Deep Learning Lab Exercise 4: Reinforcement Learning

[AUTHOR'S NAME REDACTED], Alex Rose (4653309)

***Abstract*—(Figure out later).**

I. CartPole

We begin the classic control task of CartPole. Instead of imitation learning where the agent analyses expert play to learn the relationship between states and ideal actions (as in Exercise 3), here we use Q-Learning, where the agent samples from a palette of actions in each state, and gradually learns to paramaterise a function $Q(State, Action) = Reward$. In the CartPole instance our state is a 4-dimensional vector; our function is a simple neural network consisting of two fully-connected 20-unit hidden layers with ReLU activation; and we have two possible actions: *Left* and *Right*.

We also implemented the following bonus features:

- **Epsilon Annealing**: in which our probability ϵ of selecting a random (non-optimal) action is reduced from its starting value by a factor of 3.33×10^{-4} after each episode. This gradually shifts the agent's priorities from exploration to exploitation, as time goes on.
- **Boltzmann Exploration**, where instead of selecting randomly with probability ϵ and optimally with probability $1 - \epsilon$, the agent samples an action according to the SoftMax of its action predictions (with optional SoftMax temperature parameter)
- **Double-Q learning**, where actions are selected using the *current* network, but still evaluated using the *target* network. This decorrelates the noise in the two networks, theoretically avoiding overestimation bias in the Q function.

Training was over 1,600 episodes with Adam as our optimiser, learning rate 10^{-4}, batch size 64, buffer capacity of $100,000$, $\tau = 0.01$, and $\epsilon = 0.1$ initially. . We see that Boltzmann Exploration creates a high performance agent much more quickly than the Basic or Double Q Learning, presumably because in the early episodes when the agent has little idea of the best move, Boltzmann encourages it to explore *much* more than

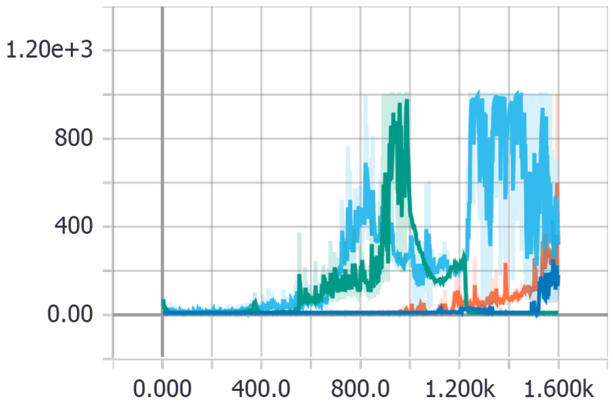

Fig. 1. Reward over training epochs for Cartpole models (*Default*; *Epsilon Annealing*; *Boltzmann Exploration*; *Double Q*)

a fixed epsilon parameter would allow. Epsilon Annealing, on the other hand, starts out very promisingly, but has a disastrous spell later and forgets everything useful it has learned. (Figure 1). We then tested each agent over 100 episodes, and found that the Boltzmann agent has learned to play perfectly with an average score over 1000, and the Double Q agent, though less impressive, has still solved the problem, with an average score over 200 (Figure 2). We should note that these results are very high-variance; in previous runs, we saw agents with identical configurations having very different training curves. Therefore, they should not be taken (for example) as definitive proof that Boltzmann exploration is the best method to solve this problem.

II. Car Racing

Now onto car racing. Again. We reused the same Q learning algorithm, code, and hyperparameters as in CartPole, but with the following modifications to suit the new environment:

- Images preprocessed to greyscale, as per the provided code
- A deep convnet, as described in Table I.
- Continuous actions discretised simply to *Left, Right, Accelerate, Break*
- A replay buffer of size $500,000$.

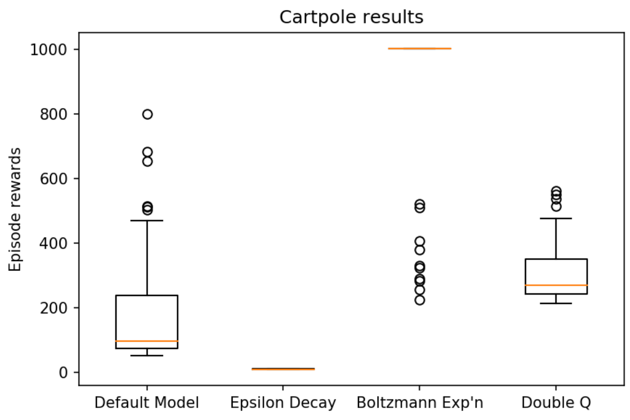

Fig. 2. Test results for Cartpole models

TABLE I

DEFAULT NETWORK ARCHITECTURE

Layer	Parameters
Convolution 1 / ReLU	Filters: 16, Size: 7
Max Pooling	Size: 2, Stride: 1
Convolution 2 / ReLU	Filters: 32, Size: 5
Max Pooling	Size: 2, Stride: 1
Convolution 3 / ReLU	Filters: 48, Size: 3
Max Pooling	Size: 2, Stride: 1
FC layer 1 / ReLU	512 units
FC layer 2 / ReLU	128 units
Output	5 units

- Frameskip of 2 during training (i.e. every third frame is seen and acted upon, and actions persist over the next two frames). At test time, no frameskip is used.

For agent testing, we used two methods:

1) Actions selected using the maximum prediction from the Q network, leading to discrete
2) Actions sampled using the Softmax of the last layer, leading to continuous (and noticeably more ”noisy”) actions.

All seeds (in numpy, tensorflow and the gym environment) were set to 0 to allow reproducability in testing.

Our agents are doing reasonably well overall already, with many average scores over 800. Notably, Boltzmann exploration does much worse here than in Cart-Pole (WHY?), and some agents again ”forget” what they’ve learned during training, and must re-learn it.

We also notice that these agents are far better at recovery than in the imitation learning exercise, since here

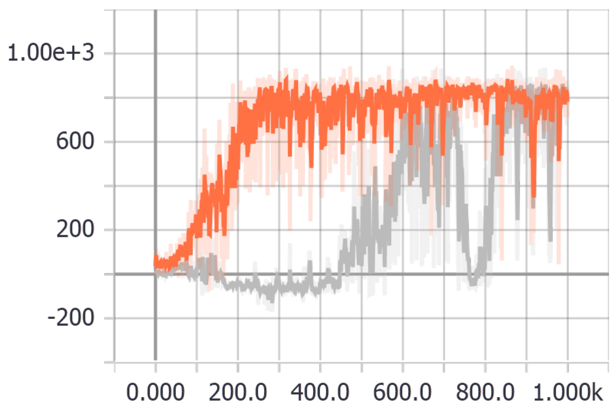

Fig. 3. Training curves for Basic Model and Double Q learning

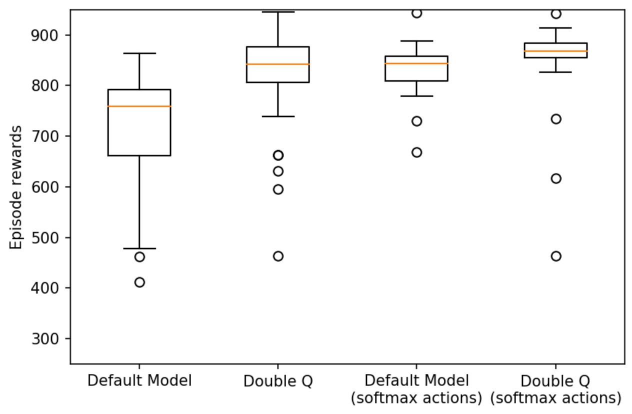

Fig. 4. Test results for Basic Model and Double Q learning

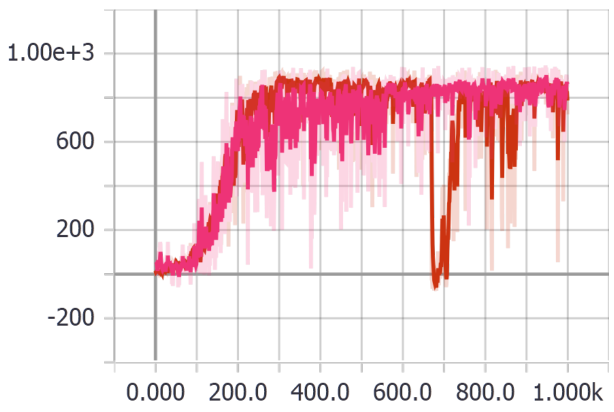

Fig. 5. Training curves for Epsilon Annealing and Boltzmann Exploration

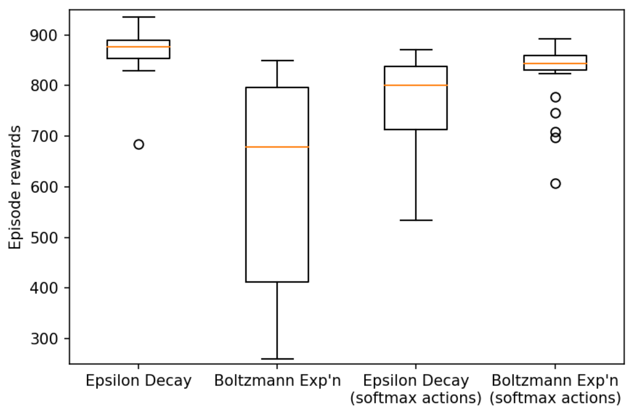

Fig. 6. Test results for Epsilon Annealing and Boltzmann Exploration

they have ample opportunity to learn how to behave in off-road situations (whereas in the imitation learning dataset, there were typically very few examples of the expert's car making a mistake).

Observing our agents' driving in test runs, we saw that their characteristic mistake was to enter turns too fast, cut across the corner (missing some reward tiles), and rejoin the road. They learn to do this manoeuvre quite well, and indeed the Q learning algorithm may be encouraging it: doing so allows the agents to avoid slowing down for the corner, and so they pick up rewards as rapidly as possible over a medium time horizon. But by the end of the episode they leave some tiles uncollected, and the clock runs down, diminishing their final reward.

We also see that the 'Softmax' agents tend to drive much more cautiously, and that there is more "noise" in their actions, such as steering rapidly back and forth on straight stretches of road. This is just as we would expect, since they are sampling from an action space at test time and therefore often picking sub-optimal ones. However, they do seem less likely to enter corners too fast than the 'Hardmax' agents. Overall, softmax action selection is not systematically better or worse.

III. Adding History

We hypothesise that the agents may be making this corner-cutting mistake because they simply don't have a clear idea of how fast they're going from the single-frame input we've used so far. Therefore, we next implemented image history in order for the agent to better judge its speed, in two different ways:

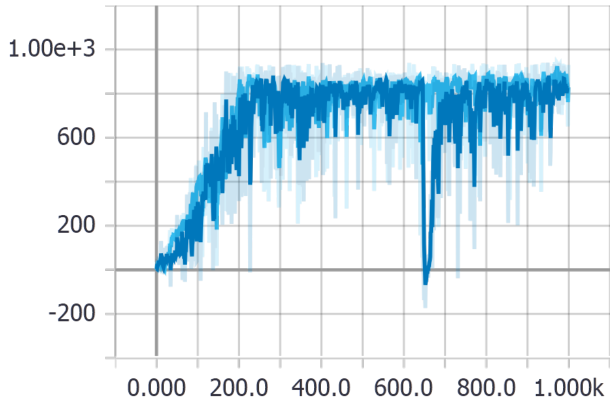

Fig. 7. Training curves for History Frame (dark) and Difference Frame (light)

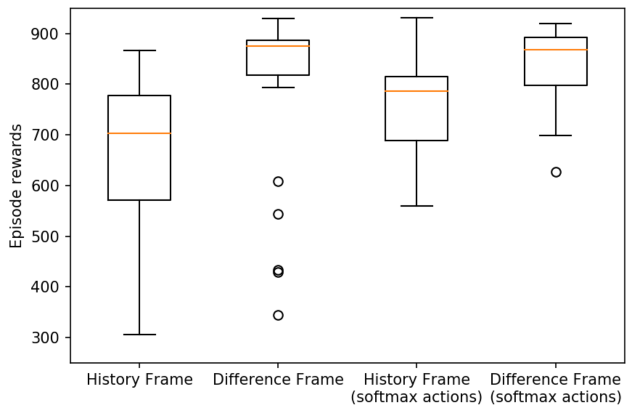

Fig. 8. Test results for History Frame and Difference Frame

1) Adding the frame immediately prior to the current one to our state buffer, so that the input to our convnet is a $(96, 96, \mathbf{2})$ image
2) Subtracting the prior frame from the current one to produce a difference image, and processing this difference image using a separate, smaller convnet (which is merged with the first one in the fully connected layers)

Results are shown in Figures 7 and 8.

Difference frames are good! (SAY SOMETHING MORE INTELLIGENT HERE)

IV. Reward Shaping

Ingmar

V. Improving on the Difference Frame + Penalty Agent

In many episodes, the Difference Frame + Penalty agent performs a perfect run, driving reasonably fast and picking up every reward tile. Good! Sometimes it still misses a few tiles, especially on sharp corners. We see if we can improve this with two modifications to the model:

1) **Larger network**: Doubling the number of filters in every convolution layer, and increasing the number of units in fully connected layers by 50%. We do this with the hypothesis that the agent may be unable to learn complex enough representations with the incumbent model to guide its actions, and in the belief that we should be protected from over-fitting by the continuous stream of new training data supplied to the gradient descent algorithm.
2) **No frameskip**: Training with zero frameskip. We do this with the hypothesis that the agent may be learning sub-optimally when it is locked into action choices that must persist over three frames, and could learn better skills if allowed to change actions on every frame if needed. This training should, of course, take roughly three times as long.

WHAT HAPPENED? TBD

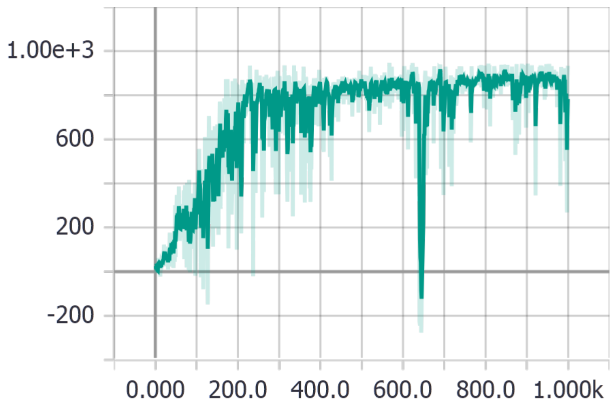

Fig. 9. Training curves for Large Network (xxx) and No Frameskip (xxx)

VI. Conclusion

SOMETHING

VII. Acknowledgements

We would like to thank Guilherme Miotto and Baohe Zhang for all the fruitful discussions, and also the tutors of this course for their valuable input.

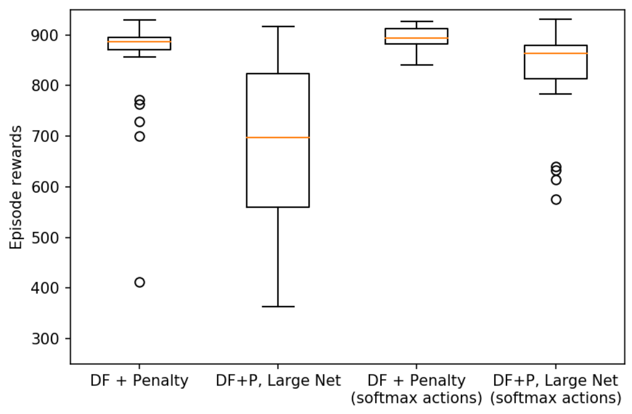

Fig. 10. Test results for Large Network and No Frameskip

TABLE II
FINAL RACECAR TESTING RESULTS

	Captions	Avg Test Score	Avg Score (Softmax)
0	Default Model	701.3	833.1
1	Epsilon Decay	871.0	771.0
2	Boltzmann Exp'n	612.6	828.2
3	Double Q	811.5	845.6
4	History Frame	650.0	754.8
5	Difference Frame	809.5	838.3
6	DF + Penalty	855.0	895.5
7	DF+P, Large Net	669.2	827.3
8	DF+P, No Skip	627.3	229.5

References

[1] AI gym car racing: https://gym.openai.com/envs/CarRacing-v0/

[2] Exercise code: https://github.com/rosea-tf/dl-lab-2018/tree/submit/exercise4_R_NR