A How Much Variance Reduction is Possible?

Recall in the main paper we consider a one-step MDP with action $a \in \mathbb{R}^d$. The reward function is $\alpha^T a$ for some $\alpha \in \mathbb{R}^d$. Consider a Gaussian policy with mean parameter μ and fixed covariance matrix $\Sigma = \sigma_2^2 \mathbb{I}$. The action is sampled as $a \sim \mathcal{N}(\mu, \Sigma)$. ES convolves the reward objective with a Gaussian with covariance matrix $\sigma_1^2 \mathbb{I}$. Let $\epsilon_1^{(i)}, \epsilon_2^{(i)} \sim \mathcal{N}(0, \mathbb{I}), 1 \le i \le N$ be N independent reparameterized noise, we can derive the vanilla ES estimator

$$\hat{g}_\mu^{\text{ES}} = \frac{1}{N} \sum_{i=1}^N \hat{g}_{\mu,i}^{\text{ES}} = \frac{1}{N} \sum_{i=1}^N \alpha^T (\mu + \sigma_1 \epsilon_1^{(i)} + \sigma_2 \epsilon_2^{(i)}) \frac{\epsilon_2^{(i)}}{\sigma_2}$$

The orthogonal estimator is constructed by N perturbations $\epsilon_{2,\text{ort}}^{(i)}$ such that $\langle \epsilon_{2,\text{ort}}^{(i)}, \epsilon_{2,\text{ort}}^{(j)} \rangle = 0$ for $i \neq j$, and each $\epsilon_2^{(i)}$ is still marginally d-dimensional Gaussian. The orthogonal estimator is

$$\hat{g}_\mu^{\text{ort}} = \frac{1}{N} \sum_{i=1}^N \hat{g}_{\mu,i}^{\text{ort}} = \frac{1}{N} \sum_{i=1}^N \alpha^T (\mu + \sigma_1 \epsilon_1^{(i)} + \sigma_2 \epsilon_{2,\text{ort}}^{(i)}) \frac{\epsilon_{2,\text{ort}}^{(i)}}{\sigma_2}$$

Finally, we consider the ES gradient estimator with control variate. In particular, we have the reparameterized gradient as

$$\hat{g}_\mu^{\text{RE}} = \frac{1}{N} \sum_{i=1}^N \hat{g}_{\mu,i}^{\text{RE}} = \frac{1}{N} \sum_{i=1}^N \alpha^T (\mu + \sigma_1 \epsilon_1^{(i)} + \sigma_2 \epsilon_2^{(i)}) \frac{\epsilon_1^{(i)}}{\sigma_1}$$

The general gradient estimator with control variate is

$$\hat{g}_\mu^{\text{CV}} = \hat{g}_\mu^{\text{ES}} + \eta \odot (\hat{g}_\mu^{\text{RE}} - \hat{g}_\mu^{\text{ES}})$$

where $\eta \in \mathbb{R}^d$. Since η can be indepdently chosen across dimensions, the maximal variance reduction is achieved by setting $\eta_i = -\frac{\text{cov}(X_i, Y_i)}{\mathbb{V}[Y_i]}$ where here $X = \hat{g}_\mu^{\text{ES}}, Y = \hat{g}_\mu^{\text{RE}} - \hat{g}_\mu^{\text{ES}}$.

Recall that for a vector g of dimension d, its variance is defined as the sum of the variance of its components $\mathbb{V}[g] = \sum_{i=1}^d \mathbb{V}[g_i]$. For simplicity, let $\rho = \frac{\sigma_2}{\sigma_1}$. We derive the variance for each estimator below.

Vanilla ES. For the vanilla ES gradient estimator, the variance is

$$\mathbb{V}[\hat{g}_\mu^{\text{ES}}] = \frac{d+1}{N} \|\alpha\|_2^2$$

Orthogonal ES. For the orthogonal ES gradient estimator, the variance is

$$\mathbb{V}[\hat{g}_\mu^{\text{ort}}] = \frac{(1+\rho^2)d + 2 - N}{N} \|\alpha\|_2^2$$

ES with Control Variate. For the ES gradient estimator with control variate, recall the above notation $X = \hat{g}_\mu^{\text{ES}}, Y = \hat{g}_\mu^{\text{RE}} - \hat{g}_\mu^{\text{ES}}$. We first compute $\rho(X_p, Y_p)^2 = \frac{\text{cov}^2(X_p, Y_p)}{\mathbb{V}[X_p]\mathbb{V}[Y_p]}$ for each component p. Let $X_{p,i}, Y_{p,i}$ be the pth component of $\hat{g}_{\mu,i}^{\text{ES}}$ and $\hat{g}_{\mu,i}^{\text{RE}} - \hat{g}_{\mu,i}^{\text{ES}}$ respectively. We will detail how to compute $\text{cov}(X_p, Y_p), \mathbb{V}[V_p]$ in the next section. With these components in hand, we have the final variance upper bound

$$\mathbb{V}[\hat{g}_\mu^{\text{CV}}] \le \mathbb{V}[\hat{g}_\mu^{\text{ES}}] \{1 - \frac{(1+\rho^2)[d((1+\rho^2) - 4]}{[(1+\rho^2)d + 1](2 + \rho^2 + \frac{1}{\rho^2})}\}.$$

B Derivation Details

Recall that for a vector g of dimensioon d, we define its variance as $\mathbb{V}[g] = \sum_{i=1}^d \mathbb{V}[g_i]$ For simplicity, recall that $\rho = \frac{\sigma_2}{\sigma_1}$.

B.1 Variance of Orthogonal ES

We derive the variance of orthogonal ES based on the formula in the Appendix of (Choromanski et al., 2018). In particular, we can easily compute the i sample estimate for the pth component of $X_{i,p} = [\hat{g}^{\text{ort}}_{\mu,i}]_p$

$$\mathbb{E}[X^2_{i,p}] = (1+\rho^2)\|\alpha\|_2^2 + \alpha_p^2$$

Hence the variance can be calculated as

$$\mathbb{V}[\hat{g}^{\text{ort}}_{\mu}] = \mathbb{V}[X] = \frac{(1+\rho^2)d + 2 - N}{N}\|\alpha\|_2^2$$

B.2 Variance of Vanilla ES

When we account for the cross product terms as in (Choromanski et al., 2018), we can easily derive

$$\mathbb{V}[\hat{g}^{\text{ES}}_{\mu}] = \mathbb{V}[X] = \frac{(1+\rho^2)d + 1}{N}\|\alpha\|_2^2.$$

We can also easily derive the variance per component $\mathbb{V}[X_p] = \frac{1}{N}((1+\rho)^2\|\alpha\|_2^2 + \alpha_p^2)$.

B.3 Variance of ES with Control Variate

Recall the definition $X_p = X^T e_p, Y_p = Y^T e_p$ where e_p is a one-hot vector with $[e_p]_i = \delta_{ip}$. For simplicity, we fix p and denote $x_i = X_{p,i}, y_i = X_{p,i} - Y_{p,i}$.

Step 1: Calculate $\mathbf{cov}(X_p, Y_p)$. The notation produces the covariance

$$\begin{aligned}\text{cov}(X_p, Y_p) &= \text{cov}(\frac{1}{N}\sum_{i=1}^N x_i, \frac{1}{N}\sum_{i=1}^N (x_i - y_i)) \\ &= \frac{1}{N^2}\mathbb{E}[\sum_{i,j} x_i x_j - x_i y_j].\end{aligned} \tag{10}$$

We identify some necessary components. Let $i \neq j$, then

$$\begin{aligned}\mathbb{E}[x_i^2] &= \mathbb{E}[(\alpha^T(\sigma_1\epsilon_1 + \sigma_2\epsilon_2)\frac{\epsilon_{1,p}}{\sigma_1})^2] \\ &= \mathbb{E}[(\alpha^T\epsilon_1)^2\epsilon_{1,p}^2 + (\alpha^T\epsilon_2)^2\rho^2] \\ &= (1+\rho^2)\|\alpha\|_2^2 + 2\alpha_p^2 \\ \mathbb{E}[x_i x_j] &= \mathbb{E}[x_i y_j] = \alpha_p^2 \\ \mathbb{E}[x_i y_i] &= \mathbb{E}[(\alpha^T(\sigma_1\epsilon_1 + \sigma_2\epsilon_2)^2\frac{\epsilon_{1,p}\epsilon_{2,p}}{\sigma_1\sigma_2}] \\ &= \mathbb{E}[2\alpha^T\epsilon_1\alpha^T\epsilon_2\epsilon_{1,p}\epsilon_{2,p}] = 2\alpha_p^2\end{aligned} \tag{11}$$

We can hence derive

$$\begin{aligned}\text{cov}(X_p, Y_p) &= \frac{1}{N^2}[\sum_{i=1}^N \mathbb{E}[x_i^2 - x_i y_i] + \sum_{i\neq j}\mathbb{E}[x_i x_j - x_i y_j]] \\ &= \frac{1}{N}[(1+\rho^2)\|\alpha\|_2^2 - 2\alpha_p^2]\end{aligned}$$

Step 2: Calculate $\mathbb{V}[Y_p]$. We only need to derive $\mathbb{E}[Y_{p,i}^2] = \mathbb{E}[(\alpha^T(\sigma_1\epsilon_1 + \sigma_2\epsilon_2)(\frac{\epsilon_{1,p}}{\sigma_1} - \frac{\epsilon_{2,p}}{\sigma_2})^2]$. After expanding all the terms, we can calculate

$$\mathbb{E}[(\alpha^T(\sigma_1\epsilon_1 + \sigma_2\epsilon_2)(\frac{\epsilon_{1,p}}{\sigma_1} - \frac{\epsilon_{2,p}}{\sigma_2})^2] = (2 + \rho^2 + \frac{1}{\rho^2})\|\alpha\|_2^2$$

Step 3: Combine all components. We finally combine all the previous elements into the main result on variance reduction. Assuming that the scaling factor of the control variate η is optimally set, the maximum variance reduction leads to the following resulting variance of component p. Using the above notations

$$\begin{aligned}\mathbb{V}[[\hat{g}_\mu^{\text{CV}}]_p] &= \mathbb{V}[X_p] - \frac{\text{cov}^2(X_p, Y_p)}{\mathbb{V}[Y_p]} \\ &= \frac{(1+\rho)^2\|\alpha\|_2^2 + \alpha_p^2}{N} - \frac{1}{N}\frac{[(1+\rho)^2\|\alpha\|_2^2 - 2\alpha_p^2]^2}{(2+\rho^2+\frac{1}{\rho^2})\|\alpha\|_2^2}.\end{aligned}$$

We can lower bound the right hand side and sum over d dimensions,

$$\begin{aligned}\mathbb{V}[\hat{g}_\mu^{\text{CV}}] = \sum_{p=1}^d \mathbb{V}[[\hat{g}_\mu^{\text{CV}}]_p] \le \mathbb{V}[X] - \\ \frac{d}{N}\frac{(1+\rho^2)^2}{2+\rho^2+\frac{1}{\rho^2}}\|\alpha\|_2^2 + \frac{4}{N}\frac{1+\rho^2}{2+\rho^2+\frac{1}{\rho^2}}\end{aligned}$$

Finally, we plug in $\mathbb{V}[X]$ and calculate the variance ratio with respect to the vanilla ES

$$\frac{\mathbb{V}[\hat{g}_\mu^{\text{CV}}]}{\mathbb{V}[\hat{g}_\mu^{\text{ES}}]} \le 1 - \frac{\rho^2[d((1+\rho^2)-4]}{[(1+\rho^2)d+1](1+\rho^2)}.$$

As a comparison, we can calculate the variance ratio of the orthogonal ES

$$\frac{\mathbb{V}[\hat{g}_\mu^{\text{ort}}]}{\mathbb{V}[\hat{g}_\mu^{\text{ES}}]} = \frac{(1+\rho^2)d+2-N}{(1+\rho^2)d+1}.$$

When does the control variate achieve lower variance? We set the inequality $\mathbb{V}[\hat{g}_\mu^{\text{ort}}] \ge \mathbb{V}[\hat{g}_\mu^{\text{CV}}]$ and calculate the following condition

$$\rho \ge \rho_0 = \sqrt{\frac{N+3-d+\sqrt{(d-N-3)^2+4(N-1)d}}{2d}}. \tag{12}$$

The expression (12) looks formidable. To simplify the expression, consider the limit $N \to \infty$ while maintaining $\frac{N}{d} \in [0,1]$. Taking this limit allows us to drop certain constant terms on the right hand side, which produces $\rho_0 = \sqrt{\frac{N}{d}}$.

C Additional Experiment Details

C.1 Updating Discount Factor γ

The discount factor γ is updated using ES methods. Specifically, at each iteration t, we maintain a current γ_t. The aim is to update γ_t such that the empirical estimate of $\mathbb{V}[\hat{g}_\theta^{\text{CV}}]$ is minimized. For each value of γ we can calculate a $\hat{g}_\theta^{\text{CV}}(\gamma)$, here we explicitly note its dependency on γ. For this iteration, we setup a blackbox function as $F(\gamma) = \sum_{i=1}^d [\hat{g}_\theta^{\text{CV}}(\gamma)]_i^2$ where d is the dimension of parameter θ. Then the ES update for γ is $\gamma_{t+1} = \gamma_t - \alpha_\gamma \hat{g}_\gamma$, where

$$\hat{g}_\gamma = \frac{1}{N_\gamma}\sum_{i=1}^{N_\gamma}\frac{F(\gamma_t + \sigma_\gamma \epsilon_i)}{\sigma_\gamma}\epsilon_i,\ \epsilon_i \sim \mathcal{N}(0,1). \tag{13}$$

As mentioned in the paper, to ensure $\gamma \in (0,1]$ we parameterize $\gamma = 1 - \exp(\phi)$ in our implementation. We optimize ϕ using the same ES scheme but in practice we set $\alpha_\phi, \sigma_\phi, N_\phi$. Here we have $\alpha_\phi \in \{10^{-4}, 10^{-5}, 10^{-6}\}$, $\sigma_\phi = 0.02$ and $N_\phi = 10$. Sometimes we find it effective to just set γ to be the initial constant, because the control variate scalar η is also adjusted to minimize the variance.

C.2 Normalizing the gradients

Vanilla stochastic gradient updates are sensitive to the scaling of the objective function, which in our case are the reward functions. Recall the vanilla ES estimator $\hat{g}_\theta^{\text{es}} = \frac{1}{N}\sum_{i=1}^{N} \frac{J(\pi_{\theta+\sigma\cdot\epsilon_i})}{\sigma}\epsilon_i$ where $J(\pi_\theta)$ is the return estimate under policy π_θ. To ensure that the gradient is properly normalized, the common practice (Salimans et al., 2017; Mania et al., 2018; Choromanski et al., 2018) is to normalize the returns $\tilde{J} \leftarrow \frac{J-\bar{J}}{\sigma(J)}$, where $\bar{J}, \sigma(J)$ are the mean/std of the estimated returns of the batch of N samples. The normalized returns $\tilde{J}$ are used in place of the original returns in calculating the gradient estimators. In fact, we can interpret the normalization as subtracting a moving average baseline (for variance reduction) and dividing by a rough estimate of the local Lipchitz constant of the objective landscape. These techniques tend to make the algorithms more stable under reward functions with very different scales.

The same technique can be applied to the control variate. We divide the original control variate by $\sigma(J)$ to properly scale the estimators.

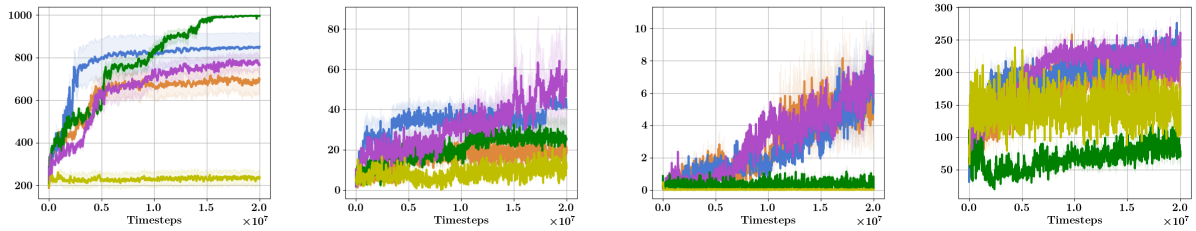
Details on policy gradient methods. PG methods are implemented following the general algorithmic procedure as follows: collect data using a previous policy $\pi_{\theta_{t-1}}$, construct loss function and update the policy parameter using gradient descent $\theta_t \leftarrow$ to arrive at π_{θ_t}, then iterate. We consider three baselines: vanilla PG, TRPO and PPO. In our implementation, these three algorithms differ in the construction of the loss function. For vanilla PG, the loss function is $L = -\mathbb{E}_{s,a}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}}(a|s)A(s,a)]$ where $A(s,a)$ is the advantage estimtion and θ_{old} is the prior policy iterate. For PPO, the loss function is $L = -\mathbb{E}_{s,a}[\text{clip}\{\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}}(a|s), 1-\epsilon, 1+\epsilon\}A(s,a)]$ where $\text{clip}\{x,a,b\}$ is to clip x between a and b and $\epsilon = 0.2$. In practice, we go one step further and implement the action dependent clipping strategy as in (Schulman et al., 2017). For TRPO, we implement the dual optimization objective of the original formulation (Schulman et al., 2015b) and set the loss function $L = -\mathbb{E}_{s,a}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}}(a|s)A(s,a)] + \eta\mathbb{E}_s\left[\mathbb{KL}[\pi_\theta(\cdot|s)||\pi_{\theta_{\text{old}}}(\cdot|s)]\right]$ where we select $\eta \in \{0.1, 1.0\}$.

When collecting data, we collect $2N$ full episodic trajectories as with the ES baselines. This is different from a typical implementation (Dhariwal et al., 2017), where PG algorithms collect a fixed number of samples per iteration. Also, we take only one gradient descent on the surrogate objective per iteration, as opposed to multiple updates. This reduces the policy optimization procedure to a fully on-line fashion as with the ES methods.

D Additional Experiments

D.1 Additional Baseline Comparison

Due to space limit, we omit the baseline comparison result on four control tasks from the main paper. We show their results in Figure 3. The setup is exactly the same as in the main paper.



(a) **Balance (DM)** (b) **Pendulum (DM)** (c) **HopperHop (DM)** (d) **AntWalk(DM)**

Figure 3: Training performance on Continuous Control Benchmarks: Swimmer, HalfCheetah, CartPole + {Swingup, TwoPoles, Balance}, Pendulum Swingup, Cheetah Run and Hopper. Tasks with DM stickers are from the DeepMind Control Suites. We compare five alternatives: baseline ES (orange), CV (blue, ours), ORTHO (marron), GCMC (green) and QMC (yellow). Each task is trained for $2 \cdot 10^7$ time steps (LQR is trained for $4 \cdot 10^6$ steps) and the training curves show the mean $\pm$ std cumulative rewards across 5 random seeds.

D.2 Discussion on Policy Gradient Methods

Instability of PG. It has been observed that the vanilla PG estimators are not stable. Even when the discount factor $\gamma < 1$ is introduced to reduce the variance, vanilla PG estimators can still have large variance due to the long horizon. As a result in practice, the original form of the PG (5) is rarely used. Instead, prior works and practical implementations tend to introduce bias into the estimators in exchange for lower variance: e.g. average across states intsead of trajectories (Dhariwal et al., 2017), clipping based trust region (Schulman et al., 2017) and biased advantage estimation (Schulman et al., 2016). These techniques stabilize the estimator and lead to state-of-the-art performance, however, their theoretical property is less clear (due to their bias). We leave the study of combining such biased estimators with ES as future work.

D.3 Additional Results with TRPO/PPO and Deterministic Policies

We provide additional comparison results against TRPO/PPO and deterministic policies in Table 3. We see that generally TRPO/PPO achieve better performance, yet are still under-performed by CV. On the other hand, though deterministic policies tend to outperform stochastic policies when combined with the ES baselines, they are not as good as stochastic policies with CV.

Comparison of implementation variations of PG algorithms. Note that for fair comparison, we remove certain functionalities of a full fledged PG implementation as in (Dhariwal et al., 2017). We identify important differences between our baseline and the full fledged implementation and evaluate their effects on the performance in Table 4. Our analysis focuses on PPO. These important implementation differences are

- **S**: Normalization of observations.
- **A**: Normalization of advantages.
- **M**: Multiple gradient updates (in particular, 10 gradient updates), instead of 1 for our baseline.

To understand results in Table 4, we start with the notations of variations of PPO implementations. The PPO denotes our baseline for comparison; the PPO+S denotes our baseline with observation normalization. Similarly, we have PPO+A and PPO+M. Notations such as PPO+S+A denote the composition of these implementation techniques. We evaluate these implementation variations on a subset of tasks and compare their performance in Table 4.

We make several observations: **(1)** Multiple updates bring the most significant for PG. While conventional ES implementations only consider one gradient update per iteration (Salimans et al., 2017), it is likely that they coudl also benefit from this modification; **(2)** Even when considering these improvements, our proposed CV method still outperforms PG baselines on a number of tasks.

E Discussion on Scalability

ES easily scale to a large distributed architecture for the trajectory collection (Salimans et al., 2017; Mania et al., 2018). Indeed, the bottleneck of most ES applications seem to be the sample collection, which can be conveniently distributed across multiple cores (machines). Fortunately, many open source implementations have provided high-quality packages via efficient inter-process communications and various techniques to reduce overheads (Salimans et al., 2017; Mania et al., 2018).

On the other hand, distributing PG requires more complicated software design. Indeed, distributed PG involve both distributed sample collection and gradient computations. The challenge lies in how to construct gradients via multiple processes and combine them into a single update, without introducing costly overheads. Open source implementations such as (Dhariwal et al., 2017) have achieved synchronized distributed gradient computations via MPI.

To scale CV, we need to combine both distributed sample collection from ES and distributed gradient computation from PG. Since both of the above two components have been implemented with high quality via open source packages, we expect it not to be a big issue to scale CV, in particular to combine scalable gradient computation

Table 3: Final performance on benchmark tasks. The policy is trained for a fixed number of steps on each task. The result is mean ± std across 5 random seeds. The best results are highlighted in bold font. We highlight multiple methods if their results cannot be separated (mean ± std overlap). CV (ours) achieves consistent gains over the baseline and other variance reduction methods. We also include a PG baseline.

Tasks	**PPO**	**TRPO**	**Det**	**CV (Ours)**	**PG**
Swimmer	11 ± 7	11 ± 7	191 ± 100	$\mathbf{237 \pm 33}$	-132 ± 5
HalfCheetah	-175 ± 20	-174 ± 16	1645 ± 1298	$\mathbf{1897 \pm 232}$	-180 ± 4
Walker	657 ± 291	657 ± 292	1588 ± 744	$\mathbf{1476 \pm 112}$	282 ± 25
Pong(R)	-17.1 ± 0.4	-15.0 ± 2.6	-10.4 ± 5.4	$\mathbf{-3.0 \pm 0.3}$	-17 ± 0.2
HalfCheetah(R)	14 ± 2	13 ± 3	502 ± 199	$\mathbf{709 \pm 16}$	12 ± 0
BipedalWalker	-66 ± 39	-66 ± 38	1 ± 2	$\mathbf{105 \pm 40}$	-82 ± 12
Cheetah(DM)	20 ± 20	36 ± 32	241 ± 47	$\mathbf{296 \pm 15}$	25 ± 6
Pendulum(DM)	0.6 ± 0.5	7.5 ± 4.8	40 ± 16	$\mathbf{43 \pm 1}$	3 ± 1
TwoPoles(DM)	264 ± 46	42 ± 52	190 ± 58	$\mathbf{245 \pm 29}$	14 ± 1
Balance(DM)	264 ± 46	515 ± 42	692 ± 250	$\mathbf{847 \pm 71}$	401 ± 12
HopperHop(DM)	0.2 ± 0.2	0.0 ± 0.0	4.6 ± 6.2	$\mathbf{6.5 \pm 1.5}$	0.1 ± 0.0
AntWalk(DM)	96 ± 36	180 ± 41	192 ± 20	$\mathbf{239 \pm 10}$	100 ± 11
AntEscape(DM)	6 ± 3	10 ± 1	13 ± 8	$\mathbf{51 \pm 2}$	6 ± 1

Table 4: Evaluation of implementation variations of PG methods, with comparison to the CV. Below show the final performances on benchmark tasks. The policy is trained for a fixed number of steps on each task. The result below shows the mean performance averaged across 3 random seeds. The best results are highlighted in bold font. Notice below the table is transposed due to space.

Tasks	Swimmer	HalfCheetah	Pendulum	Balance	Hopper	AntWalk
PPO	11	−175	0.6	264	0.2	96
PPO+S	30	126	2	308	0.5	80
PPO+A	33	67	26	278	0.4	117
PPO+S+A	31	192	1	386	0.6	106
PPO+S+M	110	1579	36	652	0.3	180
PPO+A+M	51	**2414**	**223**	771	46	173
PPO+S+A+M	106	1528	17	750	**46.4**	144
CV(Ours)	**237**	1897	43	**847**	6.5	**239**

with ES. However, this requires additional efforts - as these two parts have never been organically combined before and there is so far little (open source) engineering effort into this domain. We leave this as future work.