



Boulder

# Computer Vision; Image Classification; AutoML



[YouTube Playlist](#)

**Maziar Raissi**

**Assistant Professor**

Department of Applied Mathematics

University of Colorado Boulder

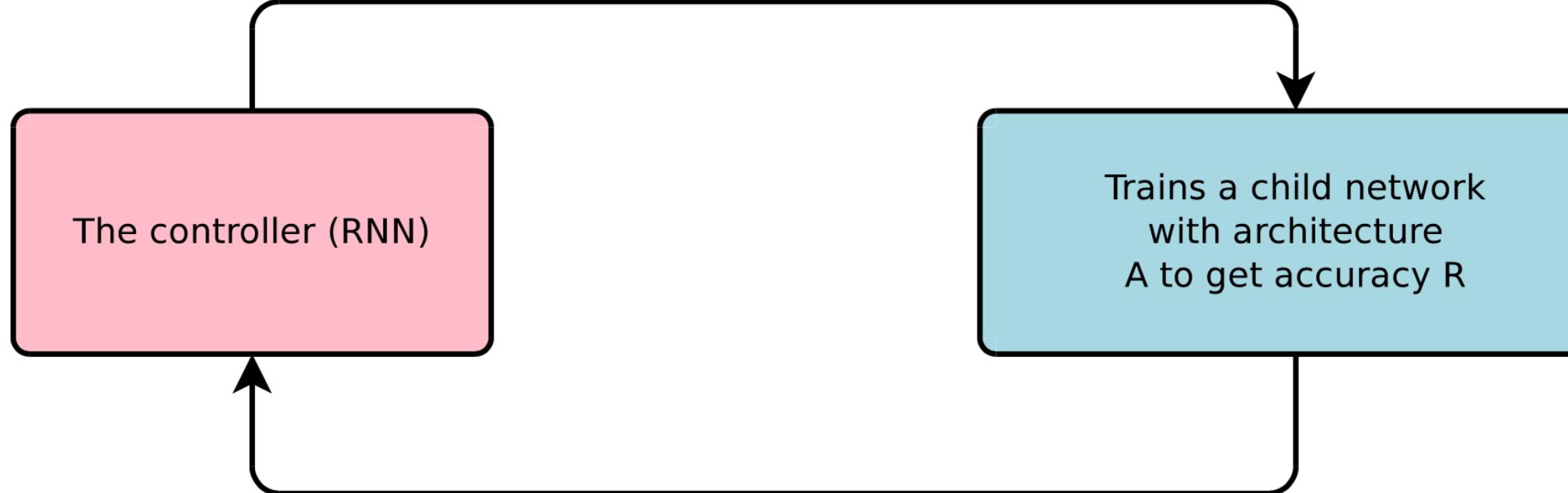
[maziar.raissi@colorado.edu](mailto:maziar.raissi@colorado.edu)



Boulder

# Neural Architecture Search With Reinforcement Learning

feature engineering → architecture engineering  
Sample architecture A with probability p



$$\nabla_{\theta_c} J(\theta_c) = \int \nabla_{\theta_c} P(a_{1:T}; \theta_c) R = \int \frac{\nabla_{\theta_c} P(a_{1:T}; \theta_c)}{P(a_{1:T}; \theta_c)} P(a_{1:T}; \theta_c) R = \int \nabla_{\theta_c} \log P(a_{1:T}; \theta_c) P(a_{1:T}; \theta_c) R$$

$$P(a_{1:T}; \theta_c) = \prod_{t=1}^T P(a_t | a_{1:t-1}; \theta_c) \implies \nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^T \mathbb{E}_{P(a_{1:T}; \theta_c)} [\nabla_{\theta_c} \log P(a_t | a_{1:t-1}; \theta_c) R]$$

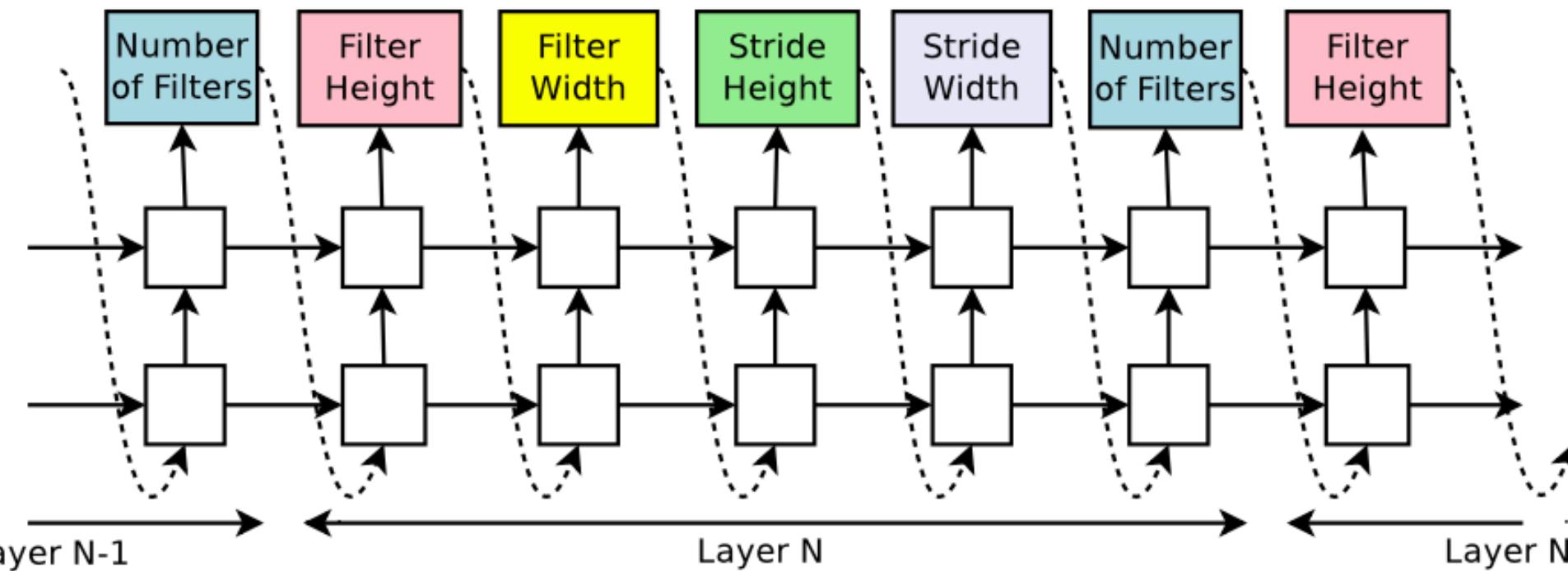
$$\nabla_{\theta_c} J(\theta_c) \approx \frac{1}{m} \sum_{k=1}^m \sum_{t=1}^T \nabla_{\theta_c} \log P(a_t | a_{1:t-1}; \theta_c) (R_k - b)$$

$m \rightarrow$  number of different architectures that the controller samples in one batch

$T \rightarrow$  number of hyperparameters the controller has to predict to design a neural network architecture

$R_k \rightarrow$  validation accuracy that the  $k$ -th neural network architecture achieves after being trained on a training dataset

$b \rightarrow$  baseline function (exponential moving average of the previous architecture accuracies)

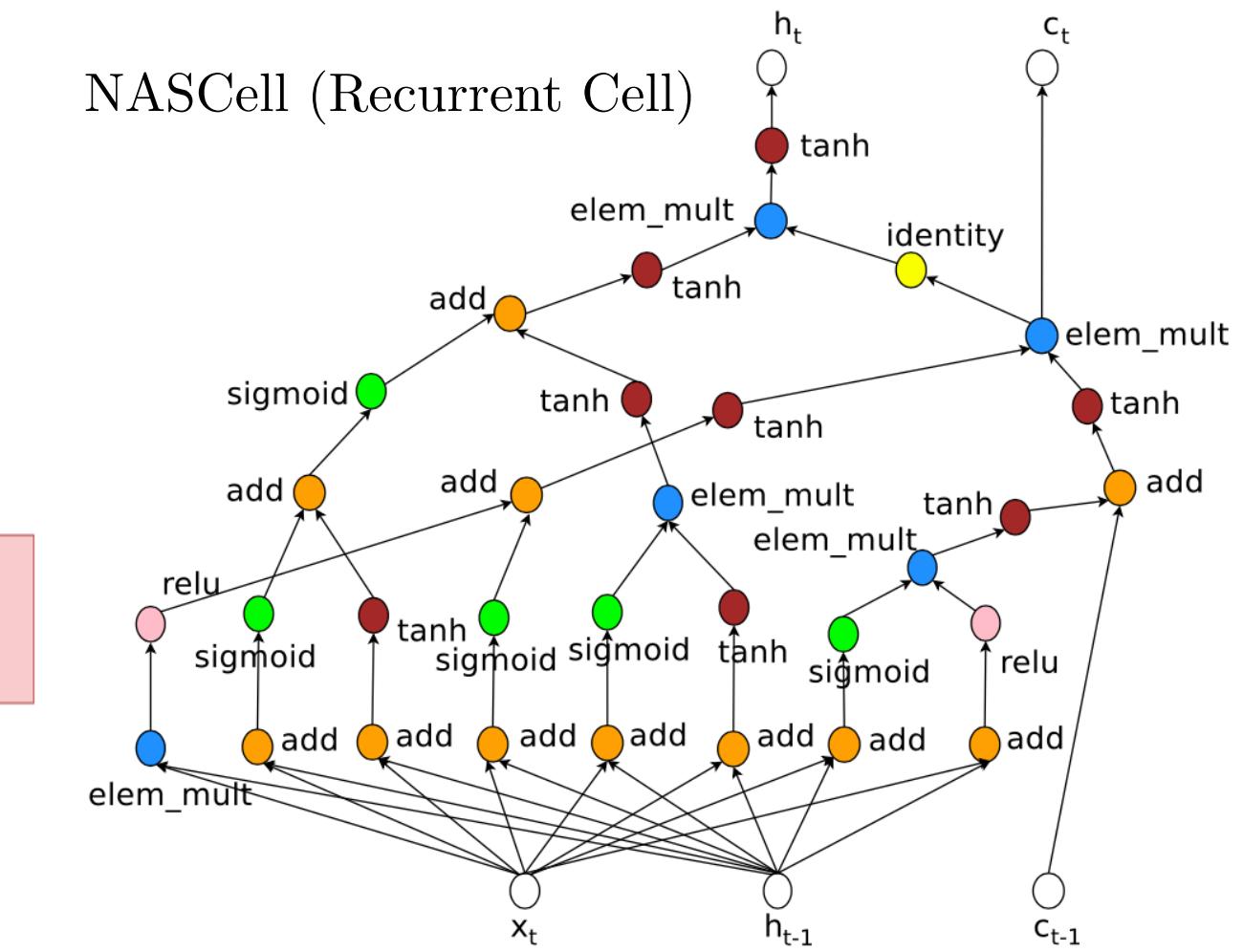
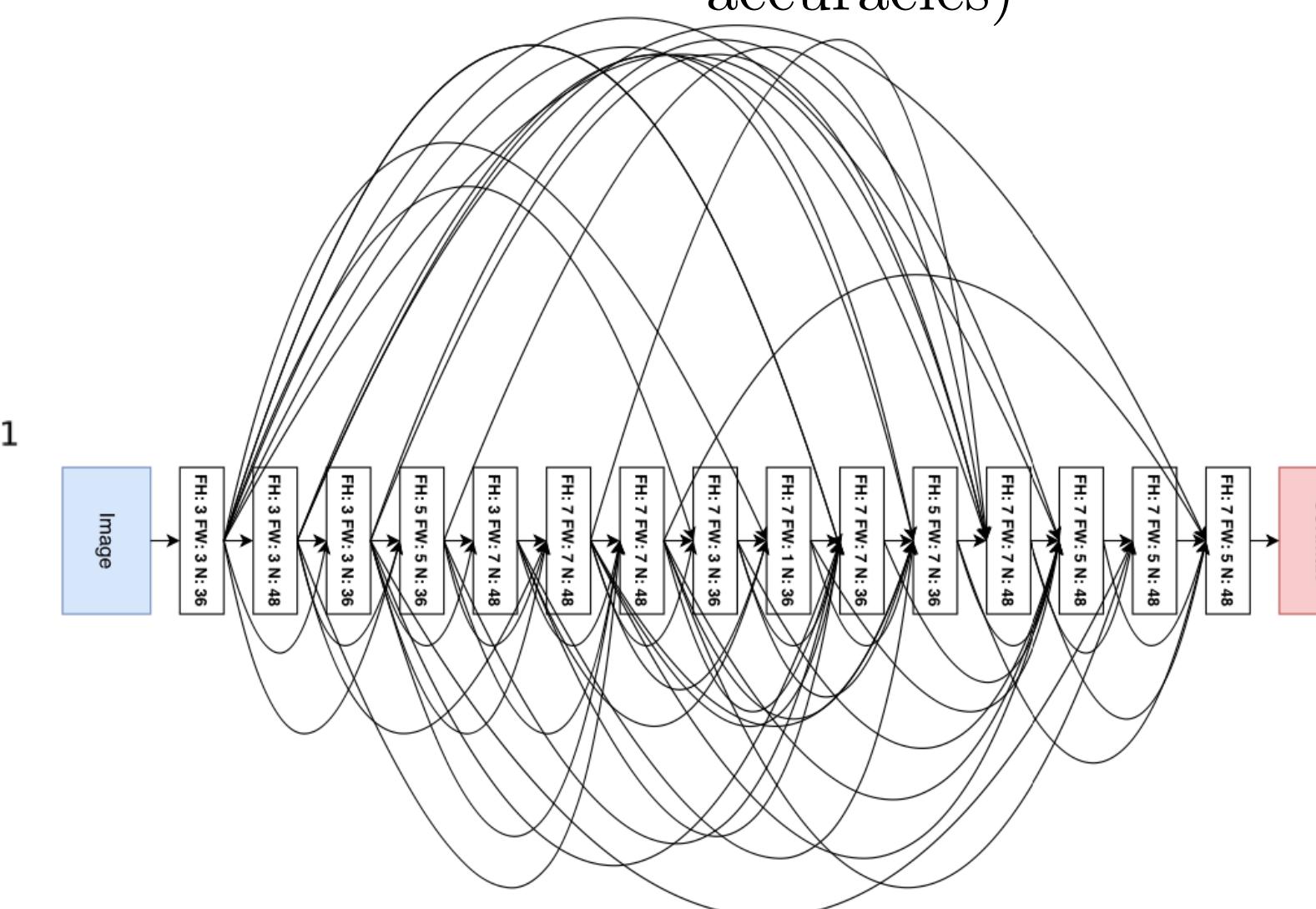


$\theta_c \rightarrow$  parameters of the controller RNN

$a_{1:T} \rightarrow$  list of actions (list of tokens that the controller predicts to design an architecture for a child network)

$R \rightarrow$  reward signal (accuracy achieved by the child network on a held-out dataset)

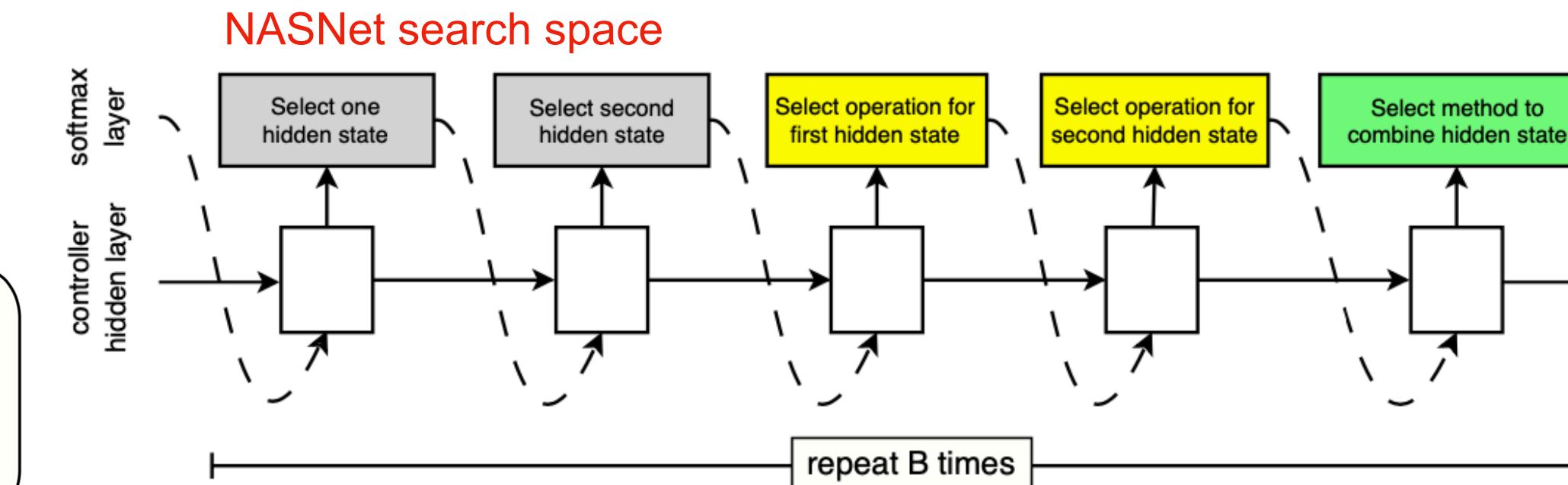
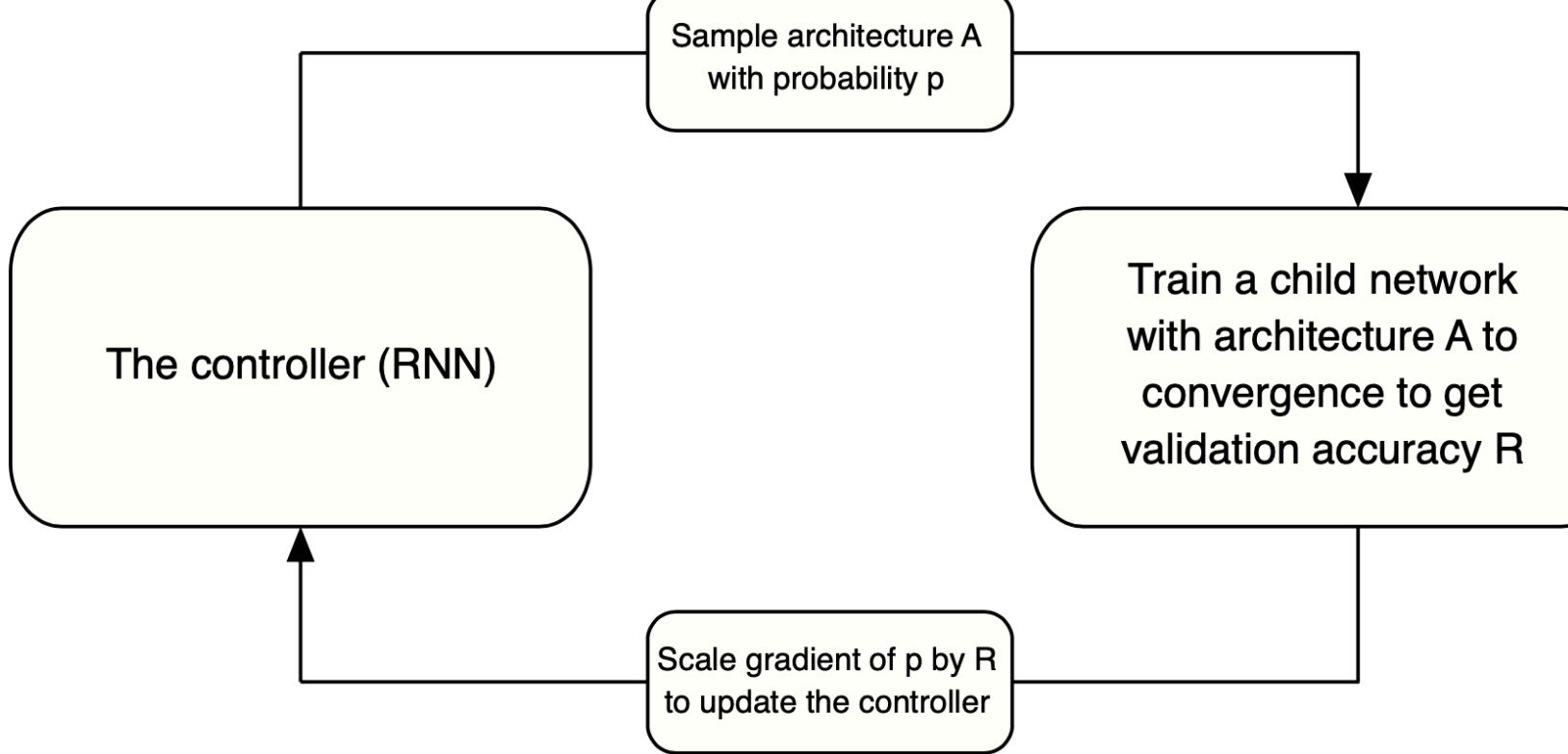
$J(\theta_c) = \mathbb{E}_{P(a_{1:T}; \theta_c)} [R] \rightarrow$  expected reward



# Learning Transferable Architectures for Scalable Image Recognition


[YouTube Video](#)

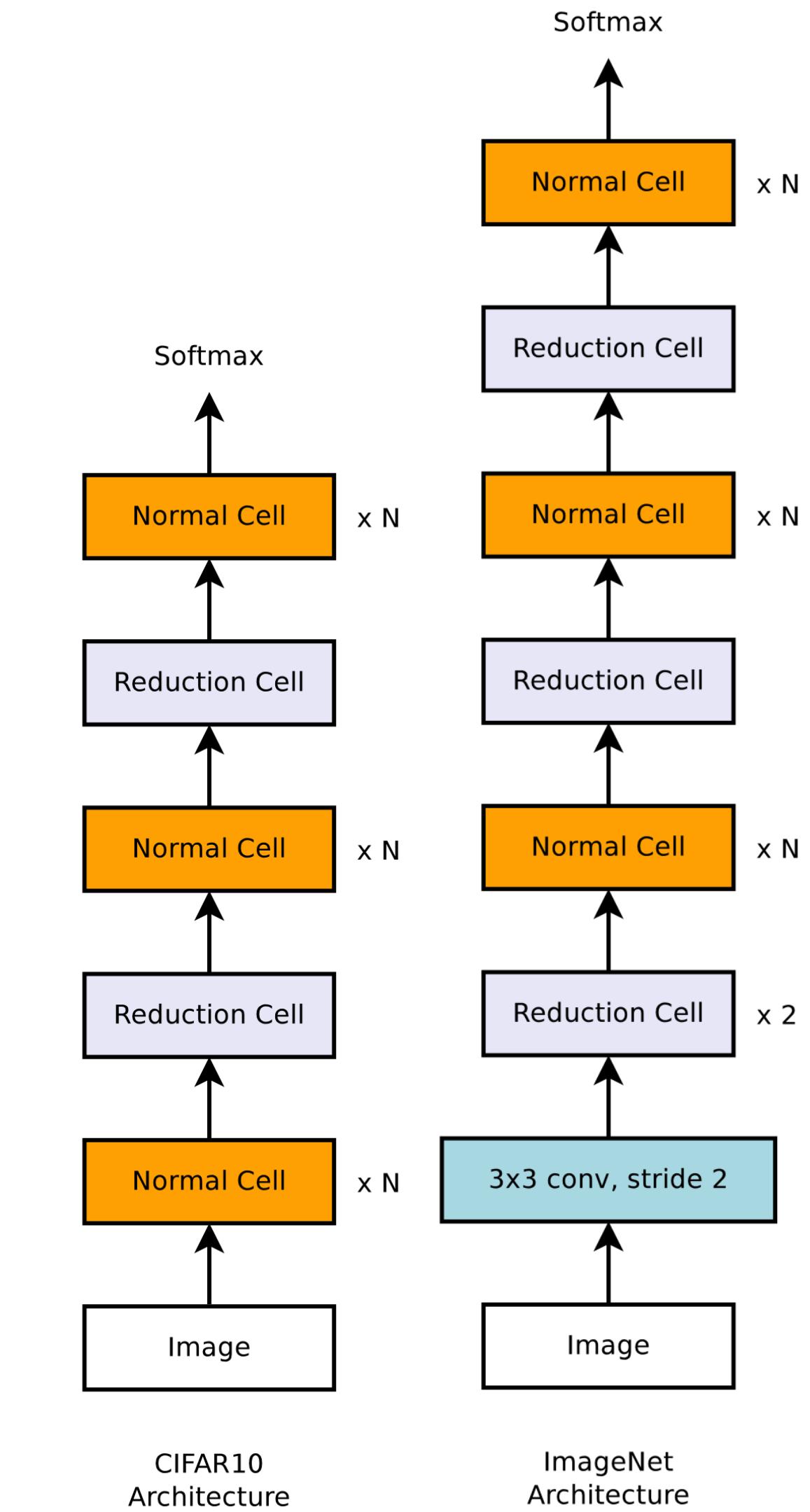
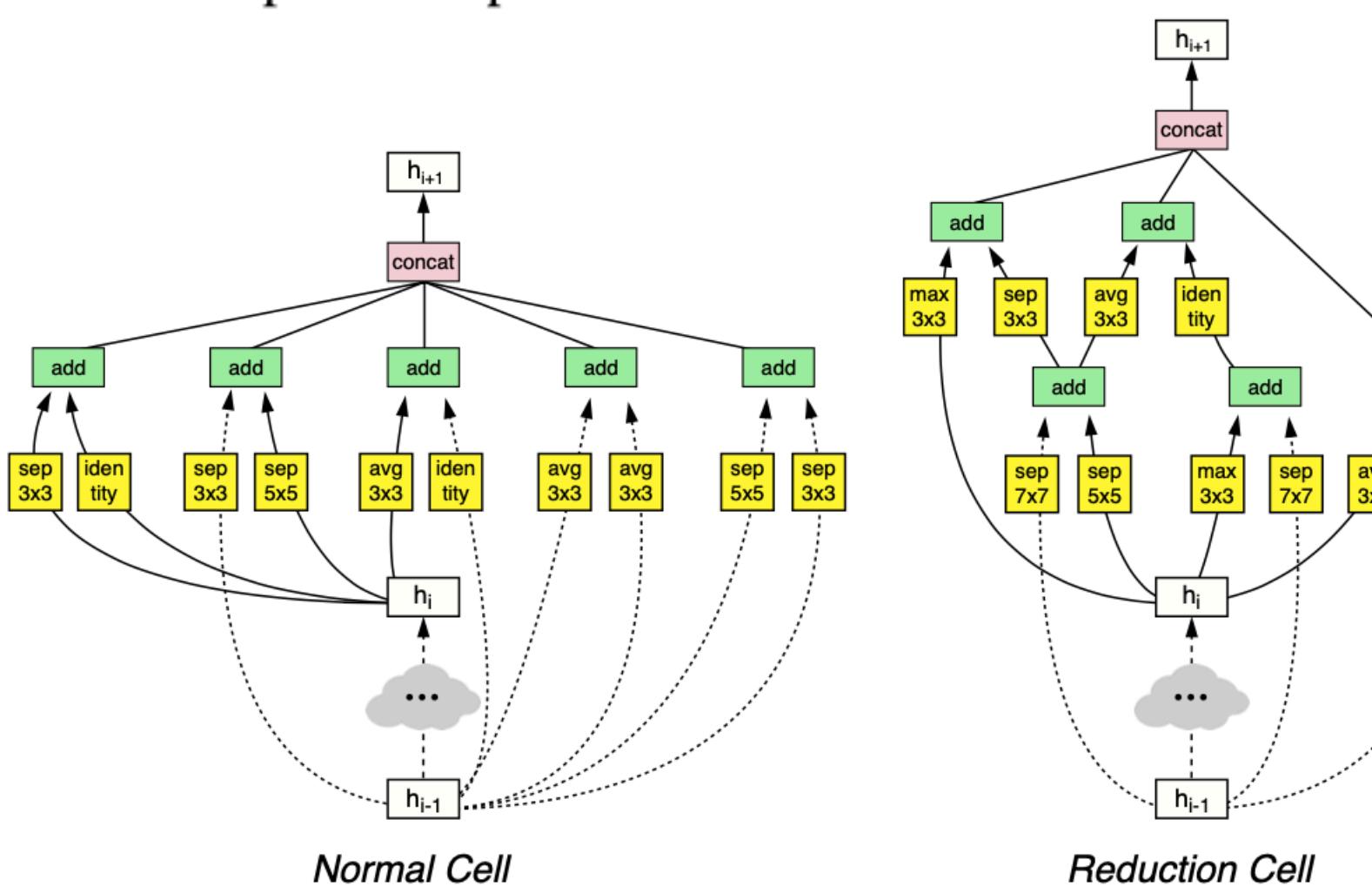
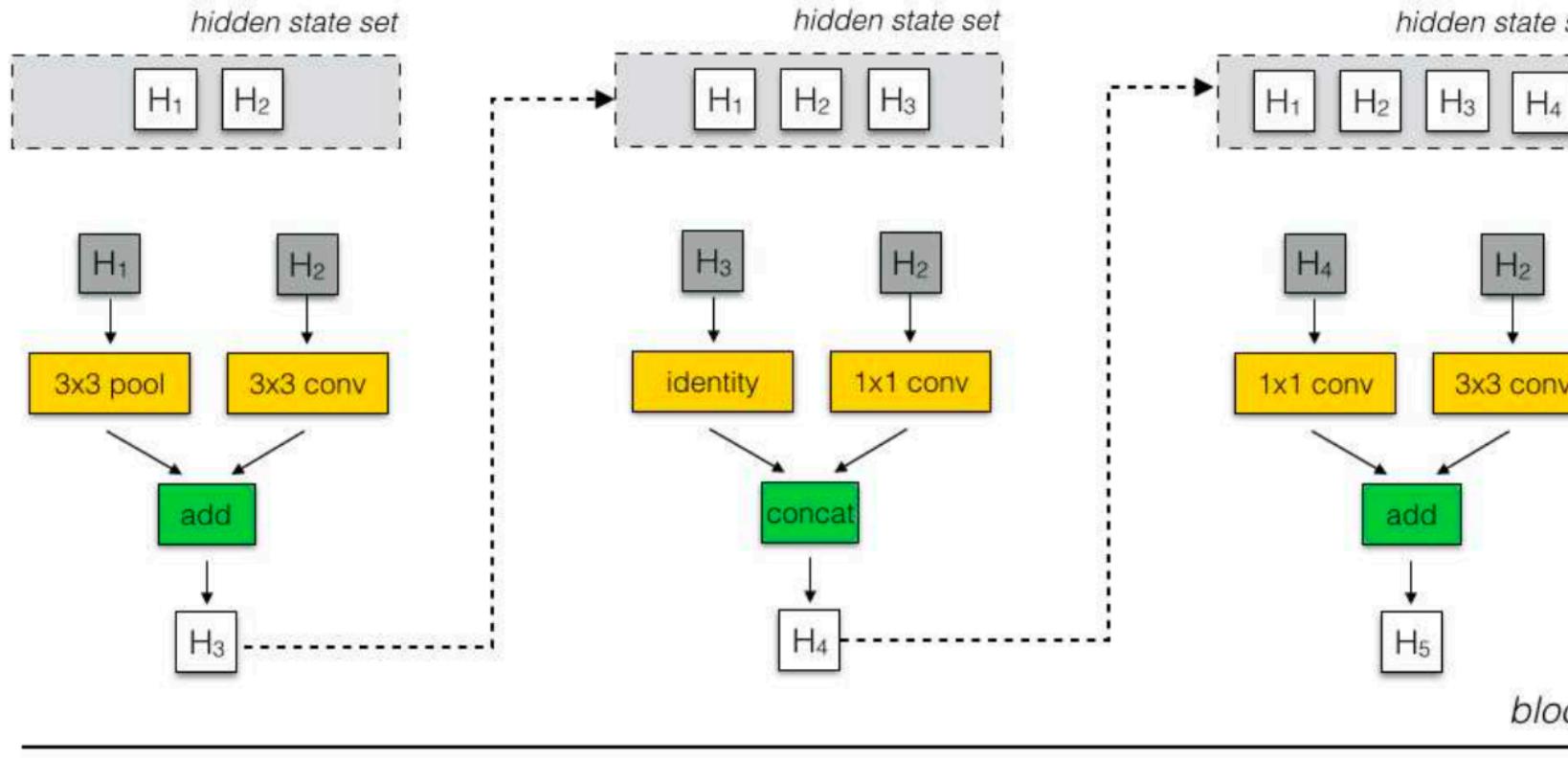
feature engineering → architecture engineering



- identity
- 1x3 then 3x1 convolution
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 3x3 max pooling
- 5x5 max pooling
- 7x7 max pooling
- 1x1 convolution
- 3x3 convolution
- 5x5 depthwise-separable conv
- 7x7 depthwise-separable conv

**Normal Cell:** convolutional cells that return a feature map of the same dimension

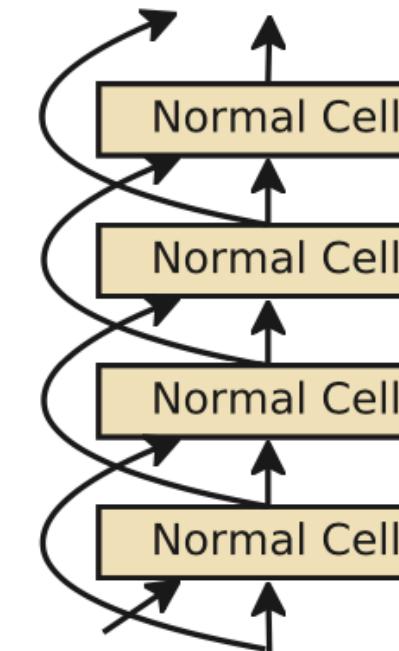
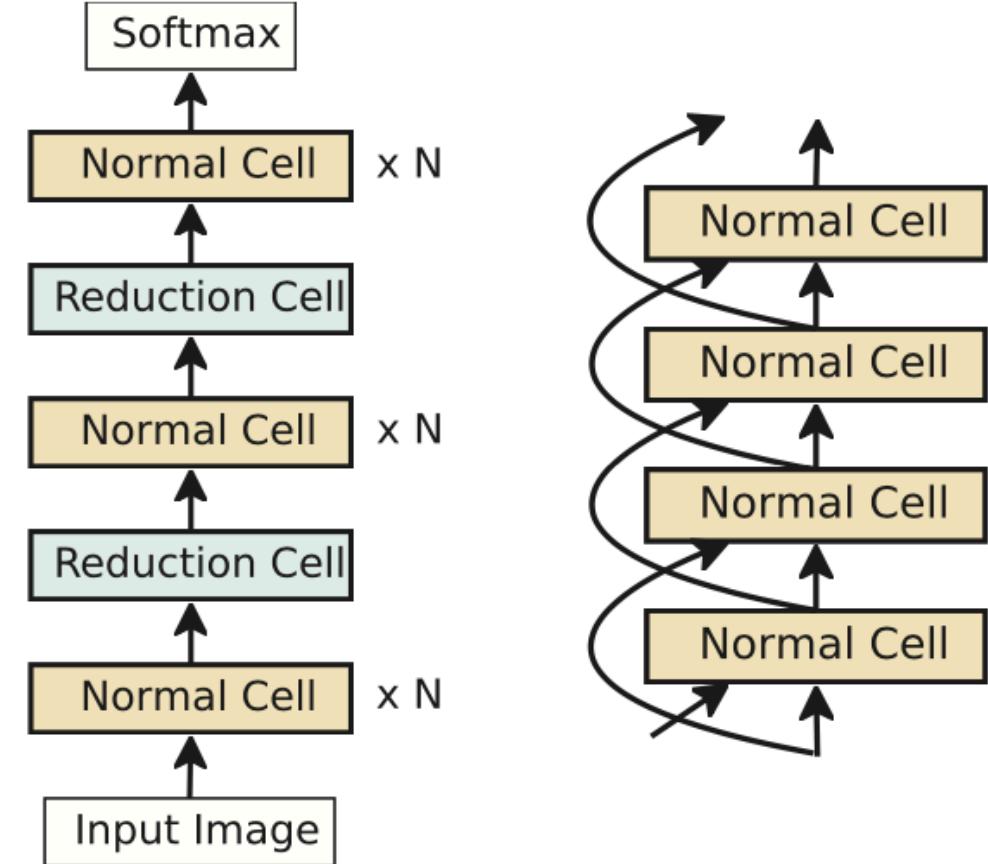
**Reduction Cell:** convolutional cells that return a feature map where the feature map height and width is reduced by a factor of two



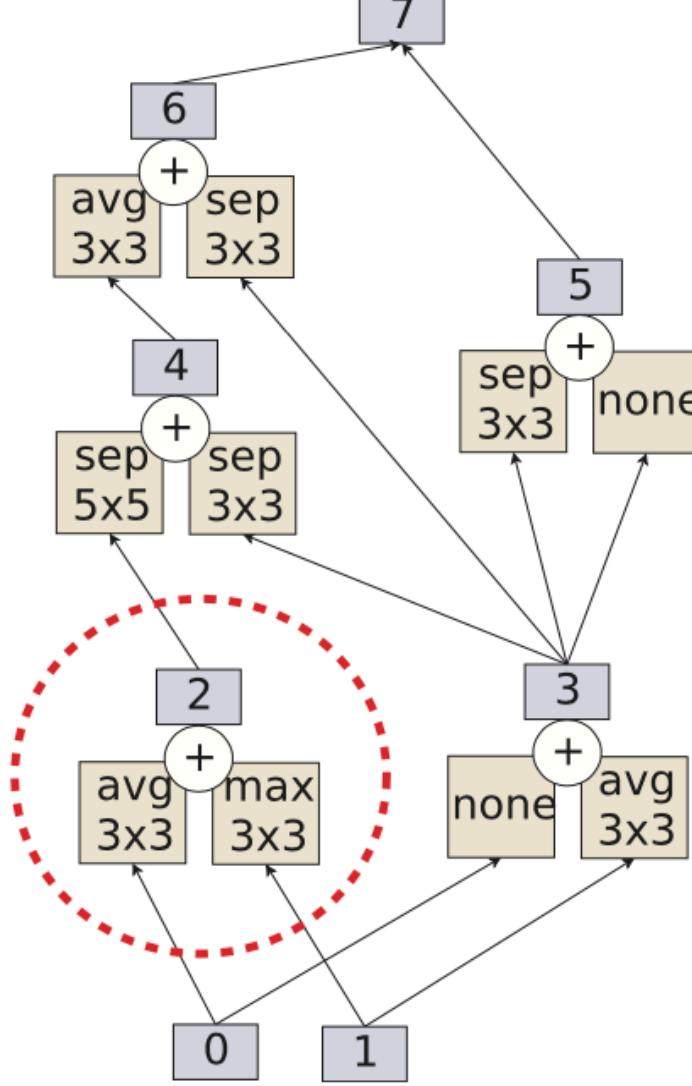
# Regularized Evolution for Image Classifier Architecture Search


[YouTube Video](#)

NASNet Search Space outer structure



NASNet Search Space cell structure

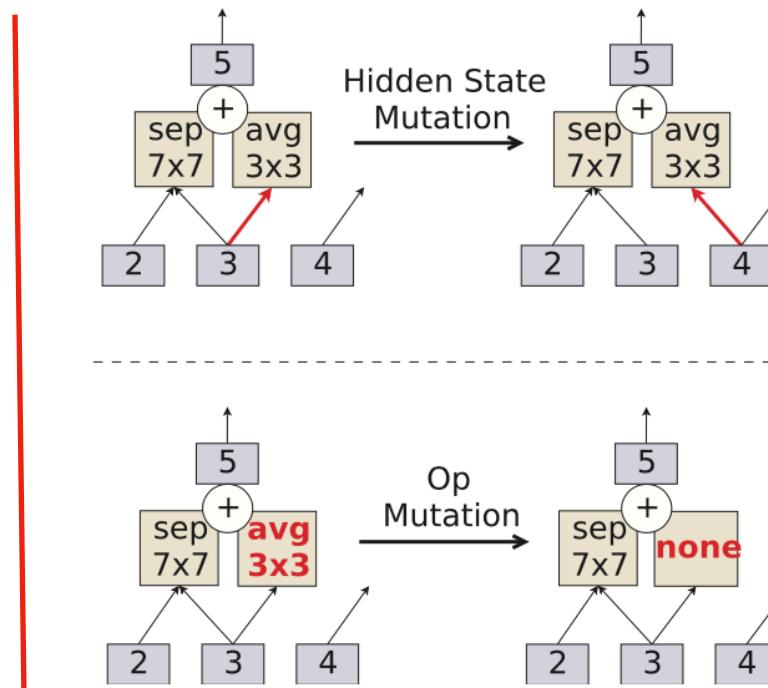


Evolution: Random Search + Selection

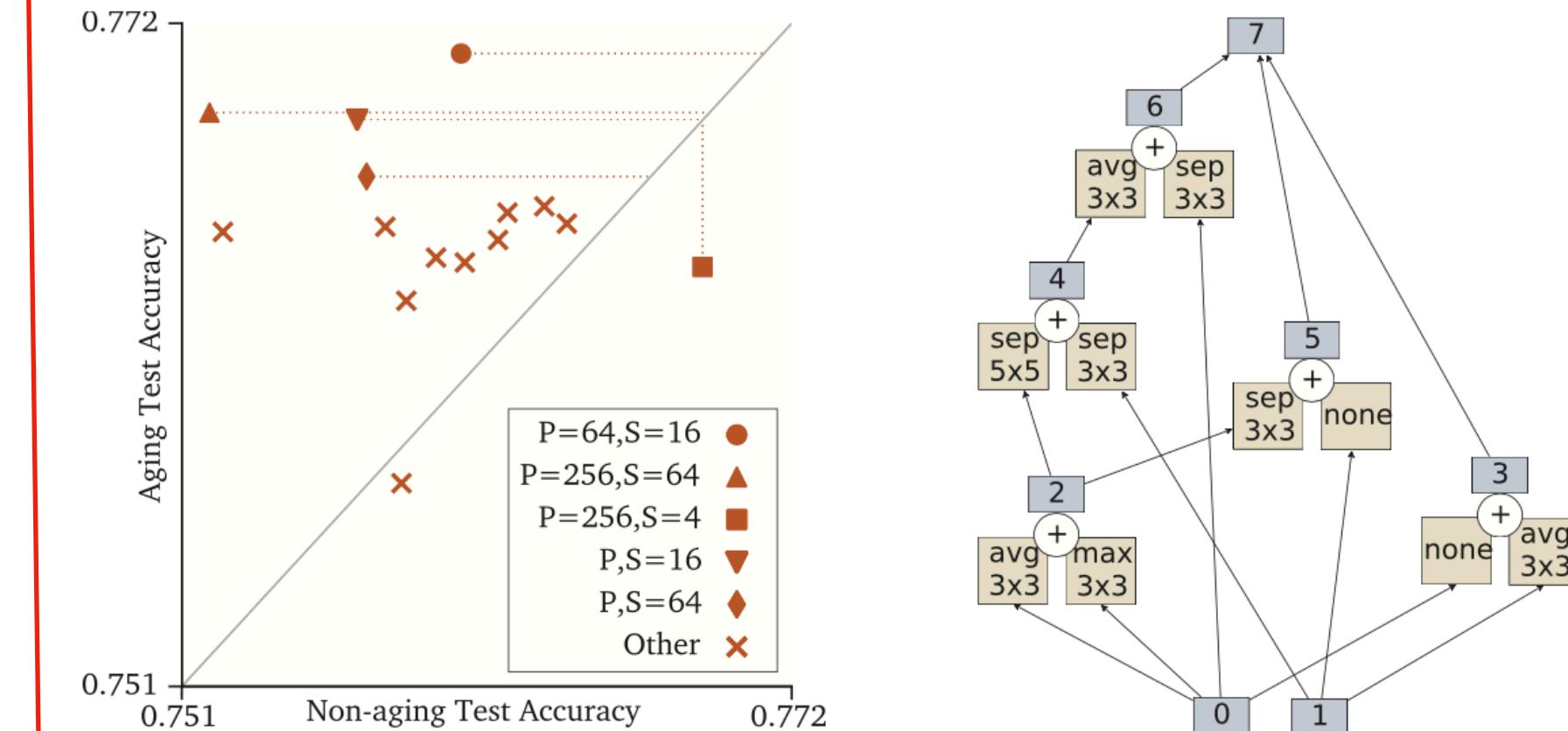
## Algorithm 1 Aging Evolution (*i.e.* Regularized Evolution)

```

population ← empty queue           ▷ The population.
history ← ∅                         ▷ Will contain all models.
while |population| < P do          ▷ Initialize population.
    model.arch ← RANDOMARCHITECTURE()
    model.accuracy ← TRAINANDEVAL(model.arch)
    add model to right of population
    add model to history
end while
while |history| < C do            ▷ Evolve for C cycles.
    sample ← ∅                         ▷ Parent candidates.
    while |sample| < S do
        candidate ← random element from population
        ▷ The element stays in the population.
        add candidate to sample
    end while
    parent ← highest-accuracy model in sample
    child.arch ← MUTATE(parent.arch)
    child.accuracy ← TRAINANDEVAL(child.arch)
    add child to right of population
    add child to history
    remove dead from left of population
    discard dead
end while
return highest-accuracy model in history
  
```



Model	# Parameters	# Multiply-Adds	Top-1 / Top-5 Accuracy (%)
Incep-ResNet V2 (Szegedy et al. 2017)	55.8M	13.2B	80.4 / 95.3
ResNeXt-101 (Xie et al. 2017)	83.6M	31.5B	80.9 / 95.6
PolyNet (Zhang et al. 2017)	92.0M	34.7B	81.3 / 95.8
Dual-Path-Net-131 (Chen et al. 2017)	79.5M	32.0B	81.5 / 95.8
GeNet-2 (Xie and Yuille 2017)*	156M	-	72.1 / 90.4
Block-QNN-B (Zhong, Yan, and Liu 2018)*	-	-	75.7 / 92.6
Hierarchical (Liu et al. 2018b)*	64M	-	79.7 / 94.8
NASNet-A (Zoph et al. 2018)	88.9M	23.8B	82.7 / 96.2
PNASNet-5 (Liu et al. 2018a)	86.1M	25.0B	82.9 / 96.2
AmoebaNet-A (N=6, F=190)*	86.7M	23.1B	82.8 / 96.1
AmoebaNet-A (N=6, F=448)*	469M	104B	83.9 / 96.6





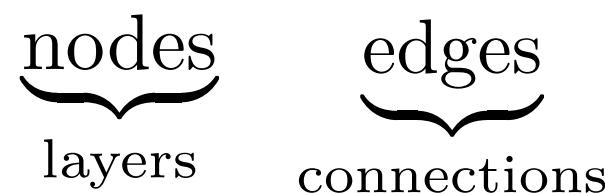
Boulder

# Evolving Deep Neural Networks

## NeuroEvolution of Augmenting Topologies (NEAT)

- 1) A population of chromosomes (each represented by a graph) with minimal complexity is created.
- 2) Over generations, structure (i.e. neurons and weights) is added to the graph incrementally through mutation.
- 3) During crossover, historical markings are used to determine how genes of two chromosomes can be lined up.
- 4) The population is divided into species (i.e. subpopulations) based on a similarity metric.
- 5) Each species grows proportionally to its fitness and evolution occurs separately in each species.

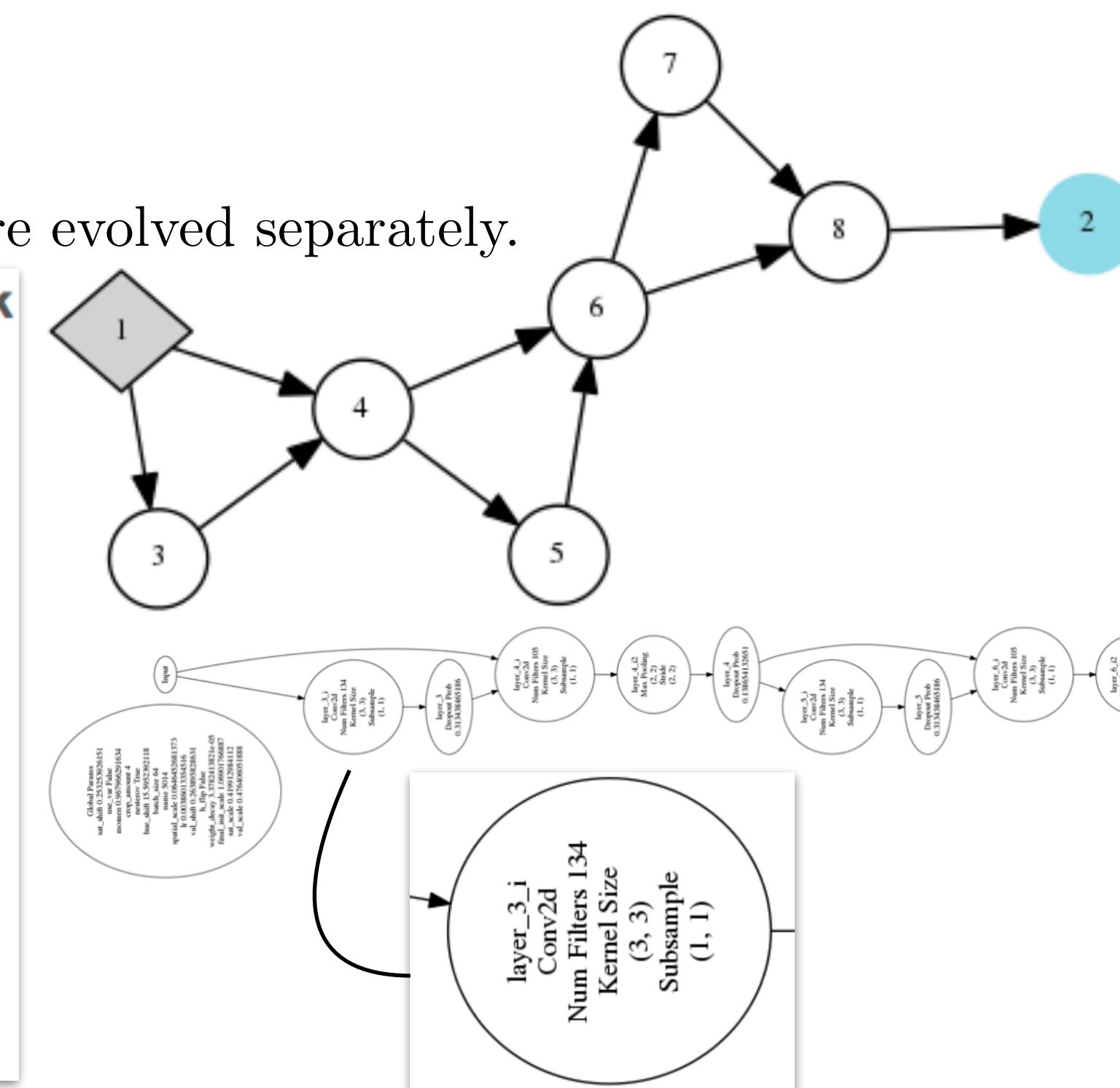
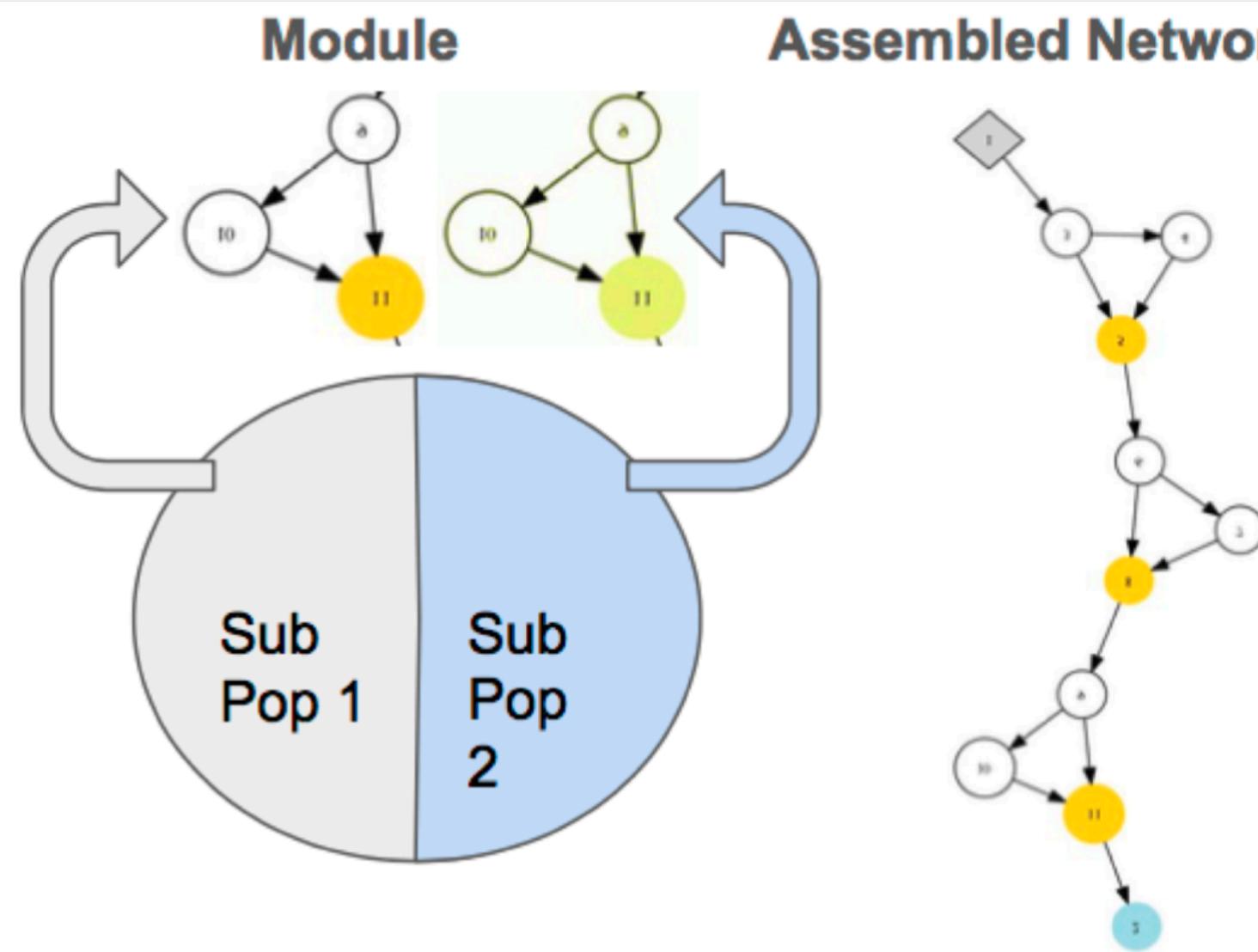
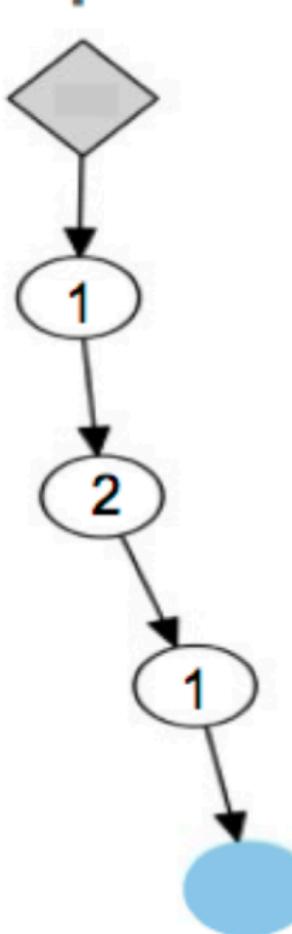
## DeepNeat



## Coevolution DeepNEAT (CoDeepNEAT)

In CoDeepNEAT, two populations of modules and blueprints are evolved separately.

### Blueprint



## Node and global hyperparameters evolved in the CIFAR-10 domain.

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}
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}



# Efficient Neural Architecture Search via Parameter Sharing

Boulder

NAS: 450 GPUs for 3 – 4 days ( $\approx 32,400 - 43,200$  GPU-Hours)

ENAS: 16 GPU-Hours (1000 $\times$  faster)

The computational bottleneck of NAS is the training of each child model to convergence, only to measure its accuracy whilst throwing away all the trained weights.

Force all child models to share weights!

The controller RNN makes two sets of decisions:

- 1) two previous nodes to be used as inputs to the current node and
- 2) two operations to apply to the two sampled nodes.

$\theta \rightarrow$  parameters of the controller LSTM

$\omega \rightarrow$  shared parameters of the child models

$$\nabla_{\omega} \mathbb{E}_{\mathbf{m} \sim \pi(\mathbf{m}; \theta)} [\mathcal{L}(\mathbf{m}; \omega)] \approx \frac{1}{M} \sum_{i=1}^M \nabla_{\omega} \mathcal{L}(\mathbf{m}_i, \omega) \rightarrow \text{fixed } \theta$$

$\underbrace{\text{model}}$      $\underbrace{\text{standard cross-entropy loss}}$

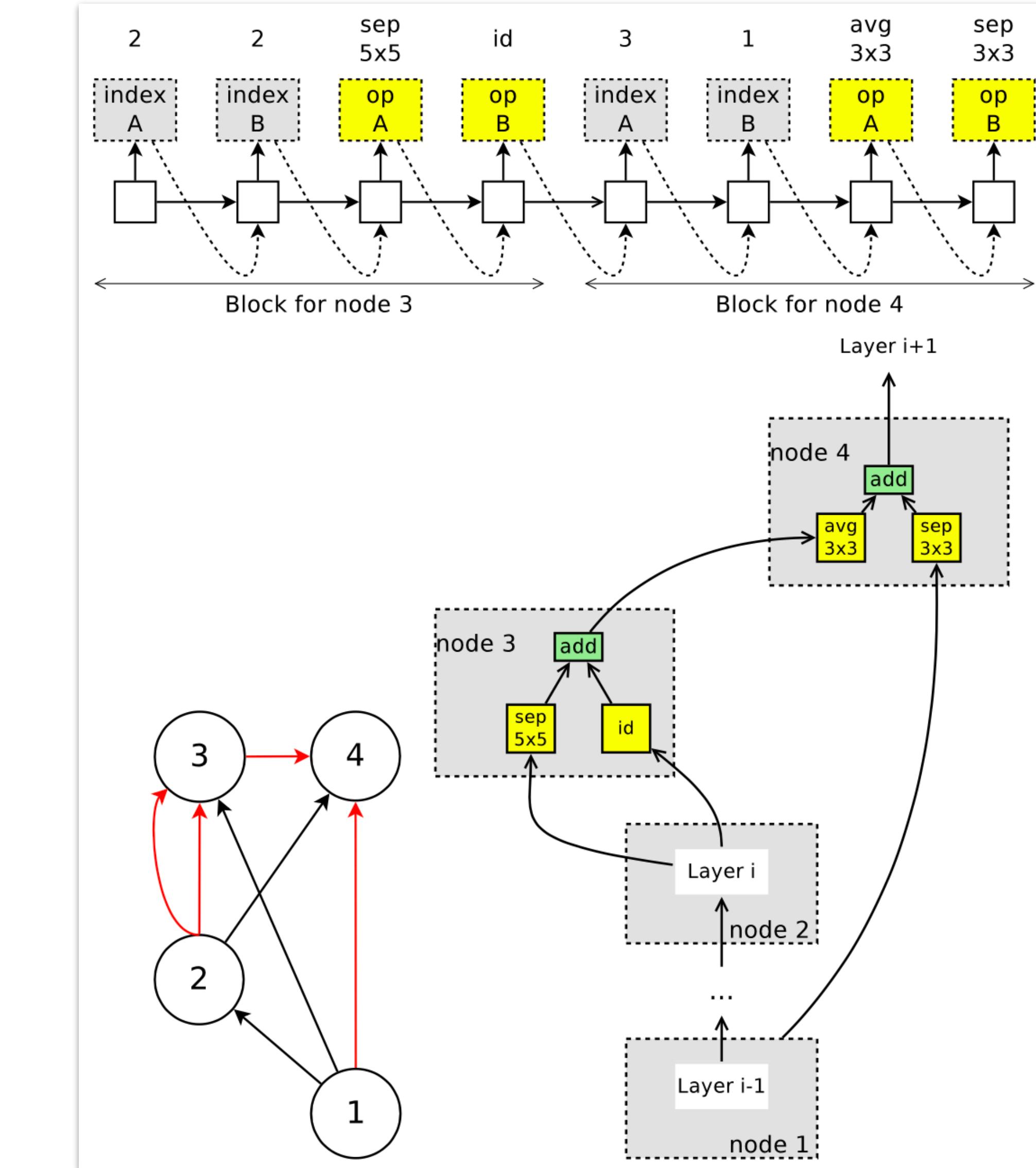
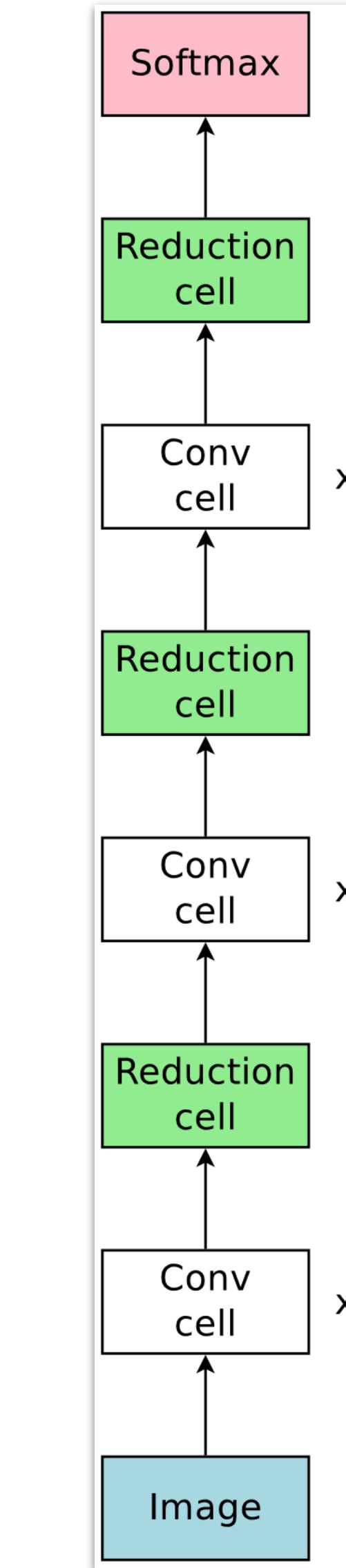
$M = 1$

$$\mathbb{E}_{\mathbf{m} \sim \pi(\mathbf{m}; \theta)} [\mathcal{R}(\mathbf{m}, \omega)] \rightarrow \text{fixed } \omega$$

$\underbrace{\text{reward (e.g., accuracy)}}$

Method	GPUs	Times (days)	Params (million)	Error (%)
Hierarchical NAS (Liu et al., 2018)	200	1.5	61.3	3.63
Micro NAS + Q-Learning (Zhong et al., 2018)	32	3	–	3.60
Progressive NAS (Liu et al., 2017)	100	1.5	3.2	3.63
NASNet-A (Zoph et al., 2018)	450	3-4	3.3	3.41
NASNet-A + CutOut (Zoph et al., 2018)	450	3-4	3.3	<b>2.65</b>
ENAS + micro search space	1	0.45	4.6	3.54
ENAS + micro search space + CutOut	1	0.45	4.6	<b>2.89</b>

Classification errors of ENAS and baselines on CIFAR-10.





Boulder

# DARTS: Differentiable Architecture Search

– 2000 GPU days of Reinforcement Learning (RL)

– 3150 GPU days of Evolution

– 1.5 to 4 GPU days

## Search Space

Cell → a directed acyclic graph consisting of an ordered sequence of  $N$  nodes

$x^{(i)}$  → node (a latent representation such as a feature map in CNNs)

$o^{(i,j)}$  → edge (some operation that transforms  $x^{(i)}$ )

$$x^{(j)} = \sum_{i < j} o^{(i,j)}(x^{(i)})$$

Each cell has two input nodes and a single output node

Inputs to convolutional cells:

Cell outputs in the previous two layers

Inputs to recurrent cells:

Input at the current step and the state carried from the previous step

Zero operation: Lack of connection between two nodes

## Continuous Relaxation and Optimization

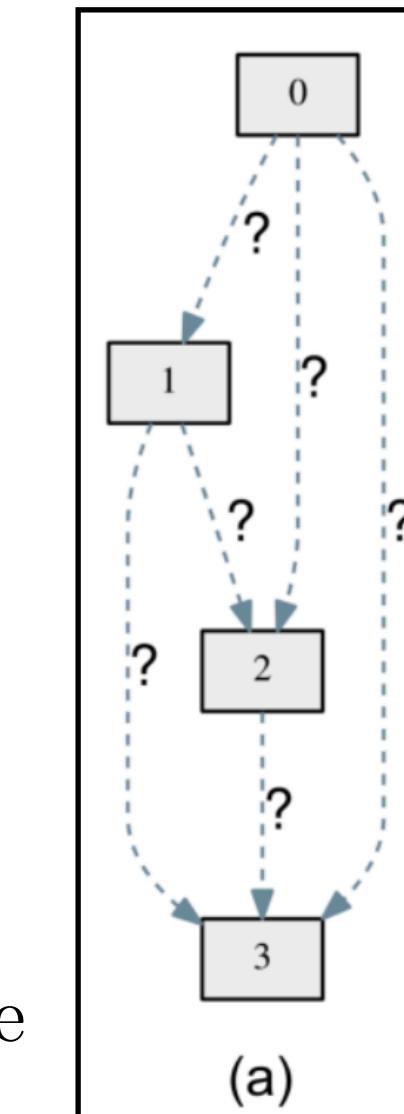
$\mathcal{O}$  → set of candidate operations

(e.g., conv, max pooling, zero)

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$

$\alpha^{(i,j)} \in \mathbb{R}^{|\mathcal{O}|}$  → operation mixing weight

$\alpha = \{\alpha^{(i,j)}\}$  → encoding the architecture



## Bilevel Optimization Problem

$$\min_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha)$$

$$\text{s.t. } w^*(\alpha) = \operatorname{argmin}_w \mathcal{L}_{train}(w, \alpha)$$

$w$  → lower-level variable

$\alpha$  → upper-level variable

## Approximate Architecture Gradient

$$\nabla_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha)$$

$$\approx \nabla_{\alpha} \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$$

$$\nabla_{\alpha} \mathcal{L}_{val}(w', \alpha) - \xi \nabla_{\alpha,w}^2 \mathcal{L}_{train}(w, \alpha) \nabla_{w'} \mathcal{L}_{val}(w', \alpha)$$

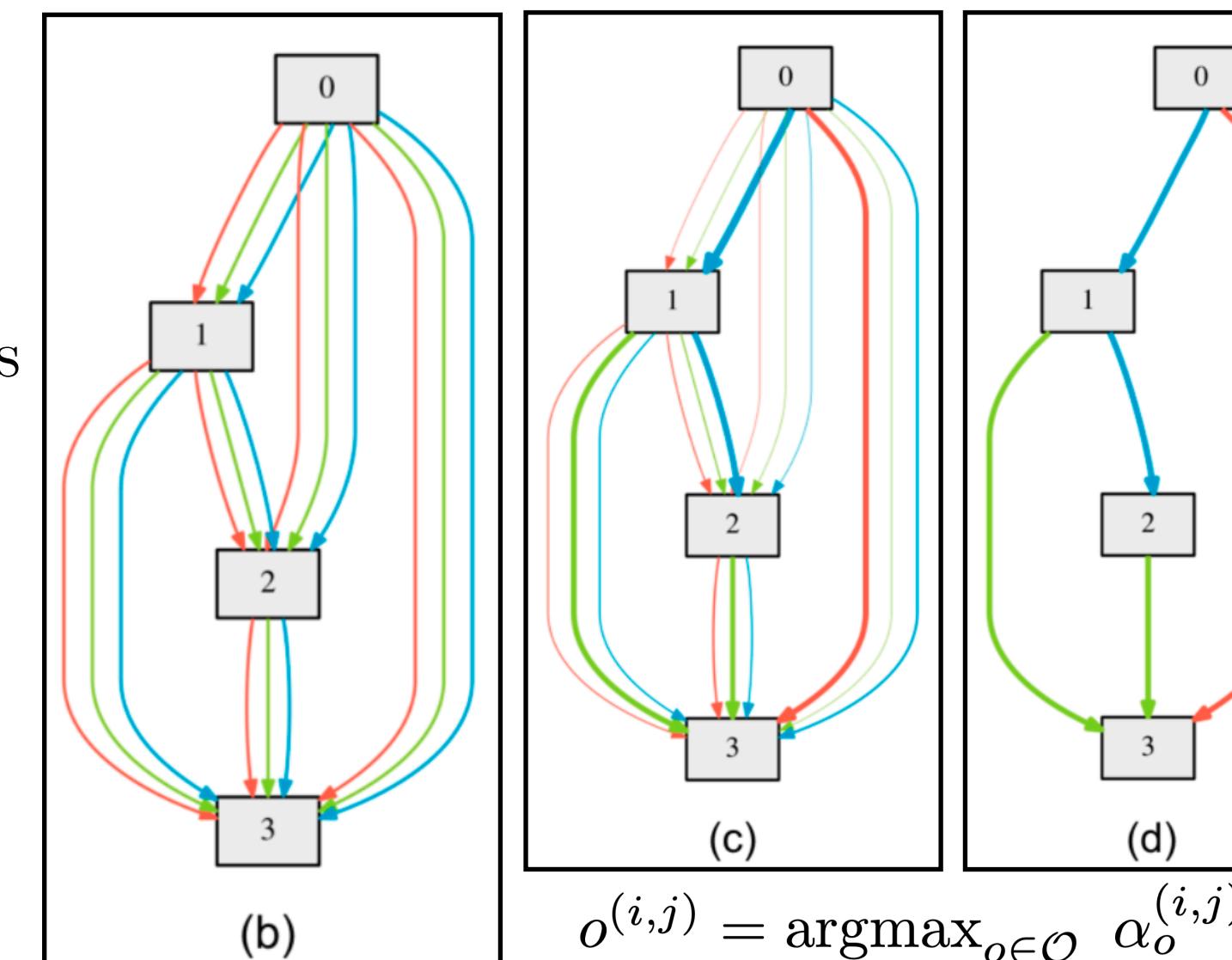
Finite Difference

## Algorithm 1: DARTS – Differentiable Architecture Search

Create a mixed operation  $\bar{o}^{(i,j)}$  parametrized by  $\alpha^{(i,j)}$  for each edge  $(i, j)$   
while not converged do

1. Update architecture  $\alpha$  by descending  $\nabla_{\alpha} \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$  ( $\xi = 0$  if using first-order approximation)
2. Update weights  $w$  by descending  $\nabla_w \mathcal{L}_{train}(w, \alpha)$

Derive the final architecture based on the learned  $\alpha$ .



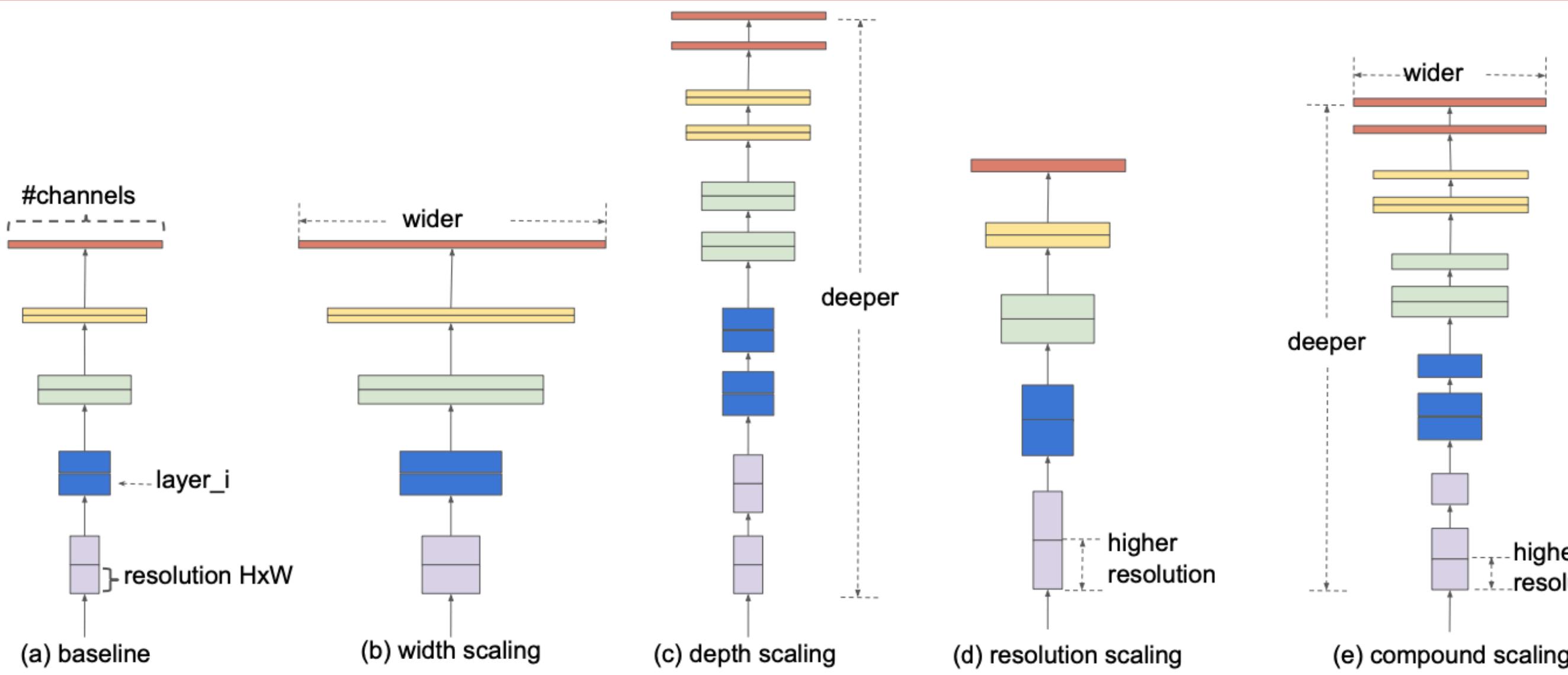
CIFAR-10 → ImageNet \*\*\* Penn Treebank → WikiText-2  
Comparison with state-of-the-art image classifiers on ImageNet in the mobile setting.

Architecture	Test Error (%)	Params (M)	+x (M)	Search Cost (GPU days)	Search Method
	top-1	top-5			
Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	1448	–
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	569	–
ShuffleNet 2× ( $g = 3$ ) (Zhang et al., 2017)	26.3	–	~5	524	–
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	564	2000 RL
NASNet-B (Zoph et al., 2018)	27.2	8.7	5.3	488	2000 RL
NASNet-C (Zoph et al., 2018)	27.5	9.0	4.9	558	2000 RL
AmoebaNet-A (Real et al., 2018)	25.5	8.0	5.1	555	3150 evolution
AmoebaNet-B (Real et al., 2018)	26.0	8.5	5.3	555	3150 evolution
AmoebaNet-C (Real et al., 2018)	24.3	7.6	6.4	570	3150 evolution
PNAS (Liu et al., 2018a)	25.8	8.1	5.1	588	~225 SMBO
DARTS (searched on CIFAR-10)	26.7	8.7	4.7	574	4 gradient-based

Comparison with state-of-the-art language models on WT2.

Architecture	Perplexity	Params (M)	Search Cost (GPU days)	Search Method
	valid	test		
LSTM + augmented loss (Inan et al., 2017)	91.5	87.0	28	–
LSTM + continuous cache pointer (Grave et al., 2016)	–	68.9	–	–
LSTM (Merity et al., 2018)	69.1	66.0	33	–
LSTM + skip connections (Melis et al., 2018)	69.1	65.9	24	–
LSTM + 15 softmax experts (Yang et al., 2018)	66.0	63.3	33	–
ENAS (Pham et al., 2018b) <sup>†</sup> (searched on PTB)	72.4	70.4	33	0.5 RL
DARTS (searched on PTB)	71.2	69.6	33	1 gradient-based

# EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks


[YouTube Video](#)


$d \rightarrow$  depth scaling coefficient

$w \rightarrow$  width scaling coefficient

$r \rightarrow$  resolution scaling coefficient

$\max_{d,w,r}$  accuracy

s.t.  $\text{memory} \leq \text{target\_memory}$

$\text{flops} \leq \text{target\_flops}$

## Compound Scaling

$\phi \rightarrow$  compound coefficient

$$d = \alpha^\phi, w = \beta^\phi, r = \gamma^\phi$$

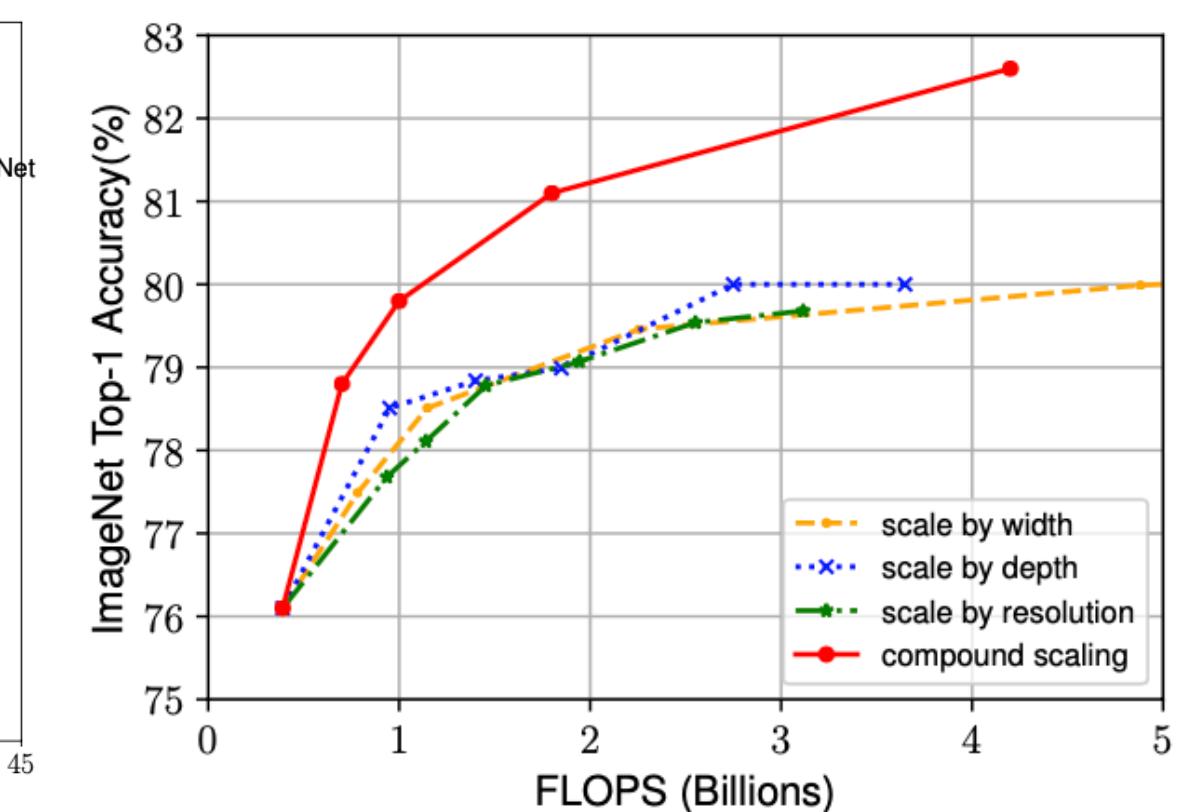
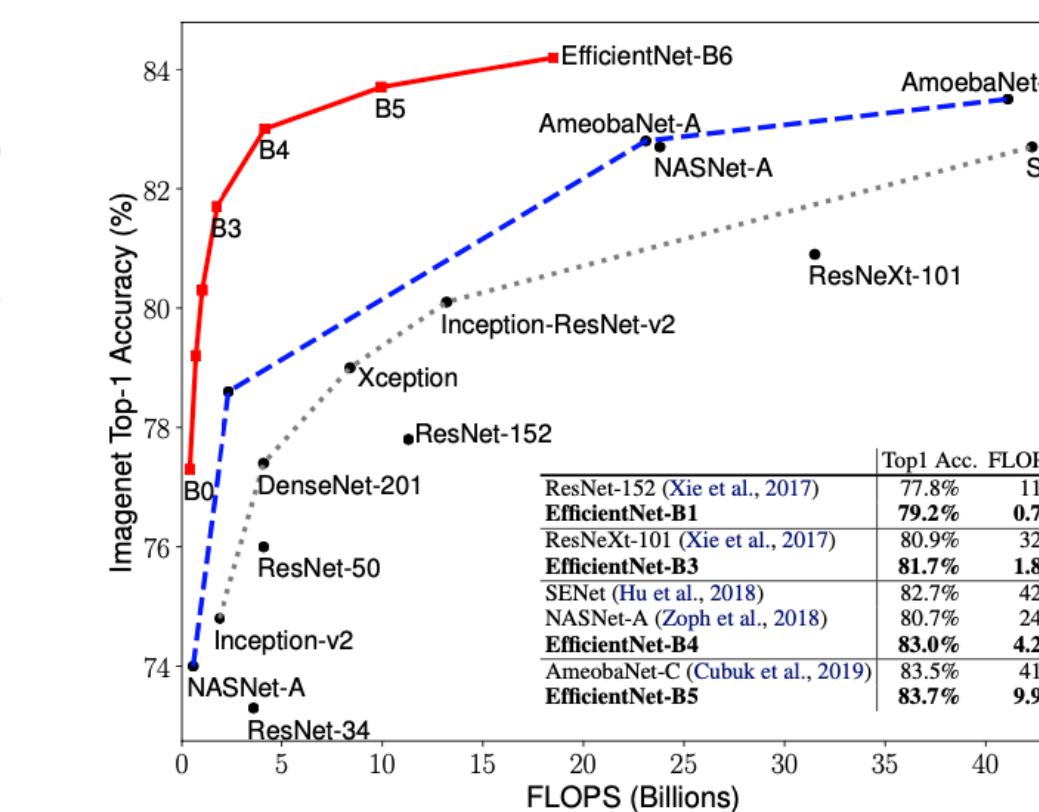
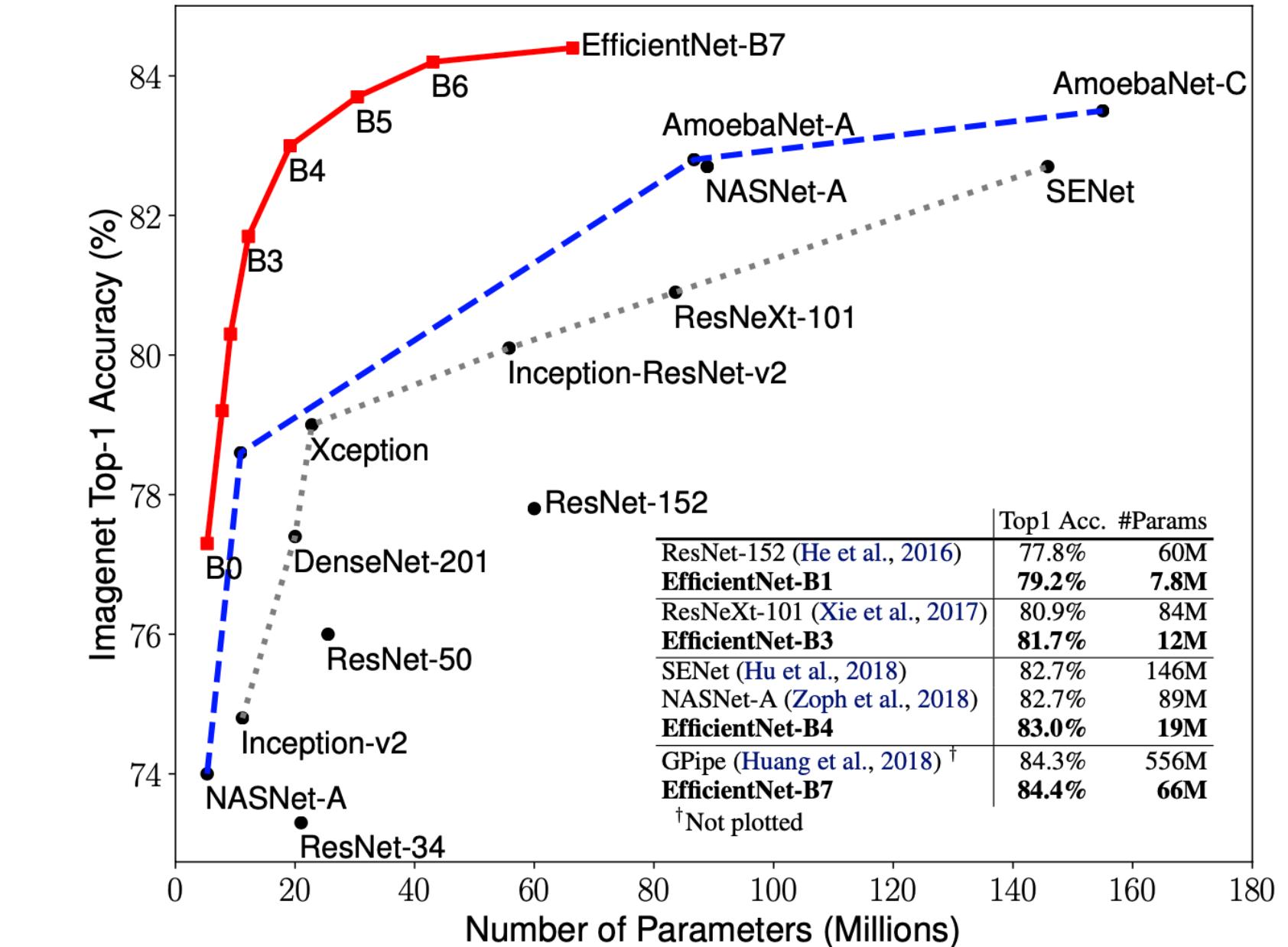
$$\alpha\beta^2\gamma^2 \approx 2, \underbrace{\alpha \geq 1, \beta \geq 1, \gamma \geq 1}_{\text{small grid search}}$$

## EfficientNet

$$\text{max accuracy} \cdot (\text{flops}/\text{target\_flops})^\omega$$

$\omega = -0.07 \rightarrow$  controls the tradeoff btw accuracy & flops

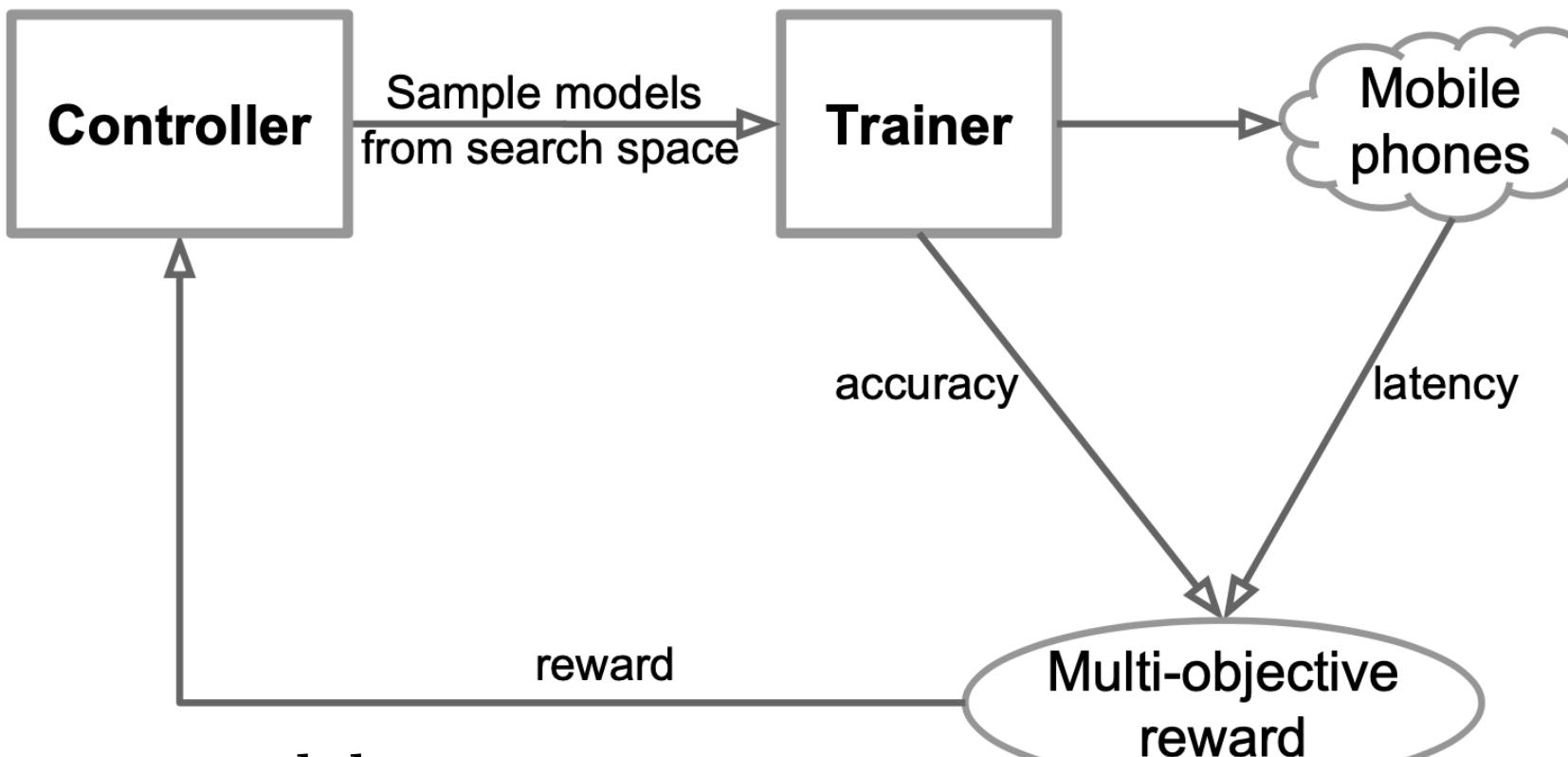
Stage $i$	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	$224 \times 224$	32	1
2	MBCConv1, k3x3	$112 \times 112$	16	1
3	MBCConv6, k3x3	$112 \times 112$	24	2
4	MBCConv6, k5x5	$56 \times 56$	40	2
5	MBCConv6, k3x3	$28 \times 28$	80	3
6	MBCConv6, k5x5	$14 \times 14$	112	3
7	MBCConv6, k5x5	$14 \times 14$	192	4
8	MBCConv6, k3x3	$7 \times 7$	320	1
9	Conv1x1 & Pooling & FC	$7 \times 7$	1280	1





Boulder

# MnasNet: Platform-Aware Neural Architecture Search for Mobile



$m \rightarrow$  model

$ACC(m) \rightarrow$  accuracy of the model on the target task

$LAT(m) \rightarrow$  inference latency on the target mobile platform

$T \rightarrow$  target latency

**Hard Constraint**

$$\underset{m}{\text{maximize}} \quad ACC(m)$$

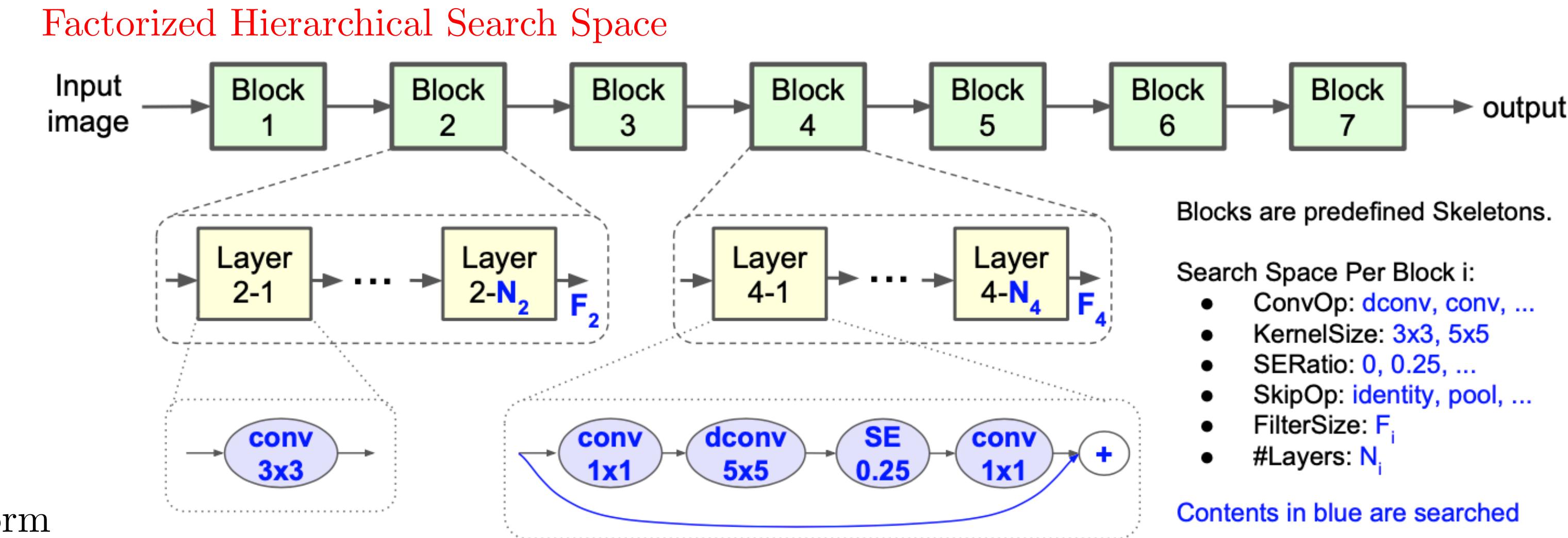
$$\text{subject to} \quad LAT(m) \leq T$$

**Soft Constraint**

$$\underset{m}{\text{maximize}} \quad ACC(m) \times \left[ \frac{LAT(m)}{T} \right]^w$$

$$w = \begin{cases} \alpha, & \text{if } LAT(m) \leq T \\ \beta, & \text{otherwise} \end{cases}$$

Empirical observation: doubling the latency usually brings about 5% relative accuracy gain.



Given two models:

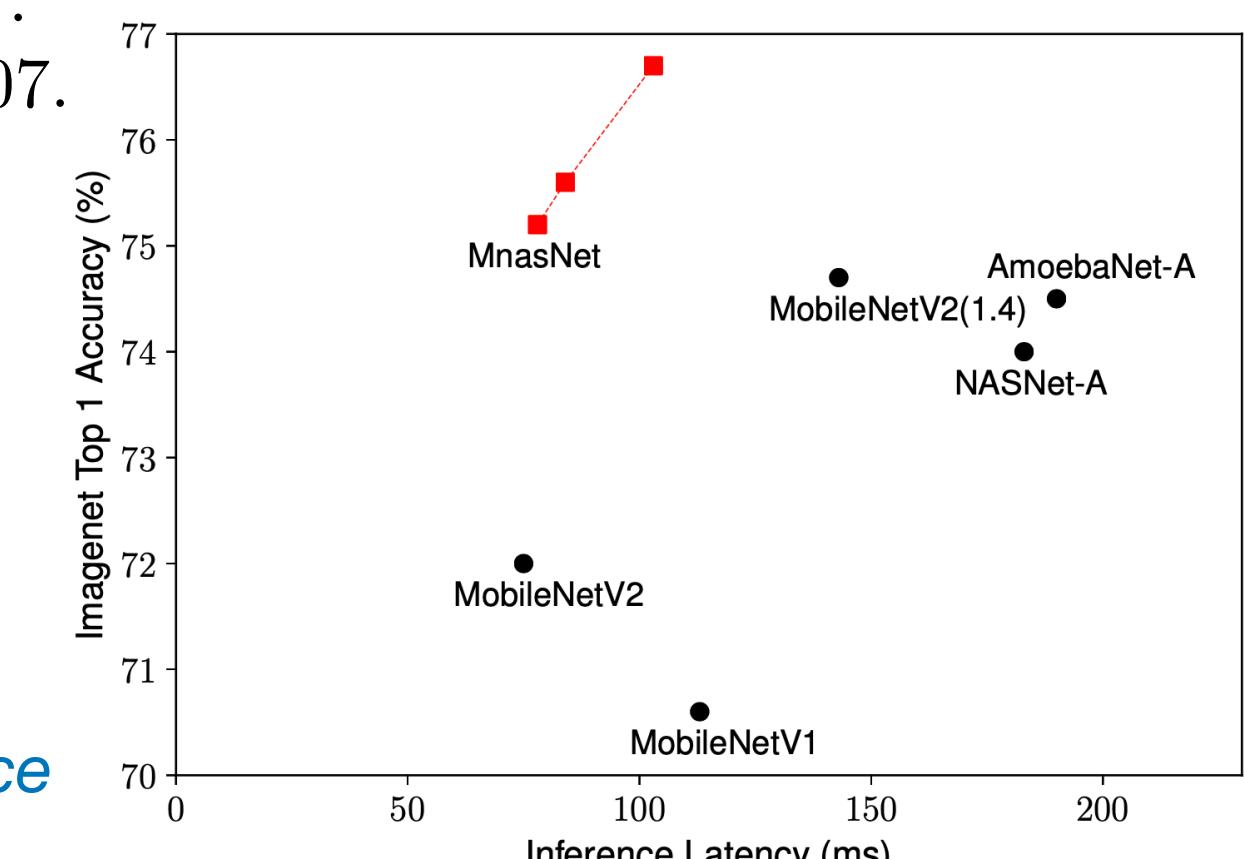
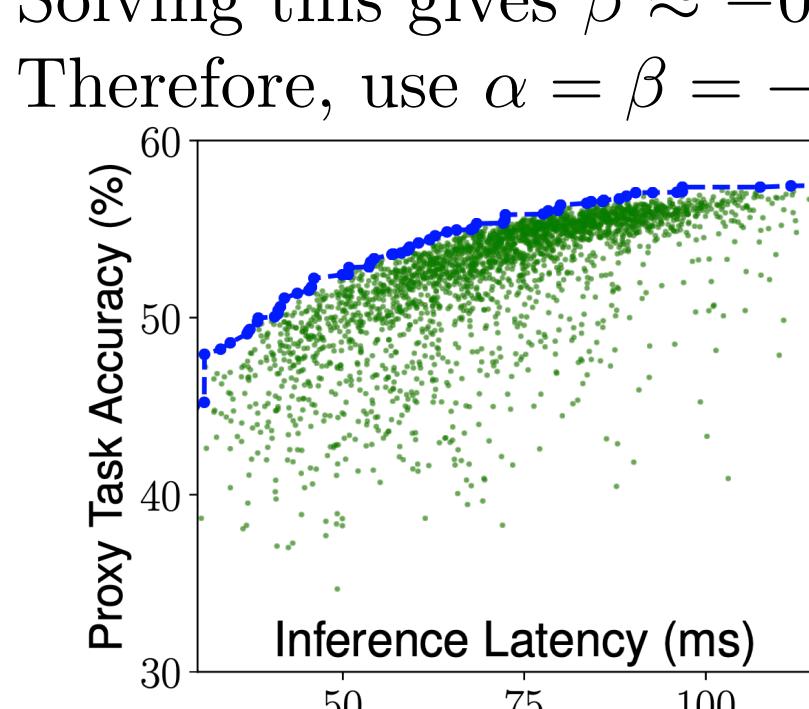
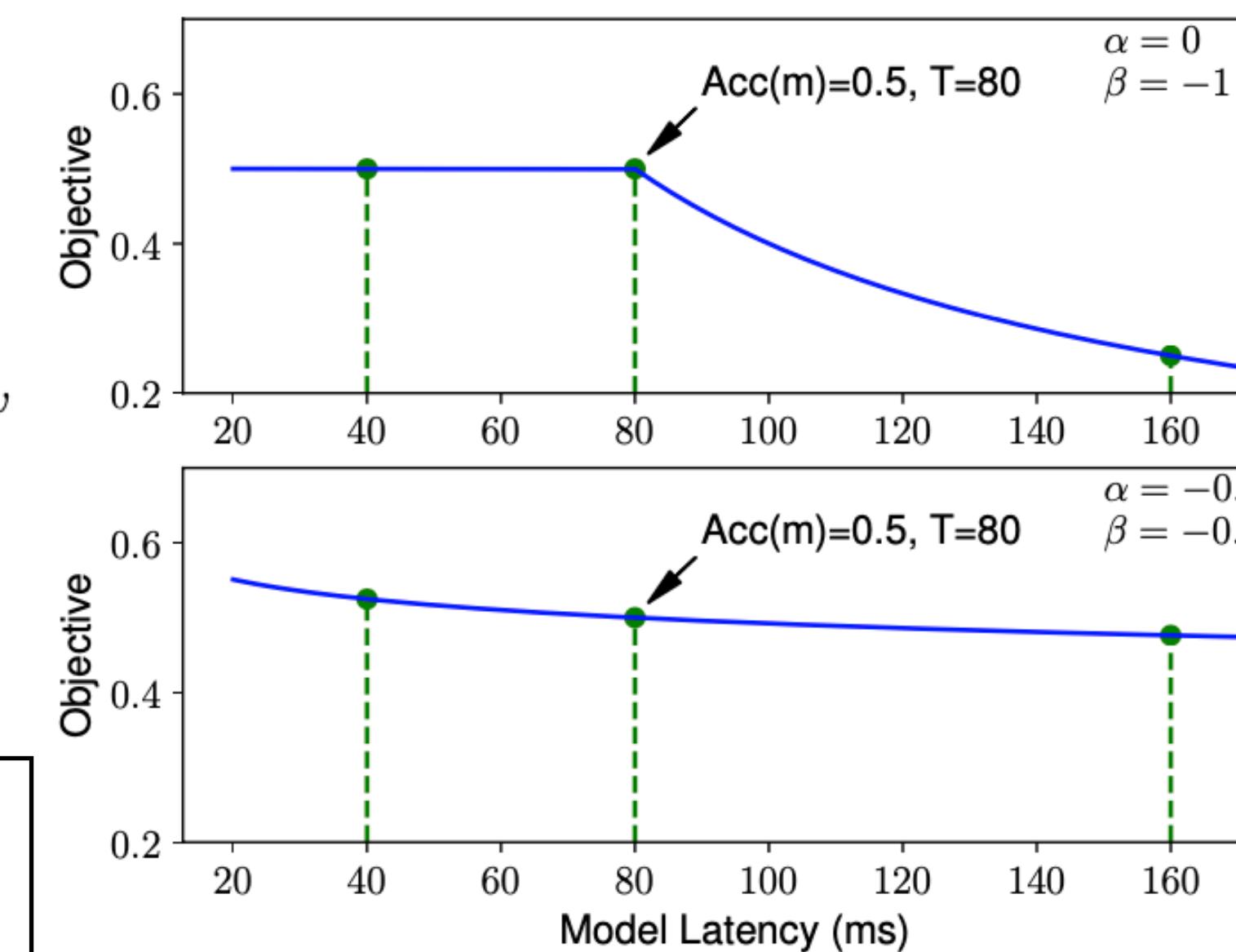
- (1)  $m_1$  has latency  $l$  and accuracy  $a$
- (2)  $m_2$  has latency  $2l$  and 5% higher accuracy  $1.05a$

They should have similar rewards:

$$R(m_2) = 1.05a(2l/T)^\beta \approx R(m_1) = a(l/T)^\beta.$$

Solving this gives  $\beta \approx -0.07$ .

Therefore, use  $\alpha = \beta = -0.07$ .





Boulder



# Questions?

[YouTube Playlist](#)

---