DNS: Determinantal Point Process Based Neural Network Sampler for Ensemble Reinforcement Learning

Hassam Ullah Sheikh [*1] **Kizza Nandyose Frisbee** [*2] **Mariano Phielipp** [1]

Abstract

The application of an ensemble of neural networks is becoming an imminent tool for advancing state-of-the-art deep reinforcement learning algorithms. However, training these large numbers of neural networks in the ensemble has an exceedingly high computation cost which may become a hindrance in training large-scale systems. In this paper, we propose DNS: a **D**eterminantal Point Process based **N**eural Network **S**ampler that specifically uses k-DPP to sample a subset of neural networks for backpropagation at every training step thus significantly reducing the training time and computation cost. We integrated DNS in REDQ for continuous control tasks and evaluated on MuJoCo environments. Our experiments show that DNS augmented REDQ matches the baseline REDQ in terms of average cumulative reward and achieves this using less than 50% computation when measured in FLOPS. The code is available at https://github.com/IntelLabs/DNS.

1. Introduction

In the past decade, reinforcement learning (RL) algorithms powered by high-capacity function approximators such as deep neural networks have been used to master complex sequential decision problems such as Atari games (Mnih et al., 2015), boards games like Chess, Go, and Shogi (Silver et al., 2016; 2017; 2018) and robotic manipulation (Liu et al., 2021). Despite showing impressive results, deep reinforcement learning (DRL) algorithms have a number of problems such as sample inefficiency (Łukasz Kaiser et al., 2020), overestimation bias (van Hasselt, 2010; Hado van Hasselt et al., 2016; Lan et al., 2020; Anschel et al., 2017; Fujimoto et al., 2018) and an imbalance between exploration and exploitation (Lee et al., 2020; Osband et al., 2016).

Considering the success of ensembles in supervised learning, the use of an ensemble of neural networks is becoming popular in deep reinforcement learning (DRL) to address the issues mentioned above. For example, in (Lan et al., 2020; Anschel et al., 2017; Fujimoto et al., 2018) have used ensemble to address the overestimation bias problem. In (Chen et al., 2021) proposed REDQ that uses an ensemble with a high update-to-data ratio to address the sample inefficiency problem. Similarly (Lee et al., 2020) have used ensemble for efficient exploration.

Despite ensembles providing elegant theoretical and practical solutions, they introduce new practical problems such as high computation costs and long training times. The high computation cost problem is especially evident in an actor-critic setting where DRL algorithms use a large number of critic networks. One such example is the REDQ algorithm that uses ten critic networks and updates all of them in every training step which leads to higher computation cost as well as longer wall-clock time.

To address this issue, we present DNS: a **D**eterminantal Point Process-based **N**eural Network **S**ampler that specifically uses k-DPP (Kulesza & Taskar, 2011) to sample a subset of neural networks for backpropagation at every training step. DNS uses Centered Kernel Alignment (CKA) (Kornblith et al., 2019) values to form the similarity matrix which is then used by the k-DPP to sample the subset on neural networks for backpropagation. The motivation for sampling a subset of networks came from a hypothesis which we show in Section 4.1 that the Q-values from the critics converge prematurely during training thus eliminating the need of training all the critics at every training step.

Additionally, we show that if the CKA matrix is not positive semi-definite, the closest positive semi-definite matrix is just a diagonal perturbation of the CKA matrix and its resulting kernel matrix is still Hermitian positive semi-definite.

We applied DNS on REDQ, evaluated on MuJoCo environments (Todorov et al., 2012) and, showed that a simple sampling technique can significantly reduce the training time and computation cost while maintaining performance as if training all the networks in the ensemble.

[*]Equal contribution [1]Intel Labs [2]Intel Corporation. Correspondence to: Hassam Sheikh <hassam.sheikh@intel.com, hassamsheikh1@gmail.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

To summarize, our contributions are the following:

1. We empirically show that neural network-based value-function approximators collapse prematurely during training in ensemble reinforcement learning.
2. To address this issue, we propose a Determinantal Point Process based Neural Network Sampler that samples a subset of value-function approximators for backpropagation at every training step.
3. We apply DNS on REDQ, which uses an ensemble of **ten** critic networks. Our experiments show that DNS sampling achieves similar or better results than REDQ using less than 50% computation when measured in FLOPS.
4. We also provide a theoretical analysis and proof that shows that k-DPP sampling of action-value functions leads to lower action-value minimization variance than random sampling k action-values. Additionally, we show how sufficiency conditions for k-DPP sampling can easily be met for the Deep RL use case.

2. Related Work

Ensembles in Deep Reinforcement Learning: The application of an ensemble of neural networks in deep reinforcement learning has been studied in several recent studies for different purposes. In (Fujimoto et al., 2018; Anschel et al., 2017; Lan et al., 2020; Chen et al., 2021) have used an ensemble to address the overestimation bias in deep Q-learning based methods for both continuous and discrete control tasks. Most recently proposed TOP (Moskovitz et al., 2021) proposed a method that learns to balance optimistic and pessimistic value estimation online. Similarly, Bootstrapped DQN and extensions (Osband et al., 2016; Chen et al., 2017) have used an ensemble of neural networks for efficient exploration. In (Chen et al., 2021; Gupta, 2015) have used a large number of ensembles to provide sample efficient reinforcement learning algorithms. The use of ensemble is rapidly growing in offline reinforcement learning to address issues such as error propagation and uncertainty estimations. The error propagation problem in offline reinforcement learning is addressed in (Kumar et al., 2019) using ensemble. Recently proposed methods such as (An et al., 2021; Ghasemipour et al., 2022) have used a large number of ensembles to measure uncertainty in offline RL settings. The application of ensemble is not only limited to the critics but several recent papers have used ensembles in the policy domain as well (Lee et al., 2020; Zhang & Yao, 2019).

Determinantal Point Process in Machine Learning: Determinantal Point Processes (DPPs) have emerged as powerful models in the machine learning community in applications requiring information diversity, coverage, and to reduce redundancy such as text summarization (Kulesza & Taskar, 2012). Applications of DPPs include video summarization (Gong et al., 2014; Sharghi et al., 2016), pose estimation (Mai et al., 2022) and wardrobes creation (Hsiao & Grauman, 2018). More recently DPPs have been used in reinforcement learning to promote behavior diversity (Pacchiano et al., 2020). k-DPPs (Kulesza & Taskar, 2011), an extension of DPP has been adopted in many applications such as image search and stochastic gradient descent using diversified fixed size mini-batches (Zhang et al., 2017).

3. Background

REDQ: REDQ (Chen et al., 2021) is an off-policy actor-critic method based on max-min RL framework. REDQ uses an ensemble of neural networks to model the critic. One key feature of REDQ is in-target minimization that samples a subset of neural networks to create the target value to train the critics networks. The target value y is calculated as

$$y = r + \gamma \left(\min_{i \in \mathcal{M}} Q_{\phi_{\text{targ},i}} \left(s', \tilde{a}'\right) - \alpha \log \pi_\theta \left(\tilde{a}' \mid s'\right) \right),$$

where $\mathcal{M}$ is the number of target networks. The policy gradient is written as

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i} \left(s, \tilde{a}_\theta(s)\right) - \alpha \log \pi_\theta \left(\tilde{a}_\theta(s) | s\right) \right),$$

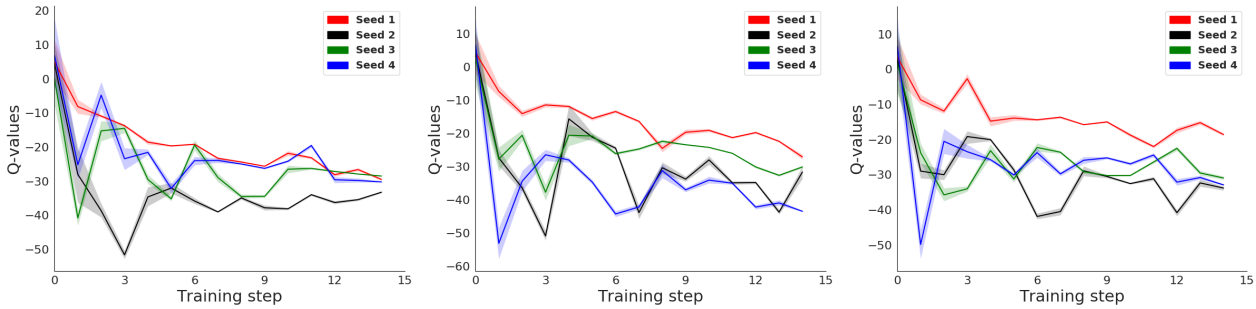
where N is the number of critic networks.

Centered Kernel Alignment: Centered Kernel Alignment (CKA) (Cristianini et al., 2002; Cortes et al., 2012; Kornblith et al., 2019) is an invertible linear transformation invariant statistic for measuring meaningful multivariate similarity between representations of higher dimension. CKA is a normalized form of Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005). Formally, CKA is defined as:

Let $\boldsymbol{X} \in \mathbb{R}^{n \times p_1}$ denote a matrix of activations of p_1 neurons for n examples and $\boldsymbol{Y} \in \mathbb{R}^{n \times p_2}$ denote a matrix of activations of p_2 neurons for the same n examples. Furthermore, we consider $\boldsymbol{A}_{ij} = a\left(x_i, x_j\right)$ and $\boldsymbol{B}_{ij} = b\left(y_i, y_j\right)$ where k and l are two kernels.

$$\text{CKA}\left(\boldsymbol{A}, \boldsymbol{B}\right) = \frac{\text{HSIC}\left(\boldsymbol{A}, \boldsymbol{B}\right)}{\sqrt{\text{HSIC}\left(\boldsymbol{A}, \boldsymbol{A}\right) \cdot \text{HSIC}\left(\boldsymbol{B}, \boldsymbol{B}\right)}}$$

HSIC is a test statistic for determining whether two sets of variables are independent. The empirical estimator of HSIC



(a) Ant-v2 (b) HalfCheetah-v2 (c) Walker-v2

Figure 1. Q-value plots of three MuJoCo environments accumulated over four different seeds. Each curve in the plot represents the mean of the Q-values from ten critics and shaded area represents 95% confidence interval band. Notice that each curve has a variance in the Q-values in the beginning of the training but it quickly disappears as training continues.

is defined as:

$$\text{HSIC}\,(\boldsymbol{A}, \boldsymbol{B}) = \frac{1}{(n-1)^2}\,\text{Tr}\,(\boldsymbol{AHBH})$$

where H is the centering matrix $H_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$.

Determinantal Point Processes: A Determinantal point process (DPP) (Macchi, 1975) is a random point process useful for the combinatorial problem of selecting a diverse sample from a set. In particular, a DPP for a given finite set defines a probability distribution over subsets, where subsets containing diverse items have high probability and are thus more likely to be selected. We briefly discuss finite determinantal point processes here, for in-depth discussions refer (Hough et al., 2006; Kulesza & Taskar, 2012; Li et al., 2016; Dereziński et al., 2019).

Definition 3.1. A point process $\boldsymbol{X}$ on discrete set $\mathcal{S}$ and with Hermitian positive semi-definite marginal kernel $\boldsymbol{K}$:$\mathcal{S}^2 \to \mathbb{C}$, $\boldsymbol{K} \preceq 1$ (all eigenvalues of $\boldsymbol{K}$ are at most 1) is called *determinantal* iff

$$P(\boldsymbol{X} \supseteq (x_i, \ldots, x_n)) = det(\boldsymbol{K}(x_i, x_j))_{1 \leq i,j \leq n} \quad (1)$$

for any $n \in \mathbb{Z}^+$ and any $x_i, \ldots, x_n \in \mathcal{S}$ or equivalently, iff

$$P(\boldsymbol{X} \supseteq x) = det(\boldsymbol{K}_x) \quad (2)$$

for any $x \subset \mathcal{S}$, where $\boldsymbol{K}_x$ is the submatrix of $\boldsymbol{K}$ indexed by $x \times x$.

Consequently, DPPs are a repulsive distribution over set $\mathcal{S}$, generating subsets that exhibit diversity.

Furthermore, for the case when $\boldsymbol{I} - \boldsymbol{K}$ invertible, the DPP $\boldsymbol{X}$ is called an L-ensemble with kernel $\boldsymbol{K} := \boldsymbol{I} - (I + \boldsymbol{L})^{-1}$ and distribution

$$P(\boldsymbol{X} = x) = det(\boldsymbol{L}_x) det(\boldsymbol{I} + \boldsymbol{L})^{-1} \quad (3)$$

for any $x \subset \mathcal{S}$, where $\boldsymbol{L}_x$ is the sub-matrix of $\boldsymbol{L}$ indexed by $x \times x$.

Lemma 1. *(Collings, 1983) Let D be an $N \times N$ diagonal matrix and let M be an arbitrary $N \times N$ matrix. The determinant of $(D + M)$ is :*

$$det(\boldsymbol{D} + \boldsymbol{M}) = \sum_{S \subseteq \mathcal{S}} det(\boldsymbol{D}_S) det(\boldsymbol{M}_S). \quad (4)$$

Thus (3) can be re-written in normalized form as:

$$P(\boldsymbol{X} = x) = det(\boldsymbol{L}_x)(\sum_{x \subseteq \mathcal{S}} det(\boldsymbol{L}_x))^{-1}. \quad (5)$$

In this paper, we only considers DPPs that are L-ensembles because of their advantages such as:

i. While (2) gives the probability that a set is contained in the DPP, (3) gives the exact probability that a sampled set is from the DPP and is thus more relevant for set selection tasks requiring samples from different regions in a feature space. From (3), more diverse sets have higher probability and are thus more likely to be selected.

ii. There is no requirement that all eigenvalues of $\boldsymbol{L}$ are less than or equal to 1.

Since standard DPP sampling does not provide the flexibility of sampling a pre-specified size, in this work we focus on k-Determinantal Point Processes (k-DPPs). A k-DPP on discrete set $\mathcal{S}$ is a distribution over all subsets of $\mathcal{S}$ with cardinality k and is thus a conditioning of a standard DPP on the event that a subset X of $\mathcal{S}$ has a fixed size. A k-DPP thus gives probabilities

$$P^{(\boldsymbol{k})}(\boldsymbol{X} = x) = det(L_x)(\sum_{\substack{x \subseteq \mathcal{S} \\ |x| = k}} det(\boldsymbol{L}_x))^{-1}, \quad (6)$$

where $|x| = k$ and $\boldsymbol{L}$ is a positive semi-definite kernel.(Kulesza & Taskar, 2011) Because k-DPPs only model contents of a set, they are less costly than standard DPPs and are useful in situations where sample size is constrained, for example by empirical bounds or hardware restrictions (Zhang et al., 2017).

4. DNS: Determinantal Point Process Based Neural Network Sampler

In this section, we propose DNS: **D**eterminantal Point Process Based **N**eural Network **S**ampler that samples a subset of critic networks for backpropagation at training time. In principal DNS can be used with any off-policy algorithm that uses the same target value to train the critics such as REDQ (Chen et al., 2021), TOP (Moskovitz et al., 2021) MaxminDQN (Lan et al., 2020), EnsembleDQN (Anschel et al., 2017). For the exposition, we describe only the REDQ version in this paper.

This section is organized as follows:

1. We *empirically* show that the Q-values from the critic networks collapse prematurely during training time.
2. we present the k-DPP based sampling algorithm to sample the indices for critic networks to train.

4.1. Empirical Evidence of Early Collapse of Q-values

The work on this paper starts with a conjecture that the Q-values from the critic networks collapse prematurely. To verify our hypothesis, we trained REDQ on HalfCheetah-v2, Ant-v2, and Walker2d-v2 on four different seeds and measured the Q-values from all the ten critics. As shown in Figure 1, it took around fifteen training steps for all the ten critics from having distinct Q-values to collapse to almost identical values in nearly every run for all three environments. Note that each curve in the plot represents the mean value of all ten critics and the shaded area around the curve represents a 95% confidence interval.

A counterargument can be made here that in Figure 1 we did not allow enough training steps that might induce any variance in the Q-values. To address this, we measured the Q-values at the tail end of the training. As shown in Figure 2, the Q-values have completely collapsed in all the runs across all three environments. From this evidence, we can conclude that longer training reduces the variance in the Q-values.

4.2. Compute Efficient Neural Network Sampling

The motivation behind the idea of training a subset of critic networks came from the observation in Section 4.1 that if all the critic networks collapse early in the training, we can only train a subset of critic networks at every training step. This will allow us to force diversity in the Q-values which recently have been shown to be a key component in ensemble reinforcement learning (Sheikh et al., 2022). To sample a diverse set of critics, we use k-DPP (Kulesza & Taskar, 2011) which is a derivative of DPPs (Macchi, 1975). The advantage of k-DPP over DPP is that k-DPP allows us to have control over the size of the sampled neural networks while standard DPP automatically selects the size of the subset. One key component required for using the k-DPP is a similarity matrix. Since we are interested in sampling critic networks with diverse Q-values, we created the similarity matrix by measuring the pairwise CKA similarity of all the Q-values. Formally, the similarity matrix $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ is defined as:

$$\boldsymbol{L} = \text{CKA}((Q_{\phi_i}(s,a), Q_{\phi_j}(s,a))_{1 \leq i,j \leq N}. \tag{7}$$

The similarity matrix $\boldsymbol{L}$ is used by the k-DPP to sample the indices of the diverse critics to train.

Formally, we consider a REDQ agent with N number of critic networks parameterized by $\{\phi_i\}_{i=1}^{N}$. At every training step, we sample a batch of experience B from replay buffer $\mathcal{D}$. Using state-action $(s,a) \in B$, we fetch the Q-values $Q_{\phi_i}(s,a)$ for $i = 1, 2, \ldots, N$. Using the Q-values, we create the similarity matrix $\boldsymbol{L}$ by measuring the pairwise CKA similarity using Equation (7). The similarity matrix $\boldsymbol{L}$ is then used by k-DPP to sample a diverse set of critic networks of size k to train. The rest of the training process is identical to the baseline REDQ which we invite the readers to see in Algorithm 1. *Note that the output of the k-DPP is indices of the critic networks.*

4.3. Formal Theoretical Analysis

In DNS we utilize CKA values as entries of the similarity matrix $\boldsymbol{L} \in \mathbb{R}^{N \times N}$:

$$\boldsymbol{L} = \text{CKA}((Q_{\phi_i}(s,a), Q_{\phi_j}(s,a))_{1 \leq i,j \leq N}.$$

Since not all similarity matrices are positive semi-definite, $\boldsymbol{L}$ can be approximated with the closest positive semi-definite matrix such that the relative similarity strengths among point pairs are preserved.

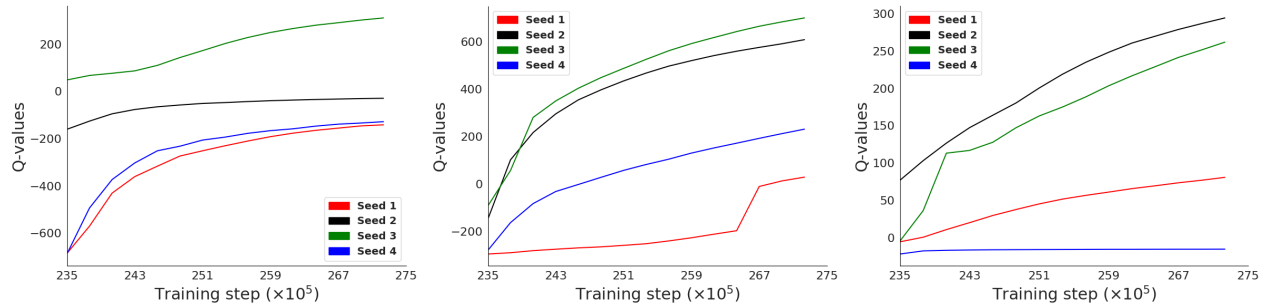
Proposition 1. *The nearest positive semi-definite matrix to a symmetric matrix to $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ is a diagonal perturbation of $\boldsymbol{L}$:*

$$\widetilde{\boldsymbol{L}} = \boldsymbol{L} + \boldsymbol{D},$$

where $\boldsymbol{D} = diag((\lambda_i + |\lambda_i|)/2)$, and $\lambda_i, i \in \{1, \ldots, N\}$ are eigenvalues from the spectral decomposition of $\boldsymbol{L}$.

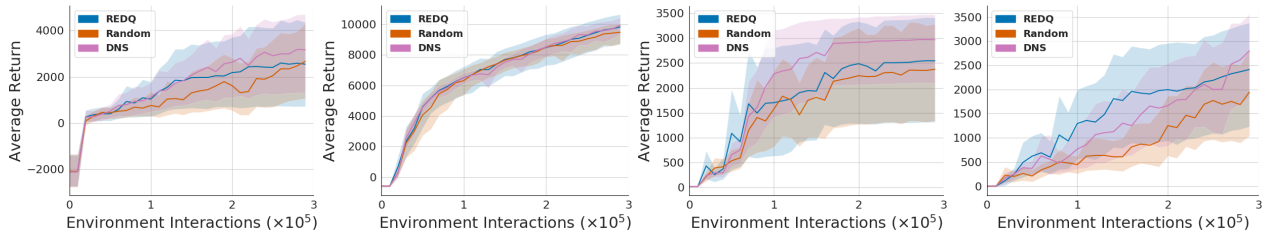
Proof. The nearest positive semi-definite matrix $\widetilde{\boldsymbol{L}}$ to a matrix $\boldsymbol{L}$ can be computed via a spectral decomposition of $\boldsymbol{B} = (\boldsymbol{L} + \boldsymbol{L}^T)/2 = \boldsymbol{V}\Lambda\boldsymbol{V}^T = \boldsymbol{V}diag(\lambda_i)\boldsymbol{V}^T$ as:

$$\widetilde{\boldsymbol{L}} = \boldsymbol{V}diag(d_i)\boldsymbol{V}^T,$$



(a) Ant-v2 (b) HalfCheetah-v2 (c) Walker-v2

Figure 2. Q-value plots of three MuJoCo environments accumulated over four different seeds at the tail end of the training. Notice that the Q-values from the critics have completely collapsed and have zero variance.



(a) Ant-v2 (b) HalfCheetah-v2 (c) Hopper-v2 (d) Walker2d-v2

Figure 3. Training curves with 95% confidence interval of baseline REDQ, random sampling and DNS.

where

$$d_i = \begin{cases} \lambda_i, & \lambda_i \geq 0 \\ 0, & \lambda_i < 0 \end{cases}$$

or equivalently as:

$$\tilde{\boldsymbol{L}} = (\boldsymbol{B} + \boldsymbol{H})/2, \tag{8}$$

where $\boldsymbol{H} = \boldsymbol{V} diag(|\lambda_i|) \boldsymbol{V}^T$ (Higham, 1988). Since $\boldsymbol{L}$ is symmetric, $\boldsymbol{B} = \boldsymbol{L} = \boldsymbol{V} diag(\lambda_i) \boldsymbol{V}^T$ and $\boldsymbol{H} = \boldsymbol{V} diag(|\lambda_i|) \boldsymbol{V}^T$. It follows from (8) that $\tilde{\boldsymbol{L}} = \boldsymbol{L} + \boldsymbol{D}$ where $\boldsymbol{D} = (diag(\lambda_i) + diag(|\lambda_i|))/2 = diag((\lambda_i + |\lambda_i|)/2)$. □

Next, recall that under the tabular version of REDQ a subset of action-value functions are updated to $Q^l_{t+1}(s,a) = Q^l_t(s,a) + \alpha[Y^{MQ} - Q^l_t(s,a)]$ and at $t+1$ an action a is sampled according to the minimum of a random sample M of N Q functions, such that $|\mathcal{M}| = \mathcal{M}$, i.e. according to $\min_{i \in \mathcal{M}} Q^i_{t+1}(s,a)$. In REDQ, $M = 2$.

For $i \in \mathcal{M}$, Let $I_i \in \{0,1\}$ be a random variable indicating if $Q^i(s,a)$ was updated, i.e. $I_i \sim$Bernoulli(p_i). Then

$$\begin{aligned} Q^i_{t+1}(s,a) &= Q^i_t(s,a) + \alpha I_i[Y^{MQ} - Q^i_t(s,a)] \\ &= Q^\pi_t(s,a) + \varepsilon^i_t(s,a) + \alpha I_i[Y^{MQ} - Q^\pi_t(s,a) - \varepsilon^i_t(s,a)] \end{aligned} \tag{9}$$

Furthermore, assume that approximation errors $\varepsilon^i_t(s,a)$ are identically distributed $U(-\tau,\tau)$ for each fixed (s,a). The theorem below characterizes the relationship between k-DPP sampling and the variance of $\min_{i \in \mathcal{M}} Q^i_{t+1}(s,a)$ and $\frac{1}{M}\sum_{i \in \mathcal{M}} Q^i_{t+1}(s,a)$.

Theorem 1. *Under the conditions above and for set $\mathcal{M}$ of M random samples of N Q functions,*

$$Var\, Q^{min} = Var(\min_{i \in \mathcal{M}} Q^i_{t+1}(s,a) | Y^{MQ})$$

decreases if for some $i, j \in \mathcal{M}$ $Q^i_t(s,a)$ and $Q^j_t(s,a)$ were sampled pre-update according to k-DPP. Variance reduction can also be shown for the sample mean

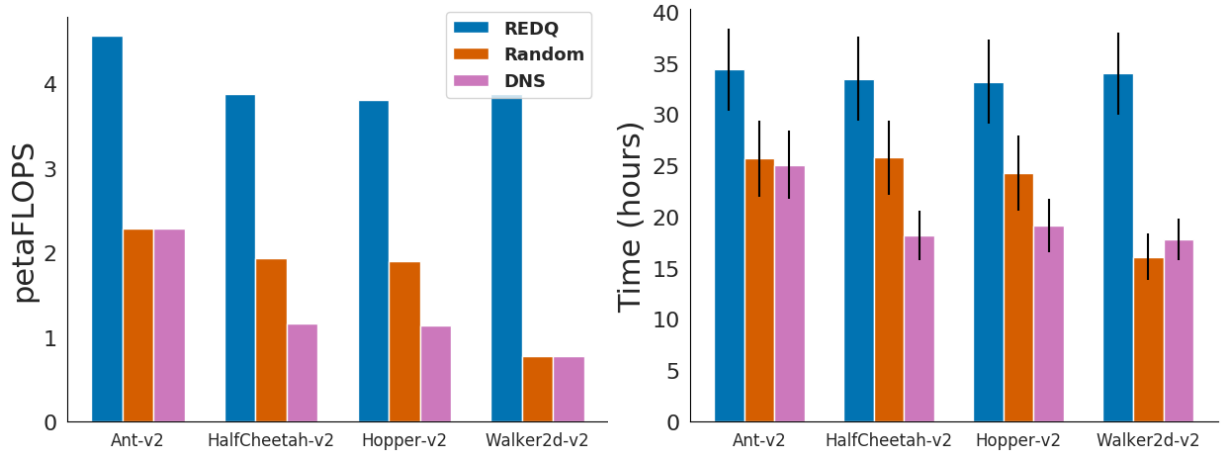
$$Var\, Q^{avg} = Var(\frac{1}{M}\sum_{i \in \mathcal{M}} Q^i_{t+1}(s,a) | Y^{MQ}).$$

Additionally, $Var\, Q^{min}$ and $Var\, Q^{avg}$ are lower under k-DPP than under k-random sampling.

Theorem 1 shows why k-DPP sampling boosts performance over k-random sampling. For the special case that all N Q

Table 1. Max average return for 10 runs of 300K time steps. Maximum value for each task is bolded. ± corresponds to a single standard deviation over runs

Environment	Baseline	Random	DNS
Ant-v2	2543.1 ± 2595.7	2666.8 ± 2262.6	**3167.2 ± 2484.7**
HalfCheetah-v2	9818.8 ± 1445.2	9474.3 ± 991.1	**9931.0 ± 819.1**
Hopper-v2	2544.2 ± 1468.21	2374.9 ± 1405.8	**2967.8 ± 1128.9**
Walker2d-v2	2414.4 ± 1580.0	1946.4 ± 1287.9	**2802.3 ± 1272.1**



(a) Computation cost of the backpropagation method in terms of petaFLOPS

(b) Average wall-clock training time in hours

Figure 4. Both bar graphs represent computation cost of the experiments shown in Figure 3 in terms of FLOPS and wall-clock time.

functions are close to being dissimilar, k-DPP sampling k of the N Q-functions approaches uniform k-random sampling with $P_{\mathcal{K}} = \frac{1}{\binom{N}{k}}$ for all sets of $\mathcal{K}$ size k. We summarize this in the corollary below and note that under the k-DPP scheme, just as in k-random sampling some variance is retained, which is beneficial for exploration.

Corollary 1. *If all N Q-functions are completely dissimilar, k-DPP sampling is equivalent to k-random uniform sampling with each set $\mathcal{K}$ with cardinality k having equal probability $P_{\mathcal{K}} = \frac{1}{\binom{N}{k}}$.*

We show the proofs of Theorem 1 in the appendix. Corollary 1 follows from the fact that when the network activations are completely dissimilar, the off-diagonal elements of the L-matrix are 0 since CKA values are zero. Thus in this case the L-matrix is just the identity matrix and the resulting sampled item probabilities are equal by equation (5).

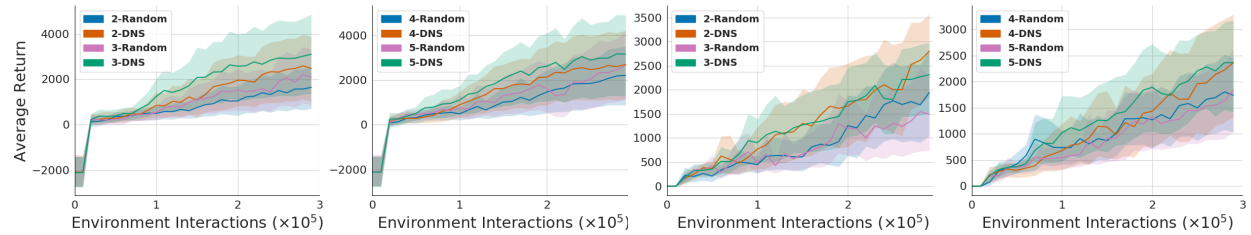
5. Experiments

We designed our experiments to answer the following questions:

1. Can DNS match the performance baseline REDQ while training only a subset of critic networks?
2. Is DNS better than random sampling?
3. Is DNS better than diversity regularization?
4. Does choice of k matter for DNS?

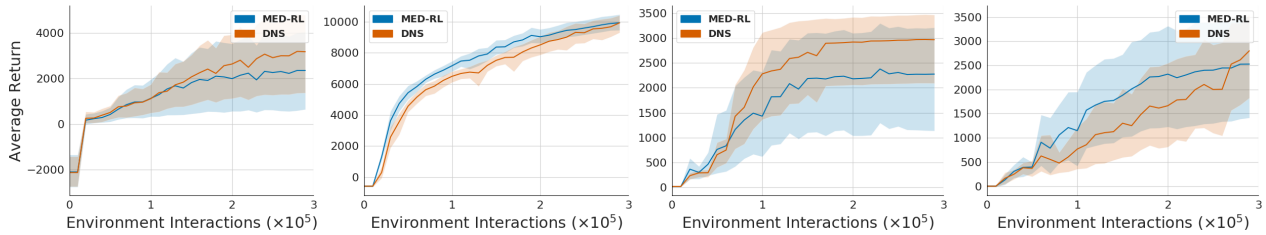
5.1. Experimental Setup

We evaluate DNS on several different continuous control tasks from MuJoCo (Todorov et al., 2012) and compare DNS with baseline REDQ where all the ten critics are trained at every training step and random sampling of neural networks for training. For DNS and random sampling, we sampled between two and five networks for our experiments. Following the setup, we report the highest returns after 300K environment interactions on Ant-v2, HalfCheetah-v2,



(a) Ant with two and three neural networks (b) Ant with four and five neural networks (c) Walker2d with two and three neural networks (d) Walker2d with four and five neural networks

Figure 5. Training curves for Ant-v2 and Walker-v2 environments for varying values of k for both DNS and random sampling.



(a) Ant-v2 (b) HalfCheetah-v2 (c) Hopper-v2 (d) Walker2d-v2

Figure 6. Training plots comparing DNS with MED-RL, a regularization method that uses Gini coefficient to maximize diversity.

Hopper-v2, and Walker2d-v2 environments. We report the mean and the standard deviation across ten runs in Figure 3. For clarity, the results are also shown in Table 1. From Figure 3 and table 1, we can see that DNS consistently outperforms REDQ and random sampling on Ant-v2, Hopper-v2, and Walker2d-v2 and matches the performance of REDQ on HalfCheetah-v2 environment. Note that the goal of this paper is to match the performance of REDQ while reducing the computation cost. For that reason, we did not perform any hyperparameter tuning. All the hyperparameters such as learning rate, batch size, neural network size, and the seeds were kept fixed across all the experiments. The only hyperparameter that has been tuned in this paper is k which samples the number of neural networks.

Details of the hyperparameters used in our experiments are shown in Table 2.

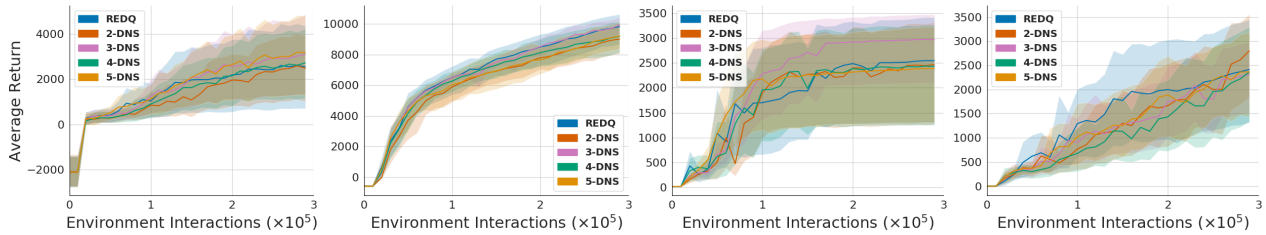
5.2. Computational Analysis

We measured the computation cost of the experiments shown in Figure 3 to verify that sampling a subset of critic networks at training indeed reduces the computation cost. We measured the computation cost in terms of wall-clock time which is a subjective metric and depends on the computing infrastructure and in FLOPS which regard as hardware independent metric. Since we are interested in reducing the backpropagation steps, we wrapped the *Backward()* function in Pytorch's profiler and measured the FLOPS needed to compute the *Backward()* function. We then multiplied the obtained FLOPS by the total number of training steps. The resulting plot is shown in Figure 4a.

Similarly for measuring the wall-clock time, we wrapped the whole training procedure by CodeCarbon (Schmidt et al., 2021). Since wall-clock time is subjective and can be affected by multiple factors, we calculated the average with standard deviation. The resulting plot is shown in Figure 4b. From Figure 4, we can see that DNS achieved better performance than baseline REDQ in at least 50% FLOPS. The key point to note is that DNS achieved 15% more average cumulative reward in less than 25% of FLOPS as compared to baseline REDQ on the Walker2d-v2 environment.

5.3. Is DNS Better than Random Sampling?

To address the question of whether random sampling is better than DNS, we plotted the training curves for Ant-v2 and Walker2d-v2 environments for varying values of k ranging between 2 and 5 in Figure 5. To avoid confusion, we split the plots in two figures for each environment. Figures 5a and 5c shows the training plots for $k = 2$ and $k = 3$ and Figures 5b and 5d shows the training plots for $k = 4$ and $k = 5$. From Figure 5, we can see that for every value of k, DNS outperforms random sampling in both environments.



(a) Ant-v2 (b) HalfCheetah-v2 (c) Hopper-v2 (d) Walker2d-v2

Figure 7. Training curves showing the effect of k on DNS

5.4. Is DNS Better than Diversity Regularization?

Recently (Sheikh et al., 2022) proposed MED-RL that uses economics-inspired regularizers such as Gini and Theil coefficients to maximize diversity between the neural networks. Since k-DPP is an alternate way of inducing diversity, we compared DNS with MED-RL (Gini) where we augmented REDQ with the Gini as proposed in (Sheikh et al., 2022). Figure 6 shows the training curve for the MED-RL and DNS where we can see that DNS clearly outperforms MED-RL in Ant-v2 and Hopper-v2 environments and matches the performance of MED-RL in the HalfCheetah-v2 environment. One quick point to note is that our results of MED-RL do not match with the results proposed in (Sheikh et al., 2022). We attribute this discrepancy to multiple factors such as different learning rates and they have shown results for five seeds only whereas we have shown results for ten seeds.

5.5. Does choice of k matter?

To analyze the effect of the choice of k, we performed an ablation study in which we trained DNS on varying values of k for all four MuJoCo environments and plotted the training curves in Figure 7. For the Ant-v2 environment, $k = 2$ and $k = 4$ performed similar to baseline REDQ. For HalfCheetah-v2 and Walker2d environments, most values of k under-performed when compared with REDQ while for Hopper-v2, $k = 3$ outperformed REDQ significantly.

6. Implementation Details and Hyperparameters

For REDQ, we used the code provided by the authors https://github.com/watchernyu/REDQ. For k-DPP we used the DPPy package https://github.com/guilgautier/DPPy. The complete list of hyperparmeters is given in Table 2.

6.1. Computing Infrastructure

All the experiments were performed on a Kubernetes managed cluster with Nvidia V100 GPUs and Intel Skylake CPUs. Each experiment was run as an individual Kubernetes job with 5 CPUs, 16GB of RAM, and 1 GPU.

Table 2. Hyperparameters for continuous control tasks

Hyperparameter	Value
Target weight τ	$1e^{-3}$
Actor learning rate	$3e^{-4}$
Critic learning rate	$3e^{-4}$
Replay buffer	$1e^{6}$
Batch size	256
Exploration steps	25000
Optimizer	Adam
Hidden Llayer size	256
Number of critics (REDQ)	10
Regularization weight	$1e^{-8}$

7. Conclusion

In this paper, we proposed DNS: a **D**eterminantal Point Process based **N**eural Network **S**ampler that specifically uses k-DPP to sample a subset of neural networks for backpropagation at every training step. This sampling allowed us to reduce the computation cost by 50% during training. We evaluated DNS on MuJoCo environments and compared our results with baseline REDQ and random sampling. Additionally, DNS outperformed MED-RL, a regularization method that maximizes diversity between the ensemble of neural networks in deep reinforcement learning.

References

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *arXiv preprint arXiv:2110.01548*, 2021.

Anschel, O., Baram, N., and Shimkin, N. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML-2017)*, pp. 176–185, 2017.

Chen, R. Y., Sidor, S., Abbeel, P., and Schulman, J. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

Chen, X., Wang, C., Zhou, Z., and Ross, K. W. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

Collings, B. J. Characteristic polynomials by diagonal expansion. *The American Statistician*, 37(3):233–235, 1983.

Cortes, C., Mohri, M., and Rostamizadeh, A. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828, 2012.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. On kernel-target alignment. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS-2002)*, pp. 367–373, 2002.

Dereziński, M., Calandriello, D., and Valko, M. Exact sampling of determinantal point processes with sublinear time preprocessing. *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Ghasemipour, S. K. S., Gu, S. S., and Nachum, O. Why so pessimistic? estimating uncertainties for offline RL through ensembles, and why their independence matters., 2022. URL https://openreview.net/forum?id=wQ7RCayXUSl.

Gong, B., Chao, W.-L., Grauman, K., and Sha, F. Diverse sequential subset selection for supervised video summarization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2069–2077, Cambridge, MA, USA, 2014. MIT Press.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic Learning Theory (ALT)*, pp. 63–77, 2005.

Gupta, S. Learning with combinatorial structure, 2015.

Hado van Hasselt, H., Guez, A., and Silver, D. Deep Reinforcement Learning with Double Q-learning. In *Proceeding of Conference on Artificial Intelligence (AAAI-2016)*, 2016.

Higham, N. J. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103:103–118, 1988.

Hough, J. B., Krishnapur, M., Peres, Y., and Virág, B. Determinantal processes and independence. *Probability surveys (PS)*, 3:206–229, 2006.

Hsiao, W.-L. and Grauman, K. Creating capsule wardrobes from fashion images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *Proceedings of the International Conference on Machine Learning (ICML-2019)*, pp. 3519–3529, 09–15 Jun 2019.

Kulesza, A. and Taskar, B. k-dpps: Fixed-size determinantal point processes. In *International Conference on Machine Learning (ICML)*, 2011.

Kulesza, A. and Taskar, B. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning (FTML)*, 2012.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS-2019)*, pp. 11761–11771, 2019.

Lan, Q., Pan, Y., Fyshe, A., and White, M. Maxmin Q-learning: Controlling the estimation bias of Q-learning. In *Proceeding of the International Conference on Learning Representations (ICLR-2020)*, 2020. URL https://openreview.net/forum?id=Bkg0u3Etwr.

Lee, K., Michael, L., Srinivas, A., and Abbeel, P. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. *arXiv preprint arXiv:2007.04938*, 2020.

Li, C., Jegelka, S., and Sra, S. Fast dpp sampling for nystrom with application to kernel methods. In *International Conference on Machine Learning (ICML)*, pp. 2061–2070. PMLR, 2016.

Liu, R., Nageotte, F., Zanne, P., de Mathelin, M., and Dresp-Langley, B. Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review. *Robotics*, 10(1):22, Jan 2021. ISSN 2218-6581. doi: 10.3390/robotics10010022. URL http://dx.doi.org/10.3390/robotics10010022.

Macchi, O. The coincidence approach to stochastic point processes. *Advances in Applied Probability (AAP)*, 7(1): 83–122, 1975.

Mai, V., Mani, K., and Paull, L. Sample efficient deep reinforcement learning via uncertainty estimation, 2022.

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, February 2015. ISSN 00280836.

Moskovitz, T., Parker-Holder, J., Pacchiano, A., Arbel, M., and Jordan, M. Tactical optimism and pessimism for deep reinforcement learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=a4WgjcLeZIn.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pp. 4026–4034, 2016.

Pacchiano, A., Parker-Holder, J., Choromanski, K. M., and Roberts, S. Effective diversity in population-based reinforcement learning. *NeurIPS 2020 (spotlight)*, 2020. URL https://arxiv.org/abs/2002.00632.

Schmidt, V., Goyal, K., Joshi, A., Feld, B., Conell, L., Laskaris, N., Blank, D., Wilson, J., Friedler, S., and Luccioni, S. CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing. 2021. doi: 10.5281/zenodo.4658424.

Sharghi, A., Gong, B., and Shah, M. Query-focused extractive video summarization. In *European conference on computer vision*, pp. 3–19. Springer, 2016.

Sheikh, H., Phielipp, M., and Boloni, L. Maximizing ensemble diversity in deep reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=hjd-kcpDpf2.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

van Hasselt, H. Double Q-learning. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS-2010)*, pp. 2613–2621, 2010.

Zhang, C., Kjellstrom, H., and Mandt, S. Determinantal point processes for mini-batch diversification. In *Conference on Uncertainty in Artificial Intelligence (AUI)*, 2017.

Zhang, S. and Yao, H. Ace: An actor ensemble algorithm for continuous control with tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5789–5796, 2019.

Łukasz Kaiser, Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

Supplementary

A. Algorithm

Algorithm 1 DNS: REDQ version

Initialize policy parameters θ, N Q-function parameters ϕ_i, $i = 1, \ldots, N$, empty replay buffer $\mathcal{D}$. Set target parameters $\phi_{\text{targ},i} \leftarrow \phi_i$, for $i = 1, 2, \ldots, N$
repeat
  Take one action $a_t \sim \pi_\theta(\cdot|s_t)$. Observe reward r_t, new state s_{t+1}.
  Add data to buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  for G updates **do**
    Sample a mini-batch $B = \{(s, a, r, s')\}$ from $\mathcal{D}$
    Fetch $Q_{\phi_i}(s, a)$ for $i = 1, 2, \ldots, N$
    Compute similarity matrix $\boldsymbol{L}$:

$$\boldsymbol{L} = CKA_{i,j \in N}(Q_{\phi_i}(s, a), Q_{\phi_j}(s, a))$$

    Sample a set K of k distinct indices from $\{1, 2, \ldots, N\}$:

$$K = DPP(\boldsymbol{L}, k)$$

    Sample a set $\mathcal{M}$ of M distinct indices from $\{1, 2, \ldots, N\}$
    Compute the Q target y (same for all of the k Q-functions):

$$y = r + \gamma \left(\min_{i \in \mathcal{M}} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}' \mid s') \right), \quad \tilde{a}' \sim \pi_\theta(\cdot \mid s')$$

    for $i \in K$ **do**
      Update ϕ_i with gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s') \in B} (Q_{\phi_i}(s, a) - y)^2$$

      Update target networks with $\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i$
    end for
  end for
  Update policy parameters θ with gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s) \right), \quad \tilde{a}_\theta(s) \sim \pi_\theta(\cdot \mid s)$$

until

B. Proofs

B.1. Proof of Theorem 1

Lemma 1. *Let $X_i \sim U(a, b)$ and $Y_i \sim B(1, p_i)$. Then for $Z_i = X_i + cY_i(d - X_i)$, $Z_{ijmin} = min(Z_i, Z_j)$, $d \in (a, b)$, $c \in (0, 1)$ we have:*

(i) $Ee^{Z_i t} = (1 - p_i)(\frac{e^{tb} - e^{ta}}{t(b-a)}) + p_i e^{cdt}(\frac{e^{(1-c)tb} - e^{(1-c)ta}}{t(1-c)(b-a)})$ *and* $E[Z_i] = 1 - p_i)(\frac{e^{tb} - e^{ta}}{t(b-a)}) + p_i e^{cdt}(\frac{e^{(1-c)tb} - e^{(1-c)ta}}{t(1-c)(b-a)})$

(ii) The distributions of Z_i and Z_{ijmin} are characterized by:

$$F_{Z_i}(z) = \frac{(1 - p_i)}{a - b}((z - b)\mathbf{1}_{(b,\infty)}(z) - (z - a)\mathbf{1}_{(a,\infty)}(z)) \tag{10}$$

$$+ \frac{p_i}{(1-c)(a-b)}((z-(dc-b(c-1)))\mathbf{1}_{(dc-b(c-1),\infty)}(z) \tag{11}$$

$$- (z-(dc-a(c-1)))\mathbf{1}_{(dc-a(c-1),\infty)}(z)) \tag{12}$$

and

$$F_{Z_{ijmin}}(z) = \frac{2(1-p_i)}{a-b}\beta(z,b,a) + \frac{2p_i}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1)) \tag{13}$$

$$-(\frac{(1-p_{i|j})}{a-b}\beta(z,b,a) \tag{14}$$

$$+ \frac{p_{i|j}}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1)))(\frac{(1-p_i)}{a-b}\beta(z,b,a) \tag{15}$$

$$+ \frac{p_i}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1)) \tag{16}$$

respectively, where

$$p_i = P(Y_i=1) \textit{ and } p_{i|j} = P(Y_i=1|Y_j=1),$$

$$\beta(z,\theta,\alpha) = ((z-\theta)\mathbf{1}_{(\theta,\infty)}(z) - (z-\alpha)\mathbf{1}_{(\alpha,\infty)}(z)),\ \theta \geq \alpha$$

Proof. (i) The moment generating function of $Z_i = X_i + cY_i(d-X_i)$

$$\begin{aligned} Ee^{Z_it} &= Ee^{(X_i+cY_i(d-X_i))t} = E[Ee^{(X_i+cY_i(d-X_i))t}|X_i] \\ &= E[e^{X_it}E[e^{Y_ic(d-X)t}|X]] = E[e^{X_it}[1-p_i+p_ie^{c(d-X_i)t}]] = (1-p_i)E[e^{X_it}] + p_ie^{cdt}e^{X(1-c)t} \\ &= (1-p_i)(\frac{e^{tb}-e^{ta}}{t(b-a)}) + p_ie^{cdt}(\frac{e^{(1-c)tb}-e^{(1-c)ta}}{t(1-c)(b-a)}). \end{aligned} \tag{17}$$

(ii) It follows from (i) that

$\mathcal{L}\{f\}(t) = Ee^{-Z_it} = (1-p_i)(\frac{e^{-tb}-e^{-ta}}{t(a-b)}) + p_ie^{-cdt}(\frac{e^{(c-1)tb}-e^{(c-1)ta}}{t(1-c)(a-b)})$

and

$$\begin{aligned} F_{Z_i}(z) &= \mathcal{L}^{-1}\{\frac{1}{t}\mathcal{L}\{f\}(t)\}(z) \\ &= \mathcal{L}^{-1}\{(1-p_i)(\frac{e^{-tb}-e^{-ta}}{t^2(a-b)}) + p_ie^{-cdt}(\frac{e^{(c-1)tb}-e^{(c-1)ta}}{t^2(1-c)(a-b)})\}(z) \\ &= \mathcal{L}^{-1}\{\frac{(1-p_i)}{a-b}(\frac{e^{-tb}-e^{-ta}}{t^2}) + \frac{p_i}{(1-c)(a-b)}e^{-cdt}(\frac{e^{(c-1)tb}-e^{(c-1)ta}}{t^2})\}(z) \\ &= \frac{(1-p_i)}{a-b}\mathcal{L}^{-1}\{(\frac{e^{-tb}-e^{-ta}}{t^2})\}(z) + \frac{p_i}{(1-c)(a-b)}\mathcal{L}^{-1}\{e^{-cdt}(\frac{e^{(c-1)tb}-e^{(c-1)ta}}{t^2})\}(z) \\ &= \frac{(1-p_i)}{a-b}(\mathcal{L}^{-1}\{\frac{e^{-tb}}{t^2})\}(z) - \mathcal{L}^{-1}\{\frac{e^{-ta}}{t^2})\}(z)) \\ &+ \frac{p_i}{(1-c)(a-b)}(\mathcal{L}^{-1}\{\frac{e^{-t(dc-b(c-1))}}{t^2})\}(z) - \mathcal{L}^{-1}\{\frac{e^{-t(dc-a(c-1)}}{t^2})\}(z)) \\ &= \frac{(1-p_i)}{a-b}((z-b)\mathbf{1}_{(z>b)} - (z-a)\mathbf{1}_{(z>a)}) \\ &+ \frac{p_i}{(1-c)(a-b)}((z-(dc-b(c-1)))\mathbf{1}_{(z>(dc-b(c-1)))} - (z-(dc-a(c-1)))\mathbf{1}_{(z>(dc-a(c-1)))}) \\ &= \frac{(1-p_i)}{a-b}((z-b)\mathbf{1}_{(b,\infty)}(z) - (z-a)\mathbf{1}_{(a,\infty)}(z)) \\ &+ \frac{p_i}{(1-c)(a-b)}((z-(dc-b(c-1)))\mathbf{1}_{(dc-b(c-1),\infty)}(z) - (z-(dc-a(c-1)))\mathbf{1}_{(dc-a(c-1),\infty)}(z)) \end{aligned} \tag{18}$$

Notice that

$$\begin{aligned}
E[Z_i] &= (1-p_i)(\frac{a+b}{2}) + \frac{p_i}{(1-c)(b-a)}[\frac{1}{2}(c-1)(b-a)[(c-1)(a+b)-2cd]] \\
&= (1-p_i)(\frac{a+b}{2}) + \frac{p_i}{2}[(1-c)(a+b)+2cd] \\
&= (1-cp_i)(\frac{a+b}{2}) + cdp_i.
\end{aligned}$$

Furthermore we can derive the variance $\text{Var}(Z_i) = E[Z_i^2] - (E[Z_i])^2$ using

$$(E[Z_i])^2 = (1-cp_i)^2(\frac{a+b}{2})^2 + (1-cp_i)(a+b)cdp_i + (cdp_i)^2$$

and

$$\begin{aligned}
E[Z_i^2] &= (1-p_i)(\frac{a^2+ab+b^2}{3}) + \frac{p_i}{3}[(1-c)^2(a^2+ab+b^2)+3cd((a+b)(1-c)+cd)] \\
&= (1-p_i)(\frac{a^2+ab+b^2}{3}) + \frac{p_i}{3}[(1-c)^2(a^2+ab+b^2)+3cd((a+b)(1-c)+cd)] \\
&= (1-2cp_i+p_ic^2)(\frac{a^2+ab+b^2}{3}) + cdp_i((a+b)(1-c)+cd)
\end{aligned}$$

(iii) Since Z_i are identically distributed but not necessarily independent, the distribution of $Z_{ijmin} = min(Z_i, Z_j)$ is characterized by

$$F_{Z_{ijmin}}(z) = F_{Z_i}(z) + F_{Z_j}(z) - F_{Z_iZ_j}(z,z) = 2F_{Z_i}(z) - F_{Z_iZ_j}(z,z)$$

where

$$F_{Z_iZ_j}(z,z) = P(Z_j \le z, Z_j \le z) = F_{Z_i|Z_j}(z)F_{Z_j}(z)$$

is the joint distribution of Z_i and Z_j. Hence,

$$\begin{aligned}
F_{Z_{ijmin}}(z) = &\frac{2(1-p_i)}{a-b}((z-b)\mathbf{1}_{(b,\infty)}(z) - (z-a)\mathbf{1}_{(a,\infty)}(z)) \\
&+ \frac{2p_i}{(1-c)(a-b)}((z-(dc-b(c-1)))\mathbf{1}_{(dc-b(c-1),\infty)}(z) - (z-(dc-a(c-1)))\mathbf{1}_{(dc-a(c-1),\infty)}(z)) \\
&- (\frac{(1-p_{i|j})}{a-b}((z-b)\mathbf{1}_{(b,\infty)}(z) - (z-a)\mathbf{1}_{(a,\infty)}(z)) \\
&+ \frac{p_{i|j}}{(1-c)(a-b)}((z-(dc-b(c-1)))\mathbf{1}_{(dc-b(c-1),\infty)}(z) \\
&- (z-(dc-a(c-1)))\mathbf{1}_{(dc-a(c-1),\infty)}(z)))(\frac{(1-p_i)}{a-b}((z-b)\mathbf{1}_{(b,\infty)}(z) - (z-a)\mathbf{1}_{(a,\infty)}(z)) \\
&+ \frac{p_i}{(1-c)(a-b)}((z-(dc-b(c-1)))\mathbf{1}_{(dc-b(c-1),\infty)}(z) - (z-(dc-a(c-1)))\mathbf{1}_{(dc-a(c-1),\infty)}(z)) \\
= &\frac{2(1-p_i)}{a-b}\beta(z,b,a) + \frac{2p_i}{(1-c)(a-b)}\beta(z, dc-b(c-1), dc-a(c-1)) - (\frac{(1-p_{i|j})}{a-b}\beta(z,b,a) \\
&+ \frac{p_{i|j}}{(1-c)(a-b)}\beta(z, dc-b(c-1), dc-a(c-1)))(\frac{(1-p_i)}{a-b}\beta(z,b,a) \\
&+ \frac{p_i}{(1-c)(a-b)}\beta(z, dc-b(c-1), dc-a(c-1))
\end{aligned}$$

where,

$$p_i = P(Y_i=1) = p_j,\ p_{i|j} = P(Y_i=1|Y_j=1),\ \beta(z,\theta,\alpha) = ((z-\theta)\mathbf{1}_{(\theta,\infty)}(z) - (z-\alpha)\mathbf{1}_{(\alpha,\infty)}(z)),\ \theta \ge \alpha.$$

Notice that

$$\begin{aligned}\frac{d}{dz}[\beta(z,\theta,\alpha)] &= (z-\theta)\delta(z-\theta) + \mathbf{1}_{(\theta,\infty)}(z) - (z-\alpha)\delta(z-\alpha) - \mathbf{1}_{(\alpha,\infty)}(z) \\ &= (z-\theta)\delta(z-\theta) - (z-\alpha)\delta(z-\alpha) + \mathbf{1}_{(\theta,\alpha)}(z) = \mathbf{1}_{(\theta,\alpha)}(z)\end{aligned}$$

where $\delta(x)$ is the Dirac delta function, implying that

$$\begin{aligned}
f_{Z_{ijmin}}(z) \;=\; & \frac{d}{dz}[F_{Z_{ijmin}}(z)] \\
&= \frac{2(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) + \frac{2p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z) \\
&- (\frac{(1-p_{i|j})}{a-b}\beta(z,b,a) + \frac{p_{i|j}}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1)))(\frac{(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) \\
&+ \frac{p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)) \\
&- (\frac{(1-p_{i|j})}{a-b}\mathbf{1}_{(a,b)}(z) + \frac{p_{i|j}}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z))(\frac{(1-p_i)}{a-b}\beta(z,b,a) \\
&+ \frac{p_i}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1)) \\
&= \frac{2(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) + \frac{2p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z) \\
&- \frac{(1-p_{i|j})}{a-b}\beta(z,b,a)\frac{(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) - \frac{(1-p_{i|j})}{a-b}\beta(z,b,a)\frac{p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1))\frac{(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1))\frac{p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z) \\
&- \frac{(1-p_{i|j})}{a-b}\mathbf{1}_{(a,b)}(z)\frac{(1-p_i)}{a-b}\beta(z,b,a) - \frac{(1-p_{i|j})}{a-b}\mathbf{1}_{(a,b)}(z)\frac{p_i}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1)) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)\frac{(1-p_i)}{a-b}\beta(z,b,a) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)\frac{p_i}{(1-c)(a-b)}\beta(z,dc-b(c-1),dc-a(c-1)) \\
&= \frac{2(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) + \frac{2p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z) \\
&- \frac{(1-p_{i|j})}{a-b}(a-b)\frac{(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) - \frac{(1-p_i)}{a-b}(a-z)\frac{p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}(dc-a(c-1)-z)\frac{(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}(c-1)(b-a)\frac{p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z) \\
&- \frac{(1-p_{i|j})}{a-b}\mathbf{1}_{(a,b)}(z)\frac{(1-p_i)}{a-b}(a-b) - \frac{(1-p_{i|j})}{a-b}\mathbf{1}_{(a,b)}(z)\frac{p_i}{(1-c)(a-b)}(dc-a(c-1)-z) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)\frac{(1-p_i)}{a-b}(a-z) \\
&- \frac{p_{i|j}}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)\frac{p_i}{(1-c)(a-b)}(c-1)(b-a) \\
&= \frac{2(1-p_i)}{a-b}\mathbf{1}_{(a,b)}(z) + \frac{2p_i)}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)
\end{aligned}$$

$$
\begin{aligned}
&- \frac{(1-p_{i|j})}{a-b}\frac{(1-p_i)}{a-b}(a-b)\mathbf{1}_{(a,b)}(z) - \frac{(1-p_{i|j})}{a-b}\frac{p_i}{(1-c)(a-b)}(a-z)\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{(1-p_i)}{a-b}(dc-a(c-1)-z)\mathbf{1}_{(a,b)}(z)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}(c-1)(b-a)\frac{p_i}{(1-c)(a-b)}\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)\\
&- \frac{(1-p_{i|j})}{a-b}\frac{(1-p_i)}{a-b}(a-b)\mathbf{1}_{(a,b)}(z) - \frac{(1-p_{i|j})}{a-b}\frac{p_i}{(1-c)(a-b)}(dc-a(c-1)-z)\mathbf{1}_{(a,b)}(z)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{(1-p_i)}{a-b}(a-z)\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{p_i}{(1-c)(a-b)}(c-1)(b-a)\mathbf{1}_{(dc-b(c-1),dc-a(c-1))}(z)
\end{aligned}
$$

From this we arrive at

$$
\begin{aligned}
EZ_{ijmin}^2 &= \int_{-\infty}^{\infty} x^2 f_{Z_{ijmin}}(x)dx\\
&= \frac{2(1-p_i)}{a-b}\frac{x^3}{3}|_a^b + \frac{2p_i}{(1-c)(a-b)}\frac{x^3}{3}|_{dc-b(c-1)}^{dc-a(c-1)}\\
&- \frac{(1-p_{i|j})}{a-b}\frac{(1-p_i)}{a-b}(a-b)\frac{x^3}{3}|_a^b - \frac{(1-p_{i|j})}{a-b}\frac{p_i}{(1-c)(a-b)}(a\frac{x^3}{3}|_{dc-b(c-1)}^{dc-a(c-1)} - \frac{x^4}{4}|_{dc-b(c-1)}^{dc-a(c-1)})\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{(1-p_i)}{a-b}((dc-a(c-1))\frac{x^3}{3}|_a^b - \frac{x^4}{4}|_a^b)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}(c-1)(b-a)\frac{p_i}{(1-c)(a-b)}\frac{x^3}{3}|_{dc-b(c-1)}^{dc-a(c-1)}\\
&- \frac{(1-p_{i|j})}{a-b}\frac{(1-p_i)}{a-b}(a-b)\frac{x^3}{3}|_a^b - \frac{(1-p_{i|j})}{a-b}\frac{p_i}{(1-c)(a-b)}((dc-a(c-1))\frac{x^3}{3}|_a^b - \frac{x^4}{4}|_a^b)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{(1-p_i)}{a-b}(a\frac{x^3}{3}|_{dc-b(c-1)}^{dc-a(c-1)} - \frac{x^4}{4}|_{dc-b(c-1)}^{dc-a(c-1)})\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{p_i}{(1-c)(a-b)}(c-1)(b-a)\frac{x^3}{3}|_{dc-b(c-1)}^{dc-a(c-1)}
\end{aligned}
$$

and

$$
\begin{aligned}
EZ_{ijmin} &= \int_{-\infty}^{\infty} x f_{Z_{ijmin}}(x)dx = \frac{2(1-p_i)}{a-b}\frac{x^2}{2}|_a^b + \frac{2p_i}{(1-c)(a-b)}\frac{x^2}{2}|_{dc-b(c-1)}^{dc-a(c-1)}\\
&- \frac{(1-p_{i|j})}{a-b}\frac{(1-p_i)}{a-b}(a-b)\frac{x^2}{2}|_a^b - \frac{(1-p_{i|j})}{a-b}\frac{p_i}{(1-c)(a-b)}(a\frac{x^2}{2}|_{dc-b(c-1)}^{dc-a(c-1)}\\
&- \frac{x^3}{3}|_{dc-b(c-1)}^{dc-a(c-1)})\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{(1-p_i)}{a-b}((dc-a(c-1))\frac{x^2}{2}|_a^b - \frac{x^3}{3}|_a^b)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}(c-1)(b-a)\frac{p_i}{(1-c)(a-b)}\frac{x^2}{2}|_{dc-b(c-1)}^{dc-a(c-1)}\\
&- \frac{(1-p_{i|j})}{a-b}\frac{(1-p_i)}{a-b}(a-b)\frac{x^2}{2}|_a^b - \frac{(1-p_{i|j})}{a-b}\frac{p_i}{(1-c)(a-b)}((dc-a(c-1))\frac{x^2}{2}|_a^b - \frac{x^3}{3}|_a^b)\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{(1-p_i)}{a-b}(a\frac{x^2}{2}|_{dc-b(c-1)}^{dc-a(c-1)} - \frac{x^3}{3}|_{dc-b(c-1)}^{dc-a(c-1)})\\
&- \frac{p_{i|j}}{(1-c)(a-b)}\frac{p_i}{(1-c)(a-b)}(c-1)(b-a)\frac{x^2}{2}|_{dc-b(c-1)}^{dc-a(c-1)}
\end{aligned}
$$

□

Next we prove Theorem 1.

Proof. Recall that

$$Q^i_{t+1}(s,a) = Q^\pi_t(s,a) + (\varepsilon^i_t(s,a) + \alpha I_i[Y^{MQ} - Q^\pi_t(s,a) - \varepsilon^i_t(s,a)]$$

where $\varepsilon^i_t(s,a)$ identically distributed $U(-\tau,\tau)$, $I_i \sim$Bernoulli(p_i). Hence,

$$\begin{aligned}
VarQ^{avg} &= Var(\frac{1}{M}\sum_{i\in\mathcal{M}} Q^i_{t+1}(s,a)\,|Y^{MQ}) \\
&= \frac{1}{M^2}Var(\sum_{i\in\mathcal{M}} Q^i_{t+1}(s,a)|Y^{MQ}) \\
&= \frac{1}{M^2}[\sum_{i\in\mathcal{M}} Var(Q^i_{t+1}(s,a)\,|Y^{MQ}) + \sum_{i\neq j} Cov(Q^i_{t+1}(s,a)\,|Y^{MQ}, Q^j_{t+1}(s,a)\,|Y^{MQ})] \\
&= \frac{1}{M^2}[\sum_{i\in\mathcal{M}} Var(Q^i_{t+1}(s,a)\,|Y^{MQ}) + \sum_{i\neq j}(E[Q^i_{t+1}(s,a)\,|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}] \\
&- E[Q^i_{t+1}(s,a)|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}])]
\end{aligned}$$

Consider $|\mathcal{M}| = \in$, Then

$$\begin{aligned}
VarQ^{avg} &= \frac{1}{4}[Var(Q^i_{t+1}(s,a)\,|Y^{MQ}) + Var(Q^j_{t+1}(s,a)|Y^{MQ}) \\
&+ E[Q^i_{t+1}(s,a)\,|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}] \\
&- E[Q^i_{t+1}(s,a)\,|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}]] \\
&= \frac{1}{4}[2Var(Q^i_{t+1}(s,a)\,|Y^{MQ}) + E[Q^i_{t+1}(s,a)\,|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}] \\
&- E[Q^i_{t+1}(s,a)\,|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}]]
\end{aligned}$$

Thus by Lemma 1,

$$\begin{aligned}
VarQ^{avg} &= \frac{1}{4}[2Var(Q^i_{t+1}(s,a)\,|Y^{MQ}) + E[Q^i_{t+1}(s,a)\,|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}] \\
&- E[Q^i_{t+1}(s,a)\,|Y^{MQ}]E[Q^j_{t+1}(s,a)\,|Y^{MQ}]] \\
&= \frac{1}{4}[2Var(Q^i_{t+1}(s,a)\,|Y^{MQ}) + (\alpha(Y^{MQ} - Q^\pi_t(s,a)))^2(p_i p_{j|i} - p_i p_j)] \\
&= \frac{1}{4}[2Var(Q^i_{t+1}(s,a)\,|Y^{MQ}) + (\alpha(Y^{MQ} - Q^\pi_t(s,a)))^2(p_{ij} - p_i^2)] \\
&= \psi + \varphi(p_i p_{j|i} - p_i^2)
\end{aligned}$$

where $\psi = \frac{1}{2}Var(Q^i_{t+1}(s,a)\,|Y^{MQ}), \varphi = \frac{1}{4}\alpha(Y^{MQ} - Q^\pi_t(s,a))^2$
Notice that $\psi, \varphi \geq 0$. So $VarQ^{avg}$ breaks down to

$$VarQ^{avg} = \begin{cases} \psi, \text{ if none of } Q^i_t(s,a), Q^i_{t+1}(s,a) \text{ were updated or if they were updated by random sampling} \\ \psi + \varphi(p_{ij} - p_i^2), \text{ if both } Q^i_t(s,a) \text{ and } Q^i_{t+1}(s,a) \text{ were updated according to } k\text{-DPP} \end{cases}$$

Since k-DPP is a repulsive process, if $Q^i_t(s,a), Q^i_{t+1}(s,a)$ are close $p_{ij} < p_i p_j = p_i^2$, and as they get further apart, because of our choice of kernel based on CKA, $p_{ij} \to p_i p_j$ and $p_i \to \frac{1}{N}$ such that even when points are farther apart, variance is still reduced. Proof for general M follows by induction and similar process can be followed to prove case for $VarQ^{min}$.

□