Meta-Algorithms

Safa Messaoud

Mariya Vasileva

The Machine Learning Holy Grail

Moving from domain-specific learning algorithms to general purpose learning algorithms (meta-learning algorithms) that can learn better learning algorithms

Stop engineering the algorithms **the same way we stopped engineering the** features **!**

Date	Slides	Reading list
January 17	Class intro	N/A
January 19	CNN architectures (Lana): PPT, PDF	Reading list
January 24	RNN Tutorial (Arun): PPT, PDF	Reading list
January 26	RNN Tutorial Part 2 (Arun): PPT, PDF	Reading list
January 31	Advanced CNN architectures (Akshay, Hong): PPT, PDF	Reading list
February 2	Advanced training techniques (Prajit): PPT, PDF	Reading list
February 7	Network compression, speedup (Shuochao, Yiwen, Daniel): PPT, PDF	Reading list
February 9	Object detection (Jiajun, Sihao, Kevin): PPT, PDF	Reading list
February 14	Semantic segmentation, dense labeling (Liwei): PPT, PDF	Reading list
February 16	Similarity learning (Moitreya, Yunan): PPT, PDF	Reading list
February 21	Visualization, adversarial examples (Ralf, Jyoti, Jiahui): PPT, PDF	Reading list
February 23	Generative adversarial networks (Shashank, Bhargav, Binglin): PPT, PDF	Reading list
February 28	Variational autoencoders (Raymond, Junting, Teck-Yian): PDF	Reading list
March 2	Advanced generation methods (Ameya, Hsiao-Ching, Anand): PPT, PDF	Reading list
March 7	3D + graphics (Juho, Qi): PPT, PDF	Reading list
March 9	Self-supervised learning (Nate, Christian, Pratik): PPT, PDF	Reading list
March 10	Intro to reinforcement learning -- bonus lecture (Unnat, Garima, Karan): PDF 10-11:30AM, SC 216	Reading list
March 14	Deep Q learning (Unnat, Garima, Karan): PPT, PDF	Reading list
March 16	Deep reinforcement learning: policy gradients, planning (Tanmay, Raj, Zhizhong): PDF	Reading list
March 28	Deep learning for manipulation, navigation (Tanmay, Andrey): PPT, PDF	Reading list
March 30	Recurrent architectures (Abhishek, Anusri): PPT, PDF	Reading list
April 4	Image captioning	Reading list
April 6	Image-text embeddings, grounding	Reading list
April 11	Visual question answering	
April 13	Deep learning for NLP	
April 18	Deep learning for machine translation	
April 20	Deep learning for audio	
April 25	Architectures with memory	
April 27	Meta-algorithms	
May 2	Wrapup, selected project presentations	

Inspiration: Slow Learning to Learn Fast

- We have a system of core knowledge that allows us to reason about objects, numbers, spaces… → Algorithm
- The slow learning (optimization, search process) of evolution led to the emergence of components that enable fast and varied learning → Meta-Algorithm

Inspiration: Slow Learning to Learn Fast

- We have a system of core knowledge that allows us to reason about objects, numbers, spaces… → Algorithm
- The slow learning (optimization, search process) of evolution led to the emergence of components that enable fast and varied learning → Meta-Algorithm

> Radical **learning to learn** is about encoding the **initial learning algorithm** in a universal language, with **primitives** that allow to **modify the code** itself in arbitrary computable fashion. Then, surround this **self-referential**, **self- modifying** code by a **recursive framework** that ensures that only "**useful**" **self-modifications** are executed or survive
>
> *Jürgen Schmidhuber*

Index

- Mathematical Formulation of Meta-Learning
- Learning the Deep learning Architecture
- Learning to Explore
- Learning to Seek Knowledge
- Learning to Communicate

Index

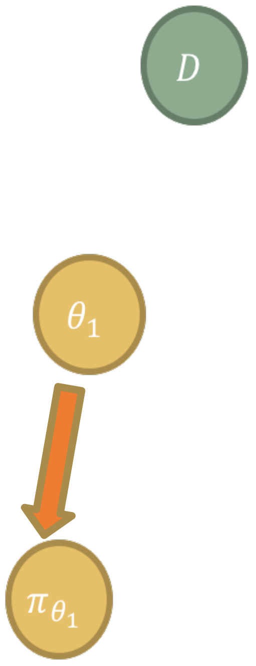
- **Mathematical Formulation of Meta-Learning**
- Learning the Deep learning Architecture
- Learning to Explore
- Learning to Seek Knowledge
- Learning to Communicate

Formal Definition of Meta-Learning

- D : Sample Space

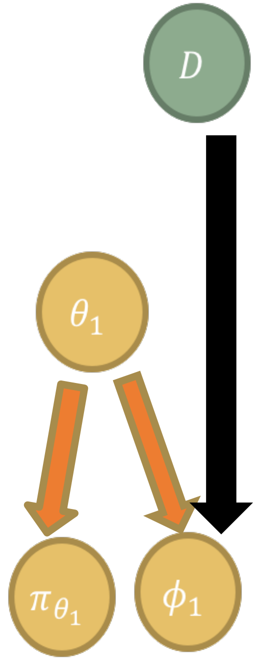
Formal Definition of Meta-Learning

- D : Sample Space
- π_θ : Agent parametrized by $\theta \epsilon \ominus$



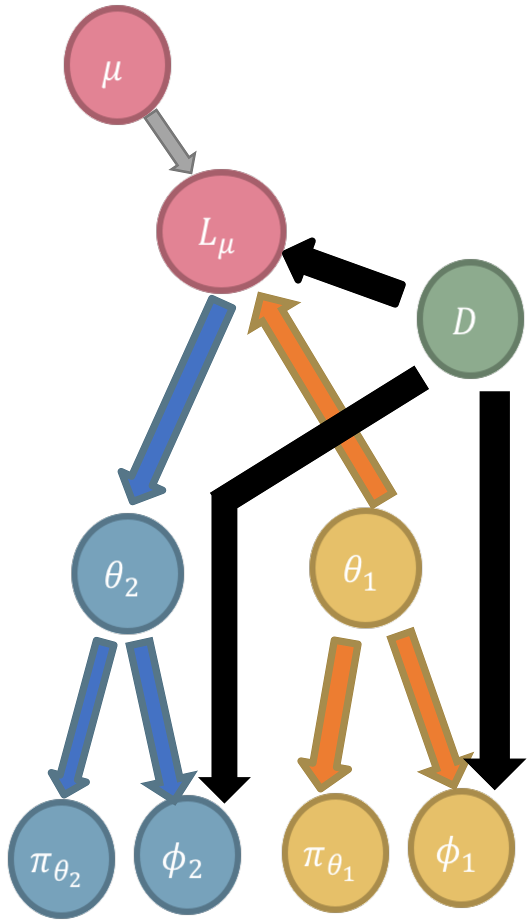
Formal Definition of Meta-Learning

- D : Sample Space
- π_θ : Agent parametrized by $\theta \epsilon \ominus$
- ϕ : The expected performance measure of the agent on a given task



Formal Definition of Meta-Learning

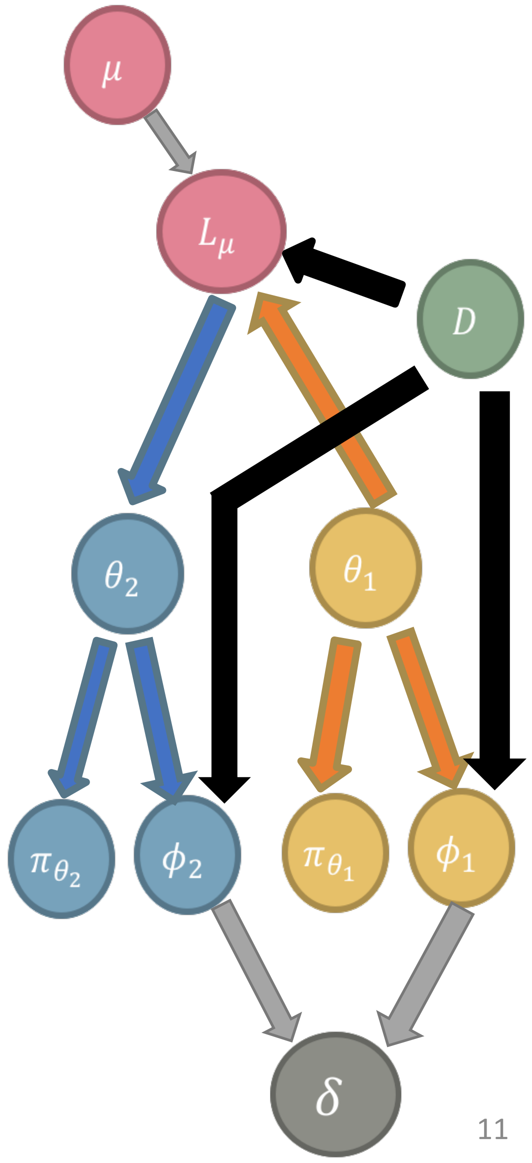
- D : Sample Space
- π_θ : Agent parametrized by $\theta \epsilon \ominus$
- ϕ : The expected performance measure of the agent on a given task
- The learning algorithm $L_\mu: (\Theta, D) \rightarrow \Theta$ is a function that changes the agent parameter Θ to maximize its expected performance
- $\mu \in M$ is a meta parameter of the learning algorithm



Formal Definition of Meta-Learning

- D : Sample Space
- π_θ : Agent parametrized by $\theta \epsilon \ominus$
- ϕ : The expected performance measure of the agent on a given task
- The learning algorithm $L_\mu : (\Theta, D) \rightarrow \Theta$ is a function that changes the agent parameter Θ to maximize its expected performance
- $\mu \in M$ is a meta parameter of the learning algorithm
- The expected performance gain of the learning algorithm

$$\delta(L_\mu) = \mathbb{E}_{\theta\ \epsilon\ \Theta,\ s \epsilon D}[L_\mu(\Phi(\theta, D)) - \Phi(\theta)]$$



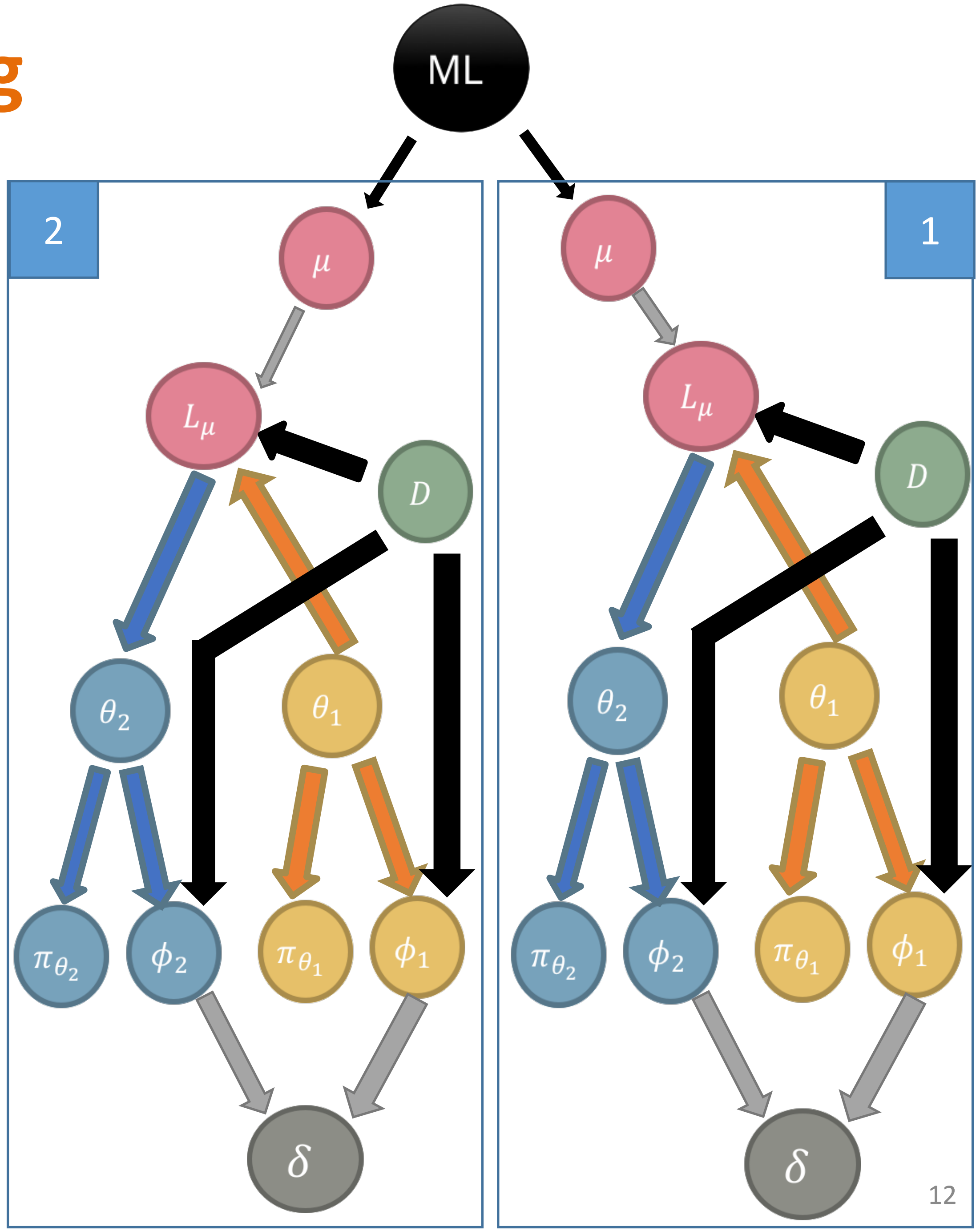
Scholarpedia Article

Formal Definition of Meta-Learning

- D : Sample Space
- π_θ : Agent parametrized by $\theta \epsilon \ominus$
- ϕ : The expected performance measure of the agent on a given task
- The learning algorithm L_μ: $(\Theta, D) \rightarrow \Theta$ is a function that changes the agent parameter Θ to maximize its expected performance
- $\mu \in M$ is a meta parameter of the learning algorithm
- The expected performance gain of the learning algorithm

$$\delta(L_\mu) = \mathbb{E}_{\theta\, \epsilon\, \Theta,\, s \epsilon D}[L_\mu(\Phi(\theta, D)) - \Phi(\theta)]$$

- The meta-Algorithm ML: $(M, D) \rightarrow M$ changes the meta-parameters of the learning algorithm to maximize it expected performance δ



Ensemble Methods

D

ϕ

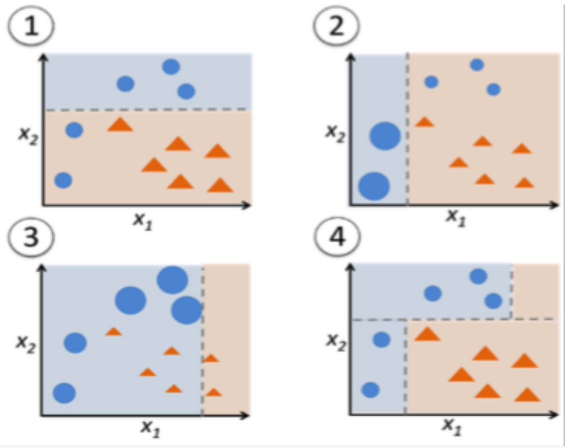
π_θ

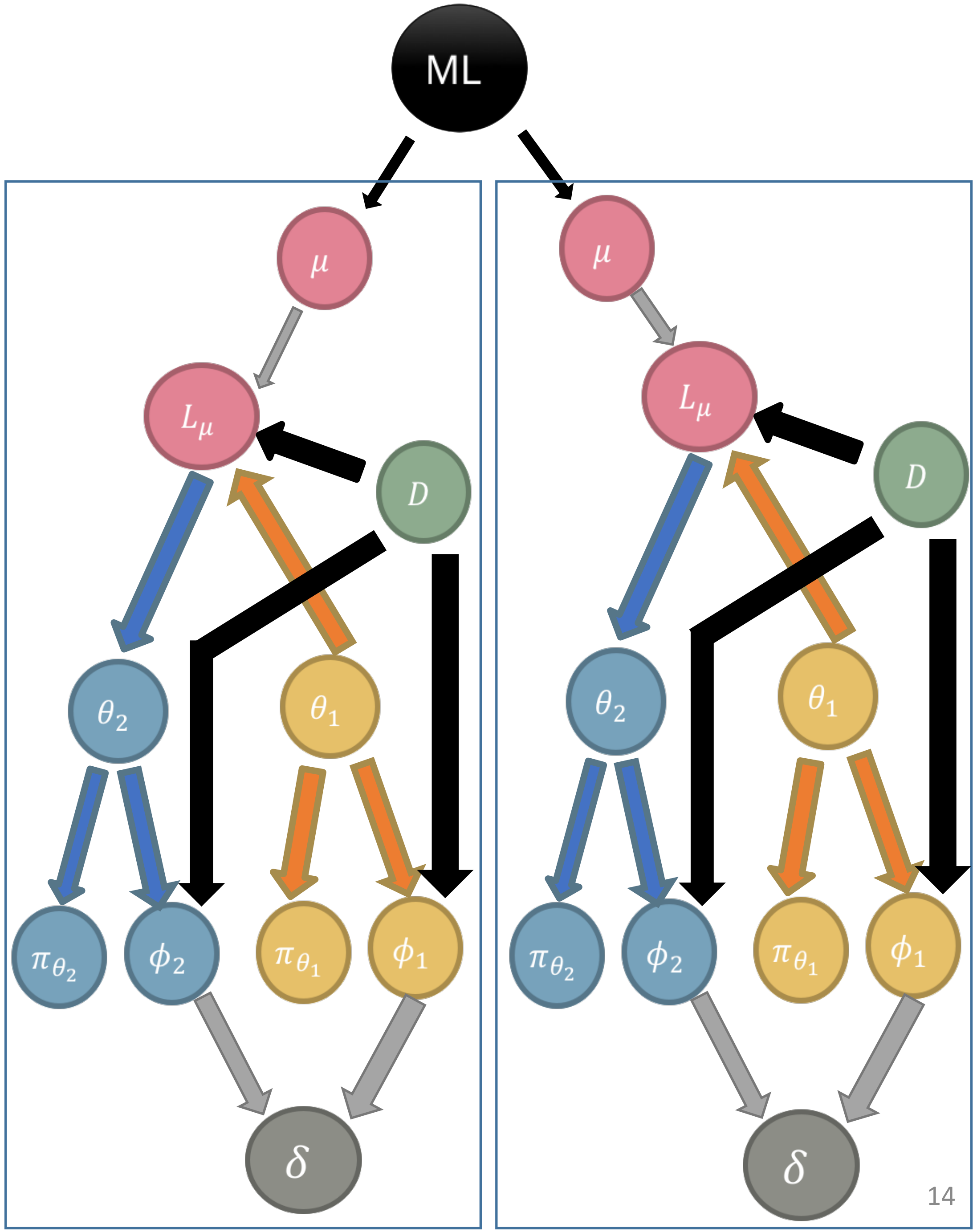
θ

$\mathbf{L}_\mu$

μ

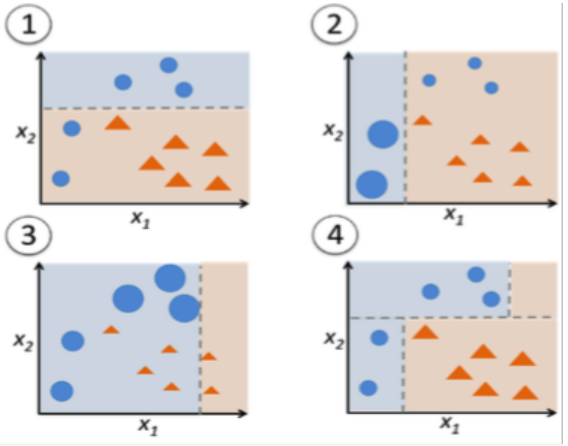
ML

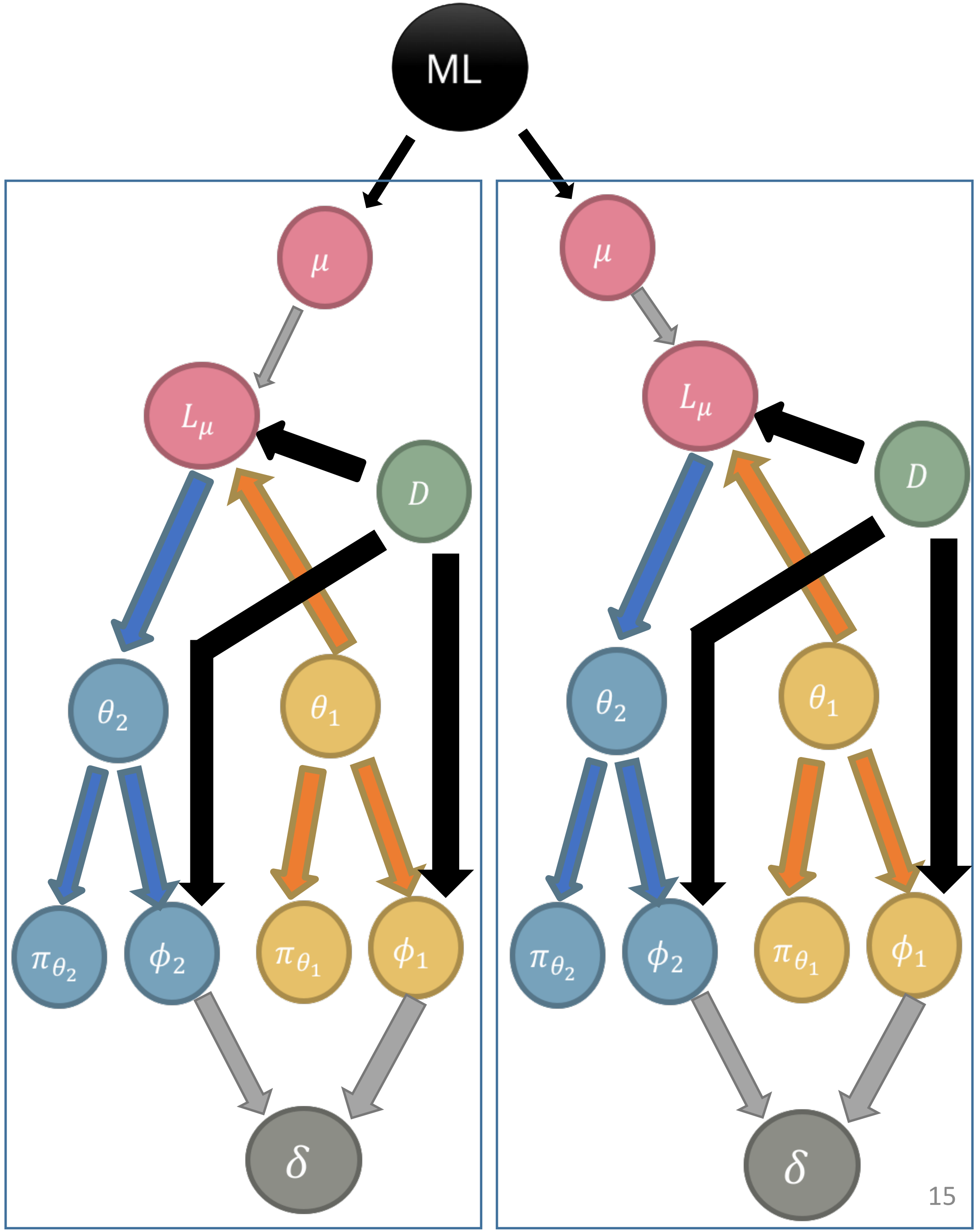




Ensemble Methods

D	Input/class samples
ϕ	Classification Errors
π_θ	Set of base-level classifiers
θ	Parameters of each classifier
$\mathbf{L}_\mu$	Supervised learning
μ	Number of classifiers, data subsets with sample weights
ML	Boosting





Scholarpedia Article

Early Days Meta-Learning Algorithms

- **Ensemble methods**
- **Success-Story Algorithm** *(Schmidhuber et al., 1997)*
- **Multiple learning algorithms** *(Rice, 1976)*
- **Meta-Genetic Programming** *(Schmidhuber, 1987)*
- **Fully self-referential learners: Gödel Machine** *(Schmidhuber, 2006, 2009)*
- **Neuro-evolution**
  - Originally used only to evolve the weights of a fixed architecture *(Miller et al., 1989)*
  - Later shown advantageous to simultaneously evolve the network architecture
    - **N**euro**E**volution of **A**ugmenting **T**opologies -NEAT- *(Stanley & Miikkulainen, 2002)*
    - **Hyper**cube-Based **N**euro**E**volution of **A**ugmenting **T**opologies -HyperNEAT- *(Stanley et al., 2009)*
    - **C**ompositional **P**attern **P**roducing **N**etworks -CPPNs- *(Stanley, 2007; Stanley et al., 2009)*

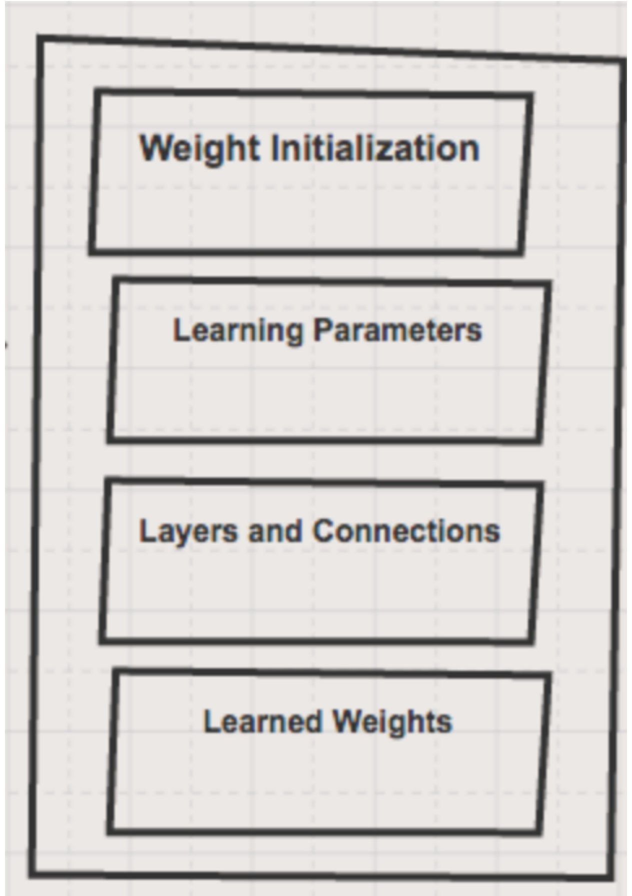
Index

- Formal Definition of Meta-Learning
- **Learning the Deep learning Architecture**
- Learning to Explore
- Learning to Seek Knowledge
- Learning to communicate

Learning to Learn the Deep Learning Network Architecture

$\mathbf{L}_\mu$	Deep NN
μ	Weights initialization, Learning parameters, Layers and connections, Learned weights
ML	

- Hyper-Parameter Optimization
- Reinforcement Learning for Architecture Design
- Hypernetworks
- Evolution



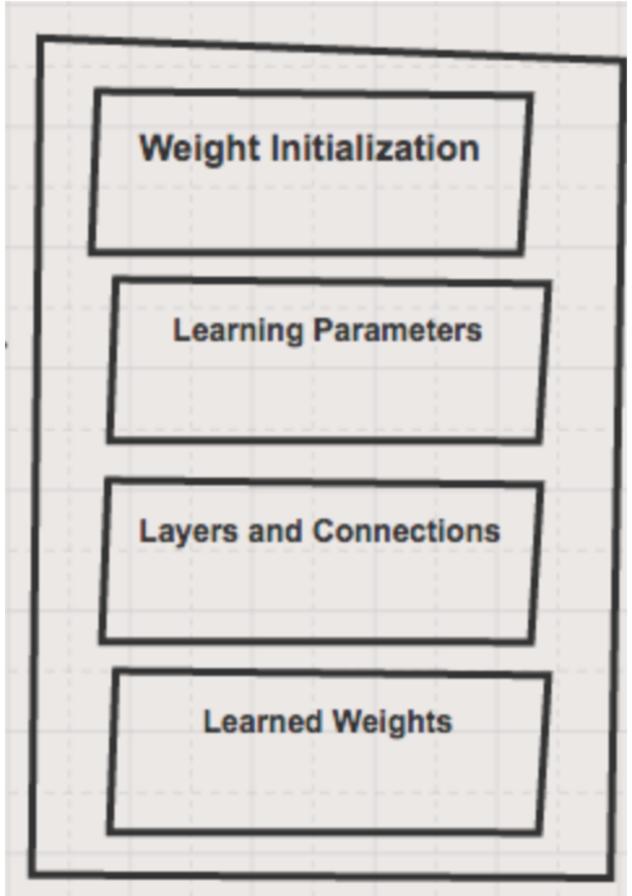
Blog by Carlos E. Perez

Learning to Learn the Deep Learning Network Architecture

$\mathbf{L}_\mu$ Deep NN

μ Weights initialization, Learning parameters, Layers and connections, Learned weights

ML
- **Hyper-Parameter Optimization**
- Reinforcement Learning for Architecture Design
- Hypernetworks
- Evolution



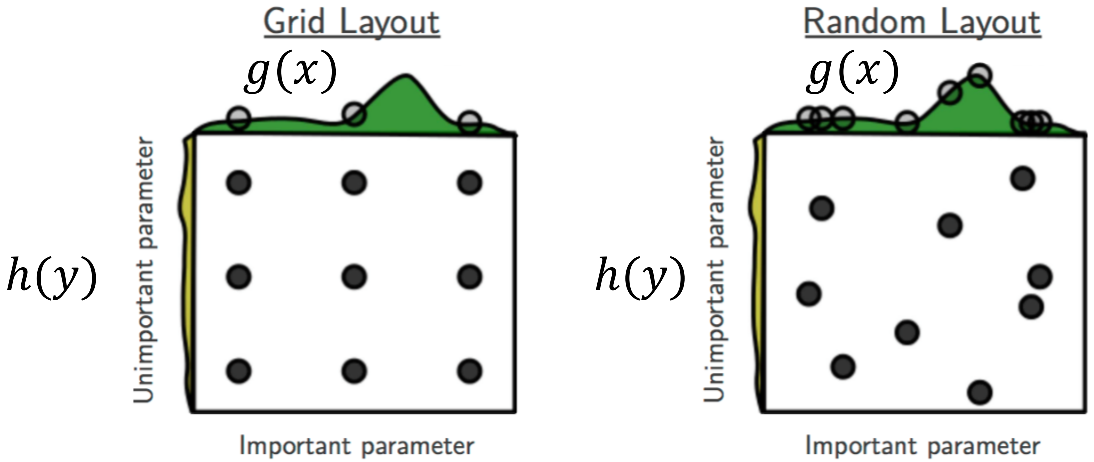
Blog by Carlos E. Perez

Hyper Parameters Optimization

> *Many machine learning algorithms have numerous hyperparameters that can be optimized. At Facebook's scale, a 1 percent improvement in accuracy for many models can have a meaningful impact on people's experiences. So with Flow, we built support for large-scale parameter sweeps and other AutoML features that leverage idle cycles to further improve these models.*

[Lec2]

- *Grid Search: exhaustively generates candidates from a grid of parameter values (usually uniformly distributed)*
- *Random Search*



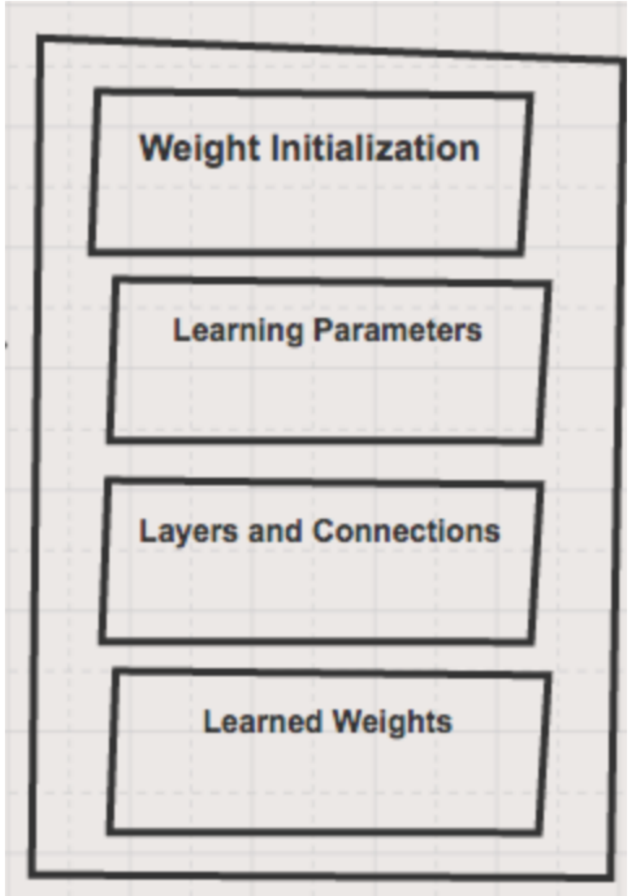
$$f(x,y) = g(x) + h(y)$$

Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research. 2012;13(Feb):281-305.

Learning to Learn the Deep Learning Network Architecture

$\mathbf{L}_{\mu}$	Deep NN
μ	Weights initialization, Learning parameters, Layers and connections, Learned weights
ML	

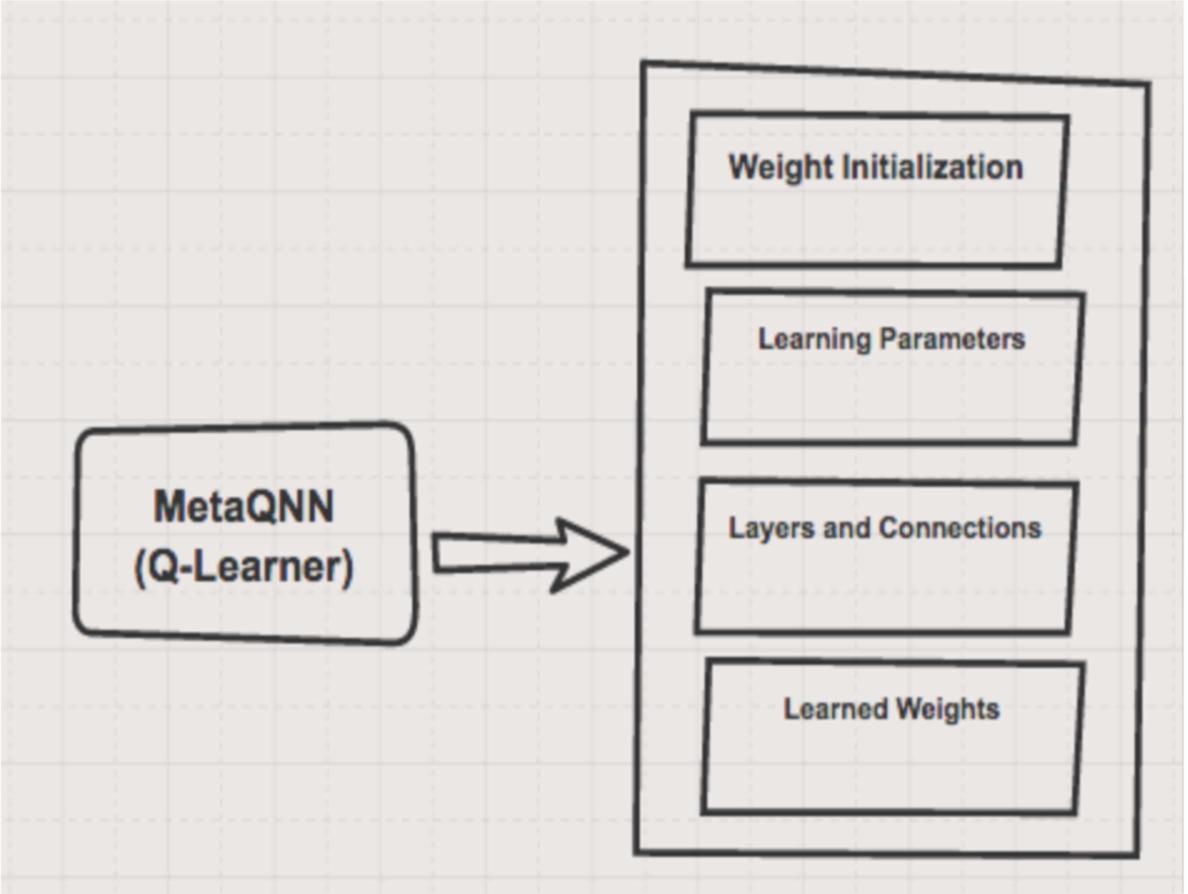
- Hyper-Parameter Optimization
- **Reinforcement Learning for Architecture Design**
- Hypernetworks
- Evolution



Blog by Carlos E. Perez

Reinforcement Q-Learning to discover CNN architectures

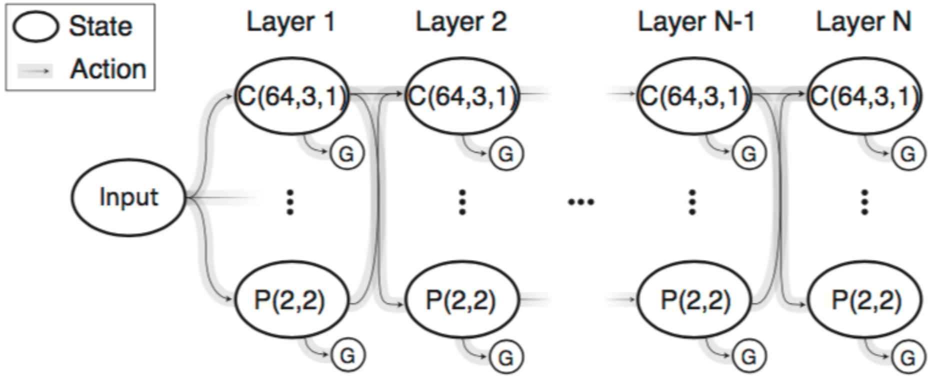
Given a **learning task**, automatically generate a **high performing** CNN architecture?



Blog by Carlos E. Perez

Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures
-State Space-



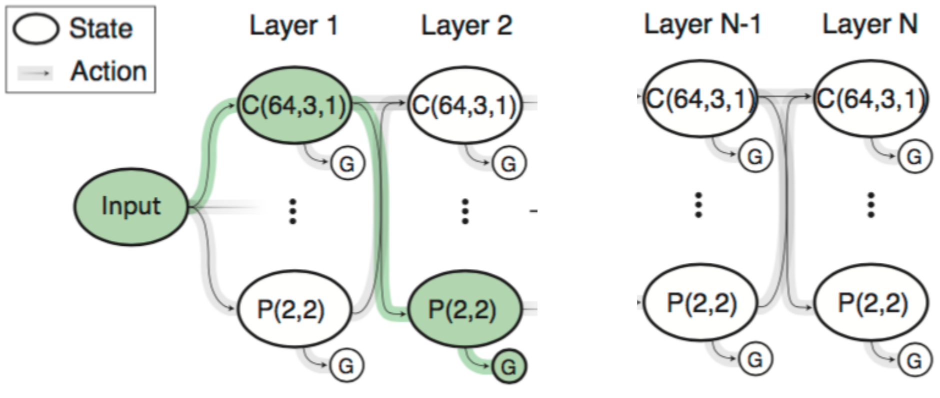
(a)

State = Tuple of relevant layer parameters

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

Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures
-Action Space-



(b)

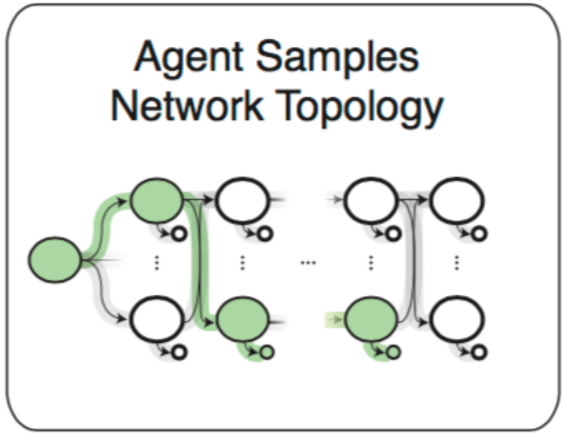
- **Action =** set of layers the agent might pick next given its current state
- **Constraints**
  - Limit the number of fc layers to max 2
  - From a state of type (C) we can transition to any other state type
  - From P we can not transition to P
  - Etc
  - …

Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures

-Training-

1. The agent **sequentially** selects **layers** via ***ε* greedy strategy** until it reaches a termination state

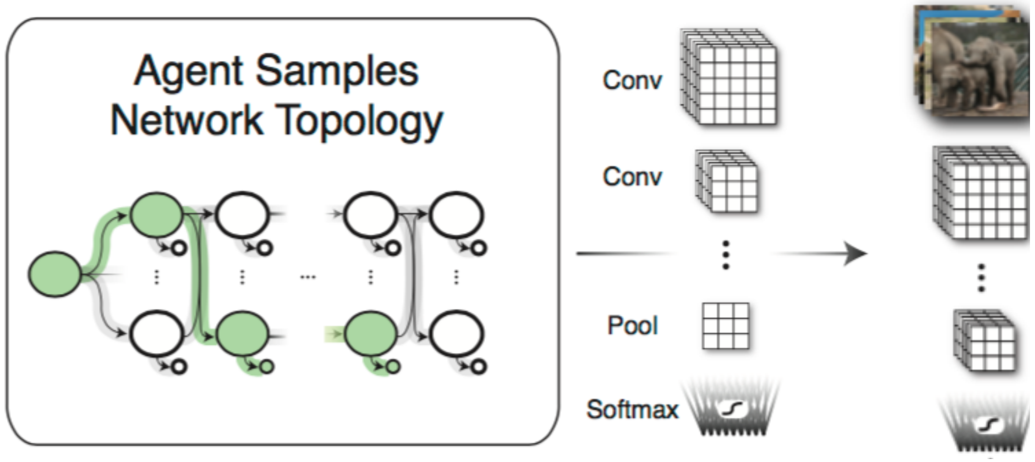


Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures

-Training-

2. The CNN architecture defined by the agent's path is **trained** on the chosen learning problem

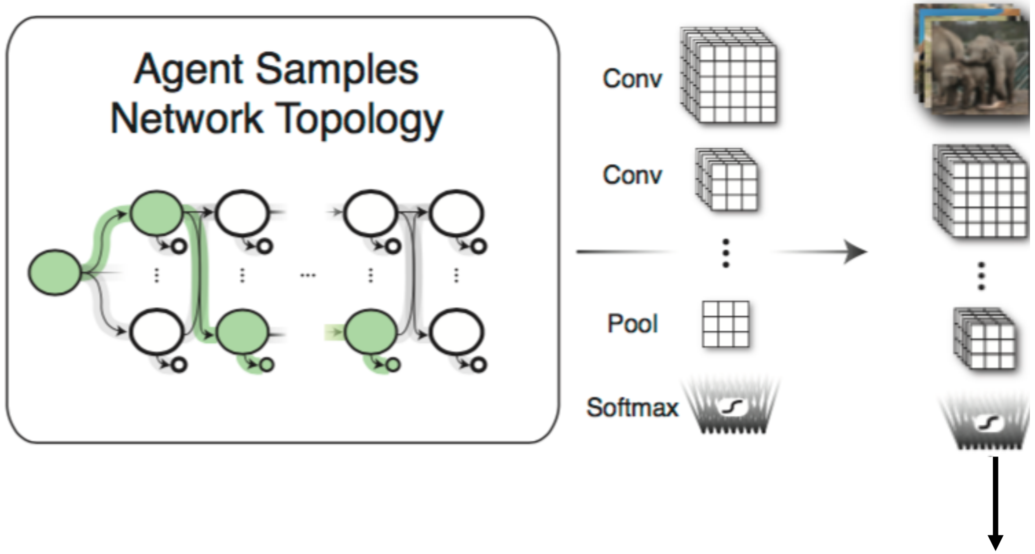


Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures

-Training-

3. The agent is given a **reward** equal to the **validation accuracy**



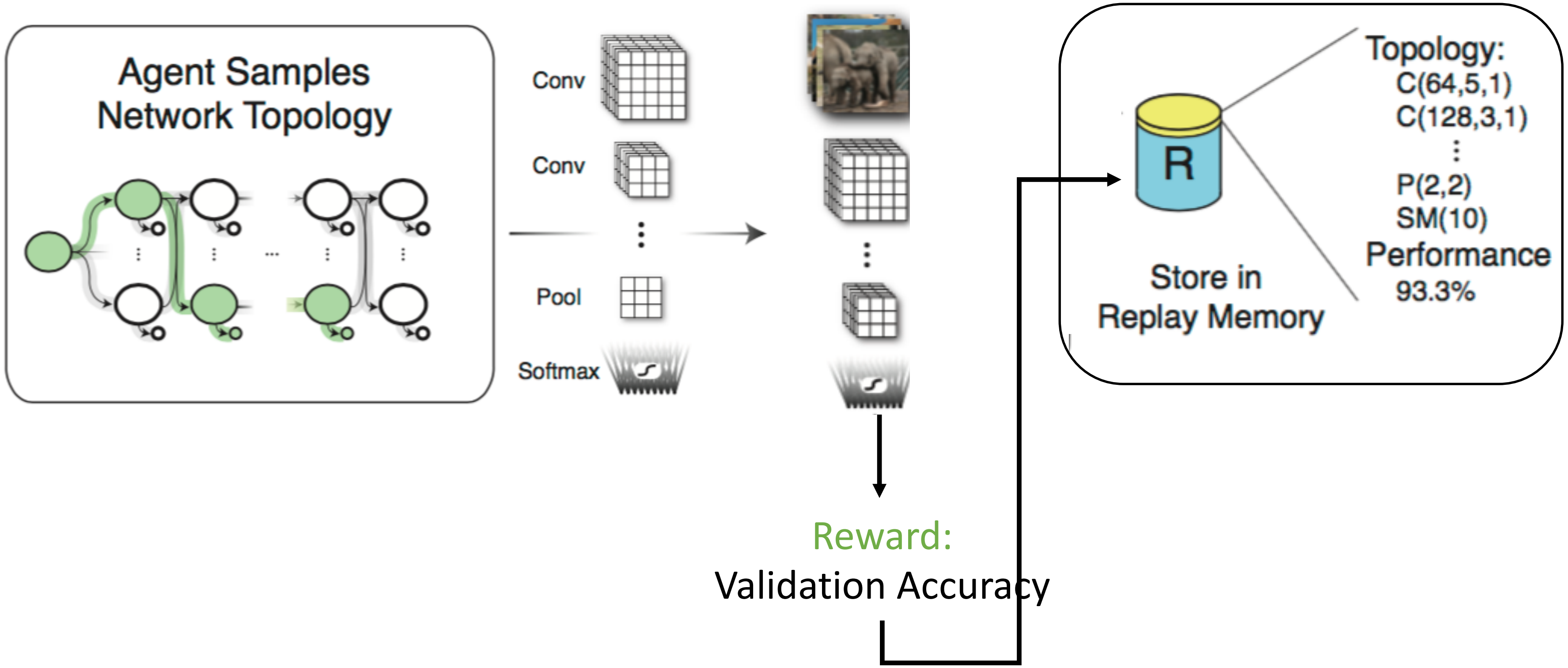
Reward:
Validation Accuracy

Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures

-Training-

4. The **validation accuracy** and the **architecture description** are stored in the **replay memory**

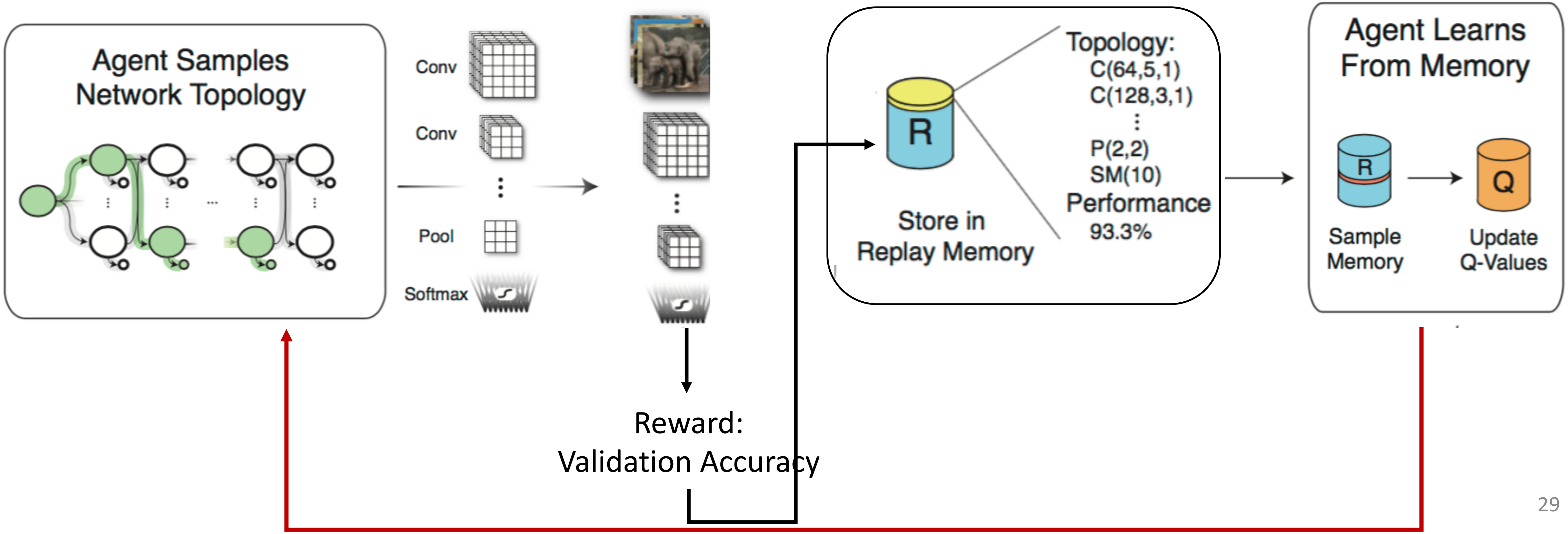


Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures

-Training-

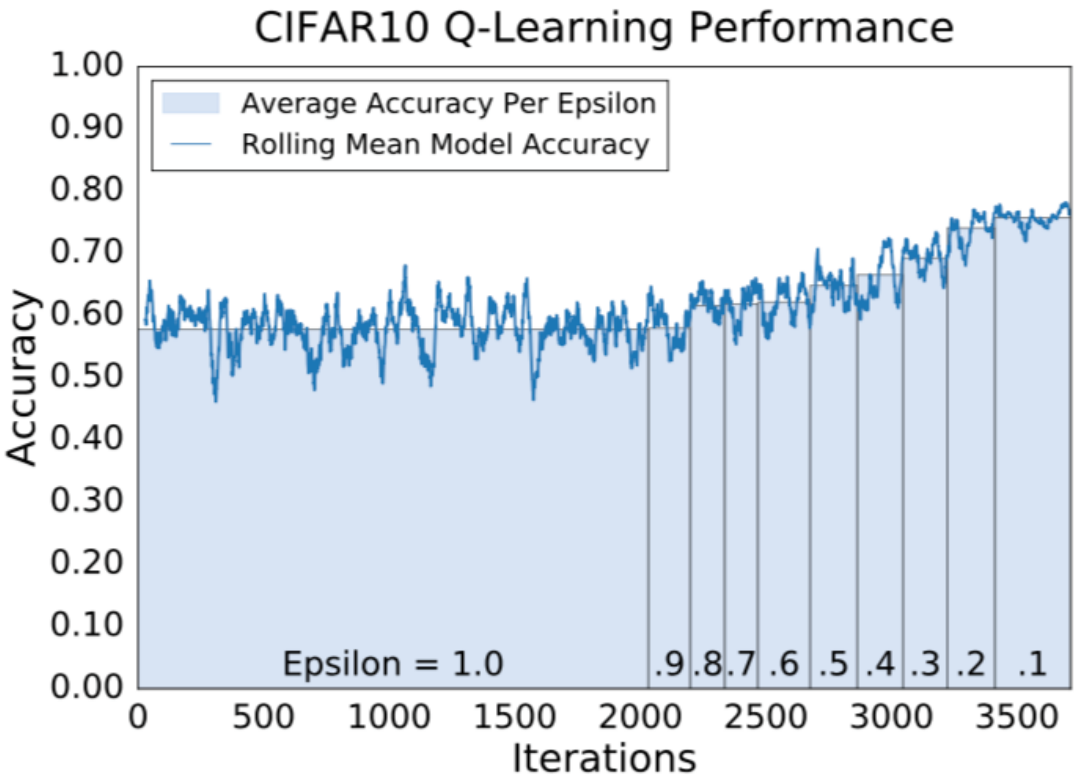
5. Experiences are sampled periodically from the **replay memory** to update Q-value



Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures

-Results-



Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Reinforcement Q-Learning to discover CNN architectures

-Results-

Method	CIFAR-10	MNIST	CIFAR-100
Resnet(110)(He et al., 2015)	6.61	-	-
Resnet(1001)(He et al., 2016)	**4.62**	-	**22.71**
VGGnet(Simonyan & Zisserman, 2014)	7.25	-	-
MetaQNN(ensemble)	7.32	**0.32**	-
MetaQNN(top model)	6.92	0.44	27.14

Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Top Model for CIFAR10
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**)

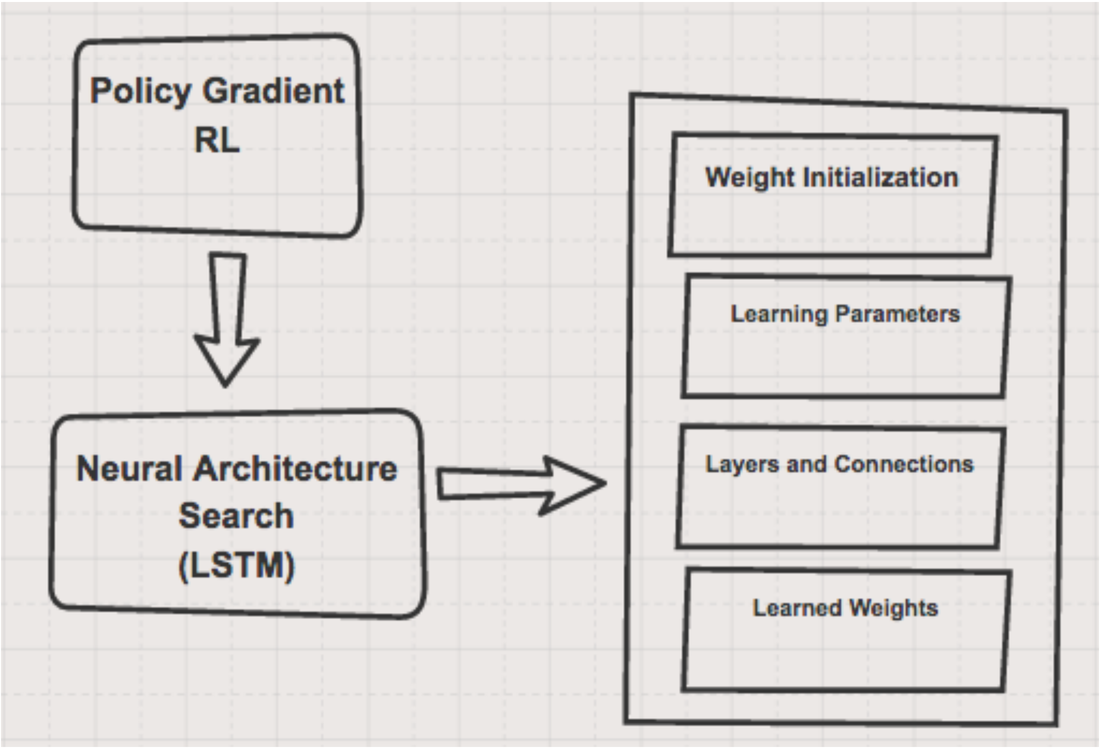
C: (#out_filter,filter_size,stride)
P: (filter_size,stride)

VGGnet
C(64,**3**,1)
C(64,**3**,1)
P(2,2)
C(128,**3**,1)
C(128,**3**,1)
P(2,2)
C(256,**3**,2)
C(256,**3**,1)
P(2,2)
C(512,**3**,2)
C(512,**3**,1)
P(2,2)
C(512,**3**,2)
C(512,**3**,1)
P(2,2)
Fc(4096)
Fc(4096)
Fc(1000)
SM(10)

Baker B, Gupta O, Naik N, Raskar R. Designing Neural Network Architectures using Reinforcement Learning. arXiv preprint arXiv:1611.02167. 2016 Nov 7.

Policy Gradient to Generate New CNN/RNN Architectures

Can we use an **RNN** to **automatically** generate a **description** of a **CNN/RNN network** with **high performance** on a given task?

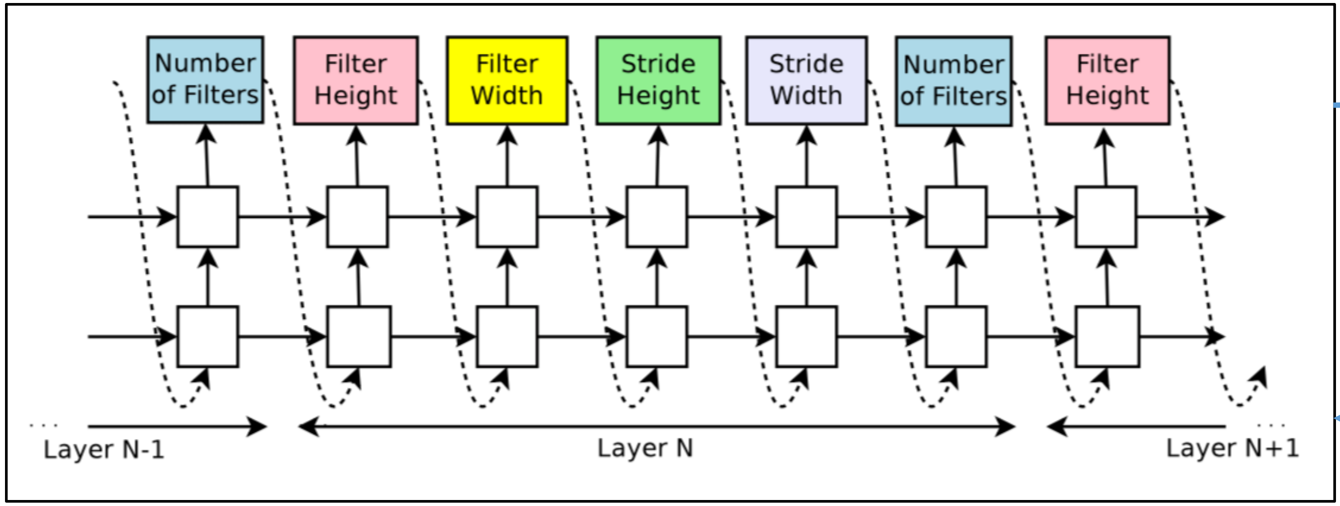


[Blog by Carlos E. Perez]()

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

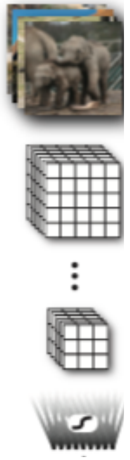
Policy Gradient to Generate New CNN/RNN Architectures -Training-

Controller RNN



CNN Architecture

Validation Accuracy



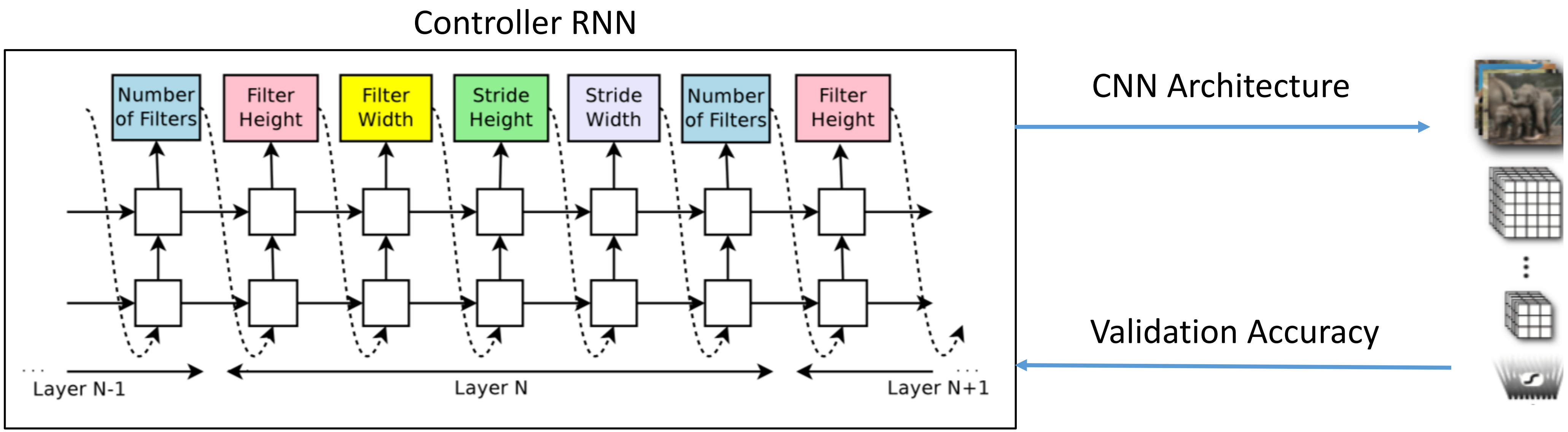
Search space

Non-linearities: rectifier linear units
Batch normalization
Skip connections
filter height [1,3,5,7]
Filter width [1,3,5,7]
Number of filter [24,36,48,64]
Strides [1,2,3]

1. RNN generates a **description** of a '**child**' neural network (CNN/RNN)

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

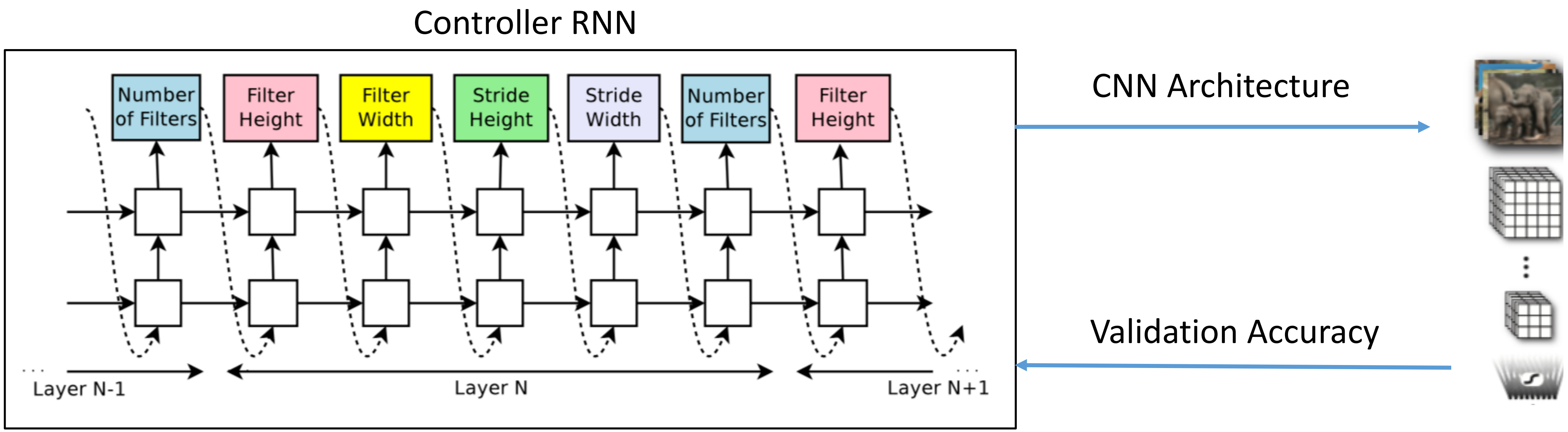
Policy Gradient to Generate New CNN Architectures -Training-



2. The **child network** is trained on a **validation data set** (50 epochs)

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

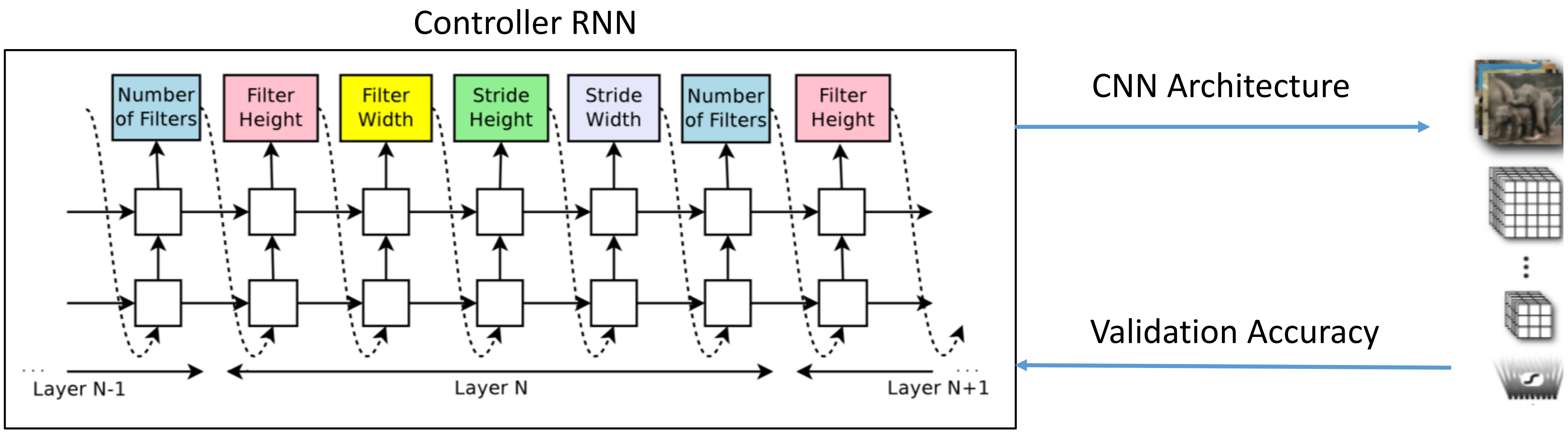
Policy Gradient to Generate New CNN Architectures -Training-



3. The **accuracy** on the validation data set at convergence is the **reward** for the controller RNN

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New CNN Architectures
-Training-



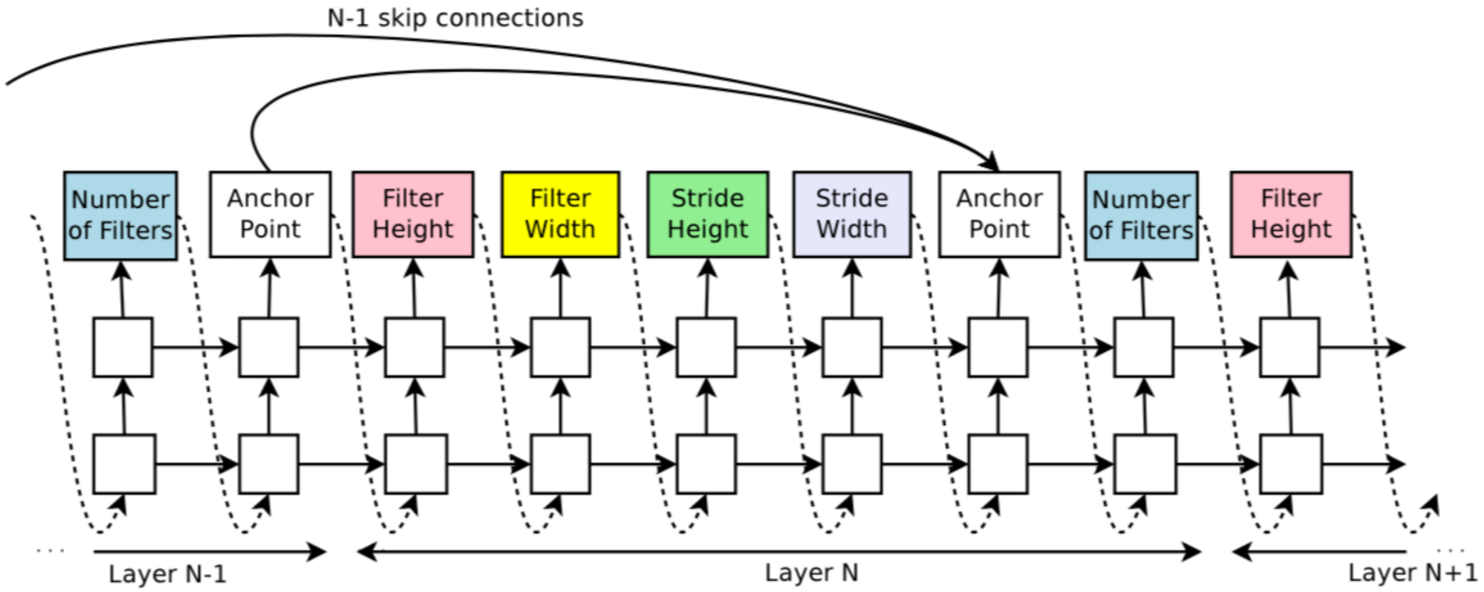
4. Policy gradient is used to update the parameters of the **RNN controller**

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New CNN Architectures -Skip Connections and Branching Layers for CNN-

- At layer N, add an anchor point which has N-1 content based sigmoids to indicate the previous layers that need to be connected

$$\mathrm{P(Layer\ j\ is\ an\ input\ to\ layer\ i)} = \mathrm{sigmoid}(v^{\mathrm{T}}\tanh(W_{prev} * h_j + W_{curr} * h_i)).$$



Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

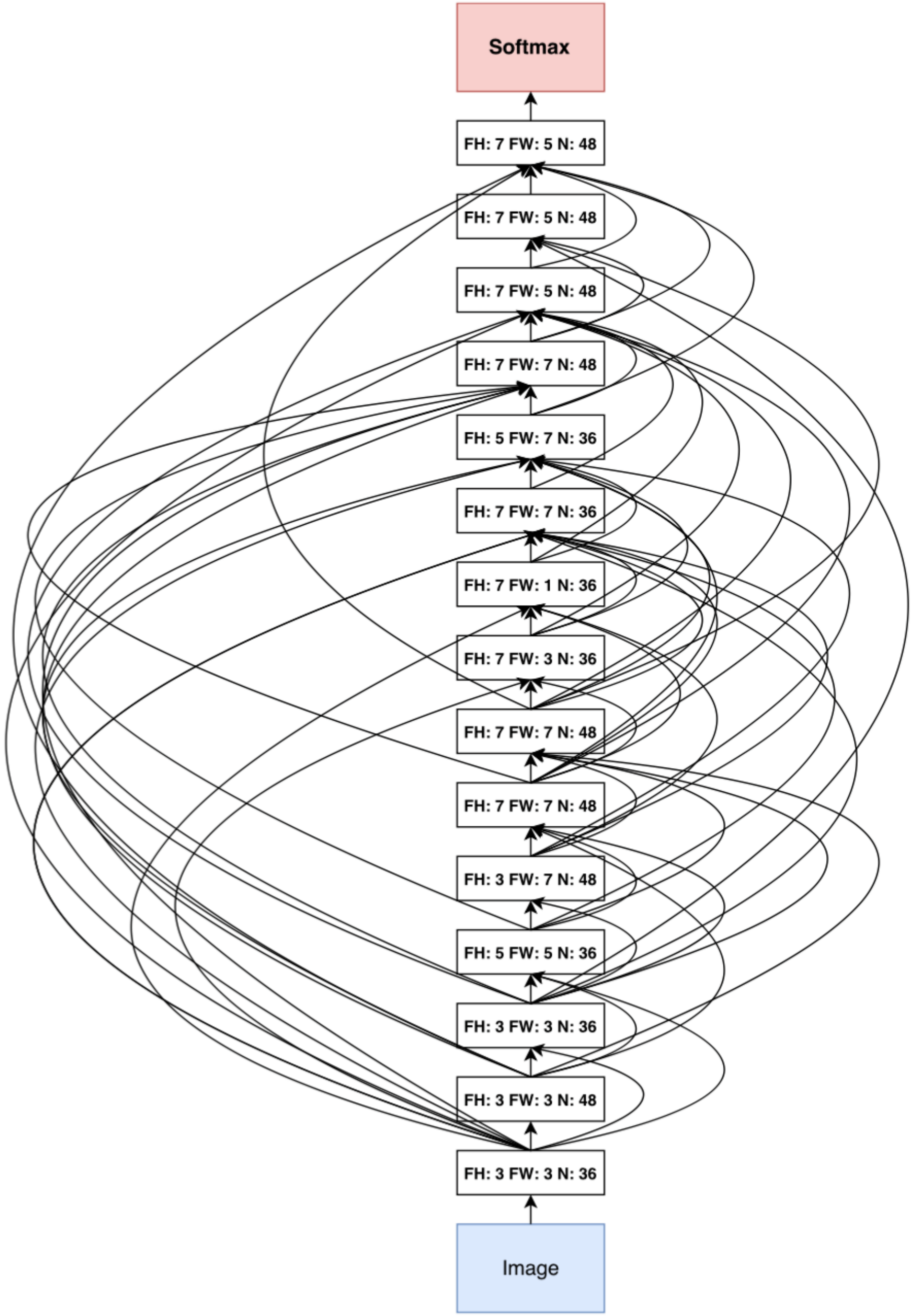
Policy Gradient to Generate New CNN Architectures
-Results-

Data set: CIFAR 10

Model	Depth	Parameters	Error rate (%)
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
Wide ResNet (Zagoruyko & Komodakis, 2016)	16	11.0M	4.81
	28	36.5M	4.17
ResNet (pre-activation) (He et al., 2016b)	164	1.7M	5.46
	1001	10.2M	4.62
Neural Architecture Search v1 no stride or pooling	15	4.2M	5.50
Neural Architecture Search v2 predicting strides	20	2.5M	6.01
Neural Architecture Search v3 max pooling	39	7.1M	4.47
Neural Architecture Search v3 max pooling + more filters	39	37.4M	3.65

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Discovered CNN



FH : Filter height
FW: Filter Width
N : Number of Filters

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New RNN Architectures -Generate Recurrent Cell Architectures-

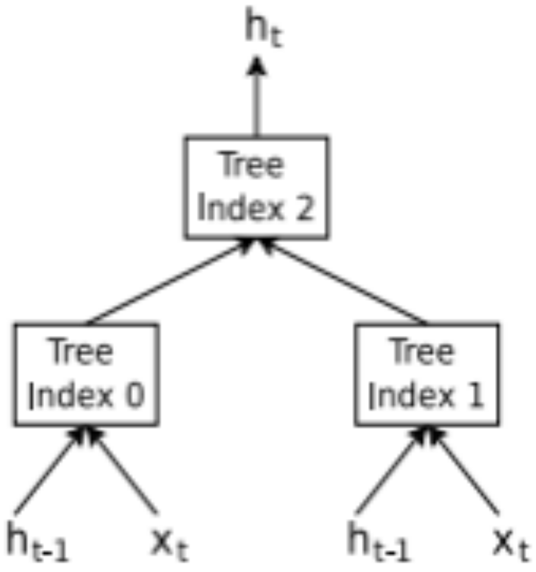
- The controller needs to find a functional form for h_t that takes x_t and h_{t-1} as inputs
    - RNN: $h_t = \tanh(W_1 \cdot x_t + W_2 \cdot h_{t-1})$
    - Better cell ?

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New RNN Architectures
-Generate Recurrent Cell Architectures-

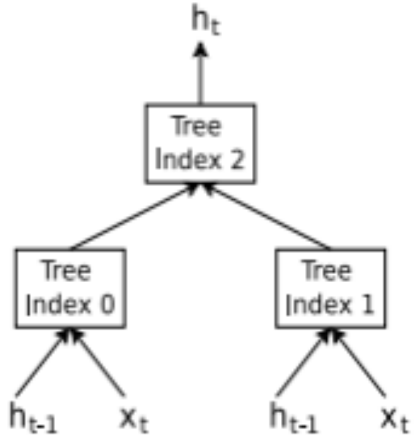
- The controller needs to find a functional form for h_t that takes x_t and h_{t-1} as inputs
  - RNN: $h_t = \tanh\left(W_1 \cdot x_t + W_2 \cdot h_{t-1}\right)$
  - Better cell ?

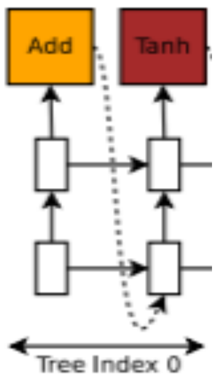
- **Cell Construction**
  - Tree of steps
  - The nodes in the tree are indexed in order
  - Controller RNN labels each step in the tree with :
    - a combination method (addition, multiplication,…)
    - an activation function (tanh, sigmoid…)
  - Consider two state variables c_t and c_{t-1}

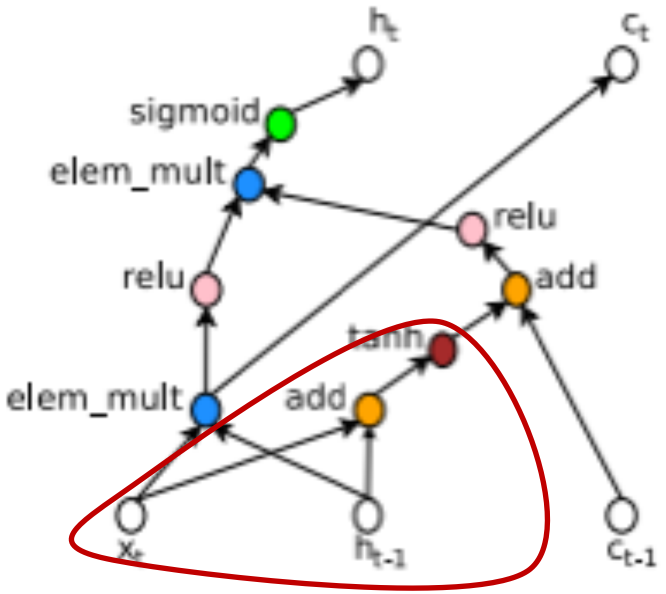


Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New RNN Architectures
-Generate Recurrent Cell Architectures-

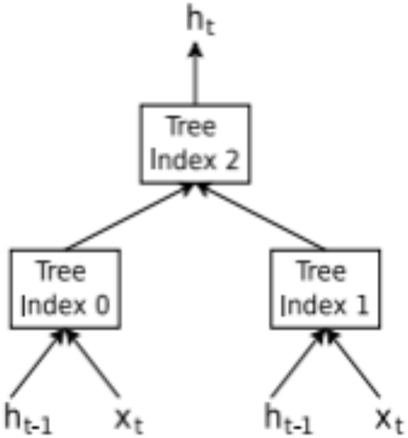


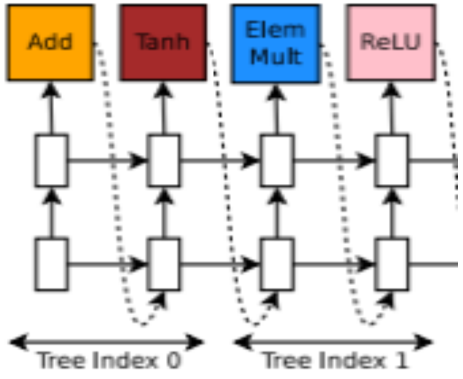


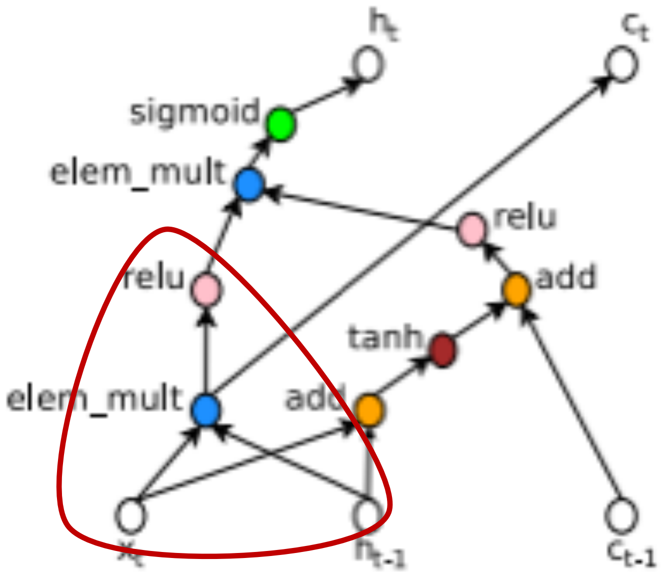


Index 0 $$a_0 = \tanh(W_1 * x_t + W_2 * h_{t-1})$$

Policy Gradient to Generate New RNN Architectures
-Generate Recurrent Cell Architectures-





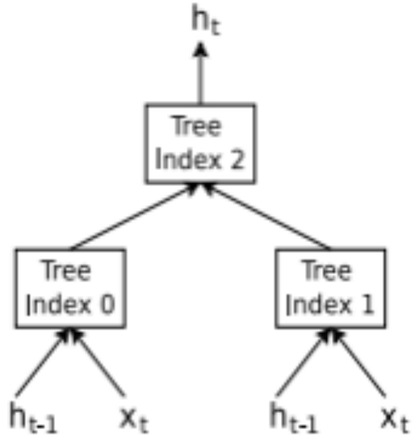


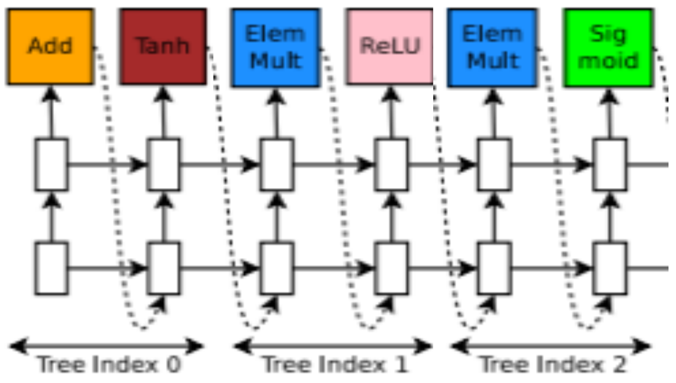
Index 0 $$a_0 = \tanh(W_1 * x_t + W_2 * h_{t-1})$$

Index 1 $$a_1 = \mathrm{ReLU}\big((W_3 * x_t) \odot (W_4 * h_{t-1})\big)$$

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New RNN Architectures
-Generate Recurrent Cell Architectures-

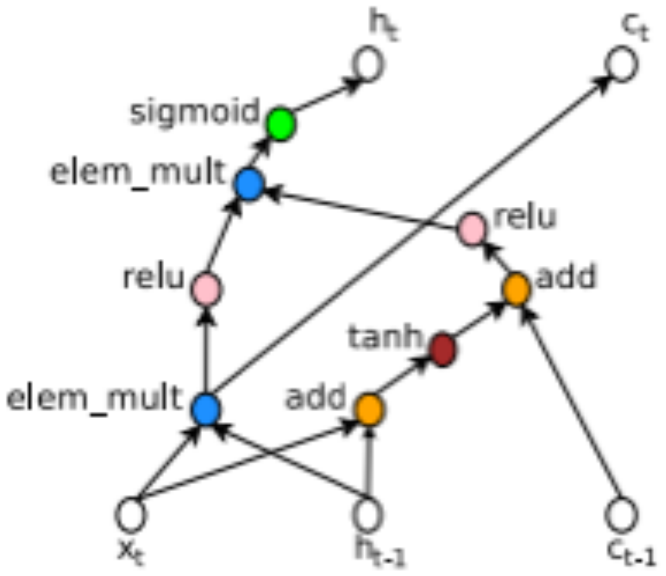




Index 0 $a_0 = \tanh(W_1 * x_t + W_2 * h_{t-1})$

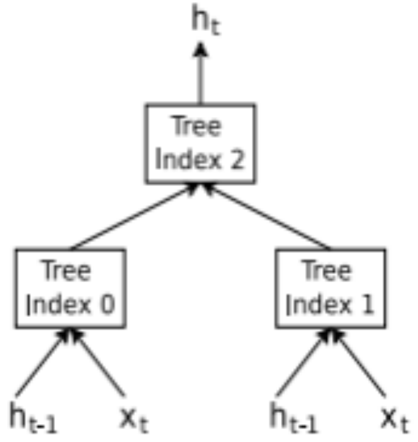
Index 1 $a_1 = \text{ReLU}\big((W_3 * x_t) \odot (W_4 * h_{t-1})\big)$

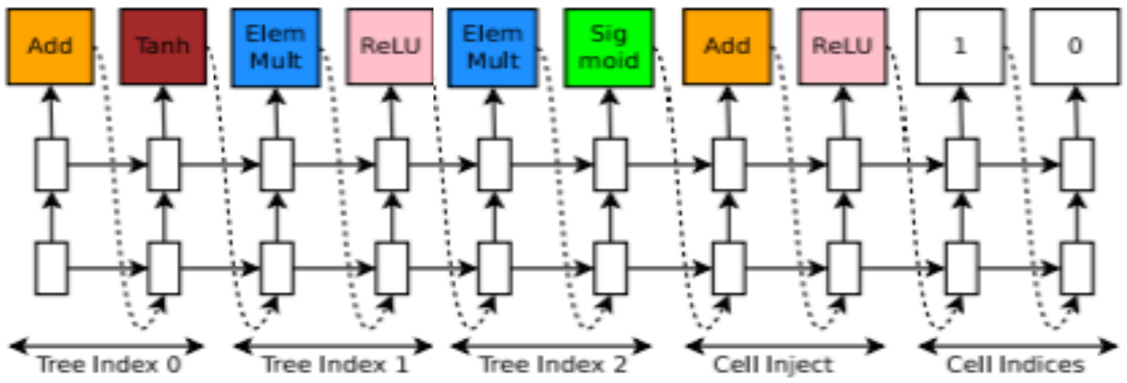
Index 2 $a_2 = \text{sigmoid}(a_0^{new} \odot a_1)$.



Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New RNN Architectures
-Generate Recurrent Cell Architectures-





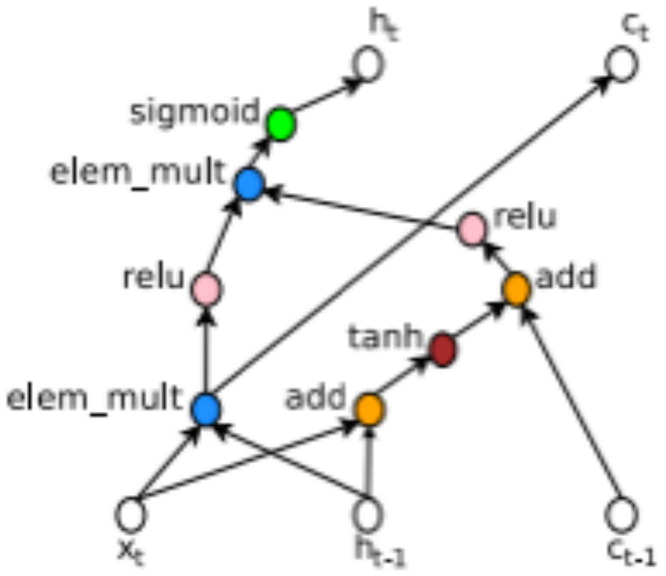
Index 0 $a_0 = \tanh(W_1 * x_t + W_2 * h_{t-1})$

Index 1 $a_1 = \mathrm{ReLU}\big((W_3 * x_t) \odot (W_4 * h_{t-1})\big)$.

Index 2 $a_2 = \mathrm{sigmoid}(a_0^{new} \odot a_1)$.

Cell Index $a_0^{new} = \mathrm{ReLU}(a_0 + c_{t-1})$

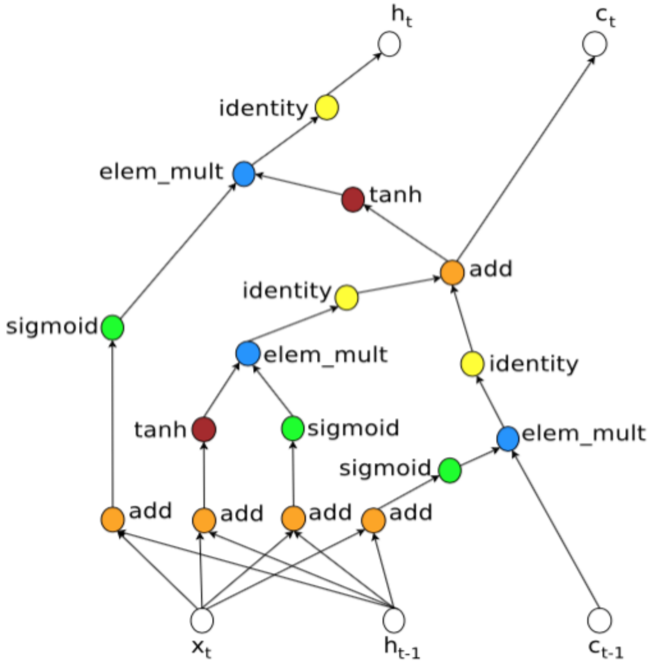
Cell Index $c_t = (W_3 * x_t) \odot (W_4 * h_{t-1})$



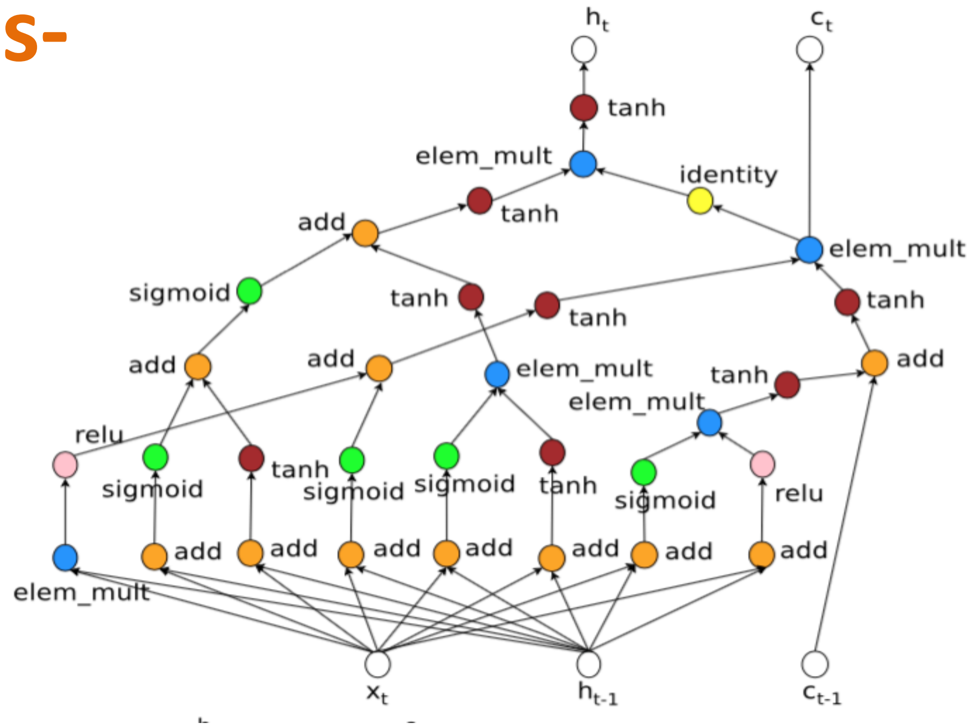
Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Policy Gradient to Generate New RNN Architectures -Results-

Data set: Penn Treebank



LSTM



Best Composed Cell

Model	Parameters	Test Perplexity
Mikolov & Zweig (2012) - RNN	6M‡	124.7
Zaremba et al. (2014) - LSTM (medium)	20M	82.7
Zaremba et al. (2014) - LSTM (large)	66M	78.4
Neural Architecture Search with base 8	32M	67.9
Neural Architecture Search with base 8 and shared embeddings	25M	64.0
Neural Architecture Search with base 8 and shared embeddings	54M	62.4

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

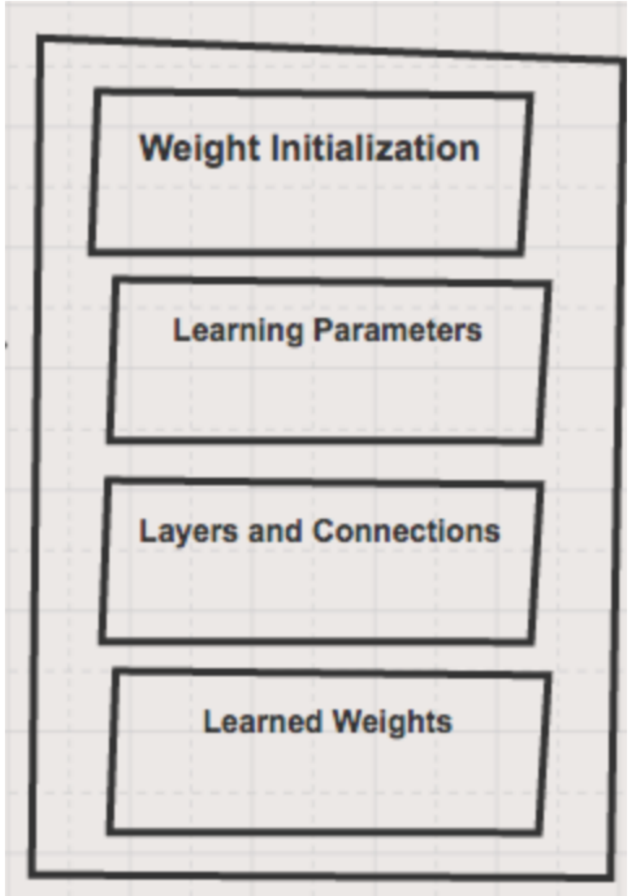
Learning to Learn the Deep Learning Network Architecture

$\mathbf{L}_{\mu}$ Deep NN

μ Weights initialization, Learning parameters, Layers and connections, Learned weights

ML

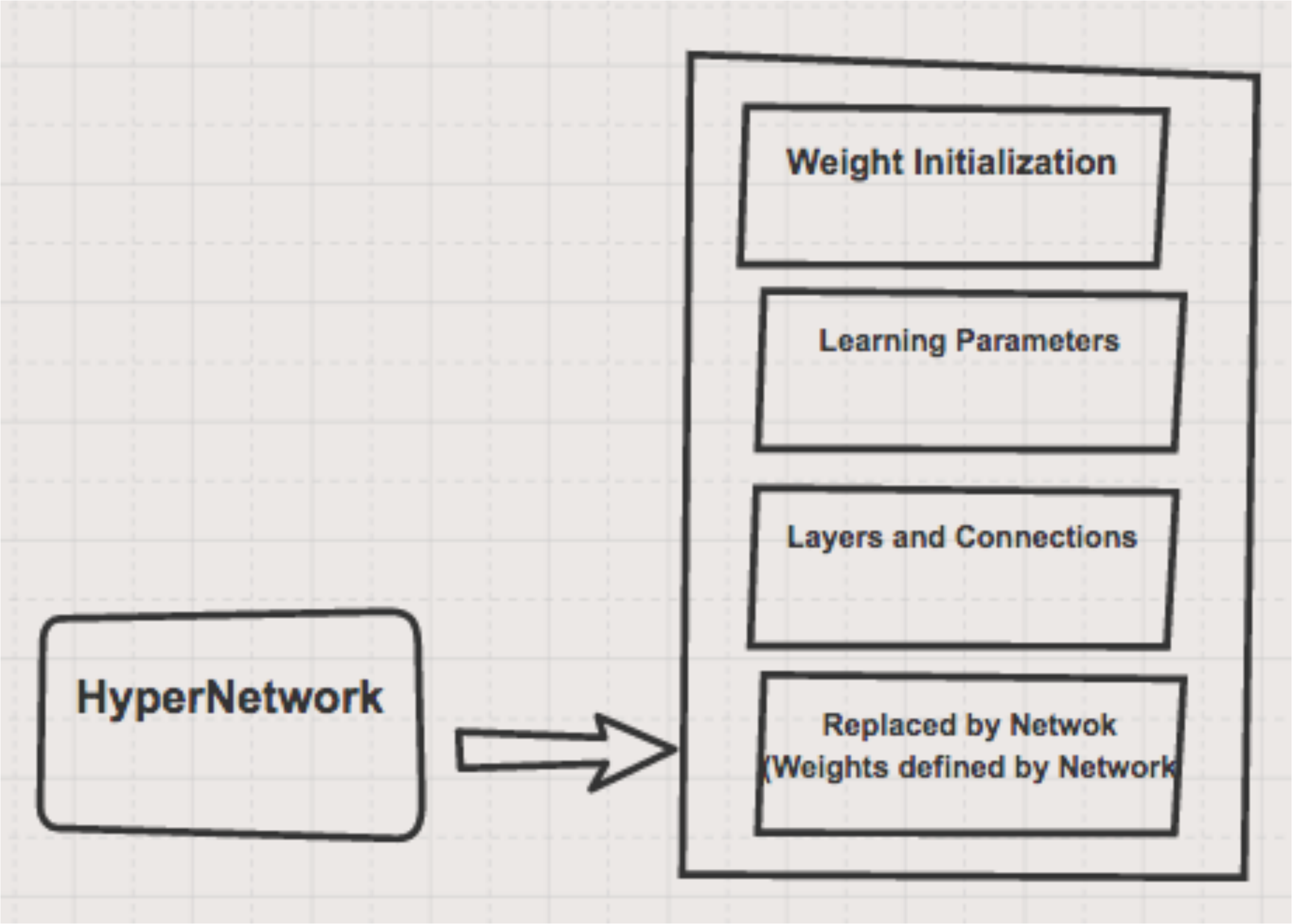
- Hyper-Parameter Optimization
- Reinforcement Learning for Architecture Design
- **Hypernetworks**
- Evolution



Blog by Carlos E. Perez

HyperNetworks

Can we use one network – a **"hypernetwork"** – to generate the weights for **another network**?



[Blog by Carlos E. Perez]

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

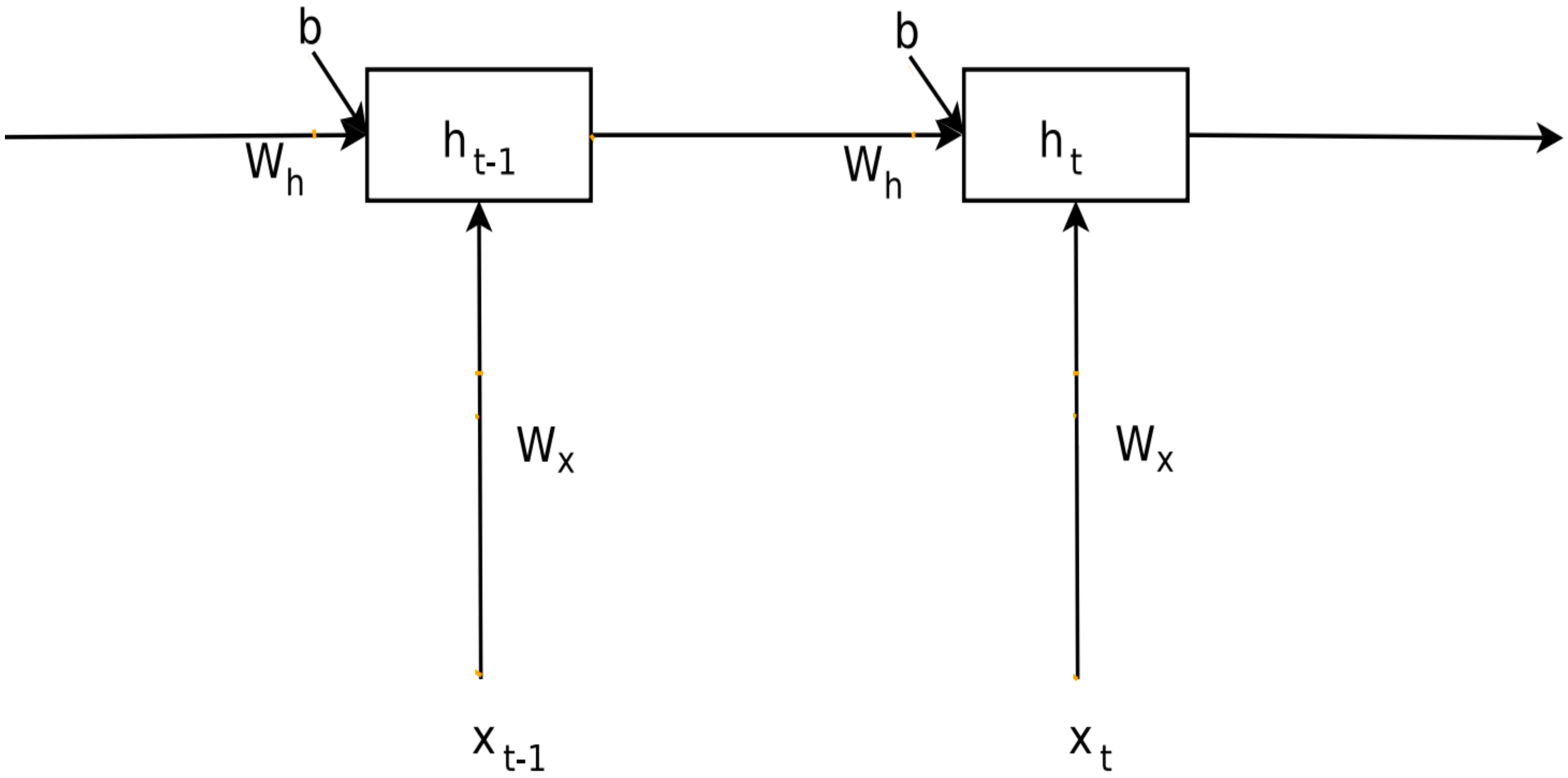
- Goal:
  - Use a **“hypernetwork”** to generate the weights for another network
  - Layer weights of main network computed as a function of a latent representation associated with each layer
  - Trained **end-to-end** with backpropagation

- Motivation:
  - RNNs impose weight sharing across layers → vanishing gradients, inflexible

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

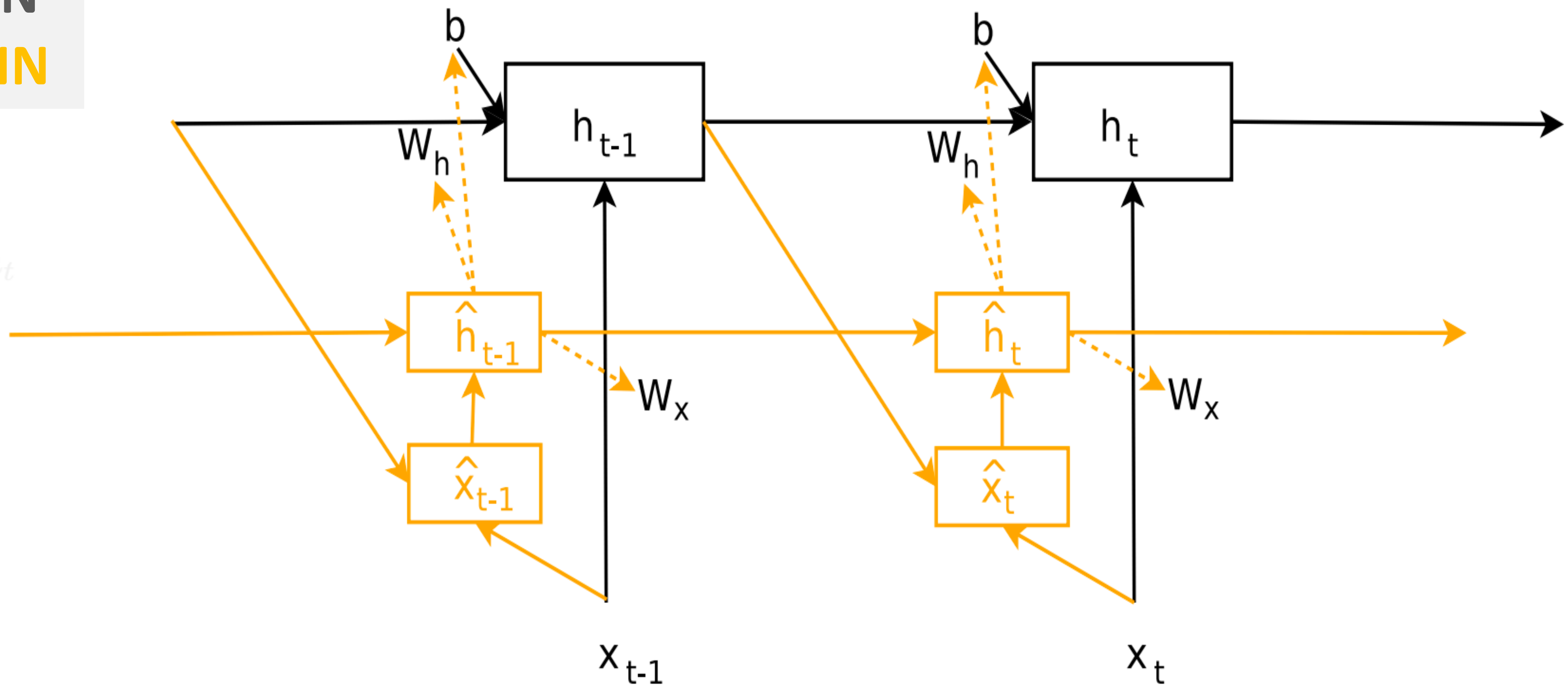
main RNN



D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

main RNN
HyperRNN

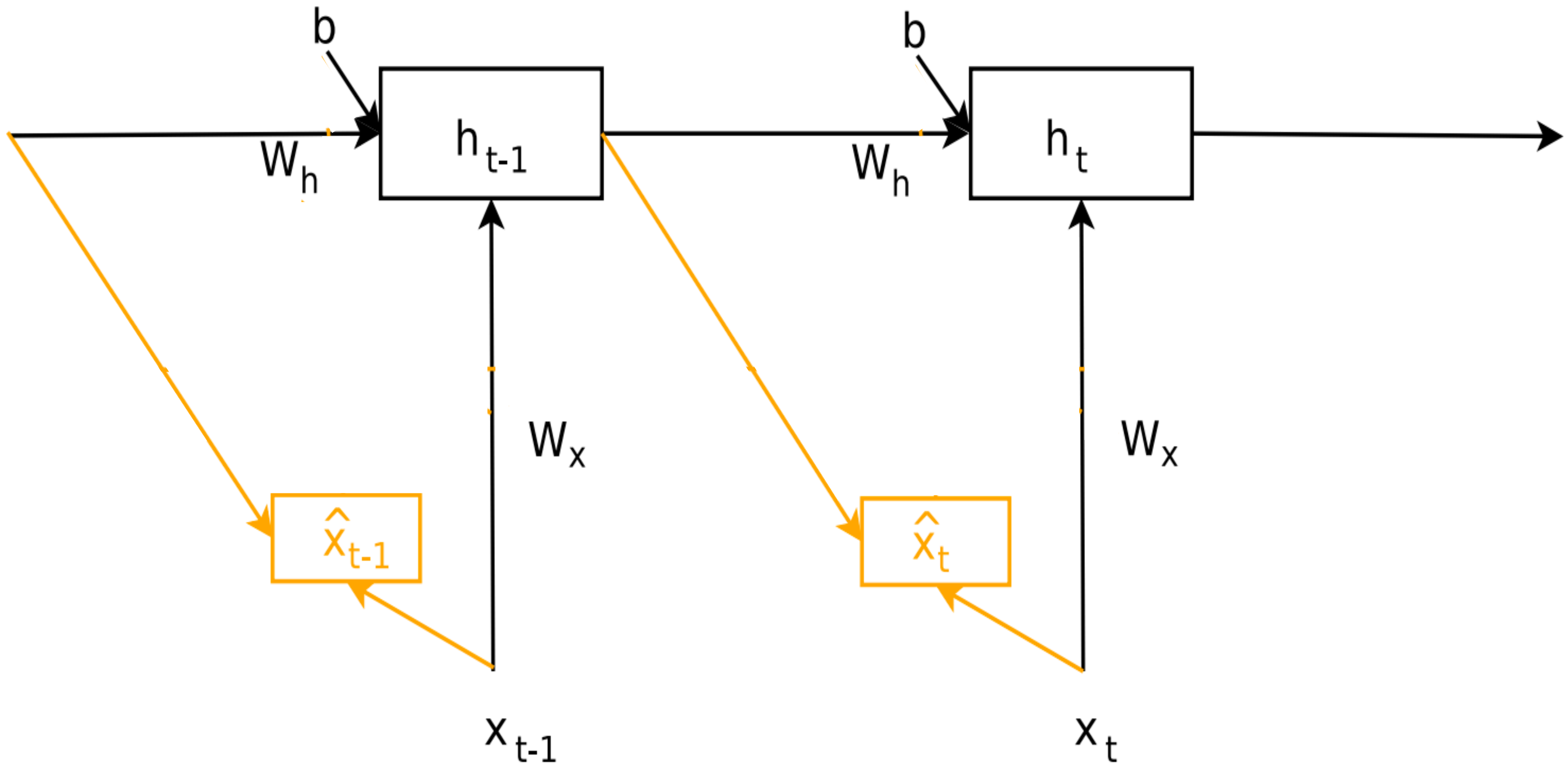


D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

main RNN

HyperRNN

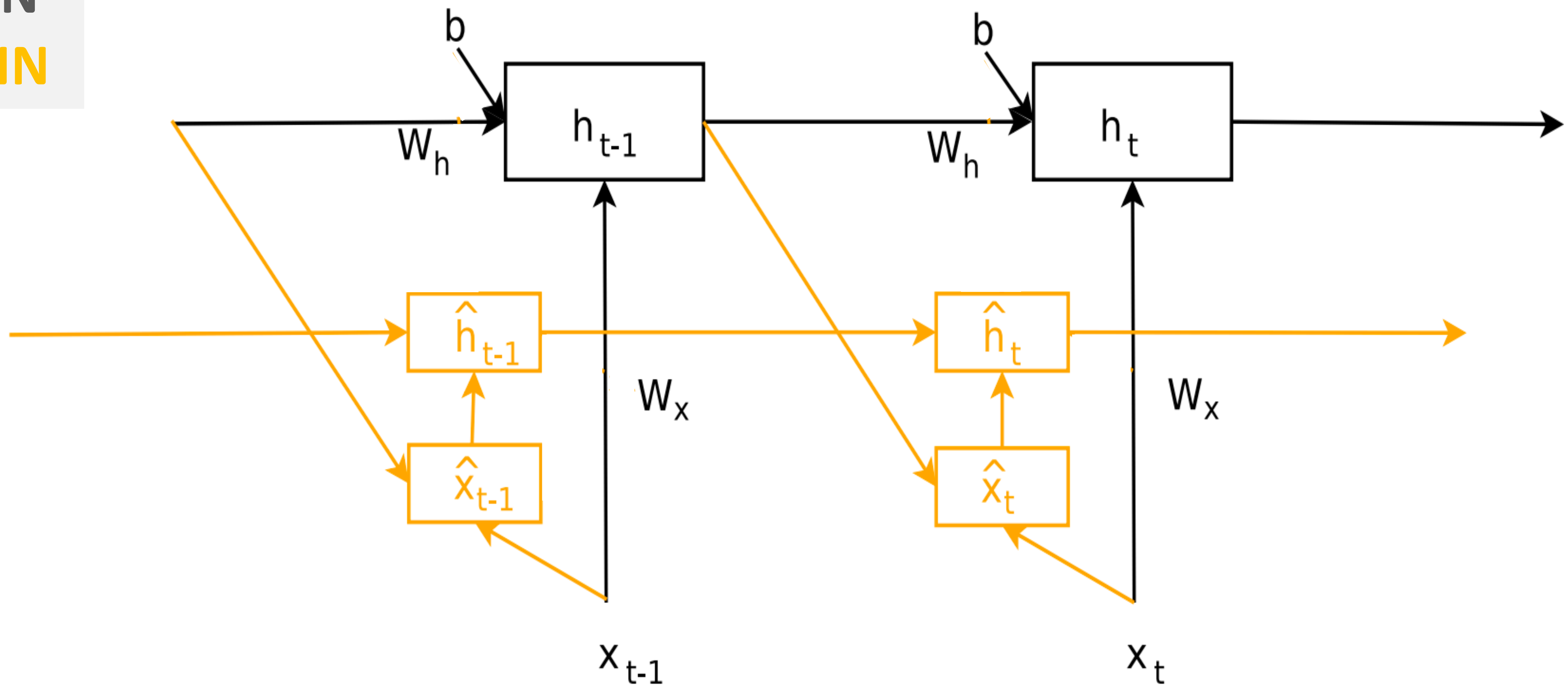


D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

main RNN

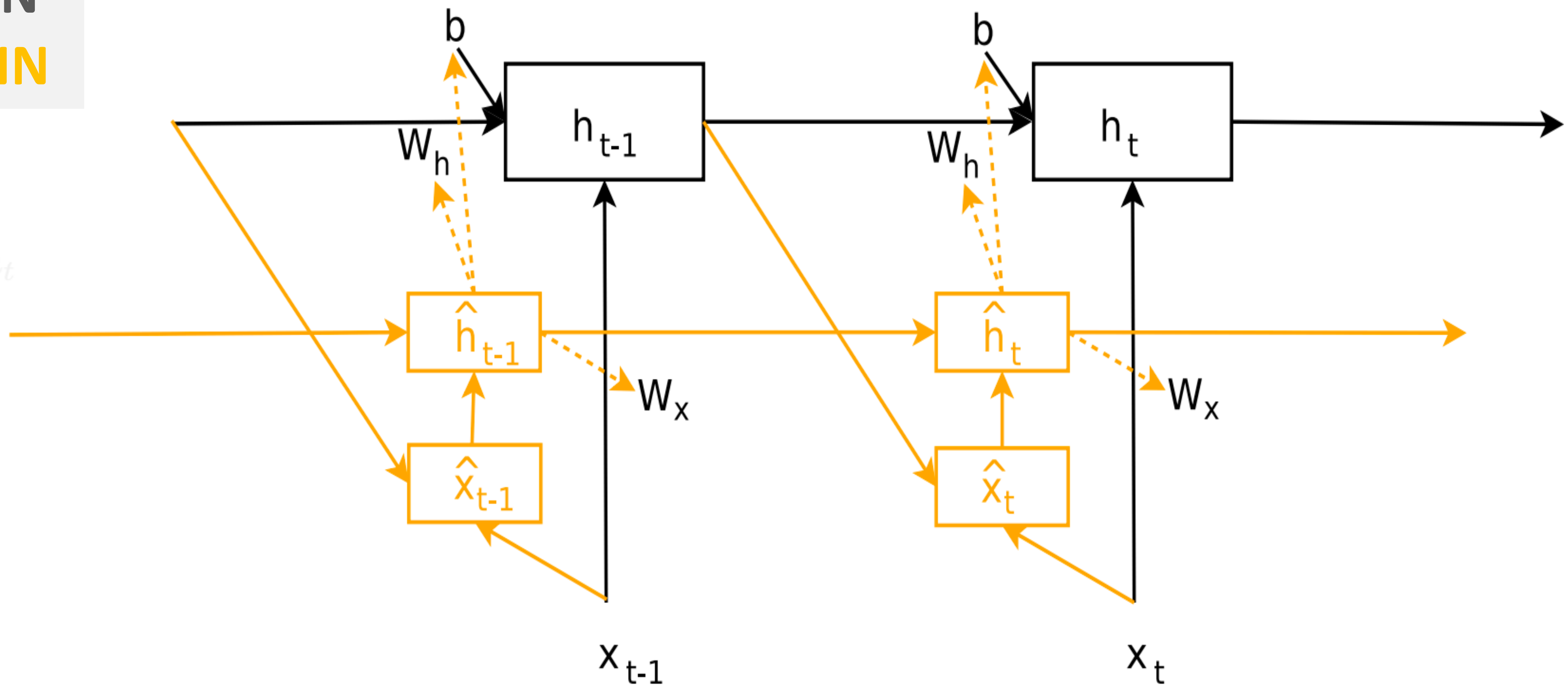
HyperRNN



D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

main RNN
HyperRNN



D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h}, \; W_x \in \mathbb{R}^{N_h \times N_x}, \; b \in \mathbb{R}^{N_h}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h},\ \ W_x \in \mathbb{R}^{N_h \times N_x},\ \ b \in \mathbb{R}^{N_h}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h},\ \ W_x \in \mathbb{R}^{N_h \times N_x},\ \ b \in \mathbb{R}^{N_h}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h}, \;\; W_x \in \mathbb{R}^{N_h \times N_x}, \;\; b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big),$$

$$\begin{aligned} z_h, z_x, z_z &\in \mathbb{R}^{N_z} \\ W_{hz} &\in \mathbb{R}^{N_h \times N_h \times N_z}, \\ W_{xz} &\in \mathbb{R}^{N_h \times N_x \times N_z}, \\ W_{bz} &\in \mathbb{R}^{N_h \times N_z}, \\ b_0 &\in \mathbb{R}^{N_h} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h},\ W_x \in \mathbb{R}^{N_h \times N_x},\ b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h) h_{t-1} + W_x(z_x) x_t + b(z_b)\big),$$

$$\begin{aligned} z_h, z_x, z_z &\in \mathbb{R}^{N_z} \\ W_{hz} &\in \mathbb{R}^{N_h \times N_h \times N_z}, \\ W_{xz} &\in \mathbb{R}^{N_h \times N_x \times N_z}, \\ W_{bz} &\in \mathbb{R}^{N_h \times N_z}, \\ b_0 &\in \mathbb{R}^{N_h} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h},\;\; W_x \in \mathbb{R}^{N_h \times N_x},\;\; b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big),$$

$$\begin{aligned} z_h, z_x, z_z &\in \mathbb{R}^{N_z} \\ W_{hz} &\in \mathbb{R}^{N_h \times N_h \times N_z}, \\ W_{xz} &\in \mathbb{R}^{N_h \times N_x \times N_z}, \\ W_{bz} &\in \mathbb{R}^{N_h \times N_z}, \\ b_0 &\in \mathbb{R}^{N_h} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h}, \;\; W_x \in \mathbb{R}^{N_h \times N_x}, \;\; b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big),$$

$$\begin{aligned} W_h(z_h) &= \\ W_x(z_x) &= \\ b(z_b) &= \end{aligned}$$

$$\begin{aligned} z_h, z_x, z_z &\in \mathbb{R}^{N_z} \\ W_{hz} &\in \mathbb{R}^{N_h \times N_h \times N_z}, \\ W_{xz} &\in \mathbb{R}^{N_h \times N_x \times N_z}, \\ W_{bz} &\in \mathbb{R}^{N_h \times N_z}, \\ b_0 &\in \mathbb{R}^{N_h} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h}, \;\; W_x \in \mathbb{R}^{N_h \times N_x}, \;\; b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big),$$

$$\begin{aligned} W_h(z_h) &= \langle W_{hz}, z_h \rangle \\ W_x(z_x) &= \langle W_{xz}, z_x \rangle \\ b(z_b) &= W_{bz} z_b + b_0 \end{aligned}$$

$$\begin{aligned} z_h, z_x, z_z &\in \mathbb{R}^{N_z} \\ W_{hz} &\in \mathbb{R}^{N_h \times N_h \times N_z}, \\ W_{xz} &\in \mathbb{R}^{N_h \times N_x \times N_z}, \\ W_{bz} &\in \mathbb{R}^{N_h \times N_z}, \\ b_0 &\in \mathbb{R}^{N_h} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h},\ \ W_x \in \mathbb{R}^{N_h \times N_x},\ \ b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big),$$

$$\begin{aligned} W_h(z_h) &= \langle W_{hz}, z_h \rangle \\ W_x(z_x) &= \langle W_{xz}, z_x \rangle \\ b(z_b) &= W_{bz} z_b + b_0 \end{aligned}$$

$$\begin{aligned} z_h, z_x, z_z &\in \mathbb{R}^{N_z} \\ W_{hz} &\in \mathbb{R}^{N_h \times N_h \times N_z}, \\ W_{xz} &\in \mathbb{R}^{N_h \times N_x \times N_z}, \\ W_{bz} &\in \mathbb{R}^{N_h \times N_z}, \\ b_0 &\in \mathbb{R}^{N_h} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h}, \;\; W_x \in \mathbb{R}^{N_h \times N_x}, \;\; b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big),$$

$$\begin{aligned} W_h(z_h) &= \langle W_{hz}, z_h \rangle \\ W_x(z_x) &= \langle W_{xz}, z_x \rangle \\ b(z_b) &= W_{bz} z_b + b_0 \end{aligned}$$

$$\begin{aligned} z_h, z_x, z_z &\in \mathbb{R}^{N_z} \\ W_{hz} &\in \mathbb{R}^{N_h \times N_h \times N_z}, \\ W_{xz} &\in \mathbb{R}^{N_h \times N_x \times N_z}, \\ W_{bz} &\in \mathbb{R}^{N_h \times N_z}, \\ b_0 &\in \mathbb{R}^{N_h} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

Recall standard RNN formulation:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \qquad W_h \in \mathbb{R}^{N_h \times N_h}, \;\; W_x \in \mathbb{R}^{N_h \times N_x}, \;\; b \in \mathbb{R}^{N_h}$$

HyperRNN: weights allowed to float over time

$$h_t = \phi\big(W_h(z_h)h_{t-1} + W_x(z_x)x_t + b(z_b)\big),$$

$$\begin{aligned} W_h(z_h) &= \langle W_{hz}, z_h \rangle \\ W_x(z_x) &= \langle W_{xz}, z_x \rangle \\ b(z_b) &= W_{bz} z_b + b_0 \end{aligned}$$

how to compute these embedding vectors?

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

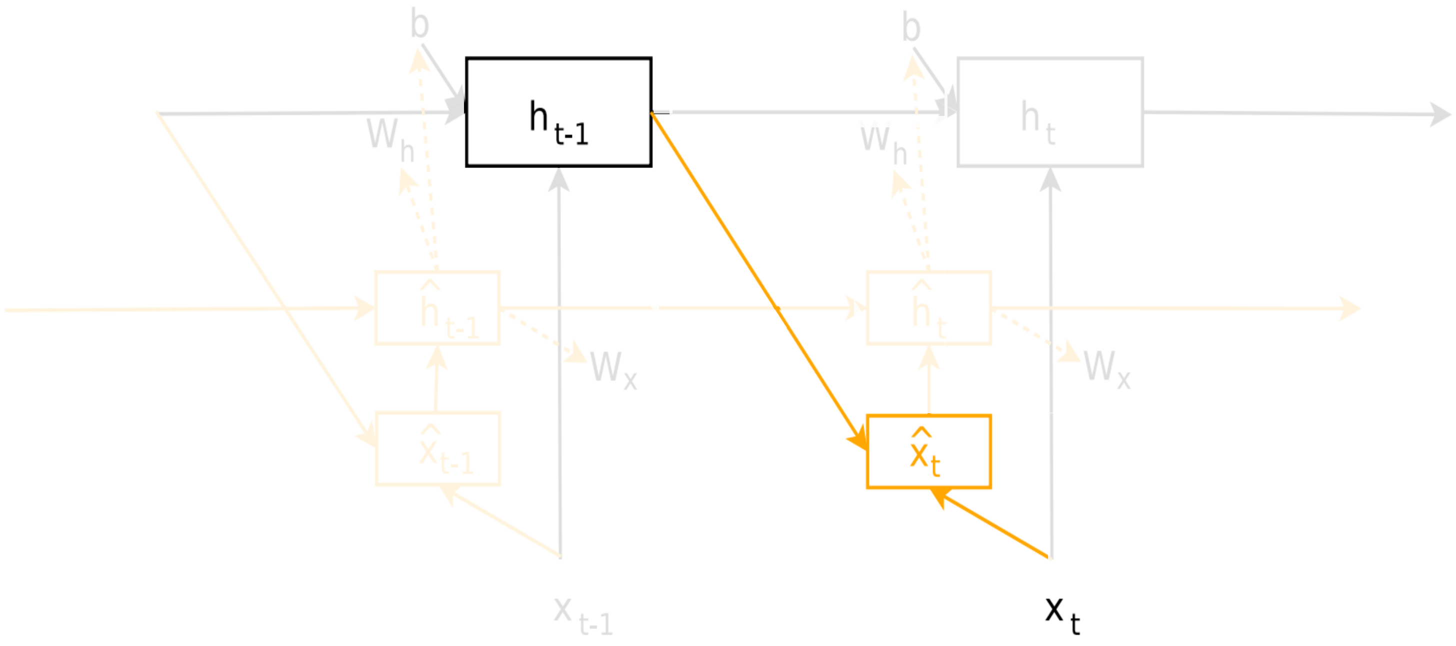
$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$



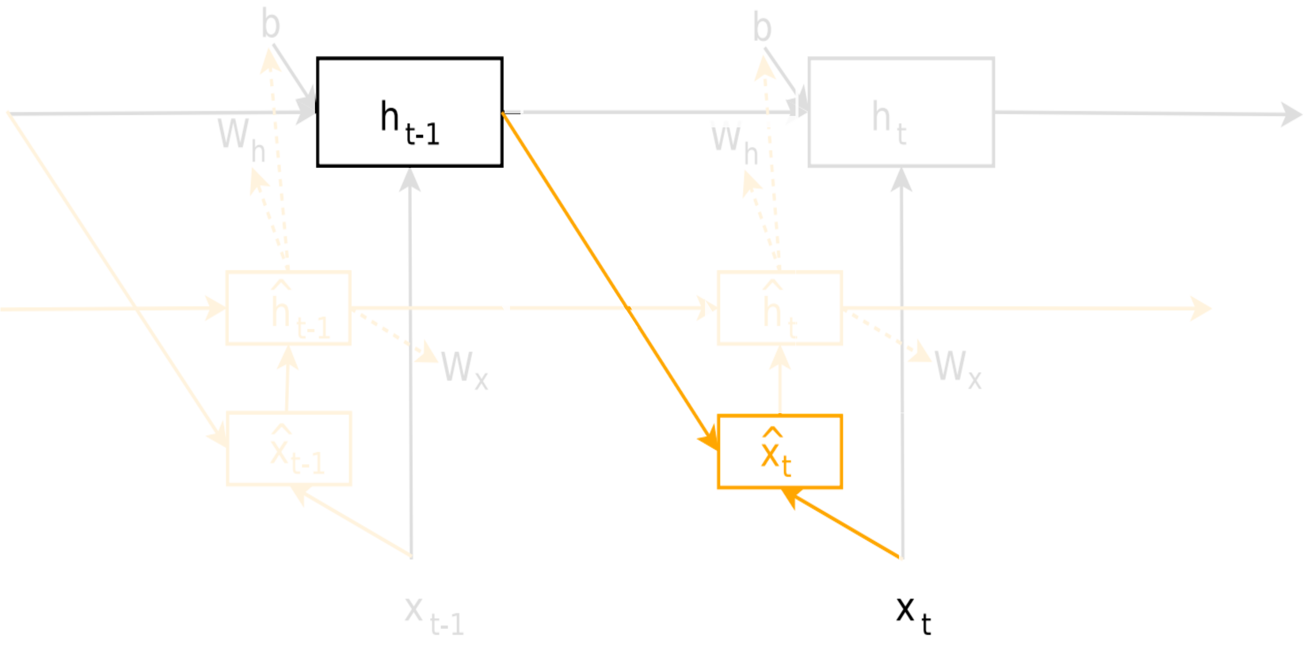
D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}}\hat{h}_{t-1} + W_{\hat{x}}\hat{x}_t + \hat{b})$$



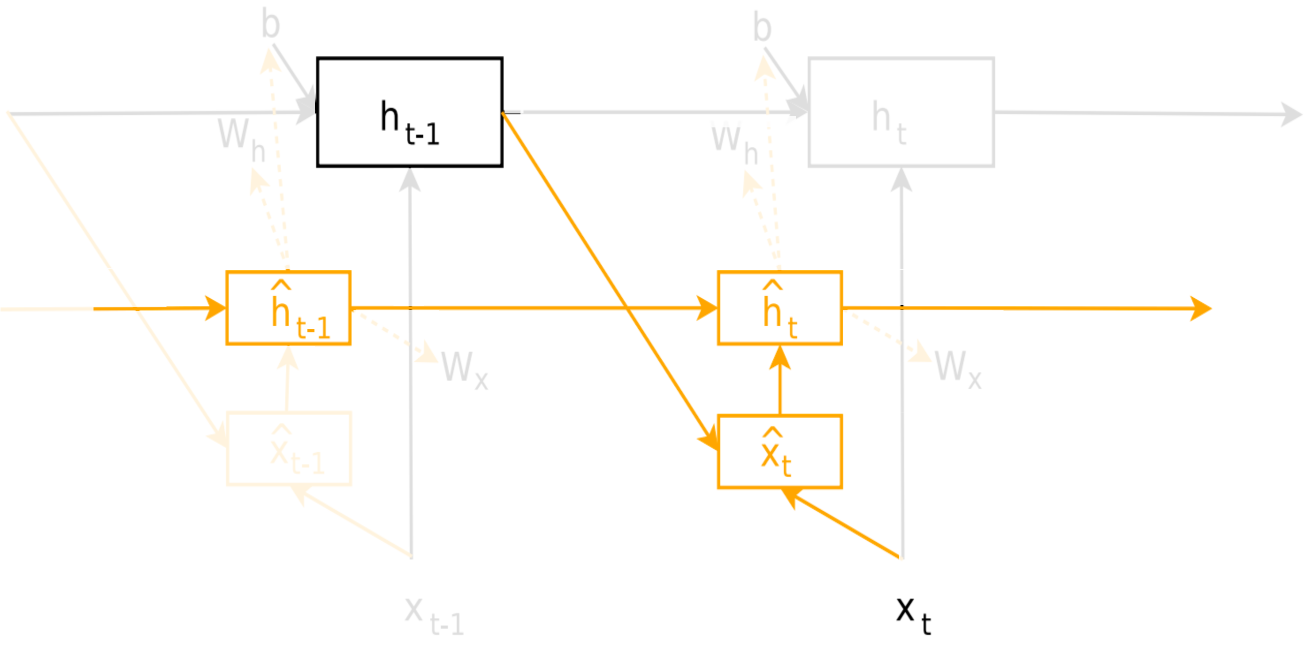
D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}} \hat{h}_{t-1} + W_{\hat{x}} \hat{x}_t + \hat{b})$$



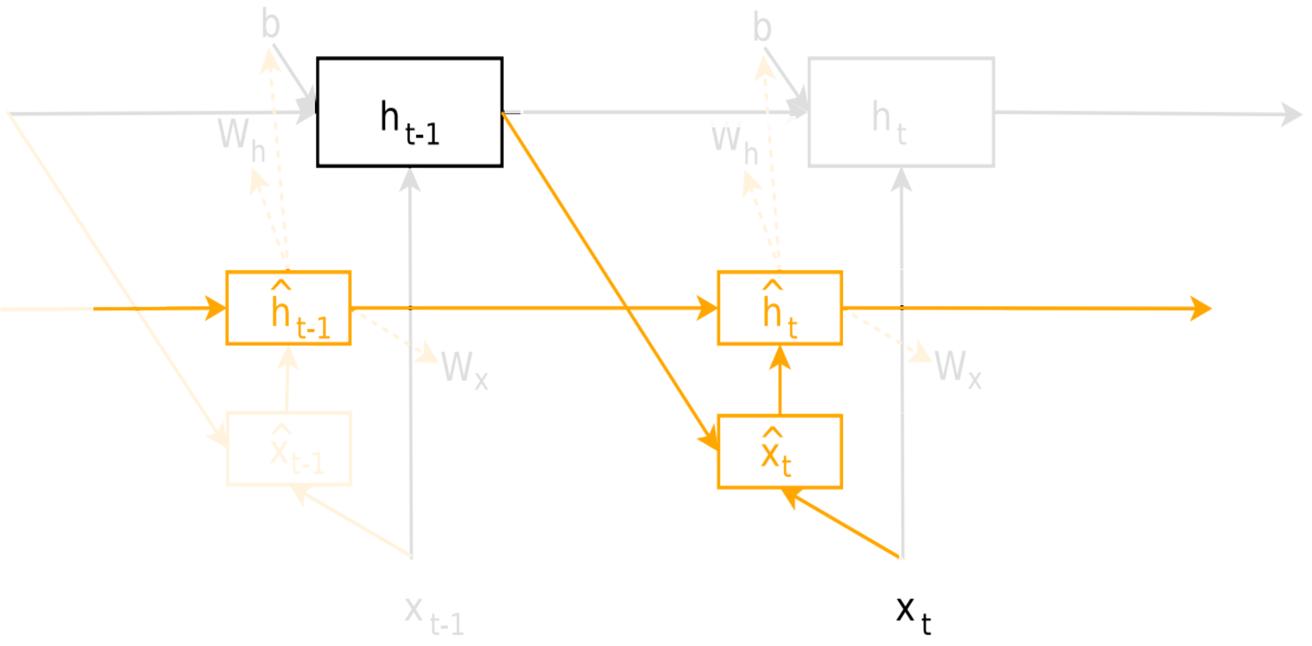
D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}} \hat{h}_{t-1} + W_{\hat{x}} \hat{x}_t + \hat{b})$$



D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}} \hat{h}_{t-1} + W_{\hat{x}} \hat{x}_t + \hat{b})$$

$$\begin{aligned} W_{\hat{h}} &\in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, \\ W_{\hat{x}} &\in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, \\ b &\in \mathbb{R}^{N_{\hat{h}}} \end{aligned}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}}\hat{h}_{t-1} + W_{\hat{x}}\hat{x}_t + \hat{b})$$

$$W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, \quad W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, \quad b \in \mathbb{R}^{N_{\hat{h}}}$$

$$z_h =$$

$$z_x =$$

$$z_b =$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}}\hat{h}_{t-1} + W_{\hat{x}}\hat{x}_t + \hat{b})$$

$$W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}},$$

$$W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)},$$

$$b \in \mathbb{R}^{N_{\hat{h}}}$$

$$z_h = W_{\hat{h}h}\hat{h}_{t-1} + b_{\hat{h}h}$$

$$z_x = W_{\hat{h}x}\hat{h}_{t-1} + b_{\hat{h}x}$$

$$z_b = W_{\hat{h}b}\hat{h}_{t-1}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}}\hat{h}_{t-1} + W_{\hat{x}}\hat{x}_t + \hat{b})$$

$$W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, \quad W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, \quad b \in \mathbb{R}^{N_{\hat{h}}}$$

$$z_h = W_{\hat{h}h}\hat{h}_{t-1} + b_{\hat{h}h}$$

$$z_x = W_{\hat{h}x}\hat{h}_{t-1} + b_{\hat{h}x}$$

$$z_b = W_{\hat{h}b}\hat{h}_{t-1}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}}\hat{h}_{t-1} + W_{\hat{x}}\hat{x}_t + \hat{b})$$

$$W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, \quad W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, \quad b \in \mathbb{R}^{N_{\hat{h}}}$$

$$z_h = W_{\hat{h}h}\hat{h}_{t-1} + b_{\hat{h}h}$$

$$z_x = W_{\hat{h}x}\hat{h}_{t-1} + b_{\hat{h}x}$$

$$z_b = W_{\hat{h}b}\hat{h}_{t-1}$$

$$W_{\hat{h}h}, W_{\hat{h}x}, W_{\hat{h}b} \in \mathbb{R}^{N_z \times N_{\hat{h}}}$$

$$b_{\hat{h}h}, b_{\hat{h}x} \in \mathbb{R}^{N_z}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

HyperNetworks

- Use a HyperRNN to compute z_h, z_x and z_b as a function of x_t and h_{t-1} :

$$\hat{x}_t = \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$\hat{h}_t = \phi(W_{\hat{h}}\hat{h}_{t-1} + W_{\hat{x}}\hat{x}_t + \hat{b})$$

$$W_{\hat{h}} \in \mathbb{R}^{N_{\hat{h}} \times N_{\hat{h}}}, \quad W_{\hat{x}} \in \mathbb{R}^{N_{\hat{h}} \times (N_h + N_z)}, \quad b \in \mathbb{R}^{N_{\hat{h}}}$$

$$z_h = W_{\hat{h}h}\hat{h}_{t-1} + b_{\hat{h}h}$$

$$z_x = W_{\hat{h}x}\hat{h}_{t-1} + b_{\hat{h}x}$$

$$z_b = W_{\hat{h}b}\hat{h}_{t-1}$$

$$W_{\hat{h}h}, W_{\hat{h}x}, W_{\hat{h}b} \in \mathbb{R}^{N_z \times N_{\hat{h}}}$$

$$b_{\hat{h}h}, b_{\hat{h}x} \in \mathbb{R}^{N_z}$$

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

Results: Language Modelling

- Evaluation of HyperLSTM on a character-level prediction task with the Penn Treebank corpus

Model[1]	**Test**	**Validation**	**Param Count**
Batch Norm LSTM (Cooijmans et al., 2016)	1.32		
Recurrent Dropout LSTM (Semeniuta et al., 2016)	1.301	1.338	
LSTM, 1000 units [2]	1.312	1.347	4.25 M
LSTM, 1250 units[2]	1.306	1.340	6.57 M
2-Layer LSTM, 1000 units[2]	1.281	1.312	12.26 M
HyperLSTM (ours), 1000 units	1.265	1.296	4.91 M
2-Layer Norm HyperLSTM, 1000 units	1.219	1.245	14.41 M

Bits-per-character on the Penn Treebank test set.

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

Results: Language Modelling

- Evaluation of HyperLSTM on a character-level prediction task with the Penn Treebank corpus

Model[1]	Test	Validation	Param Count
Batch Norm LSTM (Cooijmans et al., 2016)	1.32		
Recurrent Dropout LSTM (Semeniuta et al., 2016)	1.301	1.338	
LSTM, 1000 units [2]	1.312	1.347	4.25 M
LSTM, 1250 units[2]	1.306	1.340	6.57 M
2-Layer LSTM, 1000 units[2]	1.281	1.312	12.26 M
HyperLSTM (ours), 1000 units	1.265	1.296	4.91 M
2-Layer Norm HyperLSTM, 1000 units	1.219	1.245	14.41 M

Bits-per-character on the Penn Treebank test set.

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

Results: Language Modelling

- Evaluation of HyperLSTM on a character-level prediction task with the Penn Treebank corpus

Model[1]	Test	Validation	Param Count
Batch Norm LSTM (Cooijmans et al., 2016)	1.32		
Recurrent Dropout LSTM (Semeniuta et al., 2016)	1.301	1.338	
LSTM, 1000 units [2]	1.312	1.347	4.25 M
LSTM, 1250 units[2]	1.306	1.340	6.57 M
2-Layer LSTM, 1000 units[2]	1.281	1.312	12.26 M
HyperLSTM (ours), 1000 units	1.265	1.296	4.91 M
2-Layer Norm HyperLSTM, 1000 units	1.219	1.245	14.41 M

Bits-per-character on the Penn Treebank test set.

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

Results: Neural Machine Translation

English Input

I was expecting to see gnashing of teeth and a fight breaking out at the gate .

French (Ground Truth)

Je m' attendais à voir des grincements de dents et une bagarre éclater à la porte .

LSTM Translation

Je m' attendais à voir des larmes de dents et un combat à la porte .

HyperLSTM Translation

Je m' attendais à voir des dents grincer des dents et une bataille éclater à la porte .

English Input

According to her , the CSRS was invited to a mediation and she asked for an additional period for consideration .

French (Ground Truth)

Selon elle , la CSRS a été invitée à une médiation et elle a demandé un délai supplémentaire pour y réfléchir .

LSTM Translation

Selon elle , le SCRS a été invité à une médiation et elle a demandé un délai supplémentaire .

HyperLSTM Translation

Selon elle , le SCRS a été invité à une médiation et elle a demandé une période de réflexion supplémentaire .

English Input

I was on the mid-evening news that same evening , and on TV the following day as well .

French (Ground Truth)

Le soir-même , je suis au 20h , le lendemain aussi je suis à la télé .

LSTM Translation

J' étais au milieu de l' actualité le soir même , et à la télévision le lendemain également .

HyperLSTM Translation

J' étais au milieu de la soirée ce soir-là et à la télévision le lendemain .

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

Results: Neural Machine Translation

- Evaluation of HyperLSTM on WMT' 14 En → Fr using the same experimental setup and train/test splits outlined in (Wu et al., 2016)

Model	Test BLEU	Log Perplexity	Param Count
GNMT WPM-32K, LSTM (Wu et al., 2016)	38.95	1.027	280.7 M
GNMT WPM-32K, ensemble of 8 LSTMs (Wu et al., 2016)	40.35		2,246.1 M
GNMT WPM-32K, HyperLSTM (ours)	40.03	0.993	325.5 M

Single model results on WMT En→Fr (newstest2014)

Results: Neural Machine Translation

- Evaluation of HyperLSTM on WMT' 14 En → Fr using the same experimental setup and train/test splits outlined in (Wu et al., 2016)

Model	Test BLEU	Log Perplexity	Param Count
GNMT WPM-32K, LSTM (Wu et al., 2016)	38.95	1.027	280.7 M
GNMT WPM-32K, ensemble of 8 LSTMs (Wu et al., 2016)	40.35		2,246.1 M
GNMT WPM-32K, HyperLSTM (ours)	40.03	0.993	325.5 M

Single model results on WMT En→Fr (newstest2014)

D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

Results: Neural Machine Translation

- Evaluation of HyperLSTM on WMT' 14 En → Fr using the same experimental setup and train/test splits outlined in (Wu et al., 2016)

Model	Test BLEU	Log Perplexity	Param Count
GNMT WPM-32K, LSTM (Wu et al., 2016)	38.95	1.027	280.7 M
GNMT WPM-32K, ensemble of 8 LSTMs (Wu et al., 2016)	40.35		2,246.1 M
GNMT WPM-32K, HyperLSTM (ours)	40.03	0.993	325.5 M

Single model results on WMT En→Fr (newstest2014)

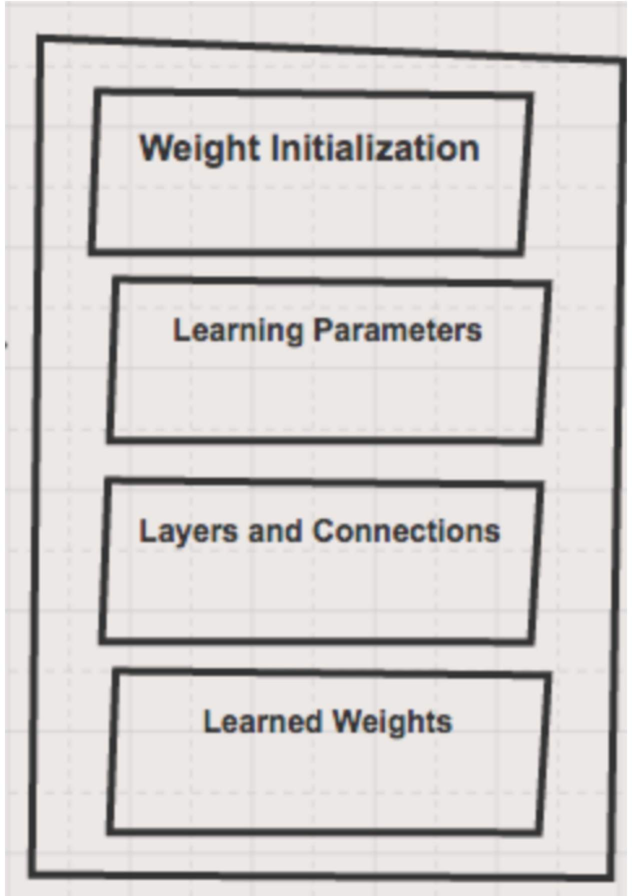
D. Ha, A. Dai, and Q. V. Le. HyperNetworks. arXiv preprint arXiv:1609.09106 (2016).

Learning to Learn the Deep Learning Network Architecture

$\mathbf{L}_{\mu}$ Deep NN

μ Weights initialization, Learning parameters, Layers and connections, Learned weights

ML
- Hyper-Parameter Optimization
- Reinforcement Learning for Architecture Design
- Hypernetworks
- **Evolution**



Blog by Carlos E. Perez

Genetic Algorithms

"As many more individuals of each species are born than can possibly survive; and as, consequently, there is a frequently recurring struggle for existence, it follows that any being, if it vary however slightly in any manner profitable to itself, under the complex and sometimes varying conditions of life, will have a better chance of surviving, and thus be naturally selected."

— Darwin, *On the Origin of Species by Means of Natural Selection* (1859)

Inspiration

- "Survival of the fittest"
- Three essential ingredients of an evolutionary process:
    - Selection
    - Variation
    - Heritability

Genetic Algorithms and Neuroevolution

- Connection to deep learning
  - **Architecture search** and **hyperparameter optimization** currently a **labor-intensive** process
  - **Evolutionary algorithms** offer natural framework for exploring neural network topologies in unsupervised manner based on principles of **natural selection**
  - Evolutionary algorithms represent the models using an **encoding** that is convenient for their purpose — analogous to **nature's DNA**

Genetic Algorithms and Neuroevolution

- **How to represent the "DNA" of neural networks?**
  - Choose meaningful encoding
- **How to "mutate" the neural network?**
  - Dependent on the chosen encoding
- **How to evaluate fitness for each learned architecture?**
  - Train for fixed number of epochs on task of interest, evaluate performance
- **What architectures can be learned with evolution?**

Genetic Algorithms and Neuroevolution

- **How to represent the "DNA" of neural networks?**
  - Choose meaningful encoding
- **How to "mutate" the neural network?**
  - Dependent on the chosen encoding
- **How to evaluate fitness for each learned architecture?**
  - Train for fixed number of epochs on task of interest, evaluate performance
- **What architectures can be learned with evolution?**

Genetic Algorithms and Neuroevolution

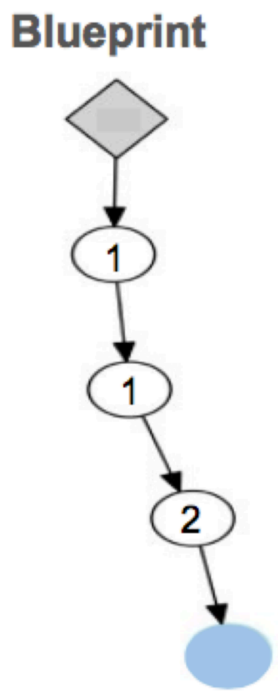
- **How to represent the "DNA" of neural networks?**
  - Choose meaningful encoding
    - Model architecture using graph structure?
    - Encode connections between layers using binary strings?
    - Form connections between fixed type and number of layers using "active" module map?

Genetic Algorithms and Neuroevolution

- **How to represent the "DNA" of neural networks?**
  - Choose meaningful encoding
    - **Model architecture using graph structure?**
    - Encode connections between layers using binary strings?
    - Form connections between fixed type and number of layers using "active" module map?

How to represent the “DNA” of neural networks?

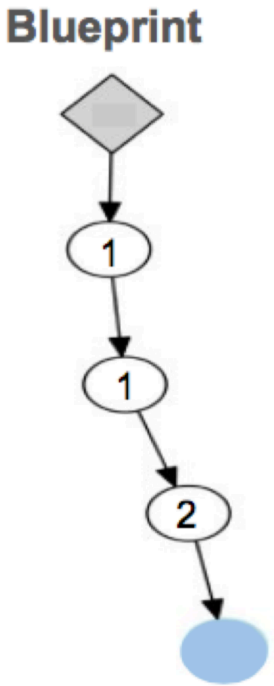
Model architecture using graph structure

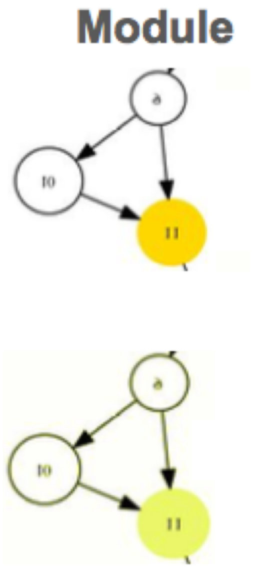


R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the “DNA” of neural networks?

Model architecture using graph structure





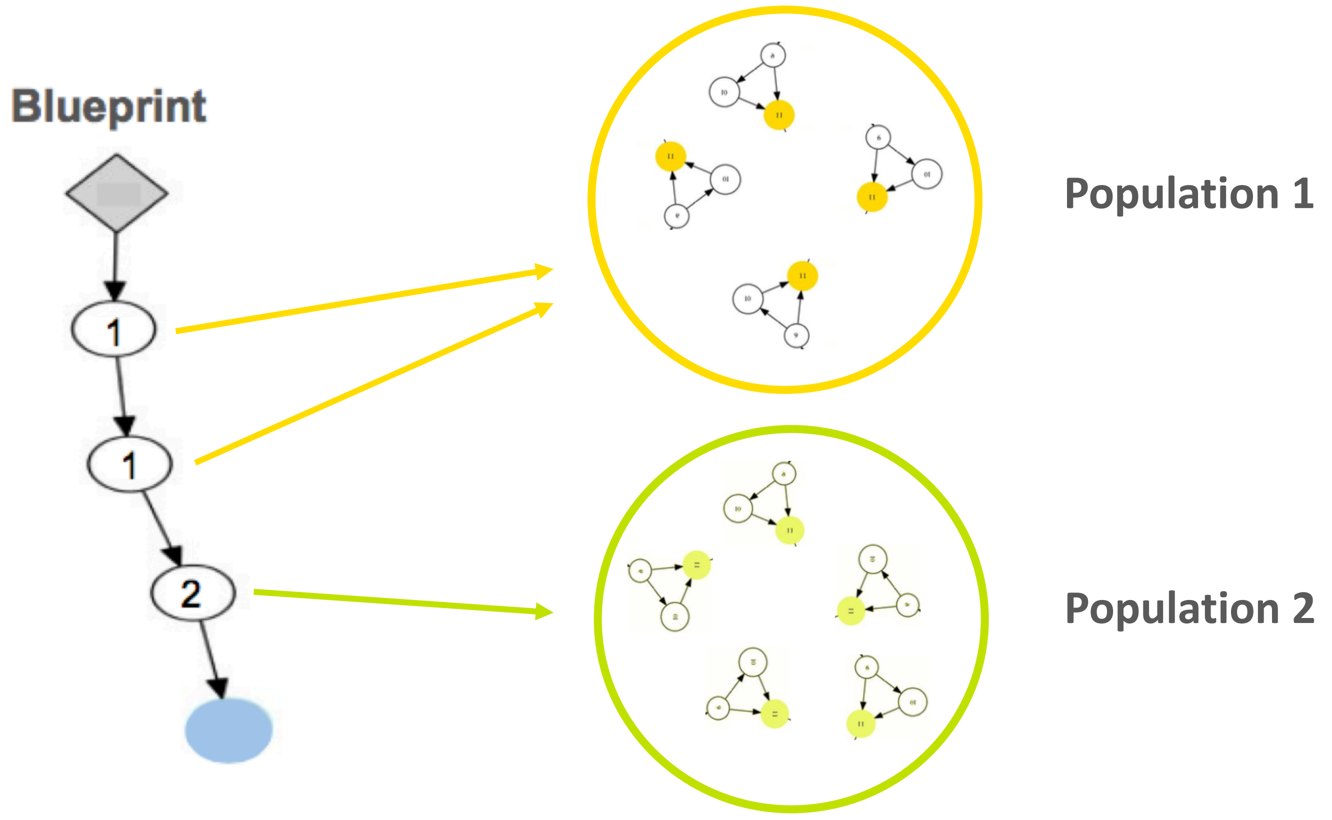
`Convolution-BatchNorm-ReLU`

`Convolution-ReLU-MaxPool`

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the “DNA” of neural networks?

Model architecture using graph structure



R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the "DNA" of neural networks?

Model architecture using graph structure

Node Hyperparameter	Range
Number of Filters	[32, 256]
Dropout Rate	[0, 0.7]
Initial Weight Scaling	[0, 2.0]
Kernel Size	{1, 3}
Max Pooling	{True, False}

CNN

Node Hyperparameter	Range
Layer Type	{Dense, LSTM}
Merge Method	{Sum, Concat}
Layer Size	{128, 256}
Layer Activation	{ReLU, Linear}
Layer Dropout	[0, 0.7]

RNN

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the "DNA" of neural networks?

Model architecture using graph structure

Node Hyperparameter	Range
Number of Filters	[32, 256]
Dropout Rate	[0, 0.7]
Initial Weight Scaling	[0, 2.0]
Kernel Size	{1, 3}
Max Pooling	{True, False}

CNN

Node Hyperparameter	Range
Layer Type	{Dense, LSTM}
Merge Method	{Sum, Concat}
Layer Size	{128, 256}
Layer Activation	{ReLU, Linear}
Layer Dropout	[0, 0.7]

RNN

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the "DNA" of neural networks?

Model architecture using graph structure

Node Hyperparameter	Range
Number of Filters	[32, 256]
Dropout Rate	[0, 0.7]
Initial Weight Scaling	[0, 2.0]
Kernel Size	{1, 3}
Max Pooling	{True, False}

CNN

Node Hyperparameter	Range
Layer Type	{Dense, LSTM}
Merge Method	{Sum, Concat}
Layer Size	{128, 256}
Layer Activation	{ReLU, Linear}
Layer Dropout	[0, 0.7]

RNN

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the “DNA” of neural networks?

Model architecture using graph structure

Global Hyperparameter	Range
Learning Rate	$[0.0001, 0.1]$
Momentum	$[0.68, 0.99]$
Hue Shift	$[0, 45]$
Saturation/Value Shift	$[0, 0.5]$
Saturation/Value Scale	$[0, 0.5]$
Cropped Image Size	$[26, 32]$
Spatial Scaling	$[0, 0.3]$
Random Horizontal Flips	{True, False}
Variance Normalization	{True, False}
Nesterov Accelerated Gradient	{True, False}

CNN

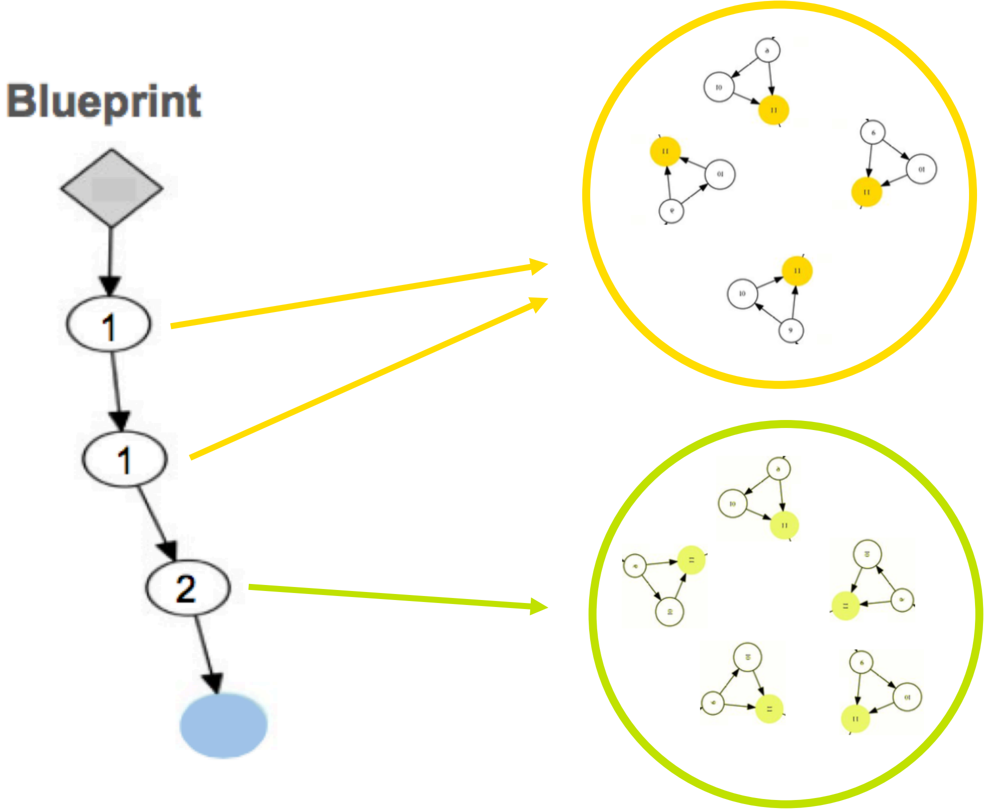
Global Hyperparameter	Range
Learning Rate	$[0.0001, 0.1]$
Momentum	$[0.68, 0.99]$
Shared Embedding Size	$[128, 512]$
Embedding Dropout	$[0, 0.7]$
LSTM Recurrent Dropout	{True, False}
Nesterov Momentum	{True, False}
Weight Initialization	{Glorot normal, He normal}

RNN

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the “DNA” of neural networks?

Model architecture using graph structure



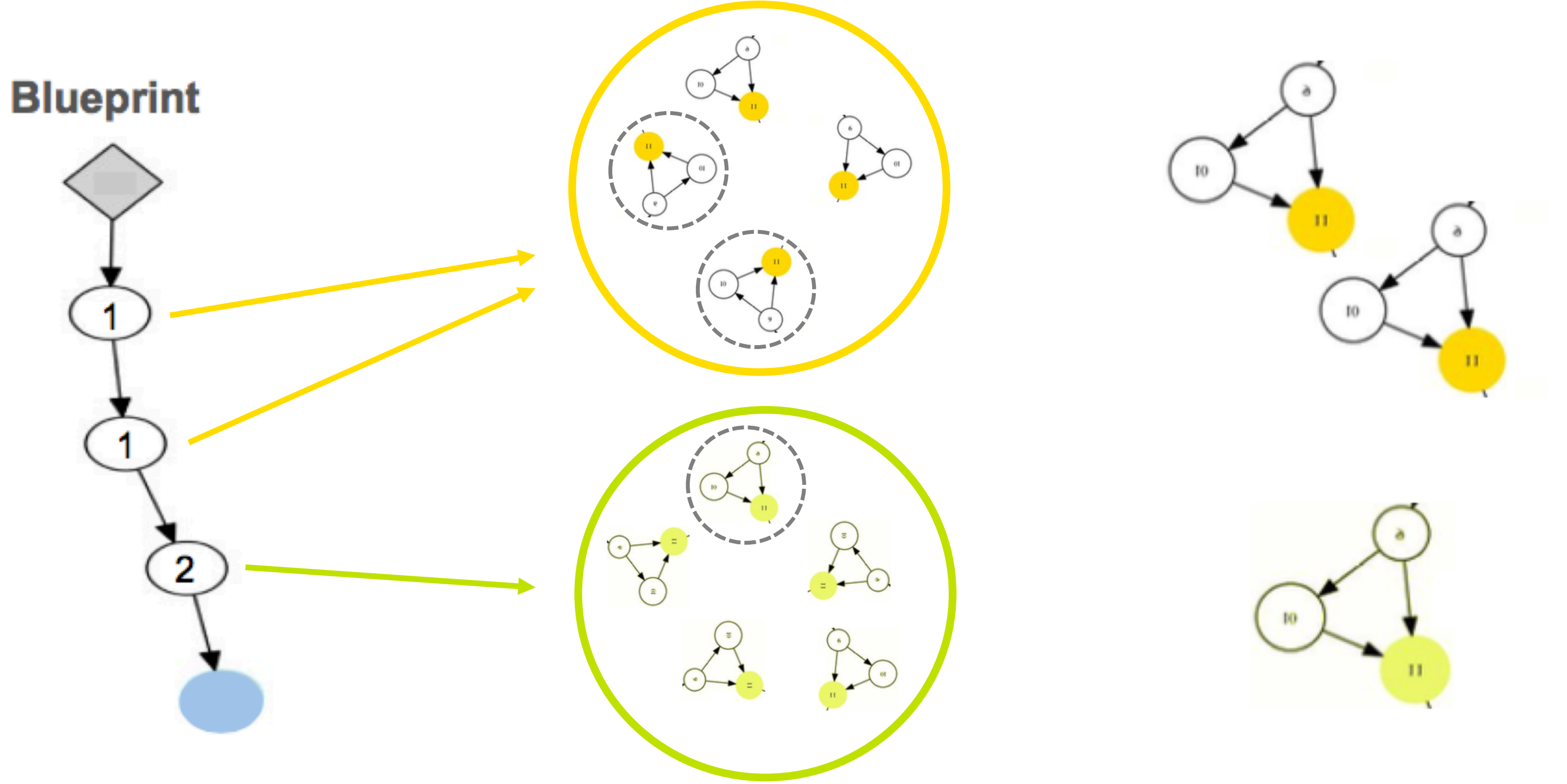
Population 1

Population 2

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the “DNA” of neural networks?

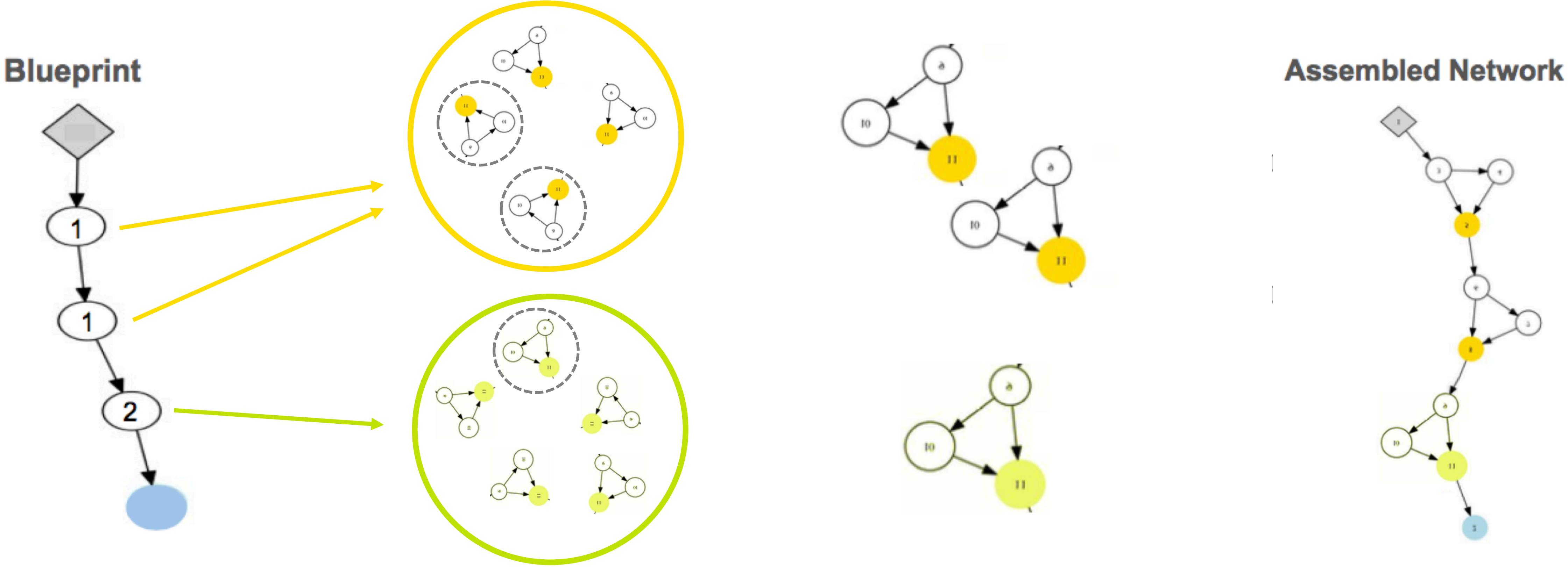
Model architecture using graph structure



R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to represent the “DNA” of neural networks?

Model architecture using graph structure



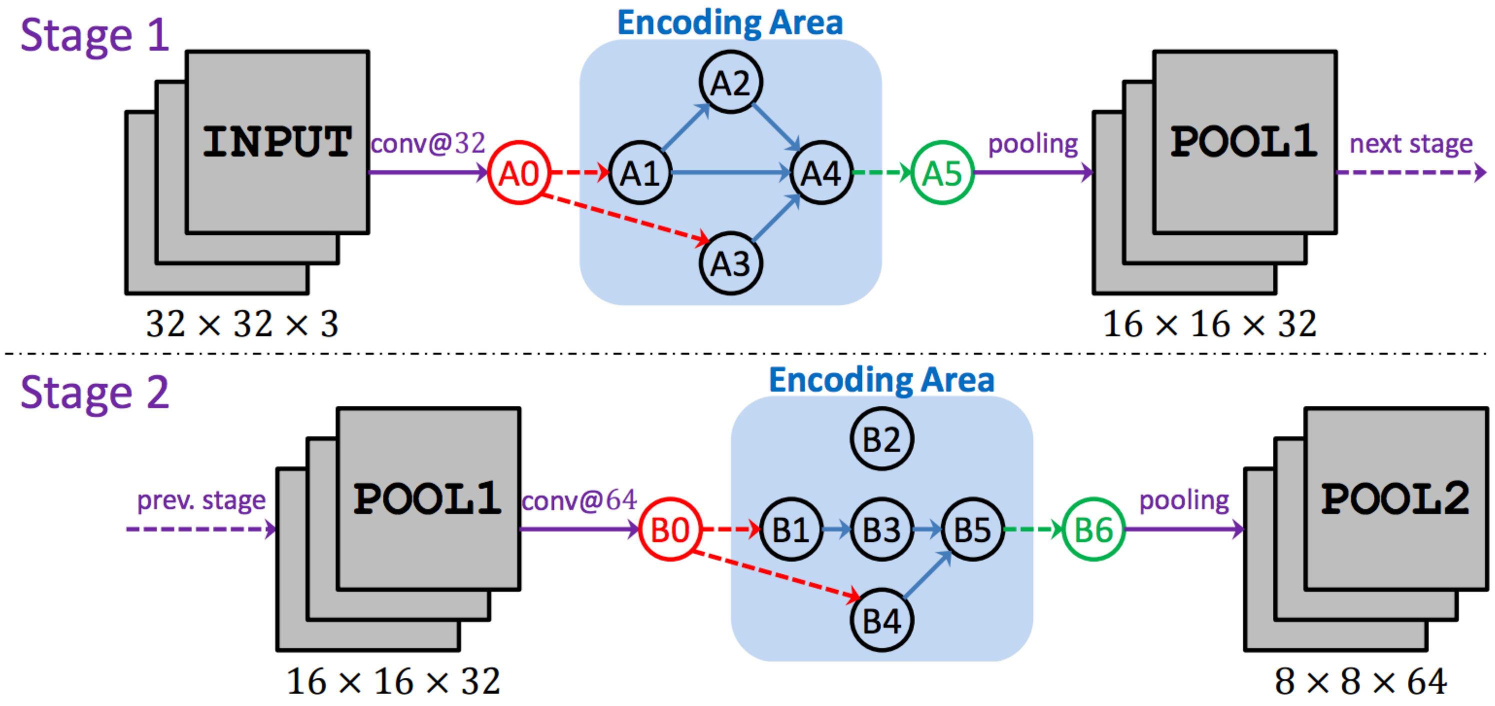
R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

Genetic Algorithms and Neuroevolution

- **How to represent the “DNA” of neural networks?**
  - Choose meaningful encoding
    - Model architecture using graph structure?
    - **Encode connections between layers using binary strings?**
    - Form connections between fixed type and number of layers using “active” module map?

How to represent the “DNA” of neural networks?

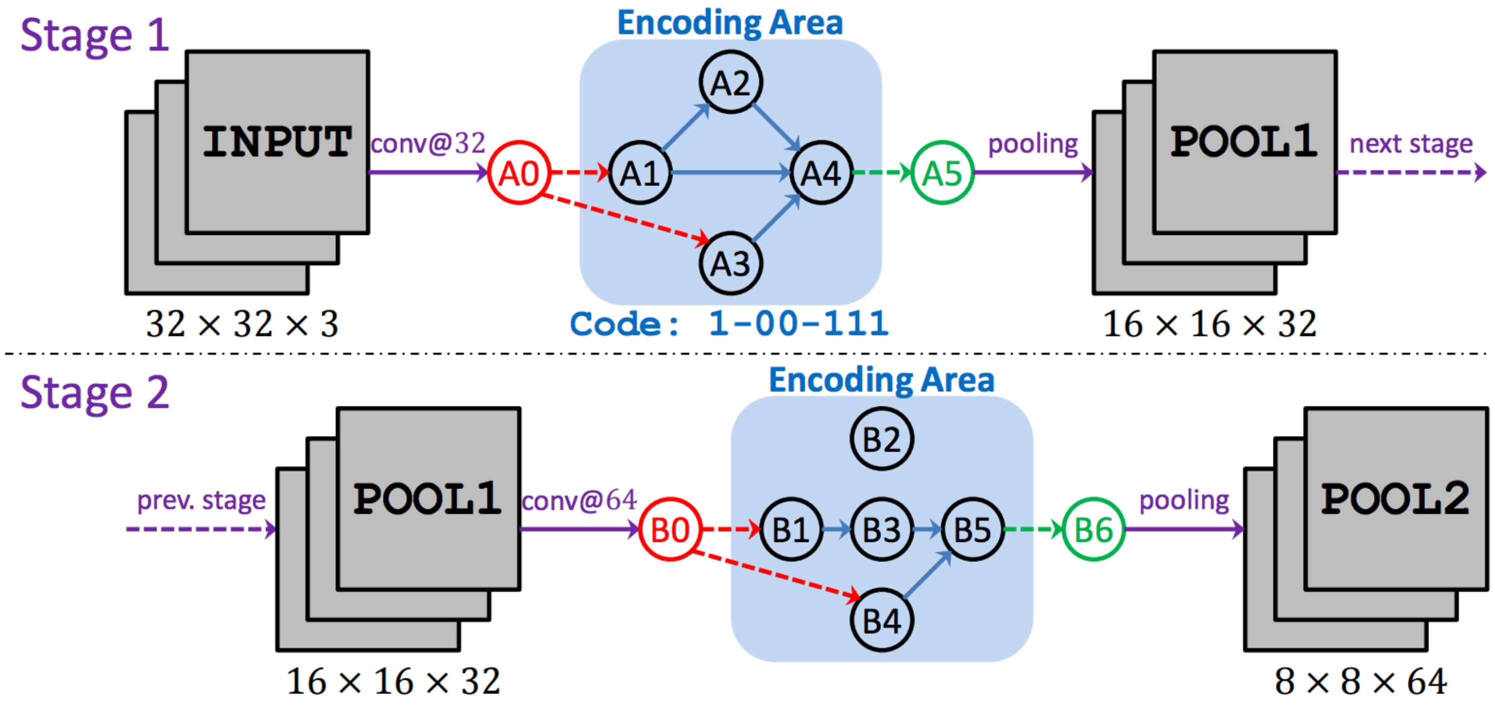
Encode connections between layers using binary strings



E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

How to represent the “DNA” of neural networks?

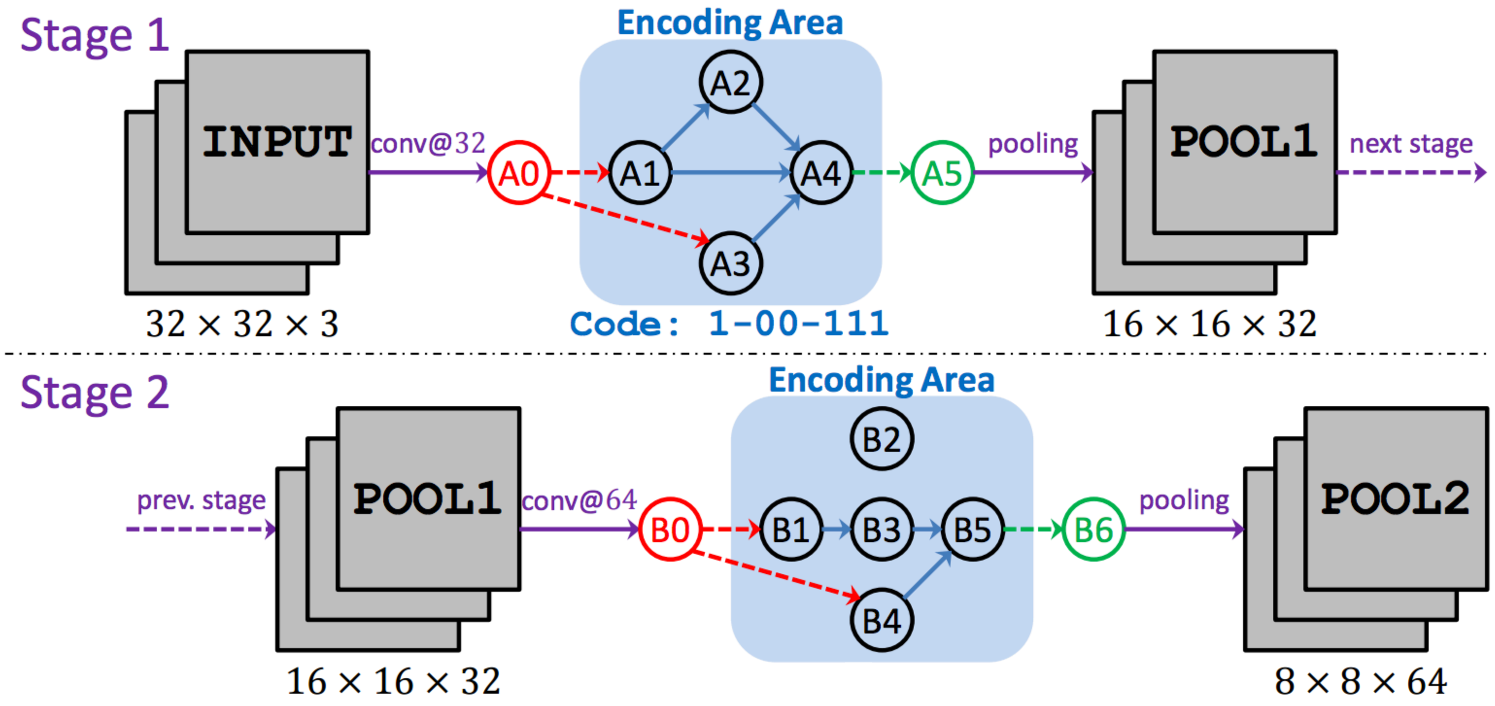
Encode connections between layers using binary strings



E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

How to represent the “DNA” of neural networks?

Encode connections between layers using binary strings



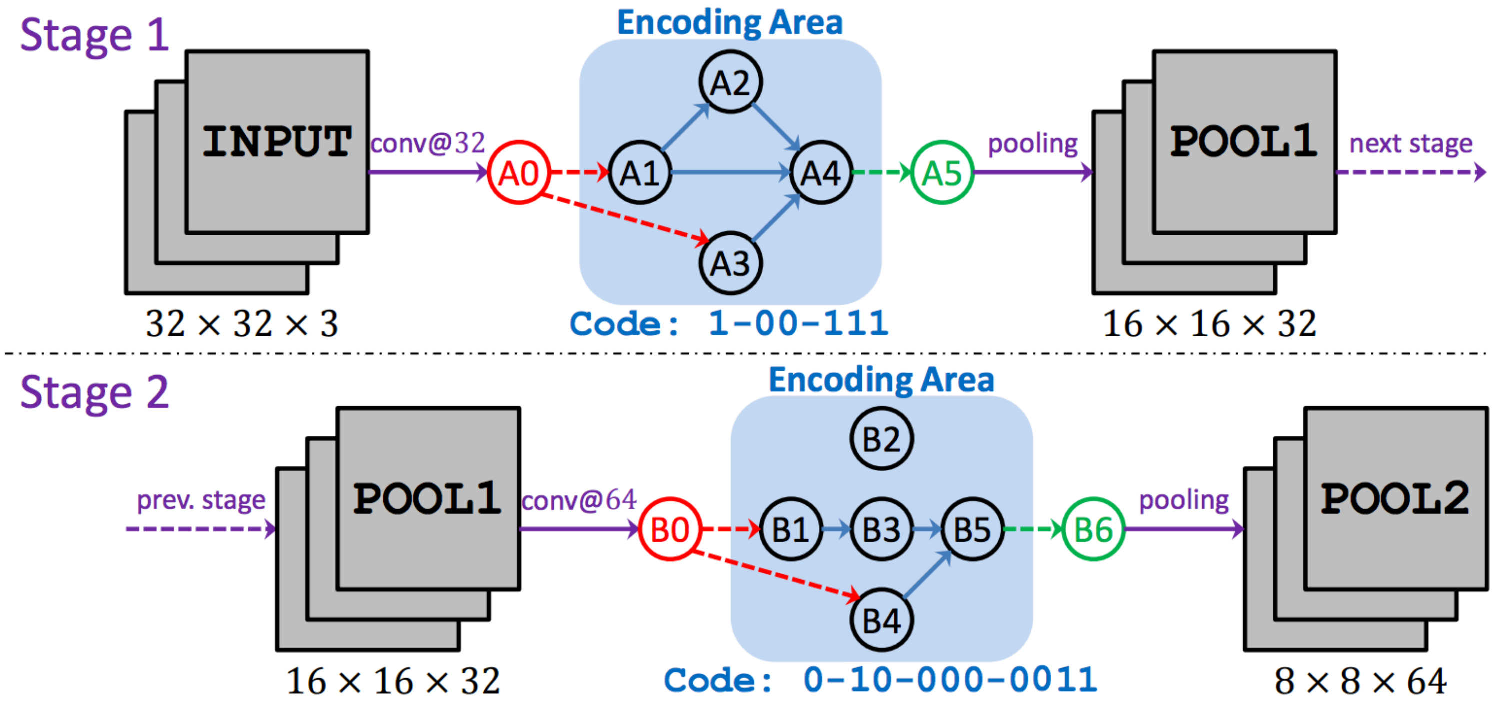
What would be the encoding of this stage?

?

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

How to represent the “DNA” of neural networks?

Encode connections between layers using binary strings



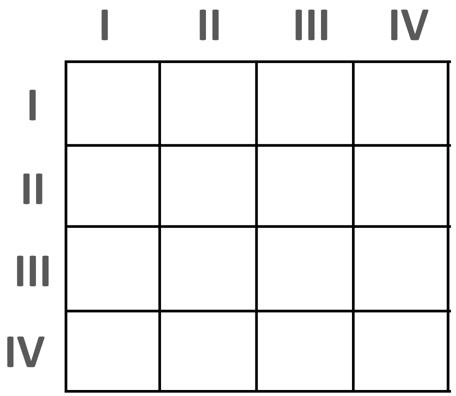
E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

Genetic Algorithms and Neuroevolution

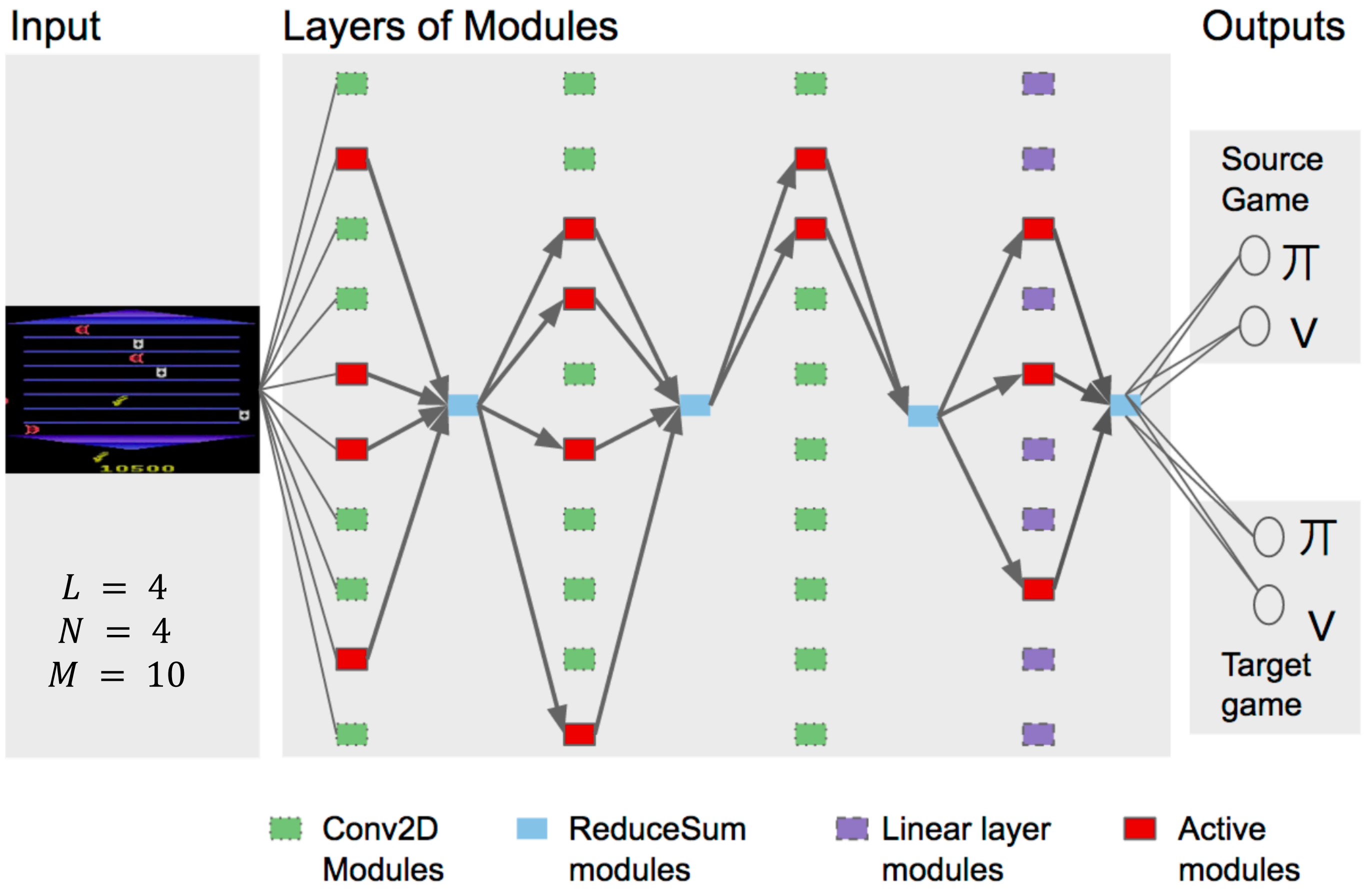
- **How to represent the "DNA" of neural networks?**
  - Choose meaningful encoding
    - Model architecture using graph structure?
    - Encode connections between layers using binary strings?
    - **Form connections between fixed type and number of layers using "active" module map?**

How to represent the “DNA” of neural networks?

Form connections between fixed type and number of layers using “active” module map



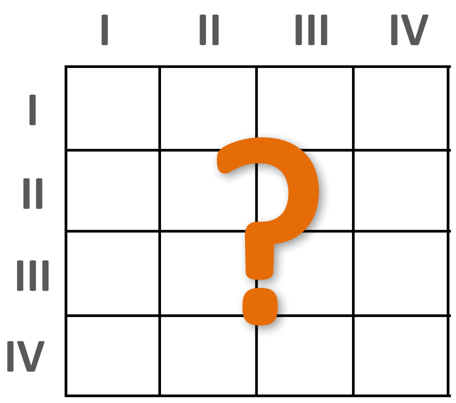
$N \times L$



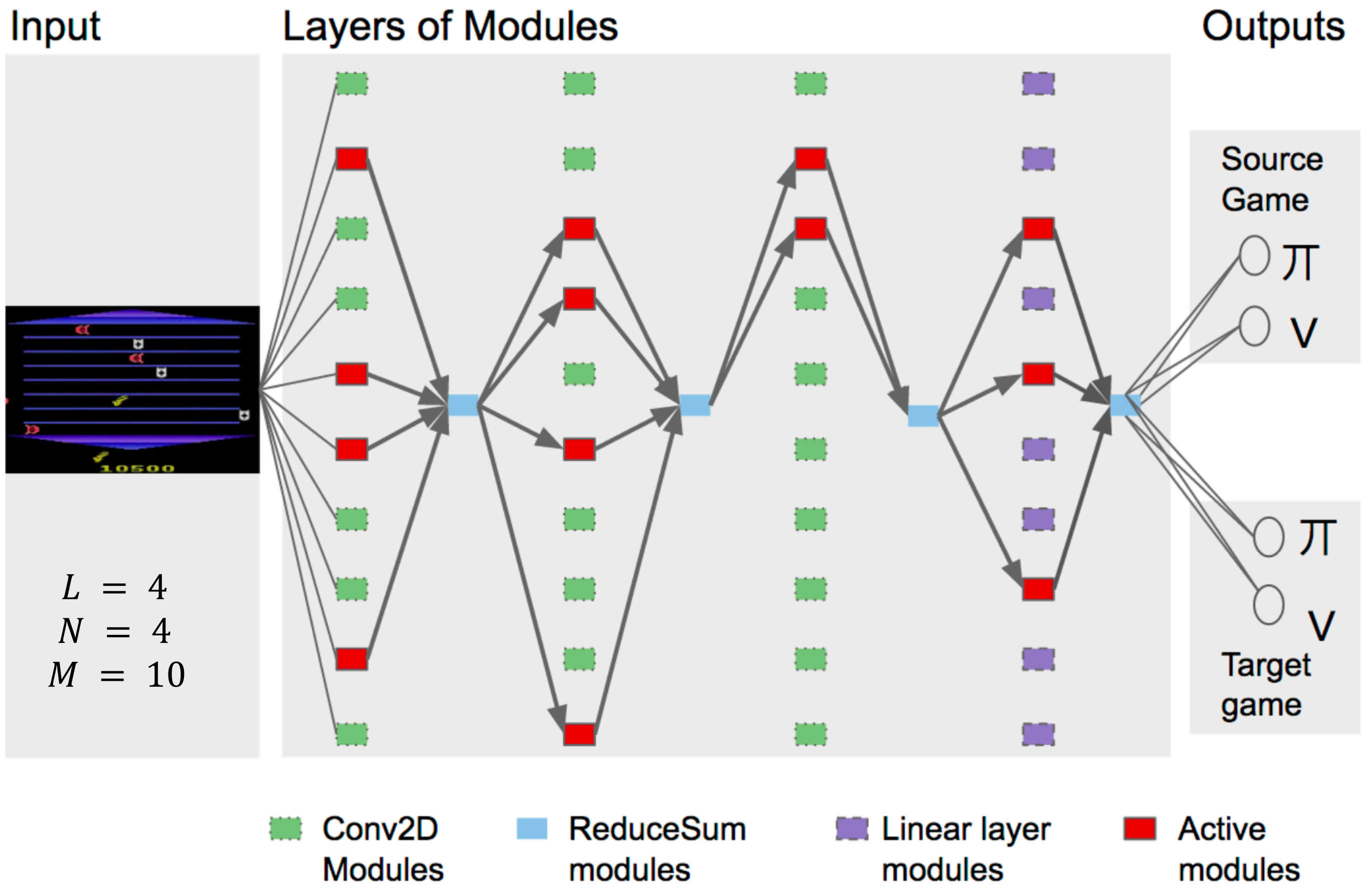
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

How to represent the “DNA” of neural networks?

Form connections between fixed type and number of layers using “active” module map



$N \times L$



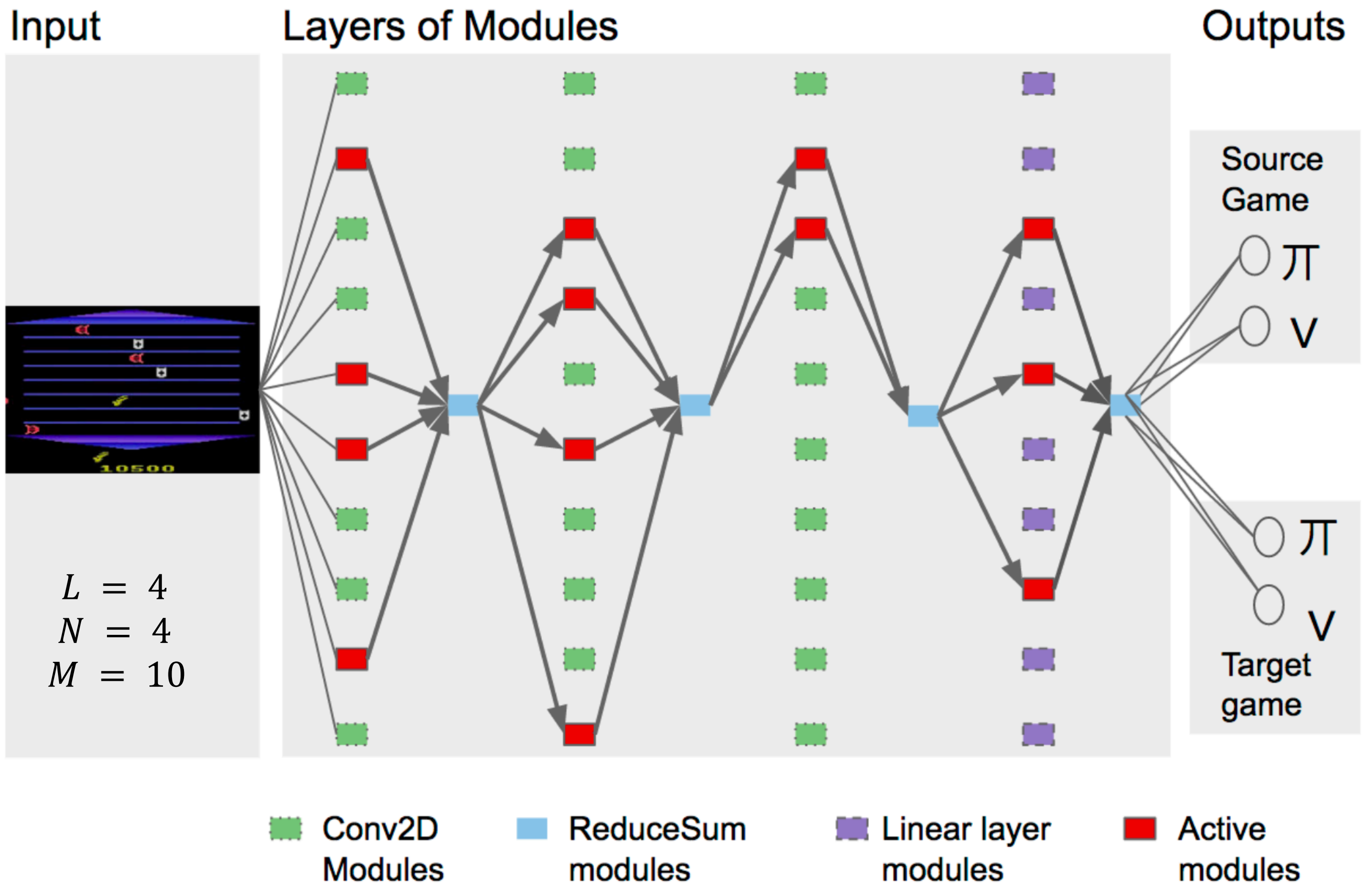
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

How to represent the “DNA” of neural networks?

Form connections between fixed type and number of layers using “active” module map

	I	II	III	IV
I	2	3	2	3
II	5	4	3	5
III	6	6	0	8
IV	9	10	0	0

$N \times L$



C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Genetic Algorithms and Neuroevolution

- **How to represent the “DNA” of neural networks?**
    - Choose meaningful encoding
- **How to “mutate” the neural network?**
    - Dependent on the chosen encoding
- **How to evaluate fitness for each learned architecture?**
    - Train for fixed number of epochs on task of interest, evaluate performance
- **What architectures can be learned with evolution?**

Genetic Algorithms and Neuroevolution

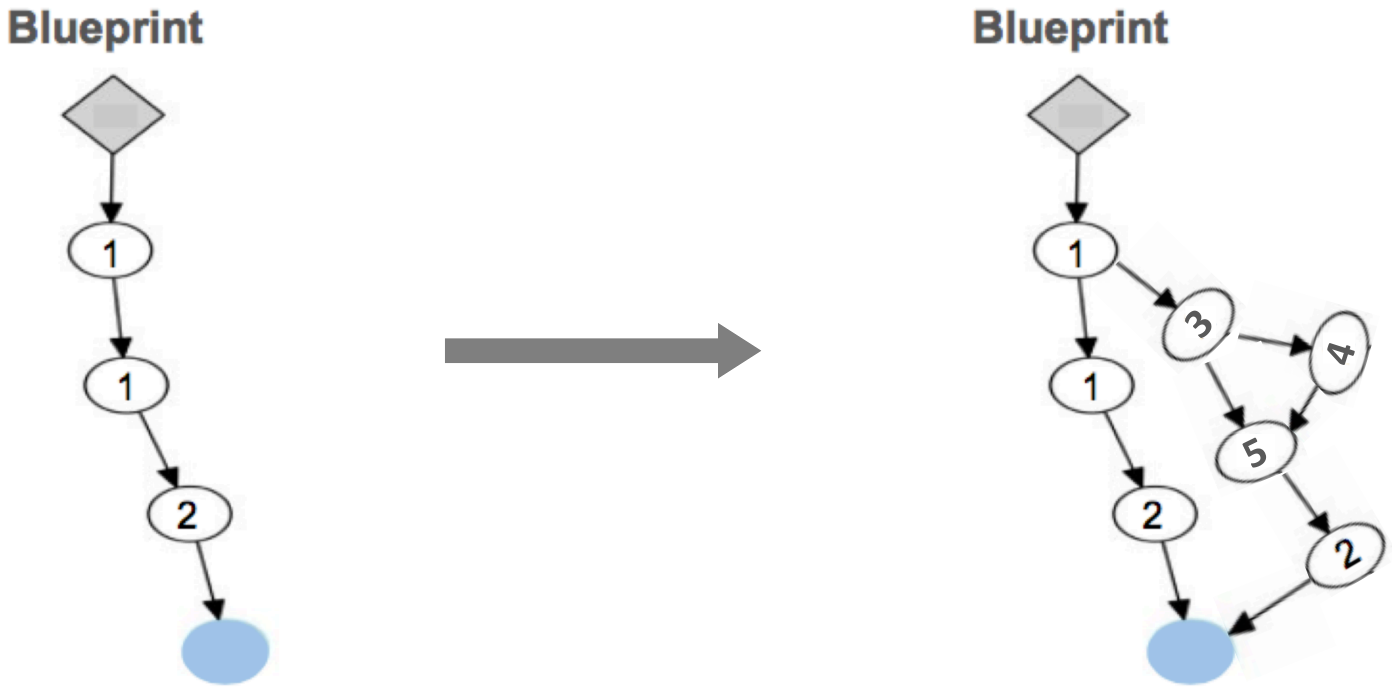
- **How to “mutate” the neural network?**
  - Dependent on the chosen encoding
    - Add structure – nodes and edges – to the graph?
    - Randomly flip bits with some mutation probability?
    - Independently select with uniform probability the “active” modules in each layer?

Genetic Algorithms and Neuroevolution

- **How to “mutate” the neural network?**
  - Dependent on the chosen encoding
    - **Add structure – nodes and edges – to the graph?**
    - Randomly flip bits with some mutation probability?
    - Independently select with uniform probability the “active” modules in each layer?

How to “mutate” the neural network?

Add structure – nodes and edges – to the graph



R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving Deep Neural Networks. arXiv preprint arXiv:1703.00548 (2017).

How to "mutate" the neural network?

Add structure – nodes and edges – to the graph

elaborate mutation operators

- ALTER-LEARNING-RATE
- IDENTITY (effectively means "keep training").
- RESET-WEIGHTS
- INSERT-CONVOLUTION (inserts a convolution at a random location in the "convolutional backbone"
  The inserted convolution has 3×3 filters, strides of 1 or 2 at random, number of channels same as input. May apply batch-normalization and ReLU activation or none at random).
- REMOVE-CONVOLUTION.
- ALTER-STRIDE (only powers of 2 are allowed).
- ALTER-NUMBER-OF-CHANNELS (of random conv.).
- FILTER-SIZE (horizontal or vertical at random, on random convolution, odd values only).
- INSERT-ONE-TO-ONE (inserts a one-to-one/identity connection, analogous to insert-convolution mutation).
- ADD-SKIP (identity between random layers).
- REMOVE-SKIP (removes random skip).

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

How to “mutate” the neural network?

Add structure – nodes and edges – to the graph

elaborate mutation operators

- ALTER-LEARNING-RATE
- IDENTITY (effectively means “keep training”).
- RESET-WEIGHTS
- INSERT-CONVOLUTION (inserts a convolution at a random location in the “convolutional backbone” The inserted convolution has 3×3 filters, strides of 1 or 2 at random, number of channels same as input. May apply batch-normalization and ReLU activation or none at random).
- REMOVE-CONVOLUTION.
- ALTER-STRIDE (only powers of 2 are allowed).
- ALTER-NUMBER-OF-CHANNELS (of random conv.).
- FILTER-SIZE (horizontal or vertical at random, on random convolution, odd values only).
- INSERT-ONE-TO-ONE (inserts a one-to-one/identity connection, analogous to insert-convolution mutation).
- ADD-SKIP (identity between random layers).
- REMOVE-SKIP (removes random skip).

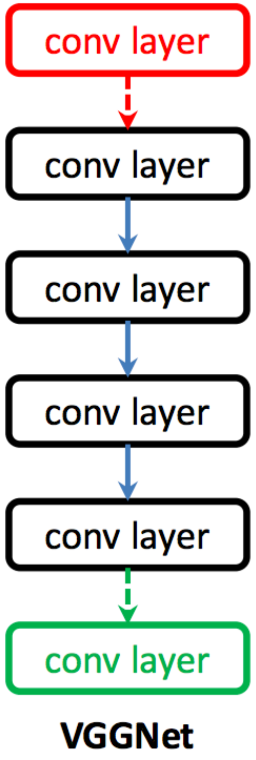
E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

Genetic Algorithms and Neuroevolution

- **How to "mutate" the neural network?**
  - Dependent on the chosen encoding
    - Add structure – nodes and edges – to the graph?
    - **Randomly flip bits with some mutation probability?**
    - Independently select with uniform probability the "active" modules in each layer?

How to “mutate” the neural network?

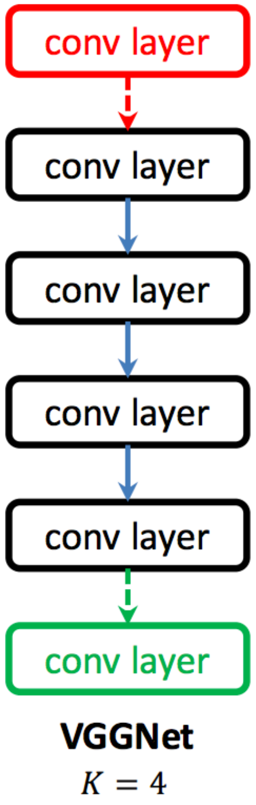
Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to “mutate” the neural network?

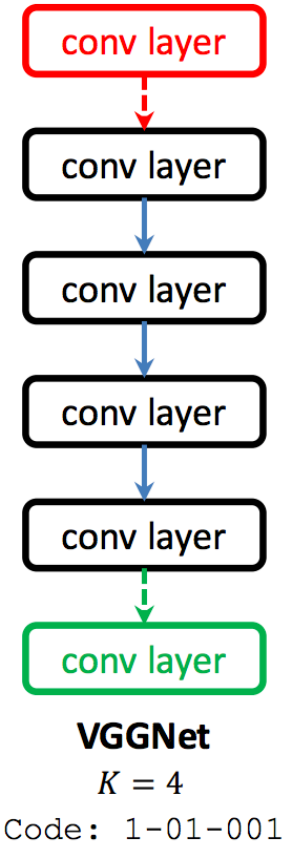
Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to “mutate” the neural network?

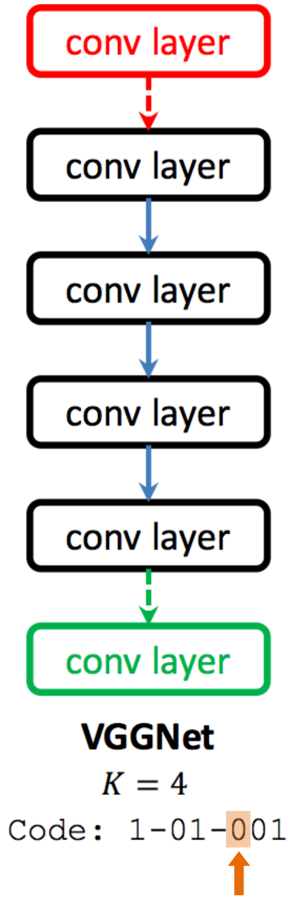
Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to “mutate” the neural network?

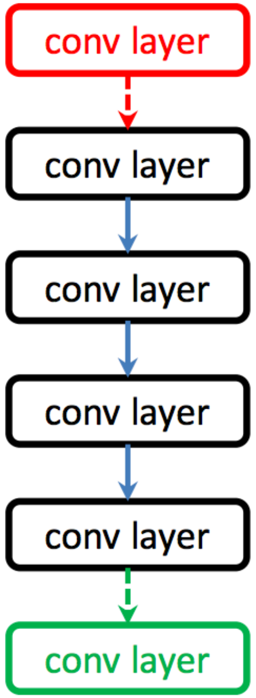
Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to “mutate” the neural network?

Randomly flip bits with some mutation probability



VGGNet

$K = 4$

Code: 1-01-001

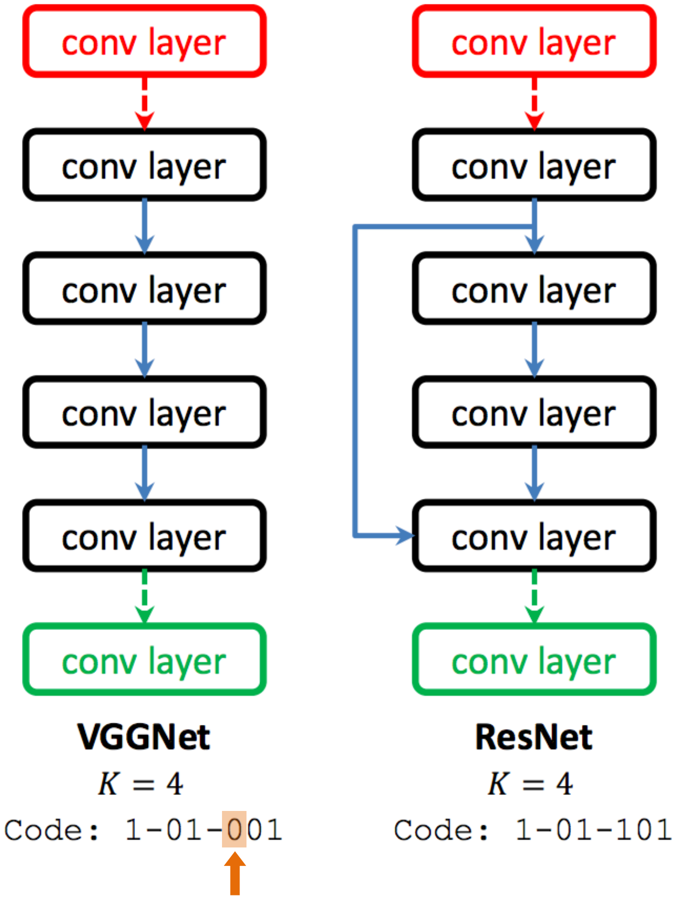
$K = 4$

Code: 1-01-101

L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to “mutate” the neural network?

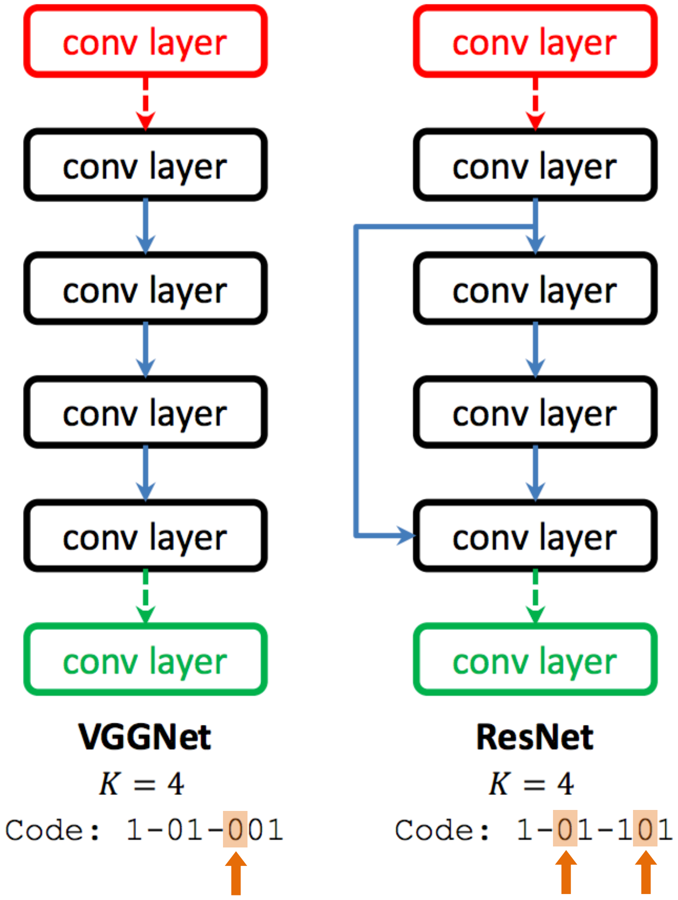
Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to "mutate" the neural network?

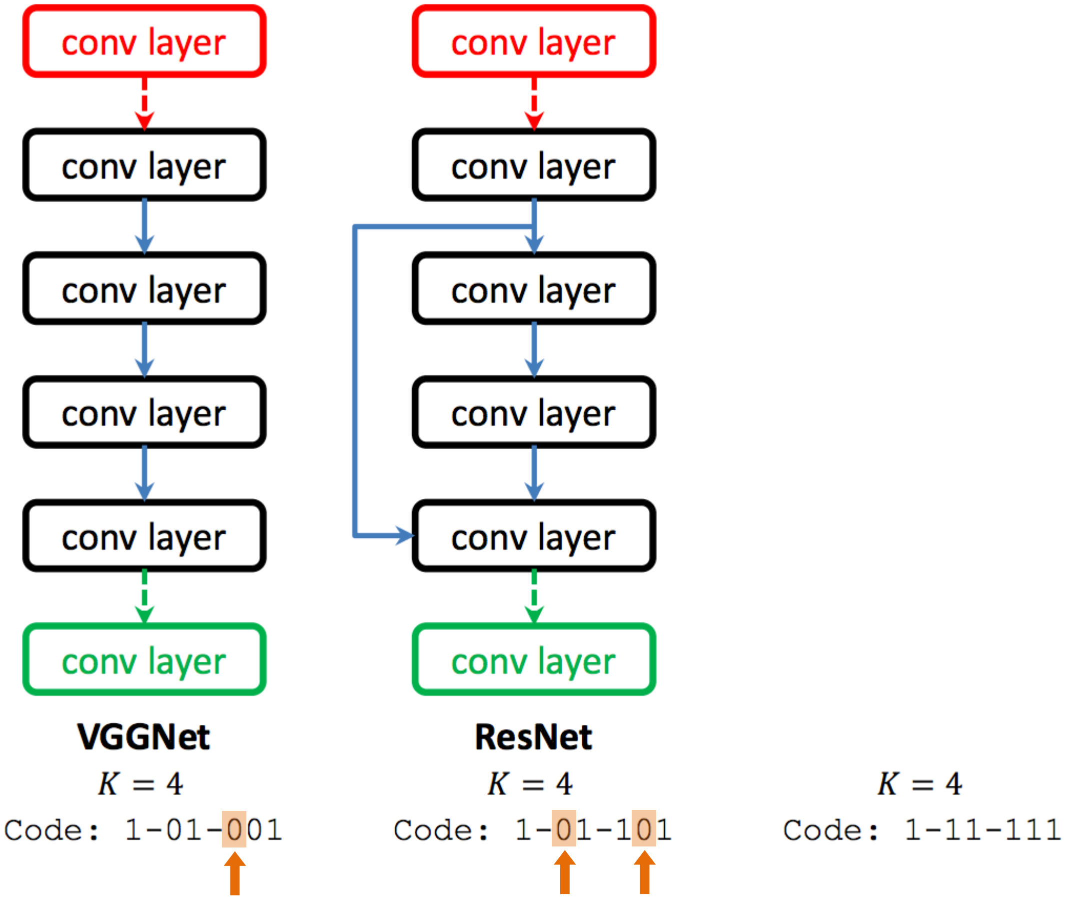
Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to “mutate” the neural network?

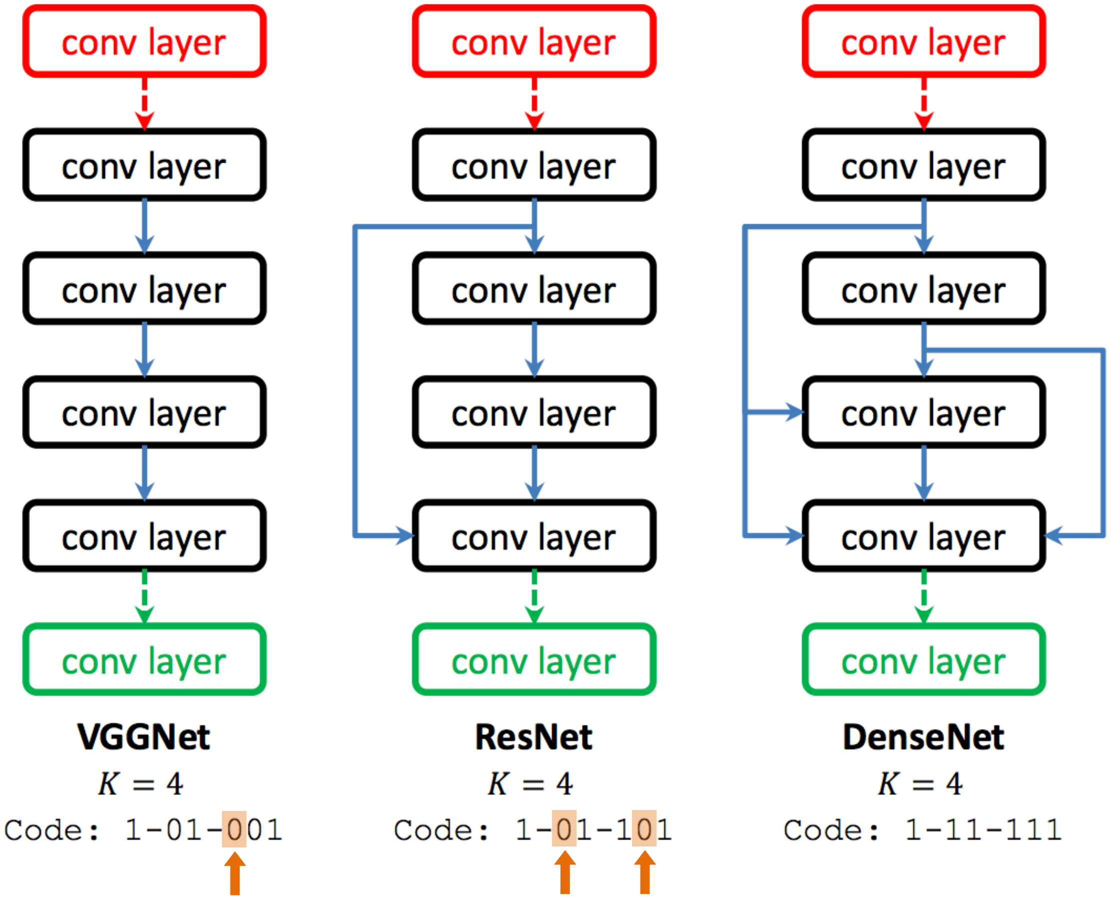
Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

How to “mutate” the neural network?

Randomly flip bits with some mutation probability



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

Genetic Algorithms and Neuroevolution

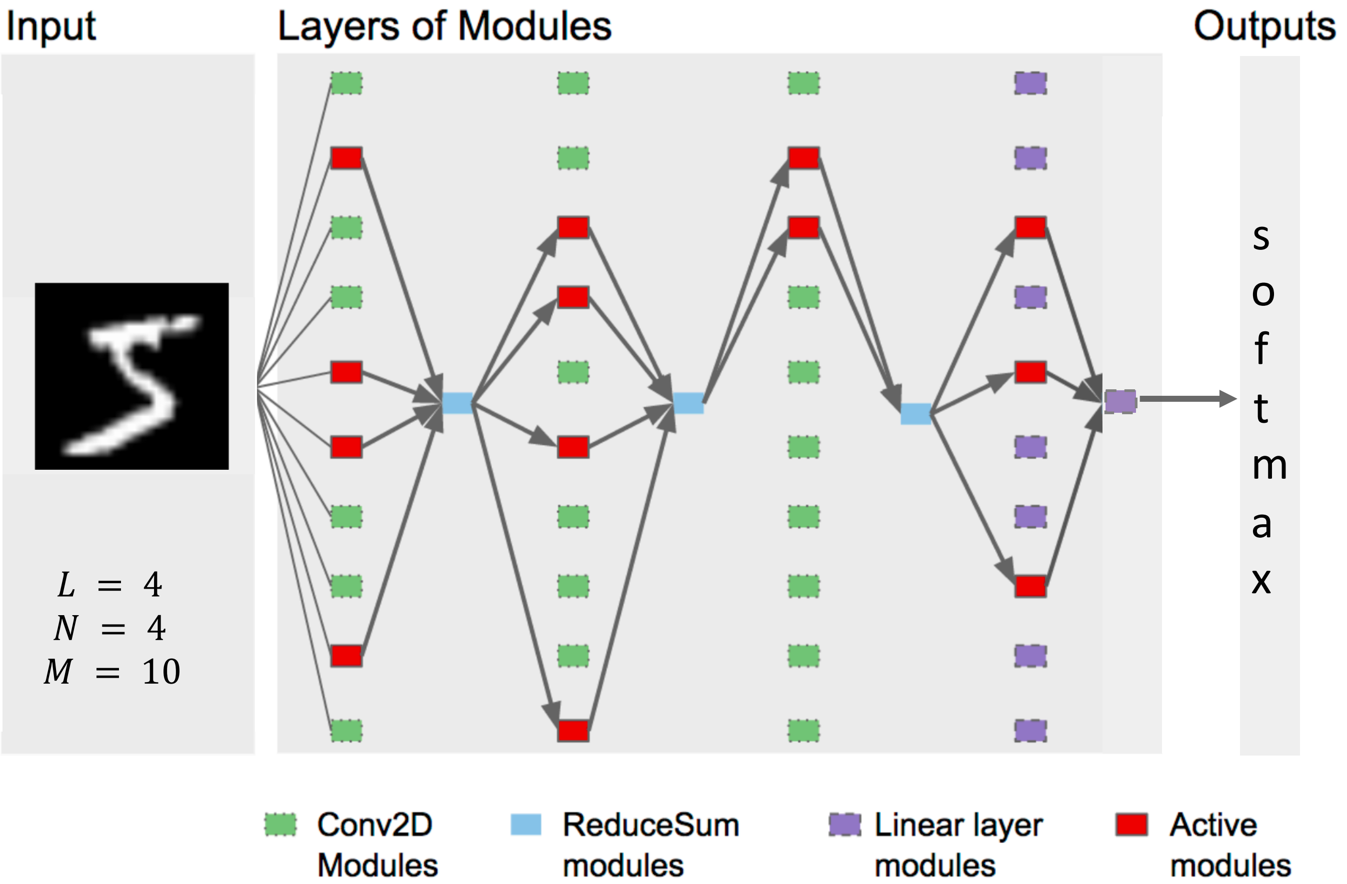
- **How to "mutate" the neural network?**
  - Dependent on the chosen encoding
    - Add structure – nodes and edges – to the graph?
    - Randomly flip bits with some mutation probability?
    - **Independently select with uniform probability the "active" modules in each layer?**

How to “mutate” the neural network?

Independently select with uniform probability the “active” modules in each layer

	I	II	III	IV
I	2	3	2	3
II	5	4	3	5
III	6	6	0	8
IV	9	10	0	0

$N \times L$



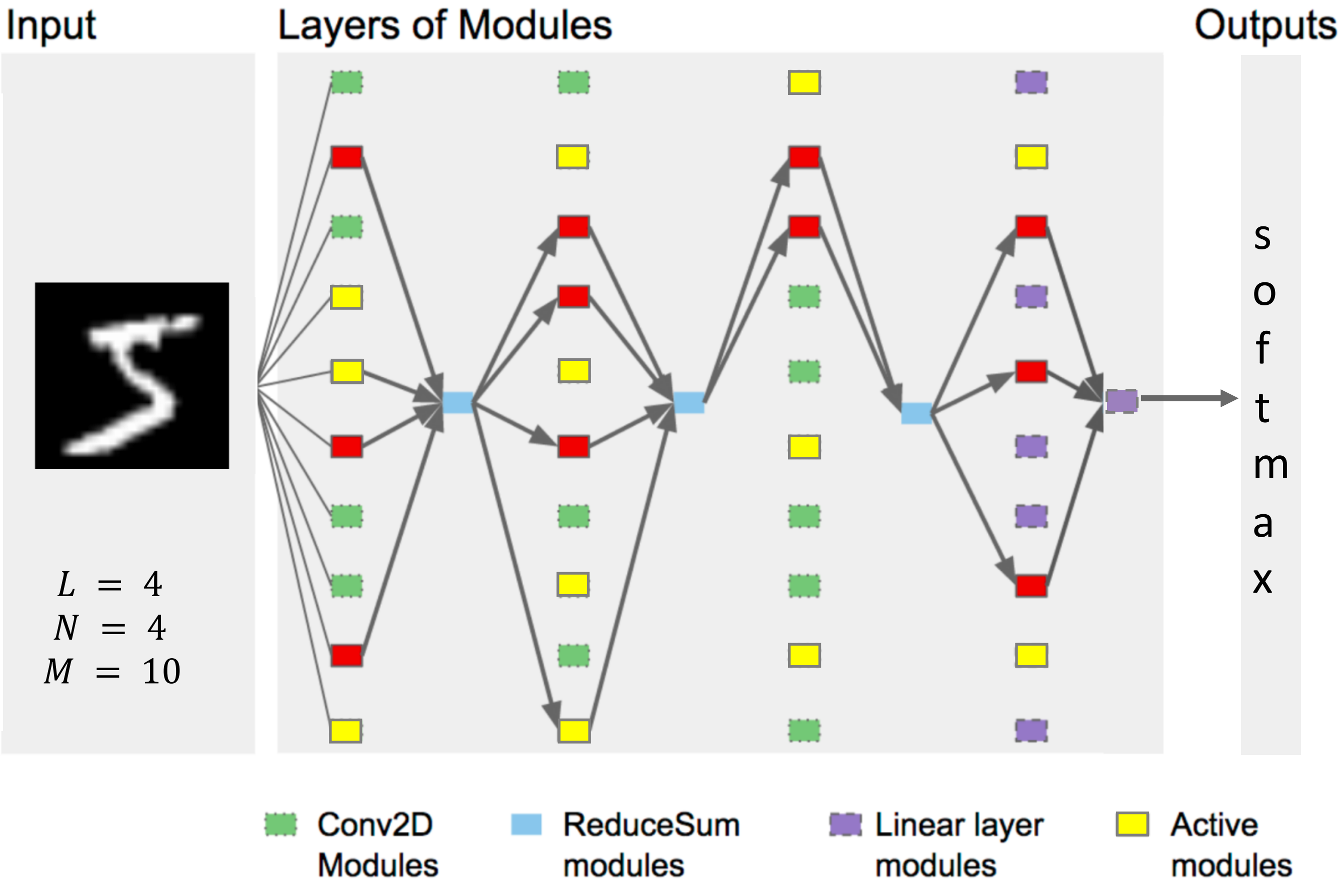
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

How to “mutate” the neural network?

Independently select with uniform probability the “active” modules in each layer

	I	II	III	IV
I	4	2	1	2
II	5	5	6	9
III	10	8	9	0
IV	0	10	0	0

$N \times L$



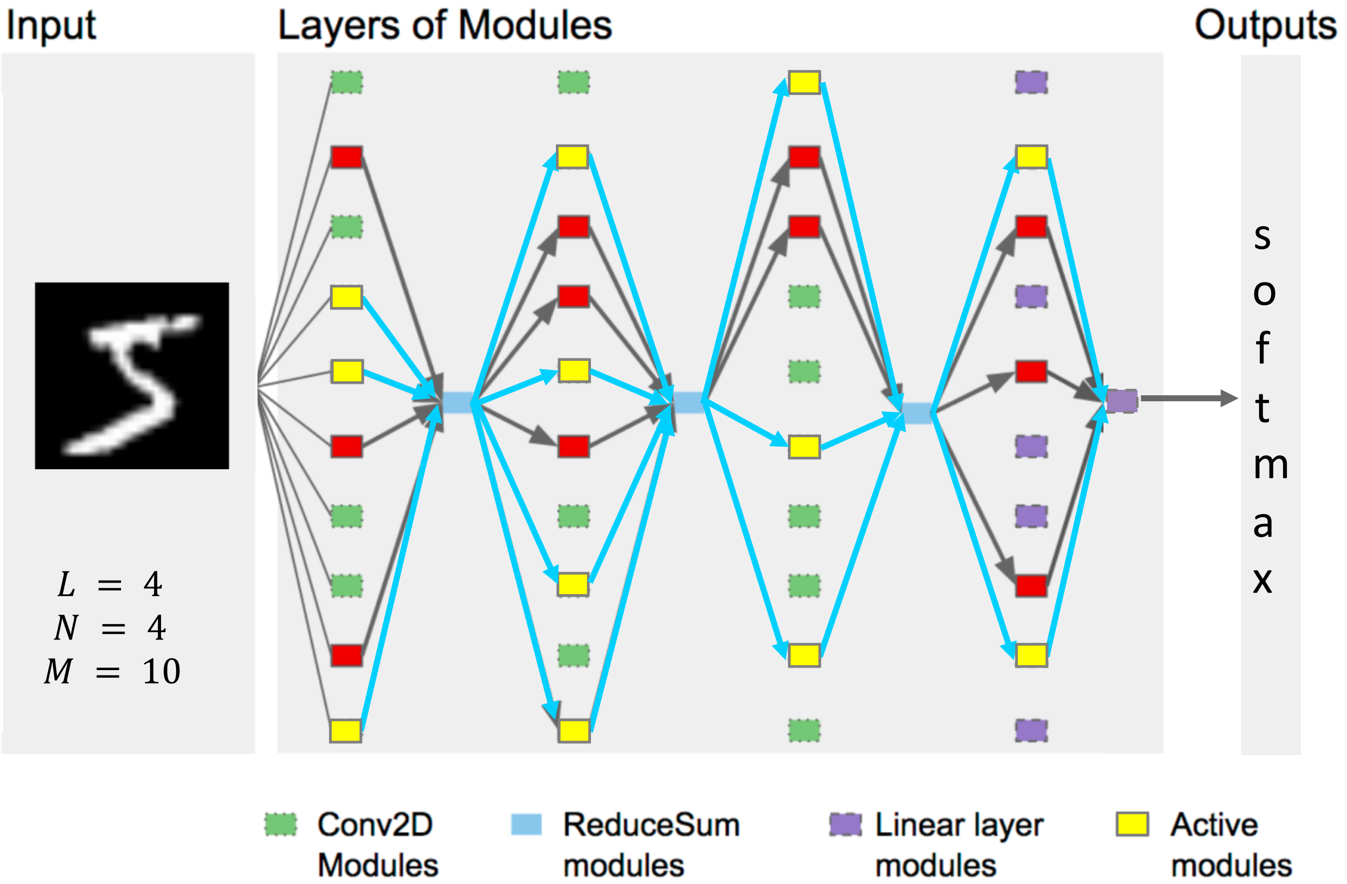
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

How to “mutate” the neural network?

Independently select with uniform probability the “active” modules in each layer

	I	II	III	IV
I	4	2	1	2
II	5	5	6	9
III	10	8	9	0
IV	0	10	0	0

$N \times L$



C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

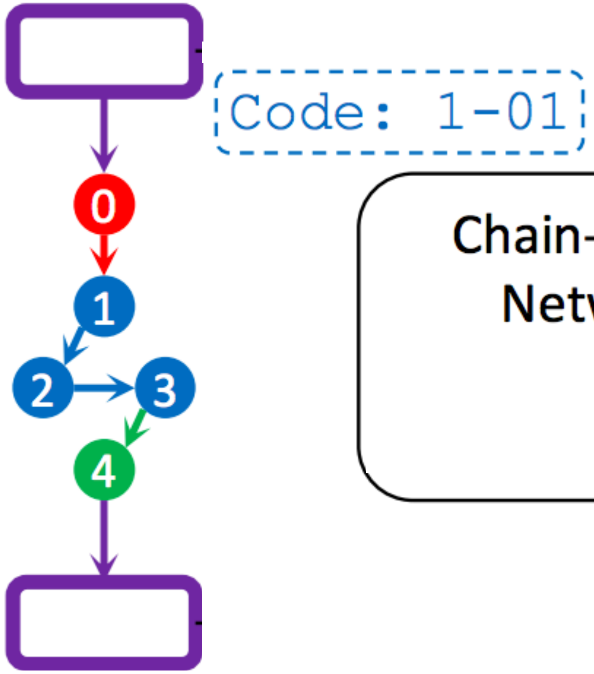
Genetic Algorithms and Neuroevolution

- How to represent the "DNA" of neural networks?
  - Choose meaningful encoding
- How to "mutate" the neural network?
  - Dependent on the chosen encoding
- **How to evaluate fitness for each learned architecture?**
  - Train for fixed number of epochs on task of interest, evaluate performance
- What architectures can be learned with evolution?

Genetic Algorithms and Neuroevolution

- How to represent the “DNA” of neural networks?
  - Choose meaningful encoding
- How to “mutate” the neural network?
  - Dependent on the chosen encoding
- How to evaluate fitness for each learned architecture?
  - Train for fixed number of epochs on task of interest, evaluate performance
- **What architectures can be learned with evolution?**

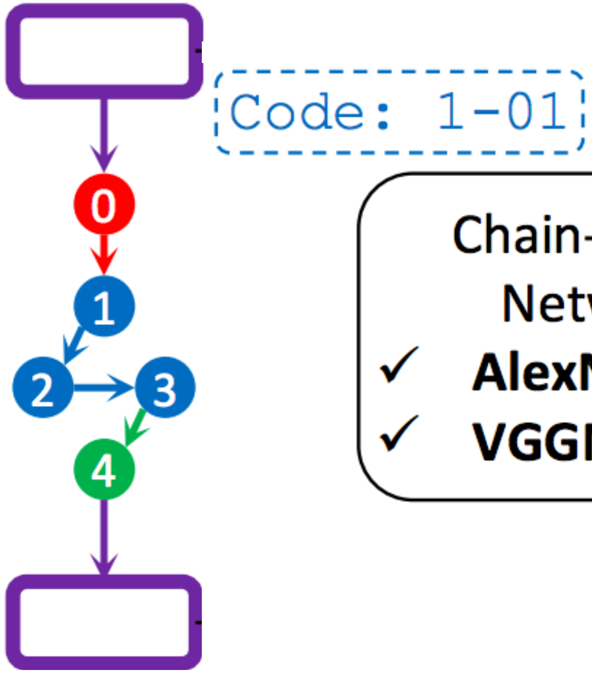
Learned Network Structures



Chain-Shaped Networks

L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

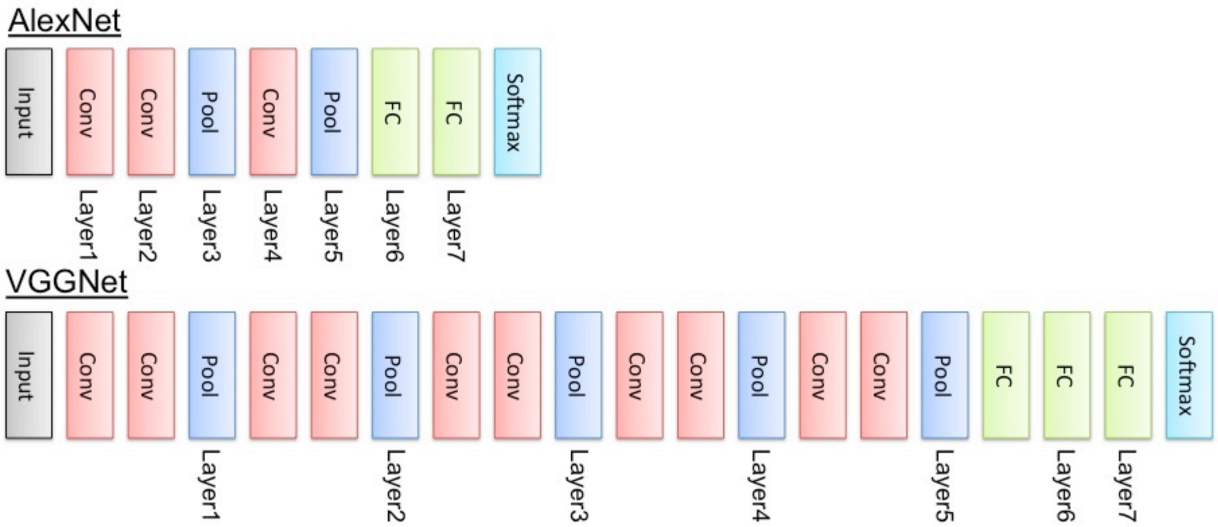
Learned Network Structures



Chain-Shaped Networks

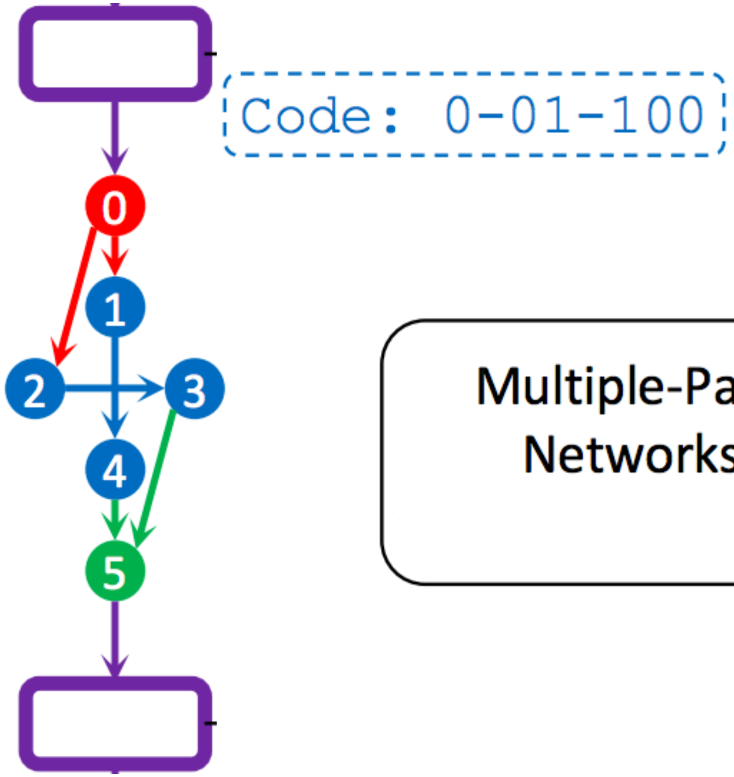
✓ **AlexNet**

✓ **VGGNet**

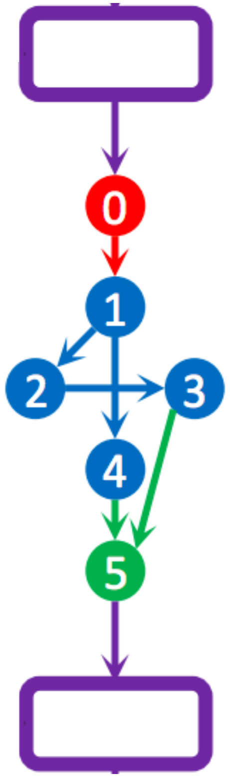


L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

Learned Network Structures

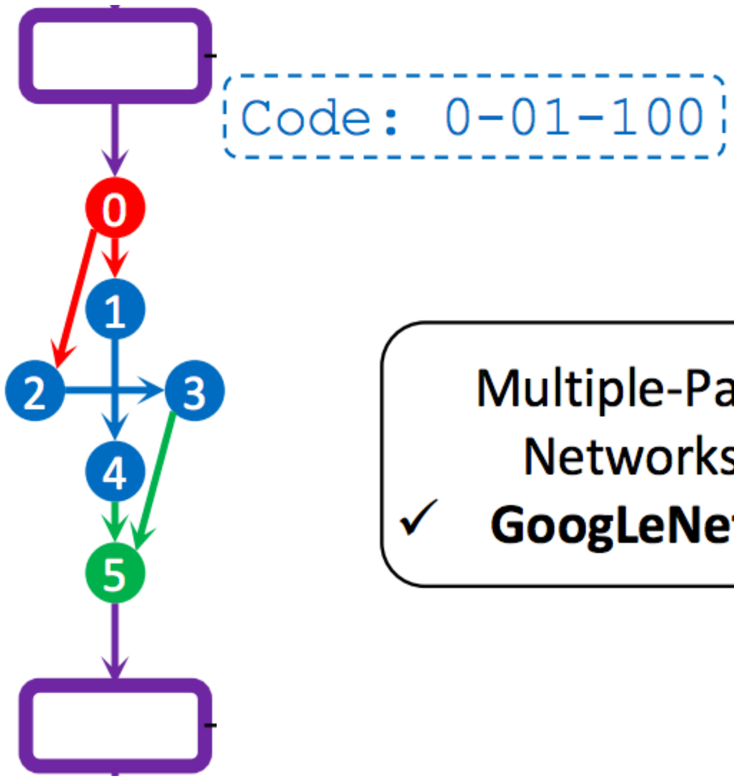


Multiple-Path Networks

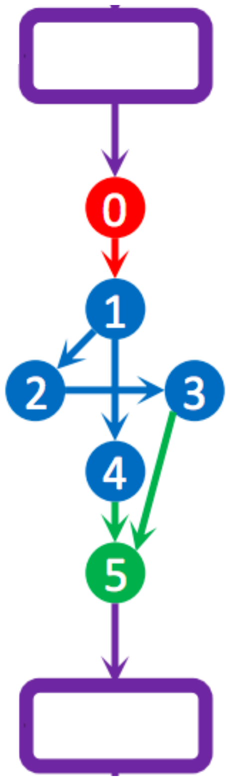


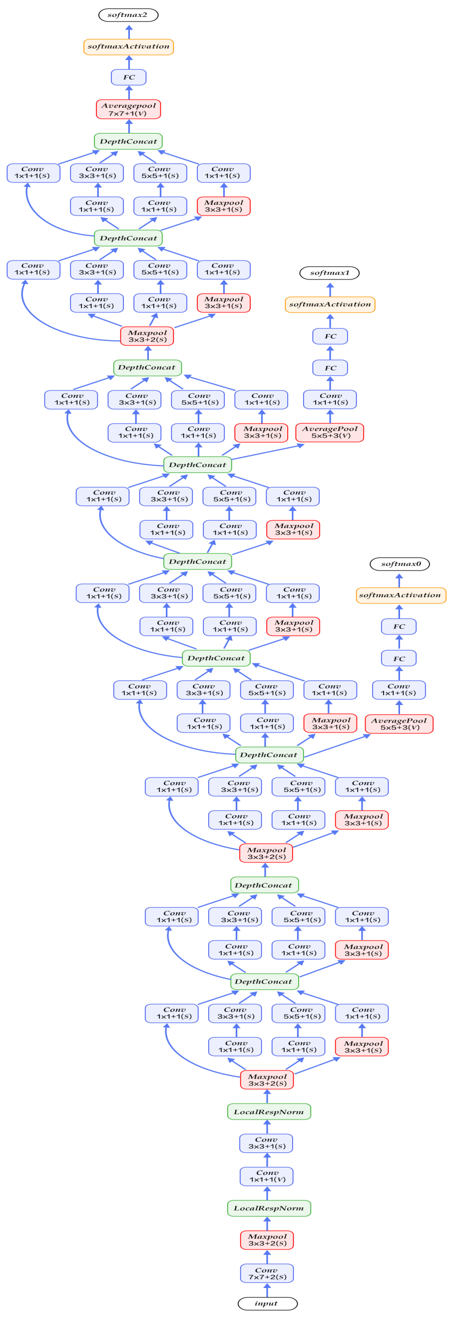
L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

Learned Network Structures



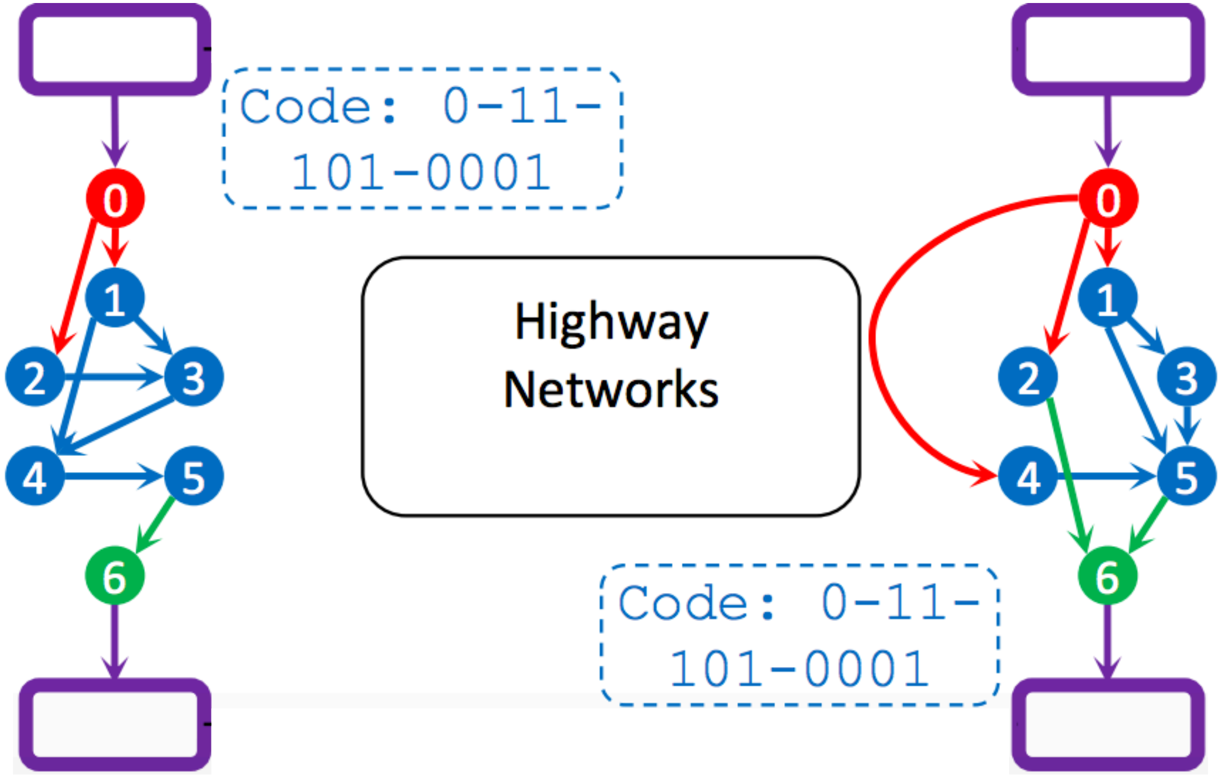
Multiple-Path
Networks
✓ **GoogLeNet**





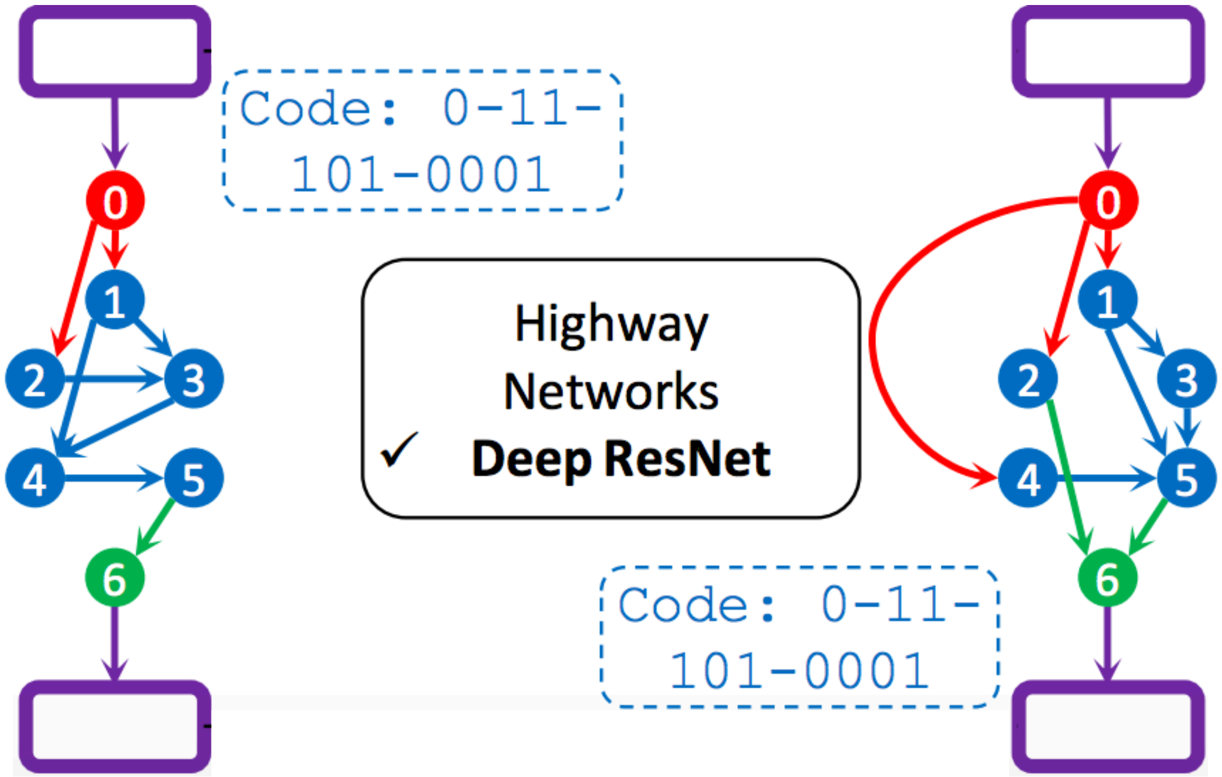
L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

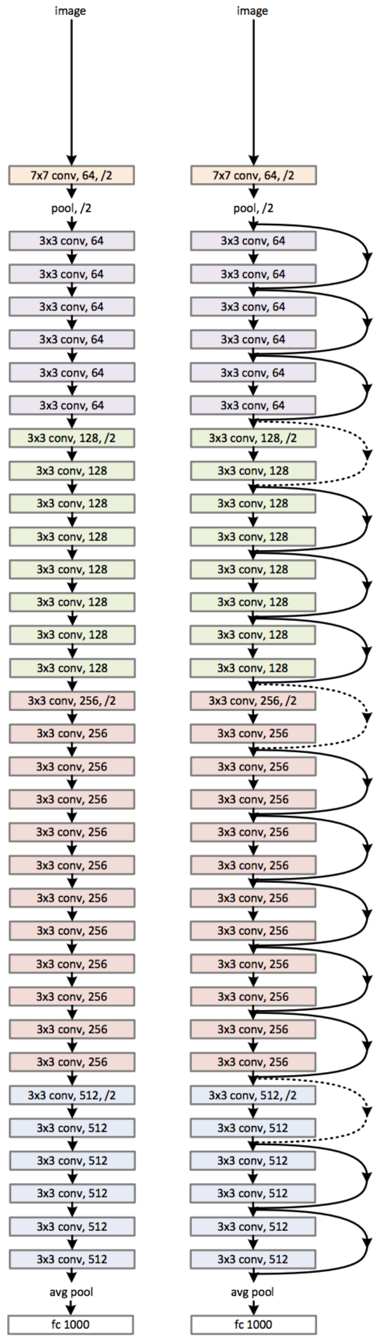
Learned Network Structures



L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

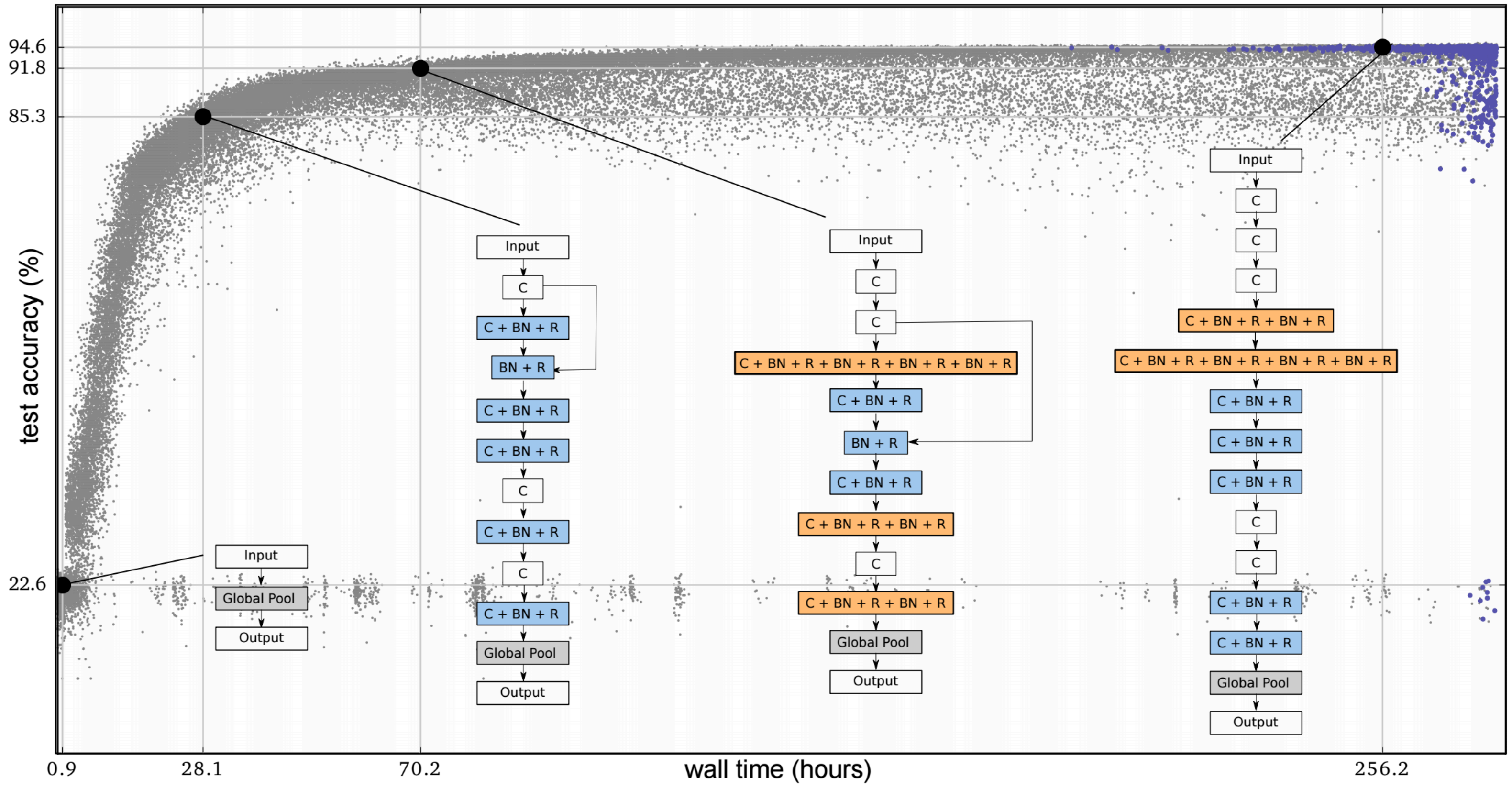
Learned Network Structures





L. Xie and A. Yuille. Genetic CNN. arXiv preprint arXiv:1703.01513 (2017).

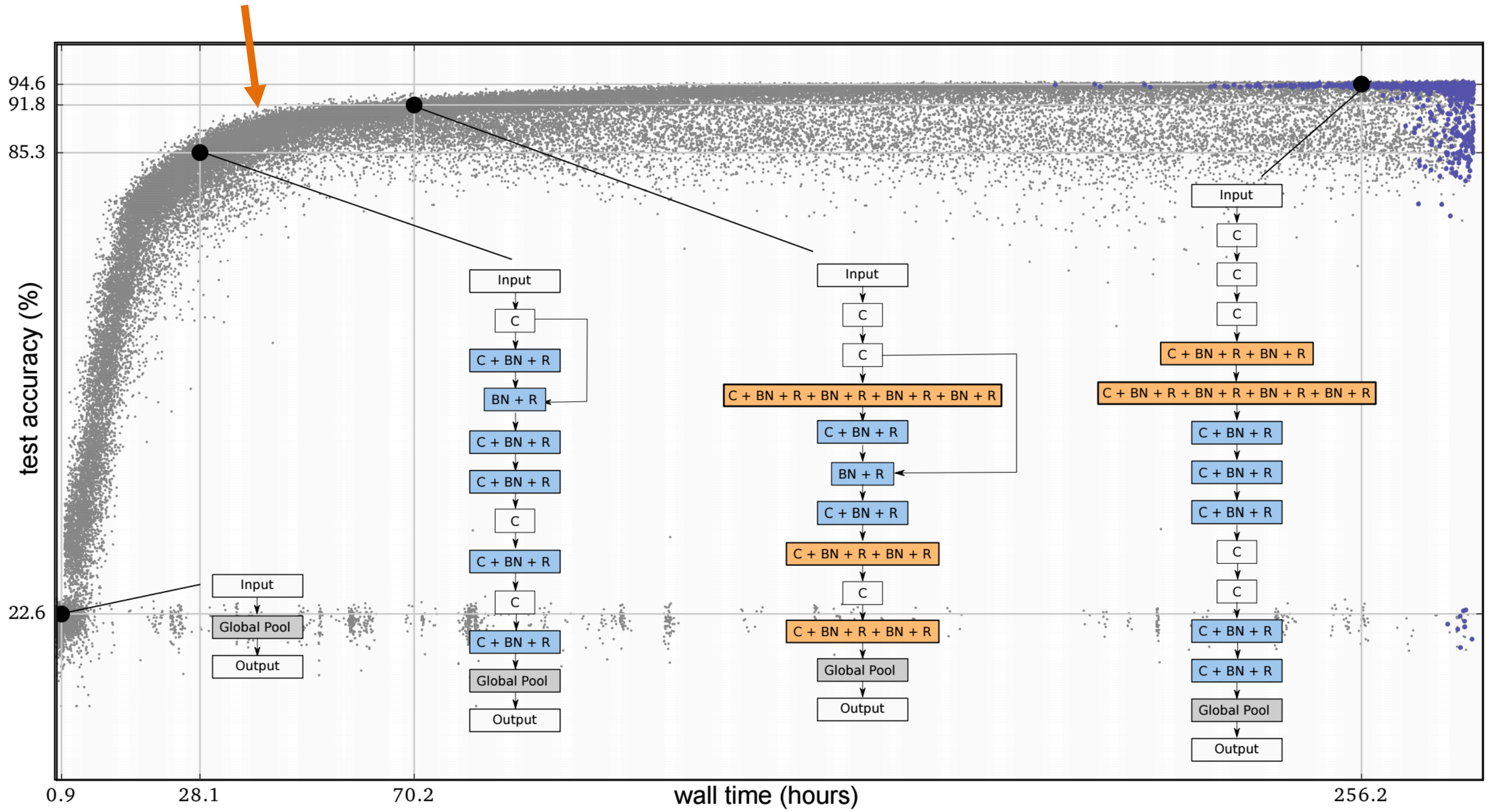
Learned Network Structures



E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

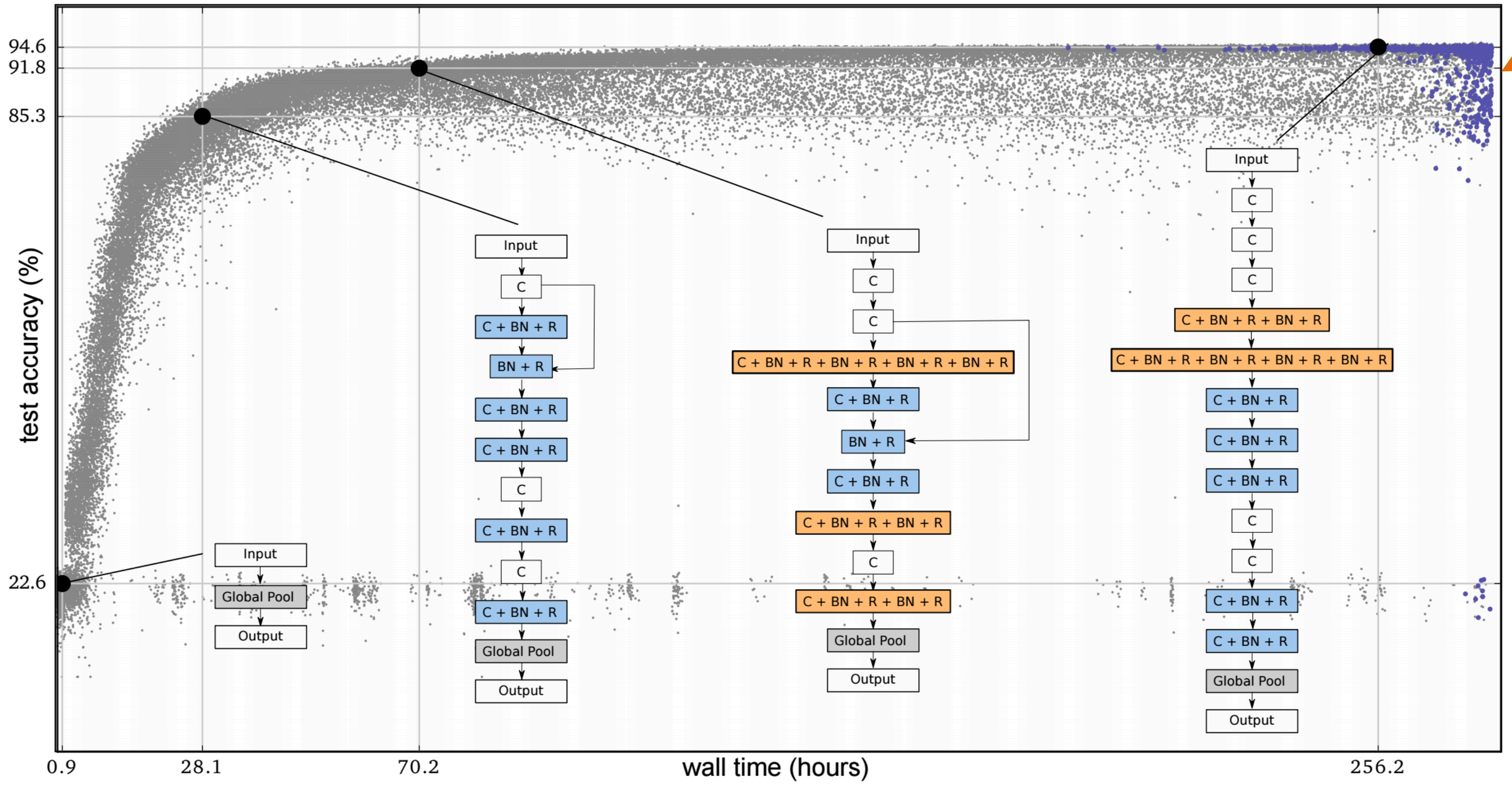
Learned Network Structures

each dot represents one individual in the population



E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

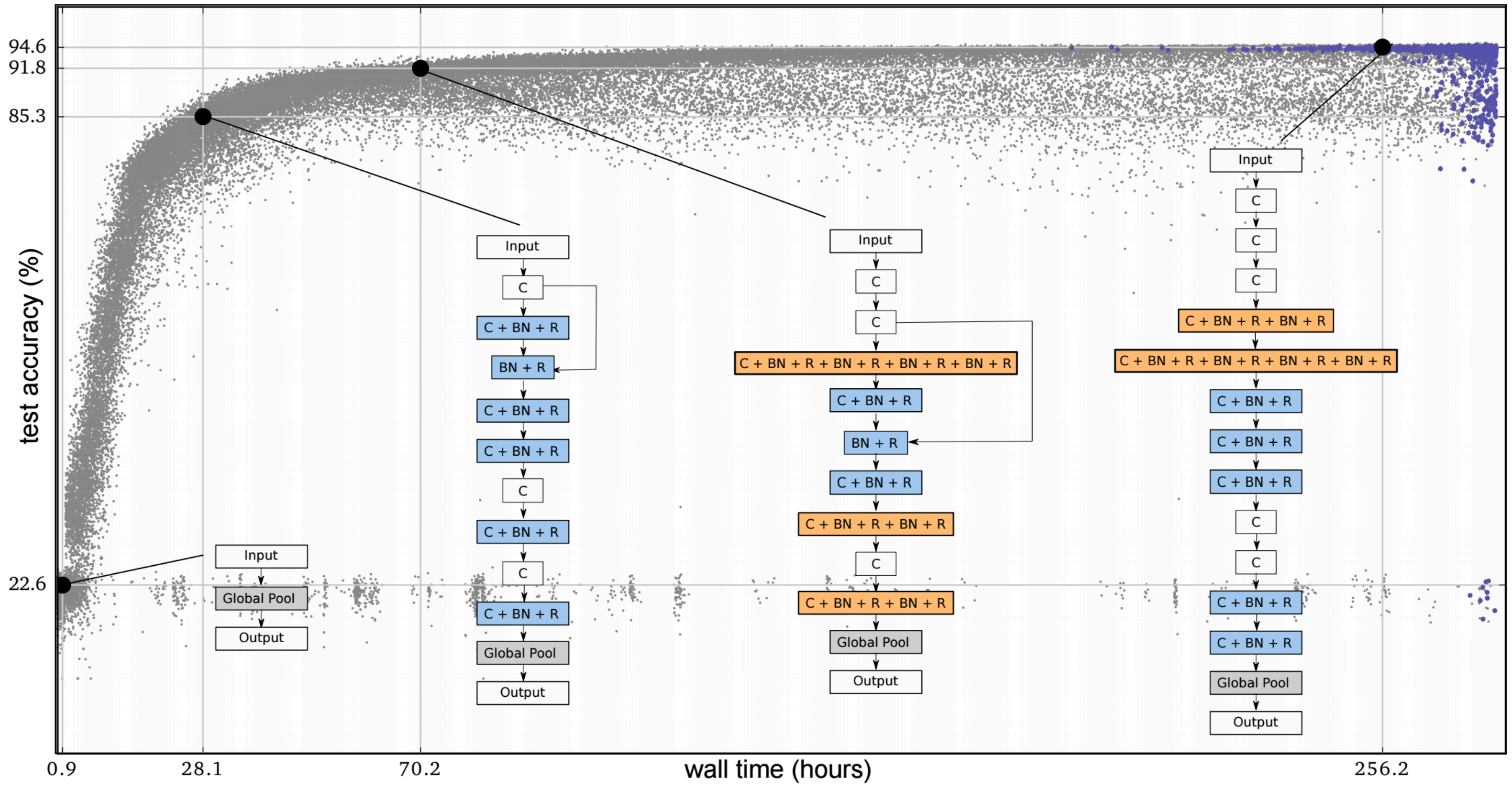
Learned Network Structures



blue dots are "alive"
(free to act as parents)

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

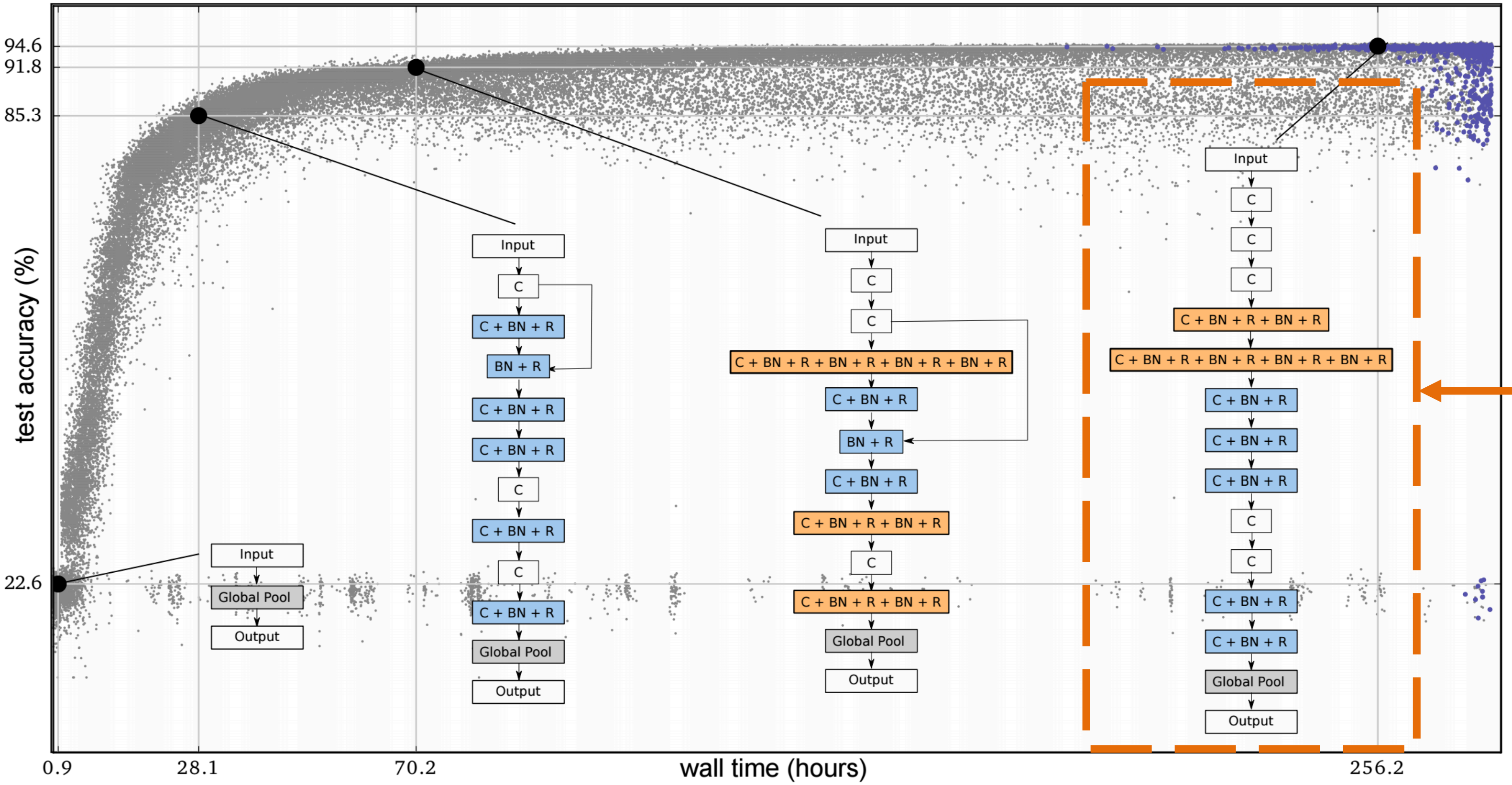
Learned Network Structures



failed mutations

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

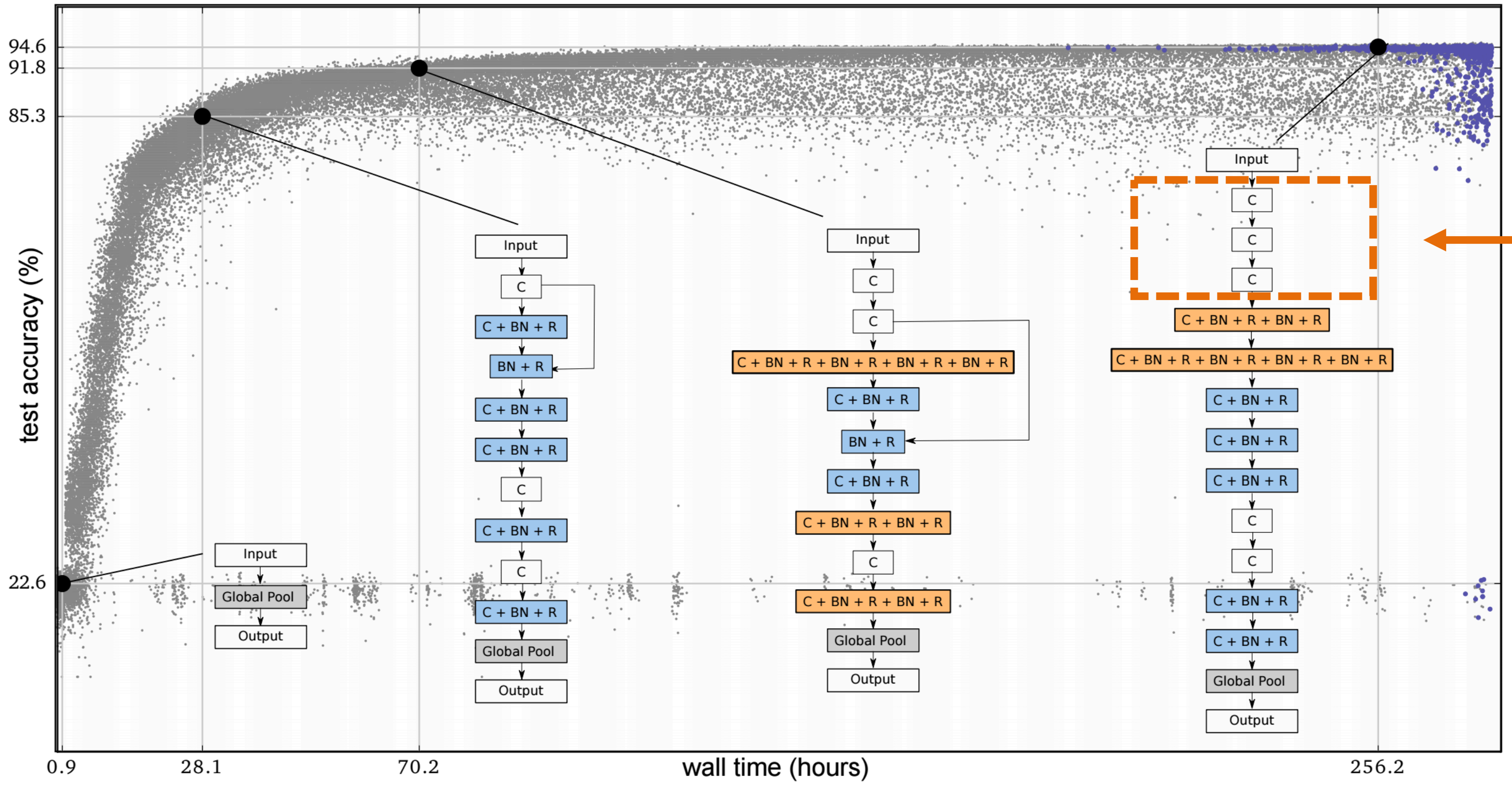
Learned Network Structures



best discovered architecture

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

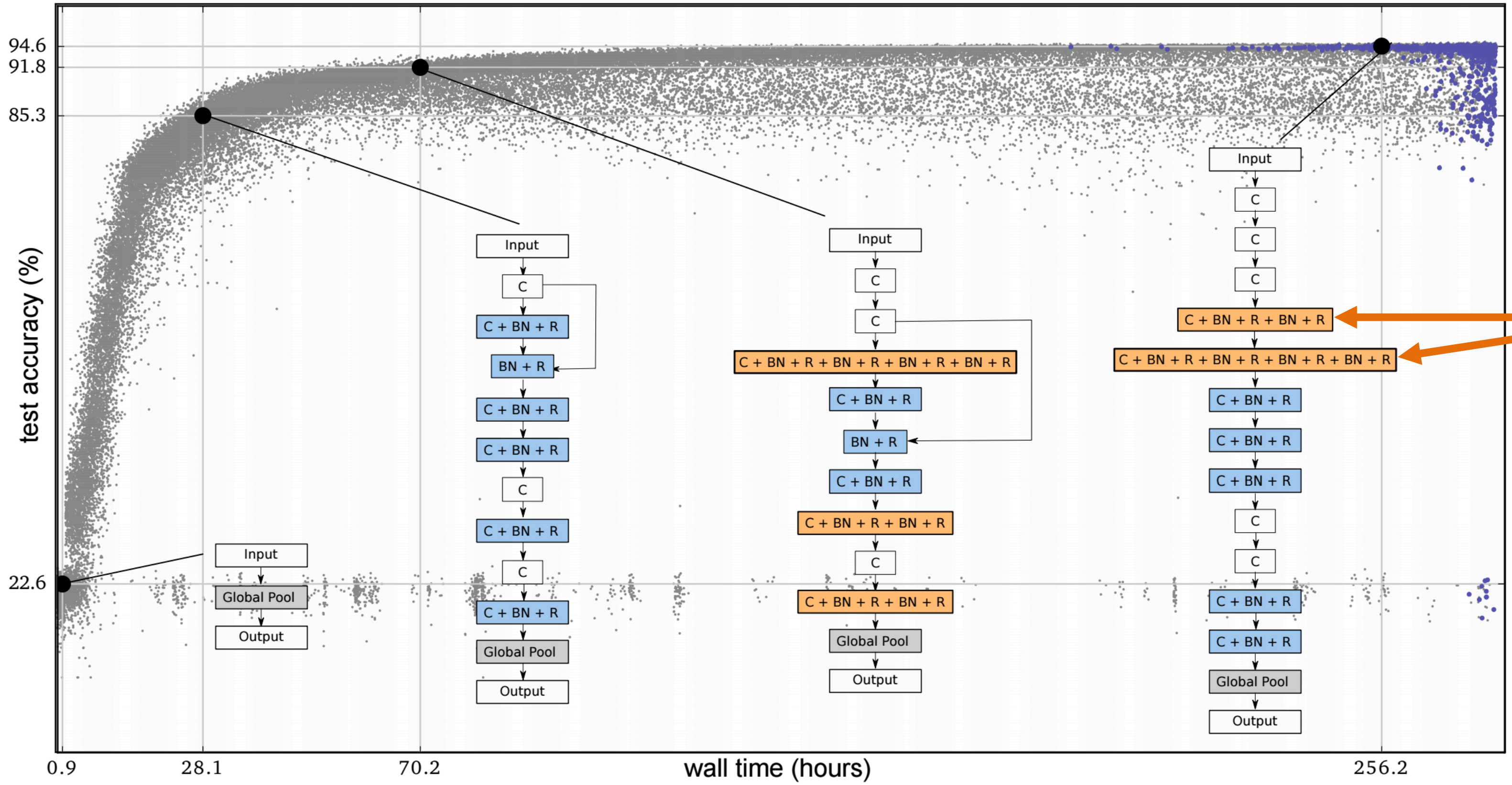
Learned Network Structures



evolution sometimes stacks convolutions without any non-linearity in between

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

Learned Network Structures

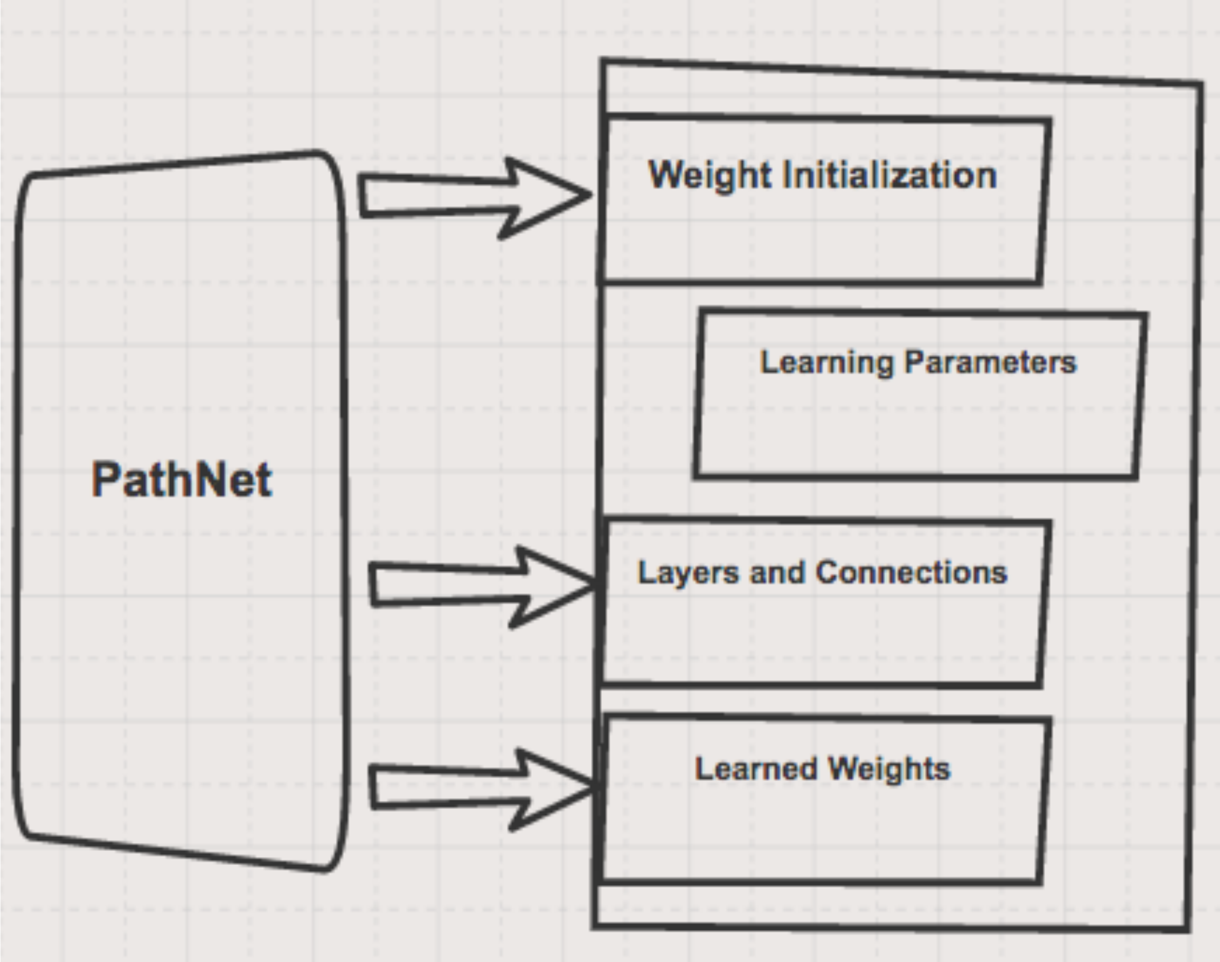


some convolutions are followed by more than one nonlinearity

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

Case Study: PathNet

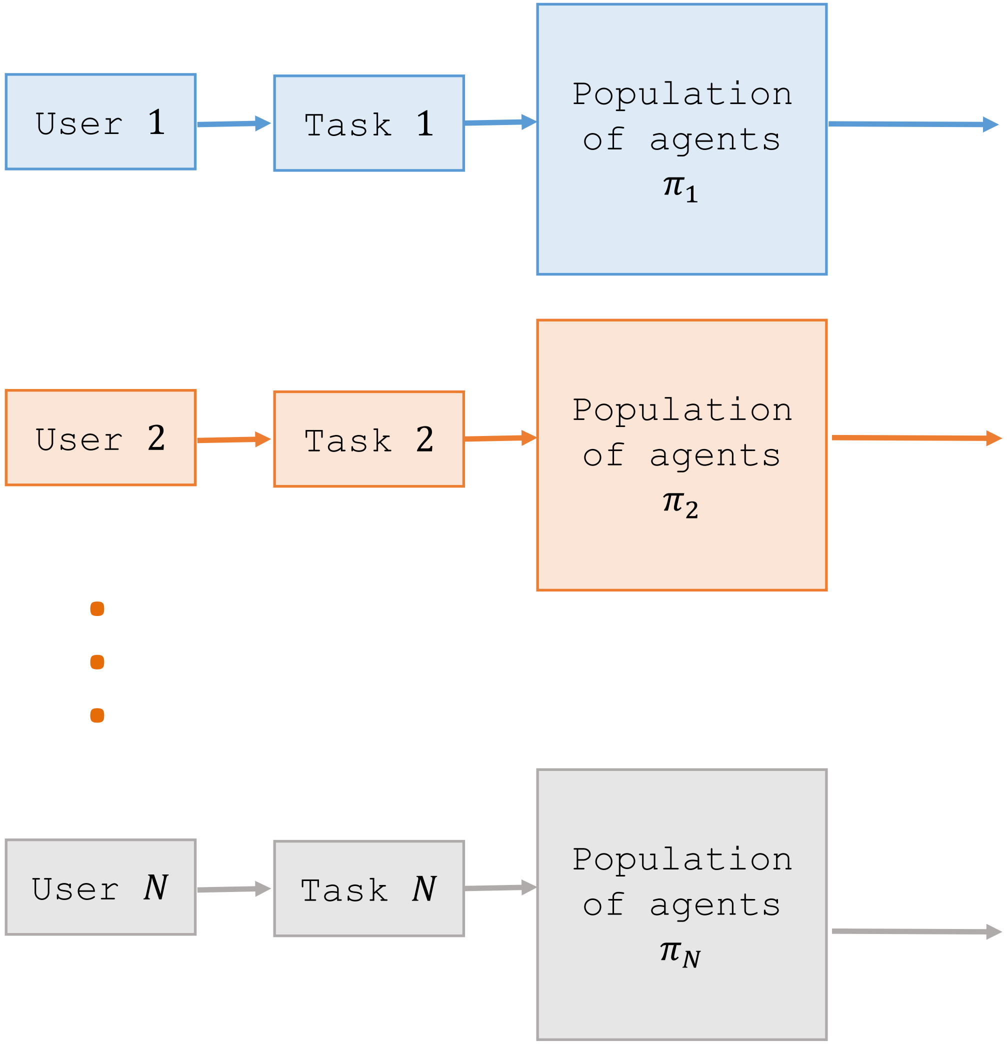
Can we **"learn without forgetting"** – reuse components of the *same* giant neural network, but for *different* tasks?

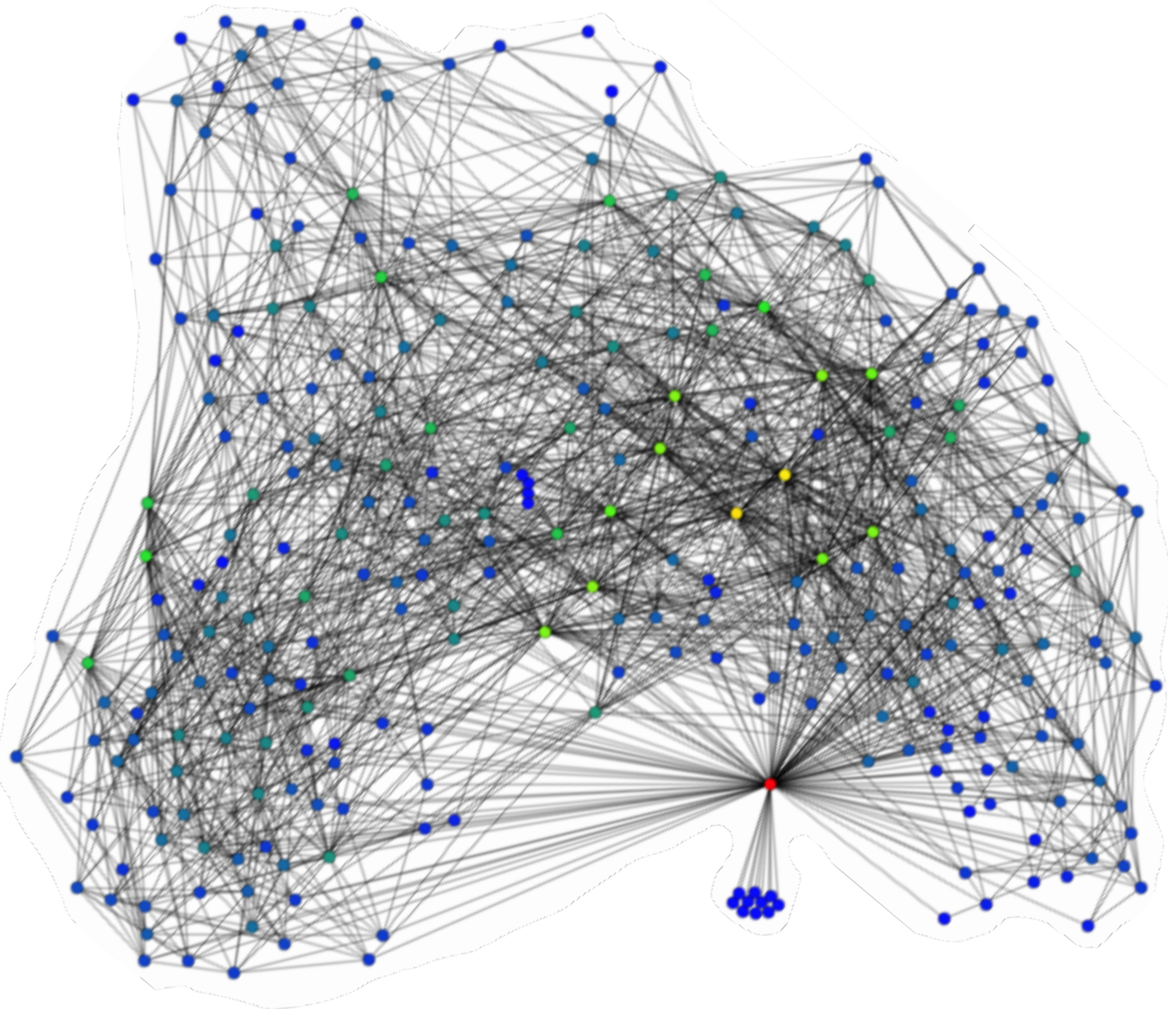


[Blog by Carlos E. Perez]

C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

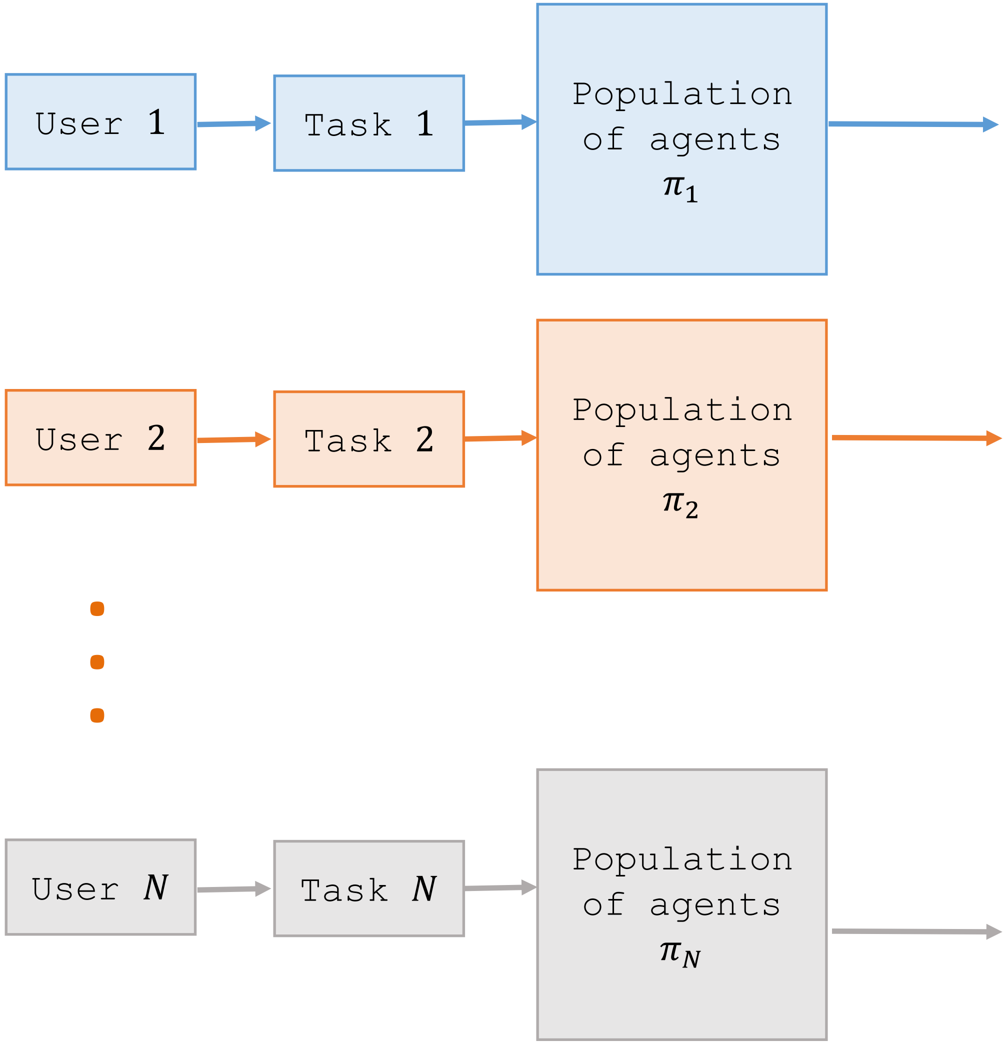
Case Study: PathNet

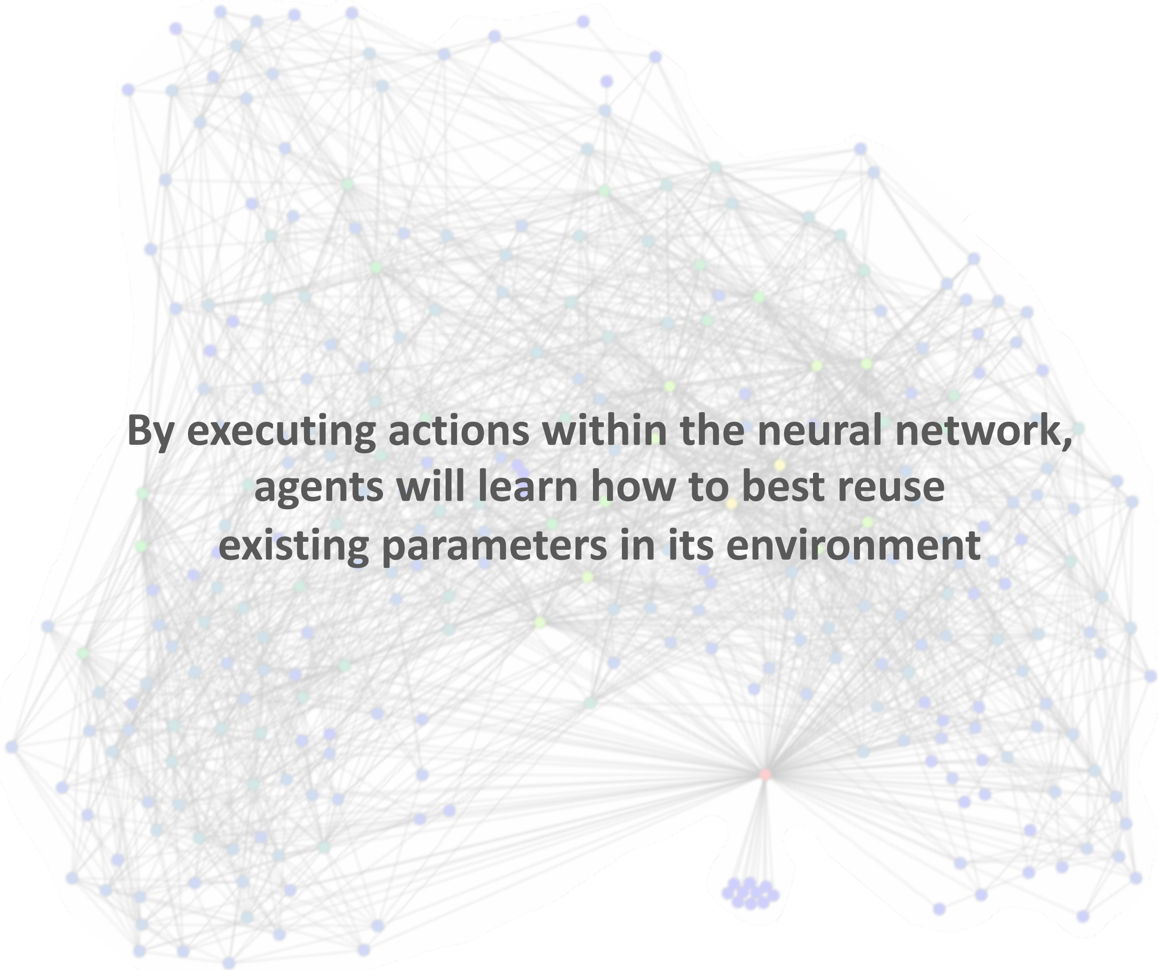




C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet





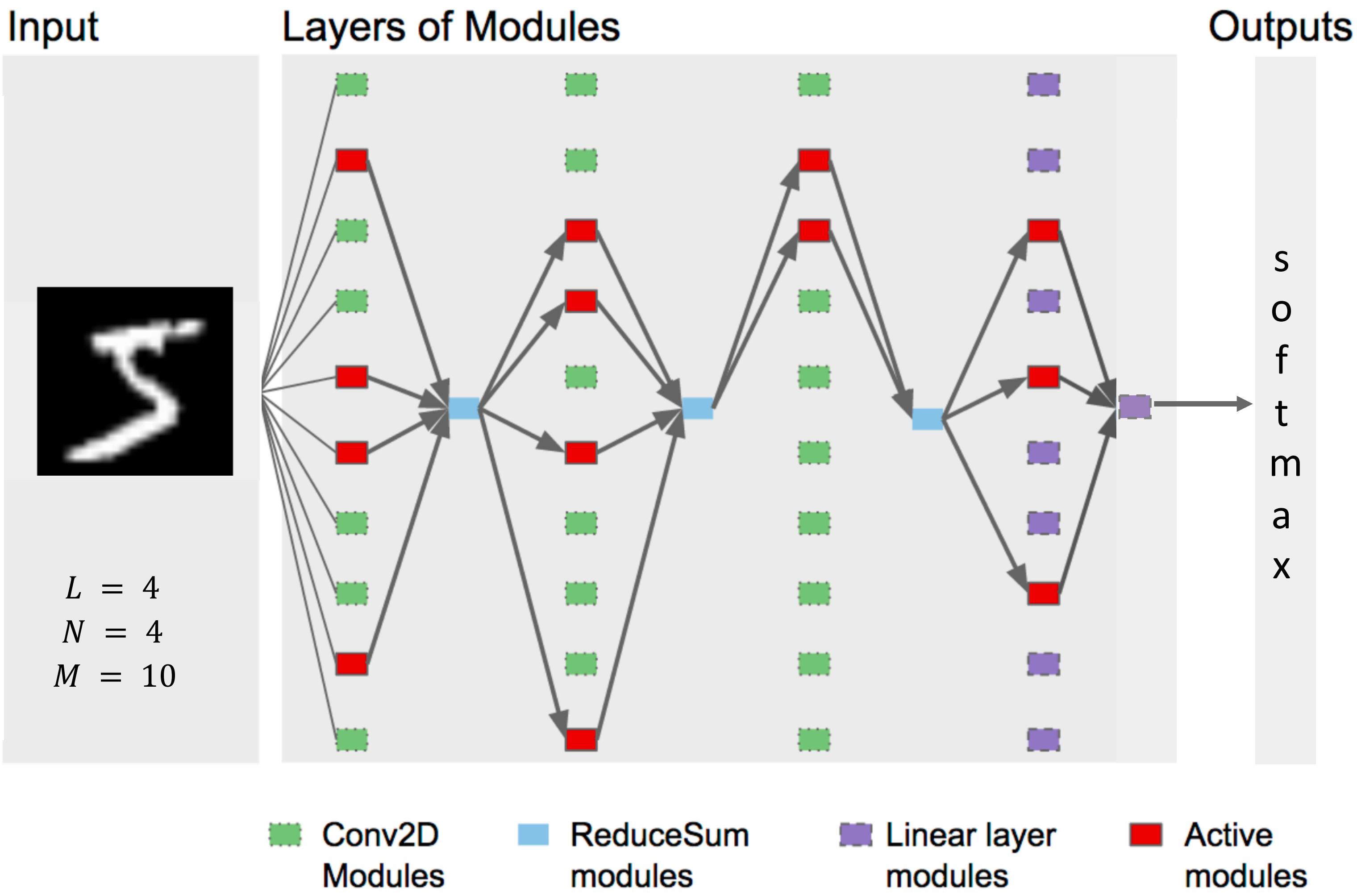
By executing actions within the neural network, agents will learn how to best reuse existing parameters in its environment

C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Architecture

- L layers
- Each layer consists of M modules followed by transfer function
- For each layer, the outputs of the modules are summed before being passed into the active modules of the next layer
- A module is **active** if it is present in the path genotype currently being evaluated
- A maximum of N distinct modules are permitted in a pathway
- The final layer is **unique** and **unshared** for each task being learned

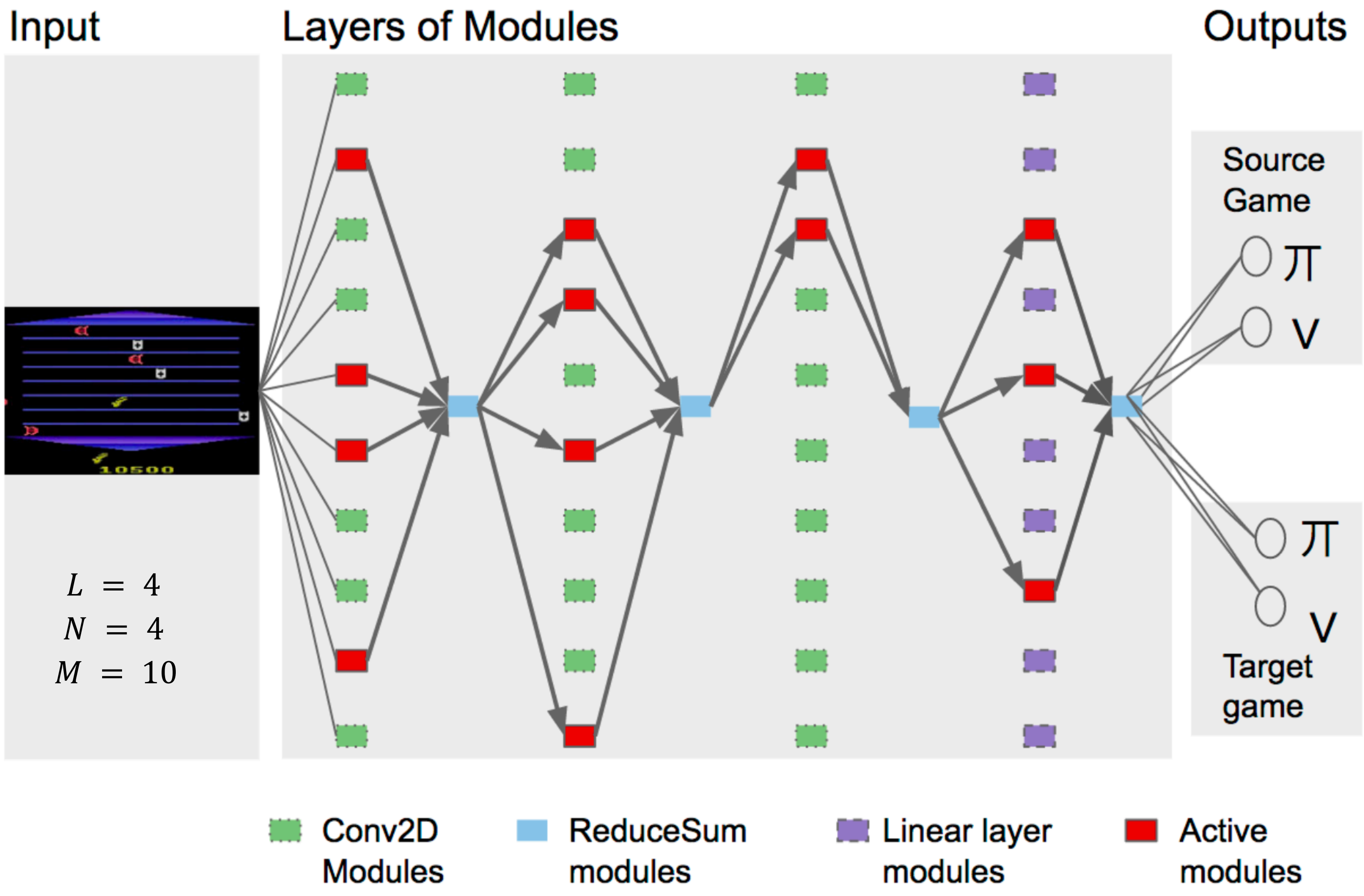


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Architecture

- L layers
- Each layer consists of M modules followed by transfer function
- For each layer, the outputs of the modules are summed before being passed into the active modules of the next layer
- A module is **active** if it is present in the path genotype currently being evaluated
- A maximum of N distinct modules are permitted in a pathway
- The final layer is **unique** and **unshared** for each task being learned



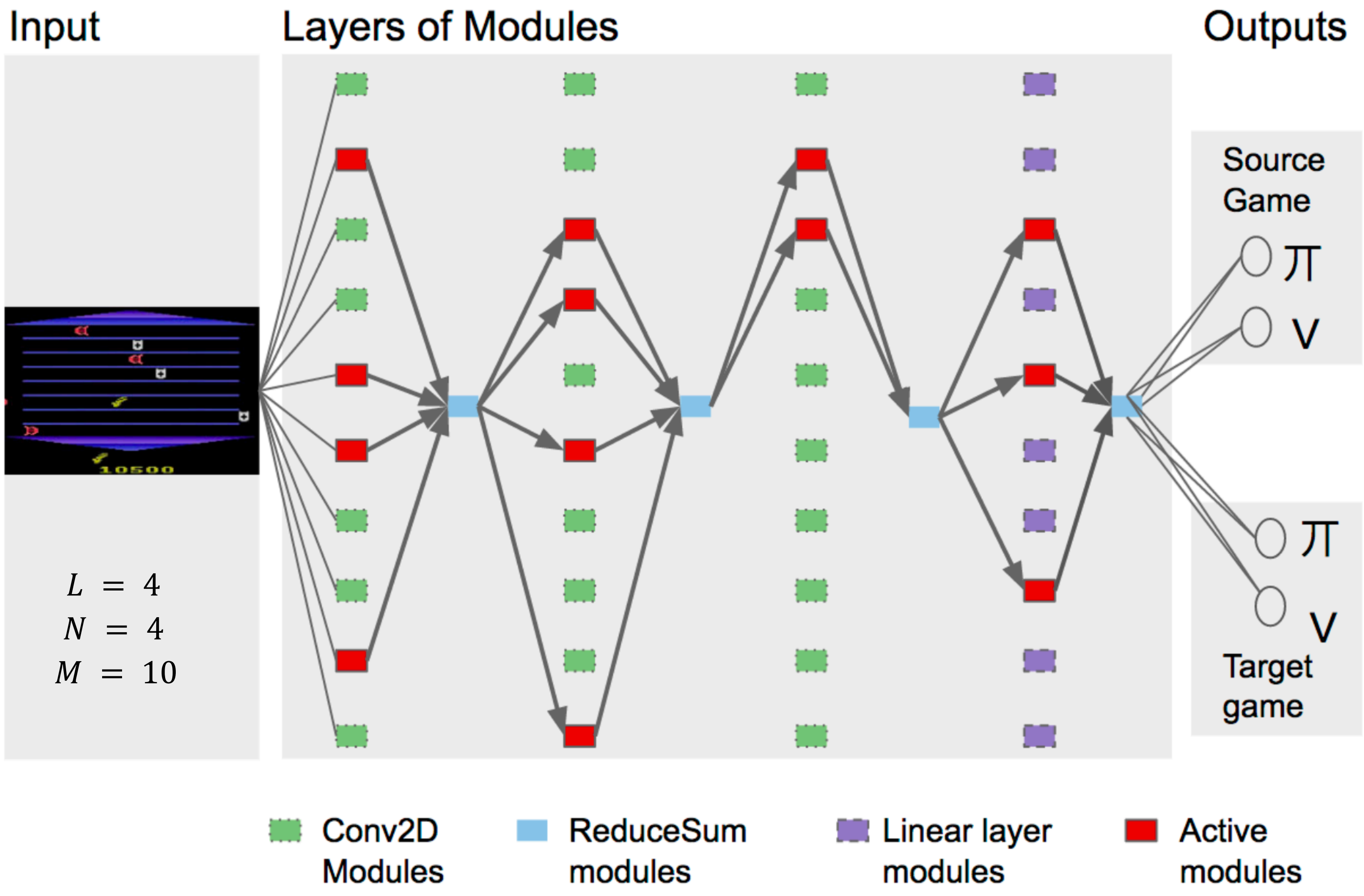
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Representation of genotype

	I	II	III	IV
I	2	3	2	3
II	5	4	3	5
III	6	6	0	8
IV	9	10	0	0

$N \times L$



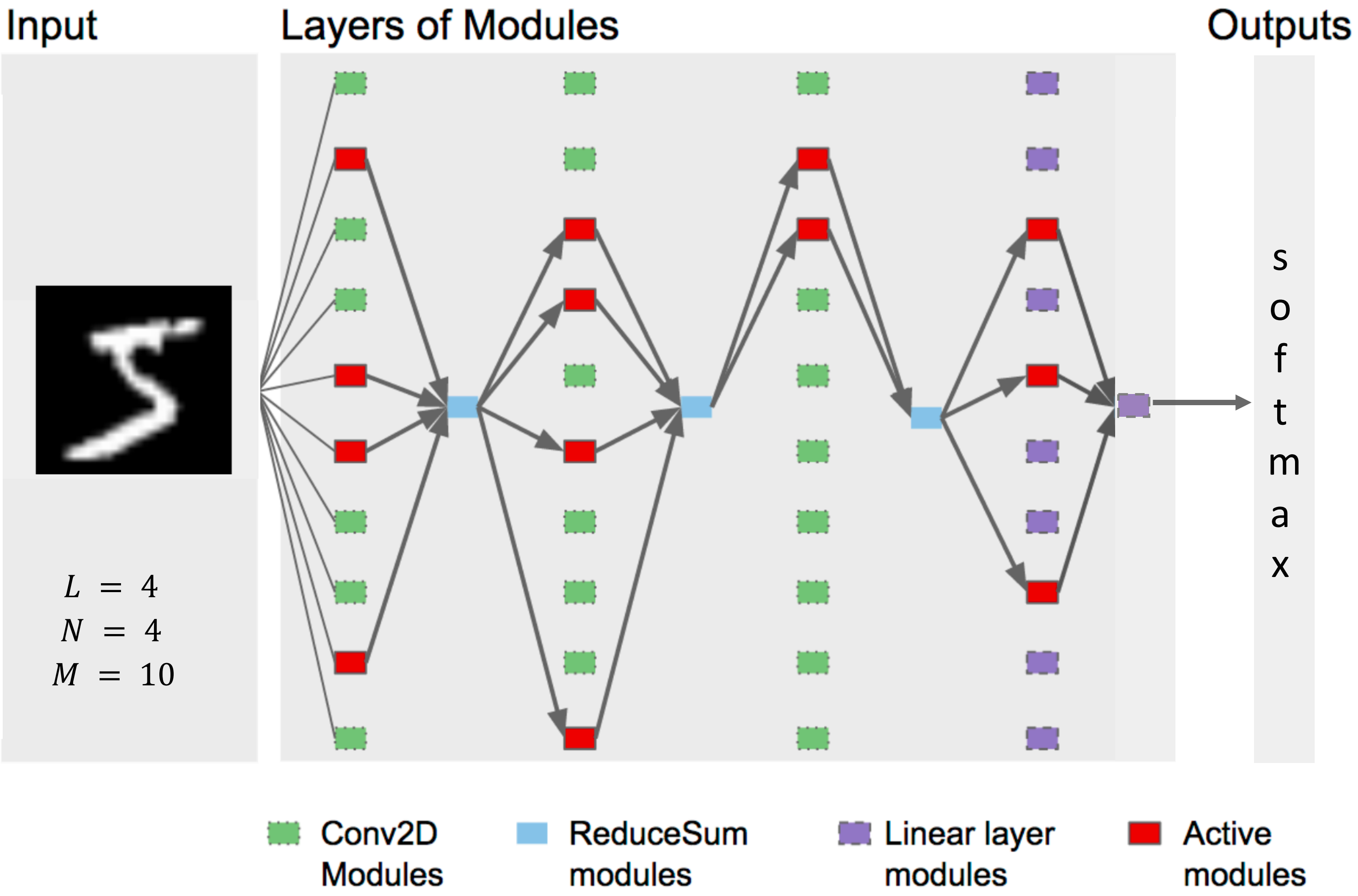
$$L = 4$$
$$N = 4$$
$$M = 10$$

C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose random genotype
- Train its pathway T epochs
- Evaluate its fitness

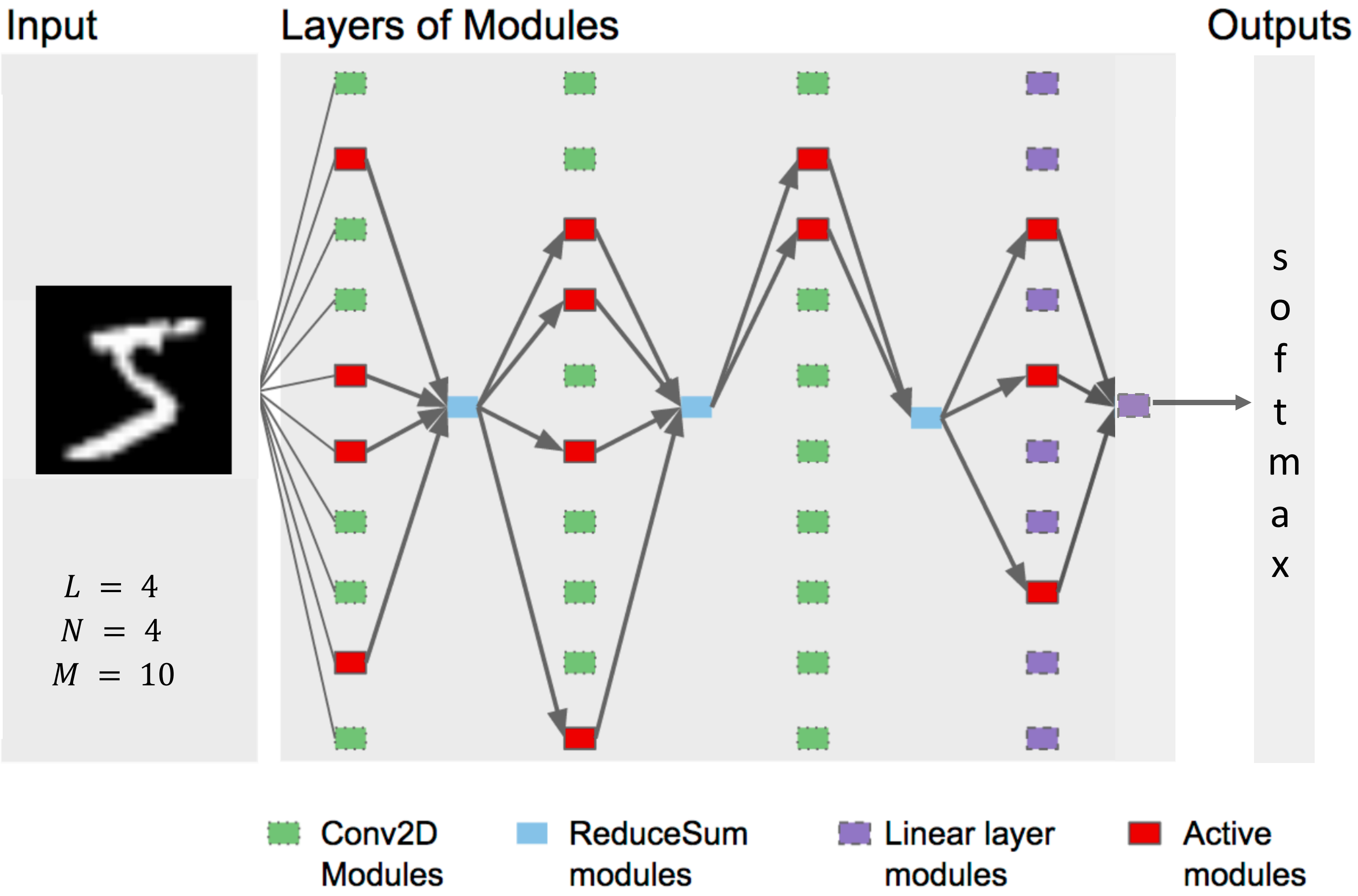


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose random genotype
- Train its pathway T epochs
- Evaluate its fitness

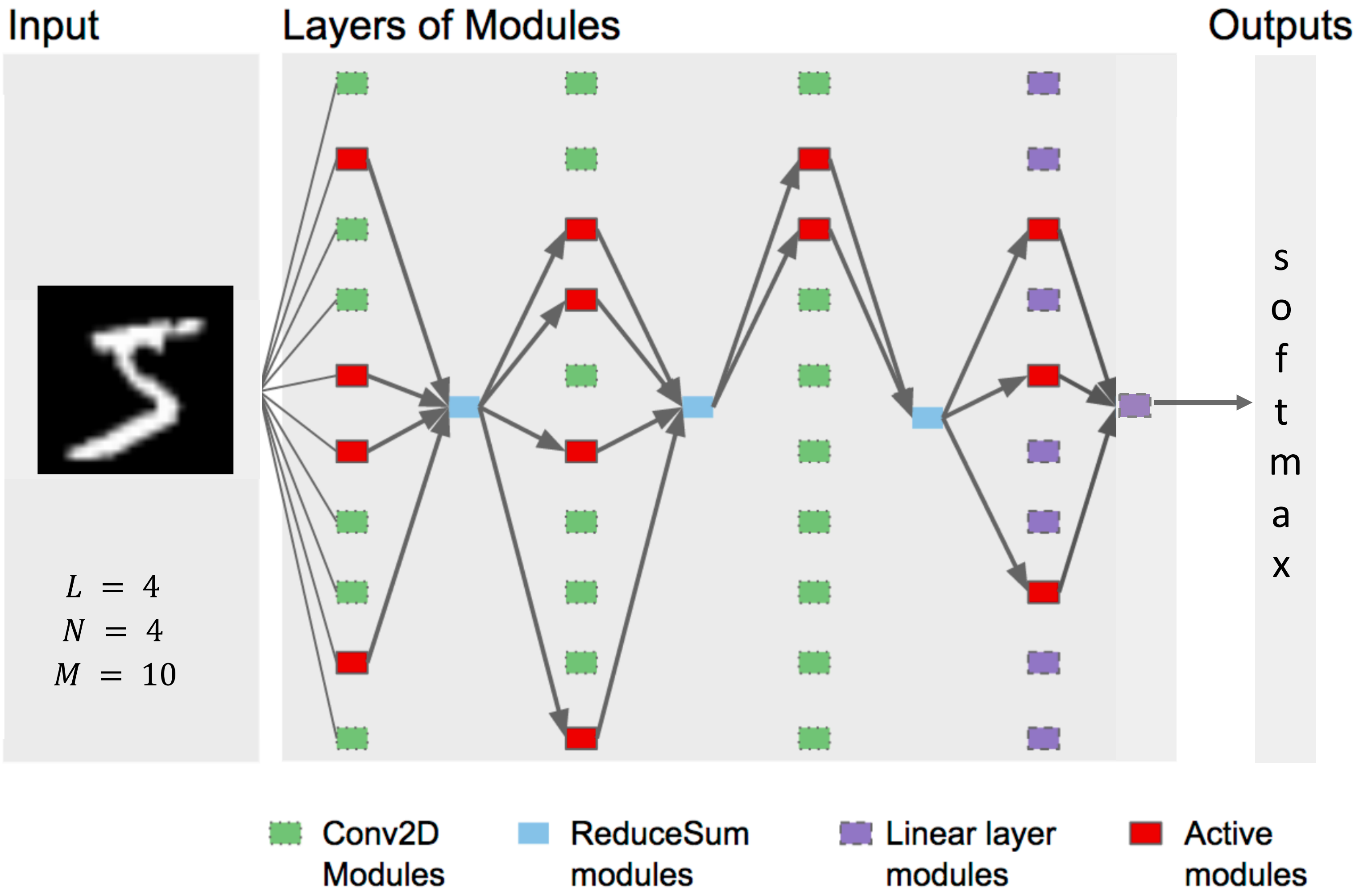


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose random genotype
- Train its pathway T epochs
- Evaluate fitness

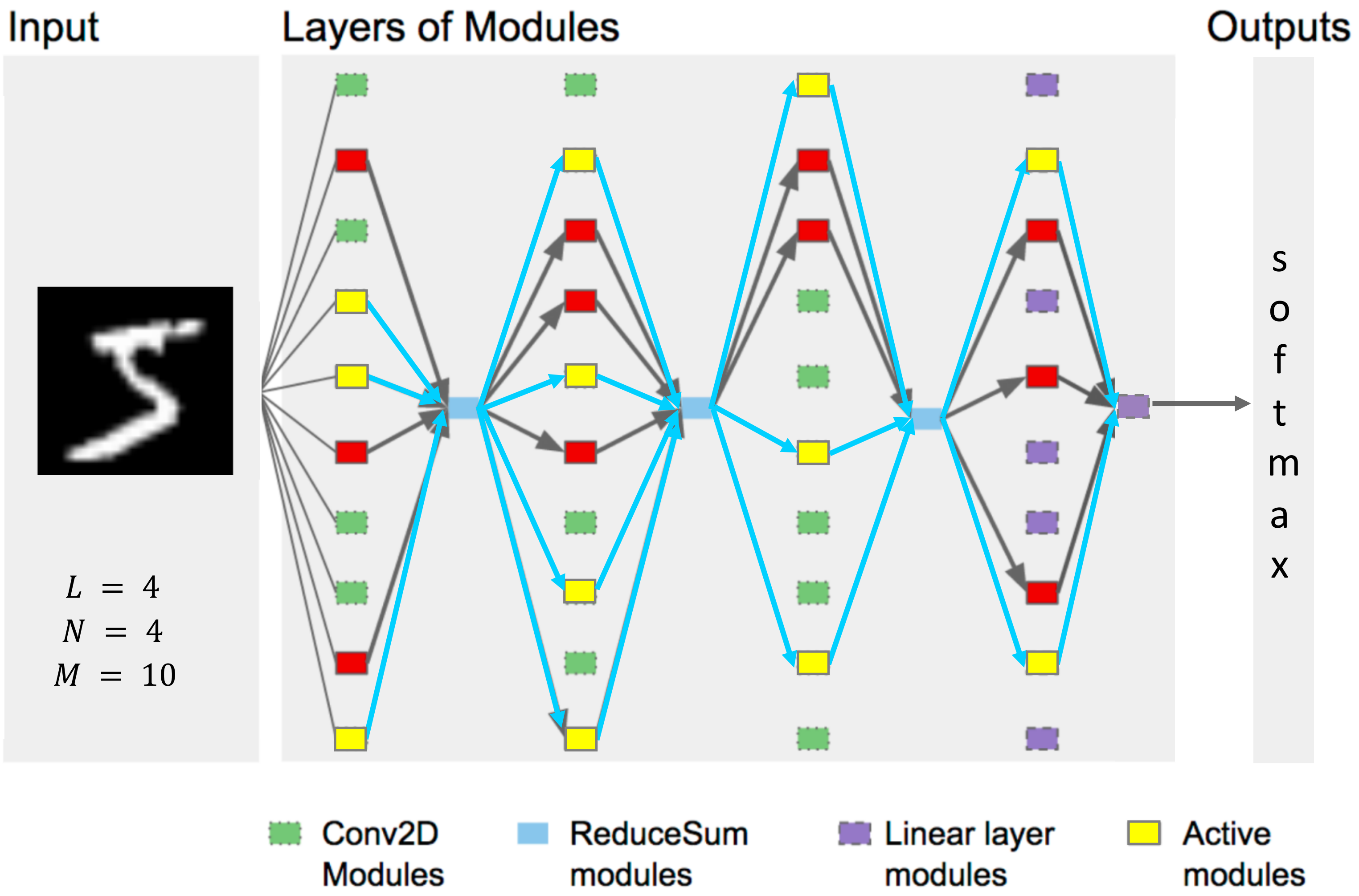


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose another random genotype
- Train its pathway for T epochs
- Binary tournament selection: keep copy of “winning” pathway
- Mutate: choose independently a module from each layer with probability $1/(NL)$

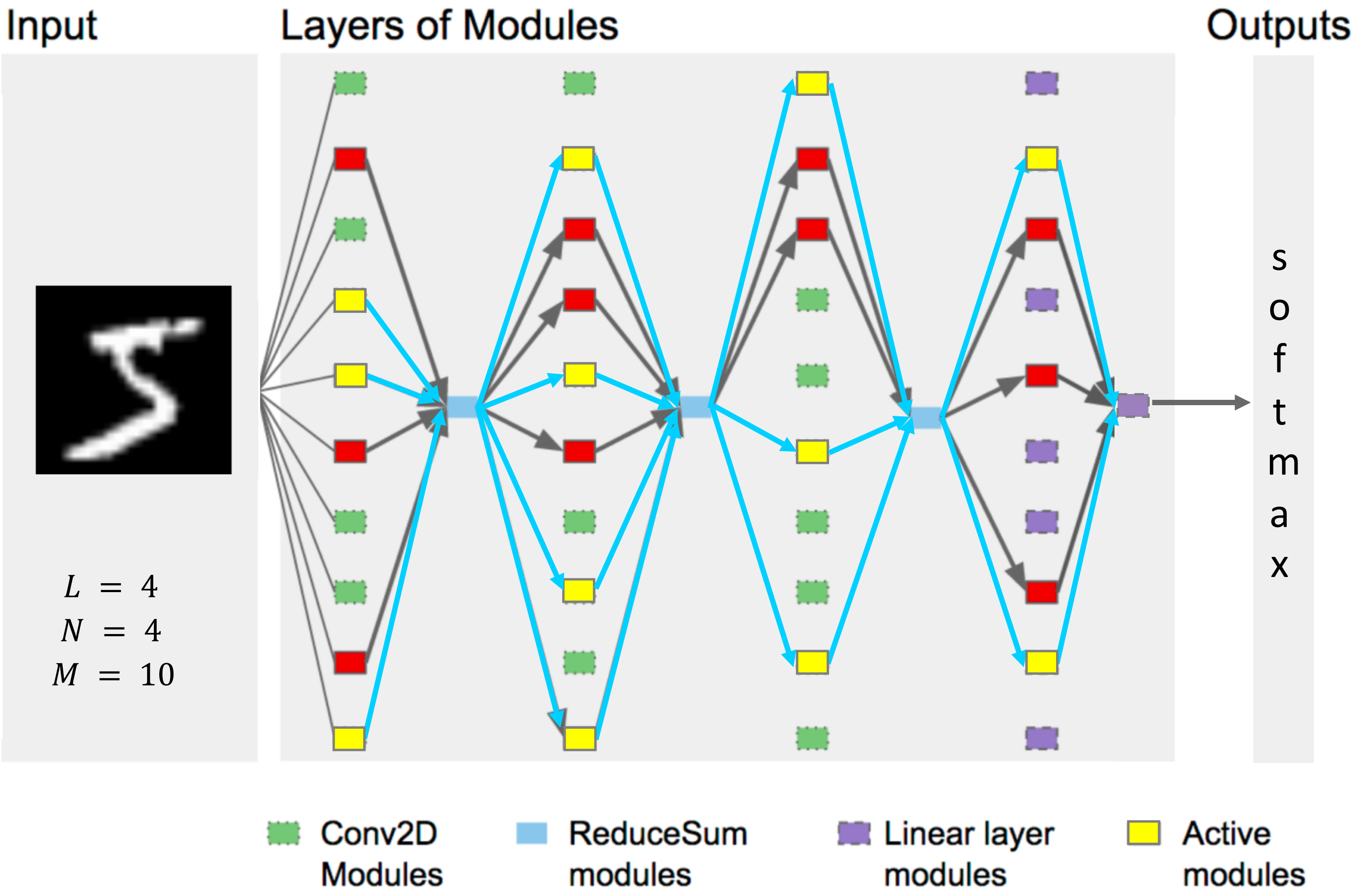


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose another random genotype
- Train its pathway for T epochs
- Binary tournament selection: keep copy of "winning" pathway
- Mutate: choose independently a module from each layer with probability $1/(NL)$

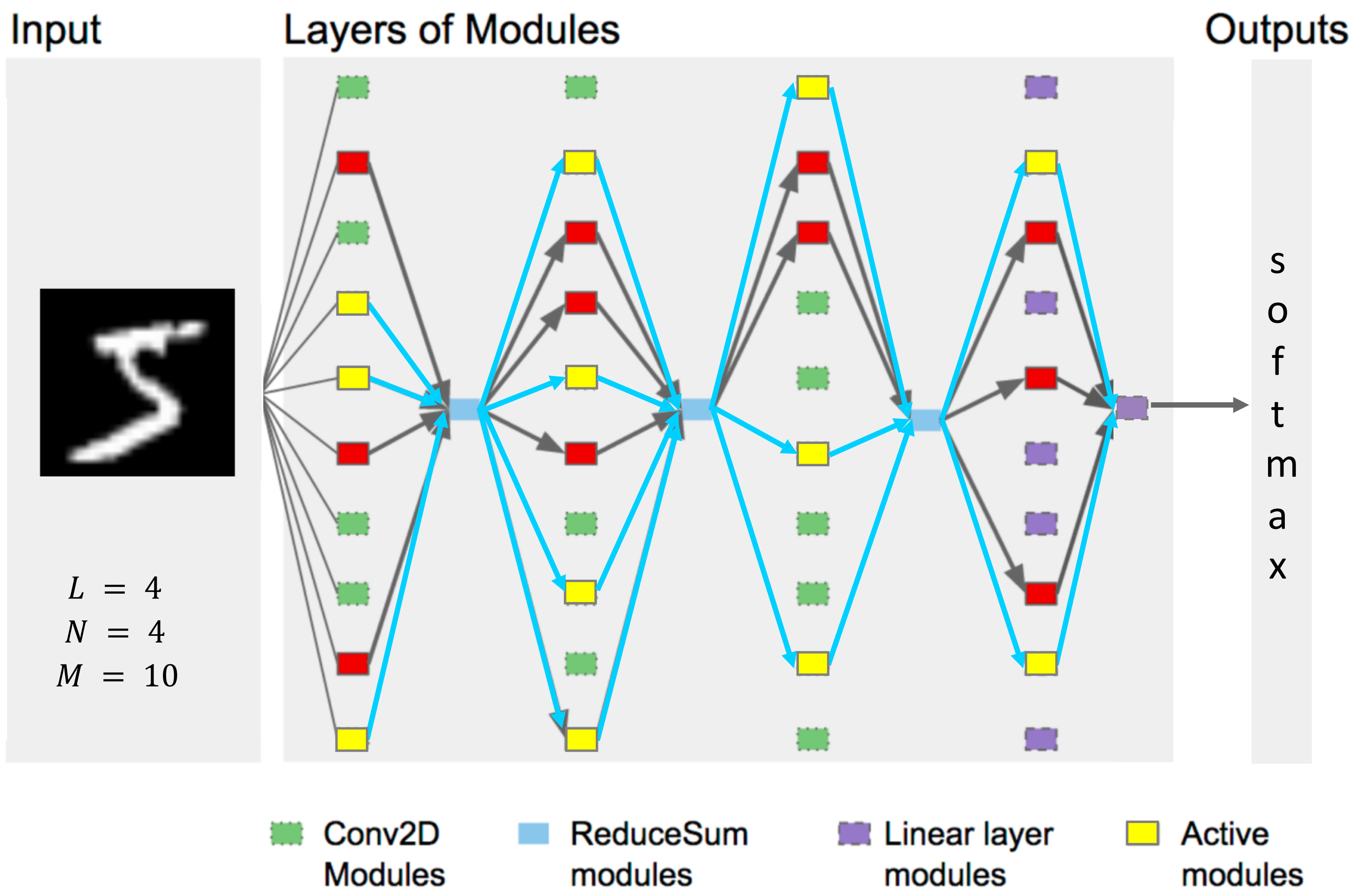


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose another random genotype
- Train its pathway for T epochs
- Binary tournament selection: keep copy of “winning” pathway
- Mutate: choose independently a module from each layer with probability $1/(NL)$

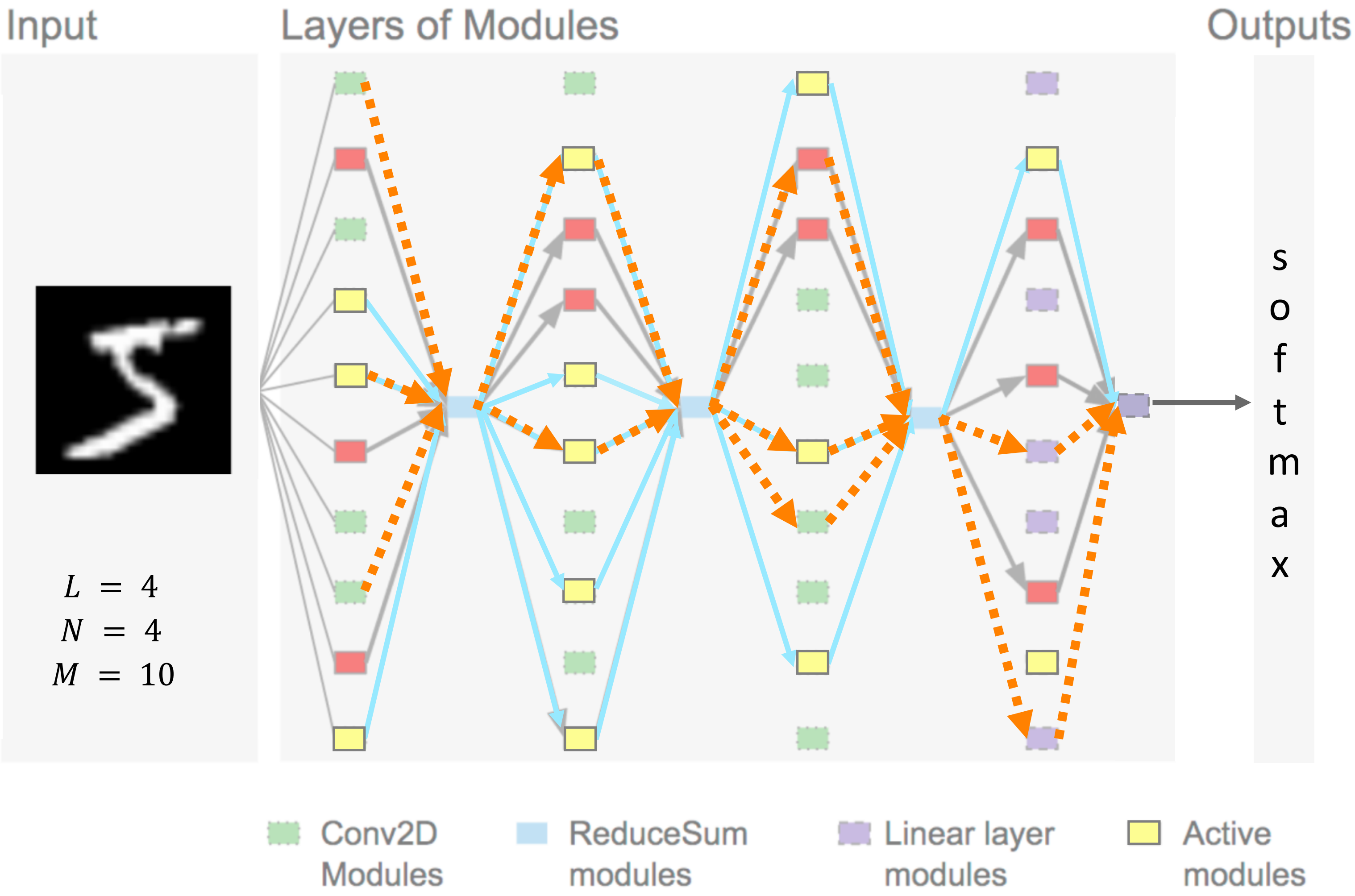


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose another random genotype
- Train its pathway for T epochs
- Binary tournament selection: keep copy of "winning" pathway
- Mutate: choose independently a module from each layer with probability $1/(NL)$

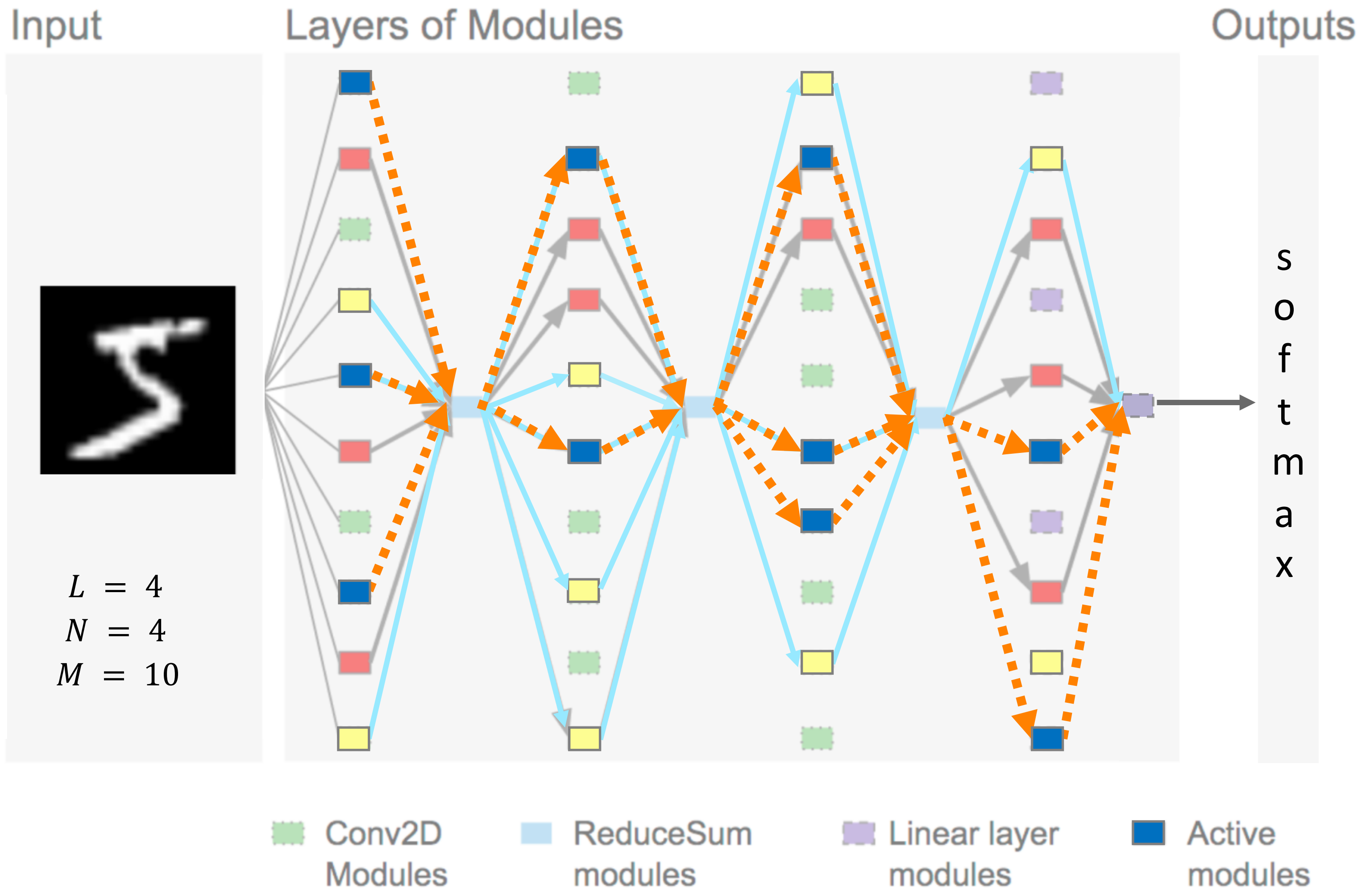


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Serial evolution

- Choose another random genotype
- Train its pathway for T epochs
- Binary tournament selection: keep copy of "winning" pathway
- Mutate: choose independently a module from each layer with probability $1/(NL)$

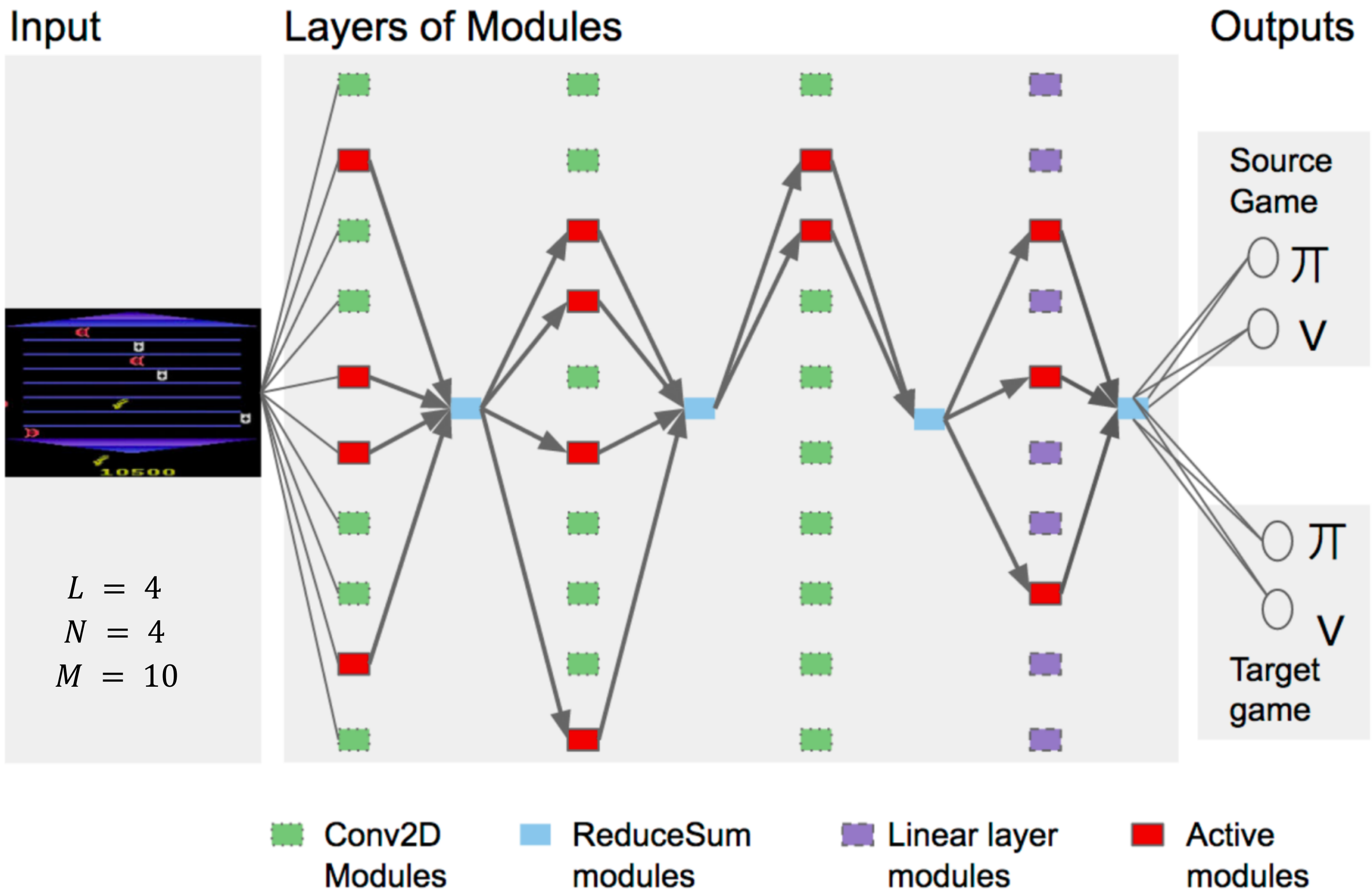


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Parallel evolution

- All pathways trained in parallel
- No simultaneous update of parameters
- Evaluate fitness for all pathways in parallel

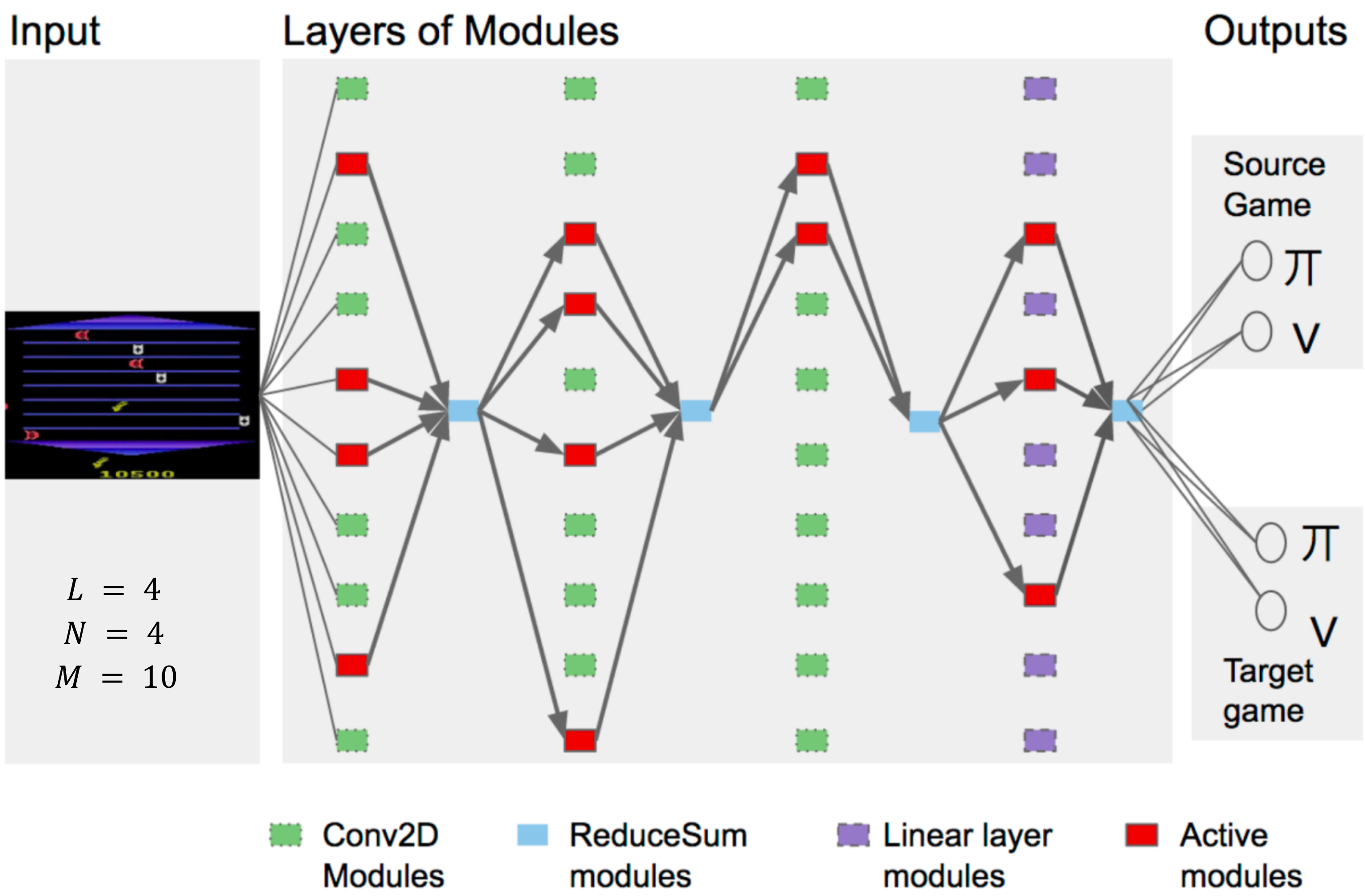


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Parallel evolution

- All pathways trained in parallel
- No simultaneous update of parameters
- Evaluate fitness for all pathways in parallel

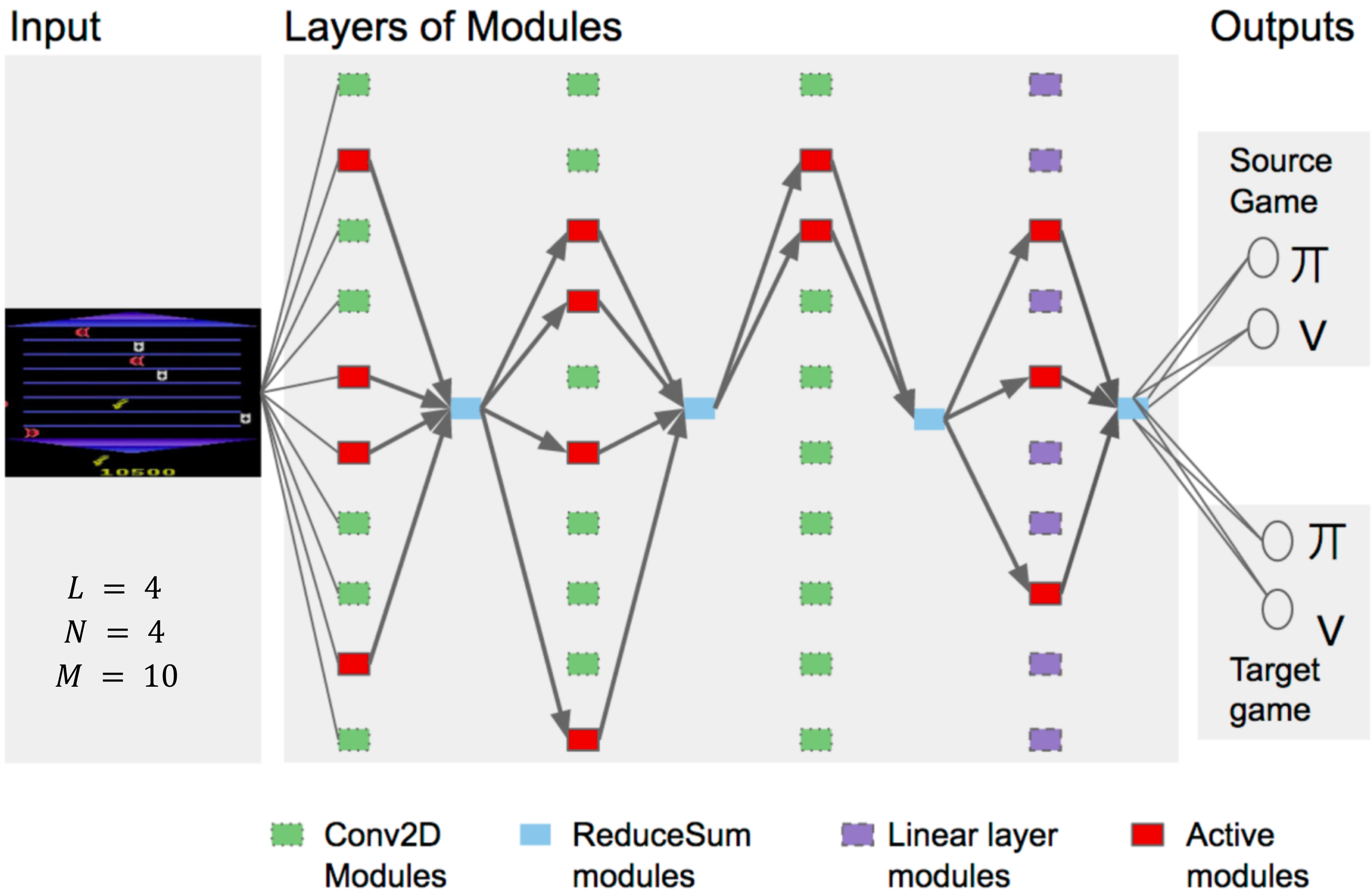


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Parallel evolution

- All pathways trained in parallel
- No simultaneous update of parameters
- Evaluate fitness for all pathways in parallel



C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Transfer learning paradigm

- Three experiments:
    1. learn task B from scratch using maximum-sized fixed pathway
    2. train a maximum-sized fixed pathway on task A, finetune on task B
    3. fix parameters of optimal pathway learned for task A, and re-evolve a new population of pathways on task B

C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Transfer learning paradigm

- Three experiments:
  I. learn task B from scratch using maximum-sized fixed pathway
  II. train a maximum-sized fixed pathway on task A, finetune on task B
  III. fix parameters of optimal pathway learned for task A, and re-evolve a new population of pathways on task B

C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Transfer learning paradigm

- Three experiments:
    1. learn task B from scratch using maximum-sized fixed pathway
    2. train a maximum-sized fixed pathway on task A, finetune on task B
    3. fix parameters of optimal pathway learned for task A, re-evolve a new population of pathways on task B (PathNet)

C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Transfer learning paradigm

- Three experiments:
    1. learn task B from scratch using maximum-sized fixed pathway
    2. train a maximum-sized fixed pathway on task A, finetune on task B
    3. **fix parameters of optimal pathway learned for task A, re-evolve a new population of pathways on task B (PathNet)**

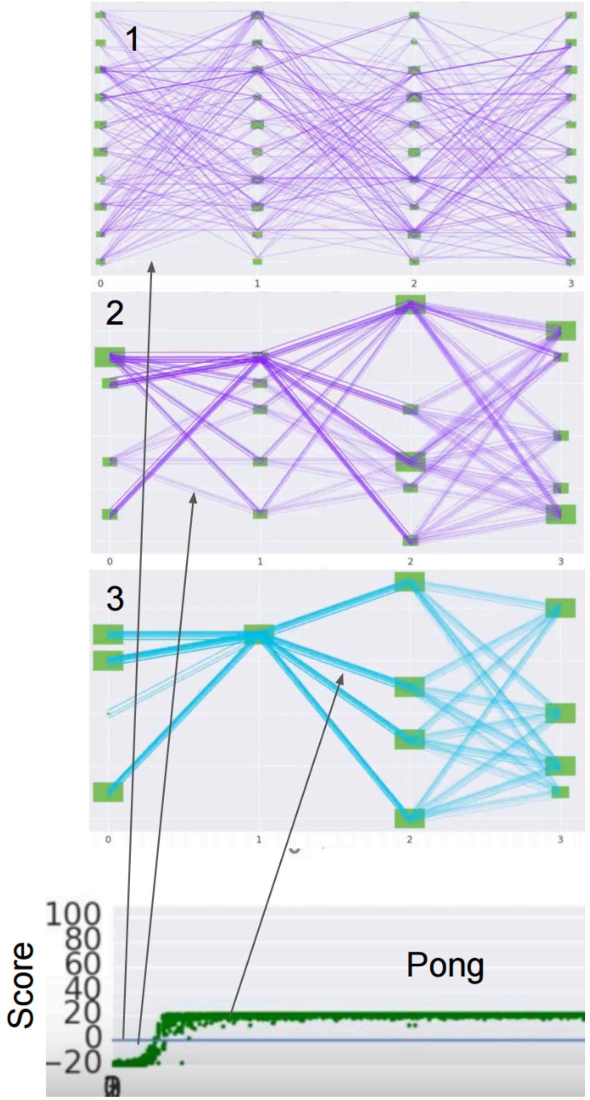
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Transfer learning paradigm

Task A: Pong

Task B: Alien



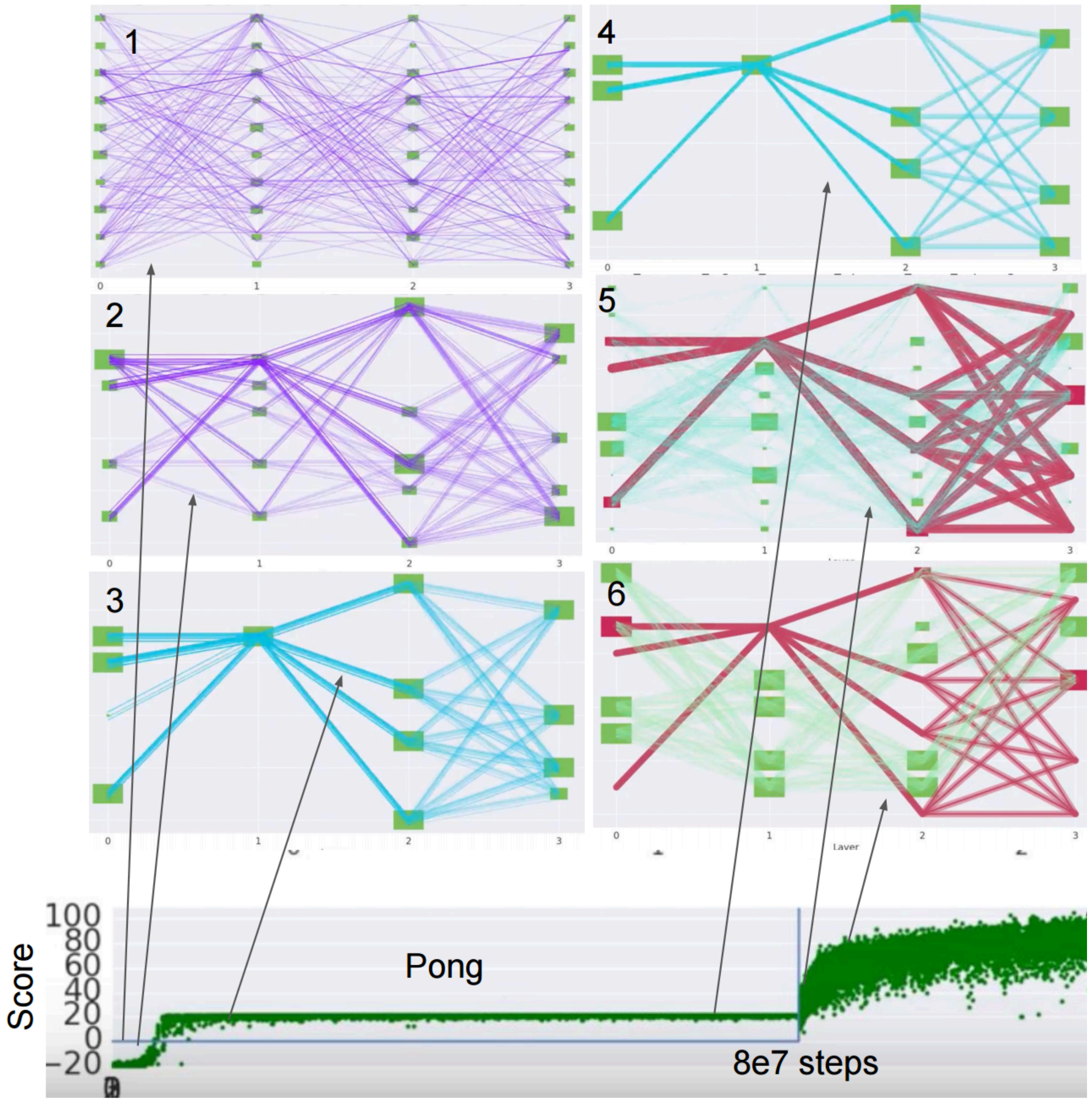
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Transfer learning paradigm

Task A: Pong

Task B: Alien



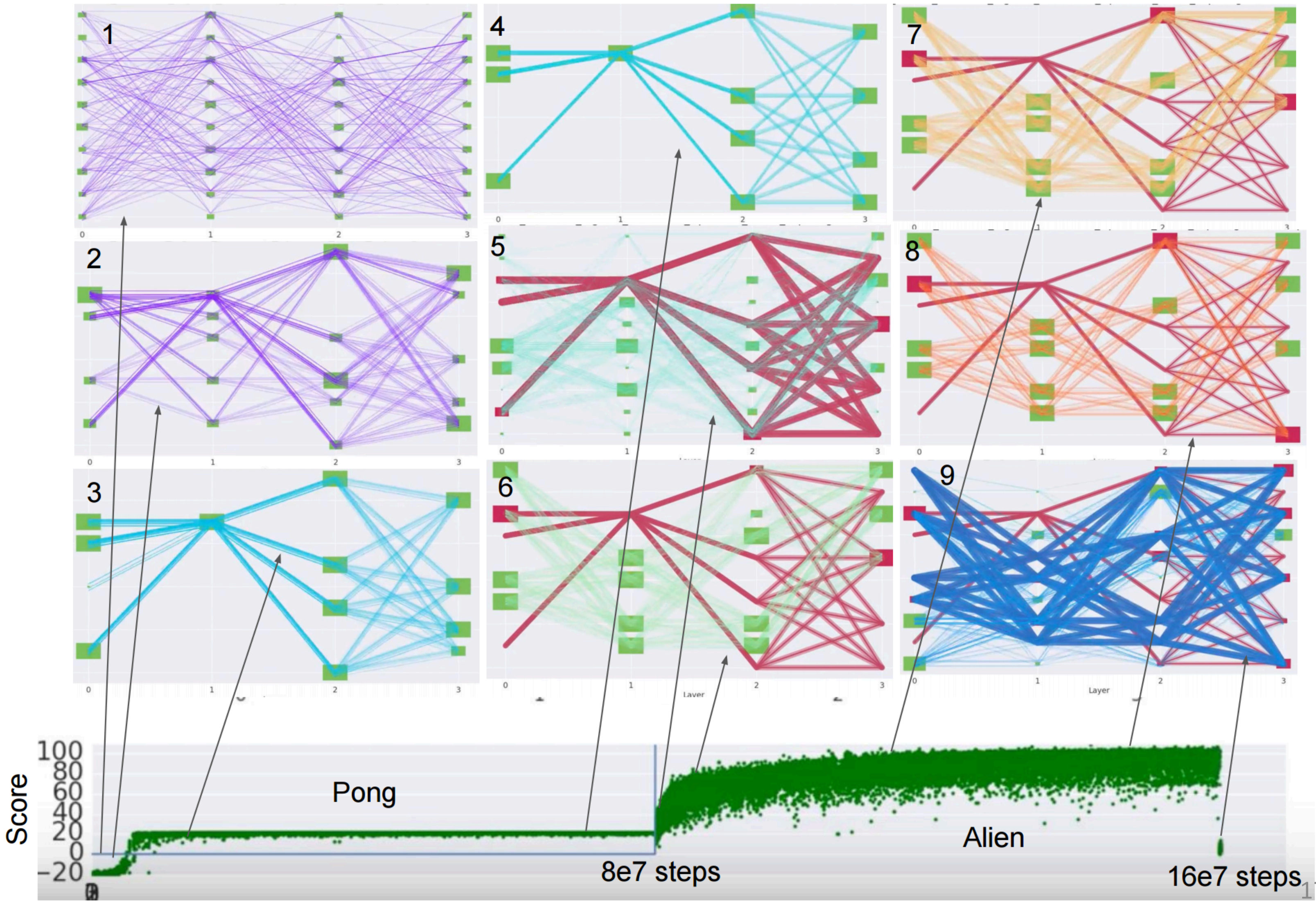
C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Transfer learning paradigm

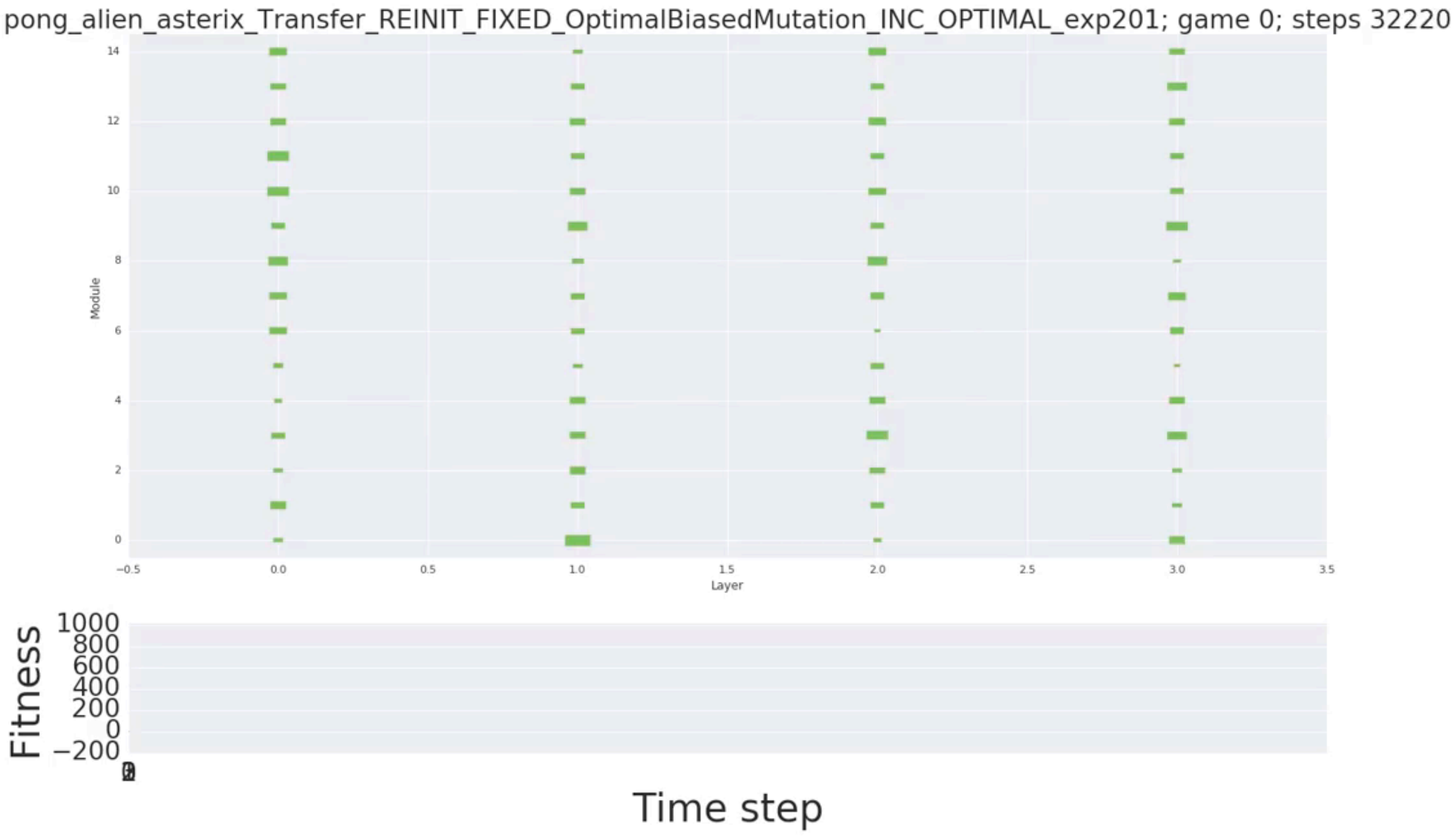
Task A: Pong

Task B: Alien



C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

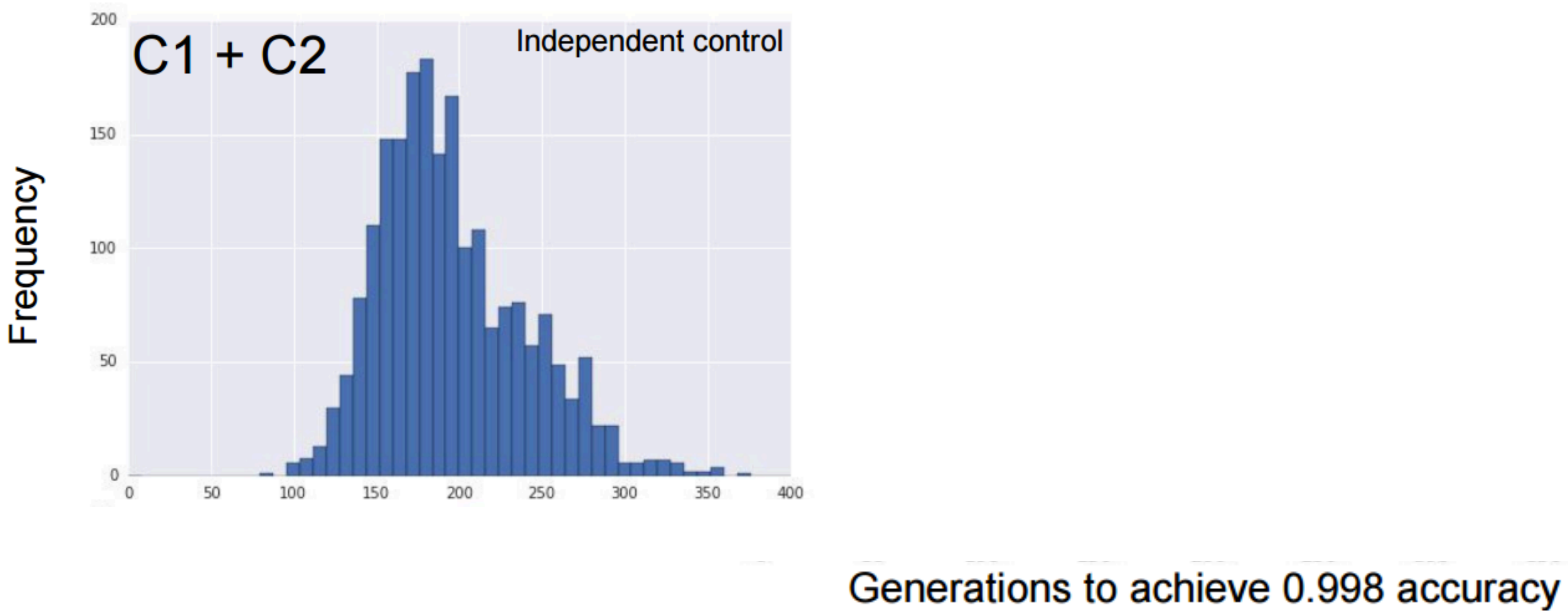


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Binary MNIST classification

Task A: 5 vs. 6
Task B: 0 vs 9

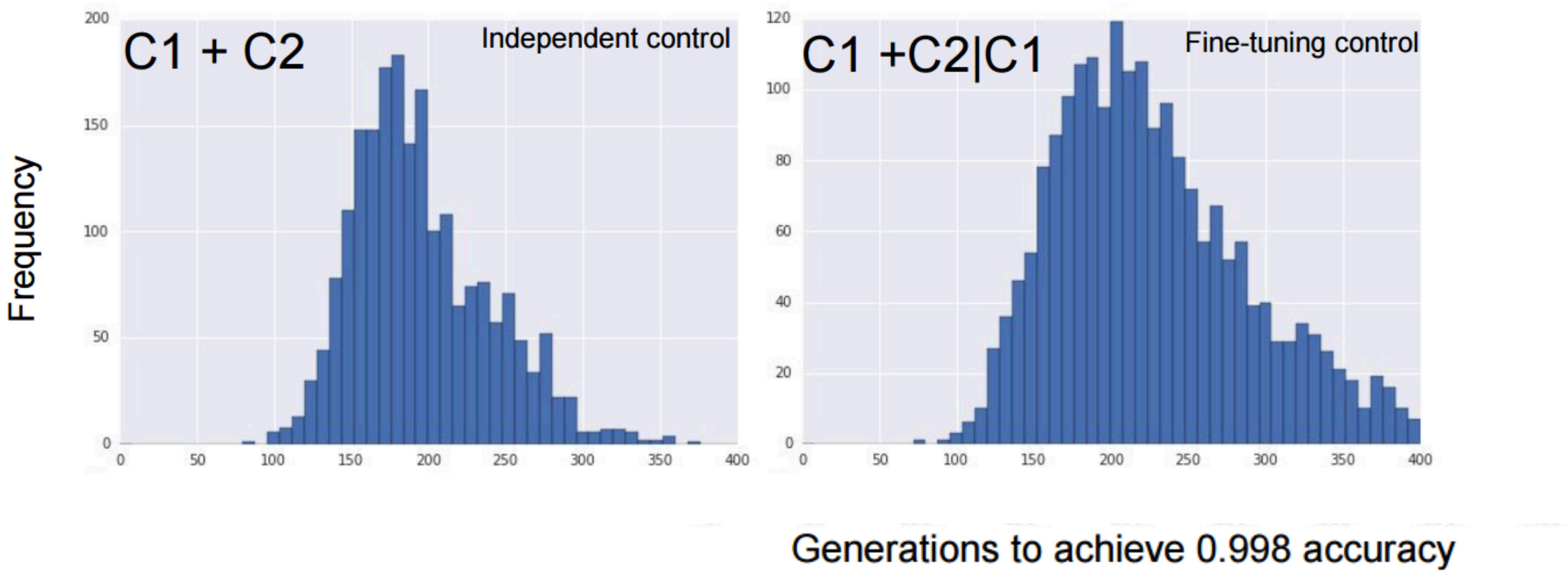


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Binary MNIST classification

Task A: 5 vs. 6
Task B: 0 vs 9

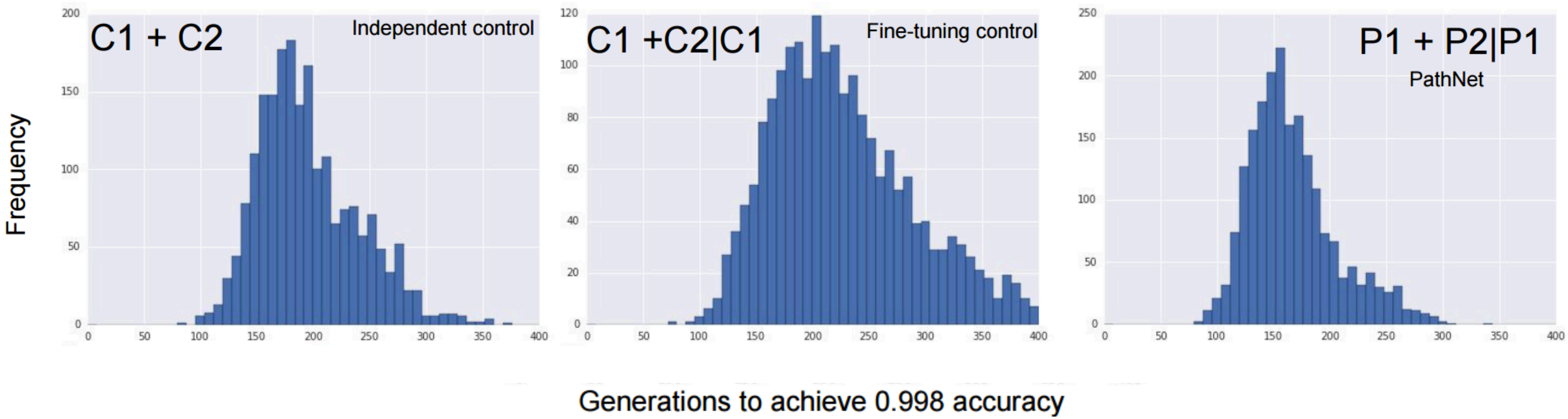


C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Binary MNIST classification

Task A: 5 vs. 6
Task B: 0 vs 9



C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Case Study: PathNet

Atari games

C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017).

Neuroevolution: Recap

- Motivation from natural evolution
- Types of “genotype” representation
- Mutation strategies
- Fitness evaluation
- Learned architectures

Neuroevolution: Recap

- Motivation from natural evolution
- Types of “genotype” representation
- Mutation strategies
- Fitness evaluation
- Learned architectures

Neuroevolution: Recap

- Motivation from natural evolution
- Types of “genotype” representation
- Mutation strategies
- Fitness evaluation
- Learned architectures

Neuroevolution: Recap

- Motivation from natural evolution
- Types of “genotype” representation
- Mutation strategies
- Fitness evaluation
- Learned architectures

Neuroevolution: Recap

- Motivation from natural evolution
- Types of “genotype” representation
- Mutation strategies
- Fitness evaluation
- Learned architectures

Neuroevolution: Recap

- Motivation from natural evolution
- Types of “genotype” representation
- Mutation strategies
- Fitness evaluation
- Learned architectures

Neuroevolution: Pros and Cons

- ✔ Parallelizable
- ✔ Intuitive
- ✔ “Survival of the fittest” ensures solutions only get better

Neuroevolution: Pros and Cons

- ✔ Parallelizable
- ✔ Intuitive
- ✔ “Survival of the fittest” ensures solutions only get better

- ✘ Very slow
- ✘ Vast search space
- ✘ Mutation strategies still too constrained
- ✘ Fitness evaluation hazy

So, is it all worth it?

Study	Params.	C10+	C100+
Maxout (Goodfellow et al., 2013)	–	90.7%	61.4%
Network in Network (Lin et al., 2013)	–	91.2%	–
All-CNN (Springenberg et al., 2014)	1.3 M	92.8%	66.3%
Deeply Supervised (Lee et al., 2015)	–	92.0%	65.4%
Highway (Srivastava et al., 2015)	2.3 M	92.3%	67.6%
ResNet (He et al., 2016)	1.7 M	93.4%	72.8%†
Evolution	5.4 M	94.6%	
	40.4 M		76.3%
Wide ResNet 28-10 (Zagoruyko & Komodakis, 2016)	36.5 M	96.0%	80.0%
Wide ResNet 40-10+d/o (Zagoruyko & Komodakis, 2016)	50.7 M	96.2%	81.7%
DenseNet (Huang et al., 2016)	25.6 M	96.7%	82.8%

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-Scale Evolution of Image Classifiers. arXiv preprint arXiv:1703.01041 (2017).

Index

- Formal Definition of Meta-Learning
- Learning the Deep learning Architecture
- **Learning to Explore**
- Learning to Seek Knowledge
- Learning to communicate

Index

- Formal Definition of Meta-Learning
- Learning the Deep learning Architecture
- **Learning to Explore**
  - Learning to Optimize
  - Learning to Explore An Environment
- Learning to Seek Knowledge
- Learning to communicate

Index

- Formal Definition of Meta-Learning
- Learning the Deep learning Architecture
- **Learning to Explore**
  - **Learning to Optimize**
  - Learning to Explore An Environment
- Learning to Seek Knowledge
- Learning to communicate

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent

$$f(\theta) \qquad \theta \in \Theta$$

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent

$$f(\theta) \qquad \theta \in \Theta$$

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent

$$f(\theta) \qquad \theta \in \Theta$$

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent

$$f(\theta) \qquad \theta \in \Theta$$

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent

$$f(\theta) \qquad \theta \in \Theta$$

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

SGD	RMSProp
AdaGrad	AdaDelta
Adam	L-BFGS

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent	Proposed method
$f(\theta) \quad \theta \in \Theta$ $\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$ $\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$	$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi)$

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent	Proposed method

$$f(\theta) \quad \theta \in \Theta$$

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi)$$

learned
update rule

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent	Proposed method

$$f(\theta) \quad \theta \in \Theta$$

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi)$$

learned
update rule

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Gradient descent	Proposed method

$$f(\theta) \qquad \theta \in \Theta$$

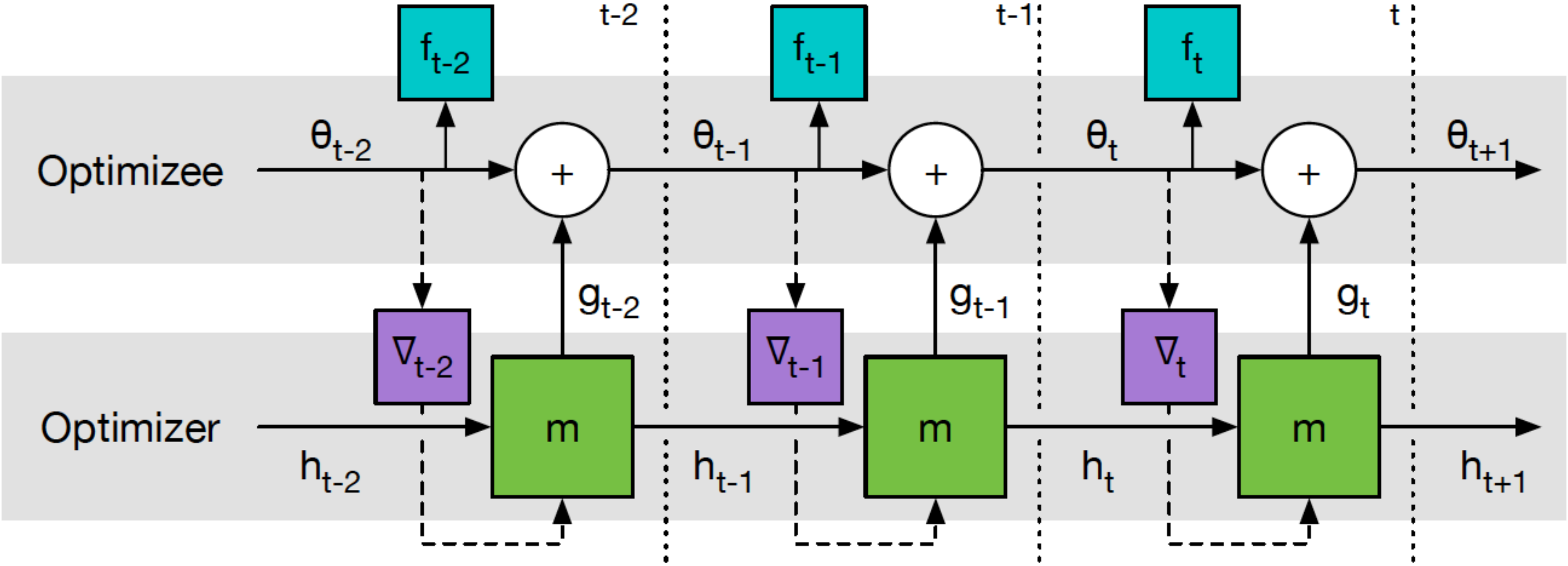
$$\theta^* = \arg\min_{\theta\in\Theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

$$\theta_{t+1} = \theta_t + \underbrace{g_t}_{\text{learned update rule}}(\nabla f(\theta_t), \underbrace{\phi}_{\text{optimizer parameters}})$$

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

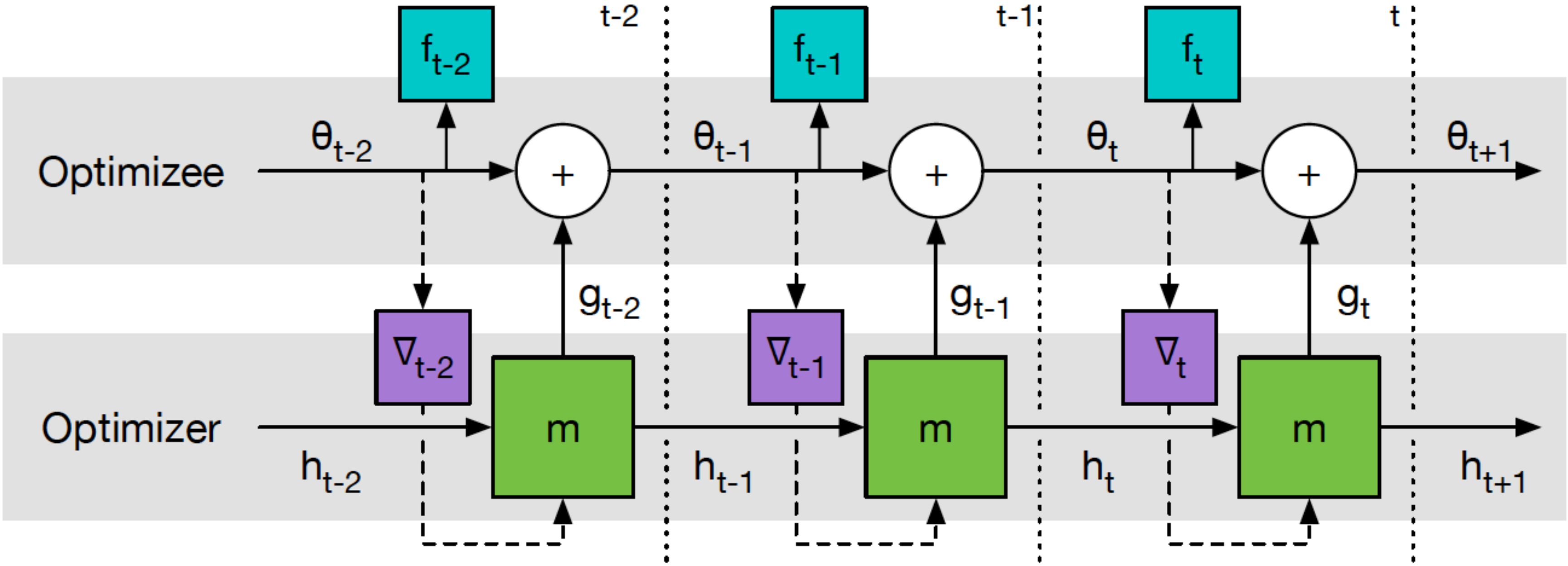


Computational graph guiding gradient flow

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\boxed{\theta_{t+1}} = \theta_t + g_t$$

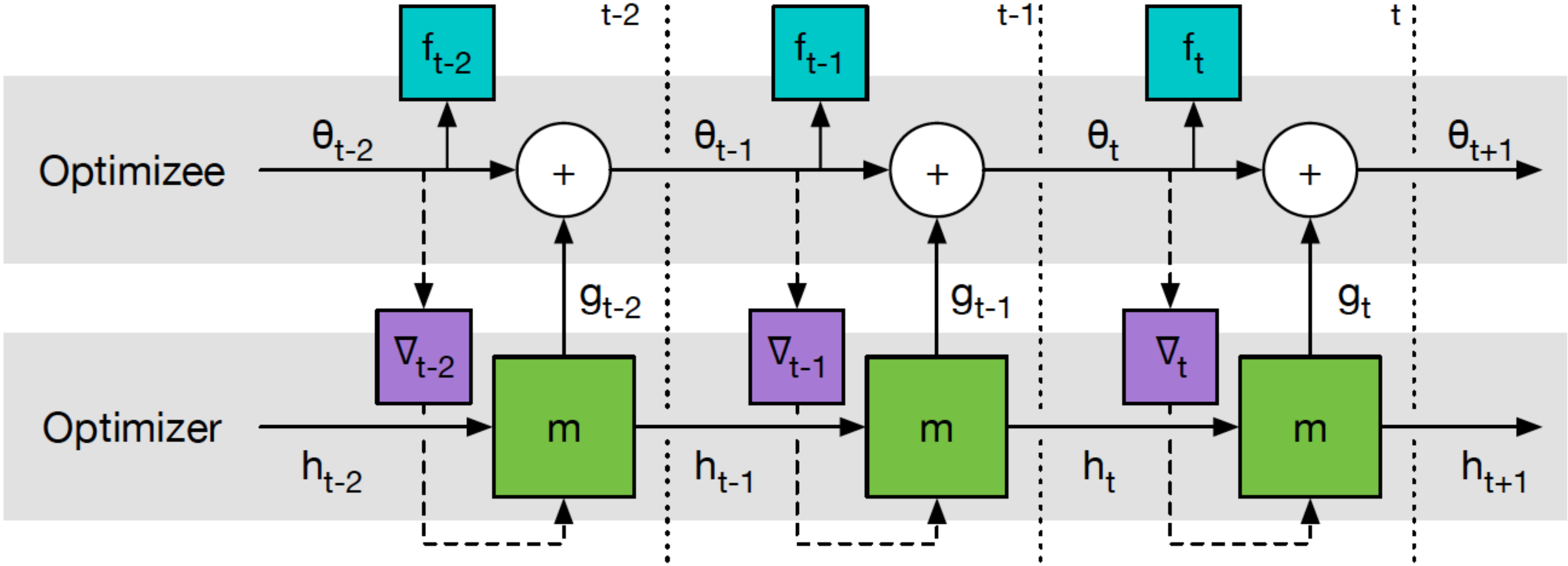


Computational graph guiding gradient flow

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\theta_{t+1} = \boxed{\theta_t} + g_t$$

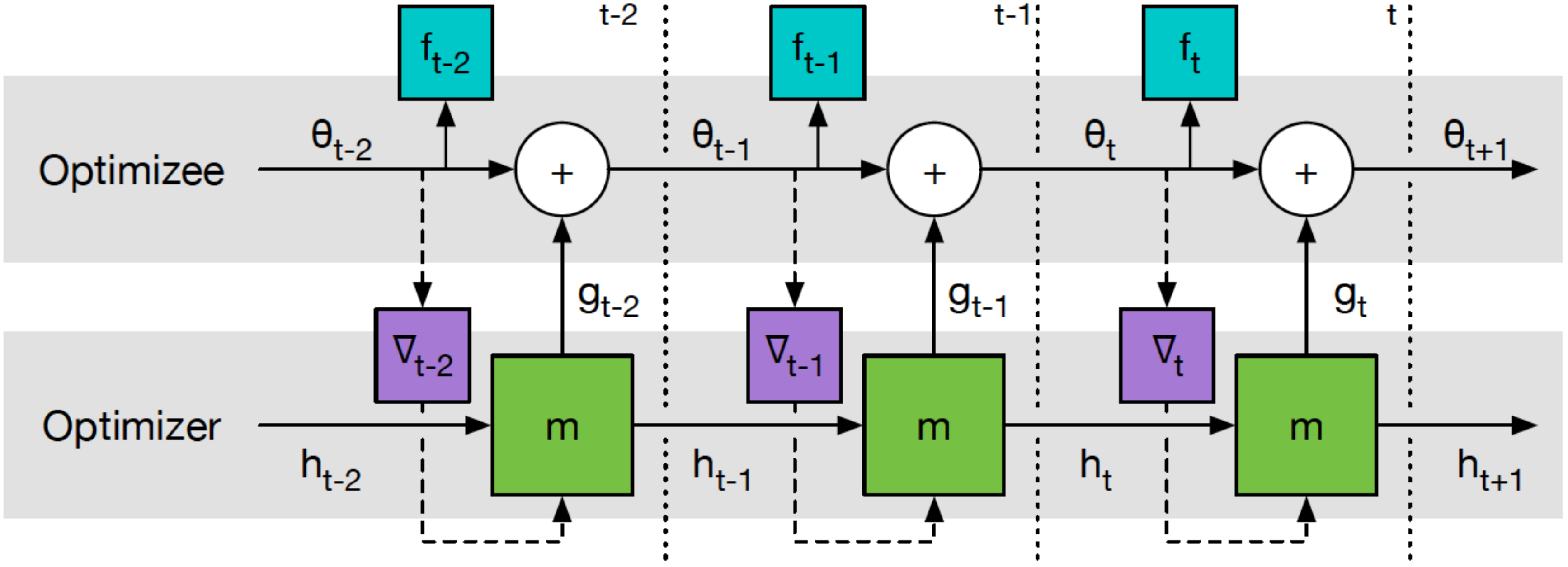


Computational graph guiding gradient flow

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\theta_{t+1} = \theta_t + \boxed{g_t}$$



Computational graph guiding gradient flow

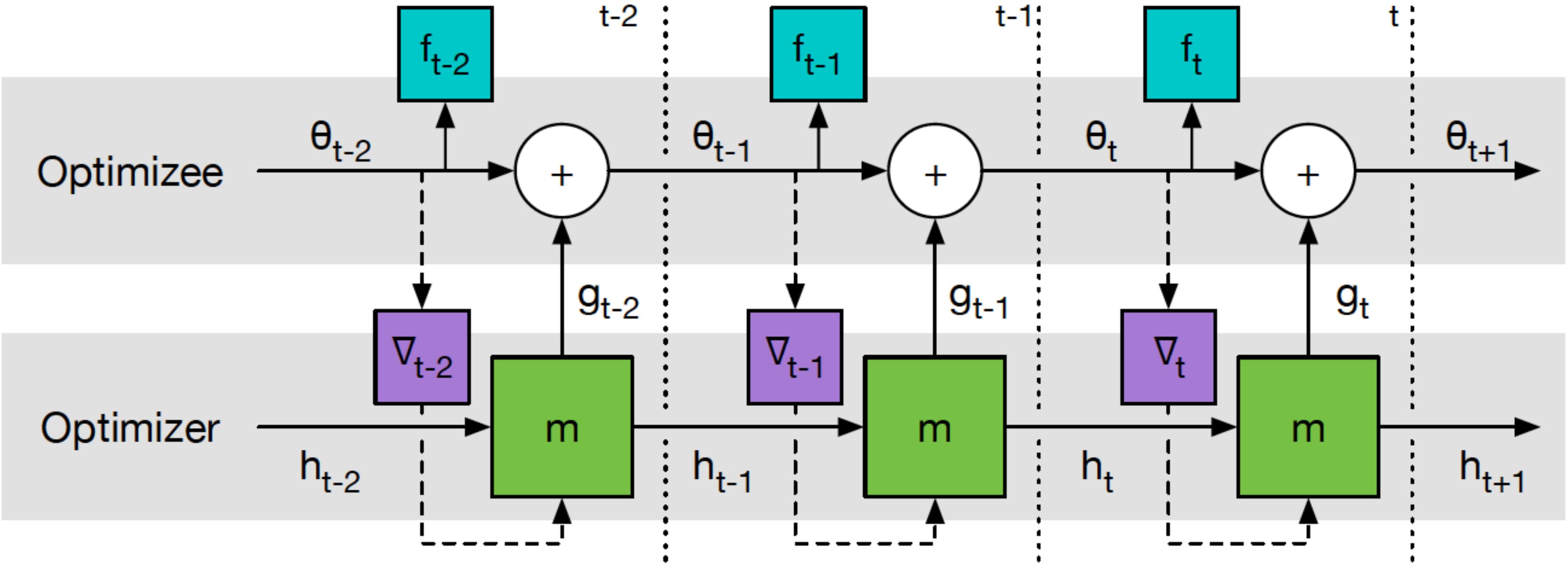
M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\theta_{t+1} = \theta_t + g_t\,,$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

$$\nabla_t = \nabla_\theta f(\theta_t)$$



Computational graph guiding gradient flow

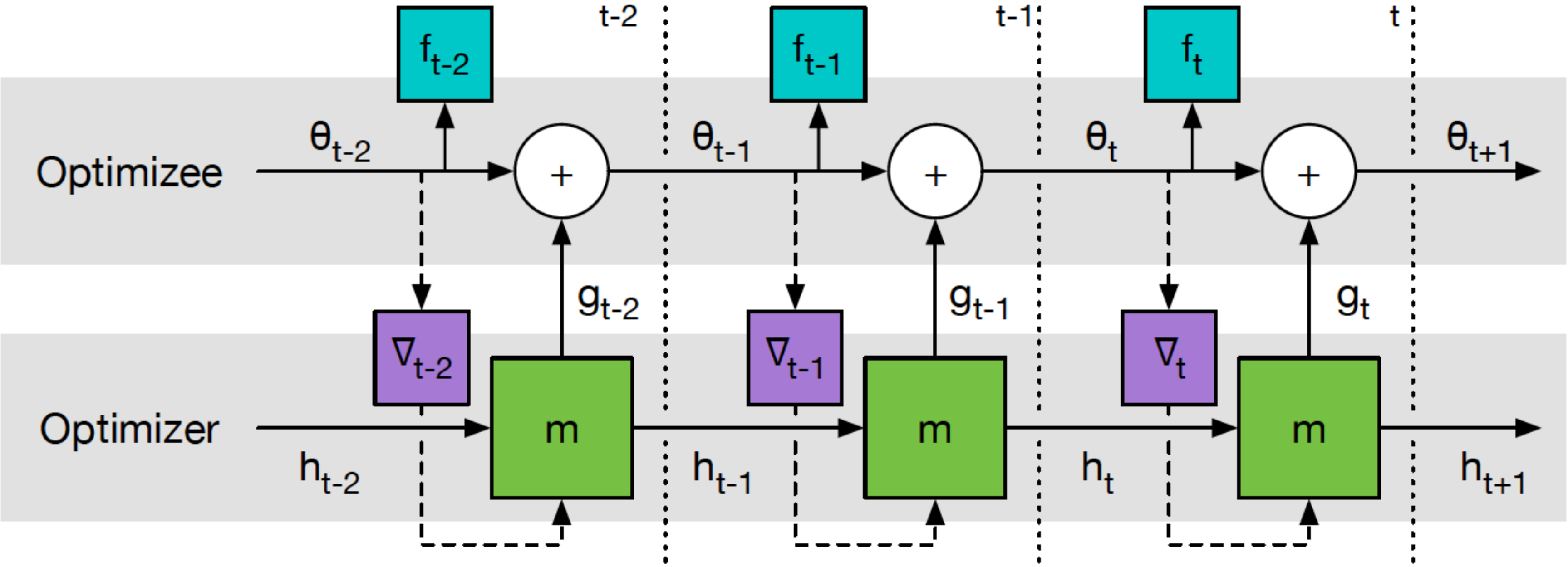
M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\theta_{t+1} = \theta_t + g_t\,,$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

$$\nabla_t = \nabla_\theta f(\theta_t)$$



Computational graph guiding gradient flow

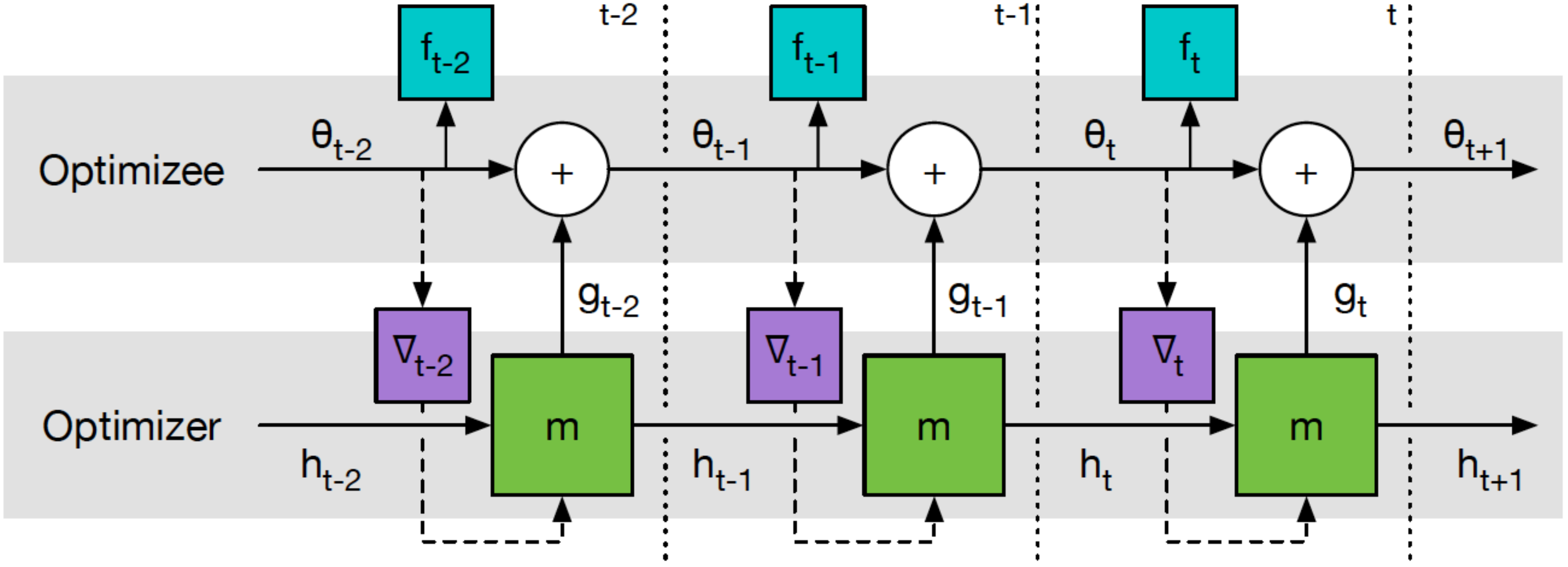
M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\theta_{t+1} = \theta_t + g_t \,,$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

$$\nabla_t = \nabla_\theta f(\theta_t)$$



Computational graph guiding gradient flow

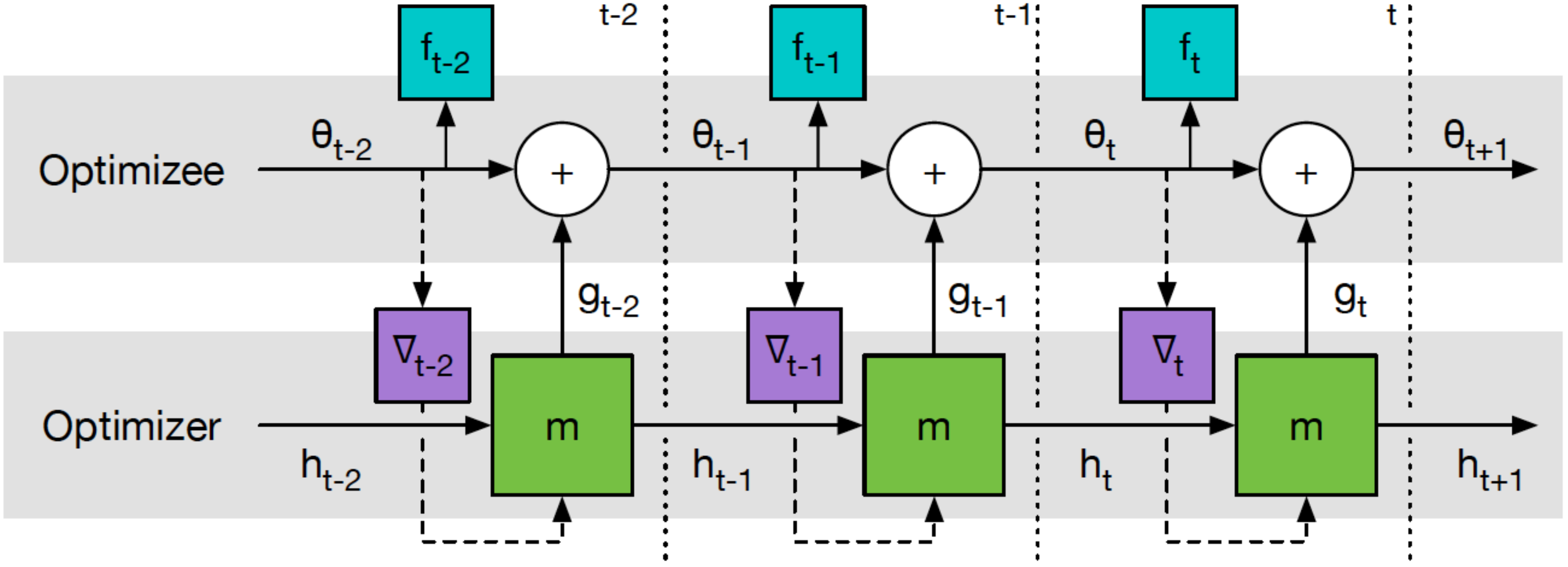
M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\theta_{t+1} = \theta_t + g_t \,,$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

$$\nabla_t = \nabla_\theta f(\theta_t)$$



Computational graph guiding gradient flow

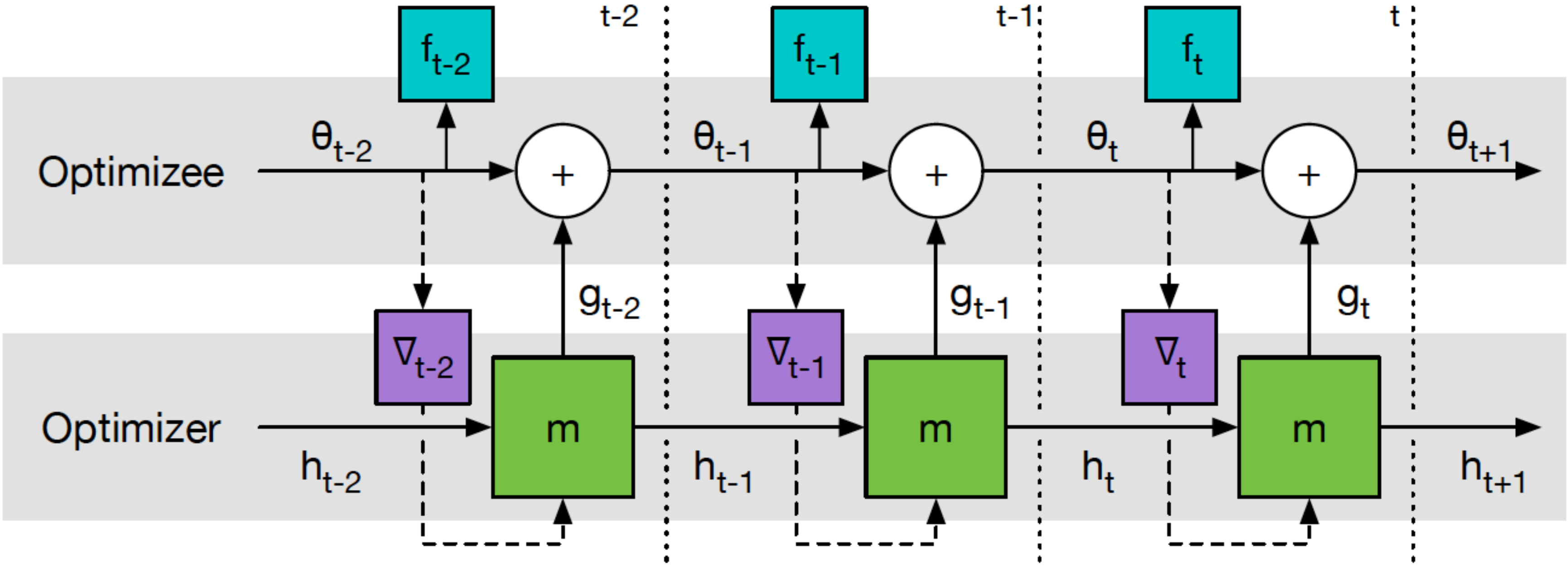
M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\theta_{t+1} = \theta_t + g_t \,,$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

$$\nabla_t = \nabla_\theta f(\theta_t)$$



Computational graph guiding gradient flow

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

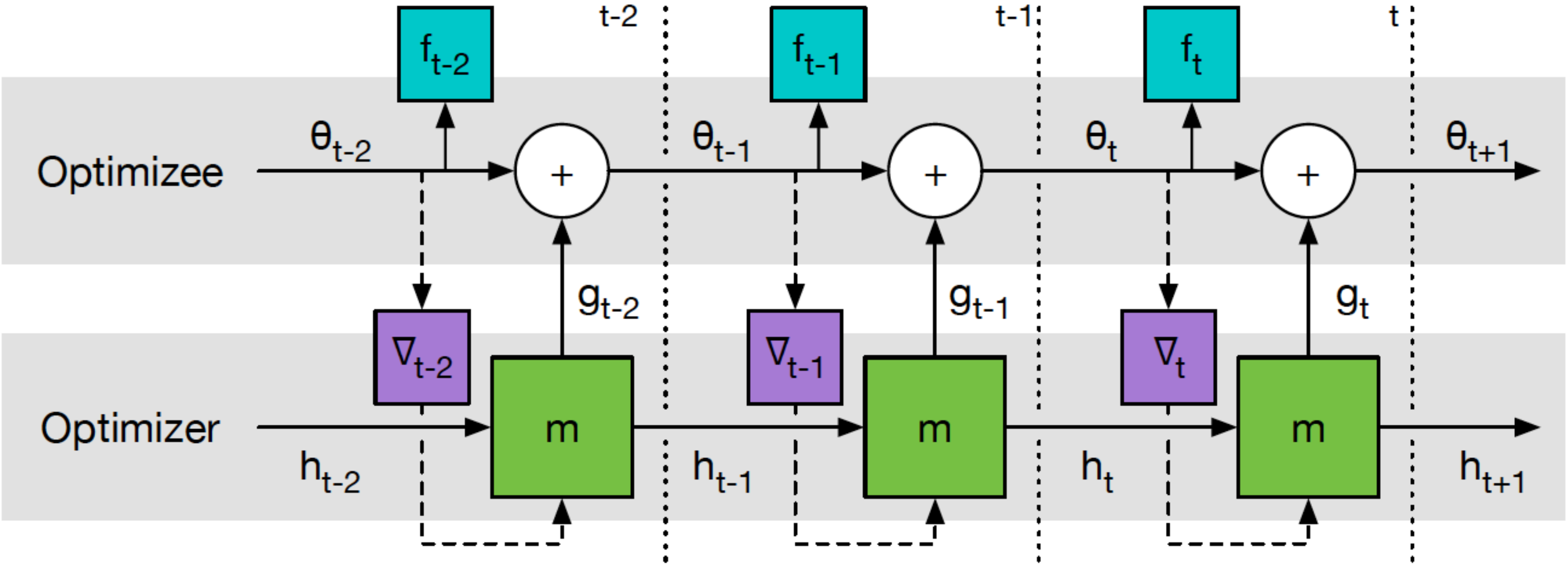
$$\theta_{t+1} = \theta_t + g_t \,,$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

$$\nabla_t = \nabla_\theta f(\theta_t)$$

$$\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{t=1}^{T} w_t f(\theta_t) \right]$$

$$w_t \in \mathbb{R}_{\geq 0}$$



Computational graph guiding gradient flow

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

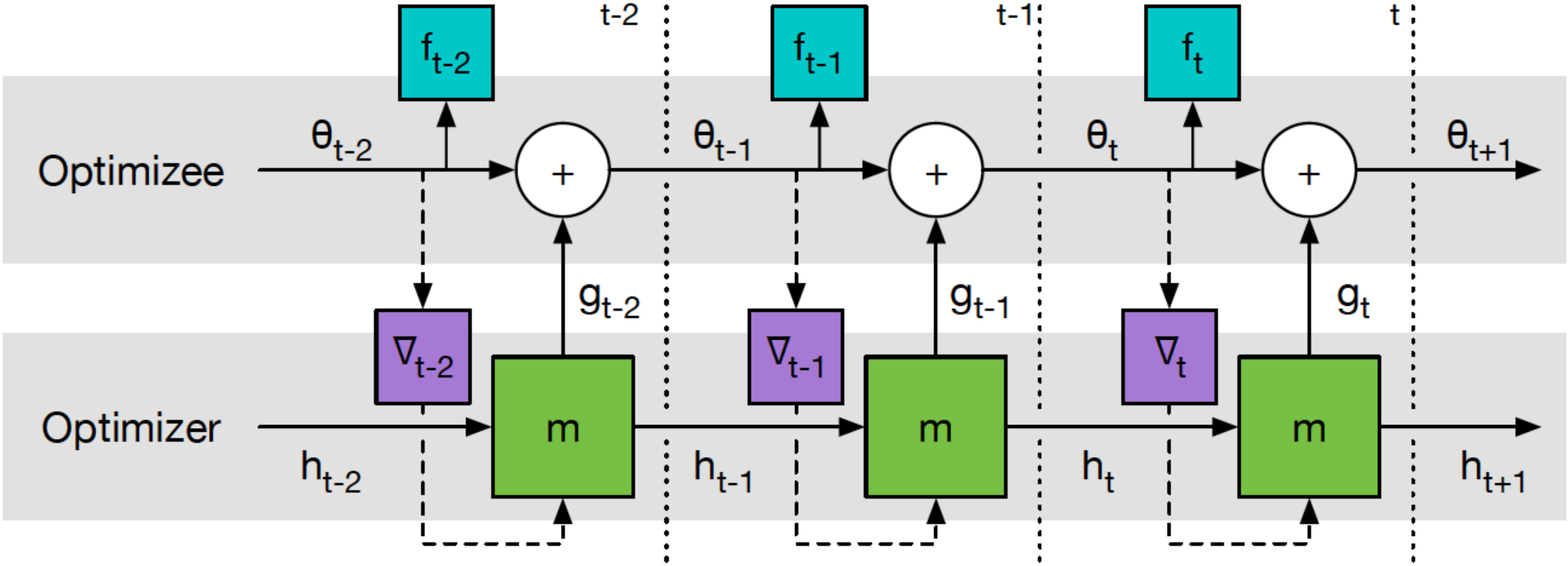
$$\theta_{t+1} = \theta_t + g_t\,,$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$$

$$\nabla_t = \nabla_\theta f(\theta_t)$$

$$\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{t=1}^{T} w_t f(\theta_t) \right]$$

$$w_t \in \mathbb{R}_{\geq 0}$$

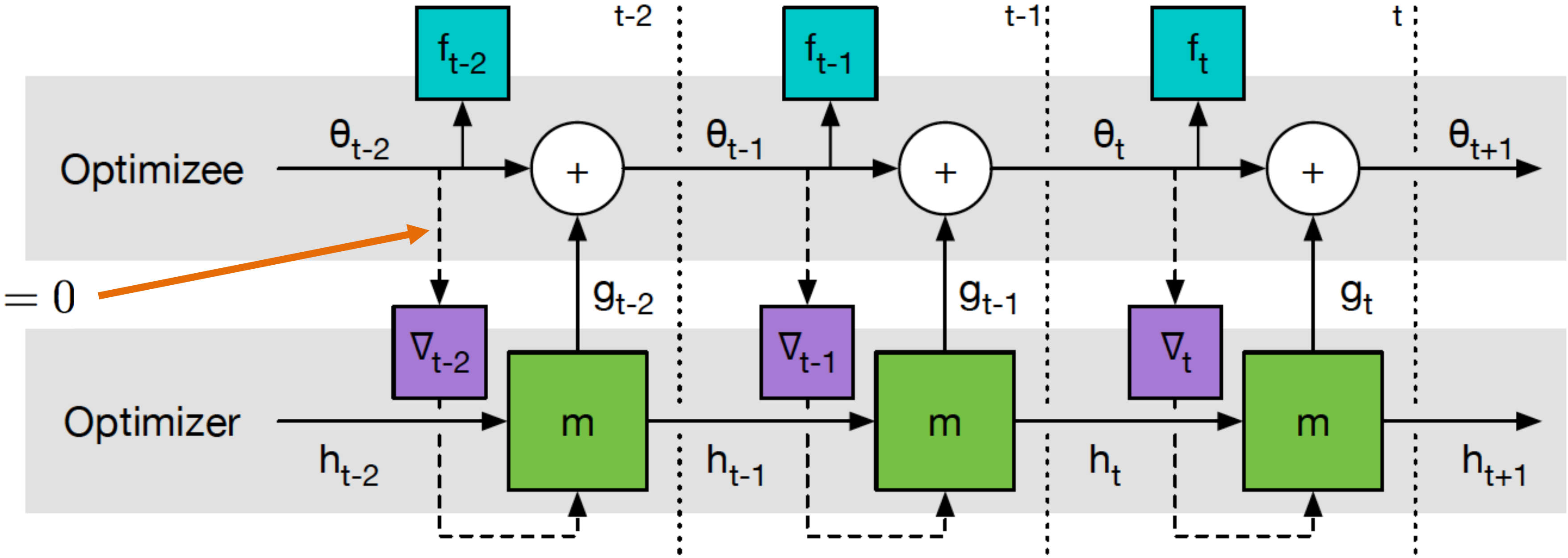


Computational graph guiding gradient flow

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

$$\nabla_t = \nabla_\theta f(\theta_t)\,, \qquad \partial \nabla_t / \partial \phi = 0$$



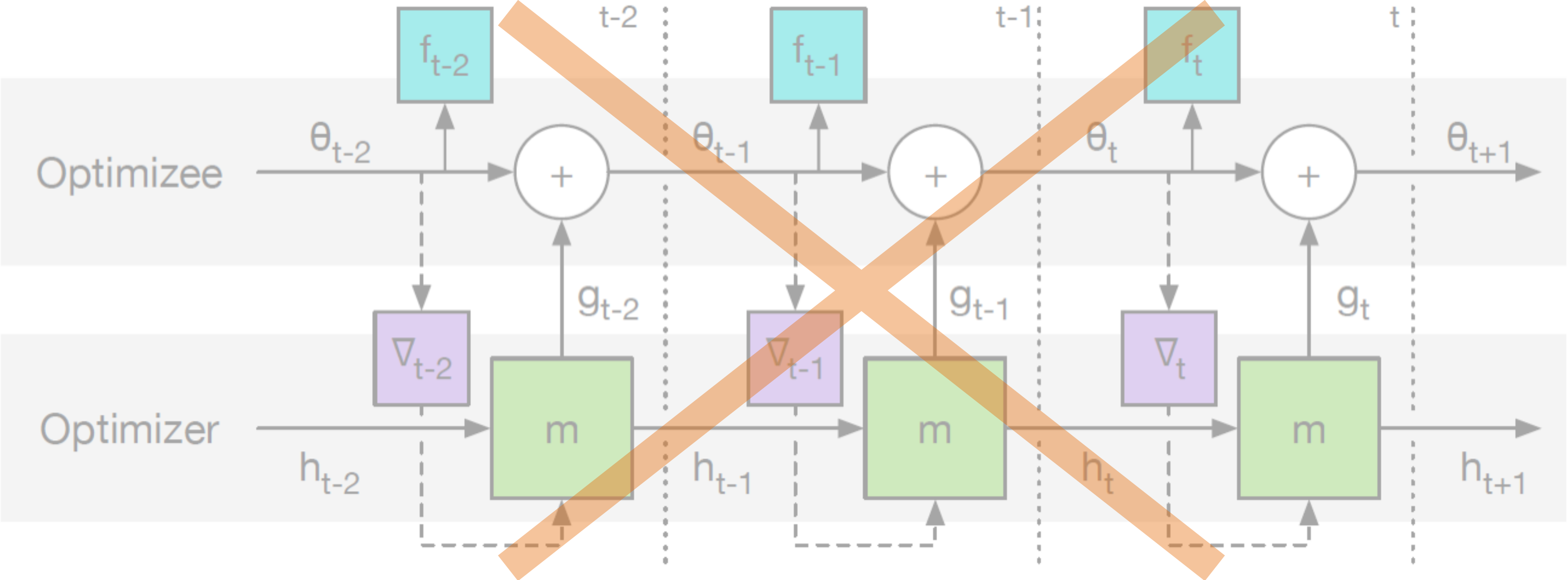
Computational graph guiding gradient flow

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

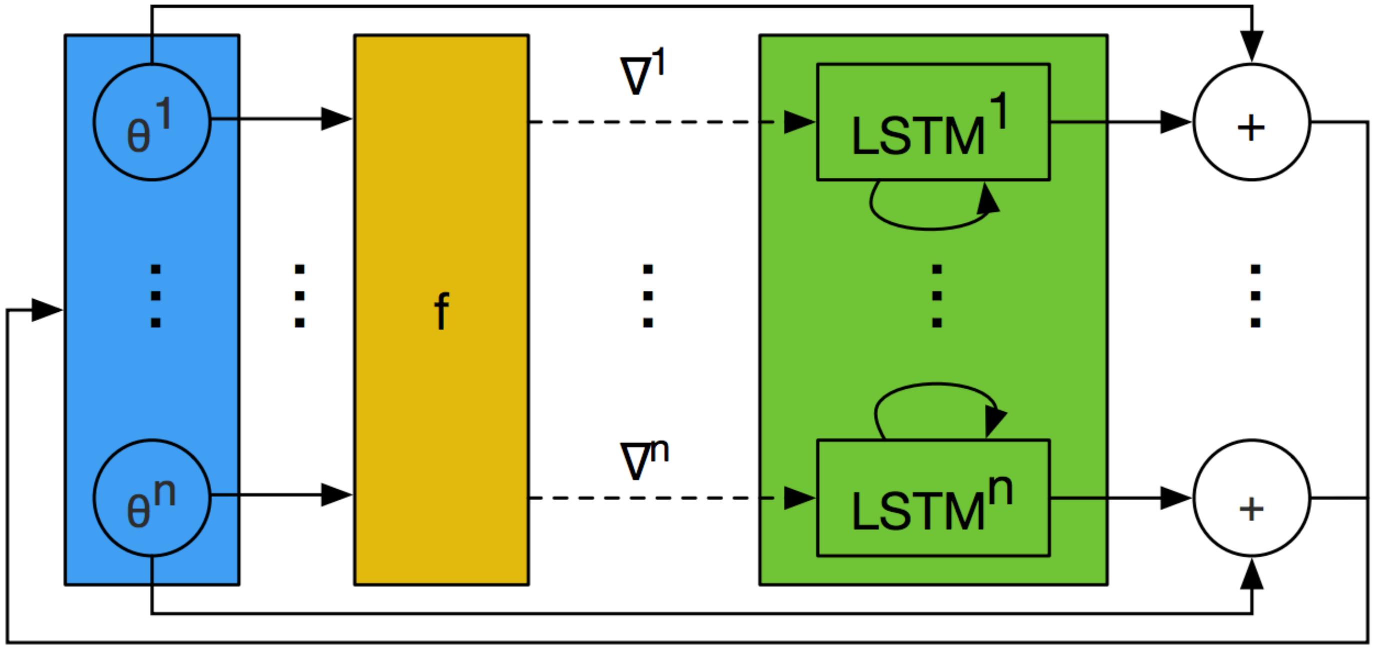
- In practice:

  infeasible



M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

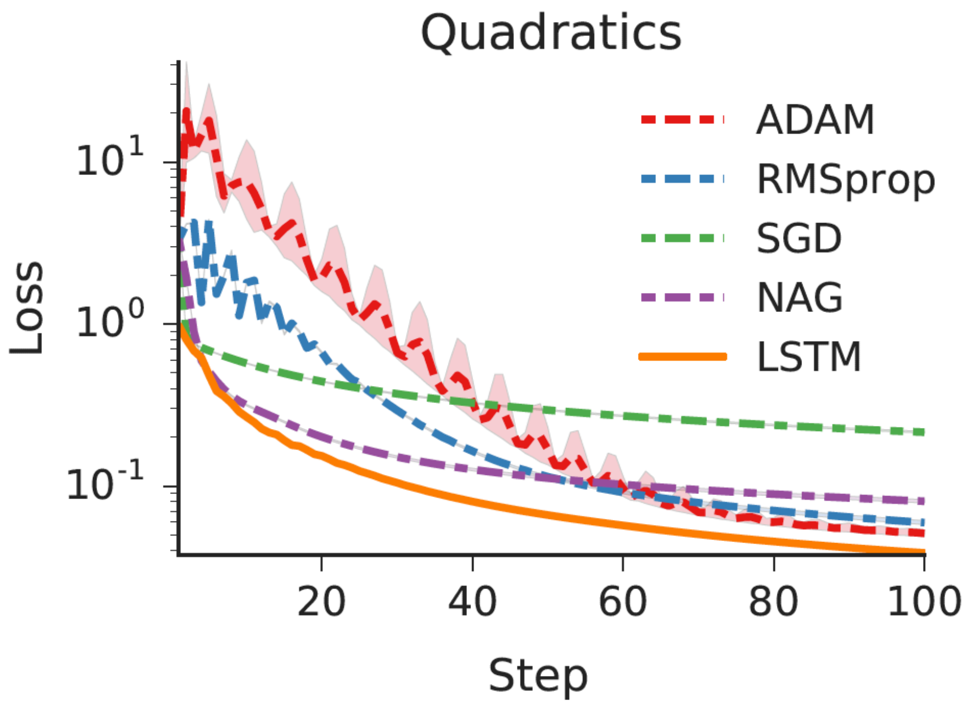


One step of an LSTM optimizer: all LSTMs have shared parameters, but separate hidden states

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Learning curves for the base network using different optimizers

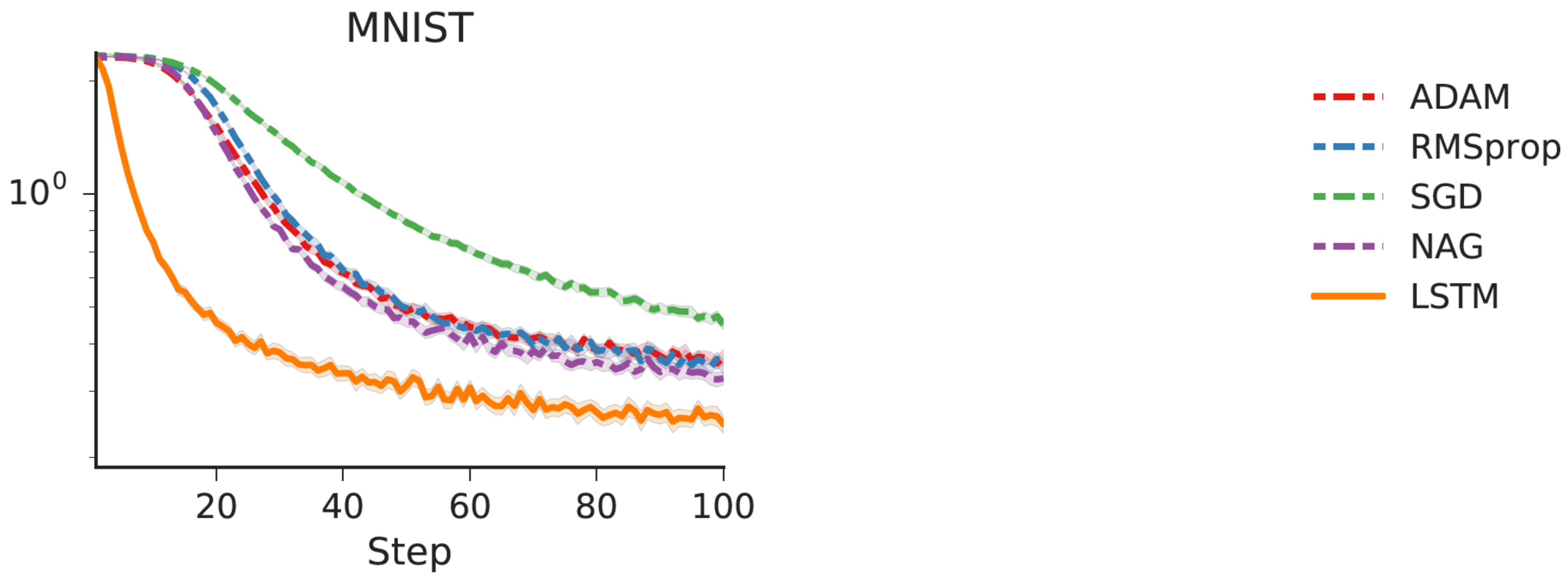


$$f(\theta) = \|W\theta - y\|_2^2$$

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

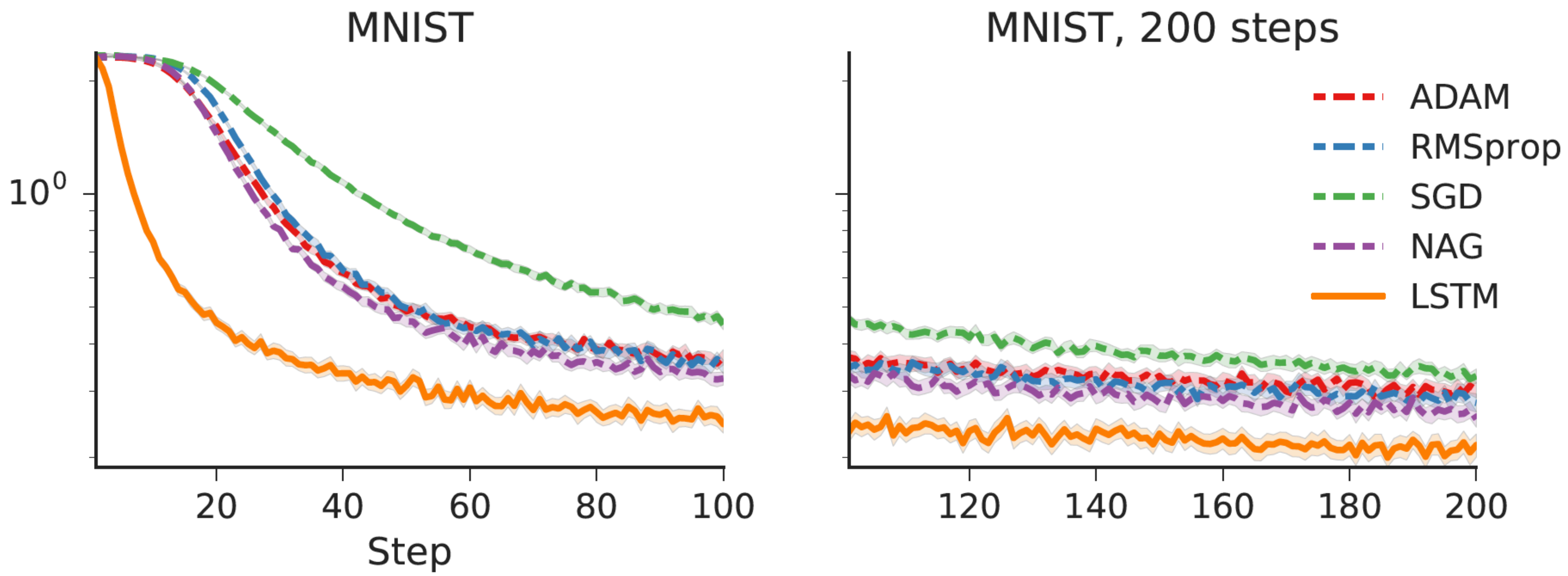
Learning curves for the base network using different optimizers



M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

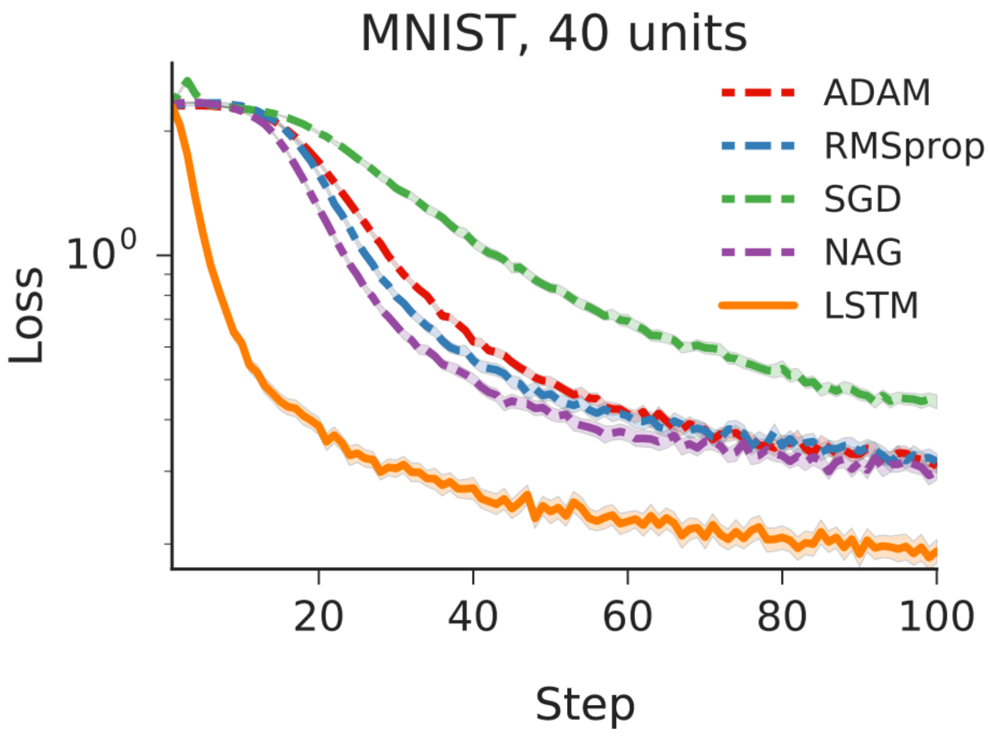
Learning curves for the base network using different optimizers



M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

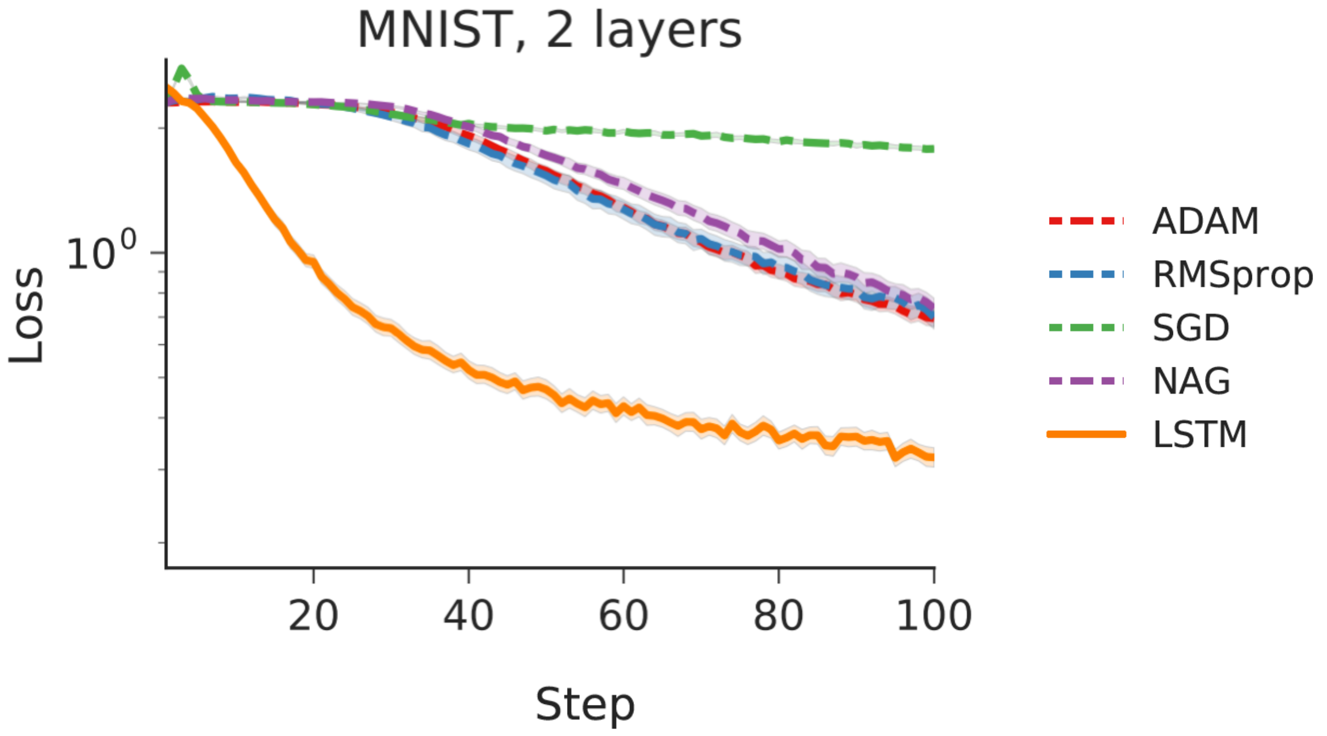
Generalization performance of optimizer



M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

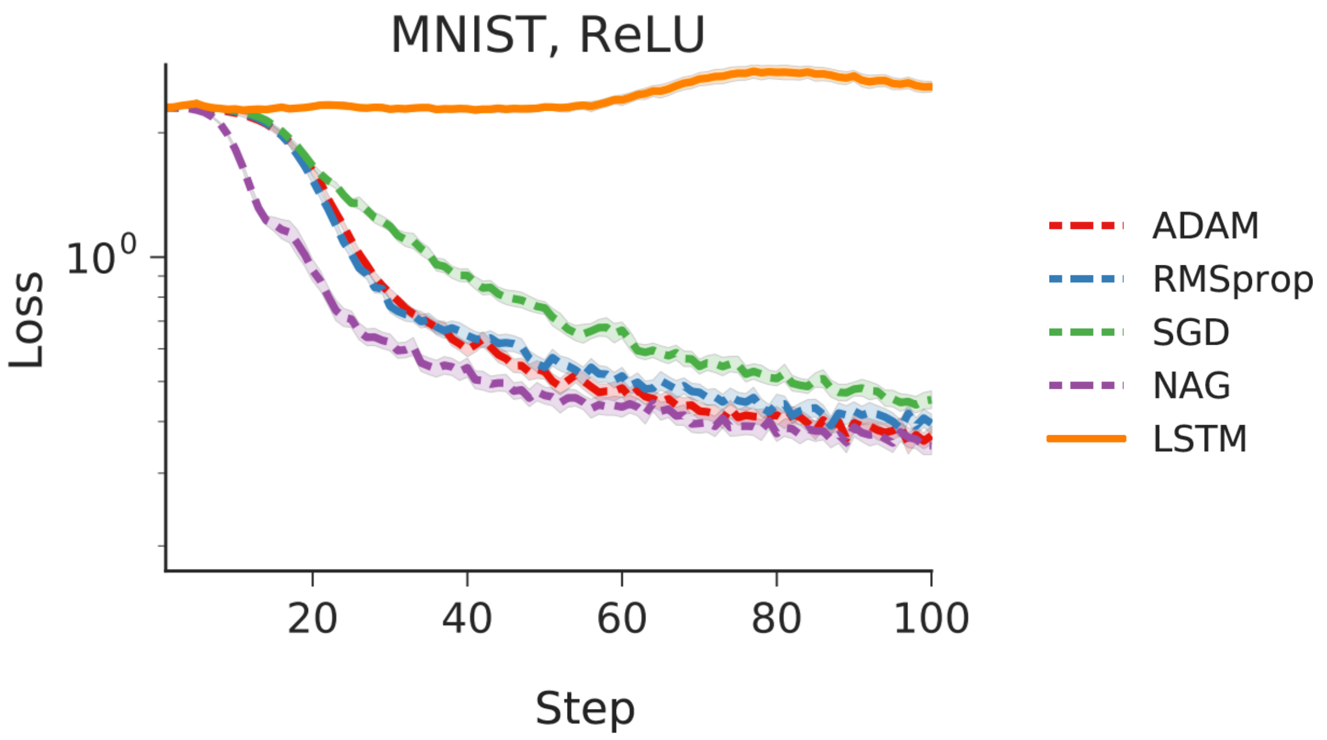
Generalization performance of optimizer



M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

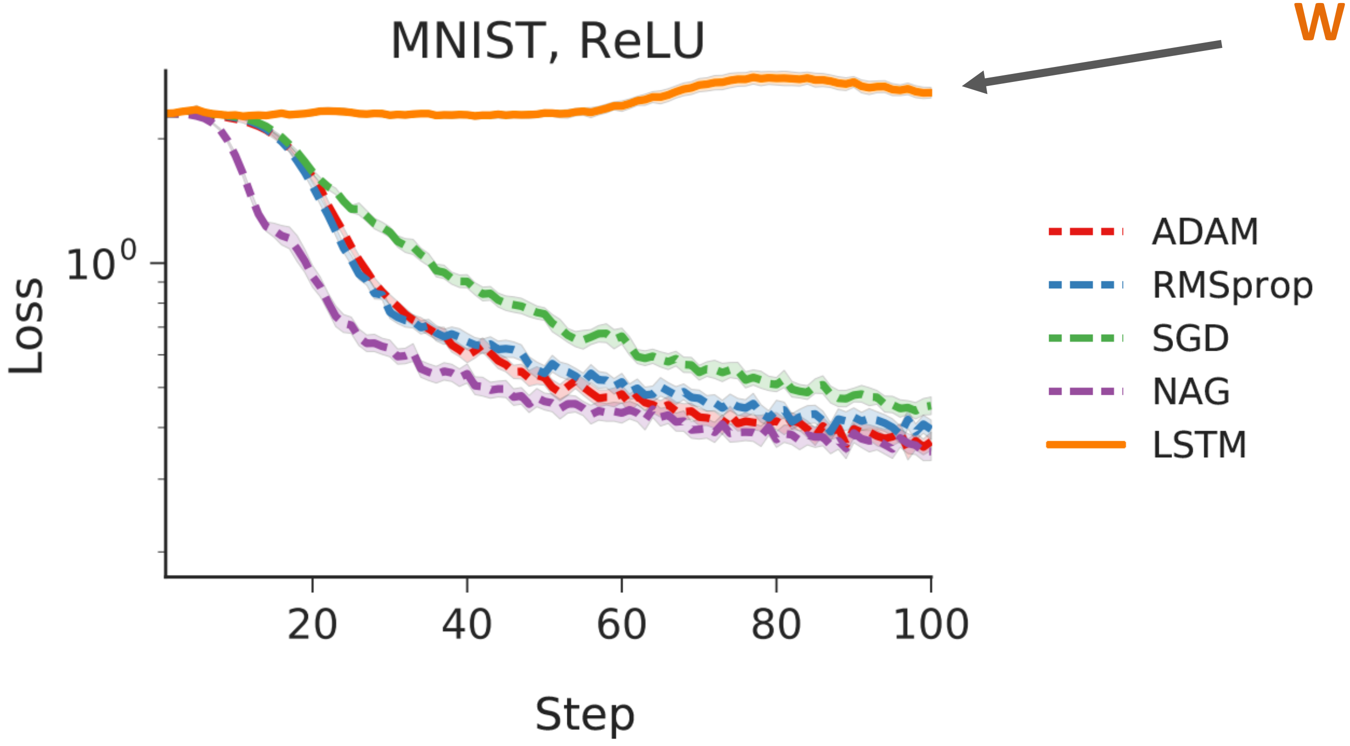
Generalization performance of optimizer



M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn by Gradient Descent by Gradient Descent

Generalization performance of optimizer

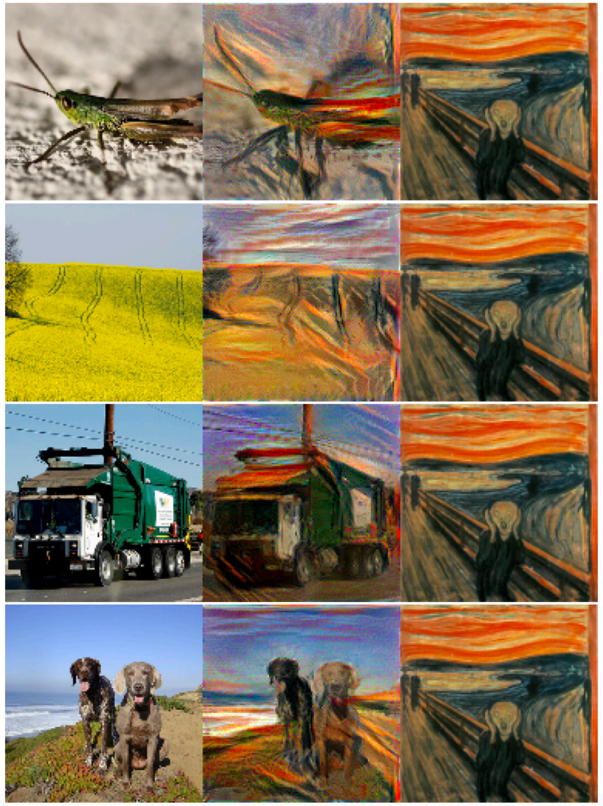


What went wrong?

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to learn by gradient descent by gradient descent

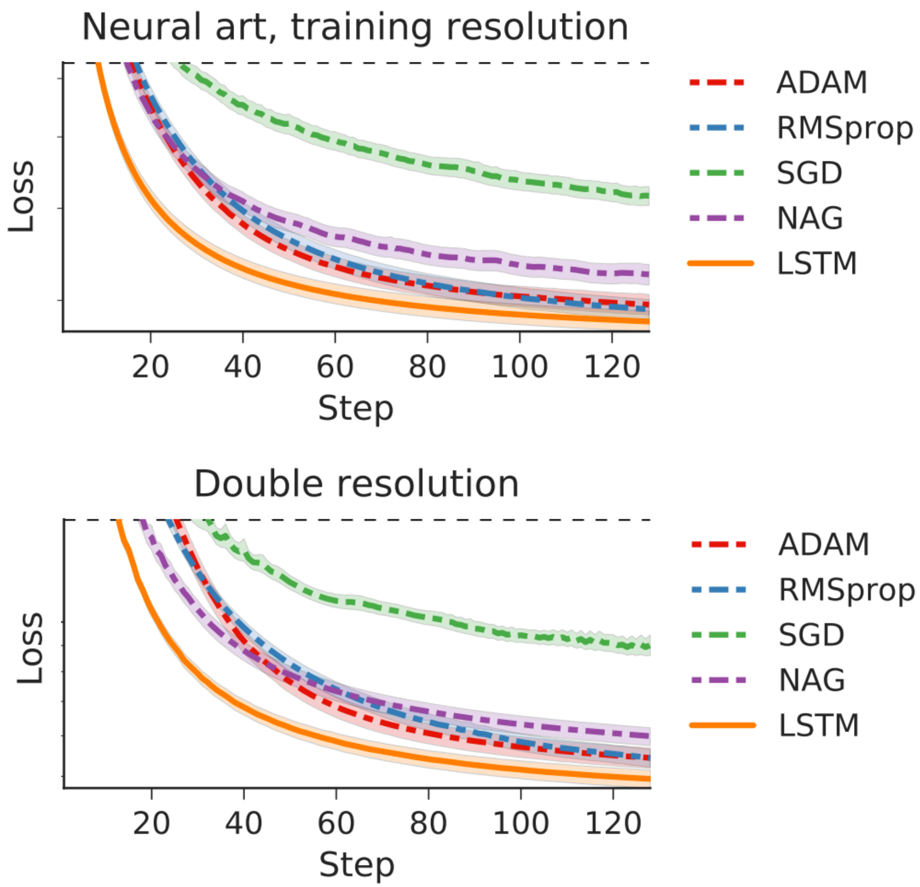
Using a learned optimizer for neural style transfer

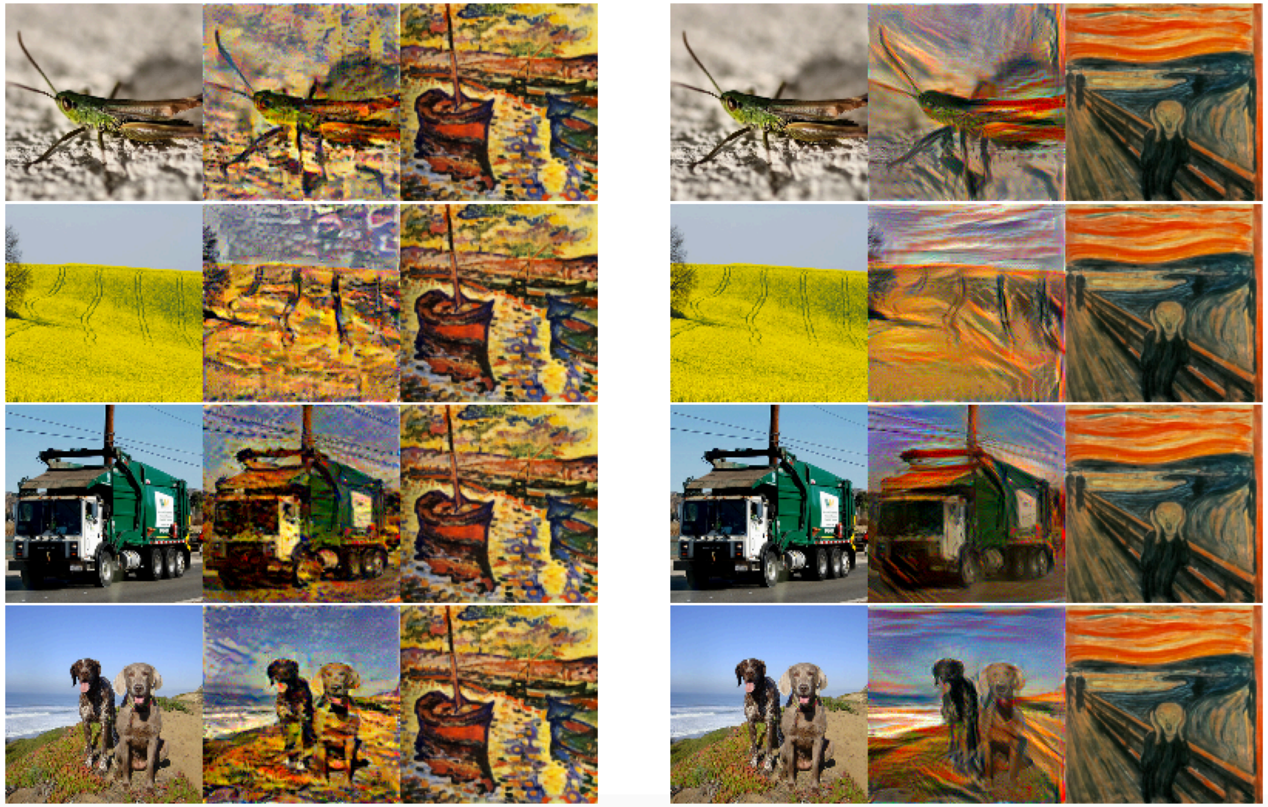


M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to learn by gradient descent by gradient descent

Using a learned optimizer for neural style transfer

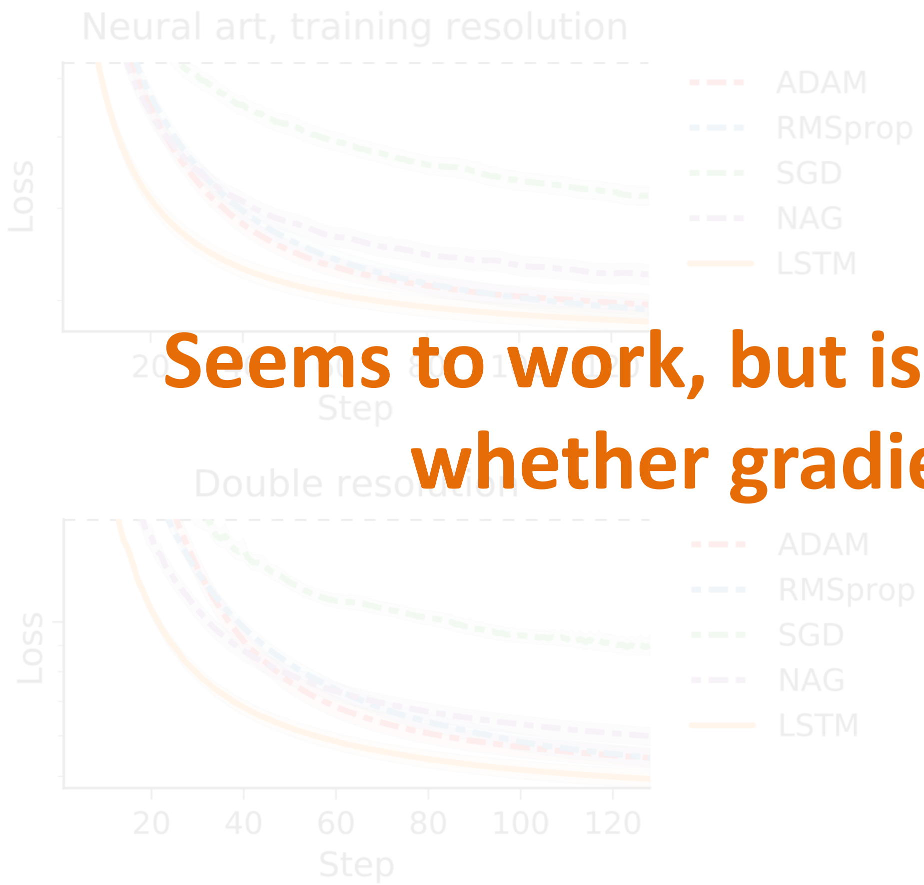


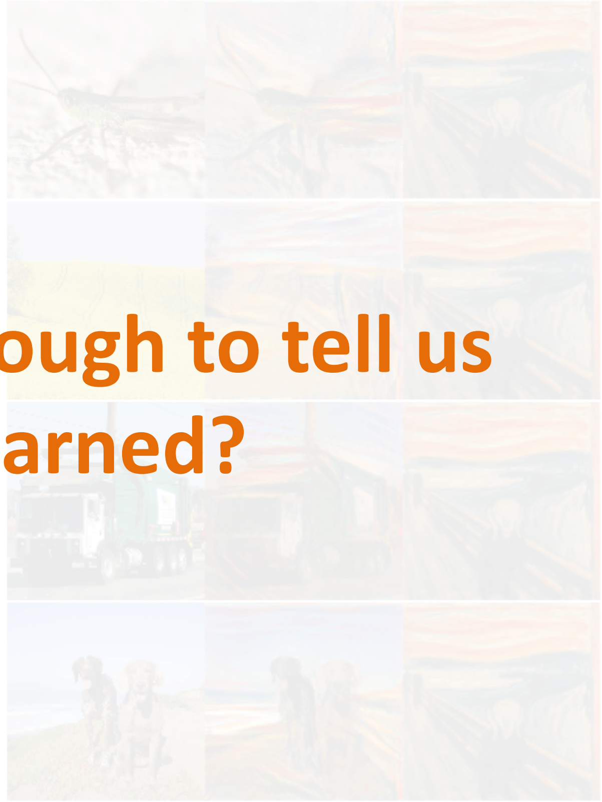


M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to learn by gradient descent by gradient descent

Using a learned optimizer for neural style transfer





Seems to work, but is this loss informative enough to tell us whether gradient descent was really learned?

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. NIPS (2016).

Learning to Learn for Global Optimization of Black Box Functions

- Address the problem of finding a global minimizer of a black-box loss function f:

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

- At test time, f is not available to the learner in closed form, but can be evaluated at a query point $\mathbf{x}\in\mathcal{X}$

- Hence, can only observe f through unbiased noisy pointwise observations $y \in \mathbb{R}$ such that:

$$f(\mathbf{x}) = \mathbb{E}[y \mid f(\mathbf{x})]$$

Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, and N. de Freitas. Learning to Learn for Global Optimization of Black Box Functions. arXiv preprint arXiv:1611.03824 (2016).

Learning to Learn for Global Optimization of Black Box Functions

- Given the current state of knowledge $\mathbf{h}_t$, propose a query point $\mathbf{x}_t$.
- Observe the response y_t .
- Update any internal statistics to produce $\mathbf{h}_{t+1}$.

Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, and N. de Freitas. Learning to Learn for Global Optimization of Black Box Functions. arXiv preprint arXiv:1611.03824 (2016).

Learning to Learn for Global Optimization of Black Box Functions

previous hidden state

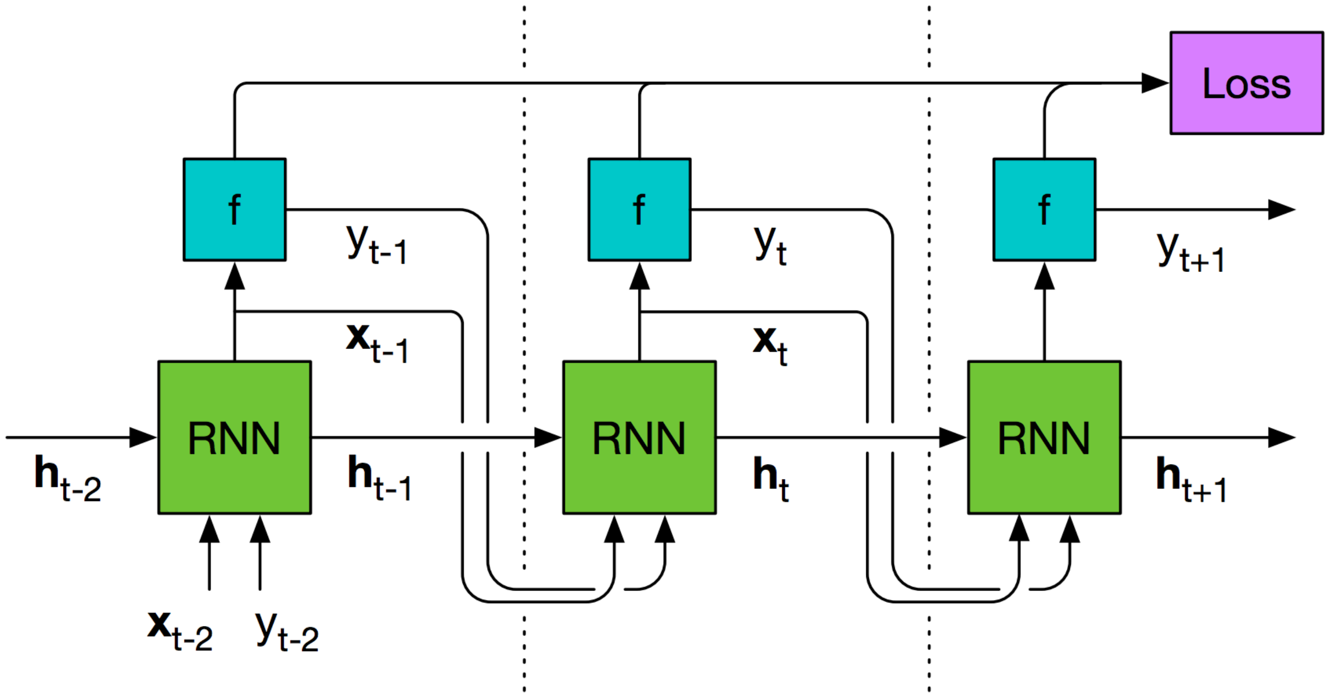
previous query point

previous function evaluation

$$\mathbf{h}_t, \mathbf{x}_t = \mathrm{RNN}_\theta(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, y_{t-1}),$$
$$y_t \sim p(y \mid \mathbf{x}_t)$$

updated hidden state

new query point



Computational graph of the learned black-box optimizer unrolled over multiple time steps: the learning process consists of differentiating the given loss with respect to the RNN parameters.

Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, and N. de Freitas. Learning to Learn for Global Optimization of Black Box Functions. arXiv preprint arXiv:1611.03824 (2016).

Learning to Learn for Global Optimization of Black Box Functions

Choice of loss function to train RNN optimizer

training horizon

$$L_{\text{obs}}(\theta) = \mathbb{E}_{f,y_{1:T-1}}\left[\sum_{t=1}^{T} f(\mathbf{x}_t)\right]$$

provide information
from every step along
the optimizer trajectory

$$L_{\text{EI}}(\theta) = -\mathbb{E}_{f,y_{1:T-1}}\left[\sum_{t=1}^{T} \text{EI}(\mathbf{x}_t \mid y_{1:t-1})\right]$$

expected posterior improvement
of querying $\mathbf{x}_t$ given
observations up to time t

$$L_{\text{OI}}(\theta) = \mathbb{E}_{f,y_{1:T-1}}\left[\underbrace{\sum_{t=1}^{T}(f(\mathbf{x}_t) - \min_{i<t}(f(\mathbf{x}_i)))}\right]$$

observed improvement
of querying $\mathbf{x}_t$ given
observations up to time t

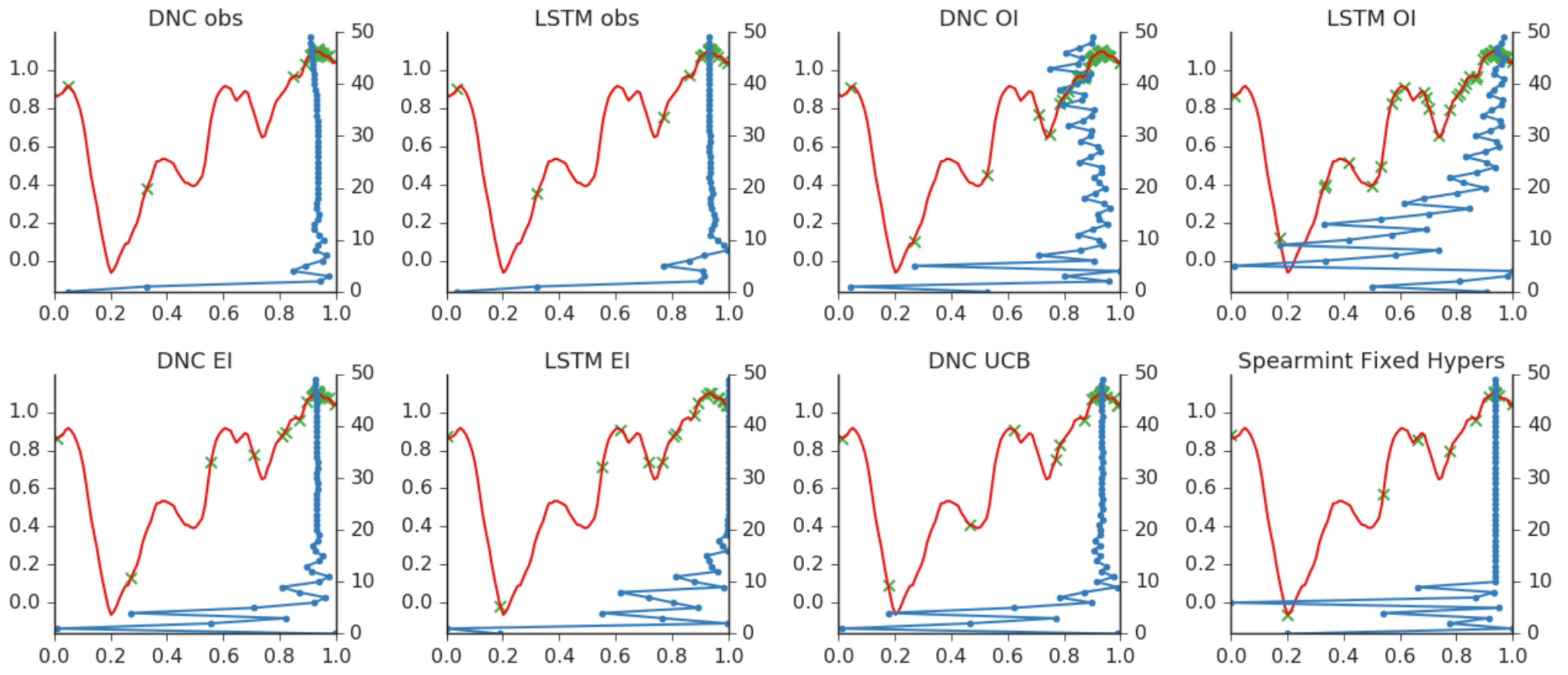
Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, and N. de Freitas. Learning to Learn for Global Optimization of Black Box Functions. arXiv preprint arXiv:1611.03824 (2016).

Learning to Learn for Global Optimization of Black Box Functions

Evaluating exploration capability over time

Search trajectories of $\mathbf{x}_t$ for different models on a 1-dimensional function

Red line: function value vs input.
Green cross: function value on query points.
Blue line: search iteration vs query locations.



Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, and N. de Freitas. Learning to Learn for Global Optimization of Black Box Functions. arXiv preprint arXiv:1611.03824 (2016).

Fast Reinforcement Learning Via Slow Reinforcement Learning

$\mathbf{L}_{\mu}$

μ

ML

Fast Reinforcement Learning Via Slow Reinforcement Learning

$\mathbf{L}_{\mu}$	RNN
μ	Optimizee parameters
$\mathbf{ML}$	Backpropagation

Index

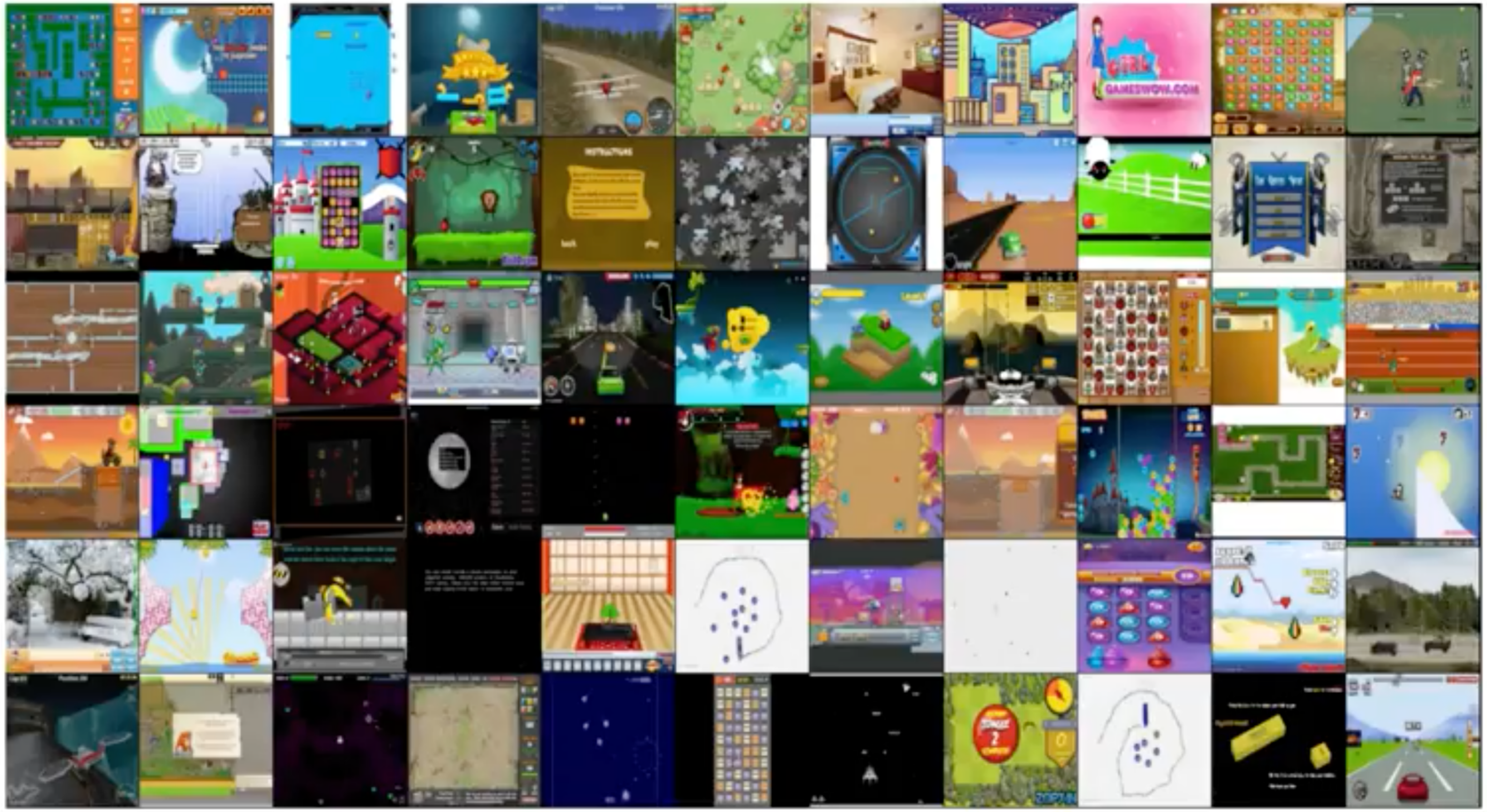
- Formal Definition of Meta-Learning
- Learning the Deep learning Architecture
- **Learning to Explore**
    - Learning to Optimize
    - **Learning to Explore An Environment**
- Learning to Seek Knowledge
- Learning to Communicate

Fast Reinforcement Learning Via Slow Reinforcement Learning

-Motivation-

Open AI Universe

A platform for benchmarking and developing the ability of agents to rapidly solve a wide variety of new problems that are difficult



RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

Fast Reinforcement Learning Via Slow Reinforcement Learning

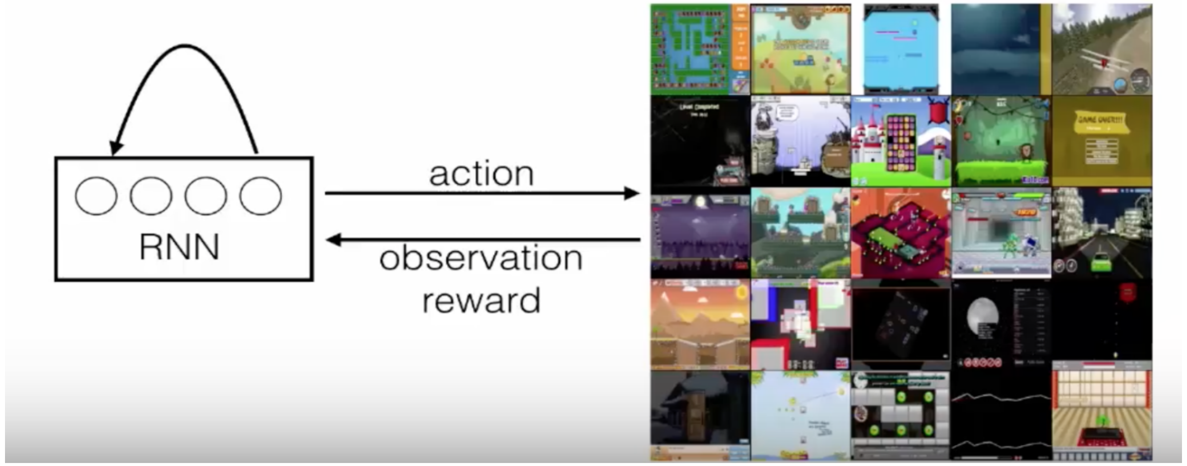
- **why are humans better than reinforcement learning agents?**
  - excellent data efficiency
  - prior experience (The agent needs to build its knowledge of the environment from scratch)

- Prior experience
  - Can be represented by a distribution over environments
    - fundamental nature of rules
    - appearance and dynamics of objects
    - typical ways in which control works
    - how scoring works
    - Etc

RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

Given a distribution over environments, which RL does the best?

- **Solution:** train an RNN policy to solve many environments simultaneously

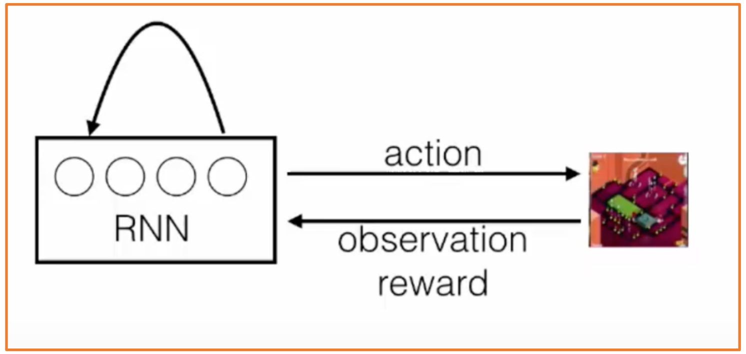


- **Performance measure**
  - How well does the RNN policy solve environments drawn from a random distribution?

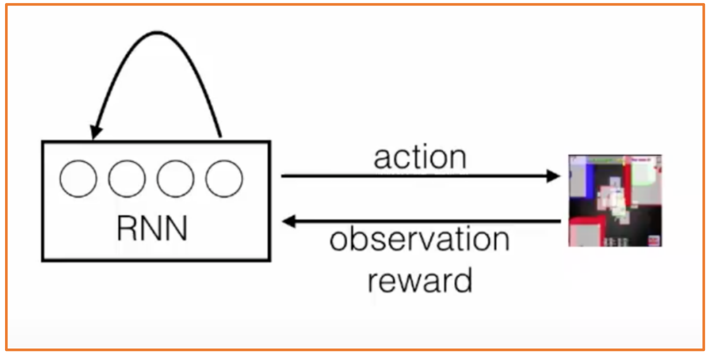
RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

Training

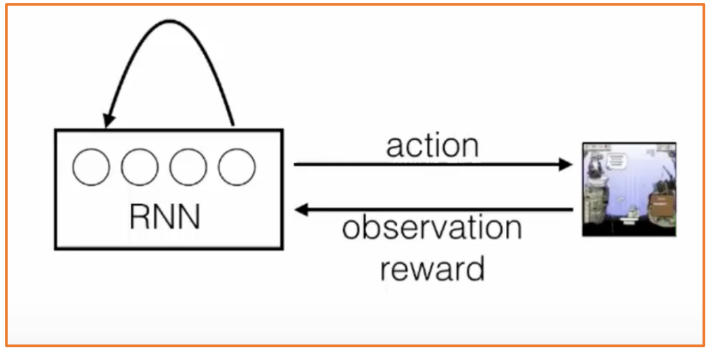
-1-

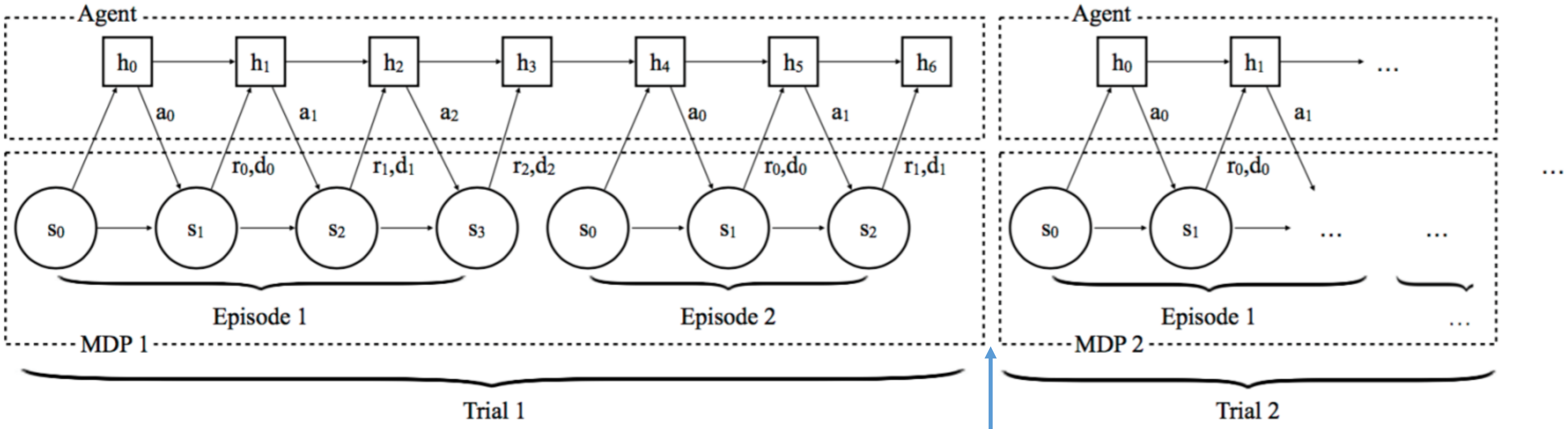


-2-



-3-



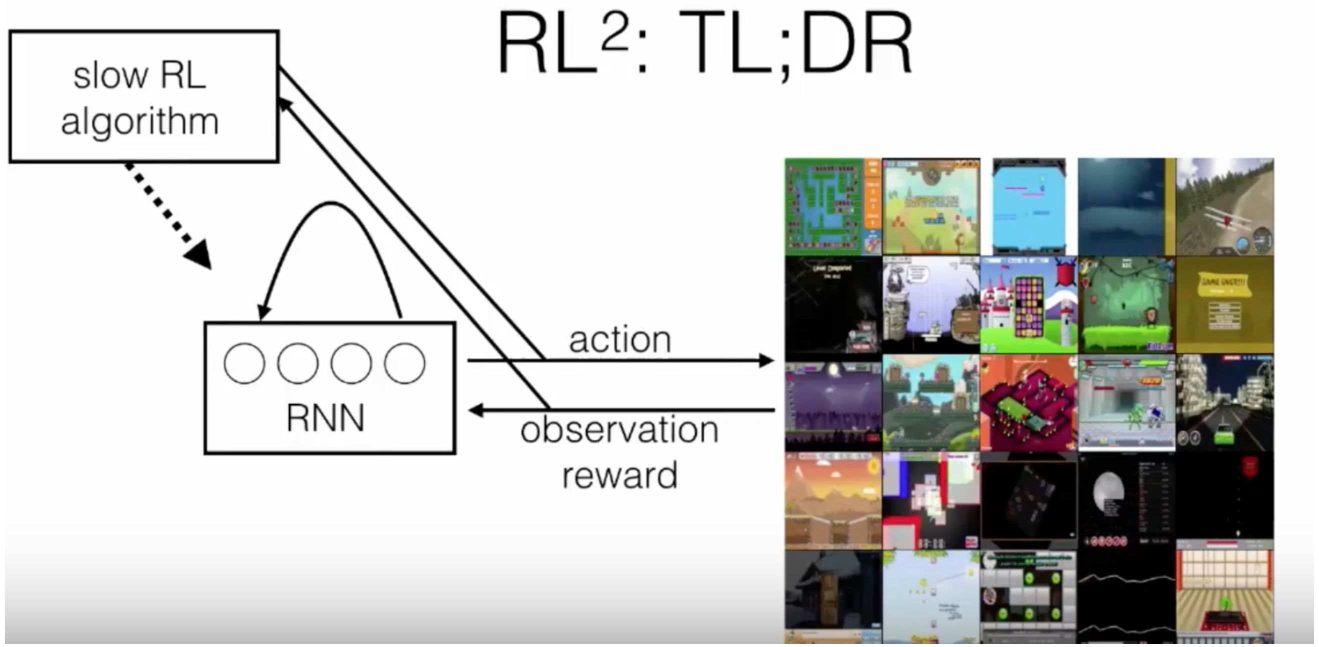


Reset the hidden state

RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

Fast Reinforcement Learning Via Slow Reinforcement Learning

- Slow RL Algorithm
  - Trains the RNN policy
  - Tune the weights to solve a given environment
- Fast RL Algorithm
  - The RL algorithm to solve a particular MDP

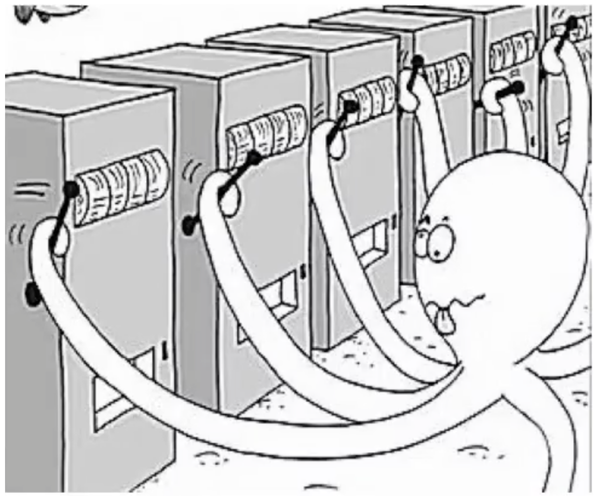


RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

Evaluation: Multi-Armed Bandit

Can RL2 learn algorithms that achieve good performance on MDP classes with special structure and optimal solution?

- Multi-armed bandit
  - Agent environment is **stateless**
  - There are **k arms**
  - At every **time step**, the agent pulls one arm and receives an award drown from an unknown distribution
  - Goal: **maximize the total reward** obtained over a fixed number of steps
  - Key challenge: **balance** exploration and exploitation

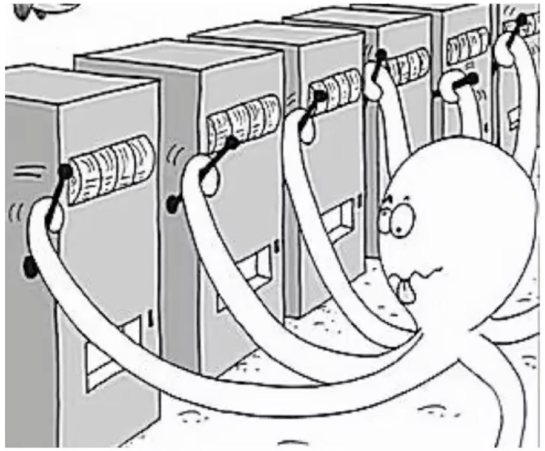


Asymptotically optimal algorithms

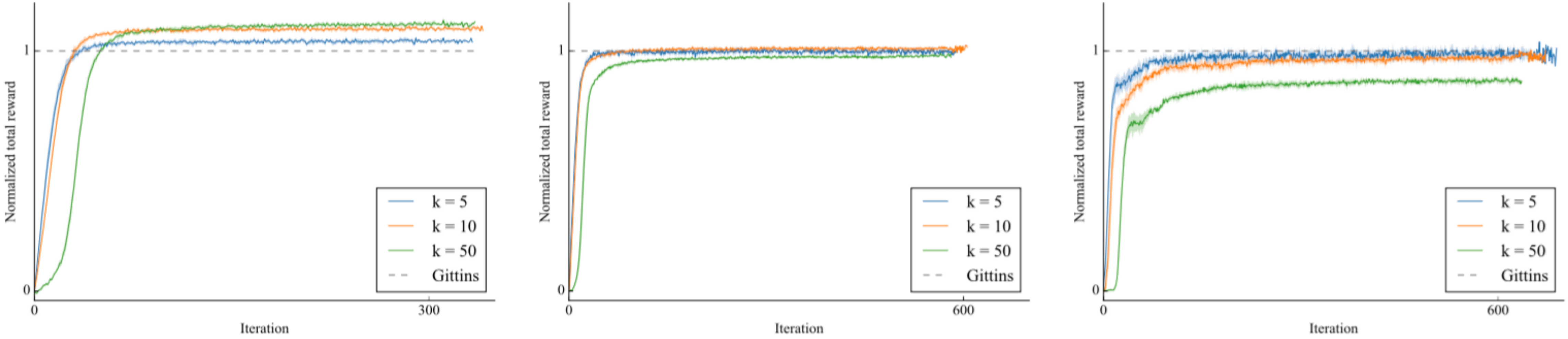
RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

Evaluation: Multi-Armed Bandit

Can RL2 learn algorithms that achieve good performance on MDP classes with special structure and optimal solution?



Asymptotically optimal algorithms



(a) $n = 10$ (b) $n = 100$ (c) $n = 500$

K: number of bandits

N: number of episodes

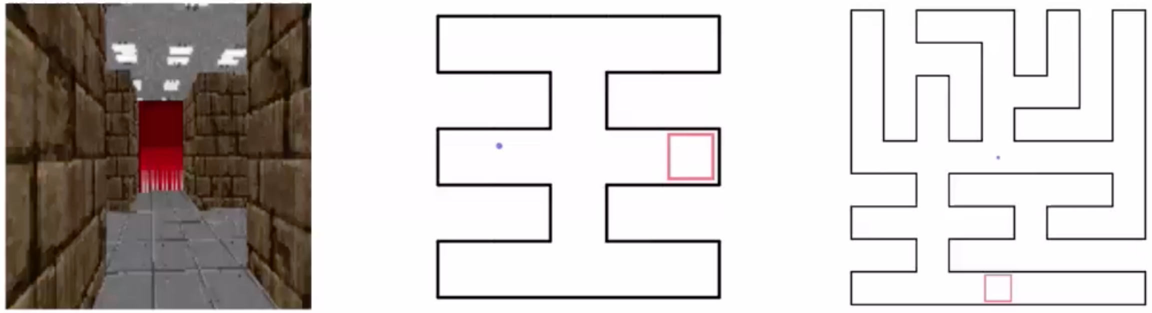
RL for long time horizons is difficult!

RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

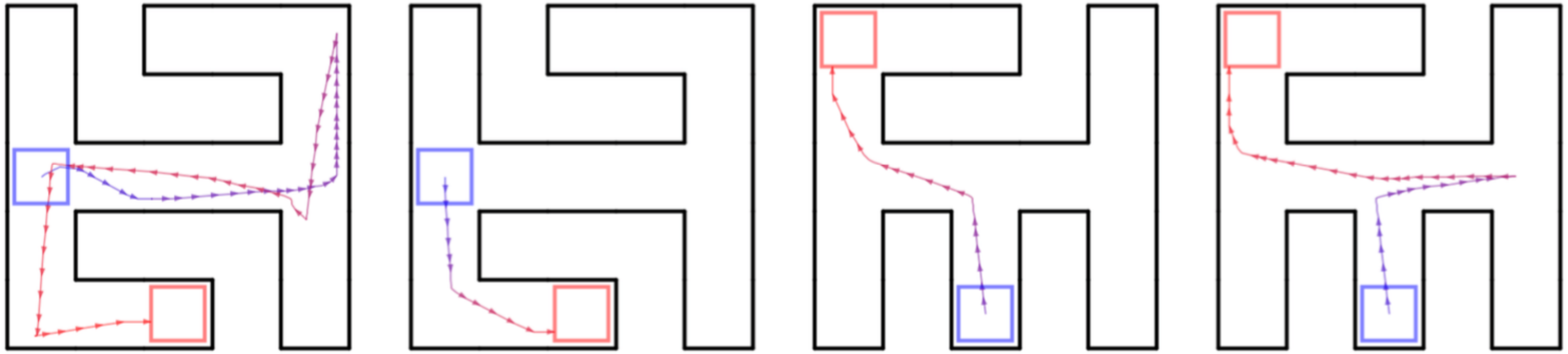
Evaluation: Visual Navigation Built on ViZDoom

Can RL2 scale to high-dimensional tasks?

Goal of the meta-learner: navigate a random maze to find a target


Agent's view Small maze Large maze

Reward
+1, target is reached,
− 0.001 hit the wall
− 0.04 per time step


(a) Good behavior, 1st episode (b) Good behavior, 2nd episode (c) Bad behavior, 1st episode (d) Bad behavior, 2nd episode

RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe. Duan, Yan, et al. "RL^ 2: Fast Reinforcement Learning via Slow Reinforcement Learning." *arXiv preprint arXiv:1611.02779* (2016).

Fast Reinforcement Learning Via Slow Reinforcement Learning

$\mathbf{L}_{\mu}$

μ

ML

Fast Reinforcement Learning Via Slow Reinforcement Learning

$\mathbf{L}_{\mu}$ RNN (Reinforcement learning)

μ RNN weights

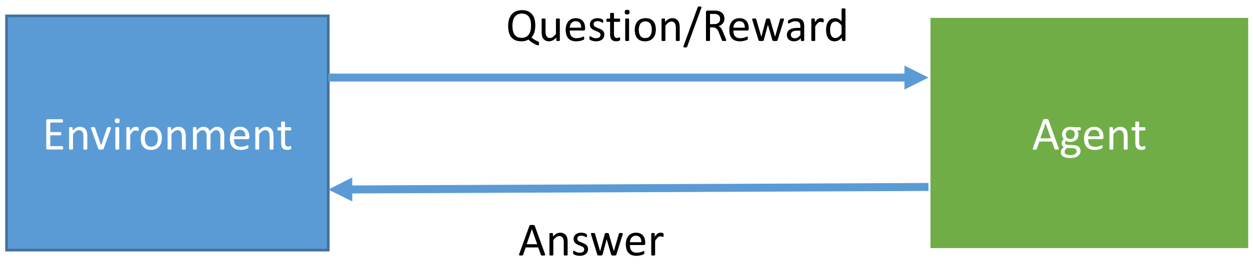
$\mathbf{ML}$ RNN (Reinforcement learning)

Index

- Formal Definition of Meta-Learning
- Learning the Deep learning Architecture
- Learning to Explore
- **Learning to Seek Knowledge**
- Learning to Communicate

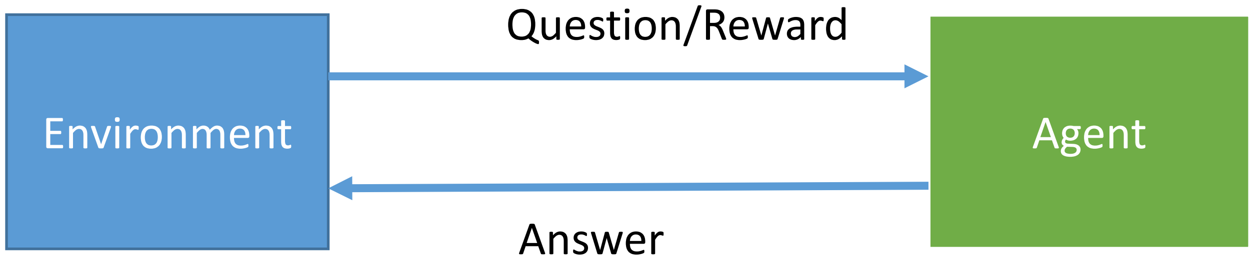
Learning to Perform Physics Experiments Via Deep Reinforcement Learning

- Goal: build agents that can learn to experiment so as to learn representations that are informative about the physical properties of the object using RL
- Formulation: Experimentation is the problem of answering questions about the non-visual properties of the object

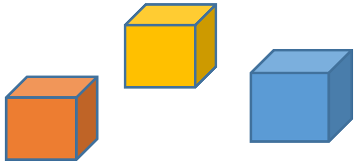


Denil, Misha, et al. "Learning to Perform Physics Experiments via Deep Reinforcement Learning." *arXiv preprint arXiv:1611.01843* (2016).

Learning to Perform Physics Experiments Via Deep Reinforcement Learning

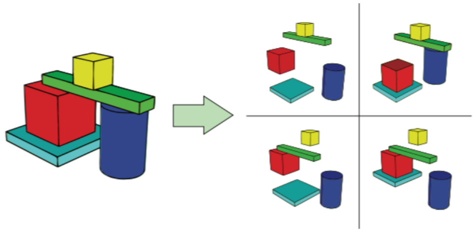


- Environments
  - Heavier: Agent applies forces to the blocks and must infer which of the blocks is the heaviest



  $\Longrightarrow$ Estimate **Mass**

  - Towers: Agent infers how many rigid bodies a tower is composed of by knocking it down



  $\Longrightarrow$ Estimate **cohesion of objects**

- Actions: Poking or answering questions

Denil, Misha, et al. "Learning to Perform Physics Experiments via Deep Reinforcement Learning." *arXiv preprint arXiv:1611.01843* (2016).

Learning to Perform Physics Experiments Via Deep Reinforcement Learning

- The agent is trained to answer questions using reinforcement learning
- The environment follows a three phase structure

Interaction → **Labeling** → **Reward**

Interaction
- Exploration phase
- The agent is free to interact with the environment

Labeling
- Agent answers the question of the environment

Reward
- Agent produces a labeling action

Denil, Misha, et al. "Learning to Perform Physics Experiments via Deep Reinforcement Learning." *arXiv preprint arXiv:1611.01843* (2016).

Learning to Perform Physics Experiments Via Deep Reinforcement Learning

- The agent is trained to answer questions using reinforcement learning
- The environment follows a three phase structure

Interaction → **Labeling** → **Reward**

Interaction
- Exploration phase
- The agent is free to interact with the environment

Labeling
- Agent answers the question of the environment

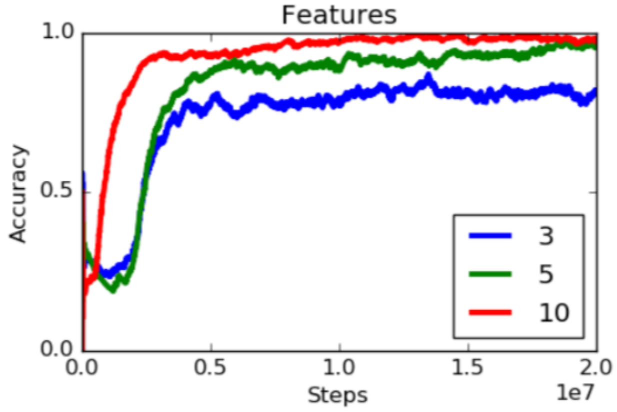
Reward
- Agent produces a labeling action

Tradeoff btw answering now or delaying to gather more information

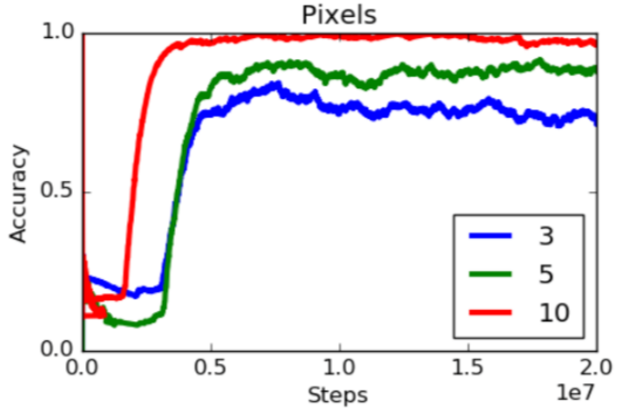
1. Difficulty to answer the question (Environment)
2. Cost of gathering information (varying the discount factor)

Denil, Misha, et al. "Learning to Perform Physics Experiments via Deep Reinforcement Learning." *arXiv preprint arXiv:1611.01843* (2016).

Which is HEAVIER?
-Results-



Training on features (block positions)



Training on pixels

- As the level of difficulty increases, the learned policy transitions from guessing immediately when a heavy block is found to strongly preferring to poke all blocks before making a decision

Denil, Misha, et al. "Learning to Perform Physics Experiments via Deep Reinforcement Learning." *arXiv preprint arXiv:1611.01843* (2016).

Learning to Perform Physics Experiments Via Deep Reinforcement Learning

$\mathbf{L}_{\mu}$

μ

ML

Learning to Perform Physics Experiments Via Deep Reinforcement Learning

$\mathbf{L}_\mu$ DNN Architecture

μ RNN weights

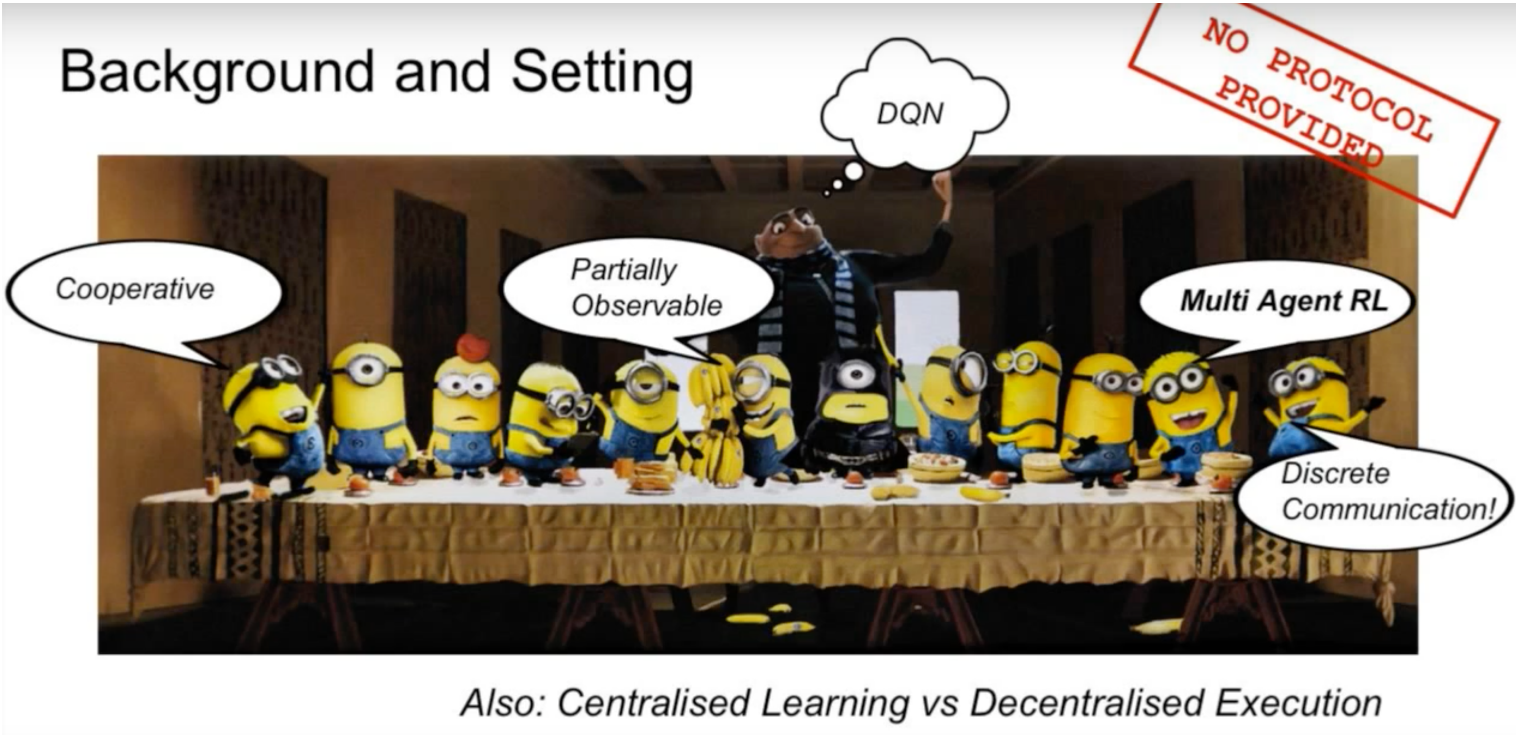
$\mathbf{ML}$ Reinforcement Learning

Index

- Formal Definition of Meta-Learning
- Learning the Deep learning Architecture
- Learning to Explore
- Learning to Seek Knowledge
- **Learning to Communicate**

Learning to Communicate with Deep Multi-Agent Reinforcement Learning

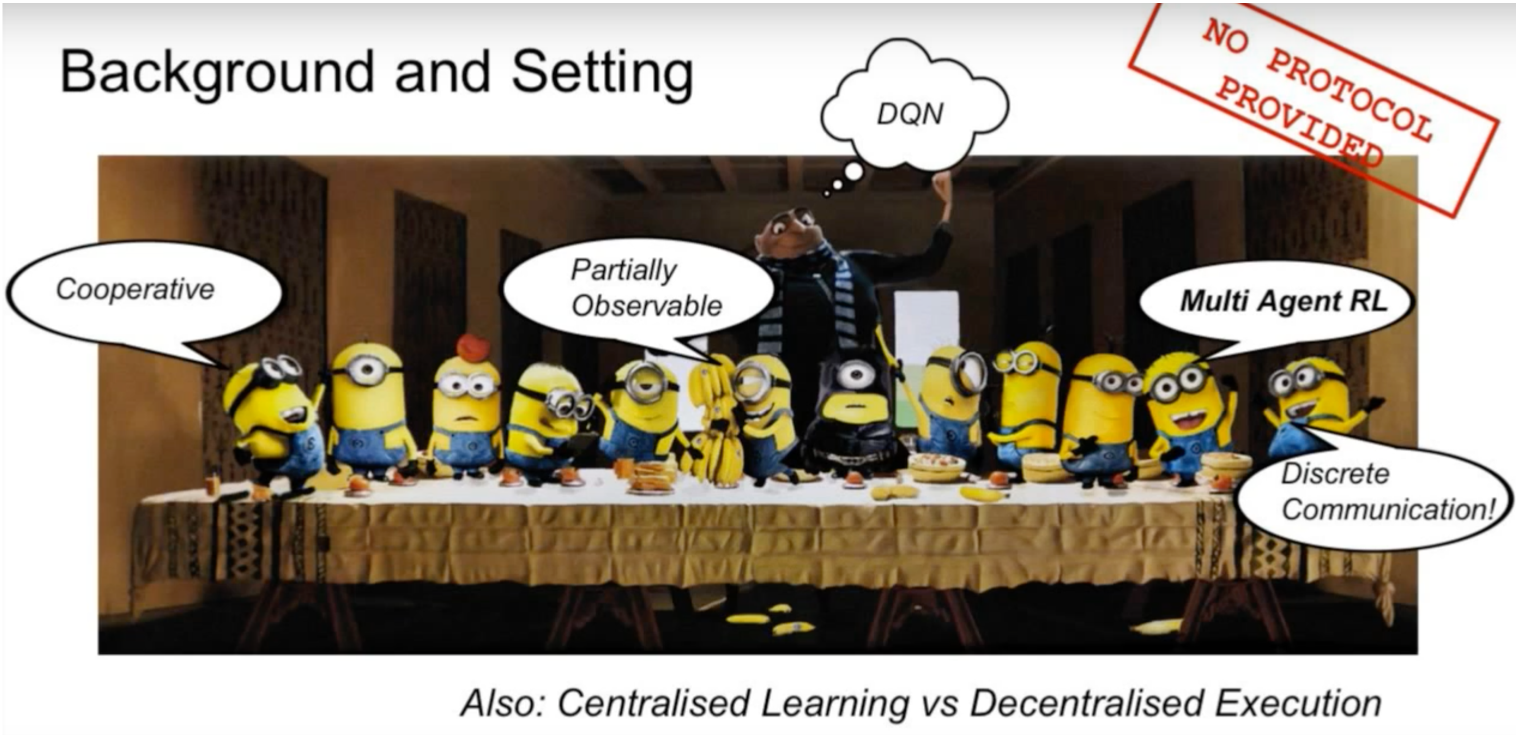
- **Goal**: how can agents use machine learning to automatically discover the communication protocols they need to coordinate their behavior?
- Agents must learn communication protocols in order to share information that is needed to solve the task



Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Learning to Communicate with Deep Multi-Agent Reinforcement Learning

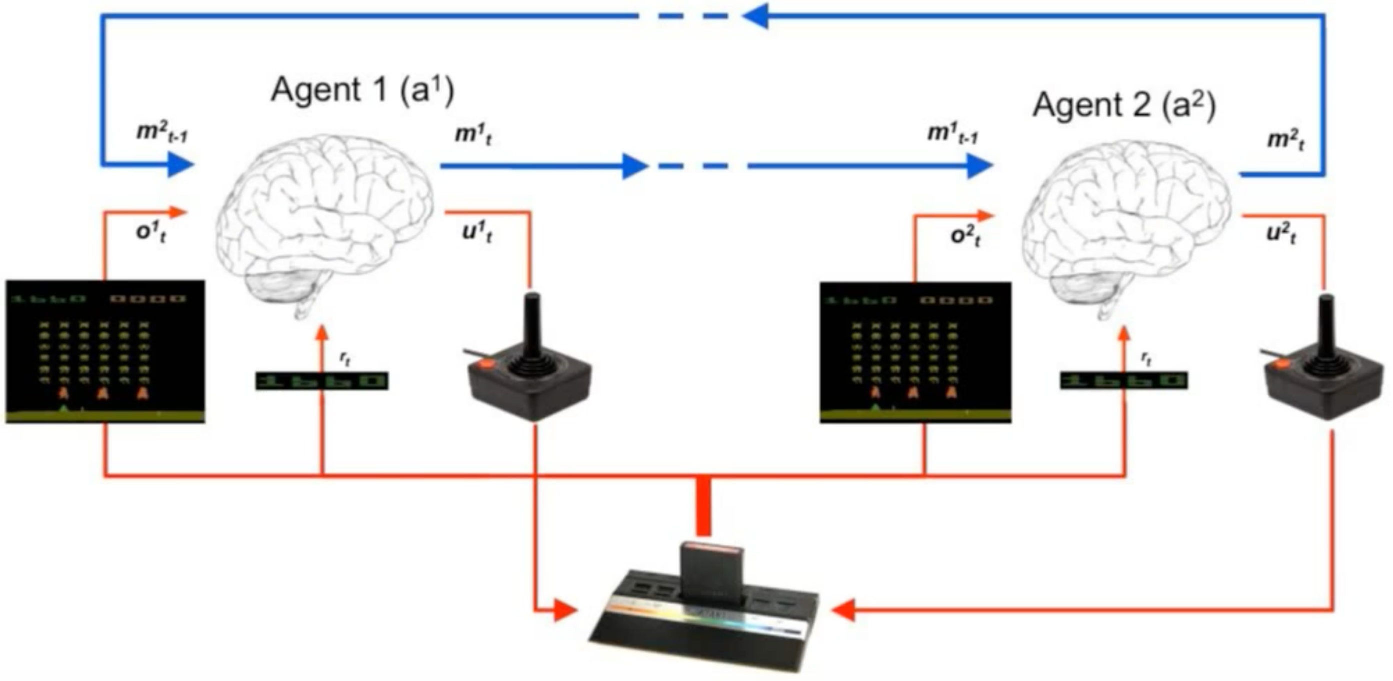
- **Setting** under consideration
  - Sequential multi-agent decision problems
  - Fully cooperative: Agents share the goal of maximizing the same discounted sum of rewards
  - Partially observable : Each agent receives a partial observation correlated with the state
  - Agents can communicate with each others as part of solving the task



Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Learning to Communicate with Deep Multi-Agent Reinforcement Learning

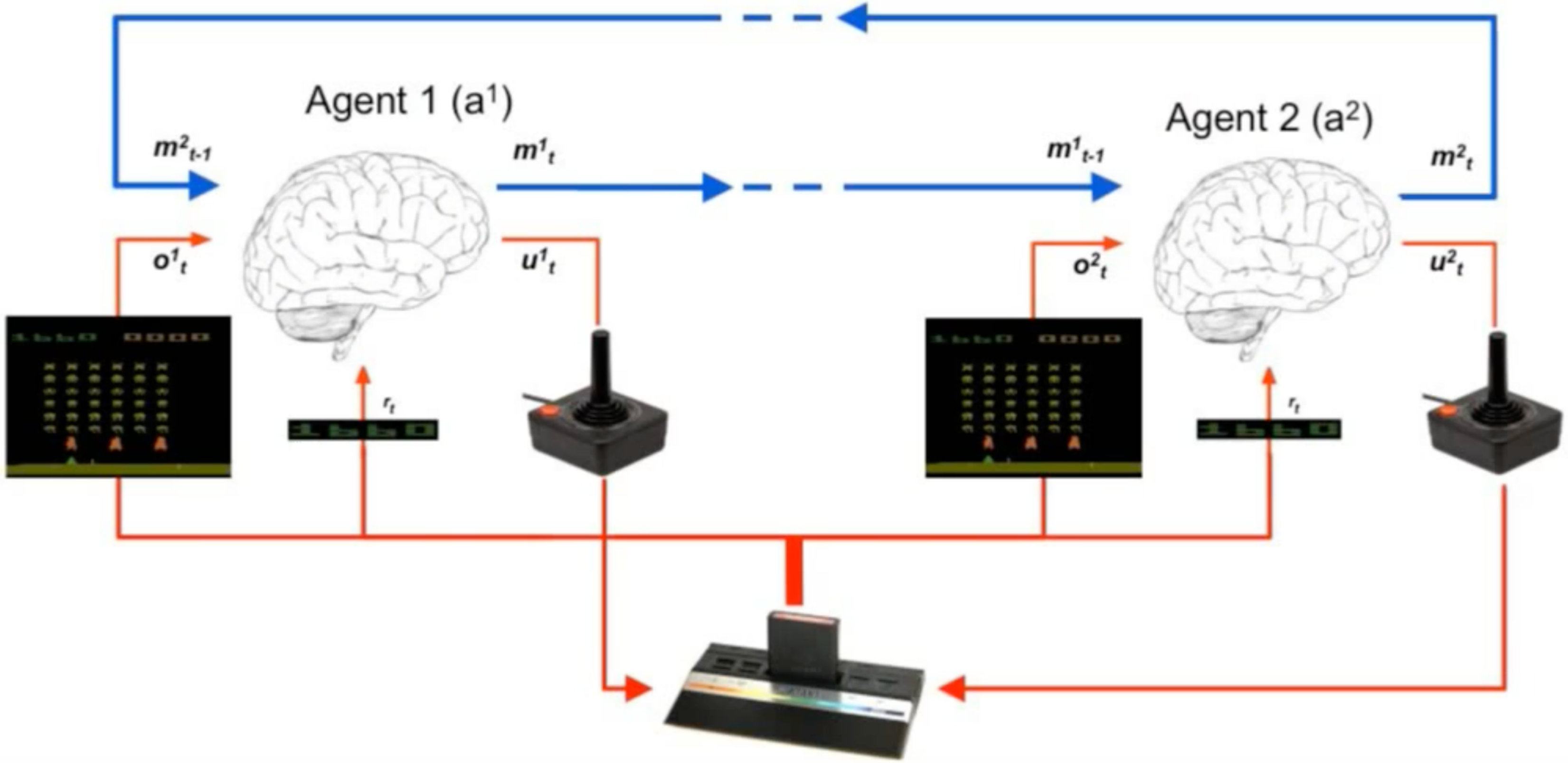
- Training phase:
  - Centralized learning phase: All agents learn together and communicate freely
  - The strategy they come up with is decentralized
  - There is a channel, but agents initially don't know how to use
  - Learn a strategy for communication through the channel



Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

How Can We Do Reinforcement Learning With Multiple Agents?

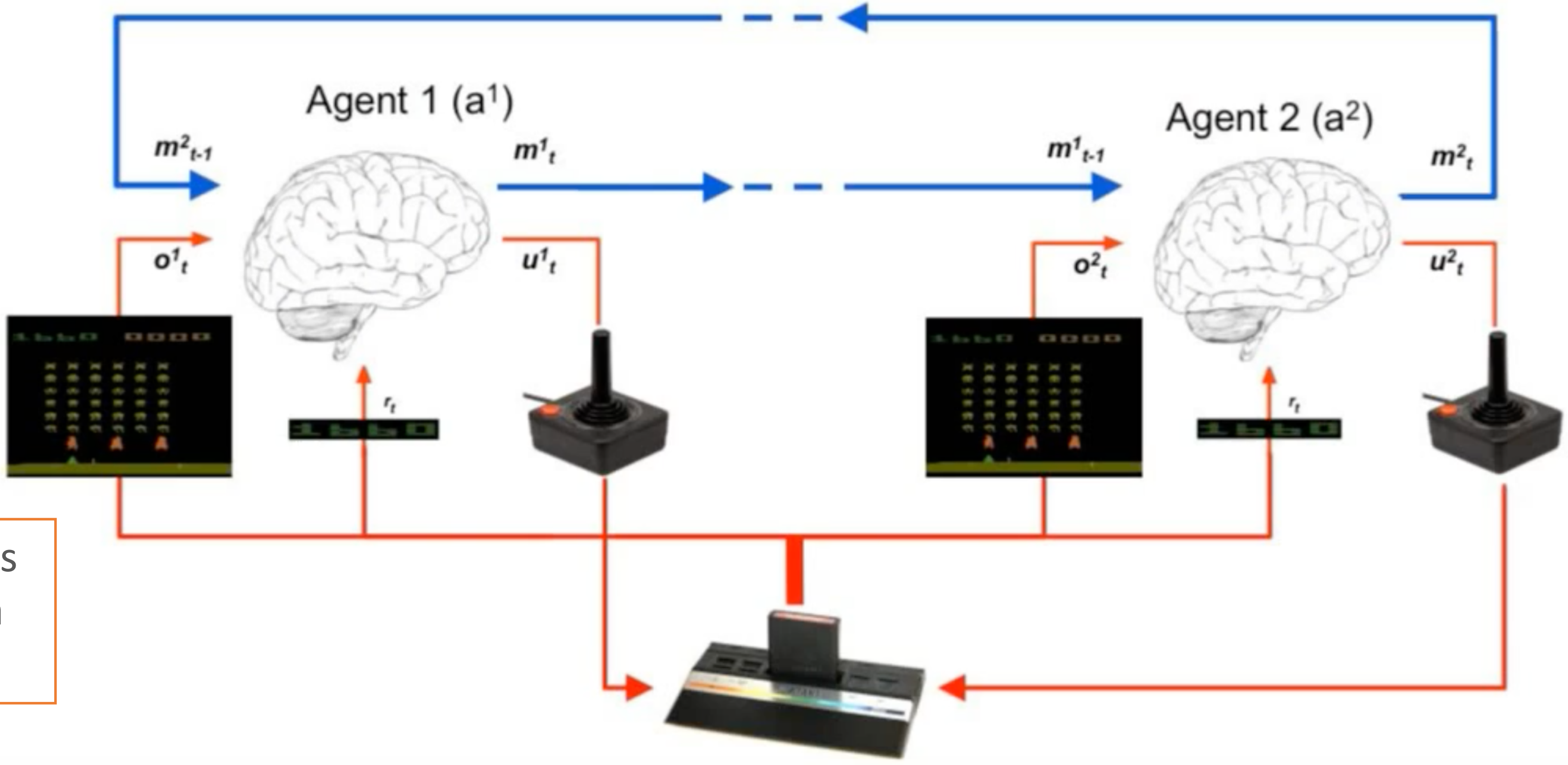
- **Answer**
  - Each Agent has a DQN network
  - 2 action spaces
  - 1 state space
  - shared reward



Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

How Can We Do Reinforcement Learning With Multiple Agents?

- **Answer**
  - Each Agent has a DQN network
  - 2 action spaces
  - 1 state space
  - Shared reward



What's the best way allowing agents to communicate in order to solve a task?

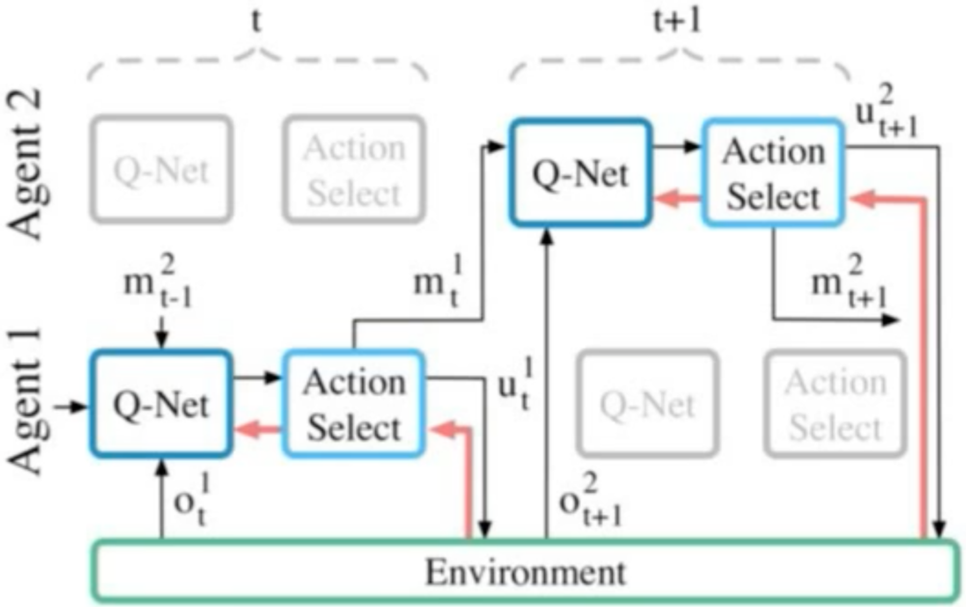
Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Differentiable **I**nter-**A**gent **L**earning & **R**einforced **I**nter-**A**gent **L**earning

Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Reinforced Inter-Agent Learning

- **RIAL**
  - The agent treats the communication message as another action (Learn the Q-value for messages)
  - Process
    - Q-Network receives observation and message
    - Select the action/message to send
    - Agent2 receives the message
    - Environment sends the reward
  - Parameter sharing
  - There is no gradient exchange between the agents

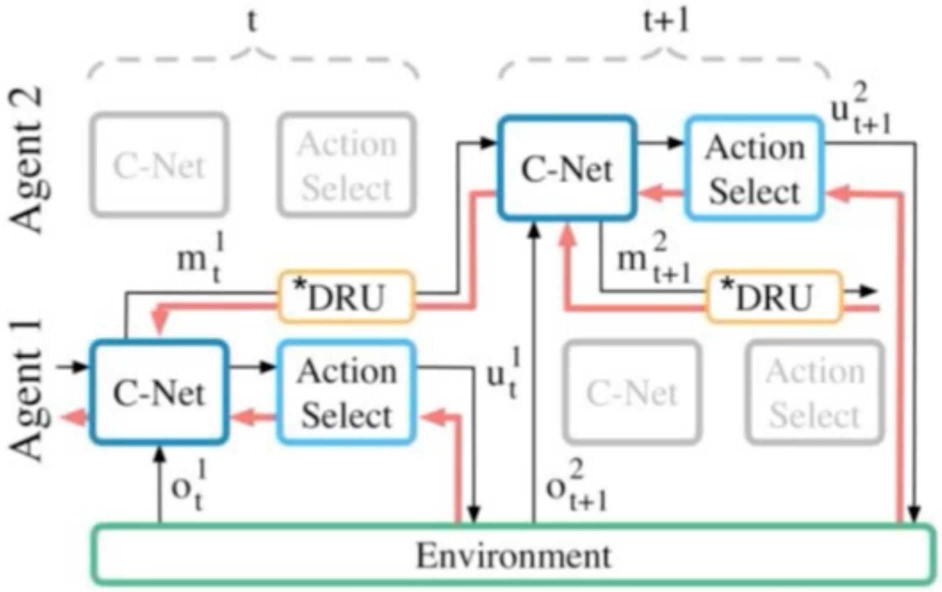


a) RIAL - RL based Communication

Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Reinforced Inter-Agent Learning

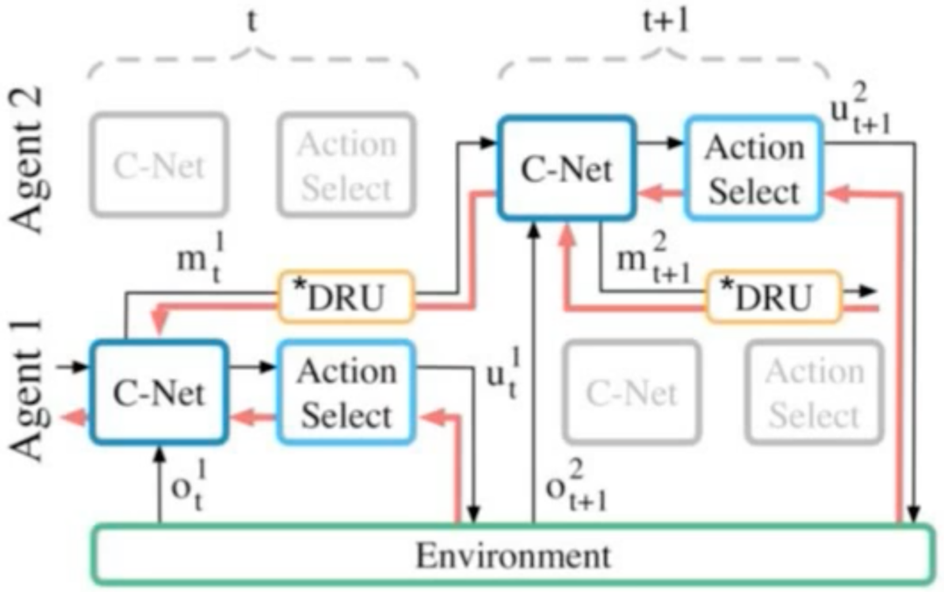
- **DIAL**
  - Gradient flows between agents: from the recipient to the sender
  - **Process**
    - Agent 1 receives a message
    - Agent 1 decides an action
    - Agent 1 receives DQN error
    - Agent 1 calculate the gradient of the loss with respect to the received message
    - Agent 1 sends the gradient it back to the sender (Agent2)
    - Agent 2 updates its weights to modify the message so that it reduces the DQN error of Agent1



Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Reinforced Inter-Agent Learning

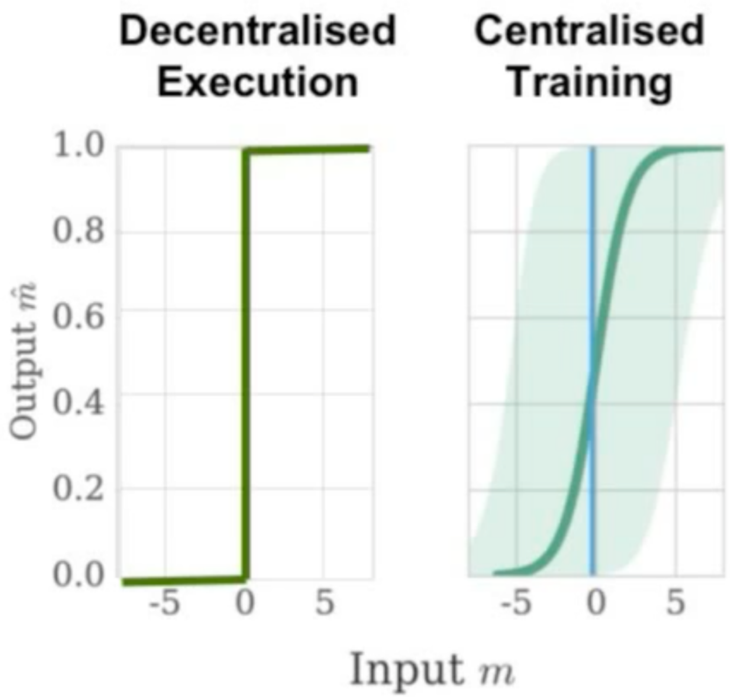
- **DIAL**
  - Gradient flows between agents: from the recipient to the sender
  - **Process**
    - Agent 1 receives a message
    - Agent 1 decides an action
    - Agent 1 receives DQN error
    - Agent 1 calculate the gradient of the loss with respect to the received message
    - Agent 1 sends the gradient it back to the sender (Agent2)
    - Agent 2 updates its weights to modify the message so that it reduces the DQN error of Agent1



Would this work?

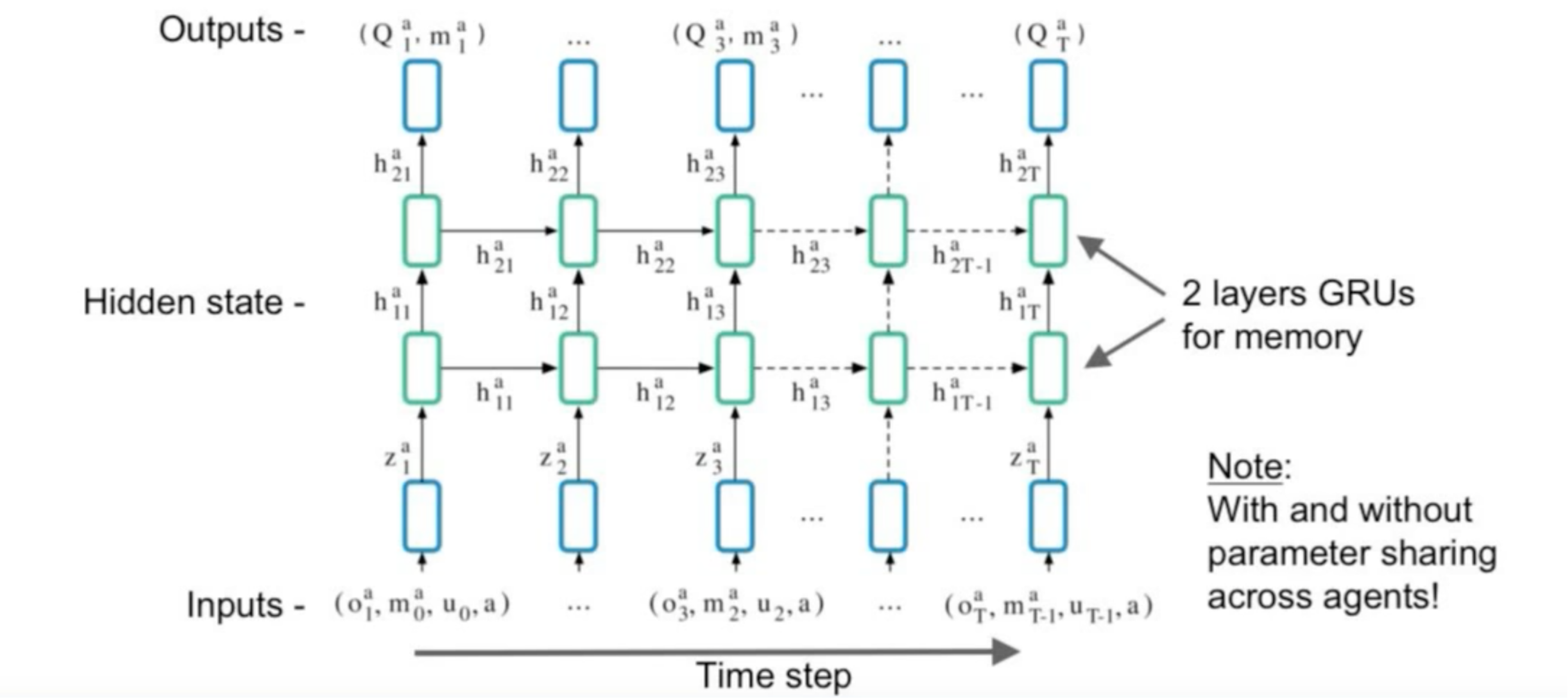
Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Representing messages



$$\mathrm{DRU}(m) = \begin{cases} \mathrm{Logistic}(\mathcal{N}(m, \sigma)), \text{ if training, else} \\ \mathbb{1}\{m > 0\} \end{cases}$$

Architecture



Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Experiments –Switch Riddle

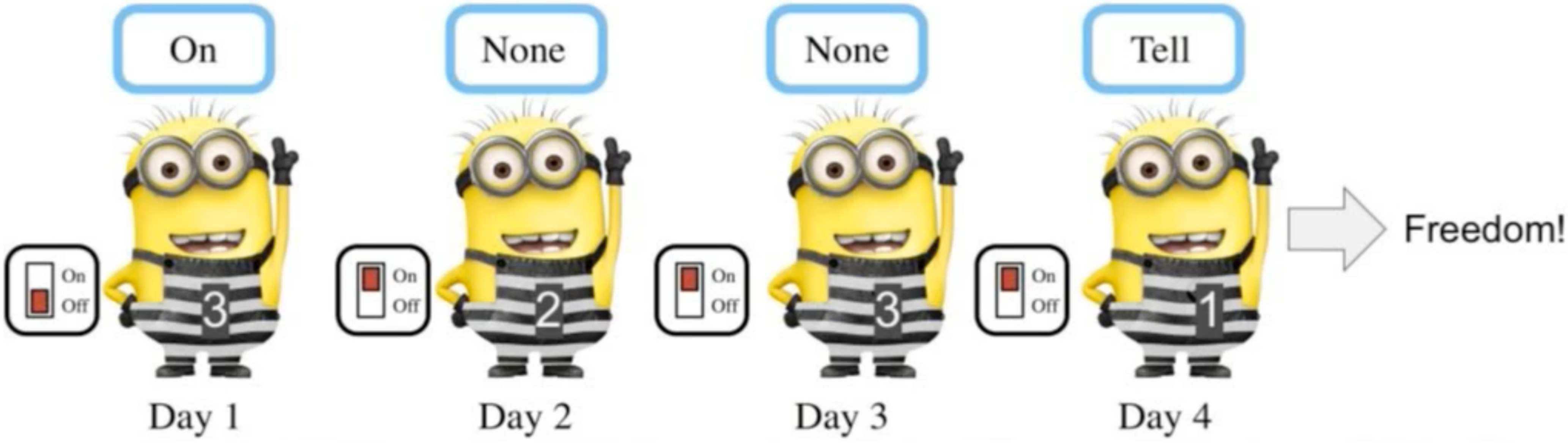
“One hundred prisoners have been newly ushered into prison.
The warden tells them that starting tomorrow, each of them will be placed in an isolated cells, unable to communicate among each others.
Each day, the warden will choose one of the prisoners uniformly at random with replacement, and place him in a central interrogation room containing only a light bulb with a toggle switch. The prisoner will be able to observe the current state of the light. If he wishes he can toggle the light bulb.
He also has the option of announcing that he believes all prisoners have visited the interrogation room at some point in time. If the announcement is true, all prisoners are set free, but if it is false, all prisoners are executed.
Thee warden leaves and the prisoners huddle to discuss their fate.
Can they agree on a protocol to guarantee their freedom?”

(Wu, 2002)

Action

Prisoner in IR

Switch

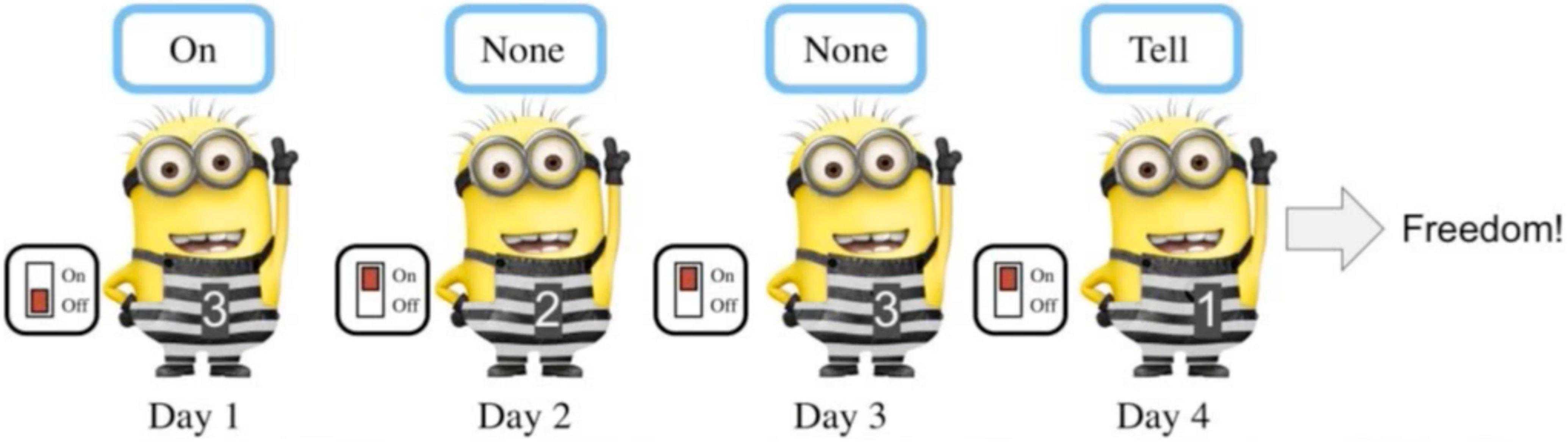


Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Action

Prisoner in IR

Switch



RL Setting

Multi-agent : N agents with 1 communication channel

State : N-bit array (has the i-th prisoner been to the IR)

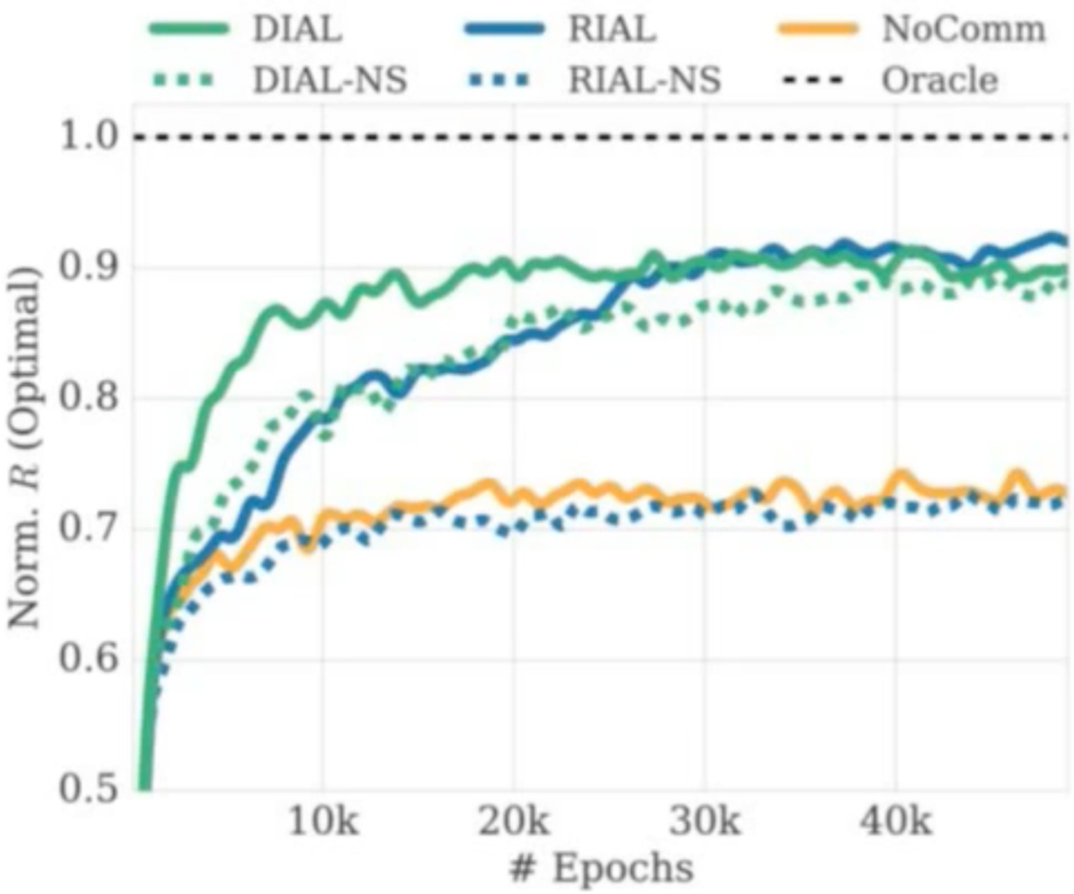
Action : Tell/ None/ Switch

Reward : +1 (freedom)/ 0 (episode expires)/ -1 (all die)

Observation: None/Switch

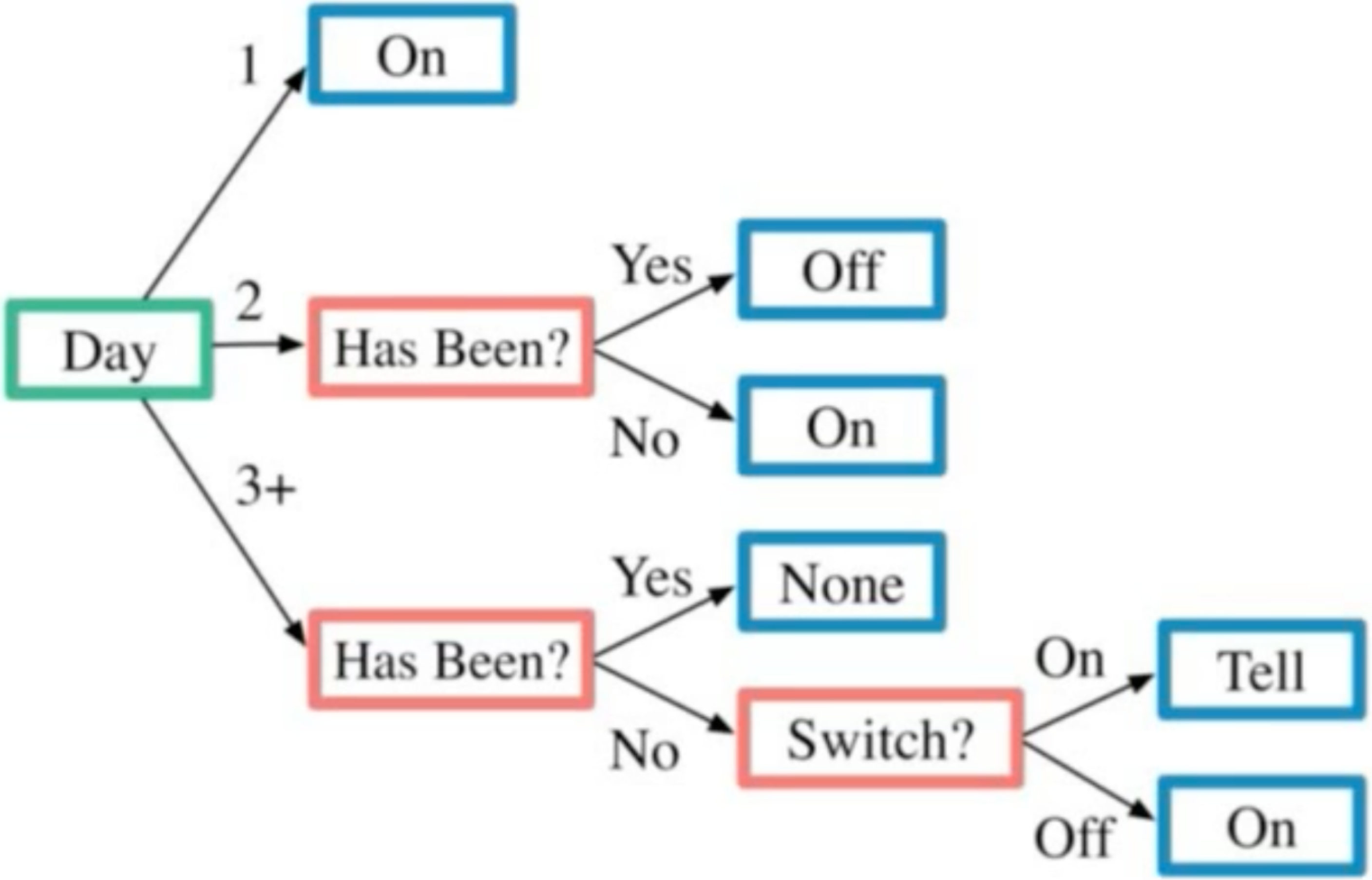
Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Did the Agents Learn to Communicate?



4 Mignions

Solution for 3 Agents



2 people: On
1 person: off

Talk by Jakob Foerster. Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

Conclusion

Conclusion

- Learning to learn the deep learning architecture
  - Two types of meta-learning algorithms
    - Evolution-inspired
    - Reinforcement learning

Conclusion

- Learning to learn the deep learning architecture
  - Two types of meta-learning algorithms
    - Evolution-inspired
    - Reinforcement learning
  - “No Free Lunch”: both with their own disadvantages
    - ✔ Possible to reach state-of-the-art on simple tasks
    - ✘ Requires lots of computational resources

Conclusion

- Learning to learn the deep learning architecture
  - Two types of meta-learning algorithms
    - Evolution-inspired
    - Reinforcement learning
  - "No Free Lunch": both with their own disadvantages
    - ✔ Possible to reach state-of-the-art on simple tasks
    - ✘ Requires lots of computational resources
- Learning to explore, seek knowledge, communicate
  - Using RL as a meta-learning algorithm seems promising

Conclusion

- Learning to learn the deep learning architecture
  - Two types of meta-learning algorithms
    - Evolution-inspired
    - Reinforcement learning
  - "No Free Lunch": both with their own disadvantages
    - ✔ Possible to reach state-of-the-art on simple tasks
    - ✘ Requires lots of computational resources
- Learning to explore, seek knowledge, communicate
  - Using RL as a meta-learning algorithm seems promising
- **Meta-learning is the next frontier in AI**

Thank you!

References:
Architecture Search

[Genetic CNN]

[HyperNetworks]

[Evolving Deep Neural Networks]

[Large-Scale Evolution of Image Classifiers]

[Random Search for Hyper-Parameter Optimization]

[Neural Architecture Search with Reinforcement Learning]

[Designing Neural Network Architectures using Reinforcement Learning]

[PathNet: Evolution Channels Gradient Descent in Super Neural Networks]

References:
Learning to Explore

Learning to Navigate in Complex Environments

Learning to Learn by Gradient Descent by Gradient Descent

Learning to Learn for Global Optimization of Black Box Functions

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning

Learning to Poke by Poking: Experiential Learning of Intuitive Physics

Learning to Perform Physics Experiments via Deep Reinforcement Learning

References:
Learning to Seek Knowledge

Learning to Perform Physics Experiments via Deep Reinforcement Learning

References:
Learning to Communicate

Learning to Communicate with Deep Multi-Agent Reinforcement Learning

Blogs & Talks

Metalearning, Scholarpedia

Taxonomy of Methods for Deep Meta Learning

RNN Symposium 2016: Ilya Sutskever - Meta Learning in the Universe

Learning to Communicate with Deep Multi-Agent Reinforcement Learning - Jakob Foerster