Rainbow II, Distributional RL

Milan Straka

March 19, 2025

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Refresh

Rainbow

There have been many suggested improvements to the DQN architecture. In the end of 2017, the *Rainbow: Combining Improvements in Deep Reinforcement Learning* paper combines 6 of them into a single architecture they call **Rainbow**.

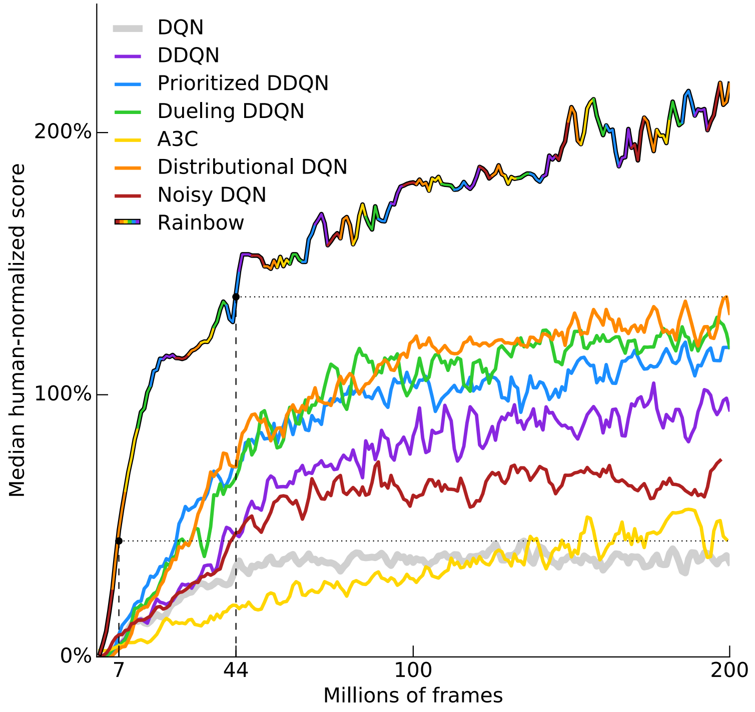


Figure 1 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Double Deep Q-Network

Double Deep Q-Network

Similarly to double Q-learning, instead of

$$r + \gamma \max_{a'} Q(s', a'; \bar{\boldsymbol\theta}) - Q(s, a; \boldsymbol\theta),$$

we minimize

$$r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \boldsymbol\theta); \bar{\boldsymbol\theta}) - Q(s, a; \boldsymbol\theta).$$

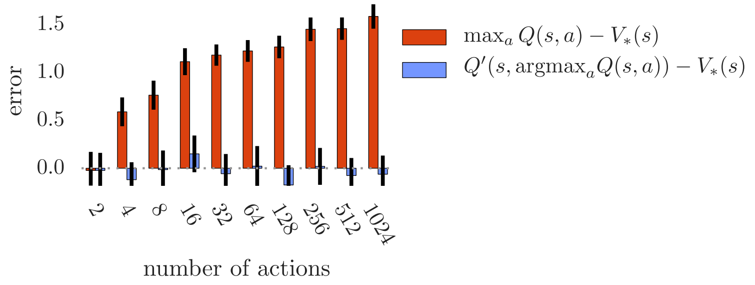


Figure 1: The orange bars show the bias in a single Q-learning update when the action values are $Q(s,a) = V_*(s) + \epsilon_a$ and the errors $\{\epsilon_a\}_{a=1}^m$ are independent standard normal random variables. The second set of action values Q', used for the blue bars, was generated identically and independently. All bars are the average of 100 repetitions.

Figure 1 of "Deep Reinforcement Learning with Double Q-learning" by Hado van Hasselt et al.

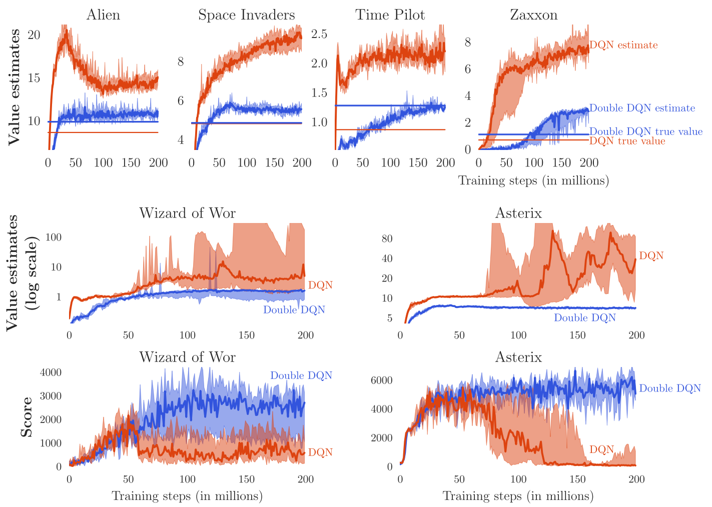


Figure 3 of "Deep Reinforcement Learning with Double Q-learning" by Hado van Hasselt et al.

Rainbow DQN Extensions

Double Q-learning

Performance on episodes taking at most 5 minutes and no-op starts on 49 games:

	DQN	Double DQN
Median	93.5%	114.7%
Mean	241.1%	330.3%

Table 1 of "Deep Reinforcement Learning with Double Q-learning" by Hado van Hasselt et al.

Performance on episodes taking at most 30 minutes and using 100 human starts on each of the 49 games:

	DQN	Double DQN	Double DQN (tuned)
Median	47.5%	88.4%	116.7%
Mean	122.0%	273.1%	475.2%

Table 2 of "Deep Reinforcement Learning with Double Q-learning" by Hado van Hasselt et al.

The Double DQN follows the training protocol of DQN; the tuned version increases the target network update from 10k to 30k steps, decreases exploration during training from $\varepsilon = 0.1$ to $\varepsilon = 0.01$, and uses a shared bias for all action values in the output layer of the network.

Prioritized Replay

Rainbow DQN Extensions

Prioritized Replay

Instead of sampling the transitions uniformly from the replay buffer, we instead prefer those with a large TD error. Therefore, we sample transitions according to their probability

$$p_t \propto \Big|r + \gamma \max_{a'} Q(s', a'; \bar{\boldsymbol\theta}) - Q(s, a; \boldsymbol\theta)\Big|^\omega,$$

where ω controls the shape of the distribution (which is uniform for $\omega = 0$ and corresponds to TD error for $\omega = 1$).

New transitions are inserted into the replay buffer with maximum probability to support exploration of all encountered transitions.

When combined with DDQN, the probabilities are naturally computed as

$$p_t \propto \Big|r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \boldsymbol\theta); \bar{\boldsymbol\theta}) - Q(s, a; \boldsymbol\theta)\Big|^\omega,$$

Prioritized Replay

Because we now sample transitions according to p_t instead of uniformly, on-policy distribution and sampling distribution differ. To compensate, we utilize importance sampling with ratio

$$\rho_t = \left(\frac{1/N}{p_t}\right)^\beta.$$

Because the importance sampling ratios ρ can be quite large, the authors normalize them, as they say "for stability reasons", in every batch:

$$\rho_t / \max_{t' \in batch} \rho_{t'}.$$

Therefore, the largest normalized importance sampling ratio in every batch is 1. The fact that normalization should happen in every batch is not explicitly stated in the paper, and implementations normalizing over the whole replay buffer also exist; but the DeepMind reference implementation does normalize batch-wise.

Prioritized Replay

Algorithm 1 Double DQN with proportional prioritization

```
 1: Input: minibatch $k$, step-size $\eta$, replay period $K$ and size $N$, exponents $\alpha$ and $\beta$, budget $T$.
 2: Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
 3: Observe $S_0$ and choose $A_0 \sim \pi_\theta(S_0)$
 4: for $t = 1$ to $T$ do
 5:     Observe $S_t, R_t, \gamma_t$
 6:     Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
 7:     if $t \equiv 0 \mod K$ then
 8:         for $j = 1$ to $k$ do
 9:             Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
10:             Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
11:             Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
12:             Update transition priority $p_j \leftarrow |\delta_j|$
13:             Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
14:         end for
15:         Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
16:         From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
17:     end if
18:     Choose action $A_t \sim \pi_\theta(S_t)$
19: end for
```

Algorithm 1 of "Prioritized Experience Replay" by Tom Schaul et al.

Dueling Networks

Dueling Networks

Instead of computing directly $Q(s, a; \boldsymbol\theta)$, we compose it from the following quantities:

- average return in a given state s, $V(s; \boldsymbol\theta) = \frac{1}{|\mathcal{A}|} \sum_a Q(s, a; \boldsymbol\theta)$,
- advantage function computing an **advantage** $Q(s, a; \boldsymbol\theta) - V(s; \theta)$ of action a in state s.

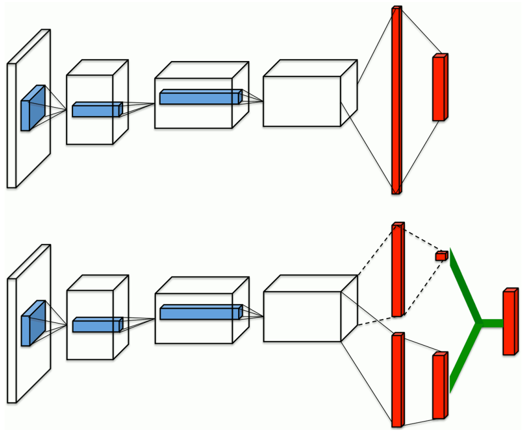
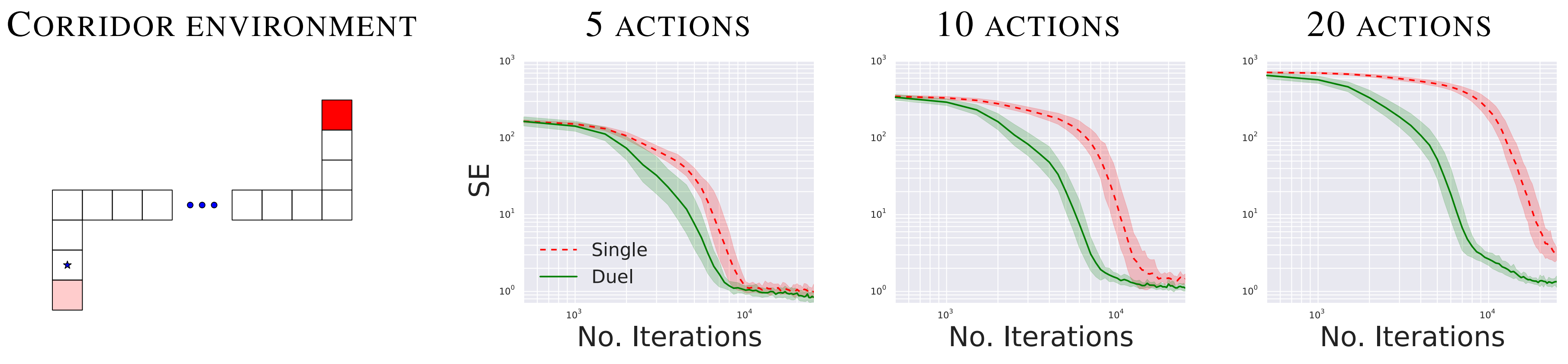


Figure 1 of "Dueling Network Architectures for Deep Reinforcement Learning" by Ziyu Wang et al.

$$Q(s, a) \stackrel{\text{def}}{=} V\big(f(s; \zeta); \eta\big) + A\big(f(s; \zeta), a; \psi\big) - \frac{\sum_{a' \in \mathcal{A}} A(f(s; \zeta), a'; \psi)}{|\mathcal{A}|}$$

Rainbow DQN Extensions

Dueling Networks



(a) **(b)** **(c)** **(d)**

Figure 3. **(a)** The corridor environment. The star marks the starting state. The redness of a state signifies the reward the agent receives upon arrival. The game terminates upon reaching either reward state. The agent's actions are going up, down, left, right and no action. Plots **(b), (c)** and **(d)** shows squared error for policy evaluation with 5, 10, and 20 actions on a log-log scale. The dueling network (Duel) consistently outperforms a conventional single-stream network (Single), with the performance gap increasing with the number of actions.

Figure 3 of "Dueling Network Architectures for Deep Reinforcement Learning" by Ziyu Wang et al.

Evaluation is performed using ε-greedy exploration with $\varepsilon = 0.001$; in the experiment, the horizontal corridor has a length of 50 steps, while the vertical sections have both 10 steps.

Dueling Networks

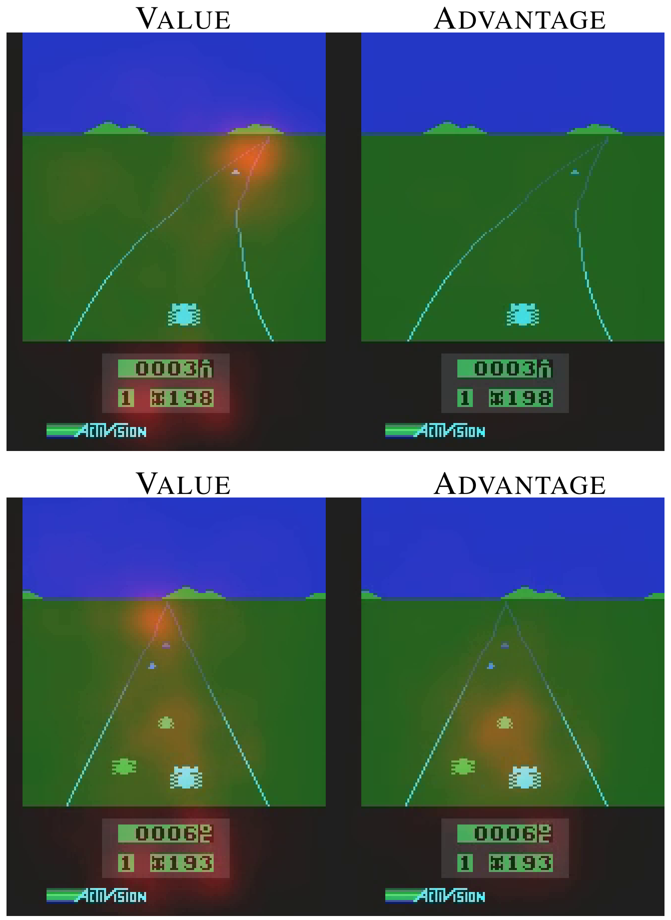
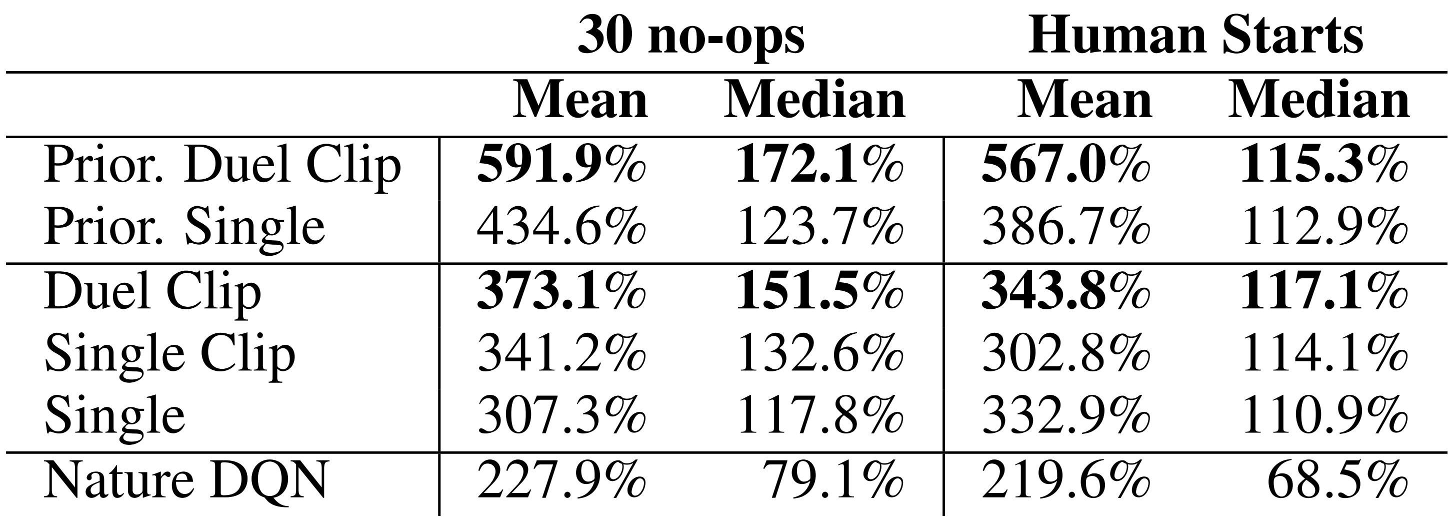


Figure 2 of "Dueling Network Architectures for Deep Reinforcement Learning" by Ziyu Wang et al.

Dueling Networks

Results on all 57 games (retraining the original DQN on the 8 missing games). `Single` refers to DDQN with a direct computation of $Q(s, a; \boldsymbol\theta)$, `Single Clip` corresponds to additional gradient clipping to norm at most 10 and larger first hidden layer (so that duelling and single have roughly the same number of parameters).



	30 no-ops		**Human Starts**	
	Mean	**Median**	**Mean**	**Median**
Prior. Duel Clip	**591.9%**	**172.1%**	**567.0%**	**115.3%**
Prior. Single	434.6%	123.7%	386.7%	112.9%
Duel Clip	**373.1%**	**151.5%**	**343.8%**	**117.1%**
Single Clip	341.2%	132.6%	302.8%	114.1%
Single	307.3%	117.8%	332.9%	110.9%
Nature DQN	227.9%	79.1%	219.6%	68.5%

Table 1 of "Dueling Network Architectures for Deep Reinforcement Learning" by Ziyu Wang et al.

Multi-step DQN

Multi-step DQN

Instead of Q-learning, we use n-step variant of Q-learning, which estimates return as

$$\sum_{i=1}^{n} \gamma^{i-1} R_i + \gamma^n \max_{a'} Q(s', a'; \bar{\boldsymbol\theta}).$$

This changes the off-policy algorithm to on-policy (because the "inner" actions are sampled from the behaviour distribution, but should follow the target distribution); however, it is not discussed in any way by the authors.

Noisy Nets

Noisy Nets

Noisy Nets are neural networks whose weights and biases are perturbed by a parametric function of a noise.

The parameters $\boldsymbol\theta$ of a regular neural network are in Noisy nets represented as

$$\boldsymbol\theta \approx \boldsymbol\mu + \boldsymbol\sigma \odot \boldsymbol\varepsilon,$$

where $\boldsymbol\varepsilon$ is zero-mean noise with fixed statistics. We therefore learn the parameters $(\boldsymbol\mu, \boldsymbol\sigma)$.

A fully connected layer $\boldsymbol y = \boldsymbol w \boldsymbol x + \boldsymbol b$ with parameters $(\boldsymbol w, \boldsymbol b)$ is represented in the following way in Noisy nets:

$$\boldsymbol y = (\boldsymbol\mu_w + \boldsymbol\sigma_w \odot \boldsymbol\varepsilon_w)\boldsymbol x + (\boldsymbol\mu_b + \boldsymbol\sigma_b \odot \boldsymbol\varepsilon_b).$$

Each $\sigma_{i,j}$ is initialized to $\frac{\sigma_0}{\sqrt{n}}$, where n is the number of input neurons of the layer in question, and σ_0 is a hyperparameter; commonly 0.5.

Rainbow DQN Extensions

Noisy Nets

The noise ε can be for example independent Gaussian noise. However, for performance reasons, factorized Gaussian noise is used to generate a matrix of noise. If $\varepsilon_{i,j}$ is noise corresponding to a layer with n inputs and m outputs, we generate independent noise ε_i for input neurons, independent noise ε_j for output neurons, and set

$$\varepsilon_{i,j} = f(\varepsilon_i) f(\varepsilon_j) \quad \text{for} \quad f(x) = \operatorname{sign}(x)\sqrt{|x|}.$$

The authors generate noise samples for every batch, sharing the noise for all batch instances (consequently, during loss computation, online and target network use independent noise).

Deep Q Networks

When training a DQN, ε-greedy is no longer used (all policies are greedy), and all fully connected layers are parametrized as noisy nets in both the current and target network (i.e., networks produce samples from the distribution of returns, and greedy actions still explore).

Noisy Nets

	Baseline		NoisyNet		Improvement
	Mean	Median	Mean	Median	(On median)
DQN	319	83	**379**	**123**	48%
Dueling	524	132	**633**	**172**	30%
A3C	293	80	**347**	**94**	18%

Table 1 of "Noisy Networks for Exploration" by Meire Fortunato et al.

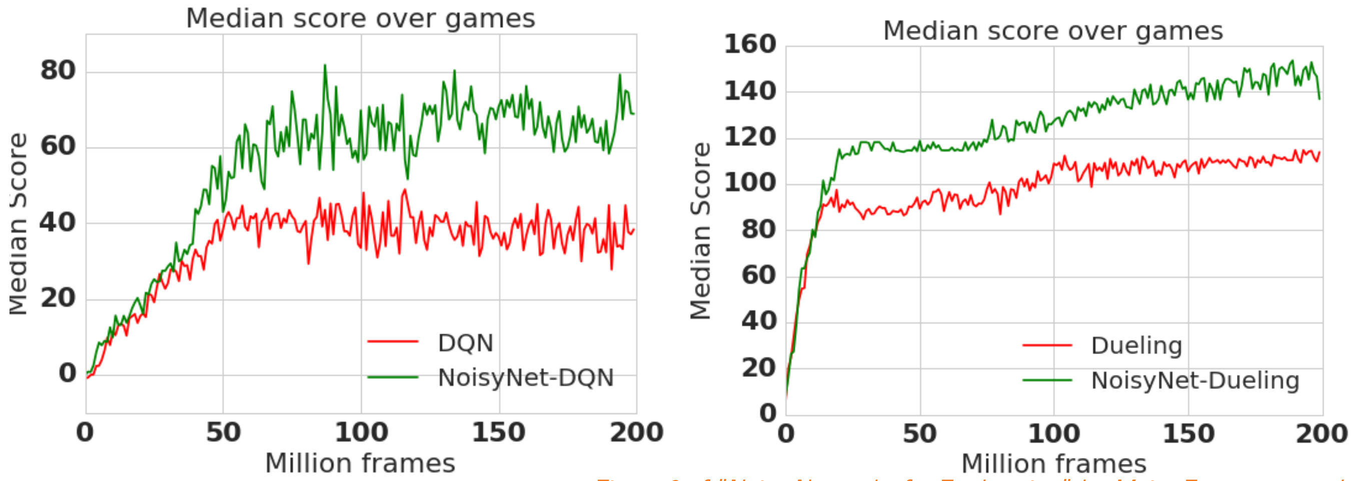


Figure 2 of "Noisy Networks for Exploration" by Meire Fortunato et al.

Noisy Nets

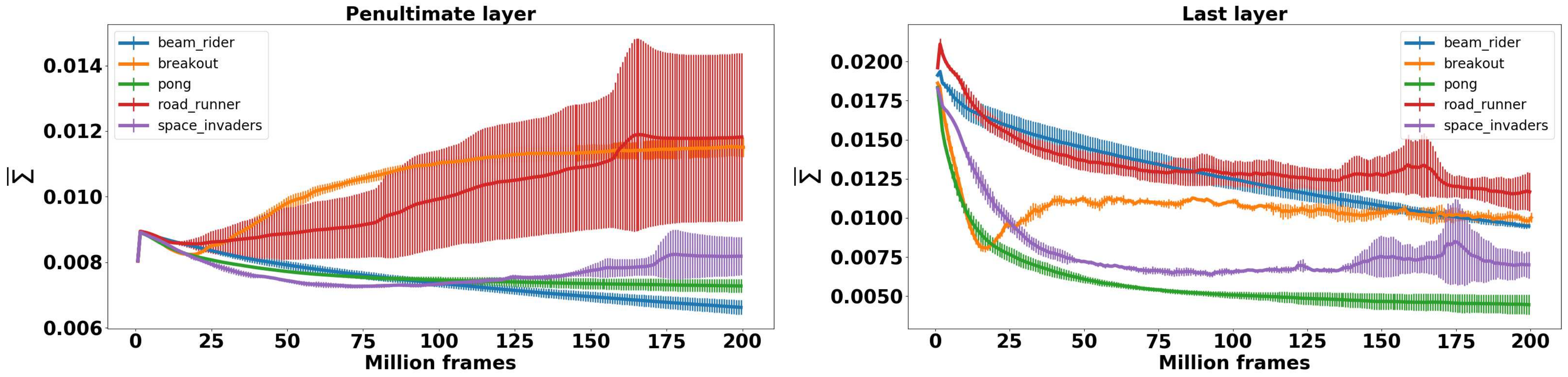


Figure 3: Comparison of the learning curves of the average noise parameter $\bar{\Sigma}$ across five Atari games in NoisyNet-DQN. The results are averaged across 3 seeds and error bars (+/- standard deviation) are plotted.

Figure 3 of "Noisy Networks for Exploration" by Meire Fortunato et al.

The $\bar{\Sigma}$ is the mean-absolute of the noise weights $\boldsymbol{\sigma}_w$, i.e., $\bar{\Sigma} = \frac{1}{\textit{layer size}} \|\boldsymbol{\sigma}_w\|_1$.

Distributional RL

Distributional RL

Instead of an expected return $Q(s,a)$, we could estimate the distribution of expected returns $Z(s,a)$ – the *value distribution*.

The authors define the distributional Bellman operator $\mathcal{T}^\pi$ as:

$$\mathcal{T}^\pi Z(s,a) \stackrel{\text{def}}{=} R(s,a) + \gamma Z(S', A') \quad \text{for} \quad S' \sim p(s,a), A' \sim \pi(S').$$

The authors of the paper prove similar properties of the distributional Bellman operator compared to the regular Bellman operator, mainly being a contraction under a suitable metric.

- For Wasserstein metric W_p, the authors define

  $$\bar{W}_p(Z_1, Z_2) \stackrel{\text{def}}{=} \sup_{s,a} W_p\big(Z_1(s,a), Z_2(s,a)\big)$$

  and prove that $\mathcal{T}^\pi$ is a γ-contraction in $\bar{W}_p$.
- However, $\mathcal{T}^\pi$ is not a contraction in KL divergence nor in total variation distance.

Wasserstein Metric

For two probability distributions μ, ν on a metric space with metric d, Wasserstein metric W_p is defined as

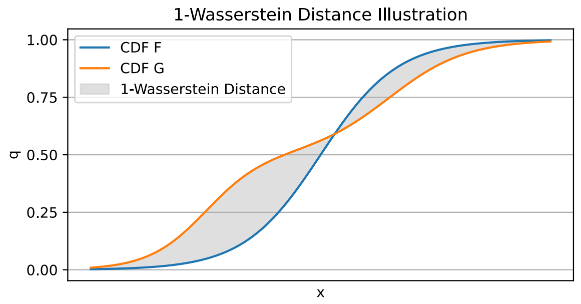
$$W_p(\mu, \nu) \stackrel{\text{def}}{=} \inf_{\gamma \in \Gamma(\mu,\nu)} \Big(\mathbb{E}_{(x,y)\sim\gamma} d(x,y)^p\Big)^{1/p},$$

where $\Gamma(\mu, \nu)$ is a set of all *couplings*, each being a joint probability distribution whose marginals are μ and ν, respectively. A possible intuition is the optimal transport of probability mass from μ to ν.

For distributions over reals with CDFs F, G, the optimal transport has an analytic solution:

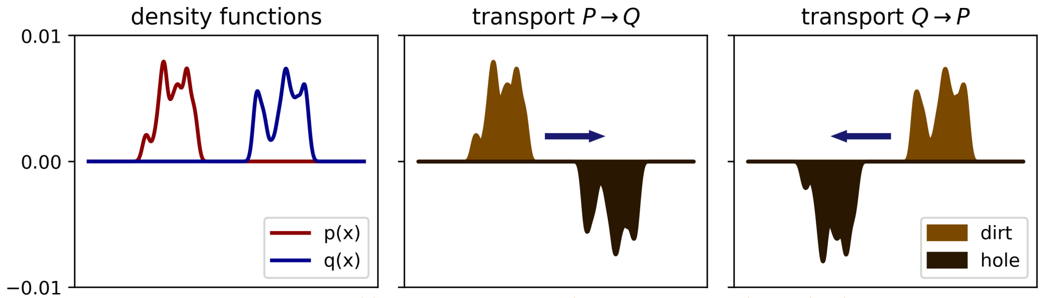
$$W_p(\mu, \nu) = \bigg(\int_0^1 |F^{-1}(q) - G^{-1}(q)|^p \,\mathrm{d}q\bigg)^{1/p},$$

where F^{-1} and G^{-1} are *quantile functions*, i.e., inverse CDFs.

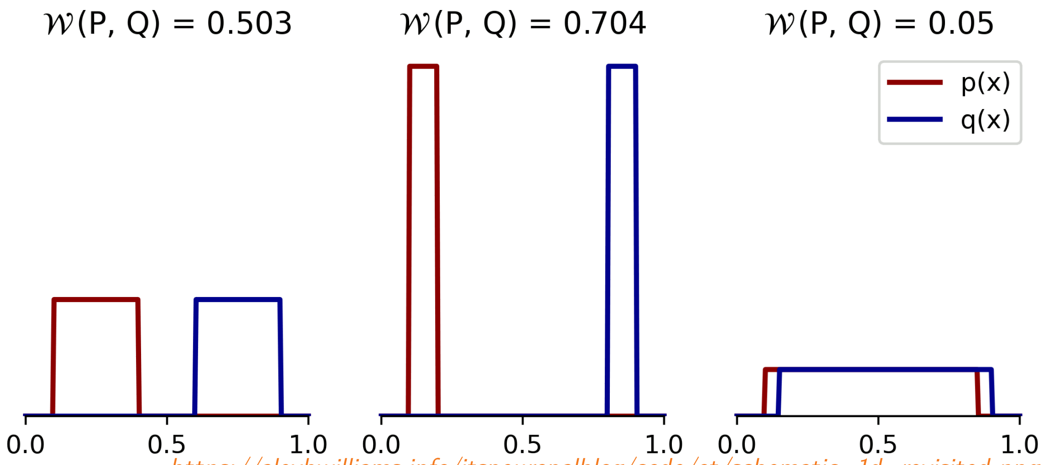


For $p = 1$, the 1-Wasserstein metric correspond to area "between" F and G, and in that case we can compute it also as $W_1(\mu, \nu) = \int_x \big|F(x) - G(x)\big| \,\mathrm{d}x$.

Wasserstein Metric



https://alexhwilliams.info/itsneuronalblog/code/ot/symmetry_1d.png



https://alexhwilliams.info/itsneuronalblog/code/ot/schematic_1d_revisited.png

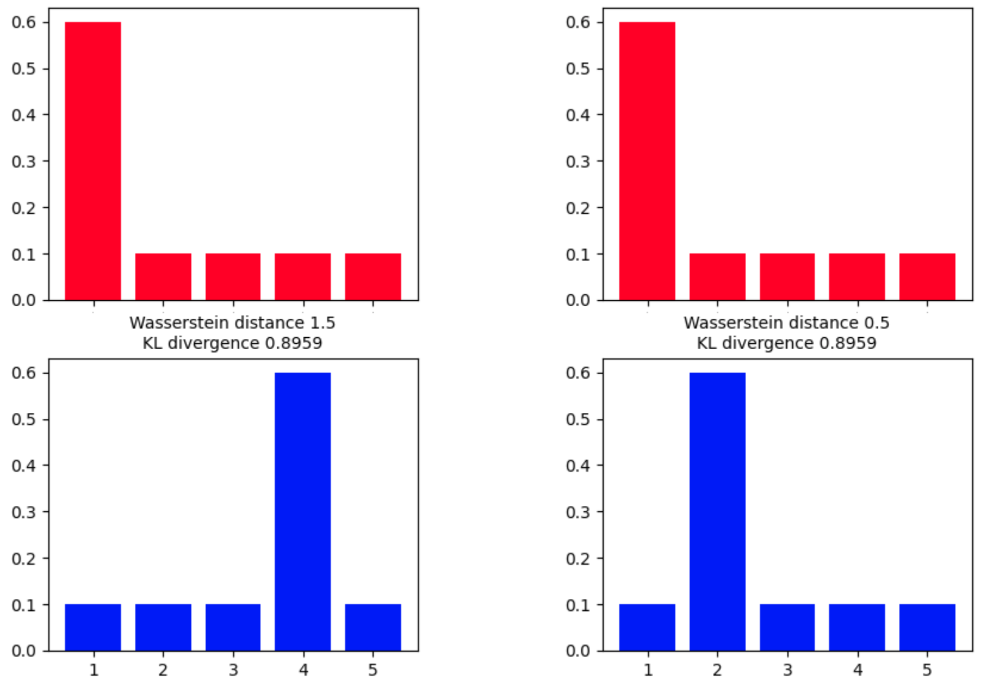


Fig. 1: Difference between Wasserstein distance and Kullback-Leibler (KL) divergence.

Figure 1 of "WATCH: Wasserstein Change Point Detection for High-Dimensional Time Series Data", https://arxiv.org/abs/2201.07125

Distributional RL: C51

The distribution of returns is modeled as a discrete distribution parametrized by the number of atoms $N \in \mathbb{N}$ and by $V_\textrm{MIN}, V_\textrm{MAX} \in \mathbb{R}$. Support of the distribution are atoms

$$\{z_i \stackrel{\textrm{def}}{=} V_\textrm{MIN} + i\Delta z : 0 \leq i < N\} \quad \textrm{for } \Delta z \stackrel{\textrm{def}}{=} \frac{V_\textrm{MAX} - V_\textrm{MIN}}{N-1}.$$

The atom probabilities are predicted using a $\operatorname{softmax}$ distribution as

$$Z_{\boldsymbol\theta}(s, a) = \left\{z_i \textrm{ with probability } p_i = \frac{e^{f_i(s,a;\boldsymbol\theta)}}{\sum_j e^{f_j(s,a;\boldsymbol\theta)}}\right\}.$$

Distributional RL: C51

After the Bellman update, the support of the distribution $R(s, a) + \gamma Z(s', a')$ is not the same as the original support. We therefore project it to the original support by proportionally mapping each atom of the Bellman update to immediate neighbors in the original support.

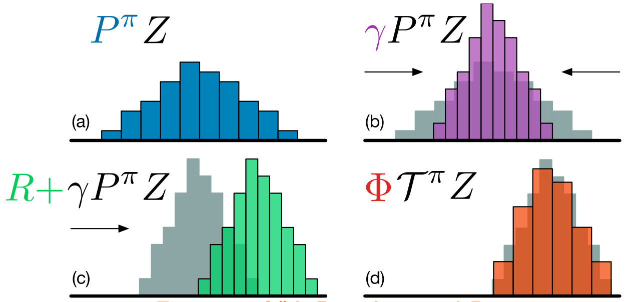


Figure 1 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

$$\Phi\big(R(s, a) + \gamma Z(s', a')\big)_i \overset{\text{def}}{=} \sum_{j=1}^{N} \left[1 - \frac{\left|\left[r + \gamma z_j\right]_{V_{\text{MIN}}}^{V_{\text{MAX}}} - z_i\right|}{\Delta z}\right]_0^1 p_j(s', a').$$

The network is trained to minimize the Kullbeck-Leibler divergence between the current distribution and the (mapped) distribution of the one-step update

$$D_{\text{KL}}\Big(\Phi\big(R + \gamma Z_{\bar{\boldsymbol\theta}}\big(s', \arg\max_{a'} \mathbb{E} Z_{\bar{\boldsymbol\theta}}(s', a')\big)\big)\Big\| Z_{\boldsymbol\theta}\big(s, a\big)\Big).$$

Distributional RL: C51

Algorithm 1 Categorical Algorithm

```
input  A transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \in [0, 1]$
  $Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$
  $a^* \leftarrow \arg\max_a Q(x_{t+1}, a)$
  $m_i = 0, \quad i \in 0, \ldots, N-1$
  for $j \in 0, \ldots, N-1$ do
    # Compute the projection of $\hat{\mathcal{T}} z_j$ onto the support $\{z_i\}$
    $\hat{\mathcal{T}} z_j \leftarrow [r_t + \gamma_t z_j]_{V_{\text{MIN}}}^{V_{\text{MAX}}}$
    $b_j \leftarrow (\hat{\mathcal{T}} z_j - V_{\text{MIN}})/\Delta z$   # $b_j \in [0, N-1]$
    $l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$
    # Distribute probability of $\hat{\mathcal{T}} z_j$
    $m_l \leftarrow m_l + p_j(x_{t+1}, a^*)(u - b_j)$
    $m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - l)$
  end for
output  $-\sum_i m_i \log p_i(x_t, a_t)$   # Cross-entropy loss
```

Algorithm 1 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

Note that by minimizing the D_{KL} instead of the Wasserstein metric W_p, the algorithm has no guarantee of convergence of any kind. However, the authors did not know how to minimize it.

Distributional RL: C51

	Mean	Median	> H.B.	> DQN
DQN	228%	79%	24	0
DDQN	307%	118%	33	43
DUEL.	373%	151%	37	50
PRIOR.	434%	124%	39	48
PR. DUEL.	592%	172%	39	44
C51	**701%**	**178%**	**40**	**50**

Figure 6 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

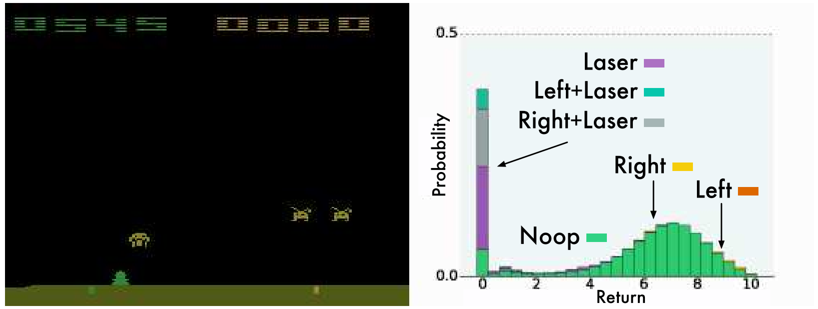


Figure 4. Learned value distribution during an episode of SPACE INVADERS. Different actions are shaded different colours. Returns below 0 (which do not occur in SPACE INVADERS) are not shown here as the agent assigns virtually no probability to them.

Figure 4 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

Distributional RL: C51

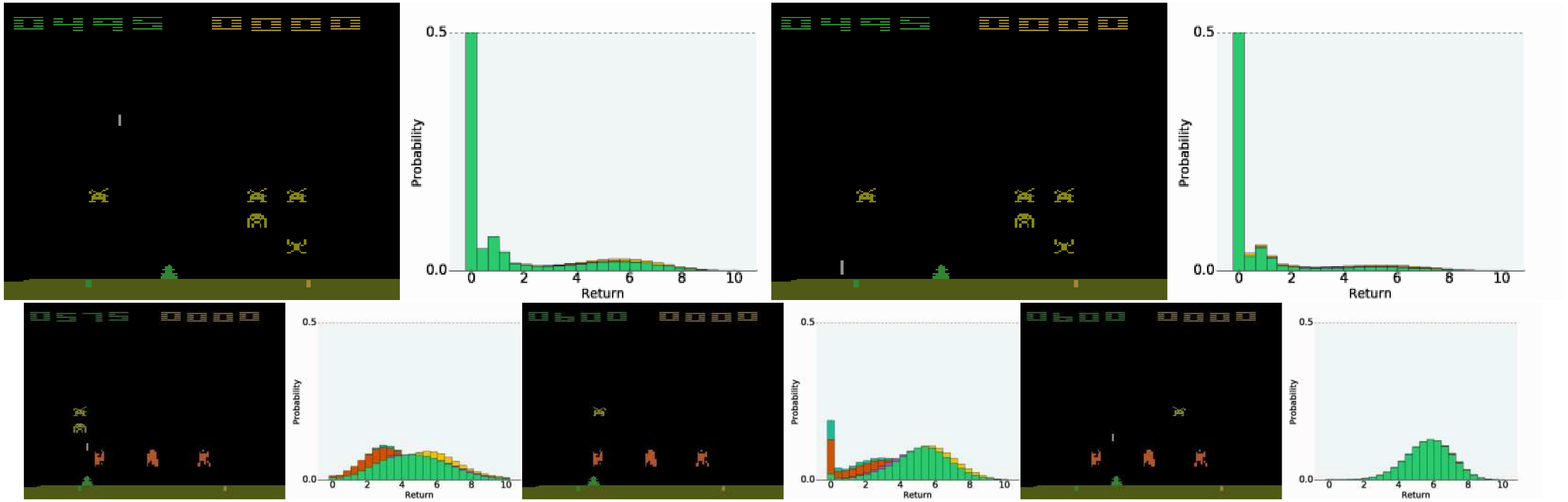


Figure 18. SPACE INVADERS: Top-Left: Multi-modal distribution with high uncertainty. Top-Right: Subsequent frame, a more certain demise. Bottom-Left: Clear difference between actions. Bottom-Middle: Uncertain survival. Bottom-Right: Certain success.

Figure 18 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

Distributional RL: C51

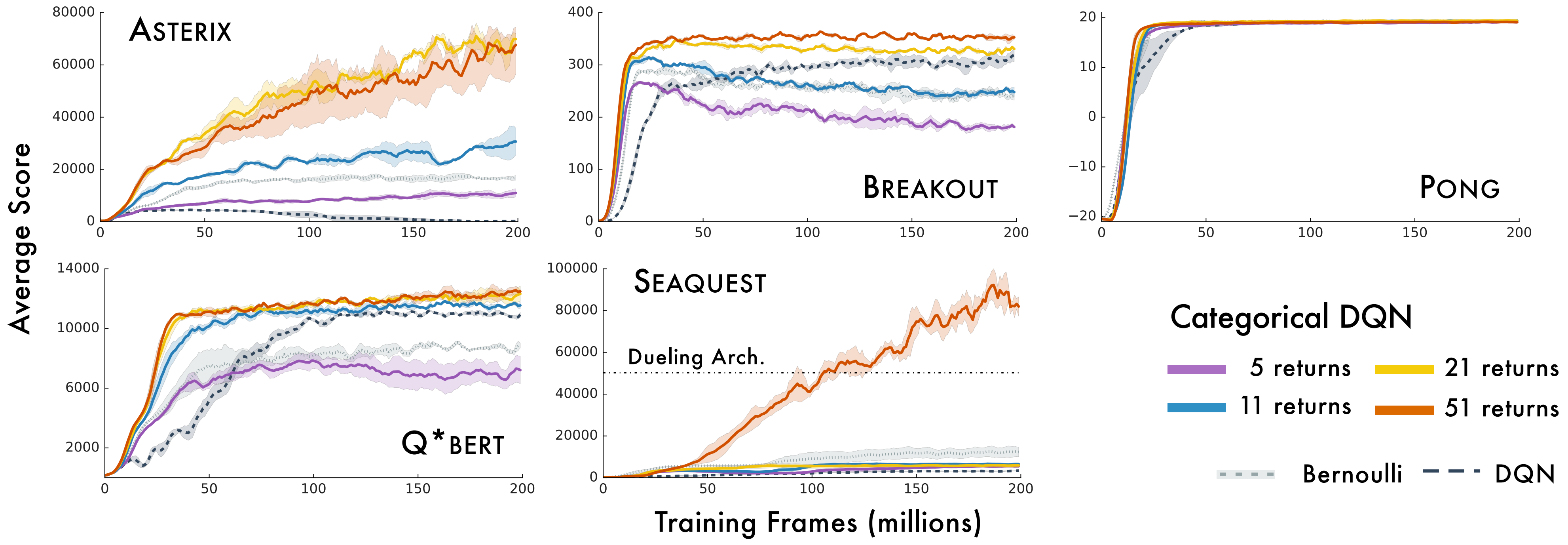


Figure 3. Categorical DQN: Varying number of atoms in the discrete distribution. Scores are moving averages over 5 million frames.

Figure 3 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

Rainbow

Rainbow Architecture

Rainbow combines all described DQN extensions. Instead of 1-step updates, n-step updates are utilized, and KL divergence of the current and target return distribution is minimized:

$$D_{\mathrm{KL}}\Big(\Phi\big(\textstyle\sum_{i=0}^{n-1}\gamma^i R_{t+i+1} + \gamma^n Z_{\bar{\boldsymbol\theta}}\big(S_{t+n}, \arg\max_{a'} \mathbb{E} Z_{\boldsymbol\theta}(S_{t+n}, a')\big)\big)\Big\| Z(S_t, A_t)\Big).$$

The prioritized replay chooses transitions according to the probability

$$p_t \propto D_{\mathrm{KL}}\Big(\Phi\big(\textstyle\sum_{i=0}^{n-1}\gamma^i R_{t+i+1} + \gamma^n Z_{\bar{\boldsymbol\theta}}\big(S_{t+n}, \arg\max_{a'} \mathbb{E} Z_{\boldsymbol\theta}(S_{t+n}, a')\big)\big)\Big\| Z(S_t, A_t)\Big)^w.$$

Network utilizes dueling architecture feeding the shared representation $f(s;\zeta)$ into value computation $V(f(s;\zeta);\eta)$ and advantage computation $A_i(f(s;\zeta), a;\psi)$ for atom z_i, and the final probability of atom z_i in state s and action a is computed as

$$p_i(s,a) \stackrel{\text{def}}{=} \frac{e^{V_i(f(s;\zeta);\eta) + A_i(f(s;\zeta),a;\psi) - \sum_{a'\in\mathcal{A}} A_i(f(s;\zeta),a';\psi)/|\mathcal{A}|}}{\sum_j e^{V_j(f(s;\zeta);\eta) + A_j(f(s;\zeta),a;\psi) - \sum_{a'\in\mathcal{A}} A_j(f(s;\zeta),a';\psi)/|\mathcal{A}|}}.$$

Rainbow Hyperparameters

Finally, we replace all linear layers by their noisy equivalents.

Parameter	Value
Min history to start learning	80K frames
Adam learning rate	0.0000625
Exploration ϵ	0.0
Noisy Nets σ_0	0.5
Target Network Period	32K frames
Adam ϵ	1.5×10^{-4}
Prioritization type	proportional
Prioritization exponent ω	0.5
Prioritization importance sampling β	$0.4 \to 1.0$
Multi-step returns n	3
Distributional atoms	51
Distributional min/max values	$[-10, 10]$

Table 1 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Rainbow Results

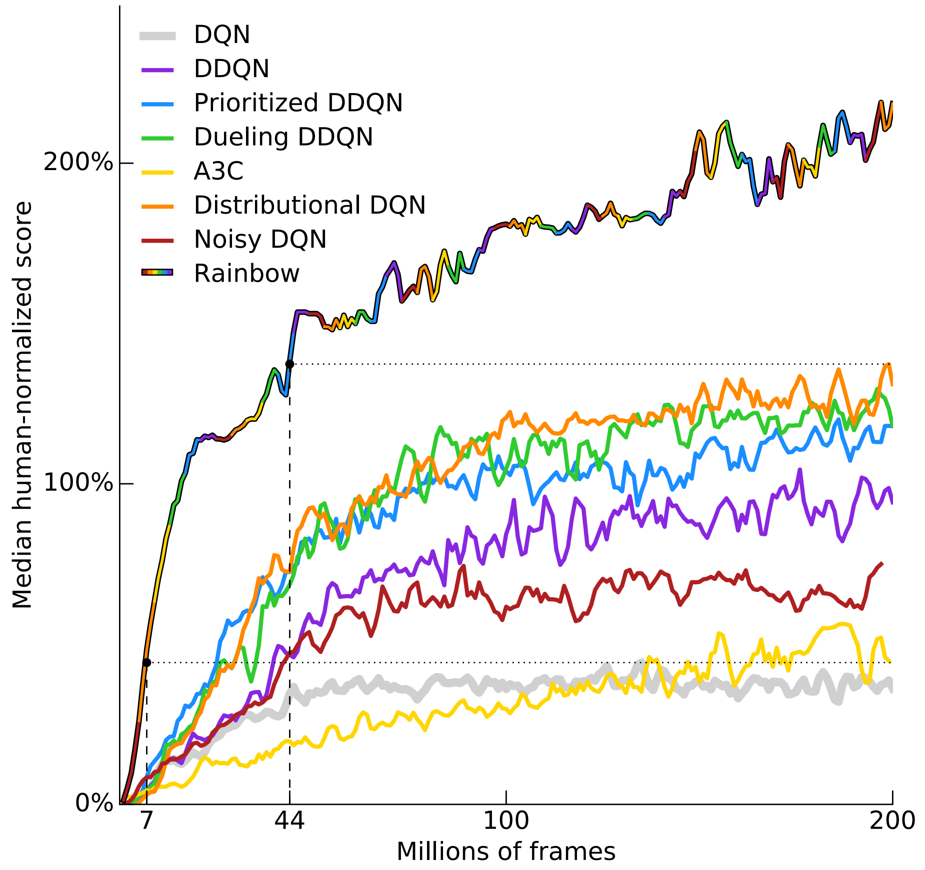


Figure 1 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Agent	no-ops	human starts
DQN	79%	68%
DDQN (*)	117%	110%
Prioritized DDQN (*)	140%	128%
Dueling DDQN (*)	151%	117%
A3C (*)	-	116%
Noisy DQN	118%	102%
Distributional DQN	164%	125%
Rainbow	223%	153%

Table 2 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Rainbow Results

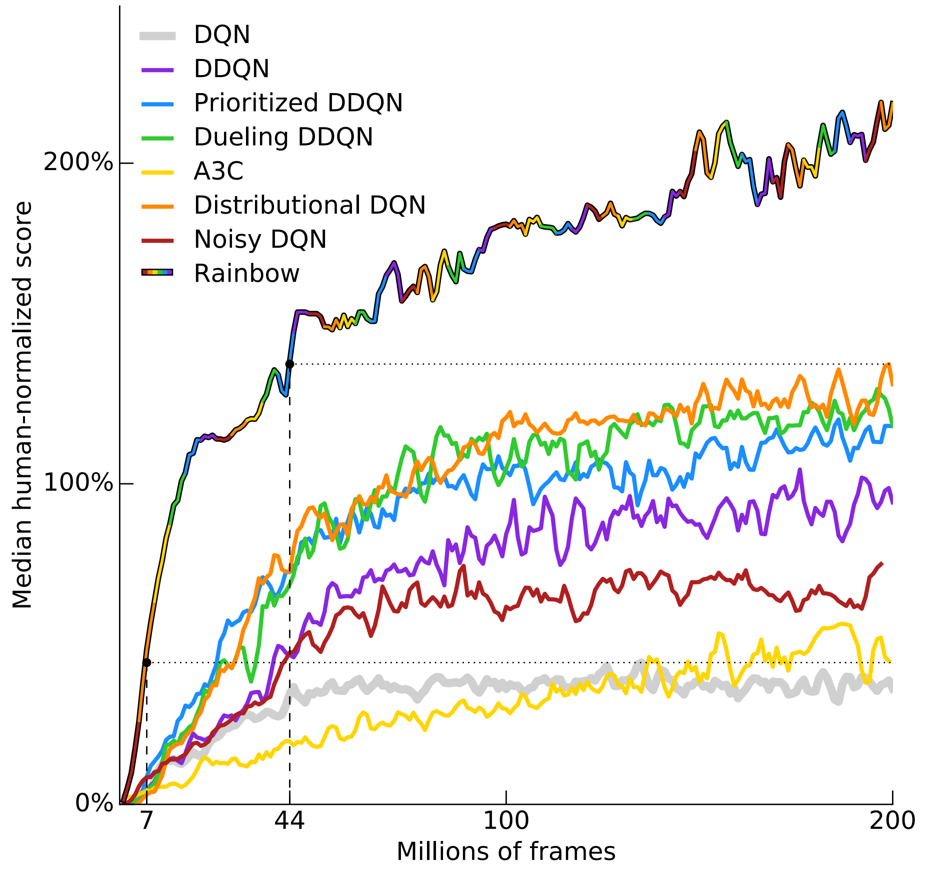


Figure 1 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

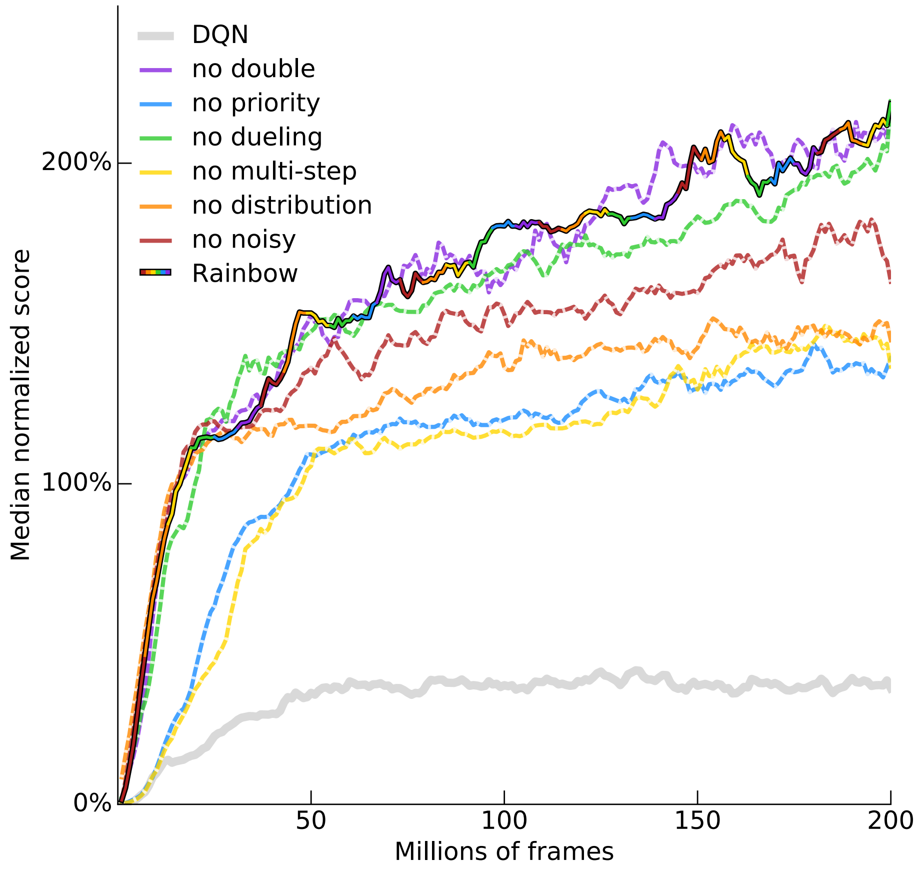


Figure 3 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Rainbow Ablations

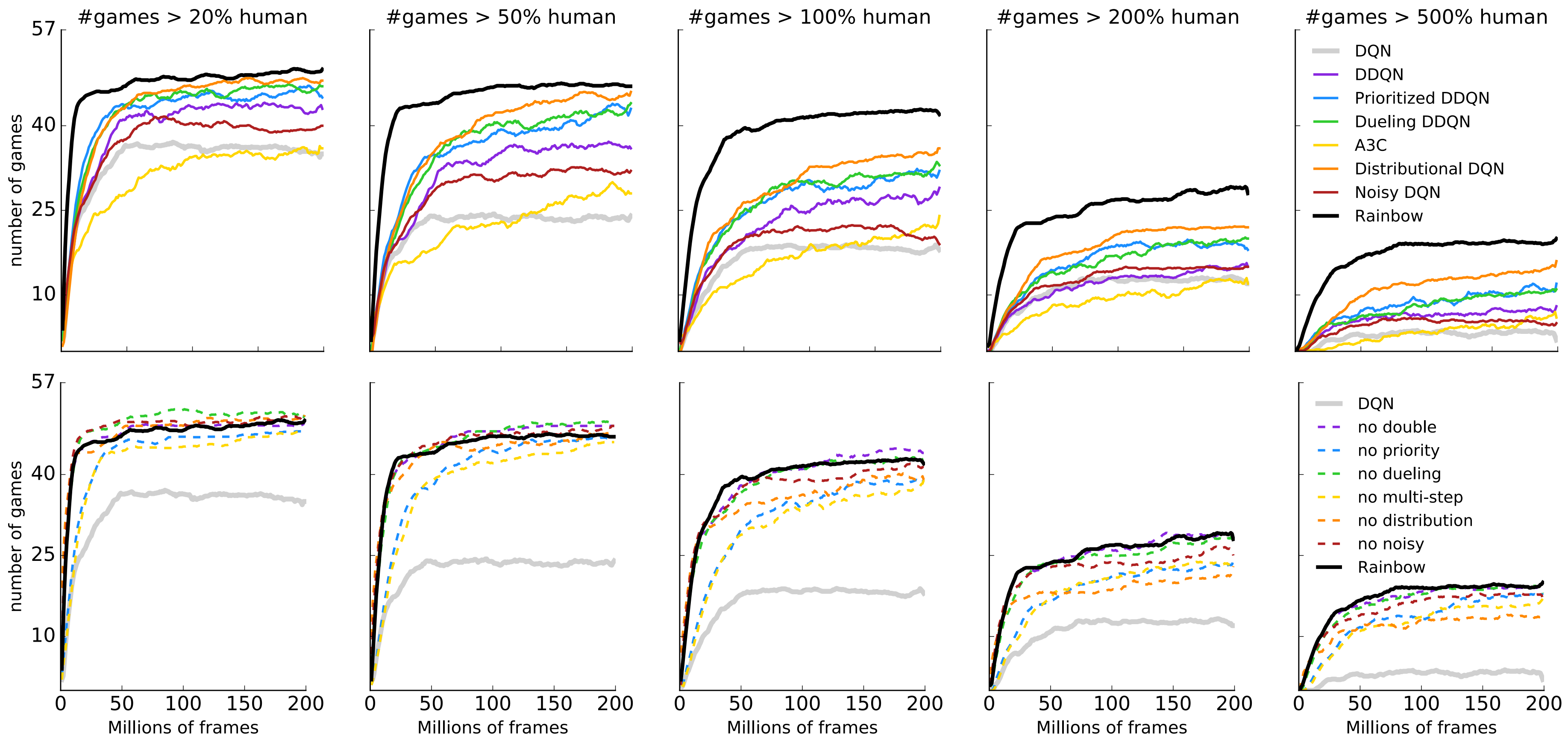


Figure 2: Each plot shows, for several agents, the number of games where they have achieved at least a given fraction of human performance, as a function of time. From left to right we consider the 20%, 50%, 100%, 200% and 500% thresholds. On the first row we compare Rainbow to the baselines. On the second row we compare Rainbow to its ablations.

Figure 2 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Rainbow Ablations

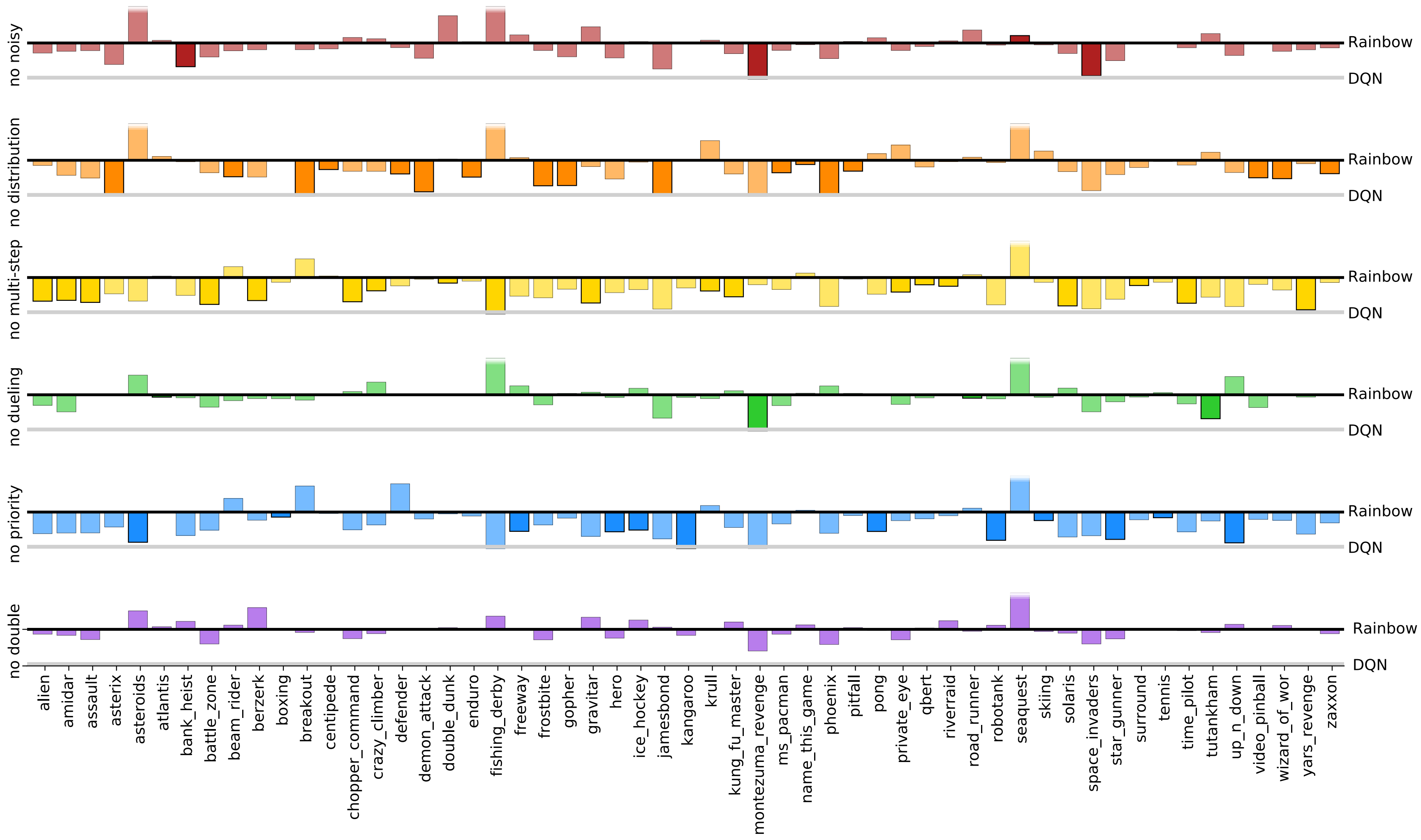


Figure 4 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Quantile Regression

Distributional RL with Quantile Regression

Although the authors of C51 proved that the distributional Bellman operator is a contraction with respect to Wasserstein metric W_p, they were not able to actually minimize it during training; instead, they minimize the KL divergence between the current value distribution and one-step estimate.

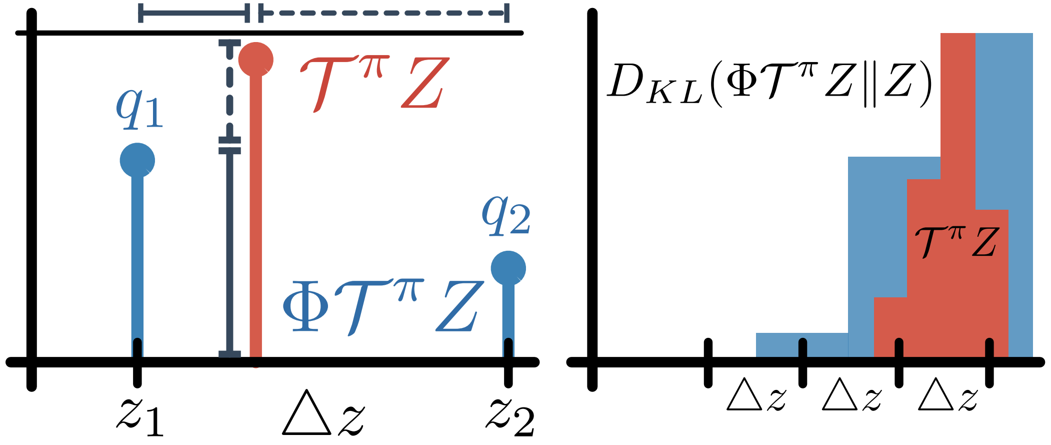


Figure 1: Projection used by C51 assigns mass inversely proportional to distance from nearest support. Update minimizes KL between projected target and estimate.

Figure 1 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Distributional RL with Quantile Regression

The same authors later proposed a different approach, which actually manages to minimize the 1-Wasserstein distance.

In contrast to C51, where $Z(s, a)$ is represented using a discrete distribution on a fixed "comb" support of uniformly spaces locations, we now represent it as a *quantile distribution* – as quantiles $\theta_i(s, a)$ for a fixed probabilities $\tau_1, \ldots, \tau_N$ with $\tau_i = \frac{i}{N}$.

Formally, we can define the quantile distribution as a uniform combination of N Diracs:

$$Z_\theta(s, a) \overset{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} \delta_{\theta_i(s,a)},$$

so that the cumulative density function is a step function increasing by $\frac{1}{N}$ on every quantile θ_i.

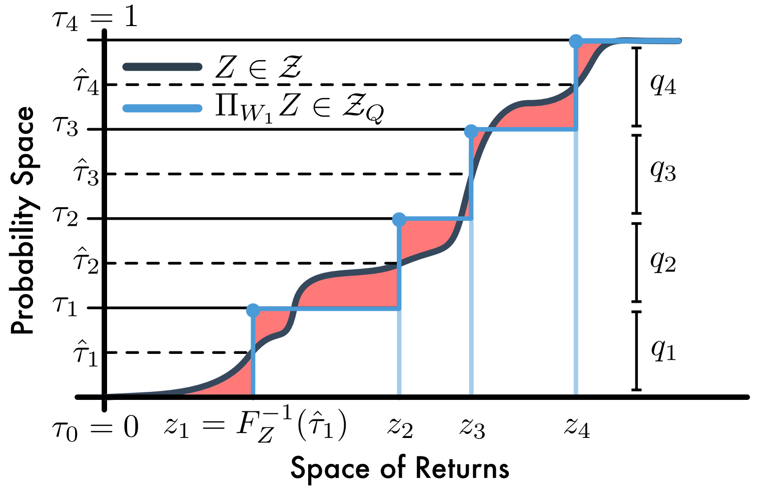


Figure 2: 1-Wasserstein minimizing projection onto $N = 4$ uniformly weighted Diracs. Shaded regions sum to form the 1-Wasserstein error.

Modified Figure 2 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Distributional RL with Quantile Regression

The quantile distribution offers several advantages:

- a fixed support is no longer required;
- the projection step Φ is not longer needed;
- this parametrization enables direct minimization of the Wasserstein loss.

Distributional RL with Quantile Regression

Recall that 1-Wasserstein distance between two distributions μ, ν can be computed as

$$W_1(\mu,\nu) = \int_0^1 \left|F_\mu^{-1}(q) - F_\nu^{-1}(q)\right| \mathrm{d}q,$$

where F_μ, F_ν are their cumulative density functions.

For arbitrary distribution Z, the we denote the most accurate quantile distribution as

$$\Pi_{W_1} Z \stackrel{\text{def}}{=} \arg\min_{Z_\theta} W_1(Z, Z_\theta).$$

In this case, the 1-Wasserstein distance can be written as

$$W_1(Z, Z_\theta) = \sum_{i=1}^N \int_{\tau_{i-1}}^{\tau_i} \left|F_Z^{-1}(q) - \theta_i\right| \mathrm{d}q.$$

Distributional RL with Quantile Regression

It can be proven that for continuous F_Z^{-1}, $W_1(Z, Z_\theta)$ is minimized by (for proof, see Lemma 2 of Dabney et al.: Distributional Reinforcement Learning with Quantile Regression, or consider how the 1-Wasserstein distance changes in the range $[\tau_{i-1}, \tau_i]$ when you move θ_i):

$$\left\{\theta_i \in \mathbb{R} \middle| F_Z(\theta_i) = \frac{\tau_{i-1} + \tau_i}{2}\right\}.$$

We denote the *quantile midpoints* as

$$\hat\tau_i \stackrel{\text{def}}{=} \frac{\tau_{i-1} + \tau_i}{2}.$$

In the paper, the authors prove that the composition $\Pi_{W_1}\mathcal{T}^\pi$ is γ-contraction in $\bar W_\infty$, so repeated application of $\Pi_{W_1}\mathcal{T}^\pi$ converges to a unique fixed point.

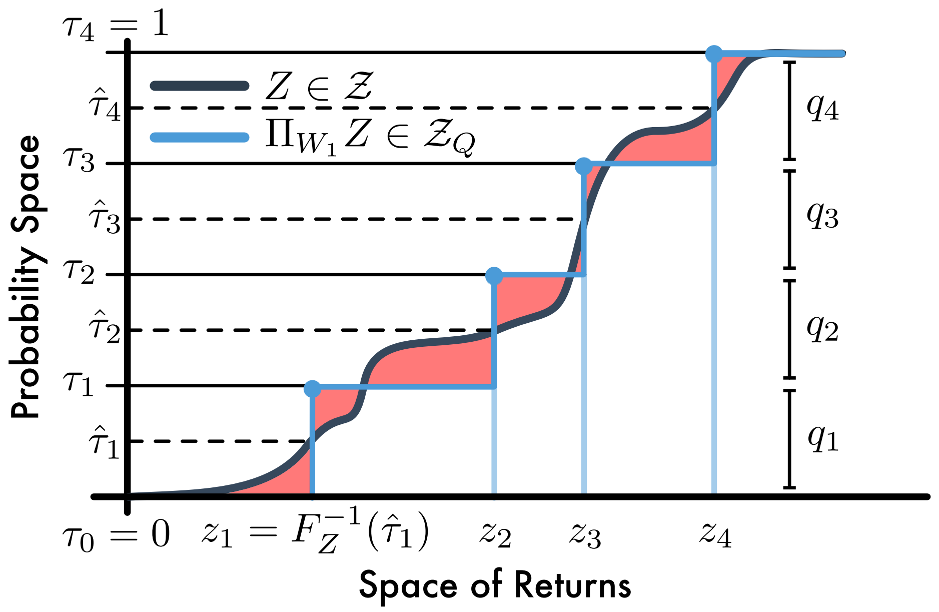


Figure 2: 1-Wasserstein minimizing projection onto $N = 4$ uniformly weighted Diracs. Shaded regions sum to form the 1-Wasserstein error.

Modified Figure 2 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Quantile Regression

Our goal is now to show that it is possible to estimate a quantile $\tau \in [0, 1]$ by minimizing a loss suitable for SGD.

Assume we have samples from a distribution P.

- Minimizing the MSE of $\hat{x}$ and the samples of P,

$$\tilde{x} = \operatorname*{arg\,min}_{\hat{x}} \mathbb{E}_{x \sim P}\left[(x - \hat{x})^2\right],$$

  yields the *mean* of the distribution, $\tilde{x} = \mathbb{E}_{x \sim P}[x]$.

  To show that this holds, we compute the derivative of the loss with respect to $\hat{x}$ and set it to 0, arriving at

$$0 = \mathbb{E}_x[2(\hat{x} - x)] = 2\mathbb{E}_x[\hat{x}] - 2\mathbb{E}_x[x] = 2\big(\hat{x} - \mathbb{E}_x[x]\big).$$

Quantile Regression

Assume we have samples from a distribution P with cumulative density function F_P.

- Minimizing the mean absolute error (MAE) of $\hat{x}$ and the samples of P,

$$\tilde{x} = \arg\min_{\hat{x}} \mathbb{E}_{x \sim P} \big[|x - \hat{x}|\big],$$

yields the *median* of the distribution, $\tilde{x} = F_P^{-1}(0.5)$.

We prove this again by computing the derivative with respect to $\hat{x}$, assuming the functions are nice enough that the Leibnitz integral rule can be used:

$$\begin{aligned}
\frac{\partial}{\partial \hat{x}} \int_{-\infty}^{\infty} P(x)|x - \hat{x}|\,\mathrm{d}x &= \frac{\partial}{\partial \hat{x}} \left[\int_{-\infty}^{\hat{x}} P(x)(\hat{x} - x)\,\mathrm{d}x + \int_{\hat{x}}^{\infty} P(x)(x - \hat{x})\,\mathrm{d}x \right] \\
&= \int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - \int_{\hat{x}}^{\infty} P(x)\,\mathrm{d}x \\
&= 2 \int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - 1 = 2F_P(\hat{x}) - 1 = 2\big(F_P(\hat{x}) - \tfrac{1}{2}\big).
\end{aligned}$$

Leibniz integral rule

The Leibniz integral rule for differentiation under the integral sign states that for $-\infty < a(x), b(x) < \infty$,

$$\frac{\partial}{\partial x}\left[\int_{a(x)}^{b(x)} f(x,t)\,\mathrm{d}t\right] =$$

$$= \int_{a(x)}^{b(x)} \frac{\partial}{\partial x} f(x,t)\,\mathrm{d}t + \left(\frac{\partial}{\partial x} b(x)\right) f\big(x, b(x)\big) - \left(\frac{\partial}{\partial x} a(x)\right) f\big(x, a(x)\big).$$

Sufficient condition for the Leibnitz integral rule to hold is that the $f(x,y)$ and its partial derivative $\frac{\partial}{\partial x} f(x,y)$ are continuous in both x and t, and $a(x)$ and $b(x)$ are continuous and have continuous derivatives.

If any of the bounds is improper, additional conditions must hold, notably that the integral of the partial derivatives of f must converge.

Quantile Regression

Assume we have samples from a distribution P with cumulative density function F_P.

- By generalizing the previous result, we can show that for a quantile $\tau \in [0, 1]$, if

$$\tilde{x} = \operatorname{arg\,min}_{\hat{x}} \mathbb{E}_{x\sim P}\Big[(x - \hat{x})(\tau - [x \le \hat{x}])\Big],$$

then $\tilde{x} = F_P^{-1}(\tau)$. Let $\rho_\tau(x - \hat{x}) \stackrel{\text{def}}{=} (x - \hat{x})(\tau - [x \le \hat{x}]) = |x - \hat{x}| \cdot |\tau - [x \le \hat{x}]|$.

This loss penalizes overestimation errors with weight $1 - \tau$, underestimation errors with τ.

$$\frac{\partial}{\partial \hat{x}} \int_{-\infty}^{\infty} P(x)(x - \hat{x})(\tau - [x \le \hat{x}])\,\mathrm{d}x =$$

$$= \frac{\partial}{\partial \hat{x}} \Bigg[(\tau - 1)\int_{-\infty}^{\hat{x}} P(x)(x - \hat{x})\,\mathrm{d}x + \tau \int_{\hat{x}}^{\infty} P(x)(x - \hat{x})\,\mathrm{d}x\Bigg]$$

$$= (1 - \tau)\int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - \tau \int_{\hat{x}}^{\infty} P(x)\,\mathrm{d}x = \int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - \tau = F_P(\hat{x}) - \tau.$$