Deep CPT-RL: Imparting Human-Like Risk Sensitivity to Artificial Agents

Jared Markowitz, Marie Chau, I-Jeng Wang
Johns Hopkins University Applied Physics Laboratory
11000 Johns Hopkins Road
Laurel, Maryland 20723
Jared.Markowitz@jhuapl.edu, Marie.Chau@jhuapl.edu, I-Jeng.Wang@jhuapl.edu

Abstract

Current deep reinforcement learning (DRL) methods fail to address risk in an intelligent manner, potentially leading to unsafe behaviors when deployed. One strategy for improving agent risk management is to mimic human behavior. While imperfect, human risk processing displays two key benefits absent from standard artificial agents: accounting for rare but consequential events and incorporating context. The former ability may prevent catastrophic outcomes in unfamiliar settings while the latter results in asymmetric processing of potential gains and losses. These two attributes have been quantified by behavioral economists and form the basis of cumulative prospect theory (CPT), a leading model of human decision-making. We introduce a two-step method for training DRL agents to maximize the CPT-value of full-episode rewards accumulated from an environment, rather than the standard practice of maximizing expected discounted rewards. We quantitatively compare the distribution of outcomes when optimizing full-episode expected reward, CPT-value, and conditional value-at-risk (CVaR) in the CrowdSim robot navigation environment, elucidating the impacts of different objectives on the agent's willingness to trade safety for speed. We find that properly-configured maximization of CPT-value allows for a reduction of the frequency of negative outcomes with only a slight degradation of the best outcomes, compared to maximization of expected reward.

1 Introduction

Increasingly impressive demonstrations of machine learning raise increasingly critical questions about its robustness and safety in real-world environments. To inspire trust from humans, machines must be capable of reasoning, acting, and generalizing in alignment with human preferences. In particular, they must be able to adeptly handle rare but consequential scenarios and should incorporate context in their decision-making.

Reinforcement learning (RL) methods that consider edge behaviors are often referred to as risk-sensitive and have become increasingly well-studied as the prospects for real-world deployment of RL have grown. Potential target applications include autonomous vehicles (e.g., cars and planes), autonomous schedulers (e.g., power grids), and financial portfolio management. It is critical that RL-based systems adhere to strict safety standards, particularly when the potential for injury or damage exists. They must be able to integrate seamlessly with humans, anticipating and adjusting to the actions of operators and bystanders beyond the situations in which they were trained. Few present-day systems meet this rigorous standard.

Human preferences in decision-making problems have been widely studied in finance and economics, but have only recently begun to be addressed in earnest by the machine learning community (Jie et al. 2018). As humans have a natural ability to emphasize rare events and incorporate context when assessing risk, having artificial agents mimic human decision-making behaviors could be beneficial. On the other hand, given the known shortcomings of human decision-making (Kahneman and Tversky 1979; Tversky and Kahneman 1992), human imitation could represent a mere stepping stone to more adept risk-handling strategies.

Classical reinforcement learning finds policies for sequential decision-making problems by optimizing expected future rewards. This criterion alone fails to adequately emphasize edge behaviors, rendering it unsuitable for many practical applications. However, numerous approaches exist for addressing edge behaviors through either explicit or implicit means. One widely-used explicit technique is to artificially increase the frequency of known problematic edge cases during training; however, this can lead to performance degradation on more frequently observed scenarios. Another explicit strategy is to apply a risk-sensitive measure during training. Example risk-sensitive measures include exponential utility (Pratt 1964), percentile performance criteria (Wu and Lin 1999), value-at-risk (Leavens 1945), conditional value-at-risk (Rockafellar and Uryasev 2000), prospect theory (Kahneman and Tversky 1979), and cumulative prospect theory (CPT; Tversky and Kahneman (1992)). An implicit strategy incorporates a notion of risk by considering the full value distribution as opposed to the expected value (Bellemare, Dabney, and Munos 2017; Dabney et al. 2018b,a).

In this paper, we extend CPT-RL (Prashanth et al. 2016), which explicitly incorporates cumulative prospect theory into the perceived reward of an RL agent. Our method allows for the application of CPT-RL to agents with deep policy networks, unlocking CPT-based control of more complex problem spaces. More precisely, we demonstrate a methodology for modifying agents trained by conventional deep reinforcement learning (DRL) algorithms to maximize CPT-value instead of expected rewards. We evaluate our method on an enhanced version of the CrowdSim robot navigation environment (Chen et al. 2019). Our results show that the incorporation of CPT allows for fewer negative outcomes with only a slight degradation of the best outcomes. In this case, the CPT-based agent accepts a slight reduction in speed in order to facilitate surer progress.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of common explicit and implicit risk-sensitive methods. In Section 3, we introduce cumulative prospect theory in the context of reinforcement learning, including a CPT-value estimation technique and an efficient high-dimensional stochastic approximation method. In Section 4, we present Deep CPT-RL, an extension of CPT-RL to algorithms that use deep policy networks. In Section 5, we provide computational results to illustrate the benefits of Deep CPT-RL when applied to robot navigation. In Section 6, we provide concluding remarks and directions for future research.

2 Related Work

Both explicit and implicit means for introducing risk sensitivity to artificial agents have been previously investigated. On the explicit side, methods often incorporate risk by distorting the reward function. For instance, exponential utility transforms the expected discounted reward $E[Z]$ to $\frac{1}{\beta}\log E[\exp(\beta Z)]$, where β determines the level of risk.[1] Another class of risk-sensitive measures focuses on rare occurrences not captured in the standard expected utility, in order to mitigate possible detrimental outcomes. Value-at-risk at confidence level α (VaR$_\alpha$) is an α-quantile that focuses on the tail distribution, $\text{VaR}_\alpha(Z) = \max\{z|P(Z < z) \leq \alpha\}$, where α sets risk. Conditional value-at-risk at confidence level α (CVaR$_\alpha$) considers the expectation of the tail below the α-quantile point VaR$_\alpha$.

On the implicit side, distributional reinforcement learning (DistRL) approaches risk from a different perspective. Instead of distorting the utility function and optimizing the expectation, DistRL models the value distribution in the optimization process and selects the policy based on its mean. Bellemare, Dabney, and Munos (2017) introduced the first DistRL algorithm, C51, and showed that it outperforms standard Deep Q-Networks (DQN; Mnih et al. (2015)) in some environments. C51 estimates probabilities on a fixed set of uniformly-spaced possible returns, requiring bounds that may not exist in practice. Quantile regression DQN (QR-DQN) overcomes this limitation by estimating the inverse cumulative distribution function (CDF) of equally-spaced quantiles (Dabney et al. 2018b). Implicit quantile networks further improve on QR-DQN by estimating the inverse CDF of random quantiles (Dabney et al. 2018a). Another recent method, Distributional Soft Actor-Critic (Ma et al. 2020), enables application of DistRL to continuous spaces.

[1]This is apparent after applying a straightforward Taylor expansion; risk-averse and risk-seeking behavior translate to $\beta < 0$ and $\beta > 0$, respectively.

Another approach for addressing edge cases is to look to human decision-making for inspiration. Human decision makers preferentially consider rare events and perform admirably on many tasks that RL agents are trained to tackle. The goal in many applications is to maximize agent alignment with human preferences, which may also point to approaches that attempt to mimic human decision-making. One method for incorporating human tendencies into agent behavior is to have the agent maximize the CPT-value function instead of expected future reward (CPT-RL; Prashanth et al. (2016)). CPT, a leading model of human decision-making, is a refined variant of prospect theory that includes a generalization of expected utility and is supported by substantial empirical evidence.

In this paper, we extend CPT-RL to enable application to deep neural network (DNN) policies. Because the gradient-free methods used in CPT-RL do not scale to training DNNs from scratch, we perform initial training to maximize expected reward and update a subset of the resulting optimal weights to maximize the value of the CPT function. Our approach resembles transfer learning at first glance, as the policy is retrained according to a related objective function. However, in our case, a subset of the optimal weights are retained and a subset are re-initialized and re-trained.

3 Background

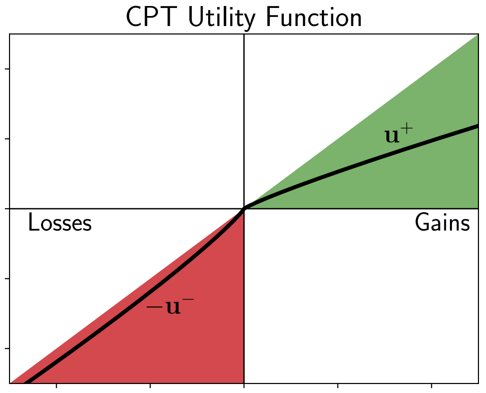
3.1 Cumulative Prospect Theory

Cumulative prospect theory uses two components to model human behavior: a *utility* function that quantifies human attitudes toward outcomes and a *weight* function that quantifies the emphasis placed on different outcomes (Figure 1). The utility function $u = (u^+, u^-)$, where $u^\pm : \mathbb{R} \to \mathbb{R}^+$, $u^+(x) = 0$ for $x \leq 0$, $u^-(x) = 0$ for $x > 0$, has two regions separated by a reference point, which we consider as zero for illustrative purposes. The region to the left of the reference point specifies the utility of losses, while the region to its right specifies the utilities of gains. Humans have empirically been found to be more risk-averse with gains than losses (Tversky and Kahneman 1992), leading to the asymmetric concavities on the two sides of the reference point. The reference point may be static or dynamic; in the case of humans it may change with accumulated experience. In our experiments we consider dynamic reference points for reasons that will be discussed in Section 5.

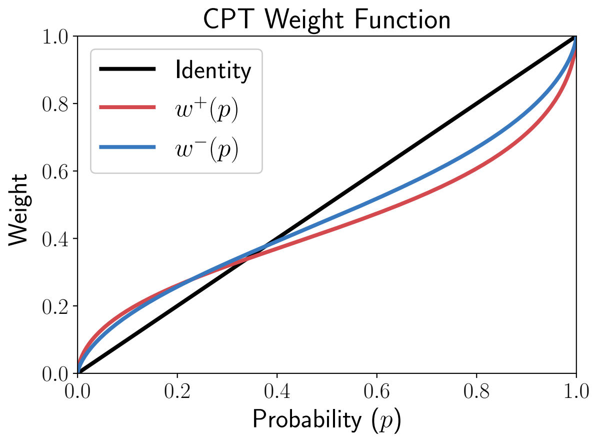
The probability weight function $w = (w^+, w^-)$, where $w^\pm : [0, 1] \to [0, 1]$, models human consideration of events based on frequency, highlighting both positive and negative rare events relative to more common events (Figure 1(b)). Tversky and Kahneman (1992) recommend the weight function $w^\pm(p) = w(p) = \frac{p^\eta}{(p^\eta+(1-p)^\eta)^{1/\eta}}$, while Prelec (1998) suggests $w^\pm(p) = w(p) = \exp(-(-\ln p)^\eta)$ where $\eta \in (0, 1)$.

The CPT-value function, defined as

$$\mathbb{C}_{u,w}(X) = \int_0^{+\infty} w^+(P(u^+(X) > z))dz - \int_0^{+\infty} w^-(P(u^-(X) > z))dz, \quad (1)$$



(a)



(b)

Figure 1: (a) Example CPT utility function. Gains and losses are treated asymmetrically, with behavior towards gains being more risk-averse. (b) Example CPT weight function. Rare outcomes on either side are emphasized, as determined by the slope of $w^+(p)$ (and similarly $w^-(p)$).

satisfies certain requirements. We consider u and w to be fixed; therefore, we simplify notation henceforth: $\mathbb{C}_{u,w} \to \mathbb{C}$.

The CPT function (1) is a generalization of expected value. In particular, if we consider $w^{\pm}(x) = x$ and $u^+(x) = x$ for $x \geq 0$, $u^-(x) = -x$ for $x < 0$, then $\mathbb{C}(X) = \int_0^{+\infty} \mathbb{P}(X > z)dz - \int_0^{+\infty} \mathbb{P}(-X > z)dz = \int_0^{+\infty} \mathbb{P}(\max(X,0) > z)dz - \int_0^{+\infty} \mathbb{P}(\max(-X,0) > z)dz = \mathbb{E}[\max(X,0)] - \mathbb{E}[\max(-X,0)]$, where u^+ and u^- are utility functions corresponding to gains ($X > 0$) and losses ($X < 0$) with reference point $X = 0$.

3.2 CPT-Value Estimation

Prashanth et al. (2016) proposed a provably convergent CPT-value estimate of (1), inspired by quantiles. Their approach is summarized in Algorithm 1. The random reward X is based on a policy π_θ, where θ is a set of controllable parameter values chosen to optimize the CPT function (1). Reward X_i is generated from episode i, where an episode is a trajectory from initial to terminal state guided by actions. Episodes are assumed to be both noisy and expensive to generate. Note that in addition to selecting appropriate utility and weight functions, this approach requires users to choose a suitable sample size or number of episodes m to produce a reasonable estimate.

Algorithm 1: CPT-value Estim. (Prashanth et al. 2016)

1. Generate $X_1, \ldots, X_m$ i.i.d. from the distribution X.
2. Let

$$\overline{\mathbb{C}}_m^+(X) = \sum_{i=1}^{m} u^+(X_{[i]}) \left(w^+\left(\frac{m+1-i}{m}\right) - w^+\left(\frac{m-i}{m}\right) \right),$$

$$\overline{\mathbb{C}}_m^-(X) = \sum_{i=1}^{m} u^-(X_{[i]}) \left(w^-\left(\frac{i}{m}\right) - w^-\left(\frac{i-1}{m}\right) \right),$$

where $X_{[i]}$ is the ith ordered statistic.

3. Return $\overline{\mathbb{C}}(X) = \overline{\mathbb{C}}_m^+(X) - \overline{\mathbb{C}}_m^-(X)$.

3.3 Gradient-Free Stochastic Approximation

Consider the stochastic optimization problem

$$\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{C}(X(\boldsymbol{\theta})),$$

where $\mathbb{C}(\cdot)$ is the CPT-value function (1), $\boldsymbol{\Theta}$ is a compact, convex subset of $\mathbb{R}^d$, $X(\cdot)$ is a random reward, and $\boldsymbol{\theta} = (\theta^1, \ldots, \theta^d)$ is a d-dimensional weight parameter. Stochastic approximation updates the weights $\boldsymbol{\theta}$ iteratively using the recursion

$$\boldsymbol{\theta}_{n+1} = \Gamma\left(\boldsymbol{\theta}_n + \boldsymbol{\gamma}_n \hat{\boldsymbol{g}}(X(\boldsymbol{\theta}_n))\right),$$

where θ_0 is chosen randomly, $\Gamma(\boldsymbol{\theta})$ is a projection operator that ensures $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, $\boldsymbol{\gamma}_n$ is a diagonal matrix with step sizes for each dimension on the diagonal, and $\hat{\boldsymbol{g}}$ is an estimate of the true gradient $\nabla\mathbb{C}$ (Robbins and Monro 1951; Chau and Fu 2015; Chau et al. 2014). Direct (unbiased) gradients are unavailable for CPT-values; however, with the availability of CPT-value estimates, indirect (biased) gradient estimates can be computed. Applicable indirect methods include finite differences (Kiefer and Wolfowitz 1952) and simultaneous perturbations stochastic approximation (SPSA) (Spall 1992).

In this paper, we employ SPSA for its computational efficiency. The kth component of the SPSA gradient is defined as

$$g_k^{SPSA}(\boldsymbol{\theta}) = \frac{\mathbb{C}(\boldsymbol{\theta} + \delta\boldsymbol{\Delta}) - \mathbb{C}(\boldsymbol{\theta} - \delta\boldsymbol{\Delta})}{2\delta\Delta_k} \quad (2)$$

for $k = 1, \ldots, d$, where $\delta > 0$ and $\boldsymbol{\Delta} = (\Delta^1, \ldots, \Delta^d)$ has random i.i.d. components with zero mean and finite inverse moments. Each CPT-value estimate is generated from sample rewards based on m episodes, i.e., X_i for $i = 1, \dots, m$. Note that the number of episodes m can vary from one iteration to the next.

4 Deep CPT-RL

The CPT-RL approach described in Section 3 was applied by Prashanth et al. (2016) to successfully optimize a Boltzmann policy and thereby control a relatively simple traffic management scenario. However, CPT-RL does not directly scale to policy mappings with high-dimensional parameter spaces, including deep neural networks. This is because SPSA produces gradient estimates with lower fidelity and higher variance than backpropagation.

To address these issues and thereby extend CPT-RL to deep policy networks, we employ a two-stage approach. First, the deep policy network of an agent is trained using a conventional actor-critic method. The actor-critic formulation was chosen in accordance with the need to learn a policy and the desire to reduce the variance of policy gradient estimates. The lower, input-side layers of the network are frozen and the upper, output-side layers (one or more) are retrained to maximize the CPT-value using a procedure similar to CPT-RL. Thus the first stage learns a feature representation of the observation space and the second stage learns a policy under the learned representation. By limiting the second stage updates to the upper layer(s) of the network, we overcome the limitations of SPSA in high-dimensional settings.

More explicitly, Stage 1 aims to find an optimal policy $\pi_{\boldsymbol{\theta}^*}$ for choosing actions $\boldsymbol{a}_t$ that maximize expected return. The optimal policy parameters $\boldsymbol{\theta}^*$ are given by

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right],$$

where the τ are trajectories drawn from the distribution $p_\theta(\tau)$ of possible trajectories an agent using policy π_θ may take in the environment, $\boldsymbol{\Theta}$ is the feasible policy parameter region, t indicates time, and $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the stochastic reward granted by the environment when action $\boldsymbol{a}_t$ is taken in state $\boldsymbol{s}_t$. Stage 2 aims to find a policy that maximizes the CPT-value function (1), or in principle any risk-sensitive measure. Unfortunately, unlike the standard expectation of rewards, the CPT-functional is based on cumulative probabilities and piecewise, precluding direct, sample-based differentiation. No nested structure is assumed, precluding the use of bootstrapping methods. Hence, we generate indirect (biased) gradients from the CPT-value estimates.

In principle, any policy-based DRL algorithm may be applied in the first stage of our procedure. We applied Proximal Policy Optimization (PPO; Schulman et al. (2017)), a leading actor-critic approach, to train our agents to convergence. As previously stated, the lower layers of the network learned in Stage 1 provide a feature representation for the upper layer(s). Therefore, PPO can be thought of as learning a "perception" mapping suitable for the ensuing "action" layers learned by the second stage. In our experiments, we found it sufficient to re-initialize the parameters and tune only the last layer in Stage 2. Further flexibility may be gained through the re-optimization of the last two layers.

The second stage is feasible because 1) the lower layers of a properly-trained initial network tease out the features necessary to produce intelligent agent behavior and 2) the action layer(s) comprise a small fraction of the parameter count of the entire network. We used SPSA to estimate the gradient because of its computational efficiency; in particular, finite differences was not considered due to the sheer number of parameters involved. As in the original CPT-RL formulation, all information required to compute the gradients was derived from batches of episodes. To increase stability, we averaged over gradient estimates generated from multiple perturbations before conducting a network update via the Adam optimizer (Kingma and Ba 2015).

One notable extension to the work by Prashanth et al. (2016) is our consideration of multiple reward terms. While other approaches are possible, we employed the simple strategy of choosing a single reference point based on all terms. We allowed this reference point to vary based on the outcome of a given episode, as required by our experimental setup discussed below. One could alternatively compute a CPT-value for each reward term where it makes sense to weigh risk using expectations for other terms, but we found this to provide similar performance with additional complexity.

Our approach is outlined in Algorithm 2. For notational simplicity, in the neural network, let $\boldsymbol{\theta}_{-1} \in \boldsymbol{\Theta}_{-1}$ and $\boldsymbol{\theta}_{:-1} \in \boldsymbol{\Theta}_{:-1}$ denote the weights in the last layer and all but the last layer, respectively. The weight parameters $\boldsymbol{\theta}_{:-1}$ are fixed in the second stage, so we drop them from the input, $X(\boldsymbol{\theta}_{:-1} \cup \boldsymbol{\theta}_{-1}) \rightarrow X(\boldsymbol{\theta}_{-1})$.

```
Algorithm 2: Deep CPT-RL
Stage 1.
1. Identify deep policy network architecture with weight parameters $\boldsymbol{\theta}$.
2. Train deep policy network to obtain optimal weight parameters $\boldsymbol{\theta}^*$ that maximize expected return (e.g., using PPO).
Stage 2.
3. Initialize stopping time $N$, gradient estimates per update $M$, SPSA parameters $\{\delta_n\}$, Adam parameters $\alpha > 0$, $\beta \in (0,1)$, $\epsilon > 0$.
4. Fix $\tilde{\boldsymbol{\theta}}_{:-1} = \boldsymbol{\theta}^*_{:-1}$, re-initialize $\tilde{\boldsymbol{\theta}}_{-1}$ s.t. $\tilde{\theta}_{-1;i} \sim$ Uniform$(-|\boldsymbol{\theta}_{-1}|^{-1/2}, |\boldsymbol{\theta}_{-1}|^{-1/2})$ for $i = 0 \dots |\tilde{\boldsymbol{\theta}}_{-1}| - 1$, where $\tilde{\boldsymbol{\theta}}_{-1} = (\tilde{\theta}_{-1;1}, \dots, \tilde{\theta}_{-1;|\tilde{\boldsymbol{\theta}}_{-1}|})$.
5. For $i = 0, \dots, N-1$:
   • For $j = 0, \dots, M-1$:
     – Generate $\boldsymbol{\Delta} = \{\Delta_k\}_{k=0}^{|\tilde{\boldsymbol{\theta}}_{-1}|-1}$, where $\Delta_k = \pm 1$ with probability $0.5$, independent $\forall k$.
     – Set $\tilde{\boldsymbol{\theta}}^{\pm}_{-1} = \tilde{\boldsymbol{\theta}}_{-1} \pm \delta_i \boldsymbol{\Delta}$ and generate $\boldsymbol{X}_j(\tilde{\boldsymbol{\theta}}^{\pm}_{-1})$.
     – Generate $\overline{\mathbb{C}}(\mathbf{X}_j(\tilde{\boldsymbol{\theta}}^{+}_{-1}))$ and $\overline{\mathbb{C}}(\mathbf{X}_j(\tilde{\boldsymbol{\theta}}^{-}_{-1}))$.
     – Compute $\boldsymbol{g}_j^{SPSA}(\tilde{\boldsymbol{\theta}}_{-1})$ using (2).
```

- Average $\{\boldsymbol{g}_j^{SPSA}\}_{j=0}^{M-1}$ and use result for Adam update $\boldsymbol{G}_i$.
- Update $\tilde{\boldsymbol{\theta}}_{-1} \leftarrow \tilde{\boldsymbol{\theta}}_{-1} + \boldsymbol{G}_i$.

6. Return $\tilde{\boldsymbol{\theta}}_{CPT} = \tilde{\boldsymbol{\theta}}_{:-1} \cup \tilde{\boldsymbol{\theta}}_{-1}$.

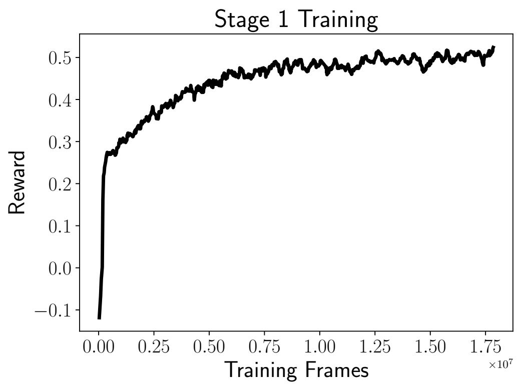


Figure 2: Stage 1 training of agent using PPO.

5 Numerical Experiments

To quantify the impact of optimizing the CPT-value, we applied our two-step procedure to an enhanced version of the CrowdSim robotic navigation environment (Chen et al. 2019). CrowdSim allows for explicit evaluation of an agent's risk-taking behavior in the form of its willingness to trade safety for speed.

5.1 CrowdSim Environment

CrowdSim was developed to study social mobility; that is, to train robotic agents to avoid collisions with pedestrians while moving toward a goal. In the default configuration, pedestrians follow the Optimal Reciprocal Collision Avoidance (ORCA; van den Berg et al. (2011)) protocol to avoid other pedestrians while traversing to their own goals. An episode concludes when the goal is reached, the robot collides with a pedestrian, or the system times out. We took the robot to be invisible to the pedestrians in order to provide a more challenging test case. To facilitate our experiments, we made a few changes to the published environment. These changes were broadly designed to make the scenario more realistic, facilitate efficient learning, and provide rotational invariance. We closed the simulation, enforcing elastic collisions when an agent reaches an outer wall. We kept pedestrians in constant motion, assigning them a new goal destination when they reached a target. While the original version fixed the initial position of the robot and its goal, we randomly placed them opposite each other on a circle centered at the origin.

In addition to environmental modifications, we adjusted the observations and rewards received by the agents. We replaced the feature-based representation with grayscale, pixel-based observations. This allowed us to encode the geometry of the system more naturally, enabling the learning of rotation-invariant navigation policies that were challenging to produce with the original, feature-based representation. We modified the published reward function to 1) allow removal of the initial imitation learning step used in Chen et al. (2019) and 2) explicitly model a speed-safety tradeoff. Removal of the imitation learning was desired to allow quantification of CPT-based shaping of an RL agent trained *tabula rasa*, as well as to prevent our training from requiring more than two phases. It was enabled through the use of a progress term that encouraged movement in the direction of the goal. The speed-safety tradeoff came from a small constant penalty being assessed at each step, encouraging the agent to get to the goal quickly. In sum, the reward function was formulated as

$$r_t = -C_{\text{time}} + C_{\text{progress}}(d_{t-1} - d_t), \tag{3}$$

where d_t is the distance between the robot and the goal at time t (i.e. $d_{t-1} - d_t$ is the "progress" made in the timestep), $C_{\text{time}} > 0$ is the time penalty, and $C_{\text{progress}} > 0$ is the progress reward. We set $C_{\text{time}} = 0.02$ and $C_{\text{progress}} = 0.1$, with the total distance from the agent's starting position to the target being 10. Our choice of C_{progress} provided a maximal progress contribution of 1 per episode; C_{time} was chosen to provide a meaningful fraction of that value for episodes of standard duration. An episode ends when one of three things happens: the robot reaches the goal, the robot collides with a pedestrian, or a timeout occurs. Hence, even though there is no explicit collision penalty, the agent is incentivized to avoid collisions because it precludes the possibility of accumulating more progress reward.

Our agent was configured to choose from 33 different motions; remaining at rest and 4 different speeds (0.25 m/s, 0.5 m/s, 0.75 m/s, 1.0 m/s) at each of 8 evenly-spaced angles in $[0, 2\pi)$. Both the robot and the pedestrians were configured to move at a preferred speed of $v_{\text{pref}} = 1.0$ m/s. We considered 10 pedestrians in each training episode, allowing that number to vary in testing to evaluate "out-of-distribution" performance. This configuration was chosen with the goal of exploring challenging regimes (where the agent would not always be able to avoid collisions) in order to generate meaningful comparisons amongst methods.

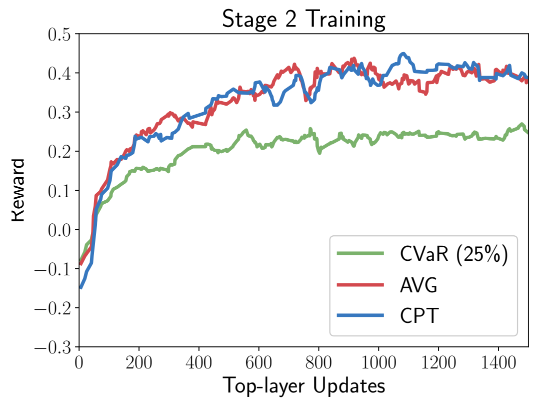
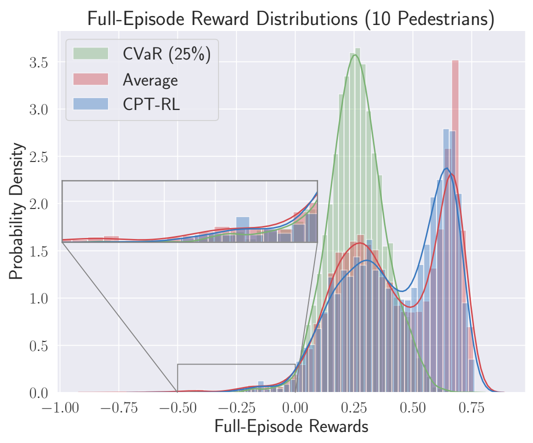
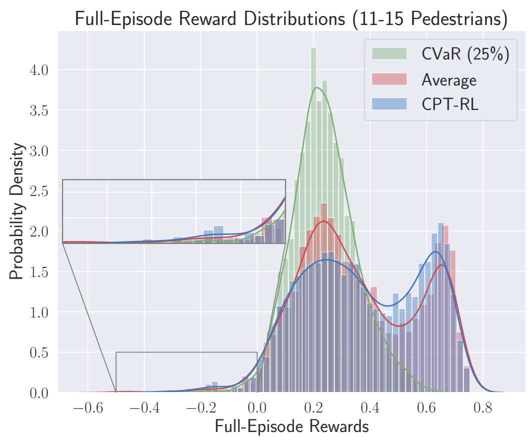


Figure 3: Stage 2 training of agent using CVaR, AVG, and CPT-based objectives.



(a)



(b)

Figure 4: (a) Histogram of full-episode rewards for agents in 10-pedestrian test scenarios. (b) Histogram of full-episode rewards for agents in test scenarios with 11-15 pedestrians.

	Rewards, 10 Pedestrians			Rewards, 11-15 Pedestrians		
Method	Mean	Median	0.01-quantile	Mean	Median	0.01-quantile
CVaR	0.262 ± 0.002	0.260	**−0.029**	0.239 ± 0.002	0.232	**−0.020**
AVG	0.418 ± 0.003	0.414	−0.172	0.358 ± 0.003	0.320	−0.066
CPT	$\mathbf{0.432 \pm 0.003}$	**0.461**	−0.085	$\mathbf{0.375 \pm 0.003}$	**0.360**	−0.123

Table 1: Reward distribution statistics for testing of Stage 2 agents. Reported error bars are standard error.

5.2 Network Architecture

In our experiments, we used a convolutional neural network that closely resembles the architecture commonly used for Atari (Mnih et al. 2015). However, our input images were 50% larger in each direction than those used by Mnih et al. (2015) to ensure proper representation of agent and goal edges. To encode motion, 4 frames were stacked. The input to the network thus consisted of $126 \times 126 \times 4$ image arrays, which were passed to 3 consecutive convolutional layers. These layers had 32, 64, and 64 channels (input-side to output-side), kernel sizes of 12, 4, and 3, and stride lengths of 6, 2, and 1. The layers were separated by rectified linear unit (ReLU) nonlinearities. The output of the last convolutional layer was passed to a rectified, 512-dimensional fully-connected layer. The resulting output was used as the input to a policy head of dimension 33 and a scalar value head.

5.3 Stage 1 Training

The neural networks governing our navigation agents were first trained to convergence using PPO. Hyperparameters were chosen to mimic those used for Atari in Schulman et al. (2017), except with shorter windows (16 steps) and more windows per batch (64).

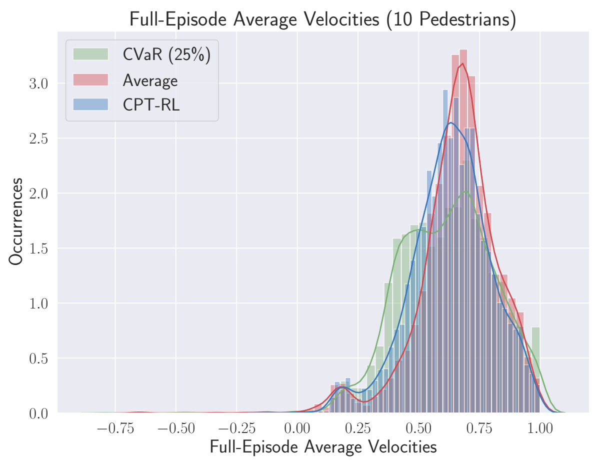
5.4 Stage 2 Training

After Stage 1 training, we trained our agent to maximize the CPT-value under Algorithm 2. For comparison purposes, we also used the same procedure to optimize for full-episode average reward (AVG) and full-episode conditional value-at-risk (CVaR). CVaR was included to evaluate a standard risk-sensitive measure, focused on improving the worst agent outcomes. While the standard practice is to consider the bottom 5% of the distribution for CVaR, we used the bottom 25% in an effort to maximize performance over a broader range of the distribution. We chose hyperparameters for the SPSA gradient estimation and the Adam optimizer in line with standard guidance (Spall 1992; Kingma and Ba 2015).

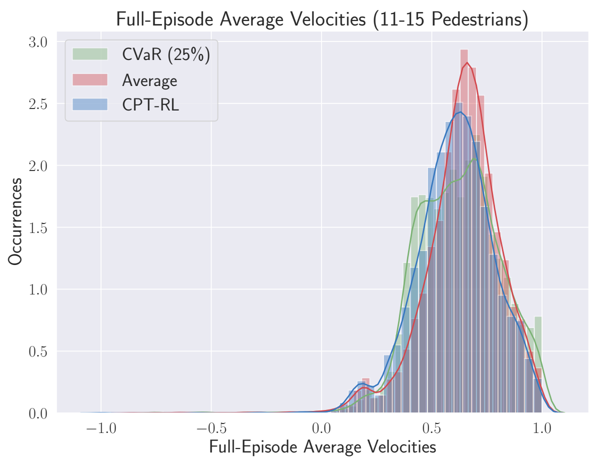
One critical detail of CPT-based training is the selection of the reference point. Our variable reference point consists of two terms, corresponding to the two terms in the reward function (3). We chose a fixed contribution of $p_{\text{ref}} = 0.5$ from the progress term, corresponding to the agent making it halfway from the start to the goal before a collision. The contribution from time varies with episode progress:

$$x_{\text{ref}} = p_{\text{ref}} + \frac{p(T)}{p_{\max}} t_{\text{ref, max}}, \tag{4}$$

where $p(T)/p_{\max}$ is the fractional progress toward the goal made by the agent over the whole episode ending at T and $t_{\text{ref, max}} = -0.3$ is a reference time multiplied by the time penalty scaling factor C_{time} in (3). Varying the reference point in this manner correlates the "acceptable" amount of time an agent may take in an episode with the amount of progress it makes. It prevents an agent from being unduly rewarded for quickly running into a pedestrian, resulting in a small episode time.



(a)



(b)

Figure 5: (a) Histogram of full-episode average velocities for agents in 10-pedestrian test scenarios. (b) Histogram of full-episode average velocities for agents in test scenarios with 11-15 pedestrians.

	10 Pedestrians				11-15 Pedestrians			
Method	Progress	Time	Velocity	Success	Progress	Time	Velocity	Success
CVaR	4.35 ± 0.03	8.61 ± 0.10	0.617 ± 0.003	0.028	3.81 ± 0.03	7.14 ± 0.08	0.621 ± 0.003	0.018
AVG	6.23 ± 0.04	10.26 ± 0.09	$\mathbf{0.661 \pm 0.002}$	0.294	5.39 ± 0.04	9.07 ± 0.09	$\mathbf{0.640 \pm 0.002}$	0.188
CPT	$\mathbf{6.63 \pm 0.04}$	11.52 ± 0.10	0.631 ± 0.002	**0.343**	$\mathbf{5.86 \pm 0.04}$	10.55 ± 0.10	0.605 ± 0.002	**0.240**

Table 2: Agent behavior statistics during testing. Reported error bars are the standard error, and the "success" quantity is the fraction of the time the agent reaches the goal without a collision.

5.5 Results

The learning curves for training in Stages 1 and 2 are shown in Figures 2 and 3, respectively. For subsequent analysis, the same amount of training data and number of Stage 2 network updates (1500; chosen for convergence) were used to compare the agents that maximized average reward, CPT-value, and CVaR. Figure 4a shows the reward distributions earned by each agent over 5000 test runs, each with 10 pedestrians. To investigate the impact of more challenging "out-of-distribution" test cases, we also evaluated on scenarios with 11-15 pedestrians sampled uniformly on the same networks, as illustrated in Figure 4b. Statistics describing each test run are provided in Table 1. Note that different random seeds were used for each of the 5000 test runs, but these seeds were the same across test conditions.

Figure 4 displays quantitative differences in the testing performance of the three agents. The CVaR approach focuses exclusively on the lower region of performance. As such, it does the best at removing lower outliers, as evidenced by the 1% quantiles in Table 1. However, it does not make any effort to enhance performance far away from the worst case and therefore cannot compete with the other two methods in any other region. The maximization of CPT-value and average reward led to similar reward distributions. However, CPT was seen to reduce the frequency of poor outcomes compared to AVG, at the cost of a slight reduction in top-end performance. Intuitively, this behavior was to be expected. As shown in Figure 1a, CPT penalizes negative outcomes that are reasonably close to the reference point more harshly than AVG while assigning slightly less utility to the very best outcomes. The weighting from Figure 1b may have played a small role in the improved performance on both edges of the distribution. The differences between the CPT and AVG agents were preserved when presented with more challenging test scenarios, as illustrated in Figures 4a and 4b.

Looking more closely at the CPT and AVG agents, we see that the former averages more progress before a collision and reaches the goal more often because it is more deliberate. Figure 5 shows the distribution of average velocities $\bar{v}$ of the agent's movement toward the goal over the test runs, defined by

$$\bar{v} = \frac{d_T - d_0}{T}, \tag{5}$$

where d_t is the distance from the agent to the goal at time t as previously defined and T is the duration of a given episode. As can be seen from Table 2, this slight reduction in speed allows the CPT-based agent to, on average, make more progress toward the goal before a collision than the other agents. It also allows the agent to reach the target without a collision more often, as reflected by the "success" fractions given in Table 2.

6 Conclusions and Future Research

We have developed a method to shape the full-episode reward distributions of DRL agents through the maximization of quantities besides expected reward. Our two-step approach has been shown to enable different agent behaviors when maximizing CPT-value, CVaR, and full-episode expected reward, both qualitatively and quantitatively. In particular, the CPT-based agents in our navigation experiments were seen to process risk differently than the AVG agents, on average proceeding more deliberately and thereby making more progress toward the goal before a collision. While CVaR is a standard risk-sensitive measure, we found that optimizing it produced agents unable to compete with the others above the very bottom of the reward distribution, even at the 25% level.

To further this work, we intend to investigate the impact of adjusting the tunable parameters of the CPT function on the risk-taking behavior of agents. In working toward assured operation, we plan to explore the combination of our methods with techniques for constrained RL (Achiam et al. 2017). Finally, to gain a deeper understanding of the effects of our methods, we also plan to apply them to more complex experimental testbeds.

7 Acknowledgments

The authors thank Ashley Llorens for insightful technical discussions and for his leadership of the Johns Hopkins Institute for Assured Autonomy (JHU IAA) project that supported our experimentation and analysis. This project was conducted under funding from both JHU/APL Internal Research and Development and JHU IAA.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained Policy Optimization. *arXiv:1705.10528 [cs]* URL http://arxiv.org/abs/1705.10528. ArXiv: 1705.10528.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 449–458.

Chau, M.; and Fu, M. C. 2015. *An Overview of Stochastic Approximation*, 149–178. Springer.

Chau, M.; Fu, M. C.; Qu, H.; and Ryzhov, I. O. 2014. Simulation Optimization: A Tutorial Overview and Recent Developments in Gradient-based Methods. In *Proceedings of the 2014 Winter Simulation Conference*, 21–35.

Chen, C.; Liu, Y.; Kreiss, S.; and Alahi, A. 2019. Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning. *2019 International Conference on Robotics and Automation* 6015–6022.

Dabney, W.; Ostrovski, G.; Silver, D.; and Munos, R. 2018a. Implicit Quantile Networks for Distributional Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 1096–1105.

Dabney, W.; Rowland, M.; Bellemare, M. G.; and Munos, R. 2018b. Distributional Reinforcement Learning With Quantile Regression. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2892–2901.

Jie, C.; Prashanth, L.; Fu, M. C.; Marcus, S.; and Szepesvári, C. 2018. Stochastic Optimization in a Cumulative Prospect Theory Framework. *IEEE Transactions on Automatic Control* 63: 2867–2882.

Kahneman, D.; and Tversky, A. 1979. Prospect Theory: An Analysis of Decision under Risk. *Econometrica* 47(2): 263–291.

Kiefer, J.; and Wolfowitz, J. 1952. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics* 23(3): 462–466.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*.

Leavens, D. H. 1945. Diversification of Investments. *Trusts and Estates* 80: 469–473.

Ma, X.; Xia, L.; Zhou, Z.; Yang, J.; and Zhao, Q. 2020. DSAC: Distributional Soft Actor Critic for Risk-Sensitive Reinforcement Learning. *arXiv: 2004.14547* .

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level Control Through Deep Reinforcement Learning. *Nature* 518(7540): 529–533.

Prashanth, L.; Jie, C.; Fu, M. C.; Marcus, S. I.; and Szepesvári, C. 2016. Cumulative Prospect Theory Meets Reinforcement Learning: Prediction and Control. In *Proceedings of the 33nd International Conference on Machine Learning*, 1406–1415.

Pratt, J. W. 1964. Risk Aversion in the Small and in the Large. *Econometrica* 32: 122–136.

Prelec, D. 1998. The Probability Weighting Function. *Econometrica* 66: 497–527.

Robbins, H.; and Monro, S. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* 22(3): 400–407.

Rockafellar, R. T.; and Uryasev, S. 2000. Optimization of Conditional Value-at-Risk. *Journal of Risk* 2: 21–41.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.

Spall, J. C. 1992. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control* 37(3): 332–341.

Tversky, A.; and Kahneman, D. 1992. Advances in Prospect Theory: Cumulative Representation of Uncertainty. *Journal of Risk and Uncertainty* 5(4): 297–323.

van den Berg, J.; Guy, S.; Lin, M.; and Manocha, D. 2011. *Robotics Research*, volume 70 of *Springer Tracts in Advanced Robotics*, chapter Reciprocal n-Body Collision Avoidance, 3–19. Springer.

Wu, C.; and Lin, Y. 1999. Minimizing Risk Models in Markov Decision Processes with Policies Depending on Target Values. *Journal of Mathematical Analysis and Applications* 231: 47–67.