An RNN architecture using Value Functions

Martha White
Assistant Professor
University of Alberta
Thanks to collaborators Matthew Schlegel, Andrew Patterson, Adam White, Rich Sutton

Goals for the talk

- Introduce **General Value Function Networks** (GVFNs) as a recurrent architecture
- Motivate how GVFNs provide an alternative training mechanism to other RNN algorithms

What is a GVF?

A GVF is a value function for a policy π, cumulant c and termination γ, with return:

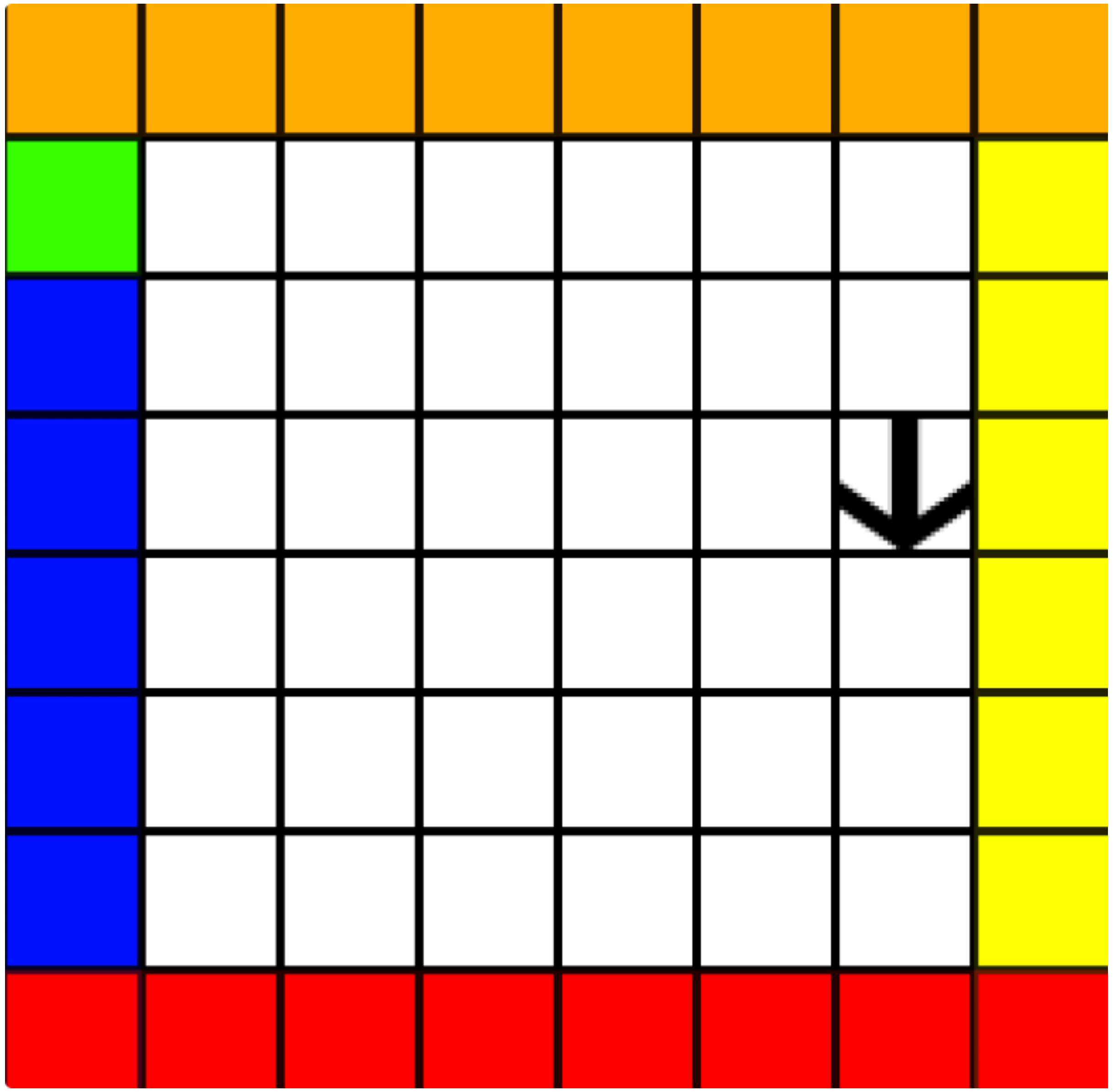
$$G_t = c(S_t, A_t, S_{t+1}) + \gamma(S_t, A_t, S_{t+1})G_{t+1}$$

Compass World

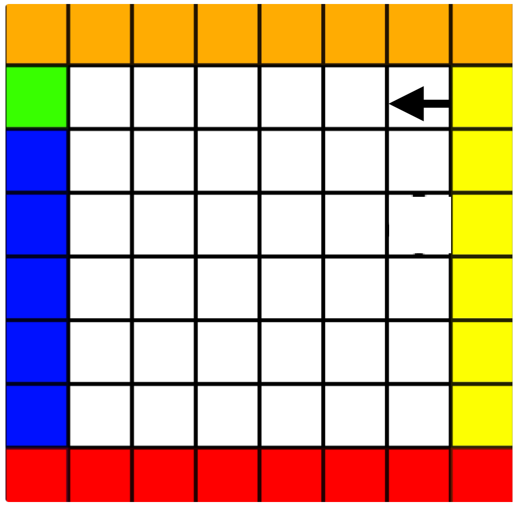
$$o_1 = \text{White}, a_1 = \text{Forward}, o_2 = \text{White}, a_2 = \text{Forward}, \ldots$$

- Agent can only see colour directly in front of it
- How can it predict (GVF) “What is the probability I will see Red if I go forward?”

$$\pi(\text{forward}|s) = 1.0$$
$$\gamma(\text{see red}) = 0, \text{else } 1$$
$$c(\text{see red}) = 1, \text{else } 0$$



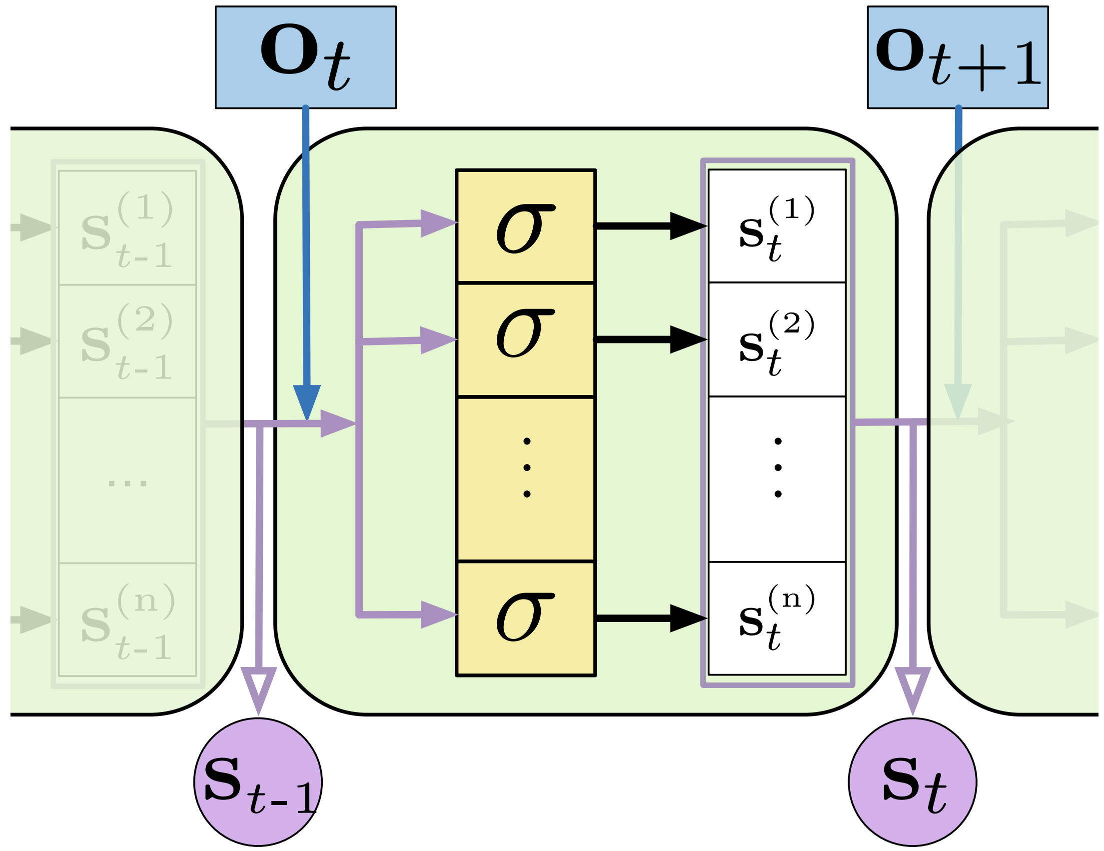
GVFs as a Predictive Representation



- Imagine if you could accurately predict
  - What is the probability I will see Orange if I turn right?
- Then you could easily infer
  - the probability of hitting the Green wall if you go forward is 1
  - the probability of hitting the Blue wall if you go forward is 0

We can use GVFs to define a new RNN unit

Recurrent NN



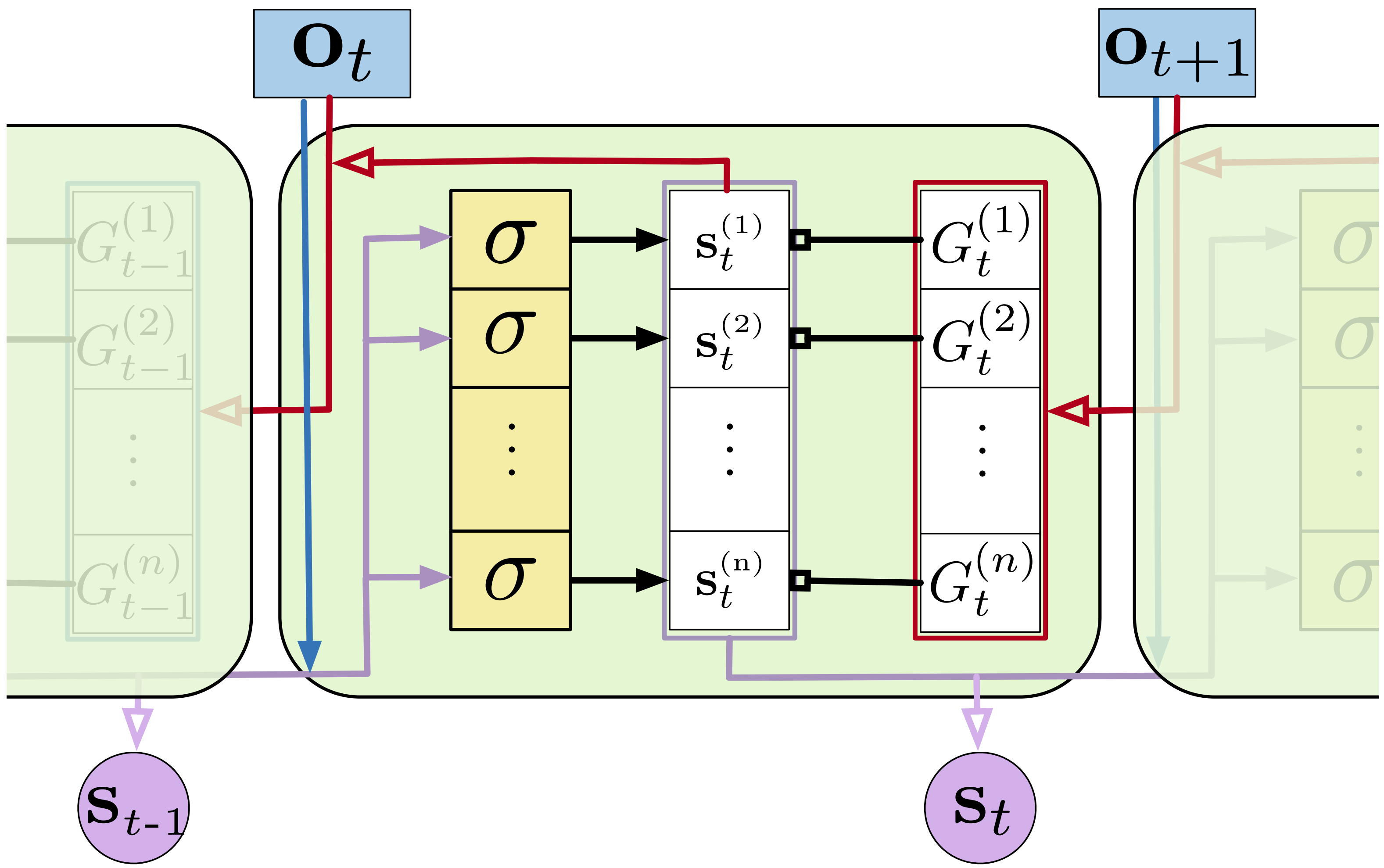
State-update function

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{o}_{t+1})$$

$$\mathbf{s}_{t+1} = \begin{bmatrix} \sigma\left(\begin{bmatrix} \mathbf{s}_t \\ \mathbf{o}_{t+1} \end{bmatrix}^\top \boldsymbol{\theta}^{(1)} \right) \\ \vdots \\ \sigma\left(\begin{bmatrix} \mathbf{s}_t \\ \mathbf{o}_{t+1} \end{bmatrix}^\top \boldsymbol{\theta}^{(n)} \right) \end{bmatrix}$$

GVF Networks

- An RNN where hidden states are constrained to be value function predictions (i.e., GVFs)



State-update function

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{o}_{t+1})$$

$$G_t^{(i)} = C_{t+1}^{(i)} + \gamma_{t+1} s_{t+1}^{(i)}$$

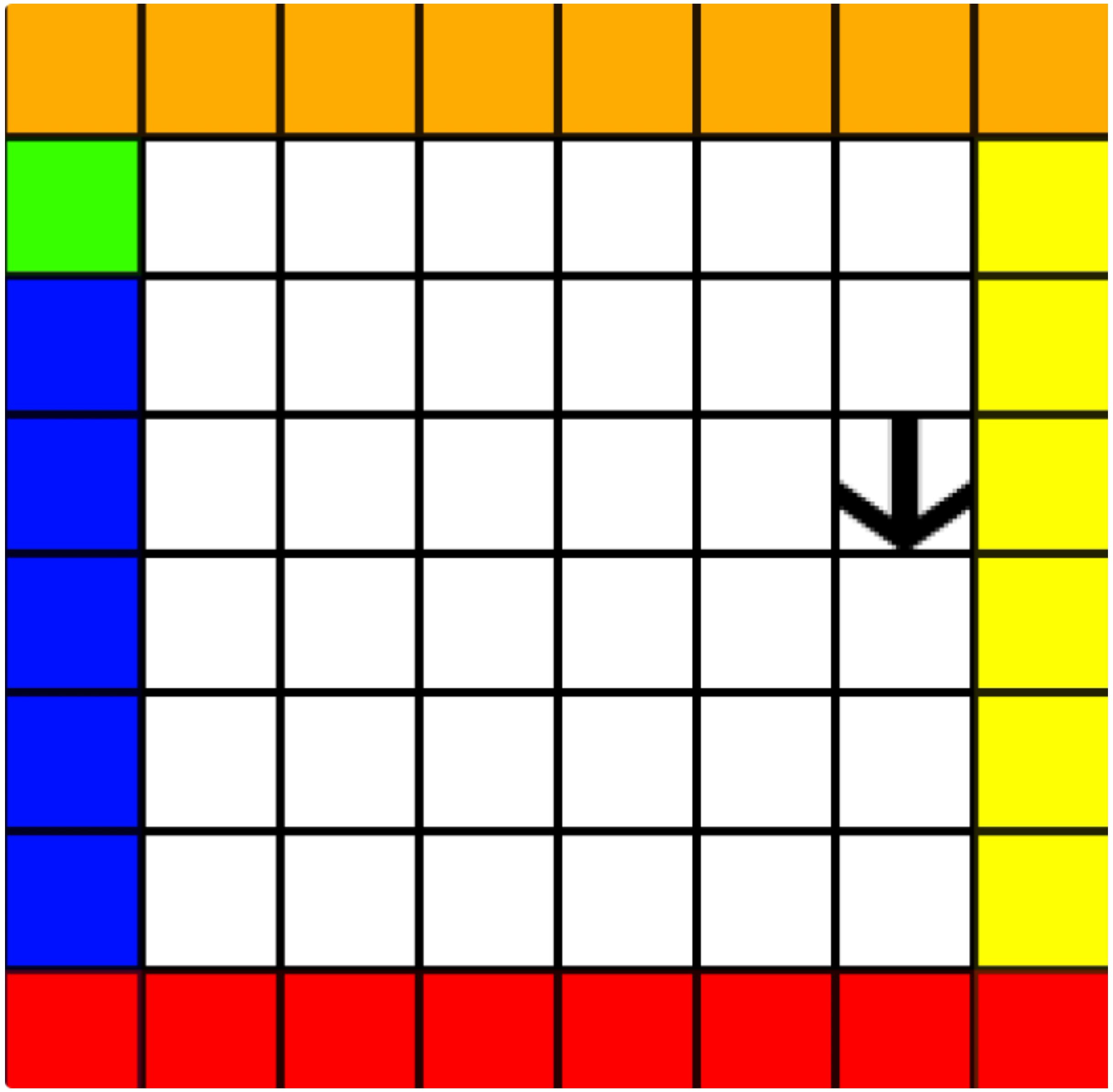
$$C_{t+1}^{(i)} = \mathbf{o}_{t+1,j}$$

Change in update

- Additional losses on each node separately
- Combined loss called the mean-squared projected Bellman network error (MSPBNE)
  - originally introduced by David Silver, for TD-networks
- Still compute gradient back-in-time, with a more complex loss (still use RTRL, or BPTT, etc.)

Experiment in Compass World

- Goal: predict probability of reaching a wall of a particular colour, if drive forward



- 5 evaluation GVFs
- Random behaviour policy results in long-term dependencies
- Compared GVFNs and GRUs

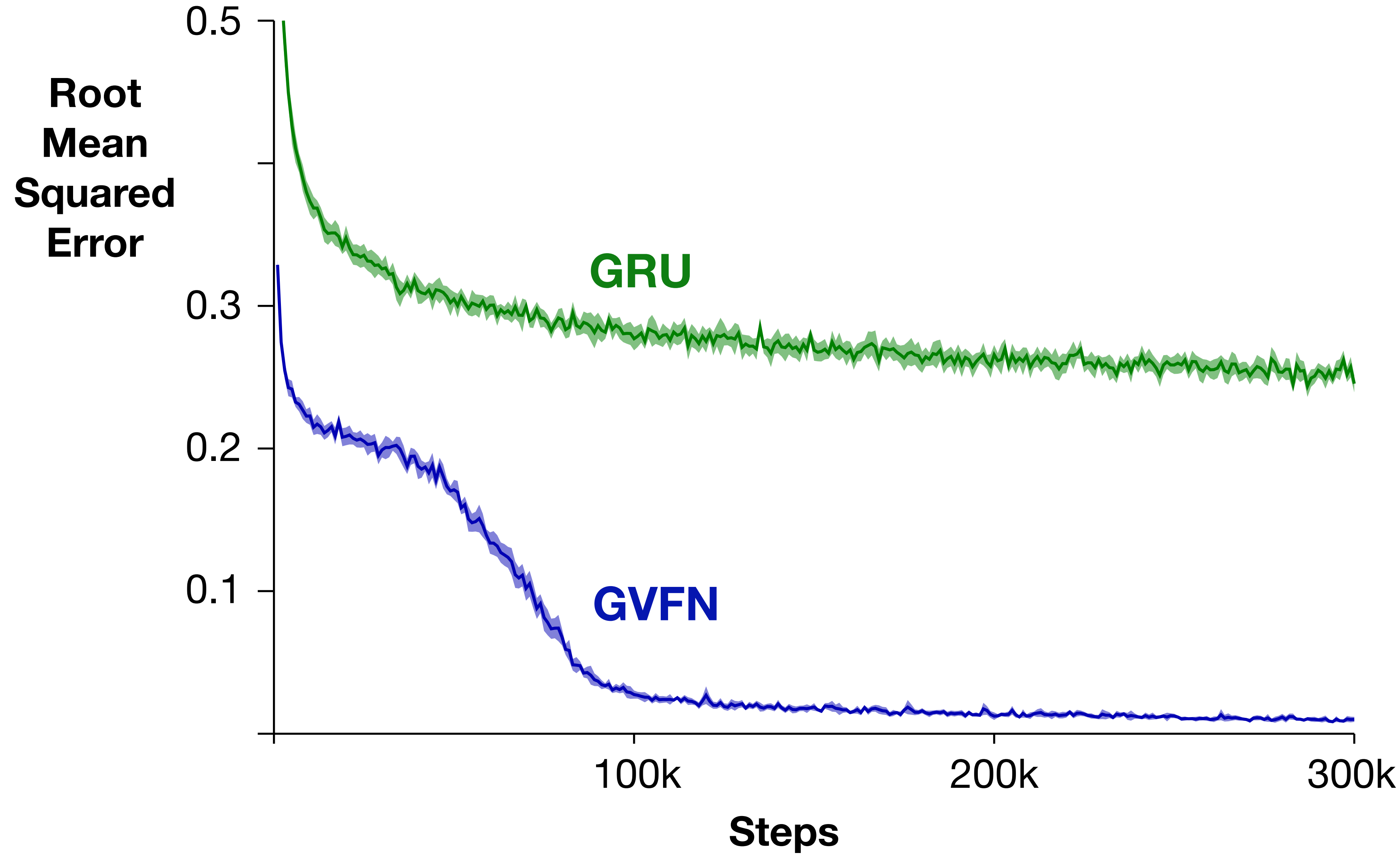
GVFN Architectures

- Expert-designed network: 64 GVFs
  - predicting probability of reaching wall, in 1-step, in 2-steps (using composition) and multiple steps, under 2 different policies
- Randomly-generated network: randomly generate GVFs from a basic set of GVFs

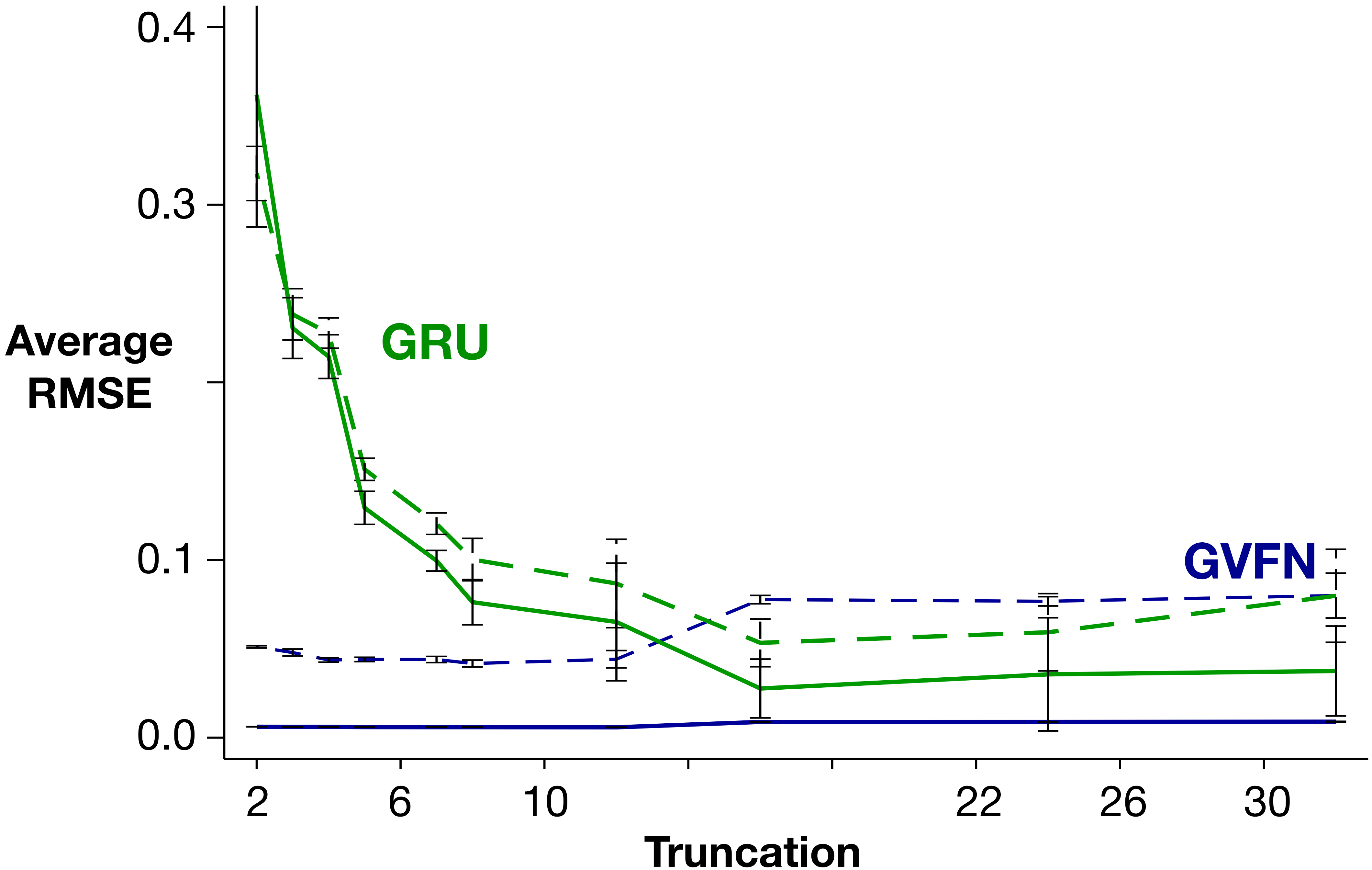
Experiment settings

- Swept GRU size and truncation in tBPTT
- Swept stepsizes for both methods
- Learning for about 300k steps (partial observability is hard)
- Averaged over 100 runs
- RMSE to true value functions for 5 evaluation GVFs

GVFNs are much more effective for this long memory problem



RingWorld domain



Q1: Is the primary goal to estimate the full gradient?

- It is an unreasonable request to adjust the weights based on its influence all the way back in time
- Option 1: Approximate this gradient
  - tBPTT, RTRL, UORO, etc.
- Option 2: Consider completely different criteria
  - GVFNs with one-step updates as an approximate fixed-point iteration

Update states using a TD update

- Treat as a fixed point problem: $s_t^{(i)} \approx C_{t+1}^{(i)} + \gamma_{t+1} s_{t+1}^{(i)}$
    - Ignore gradients back-in-time, treat features as given $[\mathbf{s}_t, \mathbf{o}_{t+1}]$
- Incorporate eligibility traces
    - Lambda-return is less biased, incorporates more information about future cumulants (less bootstrapping)
    - Also a mechanism for spreading back value (credit assignment)

$$\mathbf{s}_t^{(i)} = \sigma(\mathbf{x}_t^\top \mathbf{w}) \qquad \lambda = 0 \qquad \lambda > 0$$

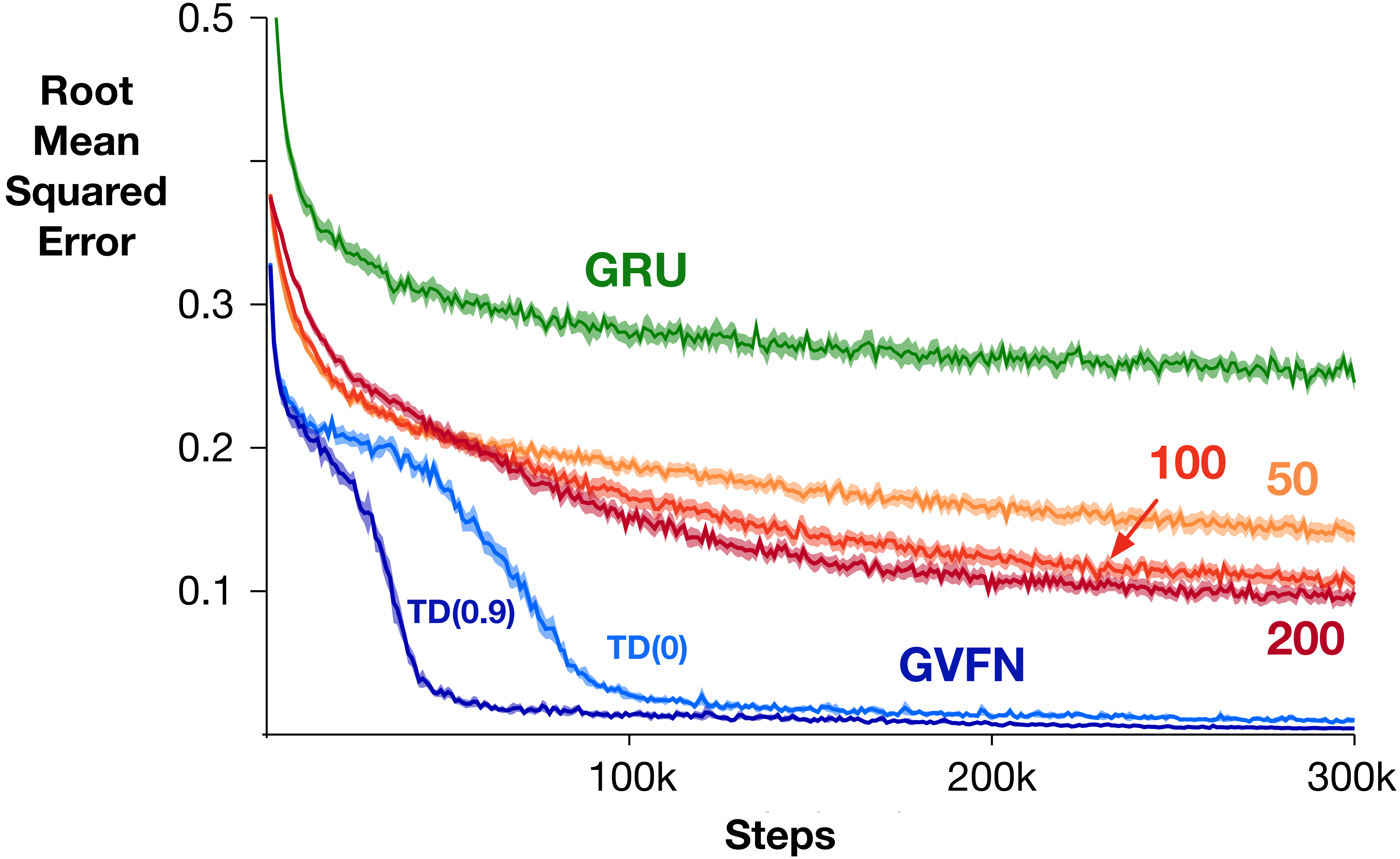
$$\mathbf{x}_t = [\mathbf{s}_{t-1}, \mathbf{o}_t] \qquad \Delta\mathbf{w} = \delta_t \mathbf{x}_t \qquad \Delta\mathbf{w} = \delta_t \mathbf{z}_t$$

$$\mathbf{z}_t = \mathbf{x}_t + \gamma_{t-1}\lambda_{t-1}\mathbf{z}_{t-1}$$

Example in Compass World

- Constantly see zero for the colour, then suddenly hit a wall and see a 1, giving a higher TD-error
- Eligibility trace sends back this TD-error, adjusting value estimates in previous states to predict a 1 (if gamma = 1)
- Now when return to state, value estimate more accurate
- Not sending back gradient credit; rather, sending back return information

Utility of traces for GVF networks



Q2: Can we use traces directly for RNNs?

- Traces for the GVFs arise from the definition of the lambda-return
- For RNNs, the predictions are not about the future
- Option 1: If GVFs are auxiliary losses, can eligibility traces be used effectively?
- Option 2: Can we obtain a principled derivation with traces, to send back credit for error on this step to past weights?

Preliminary results that not effective as auxiliary tasks