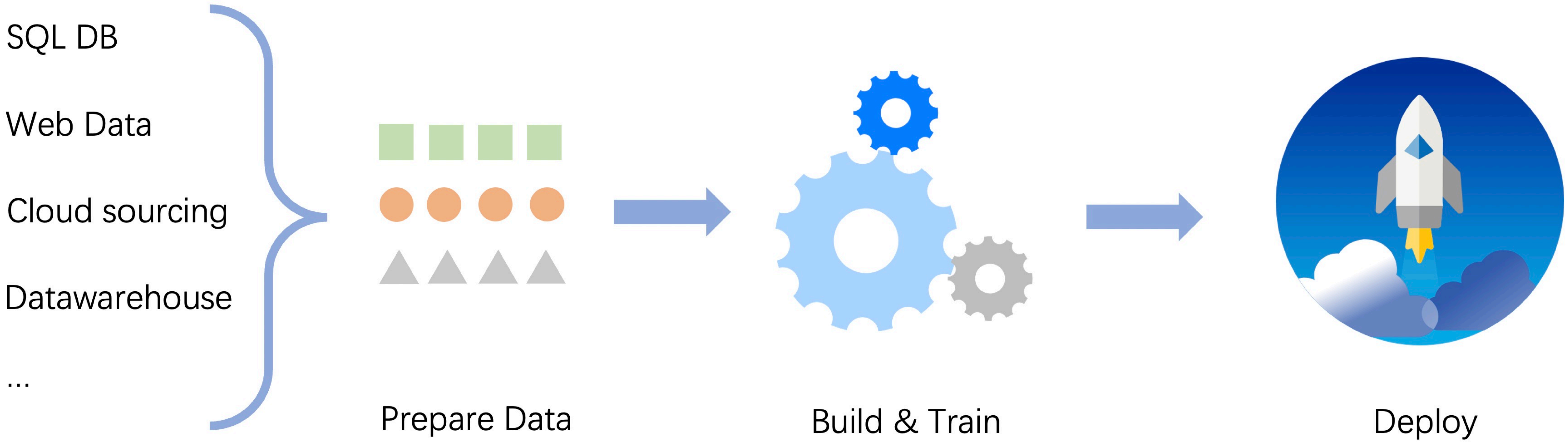
COMP5212: Machine Learning

Lecture 21

Minhao Cheng

Machine learning pipeline

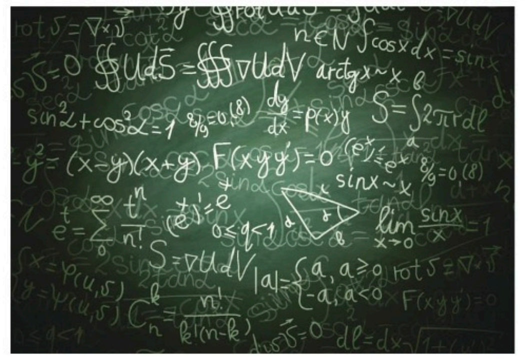


Machine learning pipeline

The devil is in the details

- What feature

Constraint/Rule

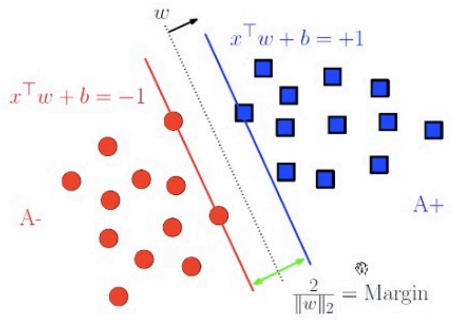


Budget

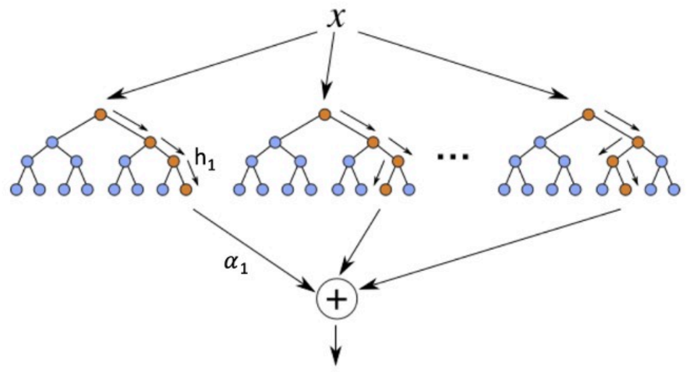
Efficiency

- Which model

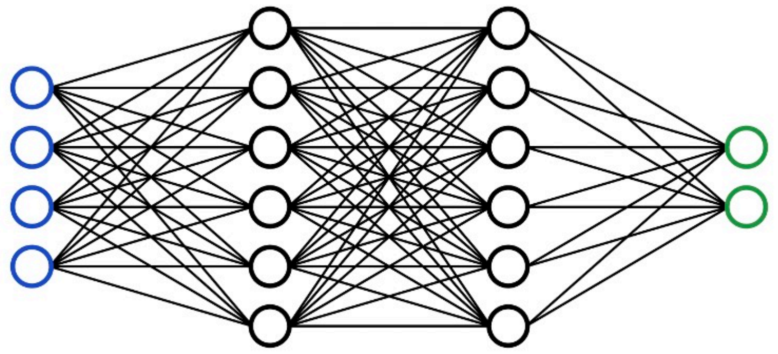
Linear model



Boosting model

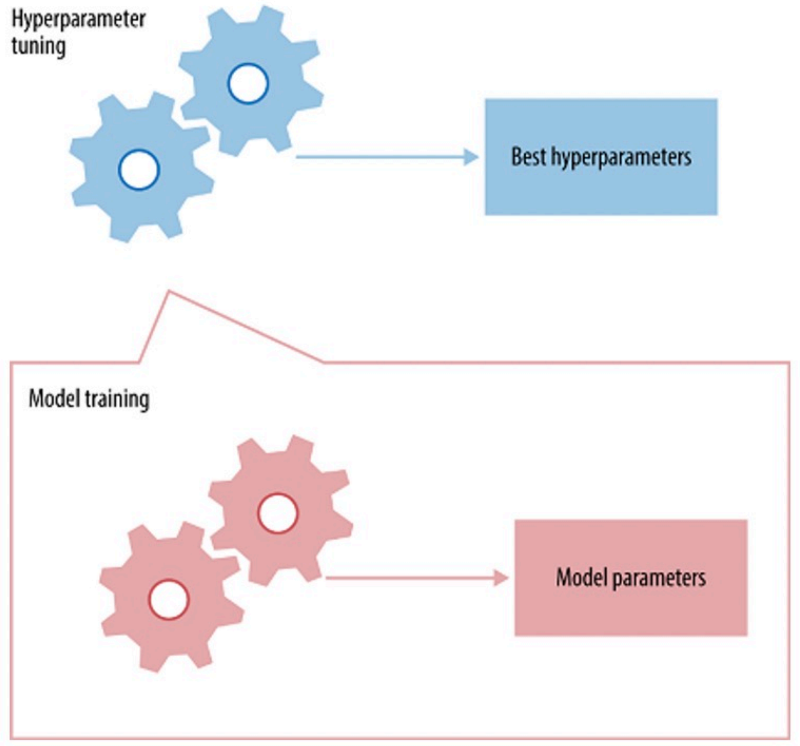


Neural network

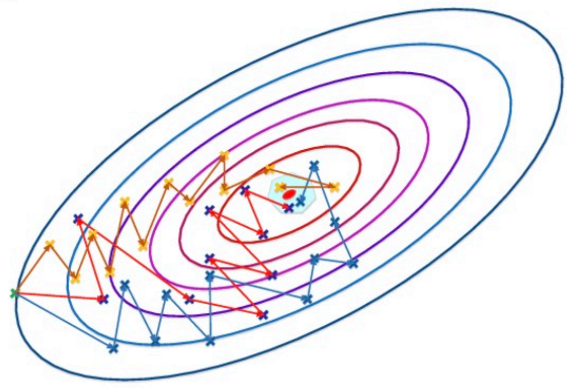


- Which parameter

Hyperparameter



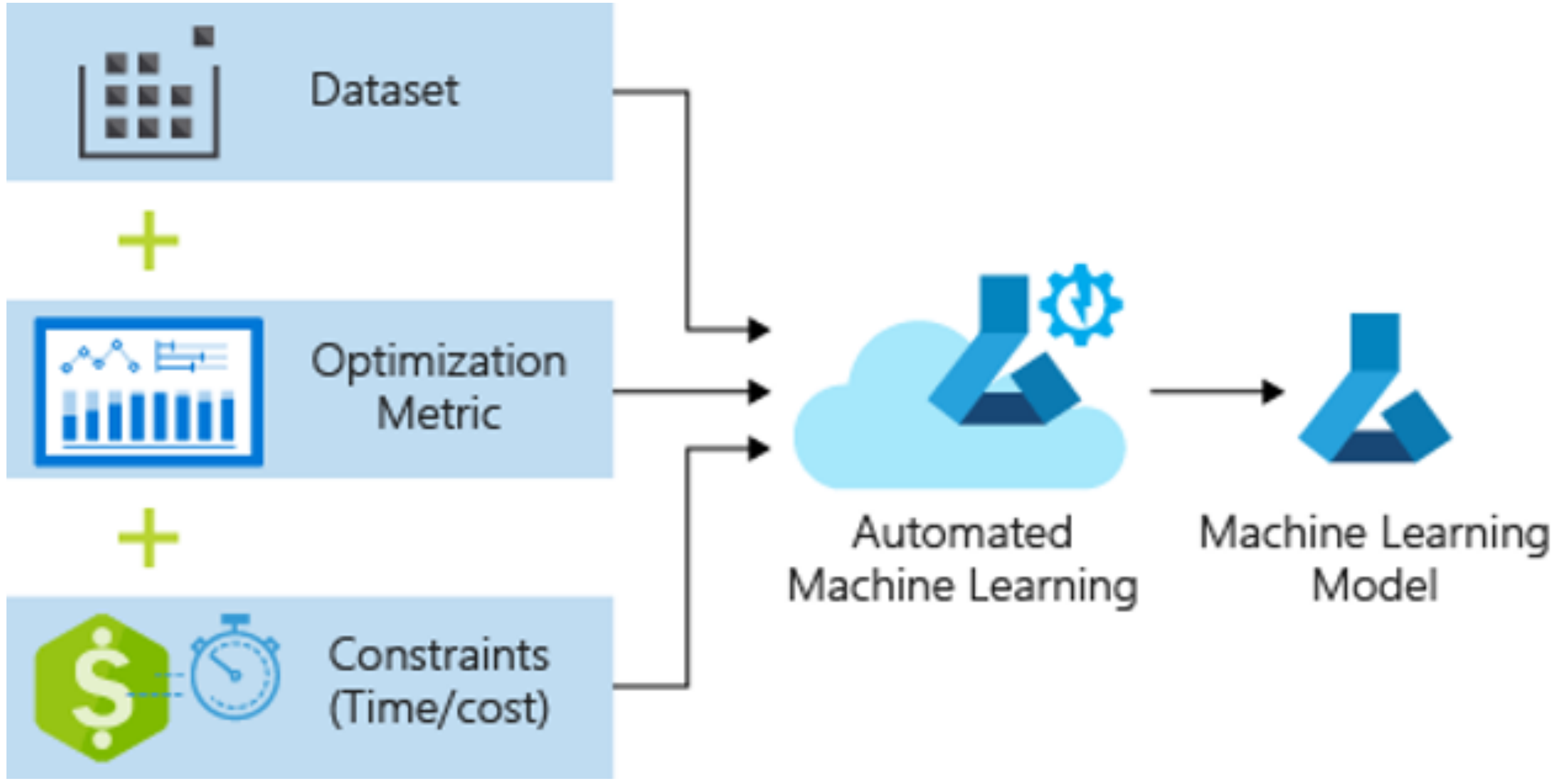
Optimizer



Automated Machine learning

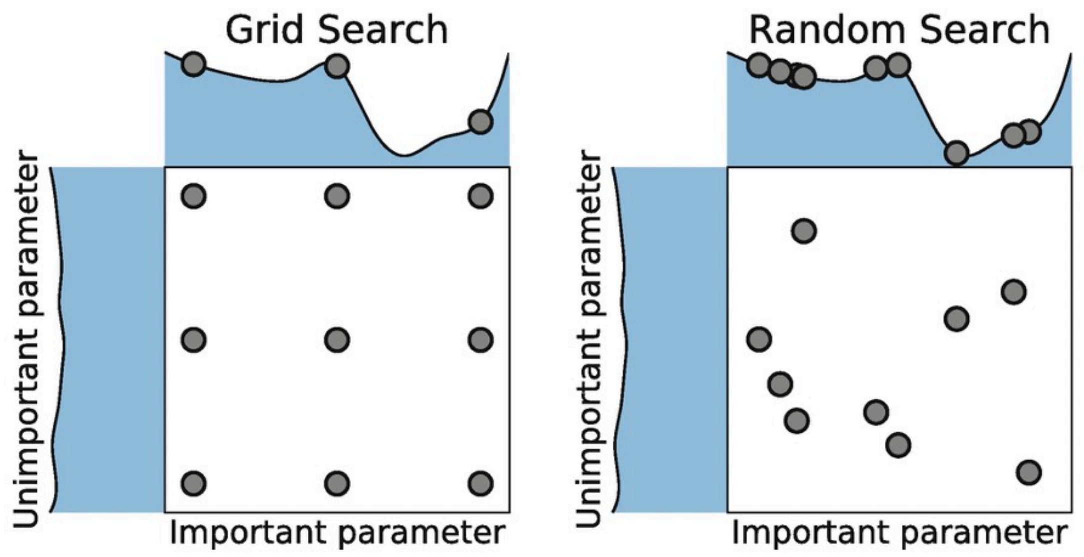
The devil is in the details

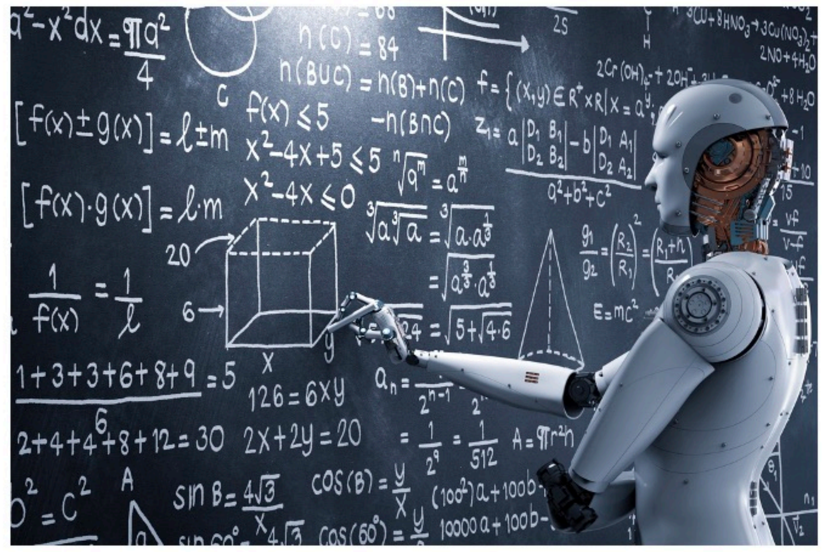
- AutoML simplifies each step in the machine learning process, from handling a raw dataset to deploying a practical machine learning model.

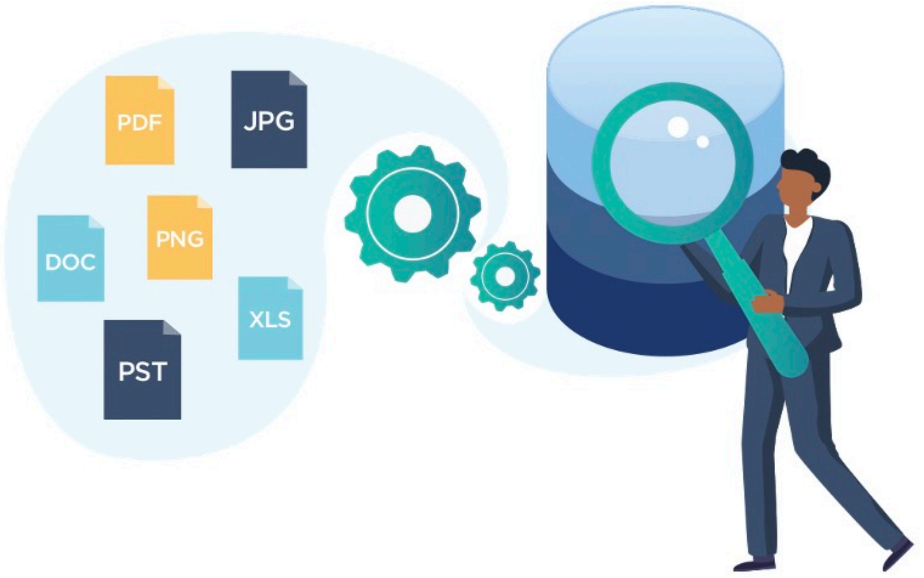


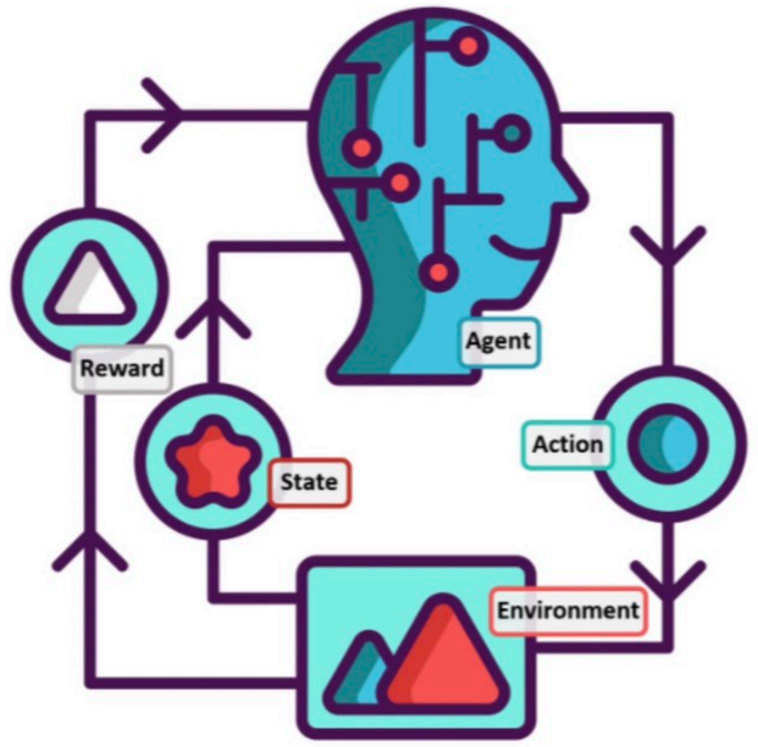
Automated Machine learning

- AutoML:
  - Neural Architecture Search (NAS)
  - Hyperparameter Optimization (HPO)
  - Meta Learning and Learning to learn
  - Automated Reinforcement learning
  - AutoML in Physical World
  - Automated model selection
  - ...





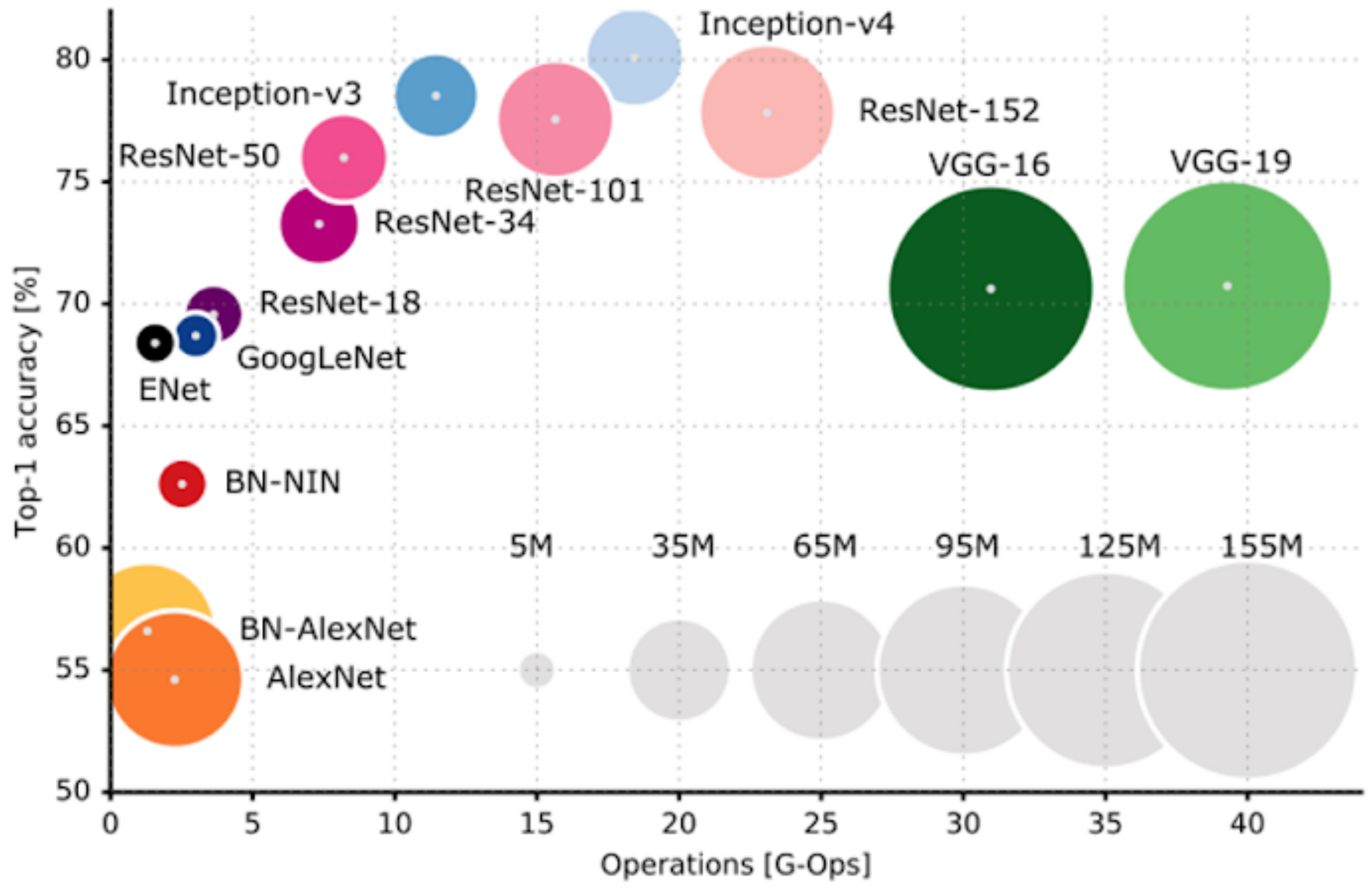


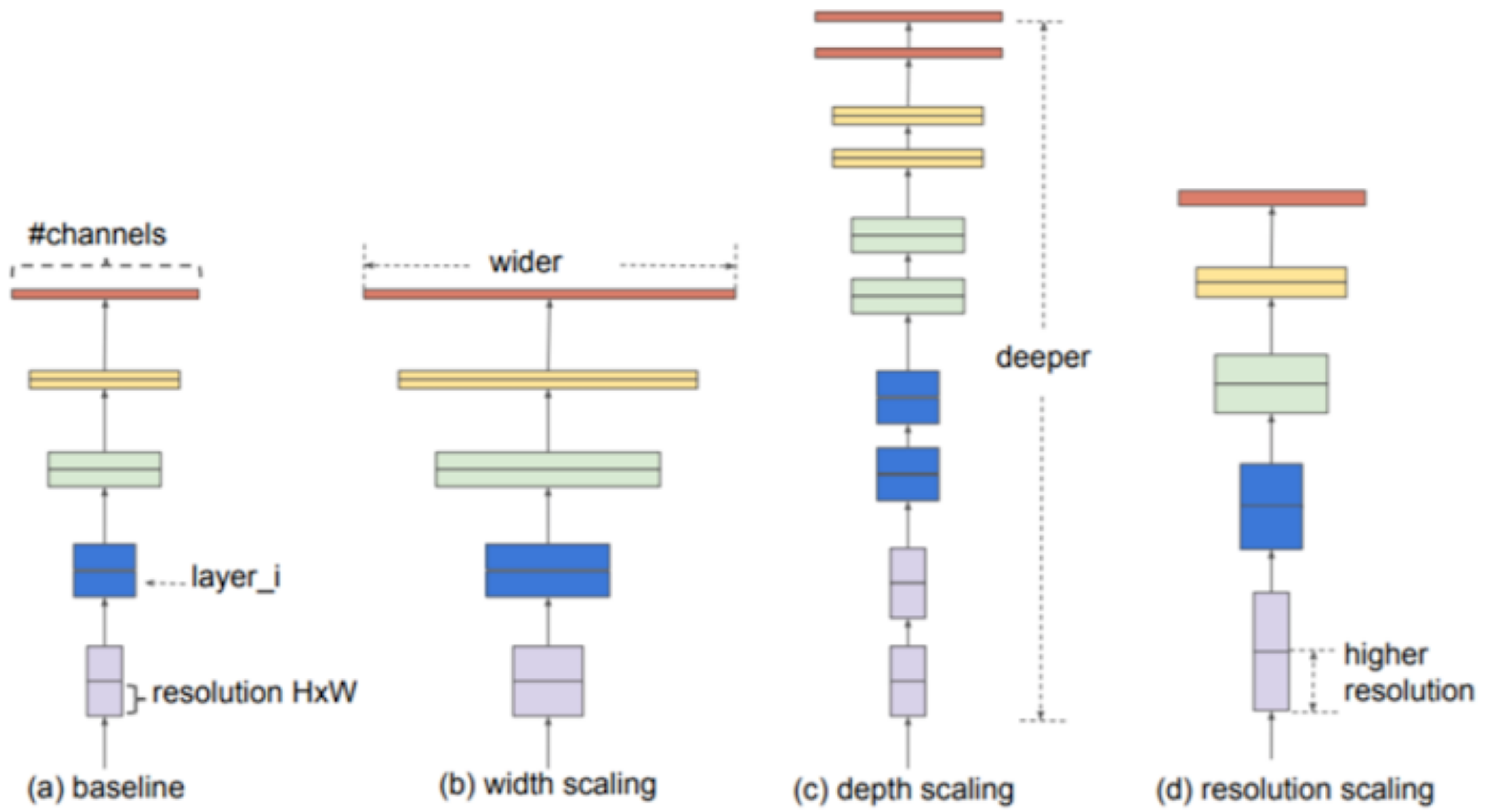


AutoML

Architecture of Neural Network

- Neural network architecture is important for both accuracy and efficiency



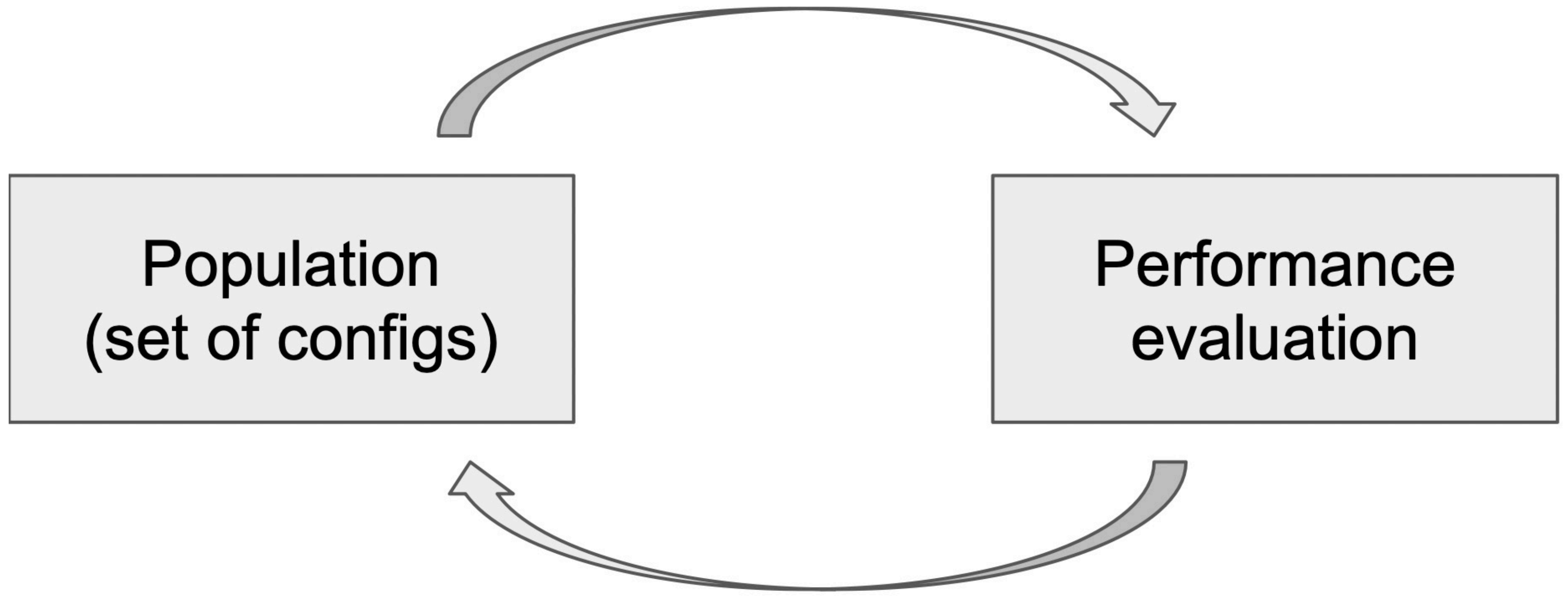


Can we automatically design architecture?

Neural Architecture Search

History of Neural Architecture Search (NAS)

- Early years: only on toy or small-scaled problems
  - Evolutionary algorithms (Miller et al., 89; Schaffer et al., 92; Verbancsics & Harguess, 13)
  - Bayesian optimization (Snoek et al, 12; Domhan et al., 15)



Evolutionary: recombination &
mutation
Bayesian: resampling

Neural Architecture Search

An early example

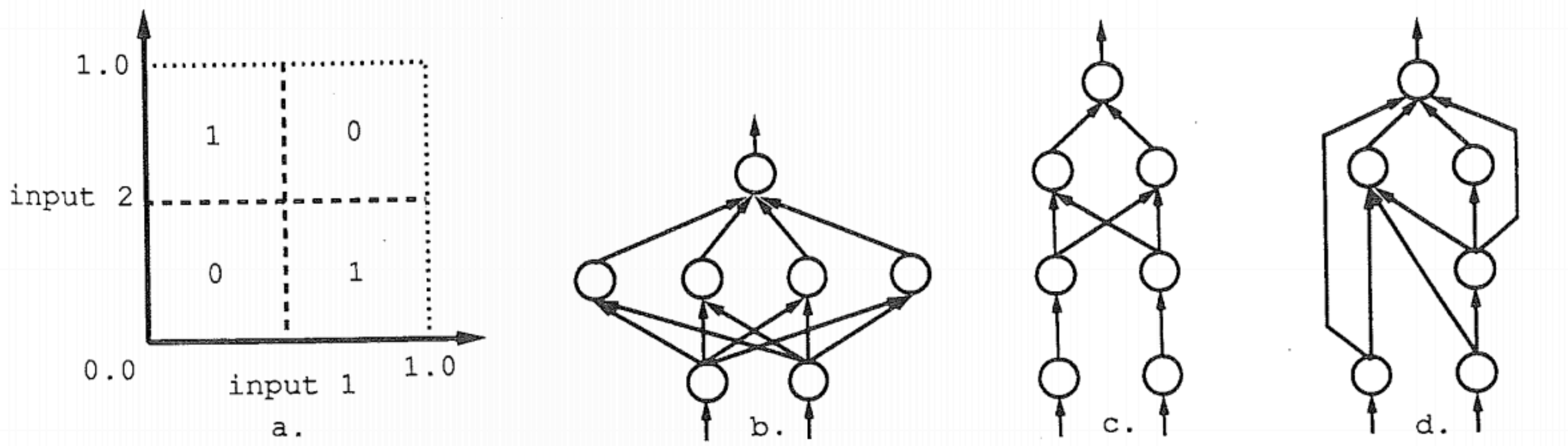


Figure 3. The four-quadrant problem. a. 2-d mapping to be learned. b. Standard 3-layer architectural solution. c. Standard 4-layer architectural solution. d. Typical discovered architectural solution.

Neural Architecture Search

- In 2016, Reinforcement learning (RL) is proposed for NAS
  - A better (structured) representation of search space
  - Learning a controller to generate architectures

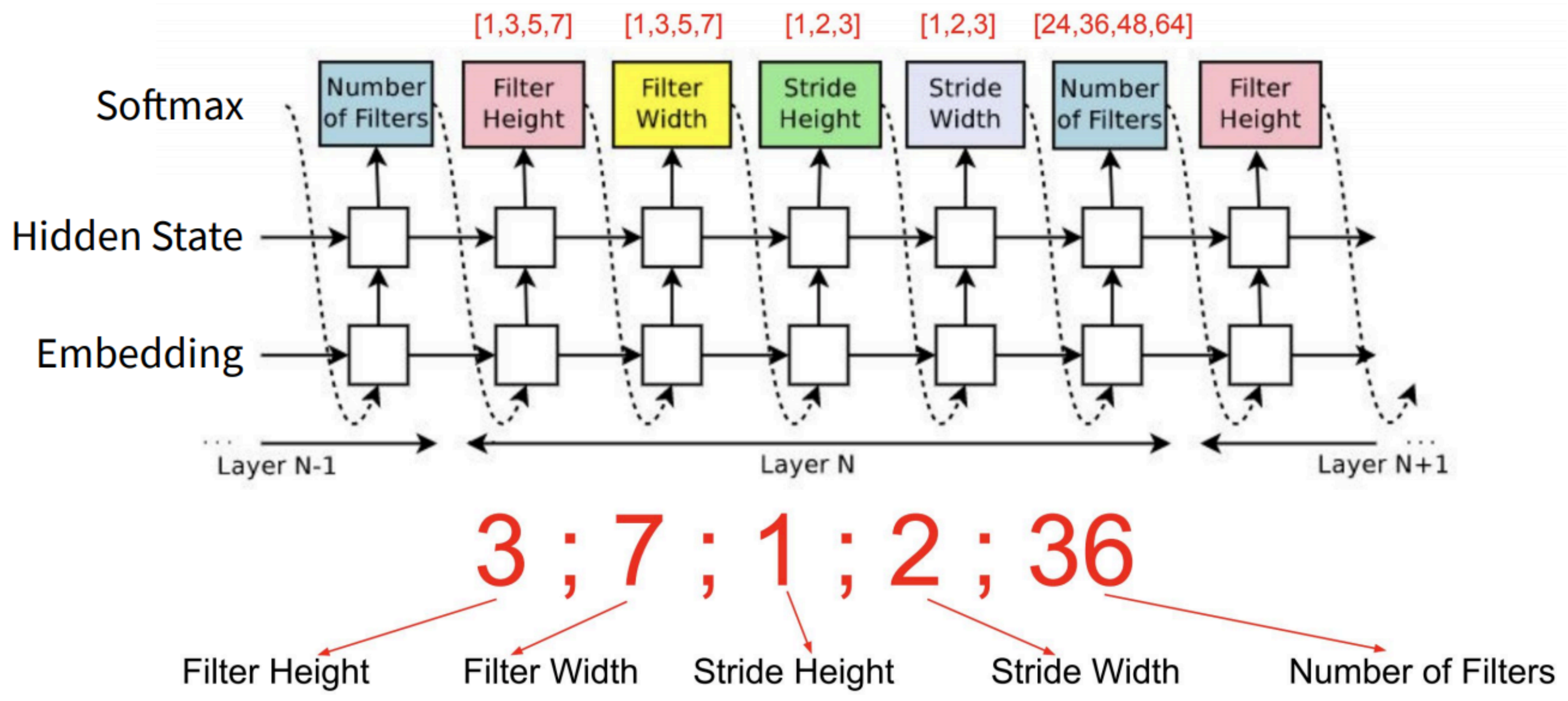
[Zoph and Quoc] Neural Architecture Search with Reinforcement Learning. ICLR, 2017.
[Baker, Gupta, Naik, Raskar] Designing Neural Network Architectures using Reinforcement Learning. ICLR, 2017.

- Successful results, but need hundreds of GPU days

Architecture	Test Error (%)	Search Cost (GPU days)	Search Method
ResNet (He et al., 2016)	4.62	-	manual
DenseNet-BC (Huang et al., 2017)	3.46	-	manual
NAS-RL (Zoph & Le, 2017)	3.65	22,400	RL

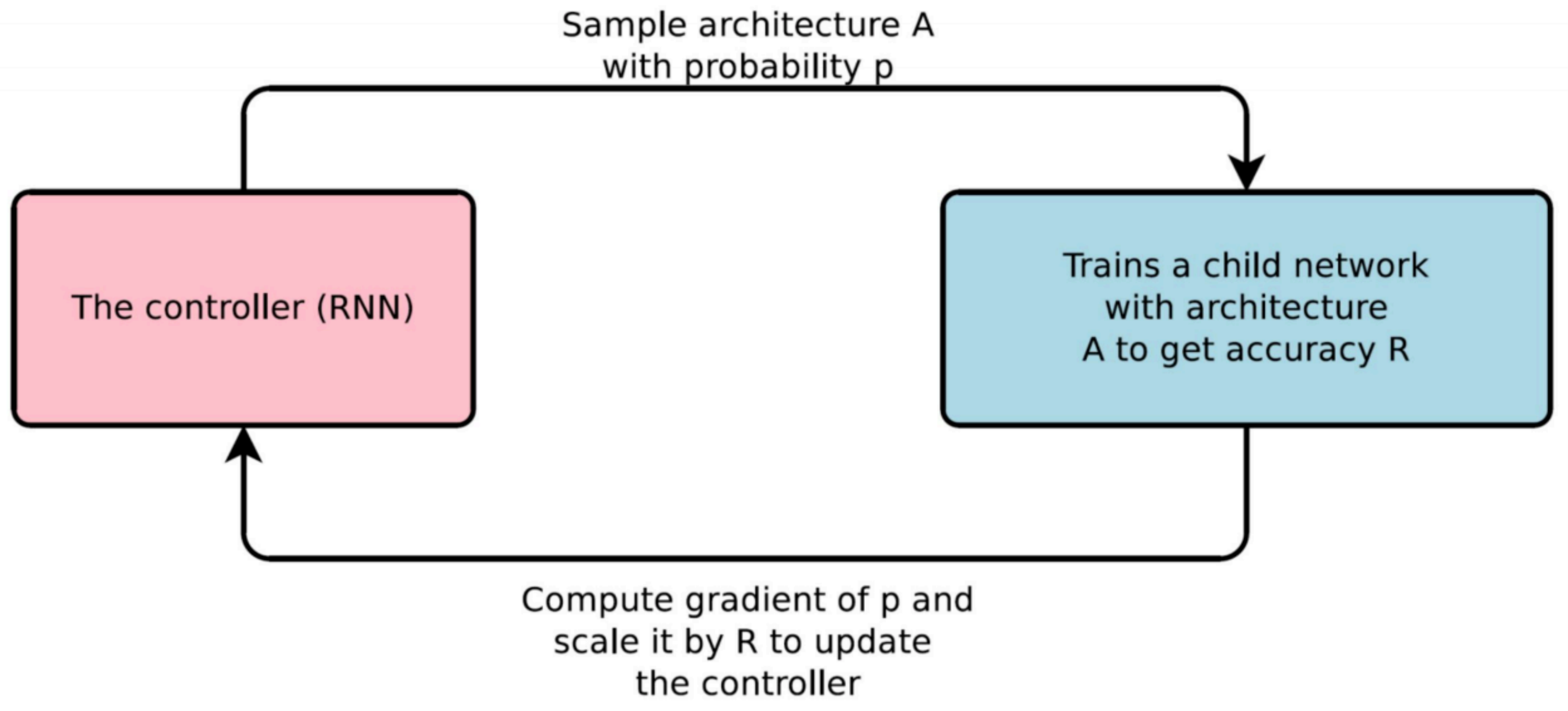
Neural Architecture Search

NAS with Reinforcement Learning



Neural Architecture Search

Training RNN controller by RL

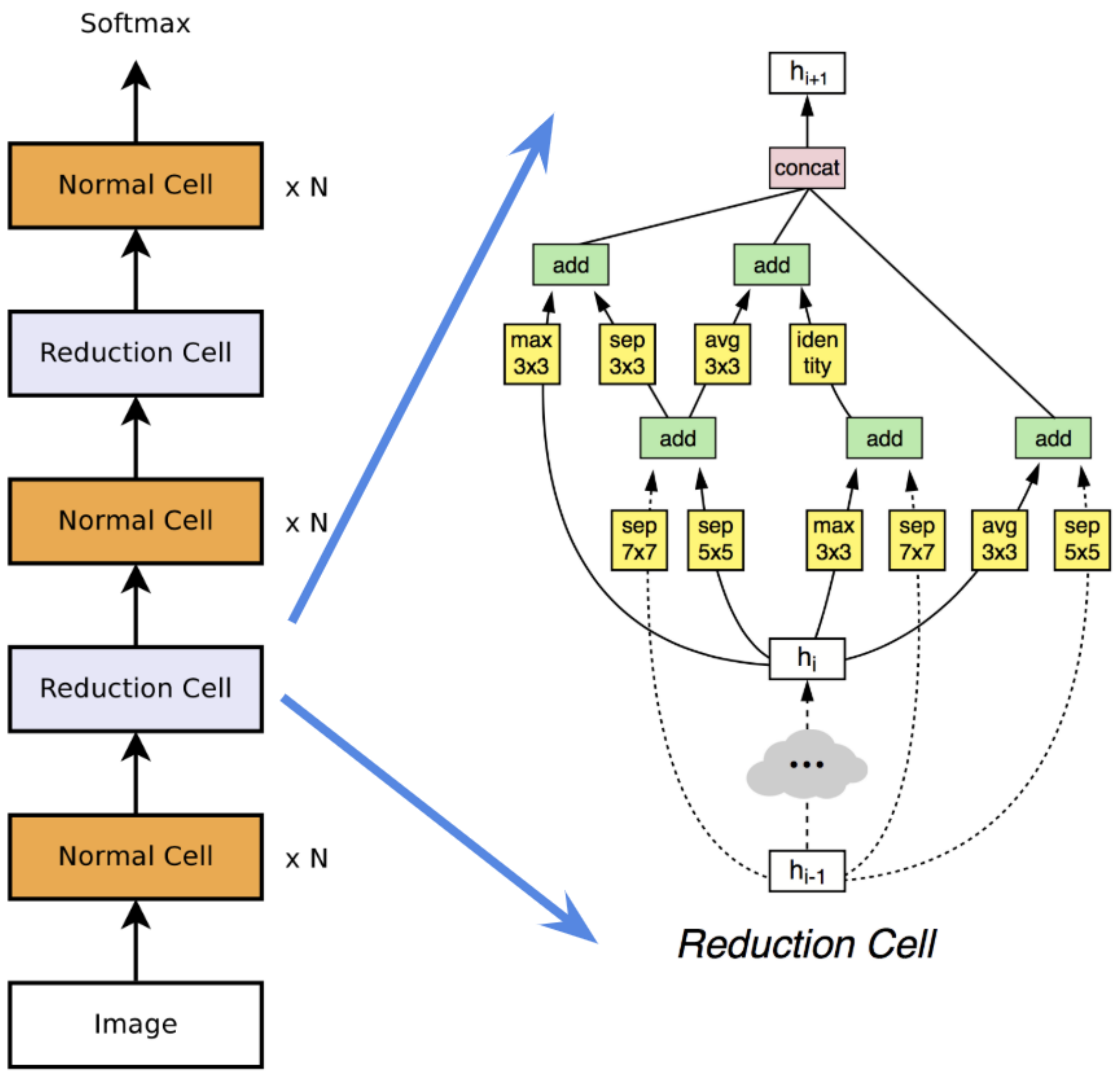


Extremely slow (>20,000 GPU days)

Neural Architecture Search

Cell-based Search Space (NASNet)

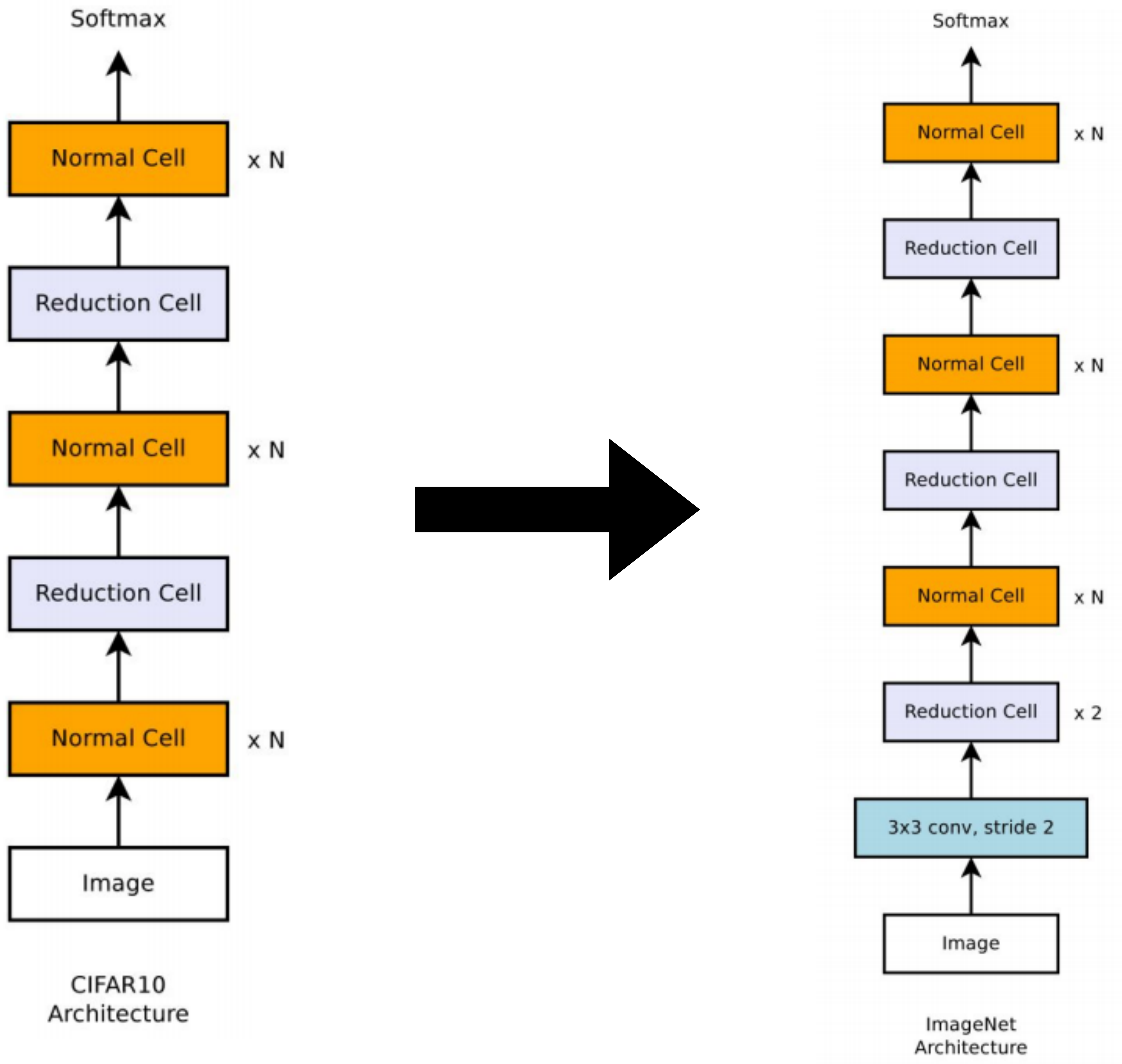
- Direct search on the global space:
    - Expensive; can't transfer to other datasets
- Cell-based search space:
    - Repeated cells (like ResNet)
    - Can use less blocks in searching
    - Can generalize to more complex datasets by stacking more blocks
- Compared with (Zoph & Le, 2017):
    - Error: 3.65 -> 2.65
    - Search cost: 22,400 -> 2000 GPU days



[Zoph, Vasudevan, Shlens, Le] Learning Transferable Architectures for Scalable Image Recognition. In CVPR, 2018.

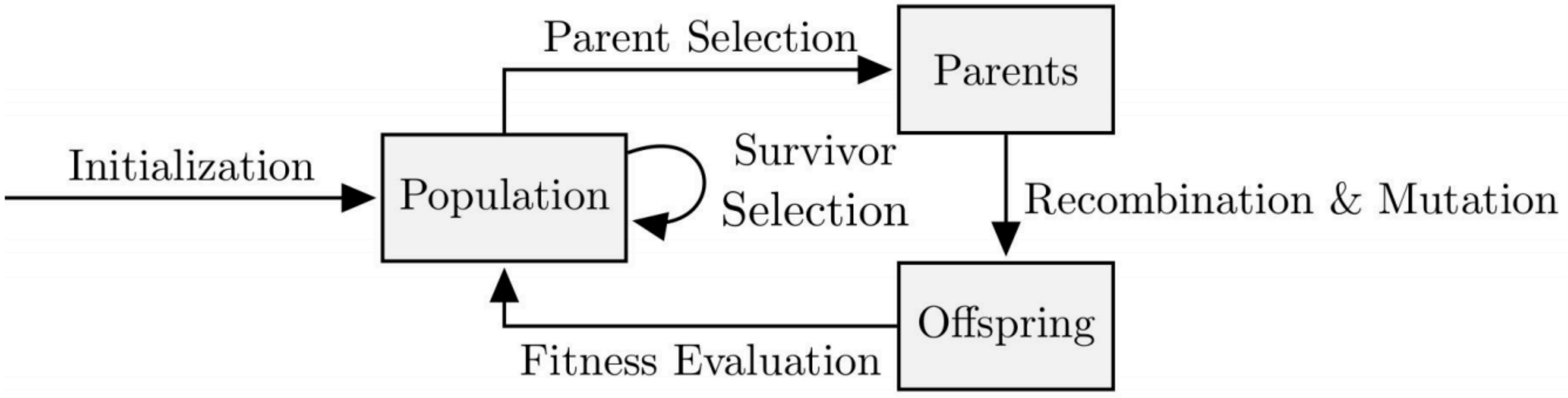
Neural Architecture Search

Generalize from CIFAR-10 to ImageNet



Neural Architecture Search
Evolutionary Algorithm

- Evolutionary algorithm also becomes possible with this search space



[Real, Aggarwak, Huang, Le] Regularized Evolution for Image Classifier Architecture Search. AAAI, 2019.

Neural Architecture Search

Other RL or evolutionary algorithms proposed

Reference	Error (%)	Params (Millions)	GPU Days	
Baker et al. (2017)	6.92	11.18	100	Reinforcement Learning
Zoph and Le (2017)	3.65	37.4	22,400	
Cai et al. (2018a)	4.23	23.4	10	
Zoph et al. (2018)	3.41	3.3	2,000	
Zoph et al. (2018) + Cutout	2.65	3.3	2,000	
Zhong et al. (2018)	3.54	39.8	96	
Cai et al. (2018b)	2.99	5.7	200	
Cai et al. (2018b) + Cutout	2.49	5.7	200	
Real et al. (2017)	5.40	5.4	2,600	Evolution
Xie and Yuille (2017)	5.39	N/A	17	
Suganuma et al. (2017)	5.98	1.7	14.9	
Liu et al. (2018b)	3.75	15.7	300	
Real et al. (2019)	3.34	3.2	3,150	

Designing competitive networks can take hundreds of GPU-days!
How to make neural architecture search more efficient?

Search typically takes **hundreds of GPU days**! Impractical for typical users.

Neural Architecture Search

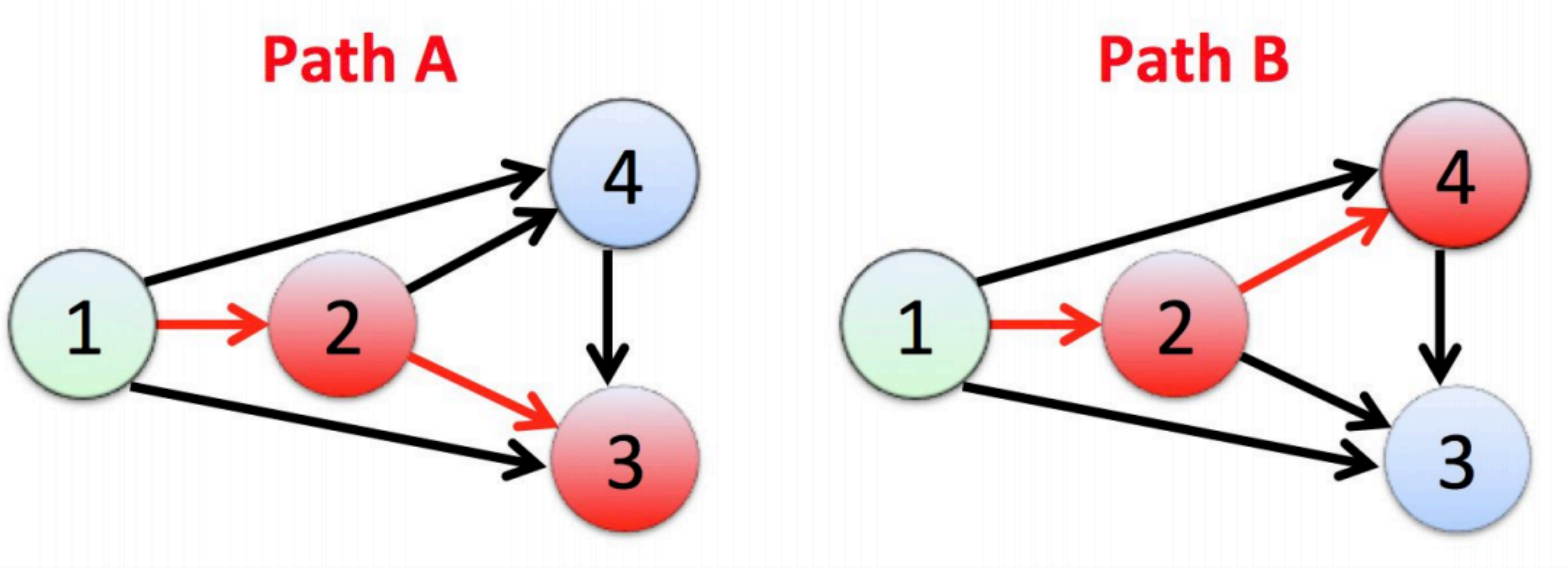
Significantly reduced search time since 2018

Architecture	Test Error (%)	Search Cost (GPU days)	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	-	manual
NAS-RL (Zoph & Le, 2017)	3.65	22,400	RL
NASNet-A (Zoph et al., 2018)	2.65	2000	RL
BlockQNN (Zhong et al., 2018)	3.54	96	RL
AmoebaNet (Real et al., 2019)	3.34 ± 0.06	3150	evolution
Hierarchical GA (Liu et al., 2018)	3.75	300	evolution
GCP (Suganuma et al., 2017)	5.98	15	evolution
DARTS (1st) (Liu et al., 2019)	3.00 ± 0.14	0.4	differentiable
DARTS (2nd) (Liu et al., 2019)	2.76 ± 0.09	1.0	differentiable
SNAS (moderate) (Xie et al., 2019)	2.85 ± 0.02	1.5	differentiable
GDAS (Dong & Yang, 2019)	2.93	0.3	differentiable
ProxylessNAS (Cai et al., 2019)†	2.08	4.0	differentiable
PC-DARTS (Xu et al., 2020)	2.57 ± 0.07	0.1	differentiable
NASP (Yao et al., 2019)	2.83 ± 0.09	0.1	differentiable
SDARTS-ADV (Chen & Hsieh, 2020)	2.61 ± 0.02	1.3	differentiable
DrNAS (Chen et al., 2019)	2.46 ± 0.03	$0.6^\ddagger$	differentiable
DARTS+PT (Wang et al., 2020)	2.61 ± 0.08	0.8	differentiable

Can run on a single GPU machine!

Neural Architecture Search

Concept of Weight Sharing

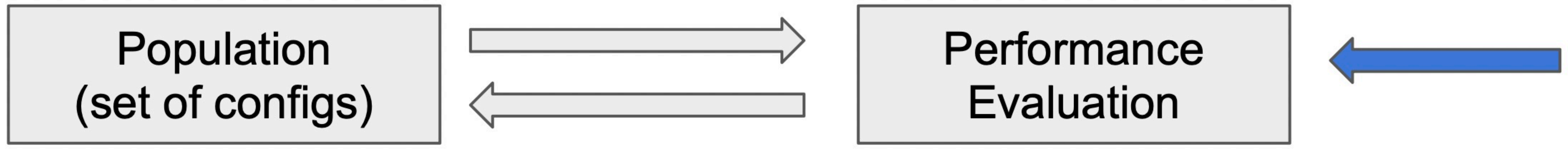


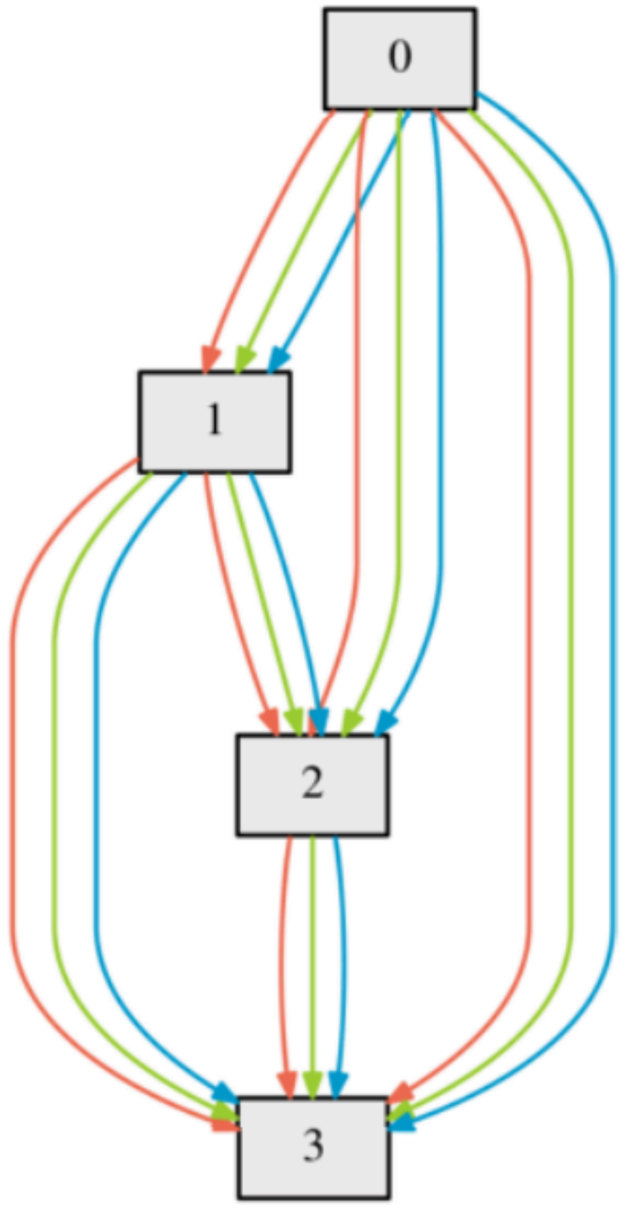
- Models defined by Path A and Path B should be trained separately
- Can we assume Path A and Path B share the same weight at 1->2?
  - Weight Sharing!
  - Avoid retraining for each new architecture

Neural Architecture Search

Concept of Weight Sharing

- Supernet: ensemble of many architectures
- All the architectures share the same w (weight sharing)
- Weight sharing can be directly used to speed up Performance Evaluation in other NAS methods
  - Train a “supernet” containing all the operations and weights
  - For any architecture, directly take the shared weights and evaluate on validation set
  - ENAS: weight sharing + RL
    - 0.5 GPU days with 2.9 error on CIFAR-10



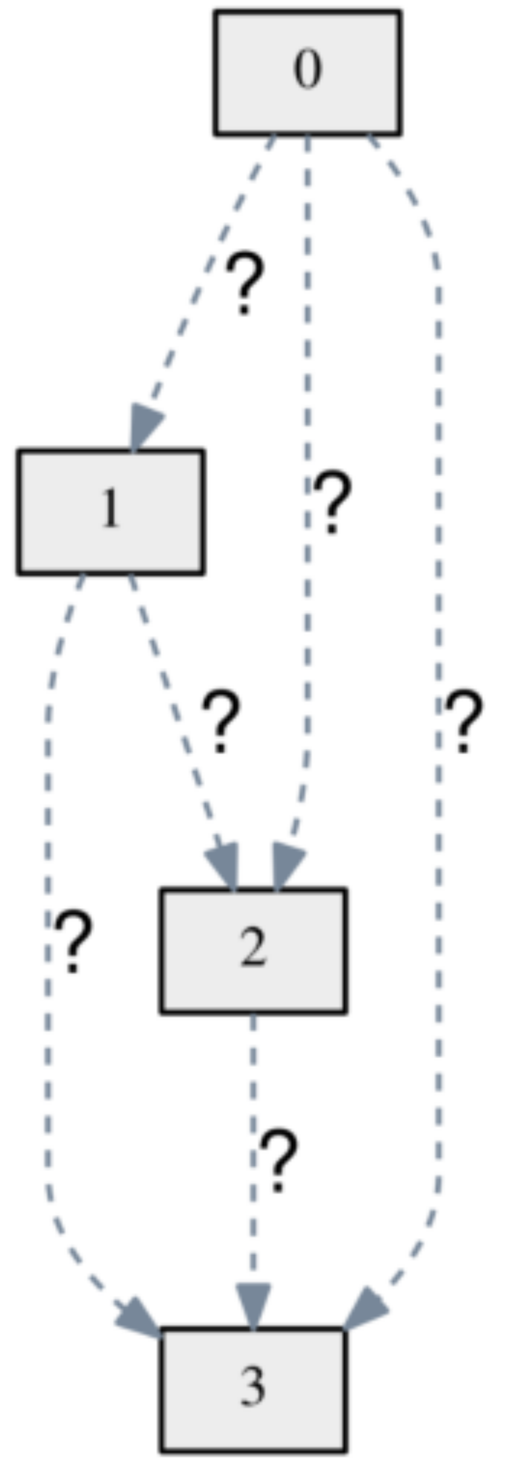


[Pham, Guan, Zoph, Le, Dean] Efficient Neural Architecture Search via Parameter Sharing. ICML, 2018.

Differentiable NAS

Can we directly obtain the final architecture through supernet training?

- Each edge is chosen from a pool of operations:
- Conv3x3, Conv5x5, Conv7x7, skip_connect, max_pool, avg_pool, zero, noise, ...
- One operation per edge => a discrete problem



(a)

Differentiable NAS

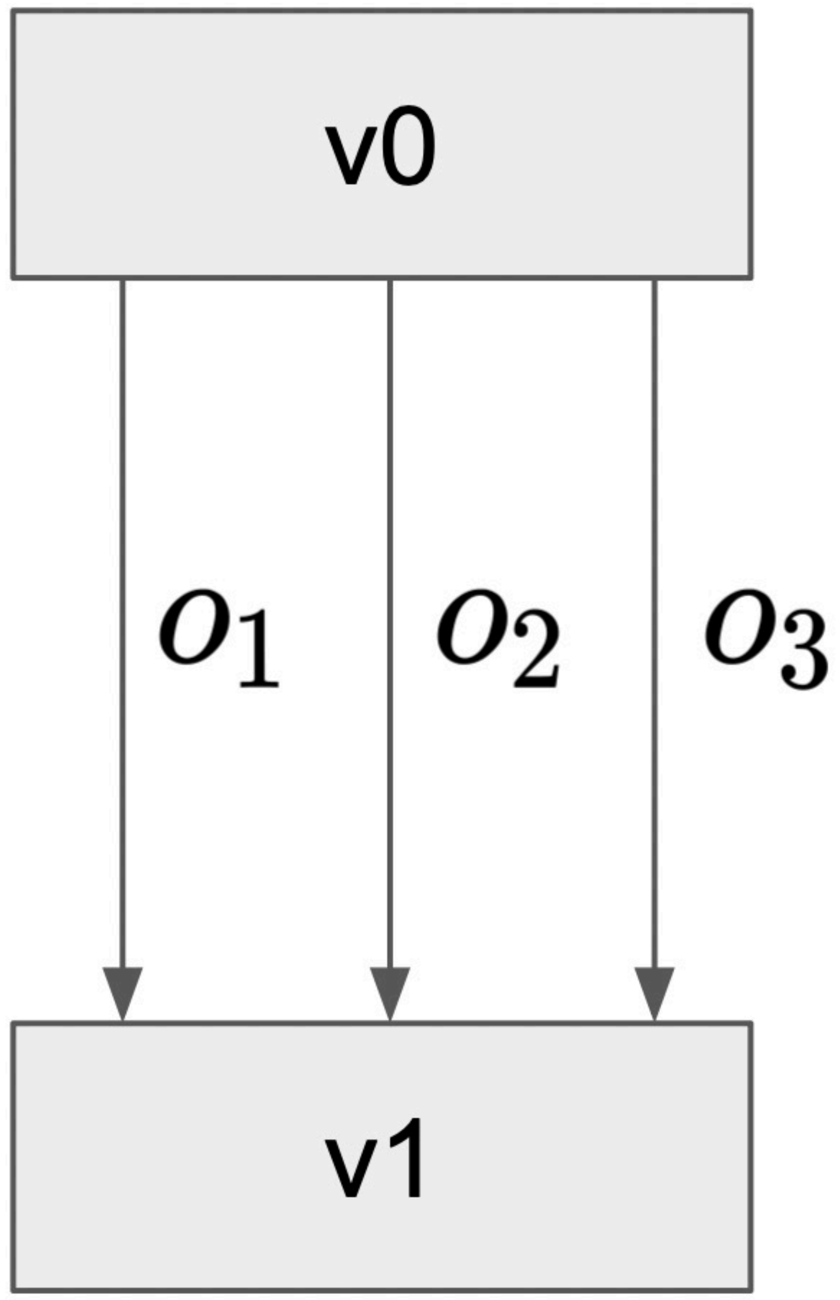
Continuous Relaxation

- For simplicity, assume 3 operations
  o_1 : Conv3×3,o_2 : skip connect, o_3 : $Zero$

- Assume each edge is a mixed of three operations:

$$v_1 = \alpha_1 o_1(v_0) + \alpha_2 o_2(v_0) + \alpha_3 o_3(v_0)$$

Weight of each operation



[Liu, Simonyan, Yang] DARTS: Differentiable Architecture Search. In ICLR 2019.

Differentiable NAS

Continuous Relaxation

- For simplicity, assume 3 operations
  o_1 : Conv3×3,o_2 : skip connect, o_3 : $Zero$

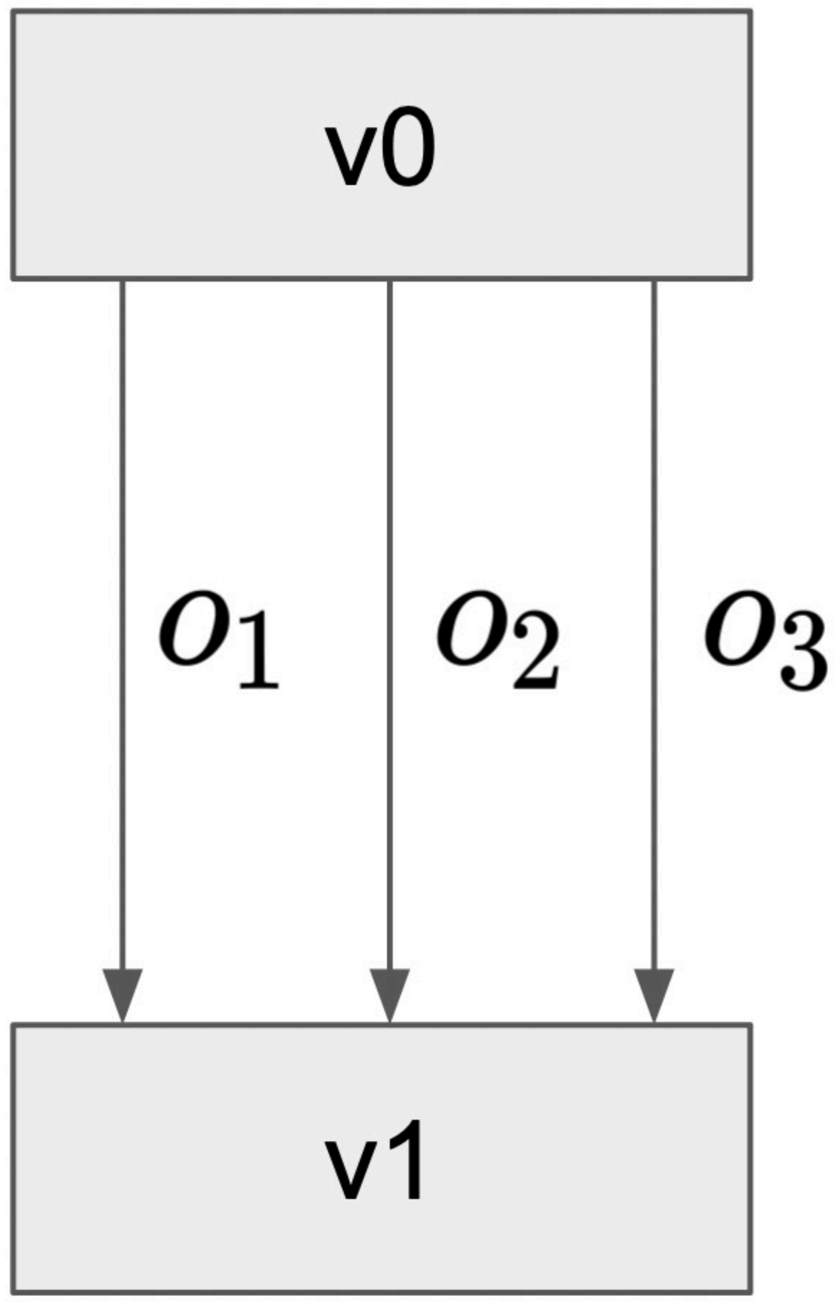
- Assume each edge is a mixed of three operations:

$$v_1 = \alpha_1 o_1(v_0) + \alpha_2 o_2(v_0) + \alpha_3 o_3(v_0)$$

Weight of each operation

- Can use softmax to ensure the weights form a prob. distribution

$$v_{\text{out}} = \sum_o \frac{\exp \alpha_o}{\sum_{o'} \exp \alpha_{o'}} o(v_{\text{in}})$$



[Liu, Simonyan, Yang] DARTS: Differentiable Architecture Search. In ICLR 2019.

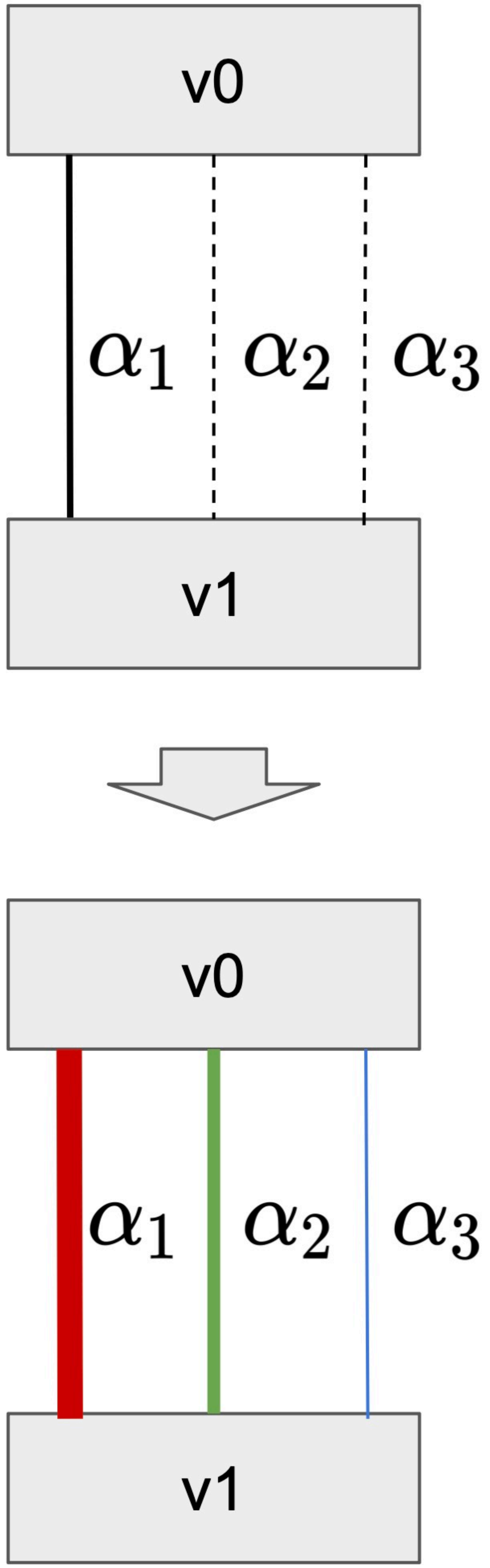
Differentiable NAS

Continuous Relaxation

- Final architecture: $[\alpha_1, \alpha_2, \alpha_3]$ is a one-hot vector
- Relax to continuous values in the search phase=> Bi-level optimization for finding α

$$\min_{\alpha} L_{\text{val}}(w^*(\alpha), \alpha)$$

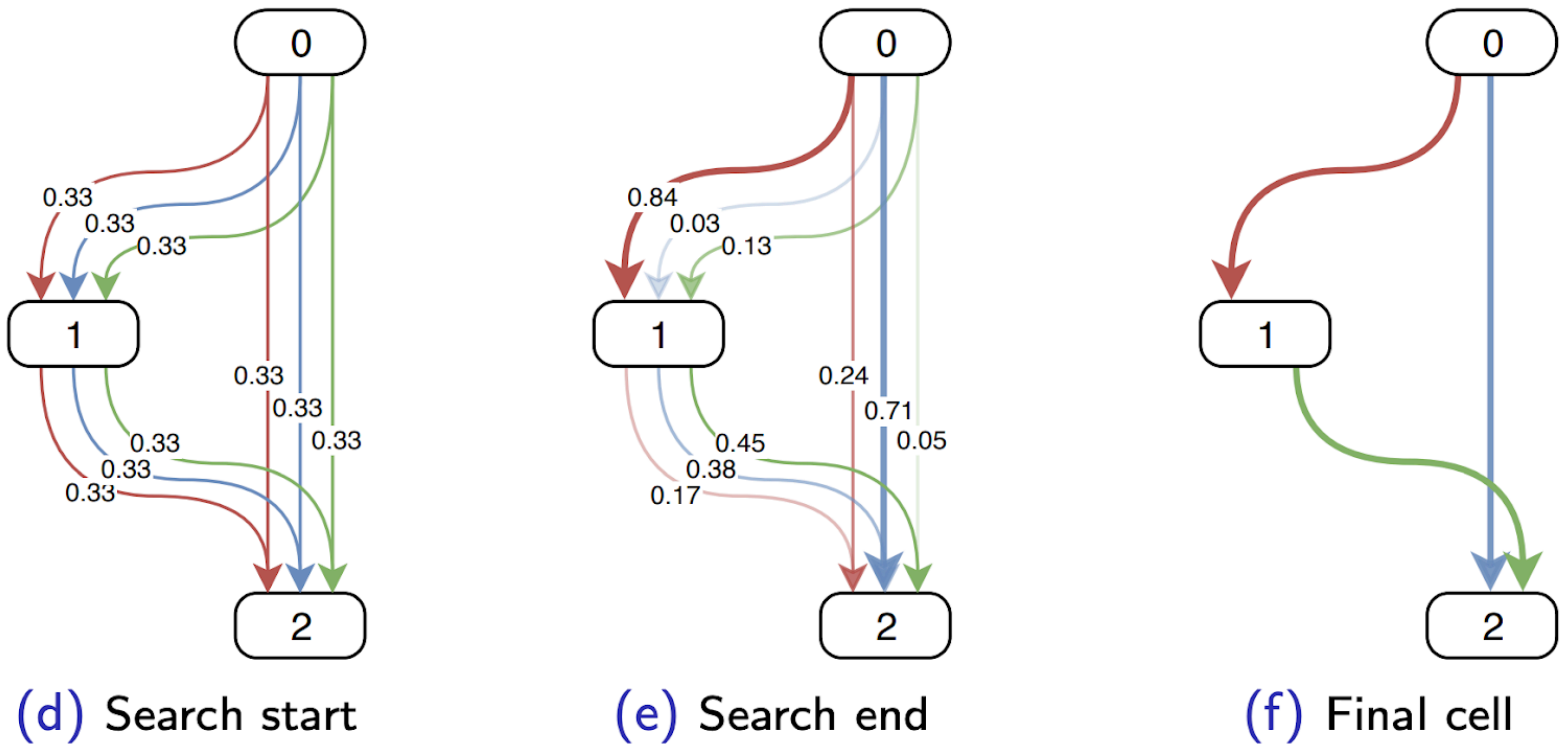
$$\text{s.t. } w^*(\alpha) = \arg\min_w L_{\text{train}}(w, \alpha)$$



Differentiable NAS

Differentiable Neural Architecture Search (DARTS)

- Solve the bi-level optimization problem to obtain (α^*, w^*) (supernet)
- Use magnitude of α^* to choose the final architecture



Differentiable NAS

How to solve bi-level optimization?

$$\min_{\alpha} L_{\text{val}}(w^*(\alpha), \alpha)$$
$$\text{s.t. } w^*(\alpha) = \arg\min_w L_{\text{train}}(w, \alpha)$$

- Iteratively update w and α
- Update w:
  - Time consuming to compute w* exactly => approximate by one SGD step
    - $w' \leftarrow w - \eta \nabla_w L_{\text{train}}(w, \alpha)$
- Update α:
  - First order DARTS: assume w is constant w.r.t. α
    - $\alpha \leftarrow \alpha - c \nabla_\alpha L_{\text{val}}(w', \alpha)$

Differentiable NAS

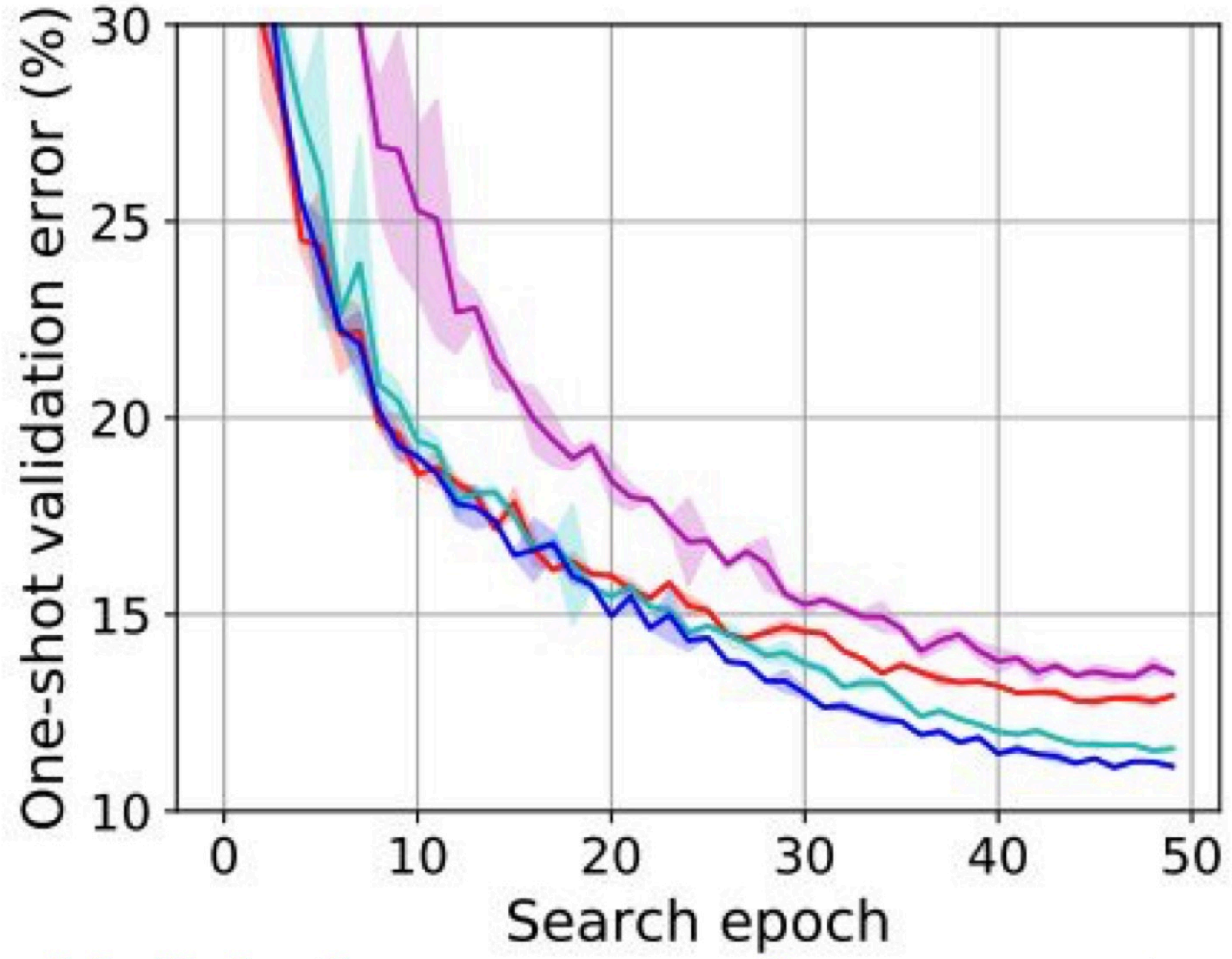
Complexity of DARTS

- Time complexity: training the supernet only once
  - Supernet is a network with K operations with each edge
    => only K times slower than standard training
  - Usually good enough
- Memory complexity (GPU memory):
  - Backprop on all the operations on each edge
    => K times memory consumption
  - Prohibits for many problems

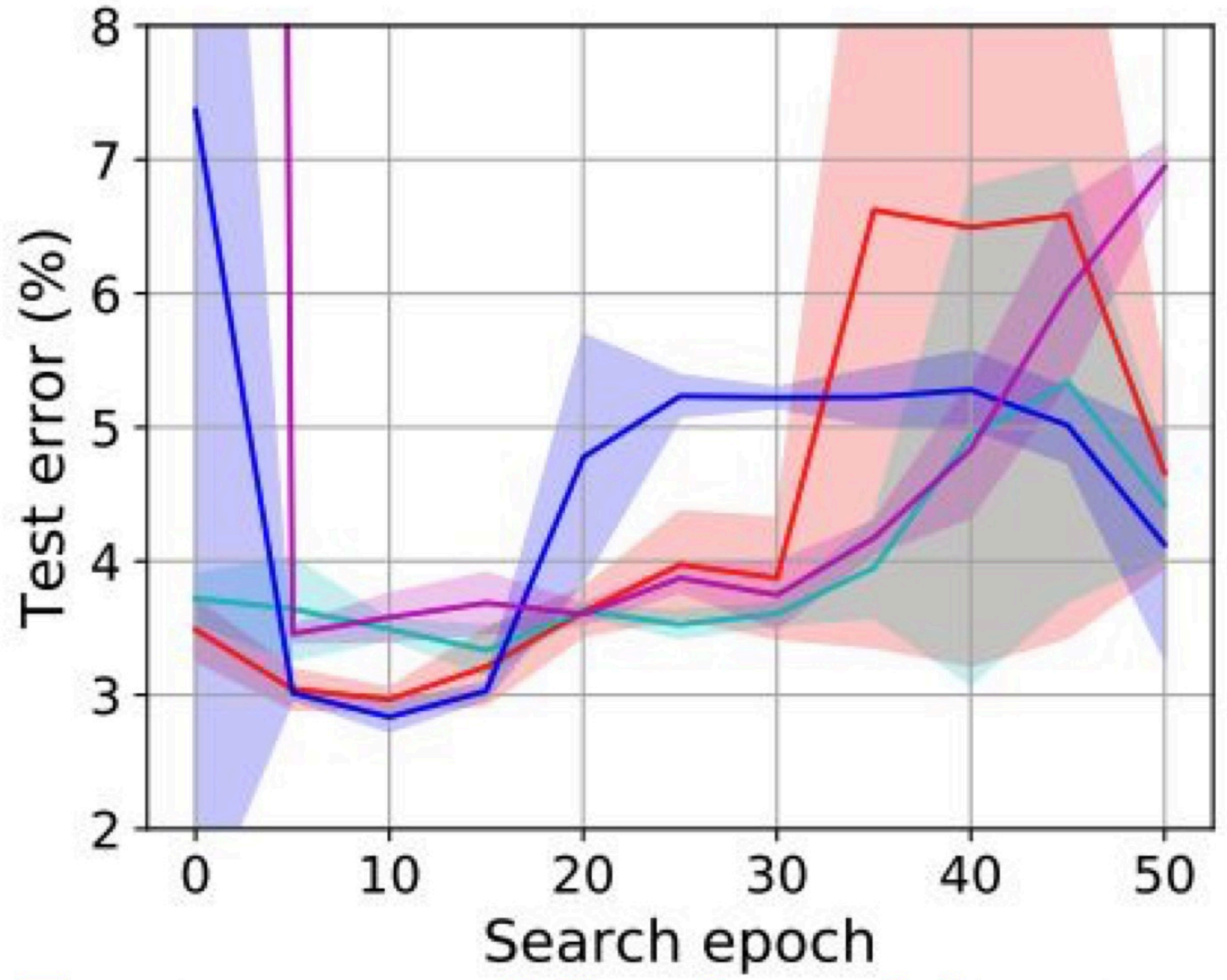
Differentiable NAS

DARTS fails in many simple cases

- Space 1: 2 operations per edge (selected from the original DARTS supernet)
- Space 2: 2 operations per edge {Conv3x3, skip_connect}
- Space 3: 3 operations per edge {Conv3x3, skip_connect, Zero}
- Space 4: 2 operations per edge {Conv3x3, Gaussian_noise}



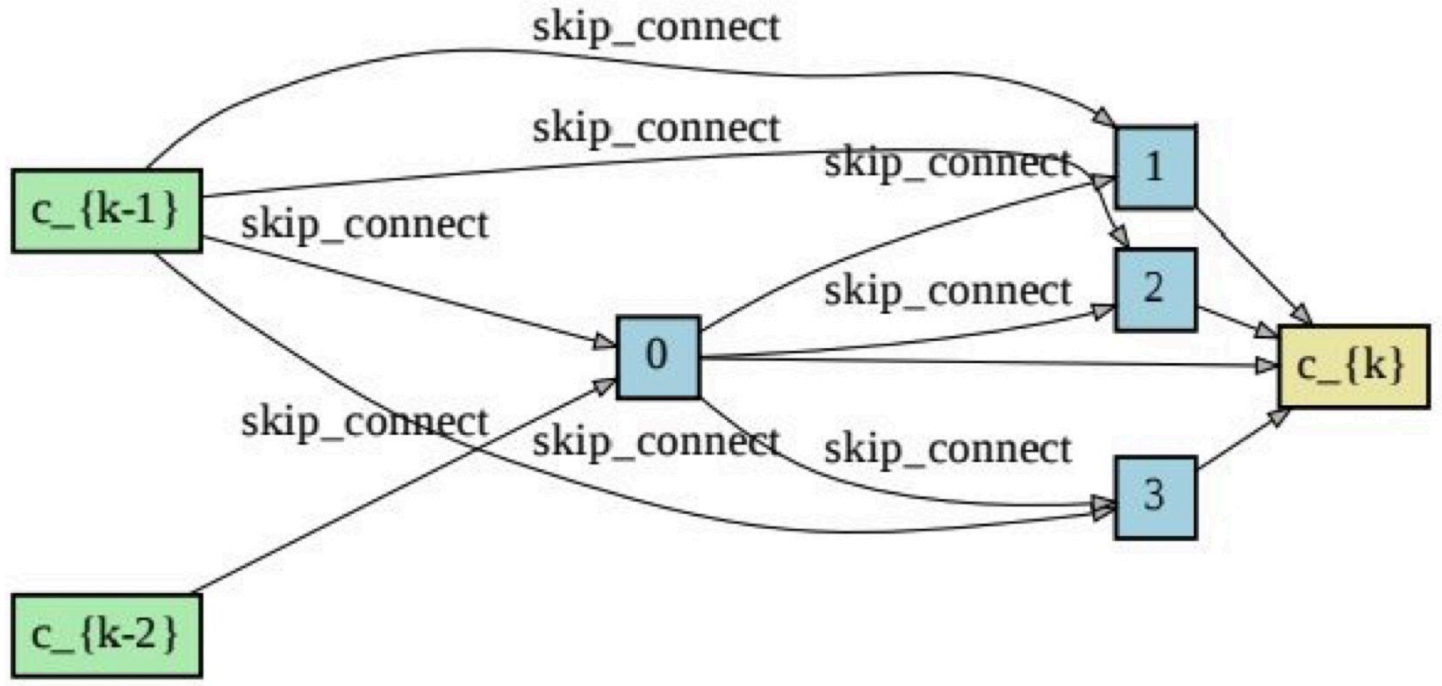
Validation error of supernet



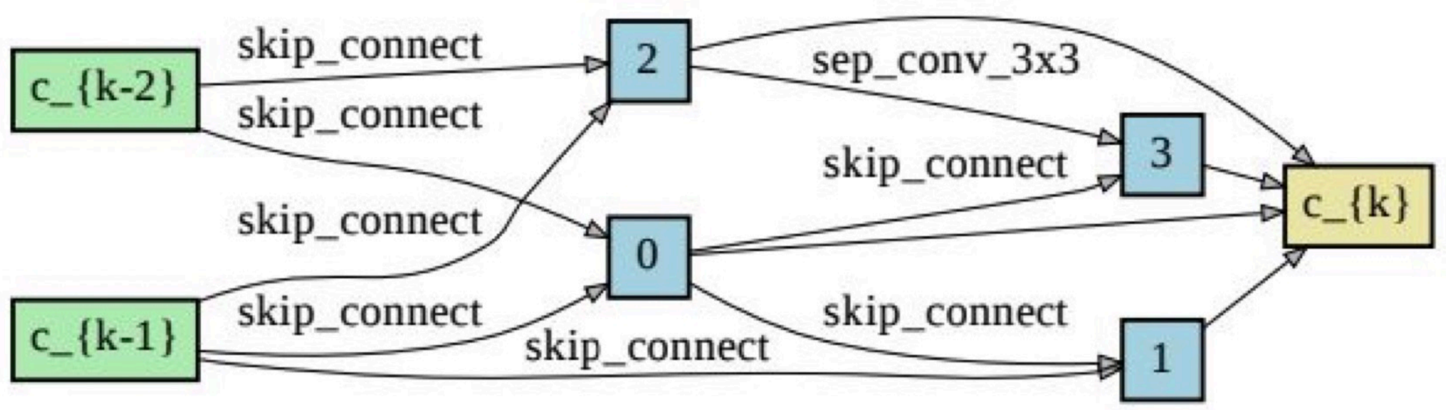
Test error of final architecture

Differentiable NAS

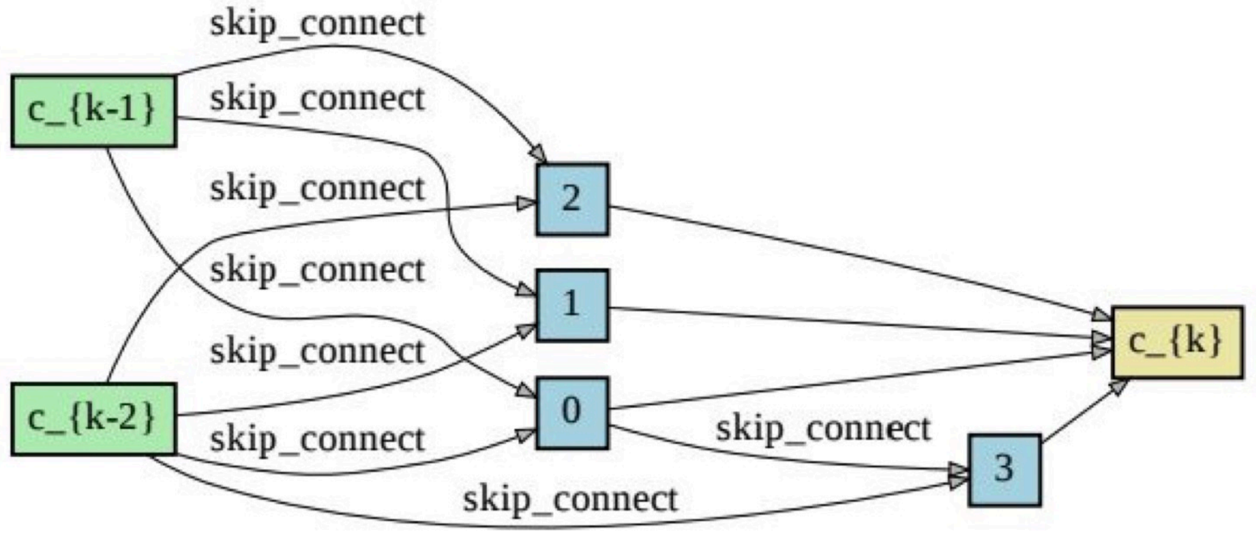
DARTS leads to degenerated solutions



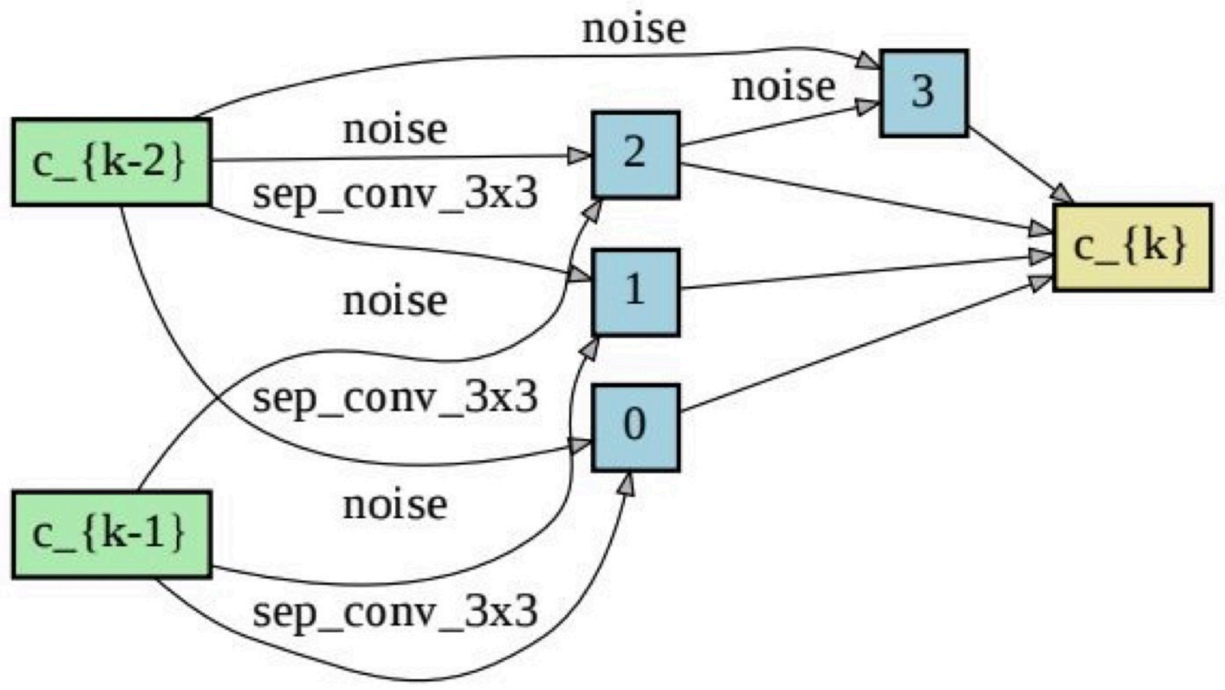
S1



S2



S3



S4

Differentiable NAS

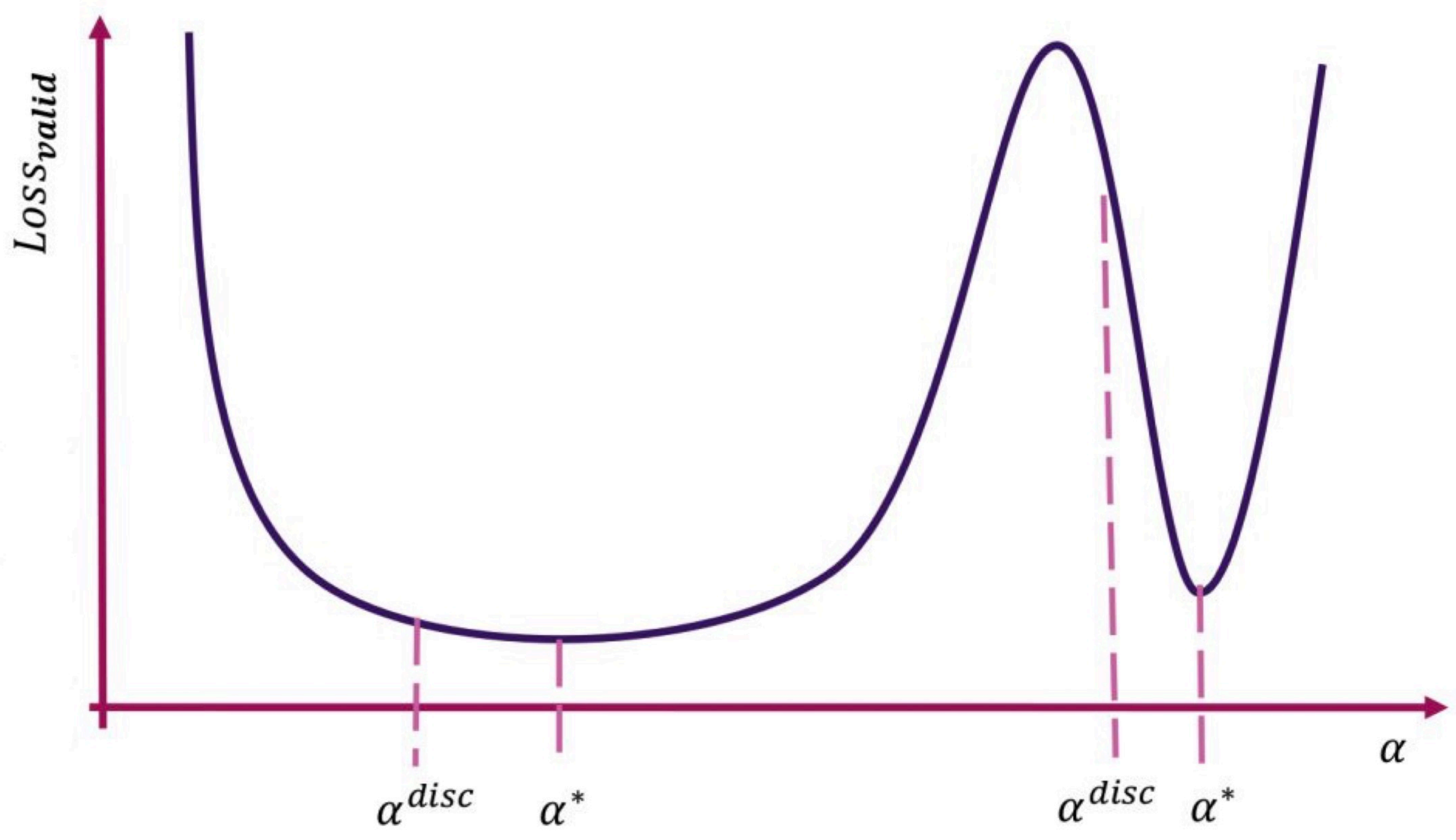
Reason 1: sharpness of the solution

- A good continuous solution doesn't imply a good discrete solution
- Gap between continuous and discrete solutions can be estimated by sharpness
  - Assume α^* is the continuous solution and $\bar{\alpha}$ is the discrete solution
  - Based on Taylor expansion:
    $L_{\text{val}}(w^*, \bar{\alpha}) \approx L_{\text{val}}(w^*, \alpha^*) + \frac{1}{2}(\bar{\alpha} - \alpha^*)^T H(\bar{\alpha} - \alpha^*)$ where
    $H = \nabla^2_{\alpha} L_{\text{val}}(w^*, \alpha^*)$ is the Hessian
  - Standard DARTS lead to "Sharp solutions" (large Hessian)

Differentiable NAS

Reason 1: sharpness of the solution

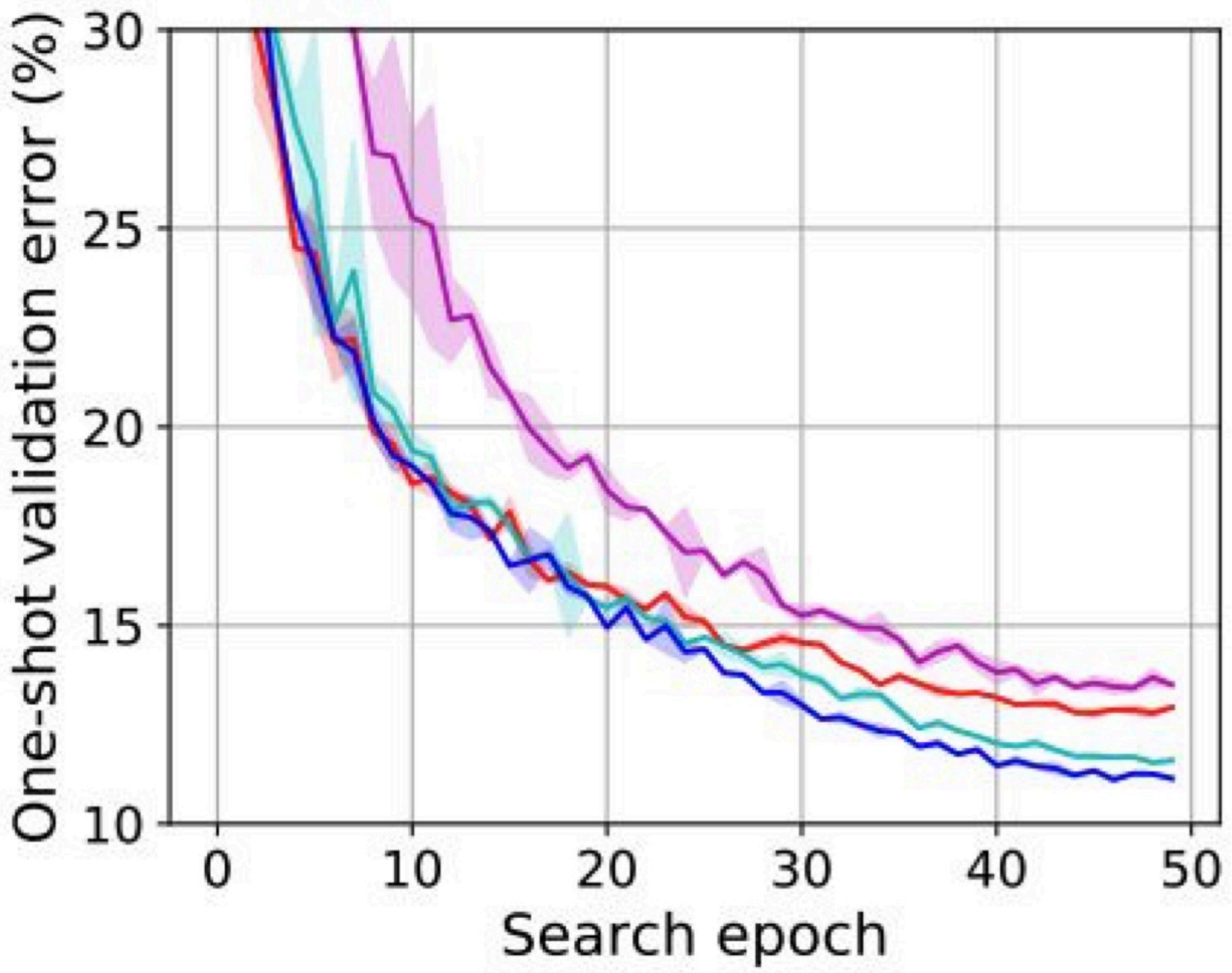
- A good continuous solution doesn't imply a good discrete solution
- Gap between continuous and discrete solutions can be estimated by sharpness
  - Standard DARTS lead to "Sharp solutions" (large Hessian)



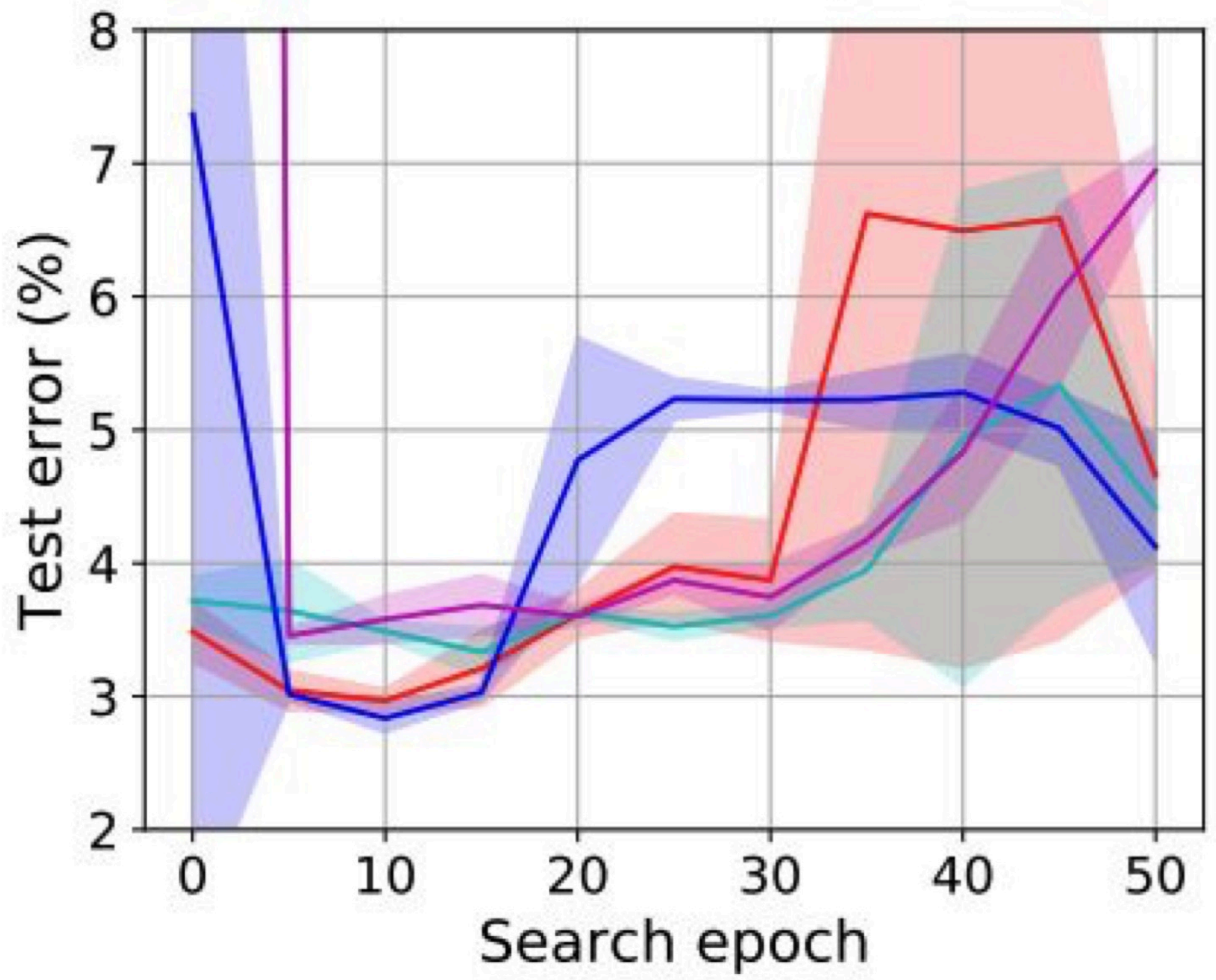
Differentiable NAS

Reason 1: sharpness of the solution

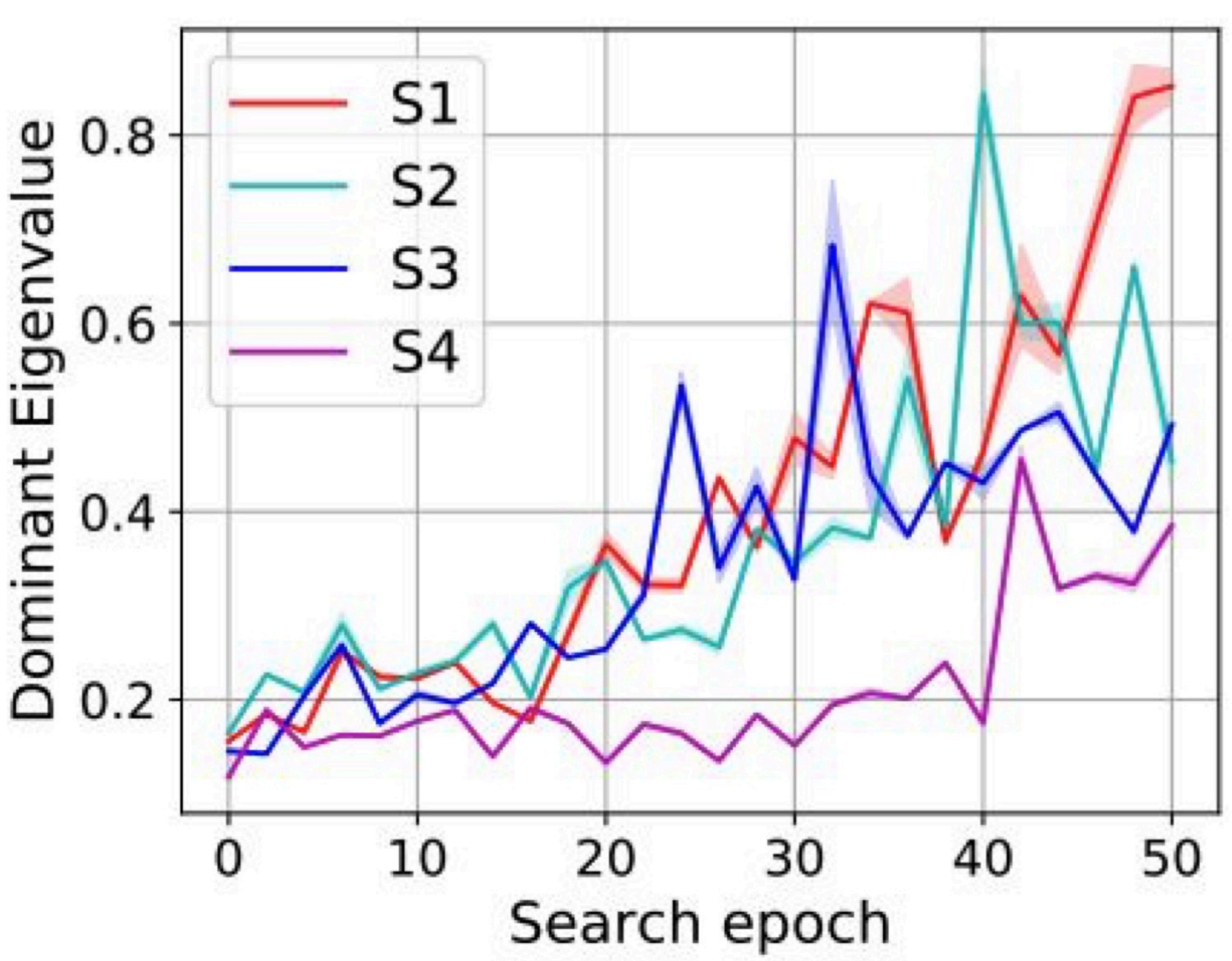
- DARTS training leads to sharp local minimums



Validation error of supernet



Test error of final architecture

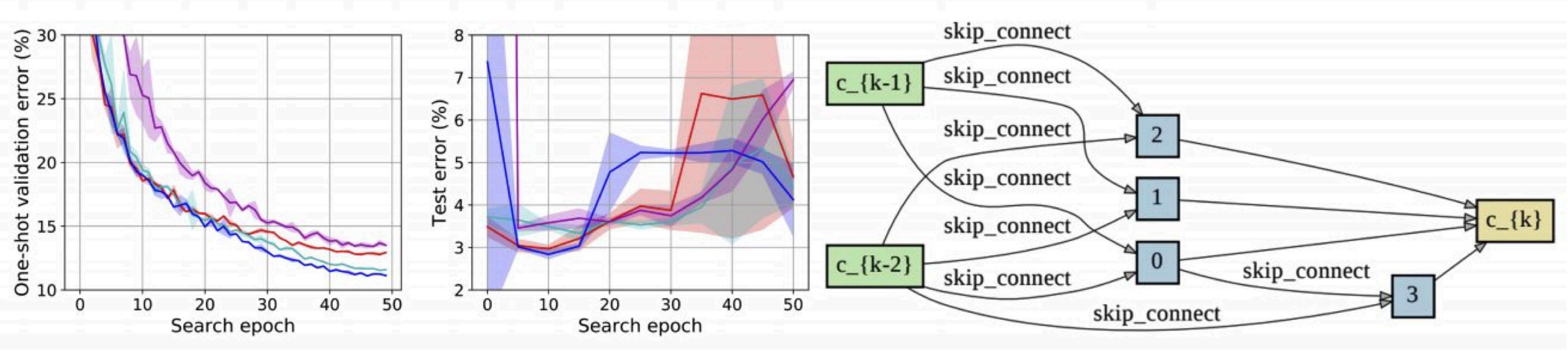


Dominant eigenvalue of Hessian

Differentiable NAS

Reason 2: Skip connection domination

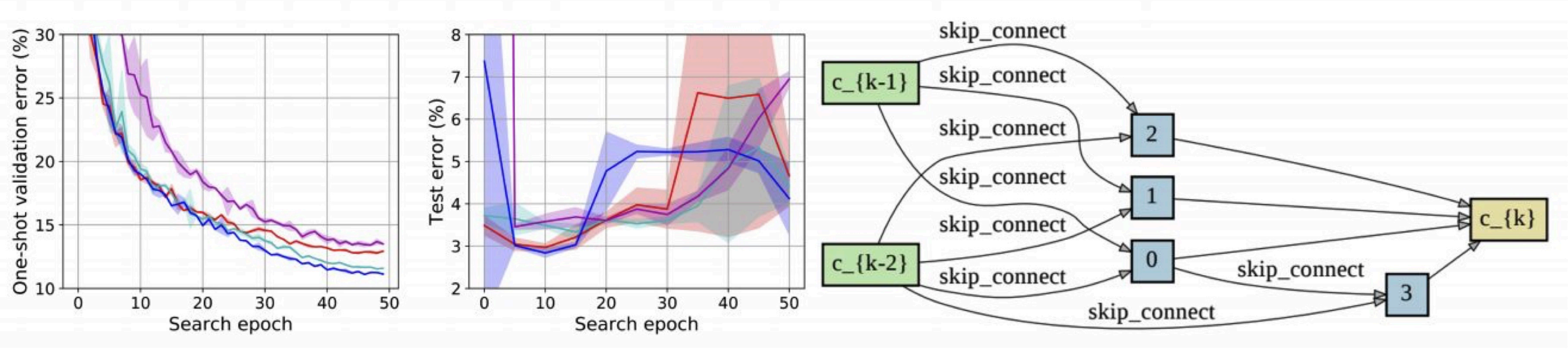
- Supernet accuracy ↑
- Weight for skip connection ↑
- Weight for convolution ↓



Differentiable NAS

Reason 2: Skip connection domination

- Supernet accuracy ↑
- Weight for skip connection ↑
- Weight for convolution ↓



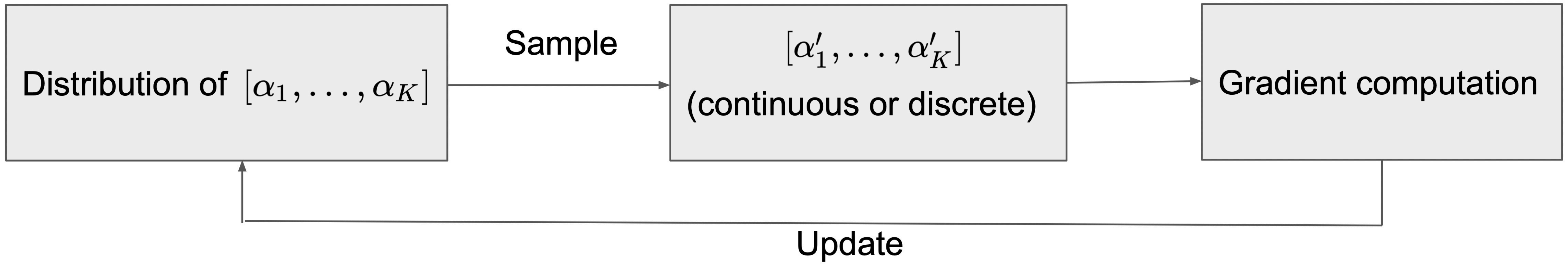
- Formally, we proved that for the optimal supernet, as number of layers goes to infinity, $\alpha_{\text{skip}} \uparrow 1$ and $\alpha_{\text{conv}} \downarrow 0$

Improvements over DARTS

- Supernet Training
  - Usually aim to make superset more “discreterizable”
  - Balance exploration and exploitation
- Scalability
  - How to use more blocks in searching?
  - Reduce memory overhead to directly search on larger problems
- Architecture Selection
  - Does architecture weight α really indicate their performance

Supernet training: Distribution learning

- Rethink DARTS as a distribution learning problem
    - For each edge, $[\alpha_1, \ldots, \alpha_k]$ defines a distribution over operations
    - We eventually "sample" an architecture from this distribution
    - How to learn $[\alpha_1, \ldots, \alpha_k]$ based on gradient-based optimization?
- Benefits:
    - Performance will be preserved better after discretization
    - Reduced training time in some cases



Supernet training: Distribution learning

Gumbel softmax

- Sampling from a distribution $i \sim \alpha_i / \sum_{i'} \alpha_{i'}$ (can't backprop from i to $\boldsymbol{\alpha}$)
- Gumbel-max: this is equivalent to

$$i = \arg\max_{i'} \{G_{i'} + \log(\alpha_i)\}$$

where each $G_{i'} \sim \text{Gumbel}(0, 1)$

  - Gumbel-softmax: using softmax with temperature annealed to be close to zero

$$z_i = \frac{\exp(G_i + \log(\alpha_i))/\gamma}{\sum_{i'} \exp(G_{i'} + \log(\alpha_{i'}))/\gamma}$$

  - This enables back-propagation to $[\alpha_1, \ldots, \alpha_K]$ (reparameterization trick)
  - SNAS: use Gumbel softmax with annealed temperature in DARTS

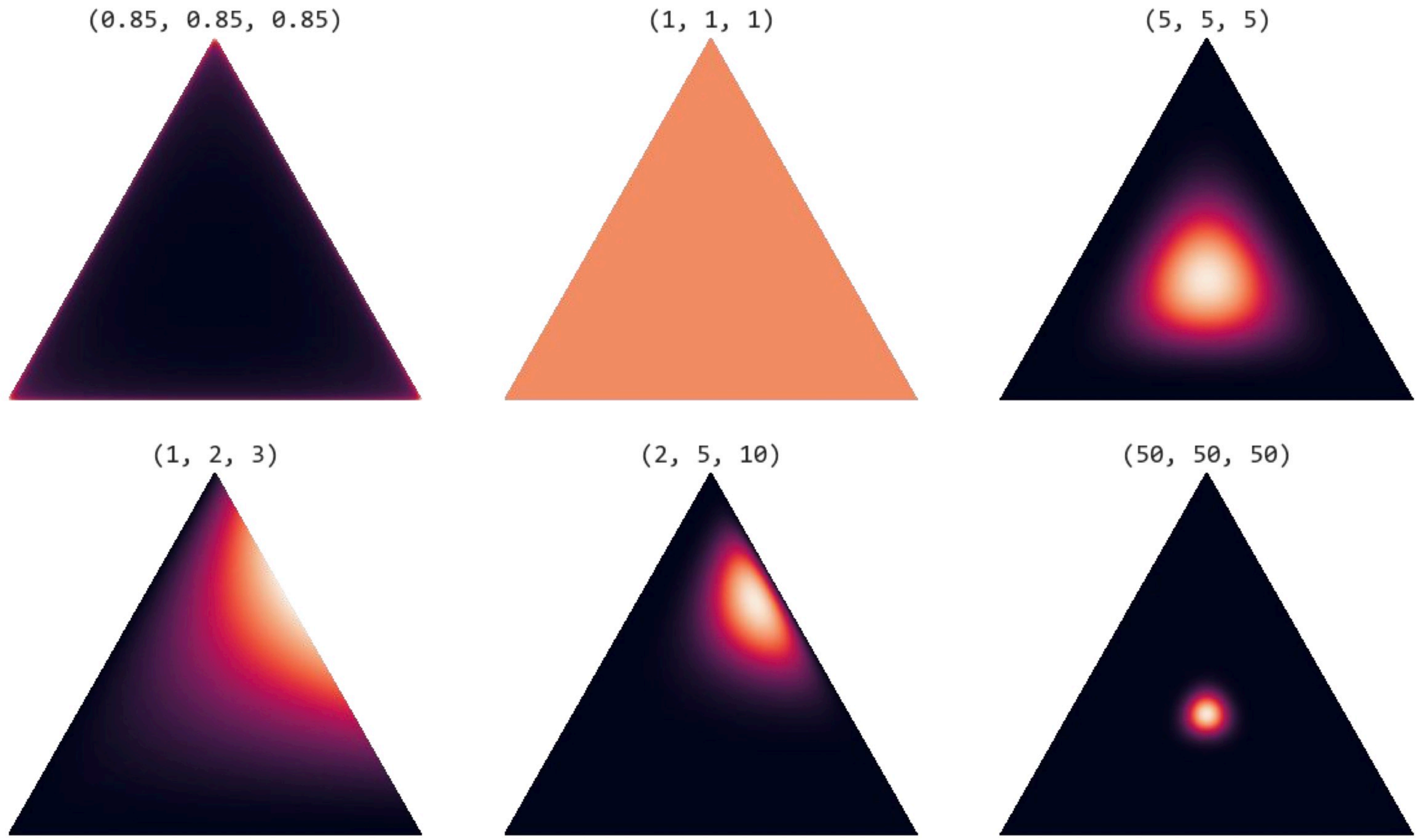
Supernet training: Distribution learning

DrNAS

- Assume architecture parameters $[\alpha_1, \ldots, \alpha_K]$ are sampled from Dirichlet Distribution:

$$[\alpha_1, \ldots, \alpha_K] \sim \mathbf{Dir}([\beta_1, \ldots, \beta_K])$$

- Dirichlet distribution samples from the standard K-1 simplex
  - $\boldsymbol{\beta} \ll 1$ leads to sparse samples with high variance
  - $\boldsymbol{\beta} \gg 1$ leads to dense samples with low variance (for sufficient exploration)



Supernet training: Distribution learning
DrNAS

- DrNAS objective:
  - Point estimation → distribution learning

$$\min_{\beta} E_{q(\alpha|\beta)}[L_{val}(w^*(\alpha),\alpha)] + \lambda d(\beta,\hat{\beta}), \quad s.t.$$

$$w^* = \arg\min_{w} L_{train}(w,\alpha), \quad q(\alpha|\beta) \sim Dir(\beta)$$

  - Gradient computation:

$$\frac{d\alpha_i}{d\beta_j} = -\frac{\frac{\partial F_{Beta}}{\partial \beta_j}(\alpha_j|\beta_j,\beta_{tot}-\beta_j)}{f_{Beta}(\alpha_j|\beta_j,\beta_{tot}-\beta_j)} \times \left(\frac{\delta_{ij}-\alpha_i}{1-\alpha_j}\right)$$

  - Architecture selection: magnitude of $\boldsymbol{\beta}$

Supernet training: Distribution learning
DrNAS

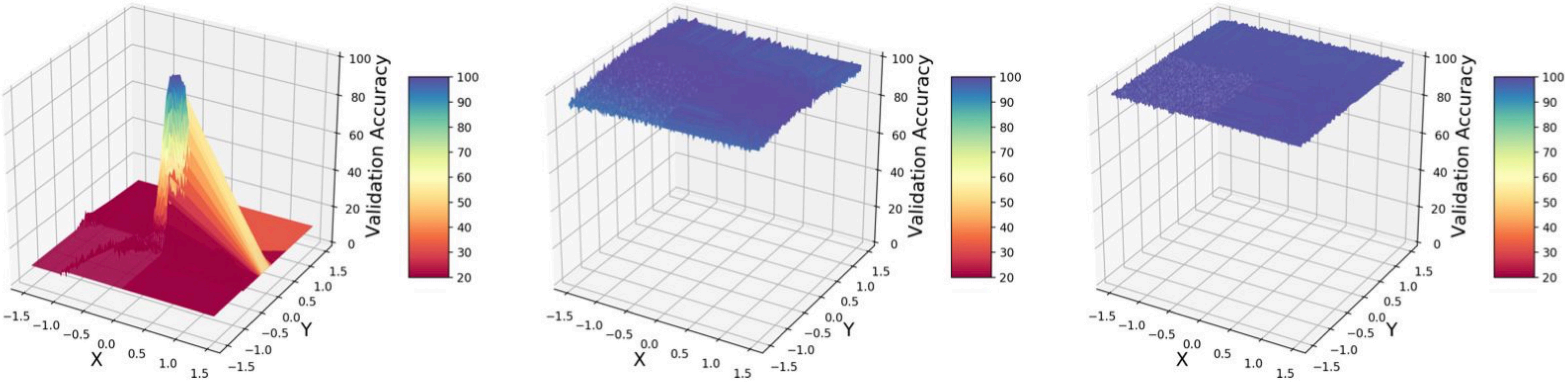
- On NAS-Bench-201
  - Achieve oracle when searching on CIFAR-100
    DrNAS (73.51) vs SNAS (69.34) vs DARTS (38.97)

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	validation	test	validation	test	validation	test
ResNet	90.83	93.97	70.42	70.86	44.53	43.63
Random (baseline)	90.93 ± 0.36	93.70 ± 0.36	70.60 ± 1.37	70.65 ± 1.38	42.92 ± 2.00	42.96 ± 2.15
RSPS	84.16 ± 1.69	87.66 ± 1.69	45.78 ± 6.33	46.60 ± 6.57	31.09 ± 5.65	30.78 ± 6.12
Reinforce	91.09 ± 0.37	93.85 ± 0.37	70.05 ± 1.67	70.17 ± 1.61	43.04 ± 2.18	43.16 ± 2.28
ENAS	39.77 ± 0.00	54.30 ± 0.00	10.23 ± 0.12	10.62 ± 0.27	16.43 ± 0.00	16.32 ± 0.00
DARTS (1st)	39.77 ± 0.00	54.30 ± 0.00	38.57 ± 0.00	38.97 ± 0.00	18.87 ± 0.00	18.41 ± 0.00
DARTS (2nd)	39.77 ± 0.00	54.30 ± 0.00	38.57 ± 0.00	38.97 ± 0.00	18.87 ± 0.00	18.41 ± 0.00
GDAS	90.01 ± 0.46	93.23 ± 0.23	24.05 ± 8.12	24.20 ± 8.08	40.66 ± 0.00	41.02 ± 0.00
SNAS	90.10 ± 1.04	92.77 ± 0.83	69.69 ± 2.39	69.34 ± 1.98	42.84 ± 1.79	43.16 ± 2.64
DSNAS	89.66 ± 0.29	93.08 ± 0.13	30.87 ± 16.40	31.01 ± 16.38	40.61 ± 0.09	41.07 ± 0.09
PC-DARTS	89.96 ± 0.15	93.41 ± 0.30	67.12 ± 0.39	67.48 ± 0.89	40.83 ± 0.08	41.31 ± 0.22
DrNAS	91.55 ± 0.00	94.36 ± 0.00	73.49 ± 0.00	73.51 ± 0.00	46.37 ± 0.00	46.34 ± 0.00
optimal	91.61	94.37	73.49	73.51	46.77	47.31

Supernet training

Perturbation-based regularization

- A smoother landscape will make supernet robust to discreterization



(a) DARTS (b) SDARTS-RS (c) SDARTS-ADV

Supernet training

Perturbation-based regularization

- Make supernet robust to α perturbation
  - Since we need to perturb it to a discrete architecture in the final stage
- Mathematically, we hope the superset robust to random or adversarial (worst-case) perturbation of α

Supernet training

Perturbation-based regularization

- Make supernet robust to α perturbation
  - Since we need to perturb it to a discrete architecture in the final stage
- Mathematically, we hope the superset robust to random or adversarial (worst-case) perturbation of α

$$\min_\alpha L_{\text{val}}(\bar{w}(\alpha), \alpha), \quad \text{s.t.}$$

SDARTS-RS: $\bar{w}(\alpha) = \arg\min_w E_{\delta \sim U_{[-\epsilon,\epsilon]}} L_{\text{train}}(w, A+\delta)$

SDARTS-Adv: $\bar{w}(\alpha) = \arg\min_w \max_{\|\delta\| \leq \epsilon} L_{\text{train}}(w, A+\delta)$

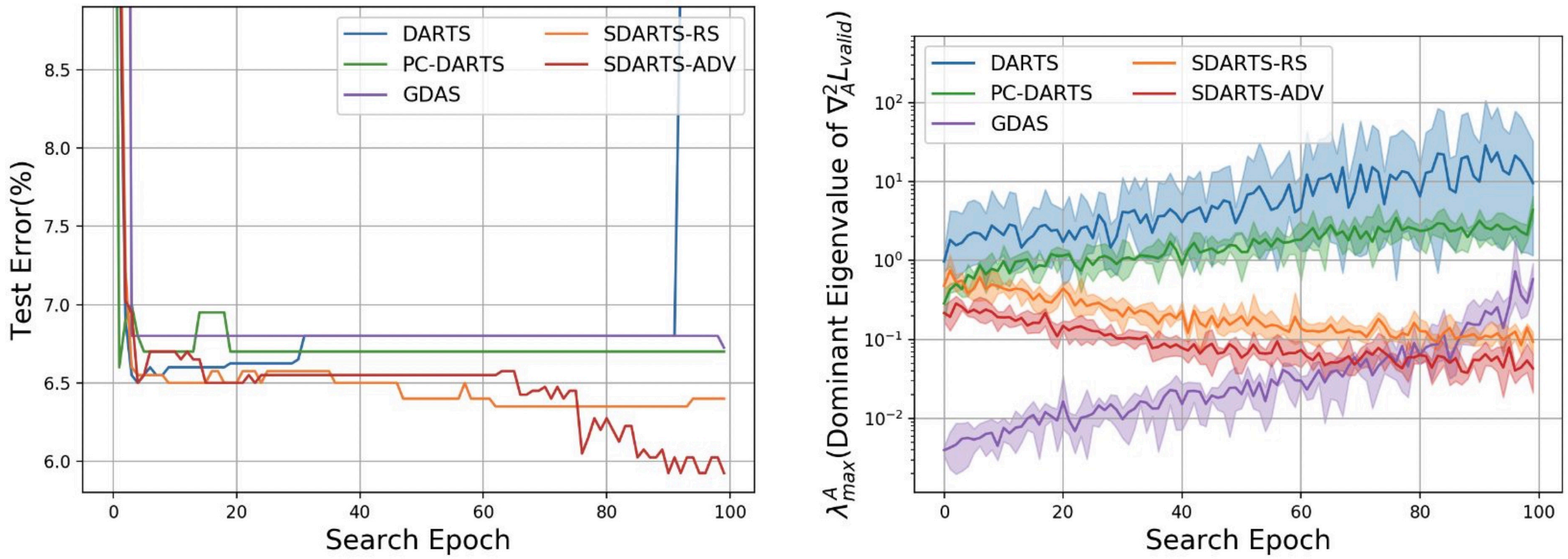
SDARTS: Each step

- ➔ Perturb α
  - ◆ Random: $\alpha' \leftarrow \alpha + N(0, \sigma^2)$
  - ◆ Adversarial:

    $\alpha' \leftarrow \alpha + \nabla_\alpha L_{\text{train}}(\alpha, w)$
- ➔ Update w based on α'
- ➔ Update α based on w

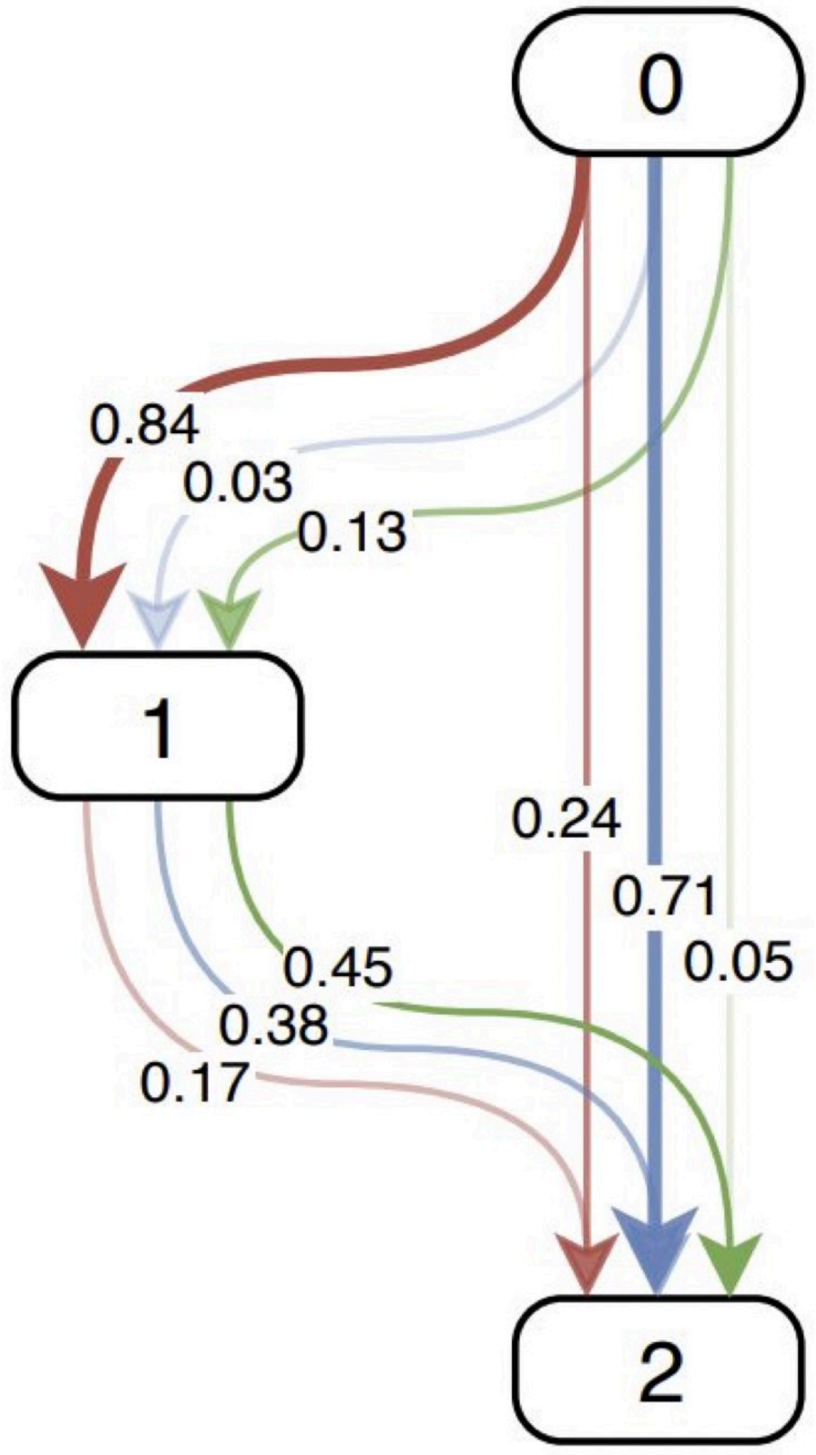
Supernet training
SmoothDARTS

- On NAS-Bench-1Shot1
  - Continues to discover better architectures
  - Anneal Hessian to a low level

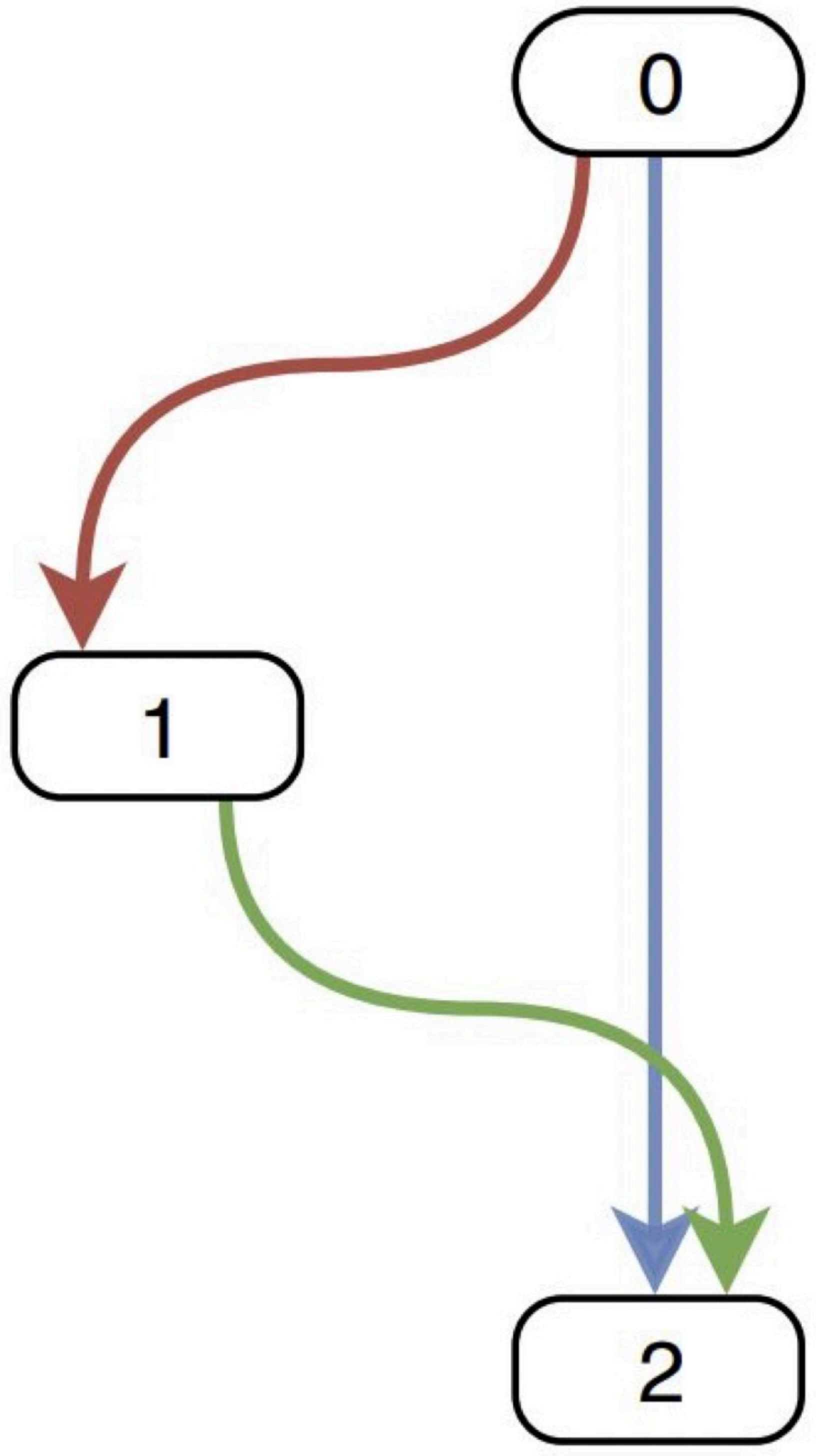


Architecture Selection

Architecture Selection in DARTS



(e) Search end



(f) Final cell

Architecture Selection

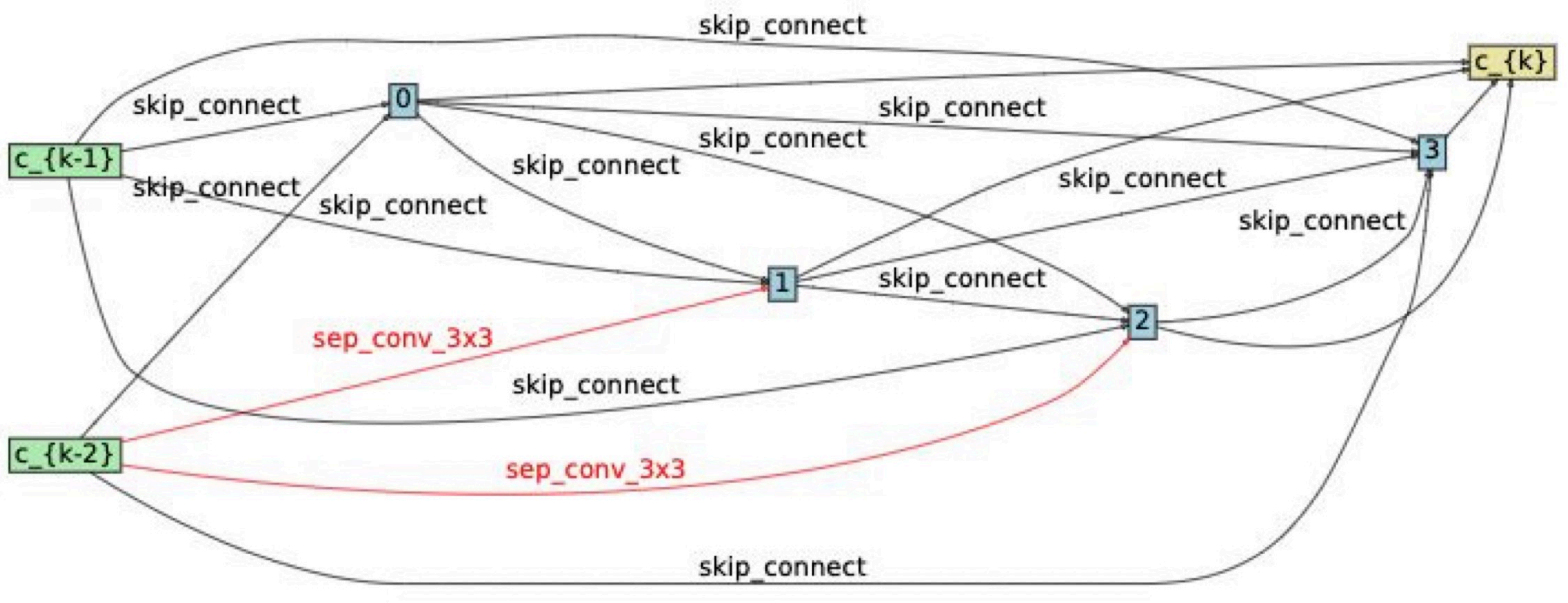
Architecture Selection in DARTS

- Recall the skip-domination problem:
  - For the optimal supernet with infinite number of layers: $\alpha_{\text{skip}} \uparrow 1 \quad \text{and} \quad \alpha_{\text{conv}} \downarrow 0$
  - ***α*** values may not really represent the ***"importance"*** of each operation
- Skip connection stands out if we select the best operation based on ***α***
- Does $\alpha_{\text{skip}} > \alpha_{\text{conv}}$ mean skip connection is better than convolution?

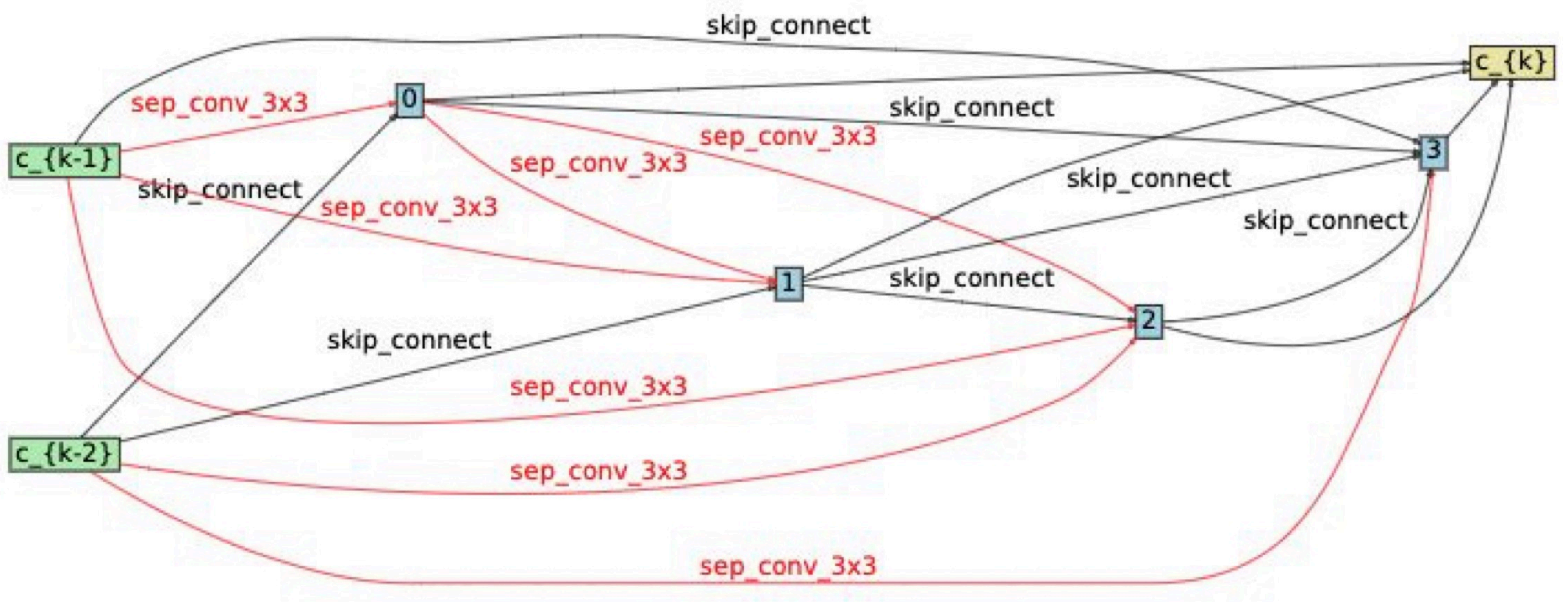
Architecture Selection

Does α represent operation strength?

- Probably Not!
- S2: (Skip_connect, sep_conv_3x3)
  - Skip connections dominate according to $\boldsymbol{\alpha}$
  - But the accuracy of S2 supernet benefits from more convolutions



Magnitude-base selection



Progressive tuning selection

Architecture Selection

Does α represent operation strength?

- Same observations on large space: DARTS space
  - Magnitude of $\boldsymbol{\alpha}$ deviates from accuracy of the supernet
  - Some operations with small $\boldsymbol{\alpha}$ are in fact more important for supernet

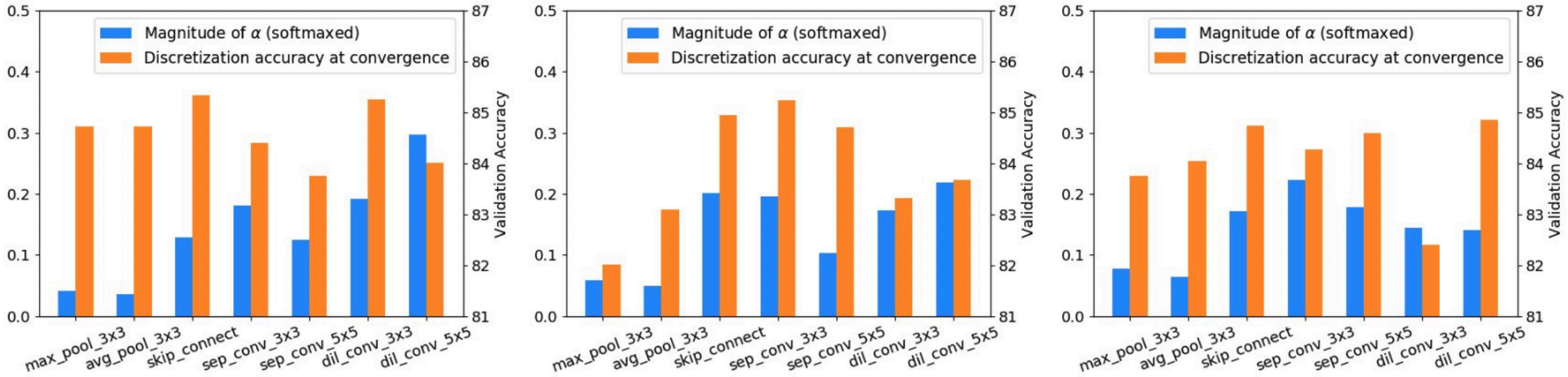


Figure: Magnitude of α vs Accuracy after choosing one operation

Architecture Selection

A New architecture Selection Method

- Evaluate the importance of an operation o by:
    - Compute the drop of validation accuracy when o is removed (no need for further training)
- Use this to choose the best o for an edge
- Fine-tune the solution, and move to the next edge
- “Pertubation-based selection” (PT for short)

Architecture Selection
A New architecture Selection Method

- PT consistently improves over the original magnitude-based selection

Dataset	Space	DARTS	DARTS+PT (Ours)
C10	S1	3.84	3.50
	S2	4.85	2.79
	S3	3.34	2.49
	S4	7.20	2.64
C100	S1	29.46	24.48
	S2	26.05	23.16
	S3	28.90	22.03
	S4	22.85	20.80
SVHN	S1	4.58	2.62
	S2	3.53	2.53
	S3	3.41	2.42
	S4	3.05	2.42

Architecture Selection

A New architecture Selection Method

- Performance improves with more searching epochs

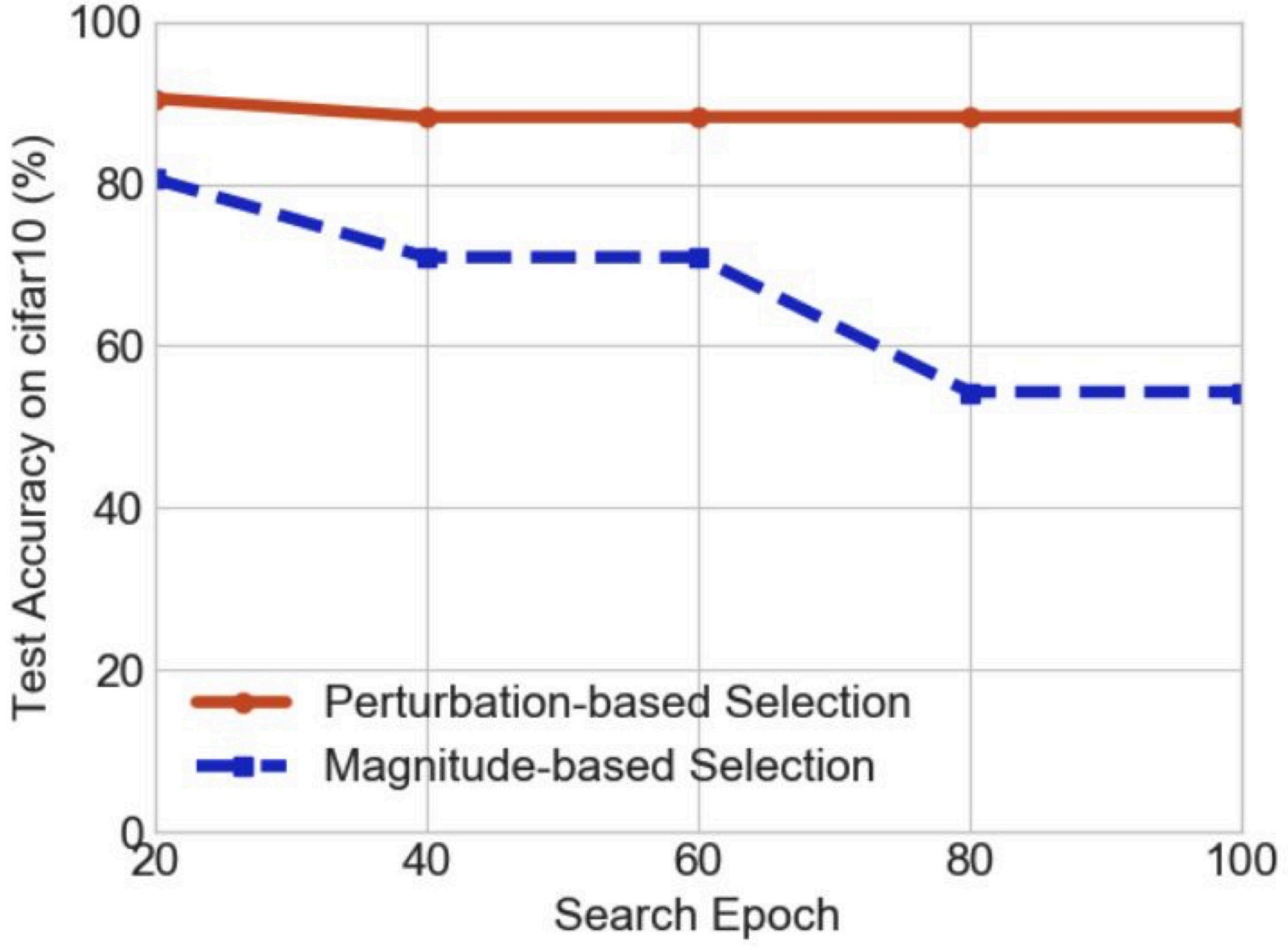


Figure: Test accuracy vs search epoch on NAS-Bench-201 space

Architecture Selection
A New architecture Selection Method

Architecture	Test Error (%)	Search Cost (GPU days)	Search Method
DARTS (1st) (Liu et al., 2019)	3.00 ± 0.14	0.4	differentiable
DARTS (2nd) (Liu et al., 2019)	2.76 ± 0.09	1.0	differentiable
SNAS (moderate) (Xie et al., 2019)	2.85 ± 0.02	1.5	differentiable
DrNAS (Chen et al., 2020)	2.54 ± 0.03	0.4	differentiable
NASP (Yao et al., 2019)	2.83 ± 0.09	0.1	differentiable
SDARTS-ADV (Chen & Hsieh, 2020)	2.61 ± 0.02	1.3	differentiable
ProxylessNAS (Cai et al., 2019)†	2.08	4.0	differentiable
PC-DARTS (Xu et al., 2020)	2.57 ± 0.07	0.1	differentiable
DrNAS (with progressive learning)	2.46 ± 0.03	0.6	differentiable
DARTS+PT (Wang et al., 2020)	2.61 ± 0.08	0.8	differentiable
SDARTS-ADV+PT	2.54 ± 0.01	0.8	differentiable

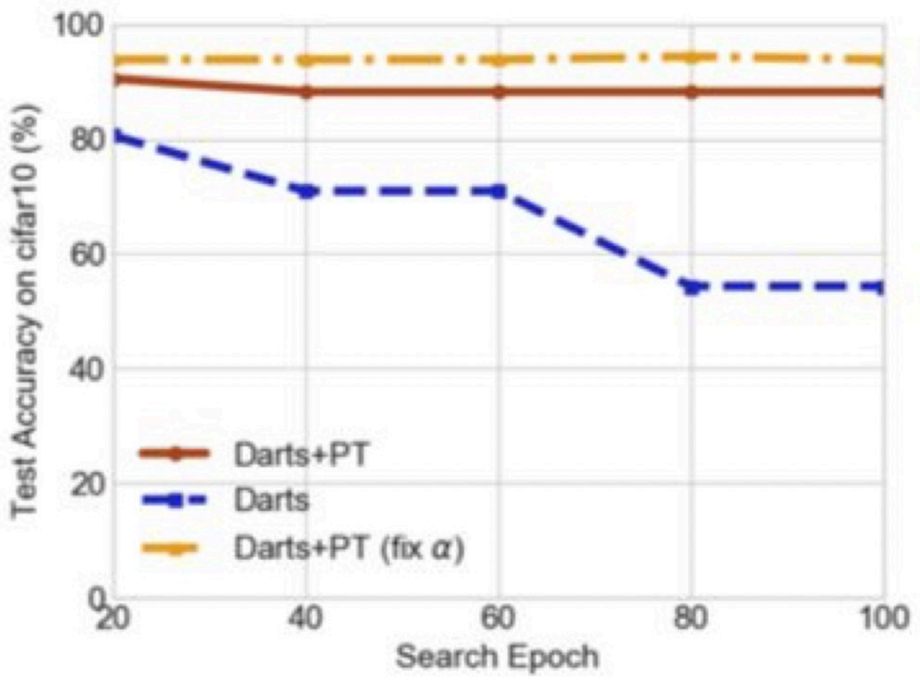
† Obtained on a different space with PyramidNet as the backbone.

Architecture Selection

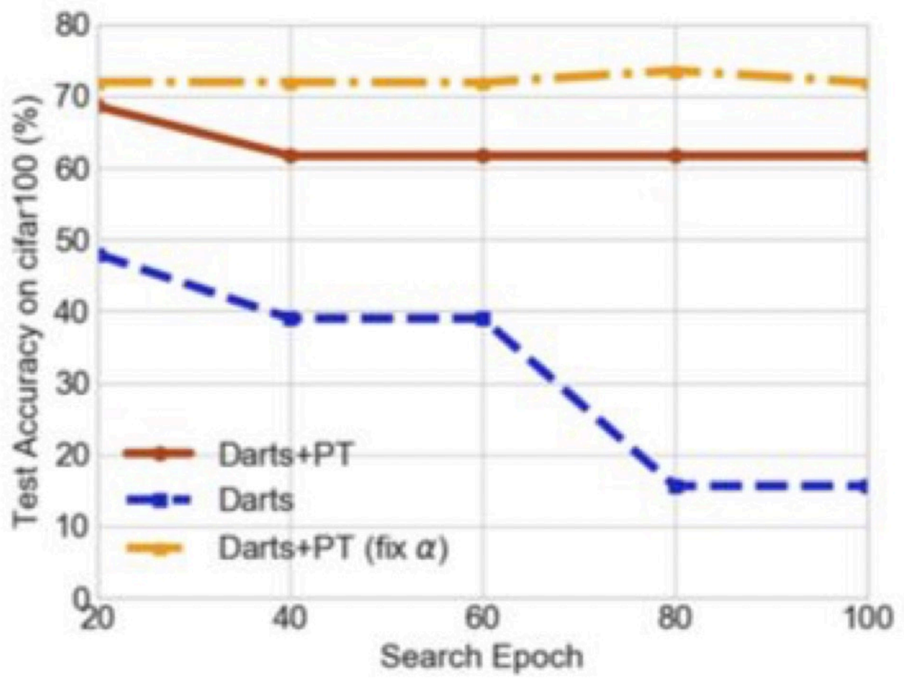
A New architecture Selection Method

Table 3: Darts+PT on S1-S4 (test error (%)).

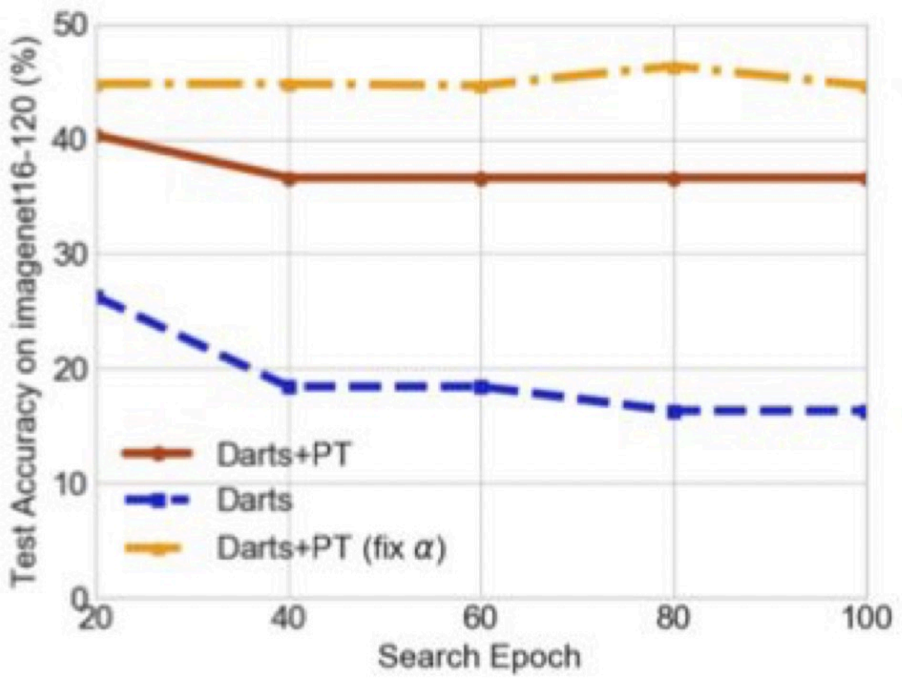
Dataset	Space	DARTS	Darts+PT (Ours)	Darts+PT (fix α)*
C10	S1	3.84	3.50	2.86
	S2	4.85	2.79	2.59
	S3	3.34	2.49	2.52
	S4	7.20	2.64	2.58
C100	S1	29.46	24.48	24.40
	S2	26.05	23.16	23.30
	S3	28.90	22.03	21.94
	S4	22.85	20.80	20.66
SVHN	S1	4.58	2.62	2.39
	S2	3.53	2.53	2.32
	S3	3.41	2.42	2.32
	S4	3.05	2.42	2.39



CIFAR10



CIFAR100



SVHN