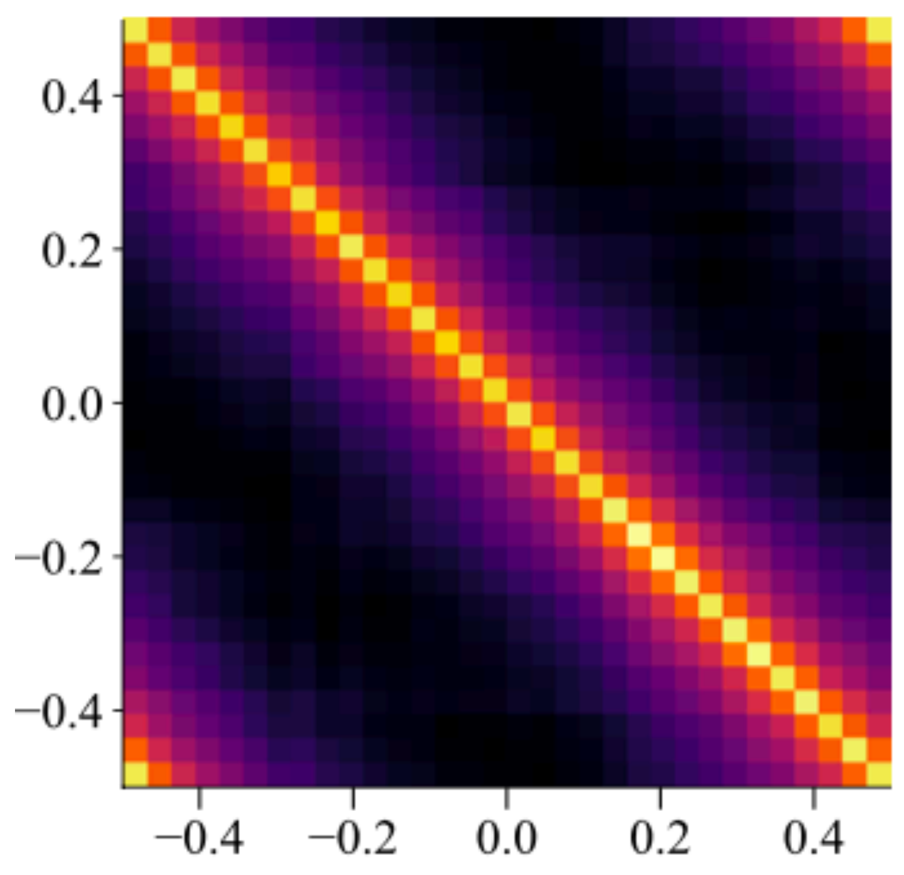
Overcoming The Spectral-Bias of Neural Value Approximation



Ge Yang*, Anurag Ajay* & Pulkit Agrawal

***Equal contribution, order determined randomly**

Q learning with neural networks suffers from the “Spectral Bias”

Q learning with neural networks suffers from the “**Spectral Bias**”

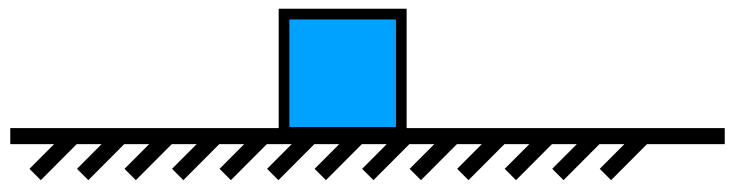
Q learning with neural networks suffers from the “Spectral Bias”

Where it is unable to fit the high-frequency components of an optimal value function

Q Learning with Neural Networks suffer from the “Spectral Bias”

Where it is unable to fit the high-frequency components of an optimal value function

Toy MDP

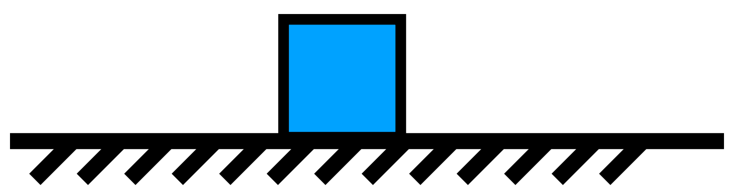


$\mathcal{S} \in [0, 1)$

Q Learning with Neural Networks suffer from the “Spectral Bias”

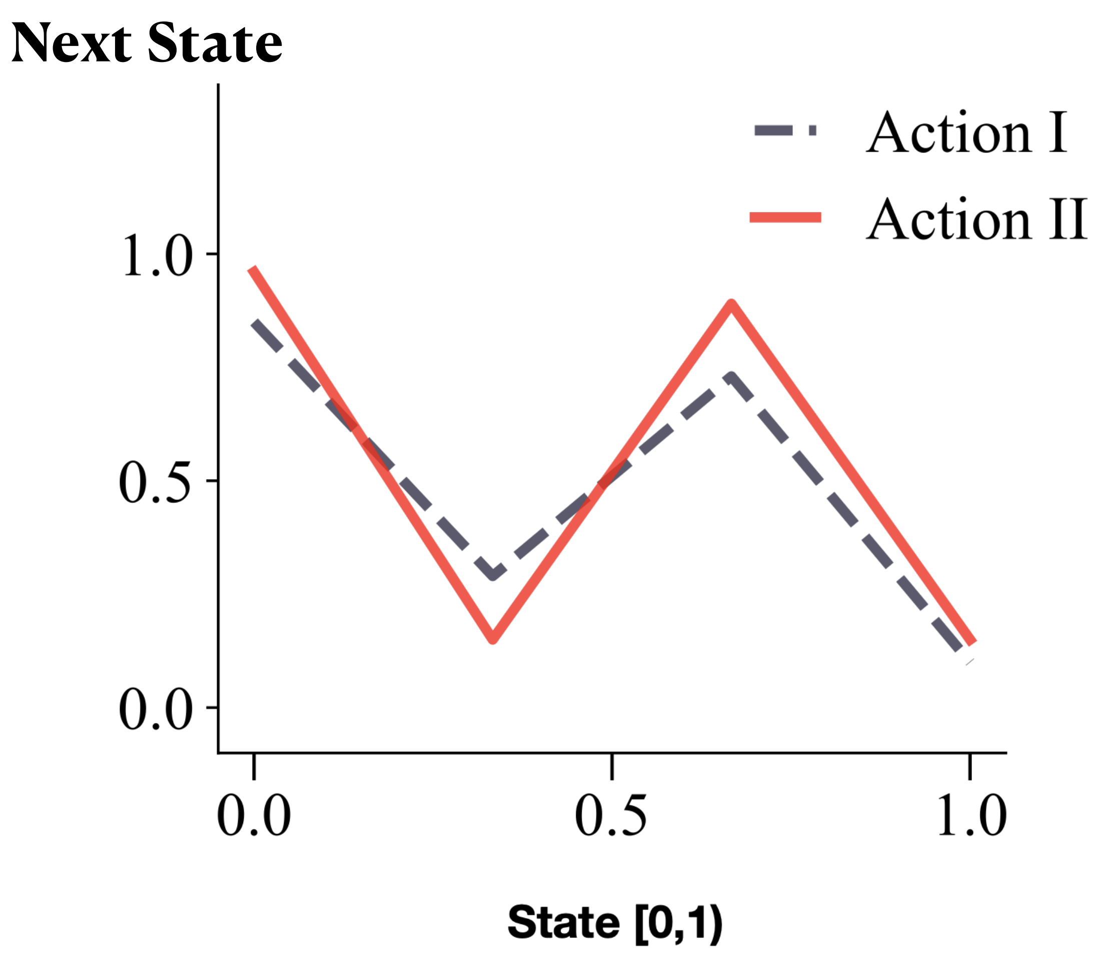
Where it is unable to fit the high-frequency components of an optimal value function

Toy MDP

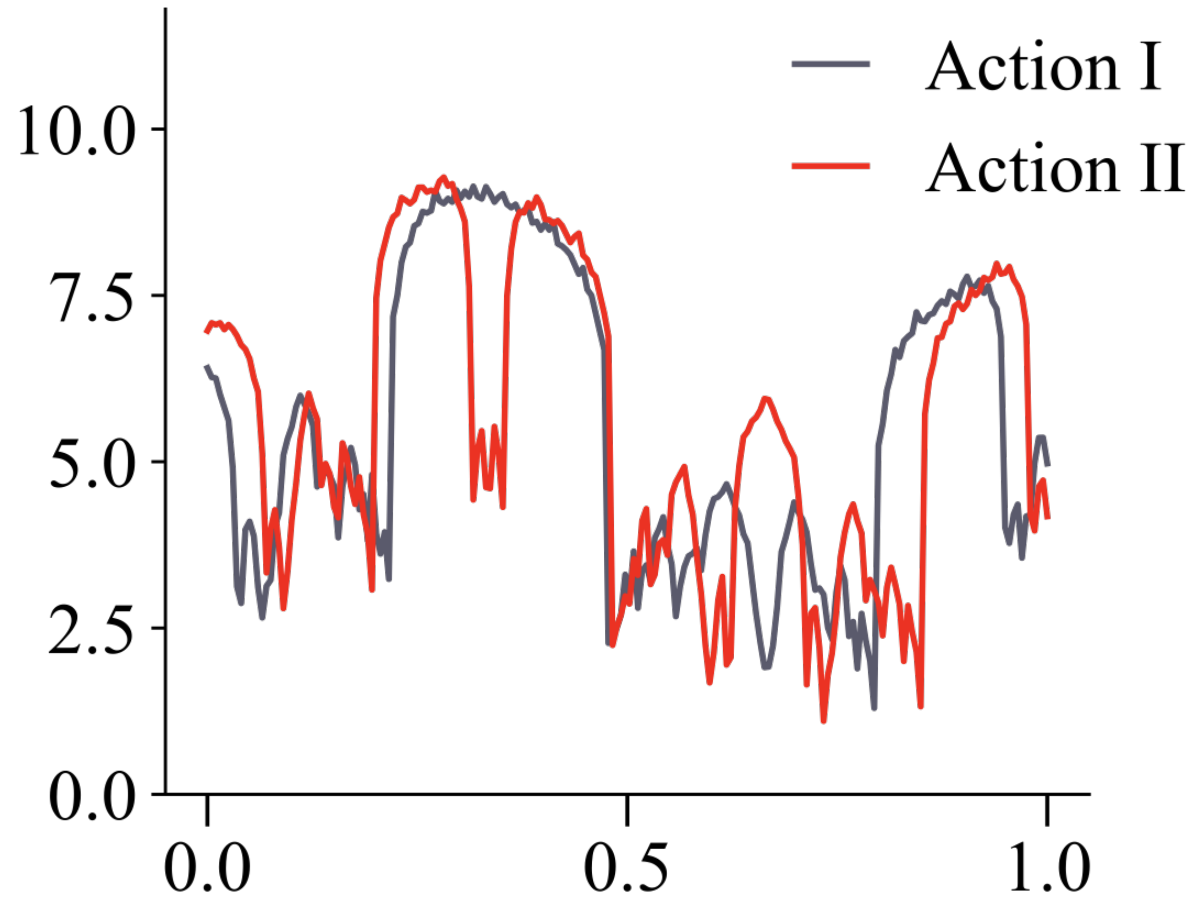


$\mathcal{S} \in [0, 1)$

$\mathcal{A} \in \{0, 1\}$



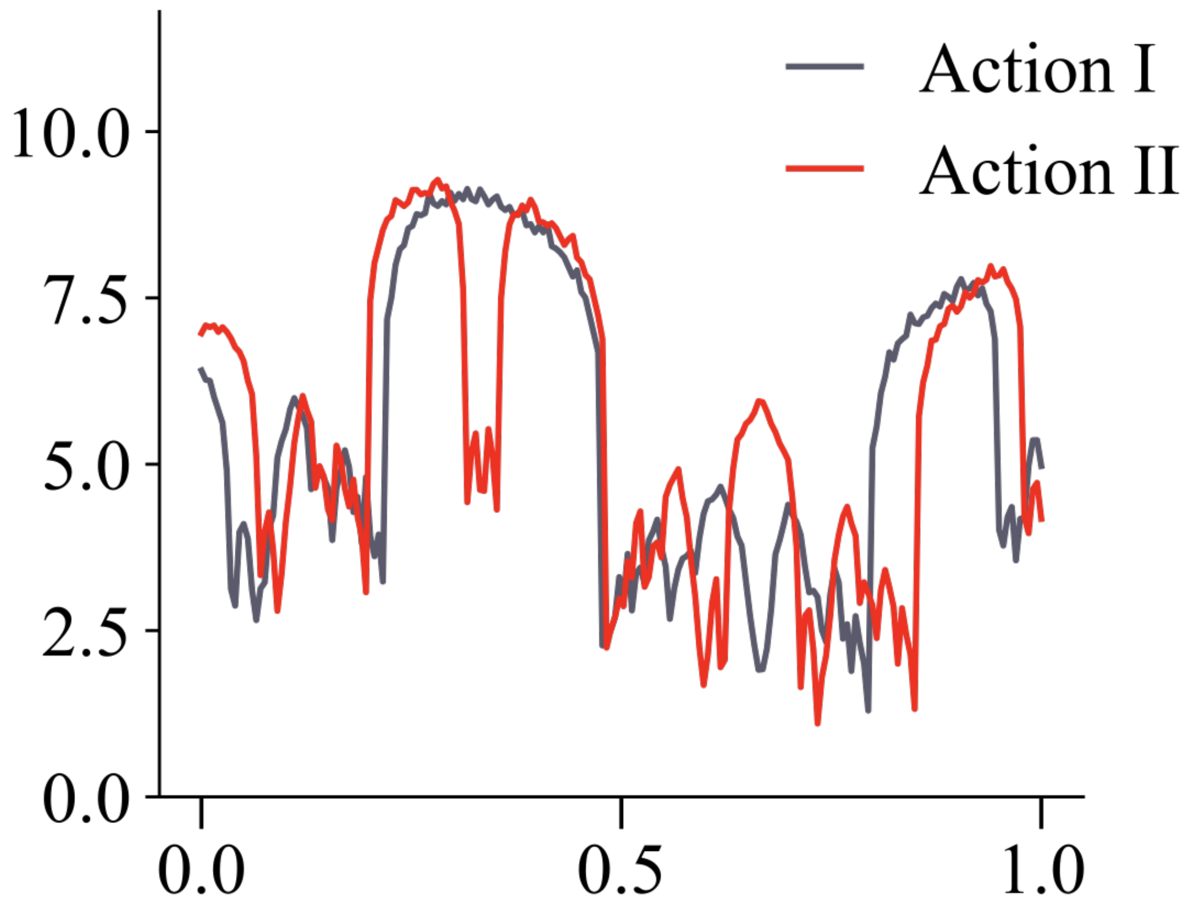
Optimal Q Function



Dong *et al, On the Expressivity of Neural Networks for Deep RL*

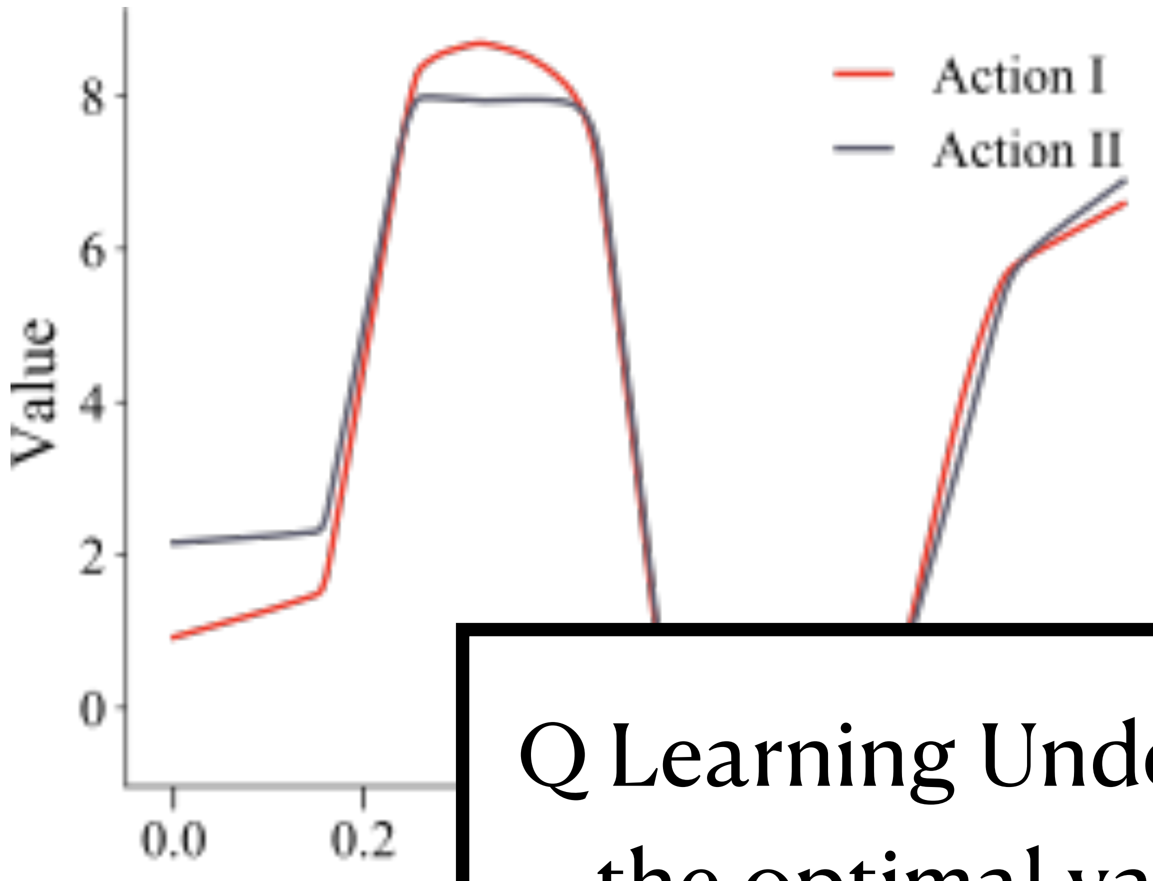
Neural Fitted Q Iteration

Optimal Q Function



Neural Fitted Q Iteration

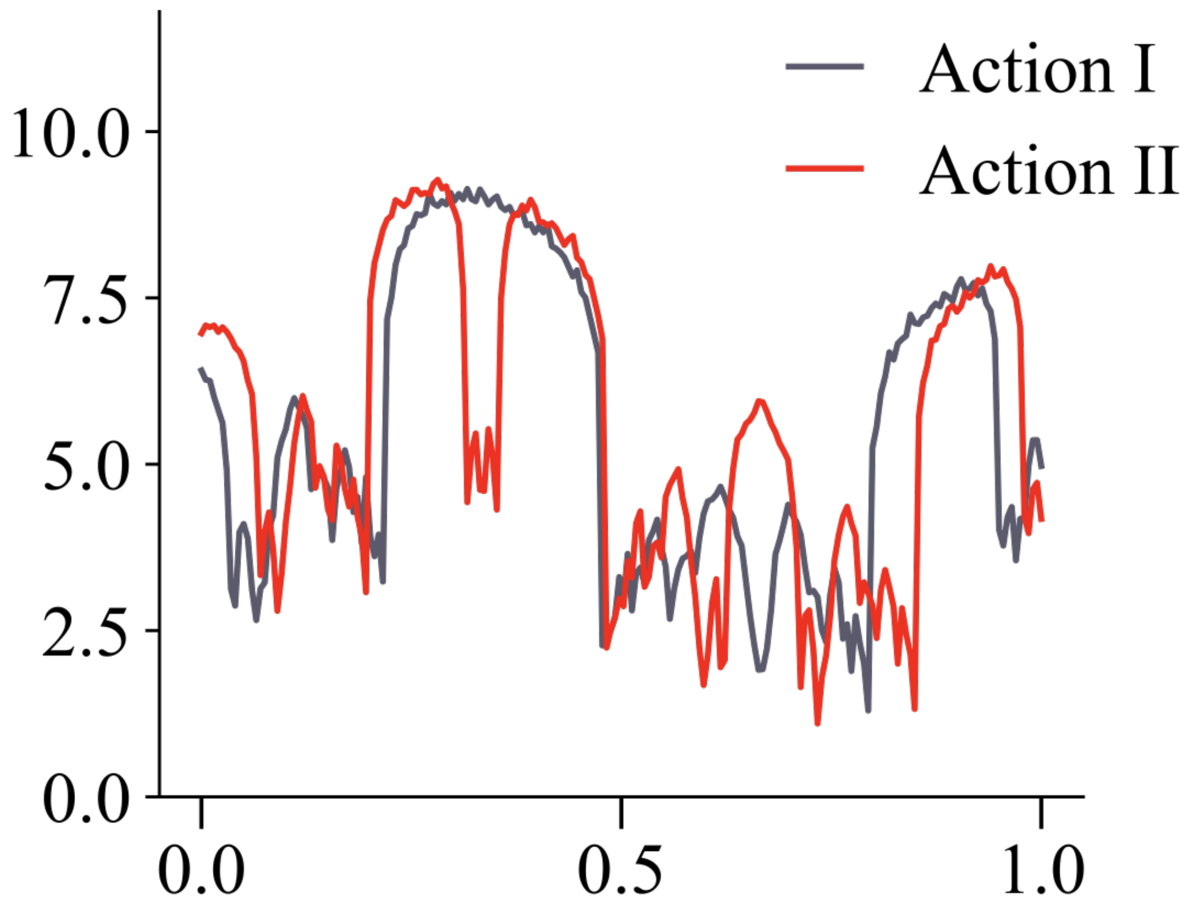
Initialize Q and *target net* $\hat{Q}$
while **do**
  for k steps **do**
    $Q(s,a) \leftarrow R(s,a) + \gamma \hat{Q}(s', a^*)$
  $\hat{Q} \leftarrow Q$
return Q



Q Learning Under-fits the optimal value!

Neural Fitted Q Iteration

Optimal Q Function



Neural Fitted Q Iteration

Initialize Q and *target net* $\hat{Q}$

while do

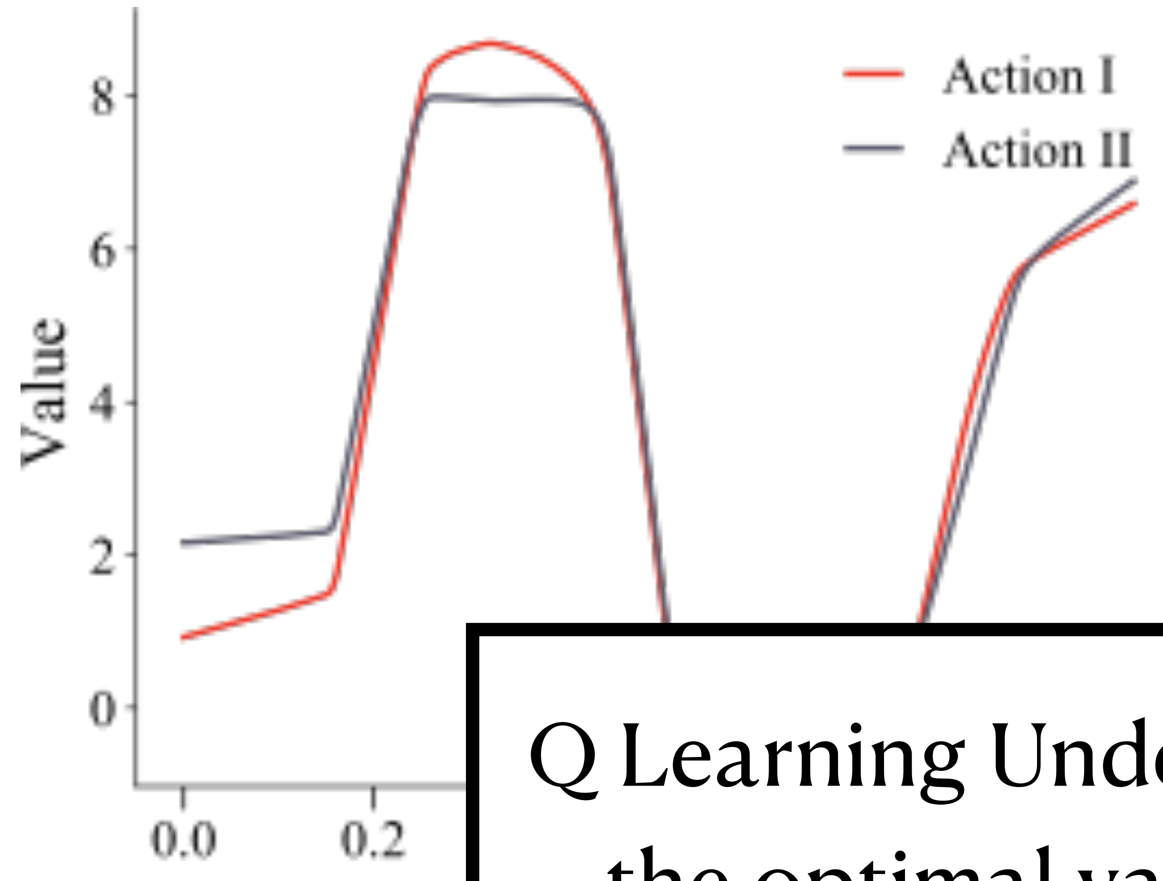
  for k steps **do** ▷ k gradient steps

    $Q(s,a) \leftarrow R(s,a) + \gamma \hat{Q}(s', a^*)$

  $\hat{Q} \leftarrow Q$ ▷ update target Q

return Q

Target Network



Q Learning Under-fits the optimal value!

Deep Q Learning is in the "early stopping" regime...

Neural Fitted Q Iteration

Initialize Q and *target net* $\hat{Q}$

while do

  for k steps **do** ▷ k gradient steps

    $Q(s,a) \leftarrow R(s,a) + \gamma \hat{Q}(s', a^*)$

  $\hat{Q} \leftarrow Q$ ▷ update target Q

return Q

...where model bias and optimization interact in complex ways.

Understanding deep reinforcement learning

requires understanding supervised learning under "early stopping."

NTK and the “early stopping” regime

Recent works in deep learning theory [Jacot *et al,* Arora *et al*] offer significant insight into how the neural network evolves under gradient descent

$$f - f^* = e^{K\langle \xi, \hat{\xi} \rangle}(f_0 - f^*) \quad \text{where} \quad K\langle \xi, \hat{\xi} \rangle = \langle \nabla_\theta f(\xi)^T \nabla_\theta f(\hat{\xi}) \rangle$$

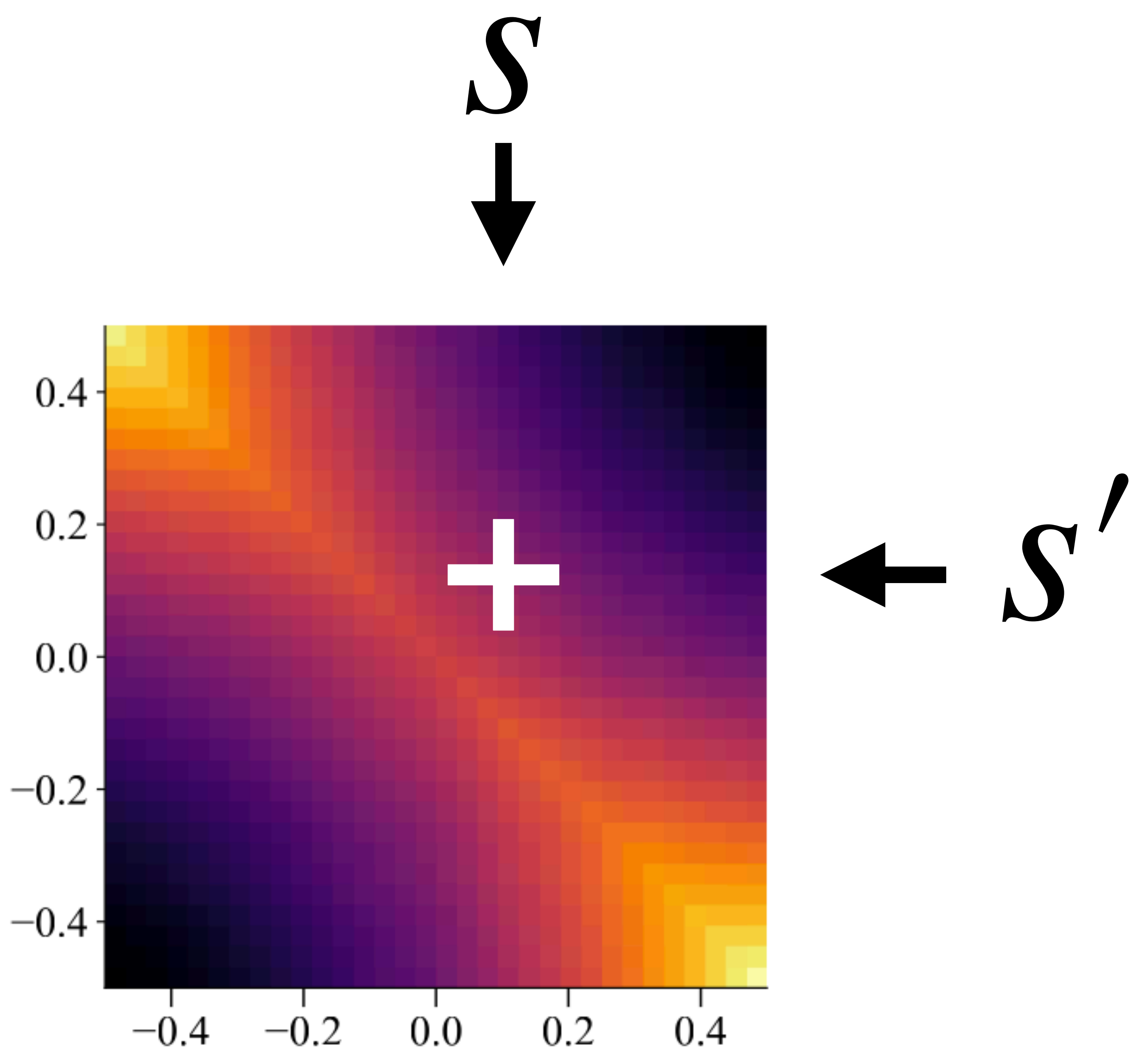
In particular, the convergence at a spectral frequency f_i is proportional to the Eigen value Λ_i of the NTK , which decays rapidly for an MLP.

$$f - f^* = e^{\Lambda_i}(f_0 - f^*) \quad \text{where} \quad K\langle \xi, \hat{\xi} \rangle = \sum_i \Lambda_i f_i$$

NTK and the "early stopping" regime

The vanilla MLP generalize in an uncontrolled fashion, which manifest as aliasing between gradient vectors over long-horizon.

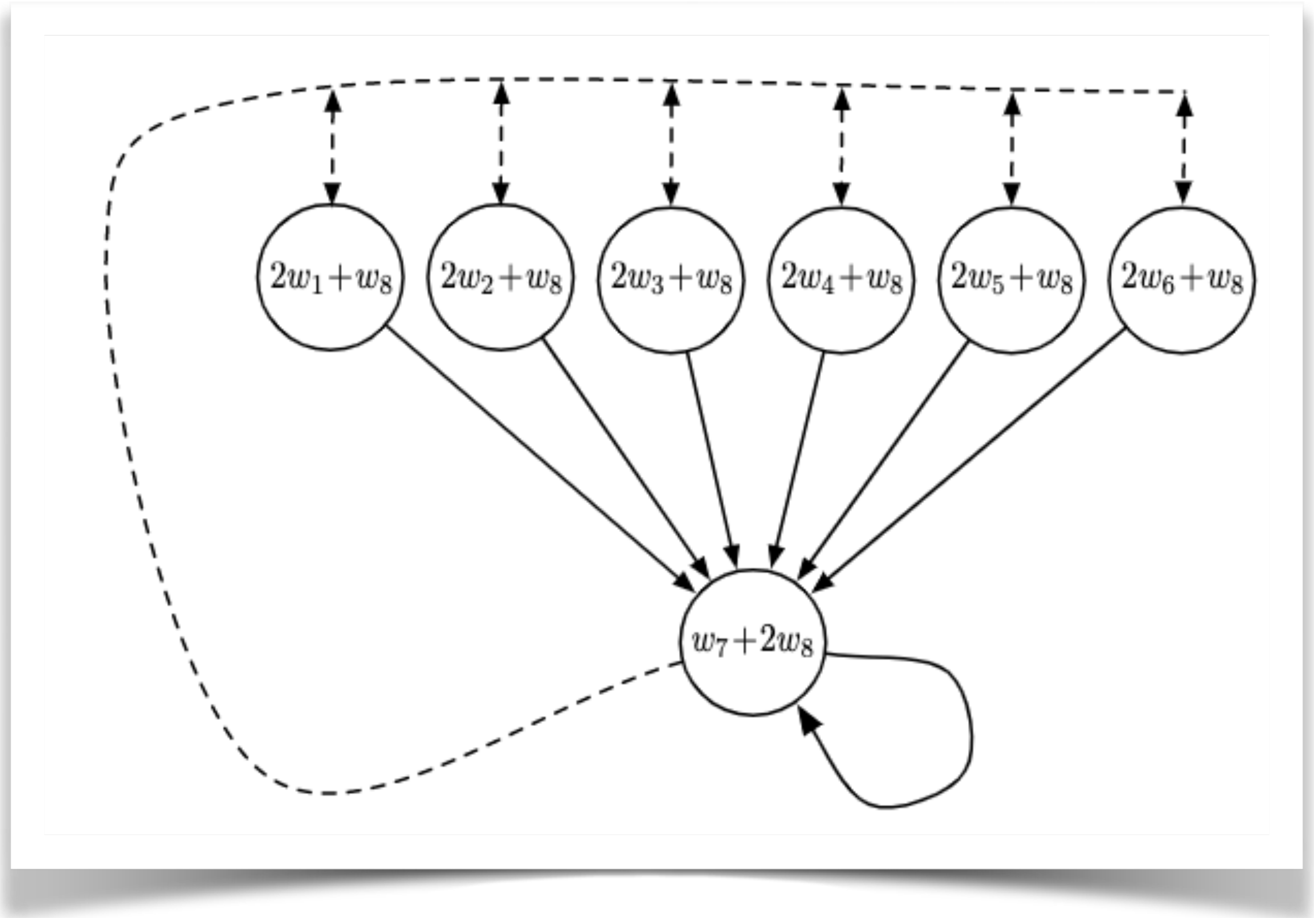
$$K\langle s, s' \rangle = \langle \nabla_\theta f(s)^T \nabla_\theta f(s') \rangle$$



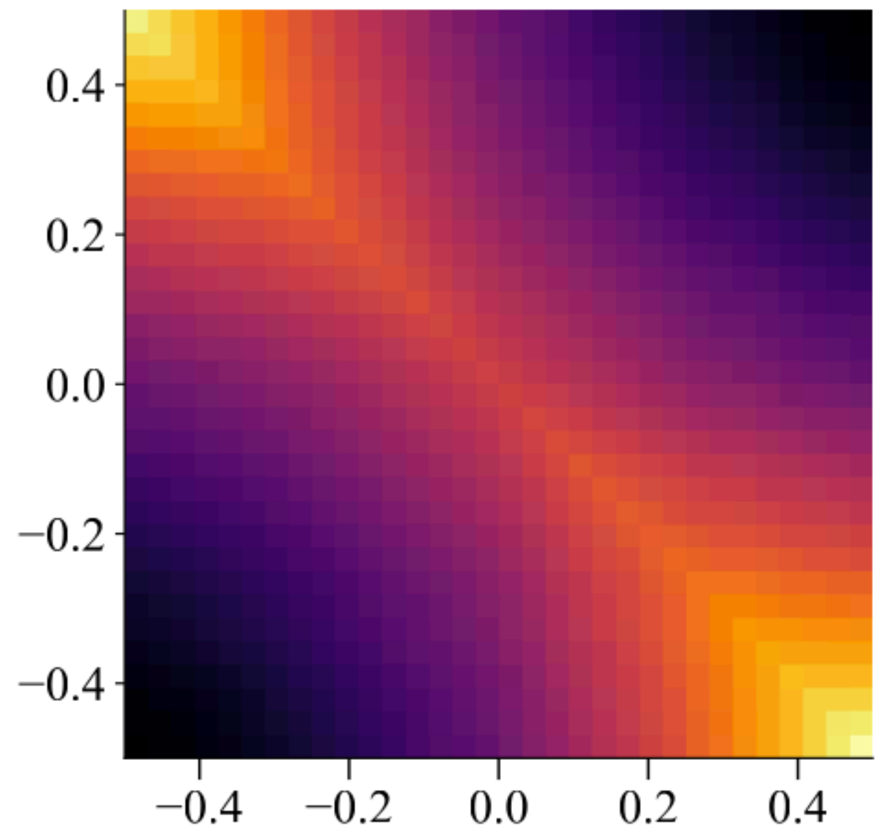
(a) MLP + ReLU

The "Spectral Bias" and NTK

State aliasing is *unavoidable* with function approximators, but the cross-talk can cause divergence, as shown in Baird *et al*.



Baird *et al* 1995



(a) MLP + ReLU

Achiam *et al, Towards Characterizing Divergence in Deep Q Learning*

Baird *et al*, Residual Algorithms: Reinforcement Learning w/ Function Approximation

To overcome the spectral bias of neural value approximation,
we need to produce controlled generalization that is ***local*** in nature.

How do we do that?

Controlled Generalization via Random Fourier Features

Luckily, the *random Fourier features* (Rahimi & Recht 2008) offered a way to construct gaussian kernels using a spectral mixture

$$k = \langle \xi, \hat{\xi} \rangle \approx \mathbf{z}(\xi)^T \mathbf{z}(\hat{\xi})$$

$$\text{where } z(\xi) = \sum_i w_i e^{2\pi k_i} \quad \text{and} \quad w_i \sim \mathcal{F}(\mathcal{K}*)\,.$$

This allows us to construct a composite neural tangent kernel that interpolates ***Locally***, so that we can specify how the network generalizes.

Controlled Generalization via Random Fourier Features

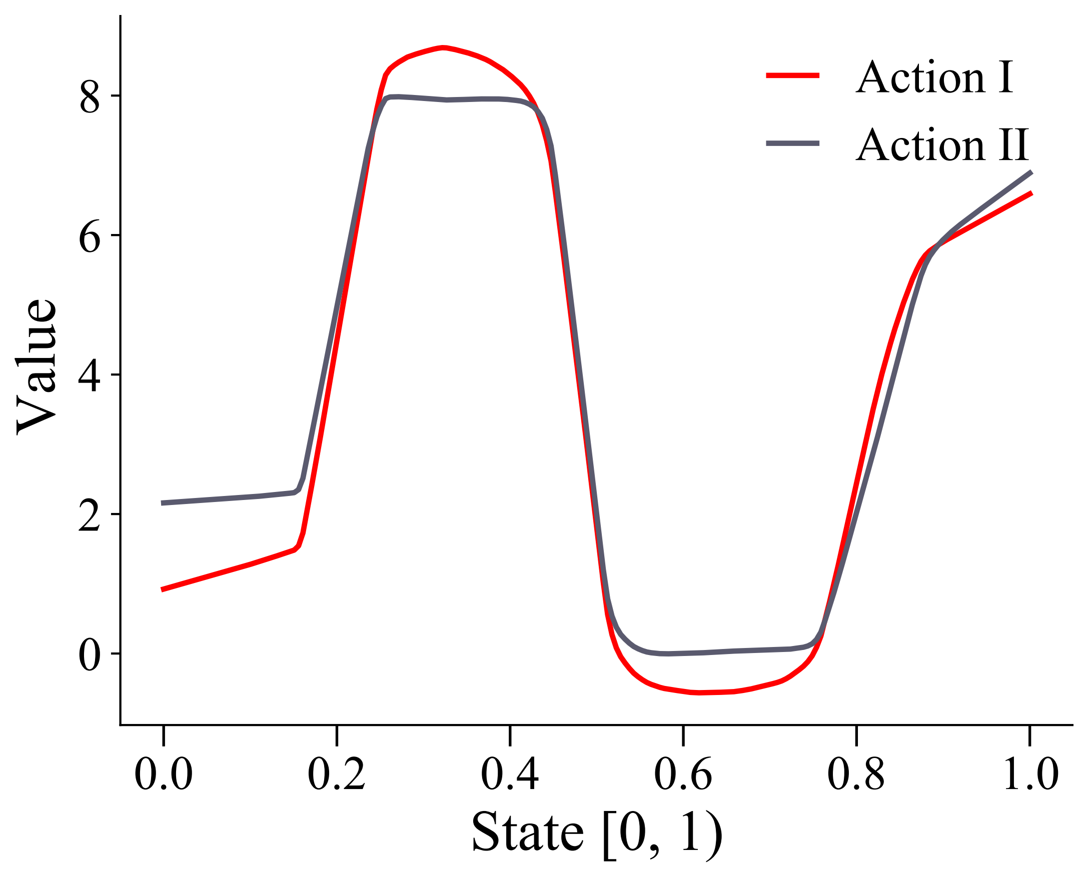
```python
import torch
import torch.nn as nn

net = nn. Sequential(
    nn.Linear(1, 200),
    nn.ReLU(),
    nn.Linear(200, 200),
    nn.ReLU(),
    nn.Linear(200, 1),
    nn.ReLU(),
)
```

```python
import torch
import torch.nn as nn

net = nn. Sequential(
    nn.Linear(1, 200),
    lambda x: torch.sin(x),
    nn.Linear(200, 200),
    nn.ReLU(),
    nn.Linear(200, 1),
    nn.ReLU(),
)
```

On the Toy domain,

```
import torch
import torch.nn as nn

net = nn.Sequential(
    nn.Linear(1, 200),
    nn.ReLU(),
    nn.Linear(200, 200),
    nn.ReLU(),
    nn.Linear(200, 1),
    nn.ReLU(),
)
```

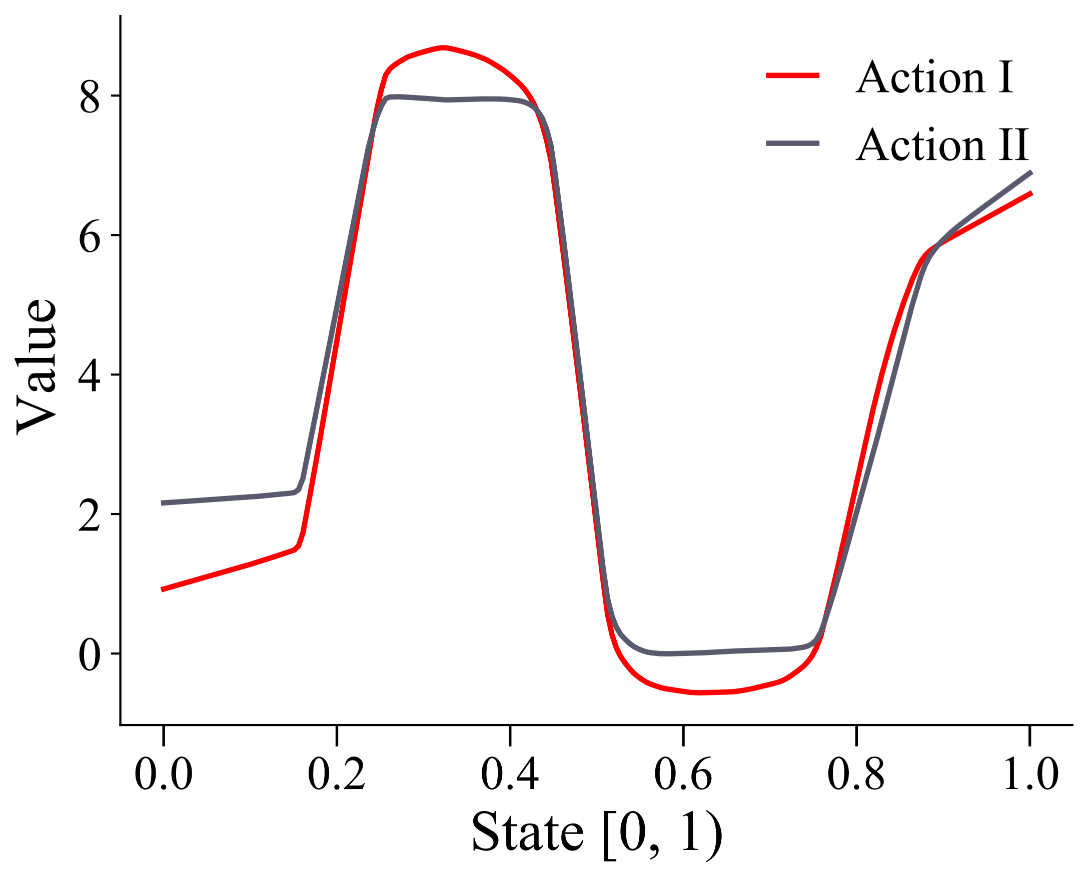
FQI + MLP



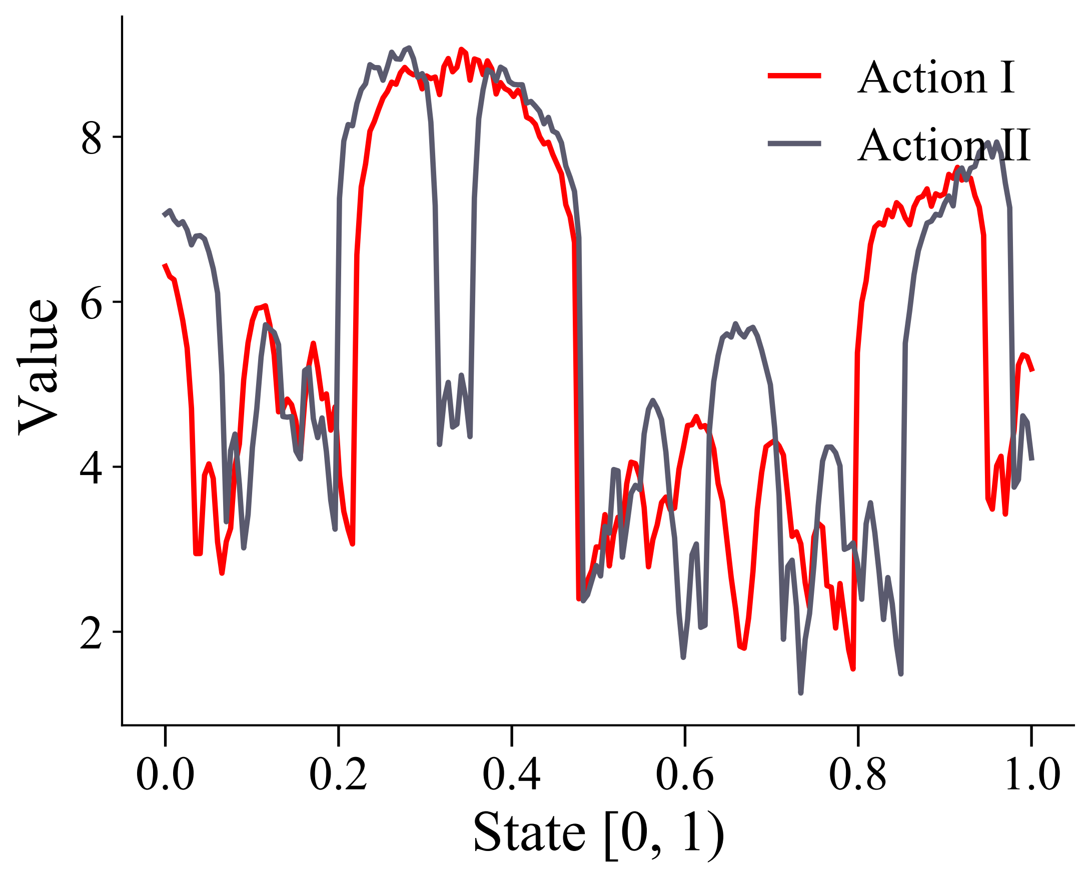
On the Toy domain,

```python
import torch
import torch.nn as nn

net = nn.Sequential(
    nn.Linear(1, 200),
    lambda x: torch.sin(x),
    nn.Linear(200, 200),
    nn.ReLU(),
    nn.Linear(200, 1),
    nn.ReLU(),
)
```

FQI + MLP



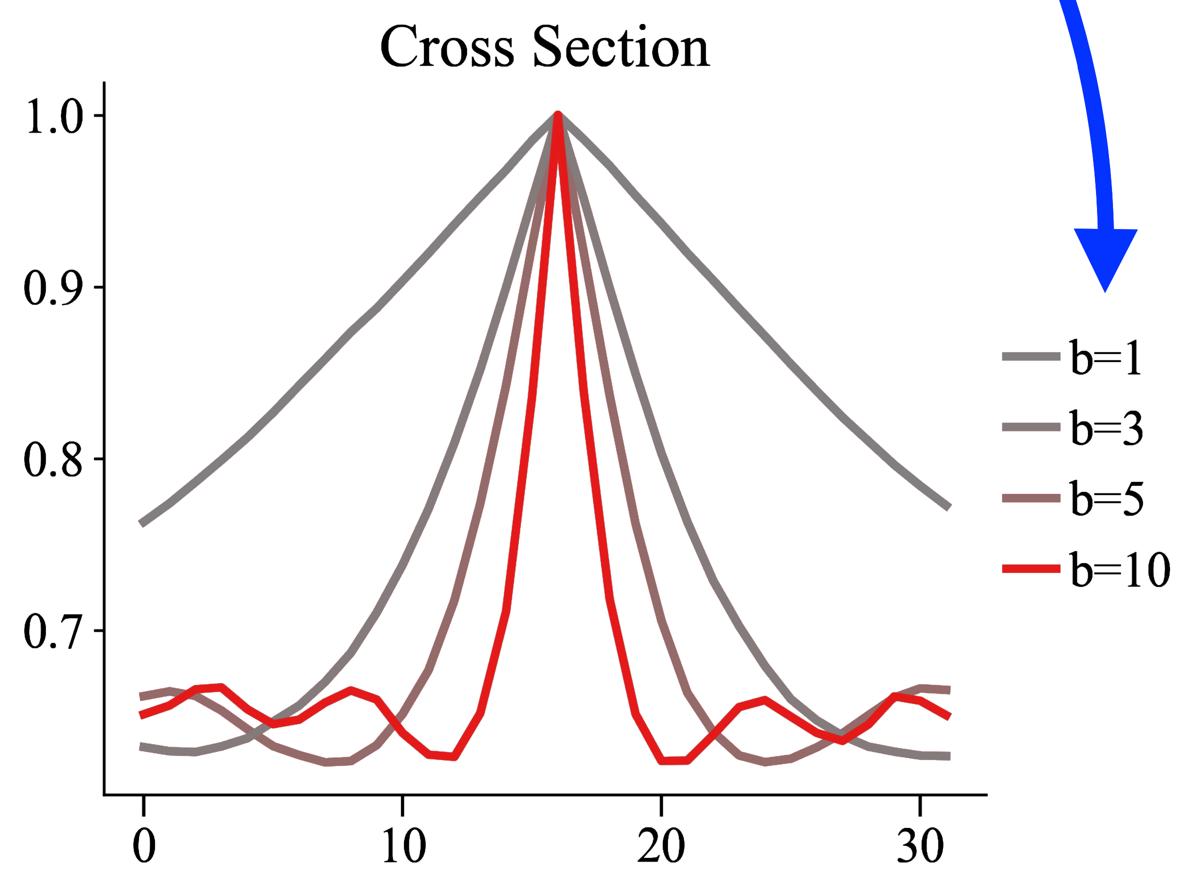
FQI + FFN

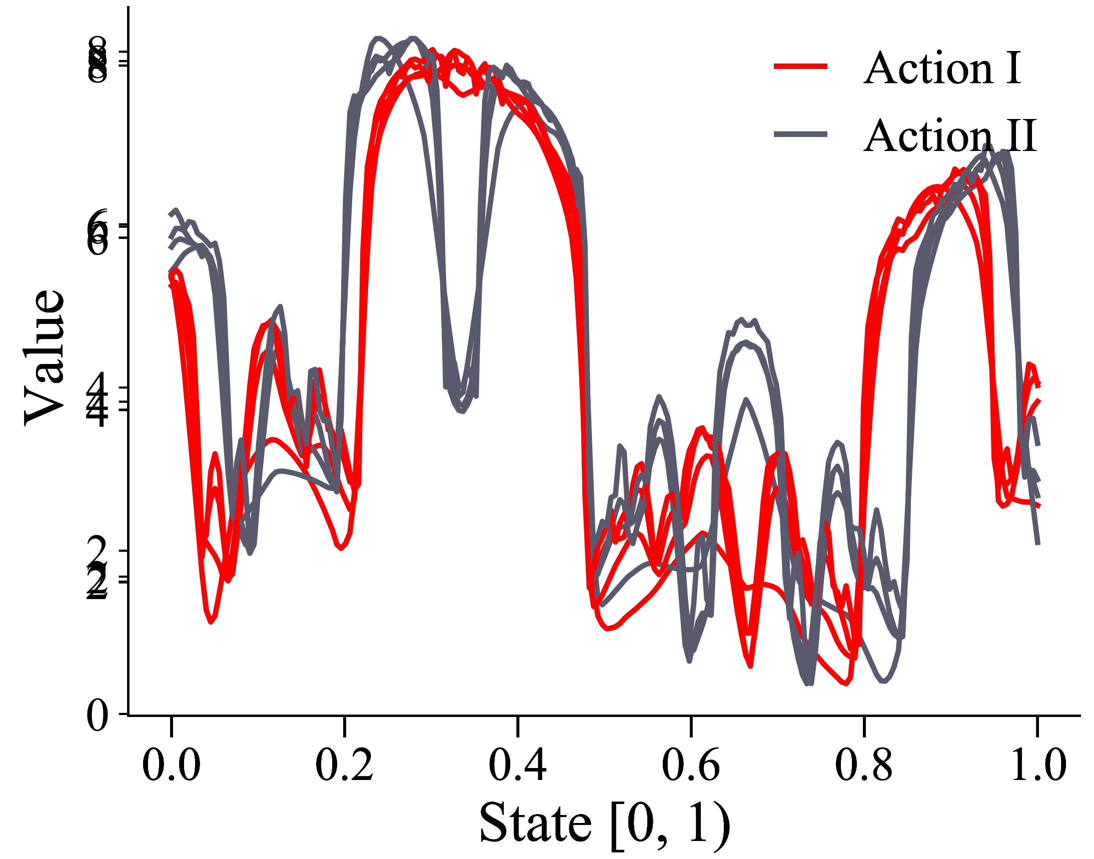


Controlled generalization via Fourier feature networks

```
import torch
import torch.nn as nn

net = nn.Sequential(
    nn.Linear(1, 200),
    lambda x: torch.sin(x),
    nn.Linear(200, 200),
    nn.ReLU(),
    nn.Linear(200, 1),
    nn.ReLU(),
)
```

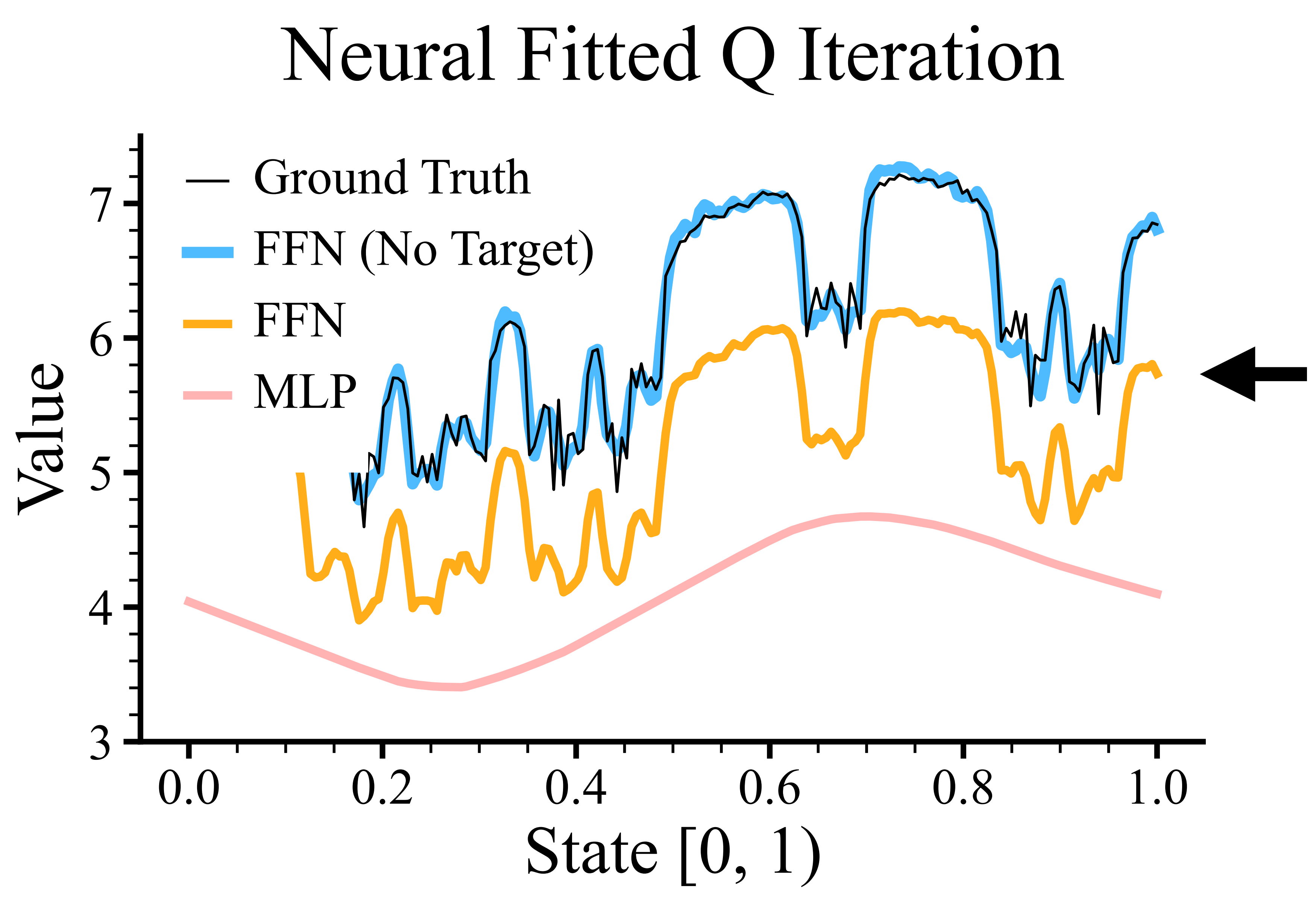
$$W_{i,j} = N\left(0, \frac{\mathbf{b}}{d_{\text{out}}}\right) \qquad b_i = U(-1,1)$$





Removing The Target Network

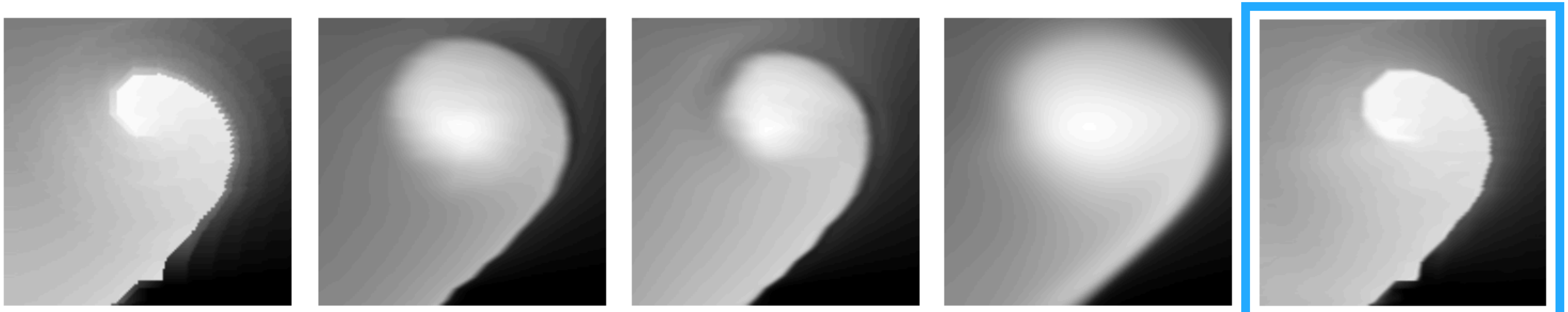
```
import torch
import torch.nn as nn

net = nn.Sequential(
    nn.Linear(1, 200),
    lambda x: torch.sin(x),
    nn.Linear(200, 200),
    nn.ReLU(),
    nn.Linear(200, 1),
    nn.ReLU(),
)
```



Mountain Car

Ground Truth

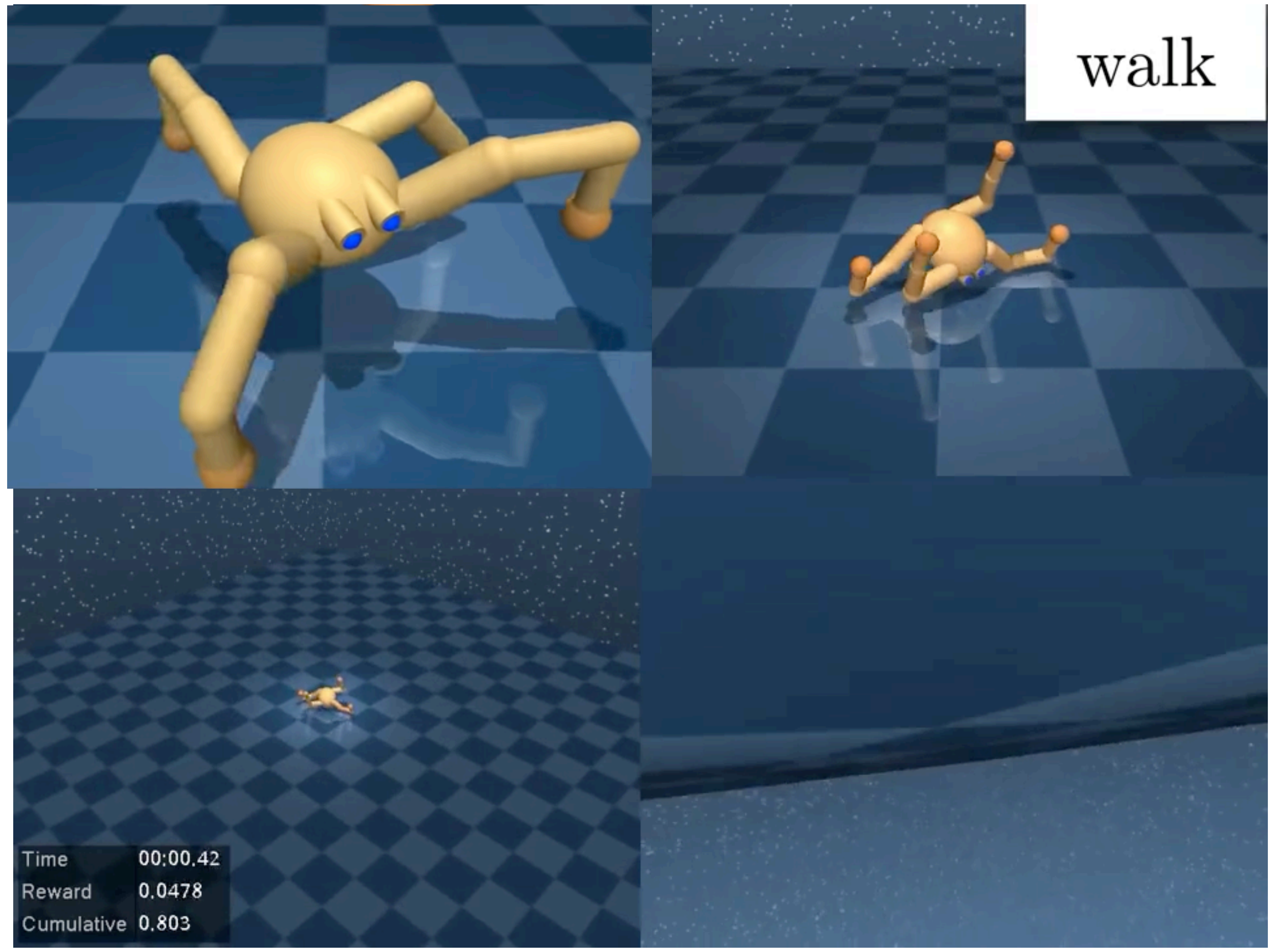
Ours



(e) Tabular (a) 4-layer MLP (b) 12-layer MLP (c) MLP + tanh (d) 4-layer RFN

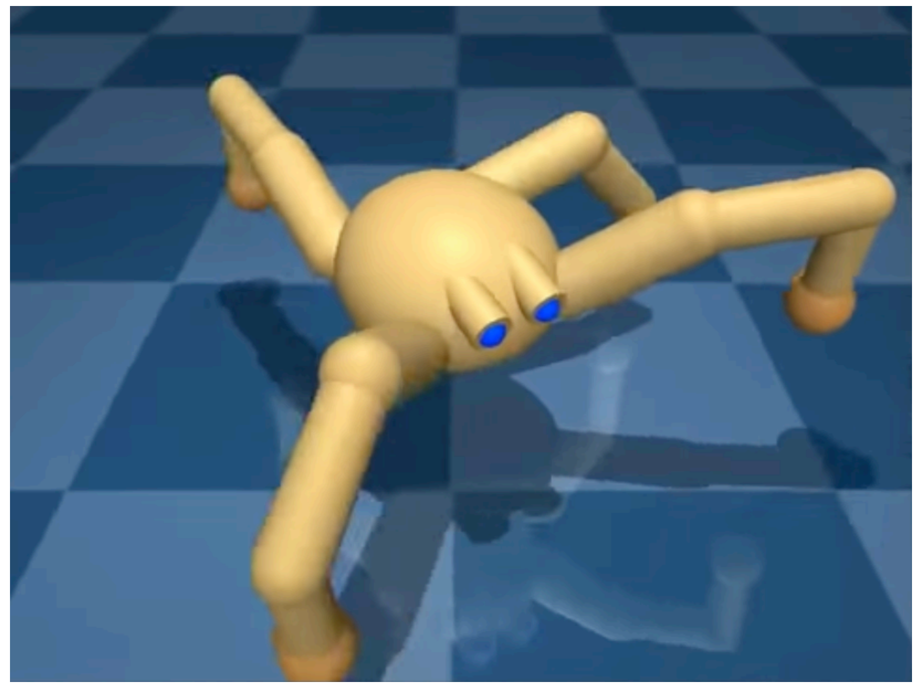
Scaling Up to Complex Continuous Control Domains

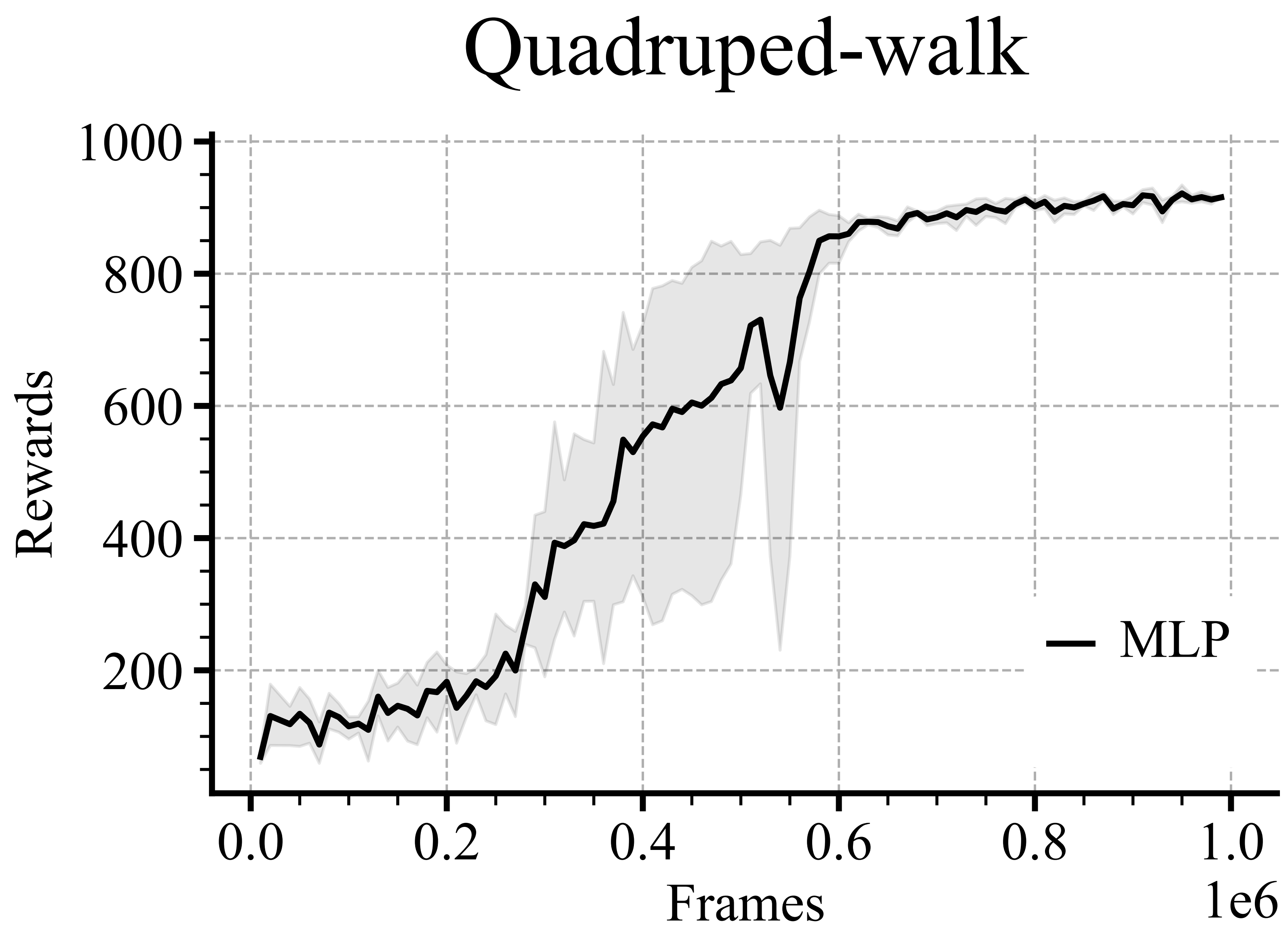
Quadruped [run, walk] from DeepMind control suite [Tassa *et al* 2018]

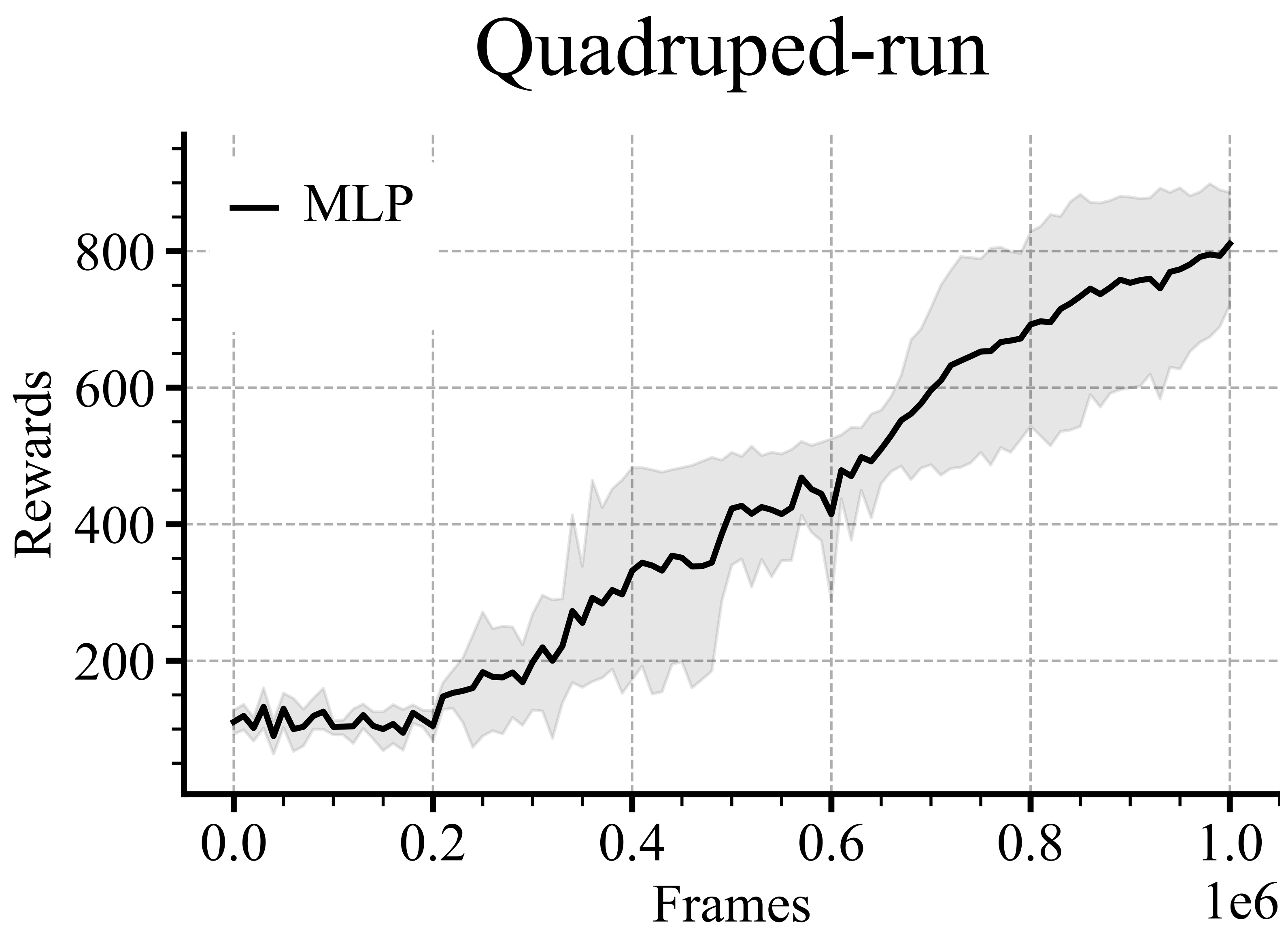


Complex Continuous Control Domains (DeepMind control suite)

Quadruped [run, walk]

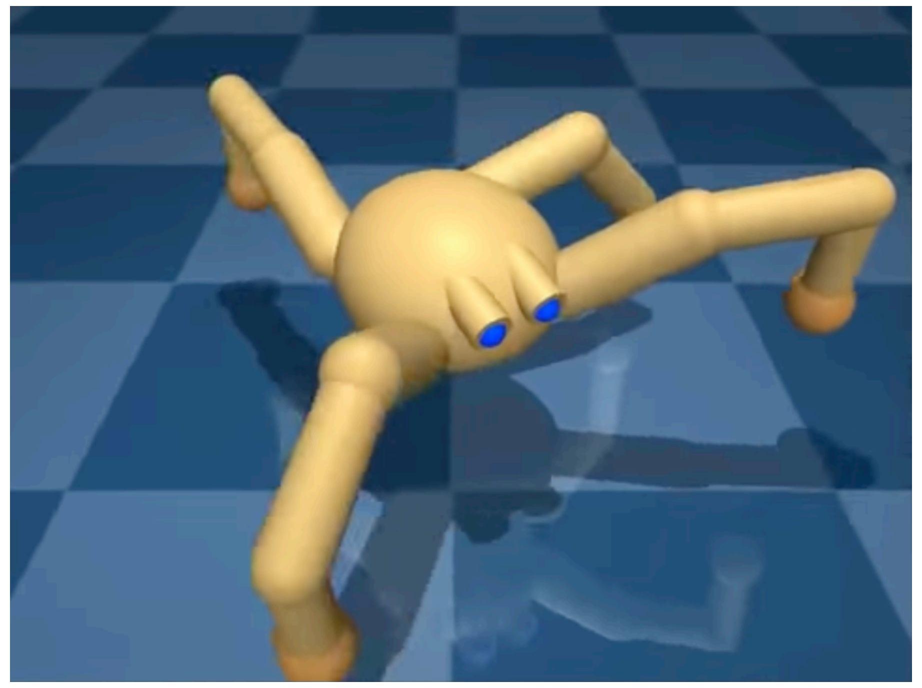


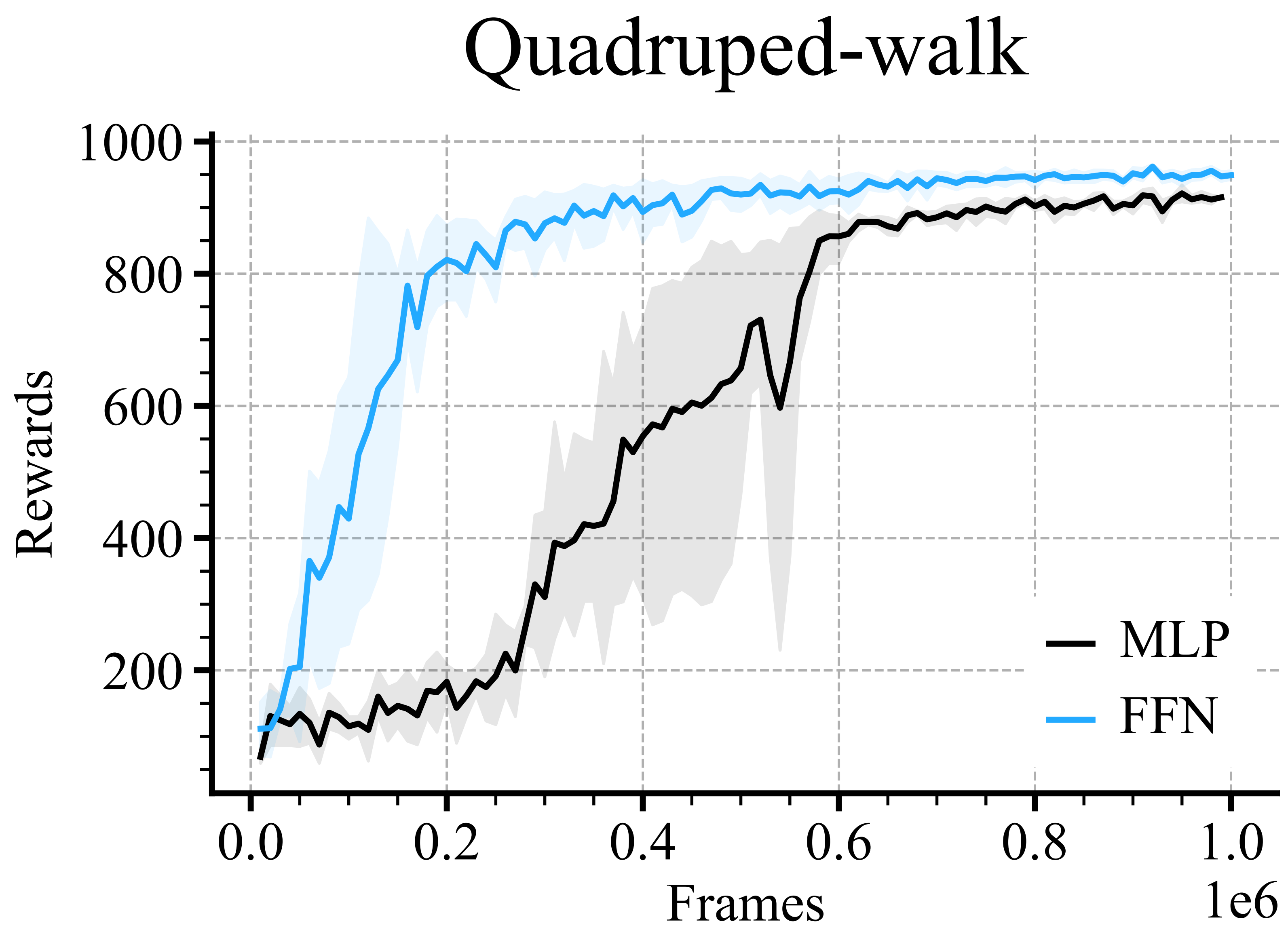


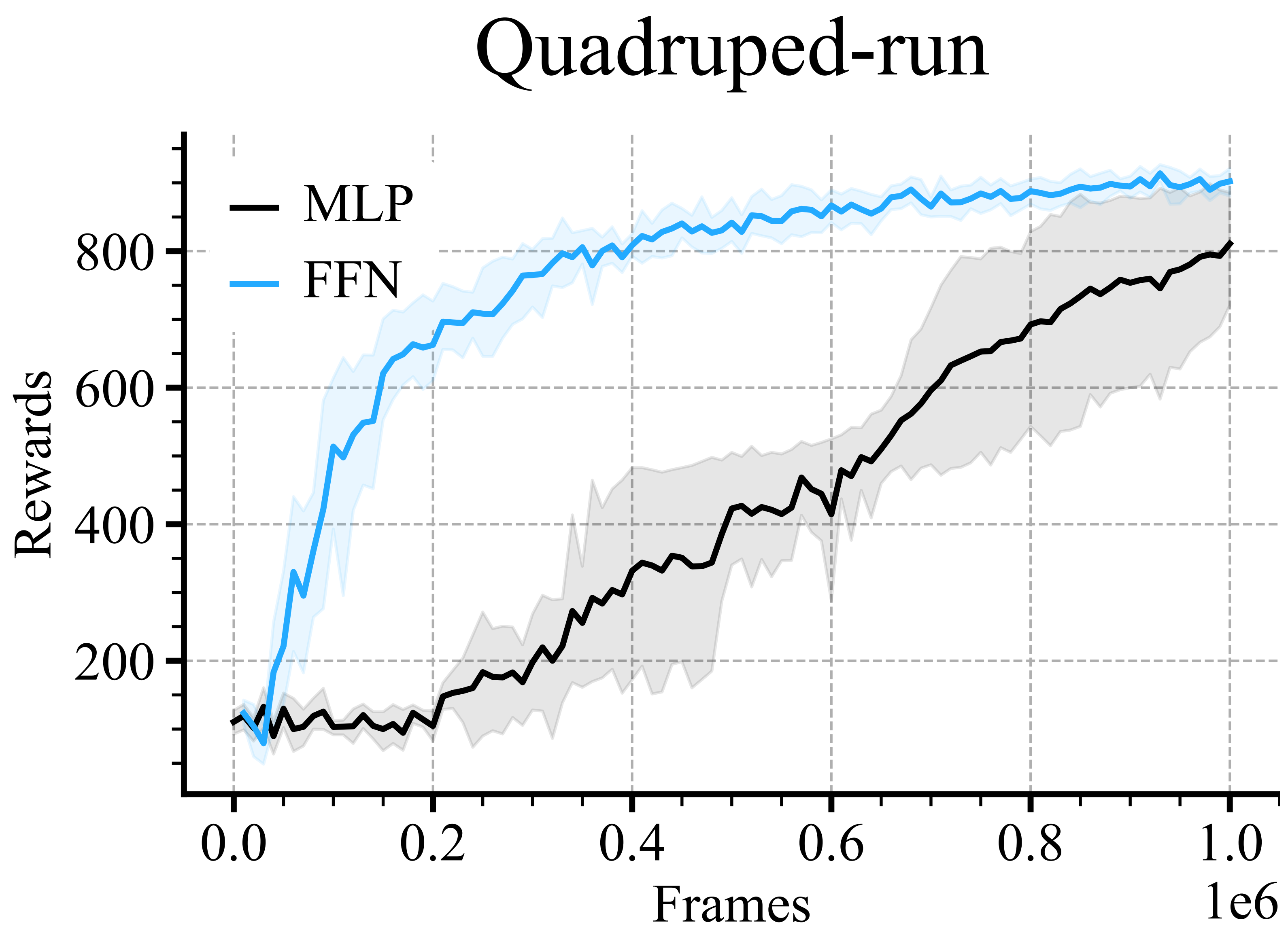


Complex Continuous Control Domains (DeepMind control suite)

Quadruped [run, walk]

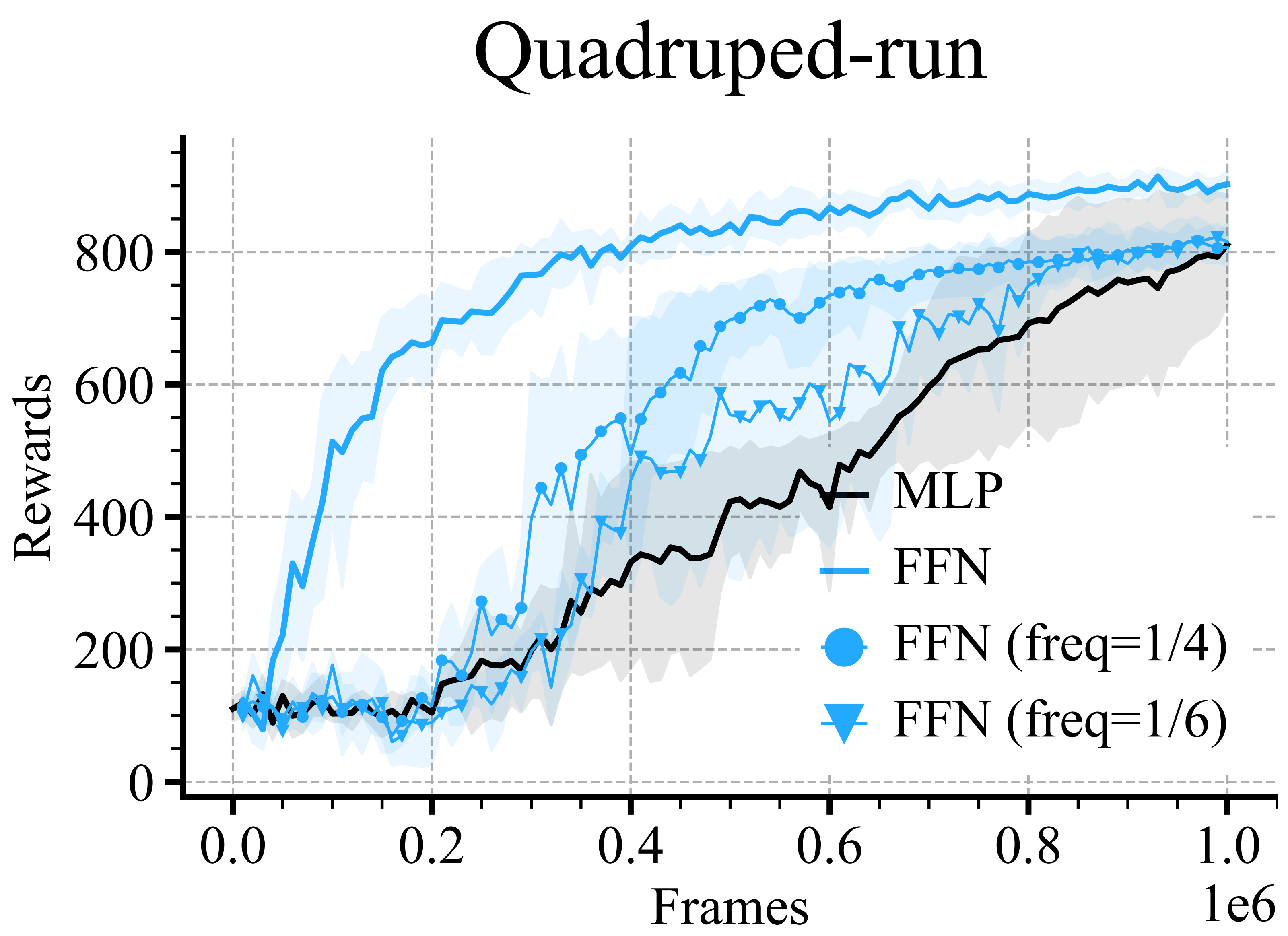


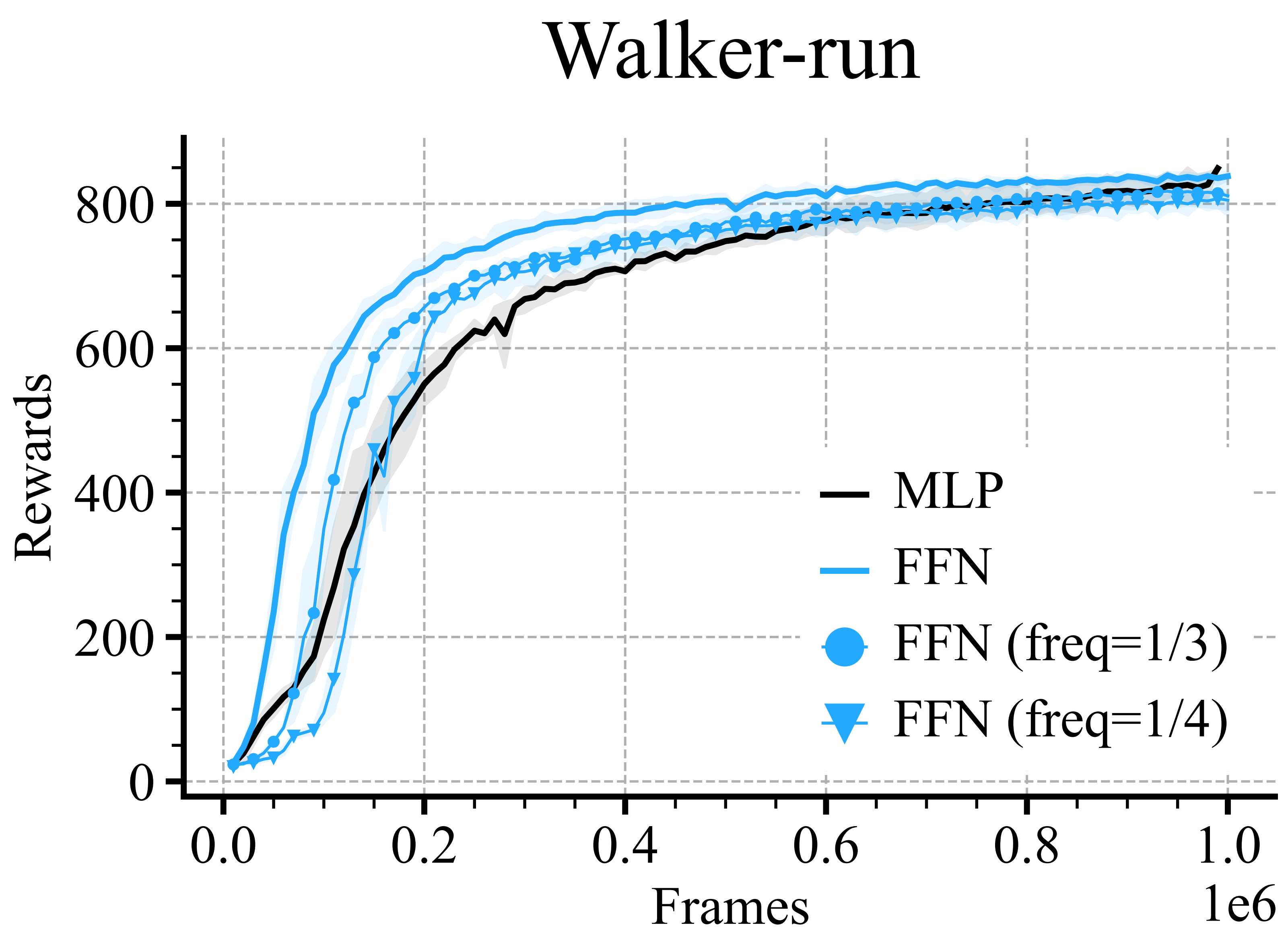




FFN is a better function approximator class

Matches *SOTA* using just **1/6** of the compute on *Quadruped* run
1/4 of the compute on *Walker* run





Summary

- Single line change overcomes the spectral bias
- Reduces off-policy divergence (no target)
- Matches *SOTA* using just **1/4** or **1/6** of the compute
- Benefit primarily comes from better critic

For more details, please visit: https://geyang.github.io/ffn

Overcoming The Spectral Bias of Neural Value estimation

For more details, please visit: https://geyang.github.io/ffn

Ge Yang*, Anurag Ajay* & Pulkit Agrawal

***Equal contribution, order determined randomly**