Designing Neural Network Architectures Using Reinforcement Learning

Bowen Baker, Otkrist Gupta, Nikhil Naik, Ramesh Raskar

Under Review for ICLR 2017

Motivation

- Neural network design is still hand crafted.

Motivation

- Neural network design is still hand crafted.
- Despite the wide usage of a few main networks, we may want networks specialized for specific tasks.

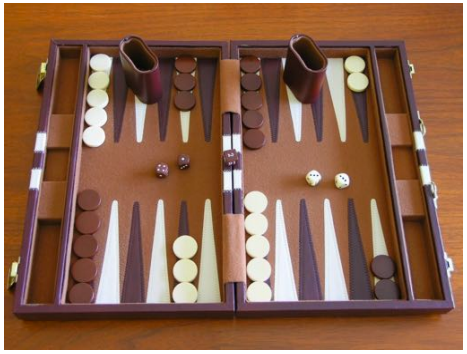
Motivation

- Neural network design is still hand crafted.
- Despite the wide usage of a few main networks, we may want networks specialized for specific tasks.
- We may want more than 1 specialized model, e.g. for the ensembling purposes.

Outline

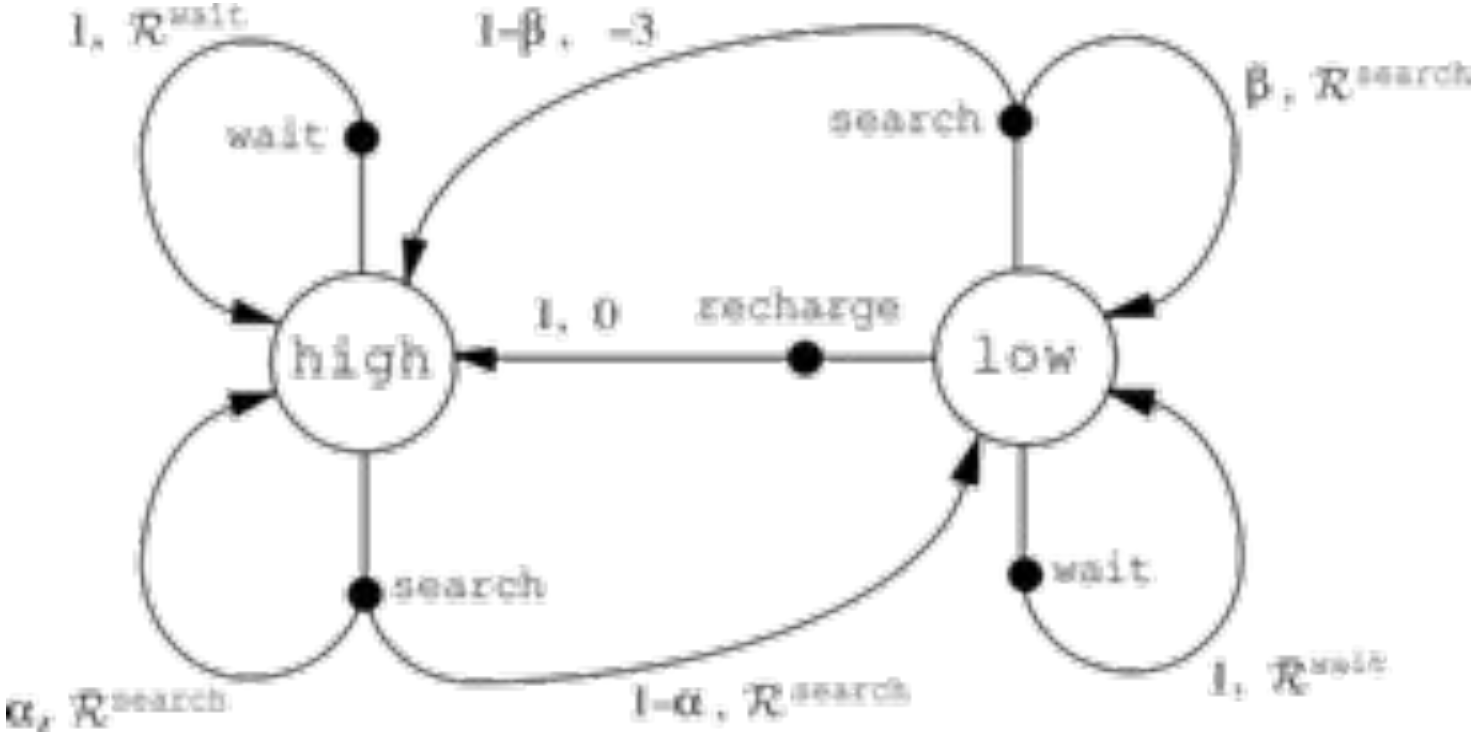
1. Reinforcement Learning Background
2. Modeling Architecture Selection as a Markov Decision Process
3. Experimental Configuration
4. Results
5. Discussion

Automating Tasks With Reinforcement Learning



Markov Decision Processes

Example -- The Recycling Robot



Q-Learning

$Q^*(s, u)$ -- Denotes the expected reward when following an optimal policy after taking action *u* at state *s*

Q-Learning

$$Q^*(s_i, u) = \mathbb{E}\left[r + \gamma \max_{u' \in \mathcal{U}(s_j)} Q^*(s_j, u')\right]$$

γ -- Discount Factor

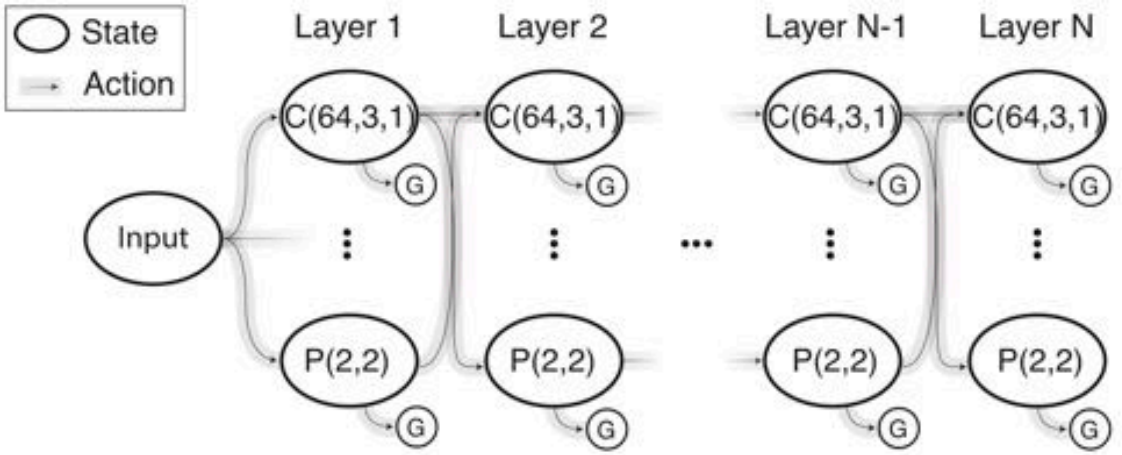
r -- Reward received from the (s_i, u, s_j) transition

Q-Learning

$$Q^*(s_i, u) = \mathbb{E}\left[r + \gamma \max_{u' \in \mathcal{U}(s_j)} Q^*(s_j, u')\right]$$

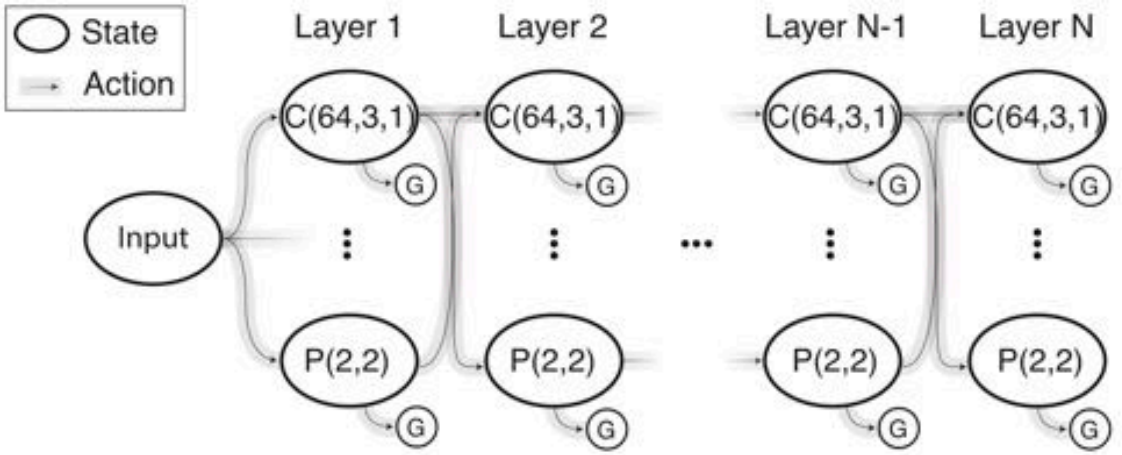
$$Q_{t+1}(s_i, u) = (1 - \alpha) Q_t(s_i, u) + \alpha \left[r_t + \gamma \max_{u' \in \mathcal{U}(s_j)} Q_t(s_j, u')\right]$$

Modeling Architecture Selection as a Markov Decision Process



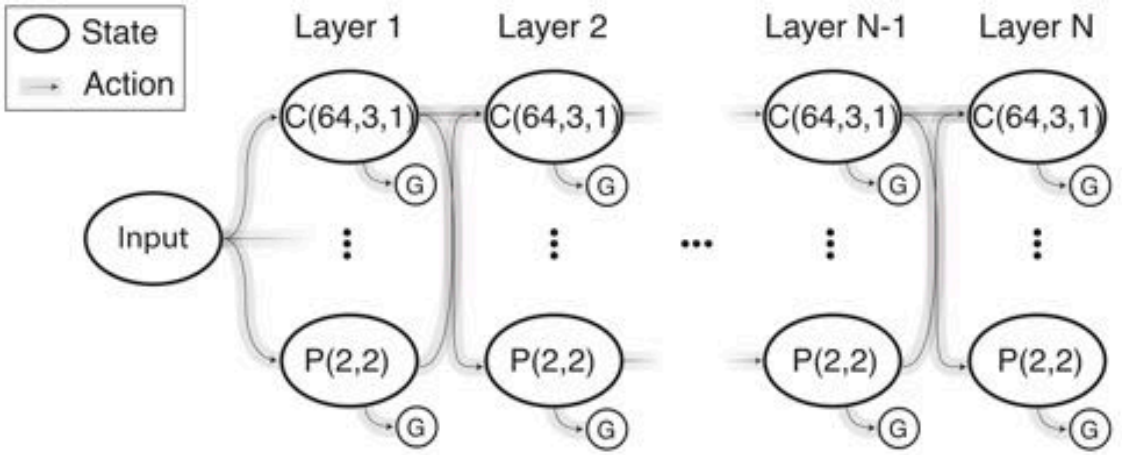
- C(64,3,1) – Convolutional Layer with 64 learnable kernels, 3x3 kernel size, and stride of 1

Modeling Architecture Selection as a Markov Decision Process



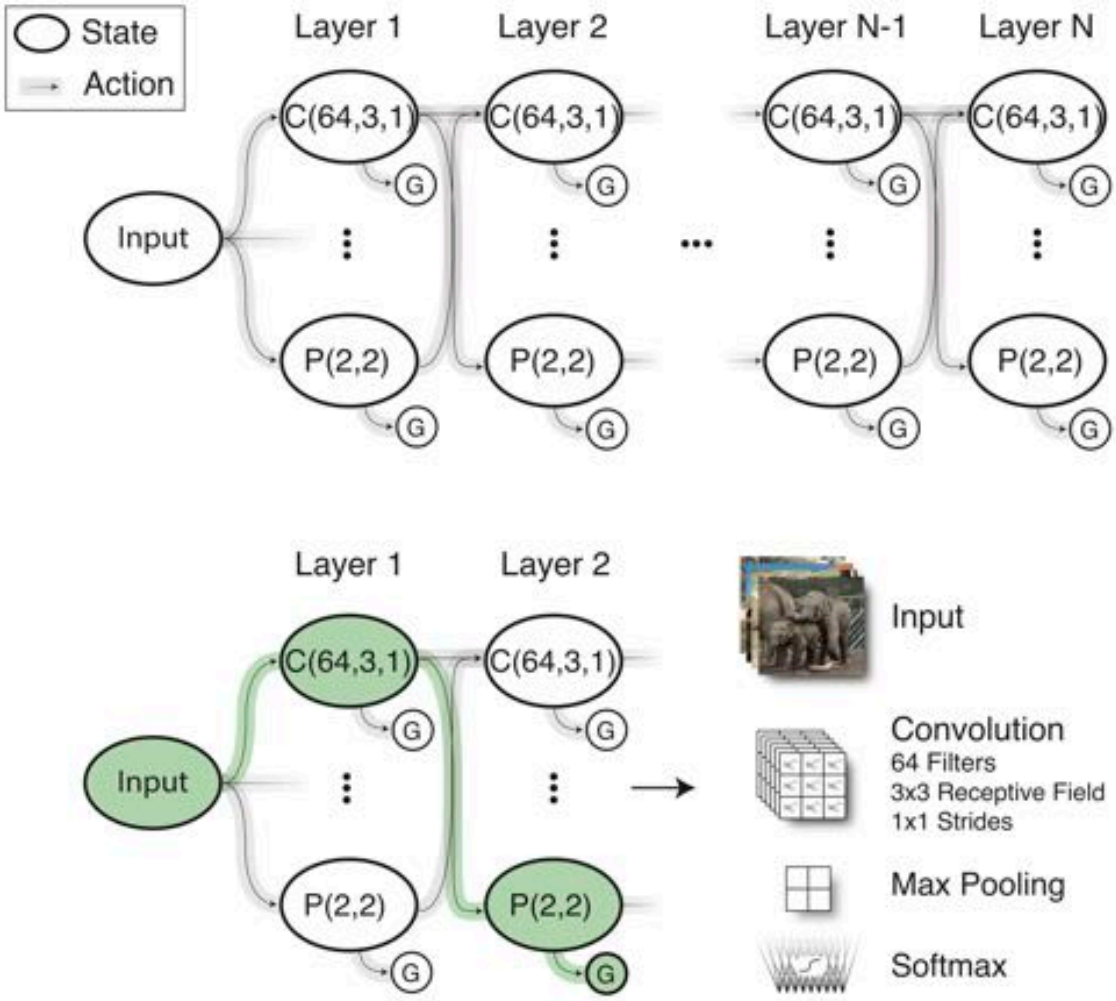
- C(64,3,1) – Convolutional Layer with 64 learnable kernels, 3x3 kernel size, and stride of 1
- P(2,2) – Max Pooling Layer with 2x2 kernel size and stride 2

Modeling Architecture Selection as a Markov Decision Process

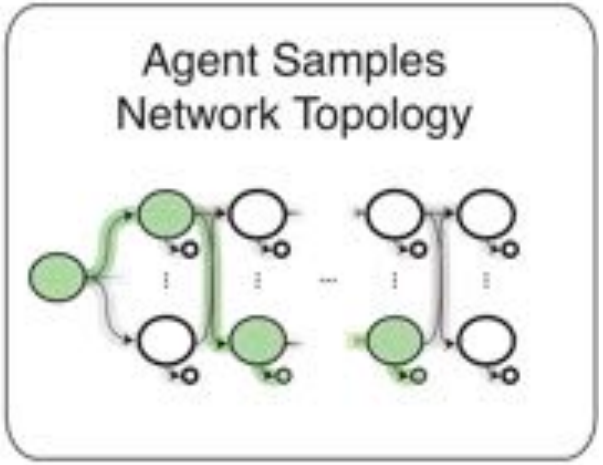


- C(64,3,1) – Convolutional Layer with 64 learnable kernels, 3x3 kernel size, and stride of 1
- P(2,2) – Max Pooling Layer with 2x2 kernel size and stride 2
- G – Termination State (e.g. Softmax)

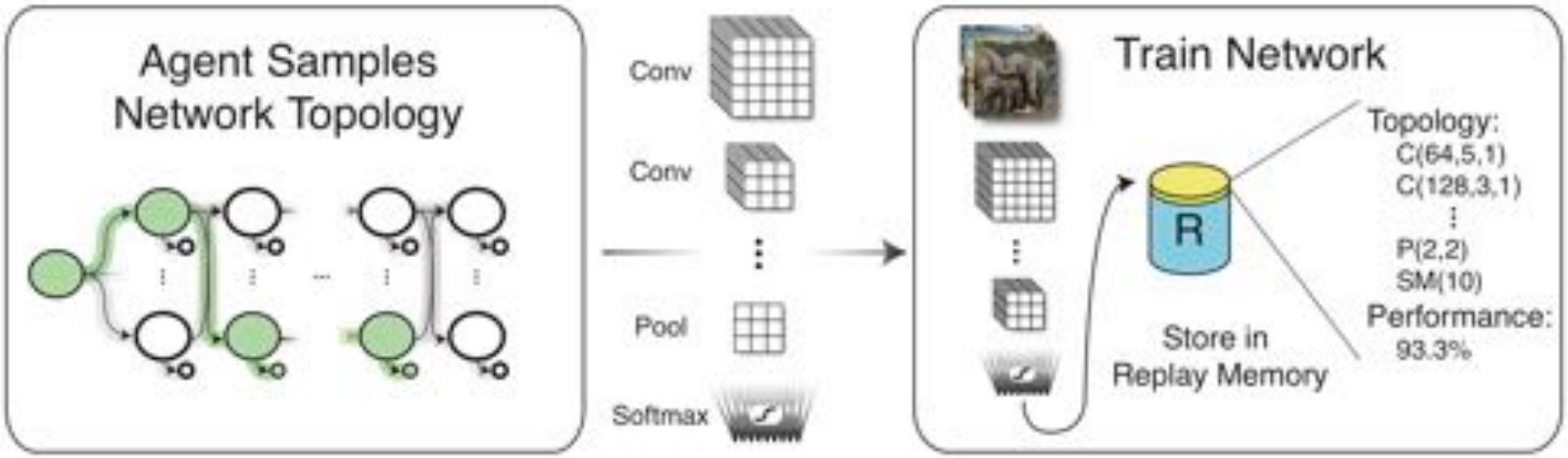
Modeling Architecture Selection as a Markov Decision Process



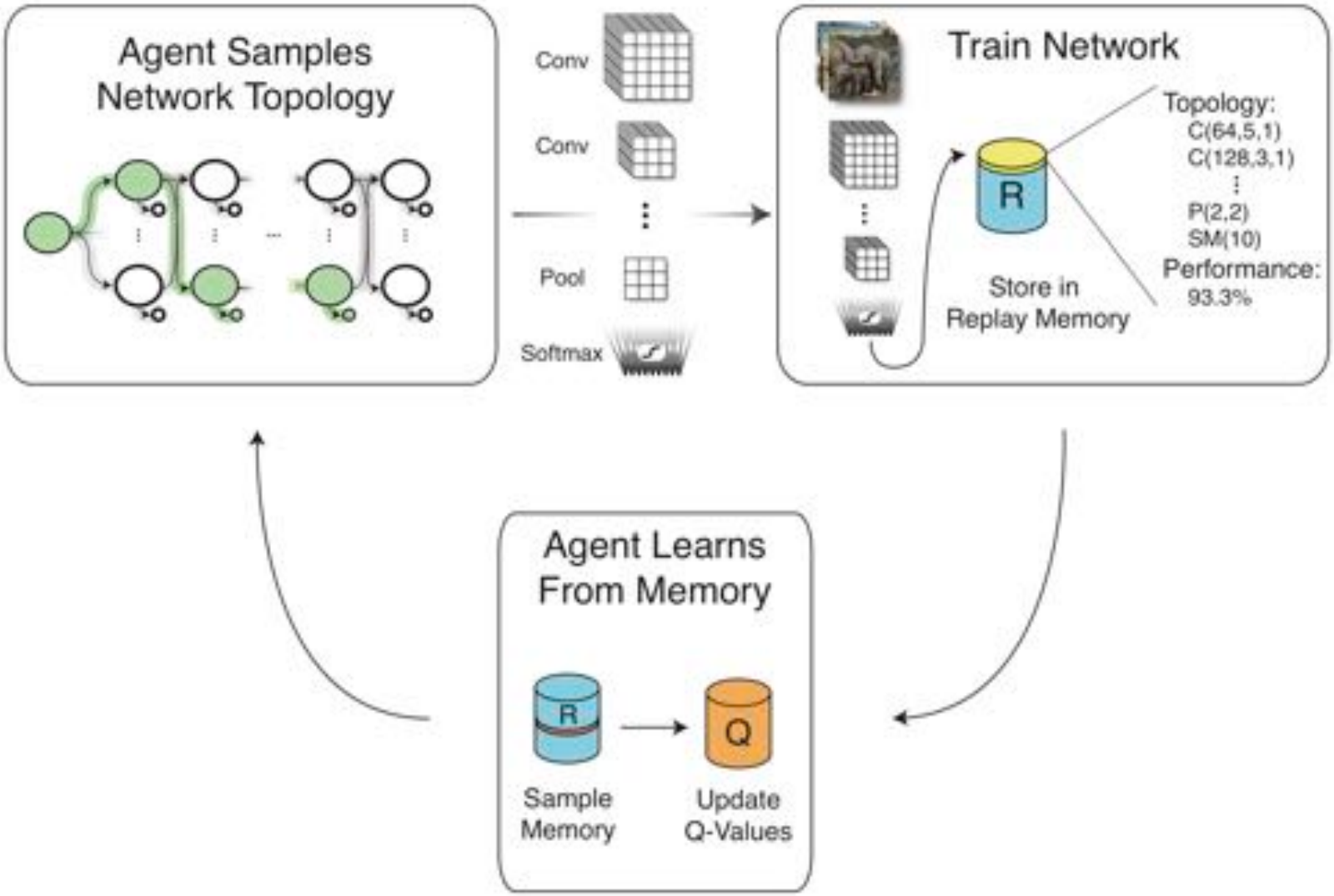
MetaQNN



MetaQNN


Agent Samples
Network Topology
Conv
Conv
Pool
Softmax
Train Network
R
Store in
Replay Memory
Topology:
C(64,5,1)
C(128,3,1)
P(2,2)
SM(10)
Performance:
93.3%

MetaQNN


Agent Samples
Network Topology
Conv
Conv
Pool
Softmax
Train Network
R
Topology:
C(64,5,1)
C(128,3,1)
P(2,2)
SM(10)
Performance:
93.3%
Store in
Replay Memory
Agent Learns
From Memory
R
Q
Sample
Memory
Update
Q-Values

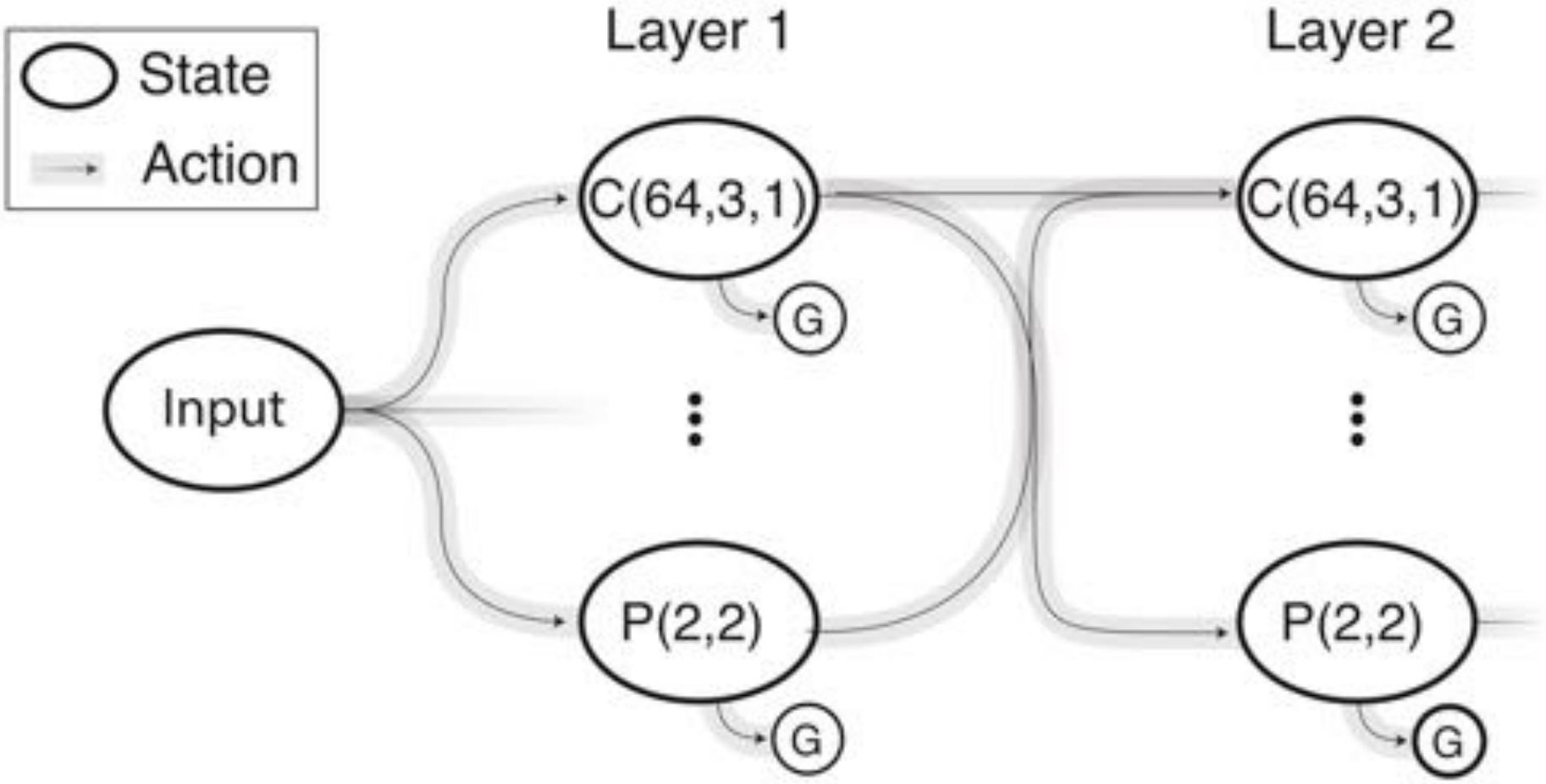
Sampling Networks

Epsilon-Greedy Exploration:

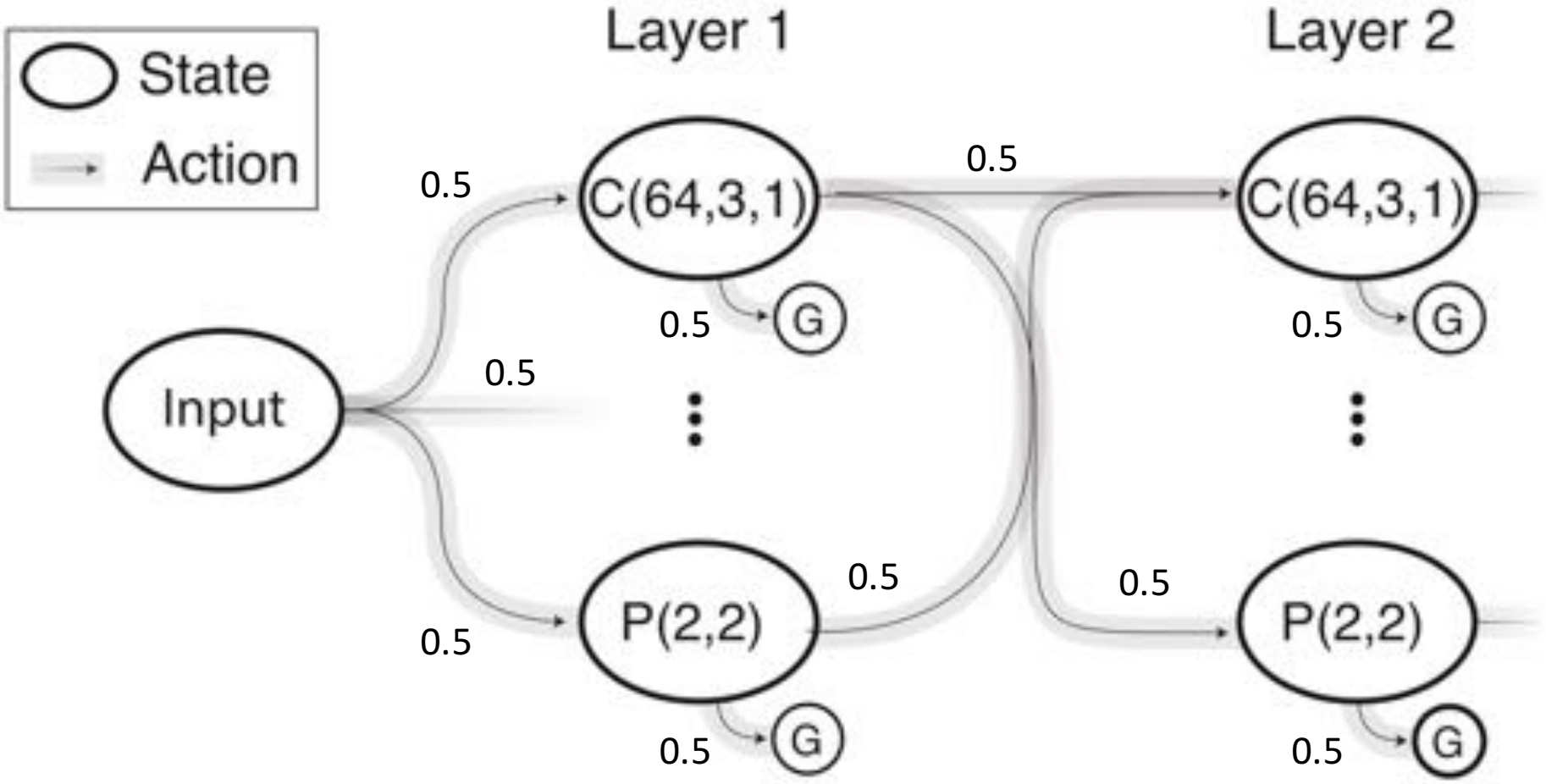
- State *s* corresponds the last layer chosen
- Action *u* corresponds to the next layer chosen

$$u = \begin{cases} \text{Uniform}[\mathcal{U}(s)] & \text{with probability } \epsilon \\ \arg\max_{u' \in \mathcal{U}(s)} [Q(s, u')] & \text{with probability } 1 - \epsilon \end{cases}$$

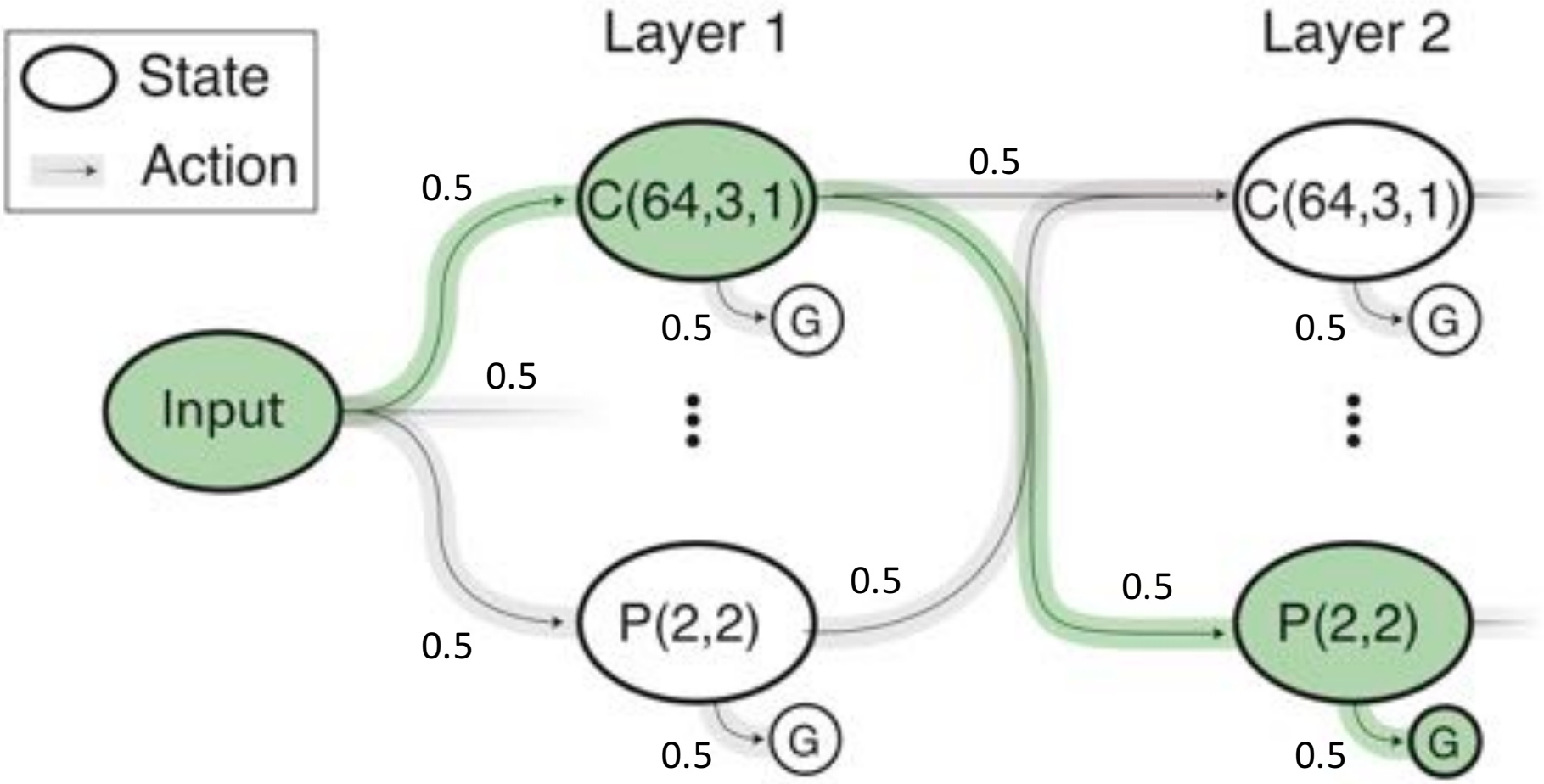
Q-Value Update (Example)



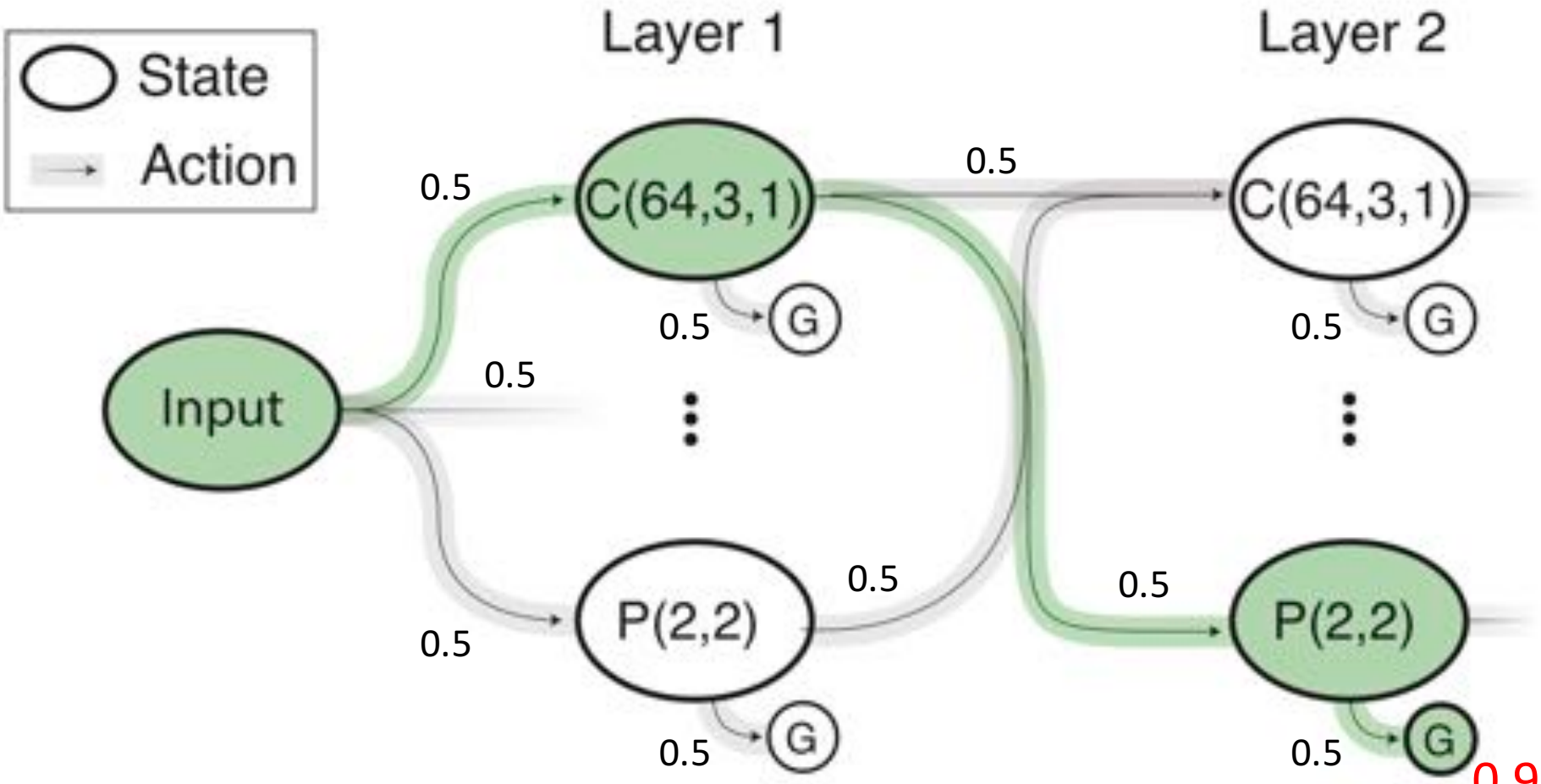
Q-Value Update (Example)



Q-Value Update (Example)

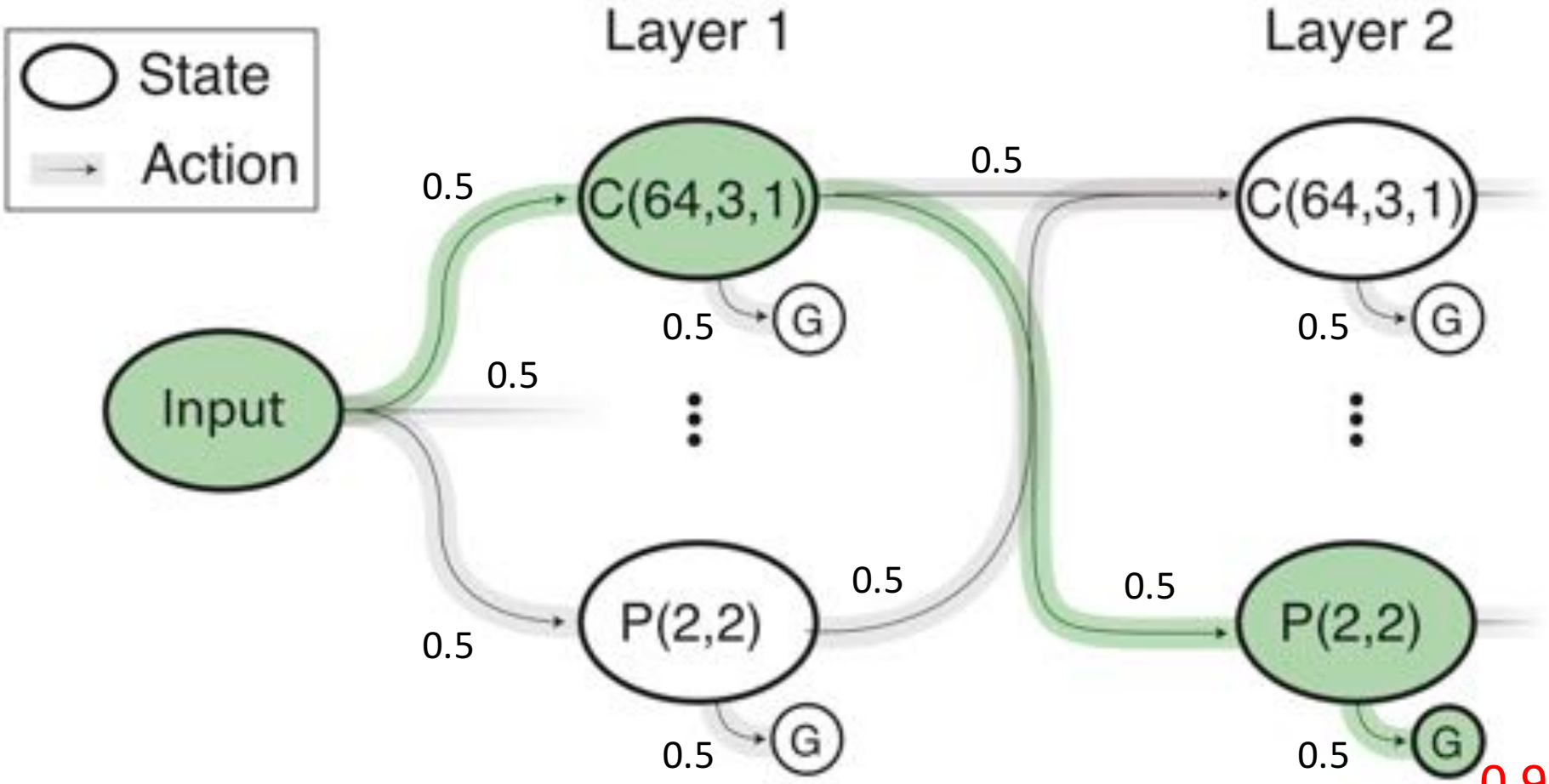


Q-Value Update (Example)



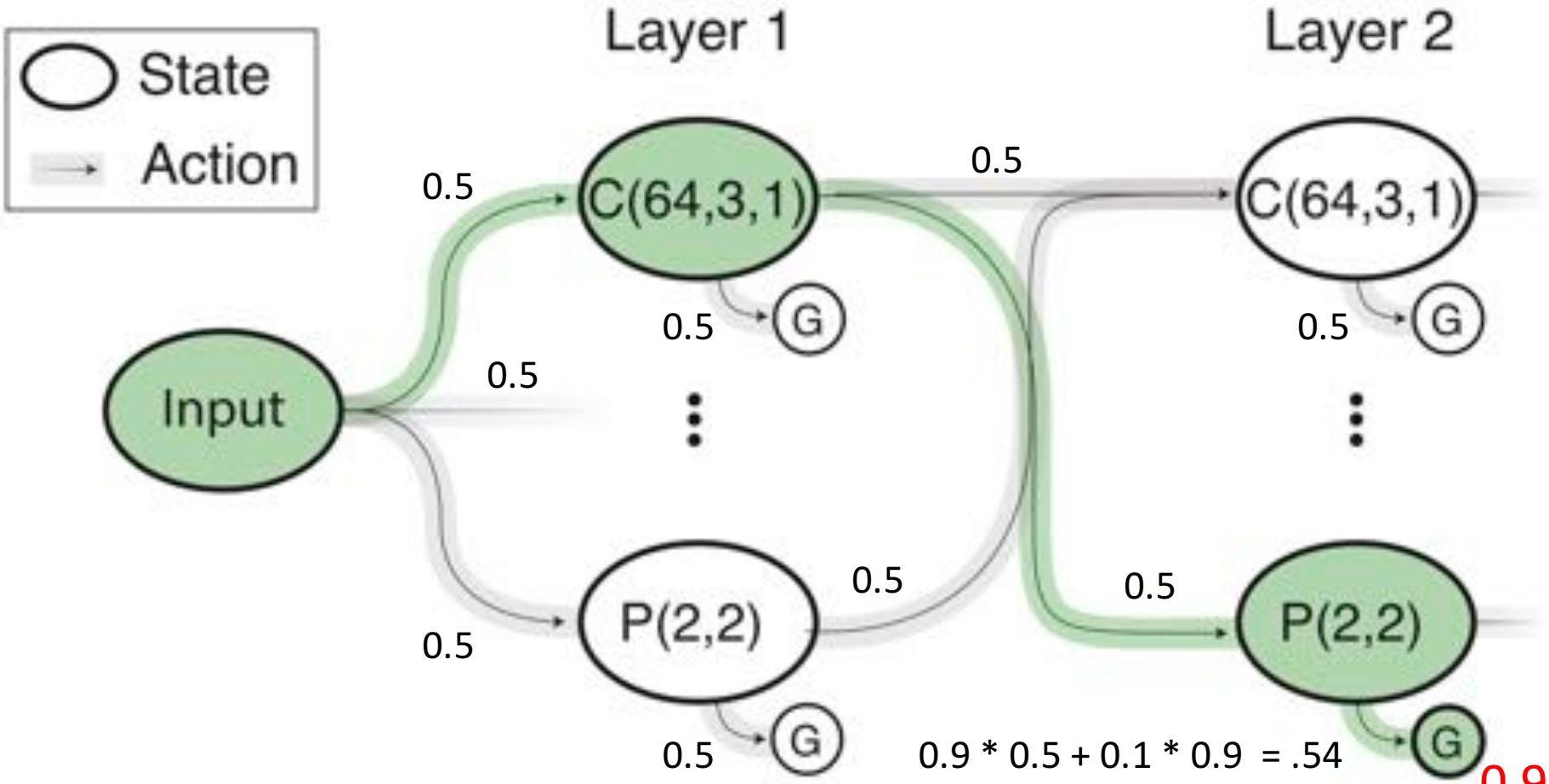
Q-Value Update (Example)

$$Q_{t+1}(s_i, u) = (1-\alpha)Q_t(s_i, u) + \alpha\left[r_t + \gamma \max_{u' \in \mathcal{U}(s_j)} Q_t(s_j, u')\right], \quad \gamma = 1, \quad \alpha = 0.1$$



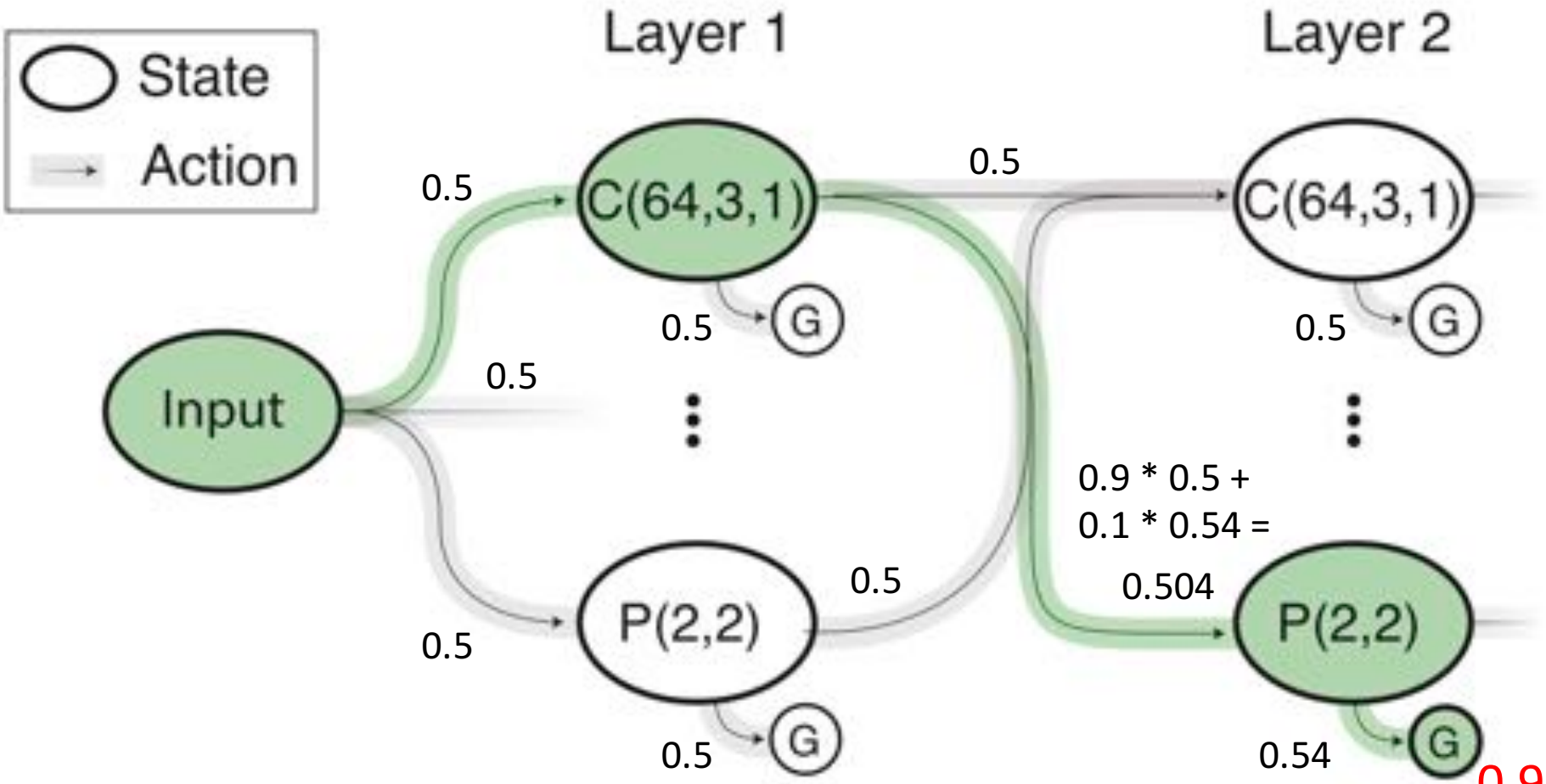
Q-Value Update (Example)

$$Q_{t+1}(s_i, u) = (1-\alpha)Q_t(s_i, u) + \alpha\left[r_t + \gamma \max_{u' \in \mathcal{U}(s_j)} Q_t(s_j, u')\right], \quad \gamma = 1, \quad \alpha = 0.1$$



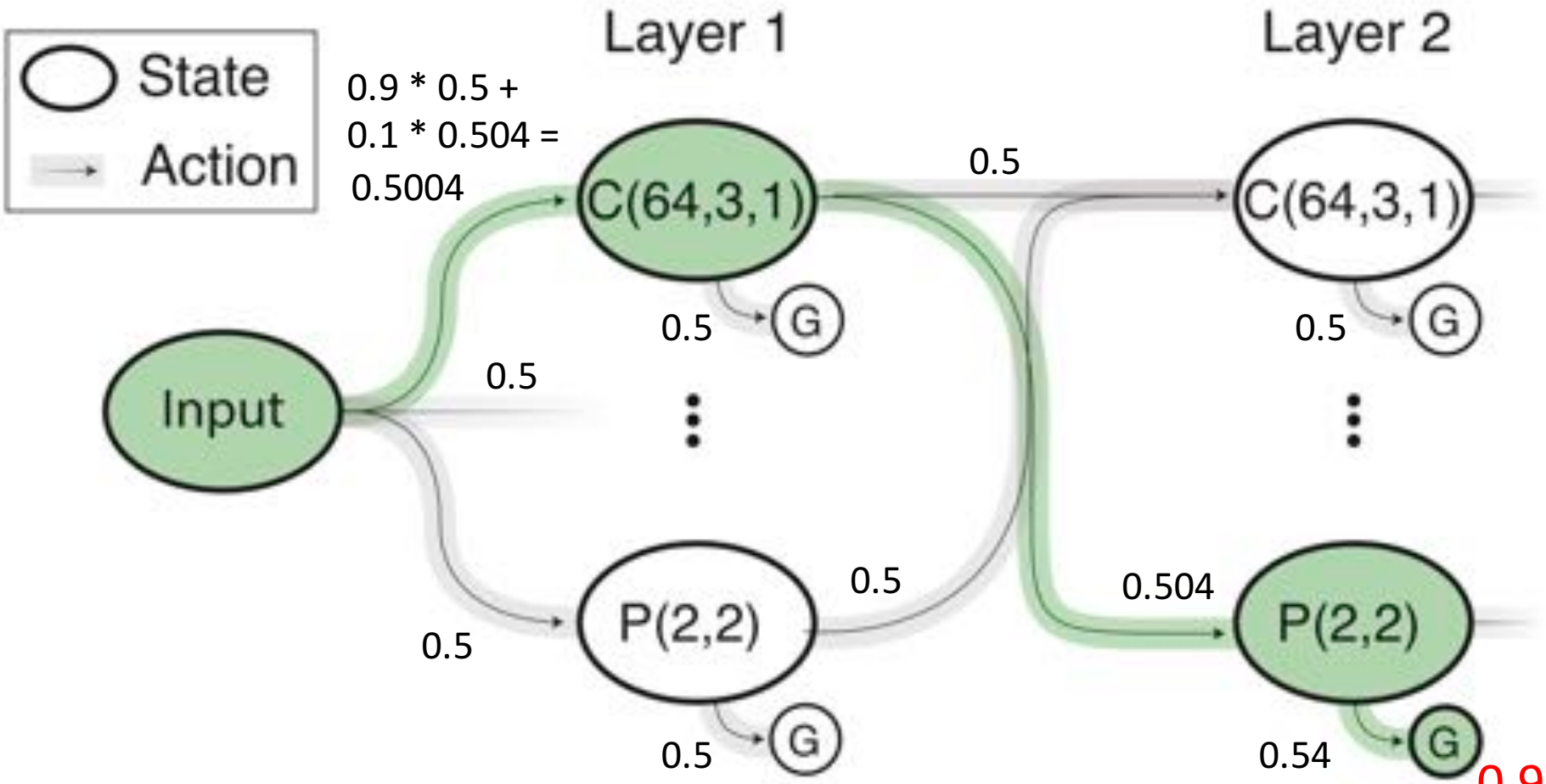
Q-Value Update (Example)

$$Q_{t+1}(s_i, u) = (1-\alpha)Q_t(s_i, u) + \alpha\left[r_t + \gamma \max_{u' \in \mathcal{U}(s_j)} Q_t(s_j, u')\right], \quad \gamma = 1, \quad \alpha = 0.1$$



Q-Value Update (Example)

$$Q_{t+1}(s_i, u) = (1-\alpha)Q_t(s_i, u) + \alpha\left[r_t + \gamma \max_{u' \in \mathcal{U}(s_j)} Q_t(s_j, u')\right], \quad \gamma = 1, \quad \alpha = 0.1$$



State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth $f \sim$ Receptive field size $\ell \sim$ Stride $d \sim$ # receptive fields $n \sim$ Representation size	< 12 Square. $\in \{1, 3, 5\}$ Square. Always equal to 1 $\in \{64, 128, 256, 512\}$ $\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth $(f, \ell) \sim$ (Receptive field size, Strides) $n \sim$ Representation size	< 12 Square. $\in \{(5, 3), (3, 2), (2, 2)\}$ $\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth $n \sim$ # consecutive FC layers $d \sim$ # neurons	< 12 < 3 $\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State $t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth	< 12
	$f \sim$ Receptive field size	Square. $\in \{1, 3, 5\}$
	$\ell \sim$ Stride	Square. Always equal to 1
	$d \sim$ # receptive fields	$\in \{64, 128, 256, 512\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth	< 12
	$(f, \ell) \sim$ (Receptive field size, Strides)	Square. $\in \{(5, 3), (3, 2), (2, 2)\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth	< 12
	$n \sim$ # consecutive FC layers	< 3
	$d \sim$ # neurons	$\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State	
	$t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth	< 12
	$f \sim$ Receptive field size	Square. $\in \{1, 3, 5\}$
	$\ell \sim$ Stride	Square. Always equal to 1
	$d \sim$ # receptive fields	$\in \{64, 128, 256, 512\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth	< 12
	$(f, \ell) \sim$ (Receptive field size, Strides)	Square. $\in \{(5, 3), (3, 2), (2, 2)\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth	< 12
	$n \sim$ # consecutive FC layers	< 3
	$d \sim$ # neurons	$\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State	
	$t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth	< 12
	$f \sim$ Receptive field size	Square. $\in \{1, 3, 5\}$
	$\ell \sim$ Stride	Square. Always equal to 1
	$d \sim$ # receptive fields	$\in \{64, 128, 256, 512\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth	< 12
	$(f, \ell) \sim$ (Receptive field size, Strides)	Square. $\in \{(5, 3), (3, 2), (2, 2)\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth	< 12
	$n \sim$ # consecutive FC layers	< 3
	$d \sim$ # neurons	$\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State	
	$t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth	< 12
	$f \sim$ Receptive field size	Square. $\in \{1, 3, 5\}$
	$\ell \sim$ Stride	Square. Always equal to 1
	$d \sim$ # receptive fields	$\in \{64, 128, 256, 512\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth	< 12
	$(f, \ell) \sim$ (Receptive field size, Strides)	Square. $\in \{(5, 3), (3, 2), (2, 2)\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth	< 12
	$n \sim$ # consecutive FC layers	< 3
	$d \sim$ # neurons	$\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State	
	$t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth	< 12
	$f \sim$ Receptive field size	Square. $\in \{1, 3, 5\}$
	$\ell \sim$ Stride	Square. Always equal to 1
	$d \sim$ # receptive fields	$\in \{64, 128, 256, 512\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth	< 12
	$(f, \ell) \sim$ (Receptive field size, Strides)	Square. $\in \{(5, 3), (3, 2), (2, 2)\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth	< 12
	$n \sim$ # consecutive FC layers	< 3
	$d \sim$ # neurons	$\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State	
	$t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth $f \sim$ Receptive field size $\ell \sim$ Stride $d \sim$ # receptive fields $n \sim$ Representation size	< 12 Square. $\in \{1, 3, 5\}$ Square. Always equal to 1 $\in \{64, 128, 256, 512\}$ $\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth $(f, \ell) \sim$ (Receptive field size, Strides) $n \sim$ Representation size	< 12 Square. $\in \{(5, 3), (3, 2), (2, 2)\}$ $\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth $n \sim$ # consecutive FC layers $d \sim$ # neurons	< 12 < 3 $\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State $t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth	< 12
	$f \sim$ Receptive field size	Square. $\in \{1, 3, 5\}$
	$\ell \sim$ Stride	Square. Always equal to 1
	$d \sim$ # receptive fields	$\in \{64, 128, 256, 512\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth	< 12
	$(f, \ell) \sim$ (Receptive field size, Strides)	Square. $\in \{(5, 3), (3, 2), (2, 2)\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth	< 12
	$n \sim$ # consecutive FC layers	< 3
	$d \sim$ # neurons	$\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State	
	$t \sim$ Type	Global Avg. Pooling/Softmax

State Space

- State Space
  - Each state is a 6-dimensional tuple

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth	< 12
	$f \sim$ Receptive field size	Square. $\in \{1, 3, 5\}$
	$\ell \sim$ Stride	Square. Always equal to 1
	$d \sim$ # receptive fields	$\in \{64, 128, 256, 512\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4], (4, 1]\}$
Pooling (P)	$i \sim$ Layer depth	< 12
	$(f, \ell) \sim$ (Receptive field size, Strides)	Square. $\in \{(5, 3), (3, 2), (2, 2)\}$
	$n \sim$ Representation size	$\in \{(\infty, 8], (8, 4]$ and $(4, 1]\}$
Fully Connected (FC)	$i \sim$ Layer depth	< 12
	$n \sim$ # consecutive FC layers	< 3
	$d \sim$ # neurons	$\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State	
	$t \sim$ Type	Global Avg. Pooling/Softmax

Action Space

- Convolution → Any Other Layer

Action Space

- Convolution → Any Other Layer
- Pooling → Any Other Layer / Pooling

Action Space

- Convolution → Any Other Layer
- Pooling → Any Other Layer / Pooling
- Any Layer → Fully Connected
  - if representation size less than 8

Action Space

- Convolution → Any Other Layer
- Pooling → Any Other Layer / Pooling
- Any Layer → Fully Connected
  - if representation size less than 8
- Any Layer → Termination

Epsilon Schedule

ϵ	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
# Models Trained	1500	100	100	100	150	150	150	150	150	150

Experiments

MNIST

- Hand Written Digits
- 60,000 Training Examples
- 10,000 Testing Examples
- 10 classes

CIFAR-10

- Tiny Images
- 50,000 Training Examples
- 10,000 Testing Examples
- 10 classes

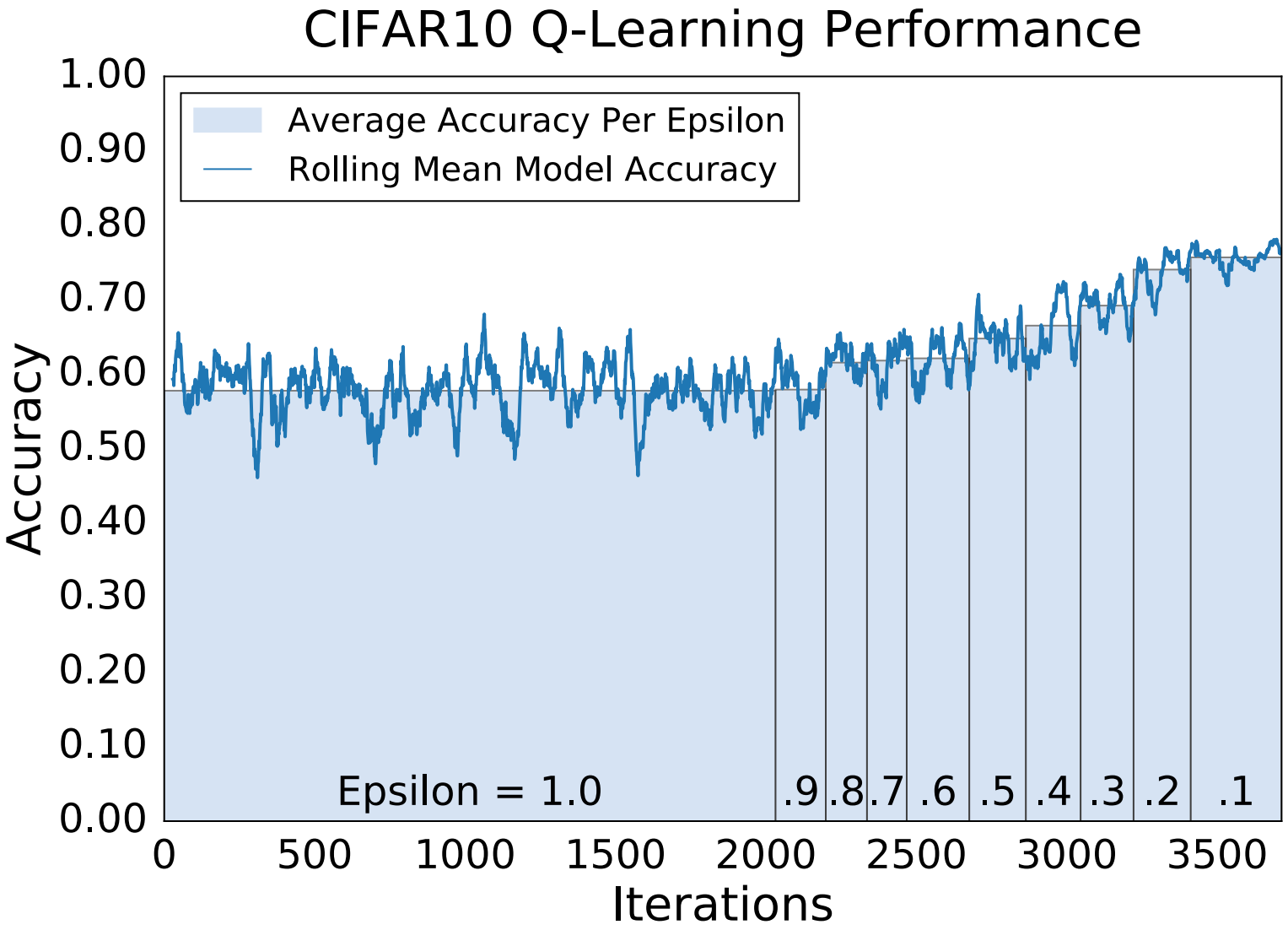
SVHN

- Street View House Digits
- 73257 Training Examples
- 26032 Testing Examples
- 531131 ‘Extra’ Examples
- 10 classes

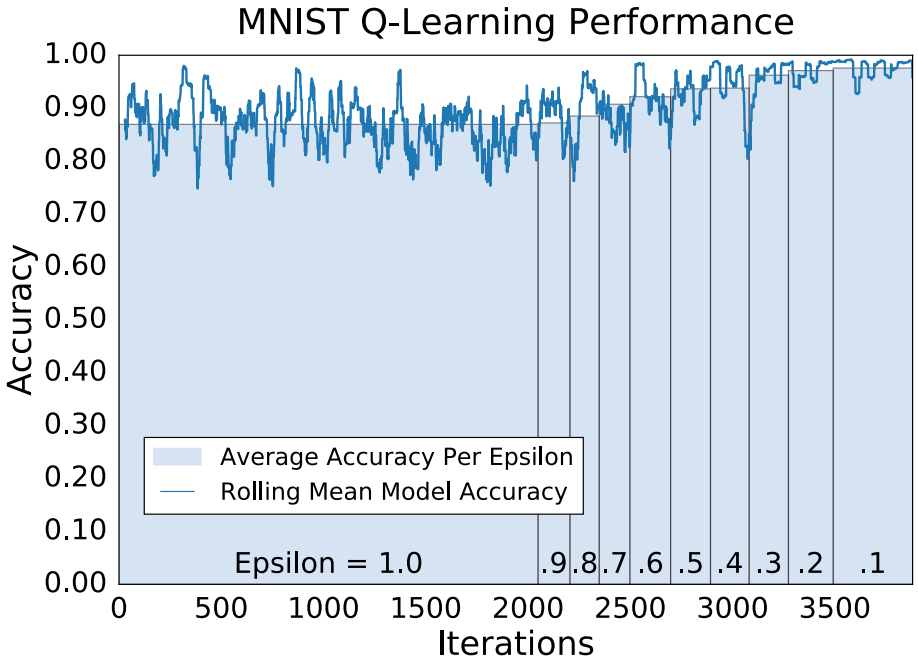
Hardware

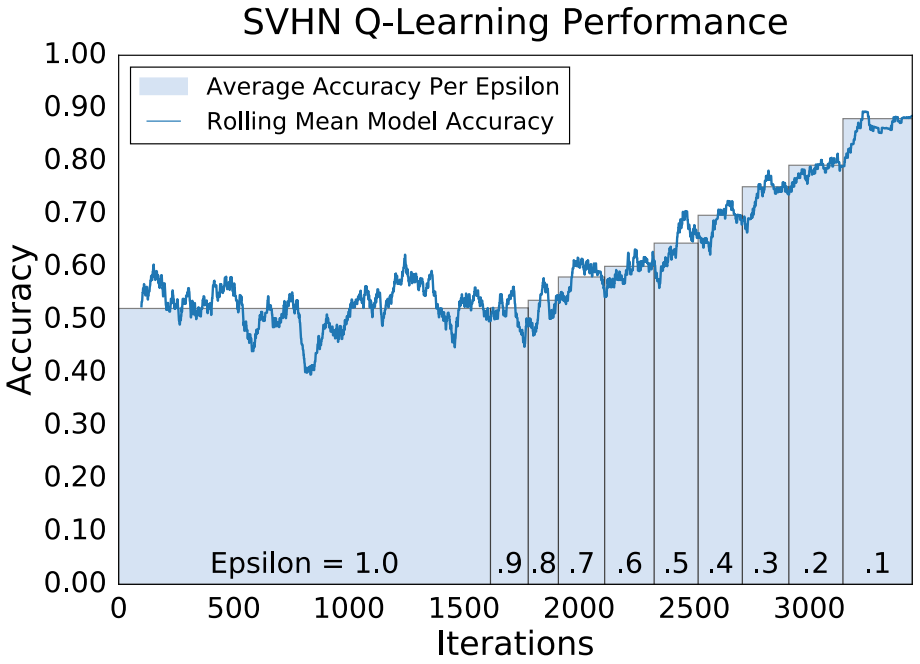
- ~10 GPU's
  - Mostly 2015 Titan Xs
  - Some GTX 1080s
- Each experiment took ~10 days
  - Roughly 100 GPUdays

Results


CIFAR10 Q-Learning Performance
Average Accuracy Per Epsilon
Rolling Mean Model Accuracy
Accuracy
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
Epsilon = 1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
500
1000
1500
2000
2500
3000
3500
Iterations

Results


MNIST Q-Learning Performance
Accuracy
Iterations
Average Accuracy Per Epsilon
Rolling Mean Model Accuracy
Epsilon = 1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1


SVHN Q-Learning Performance
Accuracy
Iterations
Average Accuracy Per Epsilon
Rolling Mean Model Accuracy
Epsilon = 1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1

Results

Method	CIFAR-10	SVHN	MNIST	CIFAR-100
DropConnect (Wan et al., 2013)	9.32	1.94	0.57	-
DSN (Lee et al., 2015)	8.22	1.92	0.39	34.57
R-CNN (Liang & Hu, 2015)	7.72	1.77	**0.31**	31.75
MetaQNN (ensemble)	7.32	2.06	0.32	-
MetaQNN (top model)	6.92	2.28	0.44	27.14*
Resnet(110) (He et al., 2015)	6.61	-	-	-
ELU (Clevert et al., 2015)	6.55	-	-	**24.28**
Tree+Max-Avg (Lee et al., 2016)	**6.05**	**1.69**	**0.31**	32.37

Results

Method	CIFAR-10	SVHN	MNIST	CIFAR-100
DropConnect (Wan et al., 2013)	9.32	1.94	0.57	-
DSN (Lee et al., 2015)	8.22	1.92	0.39	34.57
R-CNN (Liang & Hu, 2015)	7.72	1.77	**0.31**	31.75
MetaQNN (ensemble)	7.32	2.06	0.32	-
MetaQNN (top model)	6.92	2.28	0.44	27.14*
Resnet(110) (He et al., 2015)	6.61	-	-	-
ELU (Clevert et al., 2015)	6.55	-	-	**24.28**
Tree+Max-Avg (Lee et al., 2016)	**6.05**	**1.69**	**0.31**	32.37

Method	CIFAR-10	SVHN	MNIST	CIFAR-100
Maxout (Goodfellow et al., 2013)	9.38	2.47	0.45	38.57
NIN (Lin et al., 2013)	8.81	2.35	0.47	35.68
FitNet (Romero et al., 2014)	8.39	2.42	0.51	35.04
HighWay (Srivastava et al., 2015)	7.72	-	-	-
VGGnet (Simonyan & Zisserman, 2014)	7.25	-	-	-
All-CNN (Springenberg et al., 2014)	7.25	-	-	33.71
MetaQNN (ensemble)	7.32	**2.06**	**0.32**	-
MetaQNN (top model)	**6.92**	2.28	0.44	**27.14***

Results

Method	CIFAR-10	SVHN	MNIST	CIFAR-100
DropConnect (Wan et al., 2013)	9.32	1.94	0.57	-
DSN (Lee et al., 2015)	8.22	1.92	0.39	34.57
R-CNN (Liang & Hu, 2015)	7.72	1.77	**0.31**	31.75
MetaQNN (ensemble)	7.32	2.06	0.32	-
MetaQNN (top model)	6.92	2.28	0.44	27.14*
Resnet(110) (He et al., 2015)	6.61	-	-	-
ELU (Clevert et al., 2015)	6.55	-	-	**24.28**
Tree+Max-Avg (Lee et al., 2016)	**6.05**	**1.69**	**0.31**	32.37

Method	CIFAR-10	SVHN	MNIST	CIFAR-100
Maxout (Goodfellow et al., 2013)	9.38	2.47	0.45	38.57
NIN (Lin et al., 2013)	8.81	2.35	0.47	35.68
FitNet (Romero et al., 2014)	8.39	2.42	0.51	35.04
HighWay (Srivastava et al., 2015)	7.72	-	-	-
VGGnet (Simonyan & Zisserman, 2014)	7.25	-	-	-
All-CNN (Springenberg et al., 2014)	7.25	-	-	33.71
MetaQNN (ensemble)	7.32	**2.06**	**0.32**	-
MetaQNN (top model)	**6.92**	2.28	0.44	**27.14***

MNIST 10-Model Ensemble: 0.28% Error

Top Models (CIFAR-10)

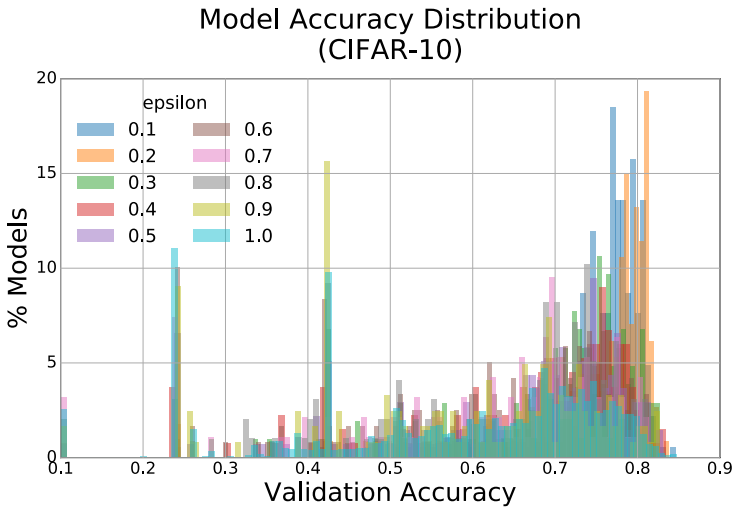
Model Architecture	Test Error (%)	# Params (10^6)
[C(512,5,1), C(256,3,1), C(256,5,1), C(256,3,1), P(5,3), C(512,3,1), C(512,5,1), P(2,2), SM(10)]	6.92	11.18
[C(128,1,1), C(512,3,1), C(64,1,1), C(128,3,1), P(2,2), C(256,3,1), P(2,2), C(512,3,1), P(3,2), SM(10)]	8.78	2.17
[C(128,3,1), C(128,1,1), C(512,5,1), P(2,2), C(128,3,1), P(2,2), C(64,3,1), C(64,5,1), SM(10)]	8.88	2.42
[C(256,3,1), C(256,3,1), P(5,3), C(256,1,1), C(128,3,1), P(2,2), C(128,3,1), SM(10)]	9.24	1.10
[C(128,5,1), C(512,3,1), P(2,2), C(128,1,1), C(128,5,1), P(3,2), C(512,3,1), SM(10)]	11.63	1.66

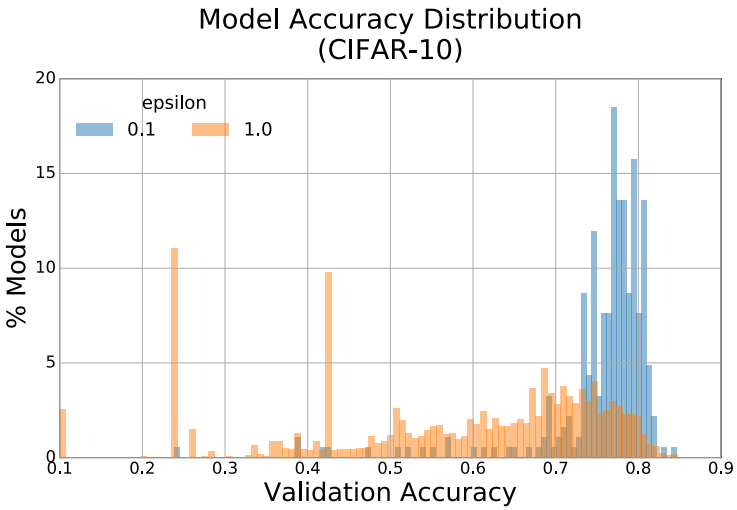
Transferability

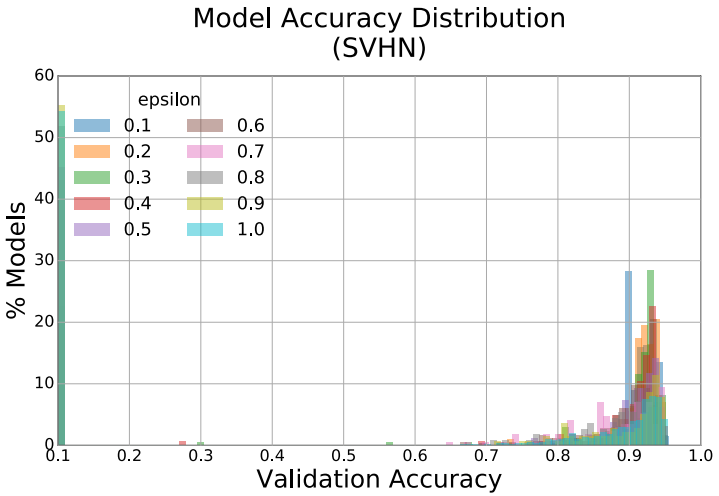
- Top model found in CIFAR-10 experiment trained for other tasks

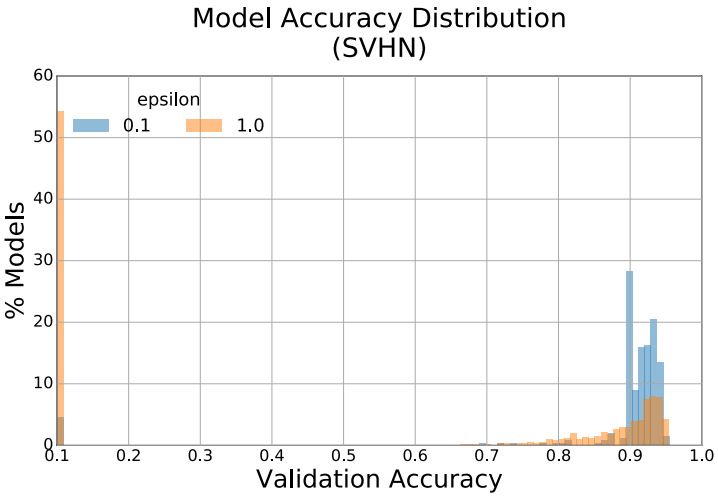
Dataset	CIFAR-100	SVHN	MNIST
Training from scratch	27.14	2.48	0.80
Finetuning	34.93	4.00	0.81
State-of-the-art	24.28 (Clevert et al., 2015)	1.69 (Lee et al., 2016)	0.31 (Lee et al., 2016)

Exploration Distributions

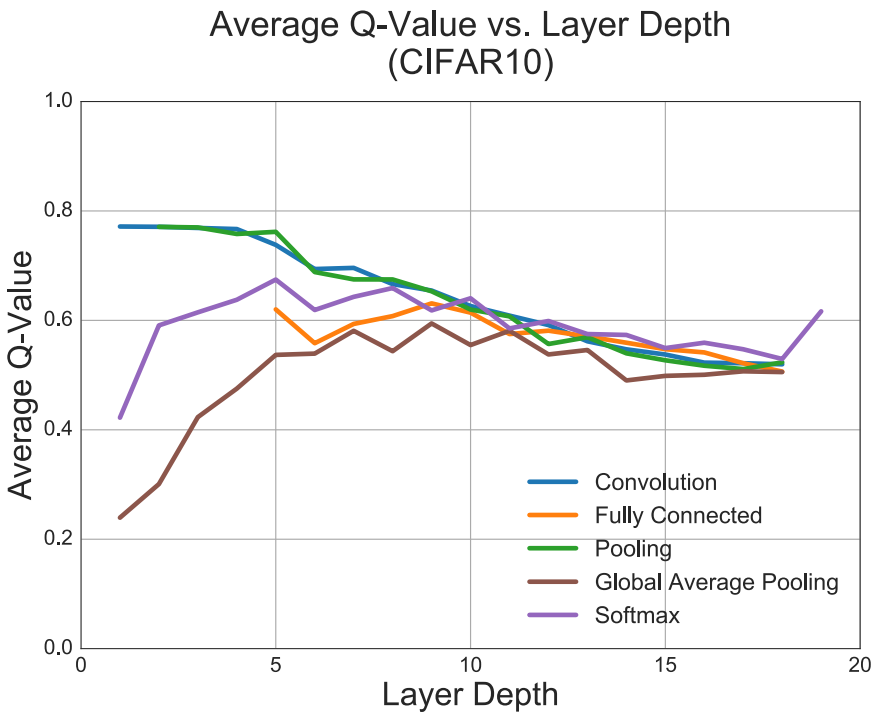


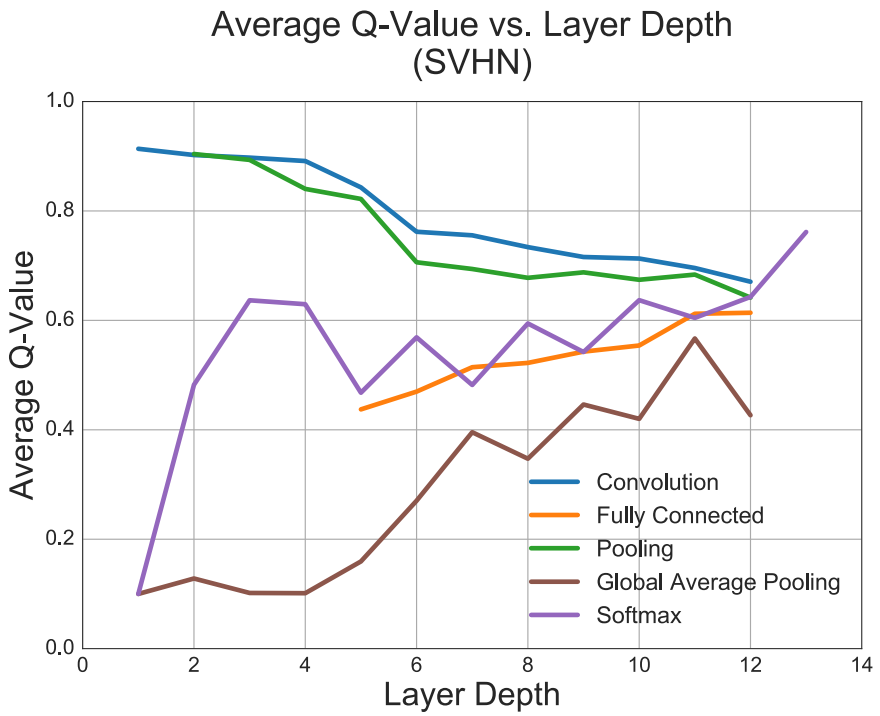




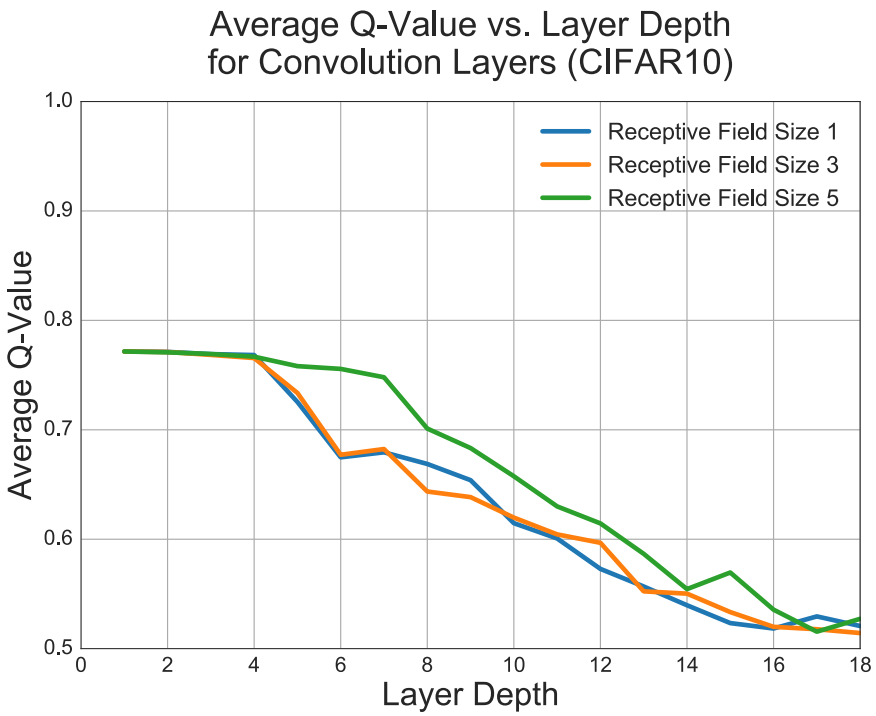


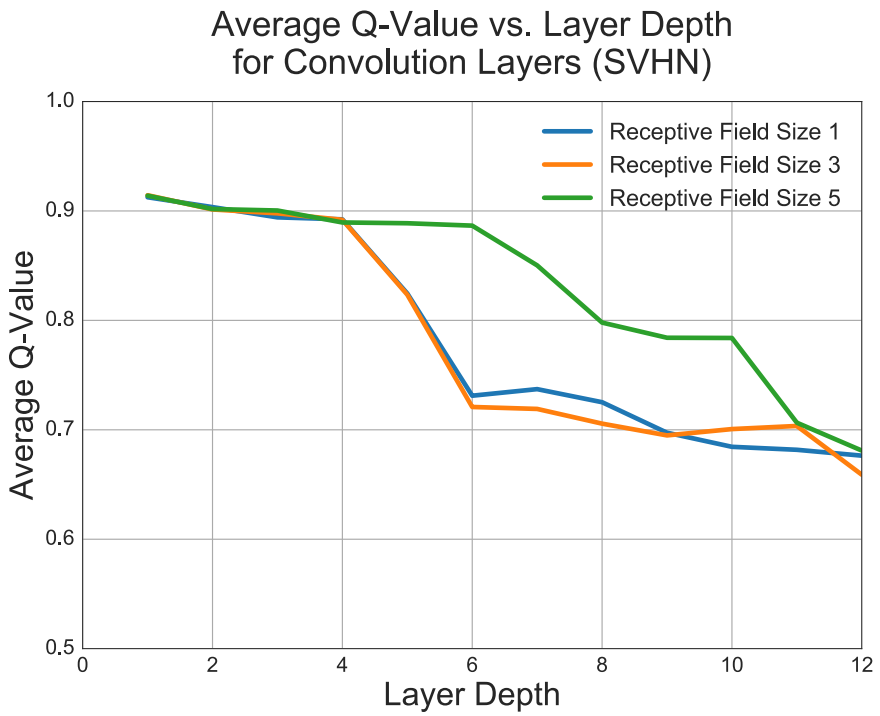
Q-Value Analysis





Q-Value Analysis

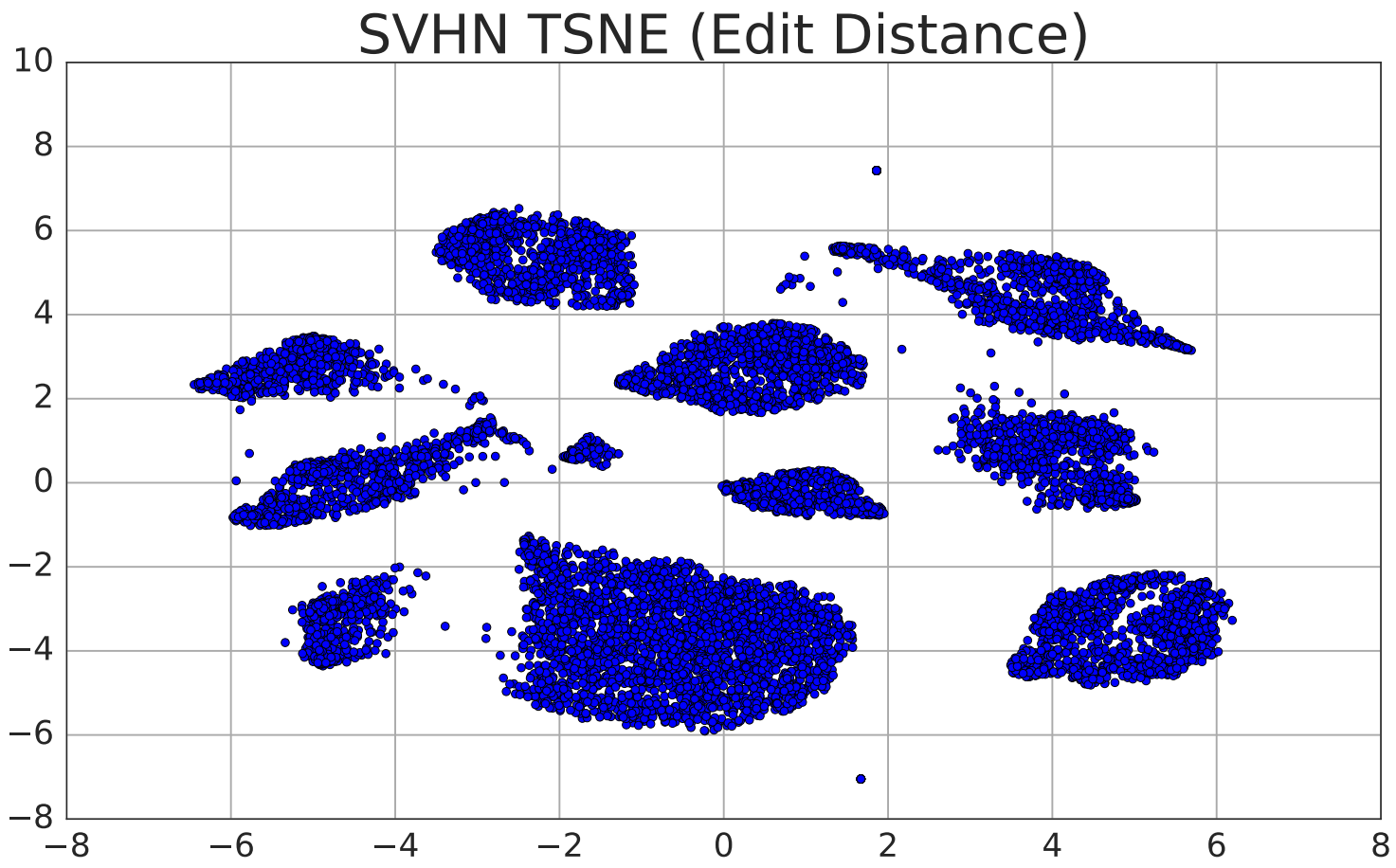




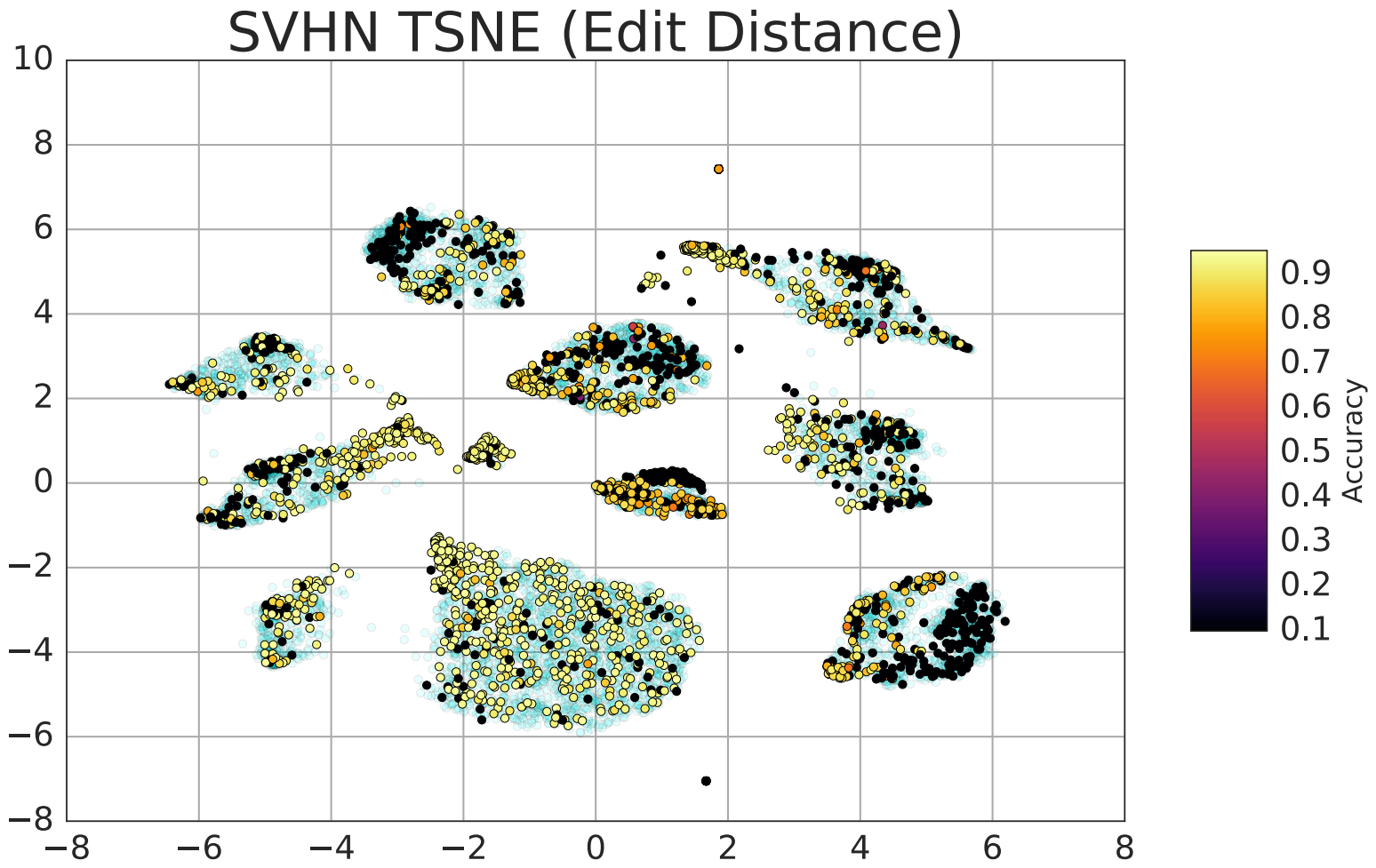
Why Does It Work?

Why Does It Work?

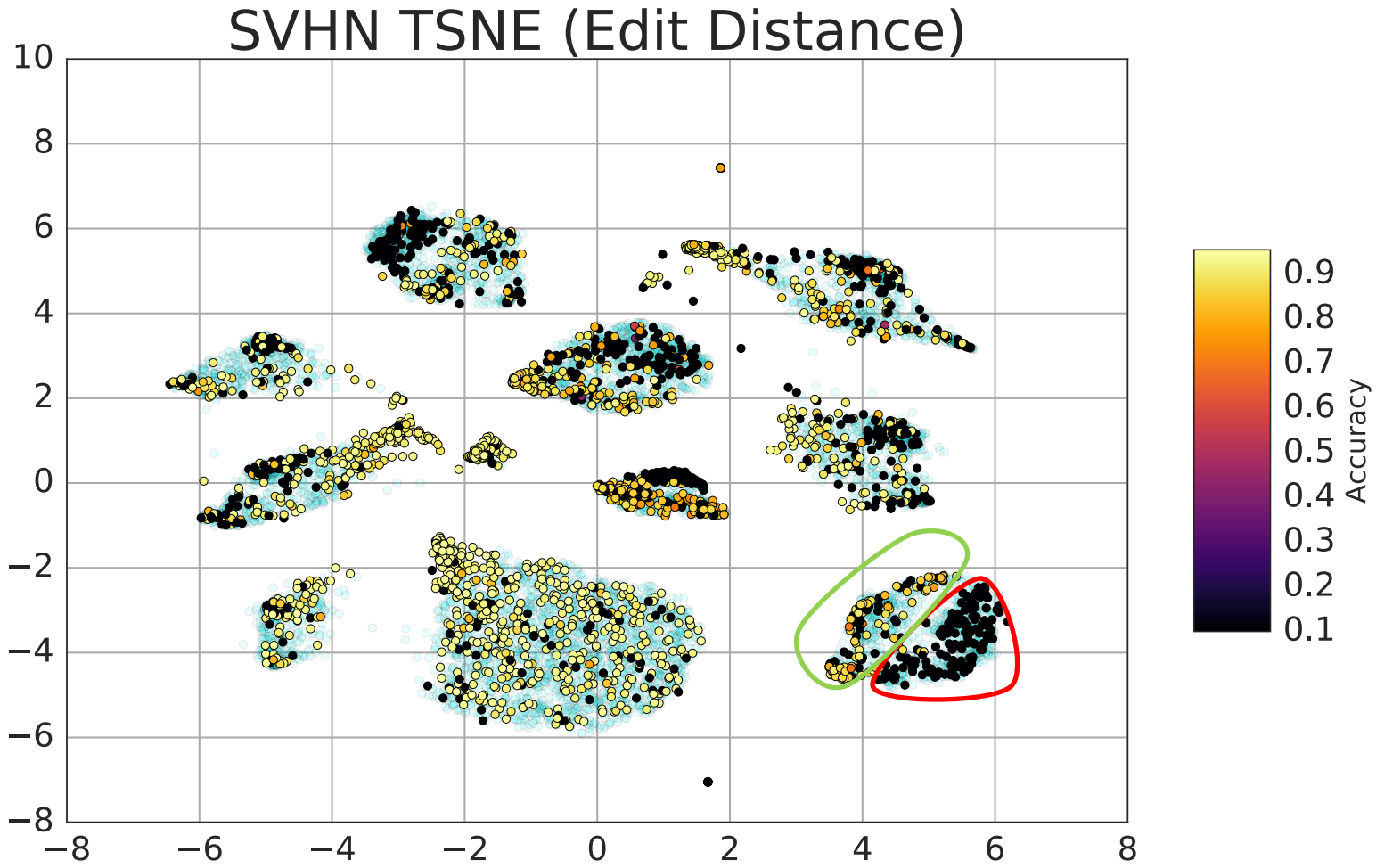
Why Does It Work?



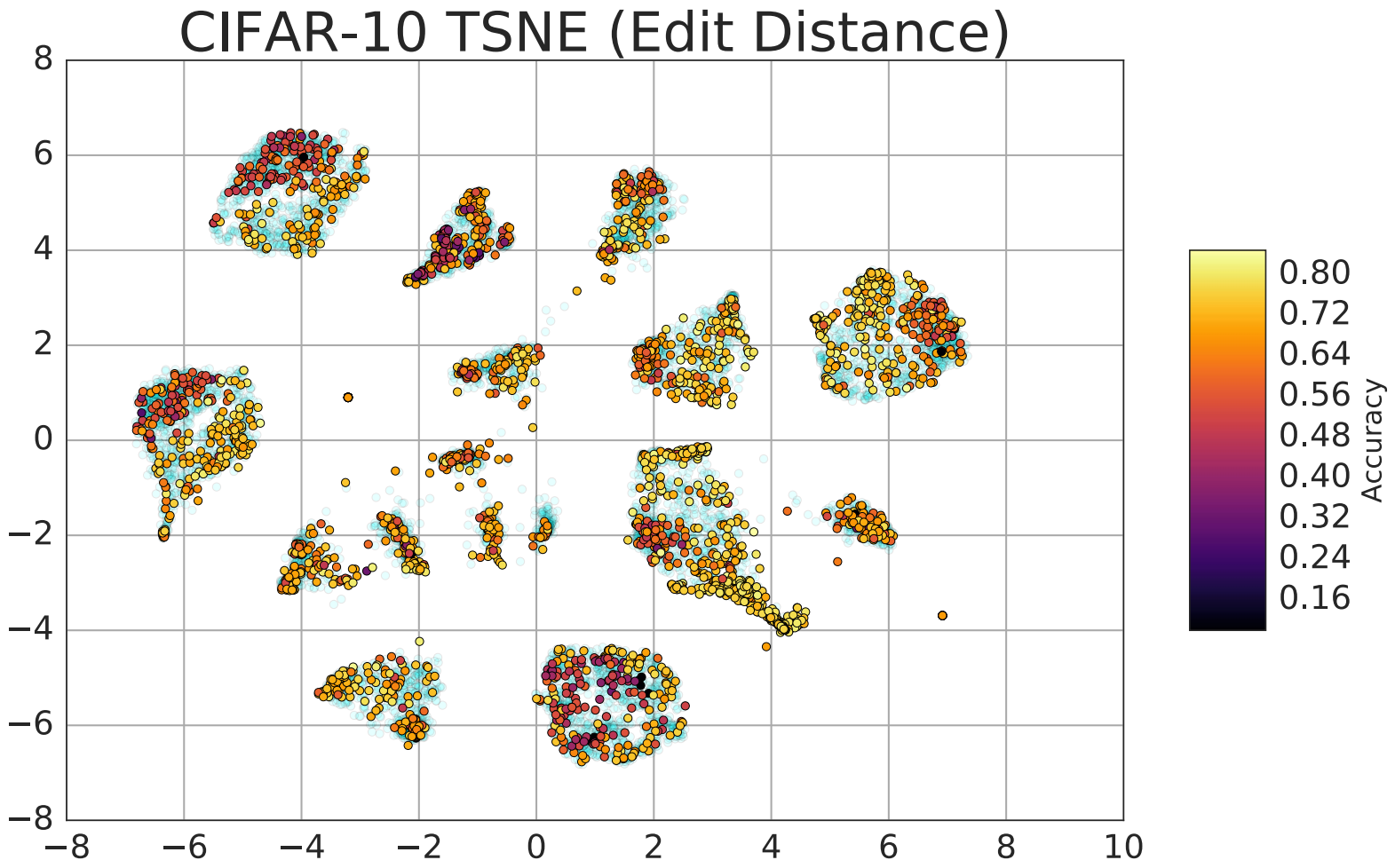
Why Does It Work?



Why Does It Work?



Why Does It Work?

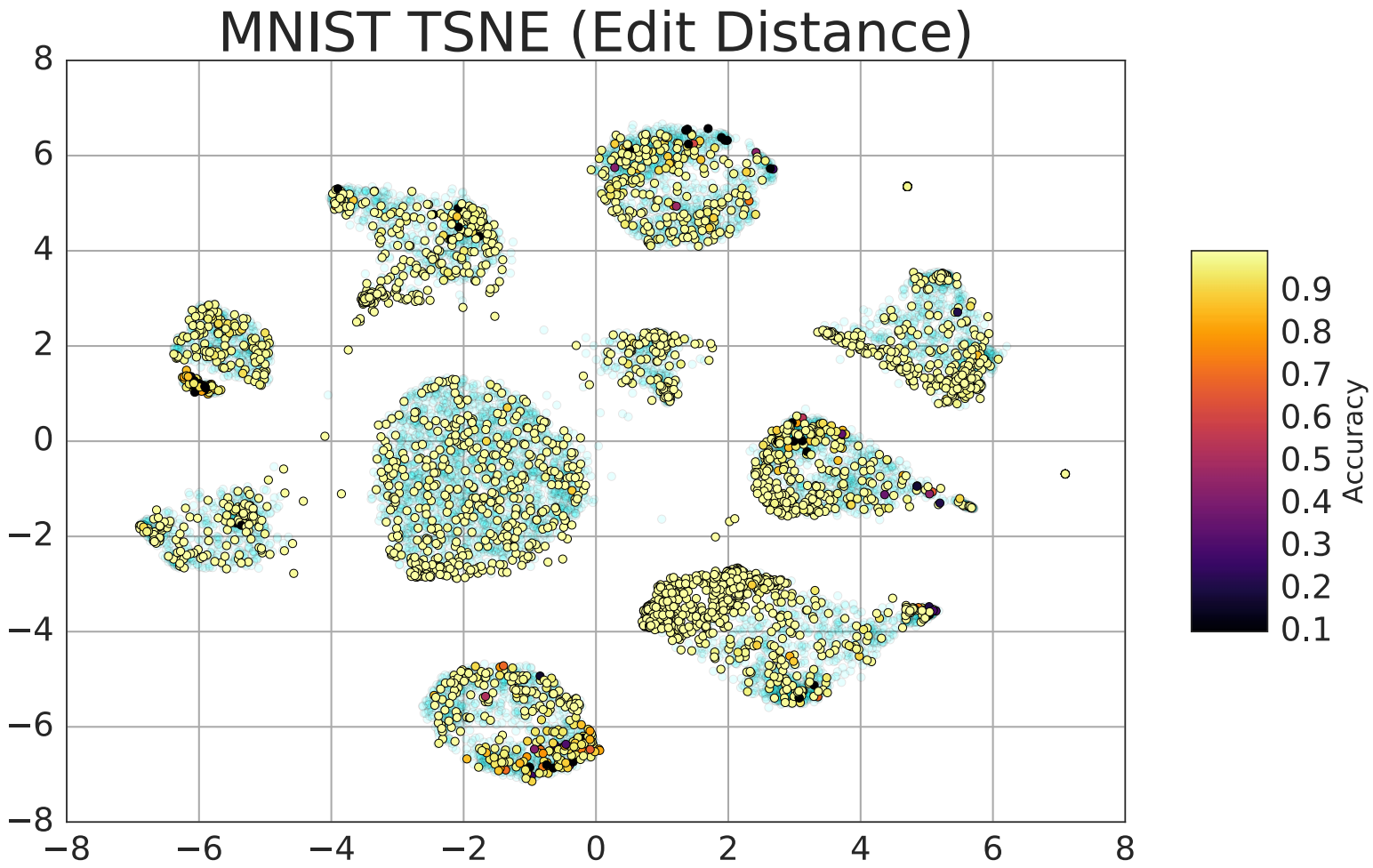


Further Work

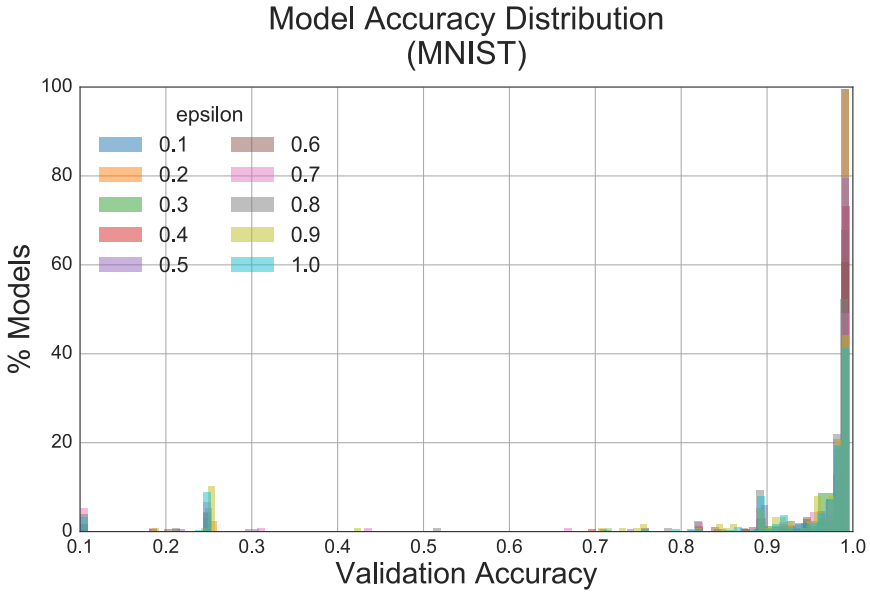
- Q-Value Approximation
- Adding More Layer Types
  - Generalized Pooling
  - Residual Connections
  - Network Splits (e.g. Inception)
- Alternative Network Design
  - Design for inference speed
  - Design for small networks

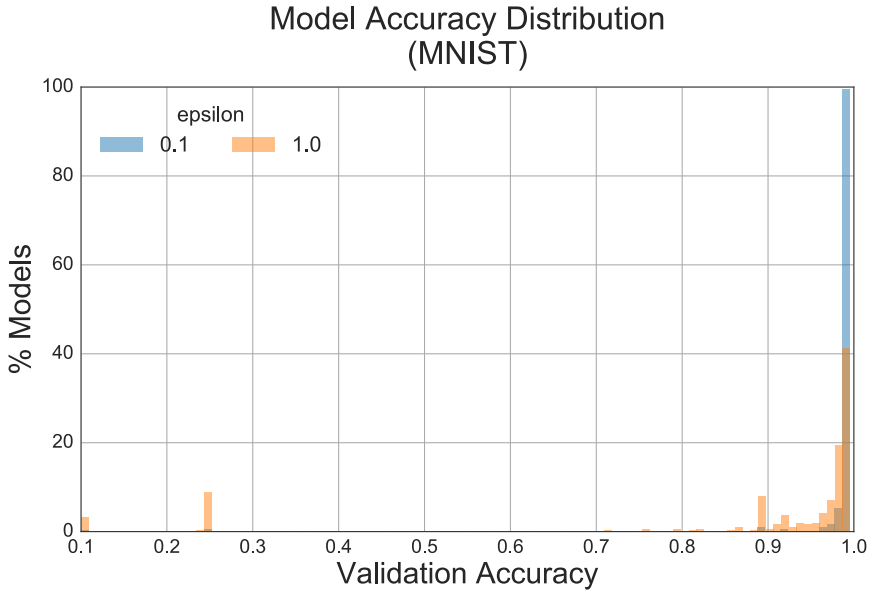
Questions?

MNIST t-SNE

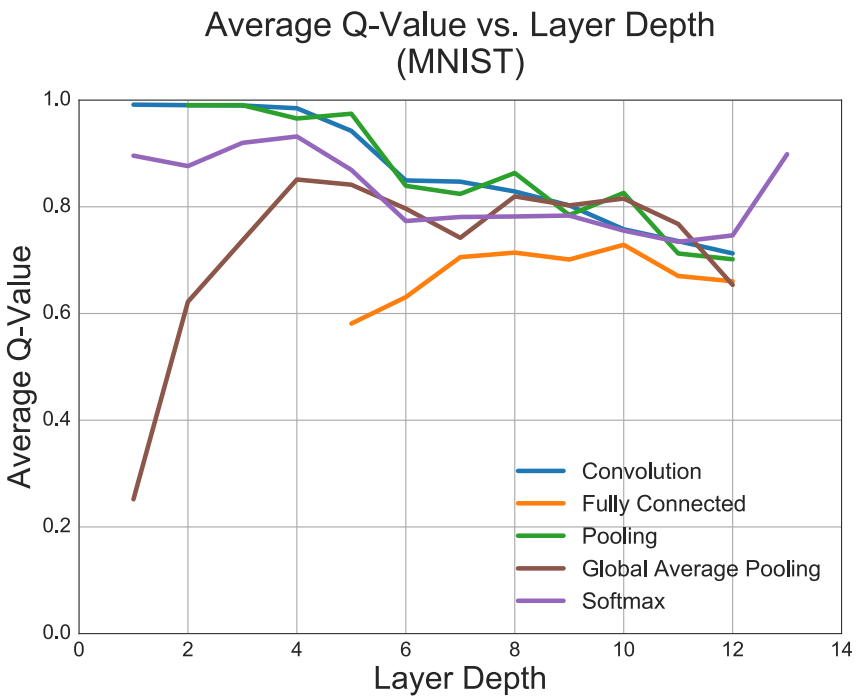

MNIST TSNE (Edit Distance)
8
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6
8
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Accuracy

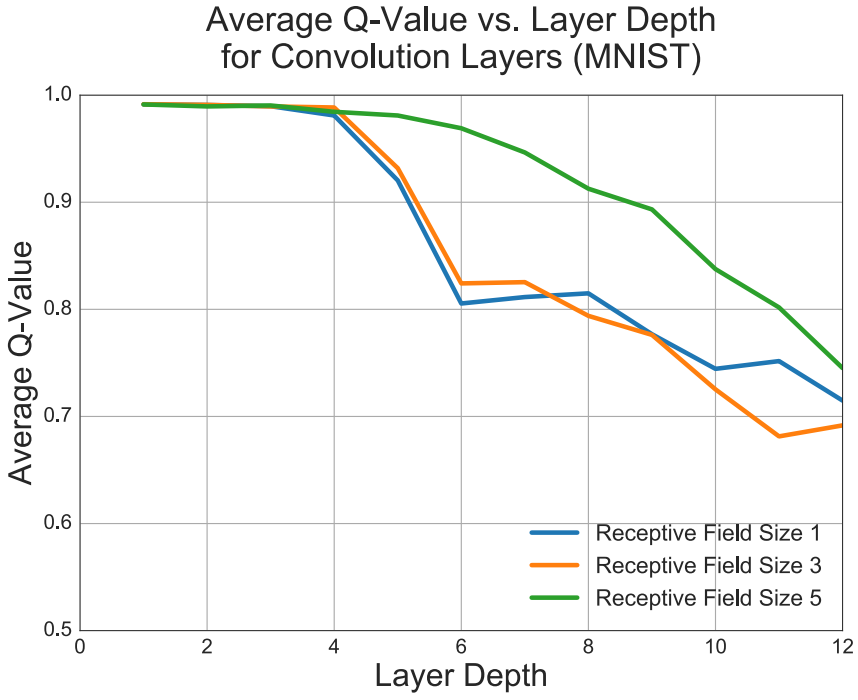
MNIST Exploration Distribution





MNIST Q-Value Analysis





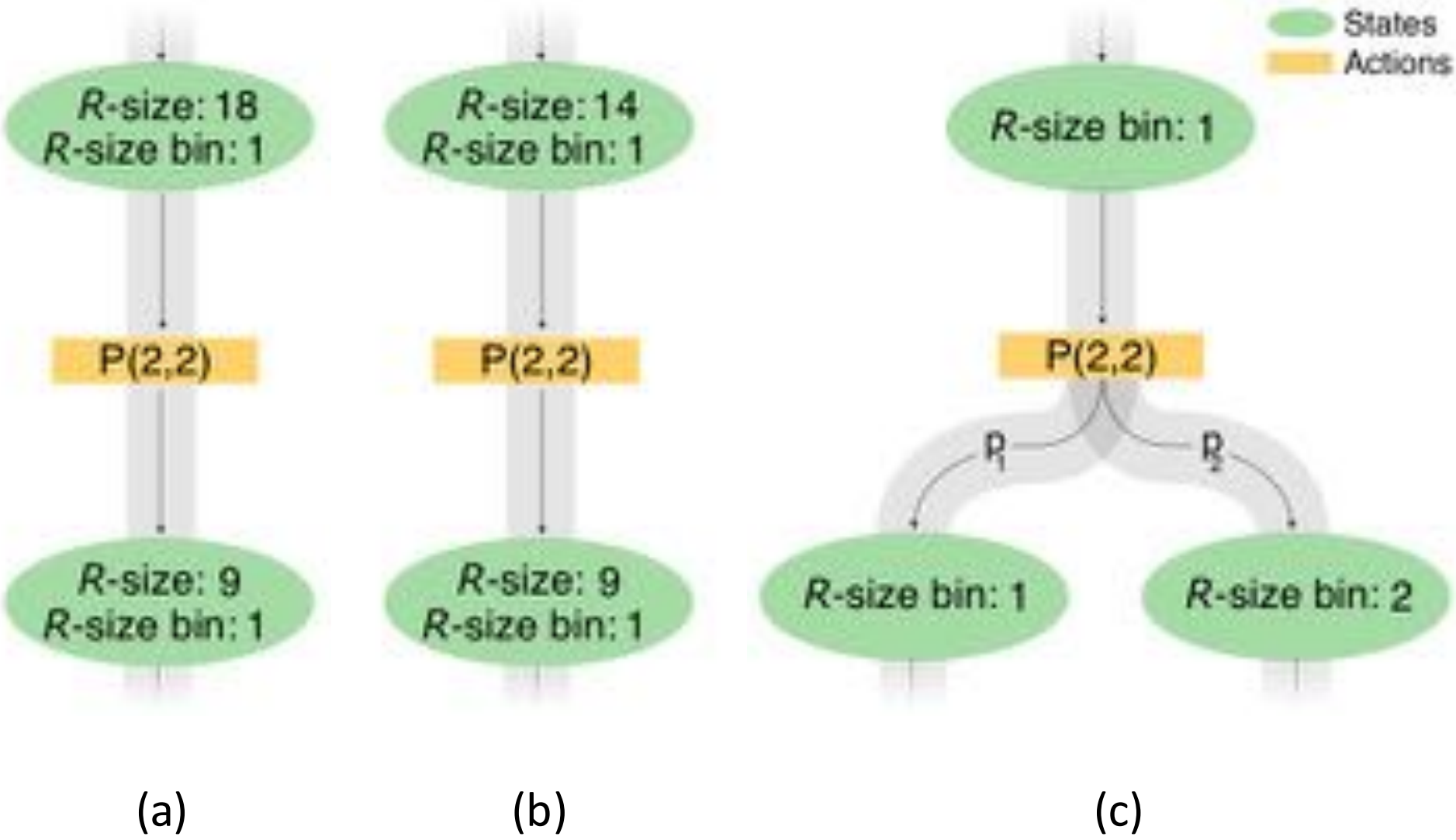
Top Models (SVHN)

Model Architecture	Test Error (%)	# Params (10^6)
[C(128,3,1), P(2,2), C(64,5,1), C(512,5,1), C(256,3,1), C(512,3,1), P(2,2), C(512,3,1), C(256,5,1), C(256,3,1), C(128,5,1), C(64,3,1), SM(10)]	2.24	9.81
[C(128,1,1), C(256,5,1), C(128,5,1), P(2,2), C(256,5,1), C(256,1,1), C(256,3,1), C(256,3,1), C(256,5,1), C(512,5,1), C(256,3,1), C(128,3,1), SM(10)]	2.28	10.38
[C(128,5,1), C(128,3,1), C(64,5,1), P(5,3), C(128,3,1), C(512,5,1), C(256,5,1), C(128,5,1), C(128,5,1), C(128,3,1), SM(10)]	2.32	6.83
[C(128,1,1), C(256,5,1), C(128,5,1), C(256,3,1), C(256,5,1), P(2,2), C(128,1,1), C(512,3,1), C(256,5,1), P(2,2), C(64,5,1), C(64,1,1), SM(10)]	2.35	6.99
[C(128,1,1), C(256,5,1), C(128,5,1), C(256,5,1), C(256,5,1), C(256,1,1), P(3,2), C(128,1,1), C(256,5,1), C(512,5,1), C(256,3,1), C(128,3,1), SM(10)]	2.36	10.05

Top Models (MNIST)

Model Architecture	Test Error (%)	# Params (10^6)
[C(64,1,1), C(256,3,1), P(2,2), C(512,3,1), C(256,1,1), P(5,3), C(256,3,1), C(512,3,1), FC(512), SM(10)]	0.35	5.59
[C(128,3,1), C(64,1,1), C(64,3,1), C(64,5,1), P(2,2), C(128,3,1), P(3,2), C(512,3,1), FC(512), FC(128), SM(10)]	0.38	7.43
[C(512,1,1), C(128,3,1), C(128,5,1), C(64,1,1), C(256,5,1), C(64,1,1), P(5,3), C(512,1,1), C(512,3,1), C(256,3,1), C(256,5,1), C(256,5,1), SM(10)]	0.40	8.28
[C(64,3,1), C(128,3,1), C(512,1,1), C(256,1,1), C(256,5,1), C(128,3,1), P(5,3), C(512,1,1), C(512,3,1), C(128,5,1), SM(10)]	0.41	6.27
[C(64,3,1), C(128,1,1), P(2,2), C(256,3,1), C(128,5,1), C(64,1,1), C(512,5,1), C(128,5,1), C(64,1,1), C(512,5,1), C(256,5,1), C(64,5,1), SM(10)]	0.43	8.10

Representation Size



(a) (b) (c)

Deep Neural Network Performance Prediction

We experiment with five main techniques:

1. Linear Least Squares (LLS)
2. Kernel Least Squares with RBF Kernel (RBF-LS)
3. Kernel Least Squares with Edit Distance Kernel (ED-LS)
4. Manifold Regularization with RBF K-NN (RBF-MR)
5. Manifold Regularization with Edit Distance Kernel (ED-MR)

Method	**CIFAR-10 L2 Loss**	**SVHN L2 Loss**	**MNIST L2 Loss**	**CIFAR-10 R^2**	**SVHN R^2**	**MNIST R^2**
LLS	0.00613	0.05713	0.01578	0.59207	0.60488	0.13619
RBF-LS	0.00658	0.06844	0.12050	0.53945	0.47971	-0.34912
ED-LS	0.00733	0.06940	0.01374	0.47528	0.47393	-1.09552
RBF-MR	0.01122	0.11132	0.01499	0.24544	0.23255	0.15724
ED-MR	0.01098	0.10317	0.01549	0.27045	0.28662	0.11671

Regression as a Meta-Modeling Approach

- Generate 10,000 untrained architectures
- Take the top 10 models predicted by each model

Method	CIFAR-10 L2 Loss	SVHN L2 Loss	MNIST L2 Loss
LLS	0.13347	0.15280	0.15281
RBF-LS	0.00657	0.09233	0.00016
ED-LS	1.49888	2185.21	0.00016
RBF-MR	0.00629	0.00440	0.00005
ED-MR	0.02607	0.00127	0.05403