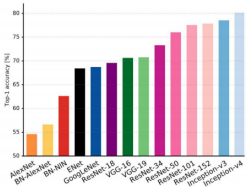
Deep Learning for Computer Vision

Neural Architecture Search

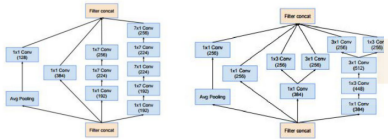
Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Neural Architecture Search: Why?



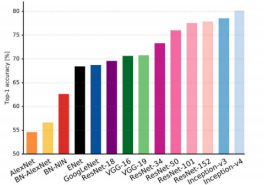
Canziani et al (2017)



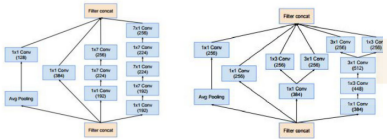
Complex hand-engineered layers from Inception-V4 (Szegedy et al., 2017)

- Most popular and successful model architectures designed by human experts
- Requires hundreds of hours of arbitrary training and testing and hyperparameter tuning
- However, it doesn't mean we explored the entire network architecture space or that we found an optimal solution
- Can we adopt a systematic and automatic way of learning high-performance model architectures?

Neural Architecture Search: Why?



Canziani et al (2017)



Complex hand-engineered layers from Inception-V4 (Szegedy et al., 2017)

Credit: Nikhil Naik, Neural Architecture Search: A Tutorial, 2019

- Most popular and successful model architectures designed by human experts
- Requires hundreds of hours of arbitrary training and testing and hyperparameter tuning
- However, it doesn't mean we explored the entire network architecture space or that we found an optimal solution
- Can we adopt a systematic and automatic way of learning high-performance model architectures? **Solution:** Neural Architecture Search

Neural Architecture Search (NAS): Brief History

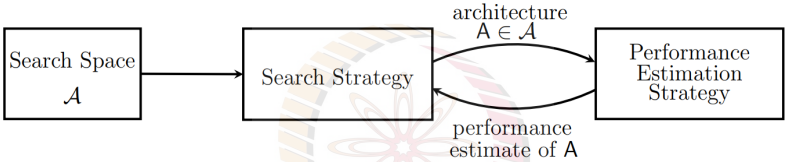
Early Work

- Neuroevolution: Evolutionary algorithms (e.g., Miller et al., 89; Schaffer et al., 92; Stanley & Miikkulainen, 02; Verbancsics & Harguess, 13)
- Random search (e.g., Pinto et al., 09; Bergstra & Bengio, 12)
- Bayesian optimization for architecture and hyperparameter tuning (e.g., Snoek et al., 12; Bergstra et al., 13; Domhan et al., 15)

Renewed Interest (2017-)

- Zoph and Le, Neural Network Architecture Search with Reinforcement Learning, ICLR'17.
- Baker et al., Designing Neural Network Architectures using Reinforcement Learning, ICLR'17.

NAS: General Problem Setup



- **Search Space:** Defines which architectures can be represented in principle
- **Search Strategy:** Details how to explore search space (which is often exponentially large or even unbounded)
- **Performance Estimation Strategy:** Estimating an architecture's performance - standard training and validation of architecture on data may be computationally expensive and limits number of architectures that can be explored

Credit: Elksen et al., Neural Architecture Search: A Survey, 2018

Neural Architecture Search Space

- Neural networks represent a function that transforms input variables $\mathbf{x}$ to output variables $\hat{\mathbf{y}}$ through a series of operations

NPTEL

Neural Architecture Search Space

- Neural networks represent a function that transforms input variables $\mathbf{x}$ to output variables $\hat{\mathbf{y}}$ through a series of operations
- Recall computational graphs (W8P1); each node $z^{(k)} \in Z$ represents a tensor and is associated with an operation $o^{(k)} \in O$ on its parent nodes I^k

NPTEL

Neural Architecture Search Space

- Neural networks represent a function that transforms input variables $\mathbf{x}$ to output variables $\hat{\mathbf{y}}$ through a series of operations
- Recall computational graphs (W8P1); each node $z^{(k)} \in Z$ represents a tensor and is associated with an operation $o^{(k)} \in O$ on its parent nodes I^k
- Computation at a node k:

$$z^{(k)} = o^{(k)}(I^{(k)})$$

where operations includes unary operations such as convolutions, pooling, activation functions or n-ary operations such as concatenation or addition

Neural Architecture Search Space

- Neural networks represent a function that transforms input variables $\mathbf{x}$ to output variables $\hat{\mathbf{y}}$ through a series of operations
- Recall computational graphs (W8P1); each node $z^{(k)} \in Z$ represents a tensor and is associated with an operation $o^{(k)} \in O$ on its parent nodes I^k
- Computation at a node k:

$$z^{(k)} = o^{(k)}(I^{(k)})$$

  where operations includes unary operations such as convolutions, pooling, activation functions or n-ary operations such as concatenation or addition
- **NAS space:** Subspace of this general definition of neural architectures

NAS Space: Global vs Cell-based[1]

Global Search Space

- Large degrees of freedom regarding arrangement of operations
- Allowed operations examples
  - convolutions, pooling, dense layers (FCs) with activation, global average pooling
- Constraints examples
  - pooling as first operation; dense layers (FCs) before convolution operations
- Rigid, impractical to scale and transfer

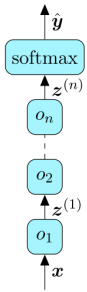
Cell-based Search Space

- Many effective handcrafted architectures designed with repetitions of fixed structures
- Network constructed by repeating a **cell** structure, a small directed acyclic graph representing a feature transformation
- E.g. **NASNet search space:** Learns two types of cells:
  - **Normal Cell:** Input and output feature maps have same dimension
  - **Reduction Cell:** Output feature map has width and height reduced by half

[1] Zoph et al, Learning Transferable Architectures for Scalable Image Recognition, CVPR 2018

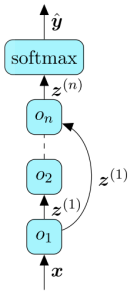
Global Search Space

Simplest example: a *chain-structured search space* as shown below



(a) Baker et al. (2017)

$$z^{(k)} = o^{(k)}\left(\left\{z^{(k-1)}\right\}\right)$$



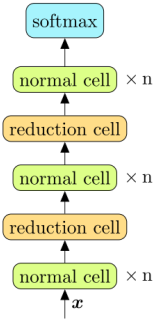
(b) Zoph and Le (2017)

$$z^{(k)} = o^{(k)}\left(\left\{z^{(k-1)}\right\} \cup \left\{z^{(i)} \mid i < k-1\right\}\right)$$

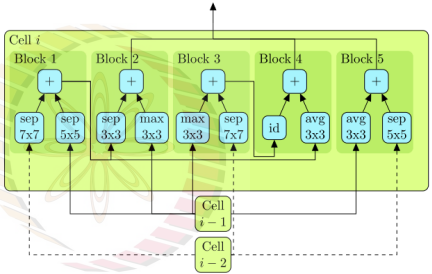
(Skip Connections)

Credit: Wistuba et al, A Survey on Neural Architecture Search, 2019

Cell-Based Search Space



Architecture template



Reduction cell of the NASNet-A architecture (Zoph et al., 2018)

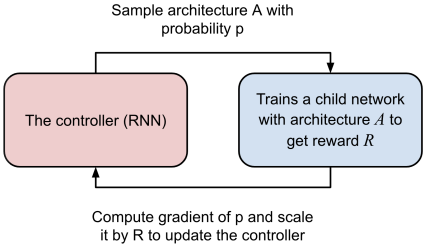
NASNet: While cell topology is maintained across network, its hyperparameters are often varied; merging operation is concatenation

Image Source: Wistuba et al., A Survey on Neural Architecture Search, 2019

NAS with Reinforcement Learning (RL)[2]

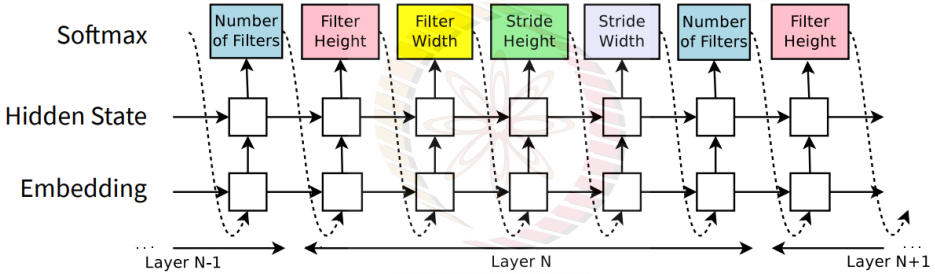
- A controller, which proposes child architecture, is implemented as a RNN; outputs a sequence of tokens that configure network architecture
- Controller trained as a RL task using REINFORCE (Monte-Carlo policy gradient)
  - **Action space:** List of tokens T $(a_{1:T})$ for defining a child network predicted by controller
  - **Reward**: Accuracy of a child network, R.
  - **Loss:** NAS optimizes controller parameters θ with a REINFORCE loss

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



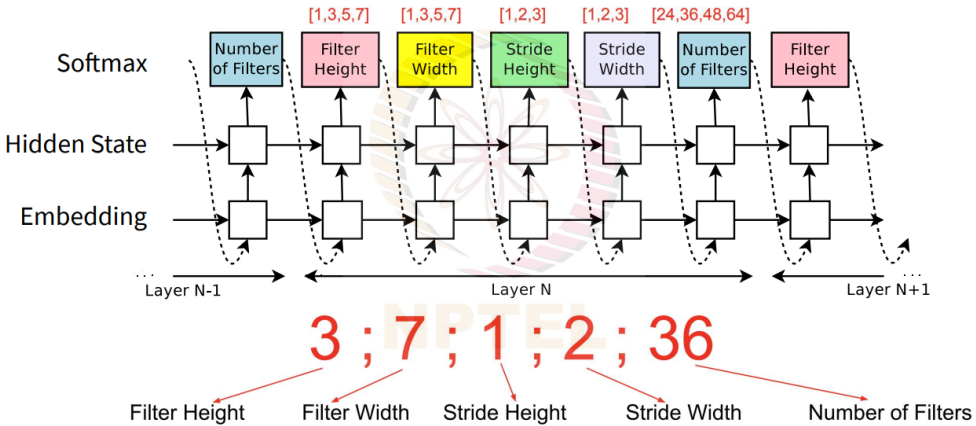
[2] Zoph and Le, Neural Network Architecture Search with Reinforcement Learning, ICLR 2017.

Training with REINFORCE



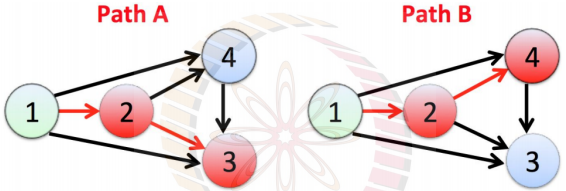
Credit: Nikhil Naik, Neural Architecture Search: A Tutorial, 2019

Training with REINFORCE



Credit: Nikhil Naik, Neural Architecture Search: A Tutorial, 2019

How To Make NAS More Efficient?



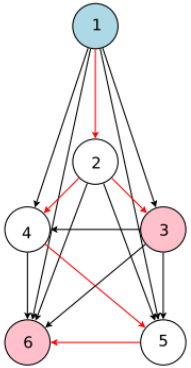
- Models defined by **path A** and **path B** are trained independently
- Instead, can we treat all model trajectories as sub-graphs of a single directed acyclic graph?
- **Efficient NAS (ENAS)**[3]: aggressively shares parameters among child models

Credit: Nikhil Naik, Neural Architecture Search: A Tutorial, 2019

[3] Pham et al., Efficient Neural Architecture Search via Parameters Sharing, ICML 2018

Efficient NAS with Weight Sharing

- All sampled architecture graphs viewed as *subgraphs* of a larger *supergraph*
- Graph represents entire search space while red arrows define a model in the search space, decided by an RNN controller (trained with REINFORCE)
- Weights of controller and a part of over-parameterized network are alternately updated
- ENAS achieved 2.89% test error on CIFAR-10, took less than 16 hours to search (significantly lesser than other NAS models)



Credit: Pham et al., Efficient Neural Architecture Search via Parameters Sharing. ICML 2018.

Differentiable Architecture Search: Gradient-based NAS[4]

- Introduced binary variables in $\{\alpha_{i,j,k}\}$ to make search space continuous. This simplifies definition to:

$$z^{(k)} = \sum_{i \in I^{(k)}} \sum_{j \in |O|} \alpha_{i,j,k} \cdot o^{(j)}(z^{(i)}) \quad \text{with} \quad \alpha_{i,j,k} \in \{0, 1\}$$

- So far, the assumption: every operation is either part of the network or not
- This method relaxes categorical choice of a particular operation as a softmax over all operations

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

- This reduces search to learning a set of mixing probabilities α

[4] Liu et al, DARTS: Differentiable Architecture Search, ICLR 2019

Differentiable Architecture Search: Gradient-based NAS[5]

- Uses an alternating (bilevel) optimization method:
  - learn model parameters w by minimizing loss on training set
  - learn structural parameters α by minimizing loss on validation set

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

- Final architecture chosen based on:

$$o^{(i,j)} = \arg\max_{o \in O} \alpha_o^{(i,j)}$$

- Elegant solution that makes all parameters differentiable!

[5] Liu et al, DARTS: Differentiable Architecture Search, ICLR 2019

Architecture Transferability of NAS Networks

Model	Params	×+	1/5-Acc (%)
Inception V3	23.8M	5.72B	78.8 / 94.4
Xception	22.8M	8.37B	79.0 / 94.5
Inception ResNet V2	55.8M	13.2B	80.4 / 95.3
ResNeXt-101 (64x4d)	83.6M	31.5B	80.9 / 95.6
PolyNet	92.0M	34.7B	81.3 / 95.8
Dual-Path-Net-131	79.5M	32.0B	81.5 / 95.8
Squeeze-Excite-Net	145.8M	42.3B	82.7 / 96.2
GeNet-2*	156M	–	72.1 / 90.4
Block-QNN-B (N=3)*	–	–	75.7 / 92.6
Hierarchical (2, 64)*	64M	–	79.7 / 94.8
PNASNet-5 (4, 216)	86.1M	25.0B	82.9 / 96.1
NASNet-A (6, 168)	88.9M	23.8B	82.7 / 96.2
AmoebaNet-B (6, 190)*	84.0M	22.3B	82.3 / 96.1
AmoebaNet-A (6, 190)*	86.7M	23.1B	82.8 / 96.1
AmoebaNet-A (6, 204)*	99.6M	26.2B	82.8 / 96.2
AmoebaNet-C (6, 228)*	155.3M	41.1B	**83.1 / 96.3**

CIFAR-10 to ImageNet

Dataset	Acc.	Network	Acc.	Best network
Food-101	**90.0**	Deep layer aggregation [40]	**90.1**	NASNet-A Large, fine-tuned ✓
CIFAR-10	97.9	AmoebaNet [41]	**98.4**[a]	NASNet-A Large, fine-tuned ✓
CIFAR-100	87.8	ShakeDrop [42]	**88.2**[a]	NASNet-A Large, fine-tuned ✓
Birdsnap	**80.2**[b]	Mask-CNN [43]	78.5	NASNet-A Large, fine-tuned ✓
SUN397	63.2	Places-pretrained VGG [44]	**66.5**	NASNet-A Large, fine-tuned ✓
Stanford Cars	**94.1**	Deep layer aggregation [40]	93.0	Inception v4, random init
FGVC Aircraft	**92.9**[b]	Deep layer aggregation [40]	89.4	Inception v3, fine-tuned
VOC 2007 Cls.	**89.7**	VGG [9]	88.4	NASNet-A Large, fine-tuned ✓
DTD	75.5	FC+FV-CNN+D-SIFT [45]	**76.7**	Inception-ResNet v2, fine-tuned
Oxford-IIIT Pets	93.8	Object-part attention [46]	**94.3**	NASNet-A Large, fine-tuned ✓
Caltech-101	93.4	Spatial pyramid pooling [47]	**95.0**	NASNet-A Large, fine-tuned ✓
Oxford 102 Flowers	97.1	Object-part attention [46]	**97.7**	NASNet-A Large, fine-tuned ✓

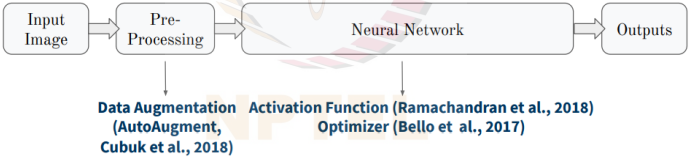
NASNet-A: SOTA on **8/13 commonly used classification benchmarks**

Other Datasets and Tasks

Credit: Nikhil Naik, Neural Architecture Search A Tutorial, 2019

Future Directions[6,7,8]

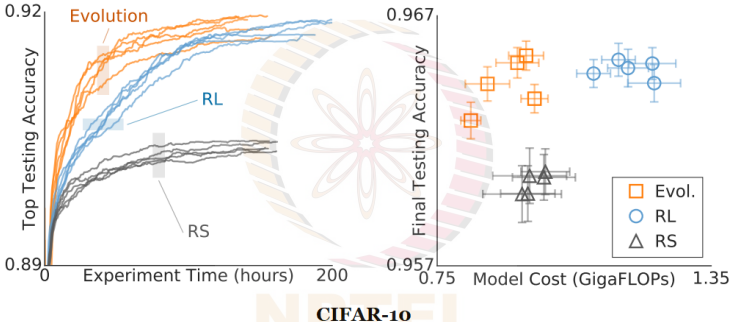
- Search efficiency
- Moving towards less constrained search spaces
- Designing efficient architectures: automated scaling, pruning and quantization
- Joint optimization of all components in deep learning pipeline (data augmentation , architectures, activation functions, training algorithms)



- Designing architectures for multimodal problems, e.g., vision and language

[6] Cubuk et al., AutoAugment: Learning Augmentation Policies from Data, CVPR 2019
[7] Ramachandran et al., Swish: A Self-Gated Activation Function, NEC Journal 2017
[8] Bello et al., Neural Optimizer Search with Reinforcement Learning, ICML 2017

The Curious Case of Random Search[9]



Difference in accuracy between best models found by random search, RL, and Evolution is **less than 1%** on CIFAR-10

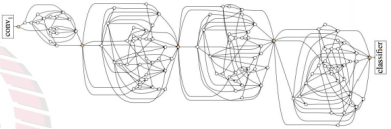
[9] Real et al, Regularized Evolution for Image Classifier Architecture Search, AAAI 2019

The Curious Case of Random Search[10,11]

Architecture	Source	Test Error Best	Test Error Average	Params (M)
NASNet-A#*	[52]	N/A	2.65	3.3
AmoebaNet-B*	[43]	N/A	2.55 ± 0.05	2.8
ProxylessNAS†	[7]	2.08	N/A	5.7
GHN#†	[50]	N/A	2.84 ± 0.07	5.7
SNAS†	[47]	N/A	2.85 ± 0.02	2.8
ENAS†	[41]	2.89	N/A	4.6
ENAS	[34]	2.91	N/A	4.2
Random search baseline	[34]	N/A	3.29 ± 0.15	3.2
DARTS (first order)	[34]	N/A	3.00 ± 0.14	3.3
DARTS (second order)	[34]	N/A	2.76 ± 0.09	3.3
DARTS (second order)‡	Ours	2.62	2.78 ± 0.12	3.3
ASHA baseline	Ours	2.85	3.03 ± 0.13	2.2
Random search WS‡	Ours	2.71	2.85 ± 0.08	4.3

Li and Talwalker (2019)

CIFAR-10



network	test size	epochs	top-1 acc.	top-5 acc.	FLOPs (B)	params (M)
NASNet-A [56]	331^2	>250	82.7	96.2	23.8	88.9
Amoeba-B [34]	331^2	>250	82.3	96.1	22.3	84.0
Amoeba-A [34]	331^2	>250	82.8	96.1	23.1	86.7
PNASNet-5 [26]	331^2	>250	82.9	96.2	25.0	86.1
RandWire-WS	320^2	100	$81.6_{\pm 0.13}$	$95.6_{\pm 0.07}$	$16.0_{\pm 0.36}$	$61.5_{\pm 1.32}$

Xie et al. (2019)

ImageNet

Credit: Nikhil Naik, Neural Architecture Search A Tutorial, 2019

[10] Li and Talwalker, Random Search and Reproducibility for Neural Architecture Search, UAI 2019

[11] Xie et al., Exploring Randomly Wired Neural Networks for Image Recognition, ICCV 2019

Homework

Readings

- Neural Architecture Search article by Lilian Weng
- Survey Papers:
  - Witsuba et al., A Survey on Neural Architecture Search, 2019.
  - Elksen et al., Neural Architecture Search: A Survey, 2018.