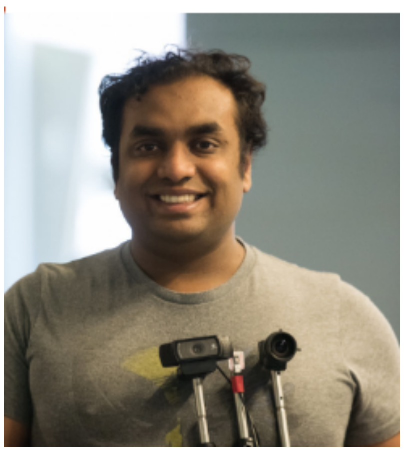
Practical Neural Network Design Using Reinforcement Learning

Bowen Baker
Media Lab
bowen@mit.edu

Co-authors



Otkrist Gupta
MIT Media Lab

Nikhil Naik
Harvard

Ramesh Raskar
MIT Media Lab

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

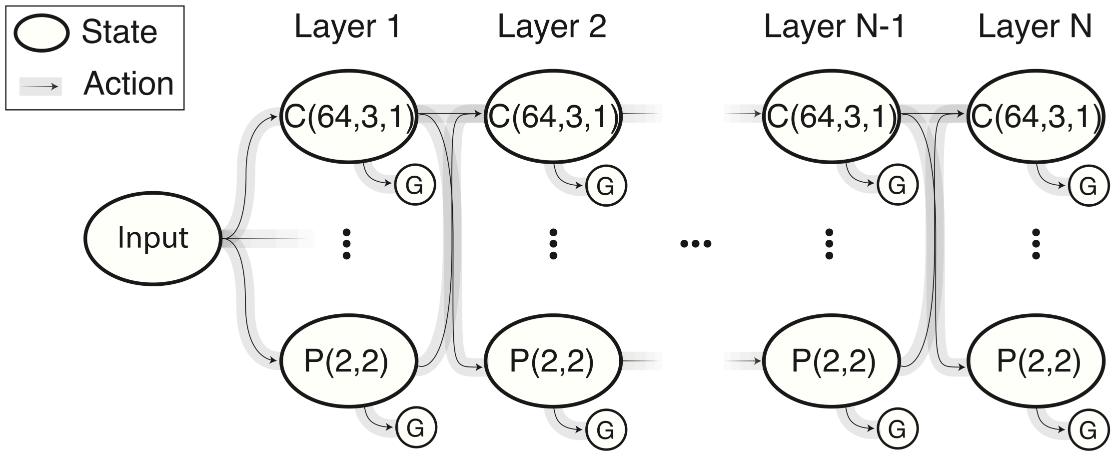
- Neural network design is still hand crafted.
- Despite the wide usage of a few main networks, we may want networks specialized for specific tasks.
- We may want more than 1 specialized model, e.g. for the ensembling purposes.

Automating Tasks With Reinforcement Learning

Outline

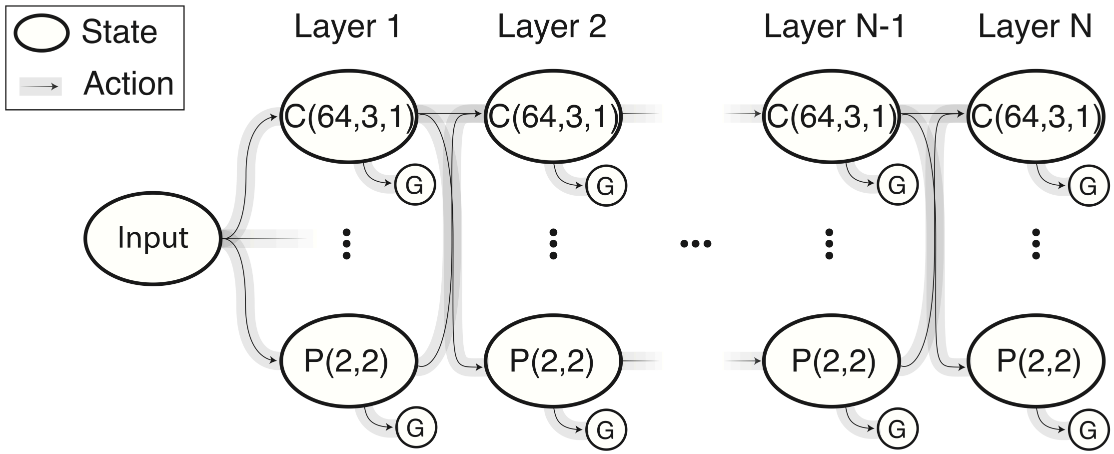
1. Modeling Architecture Selection as a Markov Decision Process
2. Reinforcement Learning Background
3. Results with Q-Learning
4. Accelerating Architecture Selection with Simple Early Stopping Algorithms

Modeling Architecture Selection as a Markov Decision Process



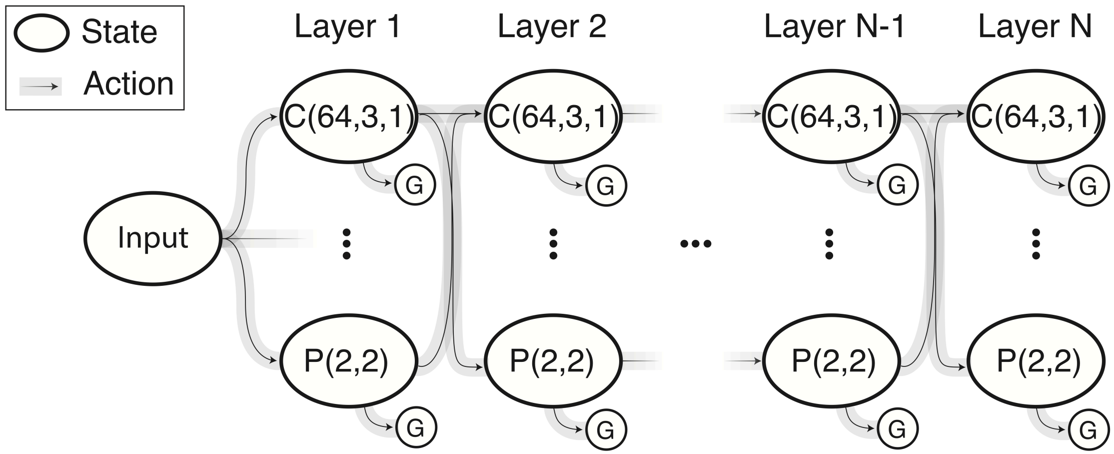
- C(64,3,1) – Convolutional Layer with 64 learnable kernels, 3x3 kernel size, and stride of 1

Modeling Architecture Selection as a Markov Decision Process



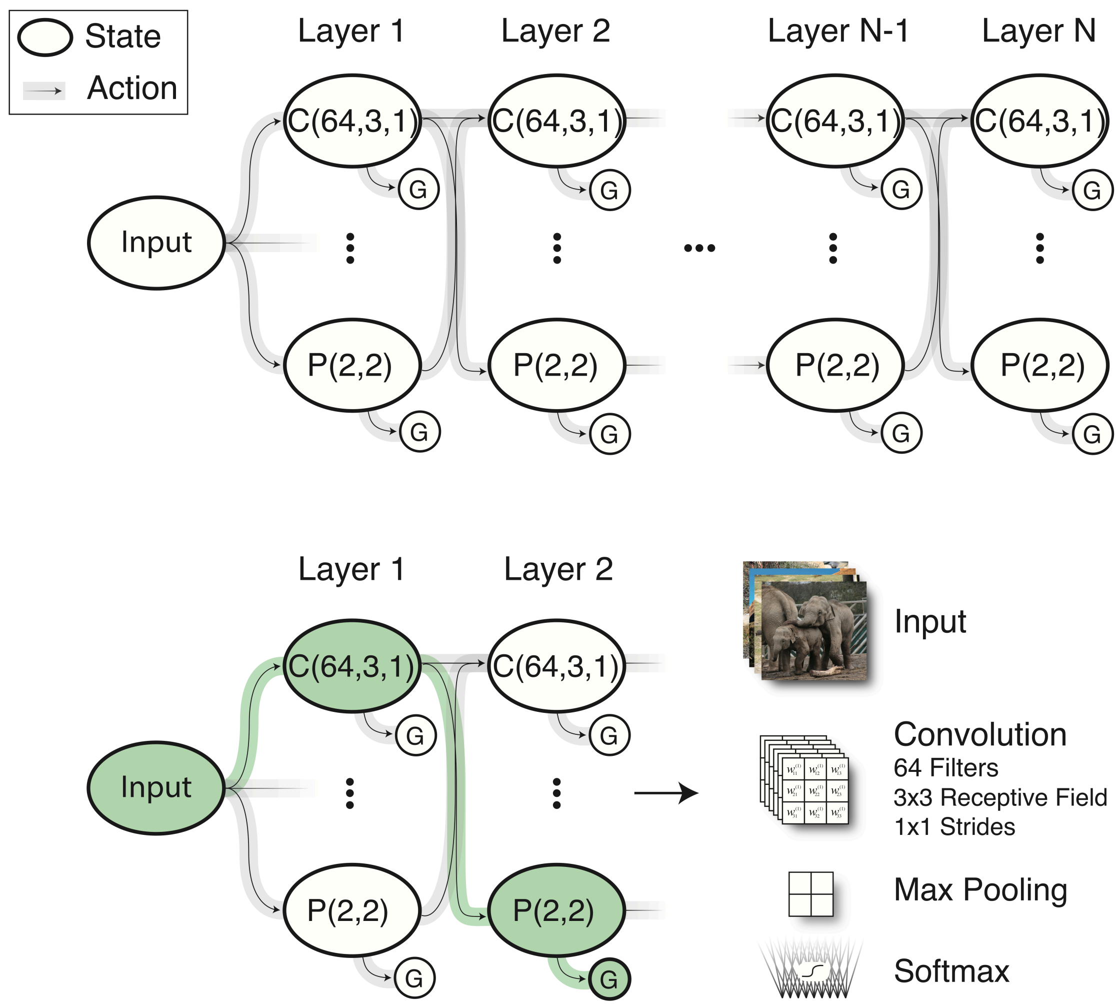
- C(64,3,1) – Convolutional Layer with 64 learnable kernels, 3x3 kernel size, and stride of 1
- P(2,2) – Max Pooling Layer with 2x2 kernel size and stride 2

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



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

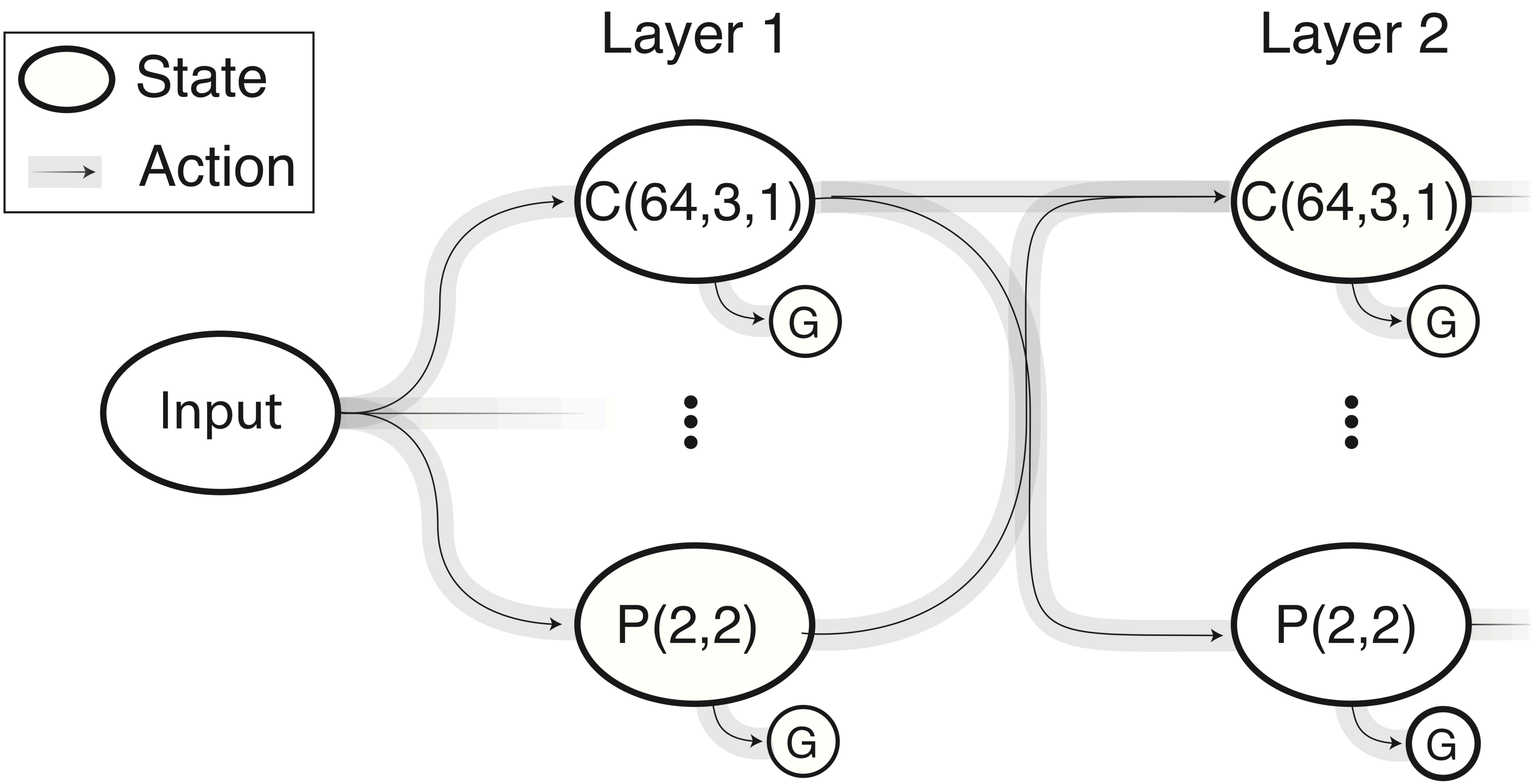
r -- Reward received from the (s_i, u, s_j) transition

Q-Learning

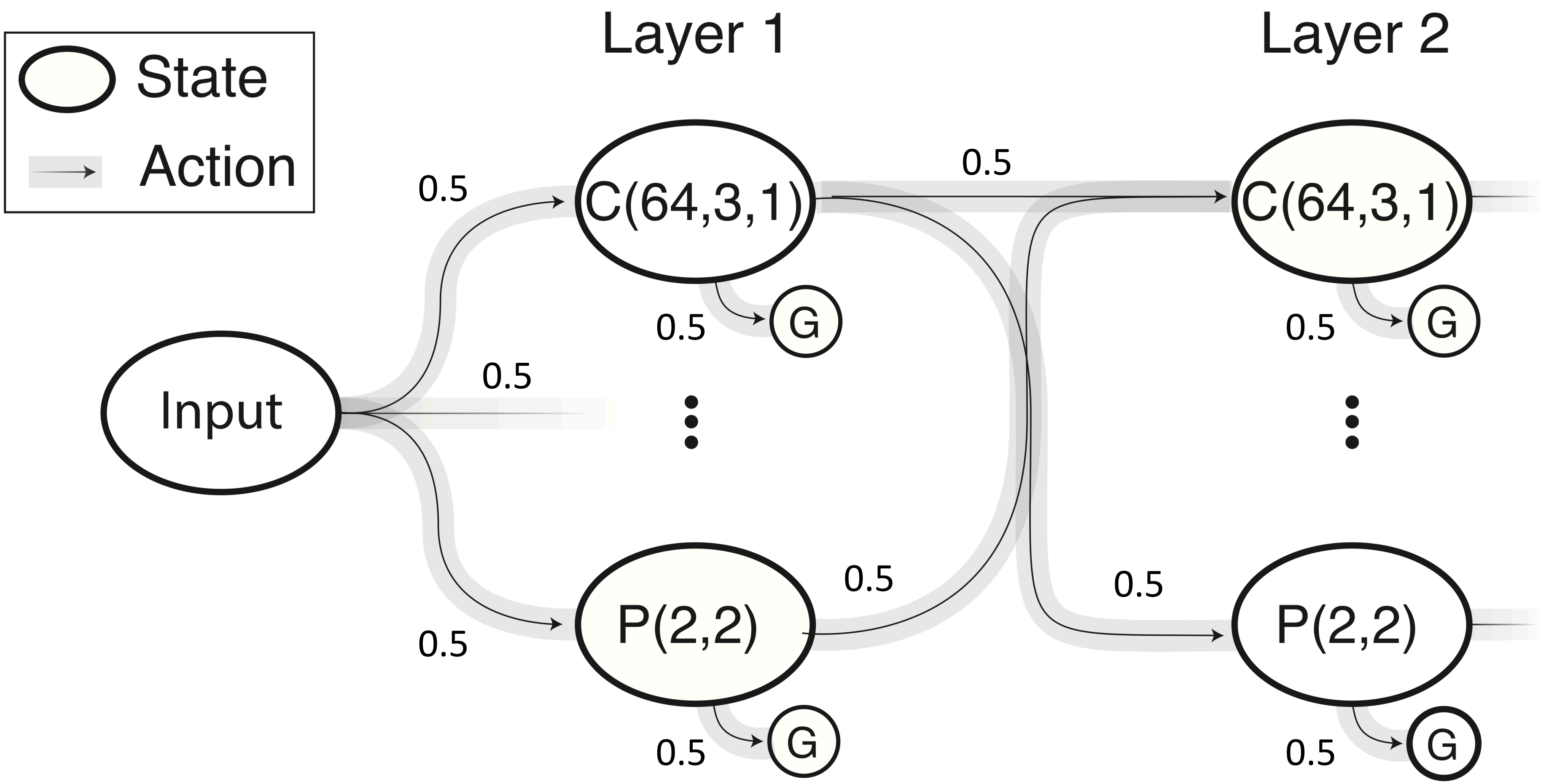
$$Q^*(s_i, u) = \mathbb{E}\left[r + \gamma \max_{u' \in \mathcal{U}(s_j)} Q^*(s_j, u')\right]$$

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

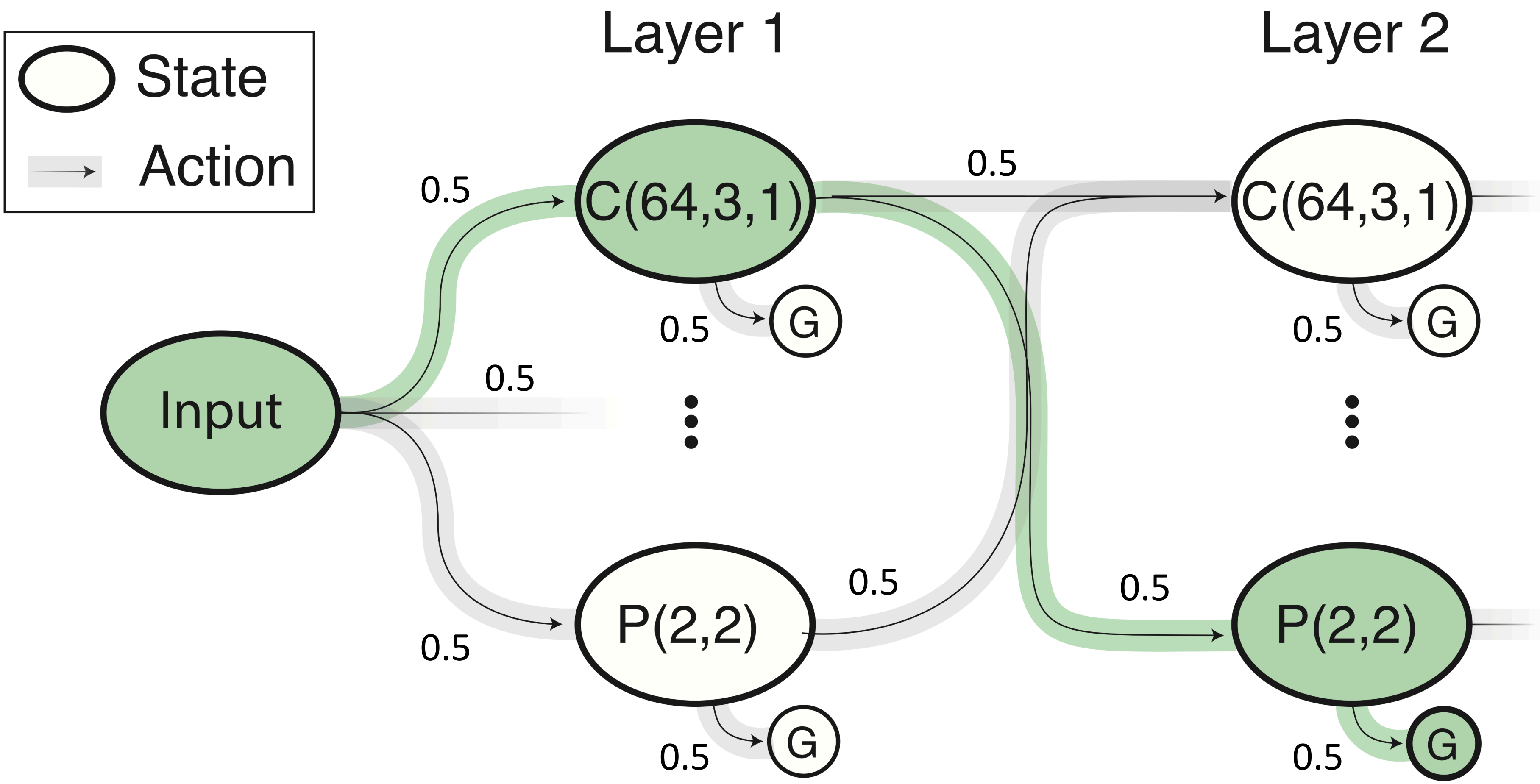
Q-Value Update (Example)



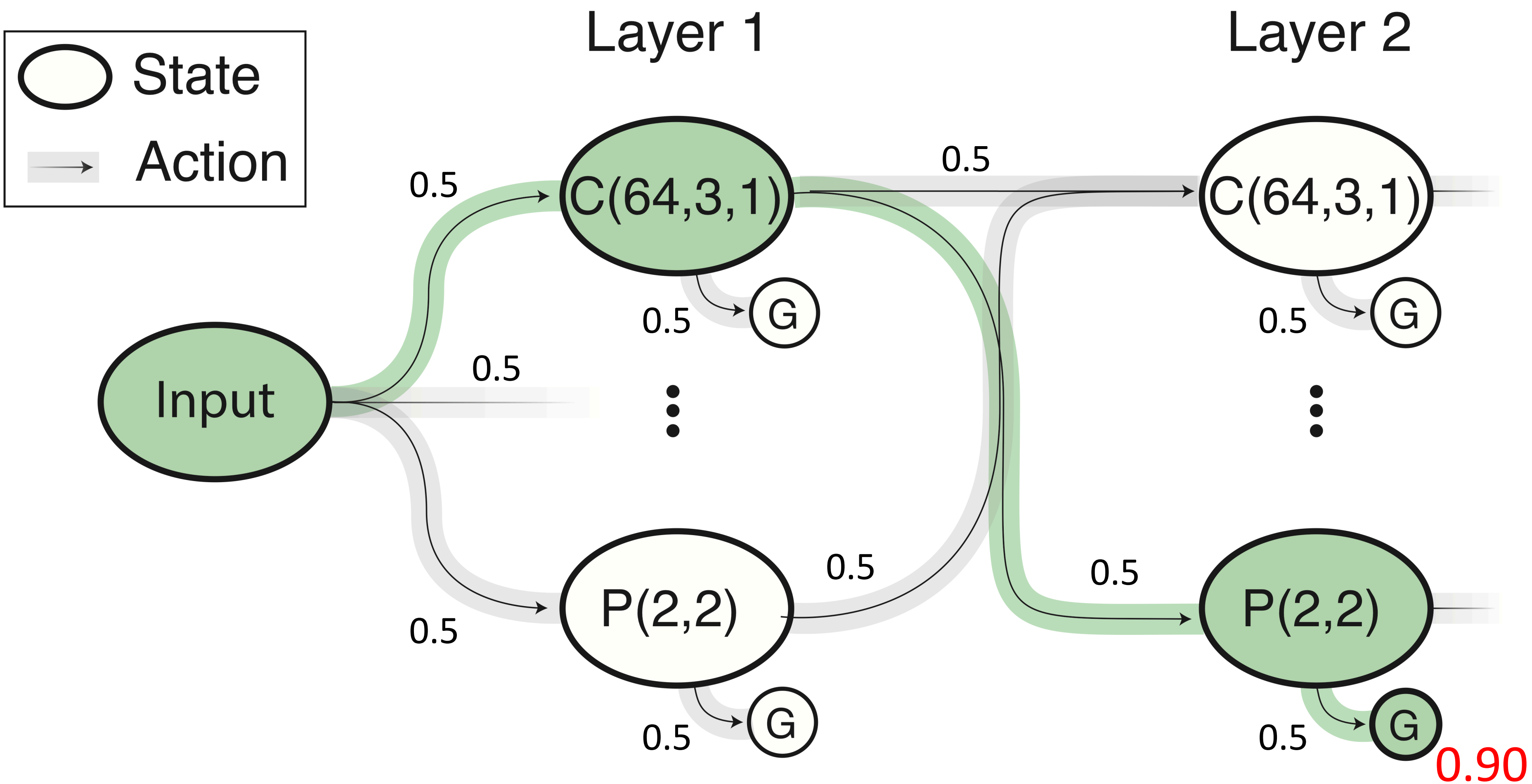
Q-Value Update (Example)



Q-Value Update (Example)

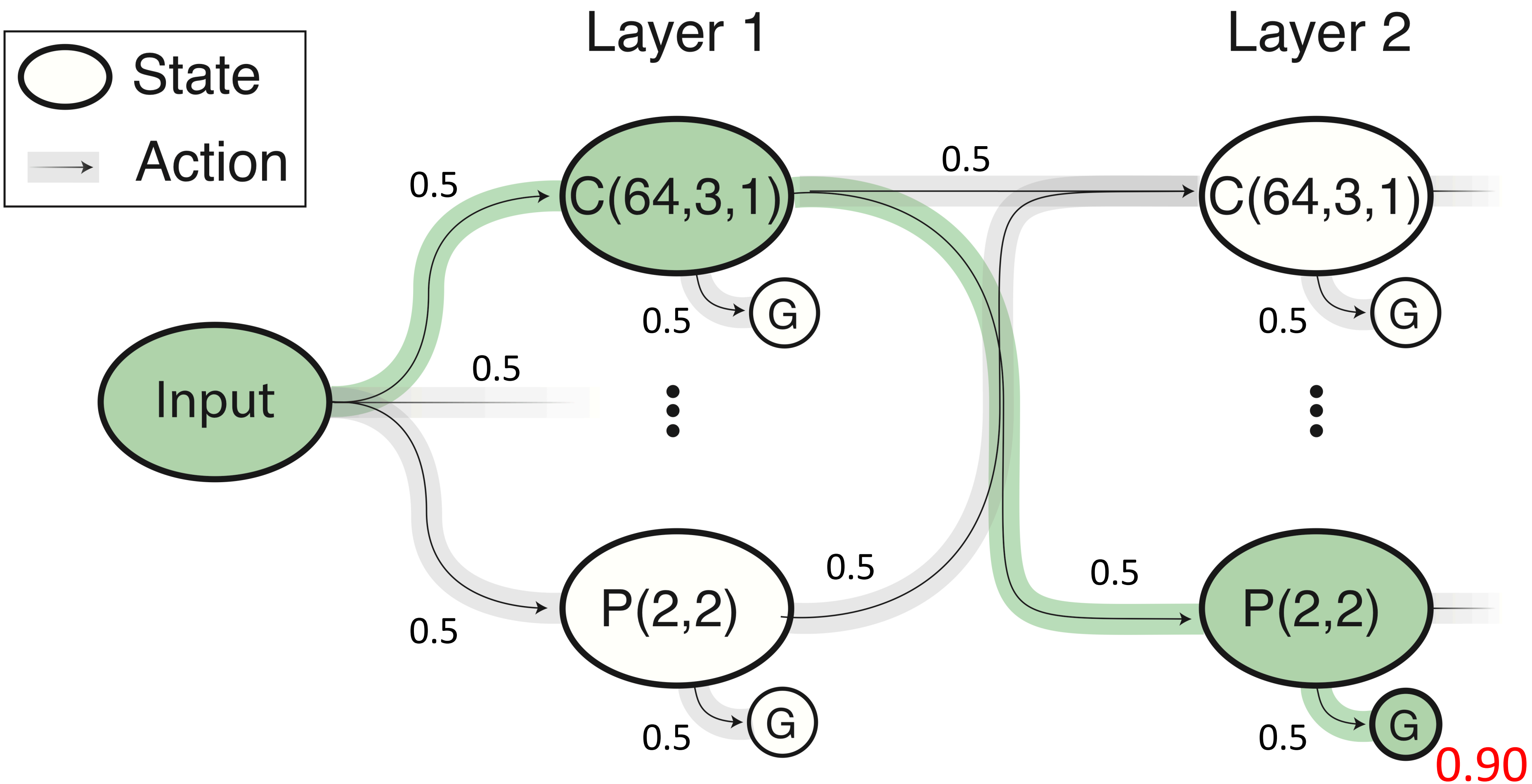


Q-Value Update (Example)



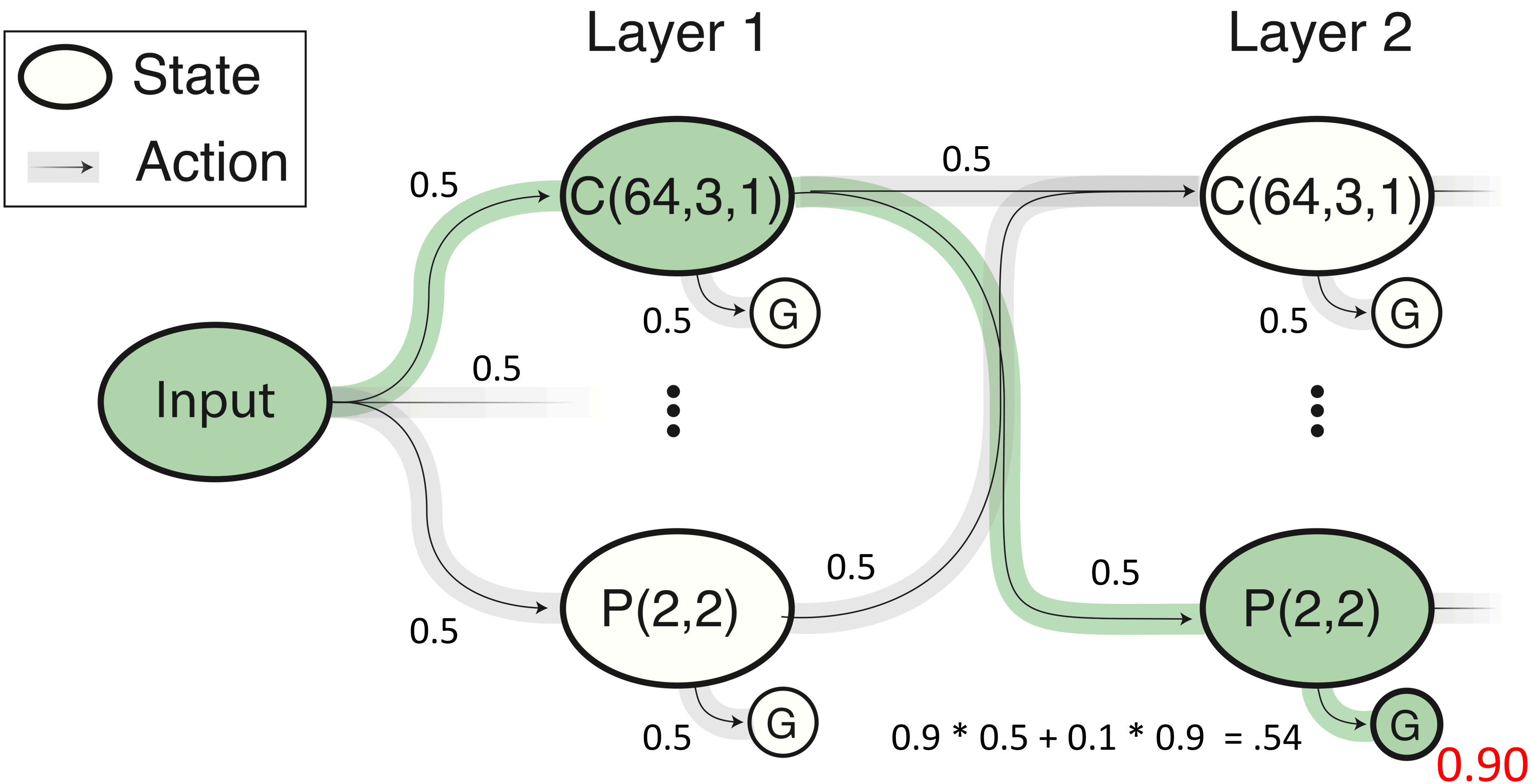
Q-Value Update (Example)

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



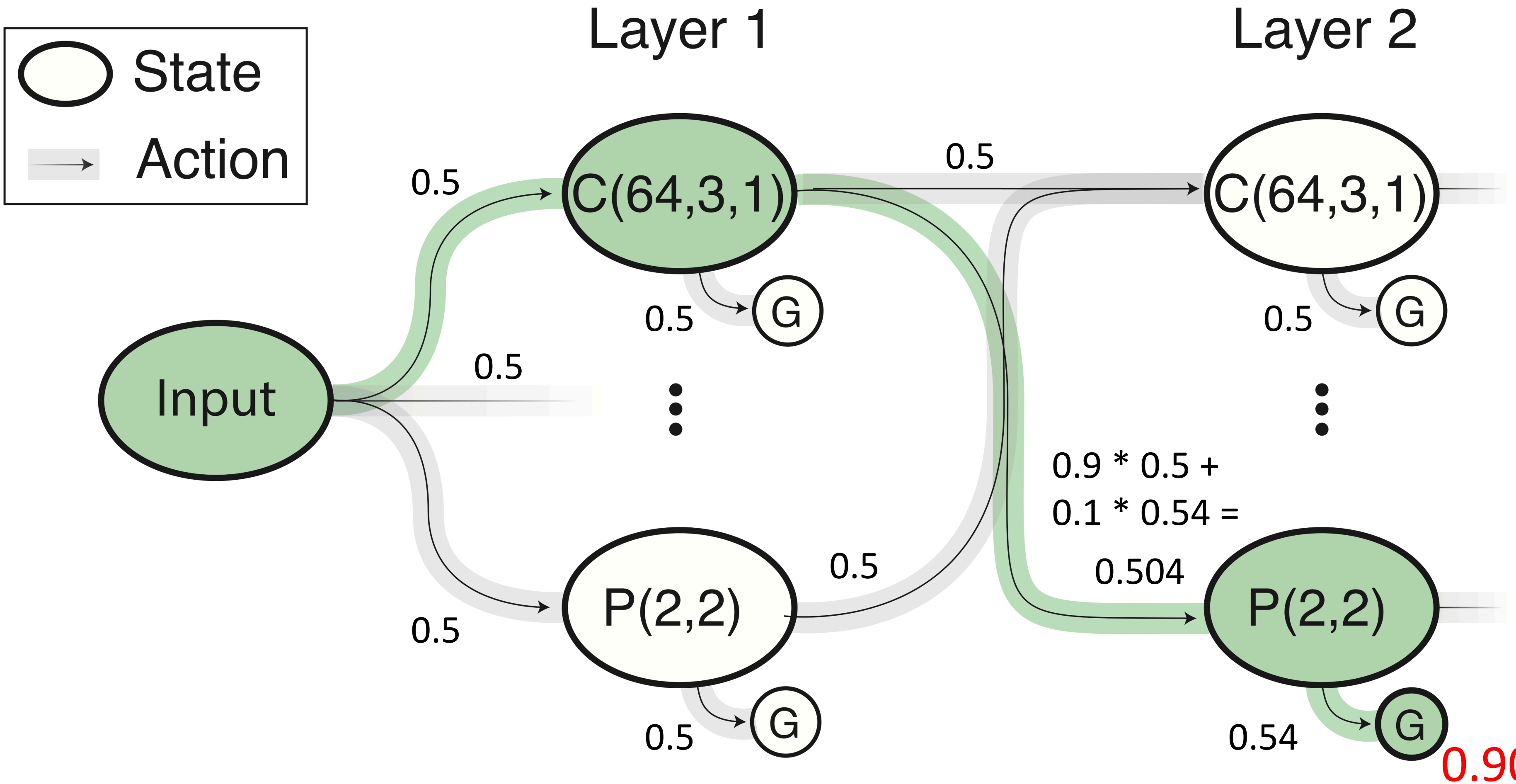
Q-Value Update (Example)

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



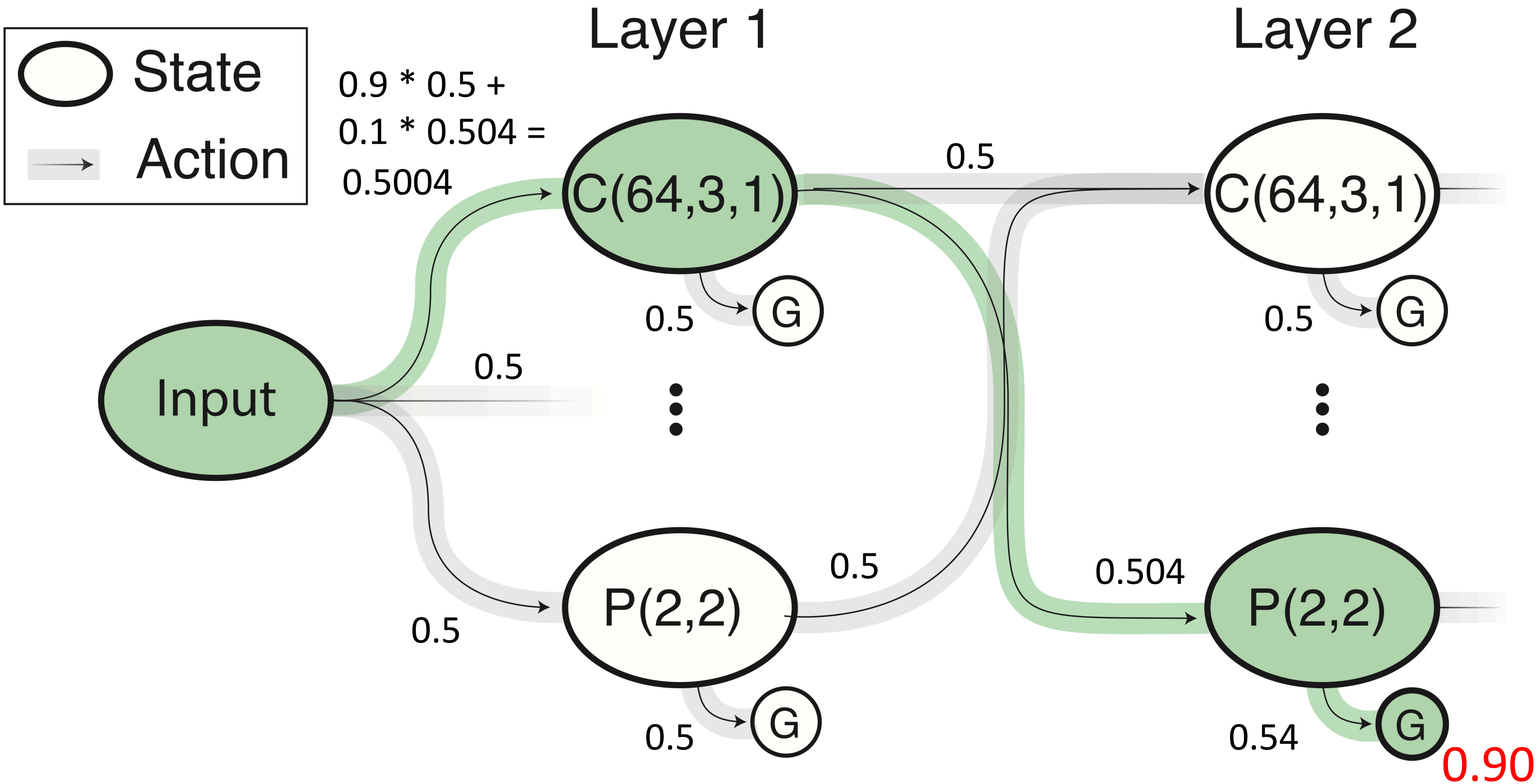
Q-Value Update (Example)

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

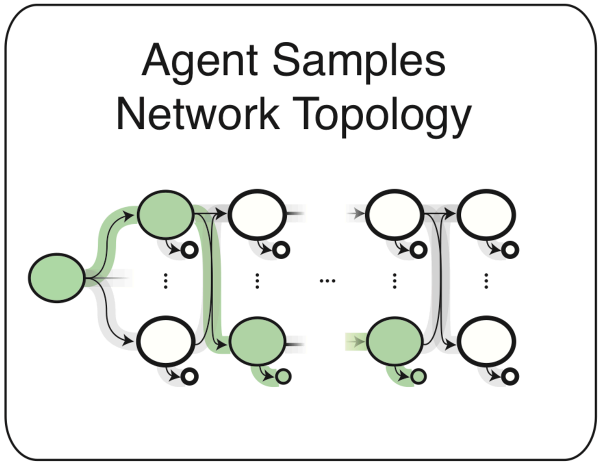


Q-Value Update (Example)

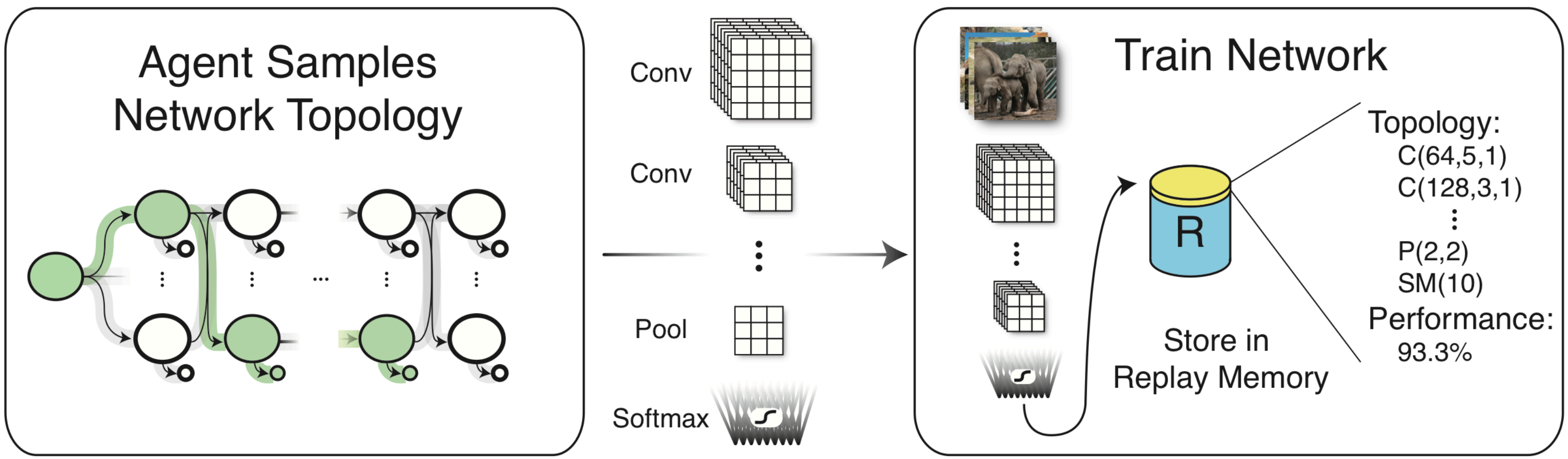
$$Q_{t+1}(s_i, u) = (1-\alpha)Q_t(s_i, u) + \alpha \left[r_t + \gamma \max_{u' \in \mathcal{U}(s_j)} Q_t(s_j, u') \right], \quad \gamma = 1, \quad \alpha = 0.1$$



MetaQNN

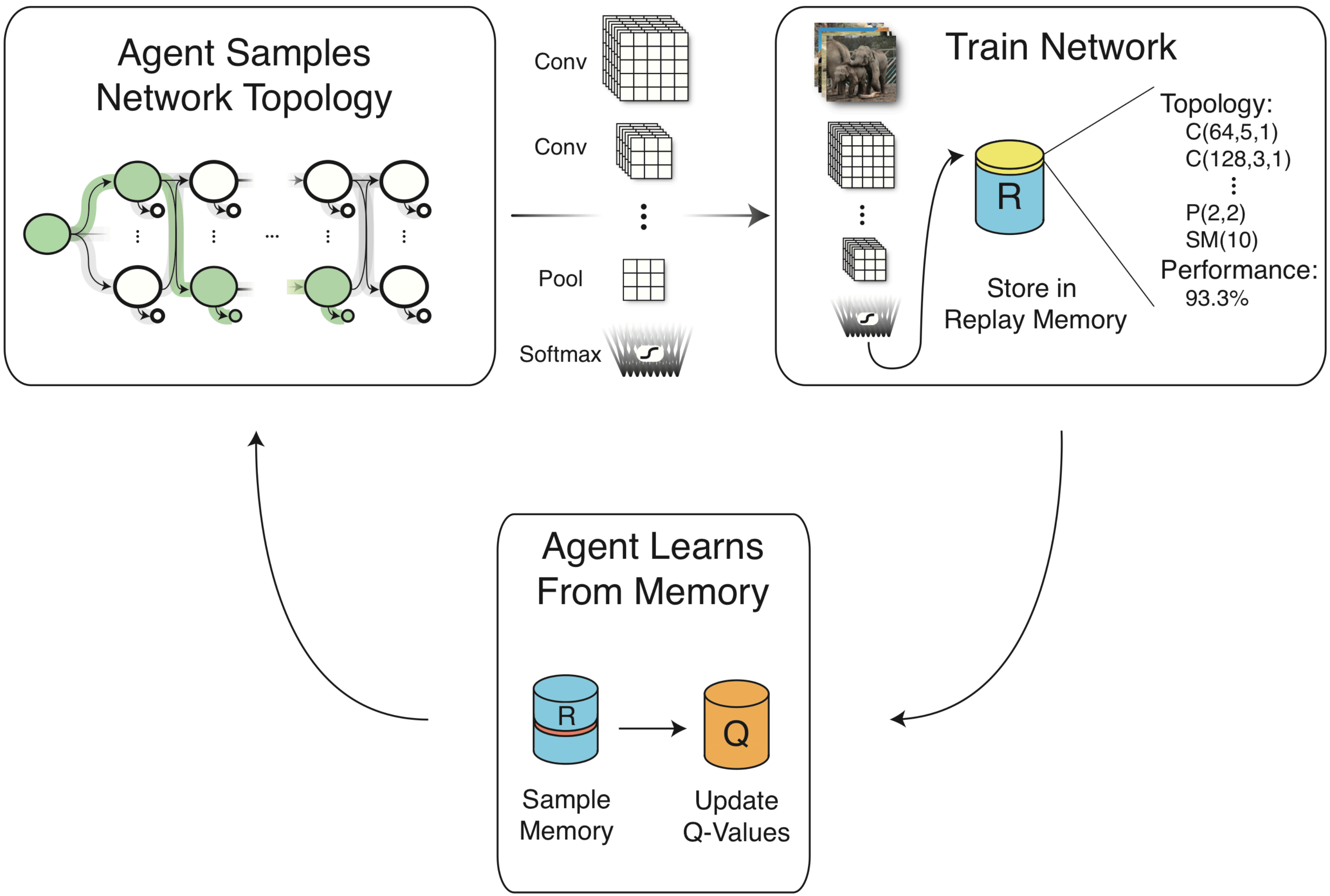


MetaQNN



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
Topology:
C(64,5,1)
C(128,3,1)
P(2,2)
SM(10)
Performance:
93.3%
R
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

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

Sampling Networks

Epsilon-Greedy Exploration:

– State *s* corresponds the last layer chosen

– Action *u* corresponds to the next layer chosen

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

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

State Space

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

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$\boxed{i \sim \text{Layer depth}}$ $f \sim$ 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)	$\boxed{i \sim \text{Layer depth}}$ $(f, \ell) \sim$ (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)	$\boxed{i \sim \text{Layer depth}}$ $n \sim$ # 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

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth $\boxed{f \sim \text{Receptive field size}}$ $\ell \sim$ 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 $\boxed{(f, \ell) \sim \text{(Receptive field size, Strides)}}$ $n \sim$ 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

Layer Type	Layer Parameters	Parameter Values
Convolution (C)	$i \sim$ Layer depth $f \sim$ Receptive field size $\boxed{\ell \sim \text{Stride}}$ $d \sim$ # 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 $\boxed{(f, \ell) \sim \text{(Receptive field size, Strides)}}$ $n \sim$ 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

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

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

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 $\boxed{d \sim \text{\# neurons}}$	< 12 < 3 $\in \{512, 256, 128\}$
Termination State	$s \sim$ Previous State $t \sim$ Type	Global Avg. Pooling/Softmax

State Space

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	$\boxed{s \sim \text{Previous State}}$ $t \sim$ Type	 Global Avg. Pooling/Softmax

State Space

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

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

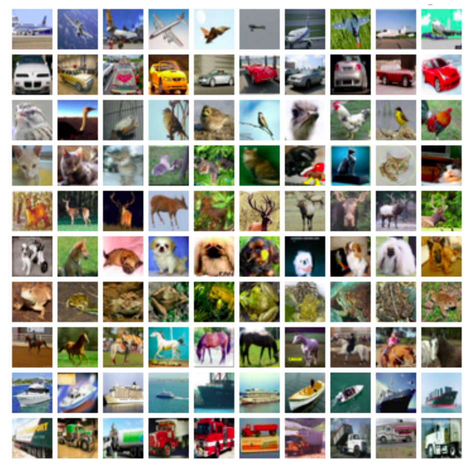
- Convolution → Any Other Layer
- Pooling → Any Other Layer / Pooling
- Any Layer → Fully Connected
  - if representation size less than 8
- Any Layer → Termination

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

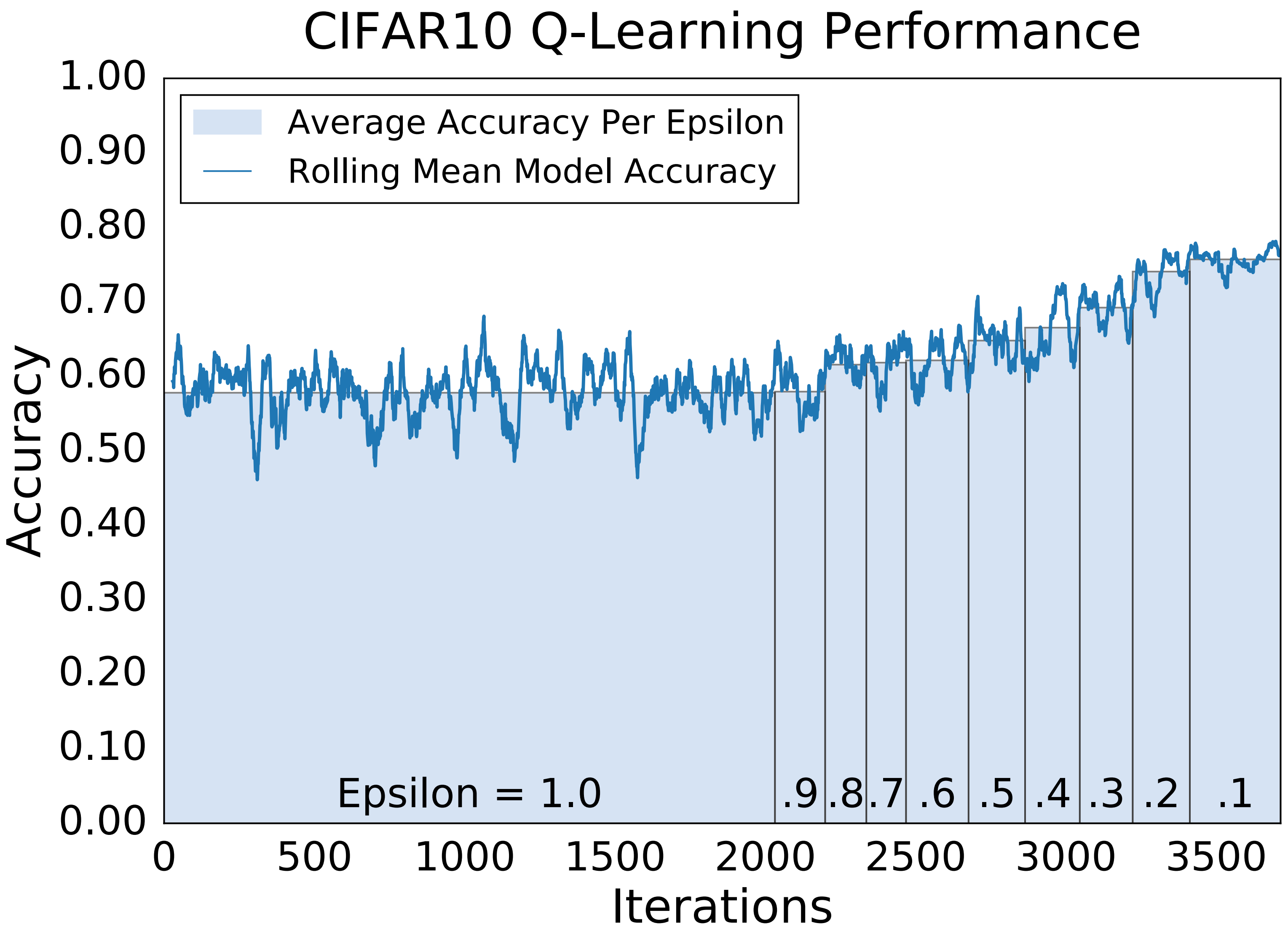
SVHN

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

Hardware

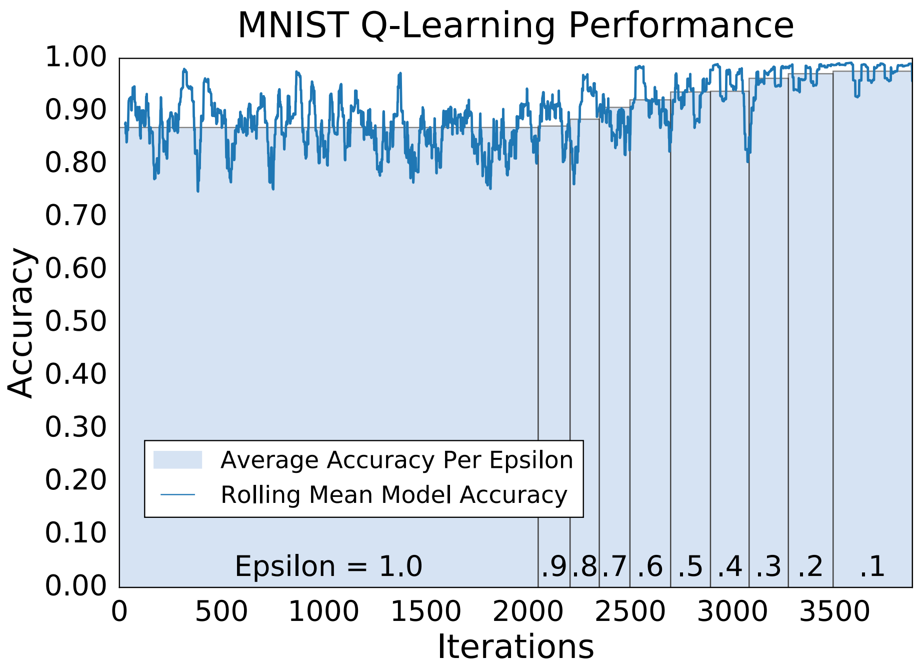
- ~10 GPU's
  - Mostly 2015 Titan Xs
  - Some GTX 1080s
- Each experiment took ~10 days
  - Roughly 100 GPUdays

Results

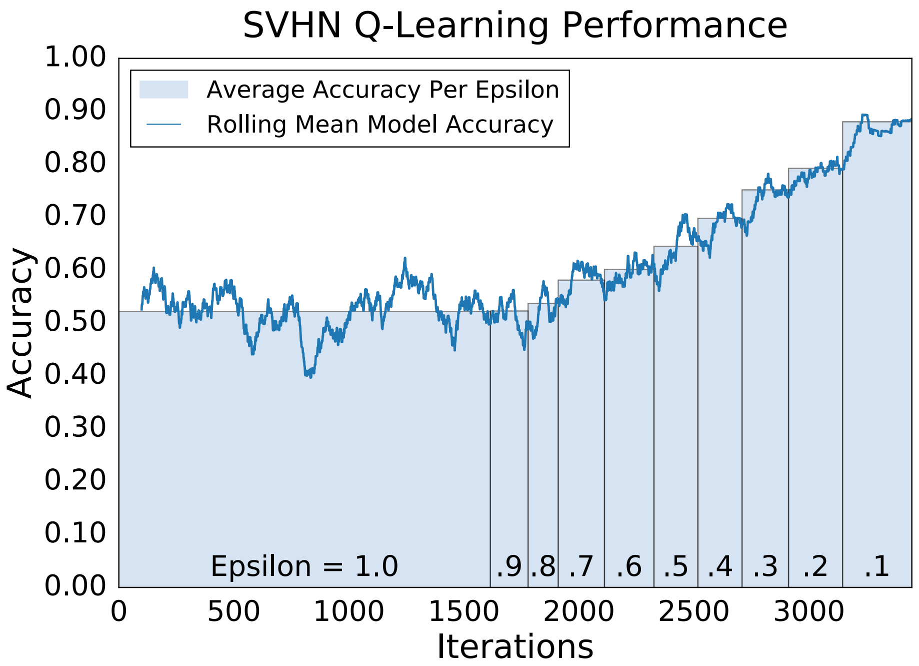


CIFAR10 Q-Learning Performance
Average Accuracy Per Epsilon
Rolling Mean Model Accuracy
Accuracy
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
Epsilon = 1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
500
1000
1500
2000
2500
3000
3500
Iterations

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
0.00 0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90 1.00
0 500 1000 1500 2000 2500 3000 3500



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
0.00 0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90 1.00
0 500 1000 1500 2000 2500 3000

Results

Comparison Against Models with similar design modules:

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

Comparison Against Models with similar design modules:

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***

Comparison Against more complex modules:

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	-	-	-
Resnet(1001) (He et al., 2016)	**4.62**	-	-	**22.71**
ELU (Clevert et al., 2015)	6.55	-	-	24.28
Tree+Max-Avg (Lee et al., 2016)	6.05	**1.69**	**0.31**	32.37

Meta-Modeling Comparison on CIFAR-10

Method	Test Error on CIFAR-10	# Samples	Estimated Computation (GPU-Days)
MetaQNN (Ours)	6.92	2,700	100
Neural Architecture Search (Zoph et al., 2016)	3.65	12,800	10,000
Large Scale Evolution (Real et al., 2017)	5.4	-	2,600
Bayesian Optimization (Snoek et al., 2012)	9.5	50	-

Updated Results:
Different Model Depths Don’t Train Equally

Model Depth	20 Epoch Accuracy	300 Epoch Accuracy
9	84.78	93.08
15	81.2	94.7

Updated Results:

Different Model Depths Don't Train Equally

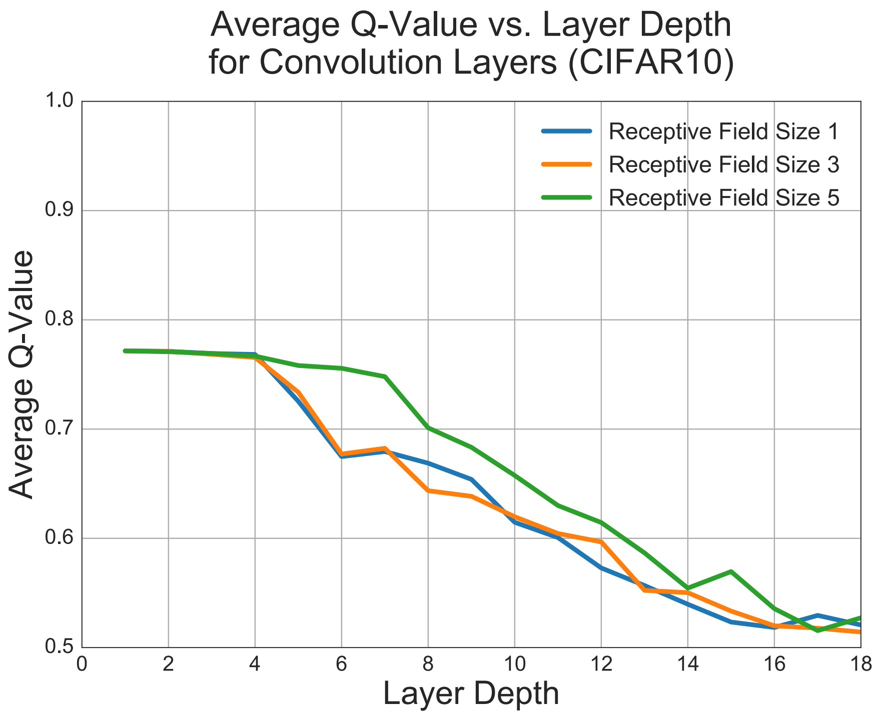
Model Depth	20 Epoch Accuracy	300 Epoch Accuracy
9	84.78	93.08
15	81.2	94.7

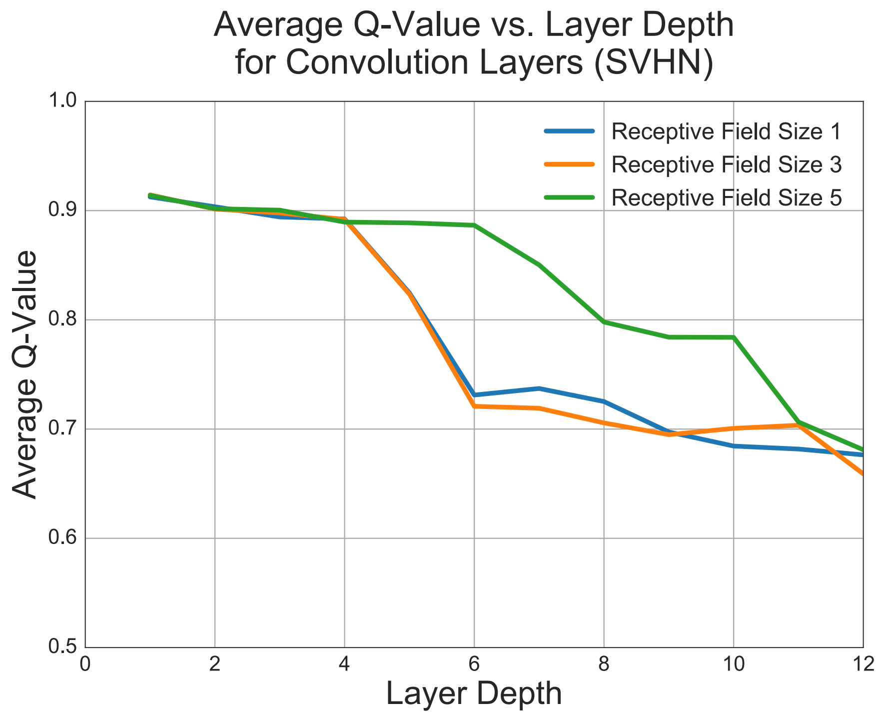
Method	CIFAR-10
DropConnect (Wan et al., 2013)	9.32
DSN (Lee et al., 2015)	8.22
R-CNN (Liang & Hu, 2015)	7.72
MetaQNN (ensemble)	7.32
MetaQNN (top model)	~~6.92~~ 5.3
Resnet(110) (He et al., 2015)	6.61
Resnet(1001) (He et al., 2016)	**4.62**
ELU (Clevert et al., 2015)	6.55
Tree+Max-Avg (Lee et al., 2016)	6.05

Updated Results:
Different Model Depths Don't Train Equally

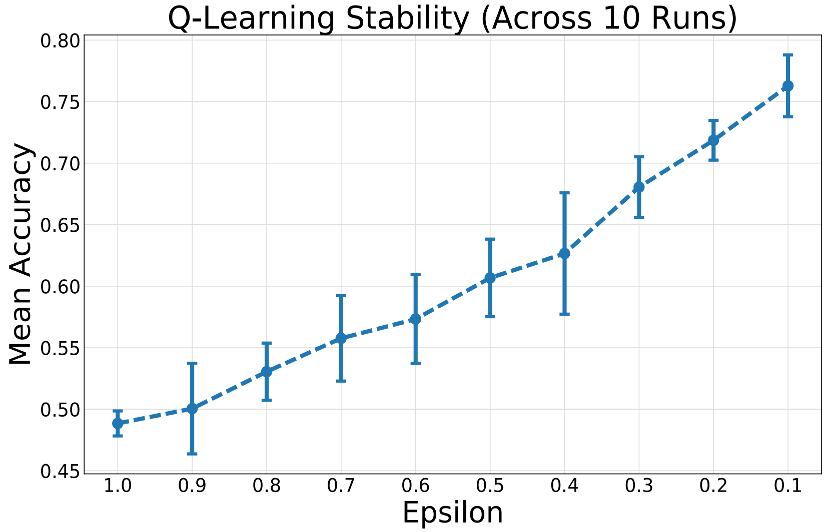
Method	Test Error on CIFAR-10	# Samples	Estimated Computation (GPU-Days)
MetaQNN (Ours)	5.3	2,700	100
Neural Architecture Search (Zoph et al., 2016)	3.65	12,800	10,000
Large Scale Evolution (Real et al., 2017)	5.4	-	2,600
Bayesian Optimization (Snoek et al., 2012)	9.5	50	-

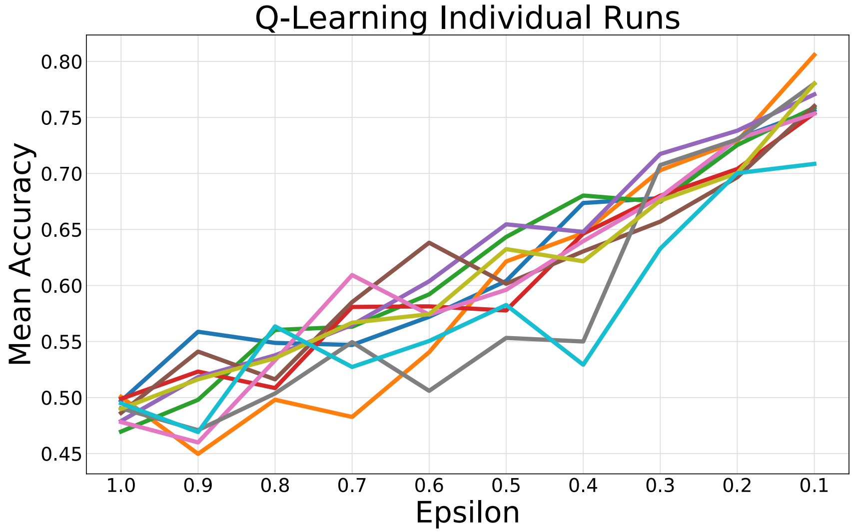
Q-Value Analysis



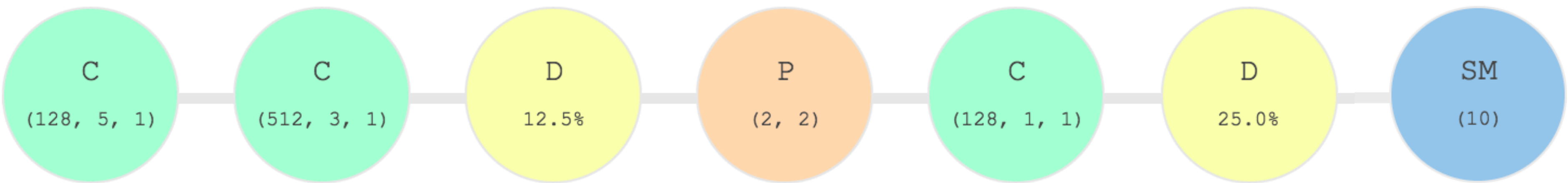


MetaQNN Stability

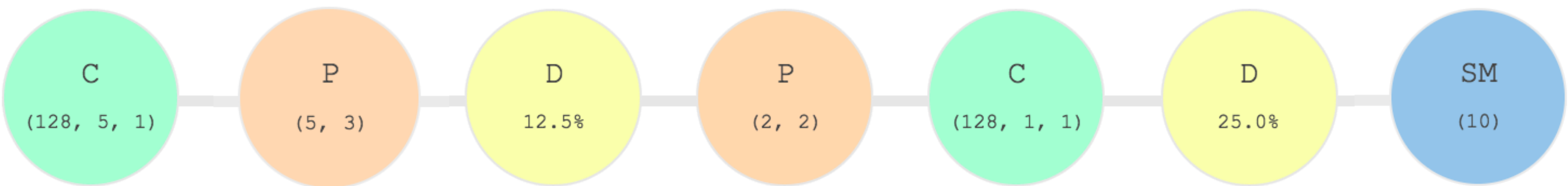




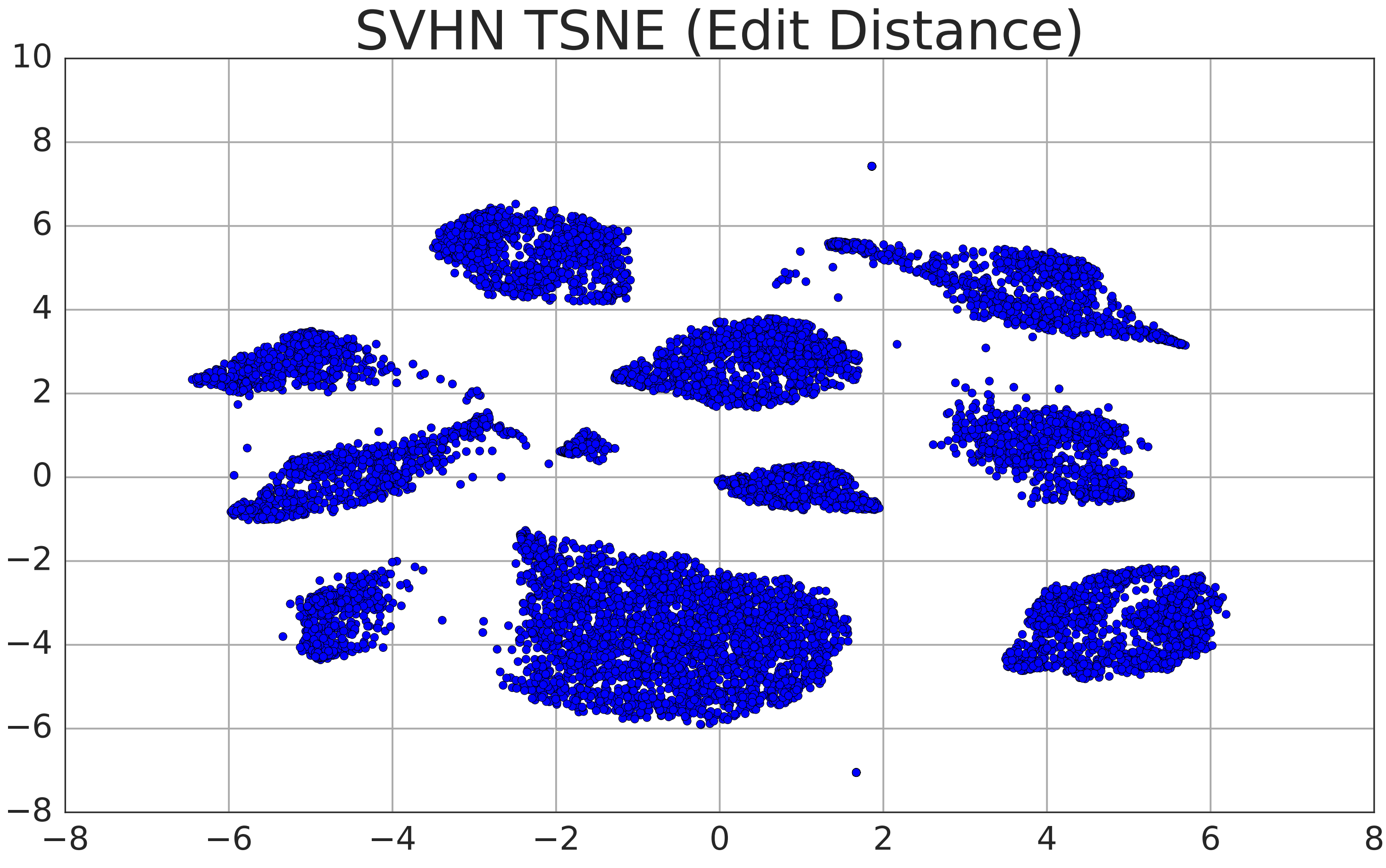
Why Does It Work?



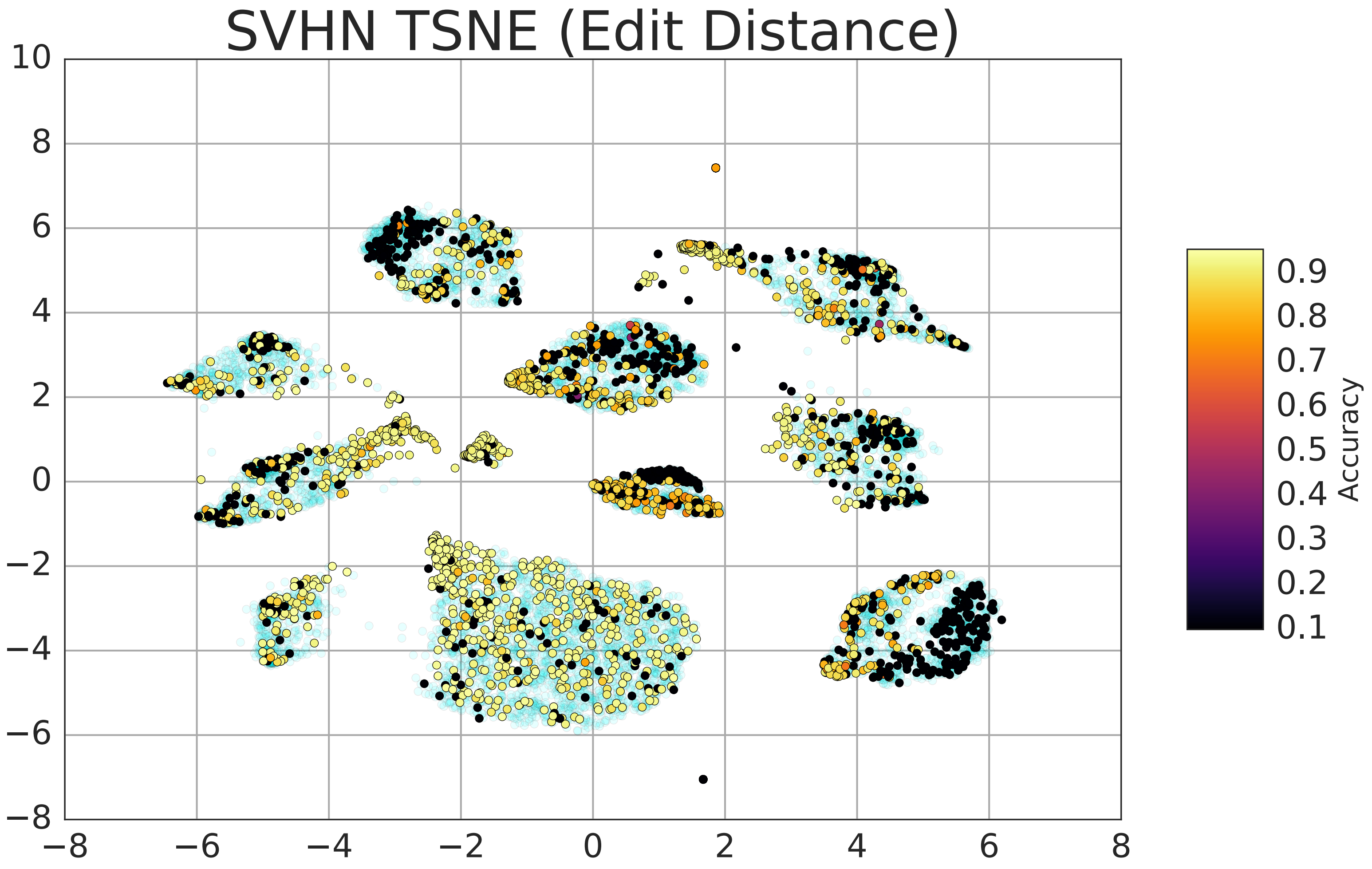
Why Does It Work?



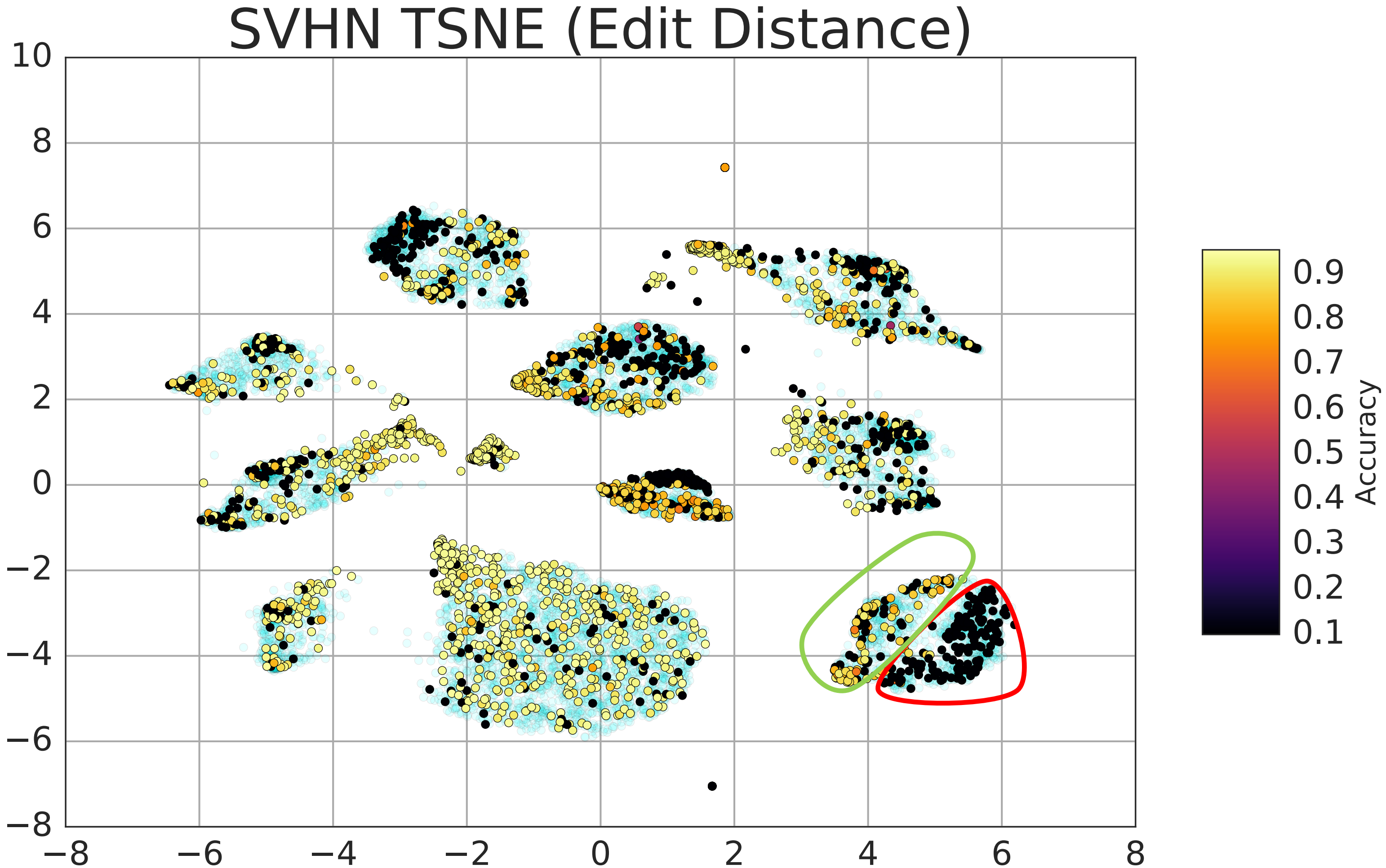
Why Does It Work?



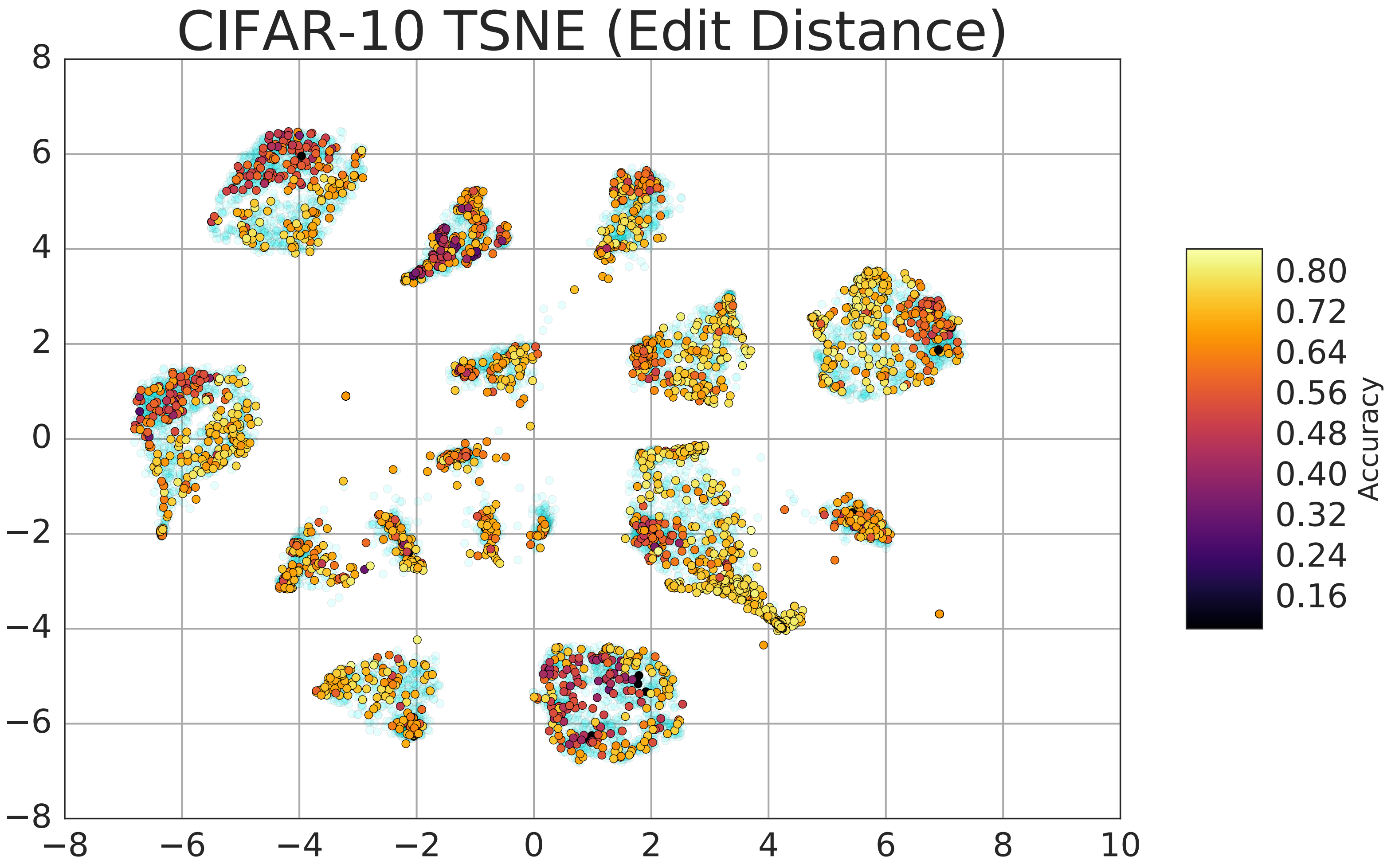
Why Does It Work?



Why Does It Work?



Why Does It Work?



Outline

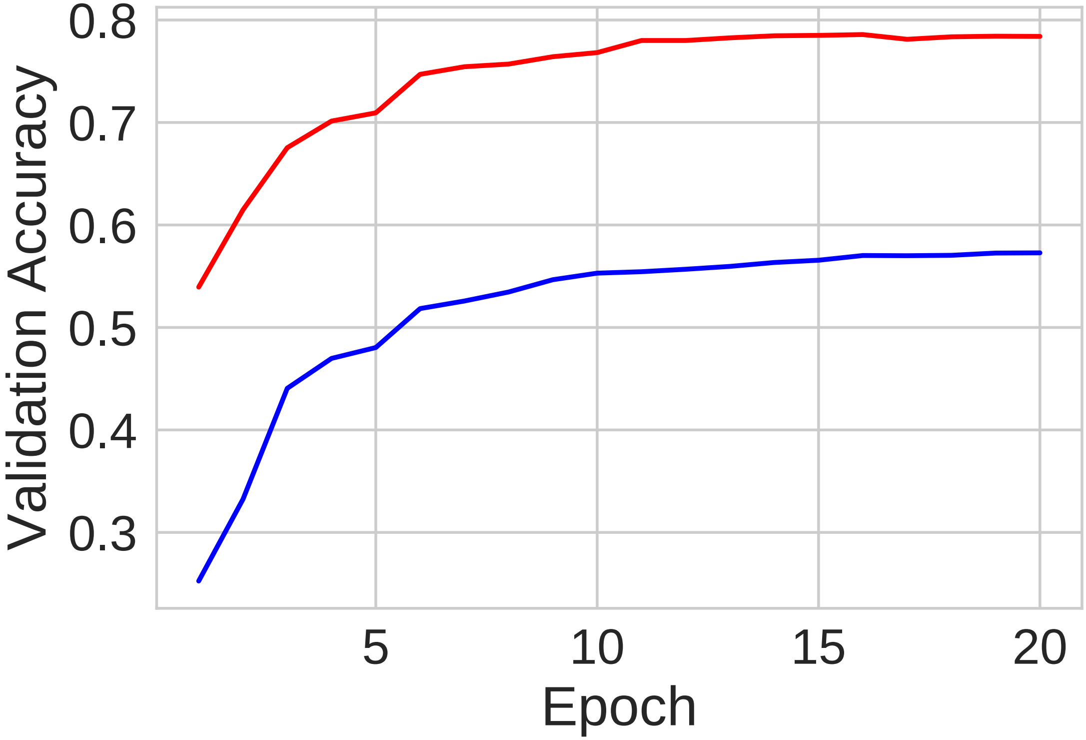
1. Reinforcement Learning Background
2. Modeling Architecture Selection as a Markov Decision Process
3. Results with Q-Learning
4. **Accelerating Architecture Selection with Simple Early Stopping Algorithms**

Meta-Modeling Comparison on CIFAR-10

Method	Test Error on CIFAR-10	# Samples	Estimated Computation (GPU-Days)
MetaQNN (Ours)	6.92	2,700	**100**
Neural Architecture Search (Zoph et al., 2016)	3.65	12,800	**10,000**
Large Scale Evolution (Real et al., 2017)	5.4	-	**2,600**
Bayesian Optimization (Snoek et al., 2012)	9.5	50	-

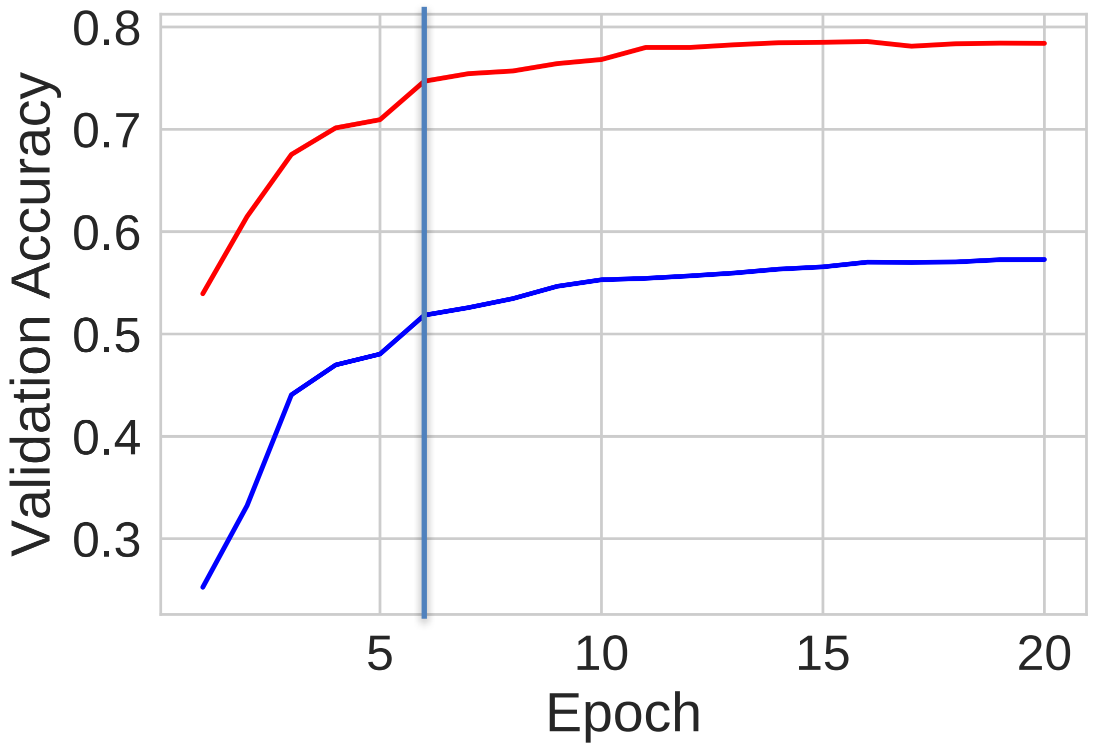
Early Stopping

- Humans are pretty good at recognizing sub-optimal training configurations



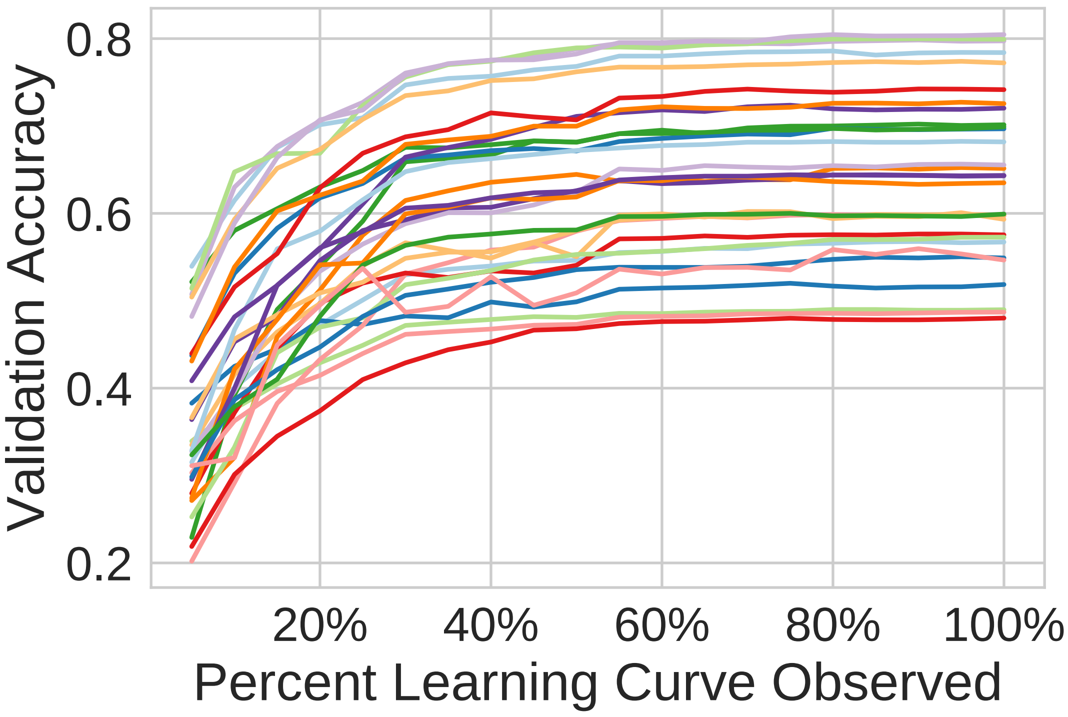
Early Stopping

- Humans are pretty good at recognizing sub-optimal training configurations



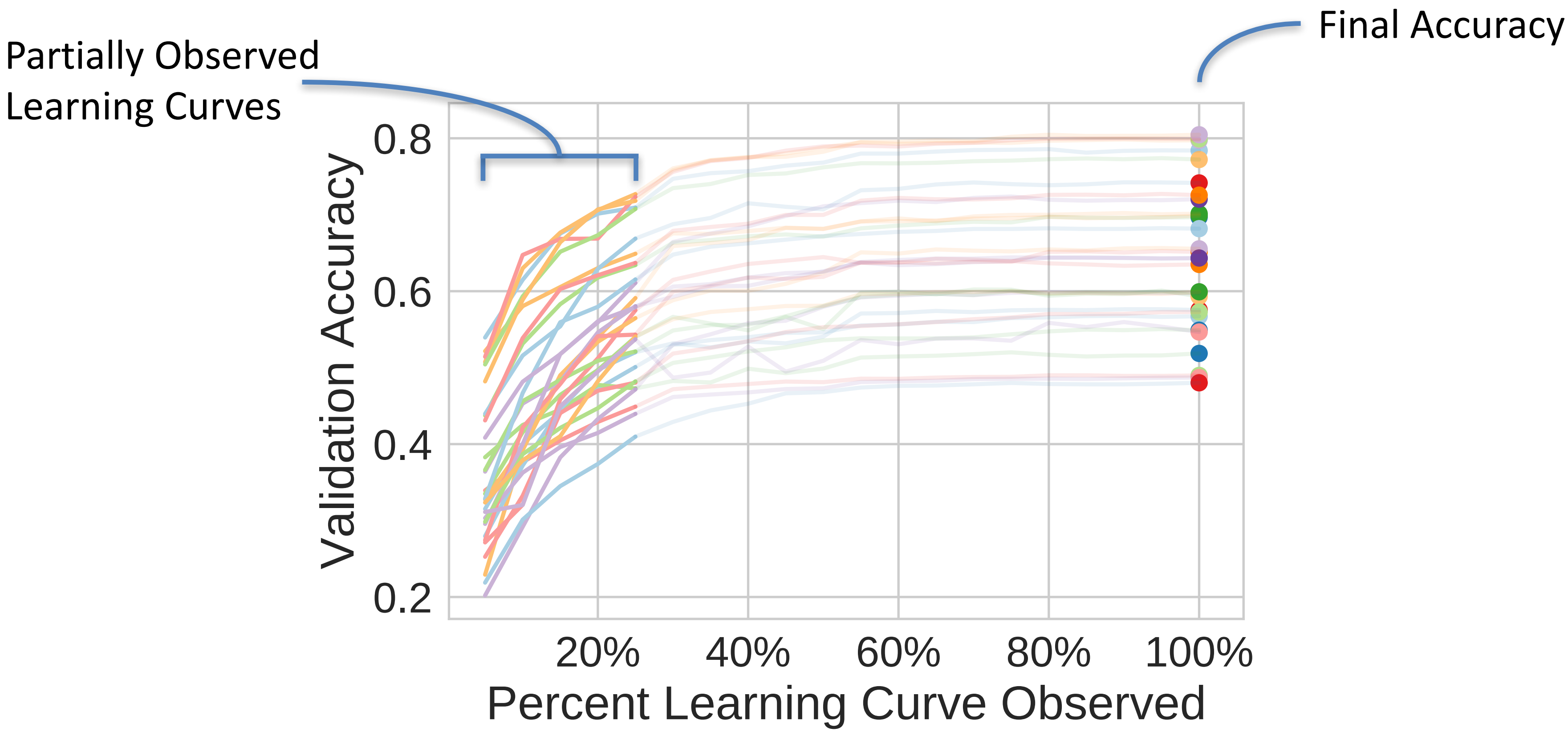
Early Stopping Using Partially Observed Learning Curves

- Use a simple model to predict final accuracy given a partially observed learning curve



Early Stopping Using Partially Observed Learning Curves

- Use a simple model to predict final accuracy given a partially observed learning curve

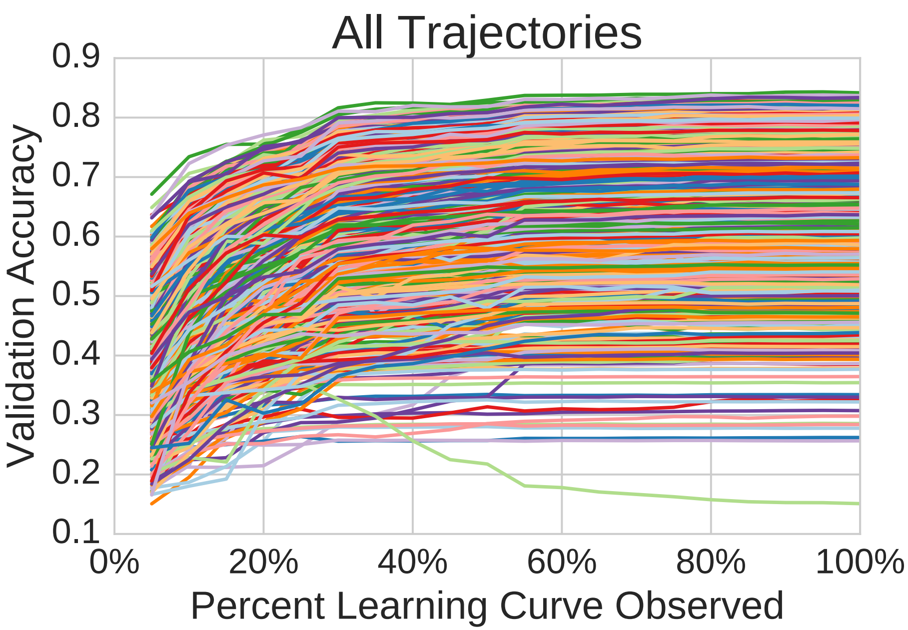


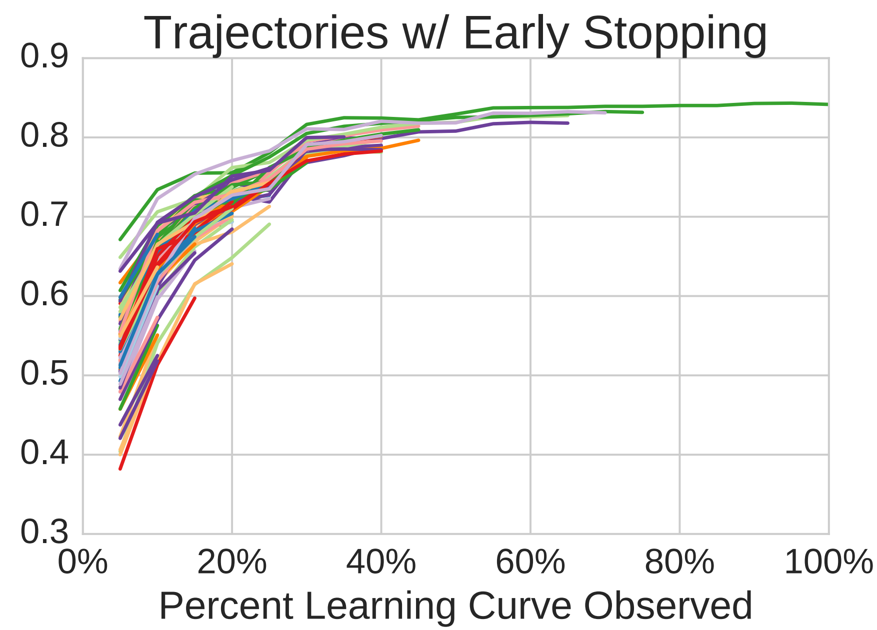
Early Stopping Using Partially Observed Learning Curves

- Use a simple model to predict final accuracy given a partially observed learning curve
- Use performance prediction to terminate sub-optimal configurations

Early Stopping Using Partially Observed Learning Curves

- Use a simple model to predict final accuracy given a partially observed learning curve
- Use performance prediction to terminate sub-optimal configurations

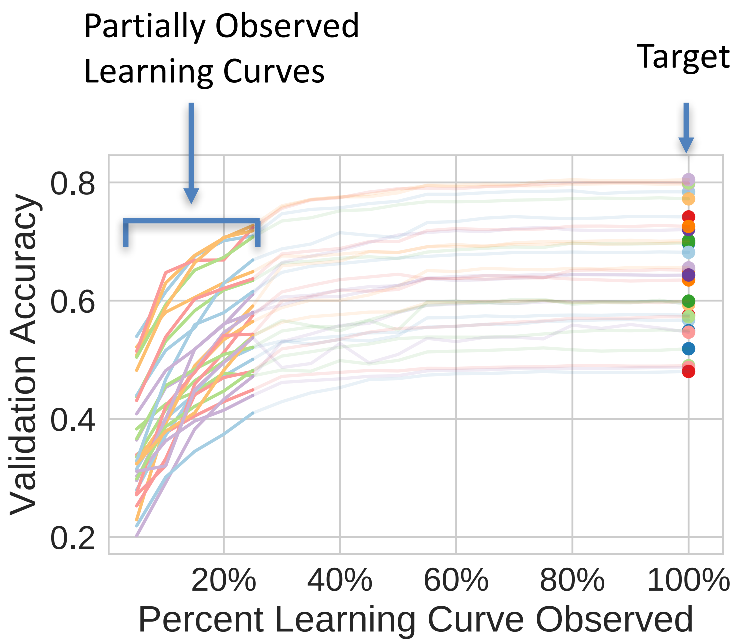




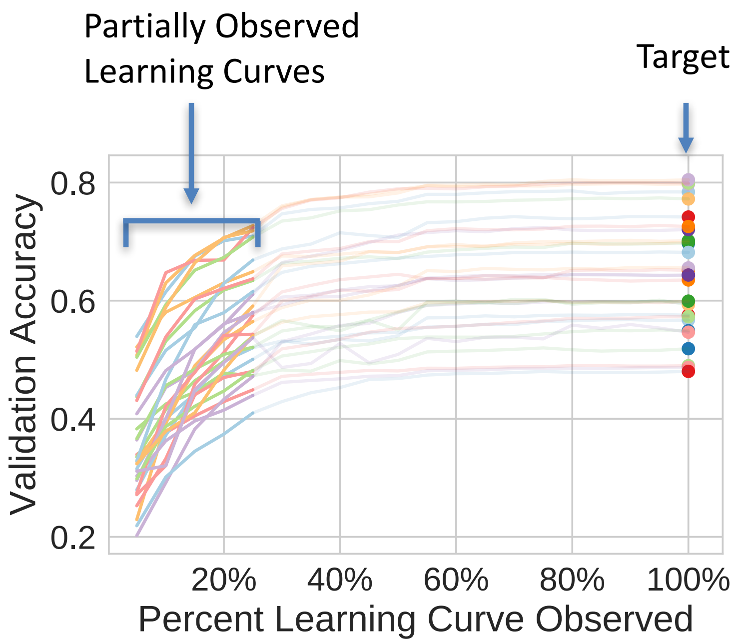
Performance Prediction Model

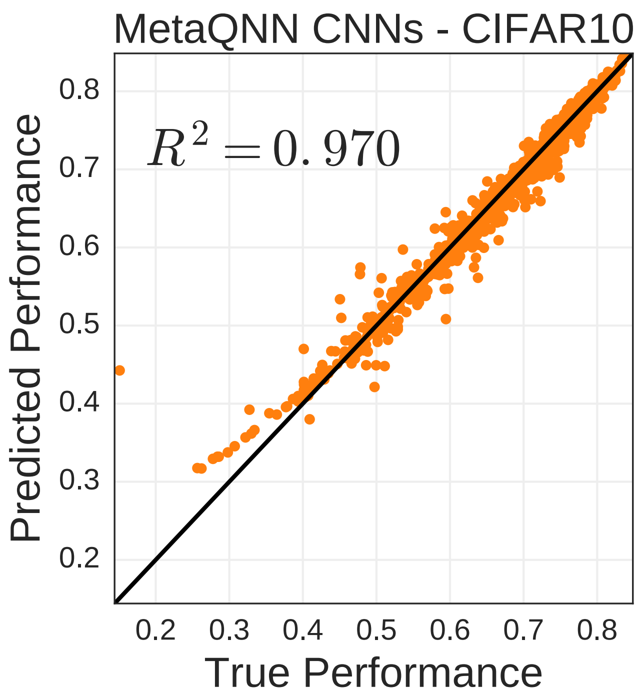
- Features:
  - $y_{1 \ldots t}$ Partially observed learning curves
  - x_f Model features, e.g. # layers, # weights, etc.
- Target
  - y_T Final Accuracy
- Works for both hyperparameter optimization and meta-modeling

Meta-Modeling Example (CIFAR-10)



Meta-Modeling Example (CIFAR-10)





- 100 training examples
- 25% learning curve observed

Experiments

- MetaQNN – Cifar10/SVHN
  - Vary Architectures

Experiments

- MetaQNN – Cifar10/SVHN
  - Vary Architectures
- Resnets – Cifar10
  - Similar search space to Neural Architecture Search

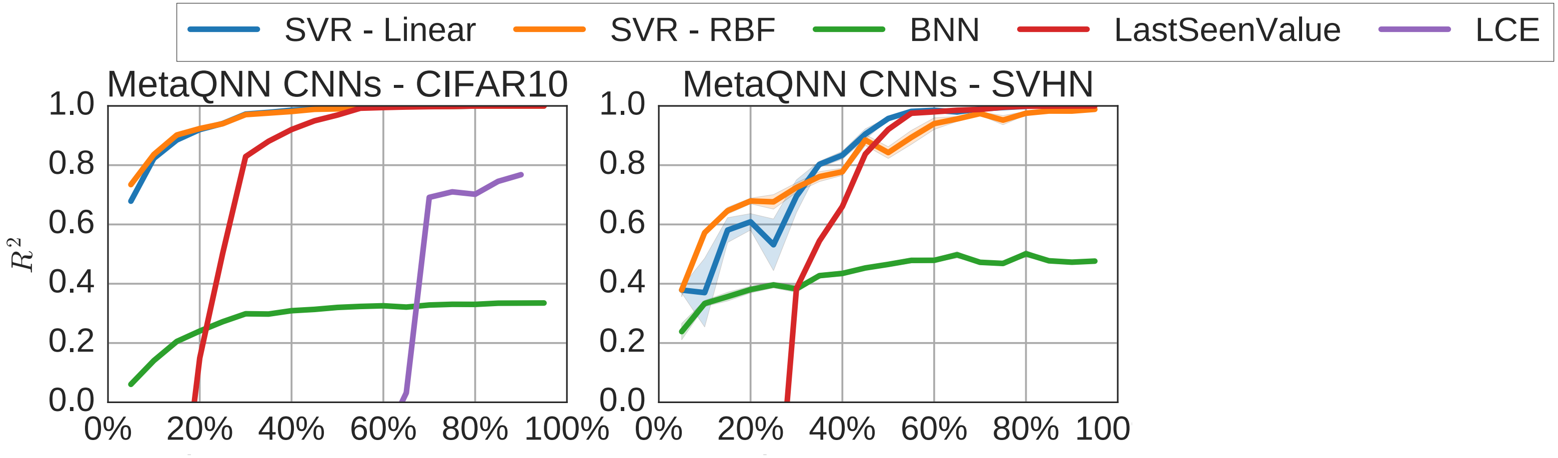
Experiments

- MetaQNN – Cifar10/SVHN
  - Vary Architectures
- Resnets – Cifar10
  - Similar search space to Neural Architecture Search
- Small Neural Network– Cifar10/SVHN
  - Vary optimization hyperparameters, e.g. learning rate, # learning rate decay steps, per layer L2 loss weight, response normalization scale and power

Experiments

- MetaQNN – Cifar10/SVHN
  – Vary Architectures
- Resnets – Cifar10
  – Similar search space to Neural Architecture Search
- Small Neural Network– Cifar10/SVHN
  – Vary optimization hyperparameters, e.g. learning rate, # learning rate decay steps, per layer L2 loss weight, response normalization scale and power
- AlexNet – 10% ImageNet
  – Vary learning rate and # learning rate decay steps

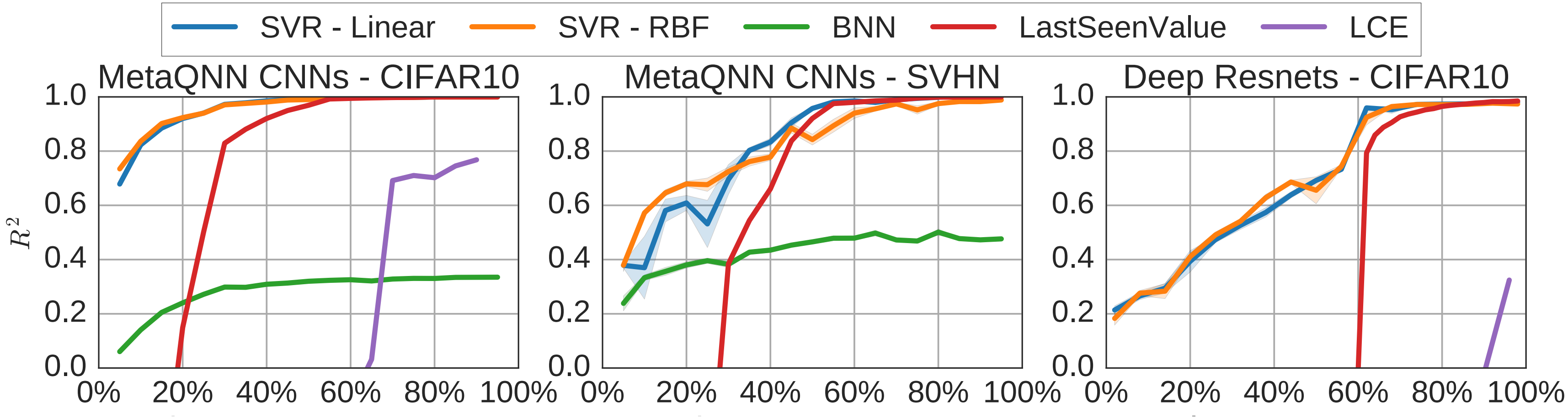
Performance Prediction Model



LCE: Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. IJCAI, 2015

BNN: Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. International Conference on Learning Representations, 17, 2017.

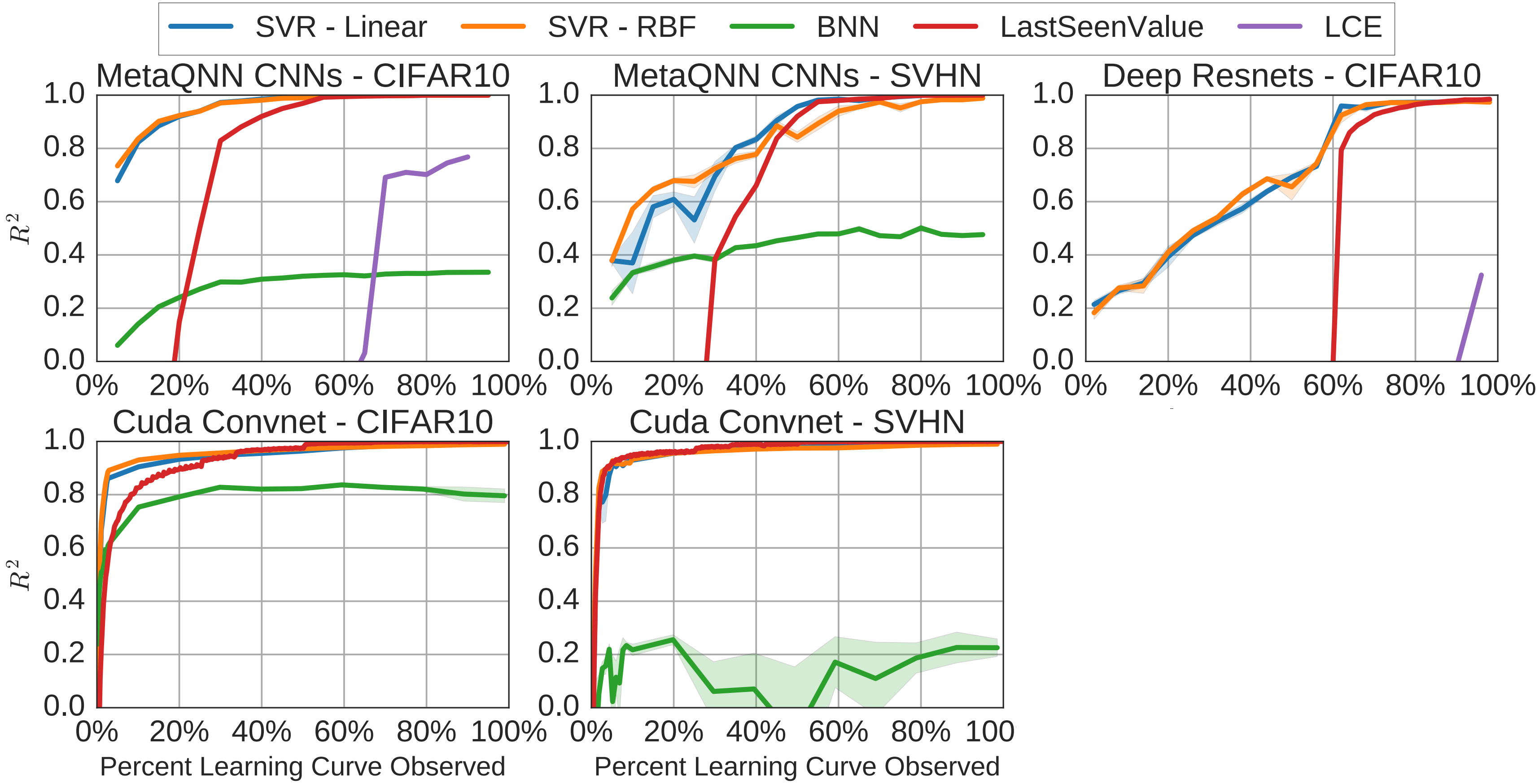
Performance Prediction Model



LCE: Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. IJCAI, 2015

BNN: Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. International Conference on Learning Representations, 17, 2017.

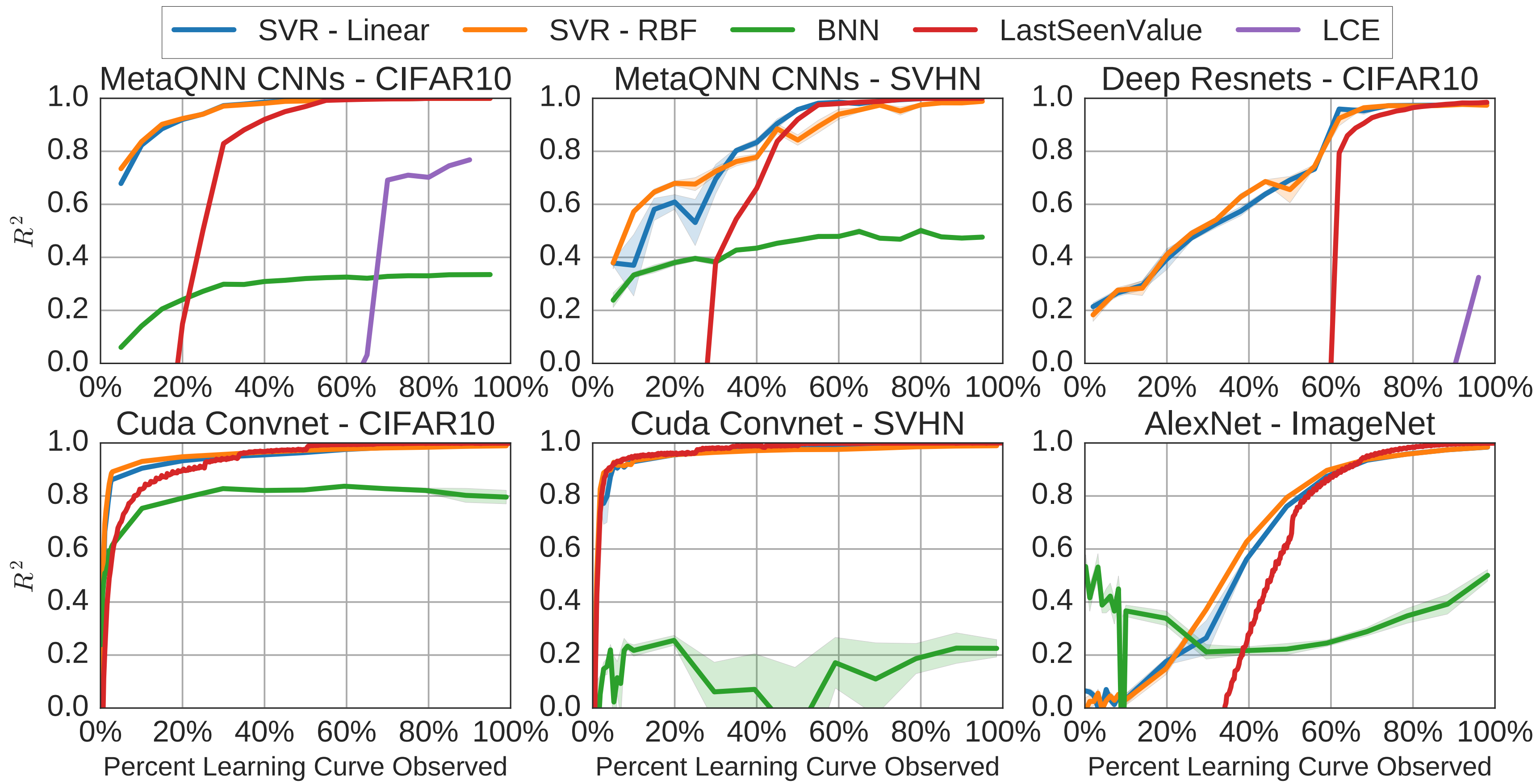
Performance Prediction Model



LCE: Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. IJCAI, 2015

BNN: Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. International Conference on Learning Representations, 17, 2017.

Performance Prediction Model



LCE: Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. IJCAI, 2015

BNN: Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. International Conference on Learning Representations, 17, 2017.

Early Stopping

1. Given performance prediction model

$$\hat{y}_T(t) = f\left(y_{1\ldots t}, x_f\right)$$

Early Stopping

1. Given performance prediction model

$$\hat{y}_T(t) = f\left(y_{1\ldots t}, x_f\right)$$

2. Assume errors are zero-mean Gaussian conditioned on t

$$\hat{y}_T(t) \; - y_T \sim N(0, \sigma_t)$$

Early Stopping

1. Given performance prediction model

$$\hat{y}_T(t) = f\left(y_{1\ldots t}, x_f\right)$$

2. Assume errors are zero-mean Gaussian conditioned on t

$$\hat{y}_T(t) - y_T \sim N(0, \sigma_t)$$

3. Estimate σ_t empirically from training set using LOOCV

Early Stopping

1. Given performance prediction model

$$\hat{y}_T(t) = f\left(y_{1\ldots t}, x_f\right)$$

2. Assume errors are zero-mean Gaussian conditioned on t

$$\hat{y}_T(t) - y_T \sim N(0, \sigma_t)$$

3. Estimate σ_t empirically from training set using LOOCV
4. Define probability of improvement,

$$p(\hat{y}_T(t) < y_{BEST}) = 1 - \phi(y_{BEST}; \hat{y}_T(t), \sigma_t)$$

where $\phi(\,\cdot\,; \mu, \sigma_t)$ is the CDF of $N(\mu, \sigma_t)$

Early Stopping

1. Given performance prediction model

$$\hat{y}_T(t) = f\left(y_{1\ldots t}, x_f\right)$$

2. Assume errors are zero-mean Gaussian conditioned on t

$$\hat{y}_T(t) \; - y_T \sim N(0, \sigma_t)$$

3. Estimate σ_t empirically from training set using LOOCV

4. Define probability of improvement,

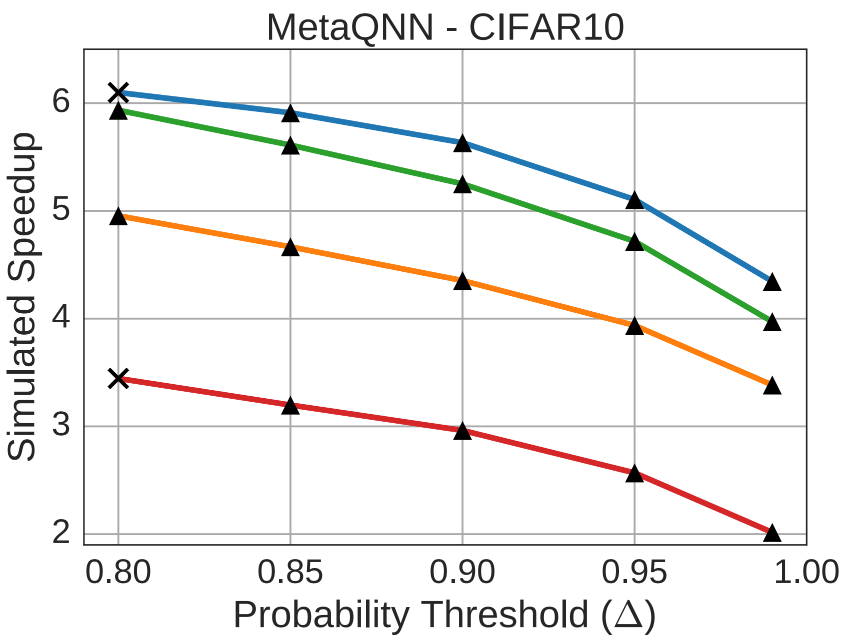
$$p(\hat{y}_T(t) < y_{BEST}) = 1 - \; \phi(y_{BEST}; \hat{y}_T(t), \sigma_t)$$

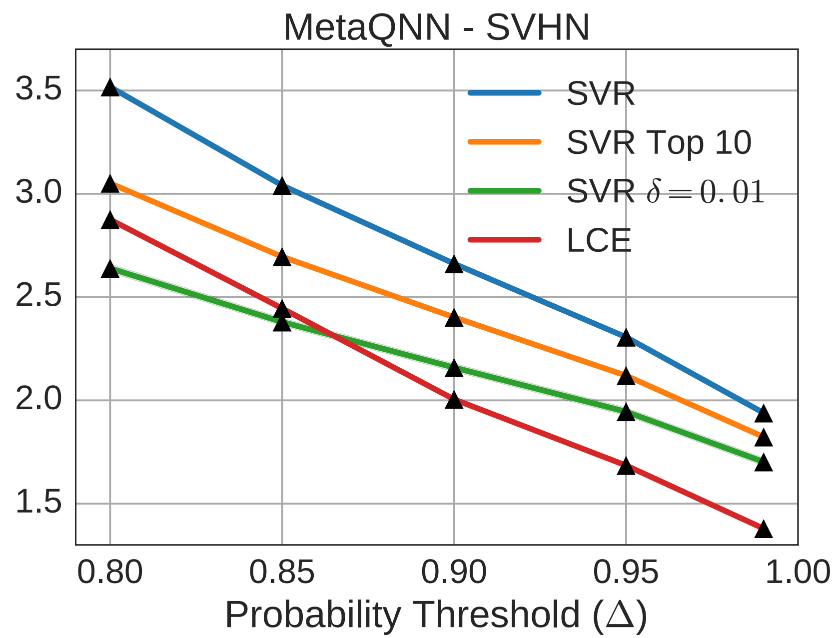
where $\phi(\,\cdot\,; \mu, \sigma_t)$ is the CDF of $N(\mu, \sigma_t)$

5. Define acceptance probability threshold Δ such that training is terminated at time-step t if

$$p(\hat{y}_T(t) < y_{BEST}) > \Delta$$

Early Stopping Results





X ~ On average does not recover best model

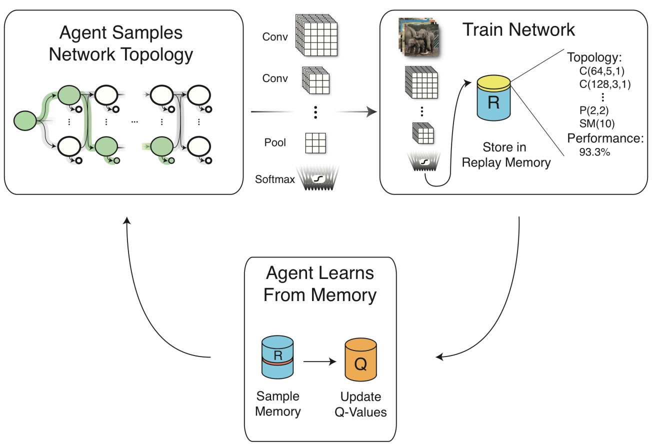
▲ ~ On average recovers best model

δ ~ Termination rule $p(\hat{y}_T(t) < y_{BEST} - \delta) > \Delta$

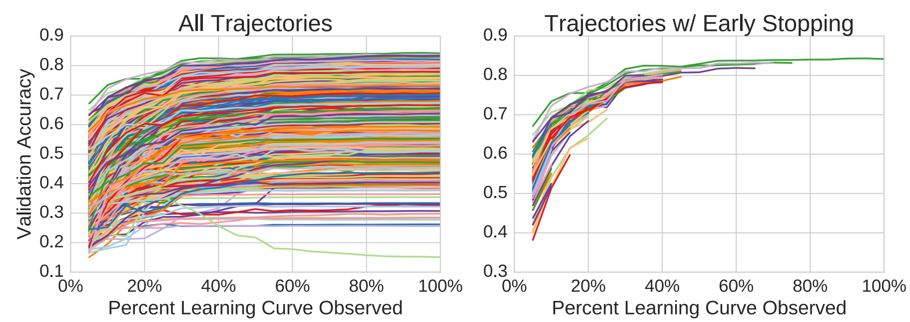
Top 10 ~ Termination rule $p(\hat{y}_T(t) < y_{10^{th}\ BEST}) > \Delta$

Summary

Designing neural network architectures using reinforcement learning [1]



Practical Neural Network Performance Prediction for Early Stopping [2]

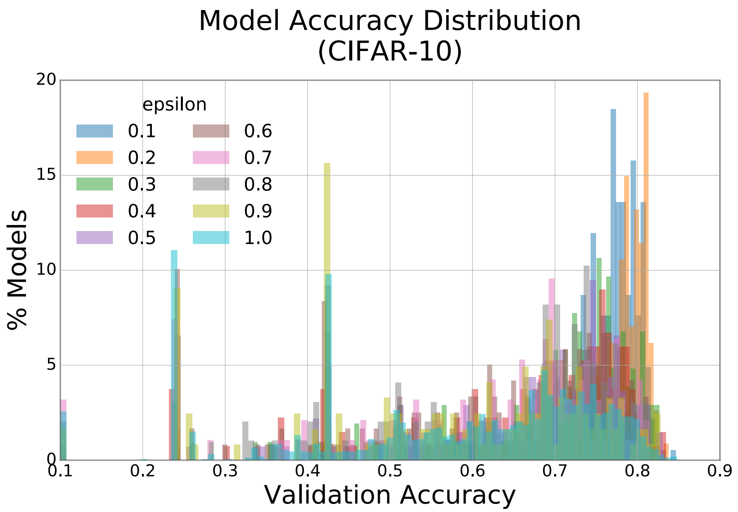


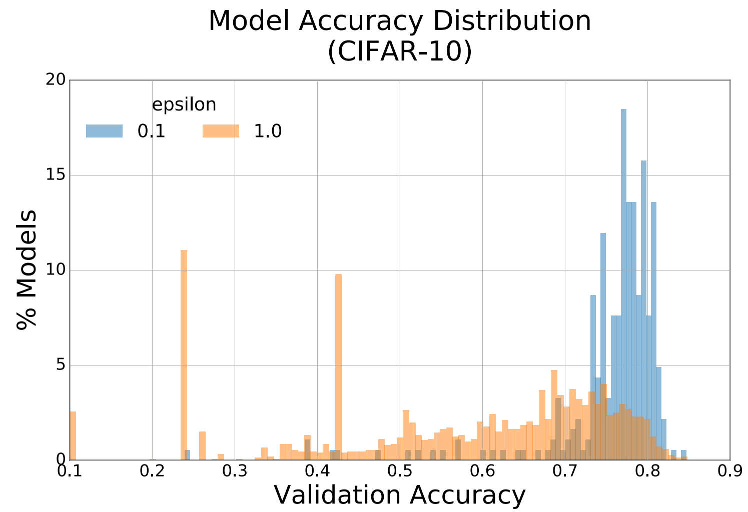
Contact: bowen@mit.edu
Slides: bowenbaker.github.io (check back later today)
MetaQNN Code: Released by end of week

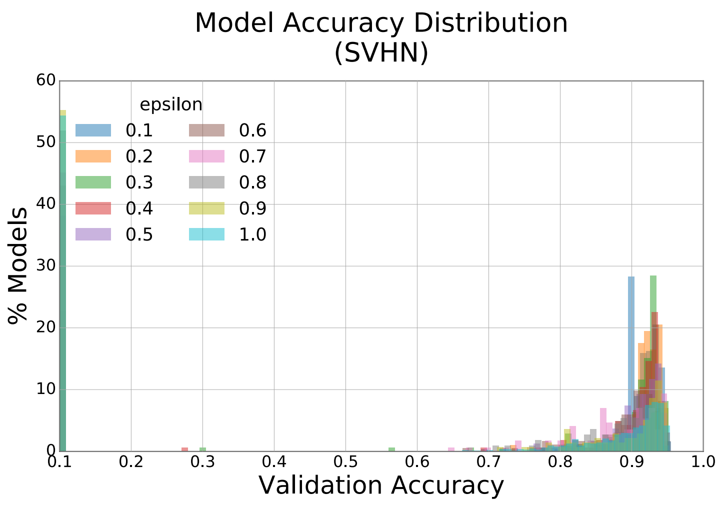
1. Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. “Designing neural network architectures using reinforcement learning.” *International Conference on Learning Representations*, 2017.
2. Bowen Baker*, Otkrist Gupta*, Ramesh Raskar, and Nikhil Naik. "Practical Neural Network Performance Prediction for Early Stopping.” *Under Submission*, 2017.

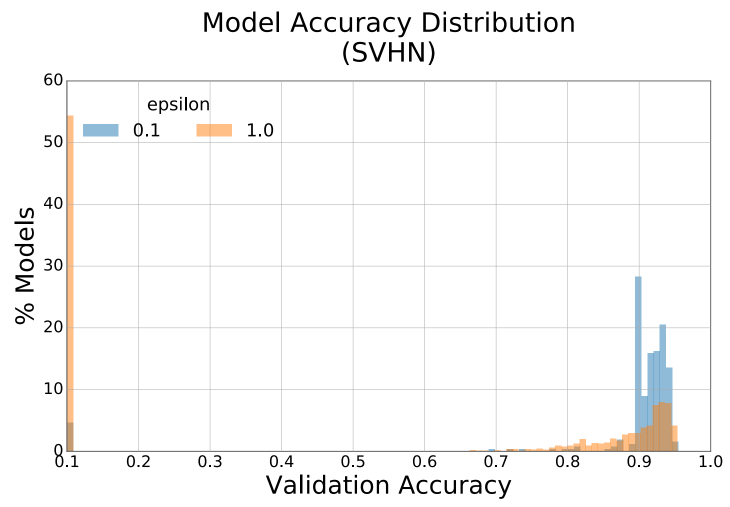
Appendix

Exploration Distributions







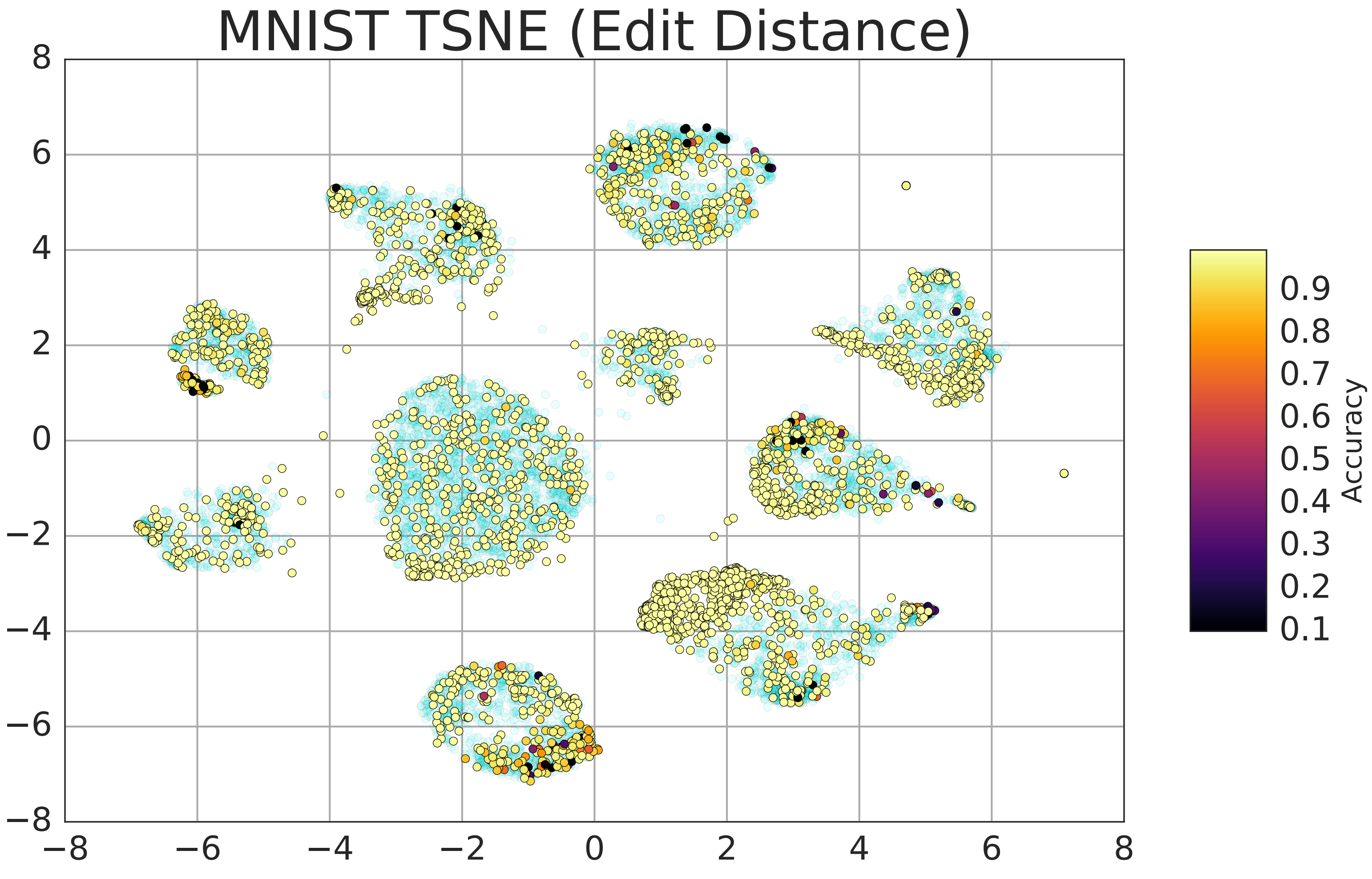


Transferability

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

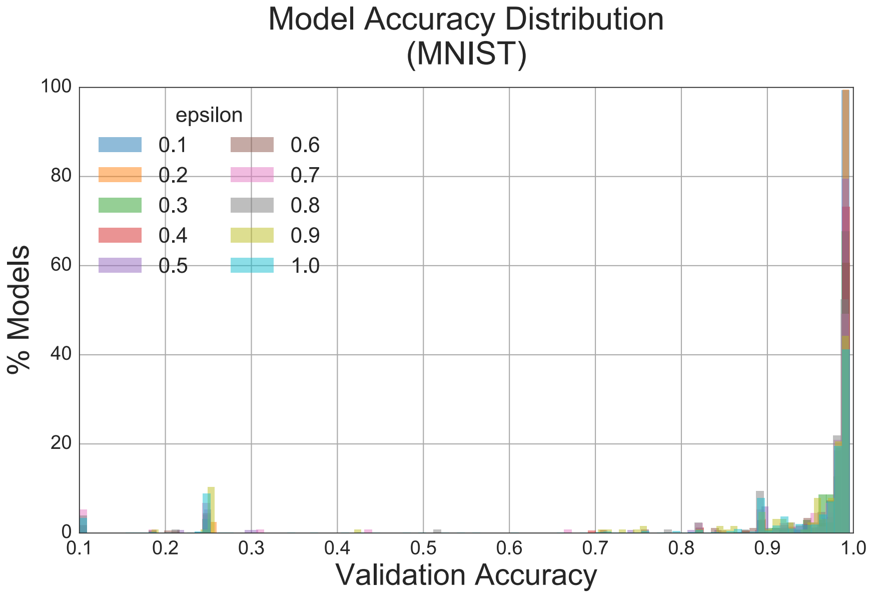
Dataset	CIFAR-100	SVHN	MNIST
Training from scratch	27.14	2.48	0.80
Finetuning	34.93	4.00	0.81
State-of-the-art	24.28 (Clevert et al., 2015)	1.69 (Lee et al., 2016)	0.31 (Lee et al., 2016)

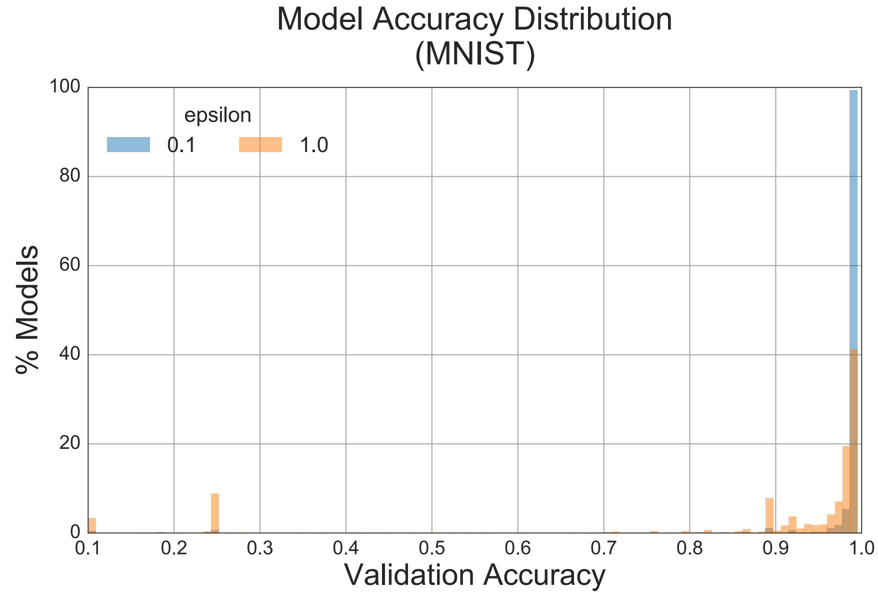
MNIST t-SNE



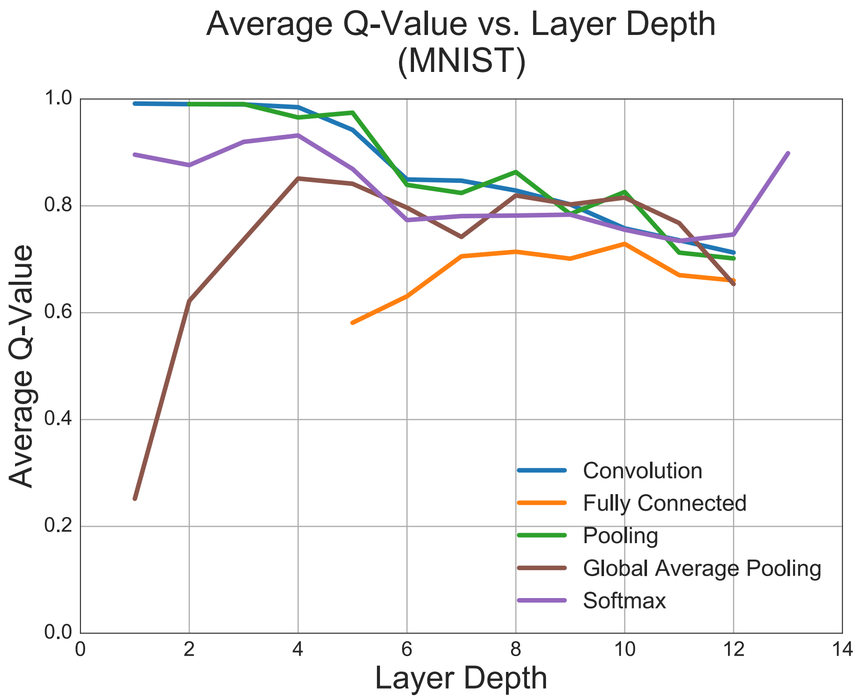
MNIST TSNE (Edit Distance)
8
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6
8
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Accuracy

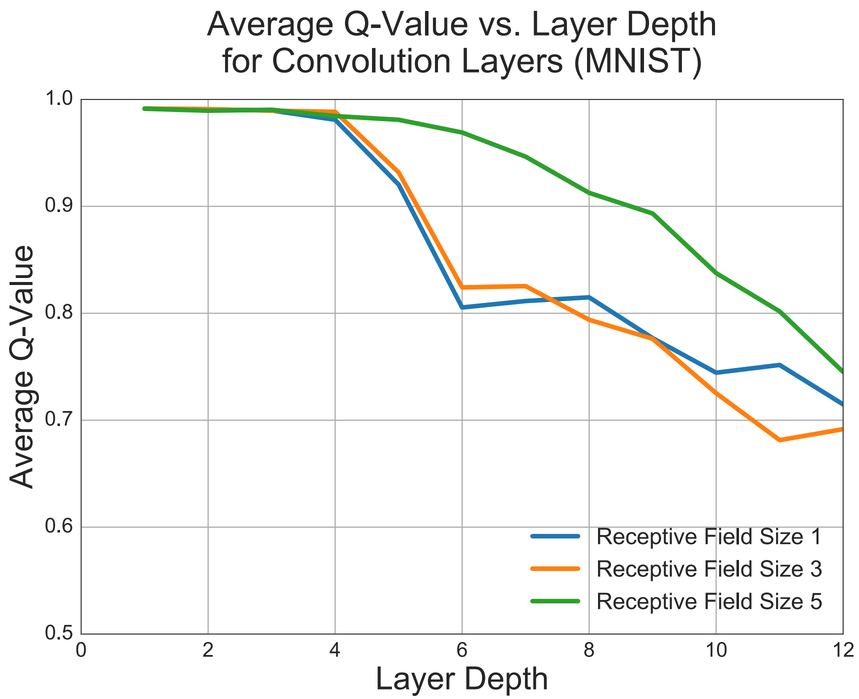
MNIST Exploration Distribution



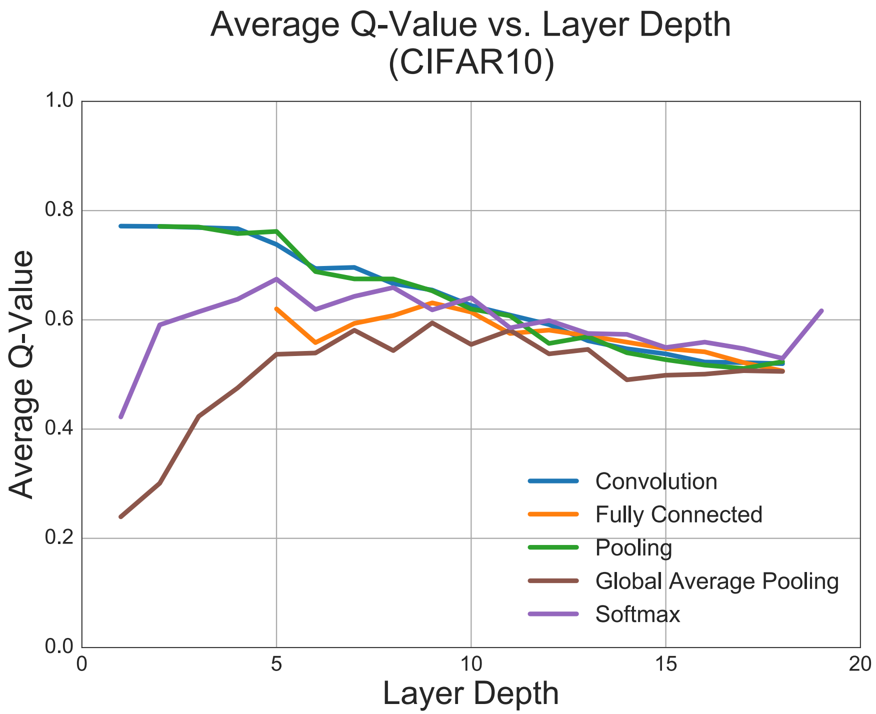


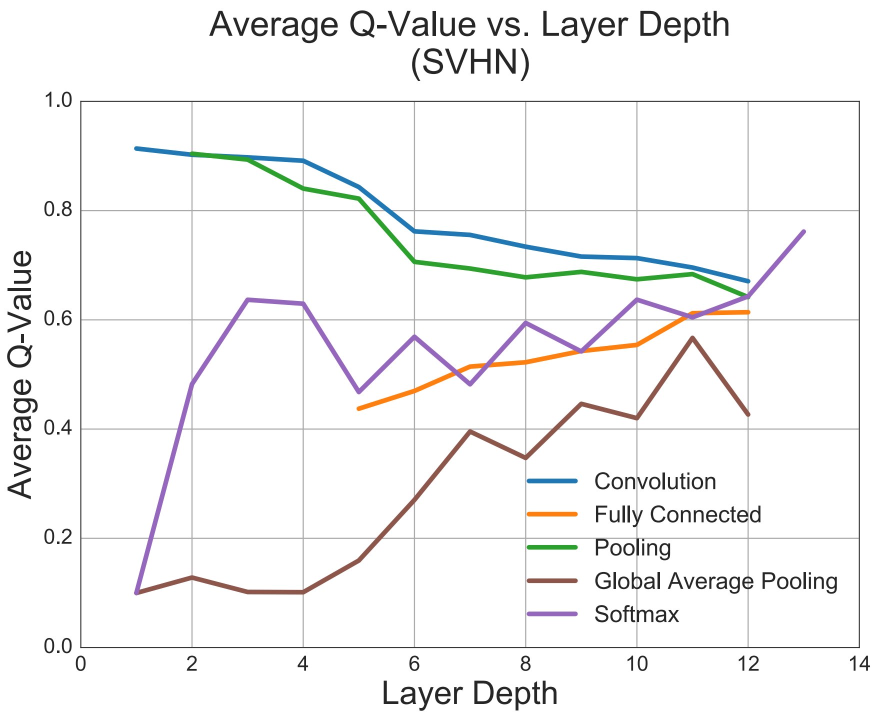
MNIST Q-Value Analysis





Q-Value Analysis





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

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

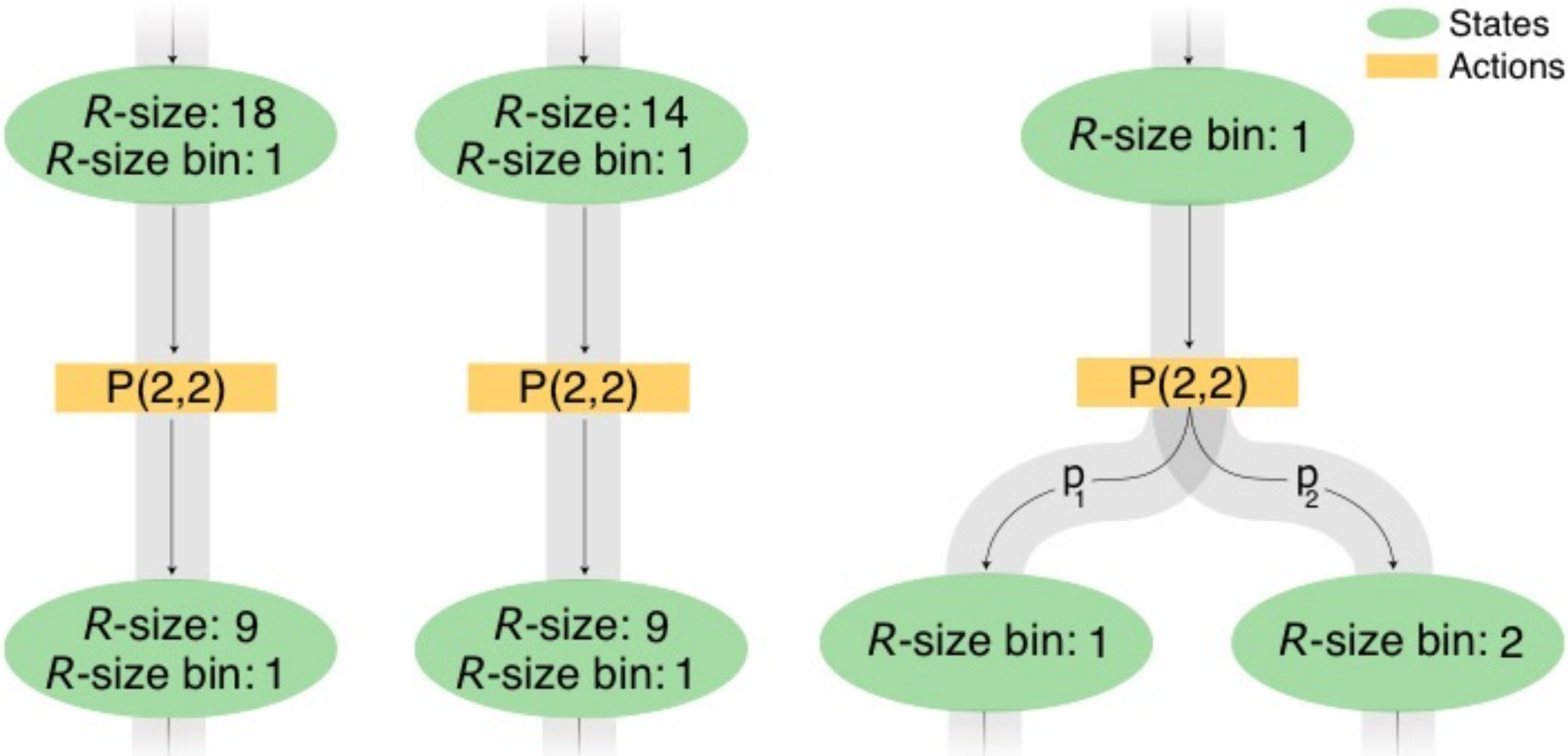
Top Models (MNIST)

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

Top Model Cifar-10 (Updated Results)

```
[C(64,3,1), C(256,3,1), D(1,9), C(512,3,1), C(64,3,1),
D(2,9), C(128,5,1), P(2,2), D(3,9), C(512,5,1), P(2,2),
D(4,9), C(128,5,1), C(256,5,1), D(5,9), C(512,3,1),
C(64,5,1), D(6,9), P(2,2), C(512,1,1), D(7,9), FC(128),
D(8,9), SM(10)]
```

Representation Size



(a) (b) (c)

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

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

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