Neural DNF-MT: A Neuro-symbolic Approach for Learning Interpretable and Editable Policies

Kexin Gu Baugh
Imperial College London
London, United Kingdom
kexin.gu17@imperial.ac.uk

Luke Dickens
University College London
London, United Kingdom
l.dickens@ucl.ac.uk

Alessandra Russo*
Imperial College London
London, United Kingdom
a.russo@imperial.ac.uk

ABSTRACT

Although deep reinforcement learning has been shown to be effective, the model's black-box nature presents barriers to direct policy interpretation. To address this problem, we propose a neuro-symbolic approach called neural DNF-MT for end-to-end policy learning. The differentiable nature of the neural DNF-MT model enables the use of deep actor-critic algorithms for training. At the same time, its architecture is designed so that trained models can be directly translated into interpretable policies expressed as standard (bivalent or probabilistic) logic programs. Moreover, additional layers can be included to extract abstract features from complex observations, acting as a form of predicate invention. The logic representations are highly interpretable, and we show how the bivalent representations of deterministic policies can be edited and incorporated back into a neural model, facilitating manual intervention and adaptation of learned policies. We evaluate our approach on a range of tasks requiring learning deterministic or stochastic behaviours from various forms of observations. Our empirical results show that our neural DNF-MT model performs at the level of competing black-box methods whilst providing interpretable policies.

KEYWORDS

Neuro-symbolic Learning; Neuro-symobilc Reinforcement Learning

ACM Reference Format:
Kexin Gu Baugh, Luke Dickens, and Alessandra Russo. 2025. Neural DNF-MT: A Neuro-symbolic Approach for Learning Interpretable and Editable Policies. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Remarkable progress has been made in reinforcement learning (RL) with the advancement of deep neural networks. Since the demonstration of impressive performance in complex games like Go [29] and Dota 2 [3], significant effort has been made to utilise deep RL approaches for solving real-life problems, such as segmenting surgical gestures [14] and providing treatment decisions [35]. However, the need for model interpretability grows with safety and ethical considerations. In the EU's AI Act, systems used in areas such as healthcare fall into the high-risk category, requiring both a high level of accuracy and a method to explain and interpret their output[1]. Therefore, the 'black-box' nature of neural models becomes a concern when using them for such high-stakes decisions in healthcare [15]. While many approaches exist to *explain* black-box neural models with post-hoc methods, it is argued that using inherently *interpretable* models is safer [26].

Various neuro-symbolic approaches address the lack of interpretability in deep RL. We use the term 'symbolic' to refer to methods that offer *logical rule representations*, in contrast to program synthesis approaches [4, 31, 32] that offer *programmatic representations* with forms of logic. Some of these neuro-symbolic methods [9, 16] rely on manually engineered inductive bias to restrict the search space and thus limit the rules they can learn. Others [18, 37] without predefined inductive bias associate weights with predicates but require pre-trained components to parse observations to predicates [18] or a special critic for training [37].

In this paper, we propose a neuro-symbolic model, neural DNF-MT, for learning interpretable and editable policies.[1] Our model is built upon the semi-symbolic layer and neural DNF model proposed in pix2rule [6] but with modifications that support probabilistic representation for policy learning. The model is completely differentiable and supports integration with deep actor-critic algorithms. It can also be used to distil policies from other neural models. From trained neural DNF-MT actors, we can extract bivalent logic programs for deterministic policies or probabilistic logic programs for stochastic policies. These interpretable logical representations are close approximations of the learned models. The neural-bivalent-logic translation is bidirectional, thus enabling manual policy intervention on the model. We can modify the bivalent logical program and port it back to the neural model, benefiting from the tensor operations and environment parallelism for fast inference. Compared to existing works, we do not rely on rule templates or mode declarations. Furthermore, our model is trained with a simple MLP critic and supports trainable preceding layers to generalise relevant facts from complex observations, such as multi-dimensional matrices.

To summarise, our main contributions are:

(1) We propose neural DNF-MT, a neuro-symbolic model for end-to-end policy learning and distillation, without requiring manually engineered inductive bias. It can be trained with deep actor-critic algorithms and supports end-to-end predicate invention.
(2) A trained neural DNF-MT actor's policy can be represented as a logic program (probabilistic for a stochastic policy and bivalent for a deterministic policy), thus providing interpretability.

*Sponsored in part by DEVCOM Army Research Lab under W911NF2220243.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

[1] Our main experiment repo is available at https://github.com/kittykg/neural-dnf-mt-policy-learning.

(3) The neural-to-bivalent-logic translation is bidirectional, and we can modify the logical program for policy intervention and port it back to the neural model, benefiting from tensor operations and environment parallelism for fast inference.

2 BACKGROUND

2.1 Reinforcement Learning

RL tasks are commonly modelled as Markov Decision Processes (MDPs) [24] or sometimes Partially Observable Markov Decision Processes (POMDPs) [17, 38], depending on whether the observed states are fully Markovian. The objective of an RL agent is to learn a policy that maps states to action probabilities $\pi(a_t|s_t)$ to maximise the cumulative reward. Value-based methods such as Q-learning [33] and Deep Q-Networks (DQN) [21] approximate the action-value function $Q(s_t, a_t)$, while policy-based methods such as REINFORCE [34] directly parameterise the policy π. Actor-critic algorithms such as Advantage Actor-Critic (A2C) [20] and Proximal Policy Optimisation (PPO) [27] combine both value-based and policy-based methods, where the actor learns the policy $\pi(a_t|s_t)$ and the critic learns the value function $V(s_t)$. Specifically, PPO clips the policy update in a certain range to prevent problematic large policy changes, providing stability and better performance.

2.2 Semi-symbolic Layer and Neural DNF Model

A neural Disjunctive Normal Form model [6] is a fully differentiable neural architecture where each node can be set to behave like a semi-symbolic conjunction or disjunction of its inputs. For some trainable weights w_i, $i = 1, \ldots, I$, and a parameter δ, a node in the neural DNF model is given by:

$$\hat{y} = \tanh\left(\sum_{i=1}^{I} w_i x_i + \beta\right), \text{ with } \beta = \delta\left(\max_{i=1}^{I} |w_i| - \sum_{i=1}^{I} |w_i|\right) \quad (1)$$

Here the I (semi-symbolic) inputs to the node are constrained such that $x_i \in [-1, 1]$, where the extreme value 1 (-1) is interpreted as associated term i taking the logical value $\top$ ($\bot$) with other values representing intermediate strengths of belief (a form of fuzzy logic or generalised belief). The node activation $\hat{y} \in (-1, 1)$ is interpreted similarly but cannot take specific values 1 or -1. The node's characteristics are controlled by a hyperparameter δ, which induces behaviour analogous to a logical conjunction (disjunction) when $\delta = 1$ ($= -1$). The neural DNF model consists of a layer of conjunctive nodes followed by a layer of disjunctive nodes. During training, the absolute value of each δ in both layers is controlled by a scheduler that increases from 0.1 to 1, as the model may fail to learn any rules if the logical bias is at full strength at the beginning of training.

Pix2rule [6] proposes interpreting trained neural DNF models as logical rules with Answer Set Programming (ASP) [19] semantics by treating each node's output $\hat{y} > 0$ (≤ 0) as logical $\top$ ($\bot$) (akin to a maximum likelihood estimate of the associated fact). However, Baugh et al. [2] point out that the neural DNF models cannot be used to describe multi-class classification problems because the disjunctive layer fails to guarantee a logically mutually exclusive output, i.e. with exactly one node taking value $\top$. Baugh et al. [2] instead propose an extended model called neural DNF-EO, which adds a non-trainable conjunctive semi-symbolic layer after the final layer of the base neural DNF to approximate the 'exactly-one' logical constraint '$class_j \leftarrow \wedge_{k,j\neq k}$ not $class_k$', and again show how ASP rules can be extracted from trained models.

3 NEURAL DNF-MT MODEL

This section explains why existing neural DNF-based models from [6] and [2] are imperfectly suited to represent policies within a deep-RL agent, and presents a new model called neural DNF with mutex-tanh activation (neural DNF-MT) to address these limitations. It then shows how trained models can be variously interpreted as deterministic and stochastic policies for the associated domains.

3.1 Issues of Existing Neural DNF-based Models

Unlike multi-class classification, where each sample has a single deterministic class, an RL actor seeks to approximate the optimal policy with potentially arbitrary action probabilities [30]. It is possible for a domain to have an optimal deterministic policy and for the RL algorithm to approach it with an 'almost deterministic' policy, where for each state the optimal action's probability is significantly greater than the others (i.e. a single almost-1 value vs all the rest close to 0). In this case, the actor almost always chooses a single action, similar to a multi-class classification model predicting a single class. A trained neural DNF-based model representing such a policy should be interpreted as a bivalent logic program representing the nearest deterministic policy. When we wish to preserve the probabilities encoded within the trained neural DNF-based actor without approximating it with the nearest deterministic policy, its interpretation should be captured as a probabilistic logic program that expresses the action distributions. There is no way to achieve both of these objectives with the neural DNF and neural DNF-EO models, since their interpretation frameworks do not satisfy two forms of mutual exclusivity: (a) *probabilistic mutual exclusivity* when interpreted as a stochastic policy, and (b) *logical mutual exclusivity* when interpreted as a deterministic policy. We first formalise the logic system represented by neural DNF-based models (Definition 3.1) and then define the two mutual exclusivities possible in this logic system (Definition 3.2 and 3.3).

Definition 3.1 (Generalised Belief Logic). *A neural DNF-based model that builds upon semi-symbolic layers represents a logic system. We refer to this logic system as* ***Generalised Belief Logic (GBL)****. A semi-symbolic node's activation $y_i \in (-1, 1)$ represents its belief in a logical proposition. For each activation y_i, we define a bivalent logic variable $b_i \in \{\bot, \top\}$ as its bivalent logical interpretation:*

$$b_i = \begin{cases} \top & \text{if } y_i > 0 \\ \bot & \text{otherwise} \end{cases}$$

Definition 3.2 (Logical mutual exclusivity). *Given the final activation of a neural DNF-based model for N classes $\mathbf{y} \in (-1, 1)^N$ and its bivalent logic interpretation $\mathbf{b} \in \{\bot, \top\}^N$, the model satisfies* ***logical mutual exclusivity*** *if there is exactly one b_i that is $\top$:*

$$\models \left(\bigvee_{i\in\{1..N\}} b_i\right) \wedge \left(\bigwedge_{i,j\in\{1..N\}, i<j} \neg(b_i \wedge b_j)\right)$$

Definition 3.3 (Probabilistic mutual exclusivity). *A probabilistic interpretation of GBL is a function $f_p : (-1, 1) \rightarrow (0, 1)$ that*

maps each belief y_i to probability p_i that b_i holds as true. Formally,

$$p_i = f_p(y_i) = \Pr(b_i = \top | y_i)$$

A neural DNF-based model satisfies ***probabilistic mutual exclusivity*** *if the interpreted probabilities associated with its activations $\mathbf{y} \in (0,1)^N$ under probabilistic interpretation f_p sum to 1. That is:*

$$\sum_i^N f_p(y_i) = 1$$

To be used for interpretable policy learning, a neural DNF-based model must guarantee the following properties:

P1: The model provides a probabilistic mutually exclusive interpretation (Definition 3.3) and can be interpreted as a probabilistic logic program (such as ProbLog [8]),

P2: When the *optimal policy is deterministic*, the model can also be interpreted as a bivalent logic program (such as ASP [19]) that satisfies logical mutual exclusivity (Definition 3.2).

A trained neural DNF model from [6] does not provide probabilistic interpretation or guarantee logical mutual exclusivity in its bivalent interpretation, and thus fails P1 and P2. A trained neural DNF-EO from [2] satisfies P2 via its constraint layer but fails to provide probabilistic interpretation for P1. To address these requirements, we propose a new model called neural DNF-MT and post-training processing steps that translate a trained neural DNF-MT model into a ProbLog program and, where applicable, into an ASP program. Our proposed model satisfies both properties above.

3.2 Mutex-tanh Activation

Let $\mathbf{d} \in \mathbb{R}^N$ be the output vector of a disjunctive semi-symbolic layer before any activation function and $d_k \in \mathbb{R}$ be the output of the k^{th} disjunctive node. Using the softmax function, we define the new activation function mutex-tanh as:

$$\text{softmax}(\mathbf{d})_k = \frac{e^{d_k}}{\sum_i^N e^{d_i}}$$

$$\text{mutex-tanh}(\mathbf{d})_k = 2 \cdot \text{softmax}(\mathbf{d})_k - 1 \tag{2}$$

With the mutex-tanh activation function, our neural DNF-MT model is constructed with a semi-symbolic conjunctive layer with a tanh activation function and a disjunctive semi-symbolic layer with the mutex-tanh activation function:

$\mathbf{c} = \tanh(\mathbf{W_c x} + \beta_\mathbf{c})$	Output of conj. layer
$\mathbf{d} = \mathbf{W_d c} + \beta_\mathbf{d}$	Raw output of disj. layer
$\tilde{\mathbf{y}} = \text{mutex-tanh}(\mathbf{d})$	mutex-tanh output of disj. layer

where $\mathbf{W_c}$ and $\mathbf{W_d}$ are trainable weights, and $\beta_\mathbf{c}$ and $\beta_\mathbf{d}$ are the logical biases calculated as Eq (1). Note that $\tilde{\mathbf{y}} \in (-1,1)^N$ shares the same codomain as the disjunctive layer's tanh output $\hat{\mathbf{y}}$. The disjunctive layer's bivalent interpretation $\hat{\mathbf{b}}$ still uses $\hat{\mathbf{y}}$, with $\hat{b_i} = \top$ when $\hat{y_i} > 0$ and $\bot$ otherwise.

To satisfy P1, we compute the probability $\tilde{\mathbf{p}}$ as:

$$\tilde{\mathbf{p}} = \left(f_p(\tilde{y}_1), \ldots, f_p(\tilde{y}_N)\right)^T \quad \text{where} \quad f_p(\tilde{y}_i) = \frac{\tilde{y}_i + 1}{2} \tag{3}$$

By construction, $\tilde{\mathbf{p}} \in (0,1)^N$, and we have $\sum_k^N \tilde{p_k} = 1$ from Eq (2) to satisfy probabilistic mutual exclusivity.

3.3 Policy Learning with Neural DNF-MT

In the following, we show how the neural DNF-MT model can be trained in an end-to-end fashion to approximate a stochastic policy and how to extract the policy into interpretable logical form.

Training Neural DNF-MT as Actor with PPO. Using the PPO algorithm [27], we train a neural DNF-MT actor with an MLP critic. The input to the neural DNF-MT actor must be in $[-1,1]^I$. Any discrete observation is converted into a bivalent vector representation, as shown in Figure 1. If the observation is complex, as shown in our experiment in Section 4.4, an encoder can be added before the neural DNF-MT actor to invent predicates in GBL form. The encoder output acts as input to the neural DNF-MT actor and the MLP critic, as shown in Figure 2.

We here present the overall training loss of the actor-critic PPO with a neural DNF-MT actor, which consists of multiple loss terms. The base training loss component matches that from PPO [27]:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[L^{\text{CLIP}}(\theta) + c_1 L^{\text{value}}(\theta) - c_2 S[\pi_\theta](s_t) \right] \tag{4}$$

where $c_1, c_2 \in \mathbb{R}$ are hyperparameters, $L^{\text{CLIP}}(\theta)$ is the clipped surrogate objective, $S[\pi_\theta](s_t)$ is the entropy of the actor in training, and $L^{\text{value}}(\theta)$ is the value loss. The action probability output of the neural DNF-MT actor defined in Eq (3) is used to calculate the probability ratio in L^{CLIP} and the entropy term $S[\pi_\theta](s_t)$.

We add the following auxiliary losses to facilitate the interpretation of the neural DNF-MT model into rules:

$$L^{(1)}(\theta) = \frac{1}{N_F} \sum_i^{N_F} |1 - |f_i|| \tag{5}$$

$$L^{(2)}(\theta) = \frac{1}{|\theta_{\text{disj}}|} \sum \left| \theta_{\text{disj}} \cdot (6 - |\theta_{\text{disj}}|) \right| \tag{6}$$

$$L^{(3)}(\theta) = \frac{1}{N_C} \sum_i^{N_C} |1 - |c_i|| \tag{7}$$

$$L^{(4)}(\theta) = -\sum_i^N \left[p_i \log\left(\frac{\hat{y}_i + 1}{2}\right) + (1 - p_i) \log\left(1 - \frac{\hat{y}_i + 1}{2}\right) \right] \tag{8}$$

where f_i is the invented predicate, N_F is the number of output of an encoder, and N_C is the number of conjunctive nodes. Eq (5) is used when there is an encoder before the neural DNF-MT actor for predicate invention. It enforces the predicates' activations to be close to ±1 so that they are stronger beliefs of true/false. Eq (6) is a weight regulariser to encourage the disjunctive weights to be close to ±6 (the choice of ±6 is to saturate tanh, as $\tanh(\pm 6) \approx \pm 1$). Eq (7) encourages the tanh output of the conjunctive layer to be close to ±1. Eq (8) is the key term to satisfy P2, pushing for bivalent logical interpretations for deterministic policies. This term mimics a cross-entropy loss between each mutex-tanh output and corresponding individual tanh outputs of the disjunctive layer, pushing the probability interpretations of the tanh outputs (i.e. $(\hat{y}_i + 1)/2$) towards their action probability $\tilde{p}_i$ counterparts. If the optimal policy is deterministic, all $\tilde{p}_i$ will be approximately 0 except for one, which is close to 1. Each $\hat{y}_i$ is pushed towards ±1, and only one will be close to 1, thus having exactly one bivalent interpretation $b_i = \top$ and satisfying logical mutual exclusivity.

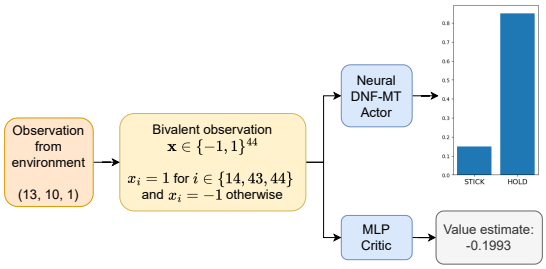

Figure 1: Neural DNF-MT model as an actor in actor-critic PPO, in environments with discrete observations.

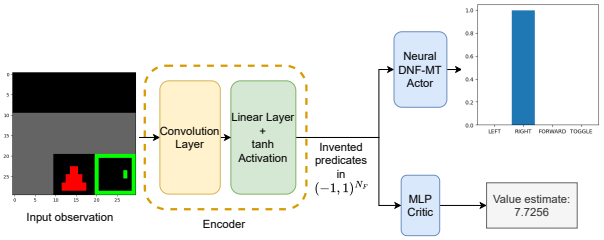

Figure 2: Neural DNF-MT model as an actor in actor-critic PPO, in environments with complex observations, such as an image-like multi-dimensional matrix.

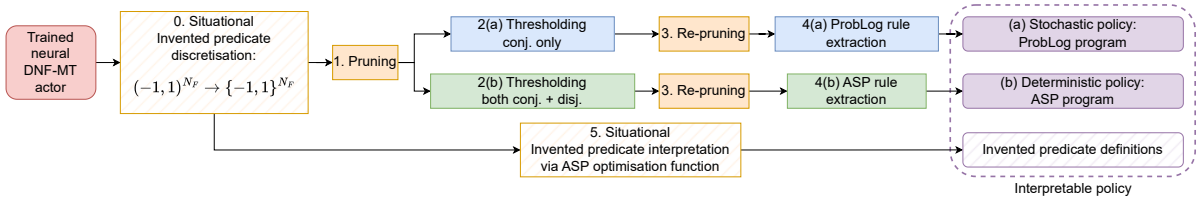

Figure 3: Post-training processing to extract an interpretable logical policy from a trained neural DNF-MT actor. There are two branches: one with sub-label (a) for extracting a stochastic policy in ProbLog and the other with sub-label (b) for extracting a deterministic policy in ASP.

Finally, the overall training loss is defined as:

$$L(\theta) = L^{\text{PPO}}(\theta) + \sum_{i \in \{1,2,3,4\}} \lambda_i L^{(i)}(\theta) \tag{9}$$

where $\lambda_i \in \mathbb{R}, i \in \{1, 2, 3, 4\}$ are hyperparameters.

Post-training Processing. This extracts either a ProbLog program for a stochastic policy or an ASP program for a close-to-deterministic policy from a trained neural DNF-MT actor, where the logic program is a close approximation of the model. It consists of multiple stages, as shown in Figure 3, described as follows.

(1) Pruning: This step repeatedly passes over each edge that connects an input to a conjunction or a conjunction to a disjunction, and removes any edge that can be removed (i) without changing the learned trajectory (for deterministic domains) or (ii) without shifting any action probability for any state more than some threshold τ_{prune} from the original learned policy (for stochastic domains). Any unconnected nodes are also removed. The process terminates when a pass fails to remove any edges or nodes.

(2) Thresholding: This process converts a semi-symbolic layer's weights from $\mathbb{R}$ to values in $\{-6, 0, 6\}$. Given some threshold $\tau \in \mathbb{R}_{\geq 0}$, a new weight is computed as $w'_{\text{k}ij} = 6 \cdot \mathbb{1}_{|w_{\text{k}ij}| \geq \tau}(w_{\text{k}ij}) \cdot \text{sign}(w_{\text{k}ij}), \text{k} \in \{\text{c}, \text{d}\}$. This weight update enables the *neural to bivalent logic translation* described later. The selection of τ should maintain the model's trajectory/action probability, subject to the same checks used in pruning. For a thresholded node with at least one non-zero weight, we replace its tanh activation with step function $h(x) = 2 \cdot \mathbb{1}_{x>0}(x) - 1$, changing its output's range to $\{-1, 1\}$. The thresholding process is applied differently to the disjunctive layer depending on the nature of the policy desired.

(2.a) **For stochastic policies**: Only the conjunctive layer is thresholded, i.e. choosing a value of τ, updating only its weights and changing the activation function. The disjunctive layer still outputs action probabilities.

(2.b) **For deterministic policies**: Thresholding is applied to both the conjunctive and disjunctive layers: a single value τ is chosen and applied in both layers' weight update, and both layers have their tanh activation replaced with the step function. This process is only possible if the model satisfies P2.

(3) Re-pruning: The pruning process from Step 1 is repeated.

(4) Logical rules extraction: All nodes (conjunctive and disjunctive) are converted into some form of logical rules. The thresholding process guarantees that all conjunctive nodes can be translated into bivalent logic representations. For a conjunctive node c_j, we consider the set $\mathcal{X}_j = \{i \in \{1..I\} | w'_{\text{c}ij} \neq 0\}$, and $|\mathcal{X}_j| \neq 0$. We partition $\mathcal{X}_j$ into subsets $\mathcal{X}_j^+ = \{i \in \mathcal{X}_j | w'_{\text{c}ij} = 6\}$ and $\mathcal{X}_j^- = \{i \in \mathcal{X}_j | w'_{\text{c}ij} = -6\}$, and translate c_j to an ASP rule of the form $conj_j \leftarrow \bigwedge_{i \in \mathcal{X}_j^+} atom_i, \bigwedge_{i \in \mathcal{X}_j^-} (\text{not } atom_i)$, where $atom_i$ is an atom for input x_i. The disjunctive nodes are interpreted differently depending on the desired policy type.

(4.a) **Stochastic policy - ProbLog rules**: We use ProbLog's annotated disjunctions to represent mutually exclusive action probabilities. Each unique achievable activation of the conjunctive layer $\mathbf{c}^{(m)} \in \{-1, 1\}^{C'}$ with $1 \leq m \leq 2^{C'}$ [2] forms the body of a unique annotated disjunction of the form $p_1 :: action_1; ...; p_N :: action_N \leftarrow \bigwedge_{i \in C^{(m)+}} conj_i, \bigwedge_{i \in C^{(m)-}} (\backslash + conj_i)$, where $C^{(m)+} = \{i | c_i^{(m)} = 1\}$, $C^{(m)-} = \{i | c_i^{(m)} = -1\}$, and $p_j =$

[2] C' is the number of remaining conjunctive nodes after pruning, which may differ from the initial choice of C.

$(\tilde{y}_j^{(m)} + 1)/2$ (the probability assigned to the j^{th} action in the disjunctive activation for the m^{th} unique activation). We compute such annotated disjunctions for all unique conjunctive activations. Listing 2 shows an example of ProbLog rules.

(4.b) **Deterministic policy - ASP rules**: Since the disjunctive layer is also thresholded, we translate each disjunctive node into a normal clause. For a disjunctive node d_j, we consider the set $C_j = \{i \in \{1..C'\} | w'_{\text{d}ij} \neq 0\}$, and $|C_j| \neq 0$. We partition C_j into subsets $C_j^+ = \{i \in C_j | w'_{\text{d}ij} = 6\}$ and $C_j^- = \{i \in C_j | w'_{\text{d}ij} = -6\}$, and translate d_j to a formula of the form $disj_j \leftarrow (\bigvee_{i \in C_j^+} conj_i) \vee (\bigvee_{i \in C_j^-} (\text{not } conj_i))$. In practice, the formula is represented as multiple rules with the same head in ASP. Listing 1 shows an example of ASP rules.

If there is an encoder before the neural DNF-MT actor in the overall architecture, we perform a mandatory step of invented predicate discretisation (step 0 in Figure 3) at the beginning of the post-training process. We take the sign of the invented predicate tanh activations, converting them to ±1 to interpret them as bivalent logical truth values of $\top$ or $\bot$. Each invented predicate is defined as a minimal set of raw observations using an ASP optimisation function (step 5 in Figure 3).

Neural-bivalent-logic translation. The translation for deterministic policies is bidirectional and maintains truth value equivalence: given an input tensor and its translated logical assignment, the interpreted bivalent truth value of the neural DNF-MT model with only ±6-and-0-valued weights is the same as the logical valuation of its translated ASP program, and vice versa.[3] A formal proof of this bidirectional claim can be found in the full version of this paper.[4]

4 EXPERIMENTS

We evaluate the RL performance (measured in episodic return) of our neural DNF-MT actors and their interpreted logical policies in four sets of environments with various forms of observations. Some tasks require stochastic behaviours, while others can be solved with deterministic policies. We compare our method with two baselines: Q-tables trained with Q-learning where applicable and MLP actors trained with actor-critic PPO. Our neural DNF-MT actors are trained with MLP critics using the PPO algorithm in the Switcheroo Corridor set, Blackjack and Door Corridor environments. In the Taxi environment, we distil a neural DNF-MT actor from a trained MLP actor. We do not directly evaluate the extracted ProbLog policies because of the long ProbLog query time. Instead, we evaluate their final neural DNF-MT actors before logical rule extraction (i.e. after step 3, re-pruning) as an approximation. The approximation is acceptable because a ProbLog policy's action distribution is the same as its corresponding neural DNF-MT's action distribution to 3 decimal places. Figure 4 summarises the performance evaluation.

4.1 Switcheroo Corridor

We adopt an example environment from [30] and create a set of Switcheroo Corridor environments that support MDP tasks with deterministic policies and POMDP tasks with stochastic policies. The observation can be either (i) the state number one-hot encoding of the agent's current position (an MDP task) or the wall status of the agent's current position (a POMDP task). In most states, going left or right results in moving in the intended direction. However, there are special states that reverse the action effect. Thus, the nature of the task decides whether the optimal policy is deterministic or stochastic. In the MDP setting, the optimal policy is deterministic: identifying the special states based on the state number and going left in them. In the POMDP setting, identifying the special states based solely on wall status observations is impossible without a memory. The optimal policy shows stochastic behaviour so that the correct action may be sampled in the special states.

The start, goal, and special states are customisable but fixed once created throughout training and inference. We create three corridors based on different configurations: Small Corridor (SC) as shown in Figure 5, Long Corridor-5 (LC-5), and Long Corridor-11 (LC-11), to test the actor's learning ability when the environment complexity increases. The configurations of LC-5 and LC-11 are listed below.

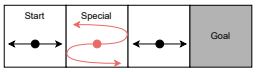

Figure 5: Small corridor (SC), same as the one from [30].

Table 1: Environment configurations for LC-5 and LC-11.

Name	Corridor Length	Start State	Goal State	Special State(s)
LC-5	5	0	4	[1]
LC-11	11	7	3	[5, 6, 7, 8]

The first six groups in Figure 4 show the performance of all models in the environment set. In MDP settings, all methods using argmax action selection perform equally well, reaching the goal with the minimum number of steps. In POMDP settings, MLP and neural DNF-MT actors perform better than Q-table with ϵ-greedy sampling as expected, with minor performance differences. Neural DNF-MT actors provide interpretability via logical programs compared to MLP actors. Listing 1 shows the ASP program for a neural DNF-MT actor in SC MDP, where state 1 is identified as special. Listing 2 shows the ProbLog rules for a neural DNF-MT actor in SC POMDP. As shown in line 1 in Listing 2, the actor favours the action going right when only the left wall is present, which only happens in state 0. Line 2 shows the case when the agent is in either state 1 or 2 with no wall on either side, and the actor shows close to 50-50 preference for both actions.

```
1 action(left) :- in_s_1.    action(right) :- not in_s_1.
```

Listing 1: ASP rules of a neural DNF-MT actor in SC MDP.

```
1 0.041::action(left) ; 0.959::action(right) :-
      left_wall_present, \+ right_wall_present.
2 0.581::action(left) ; 0.419::action(right) :- \+
      left_wall_present, \+ right_wall_present.
```

Listing 2: ProbLog rules of a neural DNF-MT actor in SC POMDP.

[3] This translation does not support predicate invention.
[4] Available at https://arxiv.org/abs/2501.03888.

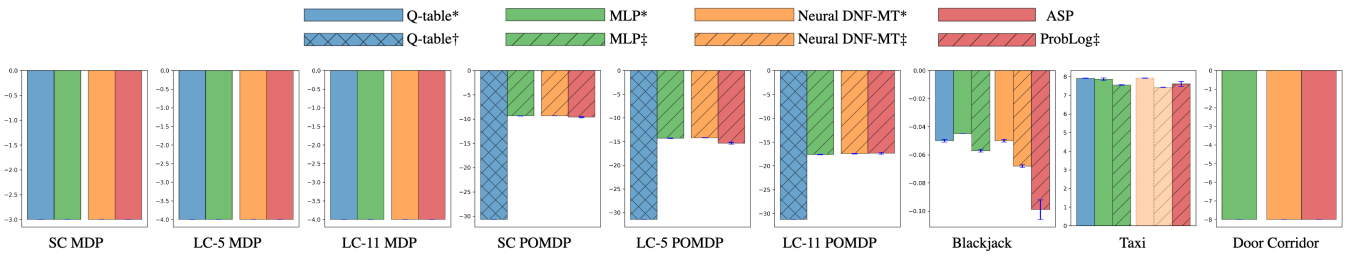

Figure 4: Mean episodic return (y-axis) ± standard error of the baselines and neural DNF-MT models, together with the ProbLog/ASP programs extracted from their corresponding neural DNF-MT models. All Q-tables are trained using Q-learning, and all MLP actors are trained with actor-critic PPO. Most neural DNF-MT actors are trained with actor-critic PPO. Except in the Taxi environment, the neural DNF-MT actor is distilled from a trained MLP actor (shown in dashed border and faded colour). Different symbols after the actor's name indicate different action selection methods: * for argmax action selection, † for ϵ-greedy sampling, and ‡ for actor's distribution sampling.

4.2 Blackjack

The Blackjack environment from [30] is a simplified version of the card game Blackjack, where the goal is to beat the dealer by having a hand closer to 21 without going over. The agent sees the sum of its hand, the dealer's face-up card, and whether it has a usable ace. It can choose to hit or stick. The performance across the models is shown in the 7$^{\text{th}}$ group in Figure 4 and Table 2. The baseline Q-table from [30] only shows a single action, so we only evaluate it with argmax action selection. We evaluate the MLP and neural DNF-MT actors with both argmax action selection and actor's distribution sampling. MLP actors with argmax action selection perform better than their distribution sampling counterparts, with a higher episodic return and win rate. The same is observed for neural DNF-MT actors. The extracted ProbLog rules perform worse than their original neural DNF-MT actors (no post-training processing), with a higher policy divergence from the Q-table from [30]. We observe a policy change from a trained neural DNF-MT actor to its extracted ProbLog policy at the thresholding stage during the post-training processing. This unwanted policy change caused by thresholding leads to performance loss and persists in later environments; we will discuss this issue further in Section 5.

Table 2: Blackjack: performance of MLP actors, neural DNF-MT actors, and the extracted ProbLog programs. Policy divergence measures the proportion of states where the argmax policy disagrees with the Q-table from [30].

Model	Episodic return	Win rate	Policy Divergence
Q-table [30]*	-0.050 ± 0.001	42.94% ± 0.00%	NA
MLP*	-0.045 ± 0.000	43.24% ± 0.02%	15.87% ± 0.30%
MLP‡	-0.057 ± 0.001	42.84% ± 0.02%	
NDNF-MT*	-0.050 ± 0.001	42.82% ± 0.06%	20.66% ± 0.56%
NDNF-MT‡	-0.068 ± 0.001	42.17% ± 0.03%	
ProbLog‡	-0.099 ± 0.007	40.79% ± 0.31%	27.92% ± 1.25%

4.3 Taxi

In the Taxi environment from [10], the agent controls a taxi to pick up a passenger first and drop them off at the destination hotel. A state number is used as the observation, and it encodes the taxi, passenger and hotel locations using the formula $((taxi\_row * 5 + taxi\_col) * 5 + passenger\_loc) * 4 + destination$. Apart from moving in four directions, the agent can pick up/drop off the passenger, but illegally picking up/dropping off will be penalised. The environment is designed for hierarchical reinforcement learning but is solvable with PPO and without task decomposition. However, for both MLP actors and neural DNF-MT actors, we find that the performance is more sensitive to PPO's hyperparameters and fine-tuning the hyperparameters is more difficult than in other environments. With the wrong set of hyperparameters, the actor settles at a local optimal with a reward of -200: never perform 'pickup'/'drop-off' and move until the step limit (200 steps). The environment is complex due to its hierarchical nature, and learning the task dependencies based on purely state numbers proves to be difficult, as a 1-value change in the x/y coordinate of the taxi results is a change of state number in 100s/10s. We successfully trained MLP actors with actor-critic PPO but failed to find a working set of hyperparameters to train neural DNF-MT actors. Instead, we distil a neural DNF-MT actor from a trained MLP actor, taking the same observation as input and aiming to output the exact action probabilities as the MLP oracle.

The performance is shown in the 8$^{\text{th}}$ group in Figure 4. Actors using argmax action selection perform better than their distribution sampling counterparts. Again, we observe a performance drop in extracted ProbLog rules. With 300 unique possible starting states, the extracted ProbLog rules are not guaranteed to finish in 200 steps: 2 out of the 10 ProbLog evaluations with action probabilities sampling have truncated episodes. Across ten post-training-processed neural DNF-MT actors with argmax action selection, there are an average of 3.3 unique starting states where the models cannot finish the environment within 200 steps. De-coupling the observation seems complicated and makes it hard to learn concise conjunctions, thus increasing the error rate in the post-training processing.

4.4 Door Corridor

Inspired by Minigrid [5], we design a corridor grid with a fixed configuration called Door Corridor, as shown in Figure 6. The agent observes a 3×3 grid in front of it (as shown as the input in Figure 2) and has a choice of four actions: turn left, turn right, move forward,

and toggle. The toggle action only changes the status of a door right in front of the agent.

For this environment, we use the architecture shown in Figure 2, where an encoder is shared between the actor and the critic. The performance of MLP actors, neural DNF-MT actors and their extracted ASP programs are shown in the last group in Figure 4. Both of the neural actors learn the optimal deterministic policy.

To evaluate an extracted ASP program in the environment, we first pass the 3×3 observation to the encoder, convert invented predicates with bivalent interpretations $\top$ to ASP facts, and then append these facts as context to the base policy. The combined ASP program has to (i) output one stable model with only one action at each step (logically mutual exclusive) and (ii) finish without truncation to be counted as a successful run. These requirements are also reflected in the neural DNF-MT actor: the final tanh activation should only have one value greater than 0, and taking that only action with greater-than-0 tanh activation at each step finishes the environment without truncation. The auxiliary loss terms in Equations 5, 7 and 8 help the neural DNF-MT actor to achieve these requirements but make the training less likely to converge on a good solution. Out of 32 runs, 6 runs cannot finish the environment within the step limit. However, 25 of the 26 remaining runs can be interpreted as ASP policies. For the single failing case, it fails to maintain logical mutual exclusivity after thresholding. While it is possible to extract a ProbLog program from it, we know the environment supports an optimal deterministic policy. Hence, no logical program is extracted for this run. The ASP programs of the 25 runs successfully finish the environment with minimal steps, as reported in Figure 4. Listing 3 shows an example of the extracted ASP program from one of the successful runs and a possible set of definitions for the invented predicates.

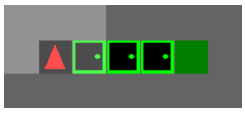

Figure 6: Door Corridor (DC): the agent needs to turn right first, and toggle and go through three doors to reach the end of the corridor.

```
action(turn_right) :- a_5, a_8.
action(forward) :- a_2.
action(toggle) :- a_3.
% Definitions of each invented predicate a_i:
a_2 :- top_right_corner_wall.
a_3 :- one_step_ahead_closed_door.
a_5 :- not curr_location_open_door,
       not one_step_ahead_closed_door.
a_8 :- two_step_ahead_unseen.
```

Listing 3: An ASP policy for a neural DNF-MT actor in DC, with a possible set of definitions for the invented predicates.

Policy Intervention. We create two variations of the base Door Corridor environment with different termination conditions: Door Corridor-T (DC-T), where the agent must be in front of the goal and toggle it instead of moving into it, and Door Corridor-OT (DC-OT), where the agent must stand on the goal and take the action 'toggle'. The input observation remains unchanged since only the goal cell's mechanism changes. The encoder can be reused immediately, but the actor and critic cannot. An MLP actor trained on DC fails to finish within step limits in DC-T and DC-OT environments without re-training. However, we can modify the ASP policy to achieve the optimal reward in both DC-T and DC-OT environments. Listings 4 and 5 show the modified ASP programs for DC-T and DC-OT environments, respectively. The modified ASP programs can be ported back to neural DNF-MT actors by virtue of the bidirectional neural-bivalent-logic translation. The modified neural DNF-MT actors also finish DC-T and DC-OT environments with minimal steps without any re-training.

```
action(turn_right) :- a_5, a_8.
action(forward) :- not a_1, a_2.
action(toggle) :- a_3.
action(toggle) :- a_1, not a_3, a_12.
```

Listing 4: Policy for DC-T, modified from Listing 3's policy.

```
action(turn_right) :- a_5, a_8, a_11.
action(forward) :- a_2.
action(toggle) :- a_3.
action(toggle) :- not a_2, not a_3, not a_11.
```

Listing 5: Policy for DC-OT, modified from Listing 3's policy.

5 DISCUSSIONS

We first analyse the persistent performance loss issue in Blackjack, Taxi, and Door Corridor environments.

Performance loss due to thresholding. The thresholding step converts the target layer(s) from a weighted continuous space to a discrete space with only 0 and ±6 values, saturating the tanh activation at ±1 and enabling the translation to bivalent logic. However, the thresholding step may not maintain the same logical interpretation of the layer output. Here we show this issue through an example in Listing 6, where a thresholded neural DNF-MT actor fails to maintain logical mutual exclusivity in the Door Corridor environment. Note that we apply thresholding on both the conjunctive and disjunctive layers since we desire a deterministic policy.

```
% Conjunctive nodes:
c_0  =  3.03 x_7  + bias_c0
c_7  =  0.56 x_13 + bias_c7
c_9  = -1.56 x_2  + bias_c9
c_11 = -1.05 x_9  + bias_c11
% Disjunctive nodes:
d_1  =  4.58 c_0 + bias_d1
d_2  = -3.48 c_9 + bias_d2
d_3  =  1.29 c_7 + 0.76 c_9 + 4.33 c_11 + bias_d3
% Thresholded nodes with tau = 0 (and ASP translation):
c_0  =  6 x_7                  (conj_0  :- a_7.)
c_7  =  6 x_13                 (conj_7  :- a_13.)
c_9  = -6 x_2                  (conj_9  :- not a_2.)
c_11 = -6 x_9                  (conj_11 :- not a_9.)
d_1  =  6 c_0                  (act(1)  :- conj_0.)
d_2  = -6 c_9                  (act(2)  :- not conj_9.)
d_3  =  6 c_7 + 6 c_9 + 6 c_11 + 12
(act(3)  :- conj_7. act(3) :- conj_9. act(3) :- conj_11.)
```

Listing 6: A neural DNF-MT actor in the Door Corridor environment that fails at the thresholding stage. We leave the bias terms uncalculated for brevity.

The 1st row of values in Table 3 are the pre-thresholding tanh output when $x_2 = -1, x_7 = 1, x_9 = 1, x_{13} = -1$, with only $\hat{y_1}$ interpreted as $\top$ and chosen as action. The thresholded conjunctive nodes in the 2nd row share the same sign as row 1, but $\hat{y_3}$ becomes positive after thresholding, resulting in two actions being $\top$ and thus violating logical mutual exclusivity. The disjunctive nodes' original weights achieve the balance of importance between c_7, c_9 and c_{11} to make

Table 3: Conjunctive and disjunctive nodes' tanh **output when $x_2 = -1, x_7 = 1, x_9 = 1, x_{13} = -1$, calculated based on the formulation in Listing 6. Row 1 is the original output without applying thresholding, and row 2 is the output after thresholding on value 0.**

c_0	c_7	c_9	c_{11}	$\hat{y_1}$	$\hat{y_2}$	$\hat{y_3}$
1.00	-0.51	0.92	-0.78	1.00	-1.00	-0.86
1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00

$\hat{y_3}$ negative. However, the thresholding process ignores the weights and makes them equally important, leading to a different output and truth value. It shows that the thresholding stage cannot handle volatile and interdependent weights and maintain the underlying truth table represented by the model. We leave it as a future work to improve/replace the thresholding stage with a more robust method.

We now discuss the implications of our work in terms of performance, interpretability, inference, and policy intervention.

Performance. From the experiments, we see that the neural DNF-MT actor can be trained with actor-critic PPO or distilled from an MLP oracle to learn an optimal policy in the Switcheroo Corridor, Taxi, and Door Corridor environments. In Blackjack, the performance is worse but close to an optimal MLP actor. Furthermore, we demonstrate that an encoder for handling complex observations and realising predicate invention can be end-to-end trained with the neural DNF-MT actor in the Door Corridor environment.

Interpretability. The logical programs extracted from trained neural DNF-MT actors provide interpretability, which MLP actors lack. We also demonstrate through different environments that we can represent stochastic and deterministic policies in different forms of logic (ProbLog and ASP, respectively).

Inference. Even if the actor has learnt an interpretable policy, running a fully logic-based agent might not be efficient. Inference in neural DNF-MT actors is significantly faster than in ProbLog or ASP, thanks to tensor operations and environment parallelism.[5]

Policy Intervention. The bidirectional neural-bivalent-logic translation allows us to modify the ASP program and translate it back to the neural architecture without re-training, as shown in Door Corridor's variations in Section 4.4. This feature could be helpful in tasks where we have background knowledge. By pre-encoding the information into logical rules or modifying the logical rules of an actor trained in a similar environment, the edited logic program can be ported back to the neural model to provide a hot start in training. This functionality will be explored in future work.

In summary, our neural DNF-MT model learns interpretable and editable policies, with the neural benefits of end-to-end training and parallelism in inference and the logical benefits of interpretable logical program representation.

6 RELATED WORK

Many neuro-symbolic approaches perform the task of inductive logic programming (ILP) [7, 23] in differentiable models, and policies are learned and represented as logical rules. They are commonly applied in Relational RL [12, 36] domains that utilise symbolic representations for states, actions, and policies. NLRL [16] and NUDGE [9] are two approaches based on the differentiable ILP system [13] and its extension from [28], where the search space needs to be defined first. NLRL generates candidate rules using rule templates. NUDGE distils symbolic policy from a trained neural model by defining its search space with mode declarations [22] and then training rule-associated weights. NeSyRL [18] and Differentiable Logic Machine (DLM) [37] do not associate weights with rules but predicates; thus, they are not reliant on rule templates or mode declarations. NeSyRL uses a disjunctive normal form Logical Neural Network (LNN) [25] as its actor, and each neuron represents an atom/logical connective. A pre-trained semantic parser extracts first-order logic predicates from text-based observations, and the LNN selects actions to generate trajectories that get stored in a replay buffer for training, similar to DQN [21]. DLM builds upon Neural Logic Machine [11] to realise forward chaining, but with logical computation units to provide interpretability. A DLM actor is trained with actor-critic PPO [27], with a specially designed critic with GRUs to handle different-arity predicates.

Different from NLRL [16] and NUDGE [9], our neural DNF-MT model does not use rule templates or mode declarations. Therefore, it does not rely on human engineering to construct the inductive bias and can learn a wider range of rules. Compared to the mentioned works that either operate on relational-based observations [9, 16, 37] or require pre-trained networks to extract logical predicates [9, 18], we demonstrate that our neural DNF-MT model is end-to-end trainable with preceding layers for predicate invention. Akin to DLM [37], we use the PPO algorithm for training; however, our method does not require a specialised critic.

7 CONCLUSION

We propose a neuro-symbolic approach named the neural DNF-MT model for learning interpretable and editable policy in RL. It can be trained with actor-critic PPO or distilled from a trained MLP actor, and an encoder for predicate invention can also be end-to-end trained together with it. The trained neural DNF-MT model can be represented as either a ProbLog program for stochastic policy or an ASP program for deterministic policy. The neural-bivalent-logic translation is bidirectional, allowing policy intervention by modifying the ASP program and then converting it back to the neural model for efficient inference in parallel environments. We evaluate the neural DNF-MT model in four environments with different forms of observations and stochastic/deterministic behaviours. The experiments show the neural DNF-MT model's capability to learn the optimal policy with performance similar to an MLP actor's. Furthermore, it provides logical representation and use cases for policy intervention, neither of which can be provided easily by an MLP. In future work, we aim to follow up on the policy intervention idea by providing the neural DNF-MT actor with a hot starting point from a modified policy. Moreover, the thresholding stage during post-training processing needs to be improved/replaced so that the underlying logical relations learned by the neural DNF-MT model can be extracted without performance loss.

[5] We provide a detailed comparison in the full version of the paper.

REFERENCES

[1] 2024. EU AI Act. https://artificialintelligenceact.eu/article/13/

[2] Kexin Gu Baugh, Nuri Cingillioglu, and Alessandra Russo. 2023. Neuro-symbolic Rule Learning in Real-world Classification Tasks. In *Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023)*, Andreas Martin, Hans-Georg Fill, Aurona Gerber, Knut Hinkelmann, Doug Lenat, Reinhard Stolle, and Frank van Harmelen (Eds.), Vol. Vol-3433. CEUR Workshop Proceedings. https://ceur-ws.org/Vol-3433/paper12.pdf

[3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *CoRR* abs/1912.06680 (2019). arXiv:1912.06680 http://arxiv.org/abs/1912.06680

[4] Yushi Cao, Zhiming Li, Tianpei Yang, Hao Zhang, Yan Zheng, Yi Li, Jianye Hao, and Yang Liu. 2024. GALOIS: boosting deep reinforcement learning via generalizable logic synthesis. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1449, 14 pages.

[5] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. 2023. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. *CoRR* abs/2306.13831 (2023).

[6] Nuri Cingillioglu and Alessandra Russo. 2021. pix2rule: End-to-end Neuro-symbolic Rule Learning. In *Proceedings of the 15th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy 2021) as part of the 1st International Joint Conference on Learning & Reasoning (IJCLR 2021)*, Artur d'Avila Garcez and Ernesto Jiménez-Ruiz (Eds.). CEUR Workshop Proceedings. https://ceur-ws.org/Vol-2986/paper3.pdf

[7] Andrew Cropper, Sebastijan Dumančić, Richard Evans, and Stephen H. Muggleton. 2022. Inductive logic programming at 30. *Machine Learning* 111, 1 (01 Jan 2022), 147–172. https://doi.org/10.1007/s10994-021-06089-1

[8] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. ProbLog: a probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (Hyderabad, India) *(IJCAI'07)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2468–2473.

[9] Quentin Delfosse, Hikaru Shindo, Devendra Dhami, and Kristian Kersting. 2023. Interpretable and Explainable Logical Policies via Neurally Guided Symbolic Abstraction. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 50838–50858. https://proceedings.neurips.cc/paper_files/paper/2023/file/9f42f06a54ce3b709ad78d34c73e4363-Paper-Conference.pdf

[10] Thomas G Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of artificial intelligence research* 13 (2000), 227–303.

[11] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural Logic Machines. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=B1xY-hRctX

[12] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. 2001. Relational Reinforcement Learning. *Machine Learning* 43, 1 (01 Apr 2001), 7–52. https://doi.org/10.1023/A:1007694015589

[13] Richard Evans and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61 (2018), 1–64.

[14] Xiaojie Gao, Yueming Jin, Qi Dou, and Pheng-Ann Heng. 2020. Automatic Gesture Recognition in Robot-assisted Surgery with Reinforcement Learning and Tree Search. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 8440–8446. https://doi.org/10.1109/ICRA40945.2020.9196674

[15] Jianxing He, Sally L. Baxter, Jie Xu, Jiming Xu, Xingtao Zhou, and Kang Zhang. 2019. The practical implementation of artificial intelligence technologies in medicine. *Nature Medicine* 25, 1 (01 Jan 2019), 30–36. https://doi.org/10.1038/s41591-018-0307-0

[16] Zhengyao Jiang and Shan Luo. 2019. Neural Logic Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3110–3119. https://proceedings.mlr.press/v97/jiang19a.html

[17] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

[18] Daiki Kimura, Masaki Ono, Subhajit Chaudhury, Ryosuke Kohita, Akifumi Wachi, Don Joven Agravante, Michiaki Tatsubori, Asim Munawar, and Alexander Gray. 2021. Neuro-Symbolic Reinforcement Learning with First-Order Logic. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 3505–3511. https://doi.org/10.18653/v1/2021.emnlp-main.283

[19] Vladimir Lifschitz. 2019. *Answer set programming*. Springer Nature, Cham, Switzerland. https://doi.org/10.1007/978-3-030-24658-7

[20] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1928–1937. https://proceedings.mlr.press/v48/mniha16.html

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (01 Feb 2015), 529–533. https://doi.org/10.1038/nature14236

[22] Stephen Muggleton. 1995. Inverse entailment and progol. *New Generation Computing* 13, 3 (01 Dec 1995), 245–286. https://doi.org/10.1007/BF03037227

[23] Stephen Muggleton and Luc de Raedt. 1994. Inductive Logic Programming: Theory and methods. *The Journal of Logic Programming* 19-20 (1994), 629–679. https://doi.org/10.1016/0743-1066(94)90035-3 Special Issue: Ten Years of Logic Programming.

[24] Martin L. Puterman. 1990. Markov decision processes. In *Stochastic Models*. Handbooks in Operations Research and Management Science, Vol. 2. Elsevier, 331–434. https://doi.org/10.1016/S0927-0507(05)80172-0

[25] Ryan Riegel, Alexander G. Gray, Francois P. S. Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh K. Srivastava. 2020. Logical Neural Networks. *CoRR* abs/2006.13155 (2020). arXiv:2006.13155 https://arxiv.org/abs/2006.13155

[26] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (01 May 2019), 206–215. https://doi.org/10.1038/s42256-019-0048-x

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[28] Hikaru Shindo, Masaaki Nishino, and Akihiro Yamamoto. 2021. Differentiable inductive logic programming for structured examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 5034–5041.

[29] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (01 Oct 2017), 354–359. https://doi.org/10.1038/nature24270

[30] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

[31] Abhinav Verma, Hoang M. Le, Yisong Yue, and Swarat Chaudhuri. 2019. Imitation-projected programmatic reinforcement learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, Article 1411, 12 pages. https://proceedings.neurips.cc/paper_files/paper/2019/file/5a44a53b7d26bb1e54c05222f186dcfb-Paper.pdf

[32] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically Interpretable Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 5045–5054. https://proceedings.mlr.press/v80/verma18a.html

[33] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (01 May 1992), 279–292. https://doi.org/10.1007/BF00992698

[34] Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* 8, 3–4 (May 1992), 229–256. https://doi.org/10.1007/BF00992696

[35] XiaoDan Wu, RuiChang Li, Zhen He, TianZhi Yu, and ChangQing Cheng. 2023. A value-based deep reinforcement learning model with human expertise in optimal treatment of sepsis. *npj Digital Medicine* 6, 1 (02 Feb 2023), 15. https://doi.org/10.1038/s41746-023-00755-5

[36] Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew M. Botvinick, Oriol Vinyals, and Peter W. Battaglia. 2018. Relational Deep Reinforcement Learning. *CoRR* abs/1806.01830 (2018). arXiv:1806.01830 http://arxiv.org/abs/1806.01830

[37] Matthieu Zimmer, Xuening Feng, Claire Glanois, Zhaohui Jiang, Jianyi Zhang, Paul Weng, Jianye Hao, Dong Li, and Wulong Liu. 2021. Differentiable Logic Machines. *CoRR* abs/2102.11529 (2021). arXiv:2102.11529 https://arxiv.org/abs/2102.11529

[38] K.J Åström. 1965. Optimal control of Markov processes with incomplete state information. *J. Math. Anal. Appl.* 10, 1 (1965), 174–205. https://doi.org/10.1016/0022-247X(65)90154-X