Appendix

A Control algorithm

The action-value function can be decomposed into two components as:

$$Q^{(PT)}(s,a) = Q_\theta^{(P)}(s,a) + Q_{\mathbf{w}}^{(T)}(s,a), \tag{6}$$

The updates for permanent value function are:

$$\theta_{k+1} \leftarrow \theta_k + \overline{\alpha}_k(\hat{Q}(S_k,A_k) - Q_\theta^{(P)}(S_k,A_k))\nabla_\theta Q_\theta^{(P)}(S_k,A_k). \tag{7}$$

where $\overline{\alpha}$ is the learning rate. An update rule similar to Q-learning can be used for the transient value function:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \delta_t \nabla_{\mathbf{w}} Q_{\mathbf{w}}^{(T)}(S_t,A_t), \tag{8}$$

where $\delta_t = R_{t+1} + \gamma \max_a Q^{(PT)}(S_{t+1},a) - Q^{(PT)}(S_t,A_t)$ is the TD error.

Algorithm 3 PT-Q-learning (Control)

```
Initialize: Buffer B, θ, w
for t : 0 → ∞ do
    Take action A_t
    Store S_t, A_t in B
    Observe R_{t+1}, S_{t+1}
    Update w using Eq. (8)
    if Task Ends then
        Update θ using B and Eq. (7)
        Reset transient value function, w
    end if
end for
```

B Proofs

B.1 Transient Value Function Results

We carry out the analysis by fixing $V^{(P)}$ and only updating $V^{(T)}$ using samples from one particular task. We use $V_t^{(TD)}$ and $V_t^{(PT)}$ to denote the value function estimates at time t learned using TD learning and Alg. 1 respectively.

Theorem B.1.1. *If $V_0^{(TD)} = V^{(P)}$, $V_0^{(T)} = 0$ and the two algorithms train on the same samples using the same learning rates, then $\forall t, V_t^{(PT)} = V_t^{(TD)}$.*

Proof. We use induction to prove this statement.

Base case: When $t = 0$, the PT-estimate of state s is $V_0^{(PT)}(s) = V^{(P)}(s) + V_0^{(T)}(s) = V^{(P)}(s)$. And, the TD estimate of state s is $V_0^{(TD)}(s) = V^{(P)}(s)$. Therefore, both PT and TD estimates are equal.

Induction hypothesis: PT and TD estimates are equal for some timestep k for all states. That is, $V_k^{(PT)}(s) = V^{(P)}(s) + V_k^{(T)}(s) = V_k^{(TD)}(s)$.

Induction step: PT estimates at timestep $k+1$ is

$$\begin{aligned} V_{k+1}^{(PT)}(s) &= V^{(P)}(s) + V_{k+1}^{(T)}(s), \\ &= V^{(P)}(s) + V_k^{(T)}(s) + \alpha_{k+1}(R_{k+1} + \gamma(V^{(P)}(S_{k+1}) + V_t^{(T)}(S_{k+1})) - V^{(P)}(s) - V_k^{(T)}(s)), \end{aligned}$$

$$= V_k^{(PT)}(s) + \alpha_{k+1}(R_{k+1} + \gamma V_k^{(PT)}(S_{k+1}) - V_k^{(PT)}(s)),$$
$$= V_k^{(TD)}(s) + \alpha_{k+1}(R_{t+1} + \gamma V_k^{(TD)}(S_{t+1}) - V_k^{(TD)}(s)),$$
$$\boxed{= V_{k+1}^{(TD)}(s).}$$

The penultimate step follows from the induction hypothesis completing the proof. □

Theorem B.1.2. *The expected target in the transient value function update rule is a contraction-mapping Bellman operator:* $\mathcal{T}^{(T)}V^{(T)} = r_\pi + \gamma\mathcal{P}_\pi V^{(P)} - V^{(P)} + \gamma\mathcal{P}_\pi V^{(T)}$.

Proof. The target for state s at timestep t in the transient value function update rule is $R_{t+1} + \gamma V^{(P)}(S_{t+1}) - V^{(P)}(s) + \gamma V^{(T)}(S_{t+1})$. The expected target is:

$$\begin{aligned}\mathcal{T}^{(T)}V^{(T)}(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma V^{(P)}(S_{t+1}) - V^{(P)}(s) + \gamma V^{(T)}(S_{t+1})|S_t = s],\\ &= r_\pi(s) + \gamma(\mathcal{P}_\pi V^{(P)})(s) - V^{(P)}(s) + \gamma(\mathcal{P}_\pi V^{(T)})(s).\end{aligned}$$

We get the operator by expressing the above equation in the vector form:

$$\mathcal{T}^{(T)}V^{(T)} = r_\pi + \gamma\mathcal{P}_\pi V^{(P)} - V^{(P)} + \gamma\mathcal{P}_\pi V^{(T)}.$$

For contraction proof, consider two vectors $u, v \in \mathbb{R}^{|\mathcal{S}|}$.

$$\begin{aligned}\left\|\mathcal{T}^{(T)}u - \mathcal{T}^{(T)}v\right\|_\infty &= \left\|r_\pi + \gamma\mathcal{P}_\pi V^{(P)} - V^{(P)} + \gamma\mathcal{P}_\pi u - (r_\pi + \gamma\mathcal{P}_\pi V^{(P)} - V^{(P)} + \gamma\mathcal{P}_\pi v)\right\|_\infty,\\ &= \|\gamma\mathcal{P}_\pi(u - v)\|_\infty,\\ &\boxed{\leq \gamma\|u - v\|_\infty.}\end{aligned}$$

Therefore, the transient value function Bellman operator is a contraction mapping with contraction factor γ. □

Theorem B.1.3. *The unique fixed point of the transient operator is* $(I - \gamma\mathcal{P}_\pi)^{-1}r_\pi - V^{(P)}$.

Proof. We get the fixed point, $V_\pi^{(T)}$, by repeatedly applying the operator to a vector v:

$$\begin{aligned}V_\pi^{(T)} &= r_\pi + \gamma\mathcal{P}_\pi V^{(P)} - V^{(P)} + \gamma\mathcal{P}_\pi v,\\ &= r_\pi + \gamma\mathcal{P}_\pi V^{(P)} - V^{(P)} + \gamma\mathcal{P}_\pi(r_\pi + \gamma\mathcal{P}_\pi V^{(P)} - V^{(P)} + \gamma\mathcal{P}_\pi v),\\ &= r_\pi + \gamma\mathcal{P}_\pi r_\pi + (\gamma\mathcal{P}_\pi)^2V^{(P)} - V^{(P)} + (\gamma\mathcal{P}_\pi)^2 v,\\ &= r_\pi + \gamma\mathcal{P}_\pi r_\pi + (\gamma\mathcal{P}_\pi)^2V^{(P)} + \cdots + (\gamma\mathcal{P}_\pi)^nV^{(P)} - V^{(P)} + (\gamma\mathcal{P}_\pi)^n v,\\ &\boxed{= (I - \gamma\mathcal{P}_\pi)^{-1}r_\pi - V^{(P)}.}\end{aligned}$$

We applied the operator to v in steps 1 to 4 and used the property $\lim_{n\to\infty}(\gamma\mathcal{P}_\pi)^n \to 0$ in the penultimate step as $\gamma\mathcal{P}_\pi$ is a sub-stochastic matrix. Many terms cancel out as it is a telescoping sum. □

Theorem B.1.4. *Let* $\mathcal{M}$ *be an MDP in which the agent is performing transient value function updates for* π. *Define an MRP,* $\widetilde{\mathcal{M}}$, *with state space* $\widetilde{\mathcal{S}} : \{(S_t, A_t, S_{t+1}), \forall t\}$, *transition dynamics* $\widetilde{\mathcal{P}}(x'|x, a') = \mathcal{P}(s''|s', a')$, $\forall x, x' \in \widetilde{\mathcal{S}}, \forall s', s'' \in \mathcal{S}, \forall a' \in \mathcal{A}$ *and reward function* $\widetilde{\mathcal{R}}(x, a') = \mathcal{R}(s, a) + \gamma V^{(P)}(s') - V^{(P)}(s)$. *Then, the fixed point of Eq.(5) is the value function of* π *in* $\widetilde{\mathcal{M}}$.

Proof. The state space of the new MDP $\widetilde{\mathcal{M}}$ is made up of all possible consecutive tuples of (S_t, A_t, S_{t+1}) pairs. The rewards in this MDP are defined as $\widetilde{\mathcal{R}}(x, a') = \mathcal{R}(s, a) + \gamma V^{(P)}(s') - V^{(P)}(s), \forall x \in \widetilde{\mathcal{S}}, \forall a' \in \mathcal{A}$. The transition matrix $\widetilde{\mathcal{P}}(x'|x, a') = prob(x'|x, a') = prob(s', a', s''|s, a, s', a') = prob(s''|s', a') = \mathcal{P}(s''|s', a')$ using Markov property of $\mathcal{M}$. Then, the value of state $x = (s, a, s')$ is:

$$v_\pi(x) = v_\pi(s, a, s') = \mathbb{E}_\tau[Z_t|X_t = (s, a, s')],$$

where $Z_t = \sum_{k=t}^{\infty} \gamma^{k-t} \tilde{R}_{k+1}$ is the returns starting from state $X_t = (s, a, s')$. Without loss of generality, consider a sample trajectory generated according to π, $\tau_k = (S_0 = s, A_0 = a, R_1, S_1 = s', A_1, R_2, S_2, A_2, R_3, \ldots, R_T, S_T)$ where S_T is the terminal state. The returns Z^{τ_k} of this trajectory is:

$$\begin{aligned}
Z^{\tau_k} &= \sum_{k=0}^{\infty} \gamma^k \tilde{R}_{k+1} \\
&= \sum_{k=0}^{\infty} \gamma^k (R_{k+1} + \gamma V^{(P)}(S_{k+1}) - V^{(P)}(S_k)), \\
&= \sum_{k=0}^{\infty} \gamma^k R_{k+1} - V^{(P)}(S_0 = s), \\
&\boxed{= G^{\tau_k} - V^{(P)}(s).}
\end{aligned}$$

Returns of every trajectory end up being the difference between the returns in the original MDP, G^{τ_k} and the permanent value function. Most importantly, these returns only depend on the state s and it is the same for all a and s' choices following s. The value of state $x = (s, a, s')$ can then be simplified using this observation:

$$\begin{aligned}
v_\pi(s, a, s') &= \mathbb{E}_\tau[Z_t | X_t = (s, a, s')], \\
&= \mathbb{E}_\tau[G_t - V^{(P)}(s) | S_t = s, A_t = \cdot, S_{t+1} = \cdot], \\
&\boxed{= V_\pi^{(T)}(s).}
\end{aligned}$$

□

B.2 Permanent Value Function Results

We focus on permanent value function in the next two theorems. First, we characterize the fixed point of the permanent value function. We use assumption 1 on the task distribution for the following proofs:

Theorem B.2.1. *Following Theorem 2, under assumption 1 and Robbins-Monro step-size conditions, the sequence of updates computed by Eq.(4) contracts to a unique fixed point* $\mathbb{E}_\tau[v_\tau]$.

Proof. The permanent value function is updated using Eq. (4) and samples stored in the buffer $\mathcal{B}$ at the end of each task. This update rule can be simplified as:

$$\begin{aligned}
V_{k+1}^{(P)}(s) &\leftarrow V_k^{(P)}(s) + \overline{\alpha}_k \sum_{i=1}^{L} \mathbb{I}_{(s_i = s)} (V_\tau^{(PT)}(s) - V_k^{(P)}(s)), \\
&= V_k^{(P)}(s) + \overline{\alpha}_k (V_\tau^{(PT)}(s) - V_k^{(P)}(s)) \sum_{i=1}^{L} \mathbb{I}_{(s_i = s)}, \\
&= V_k^{(P)}(s) + \overline{\alpha}_k L(s) (V_\tau^{(PT)}(s) - V_k^{(P)}(s)), \\
&= V_k^{(P)}(s) + \widetilde{\alpha}_k(s) (V_\tau^{(PT)}(s) - V_k^{(P)}(s)),
\end{aligned}$$

where L is the length of the buffer, $\mathbb{I}$ is the indicator function, $L(s)$ is the number of samples for state s in the buffer $\mathcal{B}$, $\widetilde{\alpha}_k(s) = \overline{\alpha}_k L(s)$. The above update is performed once for every state in $\mathcal{B}$ at the end of each task. We assume that the buffer has enough samples for each state so that all states are updated. This can be ensured by storing all the samples in $\mathcal{B}$ for a long but finite period of time. The updates performed to permanent value function until that timestep will not affect the fixed point since we are in the tabular setting.

Let the sequence of learning rate $\widetilde{\alpha}_k(s)$ obey the conditions: $\sum_{n=1}^{\infty} \widetilde{\alpha}_n(s) = \infty$ and $\sum_{n=1}^{\infty} \widetilde{\alpha^2}_n(s) = 0$.

We can rewrite the PM update rule by separating the noise term:

$$V_{k+1}^{(P)}(s) \leftarrow V_k^{(P)}(s) + \widetilde{\alpha}_k(s) (V_\tau^{(PT)}(s) - V_k^{(P)}(s)),$$

$$= V_k^{(P)}(s) + \widetilde{\alpha}_k(s)(\mathbb{E}[V_\tau^{(PT)}(s)] - V_k^{(P)}(s) + V_\tau^{(PT)}(s) - \mathbb{E}[V_\tau^{(PT)}(s)]),$$
$$= V_k^{(P)}(s) + \widetilde{\alpha}_k(s)(\mathbb{E}[V_\tau^{(PT)}(s)] - V_k^{(P)}(s) + w_k).$$

Let $\mathcal{F}_k$ denote the history of the algorithm. Then, $\mathbb{E}[w_k(s)|\mathcal{F}_k] = \mathbb{E}[V_\tau^{(PT)}(s) - \mathbb{E}[V_\tau^{(PT)}(s)]] = 0$ and under the assumption $\mathbb{E}[w_k^2(s)|\mathcal{F}_k] \leq A + B{V_k^{(P)}}^2(s)$ (finite noise variance), we conclude that the permanent value function converges to $E[V_\tau^{(PT)}(s)]$. But $E[V_\tau^{(PT)}(s)]$ is contracting towards $\mathbb{E}[v_\tau(s)]$, therefore, the permanent value function is contracting towards $\mathbb{E}[v_\tau(s)]$. □

Theorem B.2.2. *The fixed point of the permanent value function optimizes the jumpstart objective,* $J = \arg\min_{u \in \mathbb{R}^{|\mathcal{S}|}} \frac{1}{2}\mathbb{E}_\tau[\|u - v_\tau\|_2^2]$.

Proof. The jumpstart objective function is given by $J = \arg\min_{u \in \mathbb{R}^{|\mathcal{S}|}} \frac{1}{2}\mathbb{E}_\tau[\|u - v_\tau\|_2^2]$. Let $K = \frac{1}{2}\mathbb{E}_\tau[\|u - v_\tau\|_2^2]$. We get the optimal point by differentiating K with respect to u and then equating it to 0:

$$\begin{aligned}
\nabla_u K &= \nabla_u \left(\frac{1}{2}\mathbb{E}_\tau[\|u - v_\tau\|_2^2]\right), \\
&= \frac{1}{2}\mathbb{E}_\tau[\nabla_u \|u - v_\tau\|_2^2], \\
&= \frac{1}{2}\mathbb{E}_\tau[2(u - v_\tau)], \\
&= u - \mathbb{E}_\tau[v_\tau], \\
0 &= u - \mathbb{E}_\tau[v_\tau], \\
\boxed{u &= \mathbb{E}_\tau[v_\tau].}
\end{aligned}$$

Therefore, the fixed point of the permanent value function also optimizes the jumpstart objective. □

B.3 Semi-Continual RL Results

We assume that at time t, both $V^{(TD)}$ and $V^{(PT)}$ have converged to the true value function of task i and the agent gets samples from task τ from timestep $t+1$ onward for the following theorems.

Theorem B.3.1. *Under assumption 1, there exists* k_0 *such that* $\forall k \geq k_0$, $\mathbb{E}_\tau[\left\|V_{t+k}^{(TD)} - v_i\right\|_2^2] > \mathbb{E}_\tau[\left\|V^{(P)} - v_i\right\|_2^2]$, $\forall i$.

Proof. Let $\left\|V^{(P)} - v_i\right\|_2^2 = D_i$, which is a constant because we are updating transient value function only. Let $\|v_\tau - v_i\|_2^2 = \Delta_{\tau,i}$ denote the distance between value functions of two tasks i and τ. Clearly, $\Delta_{\tau,i} = \Delta_{i,\tau} \neq 0$ and $\Delta_{i,i} = 0$. And, let $Err_t^{i,\tau} = \left\|V_t^{(TD)} - v_i\right\|_2^2$ denote prediction error with respect to the previous task i at timestep t for TD estimates.

At timestep t, $V^{(PT)} = V^{(TD)} = v_i$. The agent starts getting samples from task τ from timestep $t+1$. Using the result from Bhandari et al. [6]:

$$\begin{aligned}
\left\|V_{t+1}^{(TD)} - v_\tau\right\|_2^2 &\leq \exp\left\{\frac{-\omega(1-\gamma^2)}{4}\right\} \left\|V_t^{(TD)} - v_\tau\right\|_2^2, \\
&= \exp\left\{\frac{-(1-\gamma^2)}{4}\right\} \Delta_{\tau,i},
\end{aligned}$$

since the smallest eigenvalue of the feature matrix $\omega = 1$ for tabular approximations. Let $C = \exp\{\frac{-(1-\gamma^2)}{4}\}$. Now we can apply the above result recursively for k timesteps to get

$$\left\|V_{t+k}^{(TD)} - v_\tau\right\|_2^2 \leq C^k \Delta_{\tau,i}.$$

Then the error with respect to the past task after k timesteps is,

$$Err_{t+k}^{i,\tau} \geq \Delta_{\tau,i} - C^k \Delta_{\tau,i} = \Delta_{\tau,i}(1 - C^k).$$

Taking the expectation with respect to τ gives

$$\begin{aligned}
\mathbb{E}_\tau[Err_{t+k}^{i,\tau}] &= \sum_\tau p_\tau Err_{t+k}^{i,\tau}, \\
&\geq \sum_\tau p_\tau \Delta_{\tau,i}(1 - C^k), \\
&\geq (1 - C^k)\mathbb{E}_\tau[\Delta_{\tau,i}].
\end{aligned}$$

Now we will find a timestep k_0 such that $\forall k > k_0$, $E_\tau[Err_{t+k}^{i,\tau}] > D_i$. Let k_i denote the timestep for which $(1 - C^{k_i})E_\tau[\Delta_{\tau,i}] = D_i$. Then,

$$\begin{aligned}
(1 - C^{k_i})E_\tau[\Delta_{\tau,i}] &= D_i, \\
(1 - C^{k_i}) &= \frac{D_i}{E_\tau[\Delta_{\tau,i}]}, \\
C^{k_i} &= \frac{E_\tau[\Delta_{\tau,i}] - D_i}{E_\tau[\Delta_{\tau,i}]}, \\
\boxed{k_i &= \frac{1}{\log C} \log\left(\frac{E_\tau[\Delta_{\tau,i}] - D_i}{E_\tau[\Delta_{\tau,i}]}\right).}
\end{aligned}$$

So, after time $t + k_i$, $V^{(P)}$ is a better estimate of the value function of the past task i than the current TD estimates. Then, by letting $k_0 = \max_i k_i$ we have $\mathbb{E}_\tau[Err_{t+k_0}^{i,\tau}] > D_i, \forall i$.

Note that the above result depends on the fact that k_i exists, which is only possible when $E_\tau[\Delta_{\tau,i}] - D_i > 0$. We will now show that there exists at least one solution, the fixed point of the permanent value function ($\mathbb{E}_\tau[v_\tau]$), which satisfies this constraint.

$$\begin{aligned}
E_\tau[\Delta_{\tau,i}] &> \|\mathbb{E}_\tau[v_i - v_\tau]\|_2^2 \,, \text{(Using Jensen's inequality)} \\
&= \|v_i - \mathbb{E}_\tau[v_\tau]\|_2^2 \,.
\end{aligned}$$

Then,

$$\boxed{E_\tau[\Delta_{\tau,i}] - D_i > \|v_i - \mathbb{E}_\tau[v_\tau]\|_2^2 - D_i = 0.}$$

The final step follows from the fact that $D_i = \left\|V^{(P)} - v_i\right\|_2^2 = \|\mathbb{E}_\tau[v_\tau] - v_i\|_2^2$ for $V^{(P)}$ fixed point. □

Next, we show that our method has tighter upper bound on errors compared with the TD learning algorithm in expectation.

Theorem B.3.2. *Under assumption 1, there exists k_0 such that $\forall k \leq k_0$, $\mathbb{E}_\tau\left[\left\|v_\tau - V_{t+k}^{(PT)}\right\|_2^2\right]$ has a tighter upper bound compared with $\mathbb{E}_\tau\left[\left\|v_\tau - V_{t+k}^{(TD)}\right\|_2^2\right]$, $\forall i$.*

Proof. We assume that $V^{(P)}$ is converged to its fixed point, $\mathbb{E}_\tau[v_\tau]$ (Theorem 5). Using sample complexity bounds from Bhandari et al. [6] for TD learning algorithm, we get:

$$\begin{aligned}
\mathbb{E}_\tau\left[\left\|v_\tau - V_{t+k}^{(TD)}\right\|_2^2\right] &\leq \mathbb{E}_\tau\left[\exp\left\{-\left(\frac{(1-\gamma)^2\omega}{4}\right)T\right\}\left\|v_\tau - V_t^{(TD)}\right\|_2^2\right], \\
&= \exp\left\{-\left(\frac{(1-\gamma)^2\omega}{4}\right)T\right\}\mathbb{E}_\tau\left[\left\|v_\tau - V_t^{(TD)}\right\|_2^2\right].
\end{aligned}$$

We know that our algorithm can be interpreted as TD learning with $V^{(P)}$ as the starting point (see Theorem 1). Using this relationship, we can apply the sample complexity bound to get:

$$\mathbb{E}_\tau\left[\left\|v_\tau - V_{t+k}^{(PT)}\right\|_2^2\right] \leq \mathbb{E}_\tau\left[\exp\left\{-\left(\frac{(1-\gamma)^2\omega}{4}\right)T\right\}\left\|v_\tau - V_t^{(P)}\right\|_2^2\right],$$
$$= \exp\left\{-\left(\frac{(1-\gamma)^2\omega}{4}\right)T\right\}\mathbb{E}_\tau\left[\left\|v_\tau - V_t^{(P)}\right\|_2^2\right].$$

For $k = 0$, we know that our algorithm has lower error in expectation (see Theorem 6), that is $\mathbb{E}_\tau\left[\left\|v_\tau - V_t^{(P)}\right\|_2^2\right] \leq \mathbb{E}_\tau\left[\left\|v_\tau - V_t^{(TD)}\right\|_2^2\right]$. Therefore, the error bound for our algorithm is tighter compared with that of TD learning algorithm. As $k \to \infty$, the right hand side of both the equations goes to 0, which confirms that both algorithms converges to the true value function. □

The theorem shows that the estimates for our algorithm lie within a smaller region compared to that of TD learning estimates. This doesn't guarantee low errors for our algorithms, but we can expect it to be closer to the true value function.

The next theorems contain analytical MSE equations for TD-learning and for our algorithm. Our analysis is similar to that of [42], but we don't assume that rewards are only observed in the final transition. Analytical MSE equations allow calculating the exact MSE that an algorithm would incur at each timestep when applied to a problem, instead of averaging the performance over several seeds, which is the first step towards other analyses such as bias-variance decomposition of value estimates, error bounds and bias-variance bounds, step-size adaptation for the update rule, etc. We use these equations to compare our method against TD-learning when varying the task distribution diameter (Sec. C.2).

We assume that the agent gets N i.i.d samples $\langle s, r, s'\rangle$ from task τ in each round, where s is sampled according to the stationary distribution d_τ. Also, the agent accumulates errors from all the samples and updates its estimates at the end of the round t. Let $\mathbb{E}_{seeds}$ denote the expectation with respect to random seeds, i.e., expectation with respect to the data generation process which generates the data that the agent observes over rounds and within a round.

Theorem B.3.3. *Let:*

$$m_t^{(TD)}(s) = \mathbb{E}_{seeds}[V_t^{(TD)}(s)],$$
$$Cov(V_t^{(TD)}(s), V_t^{(TD)}(x)) = \Sigma_t^{(TD)}(s,x) - m_t^{(TD)}(s)m_t^{(TD)}(x),$$

where $\Sigma_t^{(TD)}(s,x) = \mathbb{E}_{seeds}[V_t^{(TD)}(s)V_t^{(TD)}(s)]$. *We have:*

$$m_{t+1}^{(TD)}(s) = m_t^{(TD)}(s) + \alpha N d_\pi(s)\Delta_t^{(TD)}(s),$$
$$\Sigma_{t+1}^{(TD)}(s,x) = \Sigma_t^{(TD)}(s,x) + \alpha\Omega_t^{(TD)}(s,x) + \alpha^2\Psi_t^{(TD)}(s,x),$$

and, the mean squared error $\xi_t^{(TD)}$ *at timestep* t *is*

$$\xi_{t+1}^{(TD)} = \xi_t^{(TD)} + \alpha\sum_{s\in\mathcal{S}} d_\pi(s)(-2v_\pi(s)\Delta_t^{(TD)}(s) + \Omega_t^{(TD)}(s,s))$$
$$+ \alpha^2\sum_{s\in\mathcal{S}} d_\pi(s)\Psi_t^{(TD)}(s,s).$$

Proof.

$$m_{t+1}^{(TD)}(s) = \mathbb{E}_{seeds}[V_{t+1}^{(TD)}(s)],$$
$$= \mathbb{E}_{seeds}\left[V_t^{(TD)}(s) + \alpha\sum_{k=1}^{N}\mathbb{I}_{S_k=s}(R_k + \gamma V_t^{(TD)}(S'_k) - V_t^{(TD)}(s))\right],$$
$$= m_t^{(TD)}(s) + \alpha\sum_{k=1}^{N}\mathbb{E}_{seeds}[\mathbb{I}_{S_k=s}(R_k + \gamma V_t^{(TD)}(S'_k) - V_t^{(TD)}(s))],$$

$$
\begin{aligned}
&= m_t^{(TD)}(s) + \alpha \sum_{k=1}^{N} \mathbb{E}_{seeds}\left[\mathbb{E}[\mathbb{I}_{S_k=s}(R_k + \gamma V_t^{(TD)}(S'_k) - V_t^{(TD)}(s))|seed]\right], \\
&= m_t^{(TD)}(s) + \alpha \sum_{k=1}^{N} \mathbb{E}_{seeds}\left[\widehat{d_\pi}(s)\mathbb{E}[(R_k + \gamma V_t^{(TD)}(S'_k) - V_t^{(TD)}(s))|seed, S_k = s]\right], \\
&= m_t^{(TD)}(s) + \alpha \sum_{k=1}^{N} \mathbb{E}_{seeds}\left[\underbrace{\widehat{d_\pi}(s)(\widehat{r_\pi}(s) + \gamma(\widehat{\mathcal{P}_\pi} = V_t^{(TD)})(s) - V_t^{(TD)}(s))}_{\text{Independent quantities}}\right], \\
&= m_t^{(TD)}(s) + \alpha \sum_{k=1}^{N} d_\pi(s)(r_\pi(s) + \gamma(\mathcal{P}_\pi m_t^{(TD)})(s) - m_t^{(TD)}(s))], \\
&= m_t^{(TD)}(s) + \alpha N d_\pi(s)(r_\pi(s) + \gamma(\mathcal{P}_\pi m_t^{(TD)})(s) - m_t^{(TD)}(s))], \\
&\boxed{m_{t+1}^{(TD)}(s) = m_t^{(TD)}(s) + \alpha\Delta_t^{(TD)}(s),}
\end{aligned}
$$

where $\Delta_t^{(TD)}(s) = r_\pi(s) + \gamma(\mathcal{P}_\pi m_t^{(TD)})(s) - m_t^{(TD)}(s)$. We used the following results in the proof:

1. $\mathbb{E}_{seeds}[\widehat{d_\pi}(s)\widehat{r_\pi}(s)] = \mathbb{E}_{seeds}[\widehat{d_\pi}(s)]\ \mathbb{E}_{seeds}[\widehat{r_\pi}(s)] = d_\pi(s)r_\pi(s)$.
2. $$
\begin{aligned}
\mathbb{E}_{seeds}[\widehat{d_\pi}(s)\gamma(\widehat{\mathcal{P}_\pi}V_t^{(TD)})(s)] &= \gamma\mathbb{E}_{seeds}[\widehat{d_\pi}(s)]\ \mathbb{E}_{seeds}[(\widehat{\mathcal{P}_\pi}V_t^{(TD)})(s)], \\
&= \gamma d_\pi(s) \sum_{s'} \mathbb{E}_{seeds}[\widehat{\mathcal{P}_\pi}(s'|s)V_t^{(TD)}(s')], \\
&= \gamma d_\pi(s) \sum_{s'} \mathcal{P}_\pi(s'|s)m_t^{(TD)}(s')], \\
&= \gamma d_\pi(s)(\mathcal{P}_\pi m_t^{(TD)})(s).
\end{aligned}
$$
3. $\mathbb{E}_{seeds}[\widehat{d_\pi}(s)V_t^{(TD)}(s)] = \mathbb{E}_{seeds}[\widehat{d_\pi}(s)]\ \mathbb{E}_{seeds}[V_t^{(TD)}(s)] = d_\pi(s)m_t^{(TD)}(s)$.

Now consider $\Sigma_{t+1}^{(TD)}(s, x)$,

$$
\begin{aligned}
\Sigma_{t+1}^{(TD)}(s,x) &= \mathbb{E}_{seeds}[V_{t+1}^{(TD)}(s)V_{t+1}^{(TD)}(x)], \\
&= \mathbb{E}_{seeds}\Bigg[\left(V_t^{(TD)}(s) + \alpha \sum_{k=1}^{N} \mathbb{I}_{S_k=s}(R_k + \gamma V_t^{(TD)}(S'_k) - V_t^{(TD)}(s))V_{t+1}^{(TD)}(x)\right) \\
&\qquad \left(V_t^{(TD)}(x) + \alpha \sum_{l=1}^{N} \mathbb{I}_{S_l=x}(R_l + \gamma V_t^{(TD)}(X'_l) - V_t^{(TD)}(x))\right)\Bigg], \\
&= \mathbb{E}_{seeds}[V_t^{(TD)}(s)V_t^{(TD)}(x)] \\
&\quad + \alpha\mathbb{E}_{seeds}\left[V_t^{(TD)}(s)\left(\sum_{l=1}^{N} \mathbb{I}_{S_l=x}(R_l + \gamma V_t^{(TD)}(X'_l) - V_t^{(TD)}(x))\right)\right] \qquad (9) \\
&\quad + \alpha\mathbb{E}_{seeds}\left[V_t^{(TD)}(x)\left(\sum_{k=1}^{N} \mathbb{I}_{S_k=s}(R_k + \gamma V_t^{(TD)}(S'_k) - V_t^{(TD)}(s))\right)\right] \qquad (10) \\
&\quad + \alpha^2\mathbb{E}_{seeds}\Bigg[\left(\sum_{k=1}^{N} \mathbb{I}_{S_k=s}(R_k + \gamma V_t^{(TD)}(S'_k) - V_t^{(TD)}(s))\right) \\
&\qquad \left(\sum_{l=1}^{N} \mathbb{I}_{S_l=x}(R_l + \gamma V_t^{(TD)}(X'_l) - V_t^{(TD)}(x))\right)\Bigg] \qquad (11)
\end{aligned}
$$

Next, we will tackle each part separately. We will begin with Eq (9):

$$
\begin{aligned}
&\mathbb{E}_{seeds}\left[V_t^{(TD)}(s)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}(R_l+\gamma V_t^{(TD)}(X_l')-V_t^{(TD)}(x))\right)\right]\\
&=\mathbb{E}_{seeds}\left[V_t^{(TD)}(s)\sum_{l=1}^{N}\mathbb{E}[(\mathbb{I}_{S_l=x}(R_l+\gamma V_t^{(TD)}(X_l')-V_t^{(TD)}(x)))|seed]\right],\\
&=\mathbb{E}_{seeds}[V_t^{(TD)}(s)N\widehat{d_\pi}(x)(\widehat{r_\pi}(x)+\gamma(\widehat{\mathcal{P}_\pi}V_t^{(TD)})(x)-V^{(TD)}(x))],\\
&=N\mathbb{E}_{seeds}[\widehat{d_\pi}(x)\widehat{r_\pi}(x)V_t^{(TD)}(s)]+\gamma N\mathbb{E}_{seeds}[\widehat{d_\pi}(x)(\widehat{\mathcal{P}_\pi}V_t^{(TD)})(x)V_t^{(TD)}(s)]\\
&\quad-N\mathbb{E}_{seeds}[\widehat{d_\pi}(x)V_t^{(TD)}(x)V_t^{(TD)}(s)],\\
&=Nd_\pi(x)r_\pi(x)m^{(TD)}(s)+\gamma Nd_\pi(x)\mathbb{E}_{seeds}\left[\sum_{x'}\widehat{\mathcal{P}_\pi}(x'|x)V_t^{(TD)}(x')V_t^{(TD)}(x)\right]\\
&\quad-Nd_\pi(x)\Sigma_t^{(TD)}(s,x),\\
&=Nd_\pi(x)\left(r_\pi(x)m_t^{(TD)}(s)+\gamma\sum_{x'}\mathbb{E}_{seeds}[\widehat{\mathcal{P}_\pi}(x'|x)V_t^{(TD)}(x')V_t^{(TD)}(x)]-\Sigma_t^{(TD)}(s,x)\right),\\
&=Nd_\pi(x)\left(r_\pi(x)m_t^{(TD)}(s)+\gamma\sum_{x'}\mathcal{P}_\pi(x'|x)\Sigma_t^{(TD)}(x',s)-\Sigma_t^{(TD)}(s,x)\right),\\
&\boxed{=Nd_\pi(x)\left(r_\pi(x)m_t^{(TD)}(s)+\gamma(\mathcal{P}_\pi\Sigma_t^{(TD)})(x,s)-\Sigma_t^{(TD)}(s,x)\right).}
\end{aligned}
$$

A similar simplification can be done to Eq. (10) which results in:

$$
\boxed{(10)=Nd_\pi(s)\left(r_\pi(s)m^{(TD)}(x)+\gamma(\mathcal{P}_\pi\Sigma_t^{(TD)})(s,x)-\Sigma_t^{(TD)}(x,s)\right).}
$$

We will break down Eq. (11) into 9 parts and simplify each of them separately.

$$
\begin{aligned}
(11)&=\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}R_l\right)\right] && (12)\\
&+\gamma\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(X_l')\right)\right] && (13)\\
&-\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(x)\right)\right] && (14)\\
&+\gamma\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(S_k')\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}R_l\right)\right] && (15)\\
&+\gamma^2\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(S_k')\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(X_l')\right)\right] && (16)\\
&-\gamma\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(S_k')\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(x)\right)\right] && (17)\\
&-\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(s)\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}R_l\right)\right] && (18)\\
&-\gamma\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(s)\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(X_l')\right)\right] && (19)\\
&+\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(s)\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(x)\right)\right]. && (20)
\end{aligned}
$$

Consider Eq. (12)

$$\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}R_l\right)\right] = \mathbb{E}_{seeds}\left[\mathbb{E}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}R_l\right)|seed\right]\right],$$

$$= \mathbb{E}_{seeds}\left[\mathbb{E}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{I}_{S_l=x}R_l\right)|seed\right]\right] + \mathbb{E}_{seeds}\left[\mathbb{E}\left[\left(\sum_{m=1}^{N}\mathbb{I}_{S_m=s}\mathbb{I}_{S_m=x}R_mR_m\right)|seed\right]\right],$$

$$= \mathbb{E}_{seeds}\left[\mathbb{E}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)|seed\right]\mathbb{E}\left[\left(\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{I}_{S_l=x}R_l\right)|seed\right]\right] + \mathbb{I}_{s=x}\mathbb{E}_{seeds}\left[\sum_{m=1}^{N}\widehat{d_\pi}(s)\mathbb{E}[R_m^2|seed, S_m=s]\right],$$

$$= \mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\widehat{d_\pi}(s)\widehat{r_\pi}(s)\sum_{\substack{l=1,\\ l\neq k}}^{N}\widehat{d_\pi}(x)\widehat{r_\pi}(x)\right] + \mathbb{I}_{s=x}\sum_{m=1}^{N}\mathbb{E}_{seeds}[\widehat{d_\pi}(s)(\widehat{c(s)} + \widehat{r_\pi}^{\,2}(s)],$$

$$= \sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{E}_{seeds}[\widehat{d_\pi}(s)\widehat{r_\pi}(s)\widehat{d_\pi}(x)\widehat{r_\pi}(x)] + \mathbb{I}_{s=x}Nd_\pi(s)\left(c(s) + r_\pi^2(s)\right),$$

$$\boxed{= N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)r_\pi(x) + \mathbb{I}_{s=x}Nd_\pi(s)\left(c(s) + r_\pi^2(s)\right).}$$

We have used the i.i.d assumption to separate the terms. Later, we used the indicator function to indicate the part which is non-zero. Also, we have used the definition of the second moment in the penultimate step.

We do a similar simplification to Eq. (13) now.

$$\gamma\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(X_l')\right)\right]$$

$$= \gamma\mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{I}_{S_k=s}\mathbb{I}_{S_l=x}R_kV_t^{(TD)}(X_l')\right] + \gamma\mathbb{E}_{seeds}\left[\sum_{m=1}^{N}\mathbb{I}_{S_m=s}\mathbb{I}_{S_m=x}R_mV_t^{(TD)}(X_m')\right],$$

$$= \gamma\mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{E}[\mathbb{I}_{S_k=s}\mathbb{I}_{S_l=x}R_kV_t^{(TD)}(X_l')|seed]\right] + \gamma\mathbb{I}_{s=x}\sum_{m=1}^{N}\mathbb{E}_{seeds}[\mathbb{E}[\mathbb{I}_{S_m=s}\mathbb{I}_{S_m=x}R_mV_t^{(TD)}(X_m')|seed]],$$

$$= \gamma\mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\widehat{d_\pi}(s)\widehat{d_\pi}(x)\widehat{r_\pi}(s)(\widehat{\mathcal{P}_\pi V_t^{(TD)}})(x)\right] + \gamma\mathbb{I}_{s=x}N\mathbb{E}_{seeds}[\widehat{d_\pi}(s)r_\pi(s)(\widehat{\mathcal{P}_\pi V_t^{(TD)}})(s)],$$

$$\boxed{= \gamma N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)(\mathcal{P}_\pi m_t^{(TD)})(x) + \gamma\mathbb{I}_{s=x}Nd_\pi(s)r_\pi(s)(\mathcal{P}_\pi m_t^{(TD)})(s)}$$

We will simplify Eq. (14) now.

$$\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}R_k\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(x)\right)\right]$$

$$= \mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{I}_{S_k=s}\mathbb{I}_{S_l=x}R_k V_t^{(TD)}(x)\right] + \mathbb{E}_{seeds}\left[\sum_{m=1}^{N}\mathbb{I}_{S_m=s}\mathbb{I}_{S_m=x}R_m V_t^{(TD)}(x)\right]$$

$$= \mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{E}[\mathbb{I}_{S_k=s}\mathbb{I}_{S_l=x}R_k V_t^{(TD)}(x)|seed]\right] + \mathbb{I}_{s=x}\sum_{m=1}^{N}\mathbb{E}_{seeds}[\mathbb{E}[\mathbb{I}_{S_m=s}\mathbb{I}_{S_m=x}R_m V_t^{(TD)}(x)|seed]],$$

$$= \mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\widehat{d_\pi}(s)\widehat{d_\pi}(x)\widehat{r_\pi}(s)V_t^{(TD)}(x)\right] + \mathbb{I}_{s=x}N\mathbb{E}_{seeds}[\widehat{d_\pi}(s)r_\pi(s)V_t^{(TD)}(s)],$$

$$\boxed{= N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)m_t^{(TD)}(x) + \mathbb{I}_{s=x}Nd_\pi(s)r_\pi(s)m_t^{(TD)}(s)}$$

The next expression Eq. (15) is similar to Eq. (13) and therefore, we skip the simplification details.

$$\gamma\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(S'_k)\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}R_l\right)\right]$$

$$\boxed{= \gamma N(N-1)d_\pi(s)d_\pi(x)r_\pi(x)(\mathcal{P}_\pi m_t^{(TD)})(s) + \gamma\mathbb{I}_{s=x}Nd_\pi(s)r_\pi(s)(\mathcal{P}_\pi m_t^{(TD)})(s)}$$

Now, we simplify (16),

$$\gamma^2\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(S'_k)\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(X'_l)\right)\right]$$

$$= \gamma^2\mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{I}_{S_k=s}\mathbb{I}_{S_l=x}V_t^{(TD)}(S'_k)V_t^{(TD)}(X'_l)\right] + \gamma^2\mathbb{E}_{seeds}\left[\sum_{m=1}^{N}\mathbb{I}_{S_m=s}\mathbb{I}_{S_m=x}V_t^{(TD)}(S'_m)V_t^{(TD)}(X'_m)\right]$$

$$= \gamma^2\mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\mathbb{E}[\mathbb{I}_{S_k=s}\mathbb{I}_{S_l=x}V_t^{(TD)}(S'_k)V_t^{(TD)}(X'_l)|seed]\right]$$
$$+ \gamma^2\mathbb{I}_{s=x}\sum_{m=1}^{N}\mathbb{E}_{seeds}[\mathbb{E}[\mathbb{I}_{S_m=s}\mathbb{I}_{S_m=x}V_t^{(TD)}(S'_m)V_t^{(TD)}(X'_m)|seed]],$$

$$= \gamma^2\mathbb{E}_{seeds}\left[\sum_{k=1}^{N}\sum_{\substack{l=1,\\ l\neq k}}^{N}\widehat{d_\pi}(s)\widehat{d_\pi}(x)(\widehat{\mathcal{P}_\pi}V_t^{(TD)})(s)(\widehat{\mathcal{P}_\pi}V_t^{(TD)})(x)\right]$$
$$+ \gamma^2\mathbb{I}_{s=x}\sum_{m=1}^{N}\mathbb{E}_{seeds}[\widehat{d_\pi}(s)\ (\widehat{\mathcal{P}_\pi}V_t^{(TD)})(s)(\widehat{\mathcal{P}_\pi}V_t^{(TD)})(x)],$$

$$\boxed{= \gamma^2N(N-1)d_\pi(s)d_\pi(x)(\mathcal{P}_\pi\Sigma_t^{(TD)}\mathcal{P}_\pi^T)(s,x) + \gamma^2\mathbb{I}_{s=x}Nd_\pi(s)(\mathcal{P}_\pi diag(\Sigma_t^{(TD)}))(s)}$$

We now consider the next piece Eq. (17),

$$\gamma\mathbb{E}_{seeds}\left[\left(\sum_{k=1}^{N}\mathbb{I}_{S_k=s}V_t^{(TD)}(S'_k)\right)\left(\sum_{l=1}^{N}\mathbb{I}_{S_l=x}V_t^{(TD)}(x)\right)\right]$$

$$= \gamma \mathbb{E}_{seeds} \left[\sum_{k=1}^{N} \sum_{\substack{l=1, \\ l \neq k}}^{N} \mathbb{I}_{S_k=s} \mathbb{I}_{S_l=x} V_t^{(TD)}(S'_k) V_t^{(TD)}(x) \right] + \gamma \mathbb{E}_{seeds} \left[\sum_{m=1}^{N} \mathbb{I}_{S_m=s} \mathbb{I}_{S_m=x} V_t^{(TD)}(S'_m) V_t^{(TD)}(x) \right]$$

$$= \gamma \mathbb{E}_{seeds} \left[\sum_{k=1}^{N} \sum_{\substack{l=1, \\ l \neq k}}^{N} \mathbb{E}[\mathbb{I}_{S_k=s} \mathbb{I}_{S_l=x} V_t^{(TD)}(S'_k) V_t^{(TD)}(x) | seed] \right]$$

$$+ \gamma \mathbb{I}_{s=x} \sum_{m=1}^{N} \mathbb{E}_{seeds}[\mathbb{E}[\mathbb{I}_{S_m=s} \mathbb{I}_{S_m=x} V_t^{(TD)}(S'_m) V_t^{(TD)}(x) | seed]],$$

$$= \gamma \mathbb{E}_{seeds} \left[\sum_{k=1}^{N} \sum_{\substack{l=1, \\ l \neq k}}^{N} \widehat{d_\pi}(s) \widehat{d_\pi}(x) (\widehat{\mathcal{P}_\pi} V_t^{(TD)})(s) V_t^{(TD)}(x) \right] + \gamma \mathbb{I}_{s=x} \sum_{m=1}^{N} \mathbb{E}_{seeds}[\widehat{d_\pi}(s) \ (\widehat{\mathcal{P}_\pi} V_t^{(TD)})(s) V_t^{(TD)}(s)],$$

$$\boxed{= \gamma N(N-1) d_\pi(s) d_\pi(x) (\mathcal{P}_\pi \Sigma_t^{(TD)})(s,x) + \gamma \mathbb{I}_{s=x} N d_\pi(s) (\mathcal{P}_\pi \Sigma_t^{(TD)})(s,s).}$$

Simplification of Eq. (18) is similar to Eq. (14).

$$\mathbb{E}_{seeds} \left[\left(\sum_{k=1}^{N} \mathbb{I}_{S_k=s} V_t^{(TD)}(s) \right) \left(\sum_{l=1}^{N} \mathbb{I}_{S_l=x} R_l \right) \right]$$

$$\boxed{= N(N-1) d_\pi(s) d_\pi(x) r_\pi(x) m^{(TD)}(s) + \mathbb{I}_{s=x} N d_\pi(s) r_\pi(s) m^{(TD)}(s).}$$

Also, the simplification of Eq. (19) is similar to that of Eq. (17).

$$\gamma \mathbb{E}_{seeds} \left[\left(\sum_{k=1}^{N} \mathbb{I}_{S_k=s} V_t^{(TD)}(s) \right) \left(\sum_{l=1}^{N} \mathbb{I}_{S_l=x} V_t^{(TD)}(X'_l) \right) \right]$$

$$\boxed{= \gamma N(N-1) d_\pi(s) d_\pi(x) (\mathcal{P}_\pi \Sigma_t^{(TD)})^T(s,x) + \gamma \mathbb{I}_{s=x} N d_\pi(s) (\mathcal{P}_\pi \Sigma_t^{(TD)})^T(s,s).}$$

We will simplify the final part i.e., Eq. (20).

$$\mathbb{E}_{seeds} \left[\left(\sum_{k=1}^{N} \mathbb{I}_{S_k=s} V_t^{(TD)}(s) \right) \left(\sum_{l=1}^{N} \mathbb{I}_{S_l=x} V_t^{(TD)}(x) \right) \right]$$

$$= \mathbb{E}_{seeds} \left[\sum_{k=1}^{N} \sum_{\substack{l=1, \\ l \neq k}}^{N} \mathbb{I}_{S_k=s} \mathbb{I}_{S_l=x} V_t^{(TD)}(s) V_t^{(TD)}(x) \right] + \mathbb{E}_{seeds} \left[\sum_{m=1}^{N} \mathbb{I}_{S_m=s} \mathbb{I}_{S_m=x} V_t^{(TD)}(s) V_t^{(TD)}(x) \right]$$

$$= \mathbb{E}_{seeds} \left[\sum_{k=1}^{N} \sum_{\substack{l=1, \\ l \neq k}}^{N} \mathbb{E}[\mathbb{I}_{S_k=s} \mathbb{I}_{S_l=x} V_t^{(TD)}(s) V_t^{(TD)}(x) | seed] \right]$$

$$+ \gamma \mathbb{I}_{s=x} \sum_{m=1}^{N} \mathbb{E}_{seeds}[\mathbb{E}[\mathbb{I}_{S_m=s} \mathbb{I}_{S_m=x} V_t^{(TD)}(s) V_t^{(TD)}(x) | seed]],$$

$$= \mathbb{E}_{seeds} \left[\sum_{k=1}^{N} \sum_{\substack{l=1, \\ l \neq k}}^{N} \widehat{d_\pi}(s) \widehat{d_\pi}(x) V_t^{(TD)}(s) V_t^{(TD)}(x) \right] + \gamma \mathbb{I}_{s=x} \sum_{m=1}^{N} \mathbb{E}_{seeds}[\widehat{d_\pi}(s) V_t^{(TD)}(s) V_t^{(TD)}(s)],$$

$$\boxed{= N(N-1) d_\pi(s) d_\pi(x) \Sigma_t^{(TD)}(s,x) + \gamma \mathbb{I}_{s=x} N d_\pi(s) \Sigma_t^{(TD)}(s,s).}$$

We can put all these pieces together to get the desired result.

$$
\begin{aligned}
\Sigma_t^{(TD)}(s,x) &= \Sigma_t^{(TD)}(s,x) + \alpha\Omega_t^{(TD)}(s,x) + \alpha^2\Psi_t^{(TD)}(s,x),\\
\Omega_t^{(TD)}(s,x) &= Nd_\pi(x)\left(r_\pi(x)m_t^{(TD)}(s) + \gamma(\mathcal{P}_\pi\Sigma_t^{(TD)})(x,s) - \Sigma_t^{(TD)}(s,x)\right)\\
&\quad + Nd_\pi(s)\left(r_\pi(s)m_t^{(TD)}(x) + \gamma(\mathcal{P}_\pi\Sigma_t^{(TD)})(s,x) - \Sigma_t^{(TD)}(x,s)\right),\\
\Psi_t^{(TD)}(s,x) &= N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)r_\pi(x) + \mathbb{I}_{s=x}Nd_\pi(s)\left(c(s) + r_\pi^2(s)\right)\\
&\quad + \gamma N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)(\mathcal{P}_\pi m_t^{(TD)})(x) + \gamma\mathbb{I}_{s=x}Nd_\pi(s)r_\pi(s)(\mathcal{P}_\pi m_t^{(TD)})(s)\\
&\quad - N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)m_t^{(TD)}(x) - \mathbb{I}_{s=x}Nd_\pi(s)r_\pi(s)m_t^{(TD)}(s)\\
&\quad + \gamma N(N-1)d_\pi(s)d_\pi(x)r_\pi(x)(\mathcal{P}_\pi m_t^{(TD)})(s) + \gamma\mathbb{I}_{s=x}Nd_\pi(s)r_\pi(s)(\mathcal{P}_\pi m_t^{(TD)})(s)\\
&\quad + \gamma^2 N(N-1)d_\pi(s)d_\pi(x)(\mathcal{P}_\pi\Sigma_t^{(TD)}\mathcal{P}_\pi^T)(s,x) + \gamma^2\mathbb{I}_{s=x}Nd_\pi(s)(\mathcal{P}_\pi diag(\Sigma_t^{(TD)}))(s)\\
&\quad - \gamma N(N-1)d_\pi(s)d_\pi(x)(\mathcal{P}_\pi\Sigma_t^{(TD)})(s,x) - \gamma\mathbb{I}_{s=x}Nd_\pi(s)(\mathcal{P}_\pi\Sigma_t^{(TD)})(s,s)\\
&\quad - N(N-1)d_\pi(s)d_\pi(x)r_\pi(x)m_t^{(TD)}(s) - \mathbb{I}_{s=x}Nd_\pi(s)r_\pi(s)m_t^{(TD)}(s)\\
&\quad - \gamma N(N-1)d_\pi(s)d_\pi(x)(\mathcal{P}_\pi\Sigma_t^{(TD)})^T(s,x) - \gamma\mathbb{I}_{s=x}Nd_\pi(s)(\mathcal{P}_\pi\Sigma_t^{(TD)})^T(s,s)\\
&\quad + N(N-1)d_\pi(s)d_\pi(x)\Sigma_t^{(TD)}(s,x) + \mathbb{I}_{s=x}Nd_\pi(s)\Sigma_t^{(TD)}(s,s)
\end{aligned}
$$

We will now prove the final part of the theorem. The mean squared error $\xi_{t+1}^{(TD)}$ can be decomposed into bias and covariance terms:

$$
\begin{aligned}
\xi_{t+1}^{(TD)} &= \sum_s (bias_{t+1}^2(s) + Cov_{t+1}(s,s)),\\
&= \sum_s \left((v_\pi(s) - m_{t+1}^{(TD)}(s)) + (\Sigma_{t+1}^{(TD)}(s,s) - m_{t+1}^{(TD)}(s)m_{t+1}^{(TD)}(s))\right),\\
&= \sum_s (v_\pi^2(s) - 2v_\pi(s)m_{t+1}^{(TD)}(s) + \Sigma_{t+1}^{(TD)}(s,s)),\\
&= \sum_s \left(v_\pi^2(s) - 2v_\pi(s)(m_t^{(TD)}(s) + \alpha\Delta_t^{(TD)}(s)) + \Sigma_t^{(TD)}(s,s) + \alpha\Omega_t^{(TD)}(s,s) + \alpha^2\Psi_t^{(TD)}(s,s)\right),\\
&= \sum_s \left(v_\pi^2(s) - 2v_\pi(s)m_t^{(TD)}(s) + \Sigma_t^{(TD)}(s,s)\right)\\
&\qquad + \sum_s \left(-2\alpha v_\pi(s)\Delta_t^{(TD)}(s)) + \alpha\Omega_t^{(TD)}(s,s) + \alpha^2\Psi_t^{(TD)}(s,s)\right),\\
&= \xi_t^{(TD)} + \alpha\sum_{s\in\mathcal{S}}(-2v_\pi(s)\Delta_t^{(TD)}(s) + \Omega_t^{(TD)}(s,s)) + \alpha^2\sum_{s\in\mathcal{S}}\Psi_t^{(TD)}(s,s).
\end{aligned}
$$

□

Theorem B.3.4. *Let $\mathbb{E}_{seeds}$ denote the expectation with respect to seeds. Let*

$$
\begin{aligned}
m_t^{(PT)}(s) &= \mathbb{E}_{seeds}[V_t^{(PT)}(s)],\\
Cov(V_t^{(PT)}(s), V_t^{(PT)}(x)) &= \Sigma_t^{(PT)}(s,x) - m_t^{(PT)}(s)m_t^{(PT)}(x),
\end{aligned}
$$

where $\Sigma_t^{(PT)}(s,x) = \mathbb{E}_{seeds}[V_t^{(PT)}(s)V_t^{(PT)}(s)]$. Then:

$$
\begin{aligned}
m_{t+1}^{(PT)}(s) &= m_t^{(PT)}(s) + \alpha Nd_\pi(s)\Delta_t^{(PT)}(s),\\
\Sigma_{t+1}^{(PT)}(s,x) &= \Sigma_t^{(PT)}(s,x) + \alpha\Omega_t^{(PT)}(s,x) + \alpha^2\Psi_t^{(PT)}(s,x),
\end{aligned}
$$

and, the mean squared error $\xi_t^{(PT)}$ at timestep t is

$$\xi_{t+1}^{(PT)} = \xi_t^{(PT)} + \alpha \sum_{s\in\mathcal{S}} d_\pi(s)(-2v_\pi(s)\Delta_t^{(PT)}(s) + \Omega_t^{(PT)}(s,s)) + \alpha^2 \sum_{s\in\mathcal{S}} d_\pi(s)\Psi_t^{(PT)}(s,s).$$

Proof. The proof is similar to that of TD learning algorithm. We have:

$$\begin{aligned}
m_{t+1}^{(PT)}(s) &= m_t^{(PT)}(s) + \alpha N d_\pi(s)(r_\pi(s) + \gamma(\mathcal{P}_\pi m_t^{(PT)})(s) - m_t^{(PT)}(s)),\\
\Sigma_t^{(PT)}(s,x) &= \Sigma_t^{(PT)}(s,x) + \alpha\Omega_t^{(PT)}(s,x) + \alpha^2\Psi_t^{(PT)}(s,x),\\
\Omega_t^{(PT)}(s,x) &= N d_\pi(x)\left(r_\pi(x)m_t^{(PT)}(s) + \gamma(\mathcal{P}_\pi\Sigma_t^{(PT)})(x,s) - \Sigma_t^{(PT)}(s,x)\right)\\
&\quad + N d_\pi(s)\left(r_\pi(s)m_t^{(PT)}(x) + \gamma(\mathcal{P}_\pi\Sigma_t^{(PT)})(s,x) - \Sigma_t^{(PT)}(x,s)\right),\\
\Psi_t^{(PT)}(s,x) &= N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)r_\pi(x) + \mathbb{I}_{s=x}N d_\pi(s)\left(c(s) + r_\pi^2(s)\right)\\
&\quad + \gamma N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)(\mathcal{P}_\pi m_t^{(PT)})(x) + \gamma\mathbb{I}_{s=x}N d_\pi(s)r_\pi(s)(\mathcal{P}_\pi m_t^{(PT)})(s)\\
&\quad - N(N-1)d_\pi(s)d_\pi(x)r_\pi(s)m_t^{(PT)}(x) - \mathbb{I}_{s=x}N d_\pi(s)r_\pi(s)m_t^{(PT)}(s)\\
&\quad + \gamma N(N-1)d_\pi(s)d_\pi(x)r_\pi(x)(\mathcal{P}_\pi m_t^{(PT)})(s) + \gamma\mathbb{I}_{s=x}N d_\pi(s)r_\pi(s)(\mathcal{P}_\pi m_t^{(PT)})(s)\\
&\quad + \gamma^2 N(N-1)d_\pi(s)d_\pi(x)(\mathcal{P}_\pi\Sigma_t^{(PT)}\mathcal{P}_\pi^T)(s,x) + \gamma^2\mathbb{I}_{s=x}N d_\pi(s)(\mathcal{P}_\pi diag(\Sigma_t^{(PT)}))(s)\\
&\quad - \gamma N(N-1)d_\pi(s)d_\pi(x)(\mathcal{P}_\pi\Sigma_t^{(PT)})(s,x) - \gamma\mathbb{I}_{s=x}N d_\pi(s)(\mathcal{P}_\pi\Sigma_t^{(PT)})(s,s)\\
&\quad - N(N-1)d_\pi(s)d_\pi(x)r_\pi(x)m_t^{(PT)}(s) - \mathbb{I}_{s=x}N d_\pi(s)r_\pi(s)m_t^{(PT)}(s)\\
&\quad - \gamma N(N-1)d_\pi(s)d_\pi(x)(\mathcal{P}_\pi\Sigma_t^{(PT)})^T(s,x) - \gamma\mathbb{I}_{s=x}N d_\pi(s)(\mathcal{P}_\pi\Sigma_t^{(PT)})^T(s,s)\\
&\quad + N(N-1)d_\pi(s)d_\pi(x)\Sigma_t^{(PT)}(s,x) + \mathbb{I}_{s=x}N d_\pi(s)\Sigma_t^{(PT)}(s,s)
\end{aligned}$$

The MSE recursive equation can be obtained similar to that of TD expression. Therefore, we skip the details and give the final expression.

$$\xi_{t+1}^{(PT)} = \xi_t^{(PT)} + \alpha\sum_{s\in\mathcal{S}}(-2v_\pi(s)\Delta_t^{(PT)}(s) + \Omega_t^{(PT)}(s,s)) + \alpha^2\sum_{s\in\mathcal{S}}\Psi_t^{(PT)}(s,s).$$

□

C Experiments

Besides providing details for the experiments described in the main paper, we answer additional interesting questions, such as:

- What is the effect of task distribution on the performance for various methods (Sec. C.2)?
- What is the effect of network capacity on the performance (Sec. C.3)?
- What is the effect of hyperparameters on the performance (Sec. C.4)?

C.1 Additional Details

We begin by providing details of the experiments.

C.1.1 Policy evaluation grids

We modify the rewards of the goal states as described in Table C.1. The true value function for each task is shown in Fig. C.1. We convert the 5x5 discrete grid into a 48x48 RGB image for the deep RL

Task ID	Reward			
	Top-Left	Top-Right	Bottom-Left	Bottom-Right
1	0	1	0	1
2	1	0	1	0
3	0	0	1	1
4	1	1	0	0

Table C.1: Rewards at goal states for various tasks.

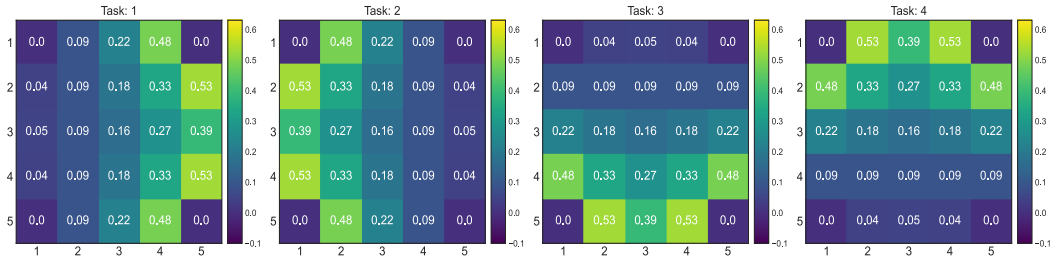


Figure C.1: Heatmap of the true value function for different tasks.

task. The agent's location is indicated by colouring the its location in red and we use green squares to indicate goal states. At every timestep, the agent receives a RGB image as an input. The action space and the action consequences are the same as discrete grid. We select best hyperparameters (learning rates) for each algorithm by choosing those values which results in the lowest area under the error curves as shown in Table C.6 and Fig. C.9. The hyperparameter search and the final reporting is performed by averaging the results on 30 different seeds. For our method, the buffer stores the samples from the latest task only to update the permanent value function.

For the first linear FA experiment, we use the same setup as the tabular setting except we encode each state in the grid as a vector of 10 bits. The first 5 bits encodes the row of the state and the last 5 bits encodes the column of the state. For example, the starting state (2,2) is represented as [0,0,1,0,0,0,0,1,0,0]. Both permanent value function and transient value function use same features to learn respective estimates.

For the second linear FA experiment, we use a continuous version of the grid. The goal rewards for each task is same as the discrete counterpart as described in Table C.1. The true value function is also different for each task and it is shown as a heatmap in Fig. C.2. We use RBFs to generate features as described in Sutton and Barto [45]. We use 676 centers (order^2) spaced equally in both x and y directions across the grid with a variance $(0.75/(order-1))^2$ and use a threshold to convert the continuous vector into a binary feature vector. Any feature value greater than 0.5 is set to 1, while those features with values less than 0.5 is set to 0. We search for the best learning rate from the same set of values as the discrete grid task. The hyperparameter curves are shown in Fig. C.11.

For the deep RL experiment, the rewards are multiplied by a factor of 10 to minimize the initialization effect of deep neural network on prediction. We used a convolution neural network (CNN) to approximate the value function for all algorithms tested. The CNN consists of two conv layers with

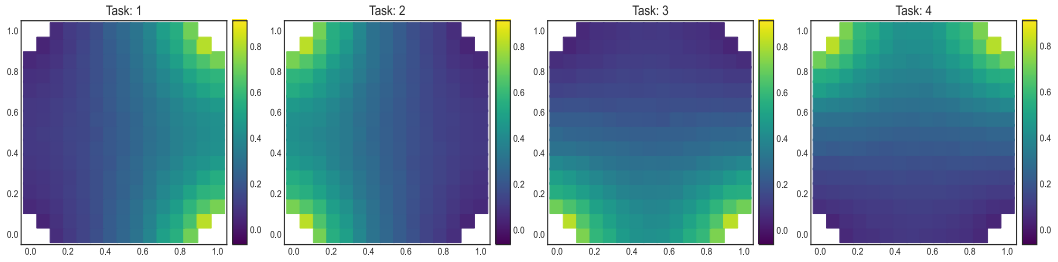


Figure C.2: Heatmap of the true value function for different tasks.

8 filters, (2,2) stride, and relu activation in each layer. We use a max pooling layer in between the two conv layers with a stride 2. The output of the second conv layer is flatted, and it is followed by a couple of linear layers. The fully connected network has 64 and 1 hidden units in the respective layers. The first fully connected layer uses a tanh activation and no activation is used in the final layer. For our method, we attach a new head after the first two conv layers for permanent value function. This head also has two linear layers where the first layer has tanh activation. The entire network is trained by backpropagating the errors from the transient value function. But, the conv layers are not trained when updating the permanent value function. The transient value function and the TD learning algorithm are trained in an online fashion using an experience replay buffer of size 100k. The buffer is reset at the end of the task for all methods to reduce the bias from old samples. permanent value function is trained at the end of the task using the samples from the latest task. 100 gradient steps are taken during each update cycle. We use SGD optimizer to update its weights. The best learning rate is picked by computing the area under the error curve on a set of values shown in Table C.7. The mean AUC curves of various choices of learning rates are shown in Fig. C.12.

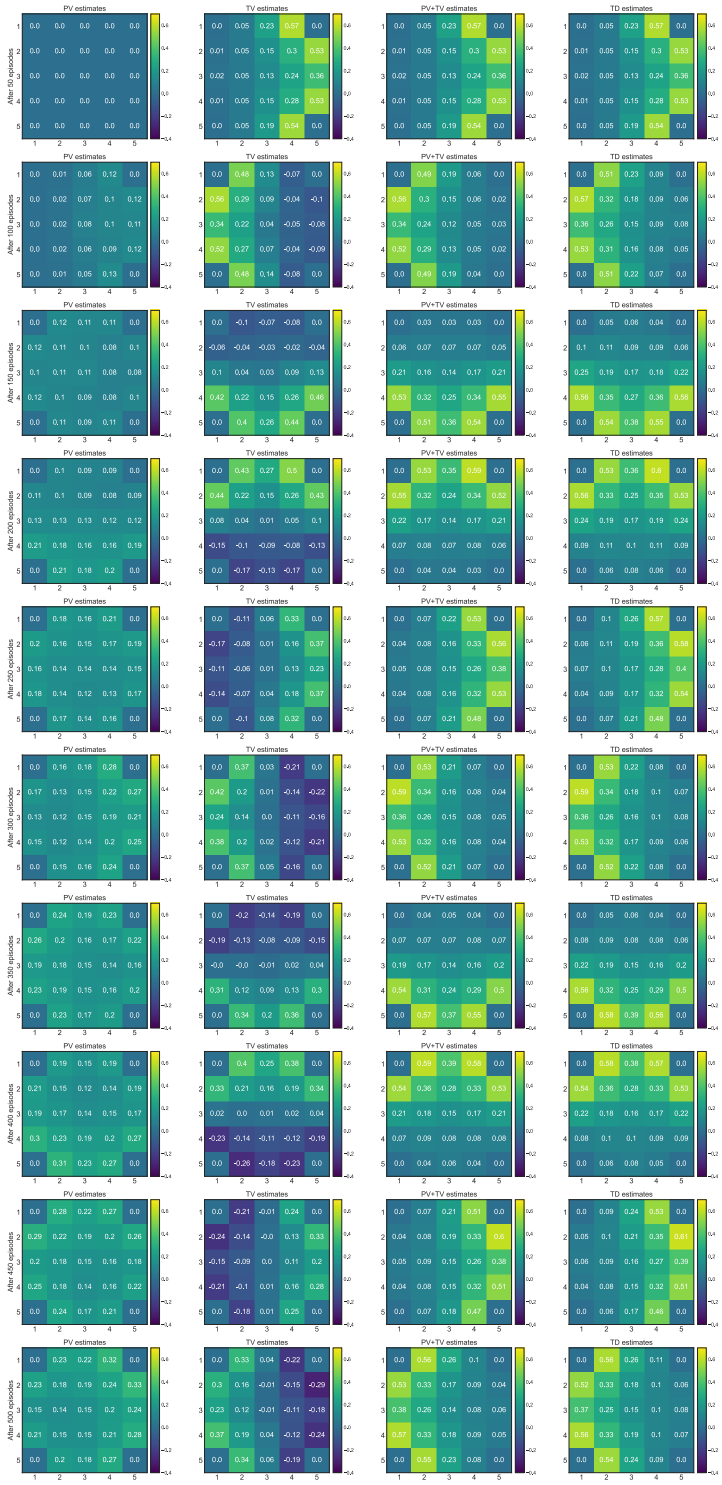


Figure C.3: Predictions evolution across episodes.

C.1.2 Policy evaluation minigrid

We use 4 rooms task as shown in Fig. 1b. The environment is a grid task with each room containing one type of item. Each item type is different in terms of how it is perceived and in terms of reward the agent receives upon collecting it. There are 4 goal cells, one in each room located at the corner. At

the beginning of the episode, the agent starts in one of the cells within the region spanning between cells (3,3) and (6,6). It has 3 actions to chose from turn left, turn right, and move forward. Turn-based actions changes the orientation of the agent along the corresponding direction, whereas the move action changes the agent's location to the state in front of it. The agent can only observe a 5x5 view in front of it (partial view). Items are one-hot represented along the 3rd dimension to make the overall input space 5x5x5. To set up the continual learning problem, we change the rewards of the items from task-to-task as described in Table C.2. The evaluation policy is kept uniformly random for all states and for all the tasks. We use a discount factor of 0.9. We evaluate the performance for a total of 750 episodes and the tasks are changes after every 75 episodes.

Task ID	Reward			
	Blue (●)	Red (●)	Yellow(●)	Purple (●)
1	-1	1	1	1
2	1	1	0.5	1
3	1	1	1	-1
4	-1	1	0.5	-1

Table C.2: Rewards for items for various tasks.

We used a convolution neural network (CNN) to approximate the value function for all algorithm. The CNN consists of three conv layers with 16, 32, and 64 filters, (2,2) stride and relu activation in each layer. The output of the third conv layer is flatted, which is then followed by a couple of linear layers with 64 and 1 hidden unit. Tanh activation in the first linear layer and no activation is used in the last layer. We attach a head after conv layers for permanent value function. The architecture of this head is same as the TD learning head. The training method and buffer capacities are same as the discrete image grid task. We use SGD optimizer to update weights in all networks. The best learning rate is picked by computing the area under the error curve on a set of values shown in Table C.8 and Fig. C.13. All the results reported are averaged across 30 seeds.

C.1.3 Control minigrid

We use two room environment as shown in Fig. 3c. The agent starts in one of the bottom row states in the bottom room and there are two goal states located in the top room. In the first task, the goal rewards are +5 and 0 for the first and second goals respectively, and it is flipped for the second task. We alternate between two tasks every 500 episodes for a total of 2500 episodes. The agent receives a one hot, 5x5 view (partial observability) in front of it at each timestep. It can take one of the three actions (move forward, turn left, turn right) at every step. The *move* action transitions the agent to the state in front of it if there is no obstacle. The environment is deterministic and episodic, and we use a discount factor of 0.99.

We use a CNN as a function approximator. The CNN has 3 conv layers with 16, 32, and 64 filters, (2,2) stride, and relu activations. It is followed by a fully connected neural network with 128, 64, and 4 hidden units. The first layer is relu activated, the second layer is tanh activated, and the final layer is linear. We use target network and experience replay to train the network. Target network's weights are updated every 200 timesteps. Experience replay buffer's capacity is capped to 100k and the stored samples are deleted when the task changes. We use batch size of 64 and update the network every timestep. We search for the best learning rate from a set of value shown in Table C.10 on 3 seeds and we pick the one that gave the highest area under the rewards curve. We then run on 30 seeds using that learning rate to report the final performance. The transient value function training procedure and details are identical to DQN. permanent value function uses a separate head following conv layers and is updated using Adam optimizer whenever the task is changed.

C.1.4 PT-DQN Pseudocode

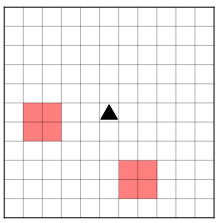
Algorithm 4 PT-DQN Pseudocode (Continual Reinforcement Learning)

Initialize transient network buffer $\mathcal{B}$, permanent network buffer $\mathcal{D}$
Initialize permanent network θ, transient network $\mathbf{w}$
Initialize target network $\mathbf{w}_{target}$
for $t : 0 \rightarrow \infty$ **do**
  Take action A_t according to ϵ-greedy policy
  Observe R_{t+1}, S_{t+1}
  Store $(S_t, A_t, R_{t+1}, S_{t+1})$ in $\mathcal{B}$
  Store $(S_t, A_t, Q^{(P)}(S_t, A_t))$ in $\mathcal{D}$
  Sample a batch of transitions $(S_j, A_j, R_{j+1}, S_{j+1})$ from B
  Compute target
  $y = R_{j+1} + \gamma \max_{a' \in \mathcal{A}} \left(Q^{(P)}(S_{j+1}, a'; \theta) + Q^{(T)}(S_{j+1}, a'; \mathbf{w}_{target})\right) - Q^{(P)}(S_j, A_j; \theta)$
  Perform a gradient step on $\left(y - Q^{(T)}(S_j, A_j; \mathbf{w})\right)^2$ with respect to $\mathbf{w}$
  Update target network every N steps, $\mathbf{w}_{target} = \mathbf{w}$
  if $mod(t, k) == 0$ **then**
    Sample a batch of transitions $(S_j, A_j, Q^{(P)}(S_j, A_j))$ from D
    Compute target $\hat{y} = Q^{(P)}(S_j, A_j) + Q^{(T)}(S_j, A_j; \mathbf{w})$
    Perform a gradient step on $\left(\hat{y} - Q^{(P)}(S_j, A_j; \theta)\right)^2$ with respect to θ
    Decay transient network weights, $\mathbf{w} = \lambda \mathbf{w}$
    Clear permanent network buffer, $D = \{\}$
  end if
end for

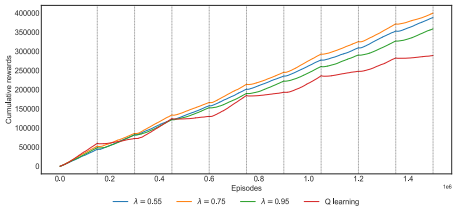
C.1.5 JellyBeanWorld (JBW)

Item		Blue (●)	Green (●)	Red (●)
Color		[0, 0, 1.0]	[0, 1.0, 0]	[1.0, 0, 0.0]
Intensity		Constant [-2.0]	Constant [-6.0]	Constant [-2.0]
Interactions	●	Piecewise box [2, 100, 10, -5]	Zero	Piecewise box [150, 0, -100, -100]
	●	Zero	Zero	Zero
	●	Piecewise box [150, 0, -100, -100]	Zero	Piecewise box [2, 100, 10, -5]

Table C.3: JellyBeanWorld environment parameters.



(a) Sample observation received by the agent.



(b) Cummulative rewards against timesteps averaged across 3 seeds.

Figure C.4: JellyBeanWorld sample observation and performance curves for various decay values.

We test algorithms on a continual learning problem in the JBW testbed [31] and use neural networks as the function approximator. The environment is a two-dimensional infinite grid with three types of items: blue, red, and green. We induce spatial non-stationarity by setting the parameters of the environment as described in Table C.3. In our configuration, three to four similar items (red or blue) form a tiny group and similar groups appear close to each other to form a sea of red (or blue) objects as shown in Fig. 5a. Therefore, the local observation distribution at various parts of the environment are different. At each timestep, the agent receives a egocentric 11x11 RGB, 360^o view

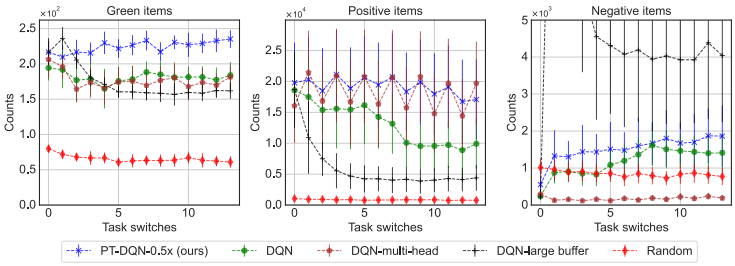


Figure C.5: Items collected by various algorithms during each task switches.

as an observation as shown in Fig. C.4a. It has four actions to choose from: up, down, left, and right. Each action transitions it by one square along the corresponding direction. The rewards for green items, which are uniformly distributed, are set to 0.1, whereas the rewards for picking red and blue items alternate between -1 and +2 every 150k timesteps, therefore, inducing reward non-stationarity. We run the experiment for 2.1 million timesteps and train algorithms using a discount factor 0.9.

The DQN agent uses four layered neural network with relu activations in all layers except the last to estimate q-values. The number of hidden units are 512, 256, 128, and 4 respectively. The network is trained using Adam optimizer with experience replay and a target network. We use epsilon-greedy policy with $\epsilon = 0.1$ for exploration. The permanent and transient network architectures are identical to that of DQN but with half the number of parameters in each layer. This ensures that the total number of parameters for the baselines and our method is same. Transient network is also trained using Adam optimizer, whereas, permanent network is updated every 10k timesteps (k=10,000) using SGD optimizer. Target network's weights are updated every 200 timesteps. Experience replay buffer capacity is capped to 8000 as higher capacity buffer stores old training samples which affects the training when the environment is changing continuously. We use batch size of 64 and update the network every timestep. For our approach, we experimented with three choices of λ values (0.55, 0.75, and 0.95) and all the choices resulted in a better performance than the DQN agent as seen in Fig. C.4b. We use the same exploration policy as the DQN agent. We flatten the observation into one long array, then stack the last four observation arrays to form an input to the neural network. The best learning rate for all algorithms are chosen by comparing the total reward obtained in 1.5 million timesteps for a set of fixed learning rate values on 3 different seeds. We search for the best learning rate from a set of value shown in Table C.12 on 3 seeds and we pick the one that gave the highest rewards. We then run on 30 seeds using that learning rate to report the final performance. The random agent picks one of the four actions according to uniform probability at each timestep. For the multi-head DQN baseline, we attach two output heads - one for each task. The baseline knows the task IDs, so it chooses the appropriate head to select actions and to train the network. The replay buffer is reset for this baseline whenever the task is changed. We use a buffer capacity of 250k for the DQN baseline that is augmented with a large buffer. The rest of the training and architecture details are same as the DQN agent.

C.1.6 MinAtar

This domain is a standard benchmark for testing single task RL algorithms [51]. We use breakout, freeway, and space invaders environments to setup a continual RL problem by randomly picking one of these tasks every 500k steps for a total of 3.5M steps. We standardize the observation space across tasks by padding 0s when needed and make all the actions available for the agent. This problem has non-stationarity in transitions besides the reward and observation signals making it challenging. We use a discount factor of 0.99 for this experiment.

We used a CNN which has 1 relu activated conv layer with 16 filters followed by a relu activated linear layer with 256 units, and a linear output layer for DQN. The architecture is same as the one used in the original paper [51]. The architecture is same for other DQN variants but we add 3 output layers for the multi-headed DQN. For our method, we use the same architecture but 8 filter in the conv layer and 128 units in the linear layer. Target network is used for all the methods which is updated every 1000 steps and we use an experience replay of 100k for all methods but we use 500k sized experience replay for large replay based DQN baseline. We use Adam optimizer to train the networks in online fashion. For our method, permanent value function network is updated using SGD

optimizer every 50k steps. The best hyperparameters are chosen by running the experiment for 1.5M steps by switching the tasks every 500k steps on 3 seeds. The results of hyperparameter tuning are detailed in Tables C.17, C.18, C.19, and C.20. We make all 7 actions available to the agent at all times and the input to the agent is 10x10x7 images.

C.2 What is the effect of task distribution on the performance for various methods?

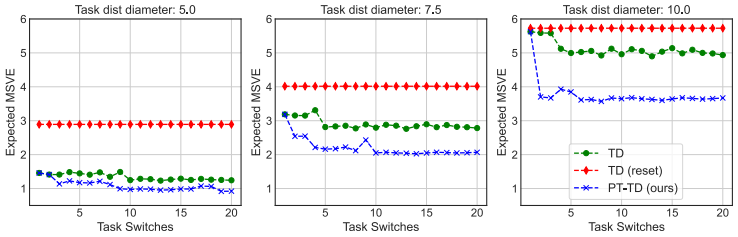


Figure C.6: Expected analytic MSVE on a task for various algorithms and ϵ values.

Setup: We set up a continual learning problem by creating 7 tasks whose value functions are ϵ (task diameter) distance from one another in norm-2. Each task has 7 states. For a particular ϵ value, for a particular number of task switches, we run each algorithm 100 times and save the final estimates which form the initial point for the new task. We then sample a task according to uniform distribution and use analytical MSE expression derived in the previous section to analyze the behaviour of PT-TD learning and the two versions of TD learning algorithm for 100 rounds. We then average the error across rounds and across different starting values to get a point in the plot shown in Fig. C.6.

Task details: We use the following $\mathcal{P}_\pi$ for all tasks and for all task diameters (ϵ):

$$\mathcal{P}_\pi = \begin{bmatrix} 0.18 & 0.19 & 0.08 & 0.16 & 0.1 & 0.11 & 0.18 \\ 0.04 & 0.05 & 0.01 & 0.0 & 0.4 & 0.0 & 0.51 \\ 0.0 & 0.2 & 0.0 & 0.05 & 0.28 & 0.06 & 0.41 \\ 0.17 & 0.14 & 0.09 & 0.26 & 0.15 & 0.19 & 0.01 \\ 0.16 & 0.16 & 0.16 & 0.24 & 0.17 & 0.0 & 0.11 \\ 0.25 & 0.02 & 0.24 & 0.24 & 0.08 & 0.05 & 0.11 \\ 0.0 & 0.25 & 0.19 & 0.44 & 0.05 & 0.0 & 0.07 \end{bmatrix}$$

We then sample 7 points in $\mathbb{R}^n$ that are ϵ units from one another. These points form the true value function of the tasks. We then compute the reward vector for each task τ as $r_\pi(\tau) = (I - \gamma\mathcal{P}_\pi)v_\pi(\tau)$. At the time of testing the algorithms, we add a zero mean Gaussian noise with 0.1 standard deviation to these rewards to simulate stochasticity. We search from the learning rates shown in Table C.4 to pick the best one for each combination of ϵ and number of task switches. We use a learning rate of 0.01 while using analytical expressions.

Observation: We observe both TD learning algorithm and our approach has the same performance after seeing data from one task. But the error gap increases as the number of task switches increase. The low error in our method can be attributed to the presence of permanent value function which captures some portion of the value that is common for all tasks. transient value function then learns the remaining part with a small number of samples resulting in low error overall. We also observe that the error gap between TD learning and our method gets wider as we increase task diameter, which demonstrates that our method is broadly applicable. The reset variant of the TD learning algorithm whose predictions are reset whenever task changes has the largest error emphasizing learning from scratch could be costly.

Algorithm	LRs
TD (without reset)	{8e-1 5e-1 3e-1 1e-1 5e-2 1e-2}
PT-TD	PV: {5e-1 3e-1 1e-1 5e-2 1e-2 5e-3}
	TV: {8e-1 5e-1 3e-1 1e-1 5e-2 1e-2}

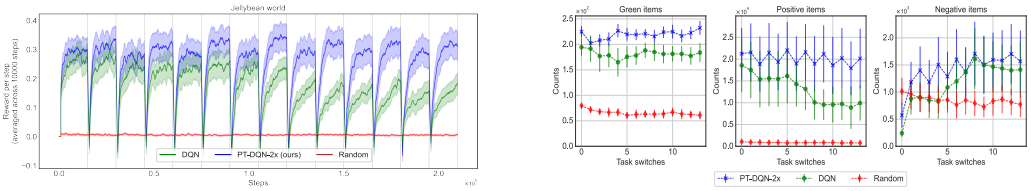
Table C.4: LRs considered to pick the best one.

C.3 What is the effect of network capacity on the performance?

Setup: We use JBW task in the continual RL setting to perform the ablation study. We run two additional experiments by varying the number of parameters in DNN for the DQN and our method. In the first experiment, we use the same DNN architecture for all algorithms, therefore, our method has twice the number of parameters as that of DQN. In the second experiment, we double the network capacity for the DQN variant making the total number of parameters same as ours. We keep the rest of the experimental setup same as before and we pick the best learning rate based on the AUC by searching over the values detailed in Table C.5 over 3 seeds for 1.5M steps.

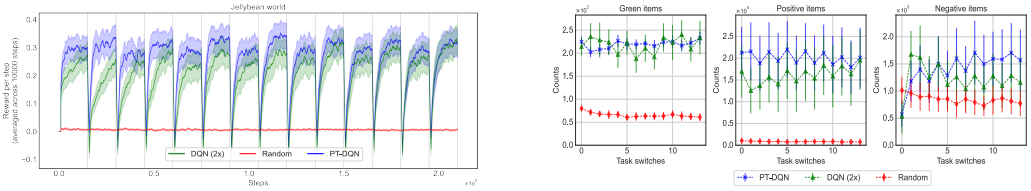
Algorithm	LRs
DQN (2x)	{1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7}
PT-DQN	PV: {5e-1 3e-1 1e-1 5e-2 1e-2 5e-3}
	TV: {8e-1 5e-1 3e-1 1e-1 5e-2 1e-2}

Table C.5: Search space of learning rates.



(a) Results on the JellyBeanWorld. (b) Items collected in each task.

Figure C.7: (a) Rewards accumulated per timestep across 10k step window is plotted against timesteps. (b) Number of items collected within a task for each type (subplots use different scales along y-axis).



(a) Results on the JellyBeanWorld. (b) Items collected in each task.

Figure C.8: (a) Rewards accumulated per timestep across 10k step window is plotted against timesteps. (b) Number of items collected within a task for each type (subplots use different scales along y-axis).

Observations: The results for ablation experiments are shown in Figures C.7 and C.8. We observe that our algorithm continues to perform better than the DQN agent in both the cases. The results also indicate that it is efficient to devote the available capacity in learning parts of the value function rather than learning the whole.

In the second ablation experiment, the DQN agent with twice the number of parameters starts off learning slowly but catches up with our method after seeing enough data. In fact, the DQN agent continues to perform similar to our method without any drop in performance as we continue to train further (we tested it by running the agent to 6M timesteps). This result highlights that our approach is beneficial in the *big world - small agent* setup where the computation budget of the agent is very small compared to the complexity of the environment. When the agent's capacity is large relative to the complexity of the environment, there's no additional benefit (neither there is any downside) to our method. Since the world is much more complicated in comparison to the agent's capacity, using our method is preferable.

These experiments also provide partial answer to the question - when does the plasticity problem arise in neural networks? We think that a combination of learning algorithm (like DQN), small agent

capacity, and a complex environment could give rise to the plasticity problem in neural networks. A thorough investigation is necessary to confirm this hypothesis.

C.4 What is the effect of hyperparameters on the performance?

In this section we analyze the sensitivity of additional hyperparameter in our method (learning rate to update permanent value function) and compare it with the sensitivity of TD and Q-learning algorithms on various environments.

Algorithm	LRs
TD (without reset)	{8e-1, 5e-1, 3e-1, 1e-1, 5e-2, 1e-2}
TD (with reset)	{8e-1, 5e-1, 3e-1, 1e-1, 5e-2, 1e-2}
PT-TD	PV: {1e-1, 5e-2, 1e-2, 5e-3, 1e-3}
	TV: {8e-1, 5e-1, 3e-1, 1e-1, 5e-2, 1e-2}

Table C.6: LRs considered in the search space for the policy evaluation grid tasks.

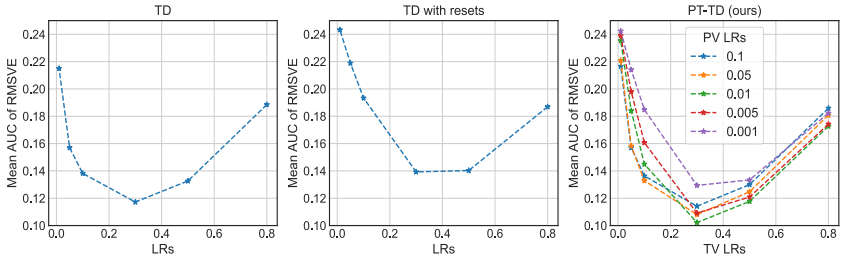


Figure C.9: Mean AUC of RMSVE against learning rate for various algorithms.

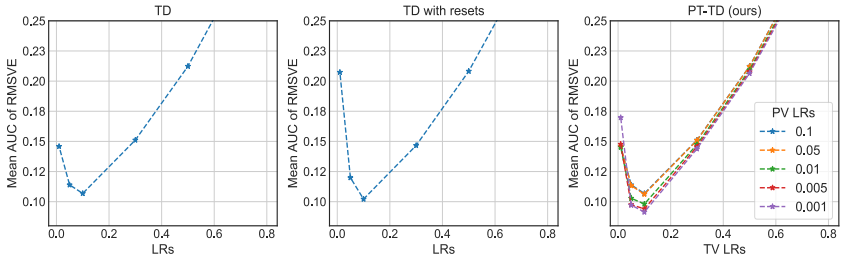


Figure C.10: Mean AUC of RMSVE against learning rate for various algorithms.

For tabular and linear grid tasks, we select best hyperparameters (learning rates) for each algorithm by choosing those values which results in the lowest area under the error curves as shown in Table C.6, Figures C.9, C.10, C.11. The hyperparameter search and the final reporting is performed by averaging the results on 30 different seeds. For the deep RL task with discrete grid, we choose the best learning rate from the values shown in Table C.7 and the corresponding U curves are shown in C.12. For the minigrid policy evaluation task, we search for the best learning rates from the values shown in Table C.8 and the corresponding U curves are shown in Fig. C.13.

For the tabular control experiment, the learning rates used to search for the best value is listed in Table C.9 and the mean returns for various choices of these values are plotted in Fig. C.14 for various algorithms. Similar table for the minigrid task (deep control) is shown in Table C.10 and the mean returns for various choices of these values is shown in Fig. C.15. We use 30 seeds to pick the best learning rate for the tabular task but use 3 seeds to select the learning rates for the minigrid task.

The learning rate tables and the corresponding performance for the full continual RL experiments are shown in Fig. C.16 and Tables C.11, C.12, C.17, C.18, C.19, and C.20. We use 30 seeds to find the best hyperparmeters for the tabular task, but we use 3 seeds for JBW experiment and MinAtar.

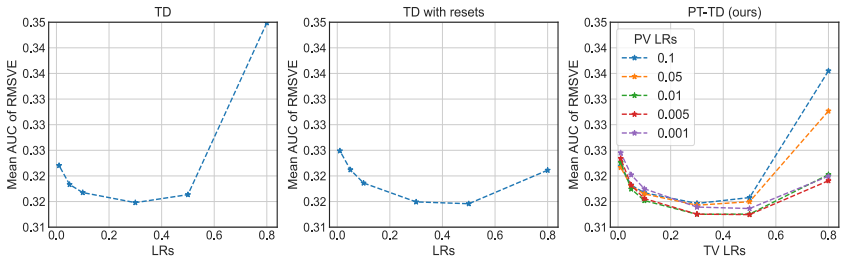


Figure C.11: Mean AUC of RMSVE against learning rate for various algorithms.

Algorithm	LRs
Deep TD (without reset)	{1e-1, 3e-2, 1e-2, 3e-3, 1e-3}
Deep TD (with reset)	{1e-1, 3e-2, 1e-2, 3e-3, 1e-3}
Deep PT-TD	PV: {1e-3, 3e-4, 1e-4, 3e-5}
	TV: {1e-2, 5e-3, 3e-3, 1e-3}

Table C.7: LRs considered to pick the best one for the Deep prediction experiment with discrete grid.

We find the best learning rate for each k, λ pair by searching over the learning rates listed in Table C.11.

Observations: We see that our algorithm performs better than their vanilla counterparts on all the problems we tested for a broad range of hyperparameters. There are a handful of $(\overline{\alpha}, \alpha)$ pairs that results in a better performance in each setting, which indicates that our method is not sensitive to hyperparameter selection. We also observe that in all the cases $\overline{\alpha}$ is smaller than α, which fits into our intuition. Because, the permanent value function has to capture regularities in the value function estimate, it needs to update slowly, whereas, the transient value function should be updated quickly to facilitate fast adaptation. The sensitivity of two new hyperparameters: λ, k for the fully continual RL setting is analyzed in the main paper (see Sec. 6.1) where we concluded that our algorithm is robust to these values.

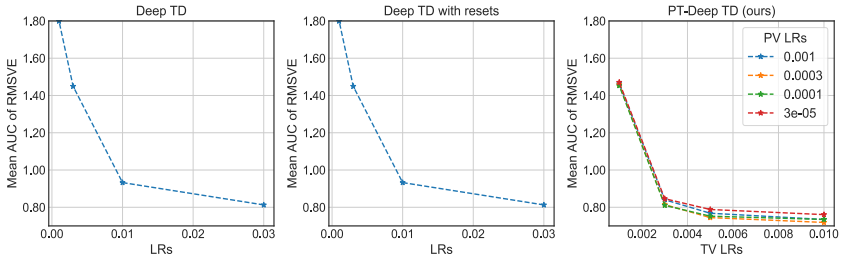


Figure C.12: Mean AUC of RMSVE against learning rate for various algorithms on image grid.

Algorithm	LRs
TD (without reset)	{1e-1, 3e-2, 1e-2, 3e-3, 1e-3}
TD (with reset)	{1e-1, 3e-2, 1e-2, 3e-3, 1e-3}
permanent value function	{3e-2, 1e-2, 3e-3, 1e-3, 3e-4}
transient value function	{3e-3, 1e-3, 3e-4, 1e-4, 3e-5, 1e-5, 3e-6, 1e-6}

Table C.8: Search space of learning rate for the policy evaluation minigrid task.

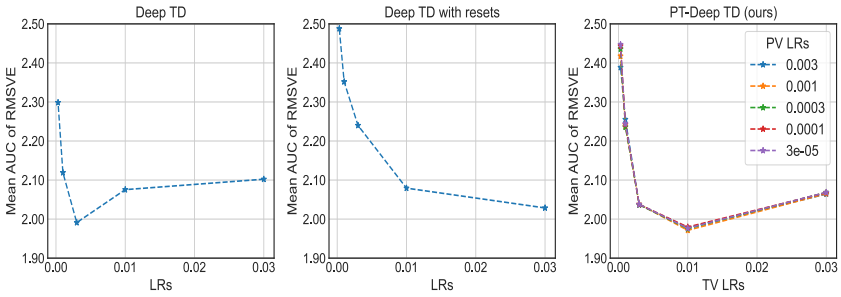


Figure C.13: Mean AUC of RMSVE against learning rate for various algorithms on minigrid grid.

Algorithm	LRs
Q-learning	{8e-1, 5e-1, 1e-1, 5e-2, 1e-2, 5e-3, 1e-3}
Q-learning (with reset)	{8e-1, 5e-1, 1e-1, 5e-2, 1e-2, 5e-3, 1e-3}
Permanent Value Function	{8e-1, 5e-1, 3e-1, 1e-1, 5e-2, 1e-2, 5e-3, 1e-3}
Transient Value Function	{8e-1, 5e-1, 3e-1, 1e-1, 5e-2, 1e-2}

Table C.9: Search space of learning rates for the tabular control task.

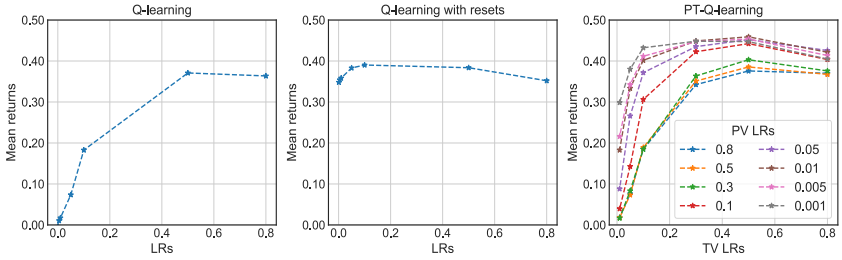


Figure C.14: Mean returns for various learning rates for the tabular control task.

Algorithm	LRs
Q-learning	{3e-3, 1e-3, 3e-4, 1e-4, 3e-5}
Q-learning (with reset)	{1e-2, 3e-3, 1e-3, 3e-4, 1e-4}
Permanent Value Function	{1e-5, 3e-6, 1e-6, 3e-7}
Transient Value Function	{3e-4, 1e-4, 3e-5, 1e-5}

Table C.10: Search space of learning rates.

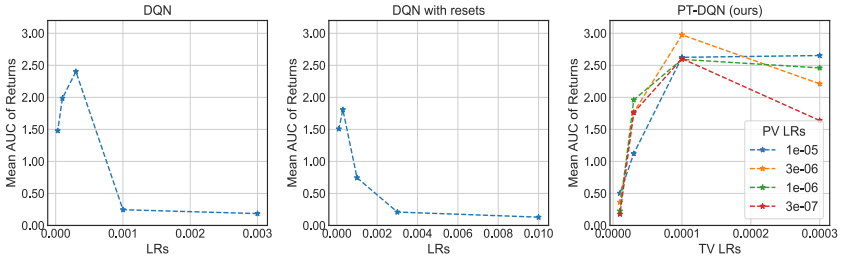


Figure C.15: Mean returns for various learning rates for the Minigrid control task.

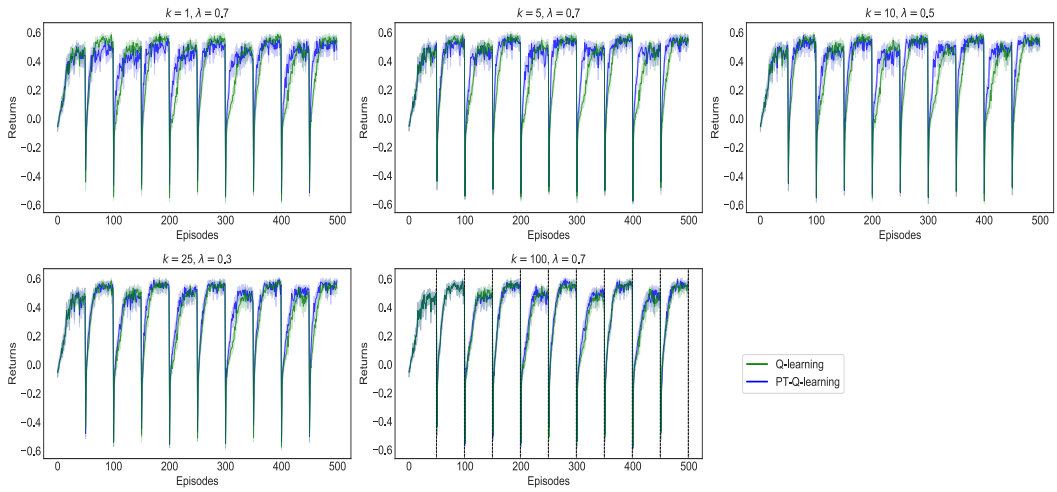


Figure C.16: Performance plot of PT-Q-learning against Q-learning for k, λ pairs

Algorithm	LRs
Q-learning	{8e-1, 5e-1, 1e-1, 5e-2, 1e-2, 5e-3, 1e-3}
Permanent Value Function (0.5x)	{1e-4, 1e-5, 1e-6, 1e-7, 1e-8}
Transient Value Function (0.5x)	{1e-2, 1e-3, 1e-4, 1e-5}

Table C.11: Search space of learning rates for the main JBW experiment.

Algorithm	LRs
Q-learning	{1e-2, 1e-3, 1e-4, 1e-5}
Permanent Value Function	{1e-4, 1e-5, 1e-6, 1e-7}
Transient Value Function	{1e-2, 1e-3, 1e-4, 1e-5}

Table C.12: Search space of learning rates for the main JBW ablation experiment.

TV-LR → PV-LR ↓	λ=0.55				λ=0.75				λ=0.95			
	1e-2	1e-3	1e-4	1e-5	1e-2	1e-3	1e-4	1e-5	1e-2	1e-3	1e-4	1e-5
1e-4	0.031	**0.223**	0.126	0.103	0.035	**0.257**	0.154	0.154	0.038	**0.217**	**0.199**	0.159
1e-5	0.031	**0.252**	0.071	0.051	0.030	**0.261**	0.077	0.068	0.040	**0.220**	**0.201**	0.096
1e-6	0.041	**0.242**	0.070	0.061	0.034	**0.260**	0.090	0.083	0.030	**0.234**	**0.207**	0.084
1e-7	0.036	**0.256**	0.056	0.067	0.040	**0.268**	0.062	0.070	0.028	**0.227**	**0.216**	0.124
1e-8	0.031	**0.247**	0.051	0.069	0.041	**0.261**	0.069	0.069	0.037	**0.232**	**0.213**	0.080

Table C.13: JBW: Reward per step for various hyperparameters for PT-DQN. We highlight the values that is larger than DQN's best performance.

LRs	1e-2	1e-3	1e-4	1e-5
Returns	0.03	0.127	**0.192**	0.093

Table C.14: JBW: Reward per step for various hyperparameters for DQN.

LRs	1e-2	1e-3	1e-4	1e-5
Returns	0.037	0.135	**0.247**	0.245

Table C.15: JBW: Reward per step for various hyperparameters for DQN-multi-head.

LRs	1e-2	1e-3	1e-4	1e-5
Returns	0.025	0.055	**0.055**	0.031

Table C.16: JBW: Reward per step for various hyperparameters for DQN with large experience replay.

TV-LR → PV-LR ↓	λ=0.55			λ=0.75			λ=0.95		
	1e-3	1e-4	1e-5	1e-3	1e-4	1e-5	1e-3	1e-4	1e-5
1e-7	**14.41**	**17.38**	7.89	**13.87**	**19.07**	10.55	9.50	**16.12**	12.16
1e-8	**15.29**	**17.35**	7.79	**14.02**	**20.54**	10.72	9.79	**16.71**	12.48
1e-9	**15.54**	**17.51**	8.02	12.63	**19.54**	10.90	8.57	**16.48**	11.76

Table C.17: MinAtar: AUC of the average return (in the previous 100 episodes) per step for various hyperparameters for PT-DQN. We highlight the values that is larger than DQN's best performance.

LRs	1e-3	1e-4	1e-5	1e-6
Returns	7.10	12.90	**13.14**	5.90

Table C.18: MinAtar: AUC of the average return (in the previous 100 episodes) per step for various hyperparameters for DQN.

LRs	1e-3	1e-4	1e-5	1e-6
Returns	8.72	14.93	**15.68**	8.54

Table C.19: MinAtar: AUC of the average return (in the previous 100 episodes) per step for various hyperparameters for DQN-multi-head.

LRs	1e-3	1e-4	1e-5	1e-6
Returns	6.78	**11.44**	10.70	4.56

Table C.20: MinAtar: AUC of the average return (in the previous 100 episodes) per step for various hyperparameters for DQN with large experience replay.