



# CENG 5030

# Energy Efficient Computing

## Lecture 10: Network Architecture Search

**Bei Yu**

(Latest update: December 19, 2020)

Spring 2021



Search Space Design

Blackbox Optimization

- NAS as a hyperparameter optimization

- Reinforcement Learning

- Evolution methods

- Regularized methods

- Baysian Optimization

- Differentiable search

- Efficient methods

NAS Benchmark

Estimation strategy



## Search Space Design

### Blackbox Optimization

NAS as a hyperparameter optimization

Reinforcement Learning

Evolution methods

Regularized methods

Baysian Optimization

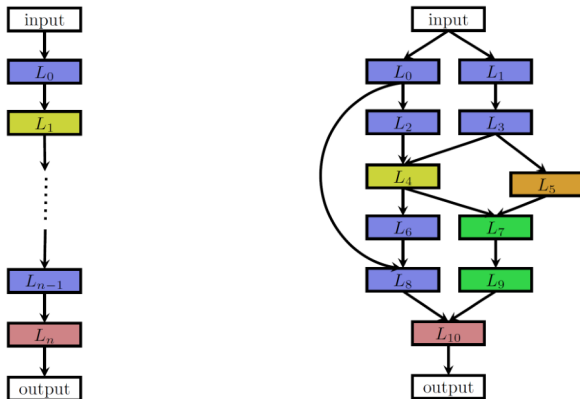
Differentiable search

Efficient methods

### NAS Benchmark

### Estimation strategy

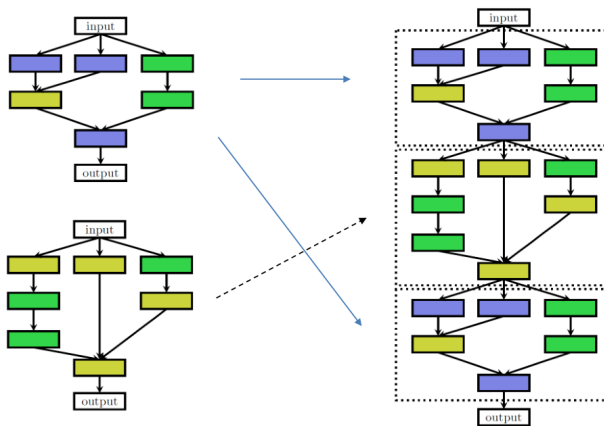
# Basic architecture search



Each node in the graphs corresponds to a layer in a neural network <sup>1</sup>

<sup>1</sup>Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter (2018). "Neural architecture search: A survey". In: *arXiv preprint arXiv:1808.05377*

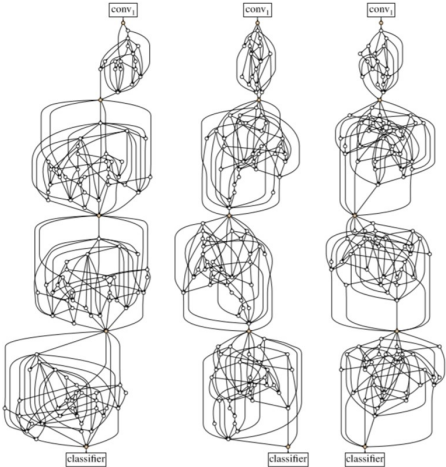
# Cell-based search



Normal cell and reduction cell can be connected in different order<sup>2</sup>

<sup>2</sup>Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter (2018). "Neural architecture search: A survey". In: *arXiv preprint arXiv:1808.05377*

# Graph-based search space



Randomly wired neural networks generated by the classical Watts-Strogatz model<sup>3</sup>

<sup>3</sup>Saining Xie et al. (2019). “Exploring randomly wired neural networks for image recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1284–1293



Search Space Design

Blackbox Optimization

- NAS as a hyperparameter optimization

- Reinforcement Learning

- Evolution methods

- Regularized methods

- Baysian Optimization

- Differentiable search

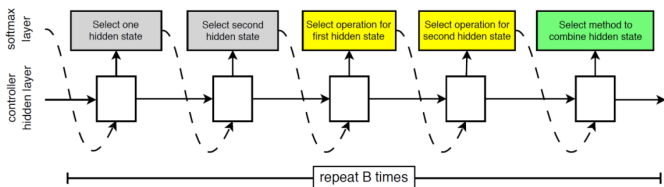
- Efficient methods

NAS Benchmark

Estimation strategy



# NAS as hyperparameter optimization



Controller architecture for recursively constructing one block of a convolutional cell <sup>4</sup>

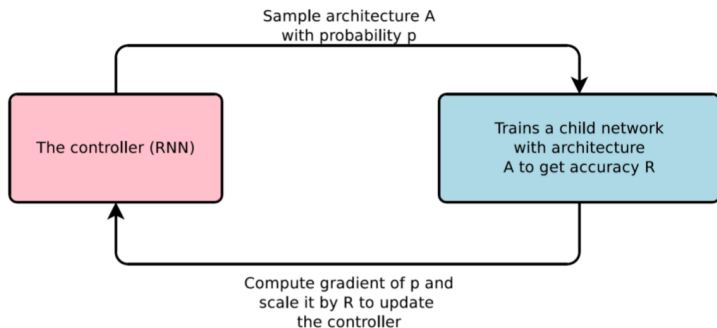
## Features

- ▶ 5 categorical choices for  $N^{th}$  block
  - ▶ 2 categorical choices of hidden states, each with domain  $0, 1, \dots, N - 1$
  - ▶ 2 categorical choices of operations
  - ▶ 1 categorical choices of combination method
  - ▶ Total number of hyperparameters for the cell:  $5B$  (with  $B = 5$  by default)
- ▶ Unrestricted search space
  - ▶ Possible with conditional hyperparameters (but only up to a prespecified maximum number of layers)
  - ▶ Example: chain-structured search space
    - ▶ Top-level hyperparameter: number of layers  $L$
    - ▶ Hyperparameters of layer  $K$  conditional on  $L \geq k$

<sup>4</sup>Barret Zoph, Vijay Vasudevan, et al. (2018). "Learning transferable architectures for scalable image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710



# Reinforcement learning



Overview of the reinforcement learning method with RNN <sup>5</sup>

## Reinforcement learning with a RNN controller

- ▶ State-of-the-art results for CIFAR-10, Penn Treebank
- ▶ Large computation demands  
**800 GPUs for 3-4 weeks, 12, 800 architectures evaluated**

<sup>5</sup>Barret Zoph and Quoc V Le (2016). "Neural architecture search with reinforcement learning". In: *arXiv preprint arXiv:1611.01578*



## Reinforcement learning with a RNN controller

$$J(\theta_c) = E_{P(a_{1:T}; \theta_c)}[R]$$

where  $R$  is the reward (e.g., accuracy on the validation dataset)

## Apply REINFORCEMENT rule

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

Use Monte Carlo approximation with control variate methods, the gradient can be approximated by

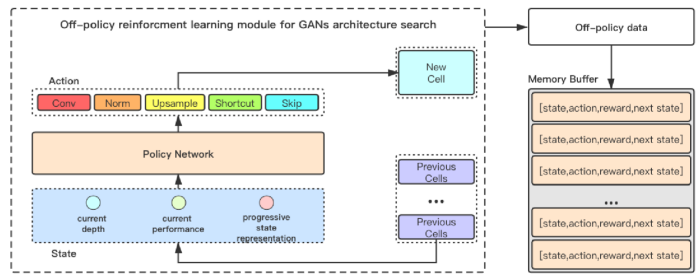
## Approximation of gradients

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



# Reinforcement Learning

Another example on GAN search: [Yuan Tian et al. \(2020\)](#). “Off-policy reinforcement learning for efficient and effective gan architecture search”. In: *arXiv preprint arXiv:2007.09180*



Overview of the  $E^2GAN$ <sup>6</sup>

## Reward define

$$R_t(s, a) = IS(t) - IS(t - 1) + \alpha(FID(t - 1) - FID(t))$$

## The objective loss function

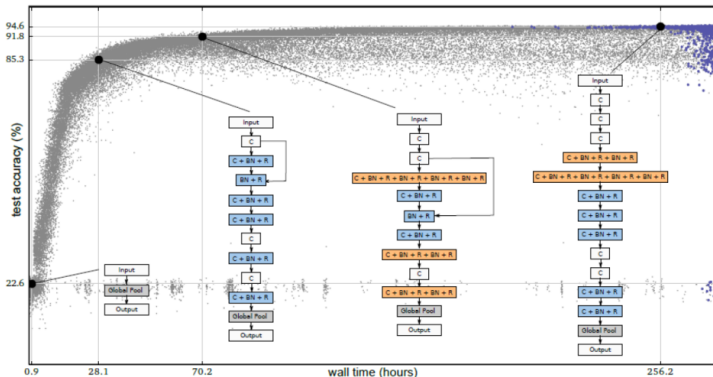
$$J(\pi) = \sum_{t=0} \mathbb{E}_{(s_t, a_t) \sim p(\pi)} R(s_t, a_t) = \mathbb{E}_{architecture \sim p(\pi)} IS_{final} - \alpha FID_{final}$$

<sup>6</sup>[Yuan Tian et al. \(2020\)](#). “Off-policy reinforcement learning for efficient and effective gan architecture search”. In: *arXiv preprint arXiv:2007.09180*

## Evolution methods

Neuroevolution (already since the 1990s)

- ▶ Typically optimized both architecture and weights with evolutionary methods e.g., Angeline, Saunders, and Pollack 1994; Stanley and Miikkulainen 2002
- ▶ Mutation steps, such as adding, changing or removing a layer e.g., Real, Moore, et al. 2017; Miikkulainen et al. 2017



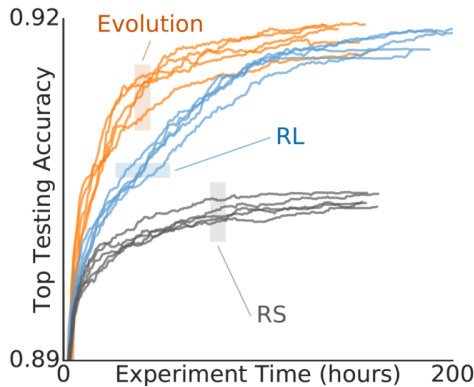
# Regularized / Aging Evolution



## Regularized / Aging Evolution methods

- ▶ Standard evolutionary algorithm e.g. Real, Aggarwal, et al. [2019](#)  
But oldest solutions are dropped from the population (even the best)
- ▶ State-of-the-art results (CIFAR-10, ImageNet)  
Fixed-length cell search space

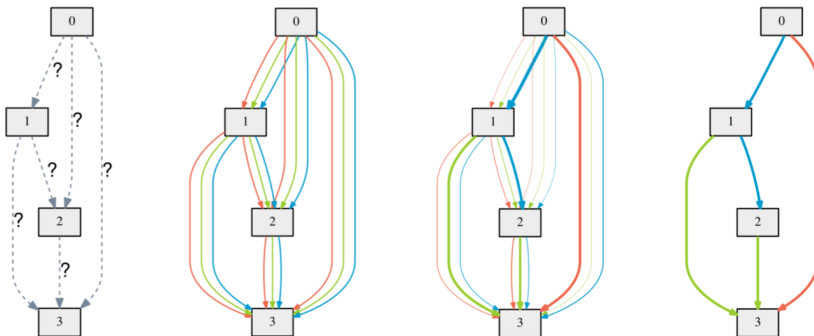
Comparison of  
evolution,  
RL and  
random search





## Bayesian optimization methods

- ▶ Joint optimization of a vision architecture with 238 hyperparameters with TPE  
Bergstra, Yamins, and Cox [2013](#)
- ▶ Auto-Net
  - ▶ Joint architecture and hyperparameter search with SMAC
  - ▶ First Auto-DL system to win a competition dataset against human experts  
Mendoza et al. [2016](#)
- ▶ Kernels for GP-based NAS
  - ▶ Arc kernel  
Swersky, Snoek, and Adams [2013](#)
  - ▶ NASBOT  
Kandasamy et al. [2018](#)
- ▶ Sequential model-based optimization
  - ▶ PNAS  
C. Liu et al. [2018](#)



Overview of SNAS <sup>7</sup>

## Continuous relaxation

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

<sup>7</sup>Hanxiao Liu, Karen Simonyan, and Yiming Yang (2018). "Darts: Differentiable architecture search". In: *arXiv preprint arXiv:1806.09055*



## A bi-level optimization

$$\begin{aligned} & \min_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \text{s.t. } & w^*(\alpha) = \underset{w}{\operatorname{argmin}} \mathcal{L}_{train}(w, \alpha) \end{aligned}$$

---

### Algorithm 1 DARTS algorithm

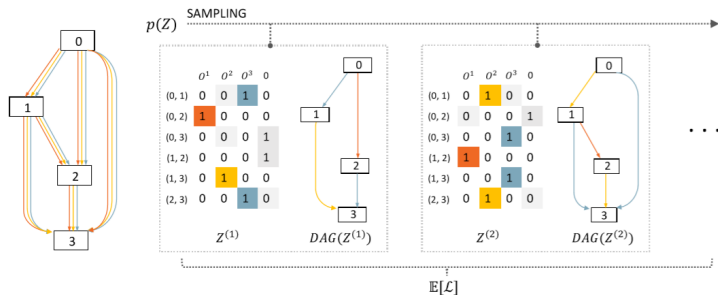
---

**Require:** Create a mixed operation  $\hat{O}^{(i,j)}$  parameterized by  $\alpha^{(i,j)}$  for each edge  $(i,j)$

**Ensure:** The architecture characterized by  $\alpha$

- 1: **while** not converged **do**
  - 2:     Update architecture  $\alpha$  by descending  $\nabla_{\alpha} \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$
  - 3:     ( $\xi = 0$  if using first order approximation)
  - 4:     Update weights  $w$  by descending  $\nabla_w \mathcal{L}_{train}(w, \alpha)$
  - 5: **end while**
  - 6: Derive the final architecture based on the learned  $\alpha$
-



Overview of SNAS <sup>8</sup>

## Stochastic NAS

$$\mathbb{E}_{Z \sim p_{\alpha}(Z)}[R(Z)] = \mathbb{E}_{Z \sim p_{\alpha}(Z)}[L_{\theta}(Z)]$$

$$x_j = \sum_{i < j} \tilde{O}_{i,j}(x_i) = \sum_{i < j} Z_{i,j}^T O_{i,j}(x_i)$$

where  $\mathbb{E}_{Z \sim p_{\alpha}(Z)}[R(Z)]$  is the objective loss,  $Z_{i,j}$  is a one-hot random variable vector to each edge  $(i, j)$  in the neural network and  $x_j$  is the intermediate node

<sup>8</sup>Sirui Xie et al. (2018). "SNAS: stochastic neural architecture search". In: *arXiv preprint arXiv:1812.09926*



## Apply Gumbel-softmax trick to relax the $p_\alpha(Z)$

$$Z_{i,j}^k = f_{\alpha_{i,j}}(G_{i,j}^k) = \frac{\exp\left(\frac{(\log \alpha_{i,j}^k + G_{i,j}^k)}{\lambda}\right)}{\sum_{l=0}^n \exp\left(\frac{\log \alpha_{i,j}^l + G_{i,j}^l}{\lambda}\right)}$$

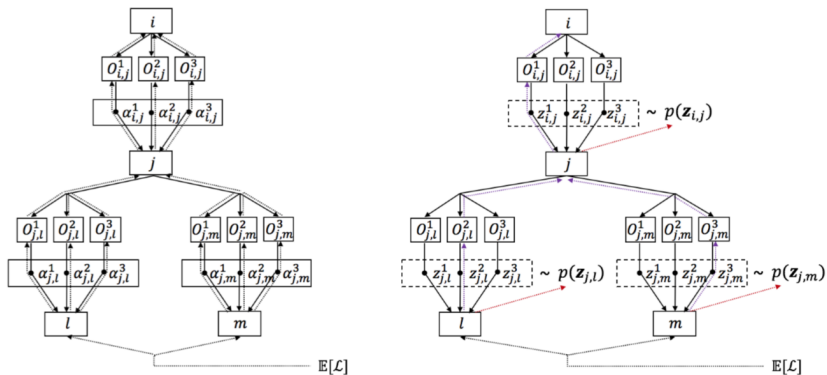
where  $Z_{i,j}$  is the softened one-hot random variable,  $\alpha_{i,j}$  is the architecture parameter,  $\lambda$  is the temperature of the Softmax function, and  $G_{i,j}^k$  satisfies that

## Gumbel distribution

$$G_{i,j}^k = -\log(-\log(U_{i,j}^k))$$

where  $U_{i,j}^k$  is a uniform random variable

# Difference between DARTS and SNAS



A comparison between DARTS (i.e., the left) and SNAS (i.e., the right) <sup>9</sup>

## Summary

- ▶ Deterministic gradients in DARTS and Stochastic gradients in SNAS
- ▶ DARTS require that the derived neural network should be retrained while SNAS has no need

<sup>9</sup>Sirui Xie et al. (2018). "SNAS: stochastic neural architecture search". In: *arXiv preprint arXiv:1812.09926*



## Main approaches for making NAS efficient

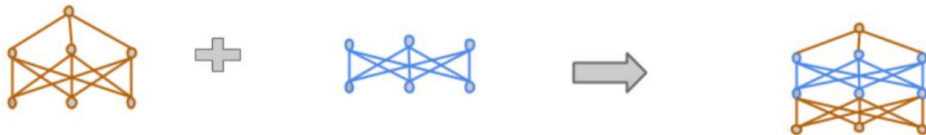
- ▶ Weight inheritance & network morphisms
- ▶ Weight sharing & one-shot models
- ▶ Discretize methods
- ▶ Multi-fidelity optimization  
Zela et al. [2018](#), Runge et al. [2018](#)
- ▶ Meta-learning  
Wong et al. [2018](#)



## Network morphisms

Wei et al. 2016

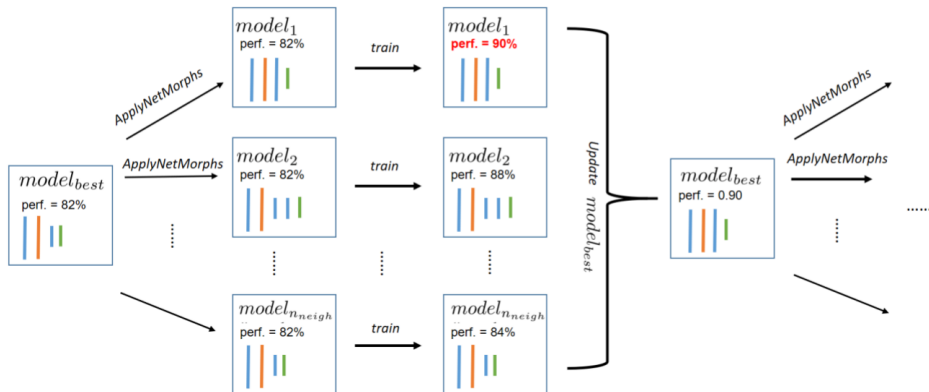
- ▶ Change the network structure, but not the modelled function  
i.e., for every input the network yields the same output as before applying the network morphism
- ▶ Allow efficient moves in architecture space



# Weight inheritance & network morphisms



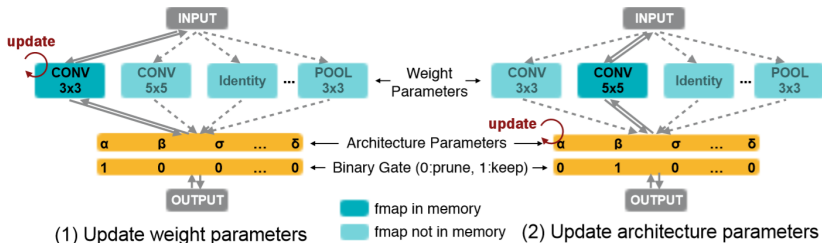
Cai, Chen, et al. 2017; Elsken, J. Metzen, and Hutter 2017; Cortes et al. 2017; Cai, J. Yang, et al. 2018





## Discretize the search space

Discretize the search space (e.g., operators, path, channels etc.) to achieve efficient NAS algorithms



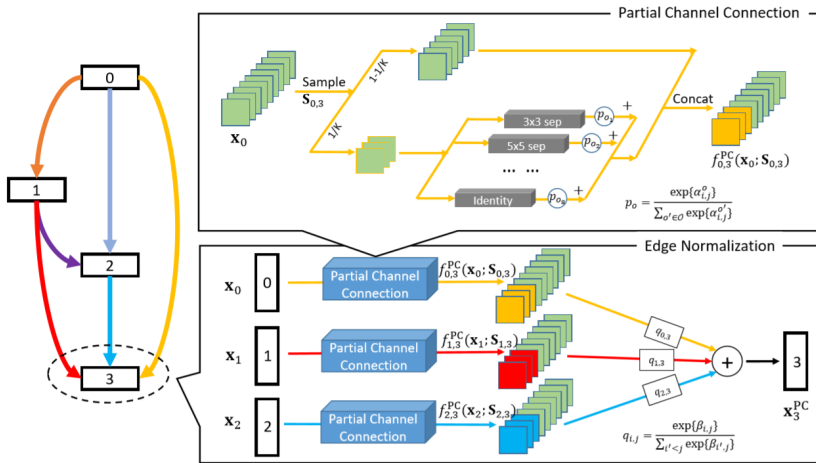
Learning both weight parameters and binarized architecture parameters <sup>10</sup>

<sup>10</sup>Han Cai, Ligeng Zhu, and Song Han (2018). “Proxylessnas: Direct neural architecture search on target task and hardware”. In: *arXiv preprint arXiv:1812.00332*



# Discretize methods

Another example: PC-DARTS



Overview of PC-DARTS. <sup>11</sup>

<sup>11</sup>Yuhui Xu et al. (2019). "Pc-darts: Partial channel connections for memory-efficient differentiable architecture search". In: [arXiv preprint arXiv:1907.05737](https://arxiv.org/abs/1907.05737)





## Partial channel connection

$$f_{i,j}^{PC}(x_i; S_{i,j}) = \sum_{o \in \mathcal{O}} \frac{\exp \alpha_{i,j}^o}{\sum_{o' \in \mathcal{O}} \exp \alpha_{i,j}^{o'}} \cdot (S_{i,j} * x_i) + (1 - S_{i,j} * x_i)$$

where  $S_{i,j}$  defines a channel sampling mask, which assigns 1 to selected channels and 0 to masked ones.

## Edge normalization

$$x_j^{PC} = \sum_{i < j} \frac{\exp \beta_{i,j}}{\sum_{i' < j} \exp \beta_{i',j}} \cdot f_{i,j}(x_i)$$

Edge normalization can mitigate the undesired fluctuation introduced by partial channel connection



Search Space Design

Blackbox Optimization

NAS as a hyperparameter optimization

Reinforcement Learning

Evolution methods

Regularized methods

Baysian Optimization

Differentiable search

Efficient methods

**NAS Benchmark**

Estimation strategy



## The motivation

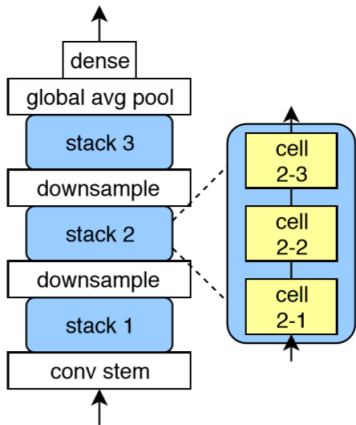
NAS algorithms are hard to reproduce normally

- ▶ Some NAS algorithms require months of compute time, making these methods inaccessible to most researchers
- ▶ Different proposed NAS algorithms are hard to compare since their different training procedures and different search spaces

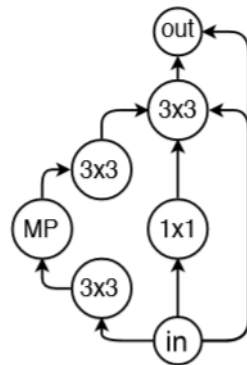
## Related works

- ▶ [Chris Ying et al. \(2019\)](#). “Nas-bench-101: Towards reproducible neural architecture search”. In: *International Conference on Machine Learning*, pp. 7105–7114
- ▶ [Xuanyi Dong and Yi Yang \(2020\)](#). “Nas-bench-102: Extending the scope of reproducible neural architecture search”. In: *arXiv preprint arXiv:2001.00326*

# NAS-Bench-101



The stem of the search space



Operation on node

The stem is composed of three cells, followed by a downsampling layer. The downsampling layer halves the height and width of the feature map via max-pooling and the channel count is doubled. The pattern are repeated three times, followed by global average pooling and a final dense softmax layer. The initial layer is a stem consisting of one  $3 \times 3$  convolution with 128 output channels.



The space of cell architectures is a directed acyclic graph on  $V$  nodes and  $E$  edges, each node has one of  $L$  labels, representing the corresponding operation. The constraints on the search space

## The search space

- ▶  $L = 3$ 
  - ▶  $3 \times 3$  convolution
  - ▶  $1 \times 1$  convolution
  - ▶  $3 \times 3$  max-pool
- ▶  $V \leq 7$
- ▶  $E \leq 9$
- ▶ input node and output node are pre-defined on two of  $V$  nodes

Encoding is implemented as a  $7 \times 7$  upper-triangular binary matrix, by de-duplication and verification, there are **423,000** neural network architectures

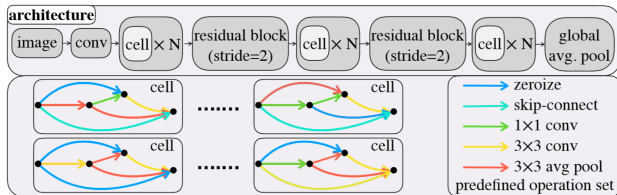


The dataset of NAS-Bench-101 is a mapping from the  $(A, Epoch, trial\#)$  to

- ▶ Training accuracy
- ▶ Validation accuracy
- ▶ Testing accuracy
- ▶ Training time in seconds
- ▶ Number of trainable parameters

## Applications

- ▶ Compare different NAS algorithms
- ▶ Research on generalization abilities of NAS algorithms



**Top:** the macro skeleton of each architecture candidate. **Bottom-left:** examples of neural cell with 4 nodes. Each cell is a directed acyclic graph, where each edge is associated with an operation selected from a predefined operation as shown in **Bottom-right**

## Comparison between NAS-Bench-101 and NAS-Bench-201

NAS-Bench-101 uses Operation on node while NAS-Bench-201 uses Operation on edge as its search space

	#architectures	#datasets	$\ \mathcal{O}\ $	Search space constraint	Supported NAS algorithms	Diagnostic information
NAS-Bench-101	510M	1	3	constrain #edges	partial	-
Nas-Bench-201	15.6K	3	5	no constraint	all	fine-grained info. (e.g., #params, FLOPs, latency)



Search Space Design

Blackbox Optimization

- NAS as a hyperparameter optimization

- Reinforcement Learning

- Evolution methods

- Regularized methods

- Baysian Optimization

- Differentiable search

- Efficient methods

NAS Benchmark

Estimation strategy





## Strategy

- ▶ Task specific
  - ▶ Classification tasks  
e.g., accuracy, error rate, etc.
  - ▶ Segmentation tasks  
e.g., pixel accuracy, MIoU
  - ▶ Generation tasks  
e.g., Inception Score, Frechet Inception Score, etc.
- ▶ Latency considered factors
  - ▶ #FLOPs
  - ▶ #Parameters

## Tips

Different NAS methods can incorporate diverse factors into search consideration