# Quantile Credit Assignment

**Thomas Mesnard** [*1] **Wenqi Chen** [*2] **Alaa Saade** [*1] **Yunhao Tang** [1] **Mark Rowland** [1] **Theophane Weber** [1] **Clare Lyle** [1] **Audrunas Gruslys** [1] **Michal Valko** [1] **Will Dabney** [1] **Georg Ostrovski** [1] **Eric Moulines** [3] **Remi Munos** [1]

## Abstract

In reinforcement learning, the *credit assignment* problem is to distinguish luck from skill, that is, separate the inherent randomness in the environment from the controllable effects of the agent's actions. This paper proposes two novel algorithms, Quantile Credit Assignment (QCA) and Hindsight QCA (HQCA), which incorporate distributional value estimation to perform credit assignment. QCA uses a network that predicts the quantiles of the return distribution, whereas HQCA additionally incorporates information about the future. Both QCA and HQCA have the appealing interpretation of leveraging an estimate of the quantile level of the return (interpreted as the level of "luck") in order to derive a "luck-dependent" baseline for policy gradient methods. We show theoretically that this approach gives an unbiased policy gradient estimator that can yield significant variance reductions over a standard value estimate baseline. QCA and HQCA significantly outperform prior state-of-the-art methods on a range of extremely difficult credit assignment problems.

## 1. Introduction

Credit assignment (Minsky, 1961) is a critical aspect of sequential decision making. On a high level, the central problem of credit assignment is to understand the relationship between actions and outcomes, and equivalently, to determine to what extent an outcome is caused by external factors instead of actions. For example, a player may have won a football game not because of the actions they took, but because the competitors were weak or they happened to score a low-quality shot on goal. Therefore, when attributing credits, it is important to separate "action" from "luck", i.e., external factors that actions cannot control.

Model-free reinforcement learning (RL) algorithms such as policy gradient methods (Williams, 1992; Sutton et al., 1999) use time as a proxy to perform credit assignment, where actions are credited based upon temporal proximity to subsequent rewards. In spite of the simplicity of such a credit assignment mechanism, such methods tend to introduce high variance due to the uncertainty from the environment. As a result, the agent often requires a larger number of samples to learn good policies, i.e., associating actions with desired outcomes in the optimal way.

A number of prior works have sought to improve the credit assignment mechanisms in model free RL for this type of environment. One important line of work proposed to incorporate hindsight information from the future to perform more efficient credit assignment (Andrychowicz et al., 2017; Harutyunyan et al., 2019). Recently, this has entailed efficient algorithms (e..g, (Mesnard et al., 2020)) that significantly improve over baseline methods in environments where it is important to carry out precise credit assignment for the agent to perform well.

A ubiquitous and indispensable notion to model-free credit assignment is *random return*, i.e., the cumulative sum of reward received by the agent. In this work, we investigate *Quantile Credit Assignment* (QCA), an credit assignment approach that fully exploits the rich structure of random returns in a generic way. QCA leverages the full distribution of the random return, to formalize the notion of "luck" in credit assignment. By performing credit assignment with QCA, model-free agents can disentangle the effect of actions from random external factors more efficiently, leading to much faster policy improvement. Furthermore, we show that QCA can also be flexibly combined with other credit assignment methods, such as incorporating hindsight information. Overall, QCA greatly improves data efficiency in high-variance environments and enable the agents to have robust performance.

Our main contributions are as follows: (1) We formalize the notion of luck with distributions of random return, and how this relates to theoretically grounded variance reduction of

[*]Equal contribution [1]DeepMind [2]Harvard University [3]Ecole polytechnique. Correspondence to: Thomas Mesnard <mesnard@deepmind.com>.

*Proceedings of the $40^{th}$ International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. 

policy gradient (PG) estimator (Section 3); (2) We propose a scalable implementation of QCA in model-free agents (Section 4); (3) We demonstrate how QCA can combine with orthogonal credit assignment mechanisms such as hindsight information, leading to hindsight QCA (HQCA, Section 5); (4) Finally, we show that QCA and HQCA improves over prior approaches in benchmark tasks (Section 6).

## 2. Background

We use capital letters for random variables and lower-case for the value they may take. Consider a generic MDP $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$. At each time step $t$, given a current state $X_t \in \mathcal{X}$ and selected action $A_t \in \mathcal{A}$, the agent receives a reward $R_t = R(X_t, A_t)$ and makes a transition to the next state $X_{t+1} \sim P(\cdot|X_t, A_t)$. Without loss of generality, we assume a fixed initial state $X_0 = x_0 \in \mathcal{X}$. In the case of a partially observed environment, we assume the agent receives an observation $O_t$ at every time step, and simply define state $X_t$ to be the history of previous observations and actions $X_t = (A_{s-1}, R_{s-1}, O_s)_{s \le t}$.

Starting from state action pair $X_t = x, A_t = a$ and following policy $\pi$, the agent receives a cumulative sum of reward (also called the return) $Z^\pi_{x,a} \coloneqq \sum_{s=t}^{\infty} \gamma^{s-t} R_s$. Similarly, we define the return distribution $\eta^\pi_x$ corresponding to the return obtained when following $\pi$ from state $x$. In general, the return is a random variable and we define its distribution as $\eta^\pi_{x,a}$. The value function and Q-function are the expectations of these random returns: $Q^\pi(x,a) = \mathbb{E}_{Z^\pi_{x,a} \sim \eta^\pi_{x,a}}\left[Z^\pi_{x,a}\right]$ and $V^\pi(x) = \mathbb{E}_{Z^\pi_x \sim \eta^\pi_x}\left[Z^\pi_x\right]$.

In general, the agent acts according to a stochastic policy $\pi_\theta(\cdot|X_t)$ where $\theta$ are policy parameters.

**Notation.** In what follows, whenever clear from context, we will omit the explicit dependence on $\pi$ to simplify notation, writing $\eta_{x,a}, \eta_x, Z_{x,a}, Z_x, Q, V$ instead of $\eta^\pi_{x,a}, \eta^\pi_x, Z^\pi_{x,a}, Z^\pi_x, Q^\pi, V^\pi$.

### 2.1. Policy Gradient Estimators and Baselines

We begin by recalling the form of policy gradient (PG) algorithms and the intuition behind their credit assignment mechanisms. As a baseline, consider a form of commonly used policy gradient estimator (Williams, 1992; Sutton et al., 1999), which we will also call the single-action policy gradient estimator. Given a trajectory $(X_t, A_t, R_t)_{t=0}^\infty$ generated under $\pi_\theta$, the policy gradient estimator is defined as follows:

$$\sum_{t \ge 0} \gamma^t \nabla_\theta \log \pi_\theta(A_t|X_t) \left(Z_{X_t,A_t} - V_\psi(X_t)\right), \quad (1)$$

where $V_\psi(x)$ is a state-dependent baseline function. It has been shown that the above estimator is an unbiased estimator of the gradient of value function $\nabla_\theta V^\pi(x_0)$ (Williams, 1992). The baseline function is usually trained to be an approximation to the value function $V_\psi(X_t) \approx V(X_t)$, such that it provides an average estimate of the random return $Z_{X_t}$ from $X_t$ when following policy $\pi$. As a result, the difference $Z_{X_t,A_t} - V_\psi(X_t)$ is an estimation to the advantage function $Q(X_t, A_t) - V(X_t)$, which helps identify directions in which the policy can improve.

## 3. Quantile Credit Assignment PG Estimators

The single-action policy gradient estimator (Equation (1)) uses $Z_{X_t,A_t} - V_\psi(X_t)$ to measure the relative advantage of taking action $A_t$ at state $X_t$ compared to other actions. However, when the random return $Z_{X_t,A_t}$ contains high variance (such as the level of "luck", capturing the intrinsic randomness of the environment as well as the randomness coming from the agent's own future actions), the policy gradient estimator can easily mistake the sign of the advantage estimate, resulting in highly sub-optimal credit assignment. Intuitively, we'd like to remove the amount of 'luck' contained in the random return $Z_{X_t,A_t}$ in order to identify the contribution that the individual action $A_t$ had on that return. For that purpose we design a policy gradient estimator that identifies the luck level $\tau \in [0,1]$ ($\tau$ is defined as the quantile level) of the random variable return $Z_{X_t,A_t}$ and assigns credit to the action $A_t$ chosen in $X_t$ at this level of randomness, by subtracting in the policy gradient estimator a baseline function $Q(x, \pi, \tau)$ that depend on this levels of luck. This defines the *quantile credit assignment* (QCA) policy gradient estimator and the corresponding QCA baseline is described next.

**QCA Baseline.** For any given policy $\pi_\theta$, recall that $\eta_{x,a}$ is the random return distribution at $(x,a)$. We define $F_{\eta_{x,a}} : \mathbb{R} \to [0,1]$ as its CDF and let

$$Q(x,a,\tau) \coloneqq F^{-1}_{\eta_{x,a}}(\tau) \quad (2)$$

be the inverse CDF evaluated at quantile level $\tau \in [0,1]$ ($Q(x,a,\tau)$ is also called the quantile function). Sampling from $\eta_{x,a}$ is equivalent to generating uniformly $\tau \sim U(0,1)$ and pushing it through the quantile function $Q(x,a,\tau)$. Formally, let $Z_{x,a} \sim \eta_{x,a}$ be a sample of random return, we have

$$Z_{x,a} =_{\mathcal{D}} Q(x,a,\tau), \quad \tau \sim U(0,1)$$

where $=_{\mathcal{D}}$ denotes equality in distribution. Our key insight is to identify the quantile level $\tau$ as the luck level in generating the random return $Z$. When $\tau$ is small (corresponding to an unlucky situation), the return is small; when $\tau$ is large (a lucky situation), the return is high. Based on $Q(x,a,\tau)$, we define the QCA baseline

$$Q(x,\pi,\tau) \coloneqq \sum_a \pi(a|x) Q(x,a,\tau). \quad (3)$$

For any given quantile (or luck) level $\tau \in [0,1]$, the QCA baseline takes an average of $Q(x,a,\tau)$ over actions under policy $\pi$. By taking the average over actions, the QCA baseline $Q(x,\pi,\tau)$ removes the randomness due to sampling immediate actions $a \sim \pi(\cdot|x)$ and retains the other sources of randomness captured in $\tau$. Given a random return $Z_{X_t,A_t}$, we can understand it as being generated by certain luck level $\widehat{\tau}_t$ via $Z_{X_t,A_t} = Q(X_t,A_t,\widehat{\tau}_t)$. Assume that we can identify the luck level $\widehat{\tau}_t$, a natural advantage estimate would be the difference between the random return and the QCA baseline evaluated at the same luck level $\widehat{\tau}_t$. This gives the QCA advantage estimate

$$Z_{X_t,A_t} - Q(X_t,\pi,\widehat{\tau}_t) = \\ Z_{X_t,A_t} - \sum_a \pi(a|x) F^{-1}_{\eta_{X_t,a}}(F_{\eta_{X_t,A_t}}(Z_{X_t,A_t})),$$

(which is analogous to $Z_{X_t,A_t} - V(X_t)$ in the usual advantage estimator). Finally, we define the QCA PG estimator

$$\sum_{t\geq 0} \gamma^t \nabla_\theta \log \pi_\theta(A_t|X_t) \left(Z_{X_t,A_t} - Q(X_t,\pi,\widehat{\tau}_t)\right). \quad (4)$$

Compared to the single-action PG estimator in Equation (1), the QCA PG estimator applies a baseline $Q(X_t,\pi,\widehat{\tau}_t)$ that depends both on the state $x$ and the level of luck $\widehat{\tau}_t$ inferred from the random return $Z_{X_t,A_t}$. Intuitively, the QCA advantage estimate $Z_{X_t,A_t} - Q(X_t,\pi,\widehat{\tau}_t)$ represents the advantage of having chosen action $A_t$ instead of a random action drawn from $\pi$ in $X_t$, for the same amount of luck (i.e., quantile level $\widehat{\tau}_t$) as in the observed return $Z_{X_t,A_t}$. An equivalent yet alternative view is that by correlating the random return $Z_{X_t,A_t} = Q(X_t,A_t,\widehat{\tau}_t)$ and the QCA baseline $Q(X_t,\pi,\widehat{\tau}_t)$ by the common luck level $\widehat{\tau}_t$, the QCA PG baseline achieves provable variance reduction compared to the regular PG estimator, as we will formally show below.

**Discussion about alternative baselines.** In addition to the QCA baseline $Q(X_t,\pi,\widehat{\tau}_t)$, an alternative approach is to define the baseline as $V(X_t,\widehat{\tau}_t)$ where $V(x,\tau) \coloneqq F^{-1}_{\eta_x}(\tau)$ is the quantile function for the return distribution $\eta_x$ from state $x$. We refer to this as the value QCA (VQCA) baseline. A conceptual drawback of the VQCA baseline $V(x,\tau)$ is that since the return distribution $\eta_x$ also contains randomness in the action sampling $a \sim \pi(\cdot|x)$, its quantile level might differ significantly from the quantile levels of the QCA baseline. As a simple example, when the returns from $\eta_{x,a}$ with fixed $(x,a)$ are deterministic, $Q(x,a,\tau)$ is constant for all $\tau \in [0,1]$. However, the return distribution $\eta_x$ can still have non-zero variance due to stochasticity in choosing actions. Since now the quantile level $\tau$ reflects the randomness in the choice of actions instead of the external randomness, we do not advise using $V(x,\tau)$ as a baseline in policy gradient estimators; we provide additional theoretical results regarding VQCA in Appendix B, including an example in which it increases variance relative to a standard value baseline.

### 3.1. Theoretical Analysis

We now provide several key statistical properties that establish QCA as a principled method for credit assignment, and also give intuition as to when we should expect the benefits of QCA to particularly be pronounced. Our first result proves the unbiasedness of the QCA policy gradient estimator.

**Proposition 3.1.** The QCA baseline results in unbiased policy gradient estimators when using exact return quantile functions $Q^\pi$, as shown in A. That is, with $Z_{X_t,A_t} = Q^\pi(X_t,A_t,\widehat{\tau}_t)$, we have

$$\nabla_\theta V^\pi(x_0) = \mathbb{E}\Bigg[\sum_t \gamma^t \nabla_\theta \log \pi_\theta(A_t|X_t) \\ \left(Z_{X_t,A_t} - Q^\pi(X_t,\pi,\widehat{\tau}_t)\right)\Bigg].$$

Next, we establish that the component of the QCA that estimates the advantage is never worse than the classical state-value function baseline, as measured by variance, and is generally strictly better as shown in A. Traditionally, this has motivated a number of improved baseline functions for policy gradient estimators (see, e.g., (Gu et al., 2016; Liu et al., 2017; Wu et al., 2018; Grathwohl et al., 2017)). Though this does not always guarantee that the full gradient estimator has smaller variance, it is often the case as empirically validated in aforementioned prior work and in our experiments.

**Proposition 3.2.** The QCA baseline provides an advantage estimate which has no greater variance than that associated with the value baseline when using exact quantile functions $Q^\pi$. More precisely, considering a random return $Z_{x,A}$ generated from a state-action pair $(x,A)$, with $A \sim \pi(\cdot|x)$, and writing $Z_{x,A} = Q^\pi(x,A,\widehat{\tau})$ we have

$$\mathrm{Var}\left(Z_{x,A} - Q^\pi(x,\pi,\widehat{\tau})\right) = \mathrm{Var}\left(Z_{x,A} - V^\pi(x)\right) - \\ \mathbb{E}_{\tau'\sim\mathcal{U}([0,1])}\left[\left(Q^\pi(x,\pi,\tau') - V^\pi(x)\right)^2\right].$$

Proposition 3.2 tells us that whenever there is an action $a$ with positive probability $\pi(a|x) > 0$, and a corresponding non-deterministic distribution over returns, we see benefits from the QCA baseline. Further, we should expect large variance reduction when using the QCA baseline (compared to the classical value baseline) precisely when there is high variance in the distribution over returns. With the unbiased property established in Proposition 3.1, it can be guaranteed that quantiles can be used in the baseline of policy gradient without biasing the agent; we note that the assumption that the quantile function is exact is crucial to the unbiasedness property established in Proposition 3.1, in contrast to classical state-based baselines in policy gradients, which are guaranteed to be unbiased even when the value function is

inexact. In summary, while the idea of learning quantiles of the distribution of the return is not new, the great novelty here is that we use it to define a baseline for policy gradient, which further decreases the variance.

## 4. Implementing QCA

In this section, we introduce the core architectural and algorithmic components of the QCA algorithm.

### 4.1. Learning the Quantile Function

Central to the QCA baseline is the quantile function $Q(x, a, \tau)$. In practice, since we do not have access to the ground truth quantile function, we parameterize a quantile network $Q_\psi(x, a, \tau) \approx Q(x, a, \tau)$ as an approximation. The network outputs $m$ quantile predictions $Q(x, a, \tau_i)$ with $\tau_i = \frac{2i-1}{2m}$. The $i$-th quantile prediction is trained using the quantile regression loss (Koenker & Bassett Jr, 1978),

$$\tau_i \left(Z_{x,a} - Q(x, a, \tau_i)\right)_+ + (1 - \tau_i) \left(Z_{x,a} - Q(x, a, \tau_i)\right)_- \tag{5}$$

where $Z_{x,a}$ is a random return generated at $(x, a)$ under policy $\pi$. By minimizing the above loss function, $Q_\psi(x, a, \tau_i)$ is guaranteed to form a close approximation to the true quantile function. Intuitively, the more quantile level $m$ we use, the more accurate the approximation is (see, e.g., (Bellemare et al., 2023) for characterizations of the approximation error). Since learning accurate quantile function is of major significance to QCA, we propose two architectural and algorithmic improvements on top of the vanilla quantile network (Dabney et al., 2018), this includes (1) a parameterization that ensures quantile predictions are monotonic $Q_\psi(x, a, \tau_i) \leq Q_\psi(x, a, \tau_{i+1})$, which introduces a useful inductive bias for learning quantiles in general; (2) a novel combination of dueling architecture (Wang et al., 2016) with quantile network, which accelerates learning quantiles through the shared parameterization. Due to space limits, we introduce details in Appendix C.

### 4.2. Finding the Quantile Level

Recall that in order to define the QCA baseline, we need to identify the quantile level $\widehat{\tau}_t$ for return $Z_{X_t,A_t}$ by solving the equality $Q(X_t, A_t, \widehat{\tau}_t) = Z_{X_t,A_t}$. With the quantile predictions, a challenge with solving the plug-in equality $Q_\psi(X_t, A_t, \widehat{\tau}_t) = Z_{X_t,A_t}$ is that there is a finite number $m$ of predicted quantiles, there might not exist a solution with $\widehat{\tau}_t \in \{\tau_i, 1 \leq i \leq m\}$.

To remedy the above issue, given the set of quantile predictions $Q_\psi(x, a, \tau_i)$ output by the network and given a return sample $Z_{X_t,A_t}$ we select the quantile level $\widehat{\tau}_t$ such that $Z_{X_t,A_t} = Q_\psi(X_t, A_t, \widehat{\tau}_t)$. This is done by considering that our quantile estimate $Q_\psi(X_t, A_t, \tau)$ interpolates piecewise-linearly the quantiles $Q_\psi(X_t, A_t, \tau_i)$ and $Q_\psi(X_t, A_t, \tau_{i+1})$ within each interval $\tau \in [\tau_i, \tau_{i+1}]$, for $1 \leq i < m$. Thus for the index $I_t$ such that $Z_{X_t,A_t} \in [Q_\psi(X_t, A_t, \tau_{I_t}), Q_\psi(X_t, A_t, \tau_{I_t+1})]$, we define

$$\widehat{\tau}_t = (1 - \alpha)\tau_{I_t} + \alpha\tau_{I_t+1} \tag{6}$$
$$\text{with } \alpha = \frac{Z_{X_t,A_t} - Q_\psi(X_t, A_t, \tau_{I_t})}{Q_\psi(X_t, A_t, \tau_{I_t+1}) - Q_\psi(X_t, A_t, \tau_{I_t})}.$$

In the specific extreme cases where $Z_{X_t,A_t} < Q_\psi(X_t, A_t, \tau_1)$ (or $Z_{X_t,A_t} > Q_\psi(X_t, A_t, \tau_m)$) we select the extreme quantile levels: $\widehat{\tau}_t = \tau_1$ (resp. $\widehat{\tau}_t = \tau_m$). By doing this, we ensure that a meaningful quantile level $\widehat{\tau}_t$ can be recovered from the learned quantile function, despite a finite quantile approximation to the full quantile function.

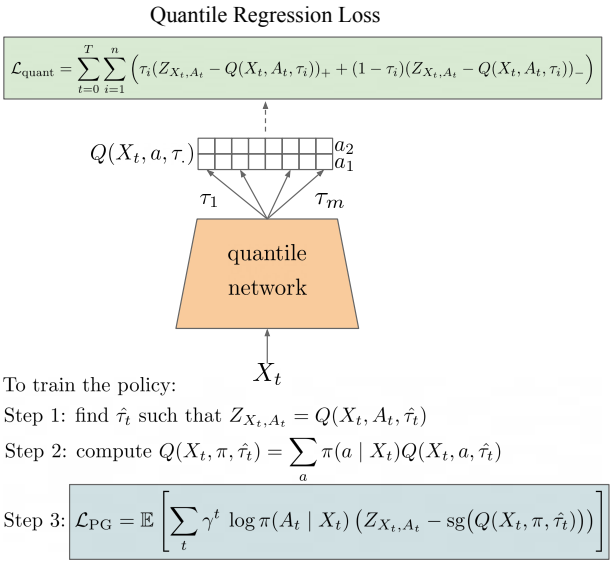



*Figure 1.* Architecture and pseudocode of the QCA algorithm. QCA trains the quantile function $Q_\psi(x, a, \tau_i)$ with the quantile regression loss; then uses the quantile function to construct QCA baseline and QCA PG estimator.

### 4.3. The QCA Algorithm

Putting all elements together, we describe the full-fledged QCA algorithm implementation. Throughout, the agent follows the policy network $\pi_\theta$. At time $t$, from state $X_t$, the agent takes action $A_t \sim \pi_\theta(\cdot|X_t)$ and observes return $Z_{X_t,A_t}$ along the trajectory thereafter,

- Train the quantile network $Q_\psi(X_t, A_t, \tau_i)$ (for all $1 \leq i \leq m$) using the quantile regression loss (Eqn 5);
- Identify $\widehat{\tau}_t$ such that $z_{X_t,A_t} = Q_\psi(X_t, A_t, \widehat{\tau}_t)$ using the interpolation strategy in Eqn 6;
- Compute the QCA baseline $Q(X_t, \pi, \widehat{\tau}_t)$ using Eqn 3;

- Update the policy network by the QCA PG estimator

$$\nabla \log \pi_\theta(A_t|X_t) \left(Z_{X_t,A_t} - Q_\psi(X_t, \pi, \widehat{\tau}_t)\right). \quad (7)$$

## 5. Hindsight QCA

The basic QCA algorithm infers the quantile level $\widehat{\tau}_t$ from the information about the return $Z_{X_t,A_t}$ only. However, it could be the case that information collected after action $A_t$ has been chosen in state $X_t$ can give additional information about the "luck" level of the return. Consider as an example the situation where one has to drive home from work and there are several possible routes. However the length of the travel (the return $Z_{X_t,A_t}$) may be affected by the weather (impacting the traffic globally) for any route. So observing the weather may directly inform the agent about the level of luck $\widehat{\tau}_t$ without having to learn the quantile function from the returns only. One should be able to generalize the identification of the luck $\widehat{\tau}_t$ level by leveraging information observed along the trajectory $(X_s, A_s, R_s)_{s \geq t}$ instead of relying on the return only.

Motivated by the above example, we introduce Hindsight QCA (HQCA) as a generalization of QCA which uses hindsight information.

### 5.1. Finding the Hindsight Quantile Level

Inspired by prior work on counterfactual credit assignment (CCA; Mesnard et al., 2020)), we let $\phi_t$ be a feature vector that summarizes future (hindsight) information collected along the trajectory (observations, actions and rewards $(X_s, A_s, R_s)_{s>t}$) after time $t$. As a concrete example to represent $\phi_t$, , we can compute the feature using a backward RNN. Note that the return $Z_{X_t,A_t} = \sum_{s \geq t} \gamma^{s-t} R_s$ is a special case of hindsight information that can be captured in $\phi_t$.

HQCA makes use of an additional network (called the hindsight $\tau$-network) $P(\tau|X_t, A_t, \phi_t)$ from which we predict the quantile level $\widehat{\tau}_t$ of the return $Z_{X_t,A_t}$ from the state of the agent $X_t$, the selected action $A_t$ and the feature $\phi_t$. We now describe how to train the network $P(\tau|X_t, A_t, \phi_t)$ such that it can accurately identify the quantile level based on hindsight features $\phi_t$.

**Training the hindsight $\tau$-network.** In order to train the hindsight $\tau$-network one could think of concatenating the $\tau$ network and the quantile network using a variational auto-encoder approach where the encoder (the hindsight $\tau$-network) would produce a distribution over the quantile levels $\tau$ (the latent variable), which injected into the decoder (the quantile network) would output the quantiles $Q(x, a, \tau)$, and both networks would be train by regressed these quantiles toward the observed returns. However this training would not produce a latent representation (the quantile level $\tau$) that is independent of the action $A_t$, given $X_t$ (since the hindsight feature $\phi_t$ may reveal information about this action), thus possibly biasing the PG estimate.

Instead, we use two separate losses to train these networks. The quantile network is still trained using quantile regression like in the previous section. And the hindsight $\tau$-network $P(\cdot|X_t, A_t, \phi_t)$ is trained (using cross entropy) to predict the quantile level $\hat{\tau}_t$ estimated by the quantile network using Eqn 6. Since the hindsight feature $\phi_t$ is injected as input to the hindsight $\tau$-network, the corresponding recurrent network $\phi_t = \Phi((X_s, A_s, R_s)_{s>t})$ is trained using the same loss as the hindsight $\tau$-network.

Once all networks have been learned perfectly, we have the property that the quantile level $\hat{\tau}_t$ output by the hindsight $\tau$-network is independent of the action $A_t$ selected in $X_t$.

### 5.2. The Hindsight QCA Algorithm

The Hindsight QCA algorithm simply consists in selecting the quantile level $\hat{\tau}_t$ as the output of the hindsight $\tau$-network instead of using the one defined by the quantile network. Once $\hat{\tau}_t$ has been selected, everything else is the same as in QCA: we compute the QCA baseline $Q(X_t, \pi, \widehat{\tau}_t)$ using Eqn 3 and improve the policy network by following the PG estimate using Eqn 7.

The benefit of this approach is that the hindsight $\tau$-network has access to information (through $\phi_t$) about the future observations $(X_s, A_s)_{s>t}$ (which is not the case of the quantile network) in addition to the return. This information can be useful to better identify the 'luck level' $\hat{\tau}_t$ because the mapping from the full trajectory to the luck level may generalize better than the same prediction from the return only. For illustration, in the example mentioned above, we expect that the hindsight feature will learn to pay attention to the weather and that this feature will be leveraged by the hindsight $\tau$-network to predict a specific luck level as a function of the observed weather regardless of the route chosen.

**Putting it all together.** The training process of Hindsight QCA is the following. At time $t$, from state $X_t$, the agent takes action $A_t \sim \pi_\theta(\cdot|X_t)$ and observes a trajectory $(X_s, A_s, R_s)_{s \geq t}$,

- Train the quantile network $Q_\psi(X_t, A_t, \tau_i)$ (for all $1 \leq i \leq m$) using the quantile regression loss (Eqn 5),
- Define the target distribution $\hat{\eta}_t = (1 - \alpha)\delta_{\tau_{I_t}} + \alpha\delta_{\tau_{I_t+1}}$, where $\alpha$ and $I_t$ are defined as in Eqn 6,
- Train the hindsight $\tau$-network by minimizing the cross entropy loss between $\hat{\eta}_t$ and $P(\tau|X_t, A_t, \phi_t)$, whose gradient further propagates through the hindsight network $\phi_t = \Phi((X_s, A_s, R_s)_{s \geq t})$ for training,

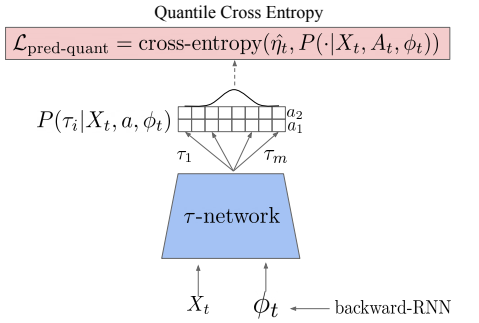
- Identify the hindsight quantile level

$$\hat{\tau}_t = \underset{\{\tau_i\}_{1\leq i\leq m}}{\arg\max}\ P(\tau_i|X_t, A_t, \phi_t),$$
$$\text{or } \hat{\tau}_t = \sum_{i=1}^{m} \tau_i P(\tau_i|X_t, A_t, \phi_t),$$

- Compute the QCA baseline $Q(X_t, \pi, \hat{\tau}_t)$ using Eqn 3,
- Update the policy network by the QCA PG estimator using Eqn 7.

Notice that we describe two ways to define the quantile level $\hat{\tau}_t$ from the hindsight $\tau$-network $P(\tau_i|x, a, \phi)$. Experimentally these two methods give very similar performances.

**Comparison to CCA (Mesnard et al., 2020).** HQCA shares similarities with the Counterfactual Credit Assignment (CCA) algorithm in that hindsight information from the future is captured by a feature $\phi_t$ and used in a policy gradient algorithm. In CCA, $\phi_t$ is learnt to predict the future return while enforcing the property of conditional independence with the action $A_t$ given $X_t$. This property is required to avoid biasing the PG estimate. In HQCA, we do not enforce this independence property between $\phi_t$ and $A_t$ (and in general we expect $\phi_t$ and $A_t$ to be dependent). However, because we use quantile regression to train the quantile network, the target quantile levels used to train the hindsight $\tau$-network are uniformly distributed and independent of $X_t, A_t$. This enforces the property that the hindsight quantile level $\hat{\tau}_t$ is independent of $A_t$ given $X_t$ (it is actually independent of $A_t$ and $X_t$), which guarantees unbiasedness of the PG estimate once the networks have been trained.




To train the policy:

Step 1: Define $\hat{\tau}_t = \text{argmax}_\tau(P(\tau|X_t, A_t, \phi_t))$

Step 2 and 3: Similar to QCA

*Figure 2.* Architecture and pseudocode of the HQCA algorithm. On top of QCA, HQCA uses a backward RNN to compute the hindsight feature $\phi_t$ that can be used by the hindsight $\tau$-network $P$ to identify the quantile level. This is then used for computing the baseline and PG estimators.

# 6. Experiments

In order to validate the hypothesis that QCA and HQCA perform well in environments where disentangling "skill" from "luck" is difficult, we investigate the performances of the proposed algorithms in three high variances environments that are described below. In parallel, we also run three strong baselines: (i) a straightforward policy gradient with baseline (PG); (ii) CCA, to see how well a previous state-of-the-art credit assignment method performs; and (iii) a PG agent with a distributional critic (as in QCA), but using only the *mean* estimated by the critic as a baseline, to disentangle improvements to credit assignment in (H)QCA from improvements in representation learning that are often observed with distributional critics (Barth-Maron et al., 2018; Hoffman et al., 2020; Duan et al., 2021; Nam et al., 2021; Shahriari et al., 2022). All results are reported as median performances over 20 seeds with interquartile range represented by a shaded area. Note that the same amount of time or less was spent to tuned QCA and HQCA in comparison to the time spent to tune the baselines.

## 6.1. High-Variance Key-To-Door

First, we propose to look at a new version of the Key-To-Door family, initially proposed by Hung et al. (2019), as a testbed for credit assignment in noisy environments. In this partially observable grid-world (Figure 7), the agent has to pick up a key in the first room for which it gets no immediate reward. In a second room, the agent can pick up 10 apples that each give an immediate reward. This is what we call the distraction phase. In the final room, the agent may open a door only if it is carrying a key, and receive a small reward for doing so. In this task, a single and early action (i.e picking up the key) impacts the reward it receives at the end of the episode. This signal is hard to detect as the episode return is largely driven by the agent's performance at picking up apples in the second room.

High-variance Key-to-door (HVKTD) keeps the overall structure of the Key-To-Door task proposed by Mesnard et al. (2020), however the reward for each apple is randomly sampled from the distribution of Uniform$\{-8, 8.2\}$. In this setting, even an agent that is skilled at picking up apples observes a large variance in its episode returns which makes the learning signal for picking up the key and opening the door weak and noisy. A perfectly trained agent will be able to get a total return of 2, half of this return coming from picking up the apples and half coming from opening the door.

**Results.** As shown in Figure 3, the PG algorithm is unable to learn to pick up the key and open the door. Distributional RL itself allows some learning progress to be made, and leveraging these quantiles to do credit assignment results in a strong improvement as shown by the QCA results. Fi-

nally, using hindsight information to inform the quantile level enables HQCA to match CCA performances which was specifically designed for this environment. Note first that the variance between seeds for HQCA is much smaller than the one for CCA. Furthermore, HQCA is capable of matching the performance of CCA with relatively little hyperparameter tuning. In our experience, we found QCA and HQCA are more robust and less finicky to tune than CCA.

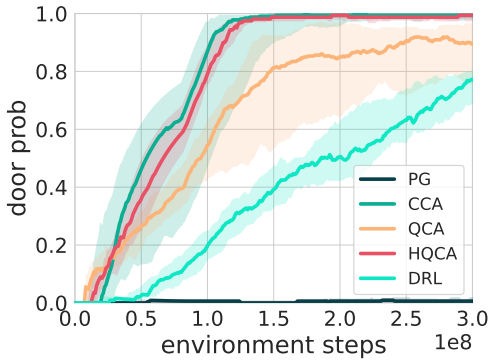



*Figure 3.* Results for High-Variance Key-To-Door.

### 6.2. Random Key-To-Door

We now consider a second variation of Key-To-Door. Instead of having immediate rewards from picking up apples during the distraction phase, Random Key-To-Door (RKTD) provides random rewards to the agent sampled from the distribution $N(0, 1)$ for each time step in this phase. The agent always receives noisy rewards in this phase as the noise level is now independent of the agent's behavior.

**Results.** RKTD is a very challenging task both for policy gradient and CCA as they are not able to not learn to open the door even after $3e6$ environment steps Figure 4 (only $1.5e6$ shown here). On the contrary, DRL, QCA and HQCA solve the environment rapidly and reliably. The quantile loss used in DRL, QCA, and HQCA greatly helps performance. Once again, leveraging these quantiles to do credit assignment improves data efficiency in particular when using hindsight, and we find that QCA and HQCA perform robustly, leading to efficient learning in highly noisy environments. One explanation why QCA and HQCA outperform CCA is that QCA approaches only need to reconstruct the quantile function of a Gaussian while CCA needs to learn to reconstruct the return.

### 6.3. Combinatorial RL with Post-Decision Noise Feedback

Finally, we consider a task where, in a first "query room", the agent is faced with two colored squares. When the agent picks up a colored square, it immediately receives the reward $R = r + \sigma$ where $\sigma \sim \text{Uniform}[0, 5]$ and $r = 1$ if the agent has picked the "good" object and $r = 0$ otherwise. The agent is then teleported to a second room, the "answer room", where it can observe another colored square whose color is deterministically mapped to the noise level $\sigma$ that has been sampled for the computation of $R$.

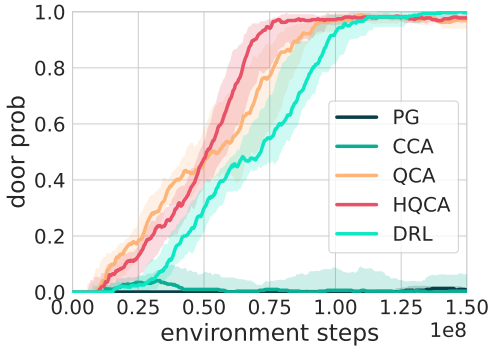



*Figure 4.* Results for Random Key-To-Door.

The task consists in a succession of these two rooms, with randomly sampled pairs of colored squares for the "query" room and a new noise level sampled each time. Note that the sampling makes sure that the agent is always faced with one good and one bad colored square. Finally, the color mapping of "good" and "bad" objects is kept fixed through training. Performances are reported as the number of rooms where the agent has picked the "good" colored object. Note that a small exploration bonus is given to the agent when it picks up any colored square to make the exploration problem simpler. A visual description of the task can be seen in Figure 5.

This task can be abstracted as a contextual bandit problem where with a large number of contexts (i.e the pairs of colored squares) and two actions (i.e left or right), though the agent is of course not *a priori* aware of this structure. However, the reward has a specific form. It is the sum of a deterministic part corresponding to the action taken and a stochastic part independent of the action. The dependency between the good action and the context is unstructured (no regularity). After the agent has taken its action, the stochastic part of the reward is revealed in a visual way. Classical methods such as plain policy gradient are not expected to perform well here because they cannot efficiently learn the reward structure, as it lacks access to hindsight information. It will have to use many samples to distinguish the deterministic from the stochastic part for each state and action. However, as HQCA (and CCA) has access to hindsight information, it should be able to leverage this to learn the reward structure of the task rapidly. Indeed, when given the stochastic part of the reward, inferring the correct action is trivial.

**Results.** As shown in Figure 6, HQCA and CCA both solve the task thanks to their use of hindsight information. They get close to the optimal score (which is 9) because a very

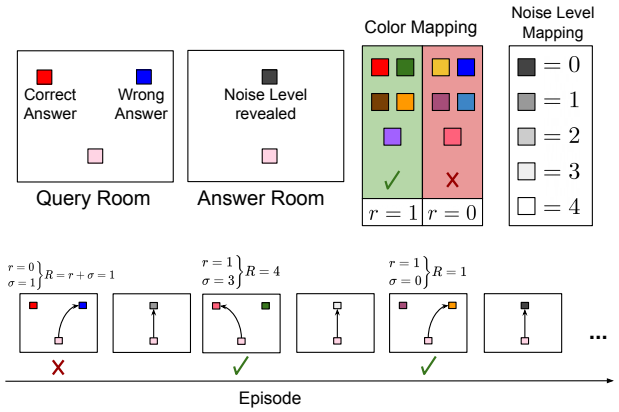



*Figure 5.* Description of the "Combinatorial RL with post-decision noise feedback" task.

short timing to visit all the rooms before the environment times out. On the contrary, PG and DRL find a local optimal which consists of simply picking up any square, regardless of their color, to get the small exploration bonus for all query rooms. As they pick the "good" object with 50% chance, they get a score of 4.5 out of 9 query rooms that can be visited in an episode. Finally, QCA seems to perform slightly better than PG and DRL but still does not solve the tasks reliably as it also lacks access to hindsight information to inform its "level".

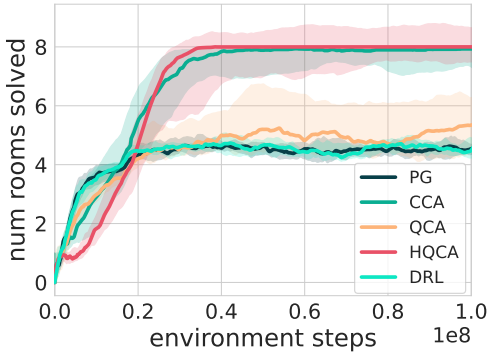



*Figure 6.* Results for the "Combinatorial RL with post-decision noise feedback" task.

## 7. Related Work

There are two tracks of motivations for our paper based on the previous work. On the one hand, there is past research that sheds light on proposing a better baseline through incorporating more information from the future to decrease the variance. Counterfactual Credit Assignment (CCA) Mesnard et al. (2020) leverages *hindsight* information to implicitly perform counterfactual evaluation-an estimate of the return for actions other than the ones which were chosen. The counterfactual reasoning will enable the agent to reason about what would have happened had different actions been taken *with everything else remaining the same*. In this way, it can form unbiased and lower variance estimates of the policy gradient by building future-conditional baselines. However, in order to make the baseline independent with actions to prevent biases, CCA has to introduce additional action-removal loss to force the information from actions can be disentangled from baseline. As a result, the algorithm will be unintentionally unstable and will lack interpretability. On the contrary, QCA does not include extra losses during training and will be easily interpreted with the estimation of quantiles.

On the other hand, our work also follows the research that focuses on distributional reinforcement learning in which the distribution over returns is modeled explicitly instead of estimating the mean, which is to examine methods of learning the return distribution instead of value function. Unlike traditional value-based reinforcement learning algorithms like DQN (Mnih et al., 2015) average over randomness to estimate the value, distributional reinforcement learning methods model this distribution over returns explicitly instead of only estimating the mean. This can lead to more insights and knowledge for the agent with a much faster and more stable learning. In Categorical DQN (C51; Bellemare et al., 2017), the possible returns are limited to a discrete set of fixed values (51), and the probability of each value is learned through interacting with environments. Based on it, QR-DQN computes the return quantiles on fixed , uniform quantile fractions using quantile regression and minimizes the quantile Huber loss between the Bellman updated distribution and current return distribution. However, the past research on distributional RL focuses more on representation learning. QCA instead follows the track of policy gradient and innovatively uses the quantiles directly as baseline, which will lower the variance in a straightforward way.

## 8. Conclusion

In this paper we introduced an approach based on quantile estimation to improve credit assignment in RL by disentangling "luck" from "skill". QCA builds an estimate of the quantile function of the return and HQCA additionally estimates the quantile level (interpreted as the level of "luck") from a full trajectory. These methods produce a "luck-dependent" baseline for policy gradient methods, which does not introduce bias and potentially significantly reduce the variance of the PG estimate (compared to a standard value estimate baseline). Experimentally, QCA and HQCA significantly outperform prior state-of-the-art methods on a range of difficult credit assignment problems.

Future research will investigate the performance of the algorithms and how to scale them in more complex environments which are closer to real-world problems.

## References


Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Tb, D., Muldal, A., Heess, N., and Lillicrap, T. Distributed distributional deterministic policy gradients. In *Proceedings of the International Conference on Learning Representations*, 2018.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2017.

Bellemare, M. G., Dabney, W., and Rowland, M. *Distributional Reinforcement Learning*. MIT Press, 2023.

Dabney, W., Rowland, M., Bellemare, M., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Duan, J., Guan, Y., Li, S. E., Ren, Y., Sun, Q., and Cheng, B. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6584–6598, 2021.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Harutyunyan, A., Dabney, W., Mesnard, T., Gheshlaghi Azar, M., Piot, B., Heess, N., van Hasselt, H. P., Wayne, G., Singh, S., Precup, D., and Munos, R. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, 2019.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

Hoffman, M., Shahriari, B., Aslanides, J., Barth-Maron, G., Behbahani, F., Norman, T., Abdolmaleki, A., Cassirer, A., Yang, F., Baumli, K., Henderson, S., Friesen, A., Haroun, R., Novikov, A., Gómez Colmenarejo, S., Cabi, S., Gulcehre, C., Le Paine, T., Srinivasan, S., Cowie, A., Wang, Z., Piot, B., and de Freitas, N. Acme: A research framework for distributed reinforcement learning. *arXiv*, 2020.

Hung, C.-C., Lillicrap, T., Abramson, J., Wu, Y., Mirza, M., Carnevale, F., Ahuja, A., and Wayne, G. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):1–12, 2019.

Koenker, R. and Bassett Jr, G. Regression quantiles. *Econometrica: journal of the Econometric Society*, pp. 33–50, 1978.

Liu, H., Feng, Y., Mao, Y., Zhou, D., Peng, J., and Liu, Q. Action-depedent control variates for policy optimization via stein's identity. *arXiv preprint arXiv:1710.11198*, 2017.

Luo, Y., Liu, G., Duan, H., Schulte, O., and Poupart, P. Distributional reinforcement learning with monotonic splines. In *Proceedings of the International Conference on Learning Representations*, 2021.

Mesnard, T., Weber, T., Viola, F., Thakoor, S., Saade, A., Harutyunyan, A., Dabney, W., Stepleton, T., Heess, N., Guez, A., Moulines, É., Hutter, M., Buesing, L., and Munos, R. Counterfactual credit assignment in model-free reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2020.

Minsky, M. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Nam, D. W., Kim, Y., and Park, C. Y. GMAC: A distributional perspective on actor-critic framework. In *Proceedings of the International Conference on Machine Learning*, 2021.

Shahriari, B., Abdolmaleki, A., Byravan, A., Friesen, A., Liu, S., Springenberg, J. T., Heess, N., Hoffman, M., and Riedmiller, M. Revisiting Gaussian mixture critic in off-policy reinforcement learning: A sample-based approach. *arXiv preprint arXiv:2204.10256*, 2022.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, (3):229–256, 1992.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A. M., Kakade, S., Mordatch, I., and Abbeel, P. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv preprint arXiv:1803.07246*, 2018.

Zhou, F., Wang, J., and Feng, X. Non-crossing quantile regression for distributional reinforcement learning. *Advances in Neural Information Processing Systems*, 2020.

# APPENDICES

## A. Proofs

**Proposition A.1.** The QCA baseline results in unbiased policy gradient estimators when using exact return quantile functions $Q^\pi$, as shown in A. That is, with $Z_{X_t,A_t} = Q^\pi(X_t, A_t, \widehat{\tau}_t)$, we have

$$\nabla_\theta V^\pi(x_0) = \mathbb{E}\Big[\sum_t \gamma^t \nabla_\theta \log \pi_\theta(A_t|X_t) \\ (Z_{X_t,A_t} - Q^\pi(X_t, \pi, \widehat{\tau}_t))\Big].$$

*Proof.* In analogy with the proof of unbiasedness for the policy gradient estimator with a state-based baseline (see e.g. Sutton & Barto, 2018), it is sufficient to prove that for each $t \geq 0$, the baseline $Q^\pi(X_t, \pi, \widehat{\tau}_t)$ is conditionally independent of the action $A_t$ given $X_t$, so that

$$\begin{aligned}
&\mathbb{E}\Big[\sum_t \gamma^t \nabla_\theta \log \pi_\theta(A_t|X_t)(Z_{X_t,A_t} - Q^\pi(X_t, \pi, \widehat{\tau}_t))\Big] \\
&\qquad = \mathbb{E}\Big[\sum_t \gamma^t \nabla_\theta \log \pi_\theta(A_t|X_t) Z_{X_t,A_t}\Big] - \sum_t \gamma^t \mathbb{E}\Big[\nabla_\theta \log \pi_\theta(A_t|X_t) Q^\pi(X_t, \pi, \widehat{\tau}_t))\Big] \\
&\qquad = \nabla V^\pi(x_0) - \sum_t \gamma^t \mathbb{E}_{X_t}\Big[\mathbb{E}_{A_t,\widehat{\tau}_t}[\nabla_\theta \log \pi_\theta(A_t|X_t) Q^\pi(X_t, \pi, \widehat{\tau}_t))|X_t]\Big] \\
&\qquad = \nabla V^\pi(x_0) - \sum_t \gamma^t \mathbb{E}_{X_t}\Big[\mathbb{E}_{A_t}[\nabla_\theta \log \pi_\theta(A_t|X_t) \mid X_t]\mathbb{E}_{\widehat{\tau}_t}[Q^\pi(X_t, \pi, \widehat{\tau}_t))|X_t]\Big] \\
&\qquad = \nabla V^\pi(x_0),
\end{aligned}$$

where the second term evaluates to 0 since

$$\mathbb{E}_{A_t}[\nabla_\theta \log \pi_\theta(A_t|X_t)] = \sum_{a_t} \pi(a_t \mid X_t) \nabla \log \pi(a_t \mid X_t) = \nabla \sum_{a_t} \pi(a_t \mid X_t) = 0.$$

To see the required conditional independence property, note that conditional on $X_t$ and $A_t$, $\widehat{\tau}_t \sim \text{Uniform}([0,1])$ by construction, which does not depend on $A_t$ and hence $Q^\pi(X_t, \pi, \widehat{\tau}_t)$ is conditionally independent of $A_t$ given $X_t$. □

**Proposition A.2.** The QCA baseline provides an advantage estimate which has no greater variance than that associated with the value baseline when using exact quantile functions $Q^\pi$. More precisely, considering a random return $Z_{x,A}$ generated from a state-action pair $(x, A)$, with $A \sim \pi(\cdot|x)$, and writing $Z_{x,A} = Q^\pi(x, A, \widehat{\tau})$ we have

$$\text{Var}\left(Z_{x,A} - Q^\pi(x, \pi, \widehat{\tau})\right) = \text{Var}\left(Z_{x,A} - V^\pi(x)\right) - \\ \mathbb{E}_{\tau' \sim \mathcal{U}([0,1])}\left[\left(Q^\pi(x, \pi, \tau') - V^\pi(x)\right)^2\right].$$

*Proof.* Since both $Z - Q^\pi(x, \pi, \widehat{\tau})$ and $Z - V^\pi(x)$ are unbiased estimators of the advantage $A^\pi(x, a)$, it suffices to compare their second moments. We calculate directly:

$$\begin{aligned}
\mathbb{E}[(Z - V^\pi(x)^2] &= \mathbb{E}[(Z - Q^\pi(x, \pi, \widehat{\tau}) + Q^\pi(x, \pi, \widehat{\tau}) - V^\pi(x))^2] \\
&= \mathbb{E}[(Z - Q^\pi(x, \pi, \widehat{\tau}))^2 + 2(Z - Q^\pi(x, \pi, \widehat{\tau}))(Q^\pi(x, \pi, \widehat{\tau}) - V^\pi(x)) + (Q^\pi(x, \pi, \widehat{\tau}) - V^\pi(x))^2] \\
&= \mathbb{E}[(Z - Q^\pi(x, \pi, \widehat{\tau}))^2] + \mathbb{E}[(Q^\pi(x, \pi, \widehat{\tau}) - V^\pi(x))^2],
\end{aligned}$$

as required, with the final equality following since

$$\begin{aligned}
\mathbb{E}[(Z - Q^\pi(x, \pi, \widehat{\tau}))(Q^\pi(x, \pi, \widehat{\tau}) - V^\pi(x))] &= \mathbb{E}[\mathbb{E}_A[Z - Q^\pi(x, \pi, \widehat{\tau})](Q^\pi(x, \pi, \widehat{\tau}) - V^\pi(x))] \\
&= \mathbb{E}[(Q^\pi(x, \pi, \widehat{\tau}) - Q^\pi(x, \pi, \widehat{\tau}))(Q^\pi(x, \pi, \widehat{\tau}) - V^\pi(x))] \\
&= 0.
\end{aligned}$$

□

## B. Additional analysis

In this section, we provide additional analysis of the value-quantile baseline described in the main paper. First, we show that, as with the QCA baseline, using an exact VQCA baseline results in an unbiased policy gradient estimator.

**Proposition B.1.** The VQCA baseline results in unbiased policy gradient estimators when using exact return CDFs. That is, with $Z_t = V^\pi(X_t, \widehat{\tau}_t)$, we have

$$\nabla_\theta V^\pi(x_0) = \mathbb{E}\left[\sum_t \gamma^t \nabla_\theta \log \pi_\theta(A_t|X_t)(Z_t - V^\pi(X_t, \widehat{\tau}_t))\right].$$

*Proof.* We may follow the same approach as the proof of Proposition 3.1; it is sufficient to show that $\mathbb{E}[\nabla_\theta \log \pi_\theta(A_t|X_t)V^\pi(X_t, \widehat{\tau}_t)] = 0$. As in the proof of Proposition 3.1, this follows since $\mathbb{E}[\nabla_\theta \log \pi_\theta(A_t|X_t)] = 0$, and the fact that by construction, $\nabla_\theta \log \pi_\theta(A_t|X_t)$ and $\tau_t$ are conditionally independent given $X_t$. □

Next, we demonstrate that there are scenarios in which using a VQCA baseline can result in higher variance estimators than would be obtained with a standard expected-value baseline. For this reason, we do not recommend VQCA as a policy gradient baseline, instead preferring QCA, with the variance improvement guarantee established in Proposition 3.2.

*Example* B.2. Consider a single-state environment with two actions, $a$ and $b$, which are equally likely under the policy $\pi$. Suppose that the return when taking action $a$ is distributed as Unif$([-z-\varepsilon, -z+\varepsilon])$, and the return when taking action $b$ is distributed as Unif$([z-\varepsilon, z+\varepsilon])$, for $0 < \varepsilon \ll z$. The expected-value baseline in this case is 0, and so the variance of the return minus this estimator is

$$\mathbb{E}[Z^2] = z^2 + O(\varepsilon z + \varepsilon^2).$$

In contrast, the VQCA baseline, at level $\tau$, is

$$V(\tau) = \begin{cases} -z + 4(\tau - 1/4)\varepsilon & 0 < \tau < 1/2 \\ z + 4(\tau - 3/4)\varepsilon & 0 < \tau < 1/2. \end{cases}$$

The resulting variance of the return minus this estimator is therefore

$$\mathbb{E}[(Z - V(\tau))^2] = 2z^2 + O(\varepsilon),$$

and hence the variance is greater than with the expected-value baseline.

## C. Architectural and algorithmic improvements over quantile networks

Below we introduce a number of architectural improvements over the vanilla quantile network used in prior work on distributional RL (Dabney et al., 2018). The vanilla quantile network $Q_\psi(x, a, \tau_i)$ produces $m$ quantile predictions for $\tau_i = \frac{2i-1}{2m}$ with $1 \leq i \leq m$. Furthermore, in practice, we use a Huber loss variant of the quantile regression loss to learn quantile predictions (Dabney et al., 2018).

**Monotonicity of quantile parameterizations.** From the definition of quantile functions, we know that they increase monotonically as a function of the quantile levels $Q(x, a, \tau_i) \leq Q(x, a, \tau_j)$ for $i < j$. To leverage this property in the network design, we construct quantile predictions $Q_\psi(x, a, \tau_i)$ as a sum of non-negative increments. To utilize this attribute, we carry out the parameterization $Q_\psi(x, a, \tau_i) = \sum_{j=1}^{i} Q_\psi(x, a, j)$ where $Q_\psi(x, a, j)$ is parameterized to be non-negative via the softplus activation for the output layer $\log(1+\exp(x))$. We can understand $Q_\psi(x, a, 1)$ as the first quantile prediction and $Q_\psi(x, a, j), j \geq 2$ as the difference between two consecutive quantile predictions. This is a useful inductive bias and helps learn quantiles. Existing alternative architectures aiming exploit this structure include those of Zhou et al. (2020) and Luo et al. (2021).

**Dueling architecture.** Dueling network (Wang et al., 2016) proposes to have two streams to separately estimate state-value and the advantages for each action. Empirically, this has proved particularly useful in accelerating learning accurate Q-functions for value-based learning and distributional RL (Hessel et al., 2018). While prior work has adapted the dueling

architecture for C51 (Bellemare et al., 2017), an alternative distributional RL agent that learns CDF approximation instead of quantile approximation to the return, we propose a novel adaptation for the quantile network. Concretely, we carry out the parameterization

$$Q_\psi(x, a, \tau) = V(x) + A_\psi(x, a, \tau),$$

where $V(x)$ (which we call forward baseline below) is regressed (using a $\ell_2$-loss) toward the Monte-Carlo return $Z_{x,A}$, where $A \sim \pi(\cdot|x)$, and $A_\psi(x, a, \tau)$ are actually the output of our quantile-network (which thus learns the quantile function of the return minus the estimated value function).

## D. Implementation details

### D.1. High-Variance Key-to-Door

#### D.1.1. Environment details

Observations returned by the Key-to-Door family of environments for each of the three phases can be visualized in Fig. 7. Agents have 10 apples in the second phase to pick.

*Figure 7.* High-Variance Key-To-Door environments visual. The agent is represented by the beige pixel, key by brown, apples by green, and the final door by blue. The agent has a partial field of view, highlighted in white

#### D.1.2. Architecture

The agent architecture is as follows:

- The observations are first fed to 2-layer CNN with $(16, 32)$ output channels, kernel shapes of $(3, 3)$ and strides of $(1, 1)$. The output of the CNN is flattened and fed to a linear layer of size $128$.
- The agent state is computed by a forward LSTM with a state size of $128$. The input to the LSTM is the output of the previous linear layer, concatenated with the reward at the previous timestep.
- The hindsight feature $\Phi$ is computed by a backward LSTM with a state size of $128$. The input provided is the concatenation of the output of the forward LSTM and the reward at the previous timestep.
- The policy is computed as the output of a 2-layer MLP of $256$ units each where the output of the forward LSTM is provided as input. This MLP is shared with the policy. The policy is then linearly decoded from its outputs.
- The forward baseline is computed linearly decoded from the MLP shared with the policy.
- The quantile network is computed as the output of a 3-layer MLP of $128$ units each where the output of the forward LSTM is provided as input.
- The $\tau - network$ is the output of a 4-layer MLP of $128$ units each where the concatenation of the output of the forward LSTM and the hindsight feature $\Phi$ is provided as input.

- For CCA, the baseline is computed as the sum of the forward baseline and a hindsight residual baseline; the hindsight residual baseline is the output of a 3-layer MLP of 128 units each where the concatenation of the output of the forward LSTM and the hindsight feature $\Phi$ is provided as input. It is trained to learn the residual between the return and the forward baseline.

- For CCA, the hindsight classifier $h_\omega$ is computed as the sum of the log of the policy outputs and the output of an MLP, with four hidden layers with 256 units each where the concatenation of the output of the forward LSTM and the hindsight feature $\Phi$ is provided as input.

- All weights are jointly trained with RMSprop (Hinton et al., 2012) with epsilon $1e8$, momentum 0 and decay 0.99.

For High-Variance Key-To-Door, the optimal hyperparameters found for each algorithm can be found in Table 1.

The agents are trained on full-episode trajectories, using a discount factor of 0.9999.

| | PG | CCA | DRL | QCA | HQCA |
|---|---|---|---|---|---|
| Learning rate | 5e-4 | 5e-4 | 5e-4 | 5e-4 | 5e-4 |
| Policy cost | 1 | 1 | 1 | 1 | 1 |
| Entropy cost | 1e-2 | 1e-2 | 1e-2 | 1e-2 | 1e-2 |
| Forward baseline cost | 1e-1 | 1e-2 | 1e-1 | 1e-1 | 1e-1 |
| Number of discrete quantiles | ____ | ____ | 5 | 5 | 10 |
| Huber loss param | ____ | ____ | 1. | 1. | 1. |
| Quantile regression cost | ____ | ____ | 1e-1 | 1e-1 | 1e-1 |
| Hindsight quantile prediction cost | ____ | ____ | ____ | ____ | 1e-2 |
| Hindsight residual baseline cost Mesnard et al. (2020) | ____ | 1e-2 | ____ | ____ | _____ |
| Hindsight classifier cost Mesnard et al. (2020) | ____ | 5e-3 | ____ | ____ | _____ |
| Action independence cost Mesnard et al. (2020) | ____ | 1e2 | ____ | ____ | _____ |

*Table 1.* List of hyperparameters used for all experiments.

### D.2. Random Key-to-Door

#### D.2.1. Environment details

Observations returned by the Random Key-To-Door family of environments for each of the three phases can be visualized in Fig. 8. Agents get immediate random rewards during the second phase, distracting them from opening the door.

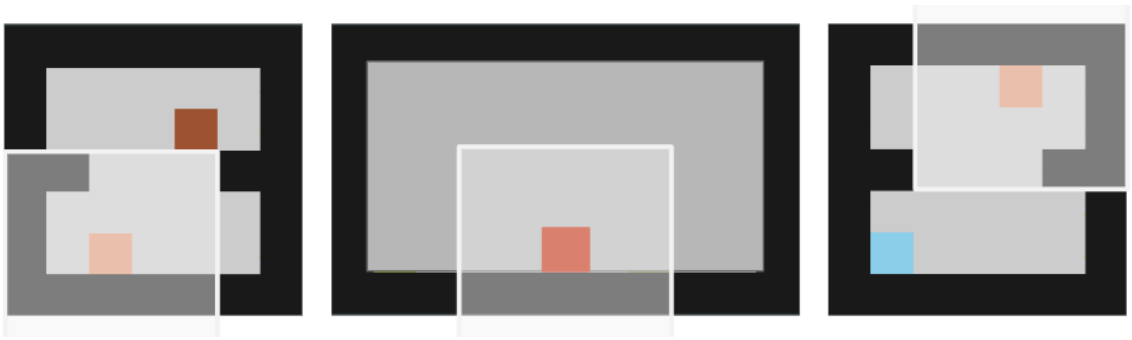


*Figure 8.* Random Key-To-Door environments visual. The agent is represented by the beige pixel, key by brown, and the final door by blue. The agent has a partial field of view, highlighted in white

For each task, a random, but fixed through training, set of 5 out of 10 colored squares are leading to a positive reward. Furthermore, a small reward of 0.5 is provided to the agent when it picks up any colored square. Each episode are 130 steps long and it takes at least 9 steps for the agent to reach one colored square in the query rooms from its initial position and 6 in the answer rooms.

#### D.2.2. ARCHITECTURE

We use the same architecture setup as reported in Appendix D.1.2. The agents are also trained on full-episode trajectories, using a discount factor of 0.9999. For Random Key-to-Door, the optimal hyperparameters found for each algorithm are the same as in Table 1.