Rainbow II, Distributional RL

Milan Straka

March 18, 2024

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

Multi-step Learning

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 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.

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 and all policies are greedy, and all fully connected layers are parametrized as noisy nets (therefore, the network is thought to generate a distribution of rewards; therefore, greedy actions still explore).

Noisy Nets

	Baseline Mean	Baseline Median	NoisyNet Mean	NoisyNet Median	Improvement (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.

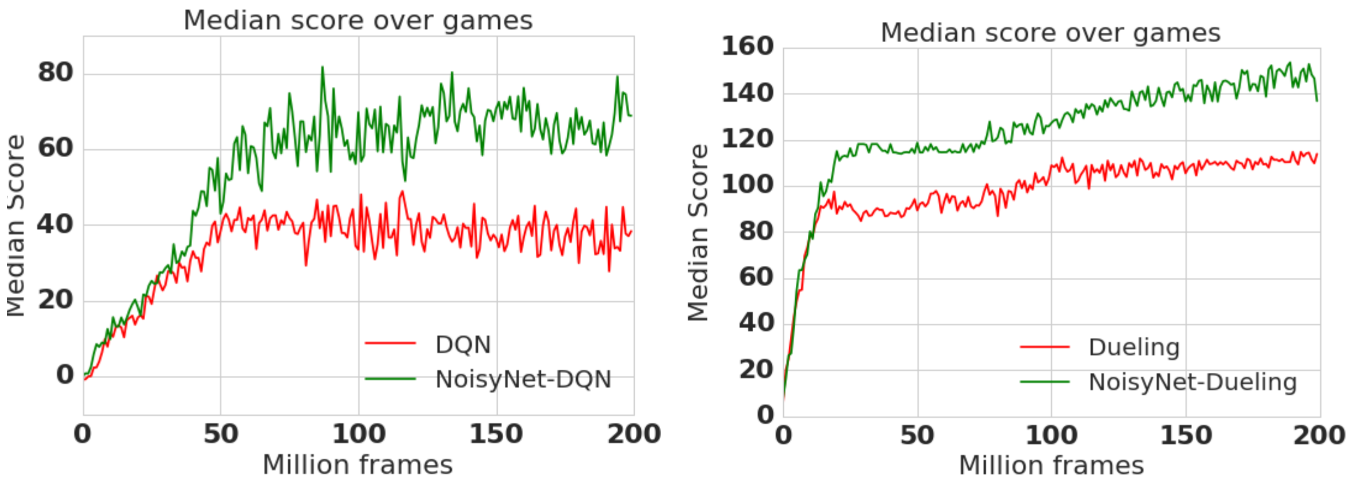


Figure 2 of "Noisy Networks for Exploration" by Meire Fortunato et al.

Noisy Nets

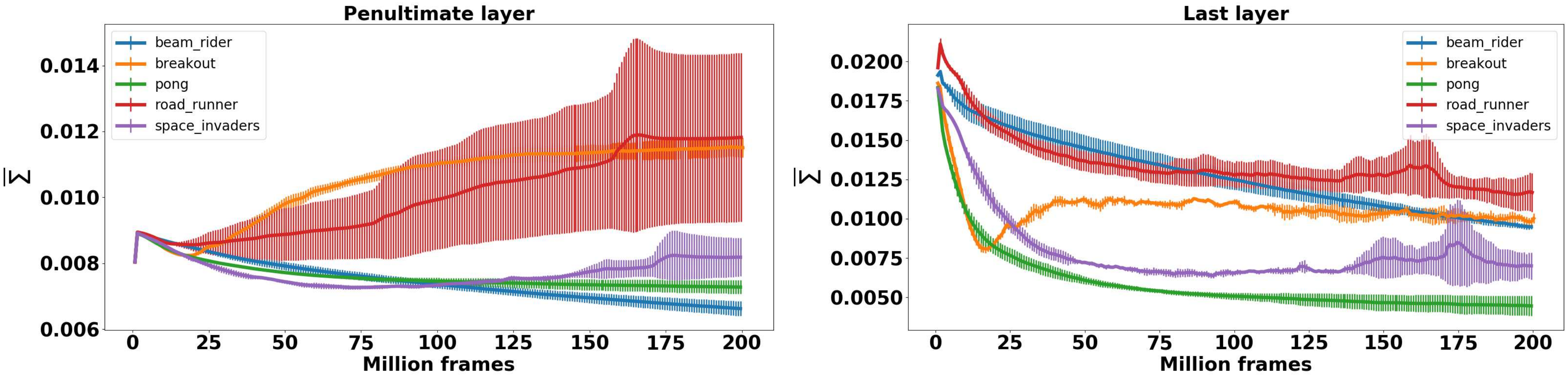


Figure 3: Comparison of the learning curves of the average noise parameter $\bar{\Sigma}$ across five Atari games in NoisyNet-DQN. The results are averaged across 3 seeds and error bars (+/- standard deviation) are plotted.

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

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 $\bar{W}_p$).

For two probability distributions μ, ν, Wasserstein metric W_p is defined as

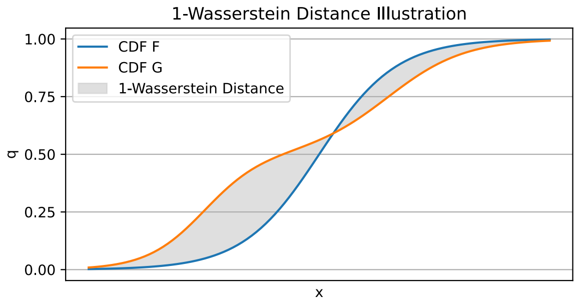
$$W_p(\mu, \nu) \stackrel{\text{def}}{=} \inf_{\gamma \in \Gamma(\mu,\nu)} \big(\mathbb{E}_{(x,y)\sim\gamma} \|x - y\|^d\big)^{1/p},$$

where $\Gamma(\mu, \nu)$ is a set of all *couplings*, each being a 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 $W_1(\mu, \nu) = \int_x \big|F(x) - G(x)\big| \,\mathrm{d}x$.

Distributional RL

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

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

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

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.

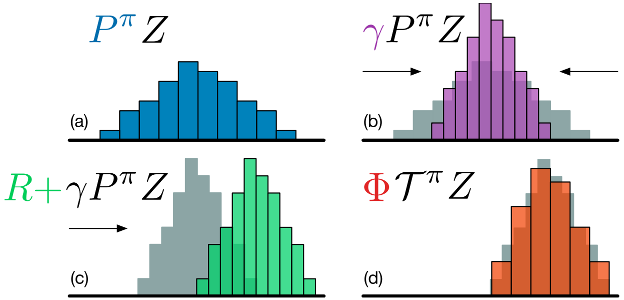


Figure 1 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

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

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

Algorithm 1 Categorical Algorithm

input A transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \in [0, 1]$
  $Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$
  $a^* \leftarrow \arg\max_a Q(x_{t+1}, a)$
  $m_i = 0, \quad i \in 0, \dots, N-1$
  for $j \in 0, \dots, N-1$ **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.

Distributional RL

	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.

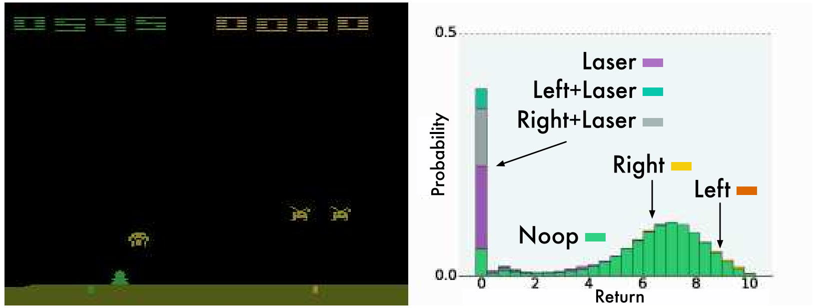


Figure 4. Learned value distribution during an episode of SPACE INVADERS. Different actions are shaded different colours. Returns below 0 (which do not occur in SPACE INVADERS) are not shown here as the agent assigns virtually no probability to them.

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

Distributional RL

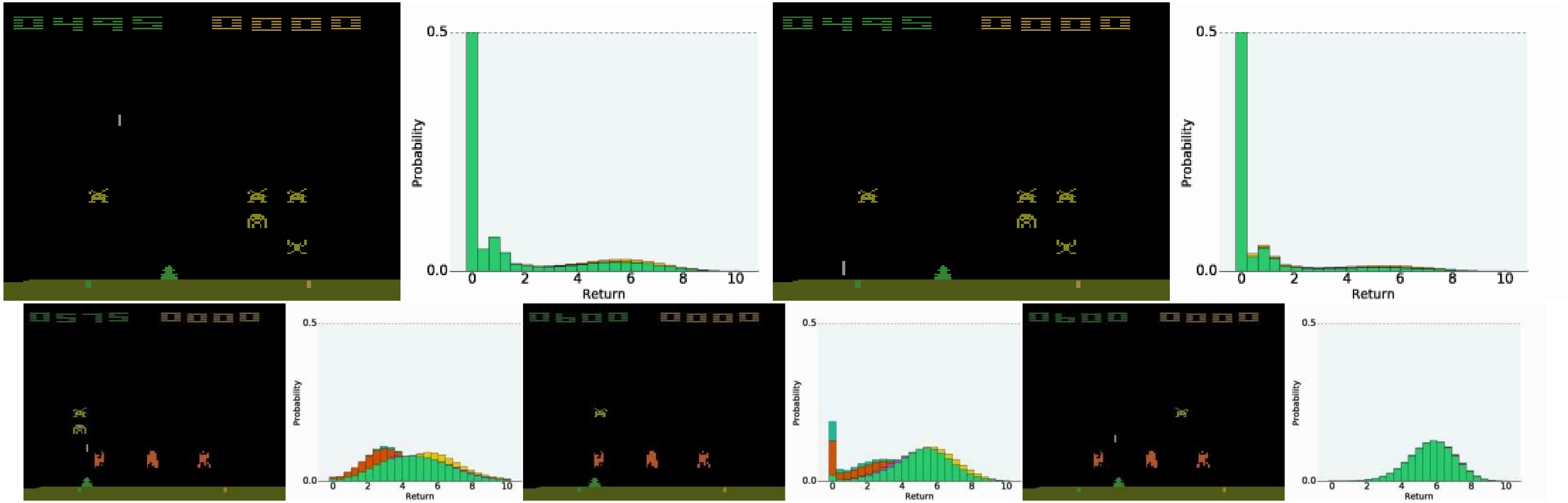


Figure 18. SPACE INVADERS: Top-Left: Multi-modal distribution with high uncertainty. Top-Right: Subsequent frame, a more certain demise. Bottom-Left: Clear difference between actions. Bottom-Middle: Uncertain survival. Bottom-Right: Certain success.

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

Distributional RL

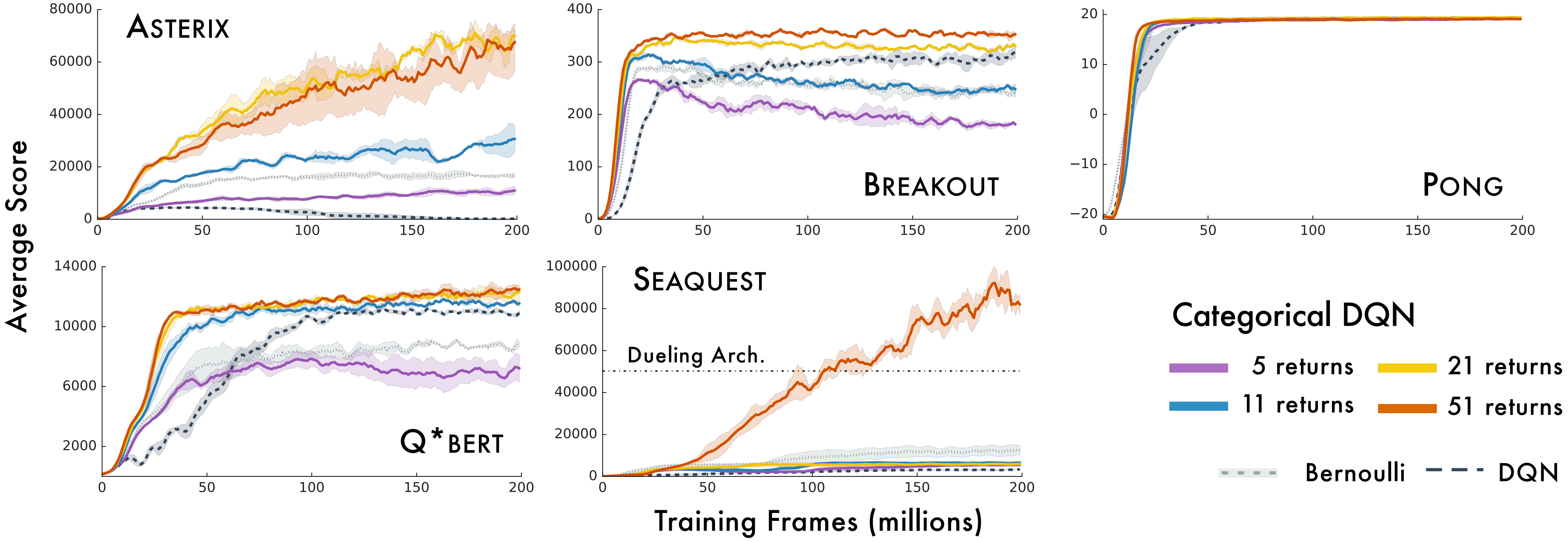


Figure 3. Categorical DQN: Varying number of atoms in the discrete distribution. Scores are moving averages over 5 million frames.

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

Rainbow 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:

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

The prioritized replay chooses transitions according to the probability

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

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.

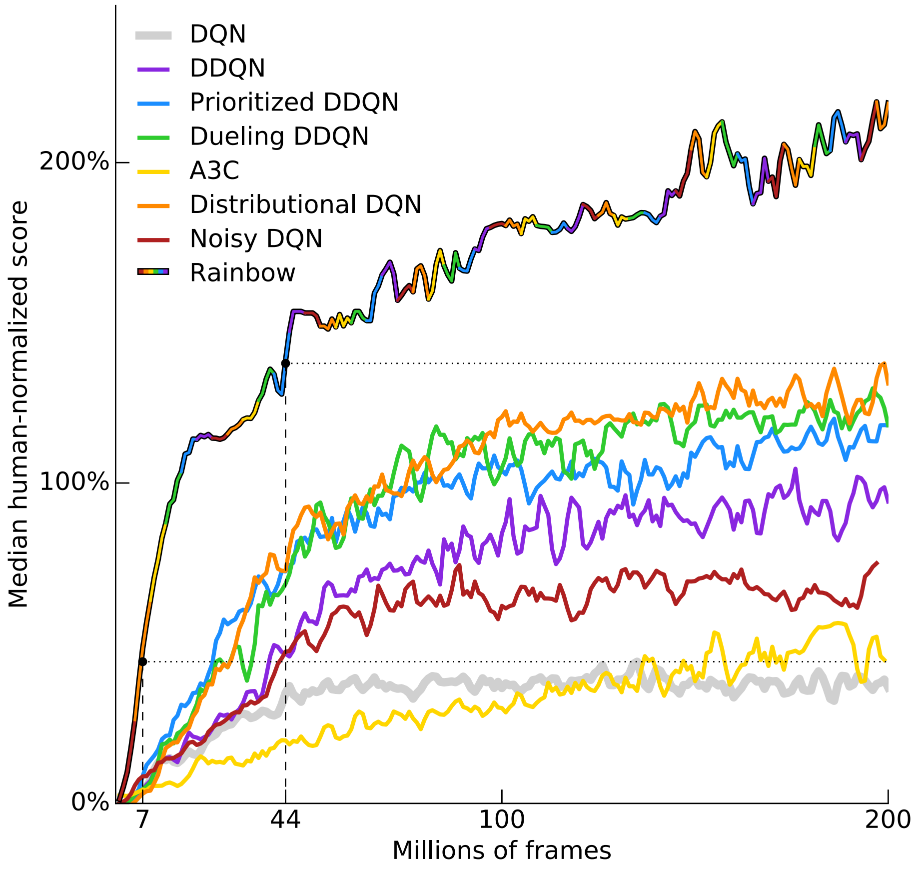


Figure 1 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

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

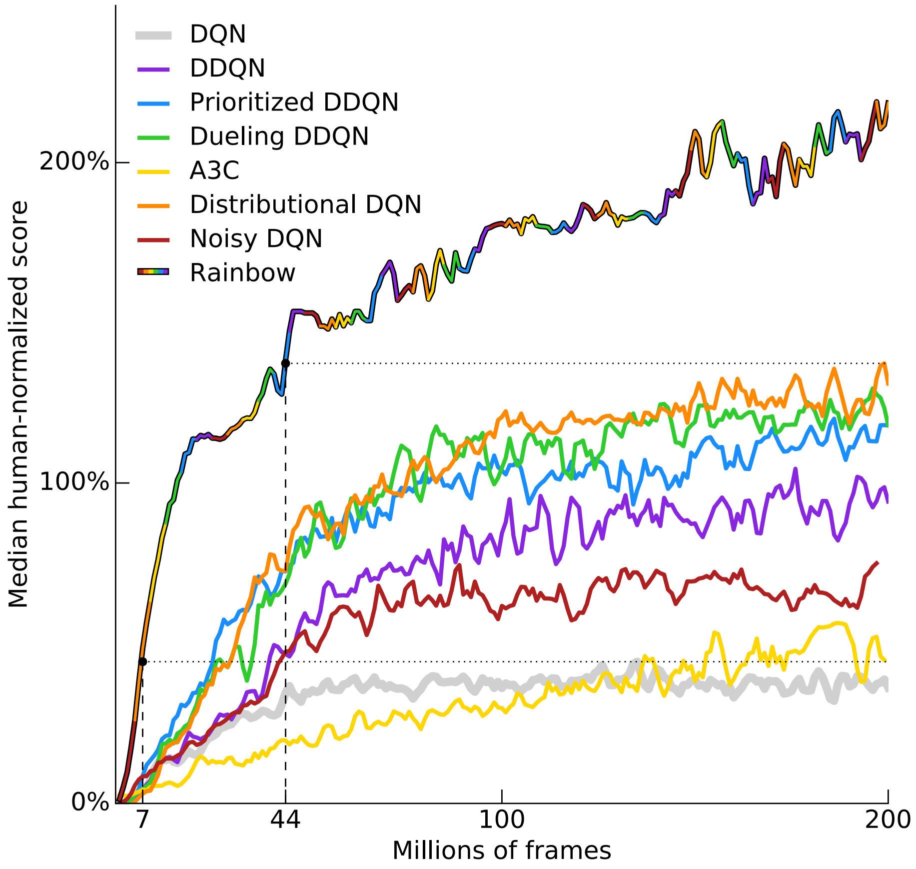


Figure 1 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

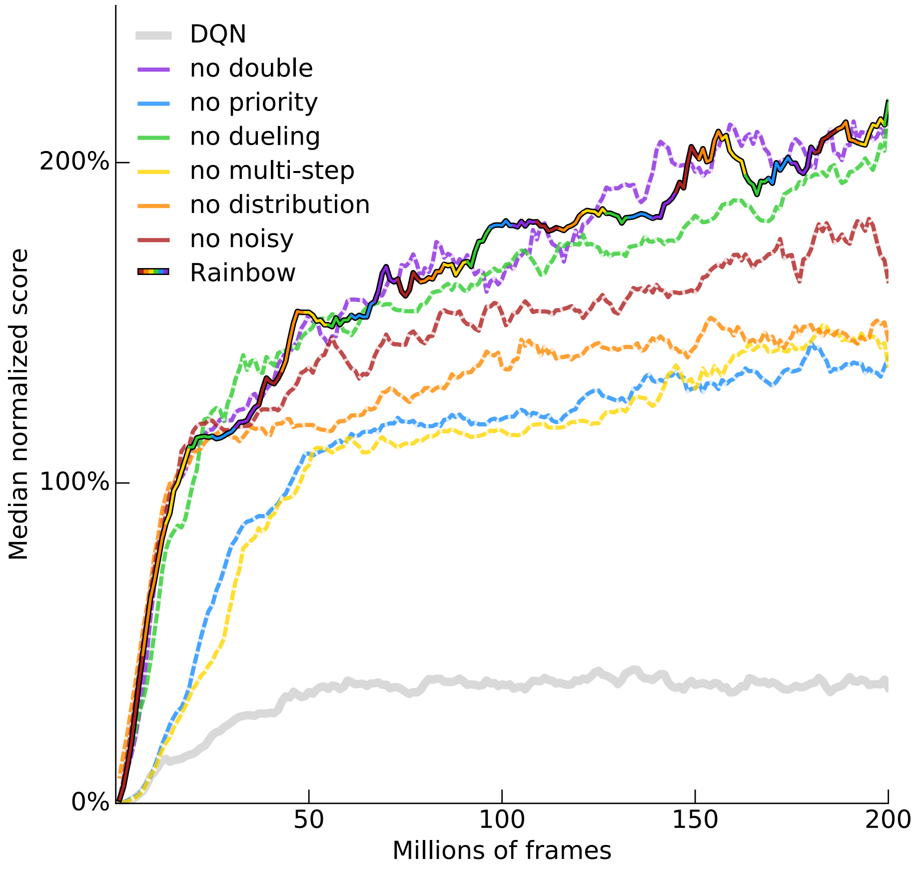


Figure 3 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

Rainbow Ablations

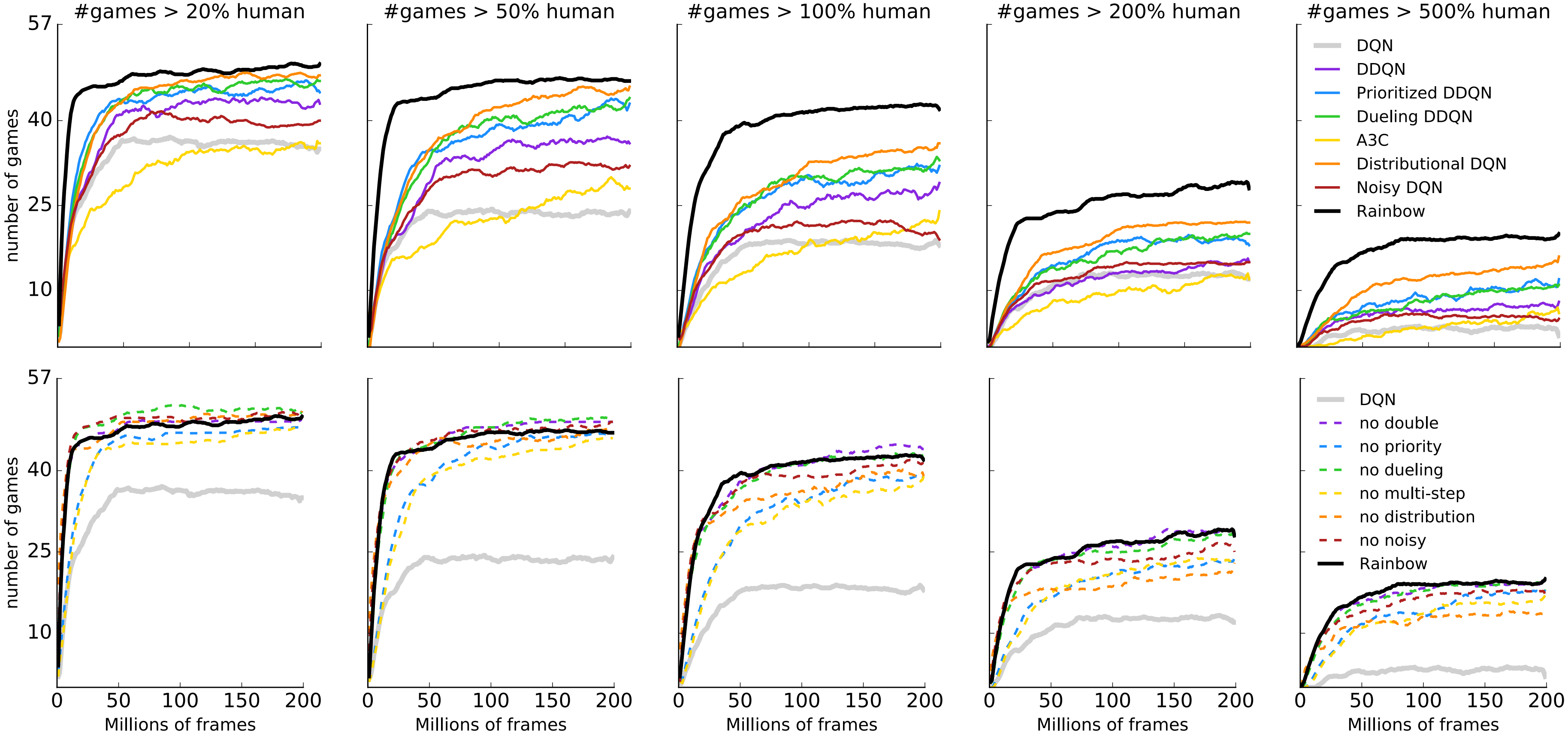


Figure 2: Each plot shows, for several agents, the number of games where they have achieved at least a given fraction of human performance, as a function of time. From left to right we consider the 20%, 50%, 100%, 200% and 500% thresholds. On the first row we compare Rainbow to the baselines. On the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

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

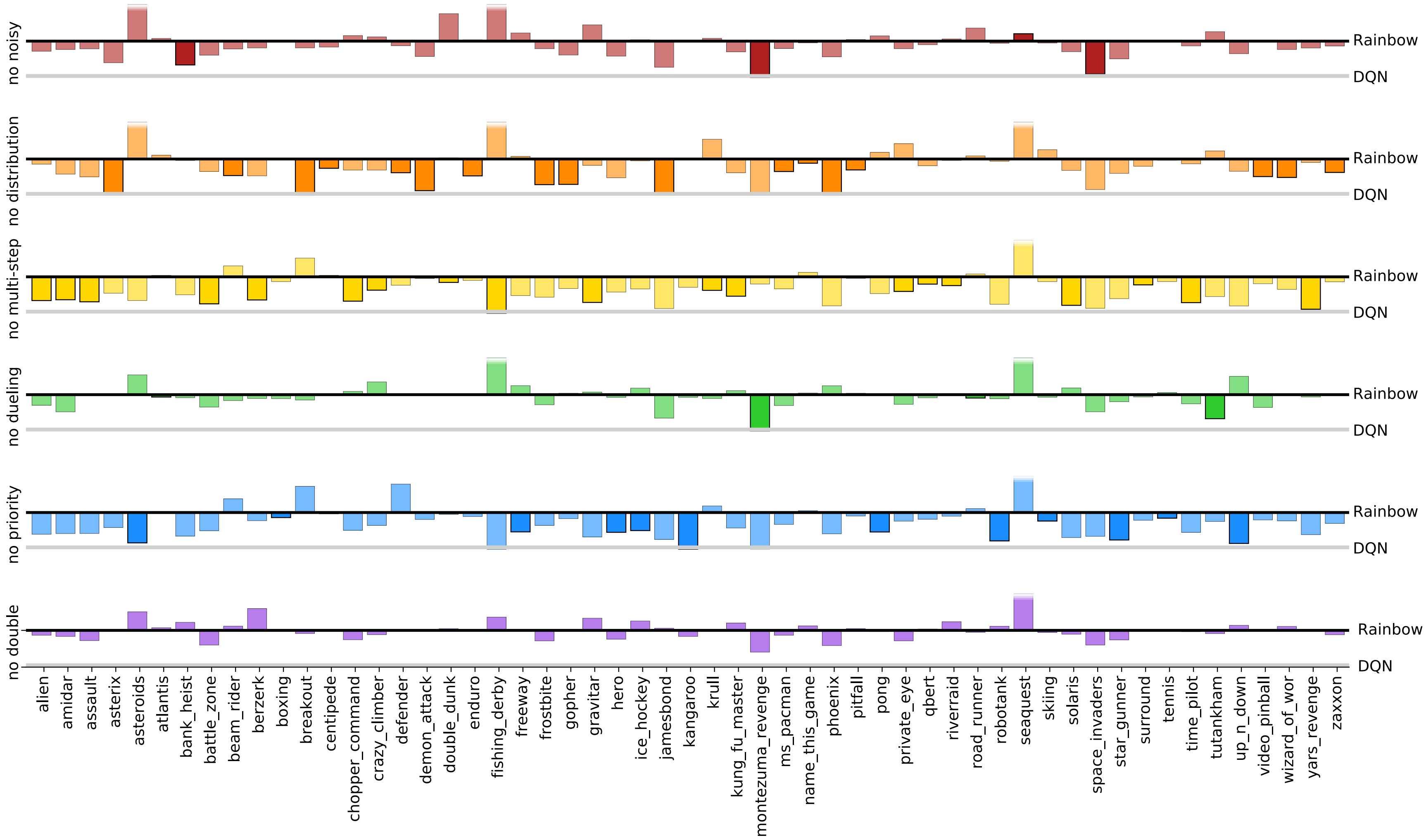


Figure 4 of "Rainbow: Combining Improvements in Deep Reinforcement Learning" by Matteo Hessel et al.

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.

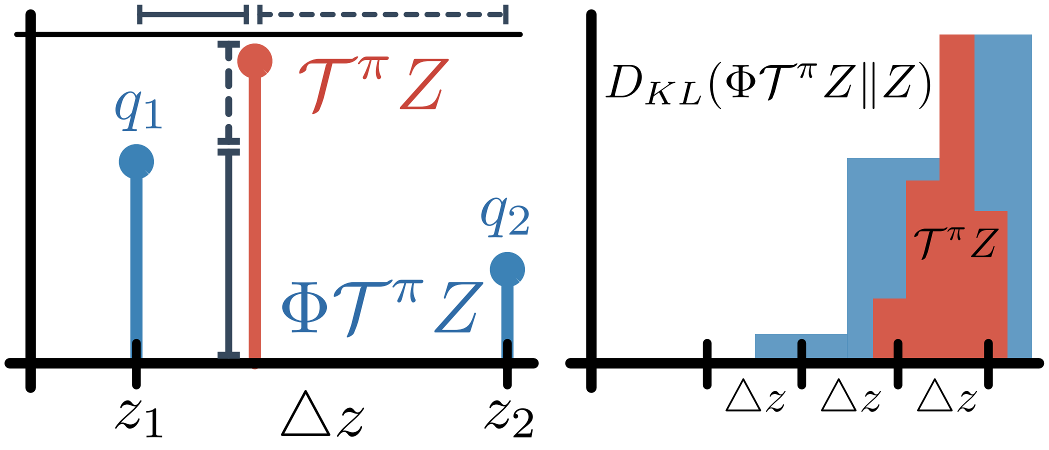


Figure 1: Projection used by C51 assigns mass inversely proportional to distance from nearest support. Update minimizes KL between projected target and estimate.

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

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:

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

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

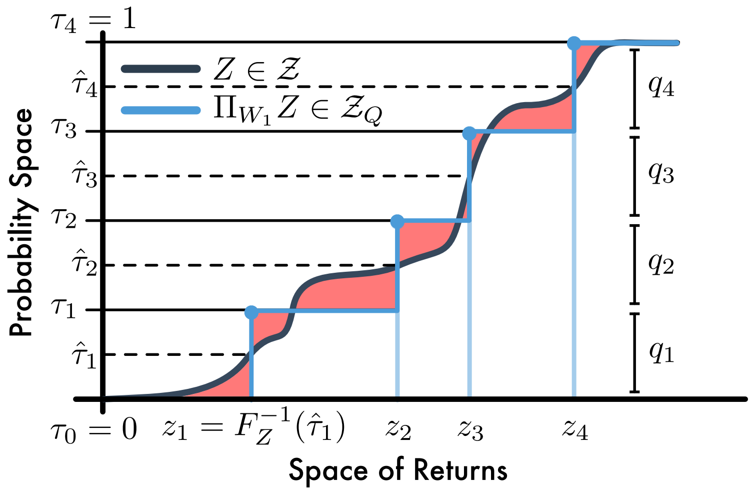


Figure 2: 1-Wasserstein minimizing projection onto $N = 4$ uniformly weighted Diracs. Shaded regions sum to form the 1-Wasserstein error.

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

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.

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

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

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.

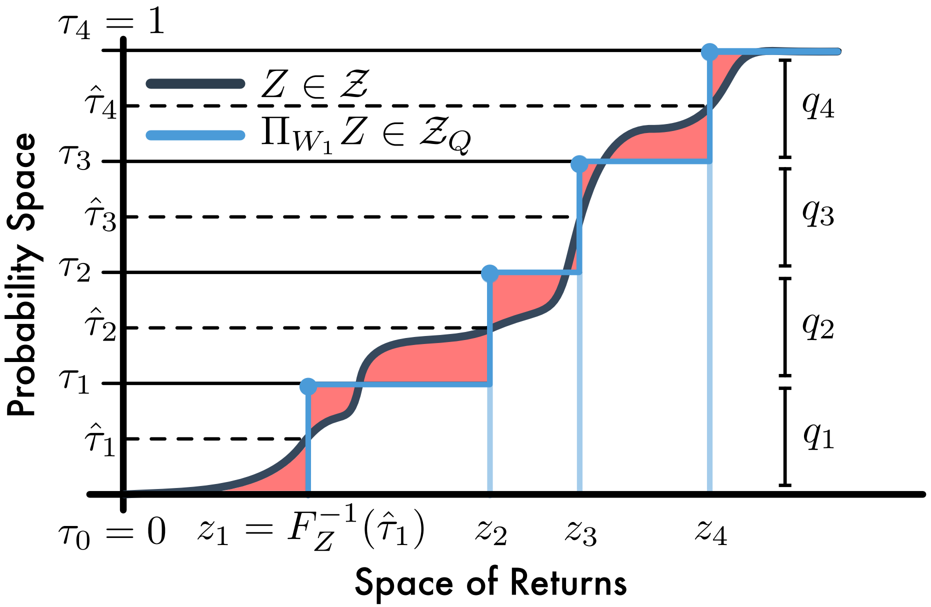


Figure 2: 1-Wasserstein minimizing projection onto $N = 4$ uniformly weighted Diracs. Shaded regions sum to form the 1-Wasserstein error.

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,

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

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,

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

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:

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

Leibniz integral rule

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

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

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

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

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

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

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

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

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

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

Quantile Regression

Using the quantile regression, when we have a value distribution Z, we can find the most accurate quantile distribution by minimizing

$$\sum_{i=1}^N \mathbb{E}_{z \sim Z} \big[\rho_{\hat\tau_i}(z - \theta)\big].$$

However, the quantile loss is not smooth around zero, which could limit performance when training a model. The authors therefore propose the **quantile Huber loss**, which acts as an asymmetric squared loss in interval $[-\kappa, \kappa]$ and fall backs to the standard quantile loss outside this range.

Specifically, let

$$\rho_\tau^\kappa(z - \theta) \stackrel{\textrm{def}}{=} \begin{cases} \big|[z \geq \theta] - \tau\big| \cdot \frac{1}{2}\big(z - \theta\big)^2 & \textrm{if } |z - \theta| \leq \kappa, \\ \big|[z \geq \theta] - \tau\big| \cdot \kappa\big(|z - \theta| - \frac{1}{2}\kappa\big) & \textrm{otherwise.} \end{cases}$$

To conclude, in DR-DQN-κ, the network for a given state predicts $\mathbb{R}^{|\mathcal{A}|\times N}$, so N quantiles for every action.

The following loss is used:

Algorithm 1 Quantile Regression Q-Learning

Require: N, κ
input $x, a, r, x', \gamma \in [0, 1)$
  # Compute distributional Bellman target
  $Q(x', a') := \sum_j q_j \theta_j(x', a')$
  $a^* \leftarrow \arg\max_{a'} Q(x, a')$
  $\mathcal{T}\theta_j \leftarrow r + \gamma\theta_j(x', a^*), \quad \forall j$
  # Compute quantile regression loss (Equation 10)
output $\sum_{i=1}^{N} \mathbb{E}_j \left[\rho^{\kappa}_{\hat\tau_i}(\mathcal{T}\theta_j - \theta_i(x, a))\right]$

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

The q_j is just $\frac{1}{N}$.

Distributional RL with Quantile Regression

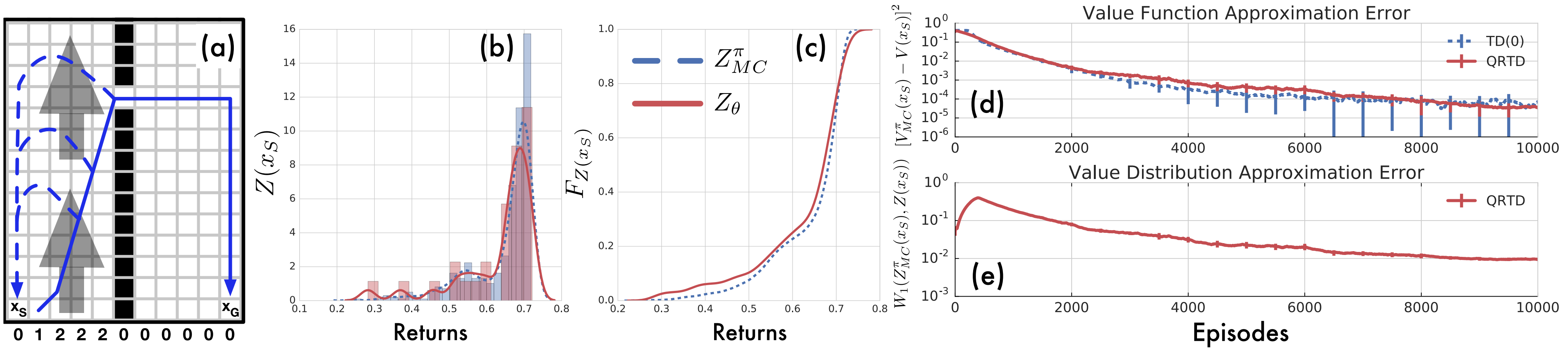


Figure 3: (a) Two-room windy gridworld, with wind magnitude shown along bottom row. Policy trajectory shown by blue path, with additional cycles caused by randomness shown by dashed line. (b, c) (Cumulative) Value distribution at start state x_S, estimated by MC, Z^π_{MC}, and by QRTD, Z_θ. (d, e) Value function (distribution) approximation errors for TD(0) and QRTD.

Figure 3 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Each state transition has probability of 0.1 of moving in a random direction.

Distributional RL with Quantile Regression

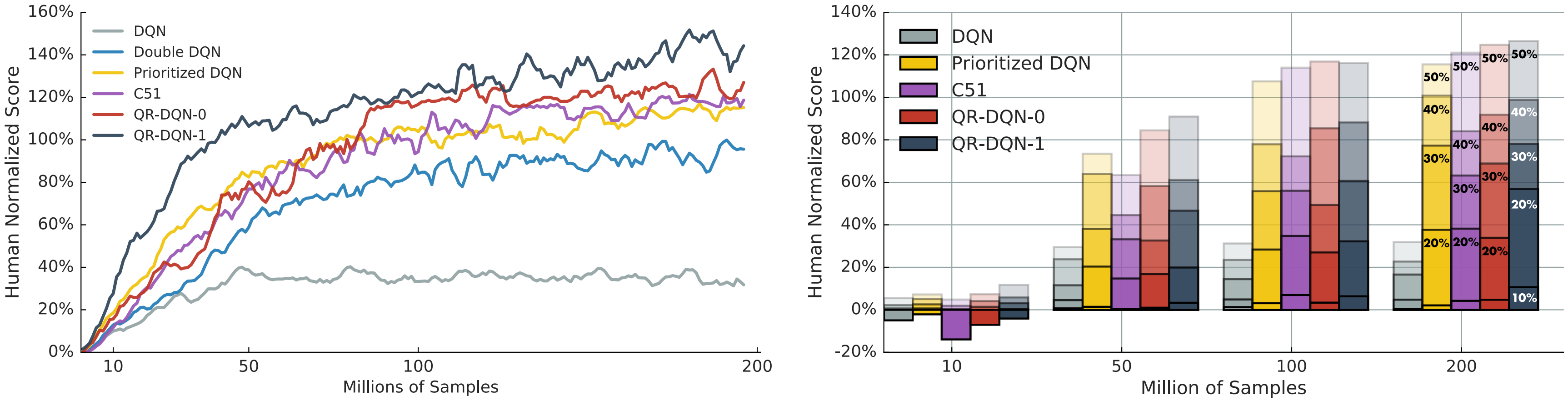


Figure 4: Online evaluation results, in human-normalized scores, over 57 Atari 2600 games for 200 million training samples. (Left) Testing performance for one seed, showing median over games. (Right) Training performance, averaged over three seeds, showing percentiles (10, 20, 30, 40, and 50) over games.

Figure 4 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

	Mean	**Median**	**>human**	**>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
QR-DQN-0	881%	199%	38	52
QR-DQN-1	**915%**	**211%**	**41**	**54**

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

In IQN (implicit quantile regression), the authors (again the same team as in C51 and DR-DQN) generalize the value distribution representation to predict *any given quantile* τ.

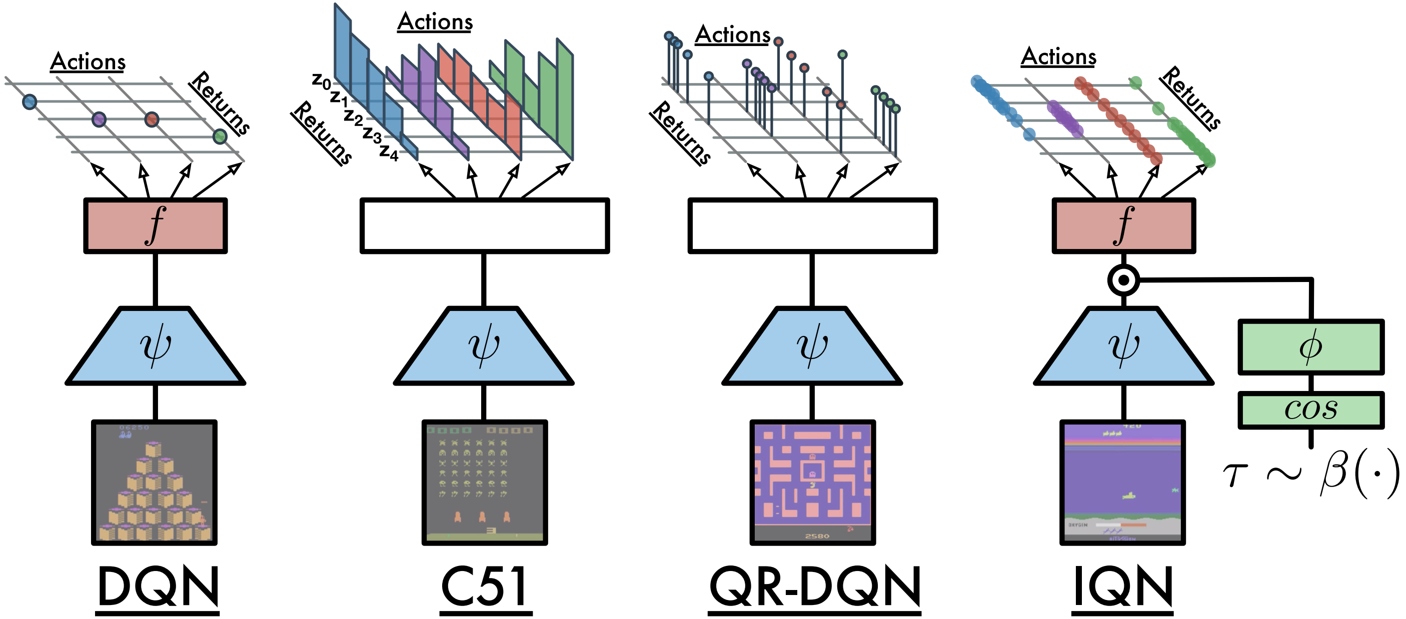


Figure 1. Network architectures for DQN and recent distributional RL algorithms.

Figure 1 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Implicit Quantile Networks for Distributional RL

The quantile τ of the value distribution, $Z_\tau(s, a)$, is modeled as

$$Z_\tau(s, a) \approx f\big(\psi(s) \odot \varphi(\tau)\big)_a.$$

- Other ways than multiplicative combinations were tried (concat, residual), but the multiplicative form delivered the best results.
- The quantile τ is represented using trainable cosine embeddings with dimension $n = 64$:

$$\varphi_j(\tau) \stackrel{\text{def}}{=} \operatorname{ReLU}\Big(\sum\nolimits_{i=0}^{n-1} \cos(\pi i \tau) w_{i,j} + b_j\Big).$$

Implicit Quantile Networks for Distributional RL

The overall loss is:

Algorithm 1 Implicit Quantile Network Loss

Require: N, N', K, κ and functions β, Z

input $x, a, r, x', \gamma \in [0, 1)$

    # Compute greedy next action

    $a^* \leftarrow \arg\max_{a'} \frac{1}{K} \sum_k^K Z_{\tilde{\tau}_k}(x', a'), \qquad \tilde{\tau}_k \sim \beta(\cdot)$

    # Sample quantile thresholds

    $\tau_i, \tau'_j \sim U([0, 1]), \qquad 1 \leq i \leq N, 1 \leq j \leq N'$

    # Compute distributional temporal differences

    $\delta_{ij} \leftarrow r + \gamma Z_{\tau'_j}(x', a^*) - Z_{\tau_i}(x, a), \qquad \forall i, j$

    # Compute Huber quantile loss

output $\sum_{i=1}^N \mathbb{E}_{\tau'} \left[\rho^\kappa_{\tau_i}(\delta_{ij})\right]$

Algorithm 1 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Note the different roles of N and N'.

Implicit Quantile Networks for Distributional RL

The authors speculate that:

- large N may increase sample complexity (faster learning),
- larger N' could reduce variance (like a minibatch size).

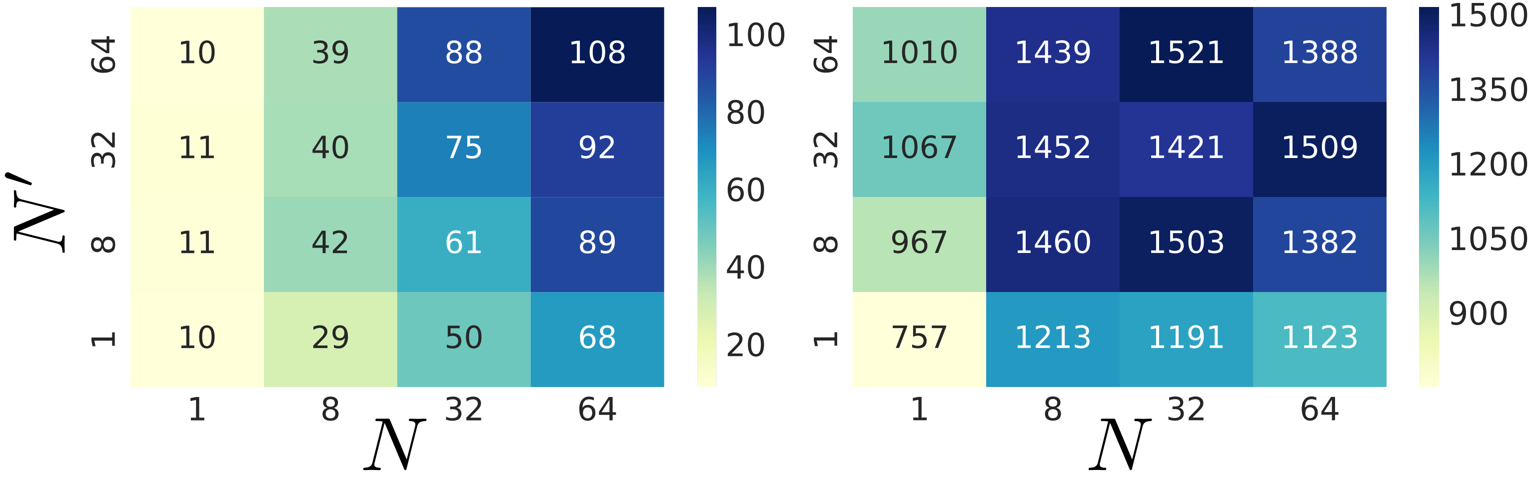


Figure 2. Effect of varying N and N', the number of samples used in the loss function in Equation 3. Figures show human-normalized agent performance, averaged over six Atari games, averaged over first 10M frames of training (left) and last 10M frames of training (right). Corresponding values for baselines: DQN $(32, 253)$ and QR-DQN $(144, 1243)$.

Figure 2 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

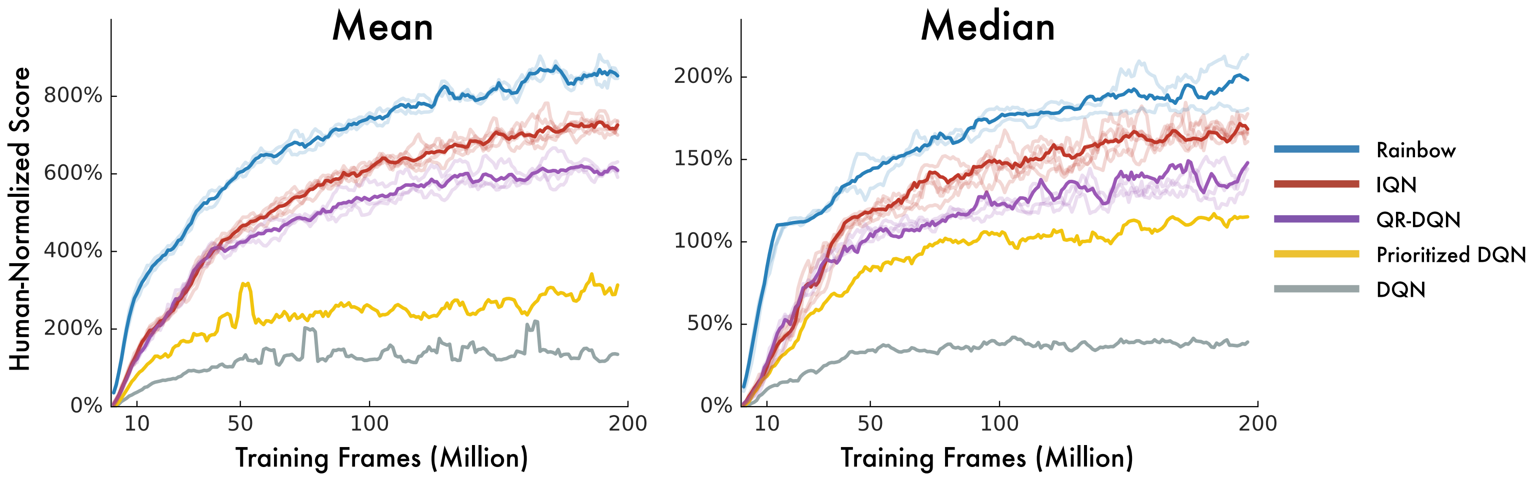


Figure 4. Human-normalized mean (left) and median (right) scores on Atari-57 for IQN and various other algorithms. Random seeds shown as traces, with IQN averaged over 5, QR-DQN over 3, and Rainbow over 2 random seeds.

Figure 4 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

	Mean	Median	Human Gap	Seeds
DQN	228%	79%	0.334	1
PRIOR.	434%	124%	0.178	1
C51	701%	178%	0.152	1
RAINBOW	**1189%**	**230%**	0.144	2
QR-DQN	864%	193%	0.165	3
IQN	1019%	218%	**0.141**	5

Table 1. Mean and median of scores across 57 Atari 2600 games, measured as percentages of human baseline (Nair et al., 2015). Scores are averages over number of seeds.

Table 1 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Human-starts (median)

DQN	PRIOR.	A3C	C51	RAINBOW	IQN
68%	128%	116%	125%	153%	**162%**

Table 2. Median human-normalized scores for human-starts.

Table 2 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

The ablation experiments of the quantile representation. A full grid search with two seeds for every configuration was performed, with the black dots corresponding to the hyperparameters of IQN.

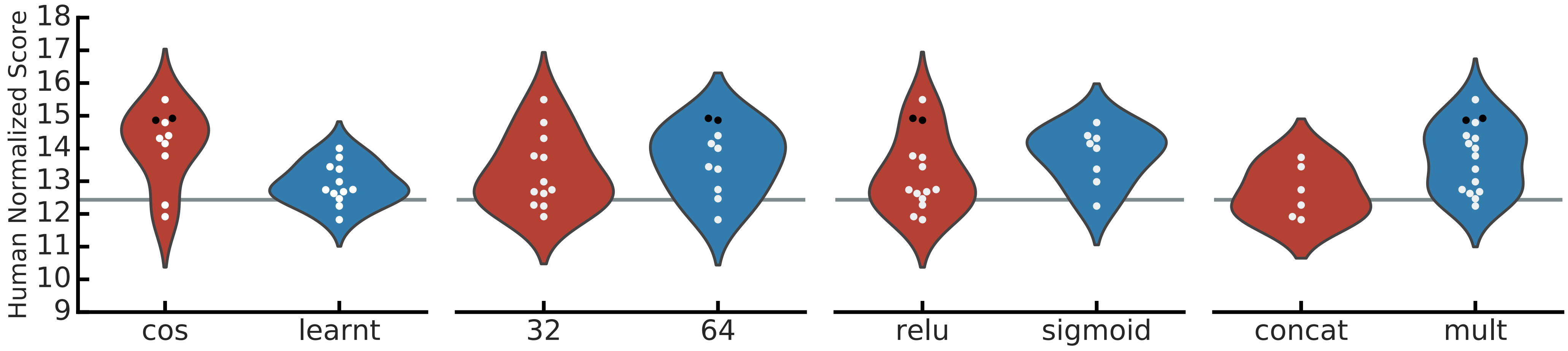


Figure 5. Comparison of architectural variants.

Figure 5 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

- "learn" is a learnt MLP embedding with a single hidden layer of size n;
- "concat" combines the state and quantile representations by concatenation, not $\odot$.