Encoding Human Domain Knowledge to Warm Start Reinforcement Learning

Andrew Silva, Matthew Gombolay
Institute for Robotics and Intelligent Machines, Georgia Institute of Technology
andrew.silva@gatech.edu, matthew.gombolay@cc.gatech.edu

Abstract

Deep reinforcement learning has been successful in a variety of tasks, such as game playing and robotic manipulation. However, attempting to learn *tabula rasa* disregards the logical structure of many domains as well as the wealth of readily available knowledge from domain experts that could help "warm start" the learning process. We present a novel reinforcement learning technique that allows for intelligent initialization of a neural network weights and architecture. Our approach permits the encoding domain knowledge directly into a neural decision tree, and improves upon that knowledge with policy gradient updates. We empirically validate our approach on two OpenAI Gym tasks and two modified StarCraft 2 tasks, showing that our novel architecture outperforms multilayer-perceptron and recurrent architectures. Our knowledge-based framework finds superior policies compared to imitation learning-based and prior knowledge-based approaches. Importantly, we demonstrate that our approach can be used by untrained humans to initially provide $> 80\%$ increase in expected reward relative to baselines prior to training ($p < 0.001$), which results in a $> 60\%$ increase in expected reward after policy optimization ($p = 0.011$).

1 Introduction

As reinforcement learning (RL) is applied to increasingly complex domains, such as real-time strategy games or robotic manipulation, RL and imitation learning (IL) approaches fail to quickly capture the wealth of expert knowledge that already exists for many domains. Existing approaches to using IL as a warm start require large datasets or tedious human labeling as the agent learns everything, from vision to control to policy, all at once. Unfortunately, these large datasets often do not exist, as collecting these data is impractical or expensive, and humans will not patiently label data for IL-based agents (Amershi et al. 2014). While humans may not label enough state-action pairs to train IL-based agents , there is an opportunity to improve warm starts by soliciting expertise from a human once, and then leveraging this expertise to initialize an RL agent's neural network architecture and policy. With this approach, we circumvent the need for IL and instead directly imbue human expertise into an RL agent.

To achieve this blending of human domain knowledge with the strengths of RL, we propose Propositional Logic Nets (PROLONETS), a new approach to directly encode domain knowledge as a set of propositional rules into a neural network, as depicted in Figure 1. Our approach leverages decision tree policies from humans to directly initialize a neural network (Figure 2). We use decision trees to allow humans to specify behaviors to guide the agent through a given domain, such as high-level instructions for keeping a pole balanced on the cart pole problem. Importantly, this policy specification does not require the human to demonstrate the balancing act in all possible states, nor does it require the human to label actions as being "good" or "bad."

By directly imbuing logical propositions from the tree into neural network weights, an RL agent can immediately begin learning productive strategies. This approach leverages readily available domain knowledge while still retaining the ability to learn and improve over time, eventually outperforming the expertise with which it was initialized. By exploiting the structural and logical rules inherent to many tasks to which RL is applied, we can bypass early random exploration and expedite an agent's learning in a new domain.

We demonstrate that our approach can outperform standard deep RL across two OpenAI gym domains (Brockman et al. 2016) and two modified StarCraft II domains (Vinyals et al. 2017), and that our framework is superior to state-of-the-art, IL-based RL, even with observation of that same domain expert knowledge. Finally, in a wildfire simulation domain, we show that our framework can work with untrained human participants. Our three primary contributions include:

1. We formulate a novel approach for capturing human domain expertise in a trainable RL framework via our architecture, PROLONETS, which we show outperforms baseline RL approaches, including IL-based (Cheng et al. 2018) and knowledge-based techniques (Humbird, Peterson, and McClarren 2018), obtaining $> 100\%$ more average reward on a StarCraft 2 mini-game.
2. We introduce dynamic growth to PROLONETS, enabling greater expressivity over time to surpass original initializations and yielding twice as much average reward in the lunar lander domain.
3. We conduct a user study in which non-expert humans leveraged PROLONETS to specify policies that resulted in higher cumulative rewards, both before and after training, relative to all baselines ($p < 0.05$).

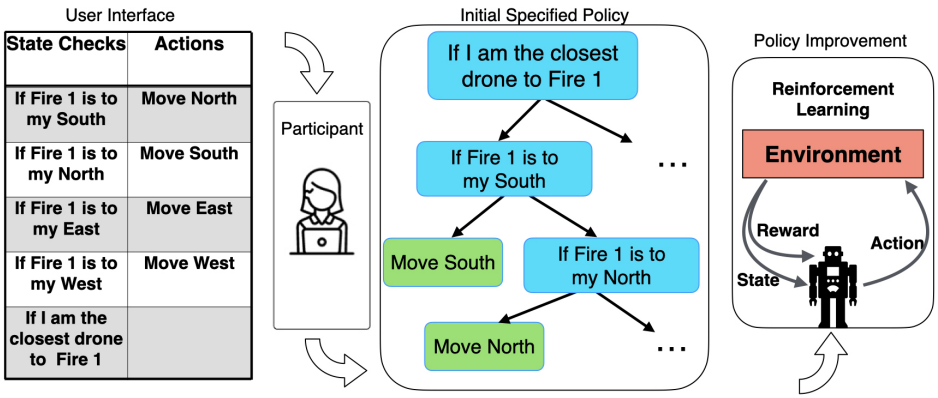


Figure 1: A visualization of our approach as it applies to our user study. Participants interact with a UI of state-checks and actions to construct a decision tree policy that is then used to directly initialize a PROLONET's architecture and parameters. The PROLONET can then begin reinforcement learning in the given domain, outgrowing its original specification.

2 Related Work

Warm starts have been used for RL (Cheng et al. 2018; Zhang and Sridharan 2019; Zhu and Liao 2017) as well as in supervised learning for many tasks (Garcez, Broda, and Gabbay 2012; Hu et al. 2016; Kontschieder et al. 2015; Wang et al. 2017). While these warm start or knowledge-based systems have provided an interesting insight into the efficacy of warm starts or human-in-the-loop learning in various domains, these systems typically involve either large labeled datasets with tedious human labeling and feedback, or they require some automated oracle to label actions as "good" or "bad." In highly challenging domains or problems, building such an oracle is rarely feasible. Moreover, it is not always possible to acquire a large labeled dataset for new domains. However, it is often possible to solicit a policy from a human in the form of a high-level series of if-then checks in critical states. These decisions can be collected as a decision tree, which our research converts into a neural network.

Researchers have previously sought to bridge the gap between decision trees and deep networks (Humbird, Peterson, and McClarren 2018; Kontschieder et al. 2015; Laptev and Buhmann 2014). This work has focused on partitioning a subspace of the data for more efficient inference (Tanno et al. 2018), enabling explicit interpretability by visualizing a network's classification policy (Frosst and Hinton 2017; Paleja et al. 2020; Silva et al. 2020), or warm starting through supervised pre-training on labeled data. As discussed, this data may not be available thus creating a need for methods which can solicit this initialization tree directly from a human.

Most closely related to our work is deep jointly-informed neural networks (DJINN) (Humbird, Peterson, and McClarren 2018), which is the latest in a long line of knowledge-based neural network research (França, Zaverucha, and Garcez 2014; Garcez, Broda, and Gabbay 2012; Maclin and Shavlik 1996; Richardson and Domingos 2006; Towell and Shavlik 1994). DJINN uses a decision tree learned over a training set in order to initialize the structure of a network's hidden layers and to route input data appropriately. However, DJINN does not explicitly initialize rules, nor does it leverage rules solicited from humans. This distinction means that DJINN creates an architecture for routing information appropriately, but the decision-criteria in each layer must be learned from scratch. Our work, on the other hand, directly initializes both the structure *and* the rules of a neural network, meaning that the human's expertise is more completely leveraged for a more useful warm start in RL domains. We build on decades of research demonstrating the value of human-in-the-loop learning (Towell and Shavlik 1994; Zhang et al. 2019) to leverage logical rules solicited from humans in the form of a decision tree to intelligently initialize the structure and rules of a deep network.

Our work is related to IL and to knowledge-based or human-in-the-loop RL frameworks (Zhang et al. 2019; Zhang and Sridharan 2019; MacGlashan et al. 2017) and apprenticeship learning and inverse RL (Abbeel and Ng 2004; Knox and Stone 2009; Chen, Paleja, and Gombolay 2020; Chen et al. 2020; Wang and Gombolay 2020a,b). Importantly, however, our approach does not require demonstrations or datasets to mimic human behavior. While our approach directly initializes with a human-specified policy, IL methods require large labeled datasets (Edwards et al. 2018) or an oracle to label data before transitioning to RL, as in the LOKI (Cheng et al. 2018) framework. Our approach translates human expertise directly into an RL agent's policy and begins learning immediately, sidestepping the IL and labeling phase.

3 Preliminaries

Within RL, we consider problems presented as a Markov decision process (MDP), which is a 5-tuple $\langle S, A, T, R, \gamma \rangle$ where $s \in S$ are states drawn from the state space or domain, $a \in A$ are possible actions drawn from the action space, $T(s', a, s)$ is the transition function representing the likelihood of reaching a next state s' by taking some action a in a given state s, $R(s)$ is the reward function which determines the reward for each state, and γ is a discount factor. In this work, we examine discrete action spaces and semantically meaningful state spaces–intelligent initialization for continuous outputs and unstructured inputs is left to future work. The

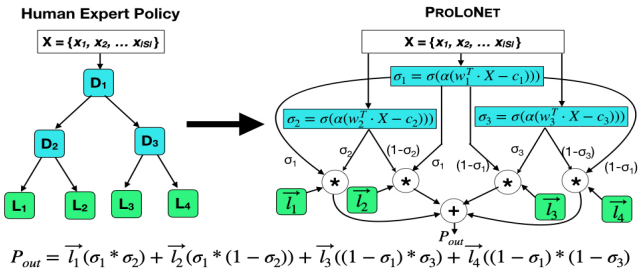


Figure 2: A traditional decision tree and a PROLONET. Decision nodes become linear layers, leaves become action weights, and the final output is a sum of the leaves weighted by path probabilities.

goal of our RL agent is to find a policy, $\pi(a|s)$, that selects actions in states to maximize the agent's expected long-term cumulative reward. IL approaches, such as ILPO (Edwards et al. 2018), operate under a similar framework, though they do not make use of the reward signal and instead perform supervised learning according to oracle data.

4 Approach

A visual overview of the PROLONET architecture is in Figure 2. To intelligently initialize a PROLONET, a human user first provides a policy in the form of some hierarchical set of decisions. These policies are solicited through simple user interactions for specifying instructions, as in Section 6. The user's decision-making process is then translated into a set of weights $\vec{w_n} \in W$ and comparator values $c_n \in C$ representing each rule, shown in Algorithm 1. Each weight $\vec{w_n}$ determines which input features to consider, and, optionally, how to weight them, as there is a unique weight value for each input feature (i.e. $|\vec{w_n}| == |S|$ for an input space S). The comparator c_n is a threshold for the weighted features.

Each decision node D_n throughout the network is represented as $D_n = \sigma[\alpha(\vec{w_n}^T * \vec{X} - c_n)]$, where $\vec{X}$ is the input data, σ is the sigmoid function, and α serves to throttle the confidence of decision nodes. Less confidence in the tree allows for more uncertainty in decision making (Yuan and Shaw 1995), leading to more exploration, even from an expert initialization. High values of α emphasize the difference between the comparator and the weighted input, thus pushing the tree to be more boolean. Lower values of α encourage a smoother tree, with $\alpha = 0$ producing uniformly random decisions. We allow α to be a learned parameter[1].

Example 1 (PROLONET Initialization)**.** *Assume we are in the cart pole domain (Barto, Sutton, and Anderson 1983) and have solicited the following from a human: "If the cart's x position is right of center, move left; otherwise, move right," and that the user indicates $x\_position$ is the first input feature of four and that the center is at 0. We therefore initialize our primary node D_0 with $\vec{w_0} = [1, 0, 0, 0]$ and $c_0 = 0$, following lines 5-8 in Alg. 1. Following lines 11-13, we create a new leaf $\vec{l_0} = [1, 0]$ (Move Left) and a new leaf $\vec{l_1} = [0, 1]$ (Move Right). Finally, we set the paths $Z(\vec{l_0}) = D_0$ and $Z(\vec{l_1}) = (\neg D_0)$. The resulting probability distribution over the agent's actions is a softmax over $(D_0 * \vec{l_0} + (1 - D_0) * \vec{l_1})$.*

[1]Code for our implementation and experiments is available at https://github.com/CORE-Robotics-Lab/ProLoNets

Algorithm 1 Intelligent Initialization

Input: Expert Propositional Rules R_d
Input: Input Size I_S, Output Size O_S
$W, C, L = \{\}$
for $r \in R_d$ **do**
  if r is a state check **then**
    $\mathbf{s}$ = feature index in r
    $w = \vec{0}^{I_S}$, $w[\mathbf{s}] = 1$
    c = comparison value in r
    $W = W \cup w$, $C = C \cup c$
  end if
  if r is an action **then**
    $\mathbf{a}$ = action index in r
    $l = \vec{0}^{O_S}$, $l[\mathbf{a}] = 1$
    $L = L \cup l$
  end if
end for
Return: W, C, L

Algorithm 2 Dynamic Growth

Input: PROLONET P_d
Input: Deeper PROLONET P_{d+1}
Input: ϵ = minimum confidence
$H(\vec{l_i})$ = Entropy of leaf $\vec{l_i}$,
for $l_i \in L \in P_d$ **do**
  Calculate $H(\vec{l_i})$ // ProLoNet leaf entropy
  Calculate $H(\vec{l_{d1}})$, $H(\vec{l_{d2}})$ // Deeper leaf entropies
  for leaves under $\vec{l_i}$ in P_{d+1} **do**
    if $H(\vec{l_i}) > (H(\vec{l_{d1}}) + H(\vec{l_{d2}}) + \epsilon)$ **then**
      Deepen P_d at $\vec{l_i}$ using $\vec{l_{d1}}$ and $\vec{l_{d2}}$
      Deepen P_{d+1} at $\vec{l_{d1}}$ and $\vec{l_{d2}}$ randomly
    end if
  end for
end for

After all decision nodes are processed, the values of D_n from each node represent the likelihood of that condition being $TRUE$. In contrast, $(1 - D_n)$ represents the likelihood of the condition being $FALSE$. With these likelihoods, the network then multiplies out the probabilities for different paths to all leaf nodes. Every leaf $\vec{l} \in L$ contains a path $z \in Z$, a set of decision nodes which should be $TRUE$ or $FALSE$ in order to reach $\vec{l}$, as well as a prior set of weights for each output action $a \in \vec{a}$. For example, in Figure 2, $z_1 = D_1 * D_2$, and $z_3 = (1 - D_1) * D_3$. The likelihood of each action a in leaf $\vec{l_i}$ is determined by multiplying the probability of reaching leaf $\vec{l_i}$ by the prior weight of the outputs within leaf $\vec{l_i}$. After calculating the outputs for every leaf, the leaves are summed and passed through a softmax function to provide the final output distribution.

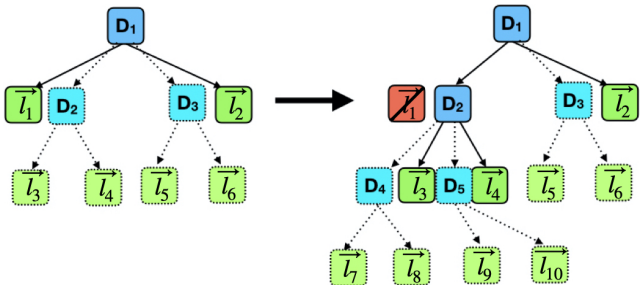


Figure 3: The dynamic growth process with a deeper PROLONET shown in paler colors and dashed lines. When $H(\vec{l_3}) + H(\vec{l_4}) < H(\vec{l_1})$, the agent replaces $\vec{l_1}$ with D_2, $\vec{l_3}$, and $\vec{l_4}$ and adds a new level to the deeper actor.

Example 2 (PROLONET Inference)**.** *Consider an example cart pole state, X=[2, 1, 0, 3] passed to the* PROLONET *from Example 1. Following* $D_n = \sigma[\alpha(\vec{w_n}^T * \vec{X} - c_n)]$*, the network arrives at* $\sigma([1,0,0,0] * [2,1,0,3] - 0) = 0.88$ *for* D_0*, meaning "mostly true." This probability propagates to the two leaf nodes using their respective paths, making the output of the network a probability given by* $(0.88 * [1,0] + (1 - 0.88) * [0,1]) = [0.88, 0.12]$*. Accordingly, the agent selects the first action with probability 0.88 and the second action otherwise. An algorithmic expression of the forward-pass is provided in the supplementary material.*

Dynamic Growth – PROLONETS are able to follow expert strategies immediately, but they may lack the expressive capacity to learn more optimal policies once they are deployed into a domain. If an expert policy involves a small number of decisions, the network will have a small number of weight vectors and comparators to use for its entire existence. To enable the PROLONET architecture to continue to grow beyond its initial definition, we introduce a dynamic growth procedure, which is outlined in Algorithm 2 and Figure 3.

Upon initialization, a PROLONET agent maintains two copies of its actor. The first is the shallower, unaltered initialized version, and the second is a deeper version in which each leaf is transformed into a randomly initialized decision node with two new randomly initialized leaves (line 1 of Alg. 2). This deeper agent has more parameters to potentially learn more complex policies, but at the cost of added randomness and uncertainty, reducing the utility of the intelligent initialization.

As the agent interacts with its environment, it relies on the shallower network to generate actions, as the shallow network represents the human's domain knowledge. After each episode, the off-policy update is run over the shallower and deeper networks. Finally, after the off-policy updates, the agent compares the entropy of the shallower actor's leaves to the entropy of the deeper actor's leaves and selectively deepens when the leaves of the deeper actor are less uniform than those of the shallower actor (lines 3-7). We find that this dynamic growth mechanism improves stability and average cumulative reward.

Example 3 (PROLONET Dynamic Growth)**.** *Assume the cart pole agent's shallower actor has found a local minimum with* $\vec{l_1} = [0.5, 0.5]$*, while the deeper actor has* $\vec{l_3} = [0.9, 0.1]$ *and* $\vec{l_4} = [0.1, 0.9]$*. Seeing that* $\vec{l_1}$ *is offering little benefit to the current policy, and* D_2 *in the deeper actor is able to make a decision about which action offers the most reward, the agent would dynamically deepen at* $\vec{l_1}$*, copying over the deeper actor's parameters and becoming more decisive in that area of its policy. The deeper actor would also grow with a random set of new parameters, as shown in Figure 3.*

5 Experimental Evaluation

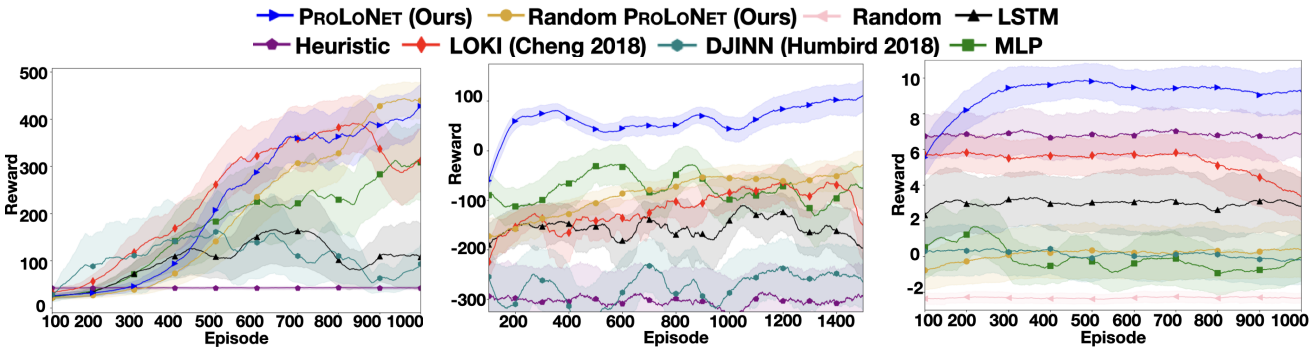
We conduct two complementary evaluations of the PROLONET as a framework for RL with human initialization. The first is a controlled investigation with expert initialization in which an author designs heuristics for a set of domains with varying complexity; this allows us to confirm that our architecture is competitive with baseline learning frameworks. We also perform an ablation of intelligent initialization and dynamic growth in this set of experiments. The second evaluation is a user study to support our claim that untrained users can specify policies that serve to improve RL.

In our first evaluation, we assess our algorithm in StarCraft II (SC2) for macro and micro battles as well as the OpenAI Gym (Brockman et al. 2016) lunar lander and cart pole environments. All agents are updated using Proximal Policy Optimization (PPO) (Schulman et al. 2017), with policy updates after each episode. Additional implementation details are available in the appendix.

To evaluate the impact of dynamic growth and intelligent initialization, we perform an ablation study and include results from these experiments in Table 1. For each M-mistake agent, weights, comparators, and leaves are randomly negated according to M, up to a maximum of $2M$ for each category.

5.1 Agent Formulations

We compare several agents across our experimental domains. The first is a PROLONET agent as described above and with expert initialization. We also evaluate a multi-layer perceptron (*MLP*) agent and a long short-term memory (*LSTM*) (Hochreiter and Schmidhuber 1997) agent, both using ReLU activations (Nair and Hinton 2010). We include comparisons to a PROLONET with random initialization (*Random* PROLONET) as well as the *Heuristic* used to initialize our agents. We compare to an IL agent trained with the *LOKI* framework, in which the agent imitates for the first K episodes (Cheng et al. 2018), where K is a tuned hyperparameter, and then transitions to RL. The *LOKI* agent supervises with the same heuristic that is used to initialize the PROLONET agent. Finally, although the original DJINN framework (Humbird, Peterson, and McClarren 2018) requires a decision tree learned over a labeled dataset, we extend the DJINN architecture to allow for initialization with a hand-crafted decision tree in order to compare to a *DJINN* agent that is initialized using the same heuristic as *LOKI* and PROLONET, but built with the DJINN architecture.



(a) Cart Pole (b) Lunar Lander (c) FindAndDefeatZerglings

Figure 4: A comparison of architectures on cart pole, lunar lander, and FindAndDefeatZerglings (Vinyals et al. 2017). As the domain complexity increases, we see that intelligent initialization is increasingly important, and PROLONETS are the most effective method for leveraging domain expertise, and perform well even when domain expertise is unnecessary, as in cart pole.

5.2 Environments

We consider four environments to empirically evaluate PROLONETS: cart pole, lunar lander, the FindAndDefeatZerglings minigame from the SC2LE (Vinyals et al. 2017), and a full game of SC2 against the in-game artificial intelligence (AI). These environments provide us with a steady increase in difficulty, from a toy problem to the challenging game of full SC2. These evaluations also showcase the ability of the PROLONET framework to compete with state-of-the-art approaches in simple domains and excel in more complex domains. For the SC2 and SC2LE problems, we use the SC2 API[2] to manufacture 193D and 37D state spaces, respectively, and 44D and 10D action spaces, respectively. In the full SC2 domain, making the right parameter update is a significant challenge for RL agents. As such, we verify that the agent's parameter updates increase its probability of victory, and if a new update has decreased the agent's chances of success, then the update is rolled back, and the agent gathers experience for a different step, similar to the checkpointing approach in Hosu and Rebedea (2016).

OpenAI Gym – As depicted in Figure 4a and 4b, PROLONETS are able to either match or exceed performance of standard reinforcement and imitation learning based RL architectures. Furthermore, we find that the PROLONET architecture–even without intelligent-initialization–is competitive with baseline architectures in the OpenAI Gym. Running reward in these domains is averaged across five runs, as recommended by Henderson et al. (2018). *MLP* and *LSTM* agents use 1-layer architectures which maintain input dimension until the output layer. We find success with intelligent initializations using as few as three nodes for the cart pole domain and as few as 10 nodes for the lunar lander. These results show that PROLONETS can leverage user knowledge to achieve superior results, and our ablation study results (Table 1) show that the architecture is robust to sub-optimal initialization in these domains.

Even where intelligent initialization is not always necessary or where high-level instruction is difficult to provide, as in cart pole, it does not hinder RL from finding solutions to the problem. Further, while baselines appear unstable in these domains, potentially owing to missing implementation hacks and tricks (Engstrom et al. 2019), we observe that the PROLONET approaches are able to succeed with the same PPO implementation and learning environment.

StarCraft II: FindAndDefeatZerglings – For this problem, we assign an agent to each individual allied unit. The best-performing initialization in this domain has 6 decision nodes and 7 leaves. Running reward is depicted in Figure 4c, again averaged over 5 runs. Intelligent initialization is crucial in this more complex domain, and the *Random* PROLONET fails to find much success despite having the same architecture as the PROLONET. *LOKI* performs on par with the *Heuristic* used to supervise actions, but *LOKI* is unable to generalize beyond the *Heuristic*. *MLP* and *LSTM* agents use a 7-layer architecture after a hyperparameter search, and we extend this to the full game of SC2. Importantly, this result (Figure 4c) shows user-initialized PROLONETS can outperform our baselines and that this initialization is key to efficient exploration and learning. The importance of the initialization policy is again shown in Table 1, where even negating 10% of the agent's parameters results in a significantly lower average reward.

StarCraft II: Full Game – After 5,000 episodes, no agent other than the PROLONET is able to win a single game against the in-game AI at the easiest setting. Even the *LOKI* and *DJINN* agents, which have access to the same heuristics used by the PROLONET, are unable to win one game. The PROLONET, on the other hand, is able to progress to the "hard" in-game AI, achieving 100% win rates against easier opponents as it progresses. Even against the "hard" in-game AI, the PROLONET agent is able to double its win rate from initialization. This result demonstrates the importance of an intelligent initialization in complex domains, where only a very narrow and specific set of actions yield successful results. Access to oracle labeling (*LOKI*) or a knowledge-based architecture (*DJINN*) does not suffice; the agent requires the actual warm start of having intelligent rules built-in. Thus, we believe these results demonstrate that our novel formulation

[2]https://github.com/Blizzard/s2client-api

DOMAIN	PROLONET	RANDOM. PROLONET	SHALLOW PROLONET	M = 0.05	M = 0.1	M = 0.15
CART POLE	449±15	401±26	415± 27	426± 30	369± 28	424± 29
LUNAR	86 ± 33	55±19	49± 20	50± 22	45± 22	45± 22
ZERGLINGS	8.9±1.5	-1.3±0.6	8.8±1.5	5.1±1.1	5.9±1.2	4.1±1.1

Table 1: PROLONET ablation study of average cumulative reward. Units are in thousands.

AI DIFFICULTY	PROLONET (OURS)	PROLONET AT INITIALIZATION	ALL OTHERS
VERYEASY	**100%**	14. %	0%
EASY	**100%**	10.9%	0%
MEDIUM	**82.2%**	11.3%	0%
HARD	**26%**	10.7%	0%

Table 2: Win rates against the StarCraft II in-game AI. "All Others" includes all agents in Section 5.1.

is singularly capable of harnessing domain knowledge.

6 User Study With Non-Experts

Our second evaluation investigates the utility of our framework with untrained humans providing the expert initialization for PROLONETS. As presented in Section 6.2, our user study shows that untrained users can leverage PROLONETS to train RL policies with superior performance. These results provide evidence that our approach can help democratize RL.

Hypotheses – We seek to investigate whether an untrained user can provide a useful initial policy for PROLONETS. Hypothesis 1 (**H1**): Expert initializations may be solicited from average users, requiring no particular training of the user, and these initializations are superior to random initializations. Hypothesis 2 (**H2**): RL can improve significantly upon these initializations, yielding superior policies after training.

Metrics – To test **H1**, we measure the reward over time for our best participant, all participants, and baseline methods. Testing **H2**, we measure the average reward for the first 50 and final 50 episodes for all agents specified by participants and our strongest baseline. Our metrics allow us to effectively examine our hypotheses in the context of expert initialization in our study domain.

Domain: Wildfire Tracking – Inspired by Seraj and Gombolay (2020), we develop a domain that is both suited to RL and of relevance to a wider audience: wildfire tracking. We randomly instantiate two fires and two drones in a 500x500 grid. The drones receive a 6D state as input, containing distances to fire centroids and Boolean flags for which a drone is the closest to each fire. The action space for drones is a 4D discrete decision of which cardinal direction to move into. Pre-made state checks include statements such as "If I am the closest drone to Fire 2" and "If Fire 1 is to my west." The two drones are controlled by separate agents without communication, and network weights are shared. The reward function is the negative distance between drones and fire centroids, encouraging drones to follow the fire as closely as possible.

6.1 Study Details

To solicit policy specifications from users, we designed a user interface that enabled participants to select from a set of premade state checks and actions. Participants were first briefed on the domain and shown a visualization and then asked to talk-through a strategy for monitoring the fires with two independent drones. After describing a solution and seeing the domain, participants were presented with the UI to build out their policies. As the participant selected options, those rules were composed into a decision tree. Once participants completed the study, we leveraged their policy specifications to initialize the structure and parameters of a PROLONET. The PROLONET was then deployed to the wildfire domain, where it further improved through RL. Our results are presented in Figure 6 and described below. We present both the highest performing participant ("Best"), as well as the median over all participants ("Median"), and compare against the agents presented in Section 5.1. *LOKI* and *DJINN* agents use the "Best" participant policy specification as a heuristic.

6.2 Study Results

Our IRB-approved study involved 15 participants (nine male, six female) between 21 and 29 years old ($M = 24, SD = 2$). The study took approximately 45 minutes, and participants were compensated for their time. Our pre-study survey revealed varying degrees of experience with robots and games, though we note that our participants were mostly computer science students. Importantly, we found that their prior experience with robots, learning from demonstration, or strategy games did not impact their ability to specify useful policies.

Nearly all participants provided policy specifications that were superior to random exploration. After performing RL over participant specifications, we can see in Figure 6 that **intelligent initialization yields the most successful RL agents,** even from non-experts. We compare to the best performing baseline, *Random* PROLONET in Figure 5. We can again see that the participants' initializations are not only better than random initialization, but are also better than the trained RL agent. A Wilcoxon signed-rank test shows that our participants' initializations (Median = -23, IQR = 19) were significantly better than a baseline initialization (Median = -87, IQR = 26), $W(15) = 1.0, p < 0.001$. Our participants' agents (Median = -7.9 , IQR = 29) were also significantly better than a baseline (Median = -52, IQR = 7.9) after training, $W(15) = 15.0, p = 0.011$. These results are significant after applying a Bonferroni correction to test the relative performance both before and after training. **This result supports hypothesis H1**, showing that average users can specify useful policies for RL agents to explore more efficiently than

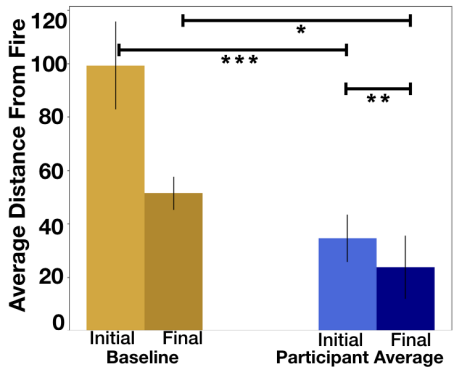


Figure 5: Initial and final distance between drones and wildfire centroids in our user study domain, where lower distance is better. Participant initializations are significantly better at tracking fires than random, showing that untrained users can leverage our approach to provide useful warm starts.

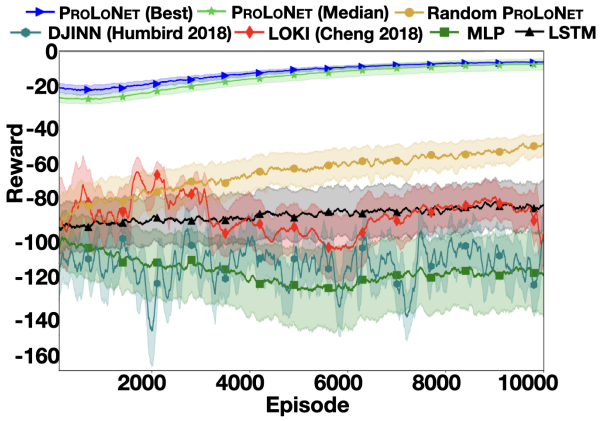


Figure 6: Wildfire tracking results, again demonstrating the importance of direct intelligent initialization (ProLoNet) rather than IL or random initialization.

random search and significantly outperform baselines.

Furthermore, our participants' agents are significantly better post-training than at initialization, as shown by a Wilcoxon signed-rank test ($W(15) = 4.0, p < 0.01$). **This finding supports hypothesis H2**, showing that RL improves on human specifications, not merely repeating what the humans have demonstrated. By combining human intuition and expertise with computation and optimization for long-term expected reward, we are able to produce agents that outperform both humans and traditional RL approaches.

Finally, we qualitatively demonstrate the utility of intelligent initialization and the ProLoNet architecture by deploying the top performing agents from each method to two drones with simulated fires to track. Videos of the top four agents are included as supplementary material.

7 Discussion

We proposed two complementary evaluations of our proposed architecture, demonstrating the significance of our contribution. Through our first set of experiments on an array of RL benchmarks with a domain expert building heuristics, we empirically validated that ProLoNets are competitive with baseline methods when initialized randomly and, with a human initialization, outperforms state-of-the-art imitation and RL baselines. As we see in Figure 4, ProLoNets are as fast or faster than baseline methods to learn an optimal policy over the same environments and optimization frameworks. In our more complex domains, we identify the importance of an intelligent initialization. While the IL baseline performs well in the FindAndDefeatZerglings minigame, *LOKI* cannot improve on the imitated policy. In the full game of SC2, no approach apart from our intelligently-initialized ProLoNet wins even a single game. The ability to leverage domain knowledge to initialize rules as well as structure, rather than simple architecture and routing information, as in DJINN, is a key difference that enables the success of our approach.

Through our user study, we demonstrated the practicality of our approach and shown that average participants, even those with no prior experience in the given domain, can produce policy specifications which significantly exceed random initialization ($p < 0.05$). Furthermore, we have demonstrated that RL can significantly improve upon these policies, learning to refine "good enough" solutions into optimal ones for a given domain. This result shows us that our participants did not simply provide our agents with optimal solutions iterated upon needlessly. Instead, our participants provided good but sub-optimal starting points for optimization. These starting policies were then refined into a solution that was more robust than either the human's solution or the best baseline solution. Our study confirms that our approach can leverage readily available human initializations for success in deep RL, and moreover, that the combination of human initialization and RL yields the best of both worlds.

8 Conclusion

We present a new architecture for deep RL agents, ProLoNets, which permits intelligent initialization of agents. ProLoNets grant agents the ability to grow their network capacity as necessary, and are surprisingly capable even with random initialization. We show that ProLoNets permit initialization from average users and achieve a high-performing policy as a result of the blend of human instruction and RL. We demonstrate, first, that our approach is superior to imitation and reinforcement learning on traditional architectures and, second, that intelligent initialization allows deep RL agents to explore and learn in environments that are too complex for randomly initialized agents. Further, we have confirmed that we can solicit these useful warm starts from average participants and still develop policies superior to baseline approaches in the given domains, paving the way for reinforcement learning to become a more collaborative enterprise across a variety of complex domains.

Acknowledgements

This work was supported by MIT Lincoln Laboratory under grant 7000437192.

Ethical Considerations

Our work is a contribution targeted at democratizing reinforcement learning in complex domains. The current state of the art in reinforcement learning in complex domains requires compute time and power beyond the capacity of many labs, hand-engineering which is rarely explained publicly, or large labeled datasets which are not always shared. By providing a means for intelligent initialization by practitioners and improved exploration in many domains, we attempt to lower the barrier to entry for research in reinforcement learning and to broaden the number of potential applications of reinforcement learning to more grounded, real-world problems. While there are risks with any technology being misused, we believe the benefits of democratizing RL outweigh the risks. We posit that giving everyone the ability to use RL rather than just large corporations and select universities is a positive contribution to society.

Beneficiaries – Our work seeks to improve and simplify reinforcement learning research for all labs and to take steps toward democratizing reinforcement learning for non-experts. We feel that the computational and dataset savings of our work stand to benefit all researchers within reinforcement learning.

Negatively affected parties – We do not feel that any group of people or research direction is negatively impacted by this work. Our work is complementary to other explorations within reinforcement learning, and insights from imitation learning translate naturally into insights on the qualities of useful or harmful intelligent initializations.

Implications of failure – While our method seeks to simplify reinforcement learning, in the worst case the initialization falls back to random and the learning agent is again faced with an intractable random exploration problem. Adversarial agents using our approach would be able to instantiate a worse-than-random agent, though our results imply that it is possible to overcome such an initialization in simple domains.

Bias and fairness – Our work does rely on the "bias" of its initialization–that is, it is biased towards the actions which a human has pre-specified. While this biased exploration may fail to accurately explore or understand the intricacies of a complex domain, the alternative (years of compute with random exploration) is simply unavailable to many researchers. This bias may be overcome through diversification of intelligent initializations which may lead to a diversity of final strategies. However, the unification of such diverse policies into a single agent and the thorough study of diverse initializations is left to future work.

References

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

Amershi, S.; Cakmak, M.; Knox, W. B.; and Kulesza, T. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35(4): 105–120.

Barto, A. G.; Sutton, R. S.; and Anderson, C. W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13(5): 834–846.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym.

Chen, L.; Paleja, R.; Ghuy, M.; and Gombolay, M. 2020. Joint goal and strategy inference across heterogeneous demonstrators via reward network distillation. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, 659–668.

Chen, L.; Paleja, R.; and Gombolay, M. 2020. Learning from suboptimal demonstration via self-supervised reward regression. In *Proceedings of the Conference on Robot Learning (CoRL)*.

Cheng, C.-A.; Yan, X.; Wagener, N.; and Boots, B. 2018. Fast Policy Learning through Imitation and Reinforcement. *arXiv preprint arXiv:1805.10413* .

Edwards, A. D.; Sahni, H.; Schroeker, Y.; and Isbell, C. L. 2018. Imitating Latent Policies from Observation. *arXiv preprint arXiv:1805.07914* .

Engstrom, L.; Ilyas, A.; Santurkar, S.; Tsipras, D.; Janoos, F.; Rudolph, L.; and Madry, A. 2019. Implementation Matters in Deep RL: A Case Study on PPO and TRPO. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

França, M. V.; Zaverucha, G.; and Garcez, A. S. d. 2014. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine learning* 94(1): 81–104.

Frosst, N.; and Hinton, G. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784* .

Garcez, A. S. d.; Broda, K. B.; and Gabbay, D. M. 2012. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

Hosu, I.-A.; and Rebedea, T. 2016. Playing atari games with deep reinforcement learning and human checkpoint replay. *arXiv preprint arXiv:1607.05077* .

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; and Xing, E. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318* .

Humbird, K. D.; Peterson, J. L.; and McClarren, R. G. 2018. Deep Neural Network Initialization With Decision Trees.

IEEE Transactions on Neural Networks and Learning Systems .

Knox, W. B.; and Stone, P. 2009. Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the International Conference on Knowledge Capture*, 9–16.

Kontschieder, P.; Fiterau, M.; Criminisi, A.; and Rota Bulo, S. 2015. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision*, 1467–1475.

Laptev, D.; and Buhmann, J. M. 2014. Convolutional decision trees for feature learning and segmentation. In *Proceedings of the German Conference on Pattern Recognition*, 95–106. Springer.

MacGlashan, J.; Ho, M. K.; Loftin, R.; Peng, B.; Wang, G.; Roberts, D. L.; Taylor, M. E.; and Littman, M. L. 2017. Interactive learning from policy-dependent human feedback. In *Proceedings of the International Conference on Machine Learning-Volume 70*, 2285–2294. JMLR. org.

Maclin, R.; and Shavlik, J. W. 1996. Creating advice-taking reinforcement learners. *Machine Learning* 22(1-3): 251–281.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML-10)*, 807–814.

Paleja, R.; Silva, A.; Chen, L.; and Gombolay, M. 2020. Interpretable and Personalized Apprenticeship Scheduling: Learning Interpretable Scheduling Policies from Heterogeneous User Demonstrations. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 6417–6428.

Richardson, M.; and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1-2): 107–136.

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

Seraj, E.; and Gombolay, M. 2020. Coordinated control of uavs for human-centered active sensing of wildfires. In *Proceedings of the American Control Conference (ACC)*, 1845–1852. IEEE.

Silva, A.; Killian, T.; Rodriguez, I. D. J.; Son, S.-H.; and Gombolay, M. 2020. Optimization Methods for Interpretable Differentiable Decision Trees in Reinforcement Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Tanno, R.; Arulkumaran, K.; Alexander, D. C.; Criminisi, A.; and Nori, A. V. 2018. Adaptive Neural Trees. *arXiv preprint arXiv:1807.06699* .

Towell, G. G.; and Shavlik, J. W. 1994. Knowledge-based artificial neural networks. *Artificial intelligence* 70(1-2): 119–165.

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* .

Wang, J.; Wang, Z.; Zhang, D.; and Yan, J. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 350.

Wang, Z.; and Gombolay, M. 2020a. Heterogeneous Graph Attention Networks for Scalable Multi-Robot Scheduling with Temporospatial Constraints. In *Proceedings of Robotics: Science and Systems*. Corvalis, Oregon, USA.

Wang, Z.; and Gombolay, M. 2020b. Learning scheduling policies for multi-robot coordination with graph attention networks. *IEEE Robotics and Automation Letters* 5(3): 4509–4516.

Yuan, Y.; and Shaw, M. J. 1995. Induction of fuzzy decision trees. *Fuzzy Sets and systems* 69(2): 125–139.

Zhang, R.; Torabi, F.; Guan, L.; Ballard, D. H.; and Stone, P. 2019. Leveraging human guidance for deep reinforcement learning tasks. *arXiv preprint arXiv:1909.09906* .

Zhang, S.; and Sridharan, M. 2019. AAAI Tutorial: Knowledge-based Sequential Decision-Making under Uncertainty. *AAAI Workshop: Knowledge-based Sequential Decision-Making under Uncertainty* .

Zhu, F.; and Liao, P. 2017. Effective warm start for the online actor-critic reinforcement learning based mhealth intervention. *arXiv preprint arXiv:1704.04866* .