# The Stochastic Road Network Environment for Robust Reinforcement Learning

John D. Martin
Department of Computing Science
University of Alberta, Edmonton, Canada
Email: jmartin8@ualberta.ca

Paul Szenher, Xi Lin, Brendan Englot
Department of Mechanical Engineering
Stevens Institute of Technology, Hoboken, USA
Email: {pszenher, xlin26, benglot}@stevens.edu

***Abstract*—Learning to navigate in uncertain environments is a critical skill for modern robotic systems, especially those designed for autonomous driving. One approach to acquiring driving skills begins with reinforcement learning in a simulated environment, then transfers the pre-trained system to a target domain for additional fine tuning. Using a highly-realistic driving simulator can reduce the approximation gap from the target domain but, beyond a certain point, accuracy can have negligible impacts on route-level navigation problems—when the concern is learning to move east or west, rather than accelerate or steer, for instance. This paper introduces the Stochastic Road Network Environment: a simulated domain aimed at modeling stochasticity that influences route-level decision making. Stochasticity due to traffic and roadway interactions is modeled with a simple stochastic reward. The proposed environment maintains a necessary amount of physical realism by streaming high-dimensional LIDAR scans of complex urban surroundings. Experiments show that robust navigation policies can be learned with modest amounts of data, further suggesting the proposed domain could be a useful platform for autonomous driving research with reinforcement learning. All code needed to use the environment will be made available in an open source repository.**

## I. Introduction

Endowing mobile robots with the ability to autonomously navigate has been viewed as a major challenge to realizing their full potential as autonomous embodied systems [1], [2]. Autonomous navigation is particularly challenging in densely-populated urban environments, where the actions of other agents and the constraints imposed by traffic rules lead to complex patterns of sensory experience. Examples are myriad and can be as complicated as observing aggressive drivers or road construction through a limited set of visual sensors. Another critical requirement in these settings is balancing competing objectives such as risk and reward, or trading off between robustness and uncertainty.

Reinforcement Learning (RL) offers a way for embodied systems to acquire navigational knowledge in complex urban environments through trial-and-error learning [3]. Instead of operating under a fixed model of an environment, as classical approaches would do, RL systems adapt their knowledge by directly interacting with the world. Until the mid-2010s, RL was impractical to use with high-dimensional sensory inputs. But this changed when RL was paired with deep neural network architectures—one of the earliest examples, DQN [4], was able to play Atari 2600 games to superhuman levels. These principles have since been adopted by embodied systems that can learn to manipulate small objects [5], [6], and navigate underactuated aerial vehicles [7].

Despite the added autonomy that deep RL offers embodied systems, navigation research in urban settings continues to be held back by the computational expense of learning tabula rasa [1]. As a result, many researchers have augmented learning with computationally-cheaper experience from simulation. One approach known as Sim-to-Real [8], begins tabula rasa learning in simulation, then deploys the system to its target environment for additional fine tuning and operation. Simulators like CARLA [9] and TORCS [10], for instance, have been used for this purpose.

Current urban driving simulators are effective for augmenting learning systems operating on small timescales, and those which require a high degree of physical realism to perform well. However, too much realism can be expensive for learning to navigate at the *route level*. Compared to obstacle avoidance and stability control systems, route-level navigation involves decision making over much larger timescales. Learning to navigate with high-rate sensory data is difficult because it contains information which is not necessarily relevant and can thus appear noisy. Deep RL has been successfully applied in other simulated environments [11]–[15], but these lack many natural qualities that are unique to urban driving settings.

This paper introduces a simulated domain for learning route-level navigation skills, called The Stochastic Road Network Environment. The environment simulates high-dimensional visual imagery over large timescales. Observations come from a virtual LIDAR sensor, whose scans are taken at different waypoints around a CARLA city map [9]. Low-level aspects of urban driving, such as traffic, are abstracted into the stochasticity of a time penalty reward signal.

This paper shows that the proposed environment can be helpful for studying robust navigation in urban settings using deep reinforcement learning. We describe a novel procedure for deploying mobile robots in uncertain environments—using a policy derived from the distribution of returns encountered under a greedy policy. The procedure is demonstrated in the proposed simulation environment; results substantiate its validity and characterize the data regime required to learn a useful robust policy. The paper concludes with some directions for future work.

## II. Problem Setting

The interaction between a robot and its environment is formalized as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, P, P_1, \gamma)$ with a finite set of states $\mathcal{S}$ and actions $\mathcal{A}$, a distribution of initial states $P_1(S_1)$, a transition distribution $P(S', R|S = s, A = a)$, and discount factor $\gamma \in [0, 1)$. At each time step $t \in \mathbb{N}$ the system takes an action $a_t$, which causes the environment to transition from $s_t$ to $s_{t+1}$ and output a scalar reward $r_{t+1}$. Actions are specified with a policy $\pi \colon \mathcal{S} \to \mathcal{P}(\mathcal{A})$, coming from the set of all policies $\Pi$. For a fixed $\pi$, the *return* is a random variable representing a sum of discounted future rewards observed while under its directive:

$$G_t^\pi \triangleq R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+3} + \cdots . \tag{1}$$

The expected return conditioned on a state-action pair is known as the action-value function:

$$q_\pi(s, a) \triangleq \mathbf{E}[G_t^\pi | S_t = s, A_t = a]. \tag{2}$$

The action-value function is the central learning target for most RL systems whost objective is to find a policy that maximizes $q_\pi(s, a)$ at every state-action pair.

In the off-policy setting, the learner takes actions according to a *behavior policy* $\beta \in \Pi$ then uses that experience to update a *target policy* $\pi$. Reinforcement learners often use the $\varepsilon$-greedy policy, which chooses uniform random actions with probability $\varepsilon$ and otherwise actions that maximize $q(s, a)$.

Distributional RL algorithms replace the value function with the distribution over random returns (1) [14]. This allows them to estimate a collection of outcomes that could result from the stochasticity inherent in the domain. For instance, if reward encodes a robot's travel time, then its return distribution reflects the spread in outcomes that result from taking certain routes—whose traffic conditions could vary from some random process. This is the problem setting our proposed environment encodes.

### A. Decision Criteria for Robust Returns

Given the potential variability in a robot's return and the consequences of directly experiencing such outcomes, a safer alternative is to use off-policy learning to find the most robust return-maximizing policy. Robustness here is measured with the dispersion of a given return distribution, and it can be measured with statistics such as the variance [16] or conditional value at risk [17]. Martin et. al [18] showed that policies based on the second-order stochastic dominance (SSD) relation are aggregations of CVaR policies, and these can lead to optimal behaviors that favor robustness when multiple solutions exist.

The SSD relation is defined using distribution functions and compared over the set of their realizable values. Consider two returns (1), $G$ and $G'$, respectively induced by actions $a$ and $a'$. We say that $G$ stochastically dominates $G'$ in the second order when their integrated CDFs, $F^{(2)}(z) \triangleq \int_{-\infty}^{z} F(x)dx$, satisfy the following equation, and we denote the relation $G \succeq_{(2)} G'$:

$$G \succeq_{(2)} G' \iff F_G^{(2)}(z) \leq F_{G'}^{(2)}(z), \ \forall \ z \in \mathbb{R}. \tag{3}$$

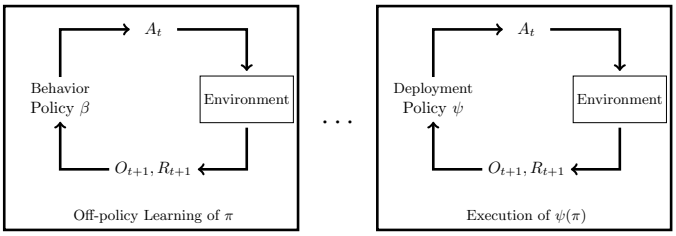


Fig. 1. **Robust Learning Procedure:** Data gathered with a behavior policy $\beta$ is used to update a target learning policy $\pi$. At some point learning will stop, then the system will be deployed with the target deployment policy $\psi$.

The function $F^{(2)}$ defines the frontier of what is known as the *dispersion space*, whose volume reflects the degree to which a random variable differs from its expected value. Outcomes that are disperse have more uncertainty and are less robust when it comes to realizing consistent outcomes. Indeed, a fundamental result from expected utility theory states that rational risk-averse agents prefer outcome $G$ to $G'$ when $G \succeq_{(2)} G'$ [19].

In settings where multiple outcomes do not result in an exact tie between expected values, the SSD policy will be equivalent to the standard greedy policy. Here we also introduce a relaxation to the SSD policy that permits uncertainty to be considered even when there is not an exact tie. The thresholded SSD policy looks at the action gap between the top two optimal actions. When this is less than a certain threshold they will be considered equivalent, and the tie will be broken with a comparison of their second moments which is consistent with the exact SSD policy—namely, the action that induces the smaller second moment will be preferred.

### B. Target Policies for Robust Deployment

This work distinguishes between two kinds of target policies. The *target learning policy* is the policy used in the update rule, and its outcomes are reflected in the learned return distribution, e.g. the standard greedy policy. In embodied learning settings, there can be an additional policy that is used after the learning process stops or is temporarily paused. We call this the *target deployment policy* $\psi \in \Pi$; it is defined as a function of the target learning policy $\psi \triangleq f(\pi)$, which we denote with the shorthand $\psi(\pi)$ in Figure 1. For example, the deployment policy can be the same as the learning policy—in which case, the robot would learn for some period of time, stop executing its behavior policy, then start executing its target learning policy. However, in settings where robustness is desired, it can be beneficial for the deployment policy to selectively choose which outcomes from the learning policy it would like to realize.

## III. The Stochastic Road Network Environment

The Stochastic Road Network Environment is built upon map structure and simulated sensor data originating from the CARLA autonomous vehicle simulator version `0.9.6` [9].

Built-in to the CARLA simulator are a set of road network maps of varied sizes, pre-configured with rich environmental details and well-defined road paths. In the targeted CARLA version `0.9.6`, five such maps are available by default,

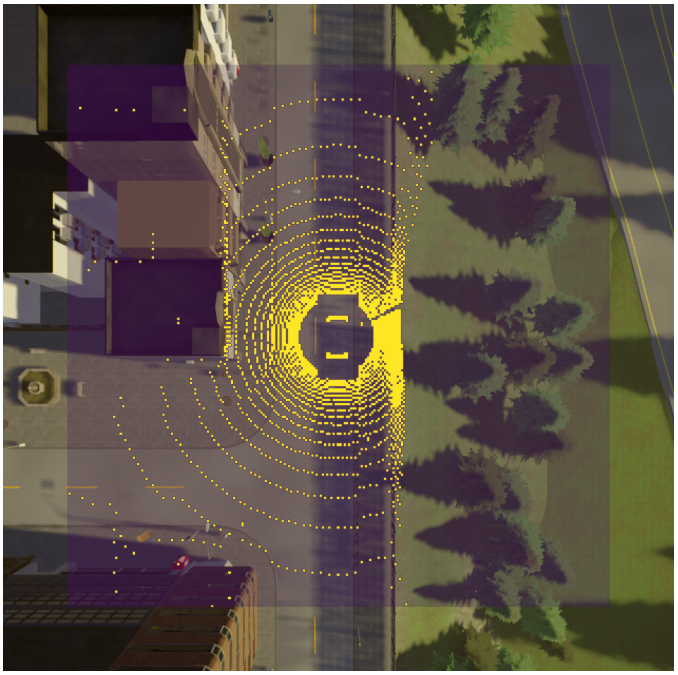

Fig. 2. **An observation sample from the proposed environment:** A learning system observes an occupancy image whose pixels correspond to the presence of a LIDAR return within a specific block of space. The image is overlaid on top of the corresponding 3D scene.

named in the sequencing of `Town01` to `Town05`. These maps vary in size, with multiple kilometers of roadway defined in each. The maps also vary with respect to the road features present, including complex intersections, multi-lane roadways (allowing for lane changes), and roundabouts. A breakdown of map action spaces, state count, and features is provided in Table I.

The underlying road-network graphs in these maps are represented in the OpenDRIVE road description format [20]. This format provides an industry-standard description language of road networks for use in autonomous vehicle simulation. Our environments are generated via extraction of discrete states and actions from the continuous OpenDRIVE maps provided alongside CARLA. The generation of learning environments is a one-time, offline data pre-processing step, whereby communication with the CARLA simulator is only necessary to generate static map graphs and observation data.

### A. Environment States and Actions

The environment is comprised of a discrete, directed road network graph with a set of fixed-position states. For each map state $s$, $n$ candidate actions are available to the agent, where $n$ is the largest number of candidate directions of travel available at any position in the environment graph. As such, the dimensionality of the action space varies among the five maps in our environment, but is fixed for all states within a single map.

The action space at each state is a discrete map of candidate state transitions. At states for which the graph topology does not allow for use of the full action space (i.e., in the case of a straightaway, where there is only one valid direction of travel) the remainder of the action space corresponds to a loopback action, whereby the agent re-transitions into its current state. State transitions are fully deterministic, meaning every state-action pair yields a transition to a deterministic subsequent state.

TABLE I
ENVIRONMENT MAP STATE AND ACTION SPACES

| Map | No. Actions | No. States | Features |
|---|---|---|---|
| `Town01` | 2 | 319 | Basic |
| `Town02` | 2 | 156 | Basic |
| `Town03` | 4 | 665 | Lane changes |
| `Town04` | 4 | 1679 | Lane changes, Roundabouts |
| `Town05` | 3 | 1220 | Lane changes, Roundabouts |

### B. Observation Stream

The CARLA simulator provides access to an array of rich sensor data-types. In this work, rich 3D LIDAR data is extracted from the CARLA simulator and post-processed into a form more amenable to usage with reinforcement learning methods.

First, raw returns from a 32-beam, 360° FoV simulated 3D LIDAR are extracted at each state location in the simulation environment. Next, this data is collected into a simple occupancy-map format on a per-state-basis, whereby the number of LIDAR returns per grid-cell are counted and stored in a 3D grid structure. This grid structure has a dimensionality of $256 \times 256 \times 3$, and a size of $50\ m \times 50\ m \times 8.4\ m$, with individual voxels of size $\sim 0.2m\ \times 0.2m\ \times 2.7m$. Finally, a single 2D slice of this grid structure ($256 \times 256$) is extracted for use as the environment observation at each environment state. The final observation is binarized, yielding a 2-dimensional matrix of size $256 \times 256$. Each cell is populated with a value of 1 if a non-zero number of LIDAR returns were present within the corresponding occupancy voxel and 0 otherwise. A representation of the resulting observation format is shown in Figure 2.

Each 2D observation matrix is associated with the state that shares the location at which the observation was taken. Observation data is non-stochastic, with each binary matrix representing a static, unchanging observation returned from a specified state. Subsequent revisiting of a given state will always yield the same observation.

### C. Reward Structure

Action rewards in the environment are the resultant sum of a fixed action reward and special-case modifiers present at specified state-action pairs.

A fixed base action reward is present at all states, yielding a deterministic reward of $-r_{base}$. In addition to this fixed action reward, there exist three categories of reward modifiers: goal actions, crosswalk actions, and loopback actions.

Goal states are those which the agent attempts to reach, and actions which result in the agent reaching this state yield an additive deterministic reward of $+r_{base}$. This positive reward offsets the base action reward, resulting in an effective reward of 0 for goal state-action pairs.

Crosswalk states are the sole source of stochasticity within the environment, and yield a reward sampled from a zero-mean truncated Gaussian distribution with a standard deviation of 1 and threshold bounds of $-r_{base}$ to $+r_{base}$. When offset by the fixed action reward, this results in an effective reward ranging from $-2r_{base}$ to 0 for crosswalk state-action pairs. Crosswalk states are intended to model sources of random travel delay in a real-world environment, such as pedestrians entering a crosswalk.

Loopback actions are actions which result in the agent re-transitioning into its current state. Loopback actions incur an additional penalty $r_{loopback}$, resulting in an effective loopback action reward of $-(r_{base}+r_{loopback})$. Loopback actions serve to model a vehicle stopping action, such as braking.

## IV. Empirical Results

Our primary empirical question asks whether uncertainty due to route stochasticity can be learned from simulated experience, then used for robust decision making. Using a distributional RL system based on Quantile Regression DQN (QR-DQN) [15], we specifically investigate the extent to which learned return distributions can be used to avoid routes with undesirable levels of stochasticity. Results from these experiments connect back to the paper's main claim that the proposed learning environment can be helpful for studying robust navigation using deep reinforcement learning.

The experiment compares the performance of different target deployment polices. This is measured with the approximate discounted return ($\gamma = 0.99$) taken over a trajectory with a maximum length of one-thousand steps. Each performance measurement is taken during an independent evaluation, at one-hundred equally-spaced times over a total of one-million time steps of learning. Evaluations occur in a separate environment instance starting from the same location. This is intended to model the counterfactual scenario of what could happen if learning were paused and the deployment policy was executed. The learner follows a $\varepsilon$-greedy policy in order to learn return distributions associated with the greedy policy (its target learning policy)[1]. Three target deployment policies are considered: the greedy policy, the exact SSD policy, and the thresholded SSD policy.

Performance results are shown in Figure 3. Convergence occurs approximately within $4 \cdot 10^5$ steps of experience. The Greedy policy is agnostic to environment stochasticity, so it takes the shortest (noisy) path. Because there are no exact ties in expectation, the SSD policy exhibits identical behavior to Greedy. We note, however, that the Thresholded SSD policy, due to the stochasticity in the shortest path, successfully takes another path without stochasticity in nearly all trials.

## V. Conclusion

This paper introduced the Stochastic Road Network Environment for route-level urban navigation. Experiments demonstrated how it can serve as a useful platform for studying and developing RL algorithms that are robust to stochastic returns. Future work will consider non-stationary sources of stochasticity to model multi-agent interactions, and the inclusion of multiple sensory modalities.

[1]Although our work uses the $\varepsilon$-greedy behavior policy, in practice, one could use a policy that is known to respect safety constraints and still explore the operating space sufficiently well.

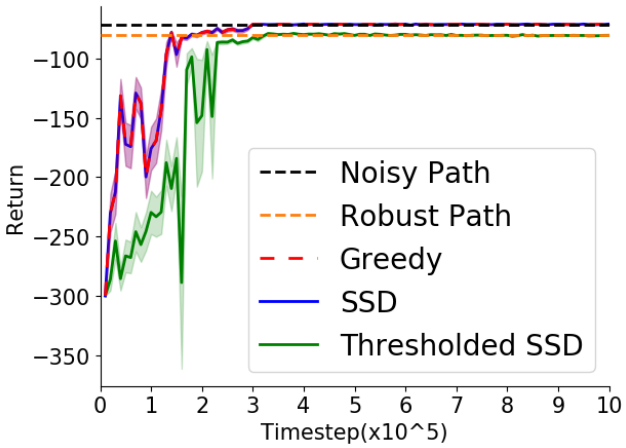


Fig. 3. **Learning curves** of QR-DQN [15] with different target deployment policies. We report the mean and standard deviation resulting from 30 trials. As indicated by the returns, both the greedy and exact SSD policies take a path with high stochasticity in all trials. The thresholded-SSD policy sacrifices optimality for robustness, and during the final evaluation, the second shortest path (Robust Path in the plot) is taken in 22 trials, the third shortest path is taken in seven trials, and the shortest path (Noisy Path in the plot) is taken in one trial. Both the second shortest path and the third shortest path are noiseless, and their lengths differ by only one step.

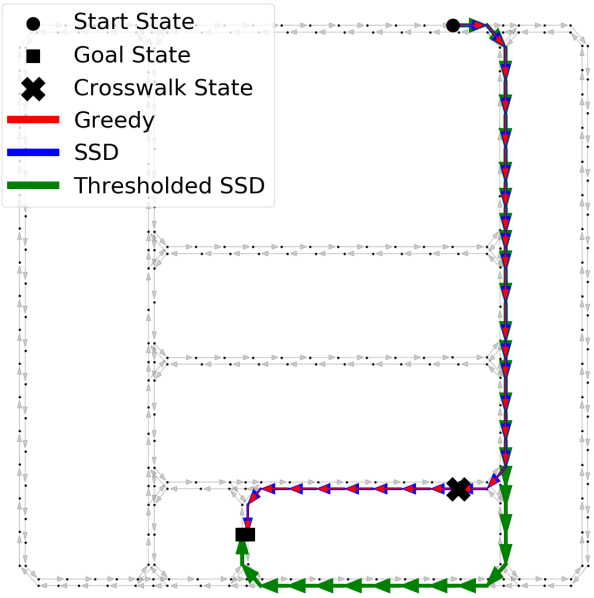


Fig. 4. **Executed paths in Town 1**. The path taken during the final evaluation is shown for each deployment policy. Greedy and SSD methods take the noisy shortest path in all trials. Thresholded SSD takes the second-shortest path to avoid unwanted stochasticity in nearly all trials, which is visualized here.


## Acknowledgements

This research was supported by the Office of Naval Research, grant N00014-20-1-2570.

## REFERENCES


[1] N. Roy, I. Posner, T. Barfoot, P. Beaudoin, Y. Bengio, J. Bohg, O. Brock, I. Depatie, D. Fox, D. Koditschek, *et al.*, "From machine learning to robotics: Challenges and opportunities for embodied intelligence," *arXiv preprint arXiv:2110.15245*, 2021.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[6] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 651–673.

[7] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[8] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on Robot Learning*. PMLR, 2017, pp. 262–270.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[10] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs. sourceforge. net*, vol. 4, no. 6, p. 2, 2000.

[11] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," *Advances in neural information processing systems*, vol. 27, 2014.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[13] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[14] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 449–458.

[15] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[16] A. Tamar, D. Di Castro, and S. Mannor, "Temporal difference methods for the variance of the reward to go," in *International Conference on Machine Learning*. PMLR, 2013, pp. 495–503.

[17] R. Keramati, C. Dann, A. Tamkin, and E. Brunskill, "Being optimistic to be conservative: Quickly learning a cvar policy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4436–4443.

[18] J. Martin, M. Lyskawinski, X. Li, and B. Englot, "Stochastically dominant distributional reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6745–6754.

[19] D. Dentcheva and A. Ruszczyński, "Inverse stochastic dominance constraints and rank dependent expected utility theory," *Mathematical Programming*, vol. 108, no. 2, pp. 297–311, 2006.

[20] M. Dupuis, M. Strobl, and H. Grezlikowski, "Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks," in *Proc. of the Driving Simulation Conference Europe*, 2010, pp. 231–242.