

BOOSTING REINFORCEMENT LEARNING WITH EXTREMUM EXPERIENCES

Anonymous authors

Paper under double-blind review

1 SUPPLEMENTARY MATERIAL

1.1 HYPERPARAMETER AND ARCHITECTURE DETAILS

Table 1: Hyperparameters and architectures used in the experiments for our proposed algorithm MaxMin TD Learning, canonical algorithm ϵ -greedy and NoisyNetworks.

Hyperparameters	Settings (For all of the algorithms)
Grey-scaling	True
Observation down-sampling	(84, 84)
Frames stacked	4
Action repetitions	4
Batch Size	32
Update	Double-Q
Max Frames per episode	108000
Exploration epsilon	0.01
Evaluation exploration epsilon	0.01
Exploration epsilon decay frame fraction	0.008
Gradient error bound	0.03125
Learning rate	0.00025
Optimizer epsilon	$0.01/32^2$
Optimizer	Adam
Discount factor	0.99
Maximum absolute rewards	1
Number of iterations	40
Number of training frames	10^4
Nesterov Momentum	True
Hardware	GPU
NoisyNetwork parameter	0.1
Q -Network channels	32,64,64
Q -Network filter size	$8 \times 8, 4 \times 4, 3 \times 3$
Q -Network stride	(4, 4), (2, 2), (1, 1)
Q -Network hidden units	512

For reproducibility and completeness in research in Table 1 we report the hyperparameter details for our proposed algorithm MaxMin TD Learning, canonical algorithm ϵ -greedy and NoisyNetworks. Furthermore, for all of the algorithms the hyperparameters and the architectures are identical with each other. Note that the architecture parameters are also identical for the 200 million frame training. Note that we did not tune hyperparameters reported below. To increase transparency in research we kept hyperparameters exactly the same with the prior studies. We ran our experiments with the JAX implementation from Bradbury et al. (2018). We used Haiku Hennigan et al. (2020) for the neural network library, Optax Hessel et al. (2020) for the optimization library, and RLax for the reinforcement learning library Babuschkin et al. (2020).

1.2 ARCADE LEARNING ENVIRONMENT RESULTS

Table 2 reports the average scores for human, random, our proposed algorithm MaxMin TD Learning, canonical algorithm ϵ -greedy and NoisyNetworks for all of the games in the Arcade Learning Environment 100K benchmark. Scores are reported with the mean over 5 random seeds. The highest score amongst the three algorithms is marked with bold font. We also reported human scores and random scores to provide complete information on the learning curves reported in the main body of the paper.

Table 2: Average returns for human, random, our proposed algorithm MaxMin TD Learning, canonical algorithm ϵ -greedy and NoisyNetworks across all of the games in the Arcade Learning Environment 100K benchmark. Scores are averaged over 5 random seeds.

Games	Human	Random	ϵ -greedy	NoisyNetworks	MaxMin TD Learning
Alien	7127.7	227.8	498.47	466.50	595.70
Amidar	1719.5	5.8	42.31	42.83	41.94
Assault	742.0	222.4	396.00	375.72	383.25
Asterix	8503.3	210.0	306.36	305.64	410.07
BankHeist	753.1	14.2	15.72	13.1	13.45
BattleZone	37187.5	2360.0	1844.61	1100.00	2200.00
Boxing	12.1	0.1	7.25	4.6	7.9
Breakout	30.5	1.7	4.83	5.33	9.03
ChopperCommand	7387.8	811.0	639.48	919.83	987.83
CrazyClimber	35829.4	10780.5	10075.00	18550.00	9870.00
DemonAttack	1971.0	152.1	1365.60	576.5	745.00
Freeway	29.6	0.0	0.00	5.1	14.00
FrostBite	4334.7	65.2	184.41	167.60	206.40
Gopher	2412.5	257.6	633.84	468.66	664.00
Hero	30826.4	1027.0	1628.42	1884.50	1528.40
Jamesbond	302.8	29.0	21.73	22.08	19.33
Kangaroo	3035.0	52.0	251.00	90.00	280.83
Krull	2665.5	1598.0	2206.02	2040.50	1491.5
KungFuMaster	22736.3	258.5	7116.94	5665.00	4045.00
Mspacman	6951.6	307.3	719.16	912.83	671.08
Pong	14.6	-20.7	-8.18	-2.3	-6.30
PrivateEye	69571.3	24.9	0.25	-7.50	30.0
Qbert	13455.0	163.9	519.71	556.08	466.83
RoadRunner	7845.0	11.5	3600.85	1527.35	670.66
Seaquest	42054.7	68.4	206.49	333.66	348.00
UpNdnDown	11693.2	533.4	1858.41	1948.00	1953.91

Also further note that in 6 games¹ simple double- Q learning already outperforms Rainbow in the low data regime. Note that Rainbow has several additional components such as dueling network, multi-step return, distributional reinforcement learning that introduces new parameters as large as the number of bins used in the algorithm, and NoisyNetworks. Thus, the fact that MaxMin TD Learning with simple double- Q learning achieves a higher score in these games than an algorithm that combines all these various techniques is further evidence that demonstrates the NoisyNetworks can be replaced with the MaxMin TD Learning algorithm in Rainbow as a future research direction to obtain better performance. Also further note that MaxMin TD Learning does not introduce any additional new parameters as NoisyNetworks does; more precisely, NoisyNetworks doubles the number of parameters used in the Q -network. Hence, the fact that MaxMin TD Learning achieves higher performance as also reported in the main body of the paper without any additional computational cost further demonstrates the benefits of the utilization of MaxMin TD Learning in a more diversified portfolio of algorithms as a zero cost exploration technique.

¹Boxing, Breakout, ChopperCommand, DemonAttack, Gopher, Pong

1.3 ADDITIONAL RESULTS ON THE MOTIVATING EXAMPLE

Figure 1 reports the learning curves for the motivating example provided in the main body of the paper with variations in ϵ . These results combined with the results reported in Figure 1 in the main body of the paper again further demonstrate that in any given range of ϵ MaxMin TD learning results in substantially faster learning compared to ϵ -greedy.

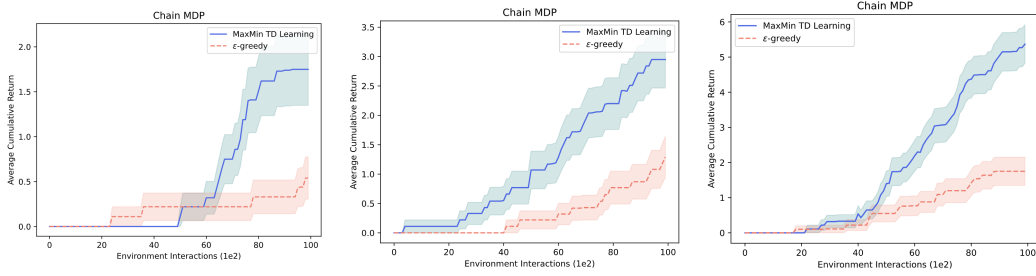


Figure 1: Learning curves in the chain MDP with our proposed algorithm MaxMin TD Learning, the canonical algorithm ϵ -greedy with variations in ϵ . Left: $\epsilon = 0.01$. Center: $\epsilon = 0.03$. Right: $\epsilon = 0.05$.

1.4 EXTENSION OF MAXMIN TD LEARNING TO DIFFERENT REINFORCEMENT LEARNING ALGORITHMS

While the main paper focuses on Deep Double Q -Network, Figure 2 reports human normalized median and human normalized 80th percentile scores for MaxMin TD Learning and ϵ -greedy with the QRDQN algorithm² across all of the tasks of the Arcade Learning Environment 100K benchmark. With MaxMin TD Learning QRDQN is able to achieve 0.15582 human normalized median score. Note that data-efficient Rainbow can only achieve 0.12 human normalized median scores van Hasselt et al. (2019). Furthermore, note that Rainbow contains dueling architecture, multi-step return, noisy networks on top of the distributional reinforcement learning. Thus, the fact that QRDQN, a baseline distributional reinforcement learning algorithm, can achieve human normalized median score that is already substantially higher than data-efficient Rainbow once more demonstrates the substantial sample efficiency gained by the MaxMin TD Learning algorithm. Furthermore, the significantly higher performance gain obtained by MaxMin TD Learning over all tasks of the Arcade Learning Environment 100K benchmark as reported via the human normalized scores in Figure 2, once more demonstrates that MaxMin TD Learning increases the performance of the baseline algorithms further beyond the performance of much more complicated algorithms.

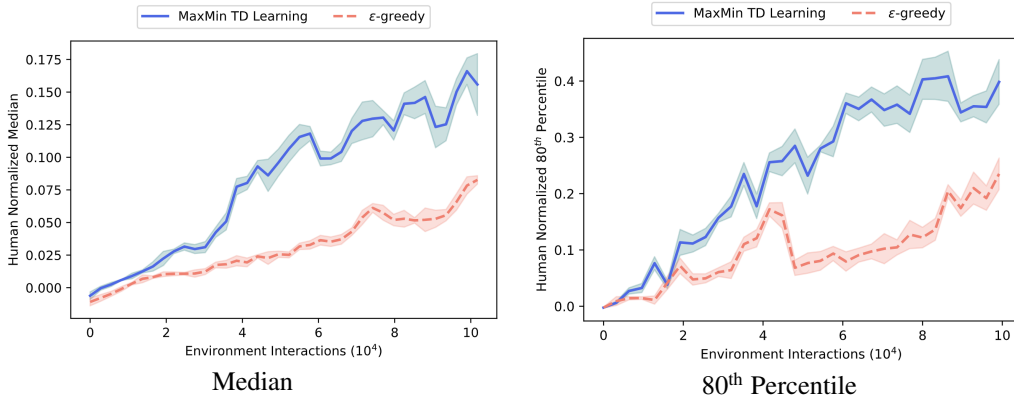


Figure 2: Human normalized median and human normalized 80th percentile scores of QRDQN with MaxMin TD Learning and ϵ -greedy in the Arcade Learning Environment 100K benchmark.

²QRDQN is one of the baseline distributional reinforcement learning algorithms.

REFERENCES

- Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Tom Hennigan, Matteo Hessel, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL <http://github.com/deepmind>.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Neca, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/google/jax>.
- Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL <http://github.com/deepmind/dm-haiku>.
- Matteo Hessel, David Budden, Fabio Viola, Mihaela Rosca, Eren Sezener, and Tom Hennigan. Optax: composable gradient transformation and optimisation, in jax!, 2020. URL <http://github.com/deepmind/optax>.
- Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, pp. 14322–14333, 2019.