Deep Reinforcement Learning and Ray Tracing

Rendering Equation

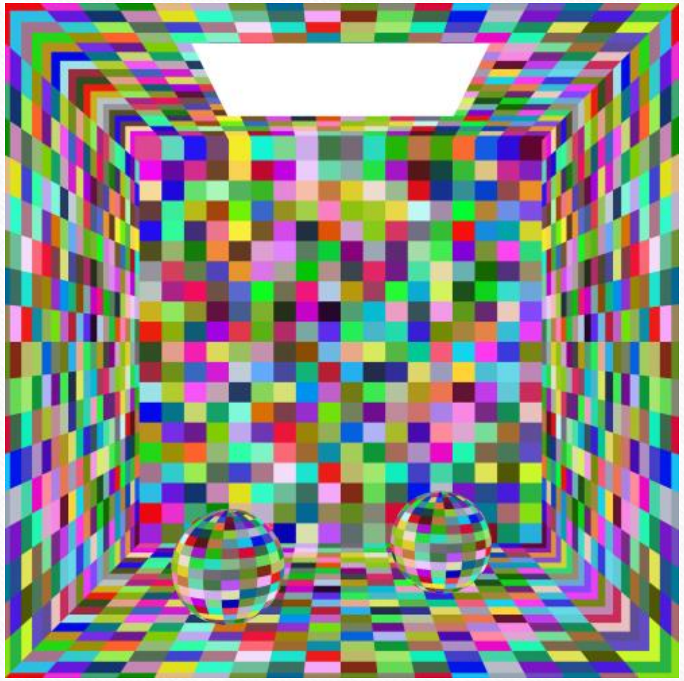
- $L(x, \omega_0) = L_e(x, \omega_0) + \int_{H^+(x)} L_i(y, -\omega_i) f_r(x, \omega_i \rightarrow \omega_0) cos\theta_i d\omega_i$
- $y = hitpoint(x, -\omega_i)$

Reinforcement Learning

- States: points on surfaces
- Actions: directions
- Q-learning :

  Discretize states and actions
- Deep Q-learning :

  Only discretize actions



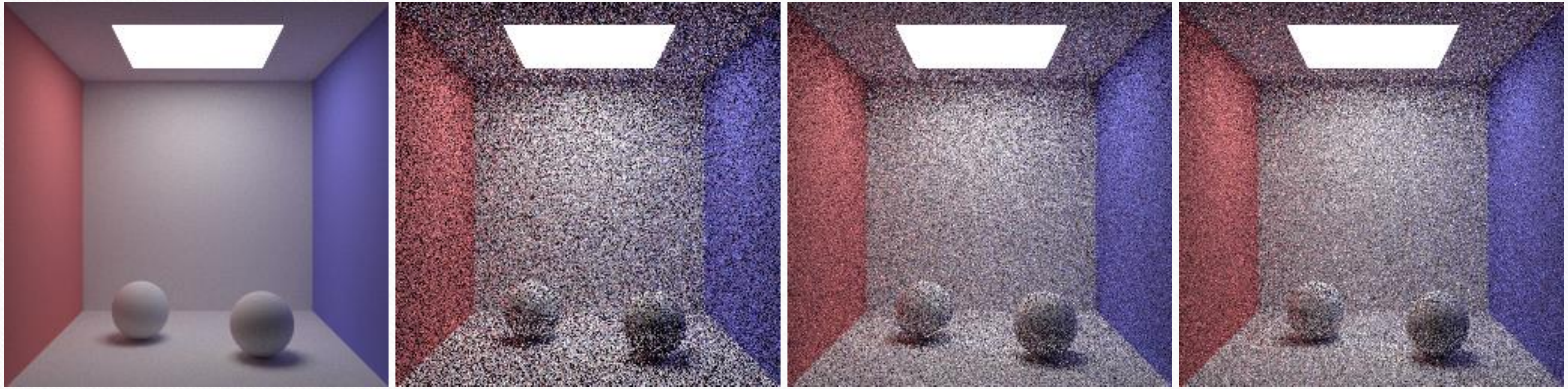
Deep Reinforcement Learning

- $Q(s,a) \sim L(x,\omega)$
- N patches to sample directions
- $target(s') \cong \frac{2\pi}{n} \sum_{i=1}^{N} Q\big(s', a'(\xi_i)\big) f_r cos\theta(\xi_i)$
- $Q(s,a,w)$ is the output of a neural network called Q-network
- Loss function : $E_{(s,a,s',r)\in U(D)}\big(Q(s,a,w) \; - \; target(s',w^-)\big)^2$
- Soft update of target parameters : $w^- = 0.999 \times w^- + 0.001 \times w$

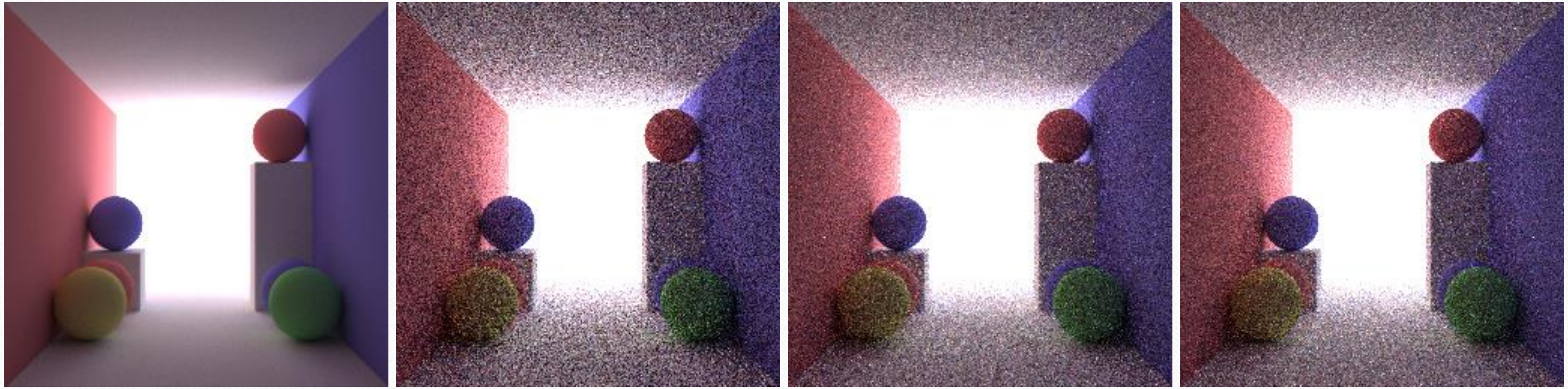
MLP Inputs

- Two different types of inputs
- $input1 = (x, n)$
- $input2 = (x, \gamma(x), n)$
- n : normal of point
- γ : positional encoding
- $\gamma(x) = (\sin(2^0 \pi x), \cos(2^0 \pi x), \ldots, \sin(2^{L-1} \pi x), \cos(2^{L-1} \pi x))$

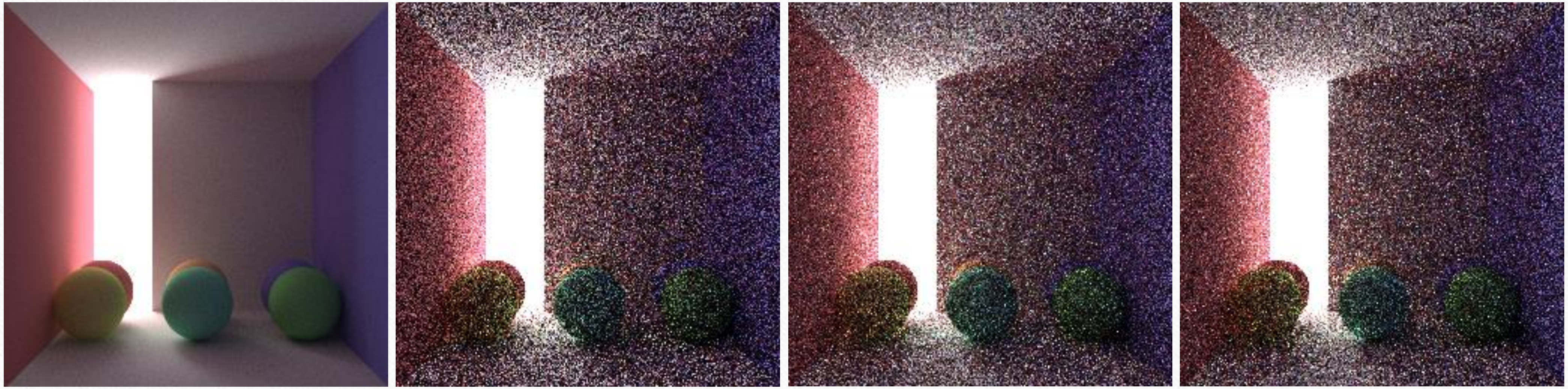
Scene 1: Uniform – DQN – DQN-POS



Scene 1: Uniform – DQN – DQN-POS

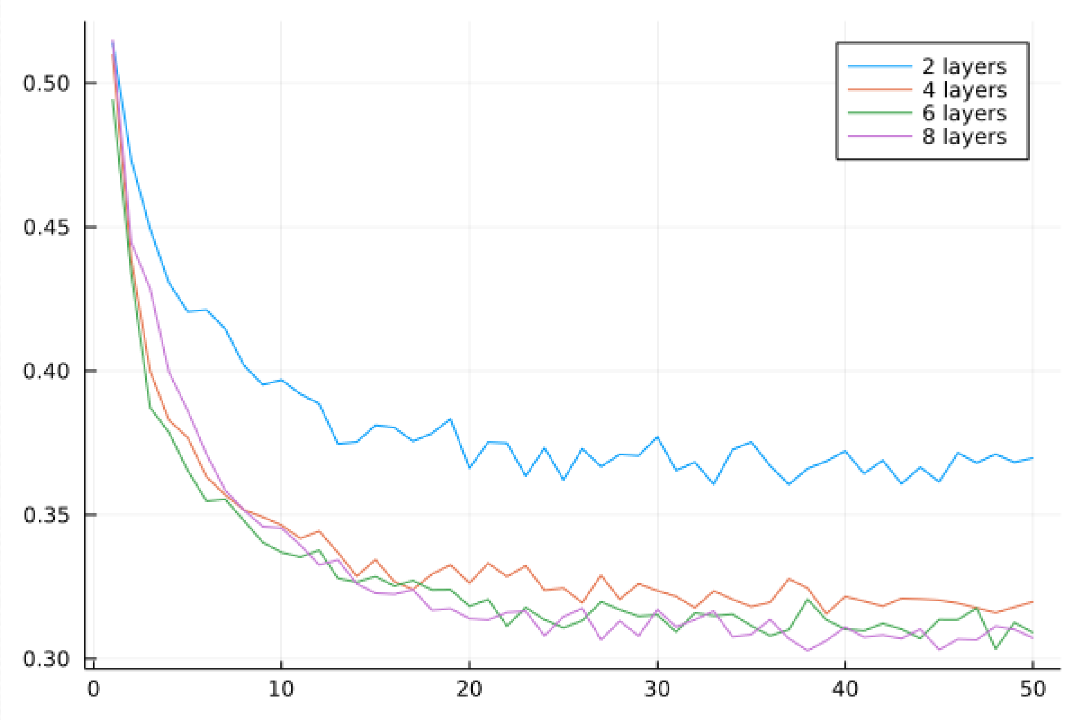


Scene 1: Uniform – DQN – DQN-POS



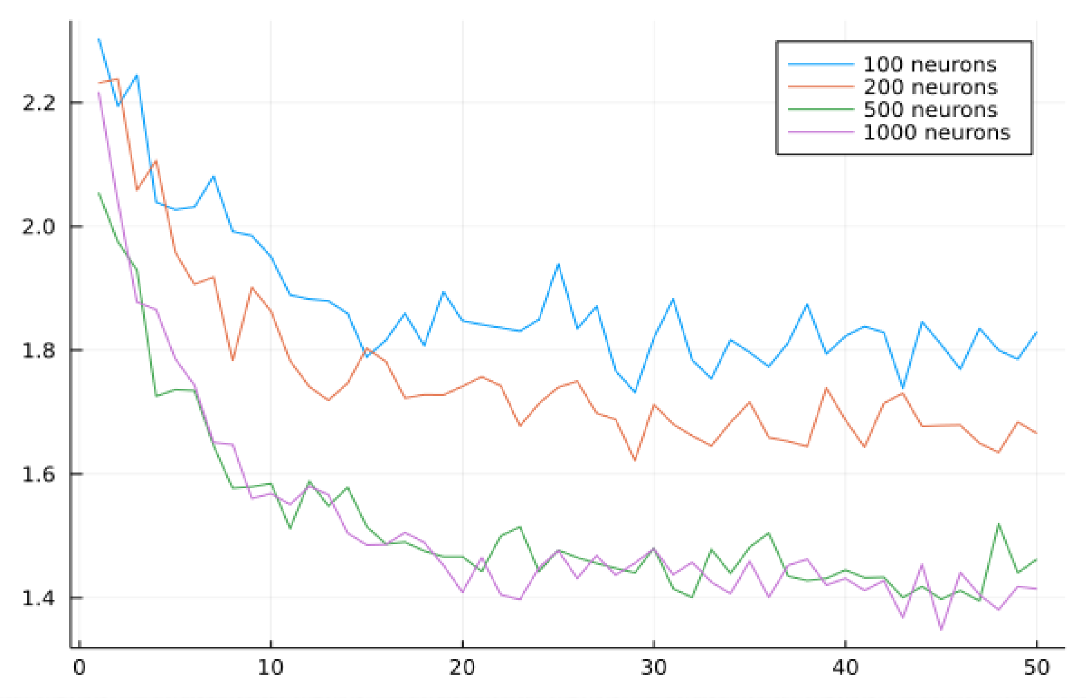
Loss Function : Depth

- Number of layers: {2,4,6,8}
- {Dense(54, H, relu), …,
  Dense(H, 108, softplus)}
- Each layers has 100 neurons



Loss Function : Width

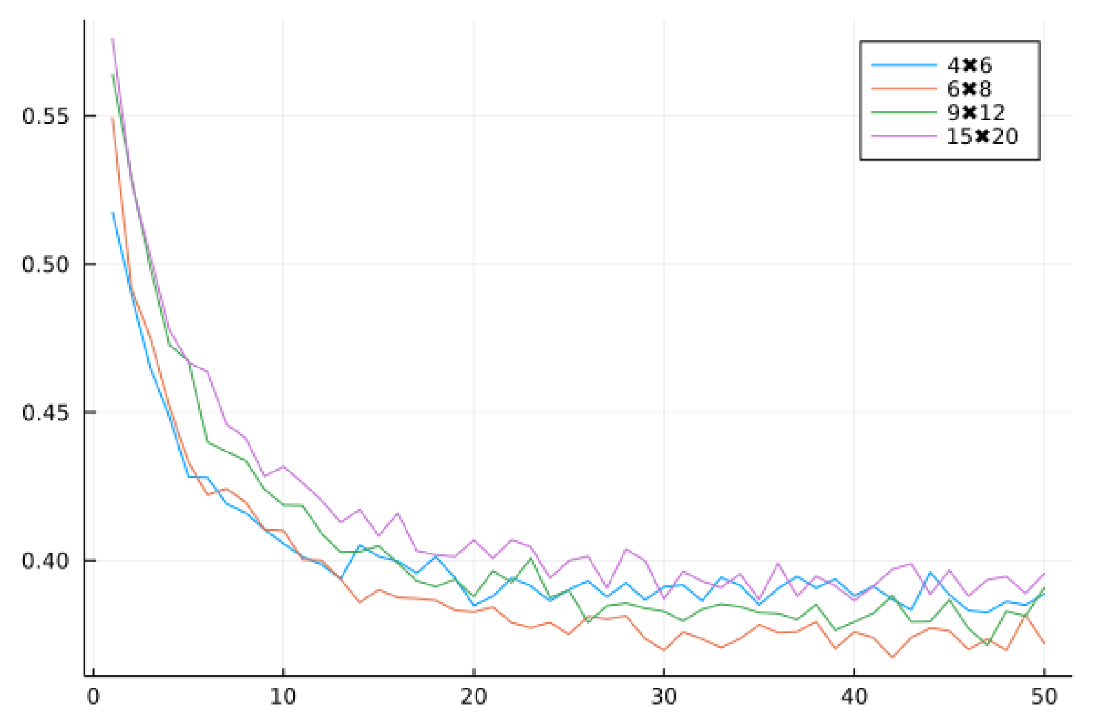
- Two-layer neural networks:
- {Dense(54, H, relu),
  Dense(H, 108, softplus)}
- H values plot



Loss Function : Discretization Resolution

- Action space resolutions

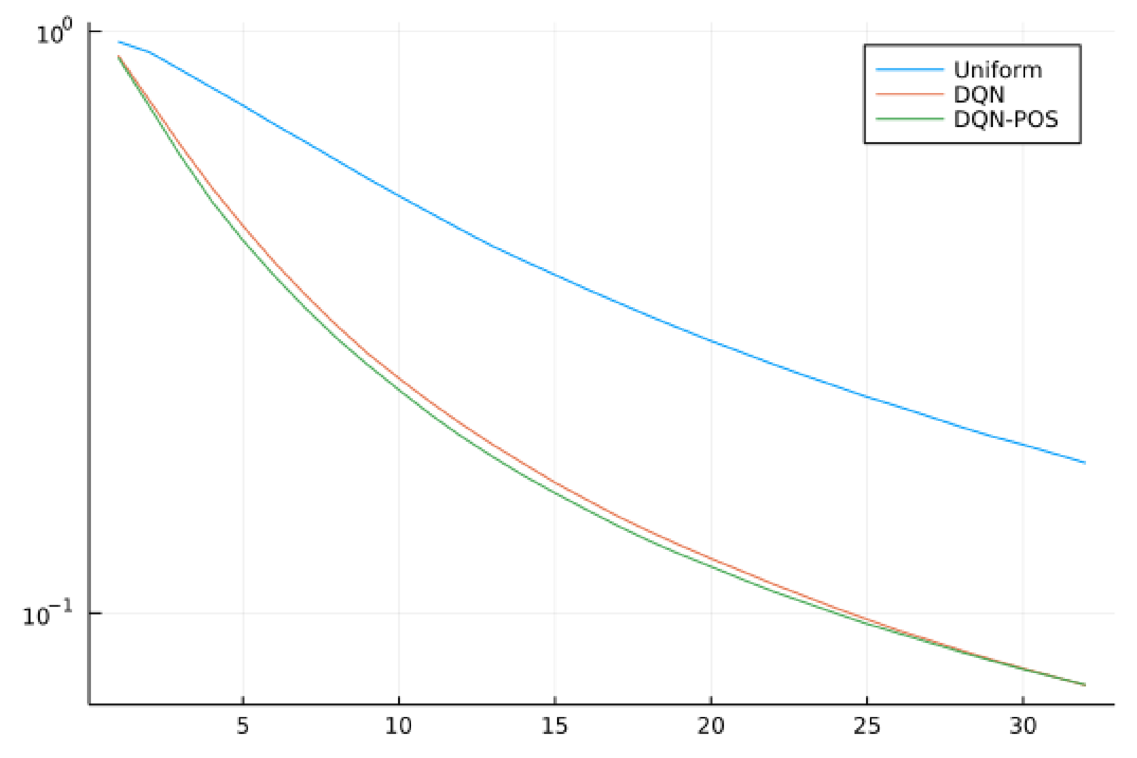
  $\{4\times6, 6\times8, 9\times12, 15\times20\}$

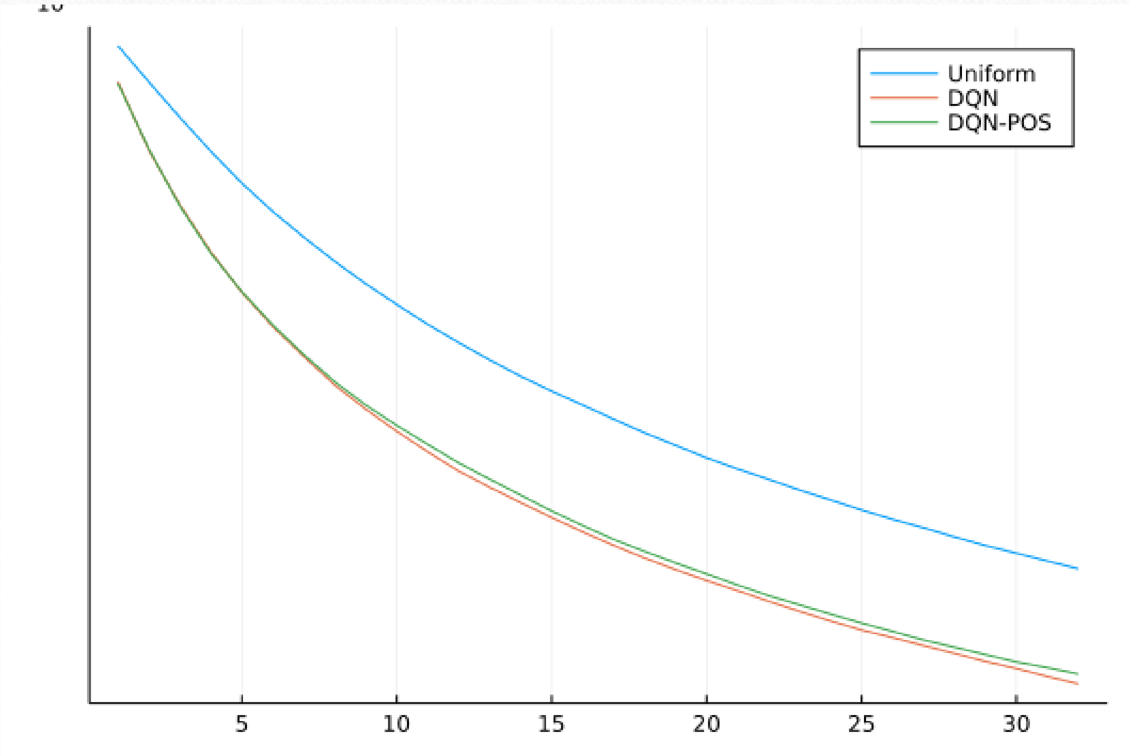


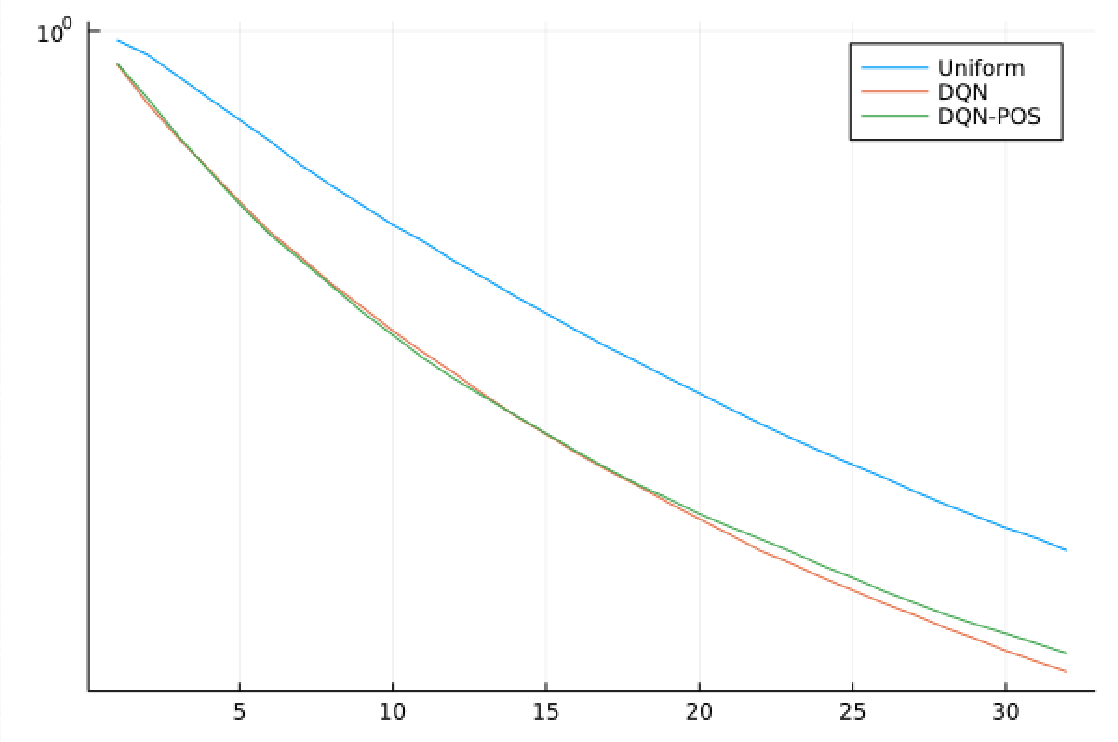
Network

- Two-layer neural networks
- {Dense(in, 1000, relu),
  Dense(1000, 108, softplus)}
- In = {6, 54}

MAPE




Uniform
DQN
DQN-POS
5
10
15
20
25
30



10^0
Uniform
DQN
DQN-POS
5
10
15
20
25
30

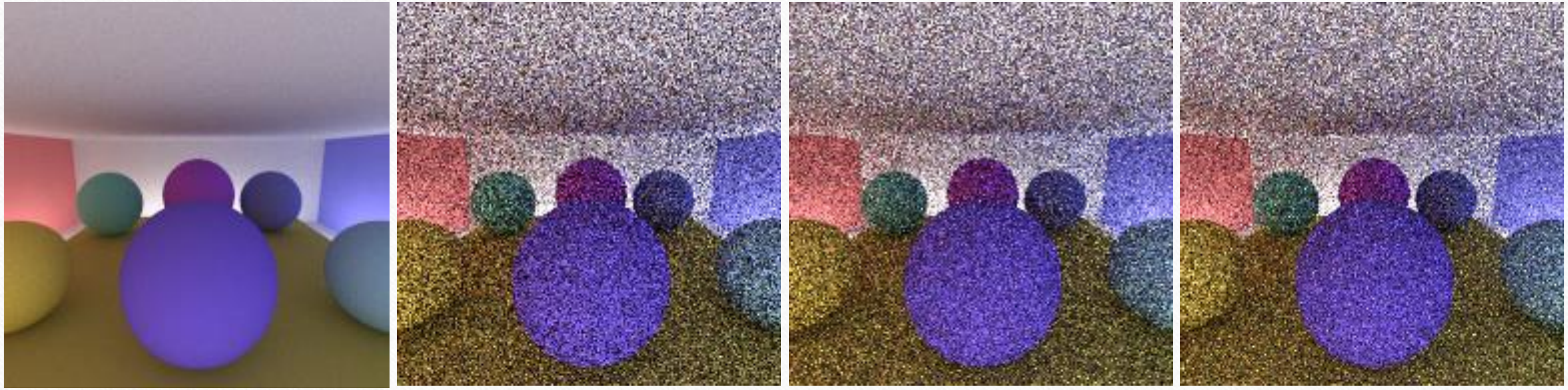
Deep Convergent Q-Learning

- C-DQN algorithm: (T. Wang)
- $L_{C-DQN}(\theta, \bar{\theta}) = \mathbb{E}\left[\max\{l_{DQN}(\theta, \bar{\theta}), l_{MSBE}(\theta)\}\right]$
- $L_{MSBE}(\theta) = L_{DQN}(\theta, \theta)$

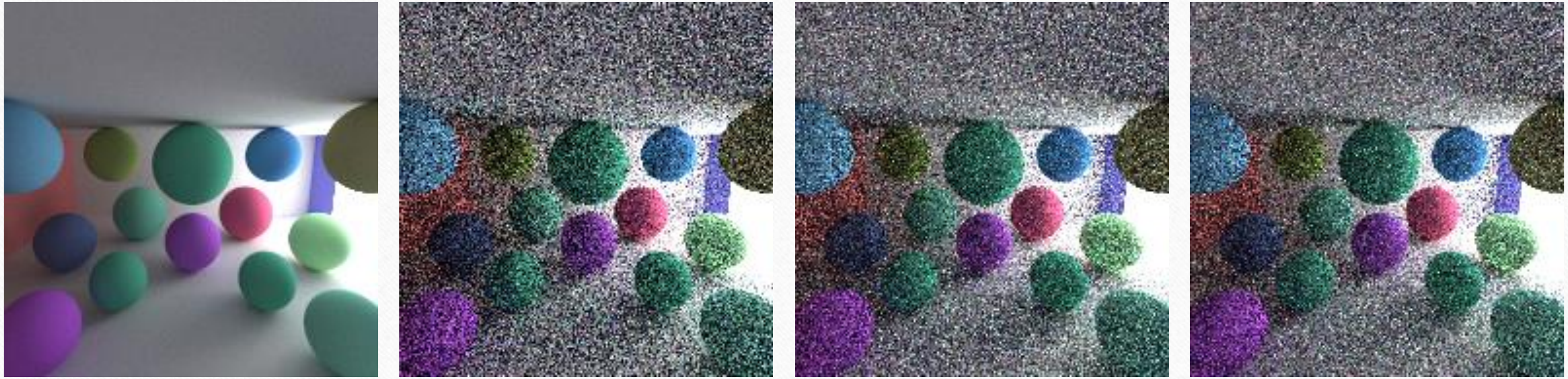
Deep RBF Network

- $RBF_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{c}) = NN_{\boldsymbol{\theta}}\left(\phi_{\boldsymbol{c}}(\boldsymbol{x})\right)$
- $\phi_{\boldsymbol{c}}(\boldsymbol{x}) = \exp(-0.5 \parallel \boldsymbol{x} - \boldsymbol{c} \parallel^2)$
- Divide space into 6*3*6 cubes and consider their centers.
- MLP = {Chain(Dense(6*3*6, 64, relu), Dense(64, 64, relu), Dense(36, 24)}

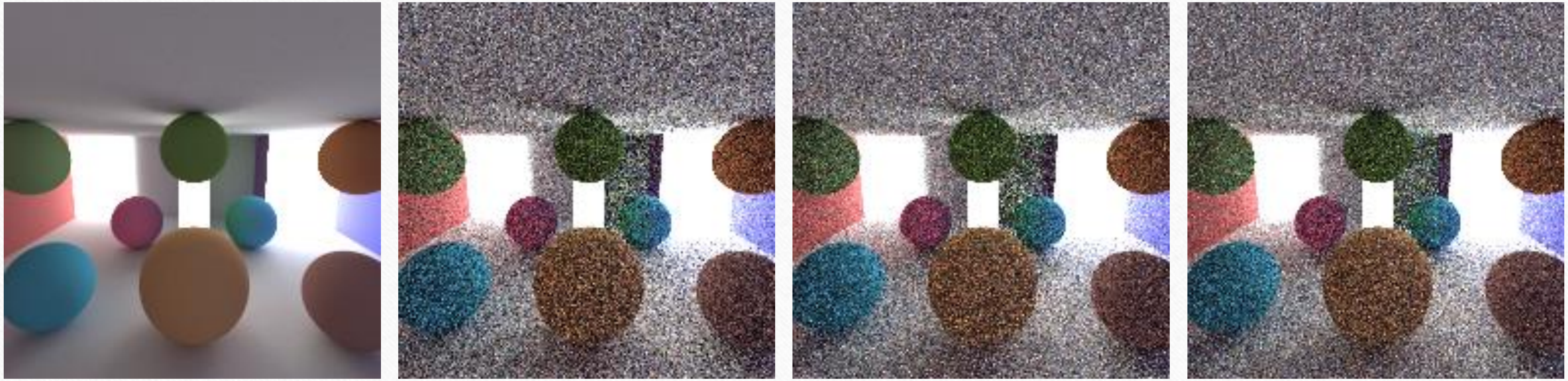
Results: Uniform , C-DQN , C-DQN-RBF



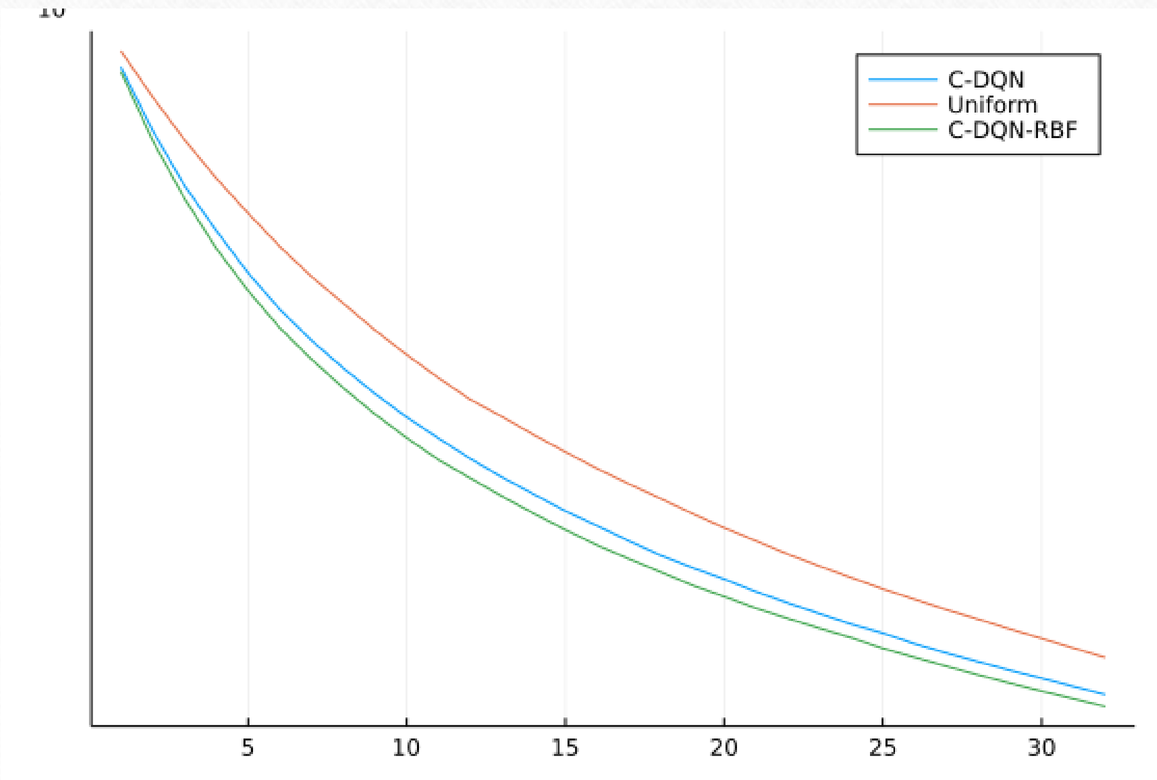
Results: Uniform , C-DQN , C-DQN-RBF

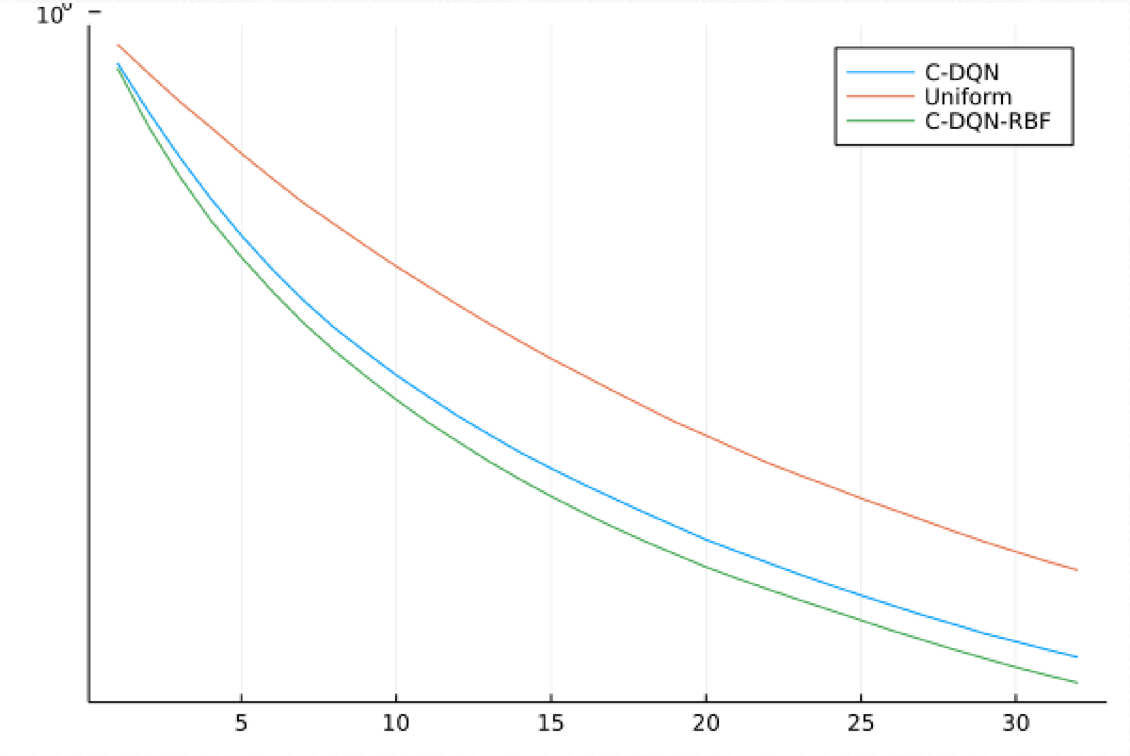


Results: Uniform , C-DQN , C-DQN-RBF

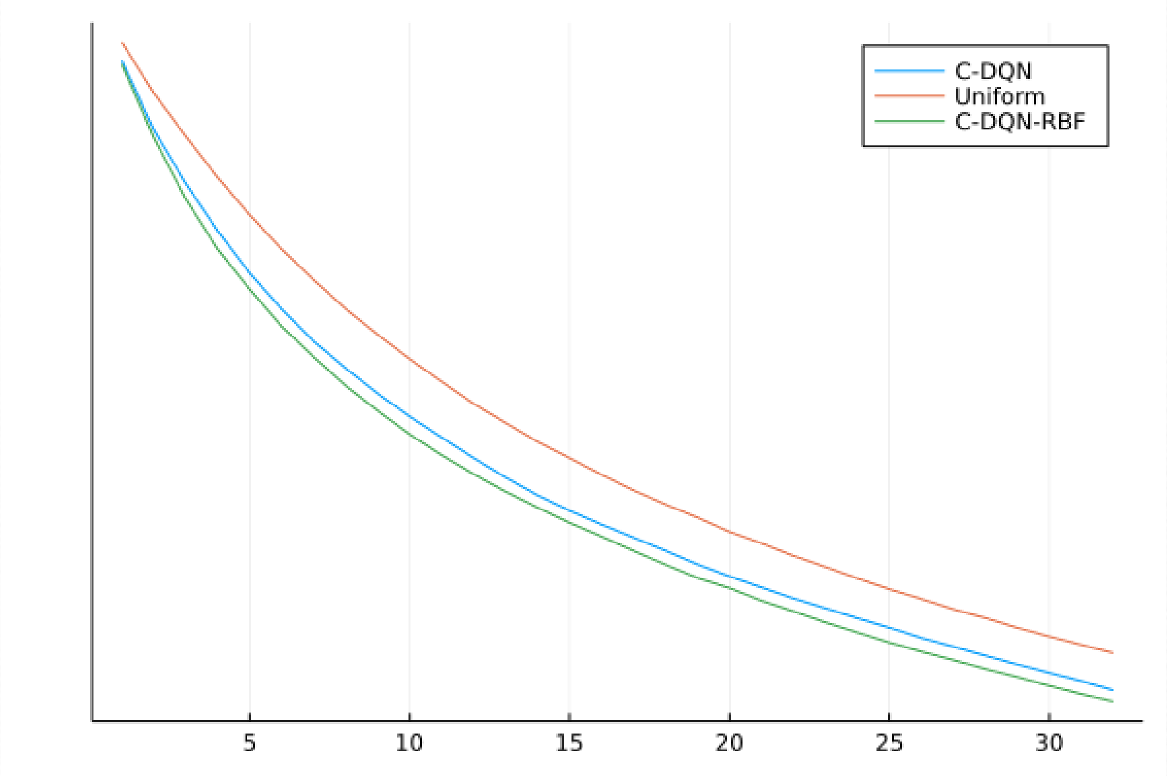


MAPE





10^0
C-DQN
Uniform
C-DQN-RBF
5
10
15
20
25
30



C-DQN
Uniform
C-DQN-RBF
5
10
15
20
25
30

Trainable Quadratic B-spline Encoder

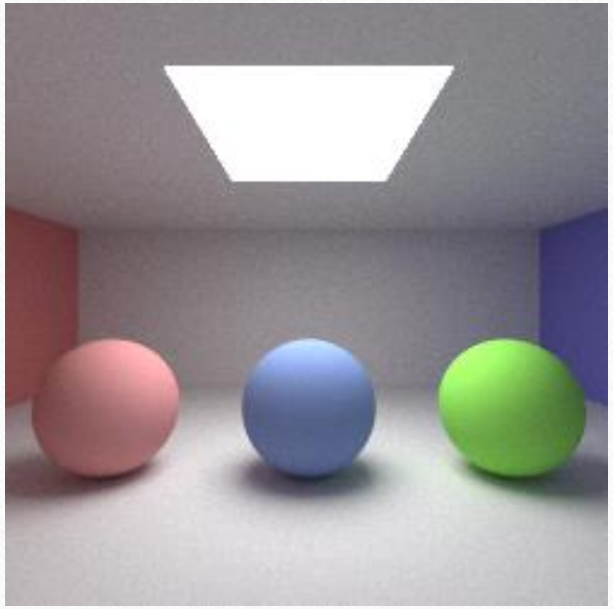
- $B_2(x) = \begin{cases} \frac{x^2}{2} & 0 \le x < 1 \\ \frac{-2x^2+6x-3}{2} & 1 \le x < 2 \\ \frac{(3-x)^2}{2} & 2 \le x < 3 \end{cases}$

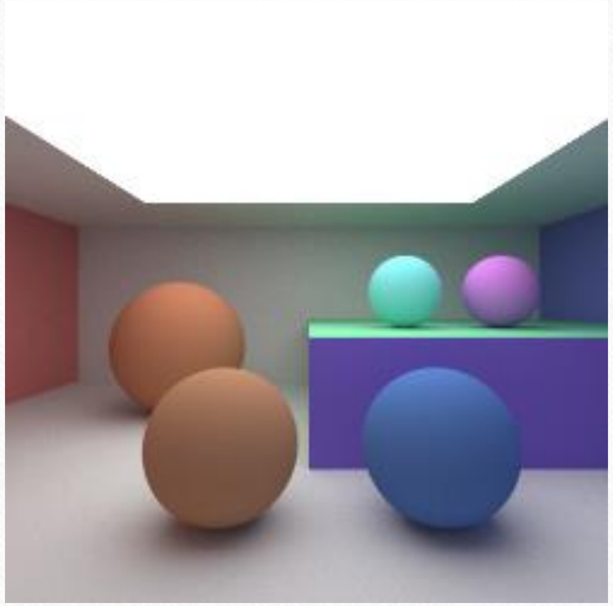
- $S_{k,n}(x) = (\sum_{i=1}^{k} \omega_{i,1} B_{i,2,1}\left(\frac{x-c_i}{\alpha_i}\right), \dots, \sum_{i=1}^{k} \omega_{i,n} B_{i,2,n}\left(\frac{x-c_i}{\alpha_i}\right))$

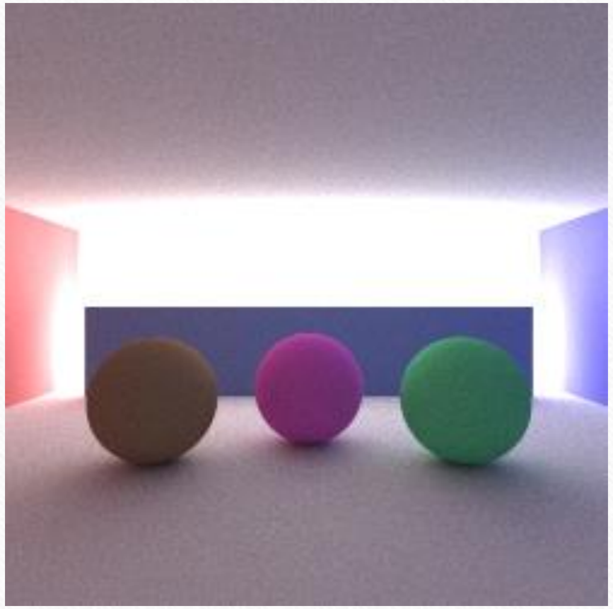
Trainable Quadratic B-spline Encoder

- Divide each of (x,y,z) axes into k segments.
- Each dimension encoded to
- $P = \begin{bmatrix} x_1 & \cdots & x_s \\ y_1 & \ddots & y_s \\ z_1 & \cdots & z_s \end{bmatrix}$
- $\omega_i = softplus(NN_\theta(P)[1{:}k, :])$
- $\delta_i = softmax(NN_\theta(P)[k+1{:}2k, :])$
- $c_i = vcat(zeros(1, s * n * 3), cumsum(\delta_i, [1{:}\mathrm{end}-1, :]))$

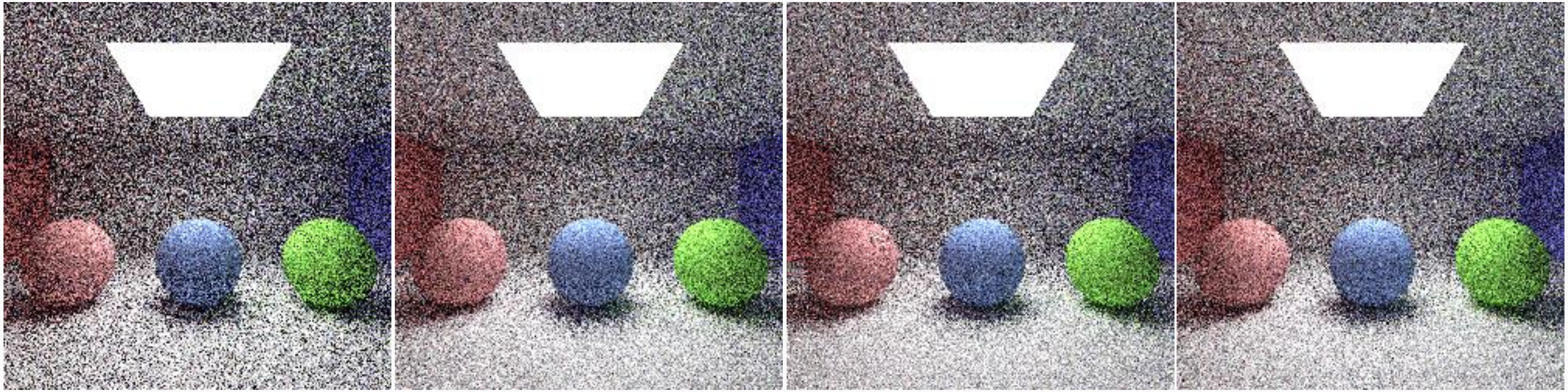
Scenes



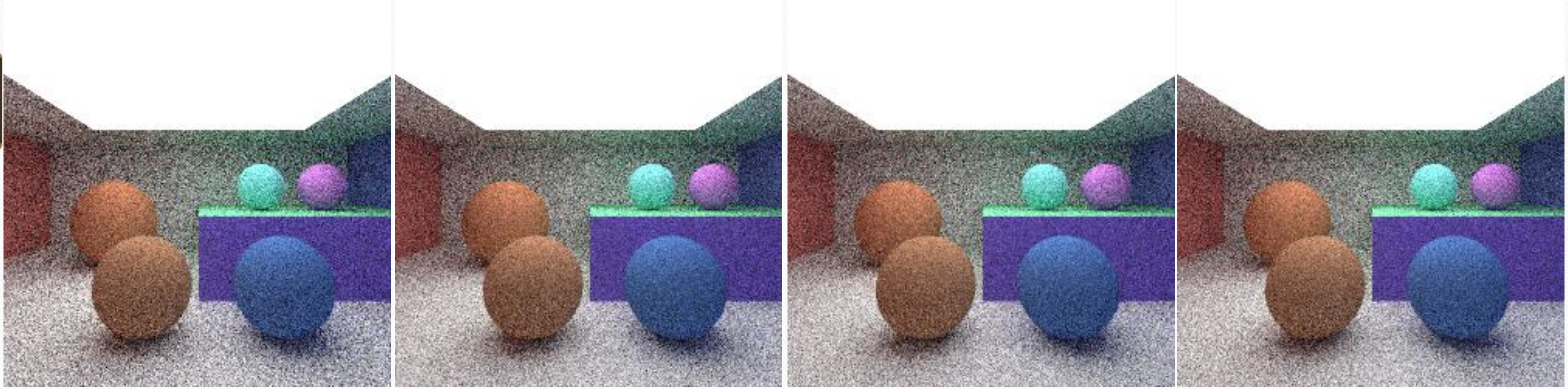




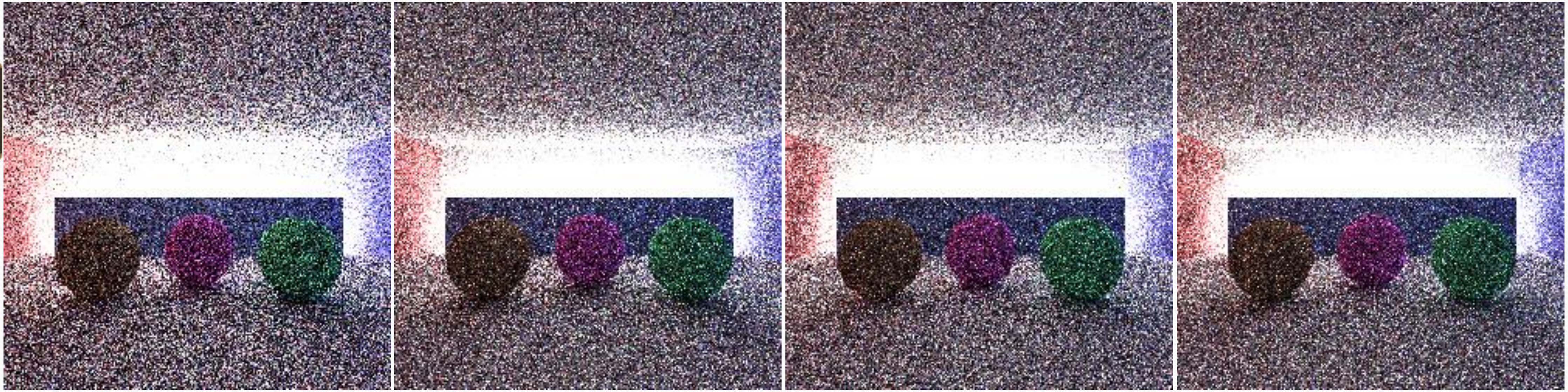
Results: Uniform, DQN-No-Encoding, DQN-RBF, DQN-Bspline



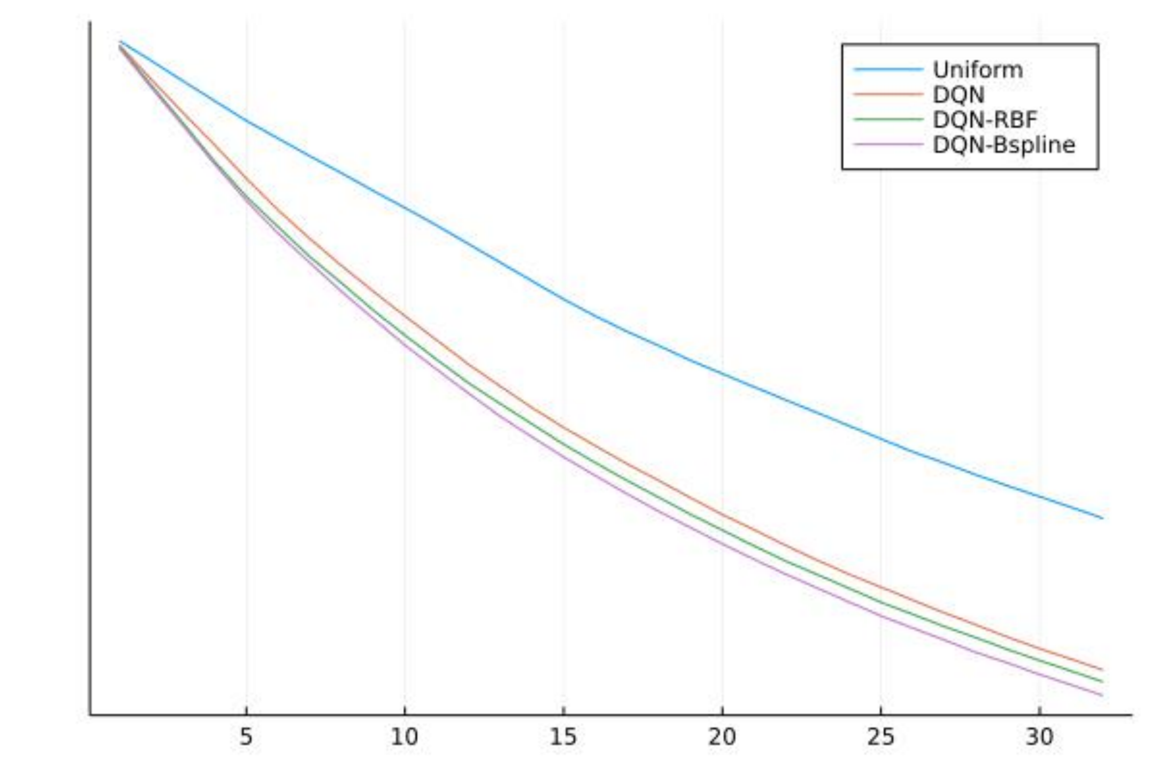
Results: Uniform, DQN-No-Encoding, DQN-RBF, DQN-Bspline



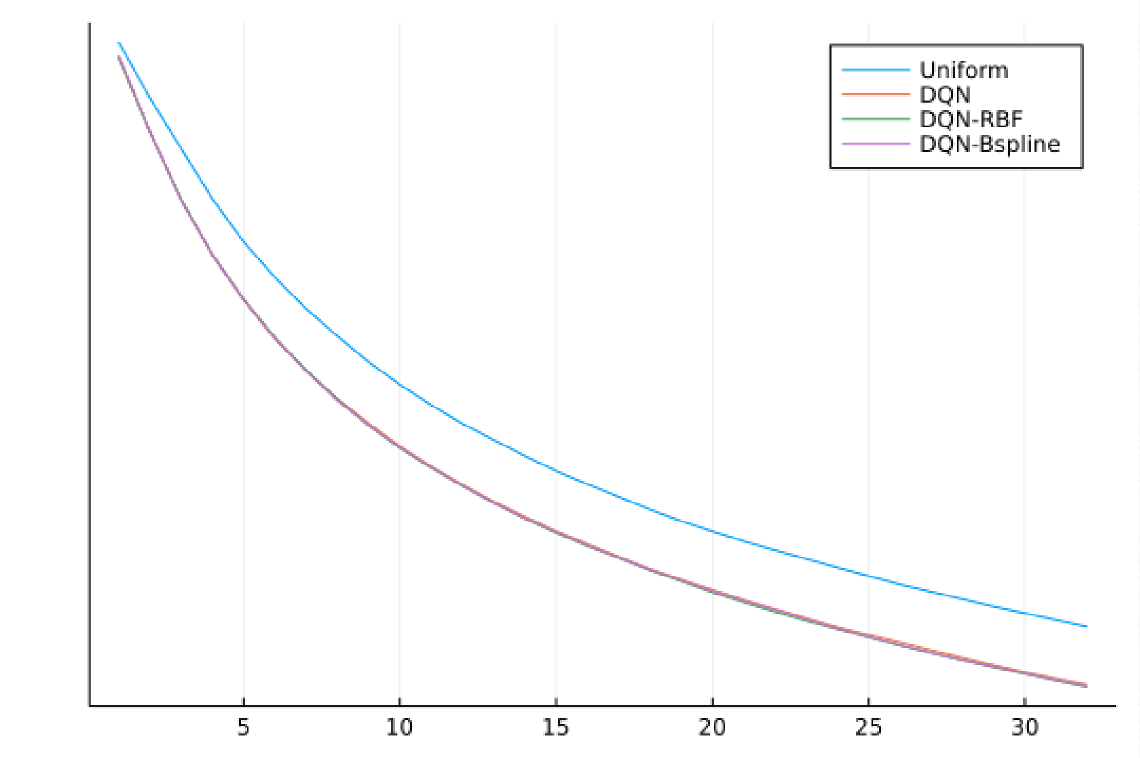
Results: Uniform, DQN-No-Encoding, DQN-RBF, DQN-Bspline



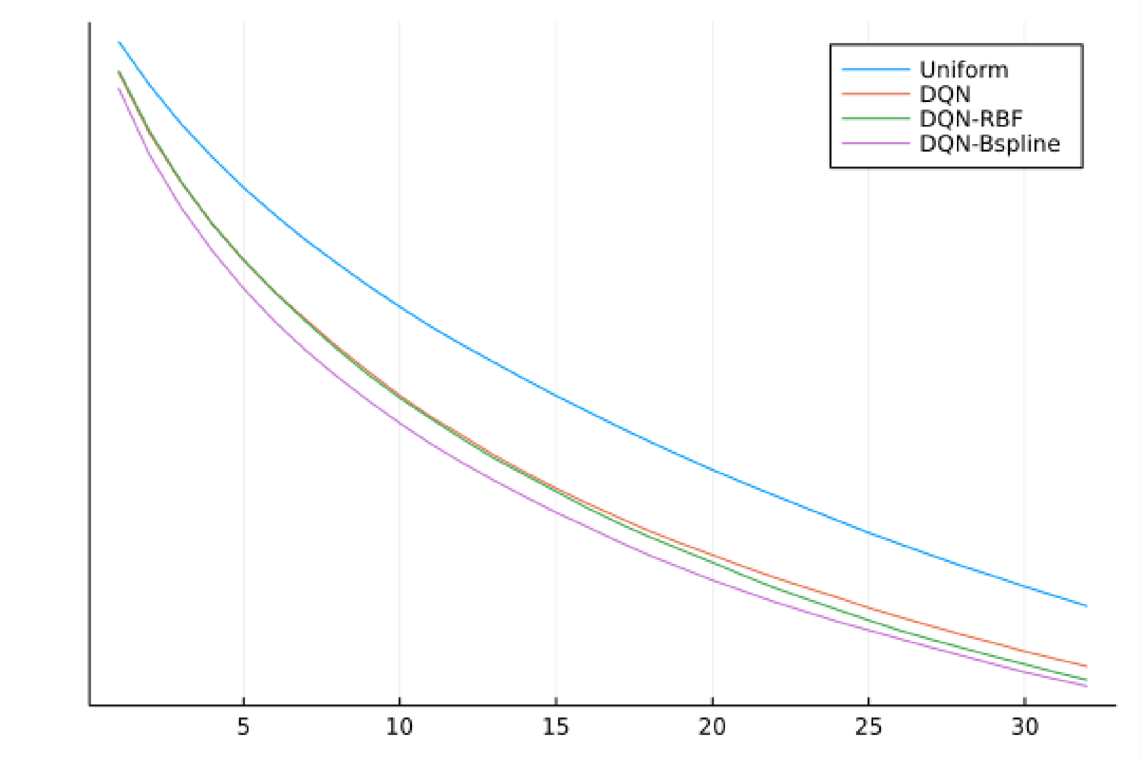
MAPE



MAPE



MAPE



MAPE

Uniform	**DQN-No-Encoding**	**DQN-RBF**	**DQN-Bspline**
0.41189	0.32089	0.31474	**0.30777**
0.12705	0.10882	**0.10823**	0.10827
0.41874	0.38466	0.37741	**0.37414**