The Permanent and Transient Framework for Continual Reinforcement Learning

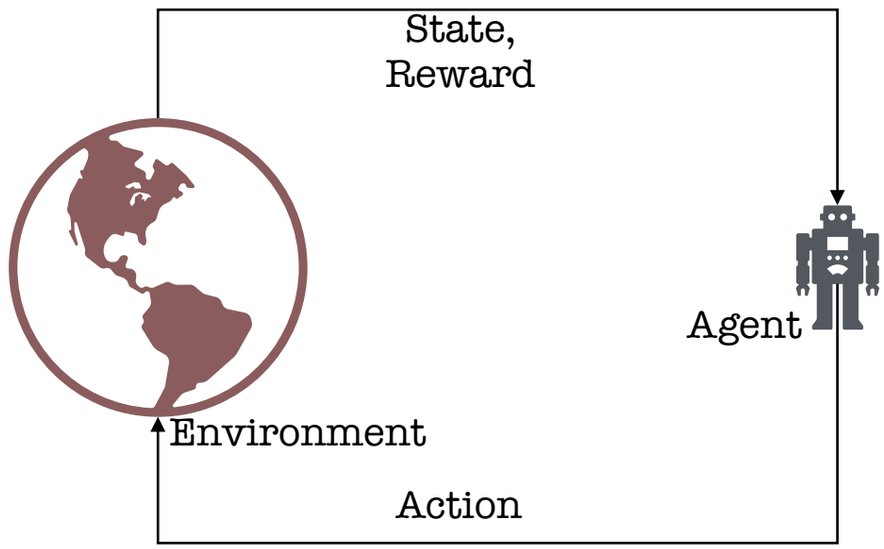
Nishanth Anand

Ph.D. Defence, McGill University and Mila

March 20th, 2026

Prediction and Control in Continual Reinforcement learning (https://openreview.net/pdf?id=KakzVASqul)
Permanent and Transient Representations for Continual Reinforcement Learning (https://openreview.net/pdf?id=5XfxEQ2SCt)
Preferential Temporal Difference Learning (https://arxiv.org/pdf/2106.06508)

Continual Reinforcement Learning



Traditional RL

- **Strong assumptions** about the environment (e.g. MDP).
- Learning stops after finding one **optimal policy**.

Continual RL

- Ideally **no assumptions** about the environment **besides regularities.**
- **Learning never stops.**
- Emphasis on **adaptation**.

Motivation

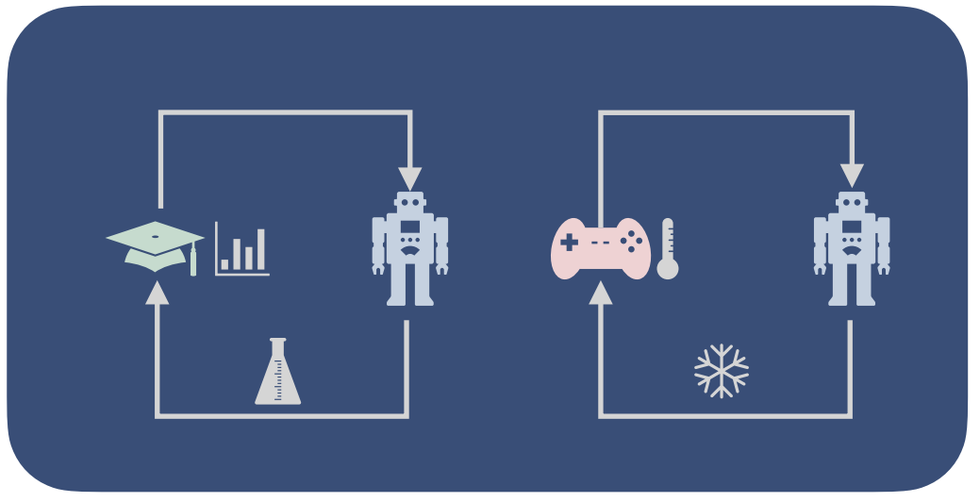
Humans learn continually

Motivation

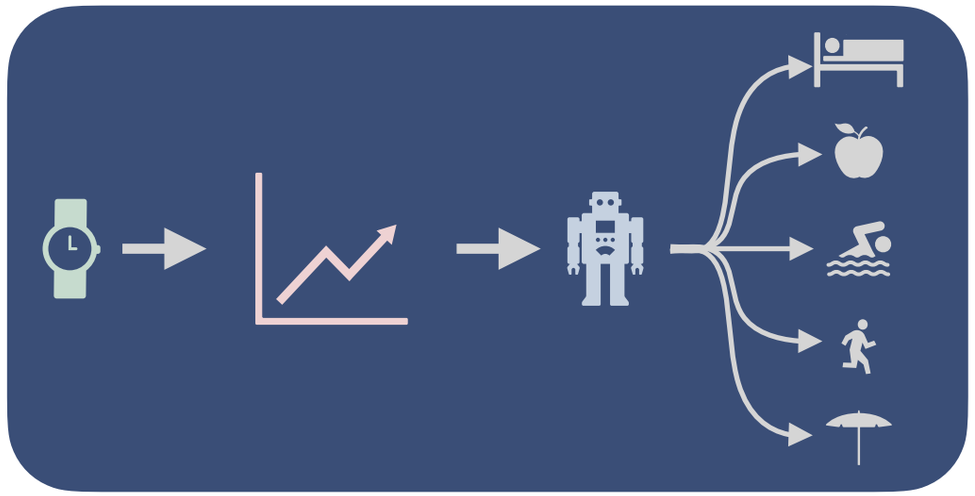
Small Agent, Big World Hypothesis

- Agent’s **capacity is limited** realtive to the **complexity of the world**
- It **can’t model everything**. It **must adapt**

Motivation



Adaptive tutors and game-playing agents



Personalized activity recommendation

Practical Applications

Research Overview

Prediction and Control in Continual RL
(w/ Doina Precup, NeurIPS 2023)

Permanent and Transient Representations
(w/ Doina Precup, under review)

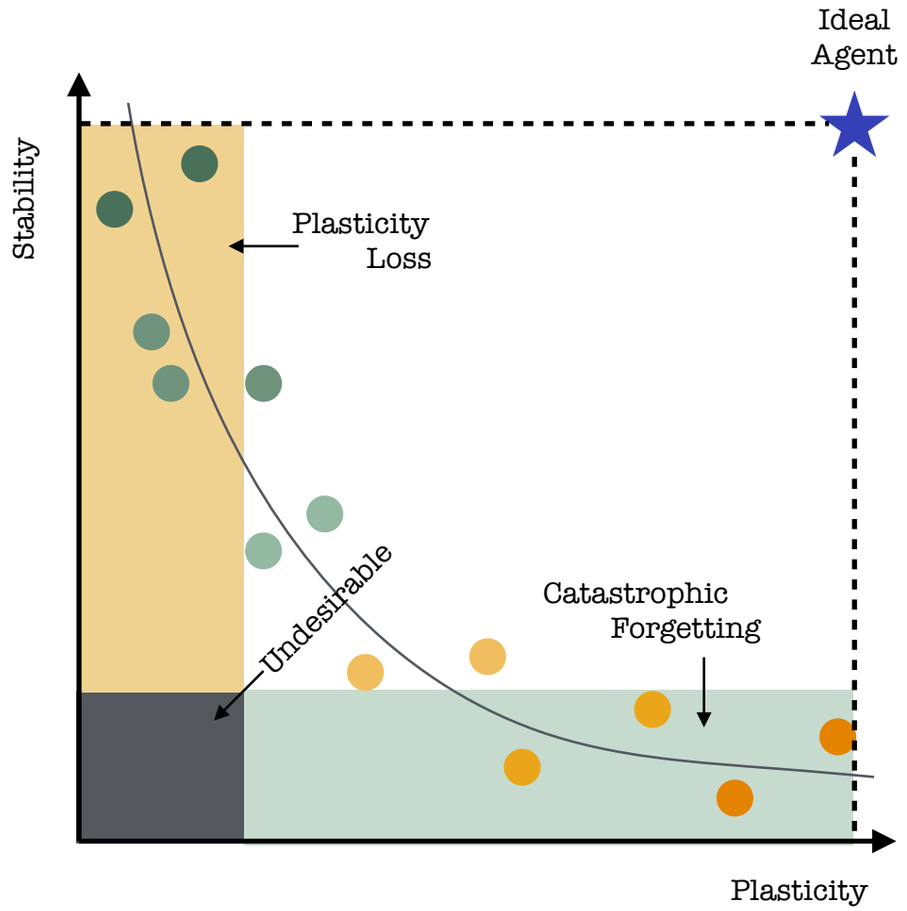
Preferential Temporal Difference Learning
(w/ Doina Precup, ICML 2021)

Permanent and Transient Successor Features for CRL
(w/ Doina Precup, under review)

Modelling Dopamine Ramping Effect using PT-Framework
(w/ Doina Precup and Paul Masset, in preparation)

Stability-Plasticity Dilemma

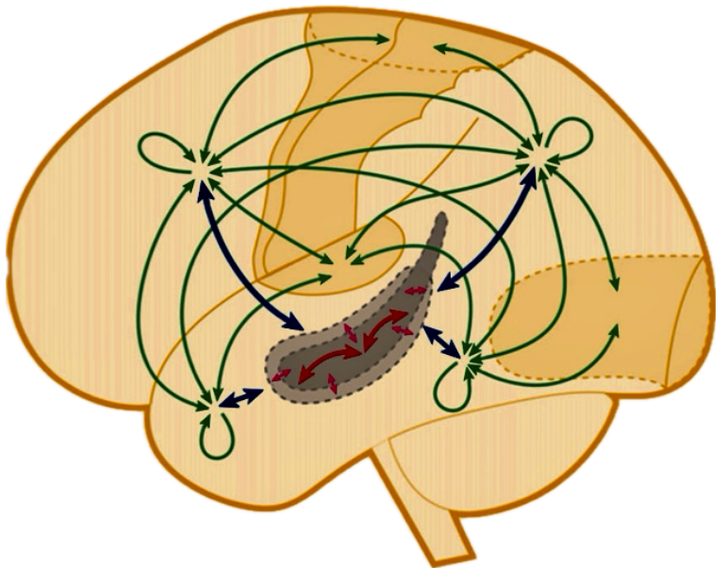
High stability
Can't learn from new experiences



Ideal Agent
Retains recurring patterns, while learning new ones

High plasticity
Overwrite acquired knowledge

Complementary Learning Systems Theory



cf. Kumaran et. al.

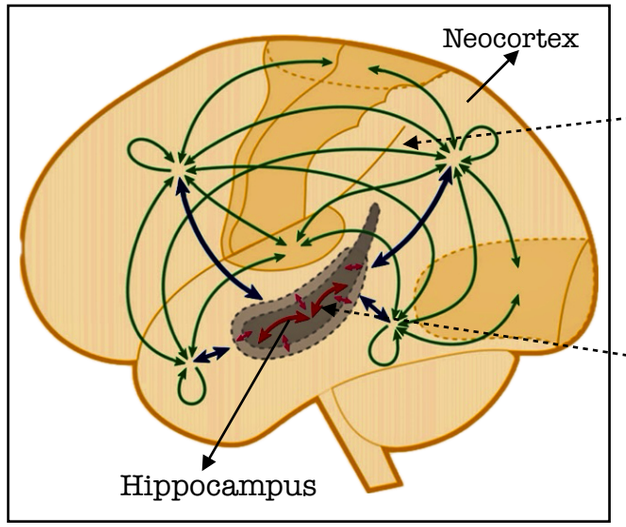
Neocortex

- Slow learning
- Structured
- Knowledge persists
- Aids generalization

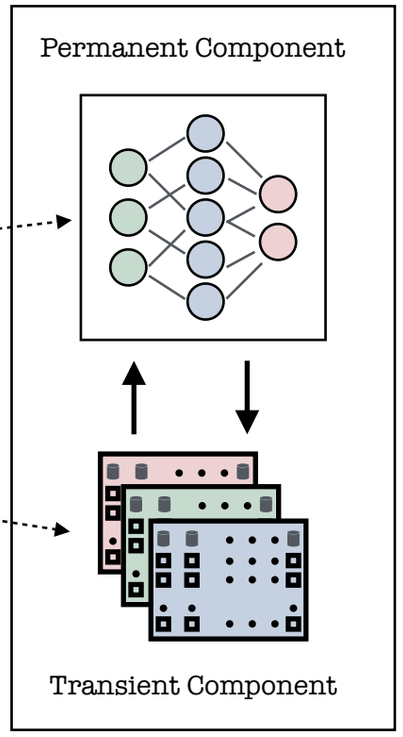
Hippocampus

- Fast learning
- Precise
- Knowledge is transient
- Aids in learning new knowledge

The PT Framework for Value Functions



Complementary Learning Systems in the Brain (c.f. Kumaran, et. al.)



Decompose learning into two systems—permanent and transient—that are complementary to one another

$$V^{(PT)}(s) = V_\theta^{(P)}(s) + V_w^{(T)}(s)$$

Overall value function

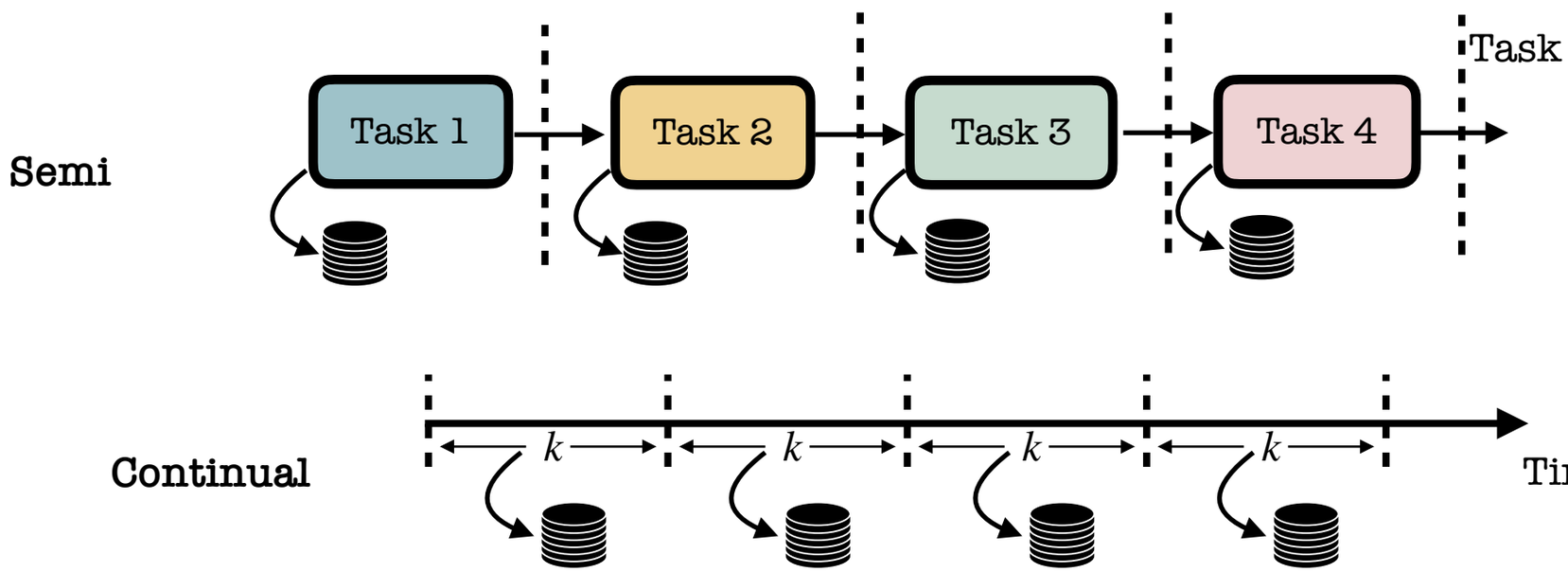
Permanent value function

Transient value function

$$Q^{(PT)}(s,a) = Q_\theta^{(P)}(s,a) + Q_w^{(T)}(s,a)$$

Permanent and Transient Value Functions

Permanent value function:



$$\theta_{k+1} \leftarrow \theta_k + \bar{\alpha}_k \left(V^{(PT)}(S_k) - V_\theta^{(P)}(S_k) \right) \nabla_\theta V_\theta^{(P)}(S_k)$$

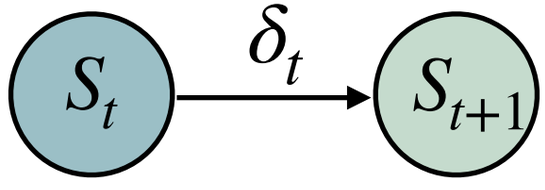
$$\theta_{k+1} \leftarrow \theta_k + \bar{\alpha}_k \left(Q^{(PT)}(S_k, A_k) - Q_\theta^{(P)}(S_k, A_k) \right) \nabla_\theta Q_\theta^{(P)}(S_k, A_k)$$

Bootstrapping at a slower timescale to capture commonality in predictions

Updated at the end of the task or every k steps

Permanent and Transient Value Functions

Transient value function:



$$w_{t+1} \leftarrow w_t + \alpha_t \left(R_{t+1} + \gamma V^{(PT)}(S_{t+1}) - V^{(PT)}(S_t)\right) \boxed{\nabla_w V_w^{(T)}(S_t)}$$

Gradient of transient value at t

TD error at t

Semi-gradient update rule

$$w_{t+1} \leftarrow w_t + \alpha_t \left(R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q^{(PT)}(S_{t+1}, a') - Q^{(PT)}(S_t, A_t)\right) \nabla_w Q_w^{(T)}(S_t, A_t)$$

Online updates to compute residuals required for rapid adaptation

Decayed or reset periodically to induce plasticity

Theoretical Results (Intuition)

- The sequence of expected updates of $V_\theta^{(P)}$ contracts to a unique fixed point which optimizes the **jumpstart objective—zero shot performance.**

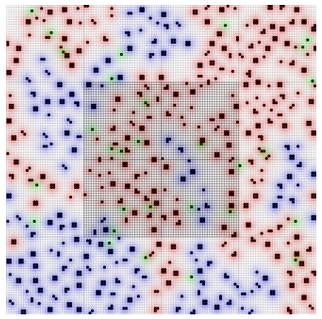
- TD estimates **forget the past** after seeing enough samples from the new task, but it **is retained through** $V_\theta^{(P)}$ in our approach.

- The **expected error on the new task is lower** for our approach compared with that of TD learning.

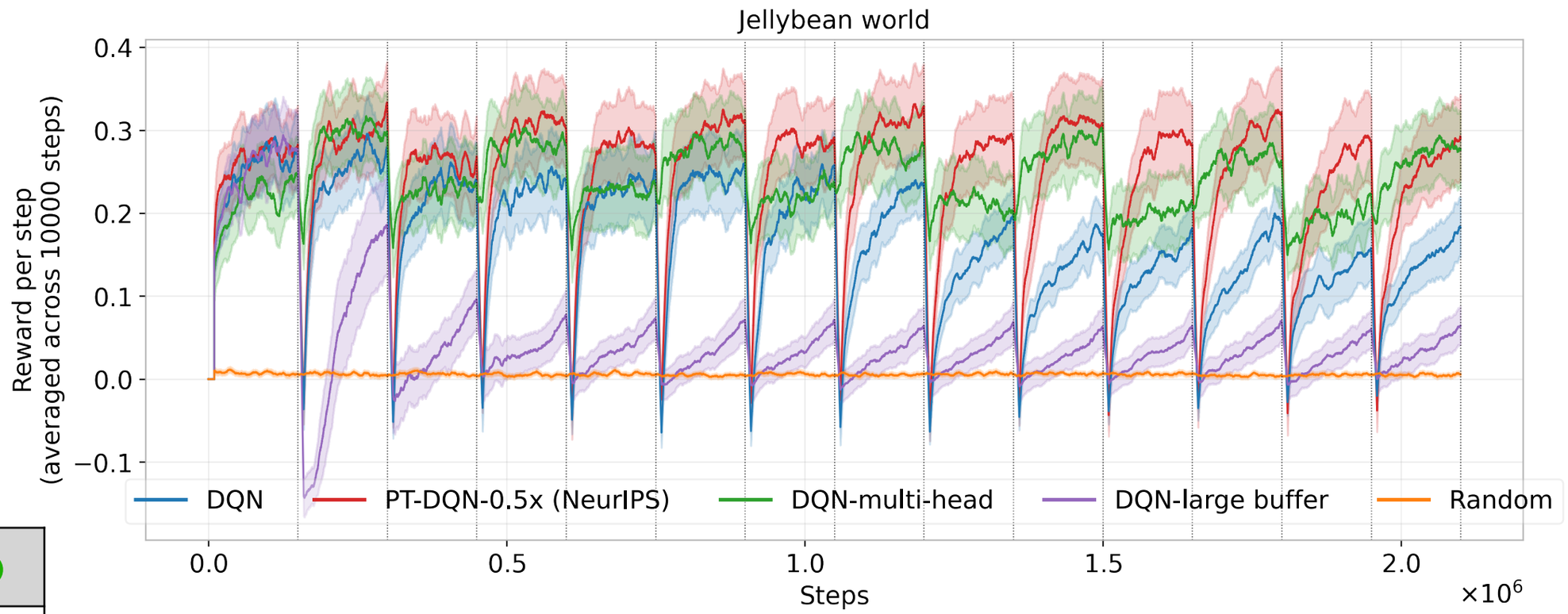
- Our approach **generalizes TD(0) learning.**

Results (Full CRL)

Control



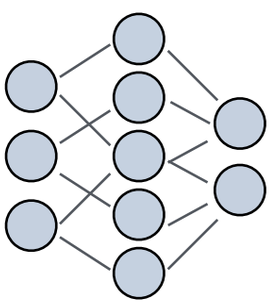
Task	●	●	●
1	2	-1	0.1
2	-1	2	0.1



The bias from permanent values is favourable as it is combines Q-values from all tasks. So, only few samples are needed to make corrections (P is updated every **10k** steps)!

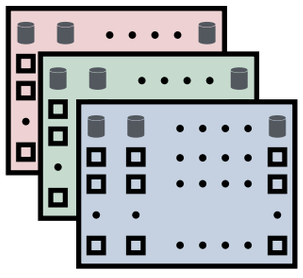
Permanent and Transient Representations

Permanent Representations



- **Expressive** enough to learn good baseline predictions
- Learning can be slow, but the **information acquired should persist**
- Should **support broad generalization** of predictions between similar situations
- E.g., Neural network, CNNs, RNNs, etc.

Transient Representations

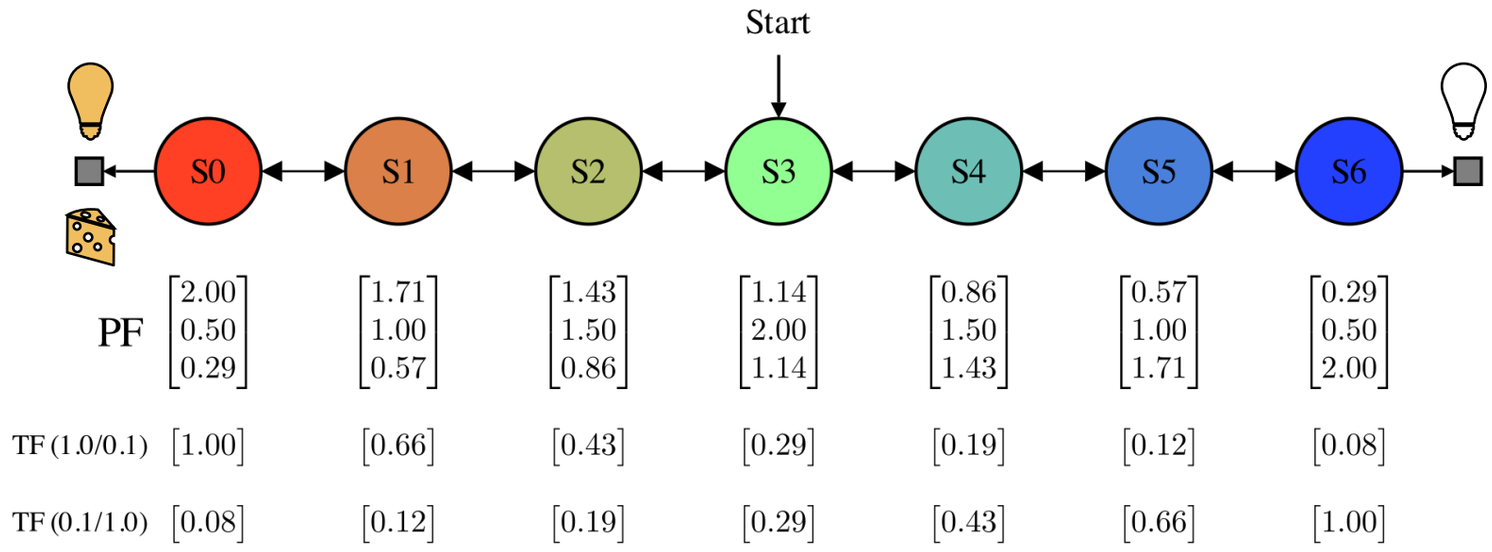


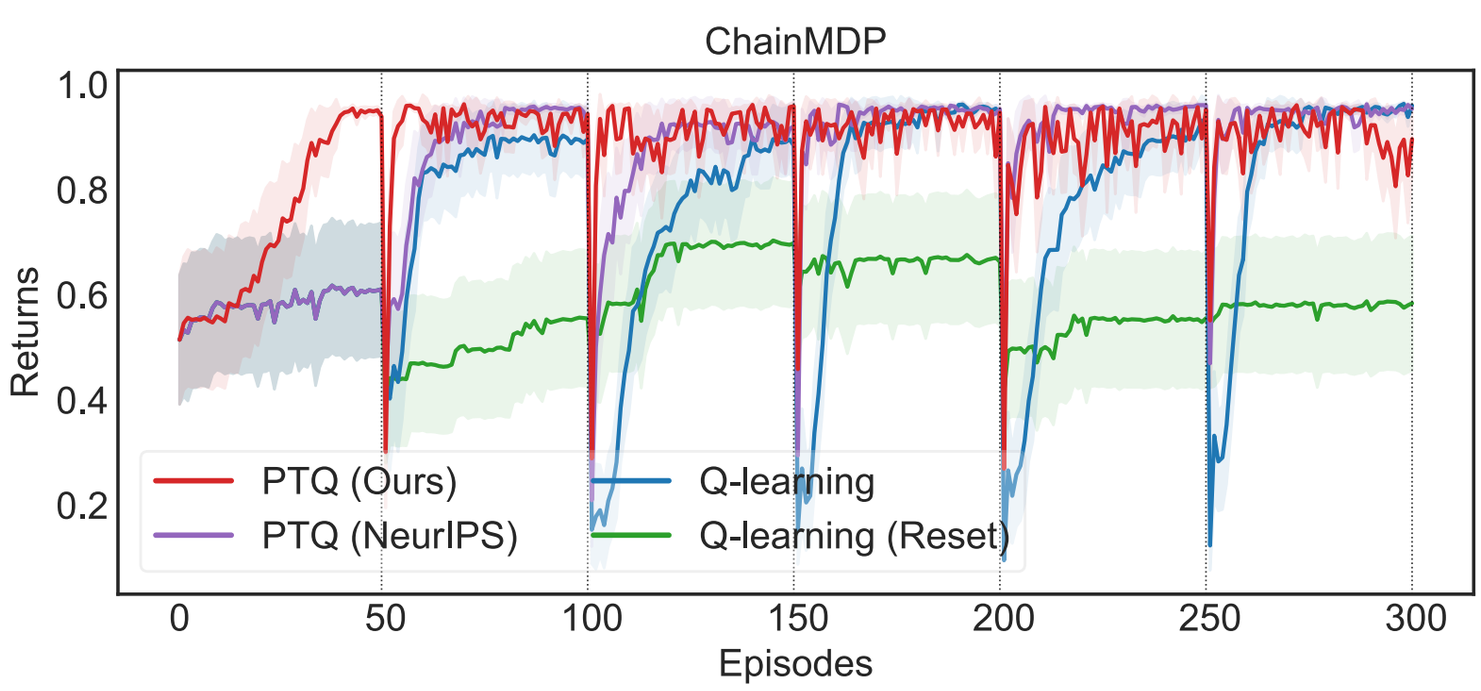
- Support **online learning at a rapid pace**
- Support **temporary adaptation**
- Facilitate **learning precise estimates**, with **minimal or carefully controlled generalization** around the current data
- E.g., New non-parametric function approximator

Results

Control

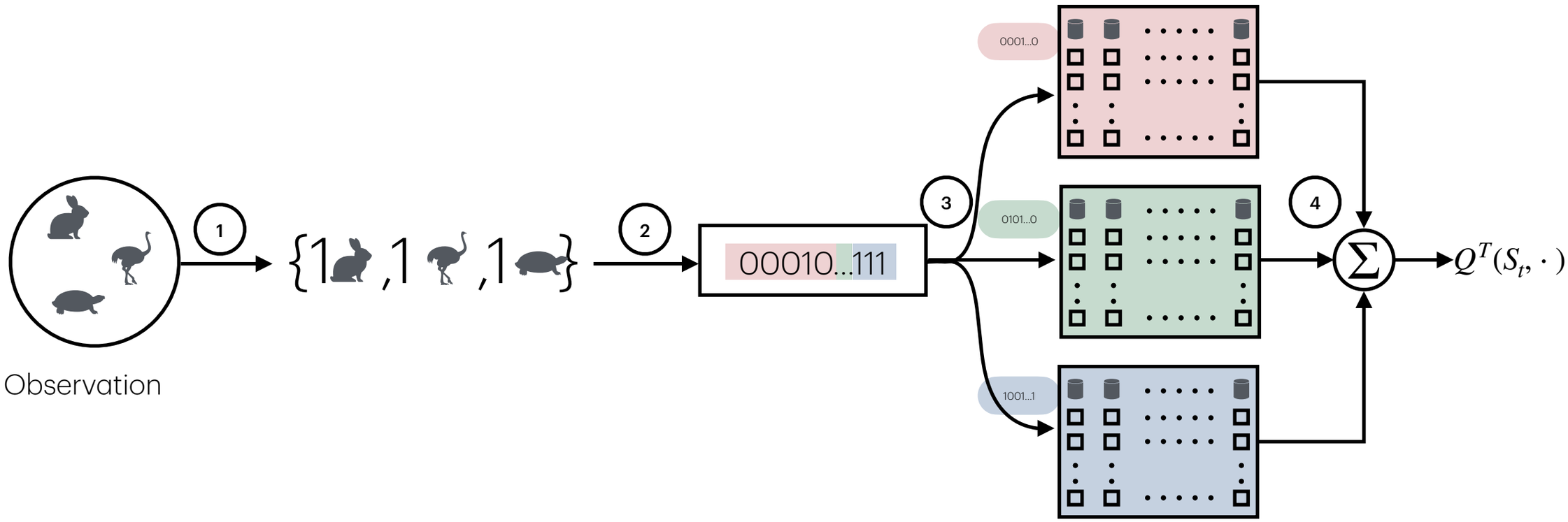
Task	🔴	🔵
1	1	0.1
2	0.1	1





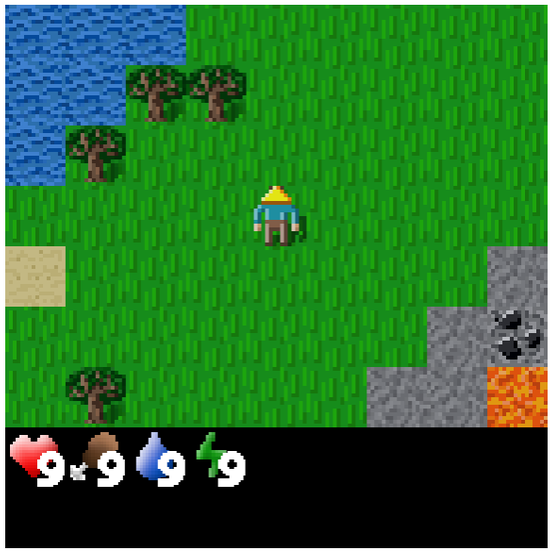
Using separate features results in faster adaptation!

Non-Parametric Approximator

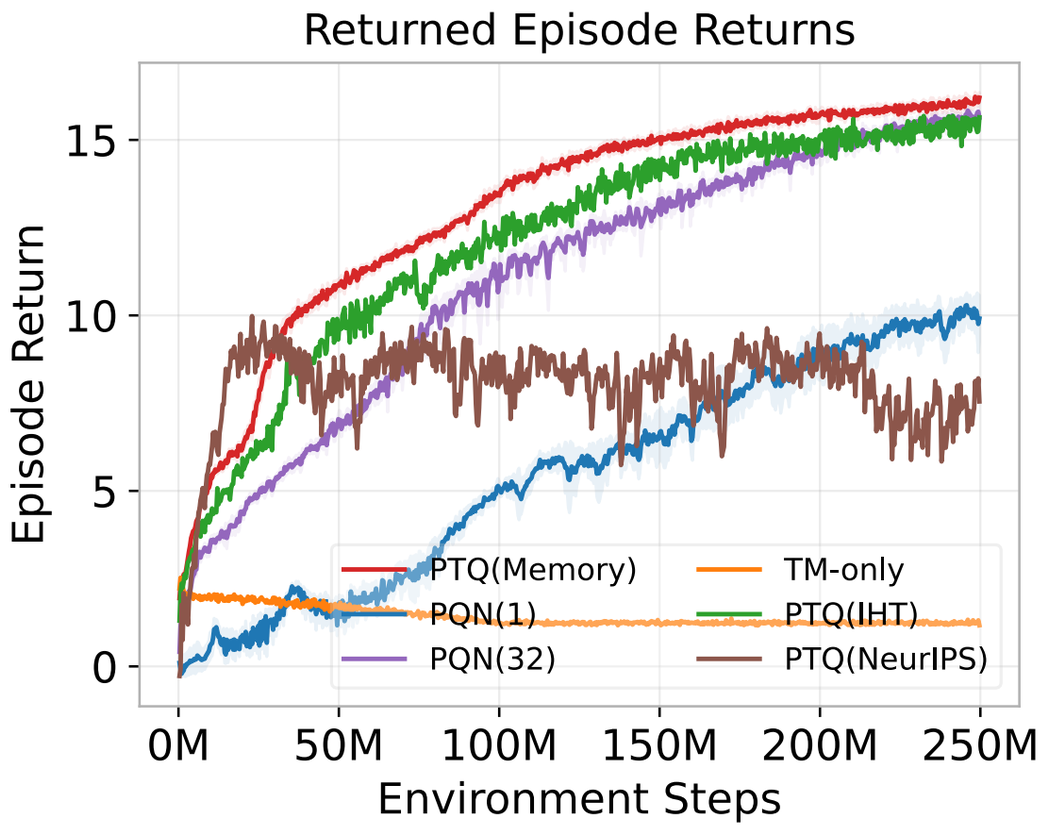


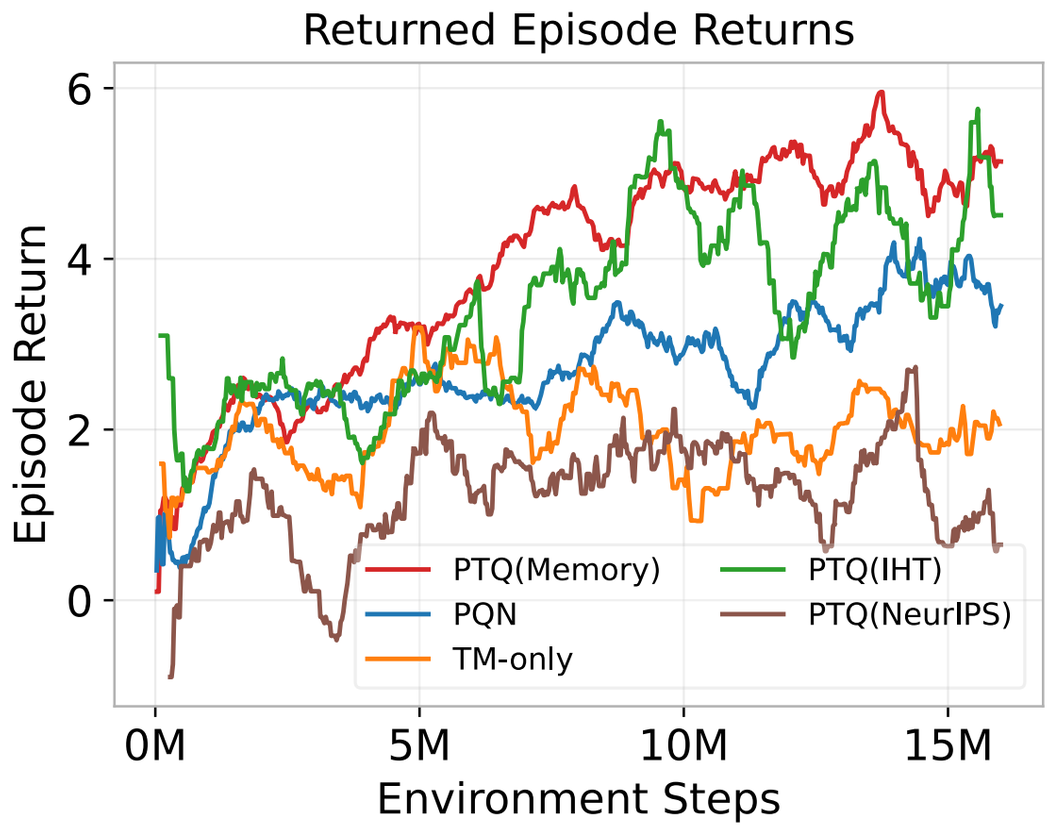
1. **Tokenization**: Mapping observation to symbolic set
2. **Hashing**: MinHash to get the hash signature and tag
3. **Binning**: Split the signature into subsequences to get bin index
4. **Value estimation**: Pool the values stored in each table to get estimates

Results - Craftax

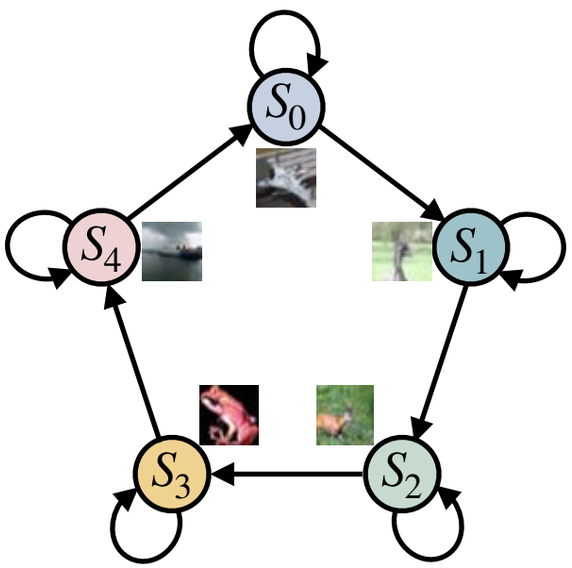


- Transient values are updated online, and the permanent values are updated every **32 steps.**
- PTQ performs better than competitive baselines both in online continual learning and in large sample complexity regimes.

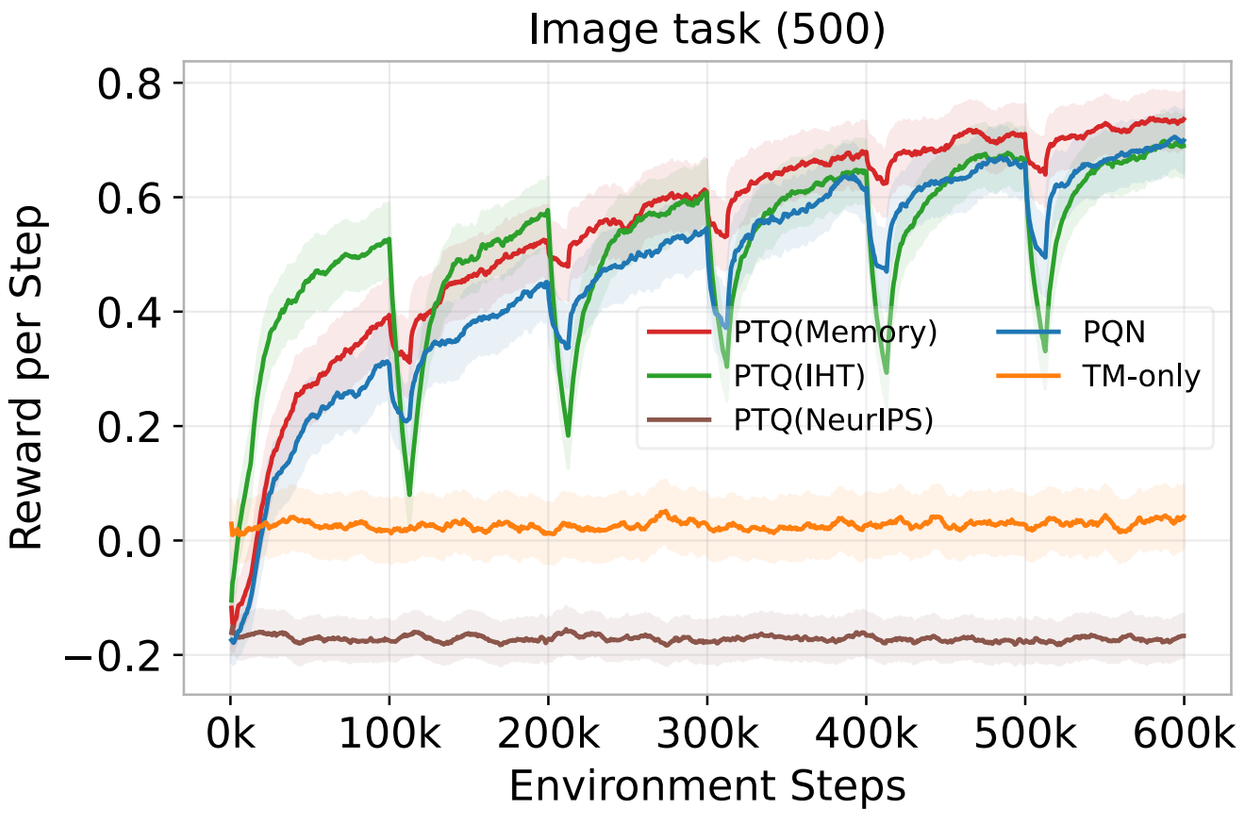


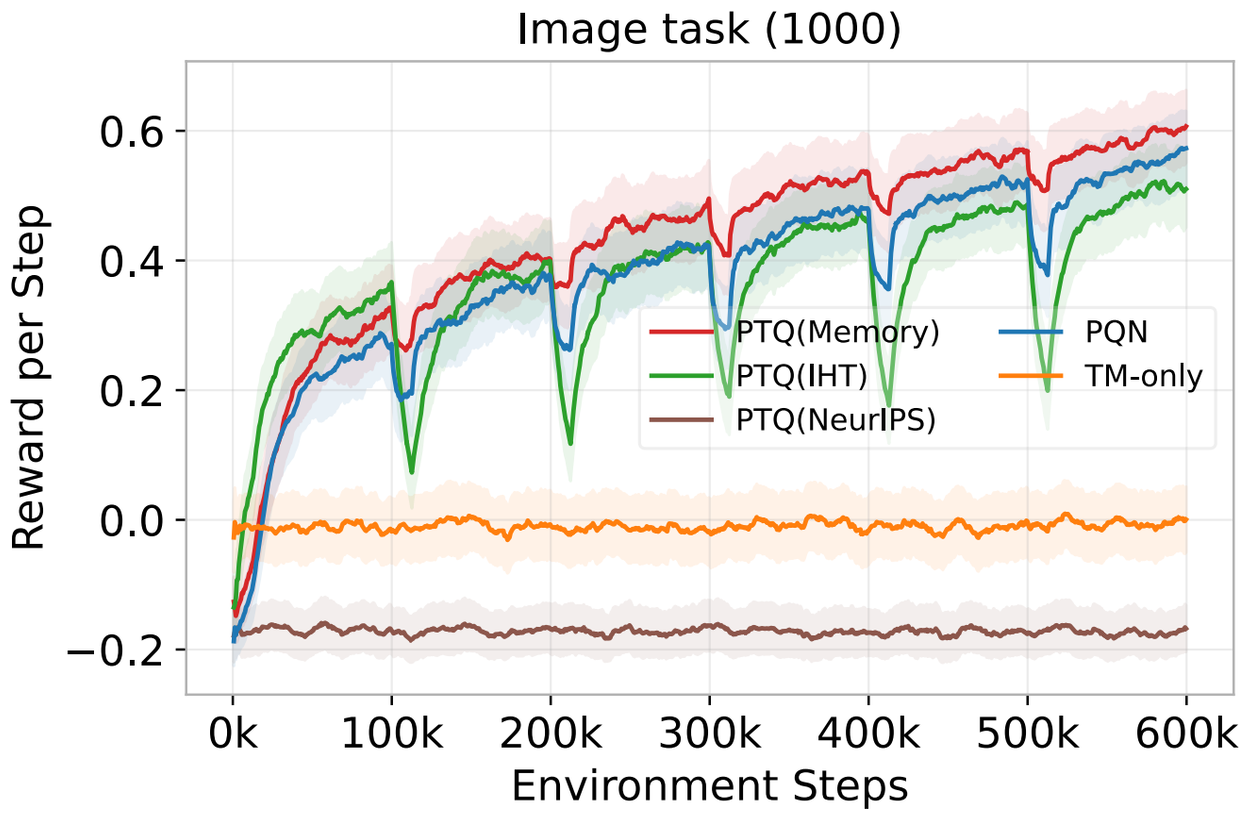


Results - Image Task



PTQ **results in instant adaptation** due to **quick, online learning** facilitated by non-parametric transient approximator! **(P updates every 32 steps)**



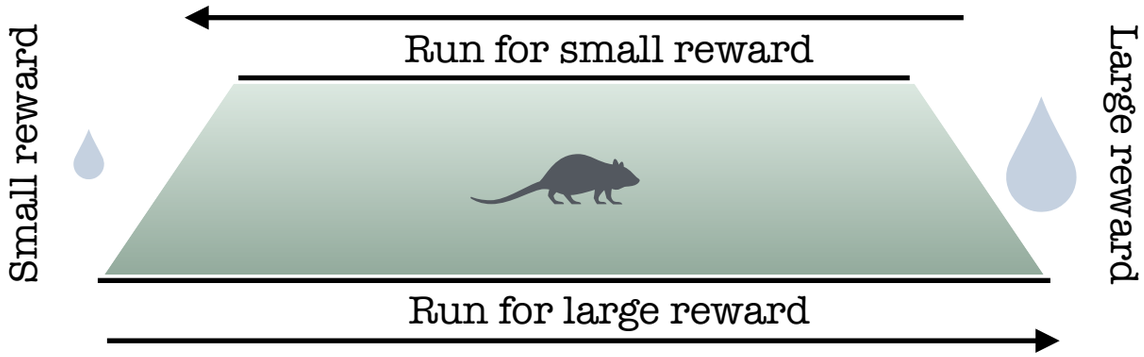


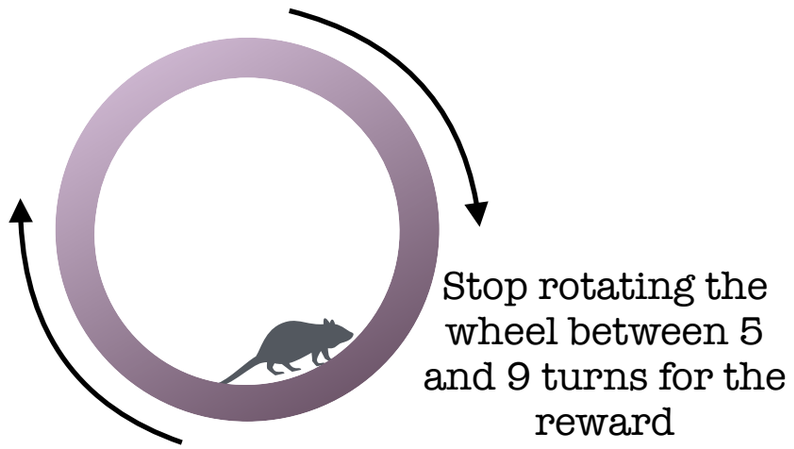
Modeling Dopamine Ramping Activity*

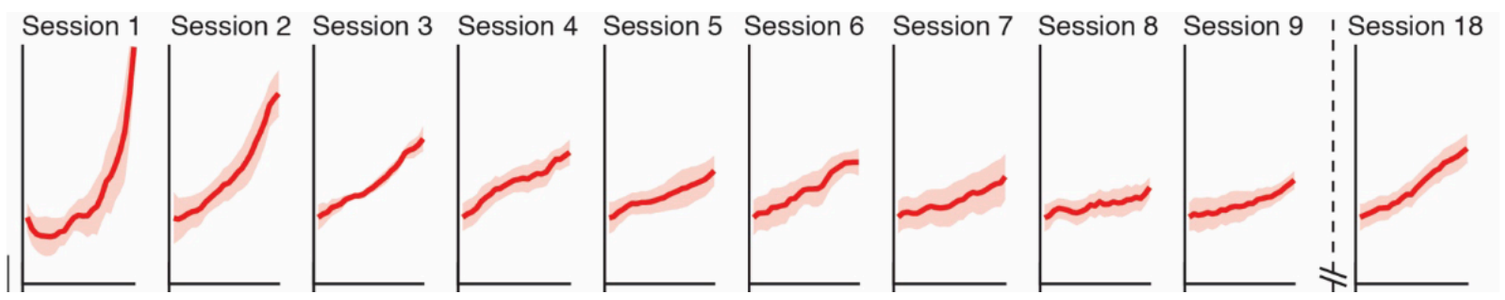
- **Predicting the future** is a key trait in intelligent species for survival.
- **Dopamine** plays an important role in learning to predict the future.
- **Dopamine** also plays a crucial role in **motivation, reward processing, and decision-making.**
- **Too little dopamine:** Parkinson's disease and motivational deficits. **Too much dopamine:** Schizophrenia
- Dopamine doesn't distinguish healthy from toxic rewards—addiction to substance and toxic relationships due to **intermittent reinforcement**!
- Better understanding of dopamine signaling is an active area of research in neuroscience.

* In preparation for a submission at a Neuroscience Journal, supervised by Doina Precup and Paul Masset

Dopamine Ramping Effect



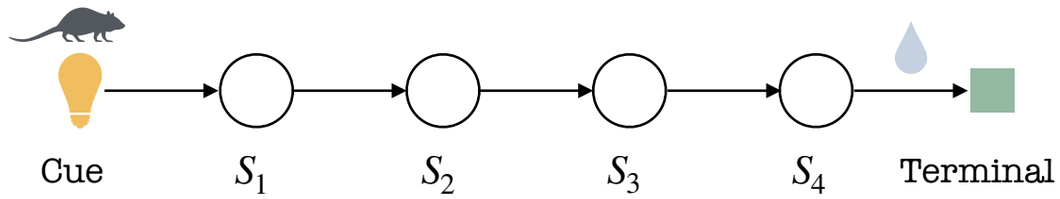




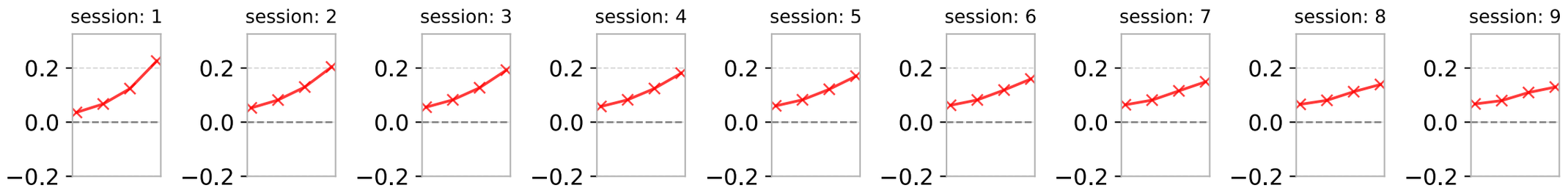
Guru et. al., 2020 hypthesized that ramping dopamine reflects the active use of a cognitive map to estimate goal proximity

Kato et. al., 2025 provided a normative framework to explain this phenomenon by incorporating decay and hierarchy to the standard TD model, but it based on heuristics and is unsound for RL

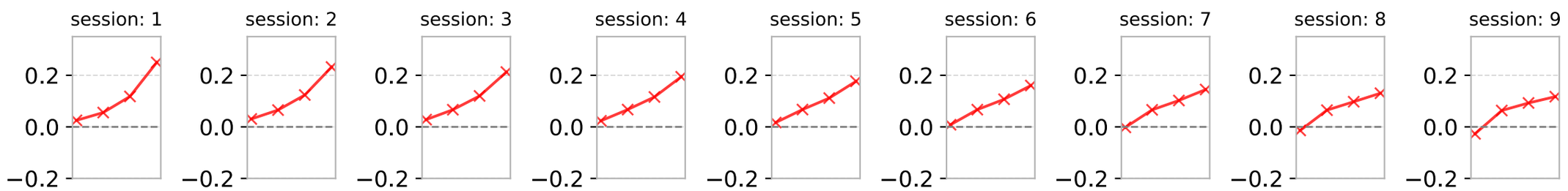
Results



Fixed P



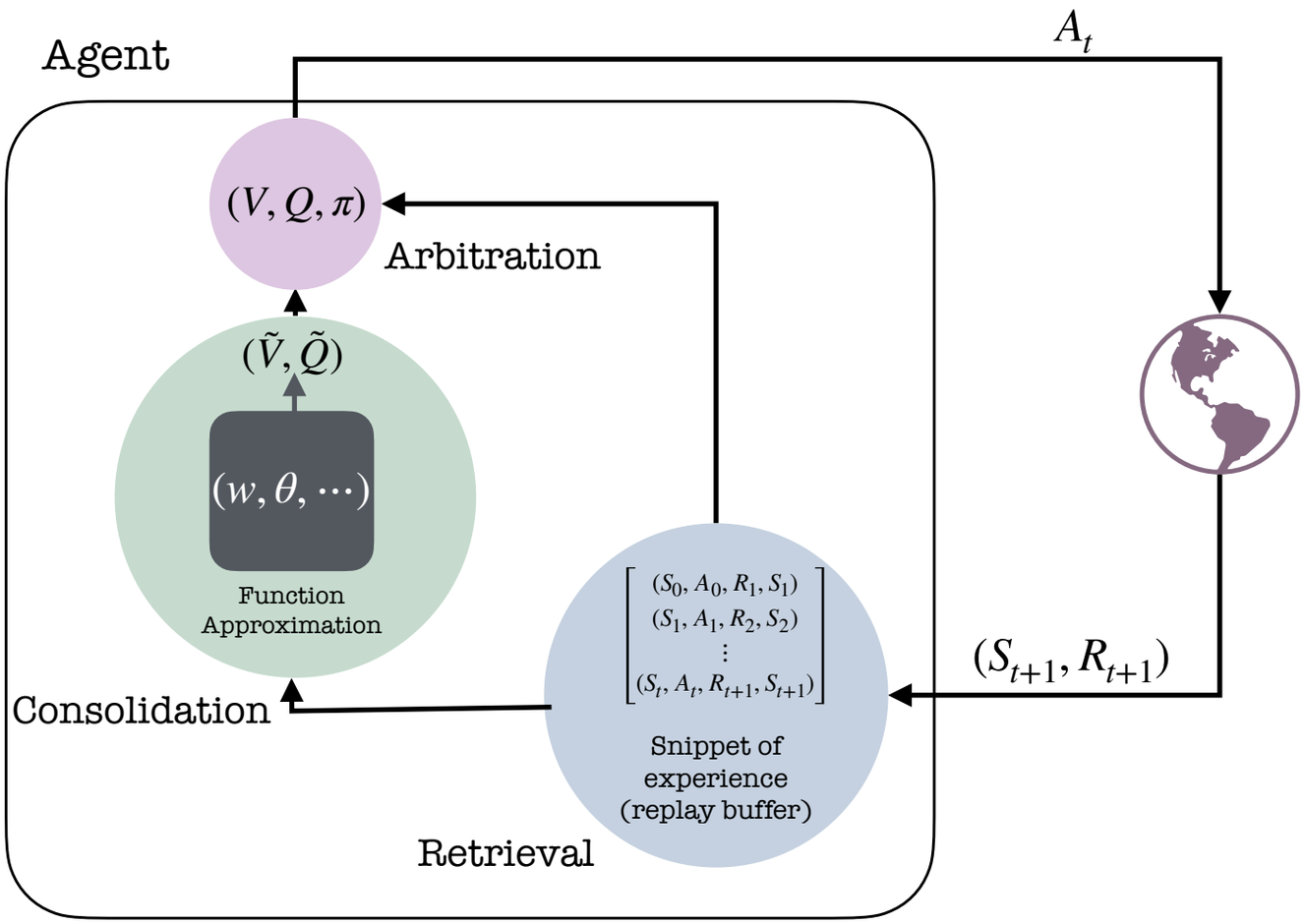
Learning P



PT framework successfully reproduced results from Guru et. al., 2020 in the same setting as Kato et. al., 2025

Discussion and Conclusion

RCA Framework to unify existing algorithms



	Retrieval	Consolitation	Arbitration
DQN	Experience replay	Neural Net	Neural Net output
Episodic Control	Kernel methods	Neural Net	Kernel output
PT-Framework	Non-parametric	Neural Net	Summation

Discussion and Conclusion

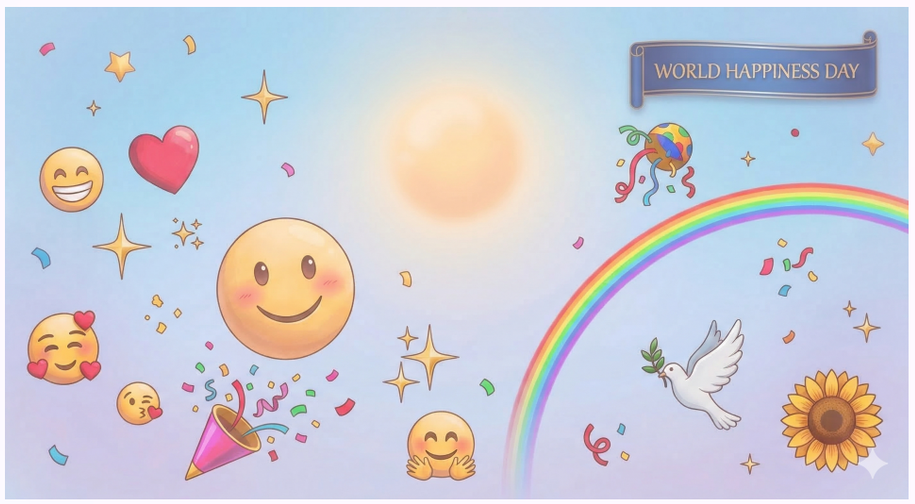
- PT Framework is **grounded in theory, has empirical benefits, and is well-suited for continual RL** for estimating value functions and SFs.

- Success is attributed to the use of **two learning systems** that results in **a better stability-plasticity trade-off**. These systems **complement** in plasticity degrees, learning speeds, update rules, and representations.

- PT Framework also **serves as a normative model for dopaming ramping phenomenon**.

- Many possible research directions. **Extension to policy gradients, advanced arbitration, adaptive consolidation, etc.**

Thank you



Acknowledgements

Doina Precup
Paul Masset
Ahmed Touati
Jalaj Bhandari
Aditya Mahajan
Emmanuel Bengio
Khimya Khetarpal
Riashat Islam
Sumana Basu
Pierre-Luc Bacon
Harsh Satija
David Abel
Chen Sun
Sarath Chandar
Prakash Panangaden
Arushi Jain
Lynn Cheriff
Veronica Chelu
Mandana Samiei
Harry Zhao
Jonathan Lebensold

Wesley Chung
Safa Alver
Maxime Webartha
Martin Klissarov
Pierluca D'Oro
Manoosh Samiei
Haque Ishfaq
Raymond Chua
Padideh Nouri
Jesse Farebrother
Harley Wiltzer
Evgenii Nikishin
Charles Onu
Priyesh Vijayan
Gandharv Patil
Ray Luo
Zihan Wang
Zijing Wu
Mohammad Sami Nur Islam
Shahrad Mohammadzadeh
Jacob Chmura

Nishchitha Anand
Nikhil Murali
Ivan Anokin
Rishika Bhagwatkar
Kanishk Jain
Vinay Gowda
Saby Sahoo
Vinny Sahoo
Alice Yannau
Dheeraj Vattikonda
Cynthia Garcia
Artem Ploujnikov
Shubham Gupta
Ravi Chunduru
Surya Penmetsa
Shashank Murugesh

Mohamed Elsayed
Will Dabney
John D. Martin
Matthew Riemer
Abhishek Naik
Michel Ma
Amin Memarian
Adriana Knatchbull-Hugessen
Florence Cloutier
Glen Berseth
Olexa Bilaniuk
Fabrice Normandin
Zina Kamel
Rafid Saif
Marc Theveneau

Appendix

Value Functions

- State-value function, $v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right]$
- Estimated using TD-learning, $w_{t+1} \leftarrow w_t + \alpha_t \left(R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t) \right) \nabla_w V(S_t)$
- TD error, $R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)$
- Policy evaluation or actor-critic algorithms
- Action-value function, $q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right]$
- Q-learning, $w_{t+1} \leftarrow w_t + \alpha_t \left(R_{t+1} + \gamma \max_{a'} Q_w(S_{t+1}, a') - Q_w(S_t, A_t) \right) \nabla_w Q_w(S_t, A_t)$

Learning Algorithms

Algorithm 6 PT-TD learning (Prediction)

Initialize: buffer $\mathcal{B}$, parameters θ, $\mathbf{w}$
for $t = 0 \to \infty$ **do**
    Store state S_t in $\mathcal{B}$
    Observe reward R_{t+1} and next state S_{t+1}
    # Update transient parameters
    Update $\mathbf{w}$ using Eq. (3.3)
    if Task ends **then**
    # Update permanent parameters
        Update θ using $\mathcal{B}$ and Eq. (3.2)
    # Reset transient parameters
        Reset $\mathbf{w}$
        Reset $\mathcal{B}$
    end if
end for

Algorithm 8 PT-Q-learning (CRL)

Initialize: buffer $\mathcal{B}$, parameters θ, $\mathbf{w}$, k, λ
for $t = 0 \to \infty$ **do**
    Take action A_t
    Store state S_t and action A_t in $\mathcal{B}$
    Observe reward R_{t+1} and next state S_{t+1}
    # Update transient parameters
    Update $\mathbf{w}_t$ using Eq. (3.6)
    if $t \mod k = 0$ **then**
    # Update permanent parameters
        Update θ using $\mathcal{B}$ and Eq. (3.5)
    # Decay transient parameters
        $\mathbf{w}_{t+1} \leftarrow \lambda \mathbf{w}_{t+1}$
    # Clear buffer
        Reset $\mathcal{B}$
    end if
end for

Learning Algorithms

Algorithm 16 Preferential TD: Linear FA

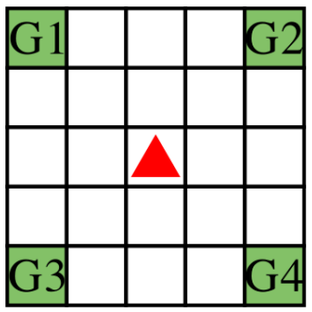
Input: policy π, discount γ, parameter β, features ϕ
Initialize: weights $\mathbf{w}_0 = 0$, eligibility trace $e_{-1} = 0$
Output: final weights $\mathbf{w}_T$
for $t = 0 \to T$ **do**
    Take action $a \sim \pi(s_t)$
    Observe reward r_{t+1} and next state s_{t+1}
    Compute values $\hat{v}(s_t) \leftarrow \mathbf{w}_t^T \phi(s_t)$ and $\hat{v}(s_{t+1}) \leftarrow \mathbf{w}_t^T \phi(s_{t+1})$
    # Compute TD error
    $\delta_t \leftarrow r_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t)$
    # Update eligibility trace
    $e_t \leftarrow \beta(s_t)\phi(s_t) + \gamma(1 - \beta(s_t))e_{t-1}$
    # Update weights
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \delta_t e_t$
end for

Algorithm 18 Permanent-Transient Successor Features (Prediction)

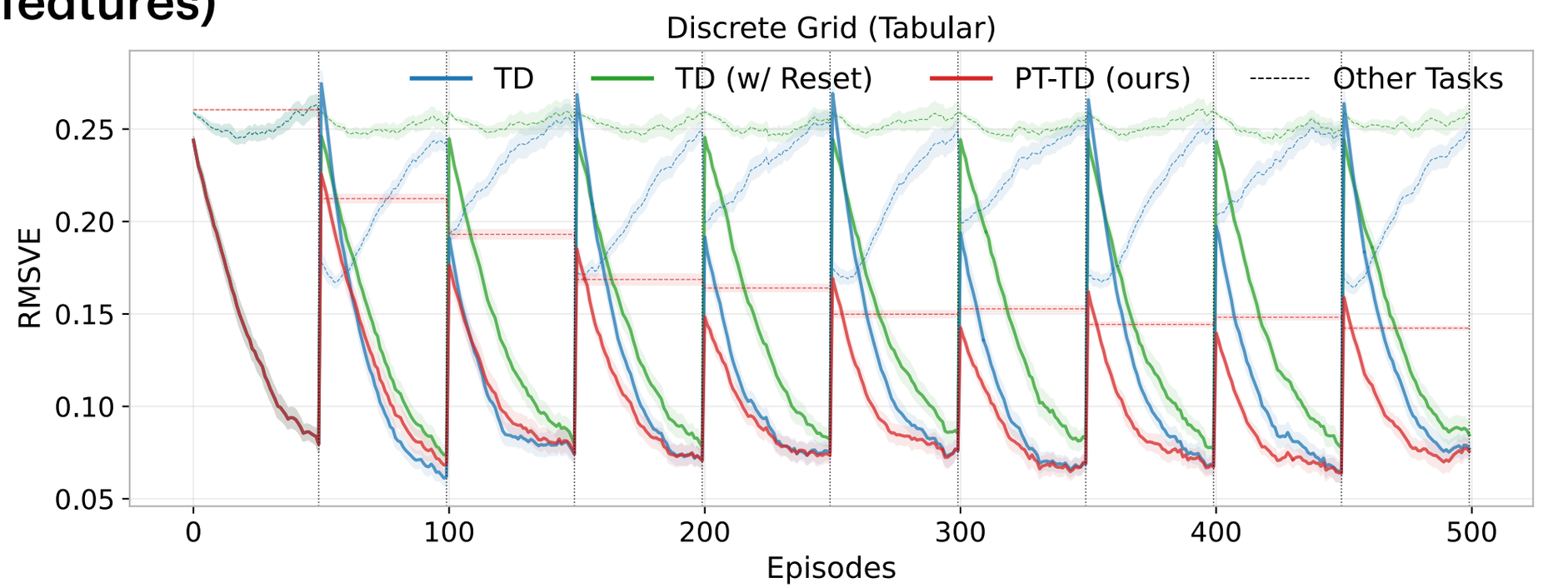
Input: feature function $\boldsymbol{\phi}(\cdot)$, policy π
Initialize: permanent parameters $\theta^{\boldsymbol{\phi}}$, transient parameters $\mathbf{w}^{\boldsymbol{\phi}}$, buffer $\mathcal{B}$
Hyperparameters: α, $\overline{\alpha}$, phase length k
for $t = 0 \to \infty$ **do**
    Observe state S_t and feature $\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$
    Take action $A_t \sim \pi(\cdot|S_t)$
    Observe next state S_{t+1} and feature $\boldsymbol{\phi}_{t+1}$
    # Update transient parameters
    Compute TD error $\boldsymbol{\delta}_t^{\phi}$ using total estimate $(\boldsymbol{\psi}^{(P)} + \boldsymbol{\psi}^{(T)})$
    Update $\mathbf{w}^{\boldsymbol{\phi}}$ using Eq. (6.9)
    Store transition $(S_t, \boldsymbol{\phi}_t)$ in $\mathcal{B}$
    if Task ends **or** $(t \mod k = 0)$ **then**
    # Update permanent parameters
        Update $\theta^{\boldsymbol{\phi}}$ using data in $\mathcal{B}$ and Eq. (6.11)
    # Reset transient parameters
        Reset (or decay) $\mathbf{w}^{\boldsymbol{\phi}}$
        Reset $\mathcal{B}$
    end if
end for

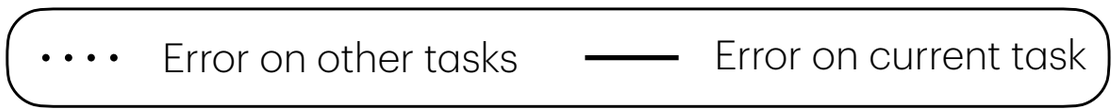
Results (Semi-CRL)

Prediction (same features)



Task	G1	G2	G3	G4
1	0	1	0	1
2	1	0	1	0
3	0	0	1	1
4	1	1	0	0

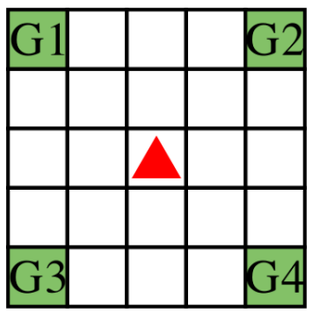




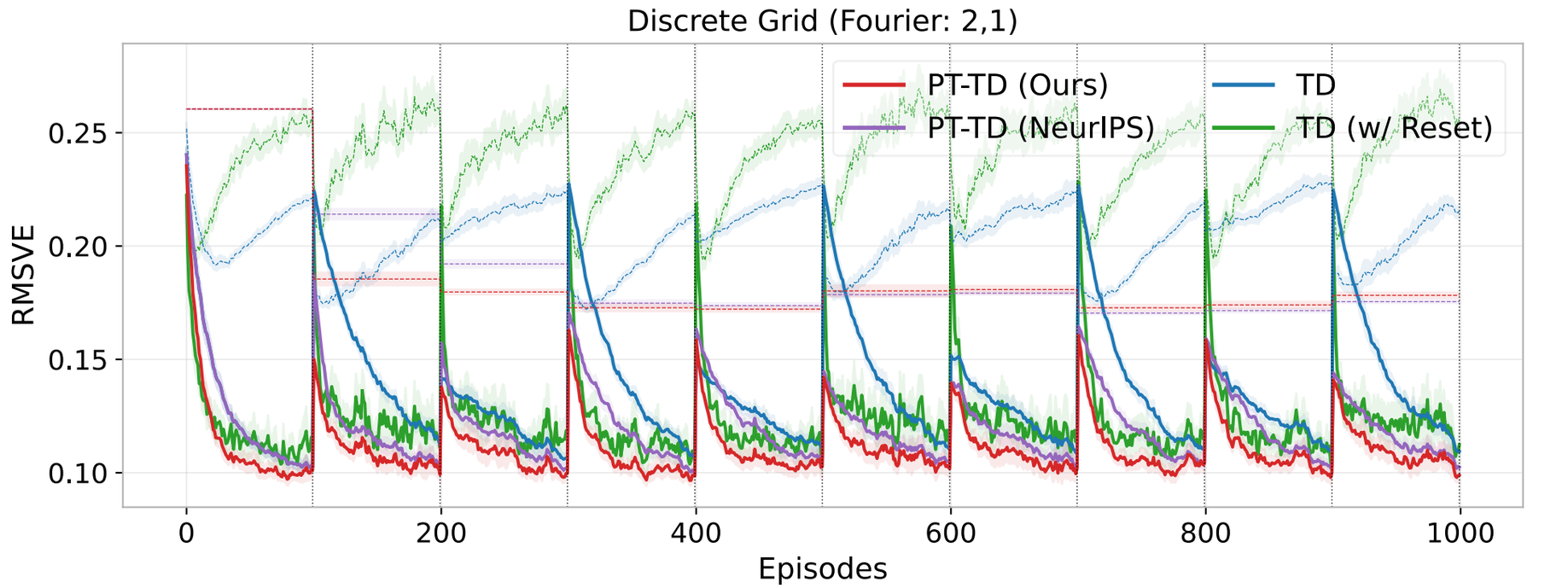
PT-TD retains the past information through $V^{(P)}$ and also adapts faster. Lower error on both current and other reward configurations!

Results

Prediction (different features, Semi-CRL)



Task	G1	G2	G3	G4
1	0	1	0	1
2	1	0	1	0
3	0	0	1	1
4	1	1	0	0

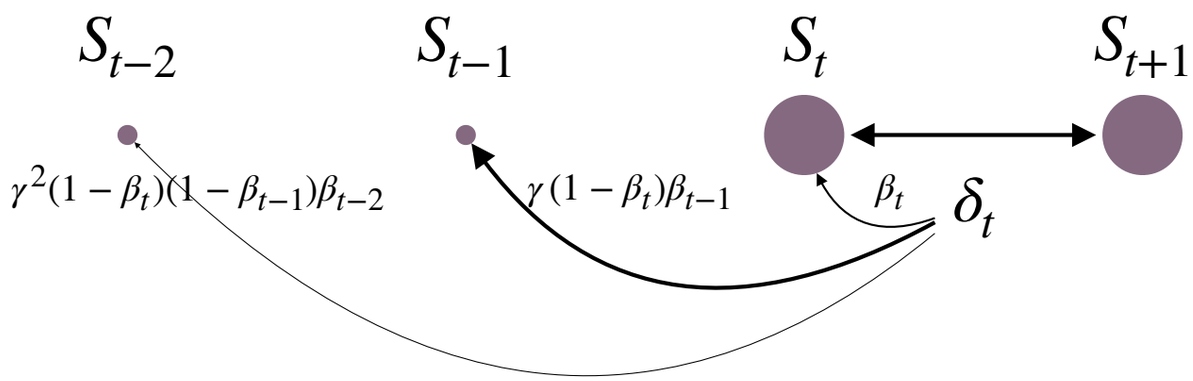


P features: Fourier features **order 1**
T features: Fourier features **order 2**

PT-TD's performance improves further when permanent and transient estimates use different features!

Preferential TD Learning

Use only a few states for bootstrapping and updating!



$$w_{t+1} \leftarrow w_t + \alpha_t \underbrace{\left(R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)\right)}_{\text{TD error at } t} e_t$$

$$e_t \leftarrow \underbrace{\gamma(1-\beta_t)e_{t-1} + \beta_t \nabla_w V_w(S_t)}_{\text{Eligibility trace}}$$

PTD - Theory

The sequence of expected updates computed by Preferential TD converges to a unique fixed point in the linear setting with state-dependent preference function under standard assumptions.

(Only algorithm besides Emphatic TD learning to converge with state-dependent bootstrapping)

Counterexample:

$$P_\pi = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \quad \Phi = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \qquad \gamma = 0.99 \qquad d_\pi = [0.5 \quad 0.5]$$

$$\lambda = [0.99 \quad 0.8] \qquad \beta = [0.01 \quad 0.2]$$

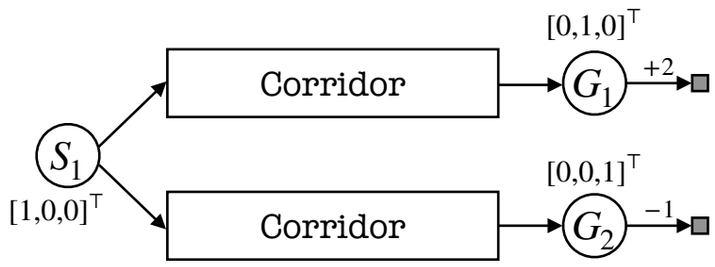
$$\mathbf{A}^{TD} = [-0.0429] \qquad \mathbf{A}^{PTD} = [0.009]$$

Not positive definite

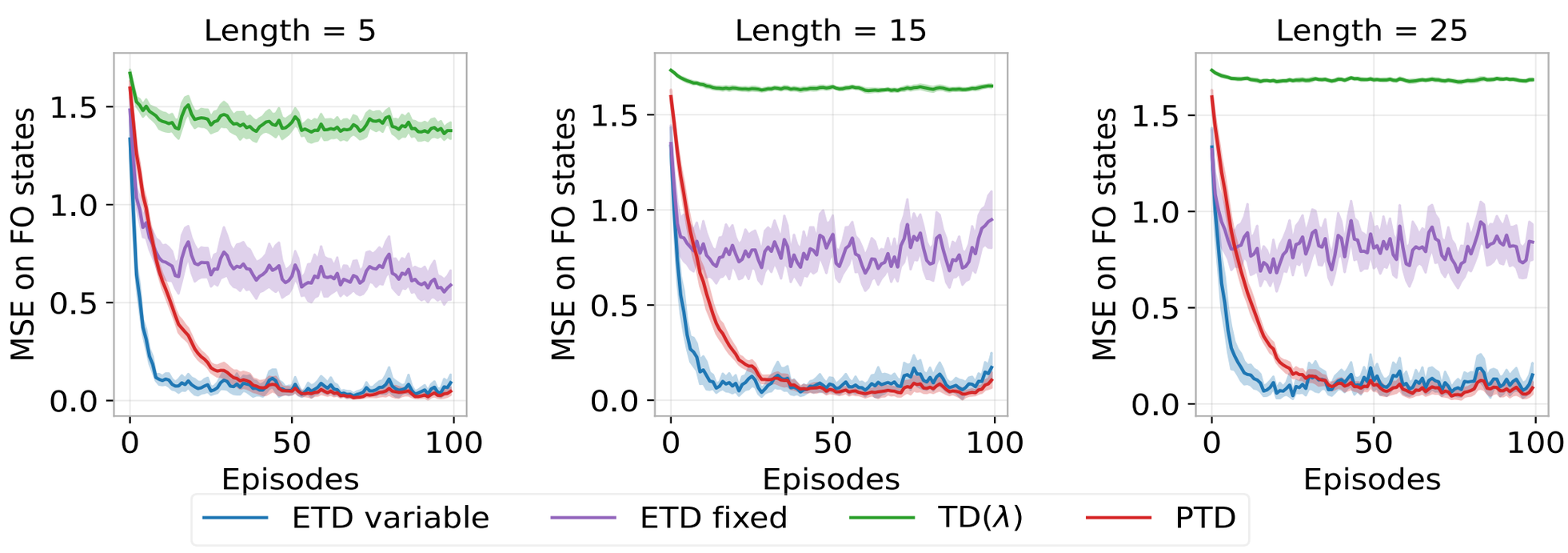
Positive definite

PTD - Results

Prediction (linear FA)

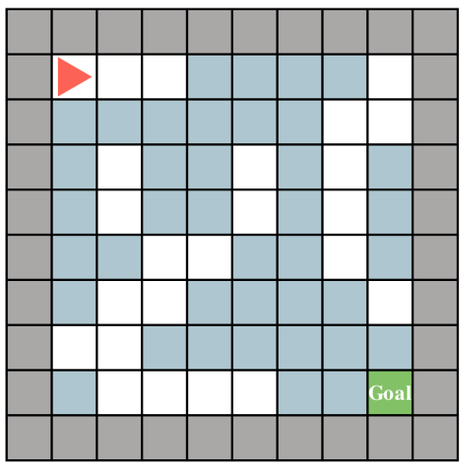


PTD learning algorithm learns the true value of states that are fully observable ($\beta = 1$ states)

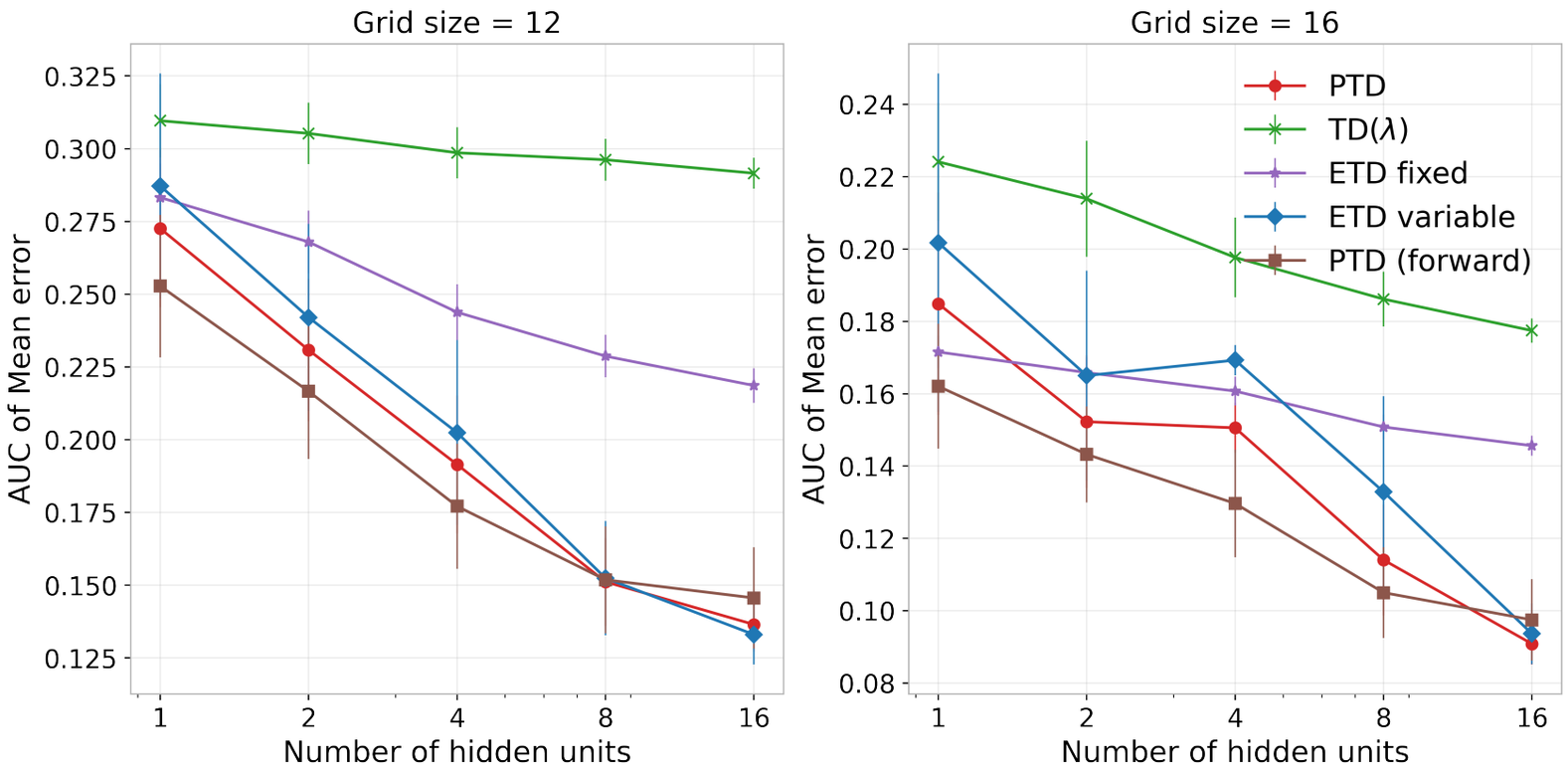


PTD - Results

Prediction (Non-linear FA)



PTD estimates, both the forward and backward variants, exhibit lower error across various combinations of grid and neural network sizes.

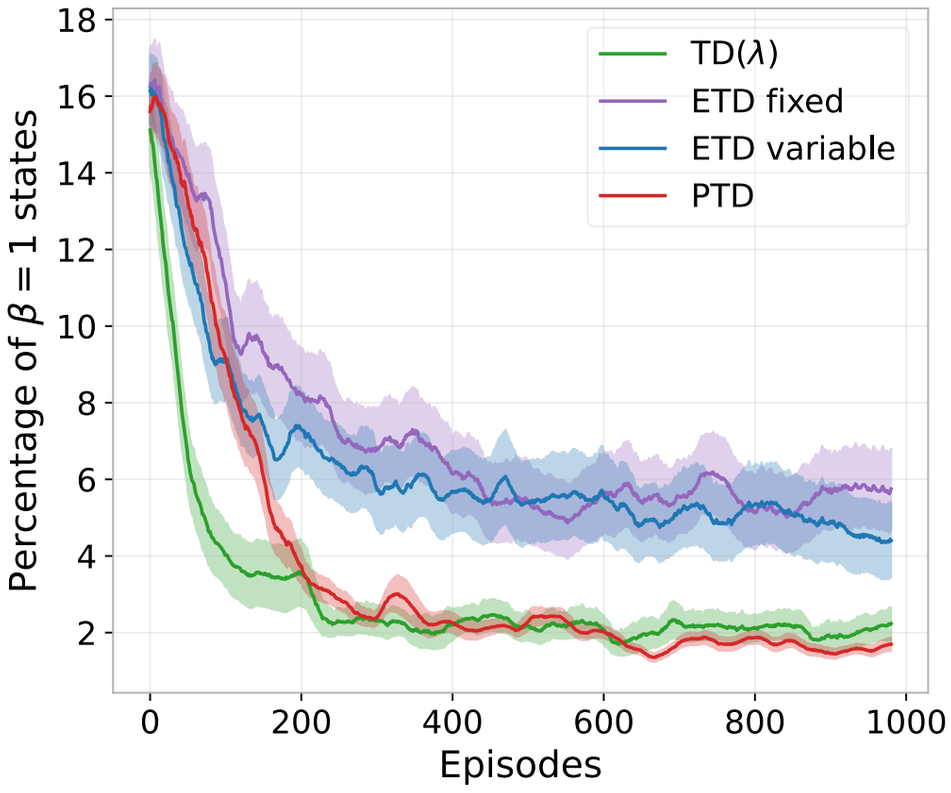


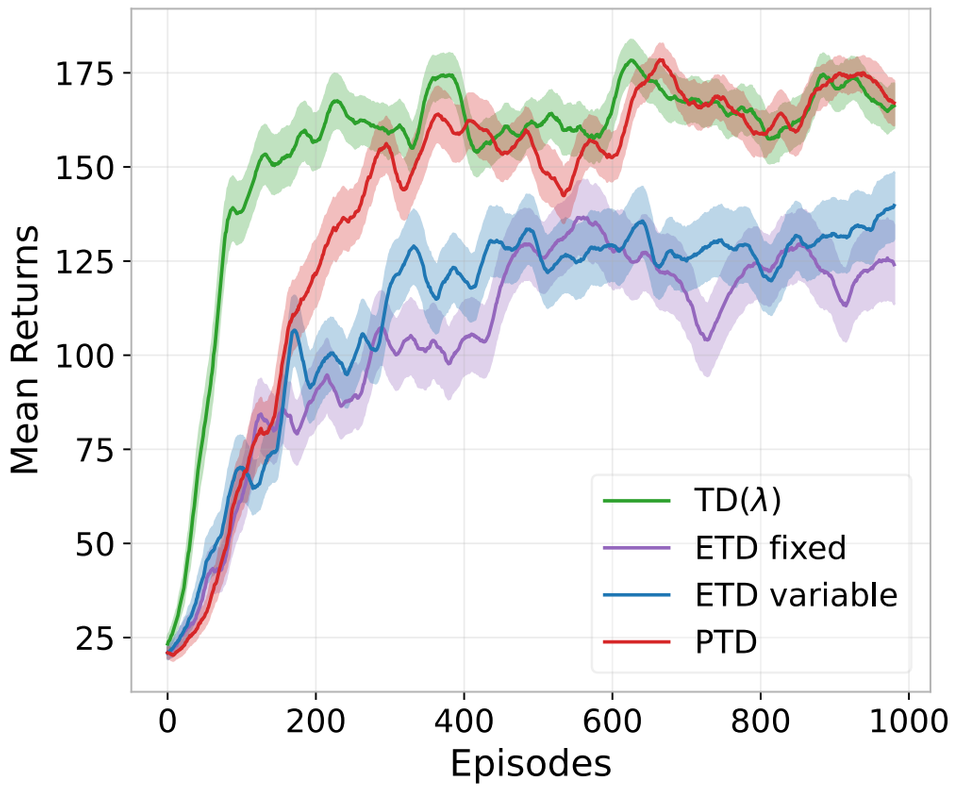
PTD - Results

Control (Cartpole)

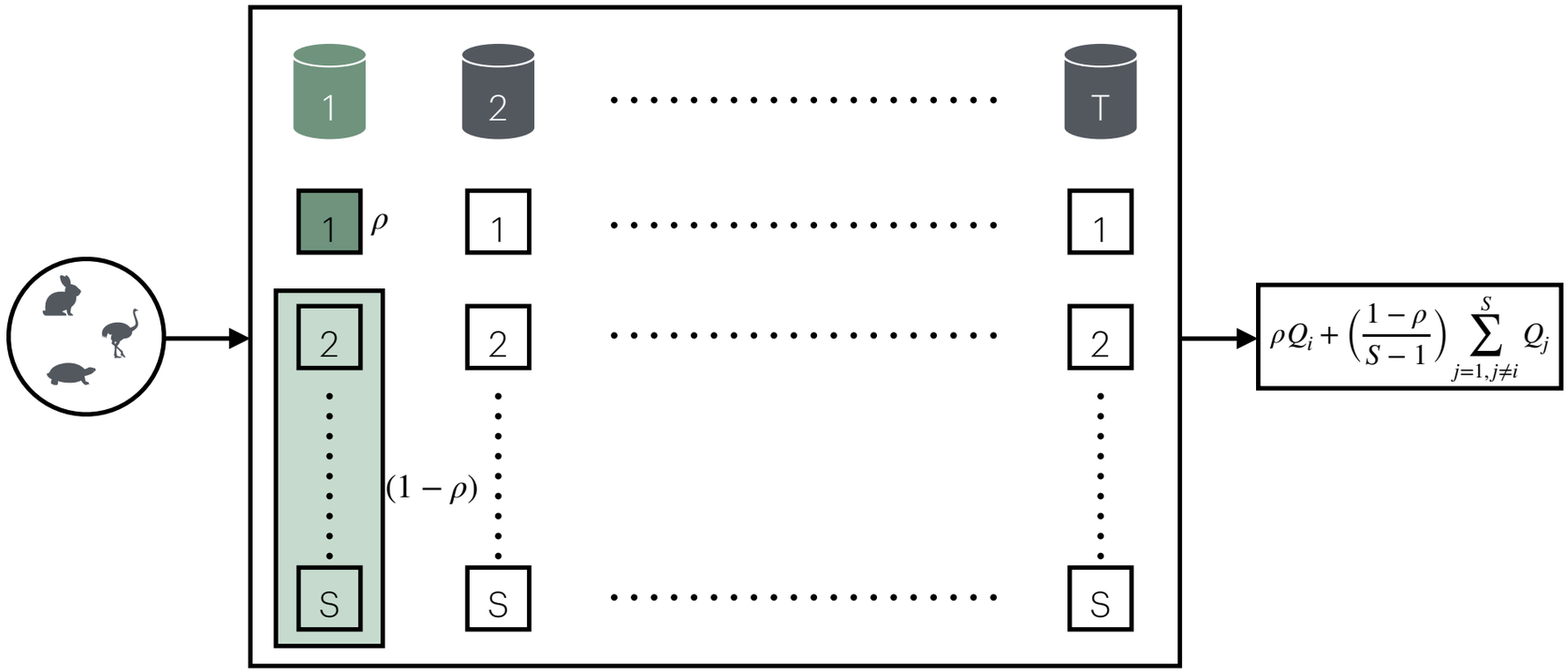
$\beta_t = 1$, if the cart position, velocity, or the pole angle changed by 1.0, 1.0, or 0.1 from the previous timestep, respectively

PTD-based Critic results in a similar performance while only bootstrapping and updating from a few states only





Non-Parametric Approximator



- Each bin has S slots to store observations and corresponding values.
- Generalization is controlled by ρ. No generalization when $\rho = 1$.
- Observations are ejected based on LRU strategy.

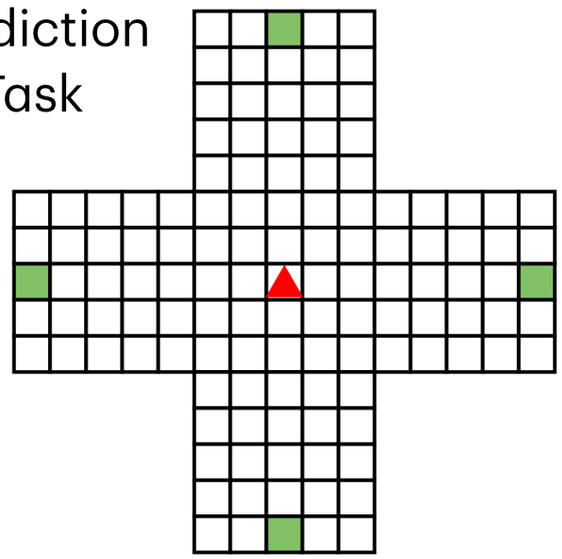
Permanent and Transient Successor Features*

- If $\mathscr{R}(s) = \boldsymbol{\phi}(s)^{\mathsf{T}}\boldsymbol{\omega}$, then the value of a state is $v_{\pi}(s) = \boldsymbol{\Psi}_{\pi}(s)^{\mathsf{T}}\boldsymbol{\omega}$.
- SFs are useful when only rewards change.
- PT-SFs, similar decomposition as that of value functions.
- Overall successor features are decomposed into permanent and transient successor features, $\boldsymbol{\Psi}(s) = \boldsymbol{\Psi}^{T}(s) + \boldsymbol{\Psi}^{P}(s)$.
- The value of a state is $v_{\pi}(s) = \left(\boldsymbol{\Psi}^{T}(s) + \boldsymbol{\Psi}^{P}(s)\right)^{\mathsf{T}}\boldsymbol{\omega}$.
- Useful in scenarios where transitions also change.

* Ongoing work

Results

Prediction Task

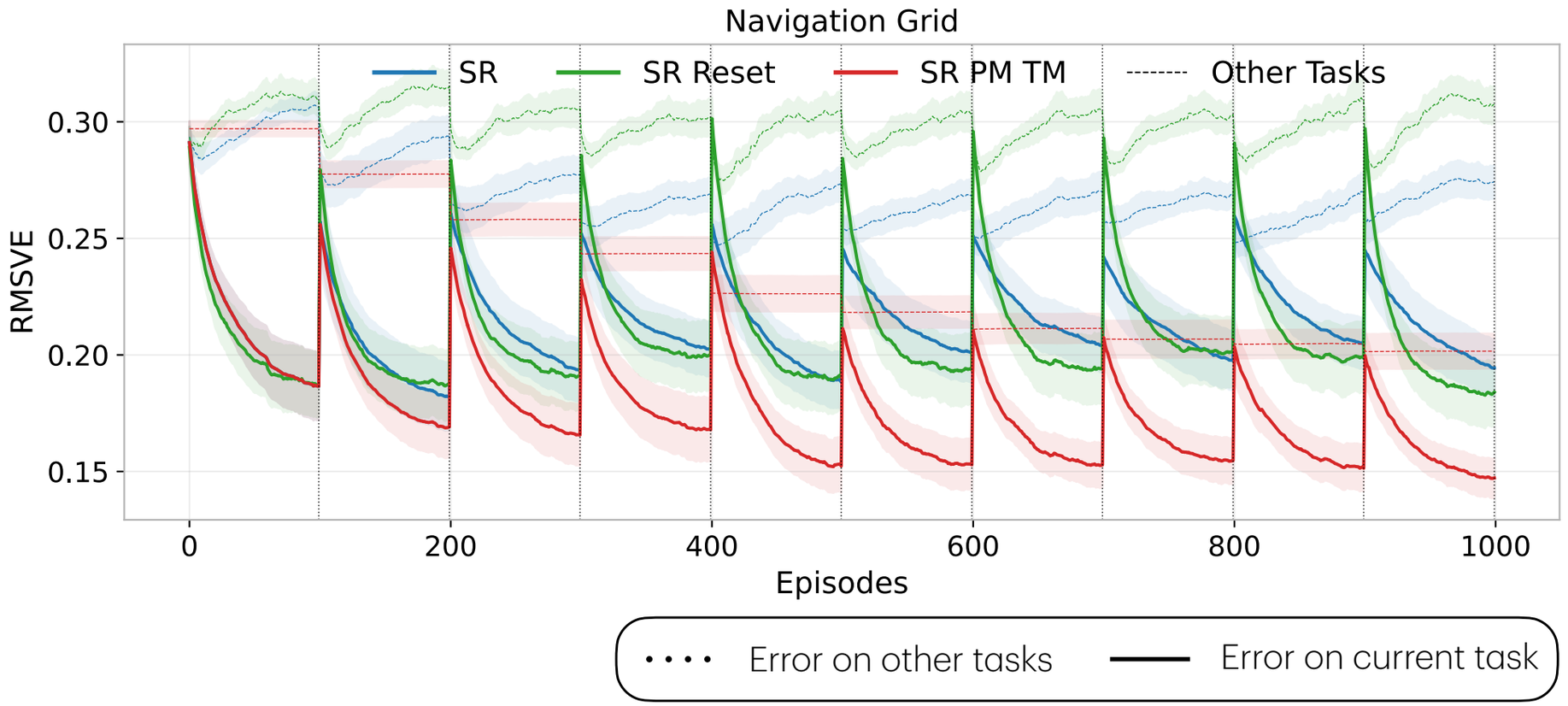


Rewards

Location	Reward
(0, 7)	2
(7, 0)	-1
(14, 7)	1.5
(7, 14)	-0.5

Task	Action Probability			
	Left	Right	Up	Down
0	0.55	0.15	0.15	0.15
1	0.15	0.55	0.15	0.15
2	0.15	0.15	0.55	0.15
3	0.15	0.15	0.15	0.55

Tasks



- Performance differences stem from the **quality of the SF** estimates since rewards are same.
- Our method **results in faster adaptation and lower initial errors** due to the PT decomposition.

End