Efficient Neural Architecture Search via Parameter Sharing

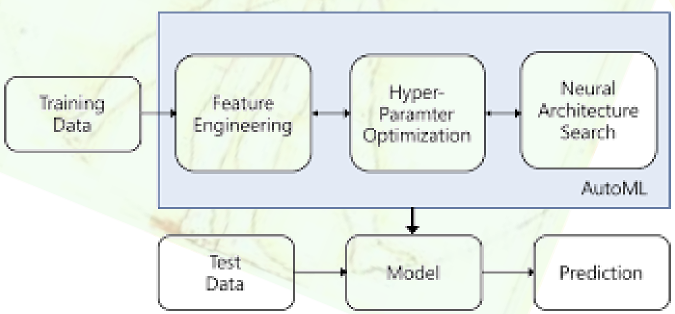
Pattern Recognition & Machine Learning Laboratory

Sangho Kim

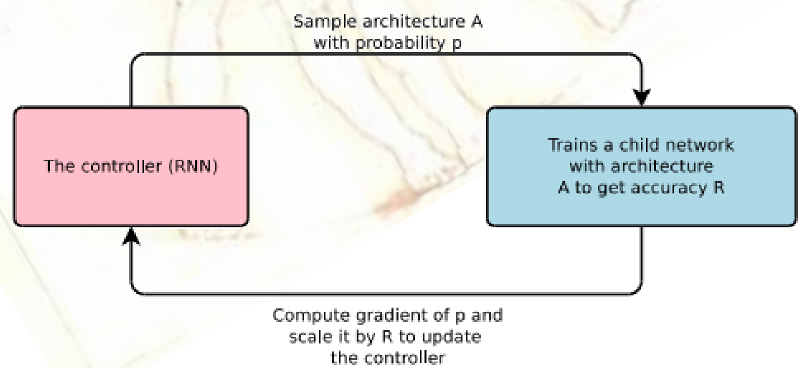
Aug 12, 2021

Introduction

- **Automated Machine Learning (AutoML)**
  - **The process of automating the process of applying machine learning**
    - **Feature engineering**
    - **Hyper-parameter optimization**
    - **Neural Architecture Search (NAS)**
- **Neural Architecture Search**
  - **Neural Architecture Search with Reinforcement Learning** ***[Zoph et al., 2017]***
    - **The controller of the NAS uses the RNN to make sample architecture**
    - **Learning the child network to extract accuracy for the validation set**
    - **Using validation accuracy like a reward for reinforcement learning**
    - **Learning the controller to maximize accuracy**



Representative fields of AutoML

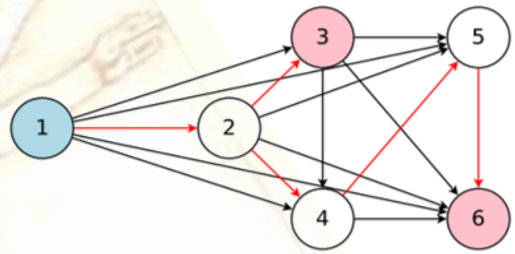


Simple NAS structure

B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," ICLR, 2017.

Efficient Neural Architecture Search via Parameter Sharing [H. Pham, et al., 2018]

- **Goal**
  - Improve the computational complexity of NAS
  - Improve the processing time of NAS
- **Motivation**
  - NAS uses 450 GPUs for 4 days
  - Computational bottleneck of NAS is the training of each child model
  - Throwing away all the trained weights
  - NAS's search space can be represented in directed acyclic graph (DAG)
    - Nodes are local computation
    - Edges are information flow
    - The whole DAG means entire search space
    - The red arrows are sub-graph



Directed acyclic graph

- **Contributions**
  - Efficient Neural Architecture Search (ENAS) forces all child models to share weights
  - Using single GPU, the search for architectures takes less than 16 hours
  - Compared to NAS, time reduction is more than 1000 times

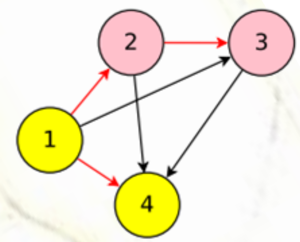
H. Pham, M. Guan, B. Zoph, Q. Le and J. Dean, "Efficient Neural Architecture Search via Parameter Shari ng," CLR, 2018.

Methods (1/4)

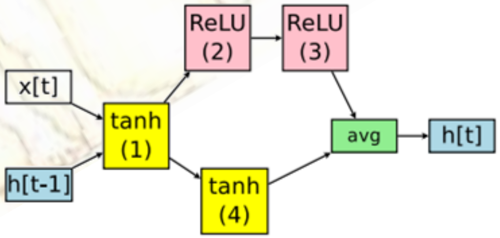
- **Designing recurrent cells**
  - The controller RNN samples two decisions
    - Which edges are activated
    - Which activation functions are performed at each node
  - Mechanism via 4 nodes
    - Node 1 : $\boldsymbol{k_1} = \tanh(\boldsymbol{x_t} \cdot \boldsymbol{W}^{(x)} + \boldsymbol{h_{t-1}} \cdot \boldsymbol{W}_1^{(h)})$
    - Node 2 : $\boldsymbol{k_2} = \mathrm{ReLU}(\boldsymbol{k_1} \cdot \boldsymbol{W}_{2,1}^{(h)})$
    - Node 3 : $\boldsymbol{k_3} = \mathrm{ReLU}(\boldsymbol{k_2} \cdot \boldsymbol{W}_{3,2}^{(h)})$
    - Node 4 : $\boldsymbol{k_4} = \tanh(\boldsymbol{k_1} \cdot \boldsymbol{W}_{4,1}^{(h)})$
    - Output : $\boldsymbol{h_t} = (\boldsymbol{k_3} + \boldsymbol{k_4}) \,/\, 2$
  - All recurrent cells share the same set of parameters
  - 4 activation functions are allowed
    - tanh, ReLU, identity, sigmoid
  - Search space
    - Recurrent cells have N nodes
      - $4^N \times (N-1)!$
      - If $N = 12$, there are approximately 10^{14} models in the search space

- k_ℓ : node ℓ of the cell computation
- x_t : input signal
- h_{t-1} : previous step output
- $W_{\ell,j}^{(h)}$: parameter matrix

Variable explanation



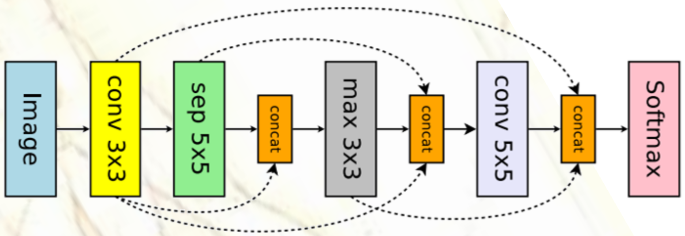
The DAG of recurrent cells



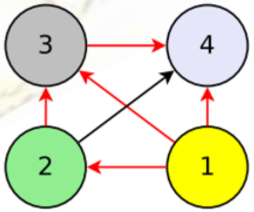
The recurrent cells

Methods (2/4)

- **Designing convolutional networks**
  - The controller RNN samples two decisions
    - Which edges are activated
    - Which computation operations are performed at each node
  - Previous connected nodes are allowed to form skip connections
  - Mechanism via 4 layers
    - Layer 2 concatenates the outputs of layer 1
    - Layer 3 concatenates the outputs of layer 1 and 2
    - Layer 4 concatenates the outputs of layer 1 and 3
  - 6 operations are allowed
    - Convolutions with filter sizes 3×3 and 5×5
    - Depthwise-separable convolutions with filter sizes 3×3 and 5×5
    - Max pooling and average pooling of kernel size 3×3
  - Search space
    - Convolutional networks have L layers
      - $6^L \times 2^{L(L-1)/2}$
      - If $L = 12$, there are approximately 1.6×10^{29} possible networks



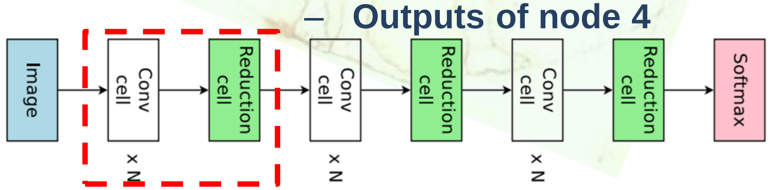
The complete network



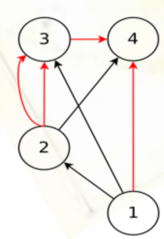
The DAG of network's architecture

Methods (3/4)

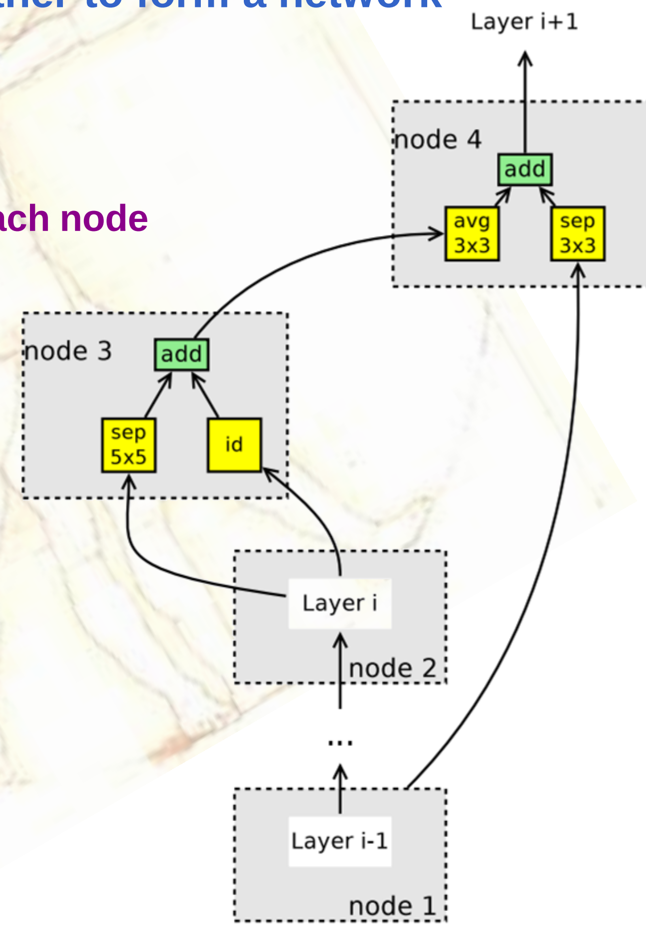
- **Designing convolutional cells**
  - **Design smaller modules and connect them together to form a network**
    - **Connect N convolution cells and 1 reduction cell**
  - **The controller RNN samples two decisions**
    - **Which edges are activated**
    - **Which computation operations are performed at each node**
  - **Mechanism via 4 nodes**
    - **Nodes 1, 2**
      - **Input nodes**
    - **Node 3**
      - **Sample node 2, node 2**
      - **Sample separable_conv_5×5, identity**
    - **Node 4**
      - **Sample node 3, node 1**
      - **Sample avg_pool_3×3, separable_conv_3×3**
    - **Output**
      - **Outputs of node 4**



Final network of convolutional cells



The DAG of convolutional cells



The convolutional cell

Methods (4/4)

- **Designing convolutional cells**
  - **Design reduction cells**
    - **Sample a computational graph from the search space**
    - **Apply all operations with a stride of 2**
    - **Reduce the spatial dimensions by a factor of 2**
  - **5 operations are allowed**
    - **Identity**
    - **Separable convolutions with kernel sizes 3×3 and 5×5**
    - **Max pooling and average pooling of kernel size 3×3**
  - **Search space**
    - **Convolutional cells have B nodes**
      - $(5 \times (B-2)!)^4$
      - **If $\mathrm{B} = 7$, there are approximately 1.3×10^{11} possible networks**

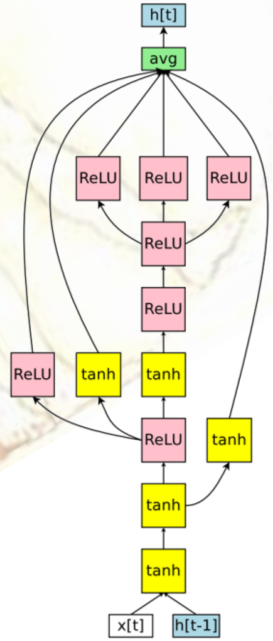
H. Pham, M. Guan, B. Zoph, Q. Le and J. Dean, “Efficient Neural Architecture Search via Parameter Shari ng," CLR, 2018.

Experiments (1/3)

- **Recurrent cells on the Penn Treebank**
  - Running on a single Nvidia GTX 1080Ti GPU
  - ENAS finds a recurrent cell in about 10 hours
  - Achieves a test perplexity of 56.3, which is on par with SOTA of 56.0
  - ENAS outperforms NAS by more than 6 perplexity points
  - ENAS is more than 1000 times faster than NAS in terms of GPU time

Architecture	Additional Techniques	Params (million)	Test PPL
LSTM (Zaremba et al., 2014)	Vanilla Dropout	66	78.4
LSTM (Gal & Ghahramani, 2016)	VD	66	75.2
LSTM (Inan et al., 2017)	VD, WT	51	68.5
RHN (Zilly et al., 2017)	VD, WT	24	66.0
LSTM (Melis et al., 2017)	Hyper-parameters Search	24	59.5
LSTM (Yang et al., 2018)	VD, WT, ℓ_2, AWD, MoC	22	57.6
LSTM (Merity et al., 2017)	VD, WT, ℓ_2, AWD	24	57.3
LSTM (Yang et al., 2018)	VD, WT, ℓ_2, AWD, MoS	**22**	**56.0**
NAS (Zoph & Le, 2017)	VD, WT	54	62.4
ENAS	VD, WT, ℓ_2	**24**	**56.3**

The perplexity on Penn Treebank of ENAS



The RNN cell ENAS discovered for Penn Treebank

Experiments (2/3)

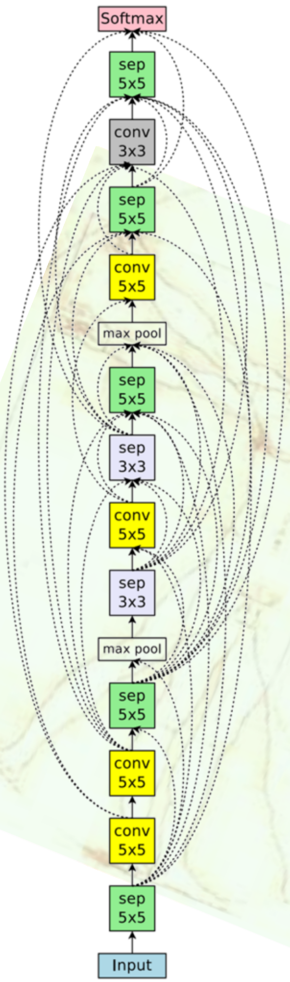
- **Convolutional architecture on the CIFAR-10**
  - **Entire convolutional networks**
    - **Increasing the number of filters in ENAS shows an error rate of 3.87%, not far from the error rate of the NAS's highest model, 3.65%**
    - **ENAS reduce the number of GPU-hours by more than 50,000 times compared to NAS.**
  - **Convolutional cells**
    - **ENAS with cutout achieves to 2.89% test error, on par with the 2.65% by NASNet-A with cutout**

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**

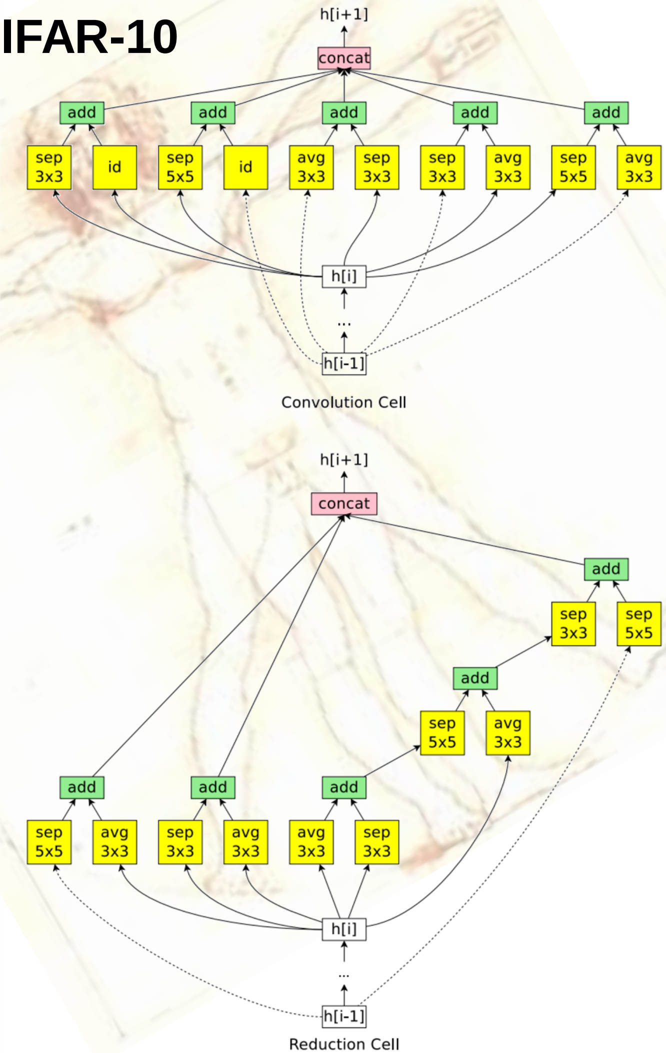
Classification errors of ENAS and baselines on CIFAR-10

Experiments (3/3)

- **Convolutional architecture on the CIFAR-10**



The convolutional network ENAS discovered for CIFAR-10



The convolutional cells ENAS discovered for CIFAR-10

H. Pham, M. Guan, B. Zoph, Q. Le and J. Dean, “Efficient Neural Architecture Search via Parameter Shari ng," CLR, 2018.