A. Derivation of results for generalized trust-region policy optimization

A.1. Controlling the residuals of Taylor expansions

We summarize the bound on the magnitude of the Taylor expansion residuals of the Q-function as a proposition.

Proposition 1. *Recall the definition of the Taylor expansion residual of the Q-function from the main text,* $E_K \triangleq \sum_{k=K+1}^{\infty} U_k$. *Let* $||\cdot||$ *be the infinity norm* $||A|| \triangleq \max_{x,a} |A(x,a)|$. *Let* $R_{\max}$ *be the maximum reward in the entire MDP,* $R_{\max} \triangleq \max_{x,a} |r(x,a)|$. *Finally, let* $\varepsilon \triangleq ||\pi - \mu||_1$. *Then*

$$||E_K|| \leq \left(\frac{\gamma}{1-\gamma}\varepsilon\right)^{K+1} \left(1 - \frac{\gamma}{1-\gamma}\varepsilon\right)^{-1} \frac{R_{\max}}{1-\gamma}. \tag{13}$$

Proof. The proof follows by bounding each the magnitude of term $||U_k||$,

$$\begin{aligned}
||E_K|| = \left\| \sum_{k=K+1}^{\infty} U_k \right\| &\leq \sum_{k=K+1}^{\infty} ||U_k|| \\
&= \sum_{k=K+1}^{\infty} \|\gamma^k \left((I - \gamma P^\mu)^{-1}(P^\mu - P^\mu)\right)^k\| \cdot ||Q^\mu|| \\
&\leq \sum_{k=K+1}^{\infty} \left(\frac{\gamma}{1-\gamma}\right)^k \varepsilon^k \frac{R_{\max}}{1-\gamma} \\
&= \left(\frac{\gamma}{1-\gamma}\varepsilon\right)^{K+1} \left(1 - \frac{\gamma}{1-\gamma}\varepsilon\right)^{-1} \frac{R_{\max}}{1-\gamma}.
\end{aligned}$$

The above derivation shows that once we have $\varepsilon < \frac{1-\gamma}{\gamma}$, $||E_K|| \to 0$ as $K \to \infty$. In the above derivation, we have applied the bound $||U_k|| \leq \left(\frac{\gamma}{1-\gamma}\right)^k \varepsilon^k \frac{R_{\max}}{1-\gamma}$, which will also be helpful in later derivations. □

A.2. Deriving Taylor expansions of RL objective

Recall that the RHS of Eq. 2 are the Taylor expansions of Q-functions Q^π. By construction, $Q^\pi - Q^\mu = \sum_{k \geq 0} U_k$. Though Eq. 2 shows the expansion of the entire vector Q^π, for optimization purposes, we care about the RL objective from a starting state x_0, $J(\pi) = \mathbb{E}_{\pi, a_0 \sim \pi(\cdot|x_0), x_0}[Q^\pi(x_0, a_0)] = \pi_0^\intercal Q^\pi$, where $\pi_0 \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ follows the definition from the main paper $\pi_0(x,a) \triangleq \pi(a|x)\delta_{x=x_0}$.

Now we focus on calculating $L_K(\pi, \mu)$ for general $K \geq 0$. For simplicity, we write $L_k(\pi, \mu)$ as L_k and henceforth we might use these notations interchangeably. Now consider the RHS of Eq. 3. By definition of the k-th order Taylor expansion L_k $(1 \leq k \leq K)$ of $J(\pi) - J(\mu)$, we maintain terms where $\pi/\mu - 1$ appears at most K times. Equivalently, in matrix form, we remove the higher order terms of $\pi - \mu$ while only maintaining terms such as $(\pi - \mu)^k, k \leq K$. This allows us to conclude that

$$\sum_{i=1}^{K} L_k = (\pi_0 - \mu_0)^\intercal \left(Q^\mu + \sum_{k \geq 1}^{K-1} U_k\right) + \mu_0^\intercal \left(\sum_{k=1}^{K} U_k\right).$$

Furthermore, we can single out each term

$$\begin{aligned}
L_k &= (\pi_0 - \mu_0)^\intercal T_{k-1} + \mu_0^\intercal U_k, \;\; k \geq 2 \\
L_1 &= (\pi_0 - \mu_0)^\intercal Q^\mu + \mu_0^\intercal U_1.
\end{aligned}$$

B. Proof of Theorem 1

Proof. We derive the Taylor expansion of Q-function Q^π into different orders of $P^\pi - P^\mu$. For that purpose, we recursively make use of the following matrix equality

$$(I - \gamma P^\pi)^{-1} = (I - \gamma P^\mu)^{-1} + \gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu)(I - \gamma P^\pi)^{-1},$$

which can be derived either from matrix inversion equality or directly verified. Since $Q^\pi = (I - \gamma P^\pi)^{-1} R$, we can use the previous equality to get

$$\begin{aligned} Q^\pi &= (I - \gamma P^\mu)^{-1} R + \gamma (I - \gamma P^\mu)^{-1} (P^\pi - P^\mu)(I - \gamma P^\pi)^{-1} R \\ &= Q^\mu + \gamma (I - \gamma P^\mu)^{-1} (P^\pi - P^\mu) Q^\pi. \end{aligned}$$

Next, we recursively apply the equality K times,

$$Q^\pi = Q^\mu + \sum_{k=1}^{K} \big(\gamma (I - \gamma P^\mu)^{-1} (P^\pi - P^\mu)\big)^k Q^\mu + \big(\gamma (I - \gamma P^\mu)^{-1} (P^\pi - P^\mu)\big)^{K+1} Q^\pi.$$

Now if $||\pi - \mu||_1 < (1-\gamma)/\gamma$ then we can bound the sup-norm in of the above term as

$$\|\gamma (I - \gamma P^\mu)^{-1} (P^\pi - P^\mu)\|_\infty = \frac{\gamma}{1-\gamma} ||\pi - \mu||_1 < 1,$$

thus the $(K+1)$-th order residual term vanishes when $K \to \infty$. As a result, the limit $K \to \infty$ is well defined and we deduce

$$Q^\pi = \sum_{k=0}^{\infty} \big(\gamma (I - \gamma P^\mu)^{-1} (P^\pi - P^\mu)\big)^k Q^\mu.$$

□

C. Proof of Theorem 2

Proof. To derive the monotonic improvement theorem for generalized TRPO, it is critical to bound $\sum_{k=K+1}^{\infty} L_k$. We achieve this by simply bounding each term separately. Recall that from Appendix A.1 we have $||U_k|| \leq \left(\frac{\gamma}{1-\gamma}\right)^k \varepsilon^k R_{\text{max}}$. Without loss of generality, we first assume $R_{\text{max}} = 1 - \gamma$ for ease of derivations.

$$|L_k| \leq \varepsilon ||U_{k-1}|| + ||U_k|| \leq \varepsilon \left(\frac{\gamma}{1-\gamma}\right)^{k-1} \varepsilon^{k-1} + \left(\frac{\gamma}{1-\gamma}\right)^k \varepsilon^k = \left(\frac{\gamma}{1-\gamma}\right)^{k-1} \frac{1}{1-\gamma} \varepsilon^k.$$

This leads to a bound over the residuals

$$\left| \sum_{k=K+1}^{\infty} L_k \right| \leq \sum_{k=K+1}^{\infty} |L_k| \leq \sum_{k=K+1}^{\infty} \left(\frac{\gamma}{1-\gamma}\right)^{k-1} \frac{1}{1-\gamma} \varepsilon^k = \frac{1}{\gamma} \left(1 - \frac{\gamma}{1-\gamma}\varepsilon\right)^{-1} \left(\frac{\gamma\varepsilon}{1-\gamma}\right)^{K+1}.$$

Since we have the equality $J(\pi) = J(\mu) + \sum_{k=1}^{\infty} L_k$ for $||\pi - \mu||_1 \leq \varepsilon < \frac{1-\gamma}{\gamma}$, we can deduce the following monotonic improvement,

$$J(\pi) \geq J(\mu) + \sum_{k=1}^{K} L_k - \frac{1}{\gamma} \left(1 - \frac{\gamma}{1-\gamma}\varepsilon\right)^{-1} \left(\frac{\gamma\varepsilon}{1-\gamma}\right)^{K+1}. \tag{14}$$

To write the above statement in a compact way, we define the gap

$$G_K = \frac{1}{\gamma} \left(1 - \frac{\gamma}{1-\gamma}\varepsilon\right)^{-1} \left(\frac{\gamma\varepsilon}{1-\gamma}\right)^{K+1}.$$

To derive the result for general R_{max}, note that the gap G_K has a linear dependency on R_{max}. Hence the general gap is

$$G_K \triangleq \frac{1}{\gamma(1-\gamma)} \left(1 - \frac{\gamma}{1-\gamma}\varepsilon\right)^{-1} \left(\frac{\gamma\varepsilon}{1-\gamma}\right)^{K+1} R_{\text{max}},$$

which gives produces the monotonic improvement result (Eq. 10) stated in the main paper. □

D. Proof of Theorem 3

Proof. It is known that for $K = 1$, replacing $Q^\mu(x, a)$ by $A^\mu(x, a)$ in the estimation can potentially reduce variance (Schulman et al., 2015; 2017) yet keeps the estimate unbiased. Below, we show that in general, replacing $Q^\pi(x, a)$ by $A^\pi(x, a)$ renders the estimate of $L_K(\pi, \mu)$ unbiased for general $K \geq 1$.

As shown above and more clearly in Appendix F, $L_K(\pi, \mu)$ can be written as

$$L_K(\pi, \mu) = \mathbb{E}_{(x^{(i)}, a^{(i)})_{1 \leq i \leq K}} \left[\prod_{i=1}^{K} \left(\frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} - 1 \right) Q^\mu(x_K, a_K) \right]. \tag{15}$$

Note that for clarity, in the above expectation, we omit an explicit sequence of discounted visitation distributions (for detailed derivations of this sequence of visitation distributions, see Appendix F). Next, we leverage the conditional expectation with respect to $(x^{(i)}, a^{(i)}), 1 \leq i \leq K - 1$ to yield

$$\begin{aligned}
L_K(\pi, \mu) &= \mathbb{E}_{(x^{(i)}, a^{(i)})_{1 \leq i \leq K-1}} \left[\prod_{i=1}^{K-1} \left(\frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} - 1 \right) \mathbb{E}_{(x_K, a_K)} \left[\left(\frac{\pi(a_K|x_K)}{\mu(a_K|x_K)} - 1 \right) Q^\mu(x_K, a_K) \right] \right] \\
&= \mathbb{E}_{(x^{(i)}, a^{(i)})_{1 \leq i \leq K-1}} \left[\prod_{i=1}^{K-1} \left(\frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} - 1 \right) \mathbb{E}_{(x_K, a_K)} \left[\left(\frac{\pi(a_K|x_K)}{\mu(a_K|x_K)} - 1 \right) A^\mu(x_K, a_K) \right] \right] \\
&= \mathbb{E}_{(x^{(i)}, a^{(i)})_{1 \leq i \leq K}} \left[\prod_{i=1}^{K} \left(\frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} - 1 \right) A^\mu(x_K, a_K) \right].
\end{aligned} \tag{16}$$

The above derivation shows that indeed, replacing $Q^\mu(x, a)$ by $A^\mu(x, a)$ does not change the value the expectation, while potentially reducing the variance of the overall estimation. □

E. Proof of Theorem 4

Proof. From the definition of the *return off-policy evaluation operator* $\mathcal{R}_1^{\pi,\mu}$, we have

$$\begin{aligned}
\mathcal{R}_1^{\pi,\mu} Q &= Q + (I - \gamma P^\mu)^{-1}(r + \gamma P^\pi Q - Q) \\
&= (I - \gamma P^\mu)^{-1}(r + \gamma P^\pi Q - Q + (I - \gamma P^\mu)Q) \\
&= (I - \gamma P^\mu)^{-1} r + \gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu)Q \\
&= Q^\mu + \gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu)Q.
\end{aligned}$$

Thus $Q \mapsto \mathcal{R}_1^{\pi,\mu} Q$ is a linear operator, and

$$\begin{aligned}
(\mathcal{R}_1^{\pi,\mu})^2 Q &= Q^\mu + \gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu)(\mathcal{R}_1^{\pi,\mu})^2 Q \\
&= Q^\mu + \gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu)Q^\mu + \left[\gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu) \right]^2 Q.
\end{aligned}$$

Applying this step K times, we deduce

$$(\mathcal{R}_1^{\pi,\mu})^K Q = Q^\mu + \sum_{k=1}^{K-1} \left[\gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu) \right]^k Q^\mu + \left[\gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu) \right]^K Q.$$

Applying the above operator to Q^μ we deduce that

$$(\mathcal{R}_1^{\pi,\mu})^K Q^\mu = Q^\mu + \sum_{k=1}^{K} \underbrace{\left[\gamma (I - \gamma P^\mu)^{-1}(P^\pi - P^\mu) \right]^k Q^\mu}_{=U_k},$$

which proves our claim. □

F. Alternative derivation for Taylor expansions of RL objective

In this section, we provide an alternative derivation of the Taylor expansion of the RL objective. Let $\pi_t/\mu_t = 1 + \varepsilon_t$. In cases where $\pi \approx \mu$ (e.g., for the trust-region case), $\varepsilon \approx 0$. To calculate $J(\pi)$ using data from μ, a natural technique is employ importance sampling (IS),

$$J(\pi) = \mathbb{E}_{\mu,x_0}\left[\left(\prod_{t=0}^{\infty}\frac{\pi_t}{\mu_t}\right)\sum_{t=0}^{\infty}r_t\gamma^t\right] = \mathbb{E}_{\mu,x_0}\left[\left(\prod_{t=0}^{\infty}(1+\varepsilon_t)\right)\sum_{t=0}^{\infty}\gamma^t r_t\right].$$

To derive Taylor expansion in an *intuitive* way, consider expanding the product $\prod_{t=0}^{\infty}(1+\varepsilon_t)$, assuming that this infinite product is finite. Assume all $\varepsilon_t \leq \varepsilon$ with some small $\varepsilon > 0$. A second-order Taylor expansion is

$$\prod_{t=0}^{\infty}(1+\varepsilon_t) = 1 + \sum_{t=0}^{\infty}\varepsilon_t + \sum_{t=0}^{\infty}\sum_{t'>t}^{\infty}\varepsilon_t\varepsilon_{t'} + O(\varepsilon^3). \tag{17}$$

Now, consider the term associated with $\sum_{t=0}^{\infty}\varepsilon_t$,

$$\begin{aligned}
\mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\varepsilon_t\sum_{t=0}^{\infty}\gamma^t r_t\right] &= \mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\varepsilon_t\sum_{t'=t}^{\infty}r_t\gamma^{t'}\right] \\
&= \mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\varepsilon_t\sum_{t'=t}^{\infty}r_t\gamma^t\gamma^{t'-t}\right] \\
&= \mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\varepsilon_t Q^{\mu}(x_t,a_t)\gamma^t\right] \\
&= (1-\gamma)\mathop{\mathbb{E}}_{\substack{x,a\sim d_\gamma^\mu(\cdot,\cdot|x_0,a_0,0)\\ a_0\sim\mu(\cdot|x_0)}}\left[\left(\frac{\pi(a|x)}{\mu(a|x)}-1\right)Q^{\mu}(x,a)\right].
\end{aligned} \tag{18}$$

Note that in the last equality, the γ^t factor is absorbed into the discounted visitation distribution $d_\gamma^\mu(\cdot,\cdot|x_0,a_0,0)$. It is then clear that this term is exactly the first-order expansion $L_1(\pi,\mu)$ shown in the main paper.

Similarly, we could derive the second-order expansion by studying the term associated with $\sum_{t=0}^{\infty}\sum_{t'>t}^{\infty}\varepsilon_t\varepsilon_{t'}$.

$$\begin{aligned}
\mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\sum_{t'>t}^{\infty}\varepsilon_t\varepsilon_{t'}\sum_{t=0}^{\infty}\gamma^t r_t\right] &= \mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\sum_{t'>t}^{\infty}\varepsilon_t\varepsilon_{t'}\sum_{\tau=t'}^{\infty}r_\tau\gamma^\tau\right] \\
&= \mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\sum_{t'>t}^{\infty}\varepsilon_t\varepsilon_{t'}\sum_{\tau=t'}^{\infty}r_\tau\gamma^{\tau-t'}\gamma^t\gamma^{t'-t}\right] \\
&= \mathbb{E}_{\mu,x_0}\left[\sum_{t=0}^{\infty}\sum_{t'>t}^{\infty}\varepsilon_t\varepsilon_{t'}Q^{\mu}(x_{t'},a_{t'})\gamma^t\gamma^{t'-t}\right] \\
&= \frac{(1-\gamma)^2}{\gamma}\mathop{\mathbb{E}}_{\substack{x,a\sim d_\gamma^\mu(\cdot,\cdot|x_0,a_0,0)\\ a_0\sim\mu(\cdot|x_0)\\ x',a'\sim d_\gamma^\mu(\cdot,\cdot|x,a,1)}}\left[\left(\frac{\pi(a|x)}{\mu(a|x)}-1\right)\left(\frac{\pi(a'|x')}{\mu(a'|x')}-1\right)Q^{\mu}(x',a')\right].
\end{aligned} \tag{19}$$

Note that similar to the first-order expansion, the discount factor $\gamma^t\gamma^{t'-t}$ is absorbed into the discounted visitation distribution $d_\gamma^\mu(\cdot,\cdot|x_0,a_0,0)$ and $d_\gamma^\mu(\cdot,\cdot|x,a,1)$ respectively. Here note that the second discounted visitation distribution is $d_\gamma^\mu(\cdot,\cdot|x,a,1)$ instead of $d_\gamma^\mu(\cdot,\cdot|x,a,0)$ — this is because $t'-t \geq 1$ by construction and we need to sample the second state *conditional* on the time difference to be ≥ 1. The above is exactly the second-order expansion $L_2(\pi,\mu)$.

By a similar argument, we can derive expansion for all higher-order expansion by considering the term associated with $\sum_{t_1=0}^{\infty}\sum_{t_2>t_1}^{\infty}\cdots\sum_{t_K>t_{K-1}}^{\infty}\varepsilon_1\varepsilon_2\ldots\varepsilon_K$. This would introduce K discounted visitation distributions $d_\gamma^\mu(\cdot,\cdot|x_0,a_0,0)$ and $d_\gamma^\mu(\cdot,\cdot|x_k,a_k,1), 1\leq k\leq K$.

The above derivation also illustrates how these higher-order terms can be estimated in practice. For the k-th order, given a trajectory under μ, sequentially sample K time difference Δt_k along the trajectory, where $t_1 \sim \text{Geometric}(1-\gamma)$. For $k \geq 2$, $t_k \sim \text{Geometric}(1-\gamma)$ while conditional on $\Delta t_k \geq 1$. Then define the time $t_k = \sum_{i \leq k} \Delta t_i$. Let $x_i = x_{t_i}$ and $a_i = a_{t_i}$, Then, a one sample estimate is

$$\prod_{i=1}^{K} \left(\frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} - 1 \right) Q^{\mu}(x_K, a_K). \tag{20}$$

F.1. Connection between off-policy evaluation and generalized advantage estimation (GAE)

Generalized advantage estimation (GAE, Schulman et al. 2016) is a technique for advantage estimation. According to Schulman et al. (2016; 2017), GAE trades-off bias and variance in the advantage estimation and can boost the performance of downstream policy optimization. On the other hand, off-policy evaluation operators (Harutyunyan et al., 2016; Munos et al., 2016) are dedicated to evaluations of Q-function $Q^{\pi}(x, a)$. What are the connections between these approaches?

The actor-critic algorithm that uses GAE maintains a policy $\pi(a|x)$ and value function $V_{\varphi}(x)$ with parameter φ. Data are collected on-policy, i.e., $\mu = \pi$. Let $\hat{A}_{\text{GAE}}(x, a)$ be the GAE estimation for (x, a). Naturally, GAE can be interpreted as first carrying out a Q-function estimation $\hat{Q}(x, a)$ and then subtracting the baseline

$$\hat{A}_{\text{GAE}}(x, a) \triangleq \hat{Q}(x, a) - V_{\varphi}(x). \tag{21}$$

Now we show that the Q-function estimation $\hat{Q}(x, a)$ can be interpreted as applying the $Q(\lambda)$ operator to an initial Q-function estimate. Here importantly, to make the connection exact, we assume the initial Q-function estimate to be bootstrapped from the value function $Q_{\text{init}}(x, a) \triangleq V_{\varphi}(x)$. To sum up,

$$\hat{A}_{\text{GAE}}(x, a) = [\mathcal{R}^{\pi,\pi}_{c=\lambda} Q_{\text{init}}](x, a) - V_{\varphi}(x), \tag{22}$$

where $\mathcal{R}^{\pi,\mu}_{c=\lambda}$ refers to the evaluation operator with trace coefficients $c(x, a) = \lambda$. Finally, the evaluation operator is replaced by sample estimates in practice. From the above, we see that there is a link between advantage estimation (i.e., GAE) with policy evaluation (i.e., the $Q(\lambda)$ operator).

G. Second-order expansions for value-based algorithms

In this section, we provide algorithmic details on value-based algorithms in our experiments. The application of Taylor expansions allow us to derive the expansion for RL objective, which is useful in policy-optimization where algorithms maintain a parameterized policy π_{θ}. Taking one step back, Taylor expansion can be used for policy evaluation as well, and can be useful in algorithms where Q-functions (value functions) are parameterized Q_{θ} where the policy is implicitly defined (e.g., ε-greedy). In our experiments, we take R2D2 (Kapturowski et al., 2019) as the baseline algorithm. Below, we briefly introduce the algorithmic procedure of R2D2 and present the Taylor expansion variants.

Basic components. The baseline R2D2 maintains a Q-function $Q_{\theta}(x, a)$ parameterized by a neural network θ. The central learner maintains an updated parameter θ and distributed actors maintain slightly delayed copies θ_{old}. Distributed actors collect data using behavior policy μ, defined as ε-greedy with respect to $Q_{\theta_{\text{old}}}(x, a)$. The target policy π is defined as greedy with respect to $Q_{\theta}(x, a)$. Actors send data to a replay buffer, and the learner samples partial trajectories $(x_t, a_t, r_t)_{t=1}^{T}$ from the buffer and computes updates to the parameter θ. In particular, the learner calculates regression targets $Q_{\text{target}}(x_t, a_t)$ and the Q-function is updated via $\theta \leftarrow \theta - \alpha \cdot \nabla_{\theta}(Q_{\theta}(x_t, a_t) - Q_{\text{target}}(x_t, a_t))^2$ with learning rate $\alpha > 0$.

Algorithmic variants. Algorithmic variants of R2D2 differ in how they compute the targets $Q_{\text{target}}(x, a)$. A useful unified view provided by Rowland et al. (2020) is that $Q_{\text{target}}(x, a)$ aims to approximate $Q^{\pi}(x, a)$ such that $Q_{\theta}(x, a) \to Q^{\pi}(x, a)$ during the update.

Along sampled trajectories, we recursively calculate the targets $Q_{\text{target}}(x_t, a_t), 1 \leq t \leq T$ based on recipes of different variants. Below are a few alternatives we evaluated in our experimenst, where we e.g., use $\hat{Q}_{\text{zero}}(x_t, a_t)$ to represent $Q_{\text{target}}(x_t, a_t)$ for the zero-order baseline.

- **Zero-order:** $\hat{Q}_{\text{zero}}(x_t, a_t) \triangleq r_t + \gamma \hat{Q}_{\text{zero}}(x_{t+1}, a_{t+1})$

- **First-order:** $\hat{Q}_{\text{first}}(x_t, a_t) \triangleq r_t + \gamma(\mathbb{E}_\pi[Q_\theta(x_{t+1}, \cdot)] - Q_\theta(x_{t+1}, a_{t+1})) + \gamma \hat{Q}_{\text{first}}(x_{t+1}, a_{t+1})$
- **Second-order:** $\hat{Q}_{\text{second}}(x_t, a_t) \triangleq r_t + \gamma\left(\mathbb{E}_\pi\left[\hat{Q}'_{\text{first}}(x_{t+1}, \cdot)\right] - \hat{Q}_{\text{first}}(x_{t+1}, a_{t+1})\right) + \gamma \hat{Q}_{\text{second}}(x_{t+1}, a_{t+1})$
- **Retrace:** $\hat{Q}_{\text{retrace}}(x_t, a_t) \triangleq r_t + \gamma c_{t+1}(\mathbb{E}_\pi[Q_{\text{retrace}}(x_{t+1}, \cdot)] - Q_{\text{retrace}}(x_{t+1}, a_{t+1})]) + \gamma c_{t+1} \hat{Q}_{\text{retrace}}(x_{t+1}, a_{t+1})$.

For retrace, we set the trace coefficient $c_t \triangleq \lambda \cdot \min\left\{\frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}, 1\right\}$ following Munos et al. (2016). All baselines bootstrap $\hat{Q}_{\text{target}}(x_T, a_T) = Q_\theta(x_t, a_T)$ from the Q-function network for the last state-action pair.

As shown above, the zero-order baseline reduces to discounted sum of returns (plus a bootstrap value at the end of the trajectory). The first-order adopts the $Q(\lambda), \lambda = 1$ recursive update rule. The second-order corresponds to applying $Q(\lambda), \lambda = 1$ twice to the partial trajectory—in particular, this corresponds to replacing the Q-function baseline $Q_\theta(x, a)$ by first-order approximations $Q_{\text{first}}(x, a)$. For the above, we define $\hat{Q}'_{\text{first}}(x_t, a) \triangleq \mathbb{I}_{a=a_t} \hat{Q}_{\text{first}}(x_t, a) + (1 - \mathbb{I}_{a=a_t}) Q_\theta(x_t, a)$ where $\mathbb{I}$ is the indicator function. This ensures that the expectations are well defined in the recursive updates.

As discussed in the main paper, it is not always necessarily optimal to carry out exact first/second-order correction, it might be potentially beneficial to strike a balance in between for bias-variance trade-off. To this end, we define the ultimate second-order target as $\hat{Q}_{\text{target-final}} = \hat{Q}_{\text{first}} + \eta(\hat{Q}_{\text{second}} - \hat{Q}_{\text{first}})$ for $\eta \geq 0$.

See Figure 4 and Figure 7 for the comparison results of these algorithmic variants. Further hyper-parameter details are specified in Appendix H.7.

H. Additional experimental details and results

H.1. Random MDP

The random MDP is identified by the number of states $|\mathcal{X}|$ and actions $|\mathcal{A}|$. The transitions $p(x'|x, a)$ are generated as samples from a Dirichlet distribution. The reward function $r(x, a)$ is generated as a Dirac, sampled uniformly at random from $[-1, 1]$. The discount factor is set to $\gamma \triangleq 0.9$. The results in Figure 1 are averaged over 10 MDPs.

We randomly fix a target policy π and randomly sample another behavior policy μ in the vicinity of π such that $||\pi - \mu||_1 \leq \varepsilon$, for some fixed $\varepsilon > 0$. Effectively, ε controls the off-policiness measured as the difference between π and , μ. When using the reward estimate $\hat{R}$ to compute the Q-function estimate, trajectories are generated under the behavior policy μ. The reward estimate is initialized to be zeros $\hat{R}(x, a) \triangleq 0$ for all x and a. Since the rewards are deterministic, we have have that when (x, a) is encountered then $\hat{R}(x, a) \leftarrow R(x, a)$.

H.2. Evaluation of distributed experiments

For this part, the evaluation environments is the entire suite of Atari games (Bellemare et al., 2013) consisting of 57 levels. Since each level has very different reward scale and difficulty, we report human-normalized scores for each level, calculated as $z_i = (r_i - o_i)/(h_i - o_i)$, where h_i and o_i are the performances of human and a random policy on level i respectively.

For all experiments, we report summarizing statistics of the human-normalized scores across all levels. For example, at any point in training, the mean human-normalized score is the mean statistic across $z_i, 1 \leq i \leq 57$.

H.3. Details on distributed algorithms

Distributed algorithms have led to significant performance gains on challenging domains (Nair et al., 2015; Mnih et al., 2016; Babaeizadeh et al., 2017; Barth-Maron et al., 2018; Horgan et al., 2018). Here, our focus is on recent state-of-the-art algorithms. In general, distributed agents consist of one central learner, multiple actors and optionally a replay buffer. The central learner maintains a parameter copy θ and updates parameters based on sampled data. Multiple actors each maintaining a slighted delayed parameter copy θ_{old} and interact with the environment to generate partial trajectories. Actors synchronize parameters from the learner periodically.

Algorithms differ by how are data and parameters passed between each component. We focus on two types of state-of-the-art scalable topologies: **Type I** adopts IMPALA-typed architecture (Espeholt et al., 2018; see blue arrows in Figure 5 in Appendix H), data are directly passed from actors to the learner. See Section 5.1 and Section 5.2; **Type II**. adopts

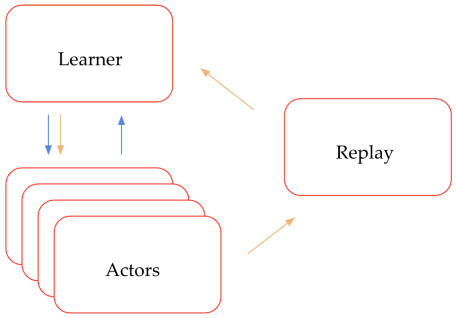

Figure 5. Architecture of distributed agents. Agents differ by the topology, i.e., how actors/learner/replay pass data/parameters between them. The above architecture summarizes common setups such as IMPALA (Espeholt et al., 2018) as blue arrows and R2D2 (Kapturowski et al., 2019) as orange arrows.

R2D2-typed architecture (Kapturowski et al. 2019, see orange arrows in Figure 5 in Appendix H), data are sent from actors to a replay, and later sampled according to priorities to the learner (Horgan et al., 2018).

H.4. Details on TayPO-2 for policy optimization

Discussion on the first-order objective. By construction, the first-order objective (Eq. 5) samples states with a discounted visitation distribution. Though such an objective is conducive to theoretical analysis, it is too conservative in practice. Indeed, the practical objective is usually undiscounted $\approx \mathbb{E}_{x_0,a_0}[\sum_{t=0}^{T_e} r_t]$ where T_e is an artificial threshold of the episode length. Therefore, in practice, the state x is sampled 'uniformly' from generated trajectories, i.e., without the discount factor γ^t.

Discussion on the TayPO-2 objective. For the second-order objective (Eq. 6), recall that we sample two state-action pairs $(x, a), (x', a')$. In practice, we sample (x, a) uniformly (without discount) as the first-order objective and sample (x', a') with discount factors $\gamma^{\Delta t}$ where Δt is the time difference between x' and x. This is to ensure that we have a comparable loss function $\hat{L}_2(\pi, \mu)$ compared to the first-order $\hat{L}_1(\pi, \mu)$.

Further practical considerations. In practice, loss functions are computed on partial trajectories $(x_t, a_t)_{t=1}^T$ with length T. Though, theoretically, evaluating $\hat{L}_2(\pi_\theta, \mu)$ requires generating time steps from a geometric distribution Geometric$(1-\gamma)$ which can exceed the length T, in practice, we apply the truncation at T. In addition, we evaluate $\hat{L}_2(\pi_\theta, \mu)$ by enumerating over the entire (truncated) trajectory instead of sampling time steps. This comes at several trade-offs: enumerating the trajectory require $\mathcal{O}(T^2)$ computations while sampling can reduce this complexity to $\mathcal{O}(T)$; enumerating over all steps could reduce the variance by pooling all data of the trajectory, but could also increase the variance due to correlations of state-action pairs on a single trajectory. In practice, we find enumerating all steps along the trajectory works well.

H.5. Near on-policy policy optimization

Additional results. The additional results on the Atari suite are in Figure 6, where, we show the median human normalized scores during training. We notice that the second-order still steadily outperforms other baselines.

Discussion on proximal policy optimization (PPO) implementation. By design, PPO (Schulman et al., 2017) alternates between data collection and policy update. The data are always collected under $\mu = \pi$ and the new policy gets updated via several gradient steps on the same batch of data. In practice, such a 'fully-synchronized' implementation is not efficient because it does not leverage a distributed computational architecture. To improve the implementation, we modify the original algorithm and adapt it to an asynchronous setting. To this end, several changes must be made to the algorithm.

- The data are collected with actor policy μ instead of the previous policy.
- The number of gradient descent per batch is one instead of multiple, to balance the data throughput from the actor.

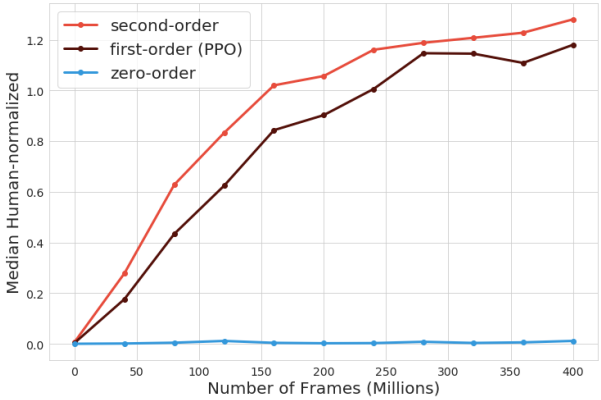

Figure 6. Near on-policy optimization. The x-axis is the number of frames (millions) and y-axis shows the median human-normalized scores averaged across 57 Atari levels. The plot shows the mean curve averaged across 3 random seeds. We observe that second-order expansions allow for faster learning and better asymptotic performance given the fixed budget on actor steps.

Details on computational architecture. For the near on-policy optimization experiments, we set up an agent with an algorithmic architecture similar to that of IMPALA (Espeholt et al., 2018). In order to minimize the delays between actors and the central learner, we schedule all components of the algorithms on a single host machine. The learner uses a single TPU for fast inference and computation, while the actors use CPUs for fast batched environment rollouts.

We apply a small network similar to Mnih et al. (2016), please see Appendix H.6 for detailed descriptions of the architecture.

Following the conventional practice of training on Atari games (Mnih et al., 2016), we clip the reward between $[-1, 1]$. The learner applies a discount $\gamma = 0.99$ to calculate value function targets. The total loss function is a linear combination of policy loss L_{policy}, value function loss L_{value} and entropy regularization L_{entropy}, i.e., $L \triangleq L_{\text{policy}} + c_v L_{\text{value}} + c_e L_{\text{entropy}}$ where $c_v \triangleq 0.5$ and $c_e \triangleq 0.01$. All missing details are the same as the hyper-parameter setup of the IMPALA architecture to be introduced below.

The networks are optimized with a RMSProp optimizer (Tieleman and Hinton, 2012) with the learning rate $\alpha \triangleq 10^{-3}$.

H.6. Distributed off-policy policy optimization

V-trace implementations. V-trace is a strong baseline for correcting off-policy data (Espeholt et al., 2018). Given a partial trajectory $(x_t, y_t, r_t)_{t=1}^T$, let $\rho_t \triangleq \min\{\overline{\rho}, \pi(a_t|x_t)/\mu(a_t|x_t)\}$ be the truncated IS ratio. Let $V_\varphi(x)$ be a value function baseline. Define $\delta_t V \triangleq \rho_t(r_t + \gamma V(x_{t+1}) - V(x_t))$ be a temporal difference. V-trace targets are calculated recursively as

$$v(x_t) \triangleq V(x_t) + \delta_t V + \gamma c_t (v(x_{t+1}) - V(x_{t+1})), \tag{23}$$

where $c_t \triangleq \min\{\overline{c}, \pi(a_t|x_t)/\mu(a_t, x_t)\}$ is the trace coefficient. The value function baseline is then trained to approximate these targets $V_\varphi(x) \approx v(x)$.

The policy gradient is corrected by clipped IS ratio as well. The policy parameter θ is updated using the gradient

$$\min\{\overline{\rho}, \pi(a_t|x_t)/\mu(a_t|x_t)\} \nabla_\theta \log \pi(a_t|x_t) \hat{a}_t, \tag{24}$$

where the advantage estimates are $\hat{a}_t \triangleq r_t + \gamma v(x_{t+1}) - v(x_t)$ and the derivative ∇_θ is taken with respect to the learner parameter $\pi(a|x) = \pi_\theta(a|x)$. Following the original setup (Espeholt et al., 2018), we set $\overline{\rho} \triangleq \overline{c} \triangleq 1$.

Hyper-parameters for Taylor expansions. The Taylor expansion variants (including first-order and second-order expansions) all adopt the surrogate loss functions introduced in the main text. The second-order expansion requires a hyper-parameter η which we set to $\eta = 1$.

The value function targets are estimated as uncorrected cumulative returns, computed recursively $v(x_t) = r_t + \gamma v(x_{t+1})$ and then the value function baseline is trained to $V_\varphi(x) \approx v(x)$. Though adopting more complex estimation techniques such as GAE (Schulman et al., 2016) could potentially improve the accuracy of the bootstrapped values.

Additional results. Additional detailed results on Atari games are in Table 1 and Table 2. In both tables, we show the performance of different algorithmic variants (first-order, second-order, V-trace) across all Atari games after training for $400M$ frames. In Table 1, there is no artificial delay between actors and the learner, though there is still delay due to the computational setup across multiple machines. In Table 2, there is an artificial delay between actors and the learner.

Details on the distributed architecture. The general policy-based distributed agent follows the architecture design of IMPALA (Espeholt et al., 2018), i.e., a central GPU learner and $N \triangleq 512$ distributed CPU actors. The actors keep generating data by executing their local copies of the policy μ, and sending data to the queue maintained by the learner. The parameters are periodically synchronized between the actors and the learner.

The architecture details are the same the ones of Espeholt et al. (2018). For completeness, we give some important details below; please refer to the original paper for the full description. For the delay experiments (Figure 3), we used two different model architectures: a shallow model based on work of Mnih et al. (2016) with an LSTM between the torso embedding and the output of policy/value function. The deep model refers to a deep network model with residual network (He et al., 2016). See Figure 3 of (Espeholt et al., 2018) for details, in particular the layer size and activation's functions.

The policy/value function networks are both trained with RMSProp optimizers (Tieleman and Hinton, 2012) with learning rate $\alpha \triangleq 5{\cdot}10^{-4}$ and no momentum. To encourage exploration, the policy loss is augmented by an entropy regularization term with coefficient $c_e \triangleq 0.01$ and a baseline loss with coefficient $c_v \triangleq 0.5$, i.e., the full loss is $L \triangleq L_{\text{policy}} + c_v L_{\text{value}} + c_e L_{\text{entropy}}$. These single hyper-parameters are selected according to Appendix D of Espeholt et al. (2018).

Actors send partial trajectories of length $T \triangleq 20$ to the learner. For robustness of the training, rewards r_t are clipped between $[-1, 1]$. We adopt frame stacking and sticky actions as Mnih et al. (2013). The discount factor is $\gamma \triangleq 0.99$ for calculating the baseline estimations.

H.7. Distributed value-based learning

Hyper-parameters for Taylor expansions. The algorithmic details (e.g., the expression for recursive updates) are specified in Appendix G. Given a partial trajectory, the zero-order variant calculates the targets recursively along the entire trajectory. For first-order and second-order variants, we find that calculating the targets recursively along the entire trajectory tends to destabilize the updates. We suspect that this is because the function approximation error accumulates along the recursive computation, leading to very poor estimates at the beginning of the partial trajectory. Note that this is very different from update rules such as Retrace (Munos et al., 2016), where the trace coefficient $c_t \triangleq \lambda \min\{\bar{c}, \pi_t/\mu_t\}$ tends to be zero frequently because π_t is a greedy policy, traces are cut automatically and function approximation errors do not accumulate as much along the trajectory. For Taylor expansion variants with order $K \geq 2$, the trace coefficient is effectively $c_t \triangleq 1$ and the trace is not cut at all. To remedy such an issue, we compute corrected n-step updates with $n \triangleq 3$. This ensures that the errors do not propagate up to n steps and stabilize the learning process.

Importantly, we note that the accumulation of errors along trajectories might also happen for policy-based algorithms. However, we speculate that policy-based agents are more robust to such errors because it is the relative values which influence the direction of policy updates. See Appendix H.6 for details on policy-based algorithms.

In the experiments, we found $\eta = 0.2$ to work the best. This best hyper-parameter was selected across $\eta \in \{0, 0.2, 0.5, 0.8, 1.0\}$ where $\eta = 0$ corresponds to the first-order. Note that this best hyper-parameter differs from those of previous experiments with policy-based agents. This means that carrying out the full second-order expansion does not outperform the first-order; the best outcome is obtained in the middle.

Additional results. We provide additional results on Atari games in Figure 7, where in order to present a more complete picture of the training properties of different algorithmic variants, we provide mean/median/super-human ratio of the human-normalized scores. At each point of the training (e.g., fixing a number of training frames), we have access to the full set of human-normalized scores $z_i, 1 \leq i \leq 57$. Then, the three statistics are computed as usual across these scores. The super-human ratio is computed as the proportion of games such that $z_i > 1$, i.e., such that the learning algorithm reaches super-human performance.

Overall, we see that the second-order expansion provides benefits in terms of the mean performance. In median performance, first-order and second-order are very similar, both providing a slight advantage over Retrace. Across these two statistics, the zero-order achieves the worst results, since the performance plateaus at a low level. However, the super-human ratio

Levels	Random	Human	V-trace	First-order	Second-order (TayPO-2)
ALIEN	227.75	7127.8	11358	5004	**9634**
AMIDAR	5.77	**1719.53**	1442	1368	1350
ASSAULT	222.39	742	13759	9930	**11505**
ASTERIX	210	8503.33	135730	152980	**170490**
ASTEROIDS	719.1	**47388.67**	29545	35385	44015
ATLANTIS	12850	29028.13	711170	**724230**	700410
BANK_HEIST	14.2	753.13	1188	1166	**1218**
BATTLE_ZONE	2360	**37187.5**	13370	13828	13755
BEAM_RIDER	363.88	16926.53	**24031**	18798	23735
BERZERK	123.65	**2630.42**	1292	1383	1347
BOWLING	23.11	**160.73**	50	50	53
BOXING	0.05	12.06	99	99	99
BREAKOUT	1.72	30.47	551	580	**637**
CENTIPEDE	2090.87	**12017.04**	10166	8773	7747
CHOPPER_COMMAND	811	7387.8	**19256**	17129	17776
CRAZY_CLIMBER	10780.5	35829.41	**139190**	132670	134310
DEFENDER	2874.5	18688.89	73020	72658	133090
DEMON_ATTACK	152.07	1971	119130	117860	**133030**
DOUBLE_DUNK	-18.55	-16.4	-7.6	**-7.4**	-8.5
ENDURO	0	**860.53**	0	0	0
FISHING_DERBY	-91.71	-38.8	**33**	32	31.4
FREEWAY	0.01	**29.6**	0	0	0
FROSTBITE	65.2	**4334.67**	302	298	302
GOPHER	257.6	2412.5	23232	20805	**26123**
GRAVITAR	173	**3351.43**	373	386	430
HERO	1026.97	30826.38	32757	33277	**36639**
ICE_HOCKEY	-11.15	0.88	0.7	1.6	**4.3**
JAMESBOND	29	302.8	**759**	548	693
KANGAROO	52	**3035**	1147	1339	1181
KRULL	1598.05	2665.53	9545	8408	**9971**
KUNG_FU_MASTER	258.5	22736.25	**44920**	33004	41516
MONTEZUMA_REVENGE	0	**4753.33**	0	0	0
MS_PACMAN	307.3	6951.6	4018	4982	**9702**
NAME_THIS_GAME	2292.35	8049	**18084**	12345	13316
PHOENIX	761.4	7242.6	**148840**	91040	94131
PITFALL	-229.44	**6463.69**	-5.9	-4.2	-4.5
PONG	-20.71	14.59	21	21	21
PRIVATE_EYE	24.94	**69571.27**	100	94	99
QBERT	163.88	13455	16044	20862	**20891**
RIVERRAID	1338.5	17118	**24116**	22151	21253
ROAD_RUNNER	11.5	7845	39513	**43974**	38177
ROBOTANK	2.16	**11.94**	7.2	7.1	7
SEAQUEST	68.4	**42054.71**	1731	1735	1743
SKIING	-17098.09	**-4336.93**	-10865	-13303	-10386
SOLARIS	1236.3	**12326.67**	2375	2263	2486
SPACE_INVADERS	148.03	1668.67	13503	**13544**	13171
STAR_GUNNER	664	10250	**265480**	190920	214580
SURROUND	-9.99	**6.53**	4.3	3.4	2.4
TENNIS	-23.84	-8.27	20.6	**22**	21.8
TIME_PILOT	3568	5229.1	28871	**32813**	32447
TUTANKHAM	11.43	167.59	243	**278**	277
UP_N_DOWN	533.4	11693.23	**193520**	163130	188190
VENTURE	0	**1187.5**	0	0	0
VIDEO_PINBALL	0	17667.9	**359610**	326060	315930
WIZARD_OF_WOR	563.5	4756.52	7302	5114	**7646**
YARS_REVENGE	3092.91	54576.93	81584	90581	**93680**
ZAXXON	32.5	9173.3	21635	21149	**25603**

Table 1. Scores across 57 Atari levels for experiments on general policy-optimization with distributed architecture with **no artificial delays** between actors and learner. We compare several alternatives for off-policy correction: V-trace, first-order and second-order. We also provide scores for random policy and human players as reference. All scores are obtained by training for 400M frames. Best results per game are highlighted in bold font.

Levels	Random	Human	V-trace	First-order	Second-order (TayPO-2)
ALIEN	227.75	7127.8	464	1820	**3257**
AMIDAR	5.77	**1719.53**	81	428	541
ASSAULT	222.39	742	1764	4868	**6490**
ASTERIX	210	8503.33	2151	**165170**	161800
ASTEROIDS	719.1	**47388.67**	2256	1329	3886
ATLANTIS	12850	29028.13	311111	543210	**621920**
BANK_HEIST	14.2	**753.13**	71	483	524
BATTLE_ZONE	2360	**37187.5**	9021	10481	13820
BEAM_RIDER	363.88	16926.53	7391	16769	**19030**
BERZERK	123.65	**2630.42**	631	757	826
BOWLING	23.11	**160.73**	40	36	50
BOXING	0.05	12.06	51	93	**95**
BREAKOUT	1.72	30.47	71	298	**387**
CENTIPEDE	2090.87	**12017.04**	8847	6545	6924
CHOPPER_COMMAND	811	7387.8	2340	4837	**8064**
CRAZY_CLIMBER	10780.5	**35829.41**	23745	63982	117830
DEFENDER	2874.5	18688.89	20594	18088	**34684**
DEMON_ATTACK	152.07	1971	36491	40324	**63758**
DOUBLE_DUNK	-18.55	-16.4	-11.7	-9.9	**-7.2**
ENDURO	0	**860.53**	0	0	0
FISHING_DERBY	-91.71	-38.8	-6.6	15.4	**15.7**
FREEWAY	0.01	**29.6**	0	0	0.01
FROSTBITE	65.2	**4334.67**	230	257	267
GOPHER	257.6	2412.5	1551	2213	**5376**
GRAVITAR	173	**3351.43**	263	300	351
HERO	1026.97	**30826.38**	2012	3452	12027
ICE_HOCKEY	-11.15	0.88	-1.5	-0.9	**1.01**
JAMESBOND	29	302.8	307	**406**	389
KANGAROO	52	**3035**	416	342	805
KRULL	1598.05	2665.53	5737	5416	**9101**
KUNG_FU_MASTER	258.5	22736.25	12991	12968	**23741**
MONTEZUMA_REVENGE	0	**4753.33**	0	0	0
MS_PACMAN	307.3	**6951.6**	960	2542	2763
NAME_THIS_GAME	2292.35	8049	13315	15510	15510
PHOENIX	761.4	7242.6	6538	16566	**32146**
PITFALL	-229.44	**6463.69**	-4.5	-4.5	-3.2
PONG	-20.71	14.59	-14	13	**18.1**
PRIVATE_EYE	24.94	**69571.27**	88	80	185
QBERT	163.88	**13455**	1155	8856	10578
RIVERRAID	1338.5	**17118**	4607	2632	5064
ROAD_RUNNER	11.5	7845	6404	16792	**36857**
ROBOTANK	2.16	**11.94**	6.2	5.5	8.07
SEAQUEST	68.4	**42054.71**	1884	1881	2283
SKIING	-17098.09	**-4336.93**	-27463	-11778	-22189
SOLARIS	1236.3	**12326.67**	2435	2269	2320
SPACE_INVADERS	148.03	1668.67	1029	2955	**4399**
STAR_GUNNER	664	10250	25622	27001	**51257**
SURROUND	-9.99	**6.53**	-8.4	-2.5	-0.74
TENNIS	-23.84	-8.27	-20	-8.84	**4.89**
TIME_PILOT	3568	5229.1	8963	**18295**	17884
TUTANKHAM	11.43	167.59	97	161	**172**
UP_N_DOWN	533.4	11693.23	18726	18693	**49468**
VENTURE	0	**1187.5**	0	0	0
VIDEO_PINBALL	0	17667.9	**28962**	210960	191240
WIZARD_OF_WOR	563.5	4756.52	4142	5234	**5349**
YARS_REVENGE	3092.91	**54576.93**	3375	26302	29403
ZAXXON	32.5	9173.3	6251	9040	**9359**

Table 2. Scores across 57 Atari levels for experiments on general policy-optimization with distributed architecture with **severe delays** between actors and learner. We compare several alternatives for off-policy correction: V-trace, first-order and second-order. We also provide scores for random policy and human players as reference. All scores are obtained by training for 400M frames. The performance across all algorithms generally degrade significantly compared to Table 1, the second-order degrades more gracefully than other baselines. Best results per game are highlighted in bold.

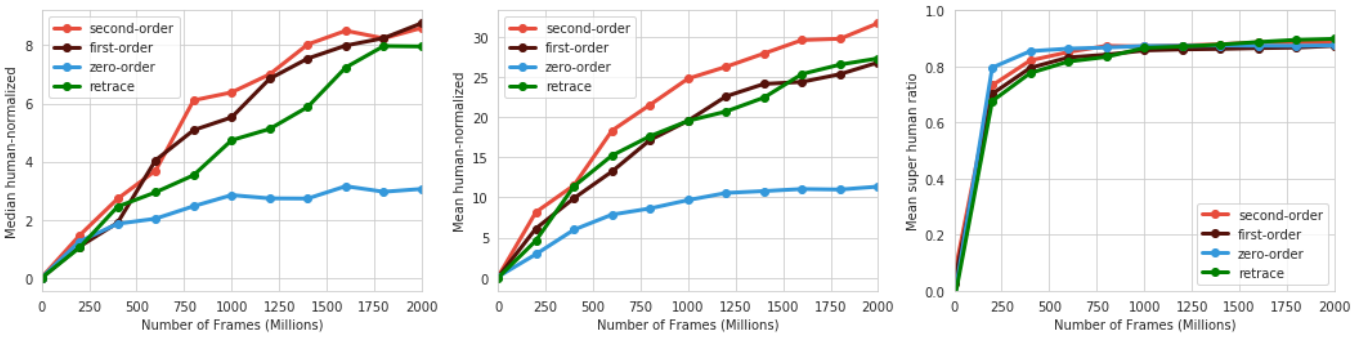

Figure 7. Value-based learning with distributed architecture (R2D2). The x-axis is number of frames (millions) and y-axis shows the mean/median/super-human ratio of human-normalized scores averaged across 57 Atari levels over the training of 2000M frames. Each curve averages across 2 random seeds. The second-order correction performs marginally better than first-order correction and retrace, and significantly better than zero-order. The super-human ratio is computed as the proportion of games with normalized scores $z_i > 1$.

statistics implies that the zero-order variant can achieve super-human performance on almost all games as quickly as other more complex variants.

Details on the distributed architecture. We follow the architecture designs of R2D2 (Kapturowski et al., 2019). We recap the important details for completeness. For a complete description, please refer to the original paper.

The agent contains a single GPU learner and 256 CPU actors. The policy/value network applies the same architecture as (Mnih et al., 2016), with a 3-layer convnet followed by an LSTM with 512 hidden units, whose output is fed into a dueling head (with hidden layer size of 512, Wang et al. 2016). Importantly, to leverage the recurrent architecture, each time step consists of the current observation frame, the reward and one-hot action embedding from the previous time step. Note that here we do no stack frames as practiced in e.g., IMPALA (Espeholt et al., 2018).

The actor sends partial trajectories of length $T \triangleq 120$ to the replay buffer. Here, the first $T_1 \triangleq 40$ steps are used for burn-in while the rest $T_2 \triangleq 80$ steps are used for loss computations. The replay buffer can hold $4 \cdot 10^6$ time steps and replays according to a priority exponent of 0.9 and IS exponent of 0.6 (Horgan et al., 2018). The actor synchronizes parameters from the learner every 400 environment time steps.

To calculate Bellman updates, we take a very high discount factor $\gamma = 0.997$. To stabilize the training, a target network is applied to compute the target values. The target network is updated every 2500 gradient updates of the main network. We also apply a hyperbolic transform in calculating the Bellman target (Pohlen et al., 2018).

All networks are optimized by an Adam optimizer (Kingma and Ba, 2014) with learning rate $\alpha \triangleq 10^{-4}$.

H.8. Ablation study

In this part we study the impact of the hyper-parameter η on the performance of algorithms derived from second-order expansion. In particular, we study the effect of η in the near on-policy optimization as in the context of Section 5.1. In Figure 8, x-axis shows the training frames (400M in total) and y-axis shows the mean human-normalized scores across Atari games. We select $\eta \in \{0.5, 1.0, 1.5\}$ and compare their training curves. We find that when η is selected within this range, the training performance does not change much, which hints on some robustness with respect to η. Inevitably, when η takes extreme values the performance degrades. When $\eta = 0$ the algorithm reduces to the first-order case and the performance gets marginally worse as discussed in the main text.

Value-based learning. The effect of η on value-based learning is different from the case of policy-based learning. Since the second-order expansion partially corrects for the value function estimates, its effect becomes more subtle for value-based algorithms such as R2D2. See discussions in Appendix G.

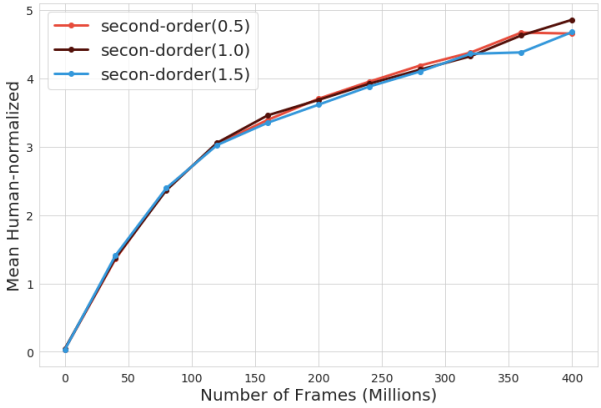

Figure 8. Ablation study on the effect of varying η. The x-axis shows the training frames (a total of 400M frames) and y-axis shows the mean human-normalized scores averaged across all Atari games. We select $\eta \in \{0.5, 1.0, 1.5\}$. In the legend, numbers in the brackets indicate the value of η.