Fuzzy Tiling Activations: A Simple Approach to Learning Sparse Representations Online

Yangchen Pan, Kirby Banman, Martha White

Our Contributions

1. We pursue a direction which we call **natural sparsity** — achieving sparsity by design, rather than by learning
2. Specifically, we design an **activation function** that produces sparse representation deterministically by construction, and the sparsity is controllable
3. The function is essentially a special "**binning operation**", but it overcomes two limitations of binning: zero gradients and lost precision
4. We demonstrate that our activation function is very suitable in an online, nonstationary, and deep learning setting. Particularly, we empirically show that deep reinforcement learning algorithms using our activation function significantly outperform those using conventional ReLu activations in most cases.

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation: $\boldsymbol{\phi}(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

where $\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$ is the **tiling vector**, k is number of bins, δ is the bin width, l is supposed to be the lower bound of the input.

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation (TA): $\phi(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

$\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$

Example: Assume the vector **c** = (0, 0.25, 0.5, 0.75) so the bin width $\delta = 0.25$,$k = 4$,$l = 0$.

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation (TA): $\phi(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

$\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$

Example: Assume the vector **c** = (0, 0.25, 0.5, 0.75) so the bin width $\delta = 0.25$,$k = 4$,$l = 0$.

Consider input z = 0.3 (**we expect TA outputs (0, 1, 0, 0)**)

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation (TA): $\phi(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

$\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$

Example: Assume the vector **c** = (0, 0.25, 0.5, 0.75) so the bin width $\delta = 0.25, k = 4, l = 0$.

Consider input z = 0.3 (**we expect TA outputs (0, 1, 0, 0)**)

Consider the first max: max(**c** - z, 0) = max((-0.3, -0.05, 0.2, 0.45), 0) = (0, 0, 0.2, 0.45) (a)

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation (TA): $\phi(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

$\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$

Example: Assume the vector **c** = (0, 0.25, 0.5, 0.75) so the bin width $\delta = 0.25$.

Consider input z = 0.3 (**we expect TA outputs (0, 1, 0, 0)**)

Consider the first max: max(**c** - z, 0) = max((-0.3, -0.05, 0.2, 0.45), 0) = (0, 0, 0.2, 0.45) (a)

Then the second max: max(z - 0.25 - **c**) = max((0.05, 0, 0, 0), 0) = (0.05, 0, 0, 0) (b)

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation (TA): $\phi(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

$\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$

Example: Assume the vector **c** = (0, 0.25, 0.5, 0.75) so the bin width $\delta = 0.25$.

Consider input z = 0.3 (**we expect TA outputs (0, 1, 0, 0)**)

Consider the first max: max(**c** - z, 0) = max((-0.3, -0.05, 0.2, 0.45), 0) = (0, 0, 0.2, 0.45) (a)

Then the second max: max(z - 0.25 - **c**) = max((0.05, 0, 0, 0), 0) = (0.05, 0, 0, 0) (b)

Then (a) + (b) = (0.05, 0, 0.2, 0.45) (c)

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation (TA): $\phi(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

$\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$

Example: Assume the vector **c** = (0, 0.25, 0.5, 0.75) so the bin width $\delta = 0.25$.

Consider input z = 0.3 (**we expect TA outputs (0, 1, 0, 0)**)

Consider the first max: max(**c** - z, 0) = max((-0.3, -0.05, 0.2, 0.45), 0) = (0, 0, 0.2, 0.45) (a)

Then the second max: max(z - 0.25 - **c**) = max((0.05, 0, 0, 0), 0) = (0.05, 0, 0, 0) (b)

Then (a) + (b) = (0.05, 0, 0.2, 0.45) (c)

Then I_+((a)+(b)) = (1, 0, 1, 1) (d)

~~Fuzzy~~ Tiling Activation (TA) Function

Tiling Activation (TA): $\phi(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

$\mathbf{c} := (l, l + \delta, l + 2\delta, \ldots, l + (k - 1)\delta)$

Example: Assume the vector **c** = (0, 0.25, 0.5, 0.75) so the bin width $\delta = 0.25$.

Consider input z = 0.3 (**we expect TA outputs (0, 1, 0, 0)**)

Consider the first max: max(**c** - z, 0) = max((-0.3, -0.05, 0.2, 0.45), 0) = (0, 0, 0.2, 0.45) (a)

Then the second max: max(z - 0.25 - **c**) = max((0.05, 0, 0, 0), 0) = (0.05, 0, 0, 0) (b)

Then (a) + (b) = (0.05, 0, 0.2, 0.45) (c)

Then I_+((a)+(b)) = (1, 0, 1, 1) (d)

Then 1 - I_+((a)+(b)) = 1 - (1, 0, 1, 1) = (0, 1, 0, 0) as we expected.

Fuzzy Tiling Activation (FTA) Function

Tiling Activation (TA): $\boldsymbol{\phi}(z) \stackrel{\text{def}}{=} 1 - I_+(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$

Problems: the gradient is zero; lose precision, potentially hurts generalization

Solution: define a fuzzy indicator function (it returns small value when the input is smaller than η)

$$I_{\eta,+}(x) \stackrel{\text{def}}{=} I_+(\eta - x)x + I_+(x - \eta)$$

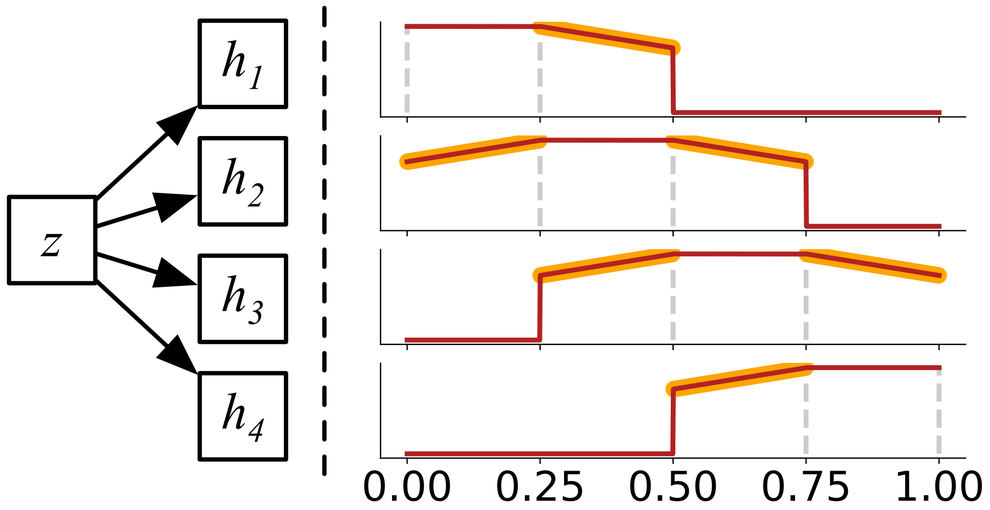
Then we get Fuzzy Tiling Activation (FTA):

$$\boldsymbol{\phi}_\eta(z) \stackrel{\text{def}}{=} 1 - I_{\eta,+}(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$$

Fuzzy Tiling Activation (FTA) Function

FTA:

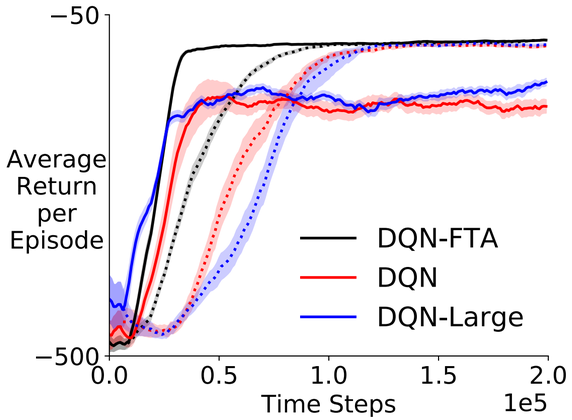
$$\phi_\eta(z) \stackrel{\text{def}}{=} 1 - I_{\eta,+}(\max(\mathbf{c} - z, 0) + \max(z - \delta - \mathbf{c}, 0))$$



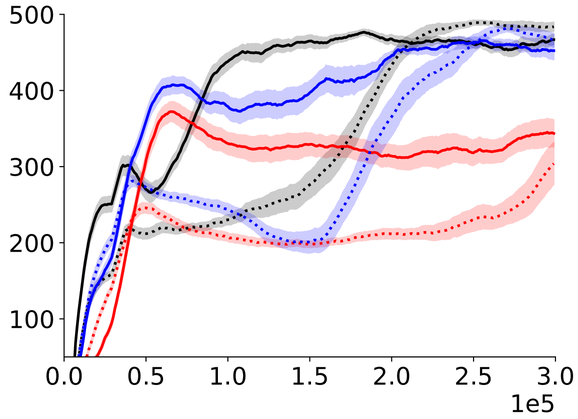
Visualization of FTA, $\mathbf{c}$ = (0, 0.25, 0.5, 0.75)

Empirical Demonstrations - RL Experiments

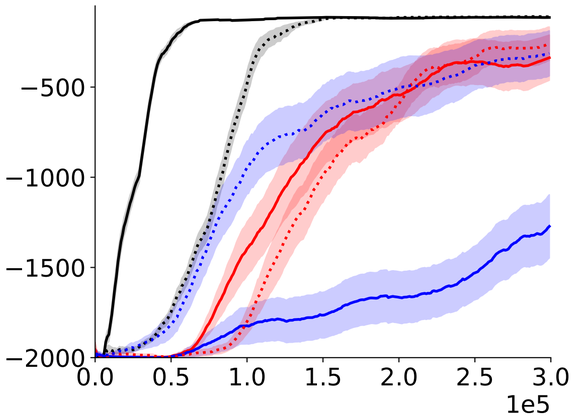
Acrobot



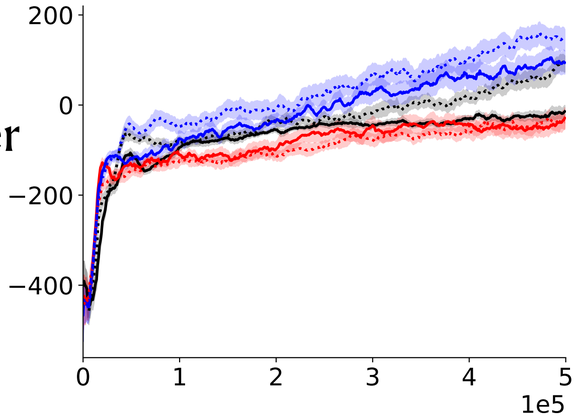
CartPole



MountainCar



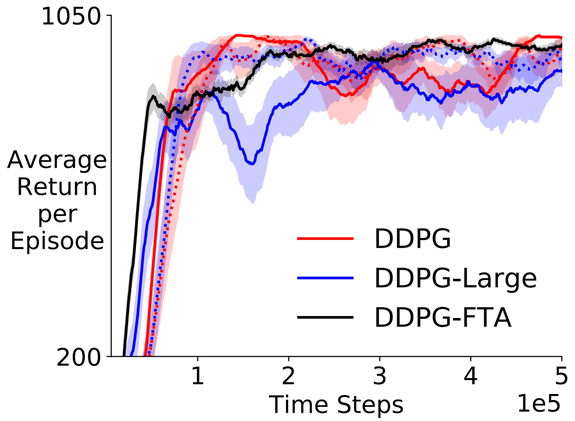
LunarLander



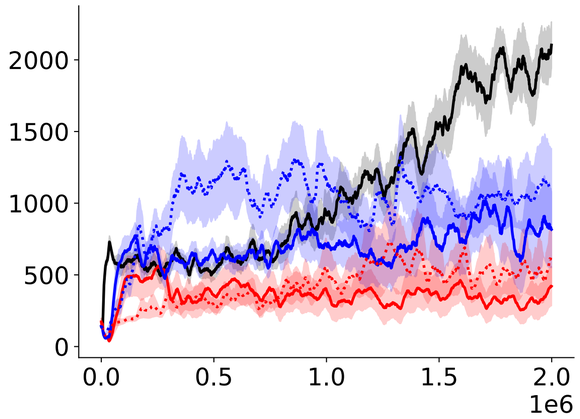
The label **DQN** is the conventional one using ReLu; **DQN-Large** uses a large neural network to match the output dimension of **DQN-FTA.** We apply FTA to the second hidden layer.

The solid lines are those without using target networks.

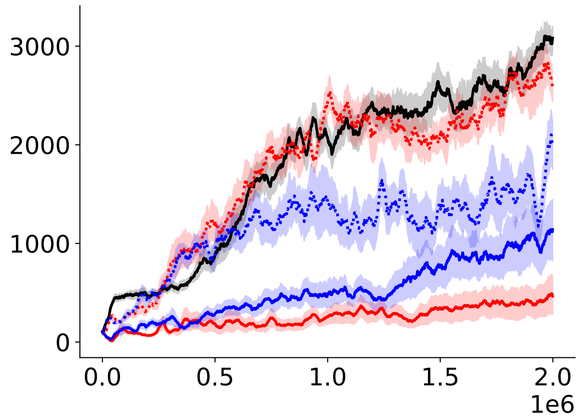
Empirical Demonstrations - RL Experiments



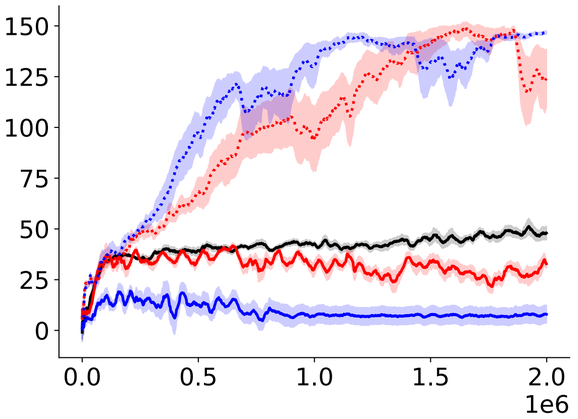
InvertedPendulum



Hopper



Walker



Swimmer

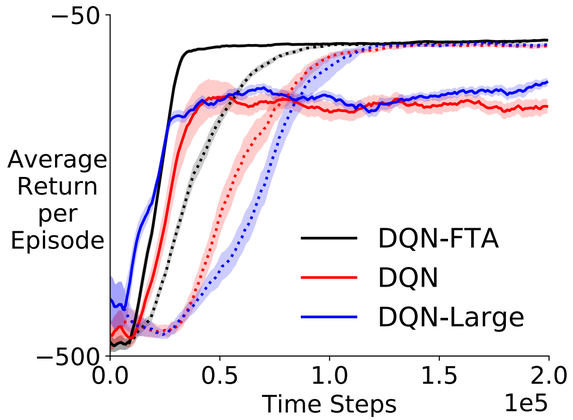
This should be the first general method to allow training without using a target network on so many domains.

We fix the same FTA setting across all experiments:

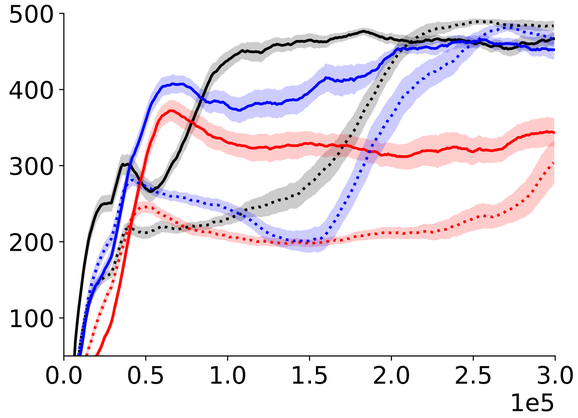
the tiling vector is c = (-20, -18, ..., 16, 18), k = 20, bin width = 2.0

Empirical Demonstrations - RL Experiments

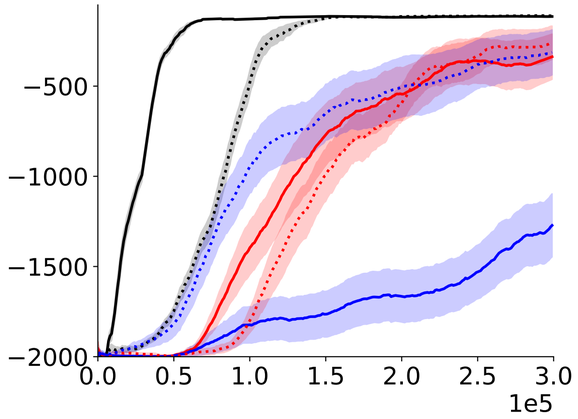
Acrobot



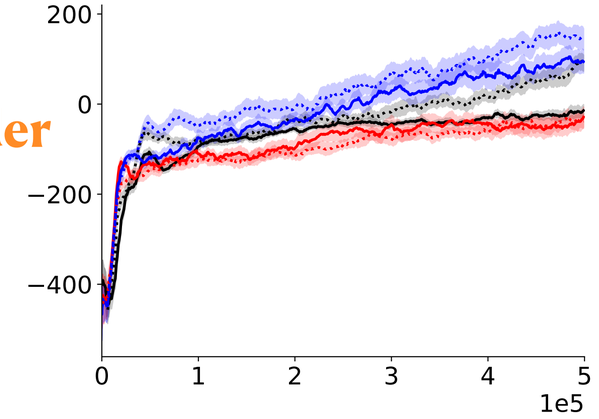
CartPole

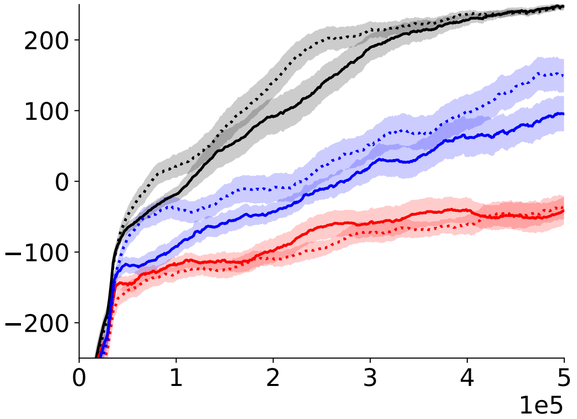


MountainCar



LunarLander





Recall that the tiling vector is c = (-20, -18, ..., 16, 18), k = 20, the lower bound of the tiling vector is -20, hence bin width = 40/2 = 2.0

DQN-FTA seems not working well on LunarLander? — see the rightmost figure, c=(-1, -0.9, ..., 0.8, 0.9)

See Section 5.4 for a detailed analysis.

Thank you!

The code link: https://github.com/yannickycpan/reproduceRL

The link is also in the footnote on the first page of our paper.