To Go or Not to Go: A Case for Q-Learning at Unsignalized Intersections

David Isele [1 2] **Akansel Cosgun** [2]

Abstract

Autonomous driving at intersections with traffic lights and stop signs can be handled by simple rules, however unsignalized intersections remain a challenging problem. We explore the effectiveness of using Deep Q Networks to handle such problems. Combining several recent advances in Deep RL, were we able to learn policies that surpass the performance of a commonly-used rule based approach in several metrics including task completion time and goal success rate. Although the results were promising, the fact that learned policies resulted in collisions, although rarely, suggest a need for further research.

1. Introduction

A majority of intersections in urban environments are signaled: traffic lights, stop signs and roundabouts are successful mechanisms at regulating the traffic. For an autonomous vehicle, passing through signaled intersections is a trivial problem that can be solved by rule-based methods. For instance, the vehicle should stop when the red light is on and continue its path when the green is on. The problem is not that trivial for unsignalized intersections: The autonomous agent has to analyze the oncoming traffic and decide when it is safe to pass and execute the move in a timely manner. Automation of unsignalized intersection handling can increase the safety of passengers, through driver assistance or as part of a fully automated system.

A number of rule-based strategies have been used at unsignalized intersection handling, including cooperative and heuristic approaches. Cooperative approaches require vehicle-to-vehicle or vehicle-to infrastructure communication, which is currently not widely supported. Several rule-based methods use a Time-to-Collision (TTC) metric (Minderhoud & Bovy, 2001), which is a widely used heuristic as a safety indicator in the automotive industry (Vogel, 2003). Variants of the TTC approach have been used for autonomous driving (Ferguson et al., 2008) and the DARPA urban challenge, where hand engineered hierarchical state machines were a popular approach to handle intersections (Urmson et al., 2008).

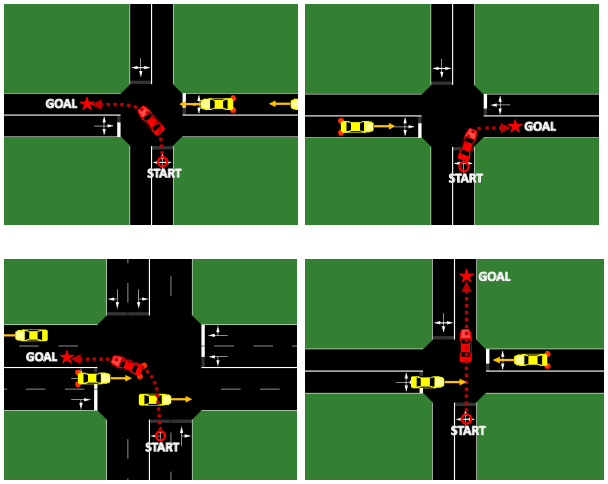

Figure 1. We investigate intersection handling with an autonomous vehicle (red car) in intelligent traffic (yellow cars) in various scenarios. No traffic lights or stop signs are present and the path is known beforehand. The autonomous agent, through randomized experiences in simulation, attempts to learn when it is safe to go. 4 of the scenarios are depicted: *Left* (top left), *Right* (top right), *Left2* (bottom left) and *Forward* (bottom right).

While rule-based methods such as TTC are sufficient in simple cases, they have their limitations. Most models assume constant velocity or acceleration, which is an oversimplification of driver intent. Moreover, actions of agent affect the behaviors of other agents and this interaction is ignored. These reasons motivate our investigation of a machine learning based approach. Only very recently, machine learning methods have been explored for decision making at intersections (Song et al., 2016; Brechtel et al., 2014). (Bojarski et al., 2016) learns a policy that maps images to actions from human demonstrations. (Bouton et al., 2017) uses Partially Observable Monte Carlo Planning (POMCP), but rely on the existence of a generative

[1]University of Pennsylvania, Philadelphia, USA [2]Honda Research Institute, Mountain View, California, USA. Correspondence to: David Isele <isele@seas.upenn.edu>, Akansel Cosgun <akansel.cosgun@gmail.com>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

model for traffic participant behavior. In this work, we explore using model-free reinforcement learning for the problem that does not require expert demonstrations.

Modern autonomous driving systems often employ several levels of abstraction (Cosgun et al., 2017; Paden et al., 2016). This allows for a wide variety of state and action representations for handling unsignalized intersections. An exploratory search of representations showed the selection of the representation had a significant impact on the agent's ability to learn.

In this paper, we are interested in evaluating the effectiveness of Deep Reinforcement Learning in the domain of intersection handling. As seen in Figure 3, the autonomous vehicle is stopped at an unsignalized intersection where the traffic is flowing in both directions. The path is assumed to be given by a higher level process. We consider two action space representations: In **Sequential** representation, the agent learns what acceleration profile to apply along the path. In **Time-To-Go** representation, the agent learns when to depart. We investigate how DQN for both representations compare to a rule-based algorithm in five different intersection simulation scenarios. We consider three metrics for comparison: success rate, collision rate and time to complete the intersection.

2. Approach

We view intersection handling as a reinforcement learning problem, and use a Deep Q Network (DQN) to learn the state action value Q-function.

2.1. Reinforcement Learning

In the reinforcement learning framework, at time t an agent in state s_t takes an action a_t according to the policy π. The agent transitions to the state s_{t+1}, and receives a reward r_t. The problem is typically formulated as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of states, and $\mathcal{A}$ is the set of available actions to the agent, $P : \mathcal{S} \times \mathcal{S} \to [0,1]$ describes the systems dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ maps state-action pairs to a real valued reward and $\gamma \in (0,1]$ is a discount factor in case of infinite time horizons. The goal of reinforcement learning is to maximize the expected return $R = \sum_{t=0}^{T} \gamma^t r_t$ over a sequence of actions. We use Q-learning to perform this optimization.

2.2. Q-learning

In Q-learning (Watkins & Dayan, 1992), the action value function $Q^{\pi}(s,a)$ is the expected return $\mathbb{E}[R_t | s_t = s, a]$ for a state-action pair following a policy π. Given an optimal value function $Q^*(s,a)$ the optimal policy can be inferred by selecting the action with maximum value $\max_a Q^*(s,a)$ at every time step.

In Deep Q-learning (Mnih et al., 2013), the optimal value function is approximated with a neural network $Q^*(s,a) \approx Q(s,a;\theta)$ with parameters θ. The action value function is learned by iteratively minimizing the error between the expected return and the state-action value predicted by the network. We use n-step return(Peng & Williams, 1996) , dynamic frame skipping and experience replay to improve the learning process.

An agent that selects actions over extended time periods can improve the learning time of an agent (Srinivas et al., 2017). Dynamic frame skipping can viewed as a simplified version of options, which is recently starting to be explored by the Deep RL community. (Jaderberg et al., 2016; Tessler et al., 2016; Kulkarni et al., 2016).

We use the experience replay mechanism to break the correlations of sequential trajectories. An experience replay buffer stores previous trajectories which can be sampled during learning. A benefit of using experience replay is that important sequences which happen less frequently can be preferentially sampled (Schaul et al., 2016). We use the simplified approach proposed by (Jaderberg et al., 2016) which avoids the computation of a rank list.

2.3. Action Representations

We present two action representations that we found promising:

- **Sequential:** The agent determines to accelerate, decelerate, or maintain constant velocity at every point in time along the desired path. 12 actions are available: the 3 types of actions, at 4 time scales (1, 2, 4 and 8 steps).
- **Time-to-Go:** The agent determines the timing of departure through a sequences of actions to *wait* or *go*, meaning every trajectory is a series of *wait* actions terminating in a *go* action. 5 actions are available: The *go* action and *wait* action at 4 time scales (1,2, 4 and 8 steps).

For TTC and the Time-to-Go DQN, after the algorithm has decided the path is clear it follows the Intelligent Driver Model.

2.4. State Representations

A top view of space is discretized into a grid in Cartesian coordinates relative to the ego car's reference frame. Every vehicle in the space is represented by it's heading angle, velocity, and calculated time to collision. The heading angle, velocity, and calculated time to collision are all represented as real values.

For the sequential action DQN, space is represented as a 5×11 grid discretizing 0 to 20 meters in front of the car

and ± 90 meters to the left and right of the car. Each spatial pixel, if occupied, contains the normalized real valued heading angles, velocity, and calculated time to collision. The $5 \times 11 \times 3$ representation results in a 165 dimensional space.

For the Time-to-Go DQN, space is represented as a 18×26 grid in global coordinates. Unlike the sequential action DQN, this representation does not use the calculated time to collision for any vehicles.

2.5. Neural Network Setup

The Sequential action DQN network is a fully connected networks with leaky ReLU (Maas et al., 2013) activation functions. The network consists of 3 hidden layers each of 100 nodes each and a final linear layer with 12 outputs, corresponding to the available actions. The Time-to-Go DQN network uses a convolutional neural network with two convolution layers, and one fully connected layer. The first convolutional layer has 32 6×6 filters with stride two, the second convolution layer has 64 3×3 filters with stride 2. The fully connected layer has 100 nodes. All layers use leaky ReLU activation functions. The final linear output layer has five outputs for each action. Our experience replay buffers store $100,000$ time steps. We have two buffers, one for collisions and one for both successes and timeouts. At each learning iteration we samples 25 steps from each buffer for a total batch size of 50.

Each sequential action scenario was trained on 1M simulations. Each Time-to-go scenario was trained on 250k simulations. This difference was in order to roughly balance the runtime of the experiments. The epsilon governing random exploration was decayed from 1.0 to 0.05 linearly over half the number of iterations. For the reward we used $+1$ for successfully navigating the intersection, -10 for a collision, and -0.01 step cost. Both networks are optimized using the RMSProp algorithm (Tieleman & Hinton, 2012).

3. Experiments

We train two separate DQNs (Sequential Actions and Time-to-Go) and compare the performance against a rule-based baseline algorithm.

Experiments were run using the Sumo simulator (Krajzewicz et al., 2012), which is an open source traffic simulation package. This package allows users to model road networks, road signs, traffic lights, a variety of vehicles, and pedestrians. Traffic cars follow IDM to control their motion, therefore they react to each other and to the autonomous vehicle. In Sumo, randomness is simulated by varying the speed distribution of the vehicles and by using parameters that control driver imperfection.

Table 1. Comparison of Different Algorithms

Scenario	Metric	*Random*	*TTC*	*DQN-Sequential*	*DQN-Time-to-Go*
Right	% Success	66.06	99.61	99.5	**99.96**
	% Collisions	9.96	**0.0**	0.47	0.04
	Avg. Time	13.2	6.46s	5.47s	**4.63s**
Left	% Success	45.9	99.7	**99.99**	**99.99**
	% Collisions	35.9	**0.0**	**0.00**	0.01
	Avg. Time	13.82s	6.97s	5.26s	**5.24s**
Left2	% Success	45.45	99.42	99.79	**99.99**
	% Collisions	26.15	**0.0**	0.11	0.01
	Avg. Time	14.48s	7.59s	7.13s	**5.40s**
Forward	% Success	66.20	**99.91**	99.76	99.78
	% Collisions	17.9	**0.0**	0.14	0.01
	Avg. Time	12.88s	6.19s	**4.40s**	4.63s
Challenge	% Success	29.99	39.2	82.97	**98.46**
	% Collisions	41.45	**0.0**	1.37	0.84
	Avg. Time	15.7s	12.55s	9.94s	**7.94s**

We ran experiments using five different intersection scenarios: *Right*, *Left*, *Left2*, *Forward* and a *Challenge*. Each of these scenarios is depicted in Figure 1. The *Right* scenario involves making a right turn, the *Forward* scenario involves crossing the intersection, the *Left* scenario involves making a left turn, the *Left2* scenario involves making a left turn across two lanes, and the *Challenge* scenario involves crossing a six lane intersection with increased traffic density. Each scenario is run for 10k simulations.

Each lane has a 45 miles per hour (20 m/s) max speed. The car begins from a stopped position. Each time step is equal to 0.2 seconds. The max number of step per trial is capped at 100 steps which is equivalent to 20 seconds. The traffic density is set by the probability that a vehicle will be emitted randomly per second. We use 0.2 for all scenarios except the challenge scenario where it is set to 0.7.

We evaluate each method according to 3 metrics:

- **Percentage of successes:** the percentage of the runs the car successfully reached the goal. This metric takes into account both collisions and time-outs.
- **Percentage of collisions:** a measure of the safety of the method.
- **Average time:** how long it takes a successful trial to run to completion.

The Time-to-Collision (TTC) policy serves as a baseline in our analysis. It uses a single threshold to decide when

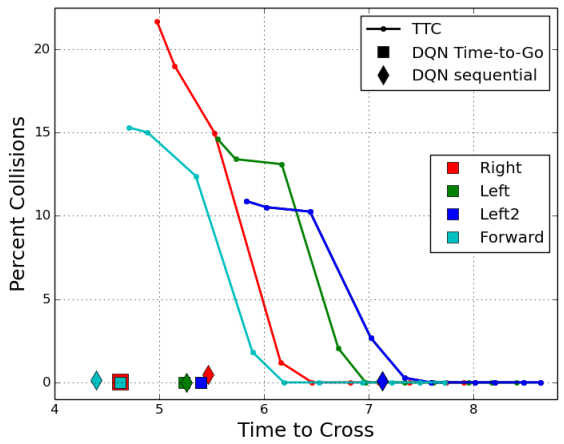

Figure 2. Trade-off between the time to cross and collision rate as the TTC threshold is varied. Note that performance of the DQN dominates in every case. The challenge scenario is excluded for scale reasons, but the results are similar.

to cross. Consider an imaginary line emanating from the front of the ego vehicle, aligned with the longitudinal axis. We calculate the TTC with another vehicle as the time it takes for the vehicle to reach this imaginary line, assuming it will travel with constant speed. Among all the vehicles in the scene, we consider the one with the minimum TTC value. If this value exceeds the TTC threshold, then the ego vehicle starts the crossing phase and follows the Intelligent Driver Model (IDM) (Treiber et al., 2000) until the goal is reached. If it doesn't exceed the threshold, the ego car continues to *wait*. The TTC threshold is chosen for each of the scenarios to yield the best results for the baseline.

4. Results

Table 1 shows the results from our experiments. The TTC method did not have a collision in any of the scenarios, given that a large enough threshold was chosen. All other methods had non-zero collision rates for all scenarios, except DQN-Sequential for the *Left* scenario. Among DQN methods, DQN Time-to-Go had substantially lower collision rate than DQN-sequential. For all scenarios except *Challenge*, DQN Time-to-Go had only 7 collisions in a total of 40k simulations.

We see that both DQN methods are substantially more efficient at reaching the goal than TTC. DQN Time-to-Go has the best task completion time in all scenarios, except *Forward*, where DQN-Sequential is faster. On average, DQN Time-to-Go was %28 faster in reaching to goal than TTC, whereas DQN Sequential was %19 faster than TTC. Therefore, the DQN methods have potential to reduce traffic jams due to their efficiency navigating intersection.

While the DQNs are more efficient, they are seldom able to minimize the number of collisions as successfully as TTC. This is due to the fact that TTC has a tunable parameter that adjusts the safety margin and we tune TTC to the lowest threshold that gives zero collisions. The few collisions that do occur for DQNs seem to relate to discretization effects, where the car nearly misses the oncoming traffic.

Figure 3. Challenge Scenario. The DQN policy begins accelerating in anticipation of the clear path. TTC would have waited until all cars were clear, missing the opportunity.

Comparing the DQN's performance against the TTC curve as we trade off speed vs. safety (Figure 2), we see that in every instance the DQN's performance dominates the performance of TTC. This suggests that it is possible to design an algorithm that has zero collision rate, but with better performance metrics than TTC.

4.1. Challenge Scenario

An interesting result was that TTC did not reach the goal the majority of the time in the *Challenge* scenario, reaching the goal only %39.2 of the time, and posting a success rate only slightly more than Random (%29.9). For the same scenario, DQN Time-to-Go reached the goal %98.46 of the time, significantly outperforming other methods.

Comparing the DQN and TTC, the DQN strategies take into account predictive behavior of the traffic. The DQNs can accurately predict that traffic in distant lanes will have passed by the time the ego car arrives. In contrast, TTC does not leave until all cars have cleared its path . In addition, TTC leaves a sufficient safety margin for oncoming cars in distant lanes. As a result TTC often waits until the road is completely clear, missing opportunities to cross.

4.2. Sequential vs Time-To-Go

Comparing the two DQN methods, the Sequential representation allows for more complex behaviors: the agent could potential slow down half way through the intersection and wait for on coming traffic to pass. The Time-to-Go representation focuses on the departure time, allowing us to specifically probe how changes in departure time can affect performance. We think Time-To-Go representation

is preferable since it performed slightly better, and allows easier abstraction with high level behavior planners.

5. Conclusion

Unsignalized intersection handling remains a hard task for autonomous vehicles, mainly because of unpredictable agent behavior. Rule-based intersection handling methods offer reliable and easy-to-interpret solutions, however result in sub-optimal behavior and task performance.

We showed a first system that uses Deep Q-Networks for the specific problem of intersection handling. By making use of the latest Deep RL techniques, we were able to build networks that, in some metrics, outperform a commonly used rule-based algorithm based on the Time-to-Collision (TTC) heuristic. While TTC achieved zero collision rate for all cases, DQN performed better on task efficiency and success rate.

We saw that determining the time to go is the most important part of the task, even in more complex environments when the ability to change speeds through the intersection might be beneficial. This can greatly reduce the complexity of the high level behavior planners.

References

Bojarski, Mariusz, Del Testa, Davide, Dworakowski, Daniel, Firner, Bernhard, Flepp, Beat, Goyal, Prasoon, Jackel, Lawrence D, Monfort, Mathew, Muller, Urs, Zhang, Jiakai, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Bouton, Maxime, Cosgun, Akansel, and Kochenderfer, Mykel J. Belief state planning for navigating urban intersections. *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

Brechtel, Sebastian, Gindele, Tobias, and Dillmann, Rüdiger. Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 392–399. IEEE, 2014.

Cosgun, Akansel, Ma, Lichao, Chiu, Jimmy, Huang, Jiawei, Demir, Mahmut, Anon, Alexandre Miranda, Lian, Thang, Tafish, Hasan, and Al-Stouhi, Samir. Towards full automated drive in urban environments: A demonstration in gomentum station, california. *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

Ferguson, Dave, Baker, Christopher, Likhachev, Maxim, and Dolan, John. A reasoning framework for autonomous urban driving. In *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 775–780. IEEE, 2008.

Jaderberg, Max, Mnih, Volodymyr, Czarnecki, Wojciech Marian, Schaul, Tom, Leibo, Joel Z, Silver, David, and Kavukcuoglu, Koray. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Krajzewicz, Daniel, Erdmann, Jakob, Behrisch, Michael, and Bieker, Laura. Recent development and applications of SUMO–simulation of urban mobility. *International Journal on Advances in Systems and Measurements (IARIA)*, 5(3–4), 2012.

Kulkarni, Tejas D, Narasimhan, Karthik, Saeedi, Ardavan, and Tenenbaum, Josh. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 3675–3683, 2016.

Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

Minderhoud, Michiel M and Bovy, Piet HL. Extended time-to-collision measures for road traffic safety assessment. *Accident Analysis & Prevention*, 33(1):89–97, 2001.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Paden, Brian, Čáp, Michal, Yong, Sze Zheng, Yershov, Dmitry, and Frazzoli, Emilio. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

Peng, Jing and Williams, Ronald J. Incremental multi-step q-learning. *Machine learning*, 22(1-3):283–290, 1996.

Schaul, Tom, Quan, John, Antonoglou, Ioannis, and Silver, David. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2016.

Song, Weilong, Xiong, Guangming, and Chen, Huiyan. Intention-aware autonomous driving decision-making in an uncontrolled intersection. *Mathematical Problems in Engineering*, 2016, 2016.

Srinivas, Aravind, Sharma, Sahil, and Ravindran, Balaraman. Dynamic action repetition for deep reinforcement learning. *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

Tessler, Chen, Givony, Shahar, Zahavy, Tom, Mankowitz, Daniel J, and Mannor, Shie. A deep hierarchical approach to lifelong learning in minecraft. *arXiv preprint arXiv:1604.07255*, 2016.

Tieleman, Tijmen and Hinton, G. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Tech. Rep*, 2012.

Treiber, Martin, Hennecke, Ansgar, and Helbing, Dirk. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805, 2000.

Urmson, Chris, Anhalt, Joshua, Bagnell, Drew, Baker, Christopher, Bittner, Robert, Clark, MN, Dolan, John, Duggins, Dave, Galatali, Tugrul, Geyer, Chris, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

Vogel, Katja. A comparison of headway and time to collision as safety indicators. *Accident analysis & prevention*, 35(3):427–433, 2003.

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.