Neural Architecture Search

Yujia Zhang

April 13, 2022

Questions

1. Which of the following approaches consumes more resources to find a neural architecture?
   a. reinforcement learning (NAS-RL)
   b. differentiable neural architecture search (DARTS)
2. To mitigate the high cost of searching in the global search space, one approach people use is to search over different "modules" (or "cells") and stack them together to form an architecture.
   a. True
   b. False

Outline

- Framework of NAS
  - Search space
  - Selecting and evaluating candidate networks
- Numerical experiments
- Opportunities and challenges

NAS helps discover good architectures

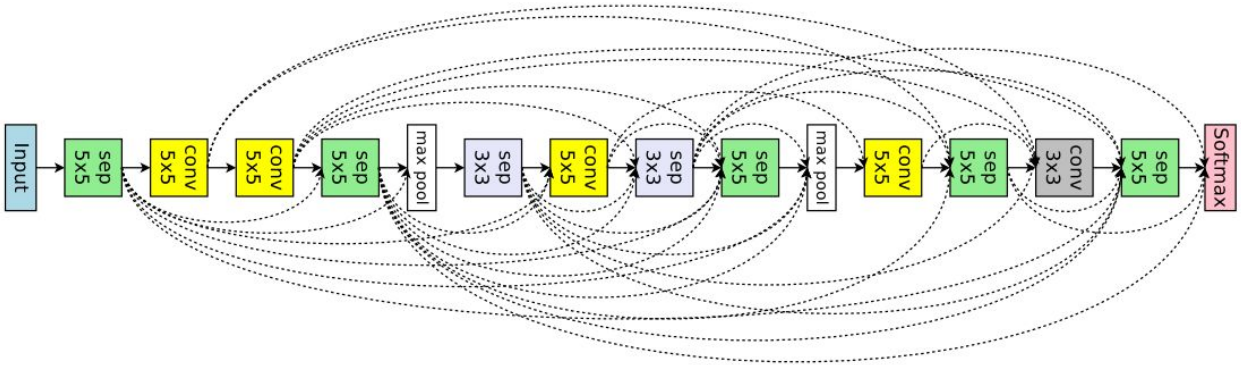


Figure 7. ENAS's discovered network from the macro search space for image classification.

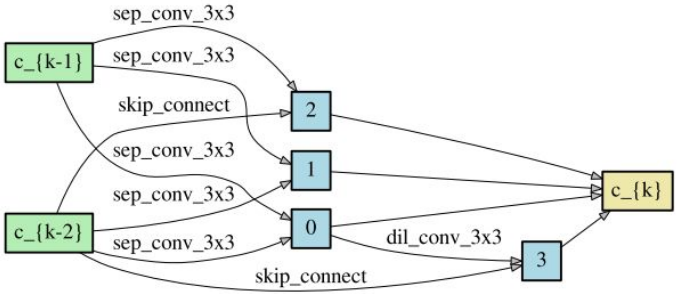


Figure 4: Normal cell learned on CIFAR-10.

Image sources: Pham et al. 2018, Liu et al. 2018

NAS pipeline

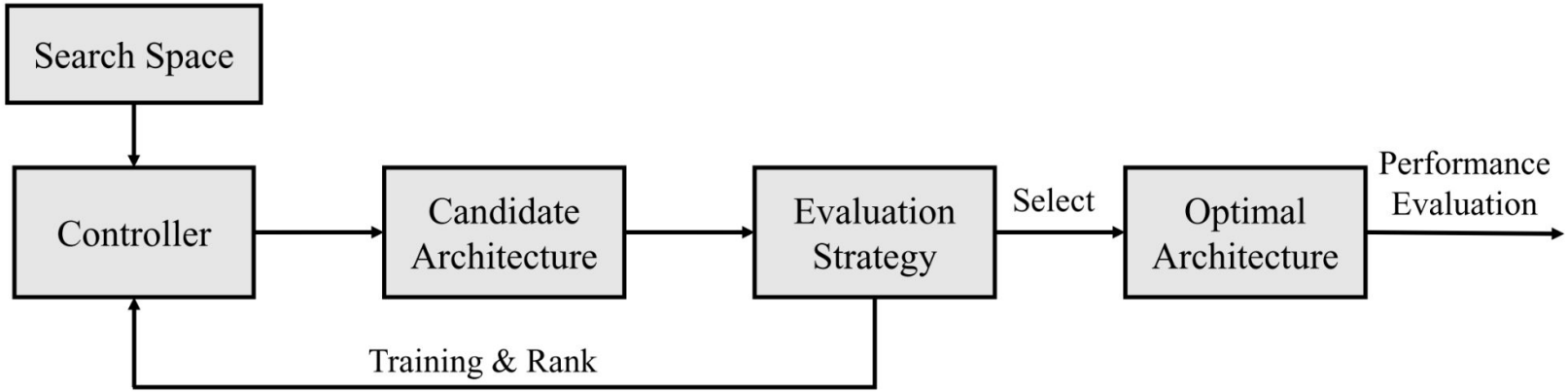


Fig. 1. The general framework of NAS.

Image source: Ren et al. 2020

NAS pipeline

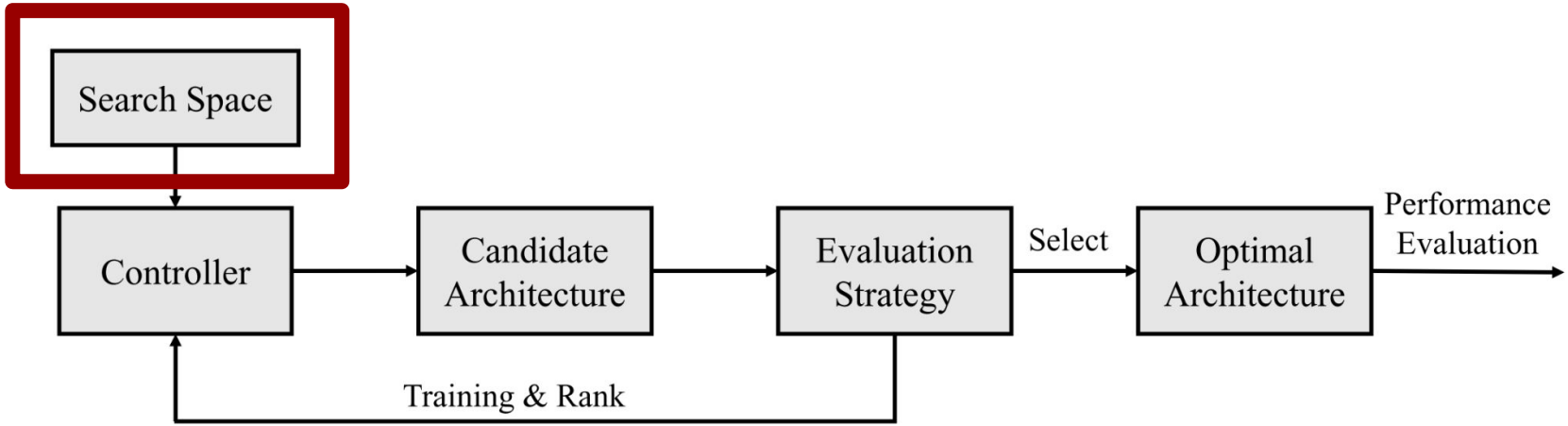


Fig. 1. The general framework of NAS.

Image source: Ren et al. 2020

Representation: sequence of tokens

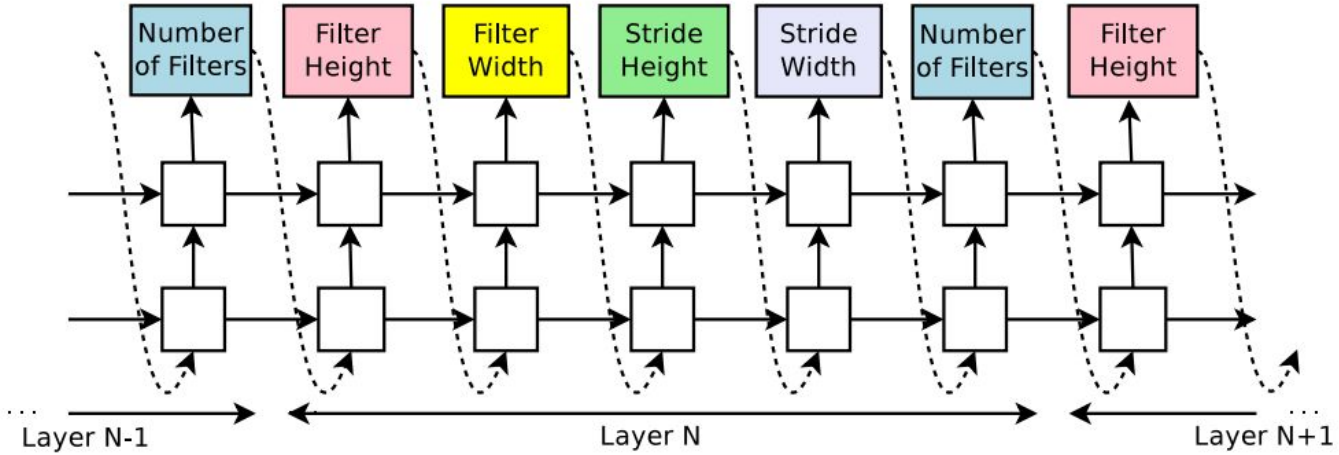


Image source: Zoph and Le, 2017

Representation: stack of cells

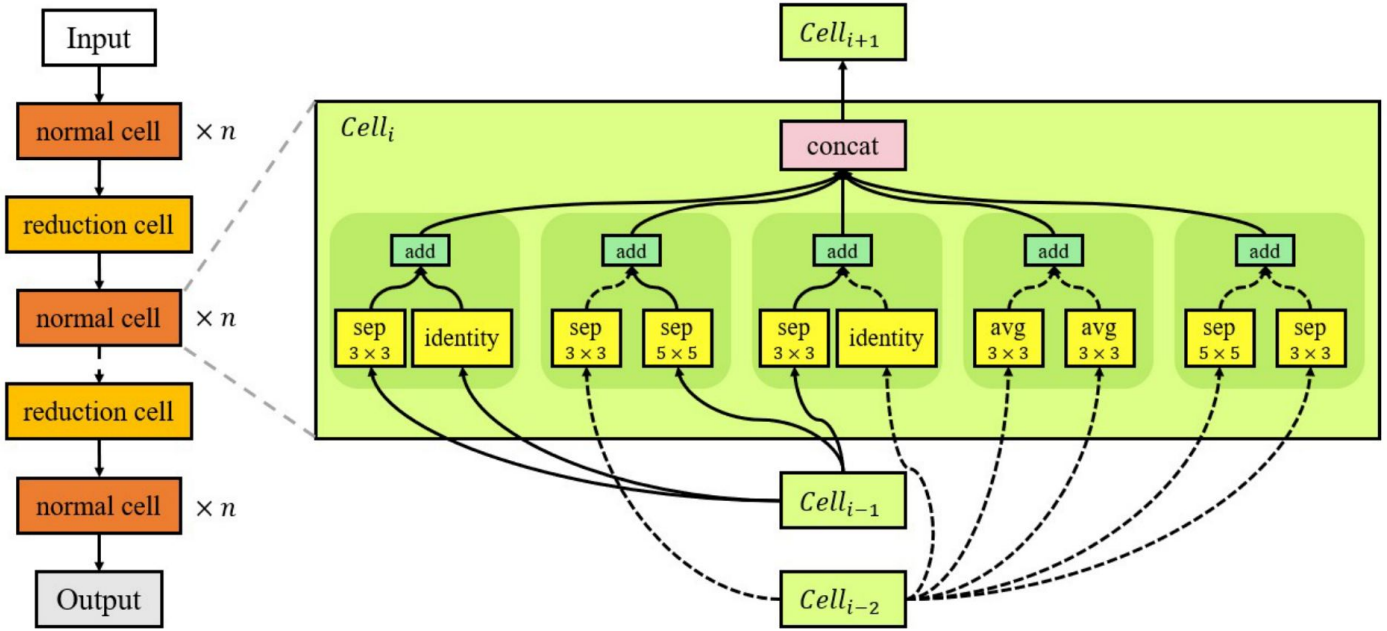


Image source: Ren et al. 2020

NAS pipeline

- **ENAS**: RL + weight-sharing [Pham et al. 2018]
- **DARTS**: gradient-based optimization [Liu et al. 2018]

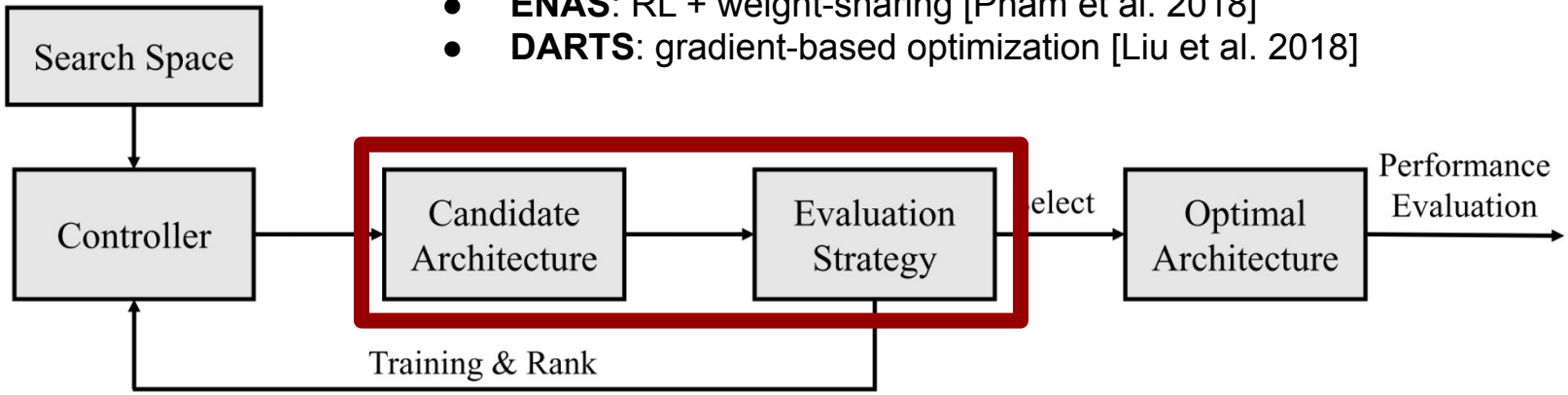


Fig. 1. The general framework of NAS.

Image source: Ren et al. 2020

NAS pipeline

Other methods for generating candidates:

- **random sampling** from pre-defined search space
- **mutation** from existing networks

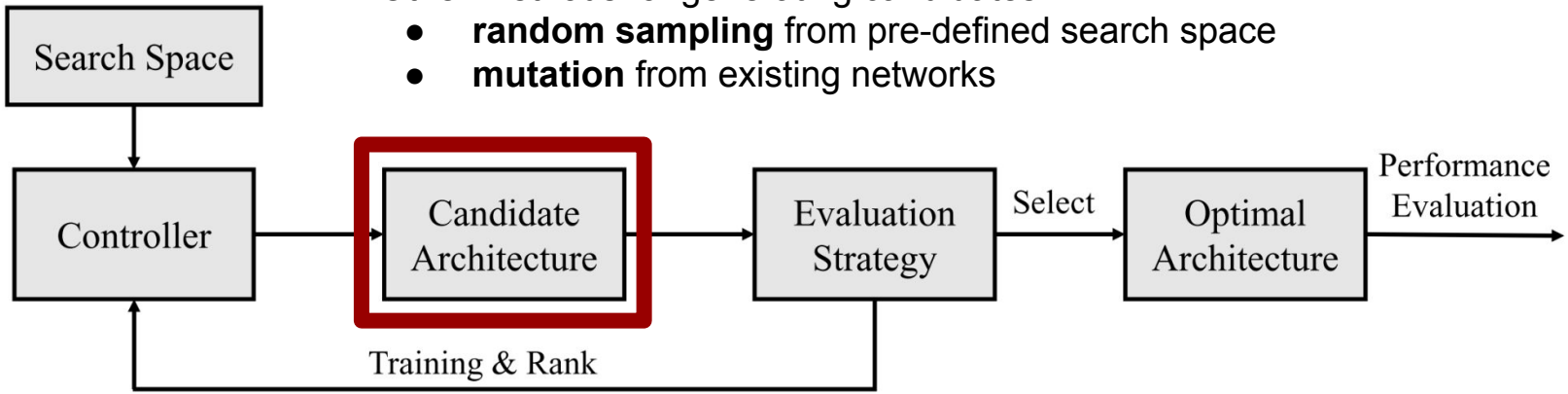


Fig. 1. The general framework of NAS.

Image source: Ren et al. 2020

NAS pipeline

Other methods for evaluating candidates, combined with early-stopping:

- **performance prediction**: [Domhan et al. 2015, Klein et al. 2016, Baker et al. 2017]

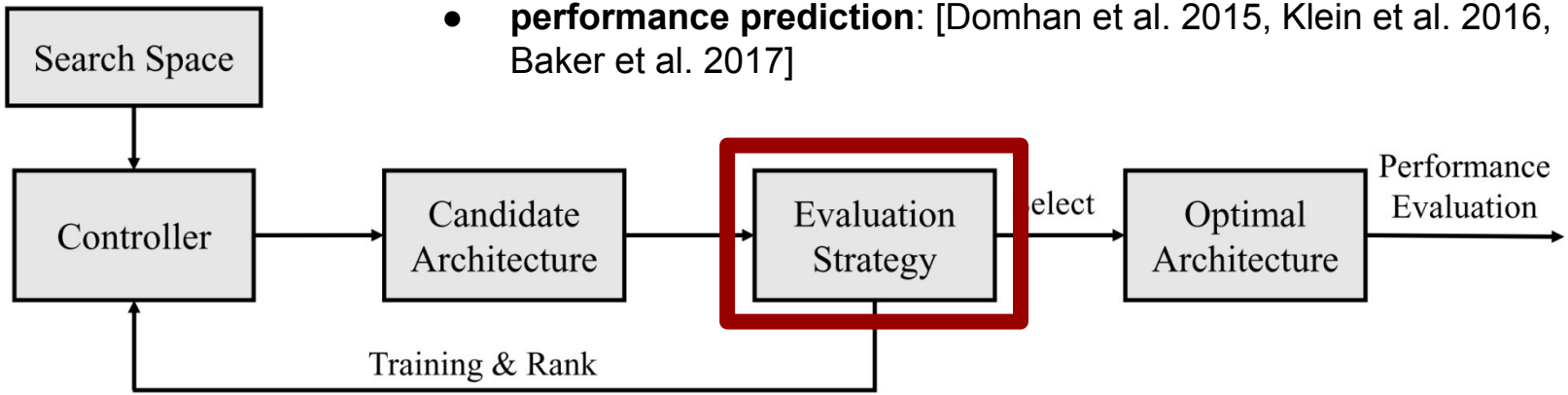


Fig. 1. The general framework of NAS.

Image source: Ren et al. 2020

ENAS: Efficient Neural Architecture Search

- Improves upon a previous paper [Zoph and Le 2017], which
  - uses RL for NAS but is prohibitively expensive (thousands of GPU days)
  - searches and evaluates each candidate architecture from scratch
- ENAS: enforces weight-sharing
  - the search space for a cell can be represented with a single directed acyclic graph
  - individual cell architectures are subgraphs of it
  - all share the same set of weights

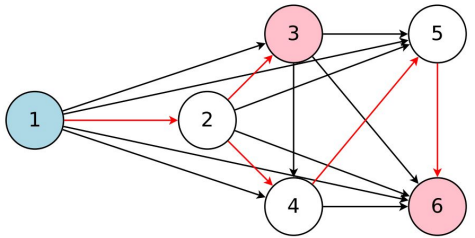
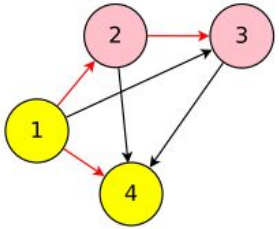


Figure 2. The graph represents the entire search space while the red arrows define a model in the search space, which is decided by a controller. Here, node 1 is the input to the model whereas nodes 3 and 6 are the model's outputs.

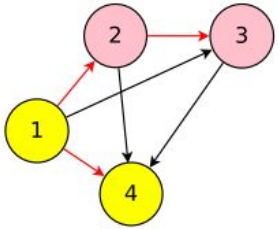
Image source: Pham et al. 2018

ENAS: generating candidates



1. specify overall cell structure

ENAS: generating candidates



1. specify overall cell structure

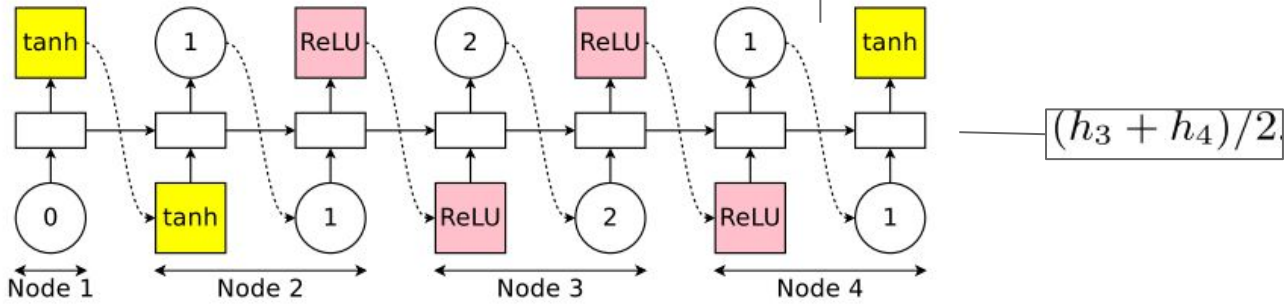
parameter matrix $\mathbf{W}_{\ell,j}^{(\mathbf{h})}$

$h_1 = \tanh\left(\mathbf{x}_t \cdot \mathbf{W}^{(\mathbf{x})} + \mathbf{h}_{t-1} \cdot \mathbf{W}_1^{(\mathbf{h})}\right)$

$h_2 = \mathrm{ReLU}(h_1 \cdot \mathbf{W}_{2,1}^{(\mathbf{h})})$

$h_3 = \mathrm{ReLU}(h_2 \cdot \mathbf{W}_{3,2}^{(\mathbf{h})})$

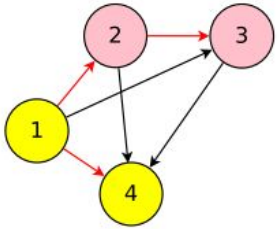
$h_4 = \tanh\left(h_1 \cdot \mathbf{W}_{4,1}^{(\mathbf{h})}\right)$



$(h_3 + h_4)/2$

2. controller makes decisions (activation + previous node)

ENAS: generating candidates



1. specify overall cell structure

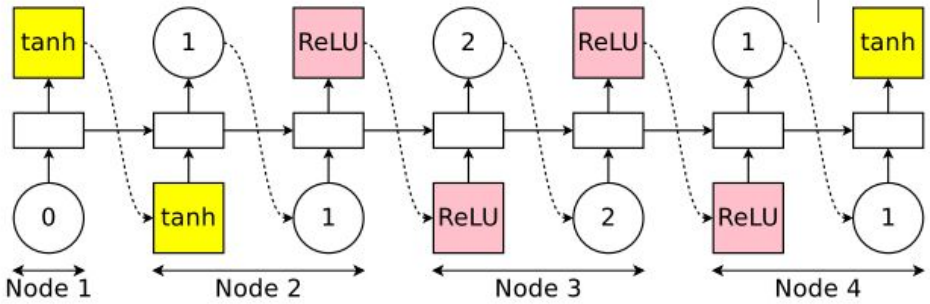
$h_1 = \tanh\left(\mathbf{x}_t \cdot \mathbf{W}^{(\mathbf{x})} + \mathbf{h}_{t-1} \cdot \mathbf{W}_1^{(\mathbf{h})}\right)$

$h_2 = \mathrm{ReLU}(h_1 \cdot \mathbf{W}_{2,1}^{(\mathbf{h})})$

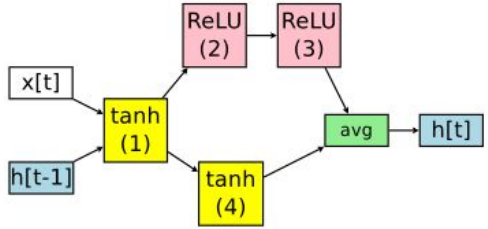
$h_3 = \mathrm{ReLU}(h_2 \cdot \mathbf{W}_{3,2}^{(\mathbf{h})})$

$h_4 = \tanh\left(h_1 \cdot \mathbf{W}_{4,1}^{(\mathbf{h})}\right)$

parameter matrix $\mathbf{W}_{\ell,j}^{(\mathbf{h})}$



2. controller makes decisions (activation + previous node)



3. generated architecture

$(h_3 + h_4)/2$

ENAS: training and selection

- **Training:** alternate between training the shared parameters and the controller's parameters
- **Selection:**
  - Sample a few candidate architectures from the controller
  - For each architecture, compute the loss on one minibatch from the validation set (using weights from the trained shared parameters)
- **Evaluation:** Take the model with the best performance and re-train it from scratch

ENAS: performance on CIFAR-10

SOTA by human experts

global search space

cell-based search space

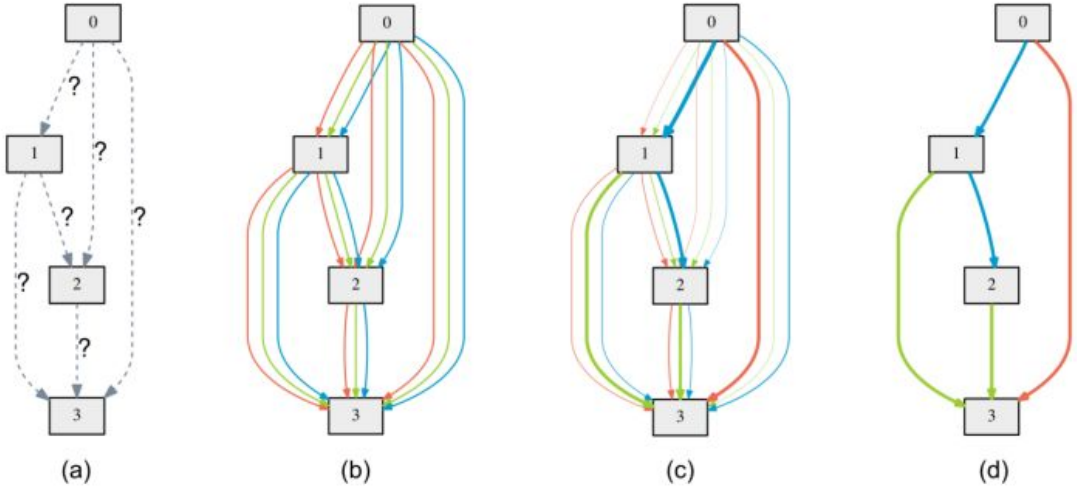
Method	GPUs	Times (days)	Params (million)	Error (%)
DenseNet-BC (Huang et al., 2016)	–	–	25.6	3.46
DenseNet + Shake-Shake (Gastaldi, 2016)	–	–	26.2	2.86
DenseNet + CutOut (DeVries & Taylor, 2017)	–	–	26.2	**2.56**
Budgeted Super Nets (Veniat & Denoyer, 2017)	–	–	–	9.21
ConvFabrics (Saxena & Verbeek, 2016)	–	–	21.2	7.43
Macro NAS + Q-Learning (Baker et al., 2017a)	10	8-10	11.2	6.92
Net Transformation (Cai et al., 2018)	5	2	19.7	5.70
FractalNet (Larsson et al., 2017)	–	–	38.6	4.60
SMASH (Brock et al., 2018)	1	1.5	16.0	4.03
NAS (Zoph & Le, 2017)	800	21-28	7.1	4.47
NAS + more filters (Zoph & Le, 2017)	800	21-28	37.4	**3.65**
ENAS + macro search space	1	0.32	21.3	4.23
ENAS + macro search space + more channels	1	0.32	38.0	**3.87**
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	**2.65**
ENAS + micro search space	1	0.45	4.6	3.54
ENAS + micro search space + CutOut	1	0.45	4.6	**2.89**

DARTS: Differentiable Architecture Search

Motivation:

- Existing methods treat architecture search as a black-box optimization problem over a **discrete** domain → bad scalability
- Single choice for type of operation → **Mixture** of operations
- Relaxing the categorical choices to a continuous domain enables doing **gradient-based** optimization.

DARTS: generating candidates



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

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

Image source: Liu et al. 2018

DARTS: training

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

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 α 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 α.

DARTS: selection

- Pick a few random seeds (4 in their case).
- **Training:** Run DARTS on each seed.
- **Selection:** Train the DARTS-selected architecture from each random seed for a short period, then select the architecture with the top validation performance.
- **Evaluation:** Train the selected architecture from scratch.

DARTS: performance on CIFAR-10

Table 1: Comparison with state-of-the-art image classifiers on CIFAR-10 (lower error rate is better). Note the search cost for DARTS does not include the selection cost (1 GPU day) or the final evaluation cost by training the selected architecture from scratch (1.5 GPU days).

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	#ops	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	–	–	manual
NASNet-A + cutout (Zoph et al., 2018)	2.65	3.3	2000	13	RL
NASNet-A + cutout (Zoph et al., 2018)†	2.83	3.1	2000	13	RL
BlockQNN (Zhong et al., 2018)	3.54	39.8	96	8	RL
AmoebaNet-A (Real et al., 2018)	3.34 ± 0.06	3.2	3150	19	evolution
AmoebaNet-A + cutout (Real et al., 2018)†	3.12	3.1	3150	19	evolution
AmoebaNet-B + cutout (Real et al., 2018)	2.55 ± 0.05	2.8	3150	19	evolution
Hierarchical evolution (Liu et al., 2018b)	3.75 ± 0.12	15.7	300	6	evolution
PNAS (Liu et al., 2018a)	3.41 ± 0.09	3.2	225	8	SMBO
ENAS + cutout (Pham et al., 2018b)	2.89	4.6	0.5	6	RL
ENAS + cutout (Pham et al., 2018b)*	2.91	4.2	4	6	RL
Random search baseline‡ + cutout	3.29 ± 0.15	3.2	4	7	random
DARTS (first order) + cutout	3.00 ± 0.14	3.3	1.5	7	gradient-based
DARTS (second order) + cutout	2.76 ± 0.09	3.3	4	7	gradient-based

DARTS: performance on Penn Treebank

Table 2: Comparison with state-of-the-art language models on PTB (lower perplexity is better). Note the search cost for DARTS does not include the selection cost (1 GPU day) or the final evaluation cost by training the selected architecture from scratch (3 GPU days).

Architecture	**Perplexity**		**Params (M)**	**Search Cost (GPU days)**	**#ops**	**Search Method**
	valid	test				
Variational RHN (Zilly et al., 2016)	67.9	65.4	23	–	–	manual
LSTM (Merity et al., 2018)	60.7	58.8	24	–	–	manual
LSTM + skip connections (Melis et al., 2018)	60.9	58.3	24	–	–	manual
LSTM + 15 softmax experts (Yang et al., 2018)	58.1	56.0	22	–	–	manual
NAS (Zoph & Le, 2017)	–	64.0	25	1e4 CPU days	4	RL
ENAS (Pham et al., 2018b)*	68.3	63.1	24	0.5	4	RL
ENAS (Pham et al., 2018b)†	60.8	58.6	24	0.5	4	RL
Random search baseline‡	61.8	59.4	23	2	4	random
DARTS (first order)	60.2	57.6	23	0.5	4	gradient-based
DARTS (second order)	58.1	55.7	23	1	4	gradient-based

Opportunities and challenges

- **Tradeoff in modularizing the search space**
  - modularizing the search space improves search efficiency but limits the scope of choice
  - Sampling random graph expands the search space and achieves comparable performance (RandWire) [Xie et al. 2019]
- **Need for a unified benchmark**
  - How much of the performance is due to the architecture itself, and how much is due to other techniques (augmentation, regularization, search space design)?
  - Comparison to human-designed networks and random search
- **Pitfalls in parameter sharing**
  - ranking with weight-sharing ⇏ ranking in re-training [Zhang et al. 2020]
- **Wider applications**
  - beyond image classification and natural languages
- **Combination of NAS and hyperparameter tuning**
  - Some recent works that jointly optimize them both: [Dong et al. 2021, Dai et al. 2021]

References

[Survey] Ren, Pengzhen, et al. "A comprehensive survey of neural architecture search: Challenges and solutions." *arXiv preprint arXiv:2006.02903* (2020).

[Survey] Elsken, T., J. H. Metzen, and F. Hutter. "Neural architecture search: A survey. arXiv 2018." *arXiv preprint arXiv:1808.05377* (2018).

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

[ENAS] Pham, Hieu, et al. "Efficient neural architecture search via parameters sharing." *International conference on machine learning*. PMLR, 2018.

[DARTS] Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search." *arXiv preprint arXiv:1806.09055* (2018).

Zhang, Miao, et al. "Overcoming multi-model forgetting in one-shot NAS with diversity maximization." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[RandWire] Xie, Saining, et al. "Exploring randomly wired neural networks for image recognition." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Dong, Xuanyi, et al. "AutoHAS: Efficient hyperparameter and architecture search." *arXiv preprint arXiv:2006.03656* (2020).

Dai, Xiaoliang, et al. "Fbnetv3: Joint architecture-recipe search using predictor pretraining." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.