Ablation Study of QRDQN

311551059 陳昱丞

Outline

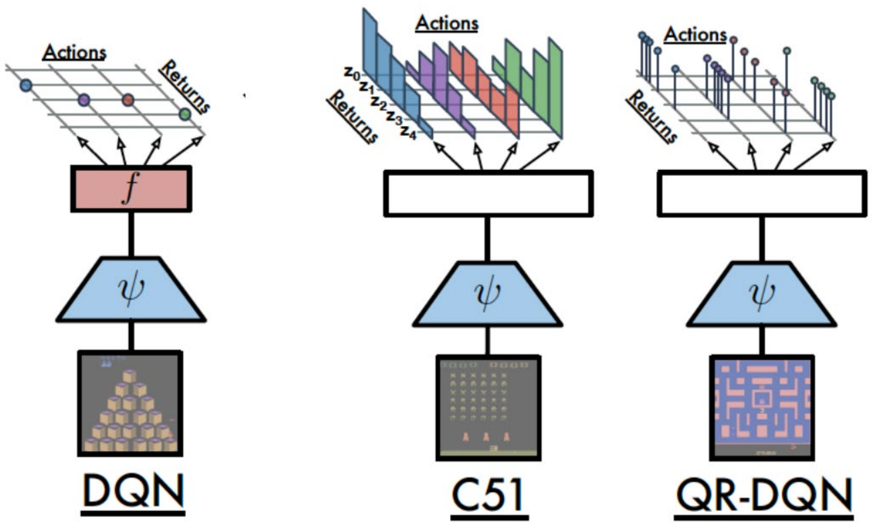
- Background
- Experiments and Result Analysis
- Conclusion
- Appendix
- Reference

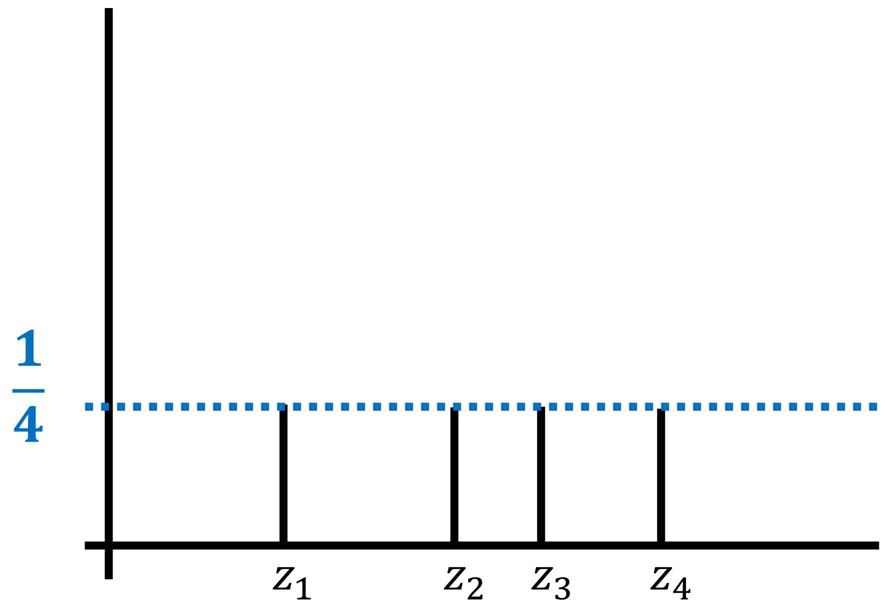
Background

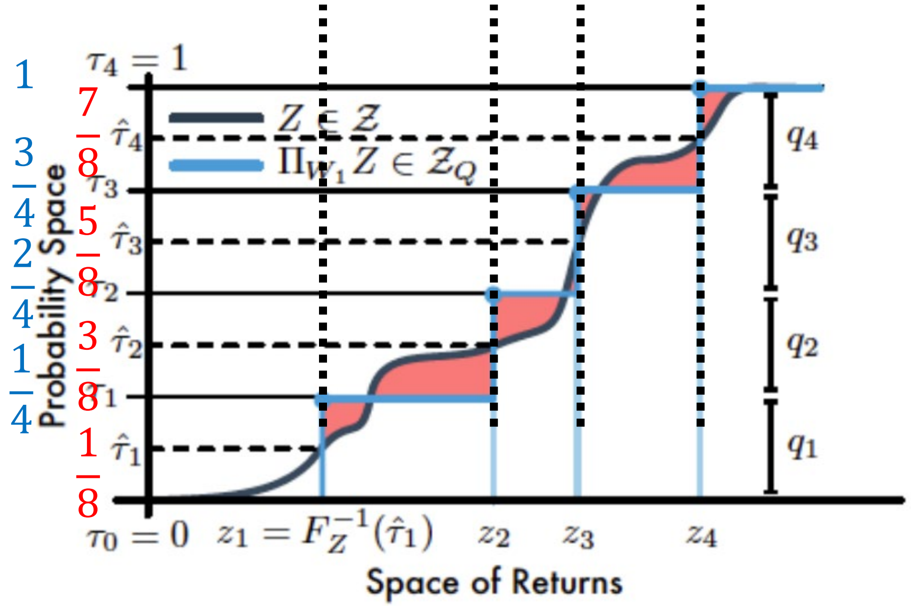
QRDQN

- Learn value distribution $Z(s, a)$ and use $E[Z(s, a)]$ as Q.
- $Q^{\pi}(s, a) = E[Z^{\pi}(s, a)] = E[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$
- Distributional Bellman Equation:
  - $Z^{\pi}(s, a) = r(s, a) + \gamma Z^{\pi}(s', a')$

QRDQN

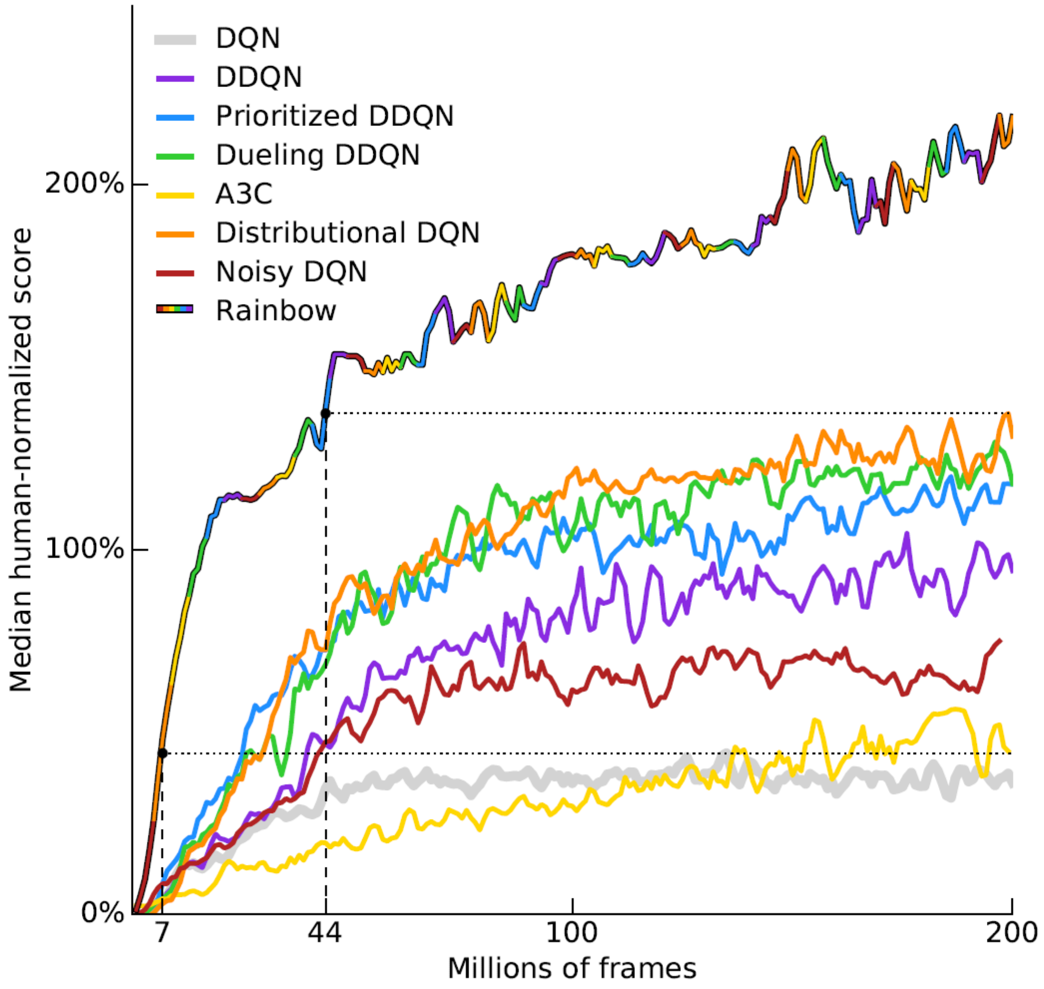

Actions
Returns
DQN
C51
QR-DQN


$\frac{1}{4}$
z_1
z_2
z_3
z_4


$\tau_4 = 1$
$Z \in \mathcal{Z}$
$\Pi_{W_1} Z \in \mathcal{Z}_Q$
q_4
q_3
q_2
q_1
Probability Space
$\tau_0 = 0$
$z_1 = F_Z^{-1}(\hat{\tau}_1)$
z_2
z_3
z_4
Space of Returns
1
$\frac{3}{4}$
$\frac{2}{4}$
$\frac{1}{4}$
$\frac{7}{8}$
$\frac{5}{8}$
$\frac{3}{8}$
$\frac{1}{8}$

Rainbow

- Double Q learning
- Prioritized Experience Replay
- Dueling networks
- Multi-step learning
- Noisy Nets
- Distributional RL
  - C51



Double Q Learning

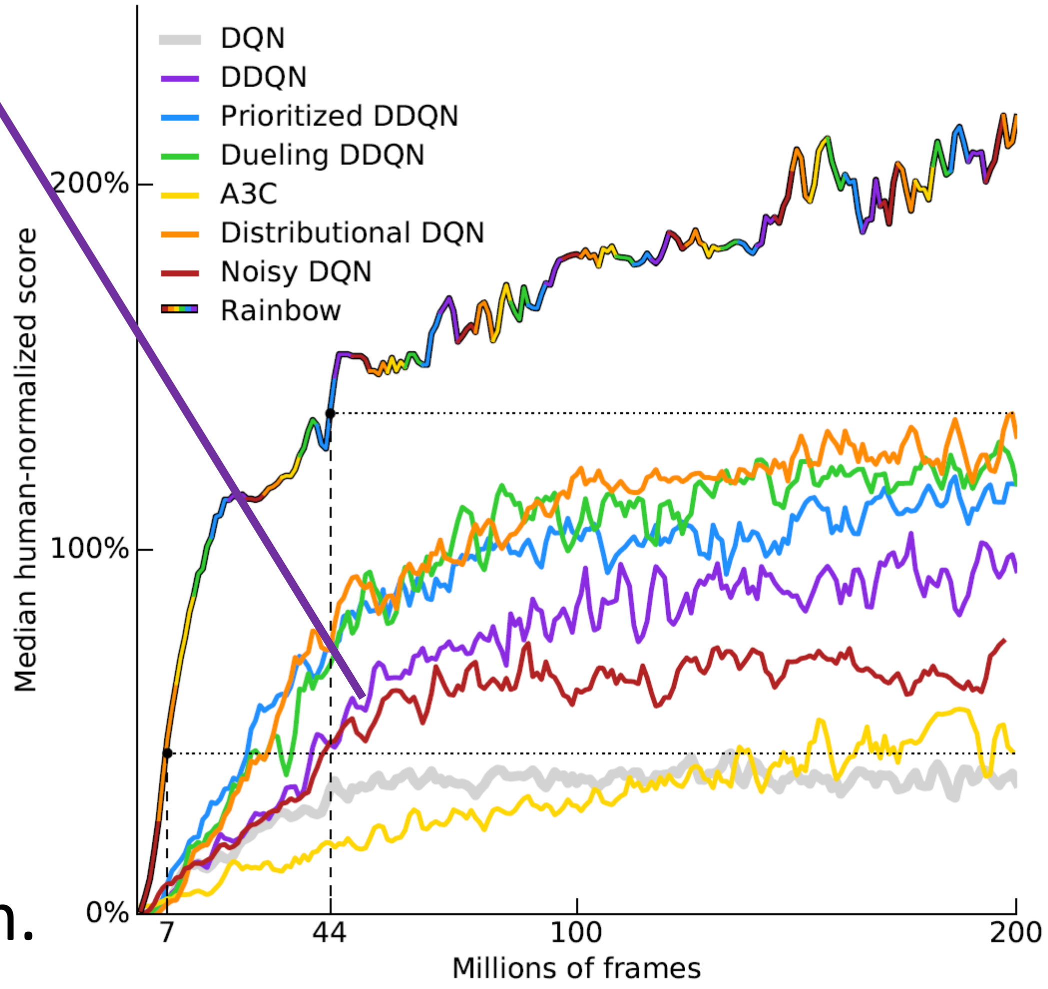
- DQN suffers from over-estimation.

- Behavior and Target network.

$$Y_t^{Q} = r_{t+1} + \gamma \max_a Q(S_{t+1}, \boxed{a} | \theta^-)$$

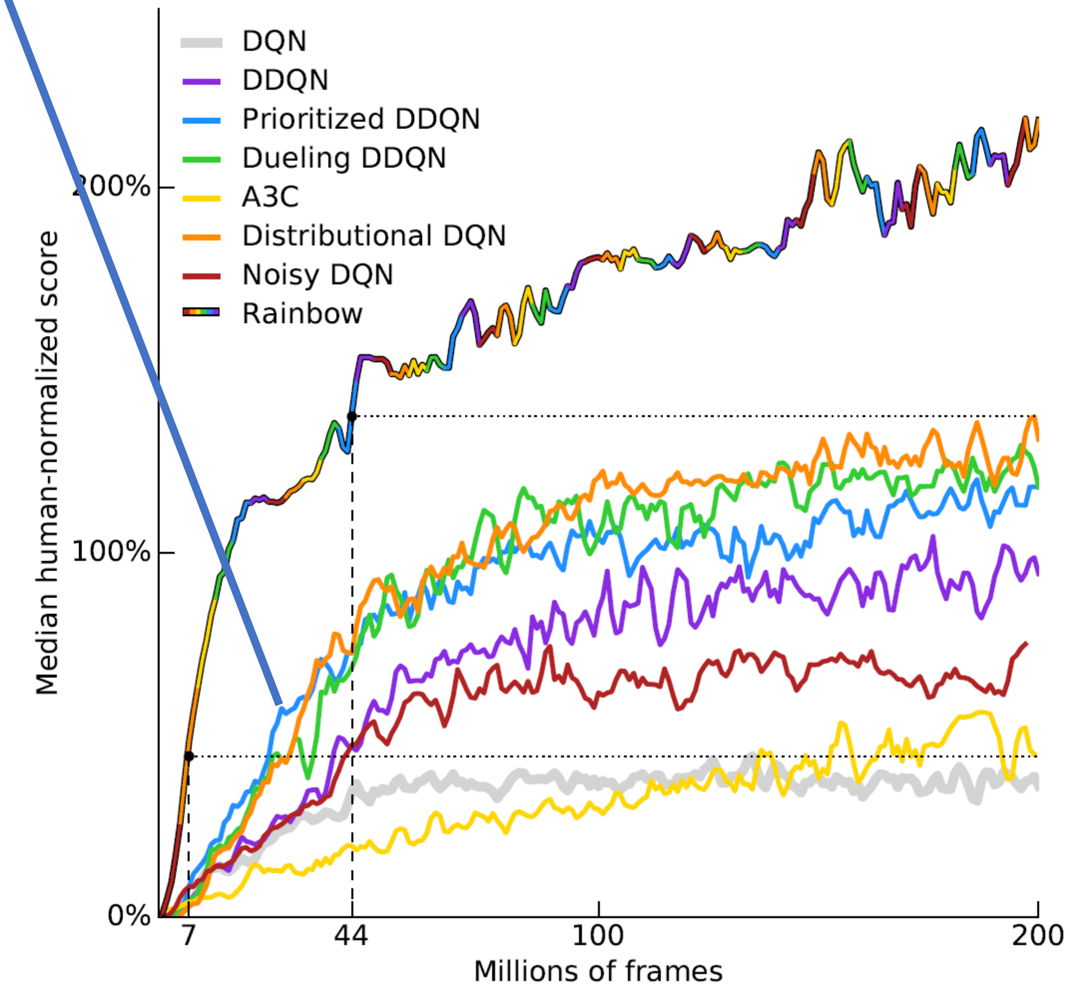
$$Y_t^{DoubleQ} = r_{t+1} + \gamma Q\left(S_{t+1}, \boxed{\underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a | \theta)} | \theta^-\right)$$

- Reduce the over-estimation problem.



Prioritized Experience Replay

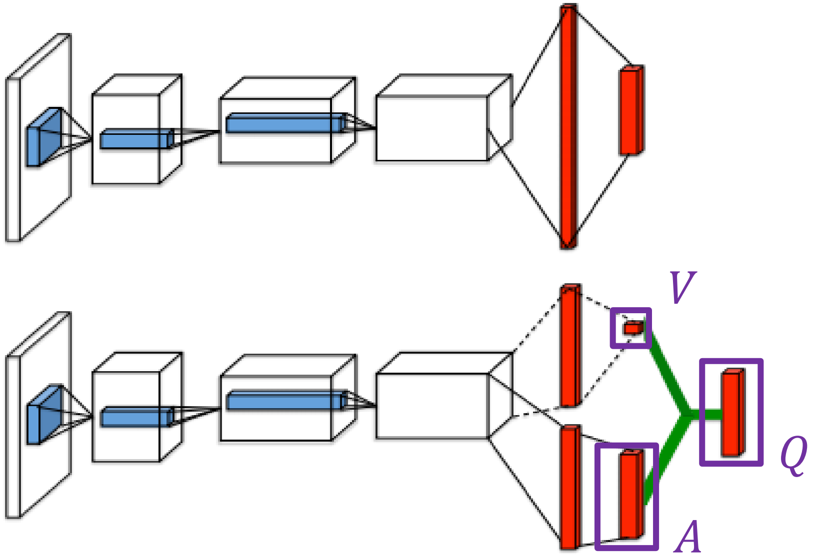
- Sample important transitions from replay buffer more frequently.
- Calculation of importance is based on TD-error.

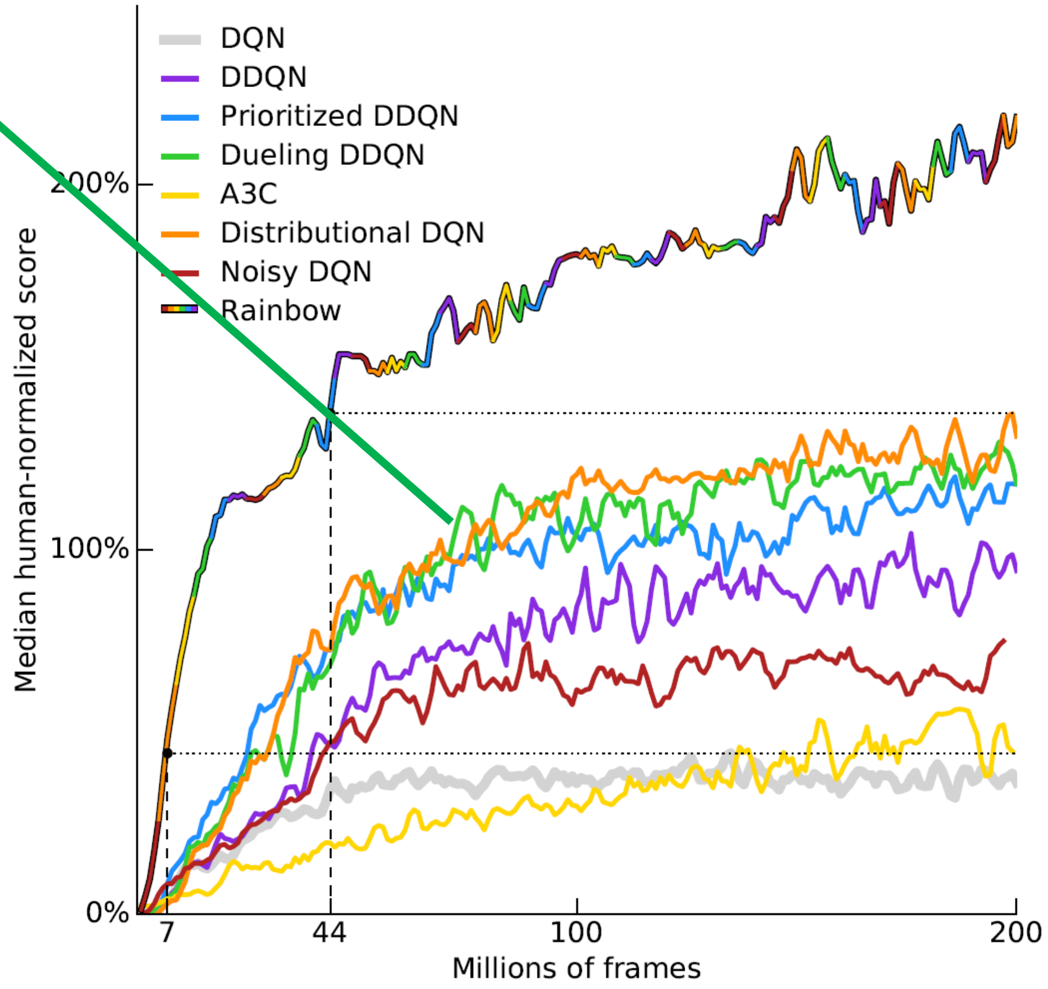


Dueling Networks

- $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$

$\rightarrow Q(s_t, a_t) = A(s_t, a_t) + V(s)$





Dueling Networks

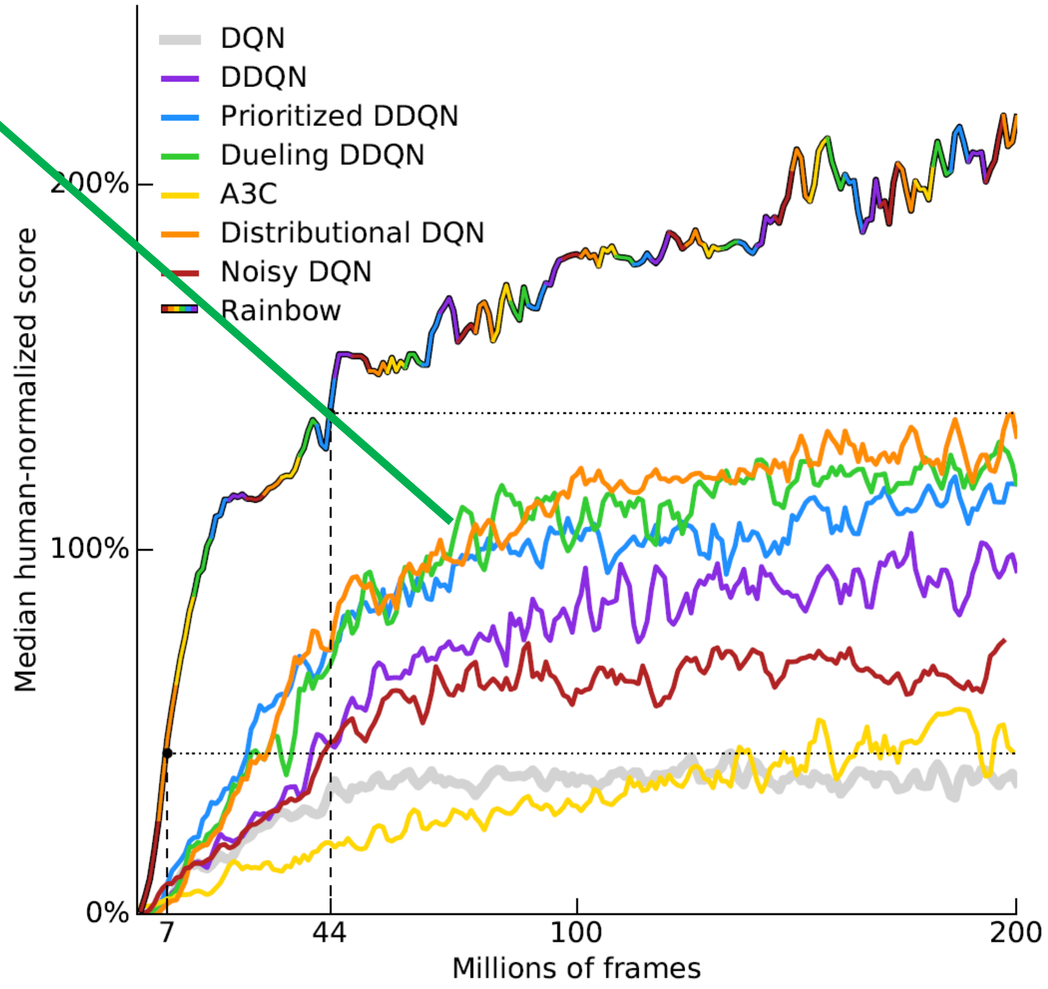
- $E_a[A(s_t, a_t)] = E_a[Q(s_t, a_t) - V(s_t)]$

$$= V(s_t) - V(s_t) = 0$$

- Constrain the value of A:

$$Q(s_t, a_t) = V(s) + A(s_t, a_t) - \frac{1}{|A|} \sum_{a_t'} A(s_t, a_t')$$

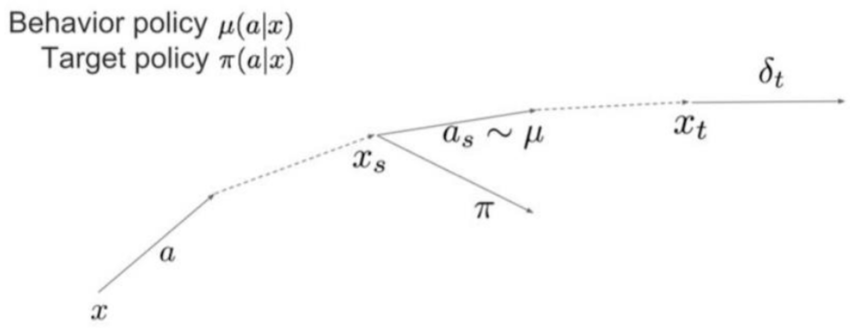
- Due to the relatively small numerical range of the value A, it is more sensitive to model updates, making it easier for the model to consider the relative changes in relation to other actions.

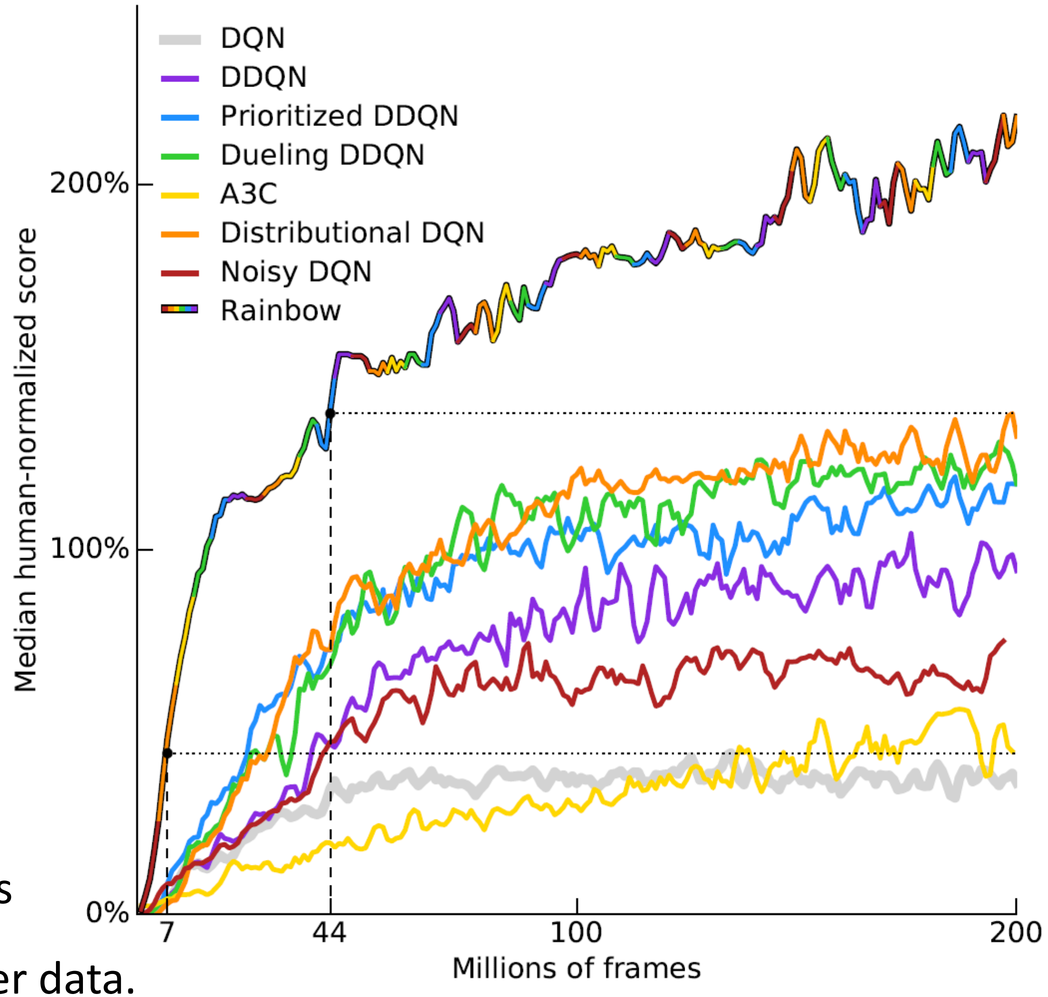


Multi-step Learning

- $R_t^{(n)} = \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}$

- $R_t^{(n)} + \gamma_t^{(n)} \max_{a'} Q_\theta(S_{t+n}, a') - Q_\theta(S_t, A_t)$





- Under different policies, the probability of encountering a specific n-steps trajectory can vary. This discrepancy gives rise to the issue of distribution mismatch in off-policy buffer data.

Noisy Nets

- Replace ϵ-greedy.

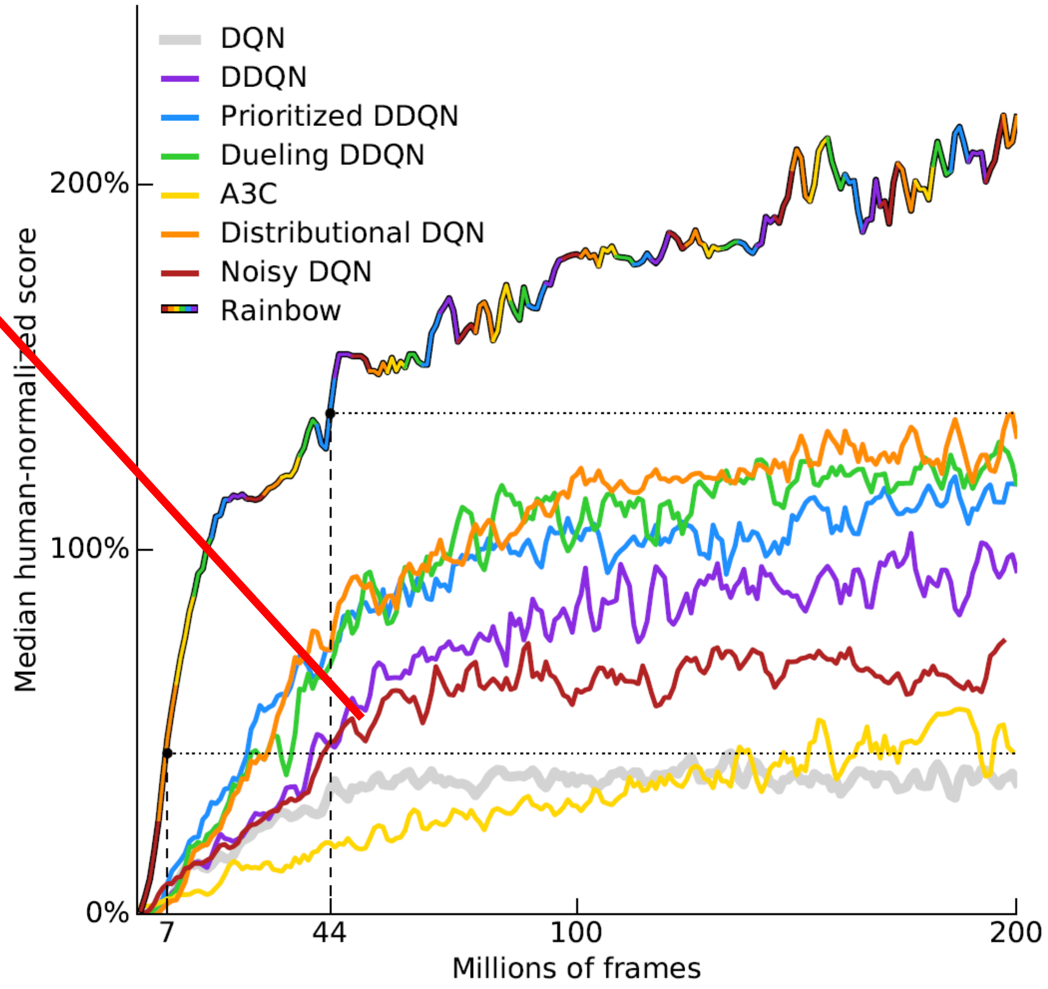
- Noise on Parameters

$$a = arg\max_a \tilde{Q}(s,a)$$

Inject noise into the parameters of Q-function **at the beginning of each episode**

$$Q(s,a) \xrightarrow{\text{Add noise}} \tilde{Q}(s,a)$$

- ϵ-greedy
  - Action is randomly selected.
  - When encountering the same state again, it is possible to choose a completely different action.
- Noisy Nets
  - Noise remains the same within the same episode.
  - More systematic exploration method, the paper refer to it as "State-dependent Exploration“.



How about Rainbow using QRDQN ?

(QR-Rainbow)

Experiments and Result Analysis

Settings

num_steps: 50000000

batch_size: 32

lr: 5e-5

memory_size: 1000000

gamma: 0.99

multi_step: 1

update_interval: 4

target_update_interval: 10000

start_steps: 50000

epsilon_train: 0.1

epsilon_eval: 0.01

epsilon_decay_steps: 250000

double_q_learning: False

dueling_net: False

noisy_net: False

use_per: False

log_interval: 100

eval_interval: 250000

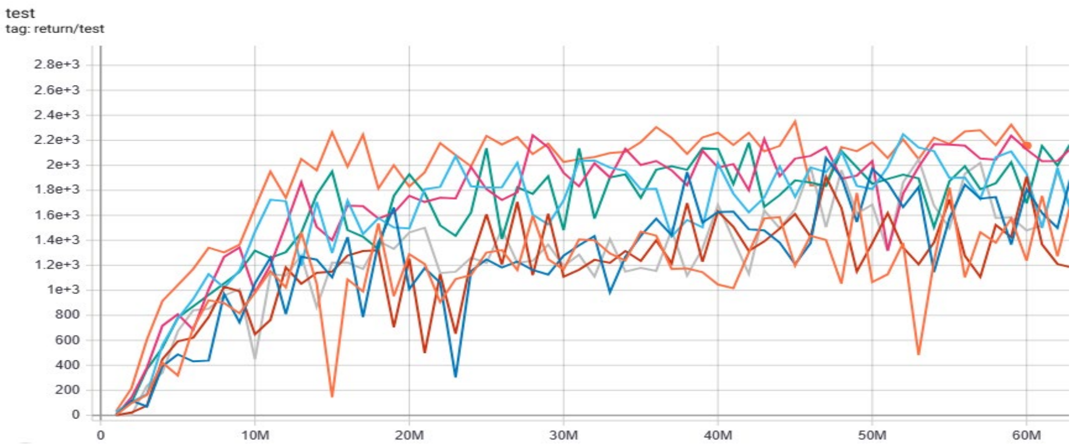
num_eval_steps: 125000

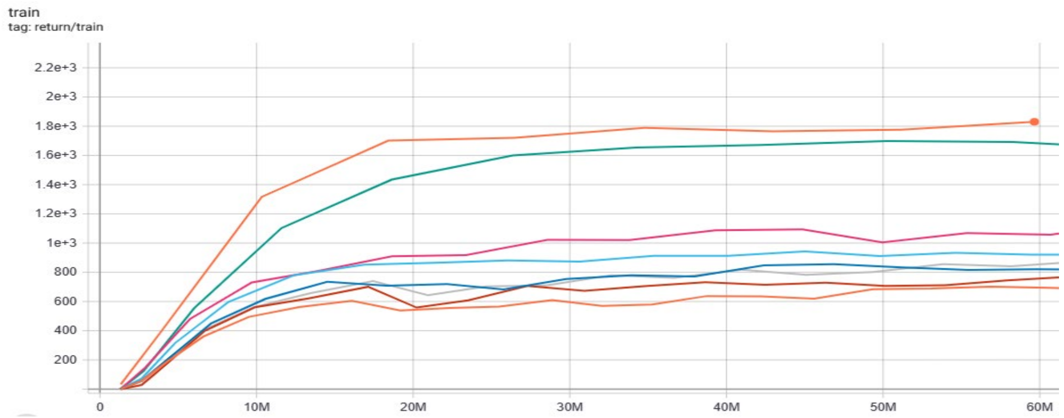
max_episode_steps: 27000

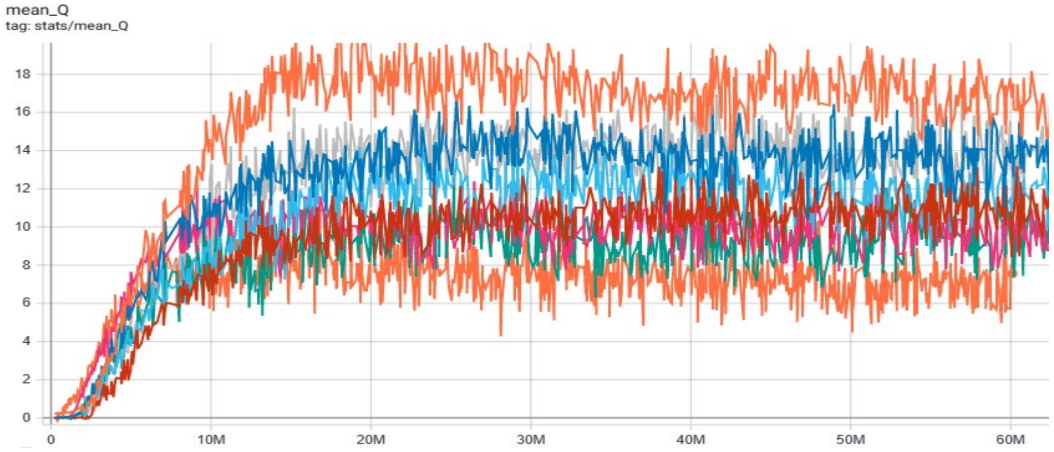
- Environment – Atari games (with NoFrameskip-v4):
  - Breakout
  - Enduro (following slides)
  - MsPacman
- Number of frames:
  - 60M to 80M
- Comparison:
  - DQN
  - QRDQN (N = 200, kappa = 1)
    - QR origin
    - QR + Double Q
    - QR + PER
    - QR + Dueling
    - QR + Noisy
    - QR + Multistep (step=3)
    - QR Rainbow

QRDQN Rainbow

- DQN
- QRDQN
- QRDQN Double-Q
- QRDQN Dueling Network
- QRDQN Multistep Return
- QRDQN Noisy Network
- QRDQN Prioritize Experience Replay
- QRDQN Rainbow

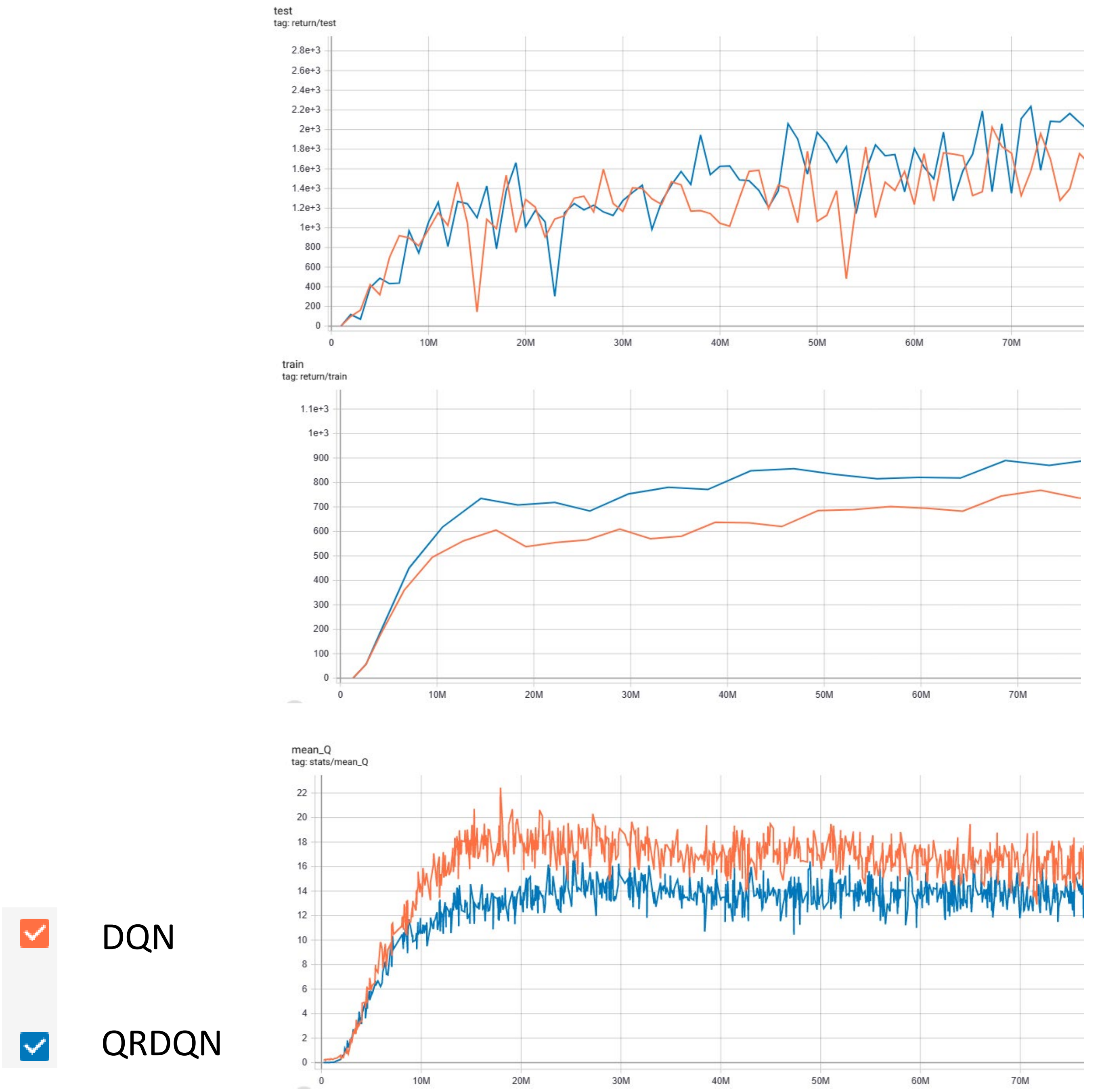






QRDQN vs DQN

- Training:
  - QRDQN > DQN
- Testing:
  - QRDQN > DQN
- Estimation of Q:
  - QRDQN < DQN

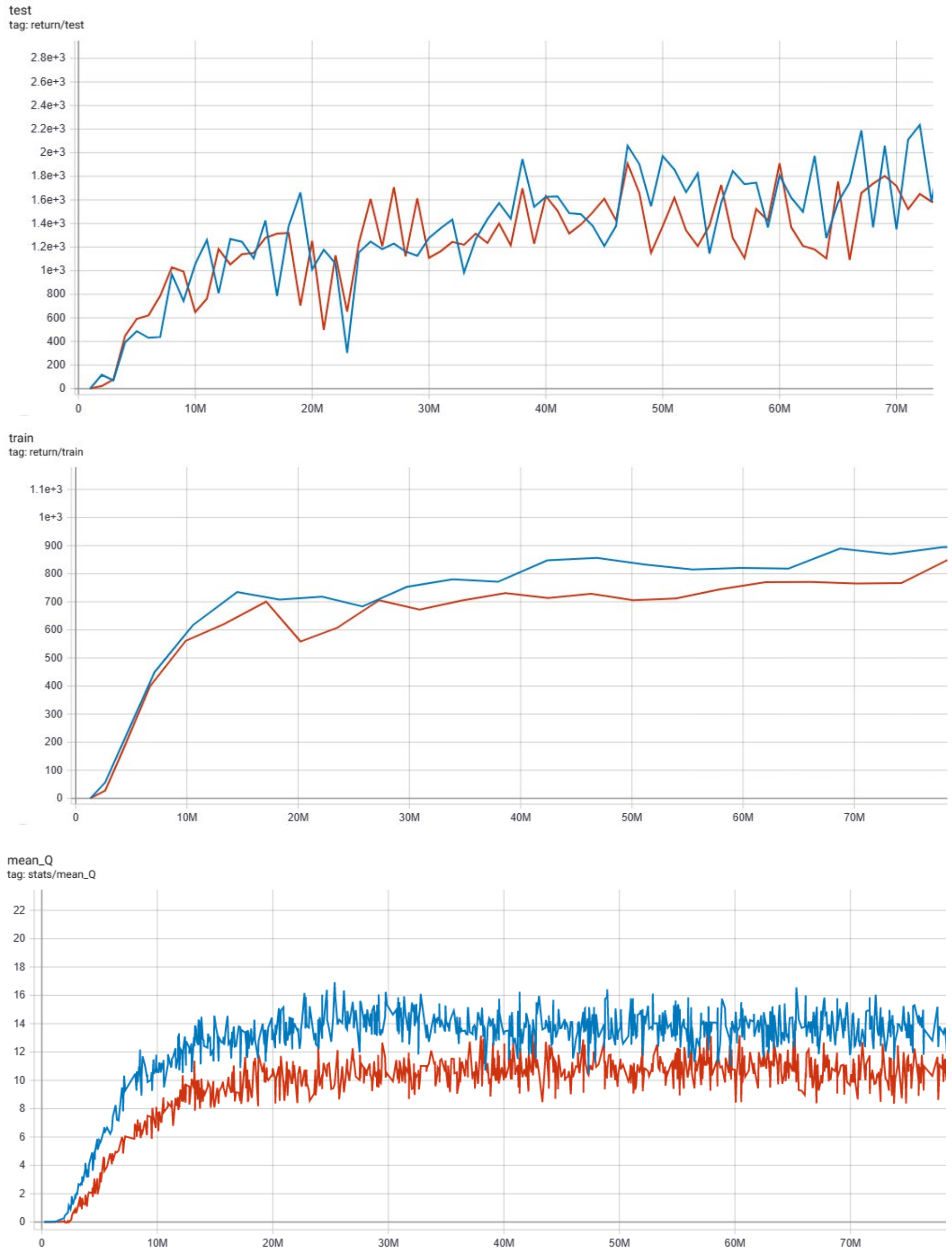


QR vs QR Double-Q

- Training:
  - QRDQN > QR Double-Q
- Testing:
  - QRDQN >= QR Double-Q
- Estimation of Q:
  - QRDQN > QR Double-Q
- Reduce over-estimation (?)

QRDQN

QRDQN Double-Q

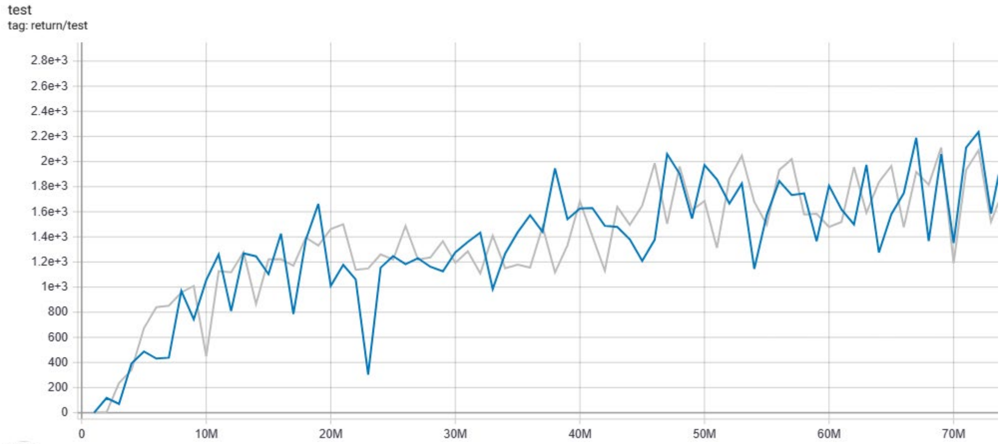


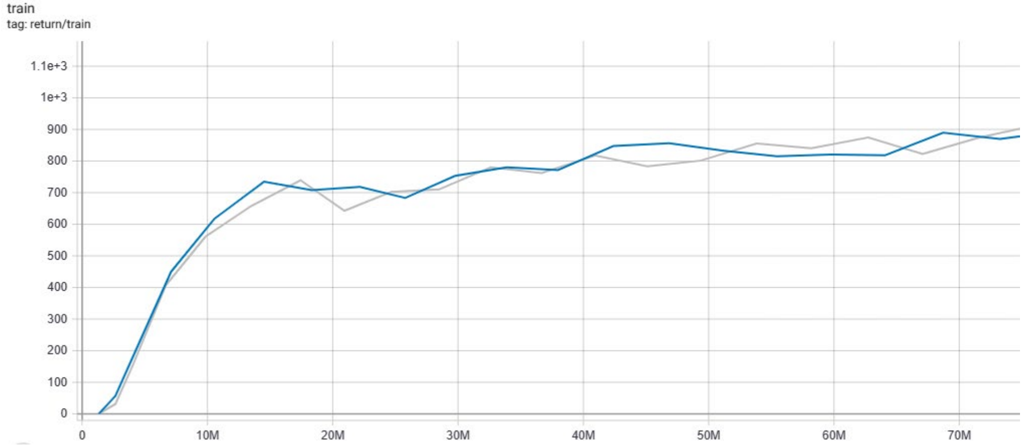
QR vs QR PER

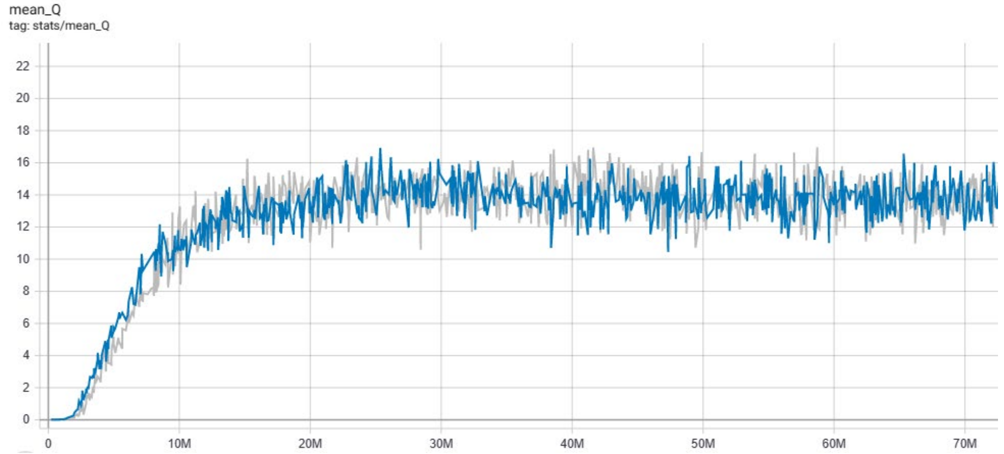
- Training:
  - QRDQN = QR PER
- Testing:
  - QRDQN = QR PER
- Estimation of Q:
  - QRDQN = QR PER
- PER is not so effective in dense reward environment.

QRDQN

QRDQN PER





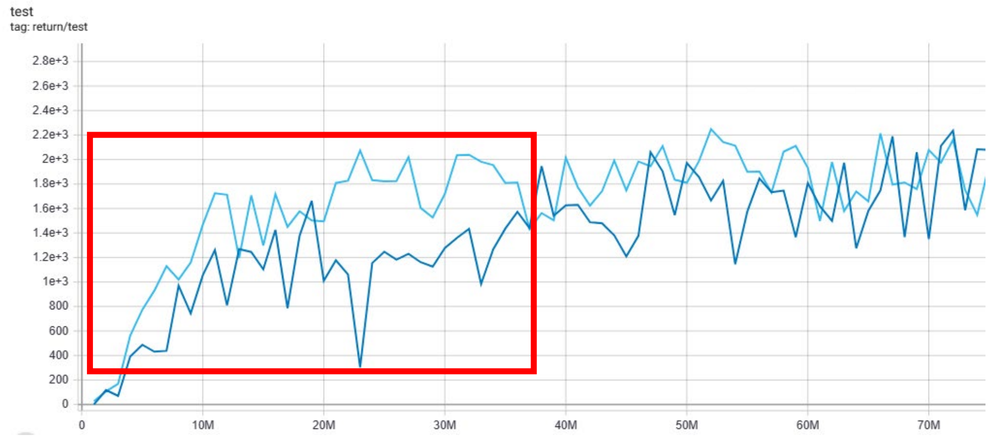


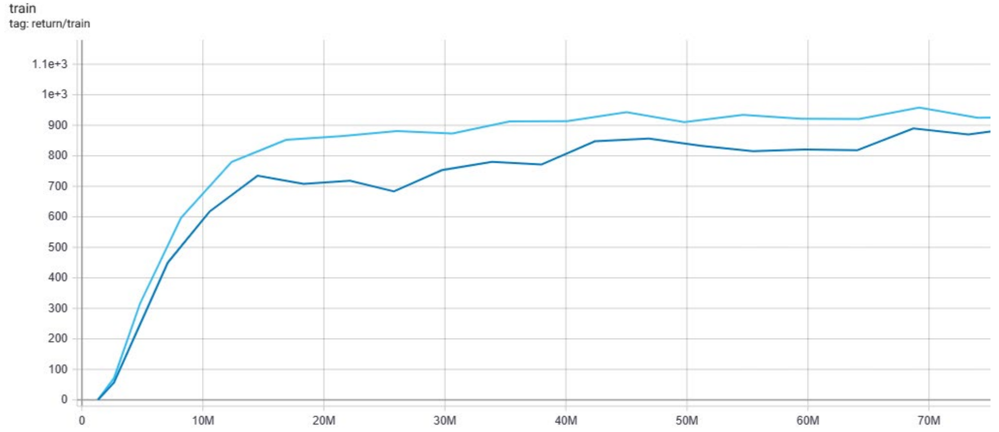
QR vs QR Dueling Network

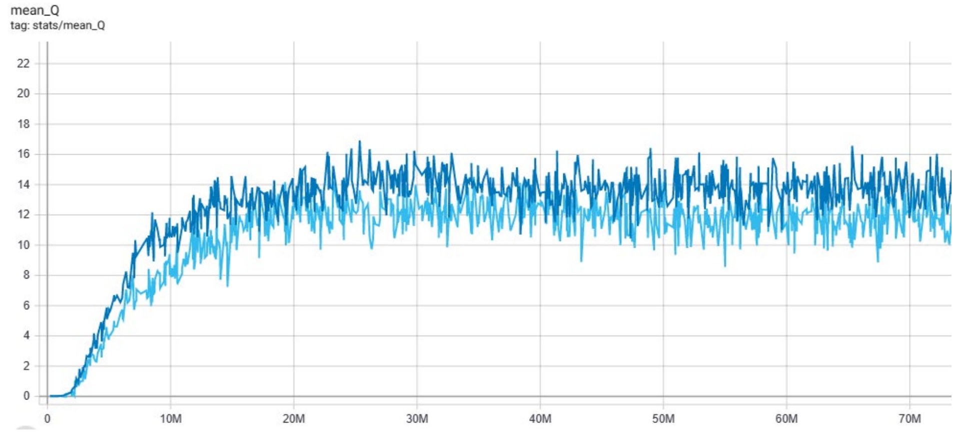
- Training:
  - QRDQN < QR Dueling
- Testing:
  - QRDQN < QR Dueling
- Estimation of Q:
  - QRDQN > QR Dueling
- Great performance against QR in early stage while testing.

☑ QRDQN

☑ QRDQN Dueling





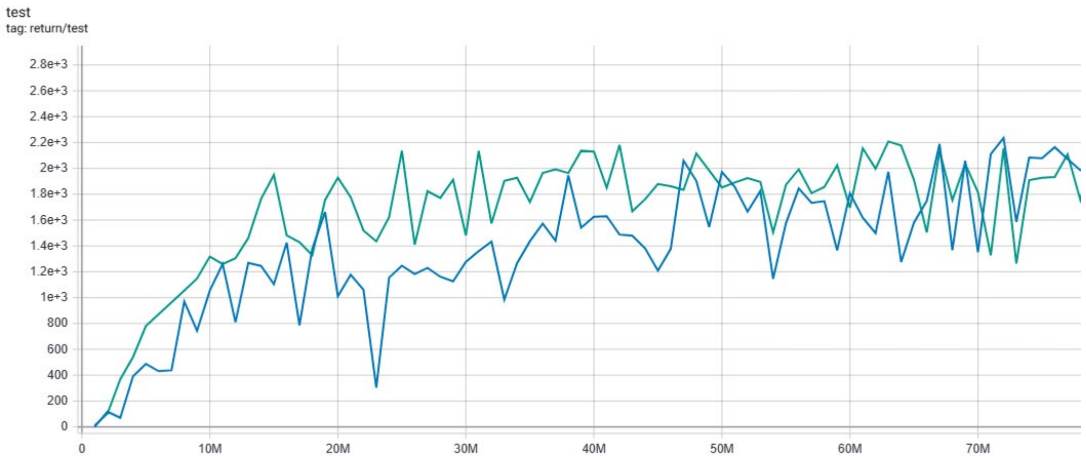


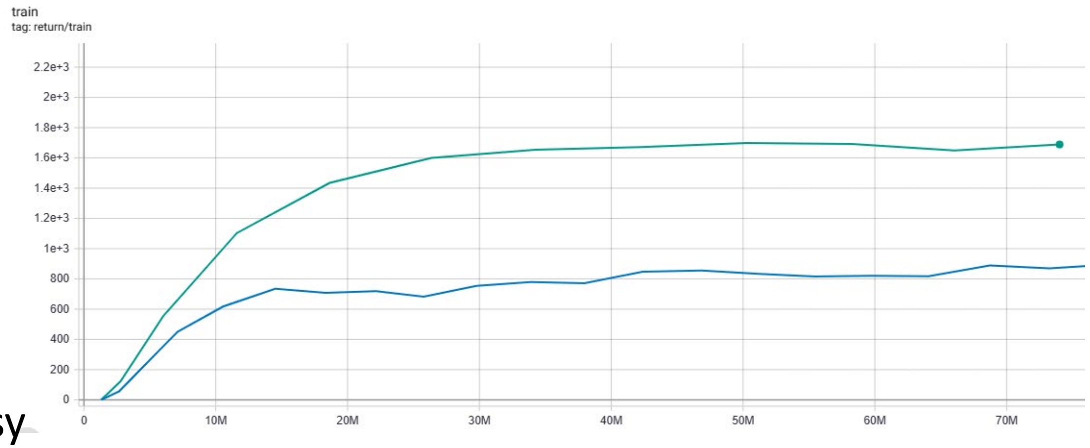
QR vs QR Noisy Network

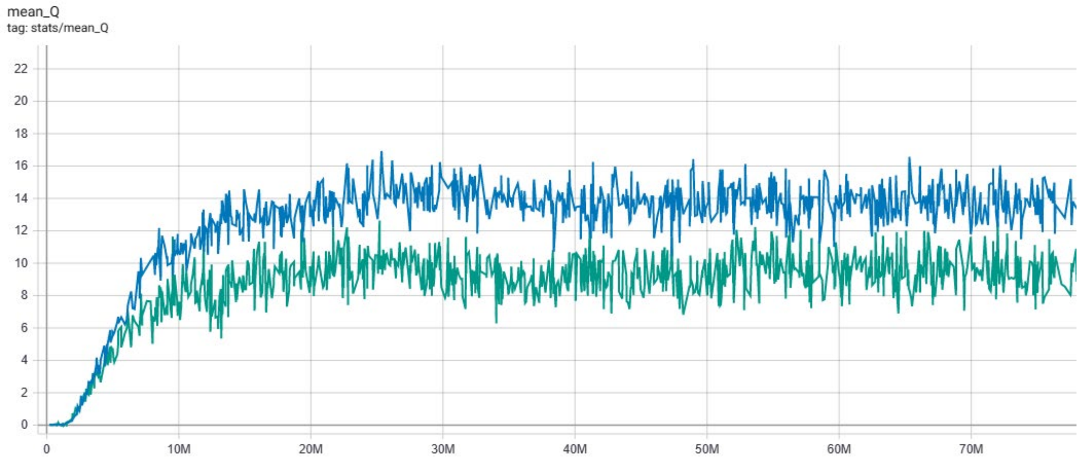
- Training:
    - QRDQN << QR Noisy
- Testing:
    - QRDQN < QR Noisy
- Estimation of Q:
    - QRDQN > QR Noisy
- Excellent performance on training return, but seems not translating into testing.
- Sample noise while testing ?

QRDQN

QRDQN Noisy





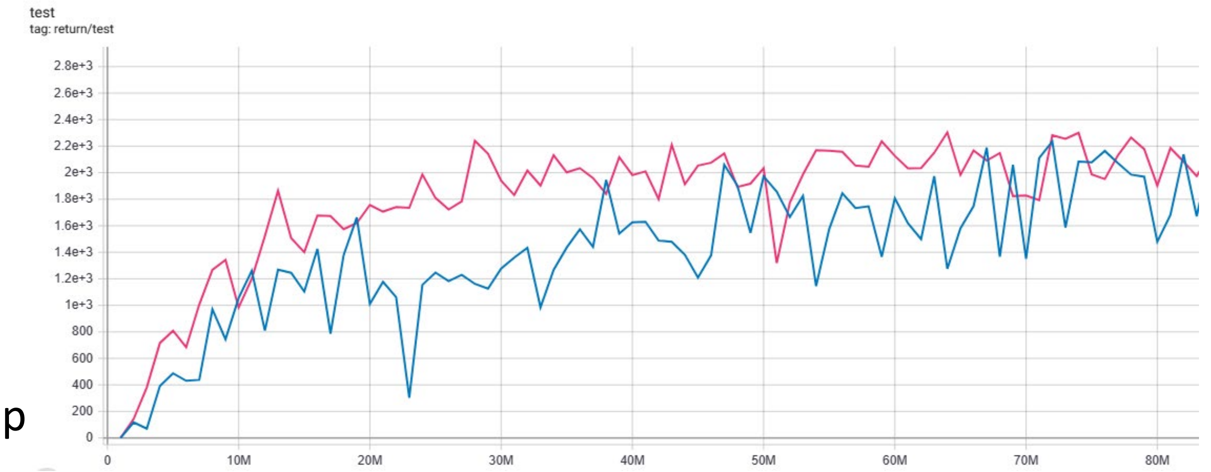


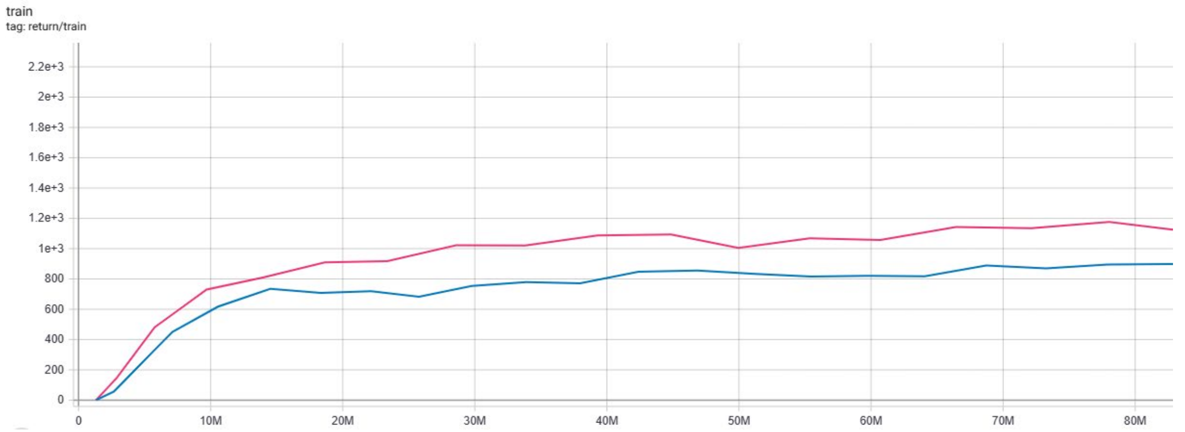
QR vs QR Multistep

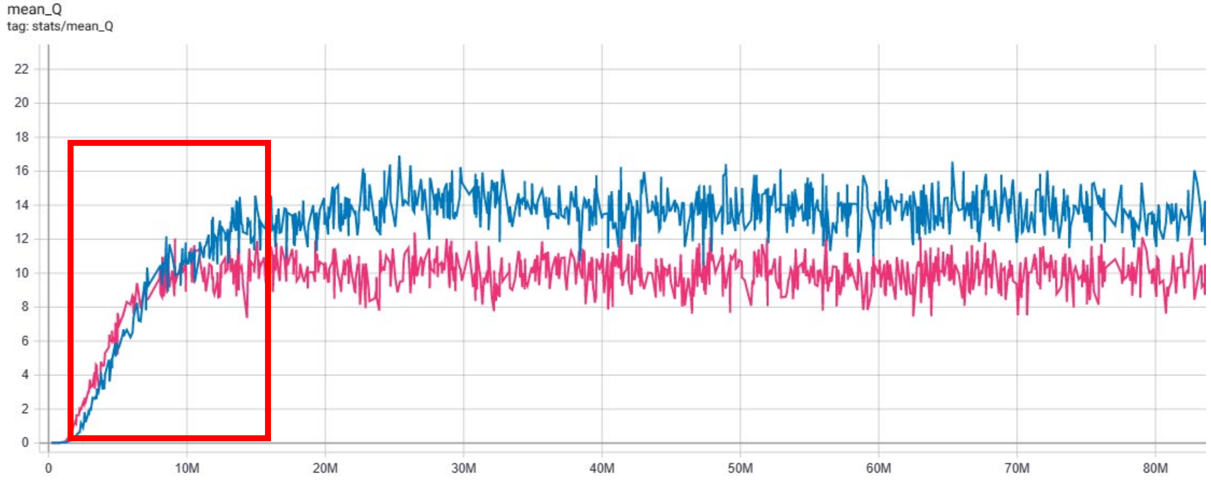
QRDQN

QRDQN Multistep

- Training:
  - QRDQN < QR Multistep
- Testing:
  - QRDQN < QR Multistep
- Estimation of Q:
  - QRDQN > QR Multistep
- High Q-value initially, but as training progress, Q-value tends to be lower.





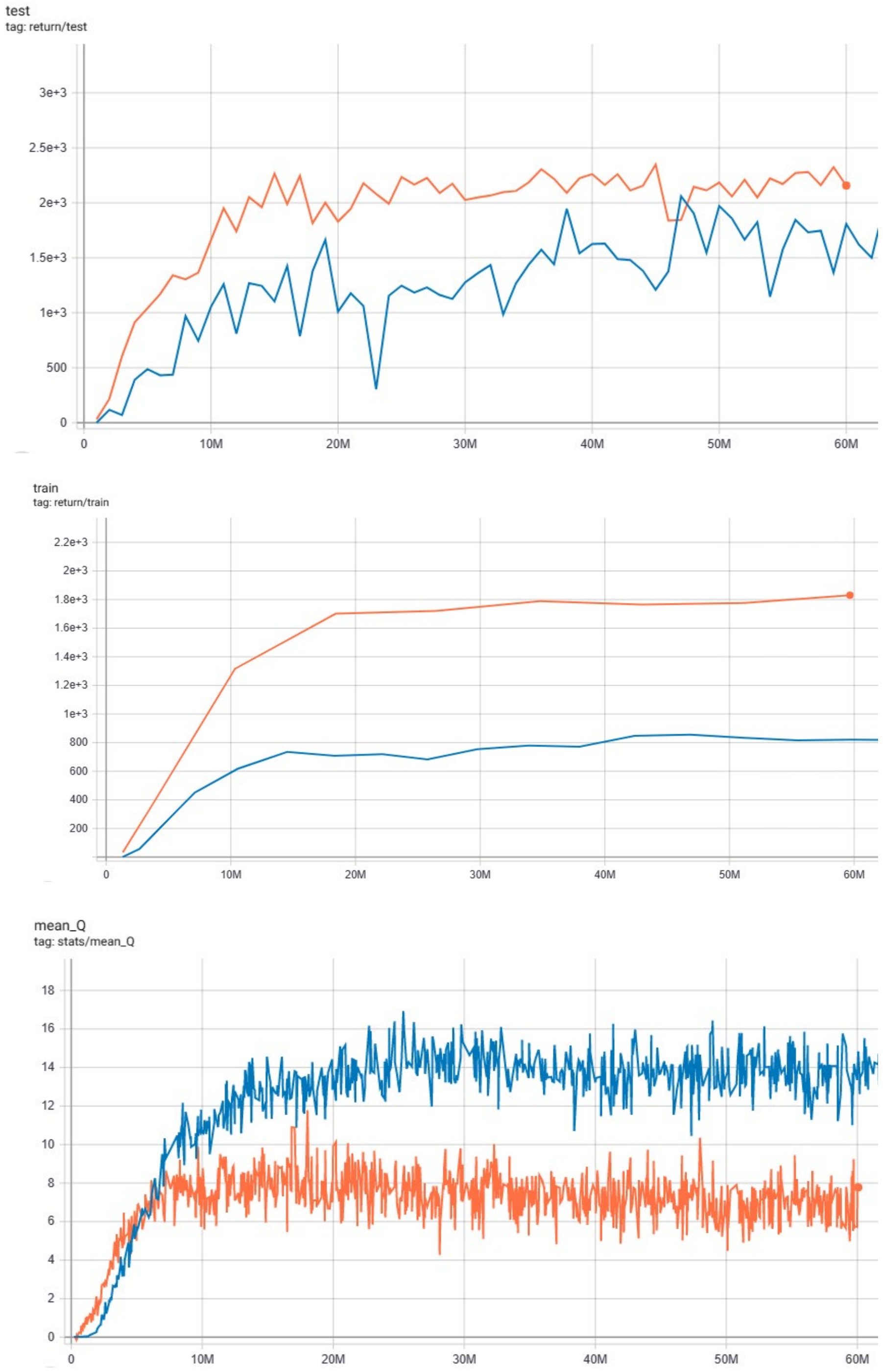


QR vs QR Rainbow

- Training:
  - QRDQN << QR Rainbow
- Testing:
  - QRDQN << QR Rainbow
- Estimation of Q:
  - QRDQN >> QR Rainbow
- Best performance.

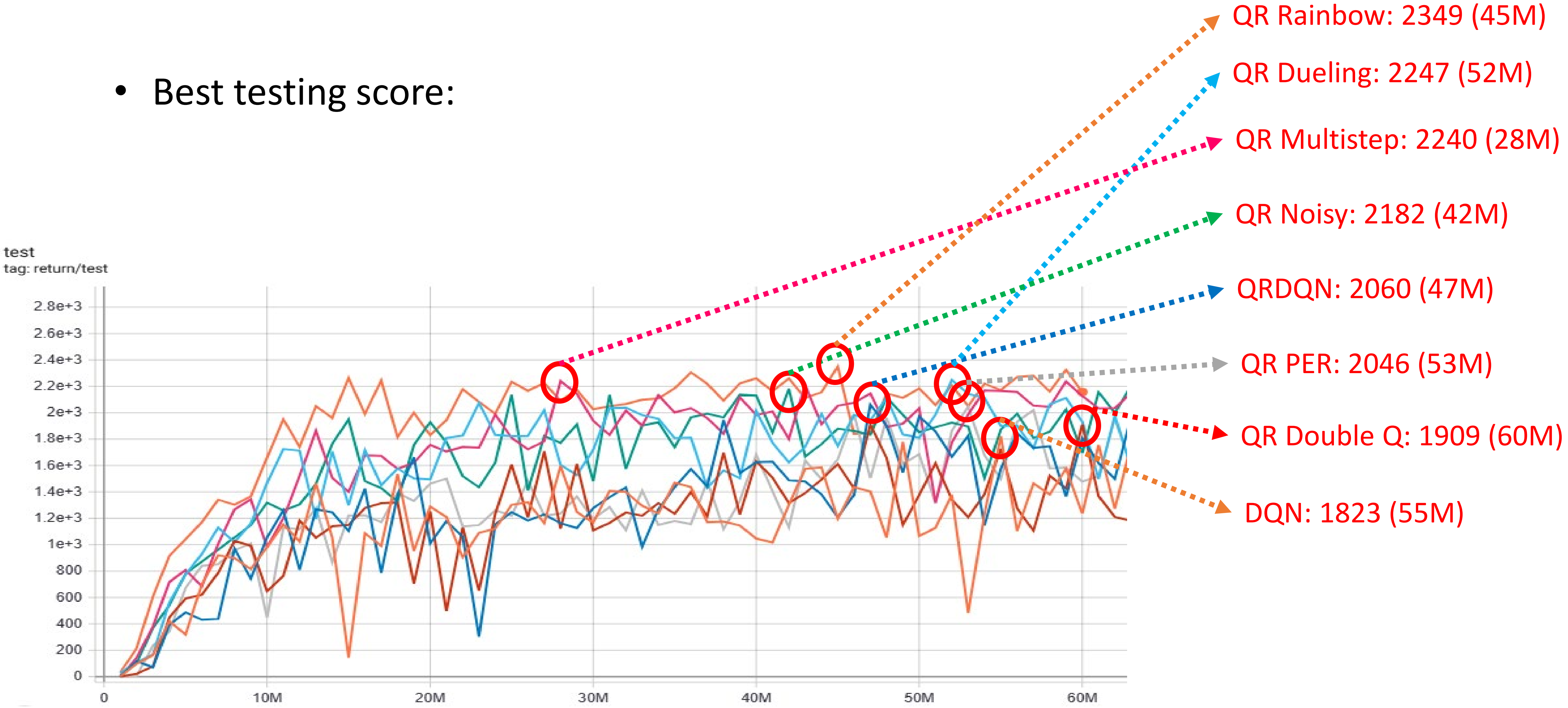
QRDQN

QRDQN Rainbow



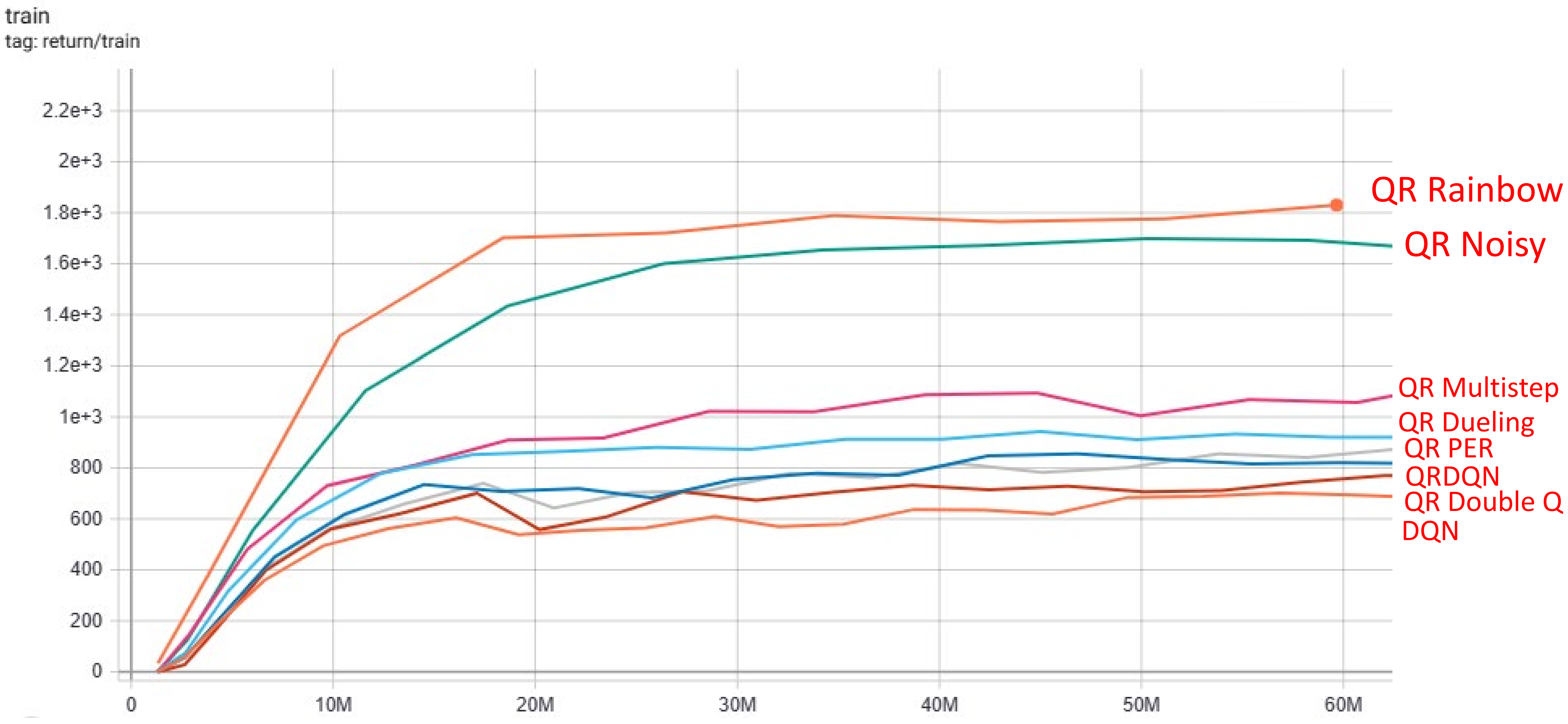
Overall Comparison (in 60M frames)

- Best testing score:



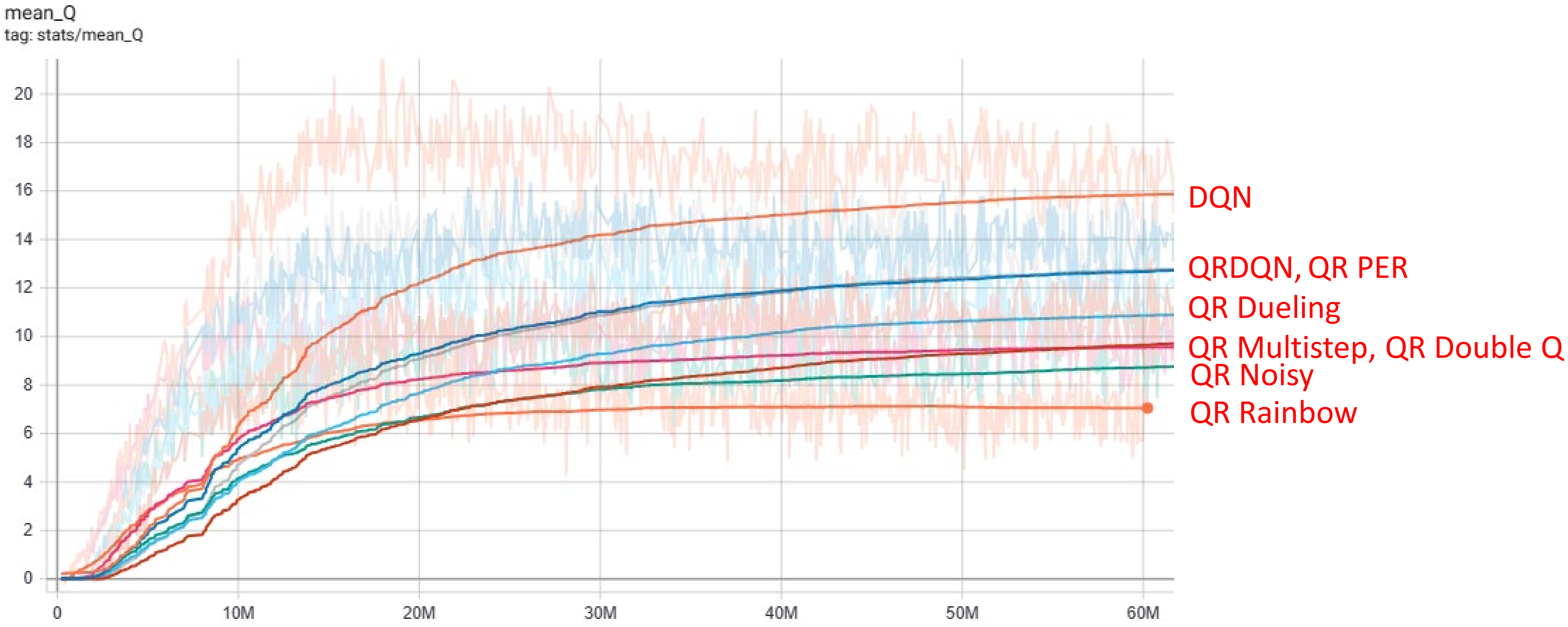
Overall Comparison (in 60M frames)

- Training score:



Overall Comparison (in 60M frames)

- Estimation of Q-value:



Overall Comparison (in 60M frames)

- Training speed (0 to 60M) (may not be accurate):
  - DQN (2d11h) — Fast
  - QR Multistep (2d13h18m)
  - QR Double-Q (2d13h24m)
  - QRDQN (2d14h)

  Takes a little bit longer of time

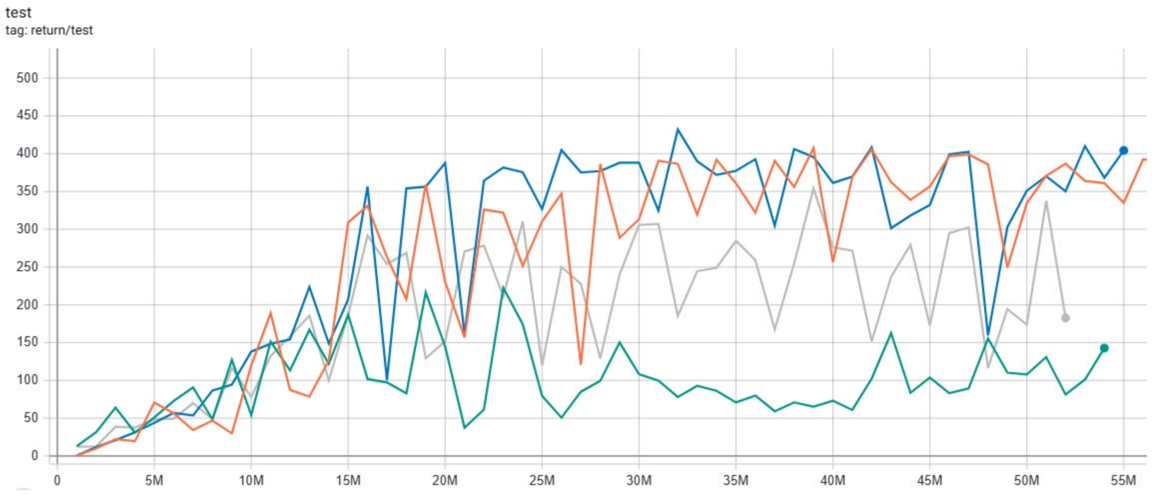
  - QR Dueling (2d16h)
  - QR Noisy (2d21h)
  - QR PER (2d22h)

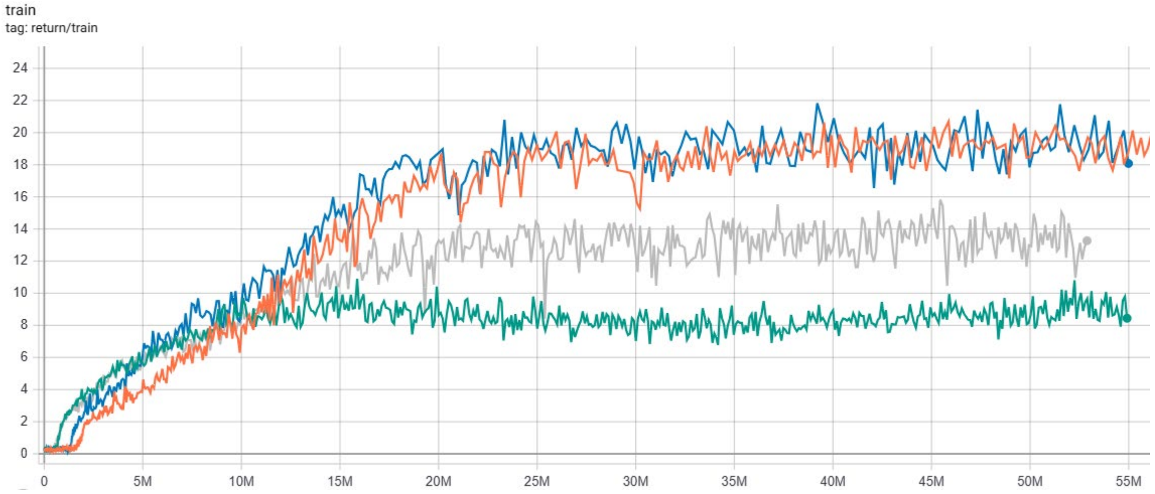
  A little bit slow

  - QR Rainbow (3d20h) — Very slow

Other Issue – Multistep Return

- Some environments may not be suitable for multistep return.
- May need some off-policy corrections. Like Retrace or ACER...
- Breakout:





QRDQN DQN QRDQN Multistep QR Rainbow

Other Issue – Noise while Testing

- Load the best model of QR Noisy.
- Run 1M frames and record the mean return:

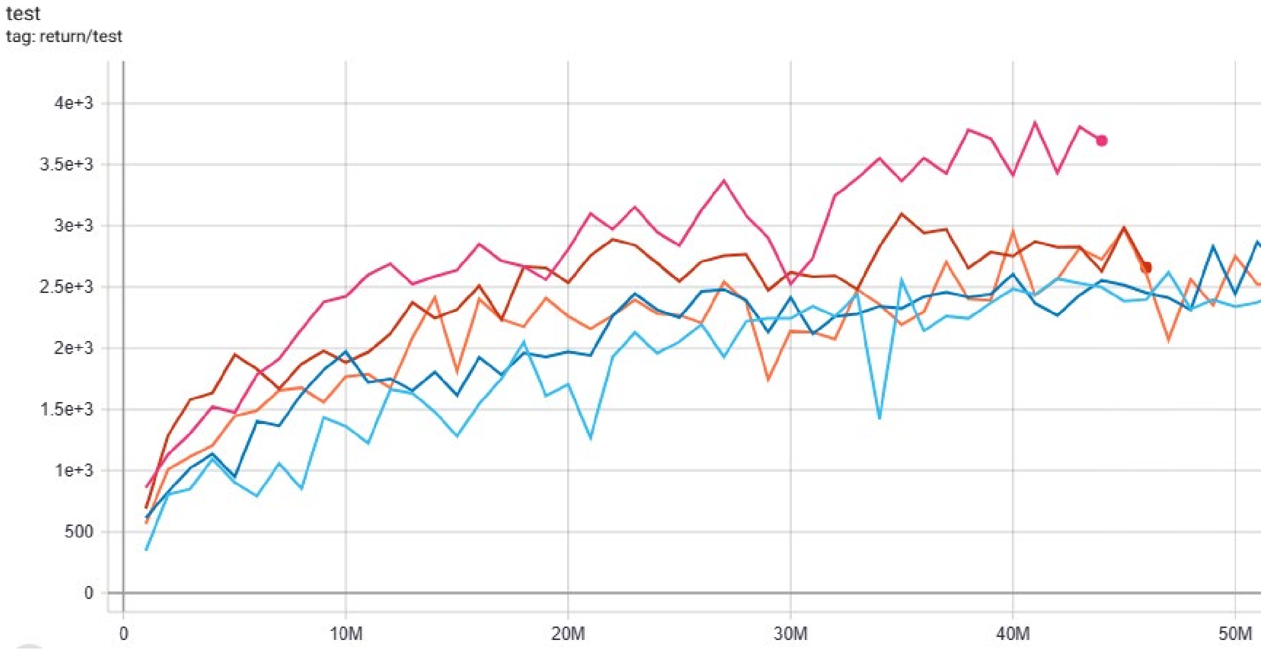
	With Noise	Without Noise
Enduro	2143.3	2114.6
MsPacman	3005.95	3180.75

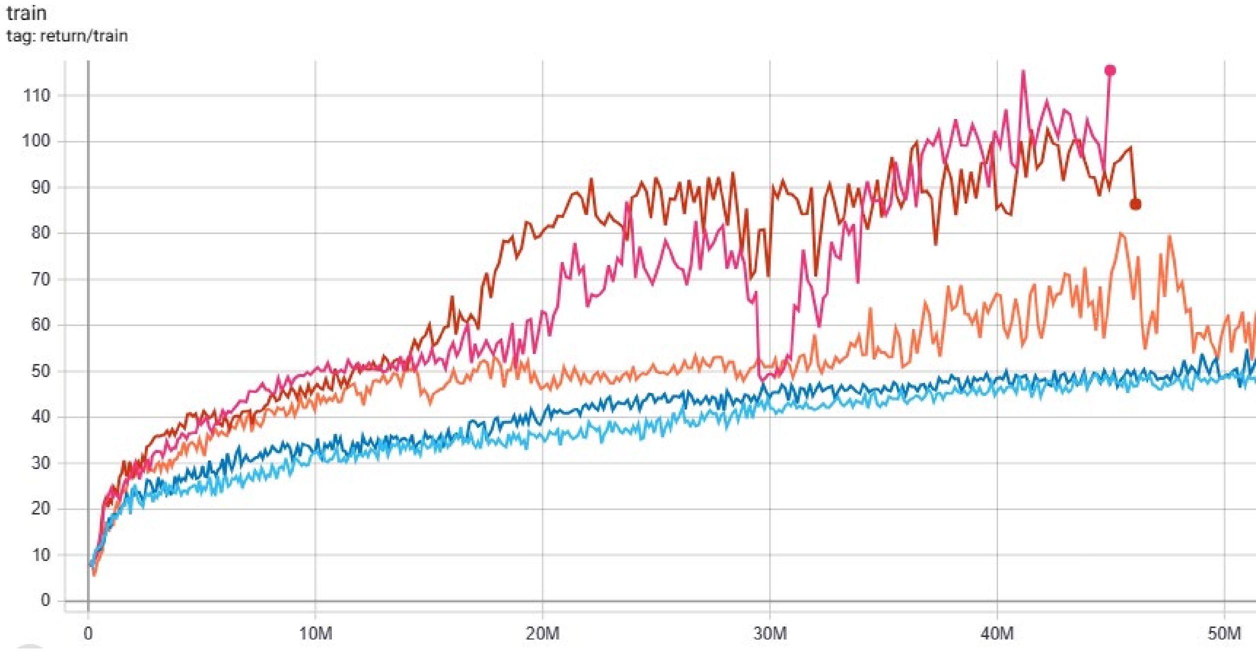
- Sample noise while testing doesn’t have much impact.

Other Research – IQN, FQF

- QR Rainbow
- IQN
- IQN Rainbow
- FQF
- FQF Rainbow

MsPacman:



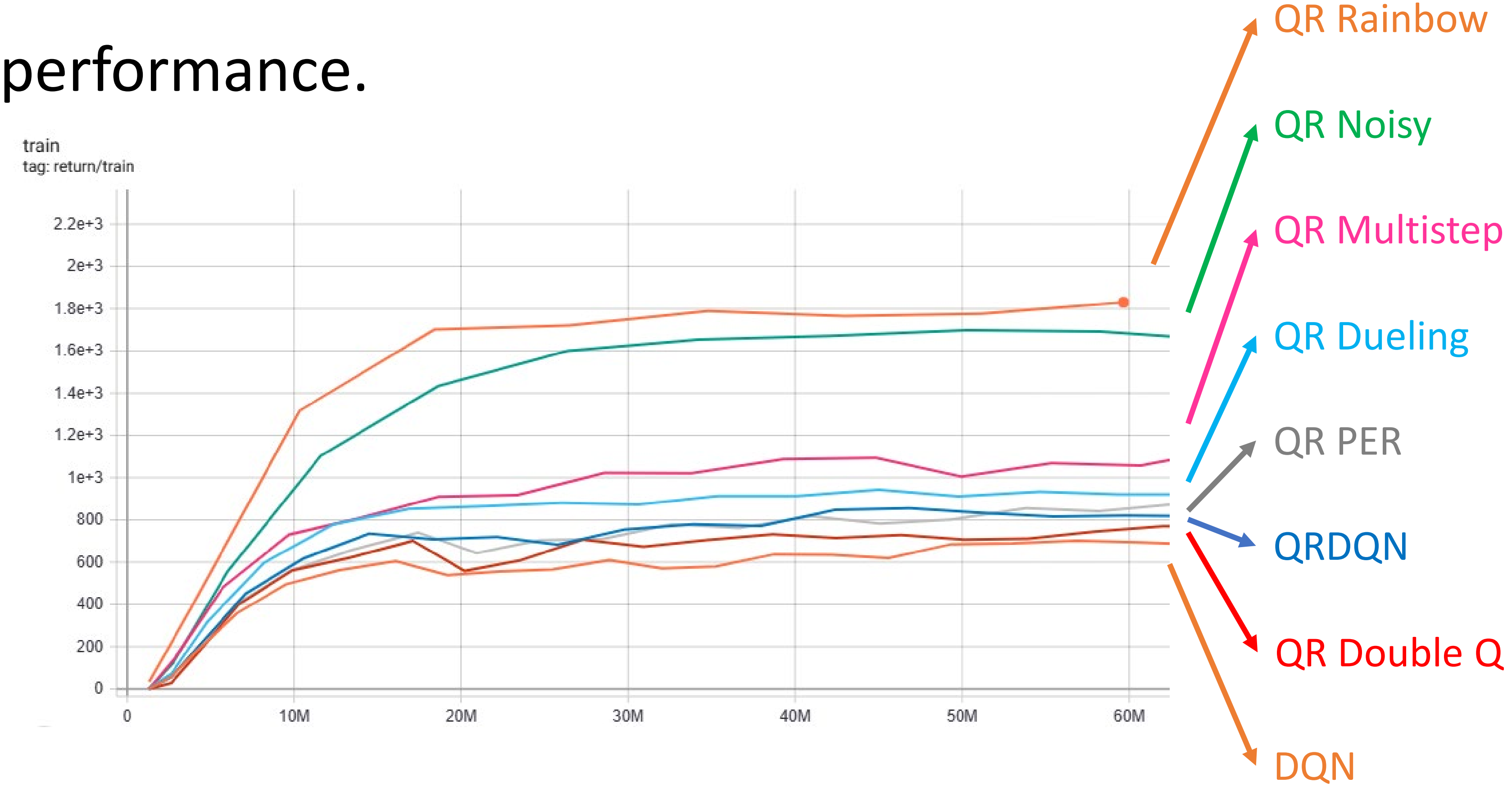


Conclusion

Conclusion

- Combining QRDQN with most of the non-distributional methods gets better performance.

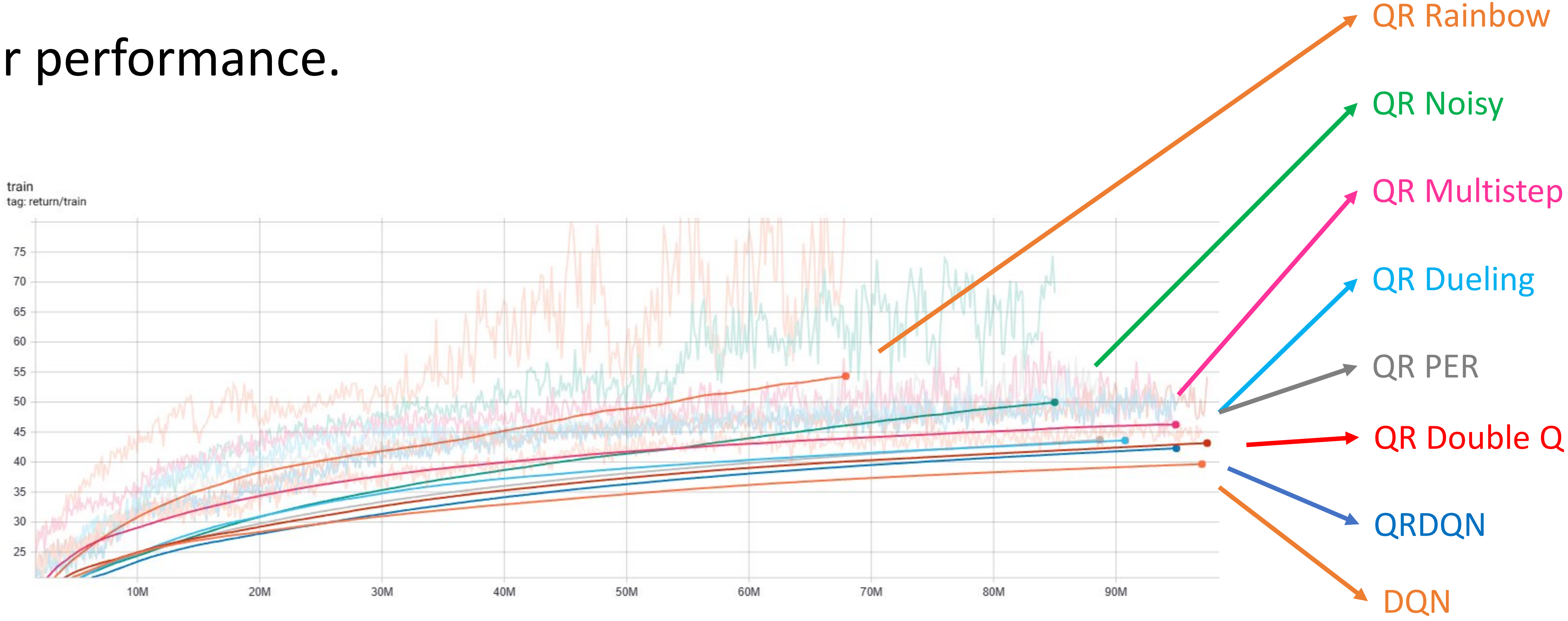
Enduro:



Conclusion

- Combining QRDQN with most of the non-distributional methods gets better performance.

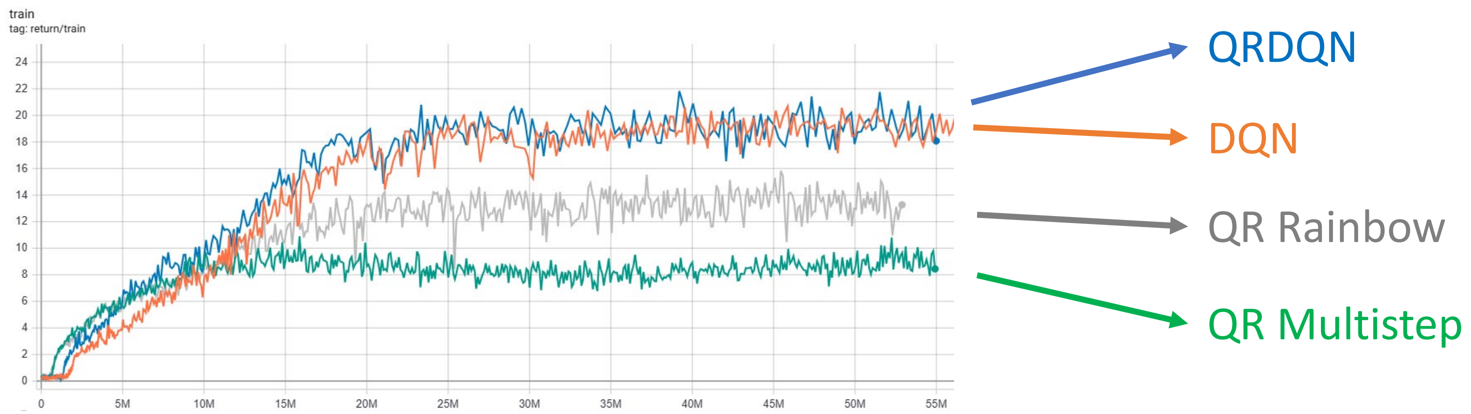
MsPacman:



Conclusion

- Some environments are not suitable for all methods, so blindly using Rainbow is risky.
- It is important to test and confirm the effectiveness of a method in improving performance before incorporating it.

Breakout:



Further Studies

- More games, more environments.
- Different kappa (0 or 1) in QRDQN.
- Different N in QRDQN.
- Compare Rainbow with QR Rainbow.
- Add intrinsic motivation (e.g. RND) in QRDQN.
- Risk Sensitive QRDQN.

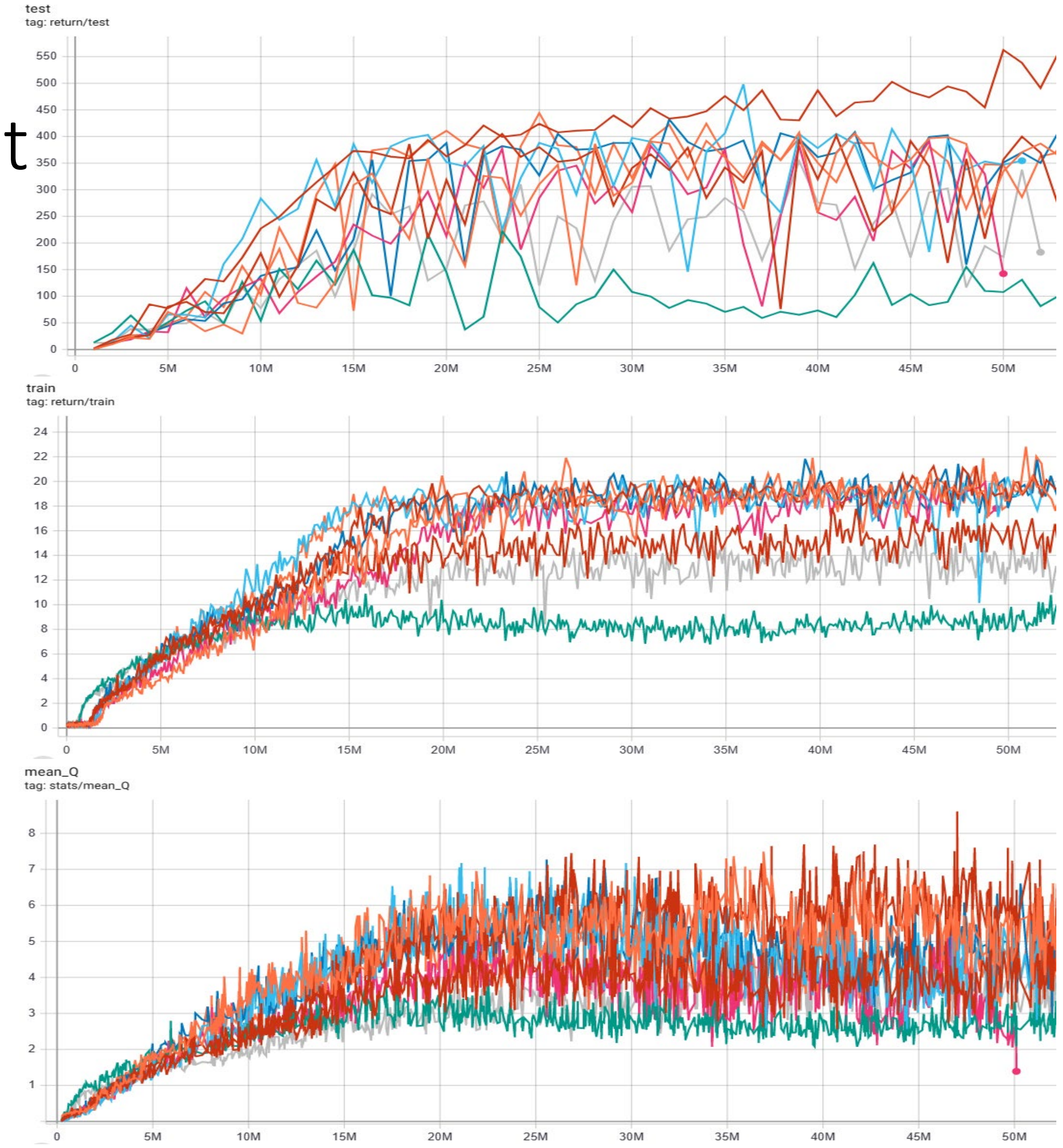
Appendix

Video

- QR Rainbow plays Enduro:
  - https://youtu.be/yWZy-niGd6U
- QR Rainbow plays MsPacman:
  - https://youtube.com/shorts/Tbbh0zdfwRE?feature=share
- QRDQN plays Breakout:
  - https://youtu.be/TA4eBbyK8Mw

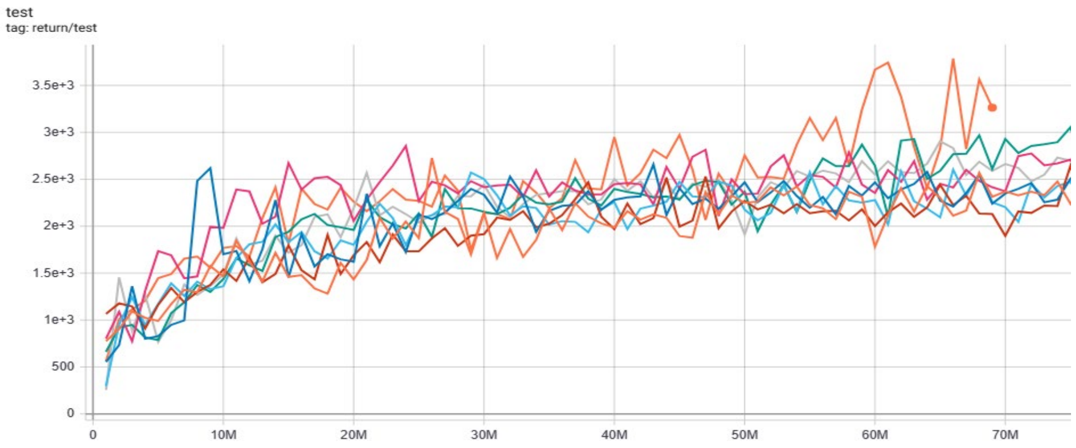
QR Rainbow in Breakout

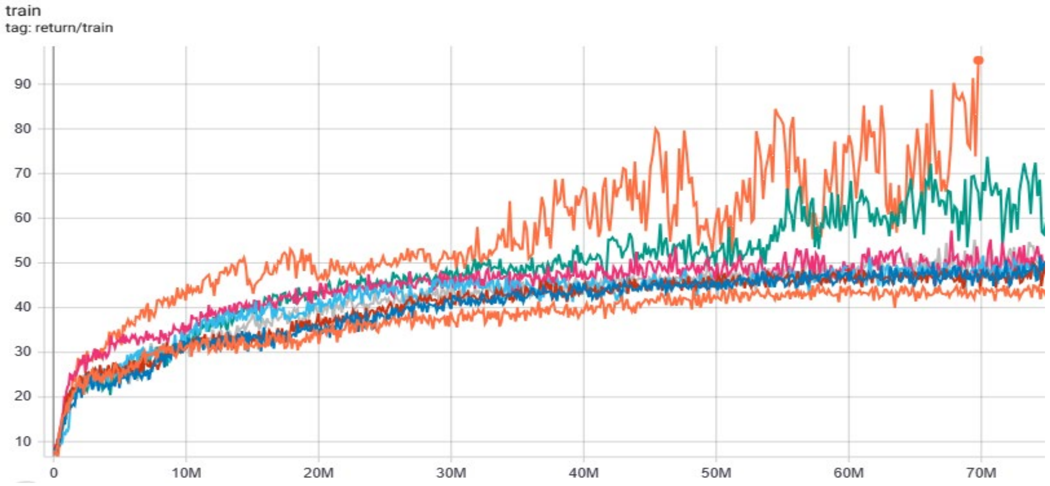
- QRDQN Prioritize Experience Replay
- QRDQN
- QRDQN Double-Q
- QRDQN Dueling Network
- QRDQN Noisy Network
- QRDQN Multistep Return
- QRDQN Rainbow
- DQN
- QRDQN PER + Double + Dueling

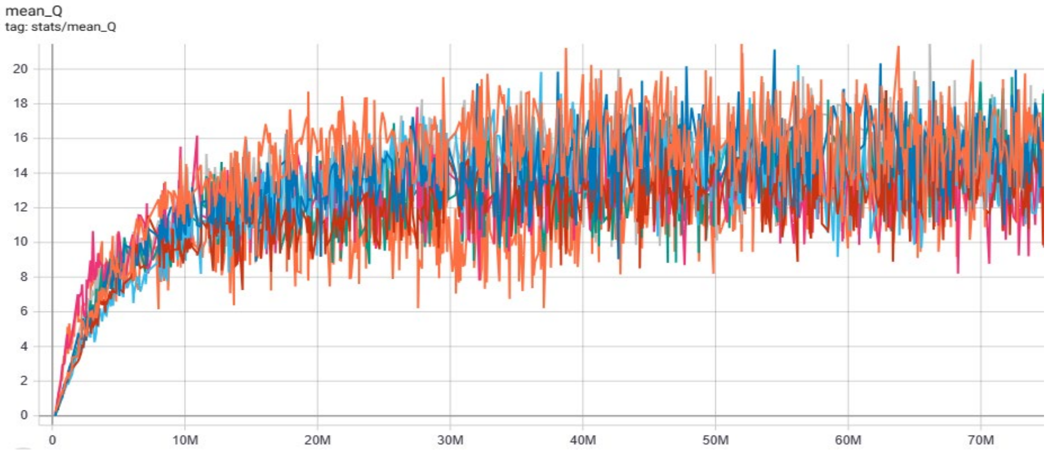


QR Rainbow in MsPacman

- DQN
- QRDQN
- QRDQN Double-Q
- QRDQN Dueling Network
- QRDQN Multistep Return
- QRDQN Noisy Network
- QRDQN Prioritize Experience Replay
- QRDQN Rainbow

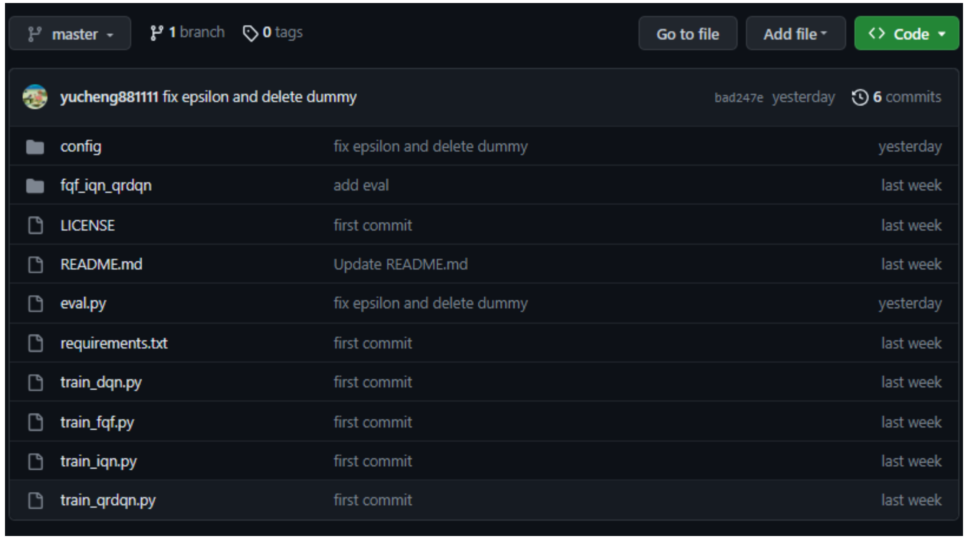


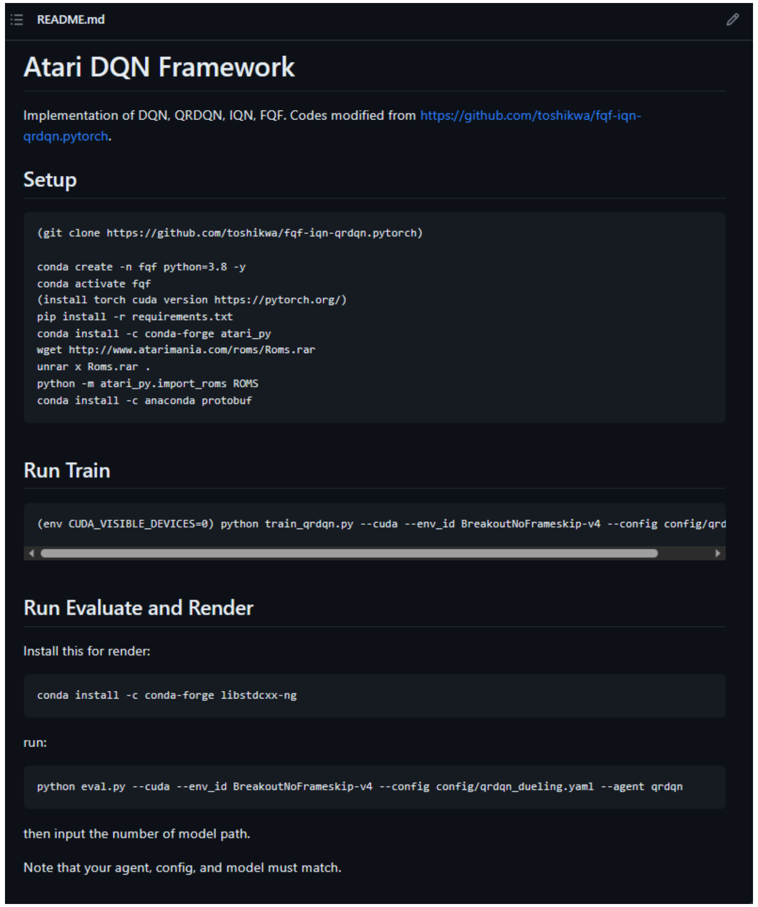




My GitHub Link

- https://github.com/yucheng881111/Atari_DQN_Framework





Reference

Reference

- Codes modified from:
  - https://github.com/toshikwa/fqf-iqn-qrdqn.pytorch
  - Distributional Reinforcement Learning with Quantile Regression | Papers With Code

Code

Edit

DLR-RM/stable-baselines3 ↳ Quickstart in Colab	★ 5,861	PyTorch
facebookresearch/Horizon	★ 3,400	PyTorch
facebookresearch/ReAgent	★ 3,400	PyTorch
opendilab/DI-engine	★ 2,605	PyTorch
Kchu/DeepRL_CK	★ 165	PyTorch
ku2482/fqf-iqn-qrdqn.pytorch	★ 135	PyTorch
ars-ashuha/quantile-regression-dqn-... ↳ Quickstart in Colab	★ 81	PyTorch
senya-ashukha/quantile-regression-d... ↳ Quickstart in Colab	★ 81	PyTorch

Reference

- Rainbow:
  - Hessel, Matteo, et al. "Rainbow: Combining improvements in deep reinforcement learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.
- QRDQN:
  - Dabney, Will, et al. "Distributional reinforcement learning with quantile regression." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.
- Multistep Return:
  - Hernandez-Garcia, J. Fernando, and Richard S. Sutton. "Understanding multi-step deep reinforcement learning: A systematic study of the DQN target." arXiv preprint arXiv:1901.07510 (2019).
- Prioritized Experience Replay:
  - Schaul, Tom, et al. "Prioritized experience replay." arXiv preprint arXiv:1511.05952 (2015).
- Double Q-learning:
  - Hasselt, Hado. "Double Q-learning." Advances in neural information processing systems 23 (2010).
  - Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 30. No. 1. 2016.

Reference

- Dueling Network:
  - Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." International conference on machine learning. PMLR, 2016.
- Noisy Network:
  - Fortunato, Meire, et al. "Noisy networks for exploration." arXiv preprint arXiv:1706.10295 (2017).
- IQN:
  - Dabney, Will, et al. "Implicit quantile networks for distributional reinforcement learning." International conference on machine learning. PMLR, 2018.
- FQF:
  - Yang, Derek, et al. "Fully parameterized quantile function for distributional reinforcement learning." Advances in neural information processing systems 32 (2019).
- Risk Sensitive QRDQN:
  - Lim, Shiau Hong, and Ilyas Malik. "Distributional Reinforcement Learning for Risk-Sensitive Policies." Advances in Neural Information Processing Systems. 2022.