Evaluating the robustness of DQN and QR-DQN under domain randomization

Youri Zwetsloot – Y.Zwetsloot@student.tudelft.nl

CSE3000 – Research project

Professor: Frans Oliehoek
Supervisor: Mustafa Celikok

Background

In **reinforcement learning**, or **RL**, an agent learns to make decisions by interacting with an environment or **domain**, receiving feedback in the form of rewards or penalties.

One of the first techniques to use **Deep Neural Networks**, **or DNNs**, to estimate the overall reward (or **return**), is now known as **Deep Q-Networks** or **DQN**. A variation called **QR-DQN** builds on DQN by estimating the return distribution, instead of just the expected return.

Problem

Robustness is the property of an agent to perform well in environments different from its training environment.

The **sim-to-reality** gap is a related problem that refers to the fact that simulated training environments typically differ a lot from the 'actual' environments, leading to degraded performance.

A common technique to improve robustness and cross the sim-to-reality gap is **domain randomization** (or **DR**): randomizing environment properties during training.

Research question

How does domain randomization affect the robustness of DQN and QR-DQN?

Methodology

We make use of a customizable simulated highway (*highway-env* [1]) environment to train and test DQN and QR-DQN.

We use 3 DR approaches:

1. **Naive**: 6 - 9 vehicles per lane
2. **Difficult**: 8 or 9 vehicles per lane
3. **Multiple properties**: lane count, vehicle count, density and politeness (see Table 1).

To evaluate robustness/*Sim2Real* transfer, we test models (in part) on unseen environments..

Property	Default	Training	Testing
Vehicle count	7	7 - 9	5 - 10
Lane count	3	2 - 3	2 - 6
Density	1.0	1 - 1.2	0.7 - 0.1.3
Politeness	0	0 - 0.5	0 - 1.0

Table 1: Environment property values in the default, training and testing Highway environments. The training and testing environments use domain randomization.

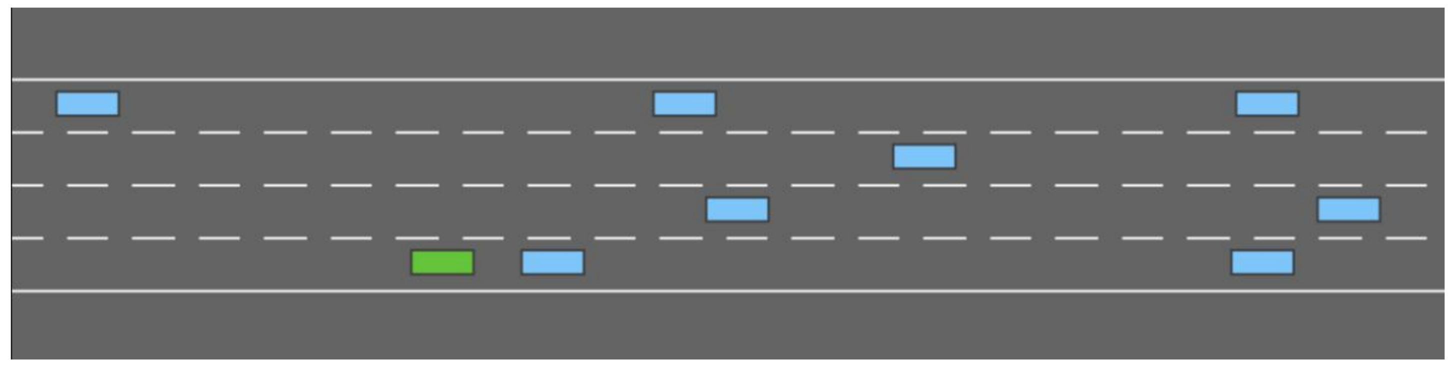

Figure 1: A still of our simulated highway environment. The green 'car' is operated by our agent.

Results

	Reward/step	Length	Crash rate
DQN	**0.78**	**74.2 / 100**	**48%**
QR-DQN	0.75	75.9 / 100	38%
DQN (6 - 9)	0.76	84.1 / 100	34%
QR-DQN (6 - 9)	**0.77**	**80.5 / 100**	**34%**
DQN (8 - 9)	0.75	78.3 / 100	30%
QR-DQN (8 - 9)	**0.75**	**91.3 / 100**	**16%**

*Table 2: **Single property**: metrics for (QR-)DQN, with and without DR. Only the vehicle count is changed between DR environments.*

	Reward/step	Length	Crash rate
DQN	**0.76**	**81.8 / 100**	**34%**
QR-DQN	0.76	86.2 / 100	24%
DQN (DR)	**0.78**	**75.5 / 100**	**38%**
QR-DQN (DR)	0.75	88.0 / 100	22%

*Table 3: **Multiple properties**: metrics for (QR-)DQN, with and without DR. Property values set according to Table 1..*

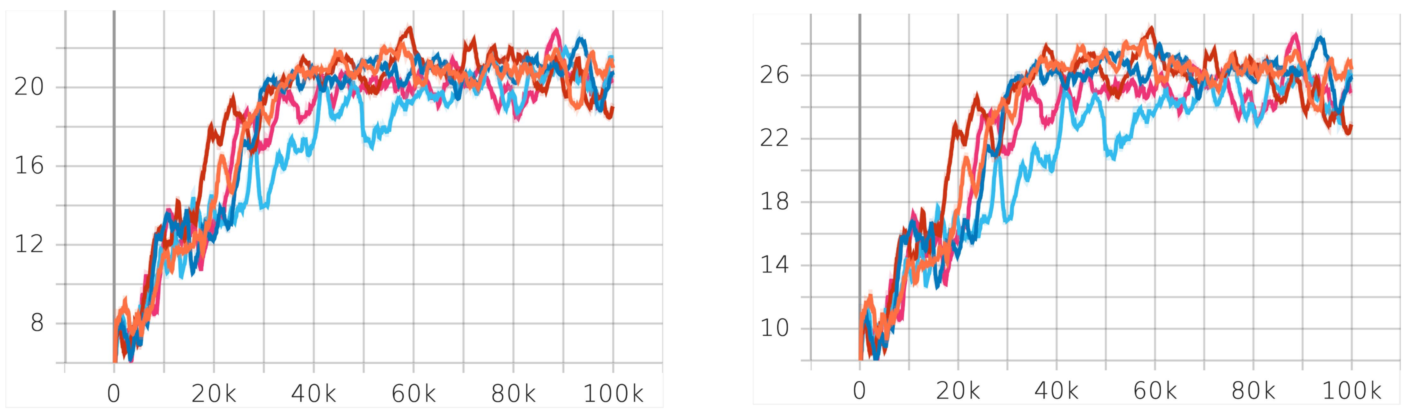

Figure 2: Plots of DQN's return and episode length over 100K steps, when trained without DR. 5 different seeds were used.

Conclusions

1. **QR-DQN** achieves a lower crash count, **DQN** a higher reward/step (risk vs. reward)
2. Difficult to get DR right...
3. ...but DR *can* improve robustness and achieve Sim2Real transfer
4. Focus on **hard** environments

Delft University of Technology

References

[1] Edouard Leurent. An Environment for Autonomous Driving Decision-Making. *https://github.com/eleurent/highway-env. 2018.*