Robust Deep RL for Autonomous Driving

Guillaume Chhor, Shreyash Pandey, Vivekkumar Patel
Stanford University

Introduction

- Self-driving cars: topic of great interest in AI field
- Use of Deep Reinforcement Learning to solve this task efficiently.
- High dimensional and complex task: use of TORCS Simulator
- We study impact of sensors' noise in RL agent's performance.

TORCS Simulator

Sensor Inputs
TORCS provides 18 different types of sensor inputs that define the agent's state. The following inputs are considered most useful:

- Angle between the car direction and the direction of the track axis.
- Speed of the car along the X,Y and Z axis.
- Vector of 4 sensors representing the rotation speed of wheels.
- Number of rotation per minute of the car engine.
- Distance between the track edge and the car within a range of 200 meters.
- Distance between the car and the track axis.

Visual Display
Training the algorithm using the raw pixels from the visual display as the agent's state was also an option.

Figure 1: Visual display use for training with raw pixels as input

Pros: OpenAI gym like interface. Pre-existing RL experiments available on GitHub.
Cons: Only available for linux. Difficult to run on Azure without display. Training with visual display impossible on Azure.

DDPG Algorithm

To deal with continuous action spaces, DeepMind came up with policy-gradient actor-critic algorithm called Deep Deterministic Policy Gradients (DDPG) that is off-policy and model-free.

1. Stochastic behavior policy for good exploration but estimates a deterministic target policy, which is much easier to learn.
2. Two neural networks, one for the actor and one for the critic. Actor generates actions, critic evaluates them.
3. Deterministic policy gradient updates weights of the actor network. Critic network updated using TD-error signal.

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$$

$$L_{critic} = \frac{1}{N}\Sigma_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

4. DDPG theorem by Silver et al.
$\nabla_{\theta^\mu}\mu \approx \mathbb{E}[\nabla_a Q(s,a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_t}]$

Actor Critic Algorithm

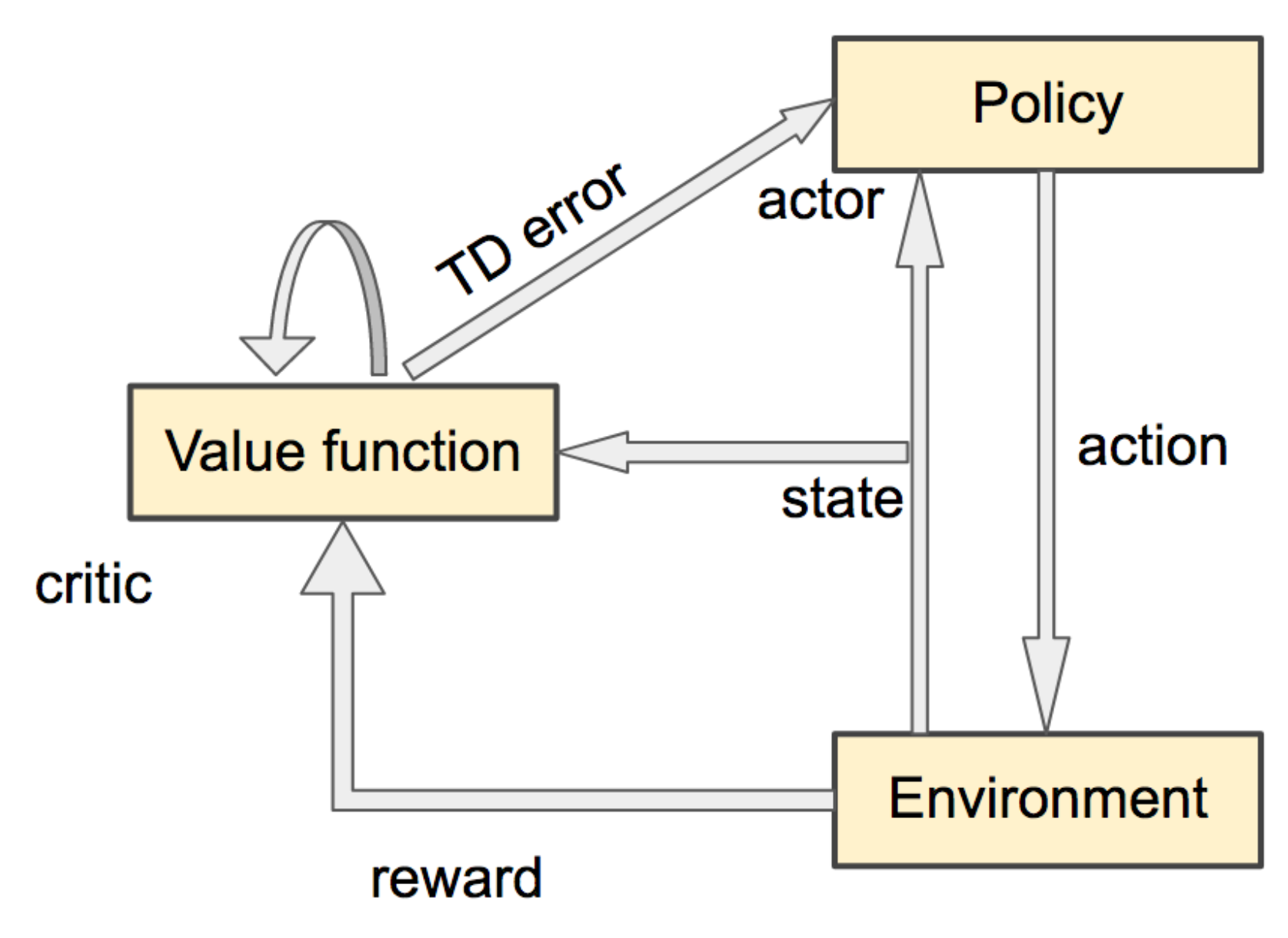

Figure 2: Interactions in a Actor Critic Algorithm

Additive White Gaussian Noise

Circuits and sensors generally suffer from White Gaussian Noise.

- White: Power is same at all frequencies - completely random!.
- Gaussian: At any time t, noise $z(t) \sim \mathcal{N}(0, \sigma)$.

Implementation Details

1. ϵ-Greedy exploration, **prioritized Experience Replay** to break correlations, **target networks** to stabilize training.
2. Actor Network: 2 hidden layers, maps sensor inputs to steering, acceleration and brake values.
3. Critic Network: concatenates state encoding (hidden layer representation for sensor inputs) with actor prediction and calculates Q value.
4. Adam optimizer with a higher learning rate for critic than actor, to make sure updates to actor network are stable.
5. Reward function:
$$R_t = V\cos\theta - V\sin\theta - V|trackPos|$$
6. Intuition: Maximize velocity along the track, minimize velocity along the transverse axis and remain in the center of track if the speed is high.
7. We add noises of different levels during training and testing and experiment with new architectures.

Train and test scores

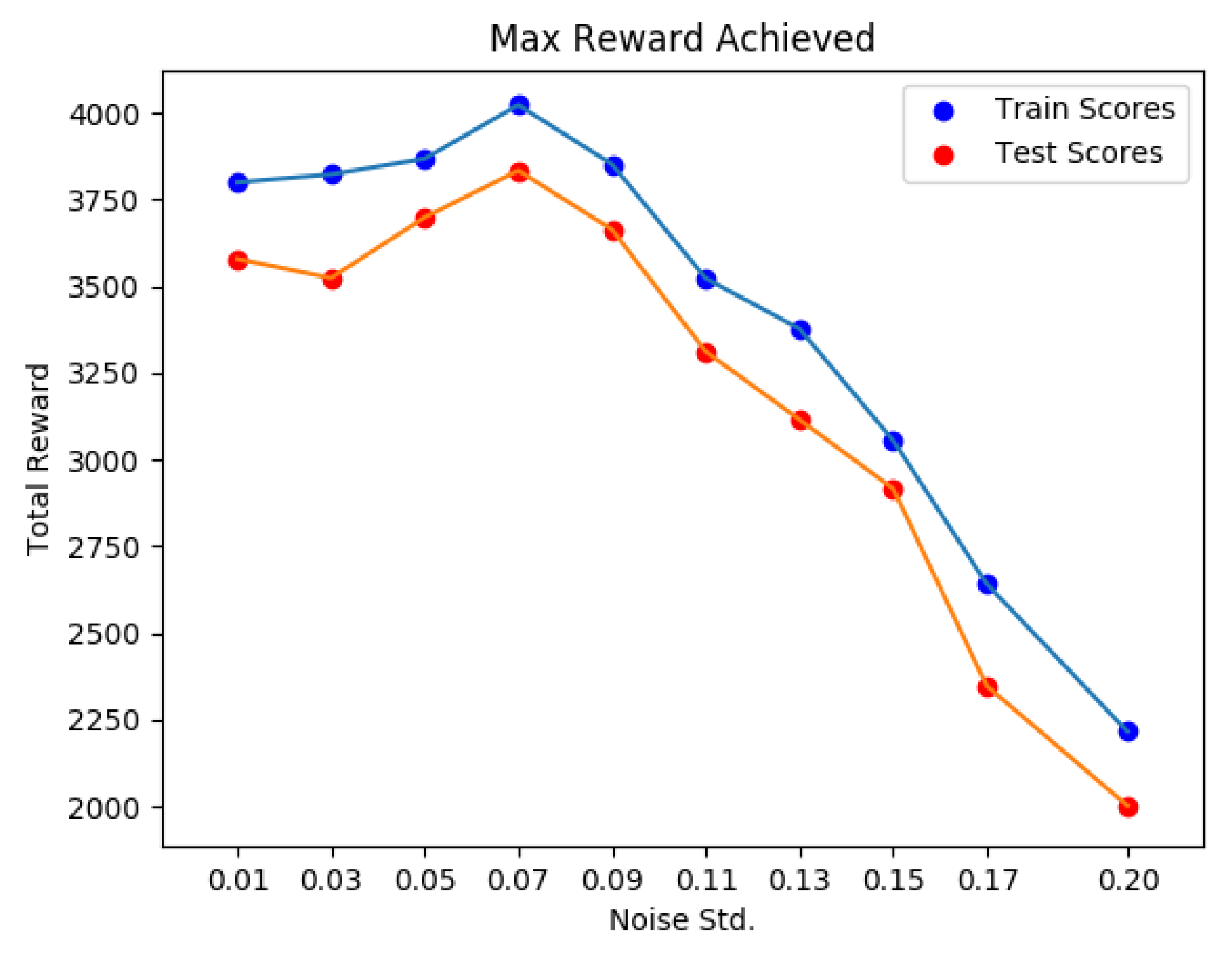

Figure 3: Training and testing with different levels of noise

- X-axis has σ for noise, with inputs normalized to have unit magnitude.
- Training and testing on different tracks.
- A small noise helps perform better.
- Algorithm breaks down with around 20% noise.

Noise / No-noise Comparison

Train | Test	No Noise	Noise ($\sigma = 0.1$)
No Noise	3579	2340
Noise ($\sigma = 0.1$)	3137	**3312**

Table 1: Performance of Algorithms with and without Noise

The results show that the agent trained with noise is, as expected, more robust at test time in presence of noise than an agent trained without noise. More over it performs almost as well as the agent trained without noise in an environment without noise.

Key Observations

- Training with small noise makes the agent more robust and gives better performance.
- Adding extra layers to the network to counter noise doesn't work.
- At 20% noise power, algorithm breaks.
- As noise power increases, it takes longer for the algorithm to converge. Blue curve indicates $\sigma = 0.05$, green indicates $\sigma = 0.13$.

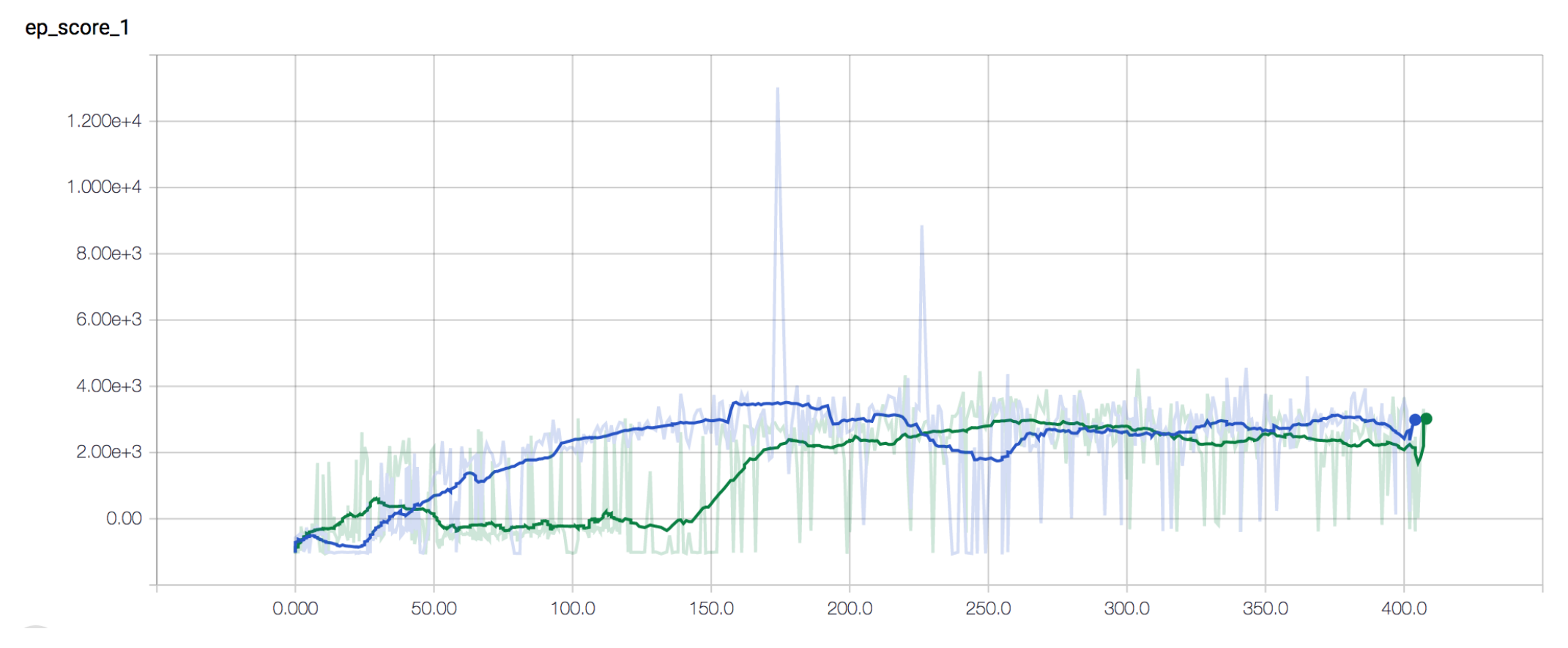

Figure 4: Training Curves with two different noises

Challenges and Future work

- Policy gradient methods prone to local optima. Need repeated runs to see if bad performance due to noise.
- Read multiple samples at a time and feed to the network, hoping that it learns to somehow average and reduce variance of noise.

References

- Silver, David, et al. "Deterministic policy gradient algorithms." ICML. 2014.
- github.com/only4hj/DeepRL