Boulder

Computer Vision; Image Classification; AutoML

YouTube Playlist

Maziar Raissi

Assistant Professor

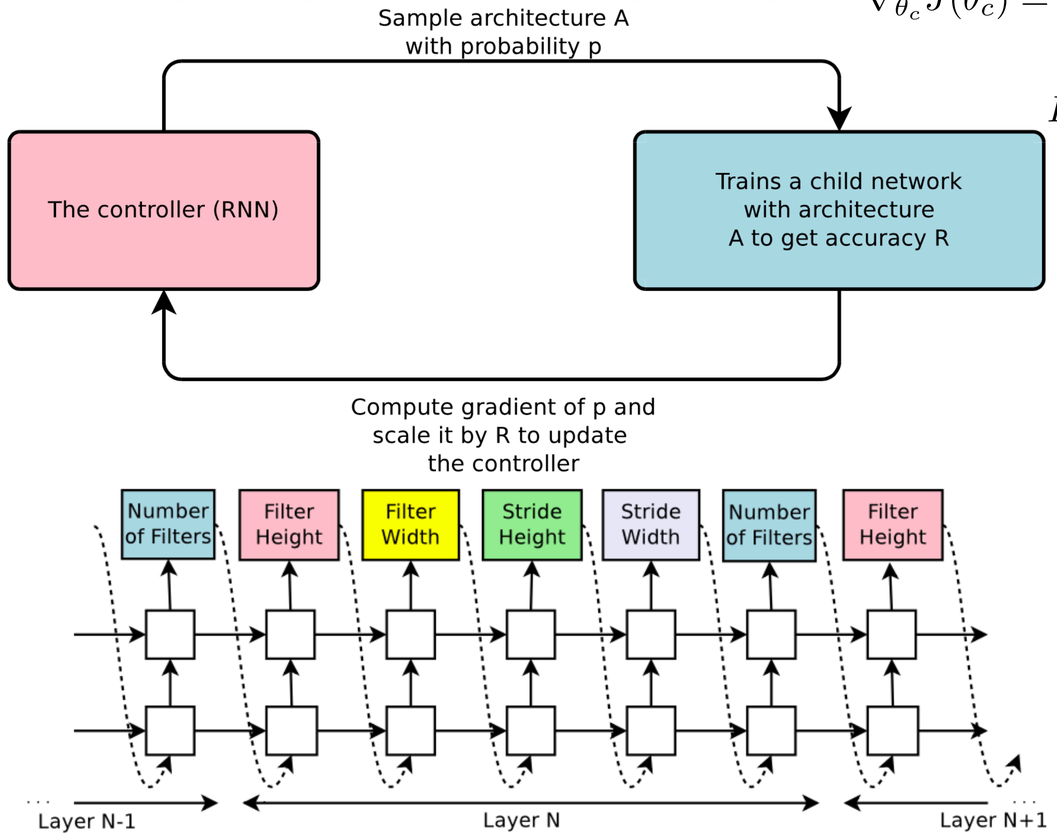
Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu

Neural Architecture Search With Reinforcement Learning

feature engineering $\rightarrow$ architecture engineering



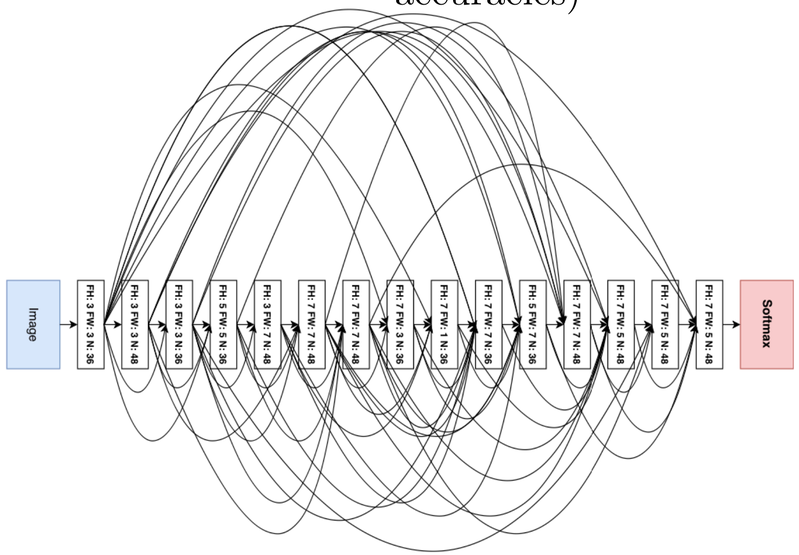
$$\nabla_{\theta_c} J(\theta_c) = \int \nabla_{\theta_c} P(a_{1:T};\theta_c) R = \int \frac{\nabla_{\theta_c} P(a_{1:T};\theta_c)}{P(a_{1:T};\theta_c)} P(a_{1:T};\theta_c) R = \int \nabla_{\theta_c} \log P(a_{1:T};\theta_c) P(a_{1:T};\theta_c) R$$

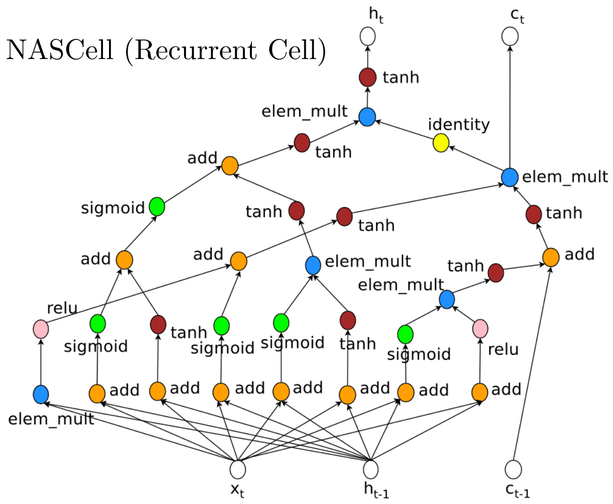
$$P(a_{1:T};\theta_c) = \prod_{t=1}^{T} P(a_t|a_{1:t-1};\theta_c) \implies \nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} \mathbb{E}_{P(a_{1:T};\theta_c)}[\nabla_{\theta_c} \log P(a_t|a_{1:t-1};\theta_c) R]$$

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

$m \rightarrow$ number of different architectures that the controller samples in one batch
$T \rightarrow$ number of hyperparameters the controller has to predict to design a neural network architecture
$R_k \rightarrow$ validation accuracy that the k-th neural network architecture achieves after being trained on a training dataset
$b \rightarrow$ baseline function (exponential moving average of the previous architecture accuracies)

$\theta_c \rightarrow$ parameters of the controller RNN
$a_{1:T} \rightarrow$ list of actions (list of tokens that the controller predicts to design an architecture for a child network)
$R \rightarrow$ reward signal (accuracy achieved by the child network on a held-out dataset)
$J(\theta_c) = \mathbb{E}_{P(a_{1:T};\theta_c)}[R] \rightarrow$ expected reward



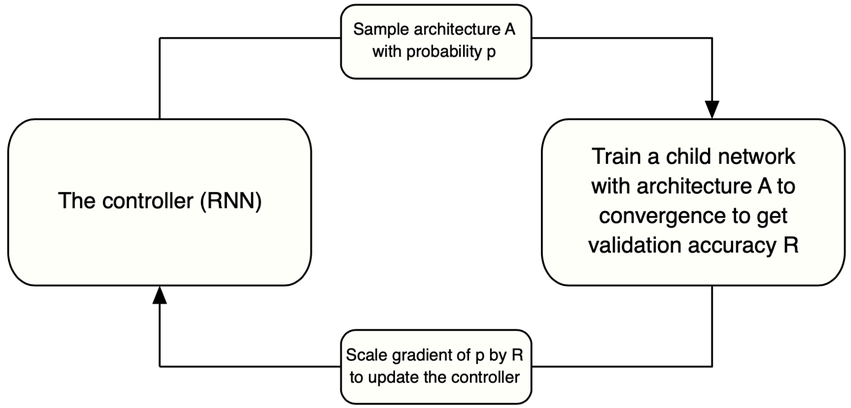


Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

Learning Transferable Architectures for Scalable Image Recognition

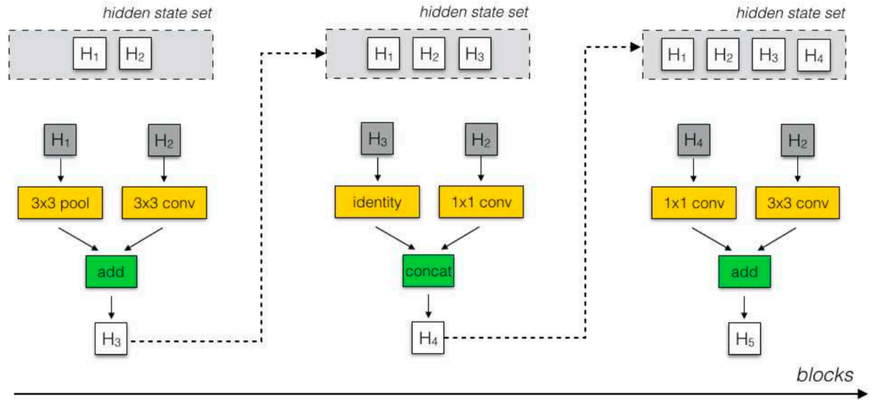
YouTube Video

feature engineering → architecture engineering

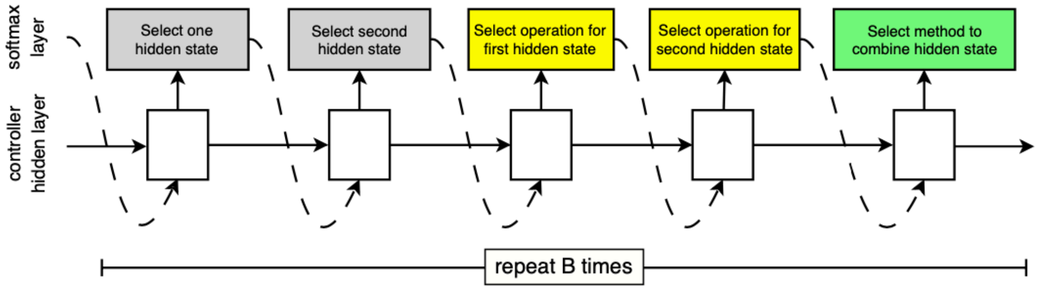


Normal Cell: convolutional cells that return a feature map of the same dimension

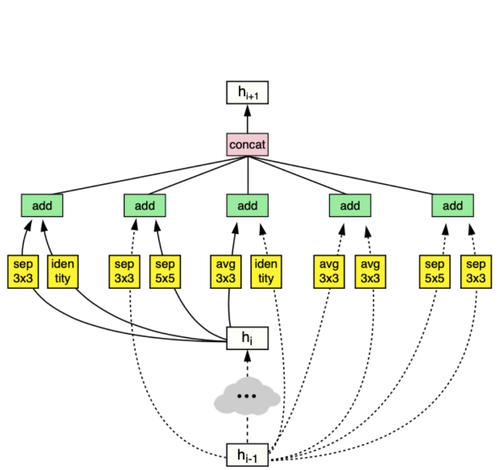
Reduction Cell: convolutional cells that return a feature map where the feature map height and width is reduced by a factor of two

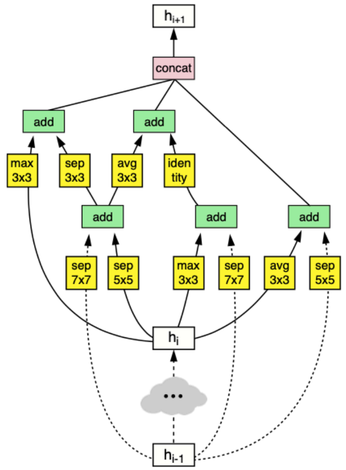


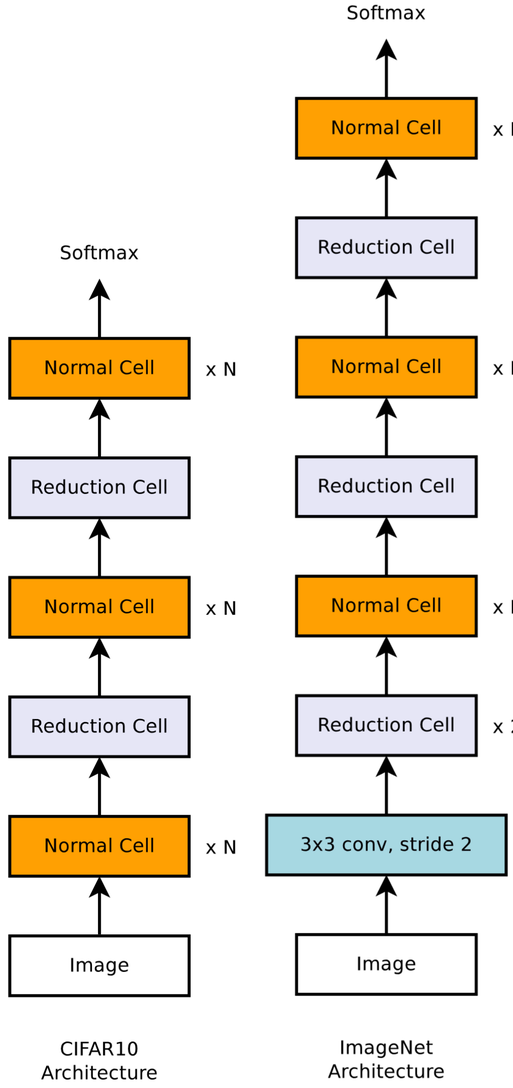
NASNet search space



- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv







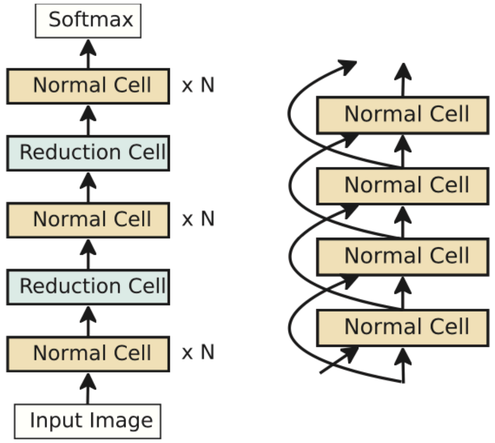
Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Regularized Evolution for Image Classifier Architecture Search

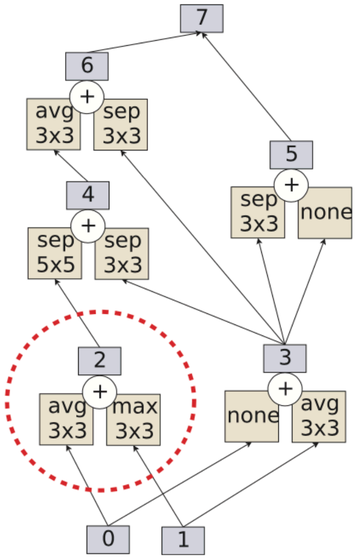
Boulder

YouTube Video

NASNet Search Space outer structure



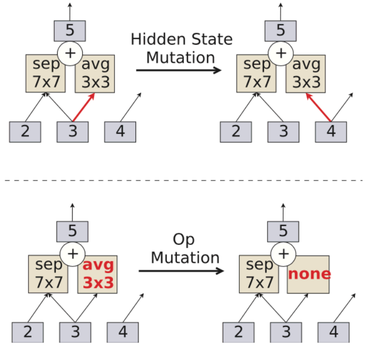
NASNet Search Space cell structure

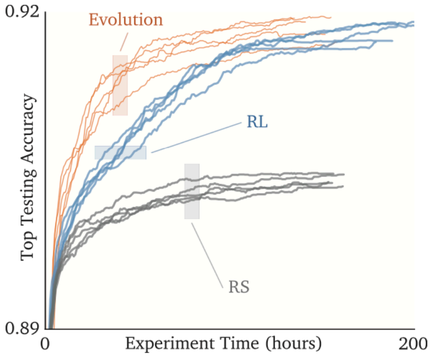


Evolution: Random Search + Selection

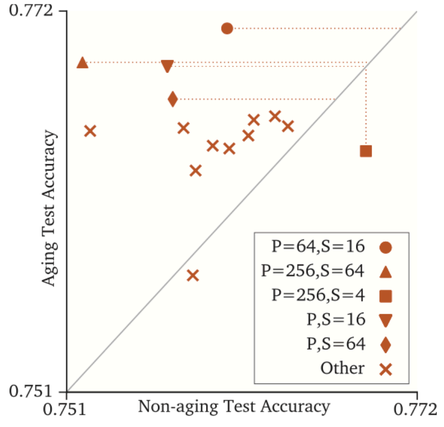
Algorithm 1 Aging Evolution (*i.e.* Regularized Evolution)

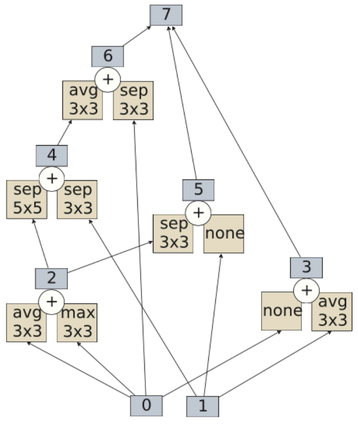
population ← empty queue ▷ The population.
history ← ∅ ▷ Will contain all models.
while |*population*| < *P* **do** ▷ Initialize population.
  model.arch ← RANDOMARCHITECTURE()
  model.accuracy ← TRAINANDEVAL(*model.arch*)
  add *model* to right of *population*
  add *model* to history
end while
while |*history*| < *C* **do** ▷ Evolve for *C* cycles.
  sample ← ∅ ▷ Parent candidates.
  while |*sample*| < *S* **do**
    candidate ← random element from *population*
      ▷ The element stays in the *population*.
    add *candidate* to *sample*
  end while
  parent ← highest-accuracy model in *sample*
  child.arch ← MUTATE(*parent.arch*)
  child.accuracy ← TRAINANDEVAL(*child.arch*)
  add *child* to right of *population*
  add *child* to *history*
  remove *dead* from left of *population* ▷ Oldest.
  discard *dead*
end while
return highest-accuracy model in *history*





Model	# Parameters	# Multiply-Adds	Top-1 / Top-5 Accuracy (%)
Incep-ResNet V2 (Szegedy et al. 2017)	55.8M	13.2B	80.4 / 95.3
ResNeXt-101 (Xie et al. 2017)	83.6M	31.5B	80.9 / 95.6
PolyNet (Zhang et al. 2017)	92.0M	34.7B	81.3 / 95.8
Dual-Path-Net-131 (Chen et al. 2017)	79.5M	32.0B	81.5 / 95.8
GeNet-2 (Xie and Yuille 2017)*	156M	–	72.1 / 90.4
Block-QNN-B (Zhong, Yan, and Liu 2018)*	–	–	75.7 / 92.6
Hierarchical (Liu et al. 2018b)*	64M	–	79.7 / 94.8
NASNet-A (Zoph et al. 2018)	88.9M	23.8B	82.7 / 96.2
PNASNet-5 (Liu et al. 2018a)	86.1M	25.0B	82.9 / 96.2
AmoebaNet-A (N=6, F=190)*	86.7M	23.1B	82.8 / 96.1
AmoebaNet-A (N=6, F=448)*	469M	104B	83.9 / 96.6





Real, Esteban, et al. "Regularized evolution for image classifier architecture search." *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 2019.

Evolving Deep Neural Networks

NeuroEvolution of Augmenting Topologies (NEAT)

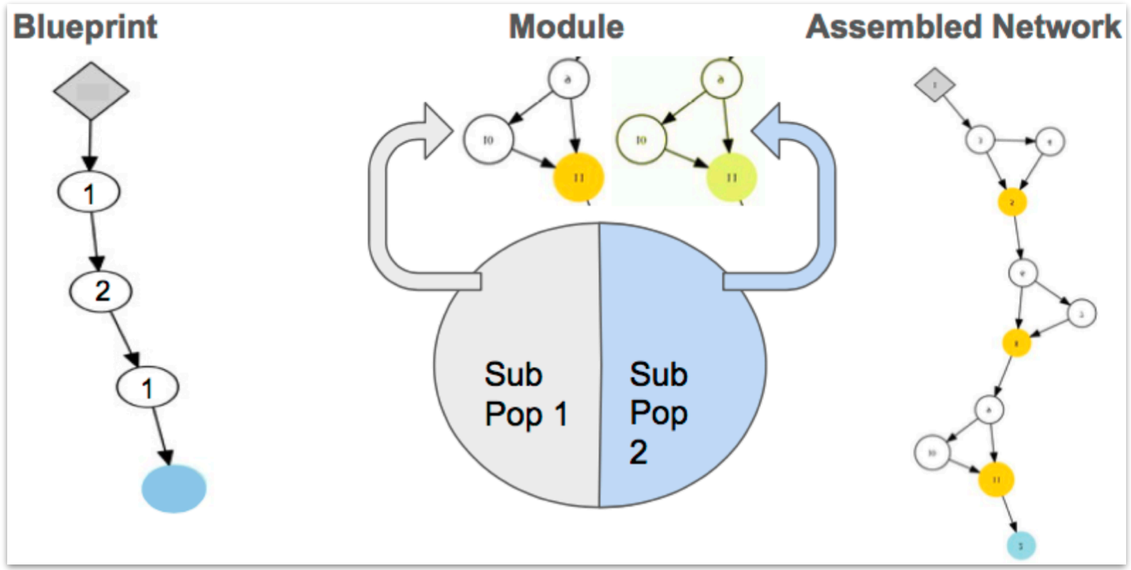
1) A population of chromosomes (each represented by a graph) with minimal complexity is created.
2) Over generations, structure (i.e. nodes (neurons) and edges (weights)) is added to the graph incrementally through mutation.
3) During crossover, historical markings are used to determine how genes of two chromosomes can be lined up.
4) The population is divided into species (i.e. subpopulations) based on a similarity metric.
5) Each species grows proportionally to its fitness and evolution occurs separately in each species.

DeepNeat

nodes (layers) edges (connections)

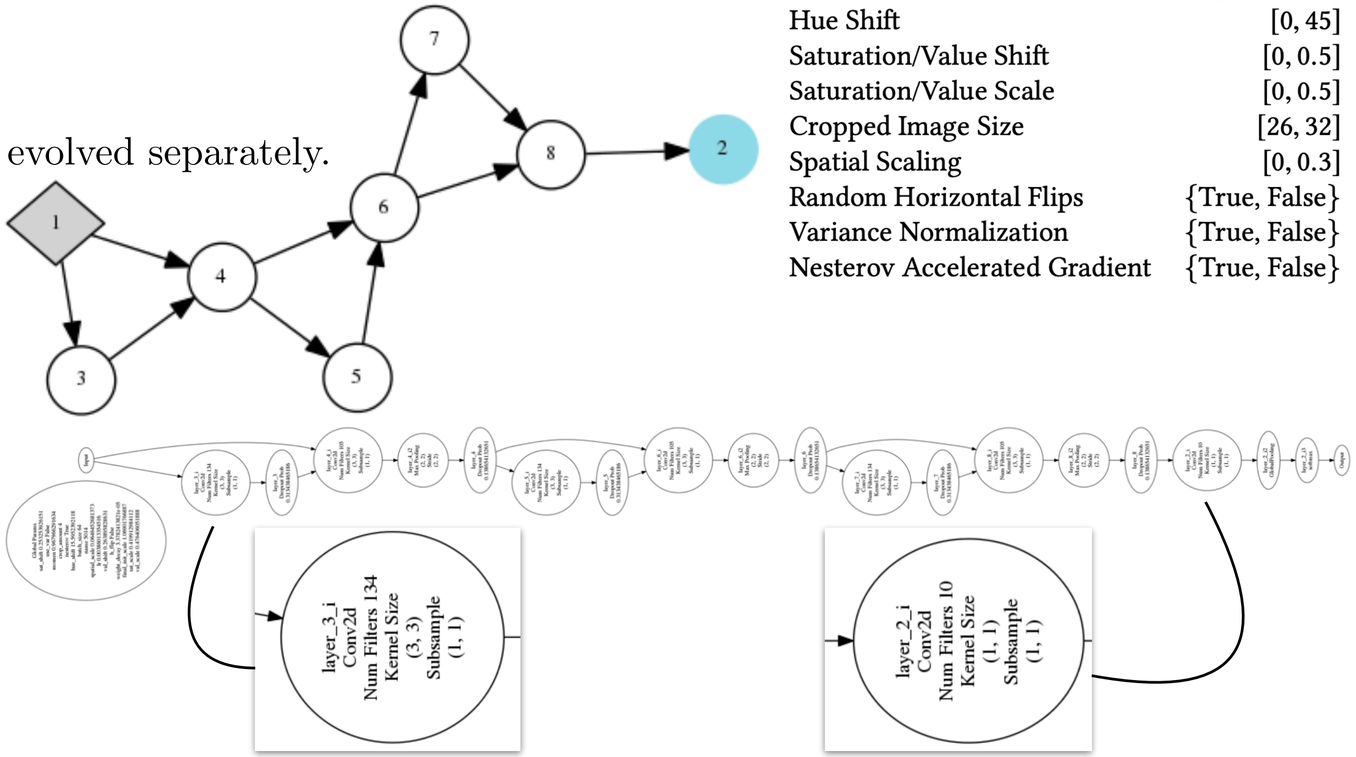
Coevolution DeepNEAT (CoDeepNEAT)

In CoDeepNEAT, two populations of modules and blueprints are evolved separately.



Node and global hyperparameters evolved in the CIFAR-10 domain.

Node Hyperparameter	Range
Number of Filters	[32, 256]
Dropout Rate	[0, 0.7]
Initial Weight Scaling	[0, 2.0]
Kernel Size	{1, 3}
Max Pooling	{True, False}
Global Hyperparameter	**Range**
Learning Rate	[0.0001, 0.1]
Momentum	[0.68, 0.99]
Hue Shift	[0, 45]
Saturation/Value Shift	[0, 0.5]
Saturation/Value Scale	[0, 0.5]
Cropped Image Size	[26, 32]
Spatial Scaling	[0, 0.3]
Random Horizontal Flips	{True, False}
Variance Normalization	{True, False}
Nesterov Accelerated Gradient	{True, False}



Miikkulainen, Risto, et al. "Evolving deep neural networks." Artificial intelligence in the age of neural networks and brain computing. Academic Press, 2019. 293-312.

Efficient Neural Architecture Search via Parameter Sharing

NAS: 450 GPUs for $3-4$ days ($\approx 32,400-43,200$ GPU-Hours)
ENAS: 16 GPU-Hours ($1000\times$ faster)

> The computational bottleneck of NAS is the training of each child model to convergence, only to measure its accuracy whilst throwing away all the trained weights.

Force all child models to share weights!

The controller RNN makes two sets of decisions:
1) two previous nodes to be used as inputs to the current node and
2) two operations to apply to the two sampled nodes.

$\theta \rightarrow$ parameters of the controller LSTM
$\omega \rightarrow$ shared parameters of the child models

$$\nabla_\omega \mathbb{E}_{\mathbf{m}\sim\pi(\mathbf{m};\theta)}\left[\mathcal{L}(\mathbf{m};\omega)\right] \approx \frac{1}{M}\sum_{i=1}^{M}\nabla_\omega \mathcal{L}(\mathbf{m}_i,\omega) \rightarrow \text{fixed } \theta$$

($\mathbf{m}\sim\pi(\mathbf{m};\theta)$: model; $\mathcal{L}(\mathbf{m};\omega)$: standard cross-entropy loss)

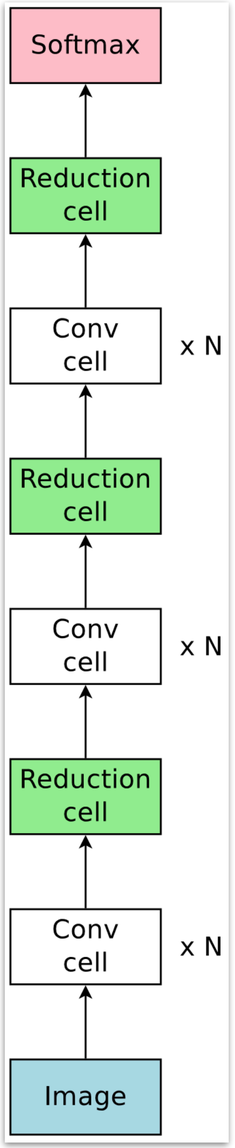
$M = 1$

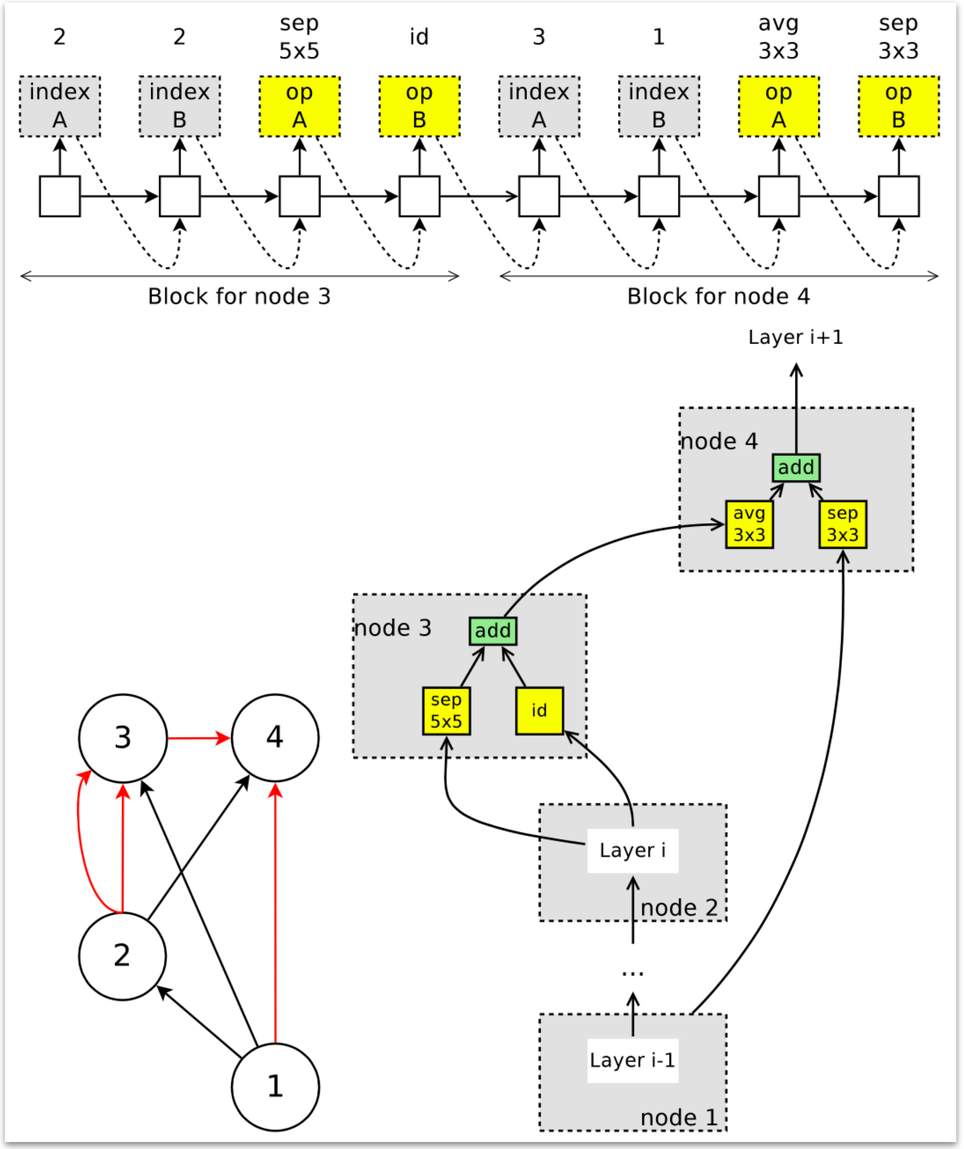
$$\mathbb{E}_{\mathbf{m}\sim\pi(\mathbf{m};\theta)}\left[\mathcal{R}(\mathbf{m},\omega)\right] \rightarrow \text{fixed } \omega$$

($\mathcal{R}(\mathbf{m},\omega)$: reward (e.g., accuracy))

Method	GPUs	Times (days)	Params (million)	Error (%)
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.





Pham, Hieu, et al. "Efficient neural architecture search via parameters sharing." *International Conference on Machine Learning*. PMLR, 2018.

DARTS: Differentiable Architecture Search

– 2000 GPU days of Reinforcement Learning (RL)
– 3150 GPU days of Evolution
– 1.5 to 4 GPU days

Search Space

Cell $\rightarrow$ a directed acyclic graph consisting of an ordered sequence of N nodes

$x^{(i)} \rightarrow$ node (a latent representation such as a feature map in CNNs)

$o^{(i,j)} \rightarrow$ edge (some operation that transforms $x^{(i)}$)

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)})$$

Each cell has two input nodes and a single output node

Inputs to convolutional cells:
Cell outputs in the previous two layers

Inputs to recurrent cells:
Input at the current step and the state carried from the previous step

Zero operation: Lack of connection between two nodes

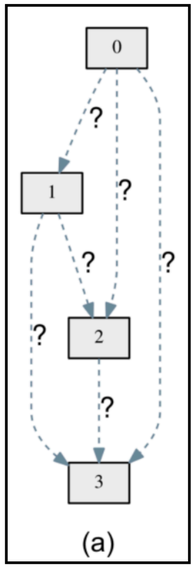
Continuous Relaxation and Optimization

$\mathcal{O} \rightarrow$ set of candidate operations (e.g., conv, max pooling, zero)

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

$\alpha^{(i,j)} \in \mathbb{R}^{|\mathcal{O}|} \rightarrow$ operation mixing weight

$\alpha = \{\alpha^{(i,j)}\} \rightarrow$ encoding the architecture



Bilevel Optimization Problem

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

$w \rightarrow$ lower-level variable

$\alpha \rightarrow$ upper-level variable

Approximate Architecture Gradient

$$\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\approx \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$$

$$\nabla_\alpha \mathcal{L}_{val}(w', \alpha) - \xi \underbrace{\nabla^2_{\alpha,w} \mathcal{L}_{train}(w, \alpha) \nabla_{w'} \mathcal{L}_{val}(w', \alpha)}_{\text{Finite Difference}}$$

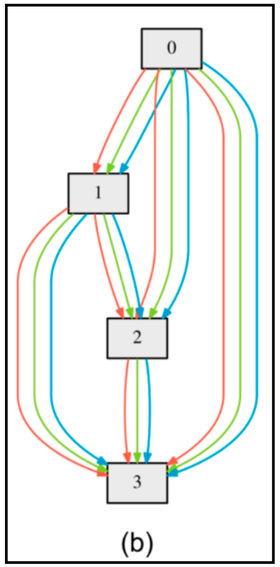
Algorithm 1: DARTS – Differentiable Architecture Search

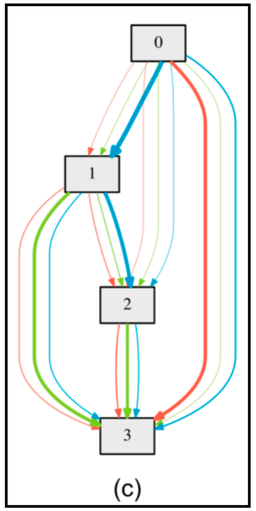
Create a mixed operation $\bar{o}^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each edge (i, j)

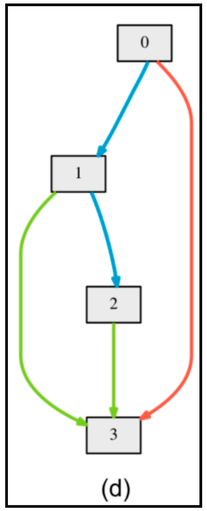
while *not converged* **do**

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.







$$o^{(i,j)} = \text{argmax}_{o \in \mathcal{O}} \ \alpha_o^{(i,j)}$$

CIFAR-10 $\rightarrow$ ImageNet $***$ Penn Treebank $\rightarrow$ WikiText-2

Comparison with state-of-the-art image classifiers on ImageNet in the mobile setting.

Architecture	Test Error (%) top-1	Test Error (%) top-5	Params (M)	+× (M)	Search Cost (GPU days)	Search Method
Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	1448	–	manual
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	569	–	manual
ShuffleNet 2× ($g = 3$) (Zhang et al., 2017)	26.3	–	~5	524	–	manual
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	564	2000	RL
NASNet-B (Zoph et al., 2018)	27.2	8.7	5.3	488	2000	RL
NASNet-C (Zoph et al., 2018)	27.5	9.0	4.9	558	2000	RL
AmoebaNet-A (Real et al., 2018)	25.5	8.0	5.1	555	3150	evolution
AmoebaNet-B (Real et al., 2018)	26.0	8.5	5.3	555	3150	evolution
AmoebaNet-C (Real et al., 2018)	24.3	7.6	6.4	570	3150	evolution
PNAS (Liu et al., 2018a)	25.8	8.1	5.1	588	~225	SMBO
DARTS (searched on CIFAR-10)	26.7	8.7	4.7	574	4	gradient-based

Comparison with state-of-the-art language models on WT2.

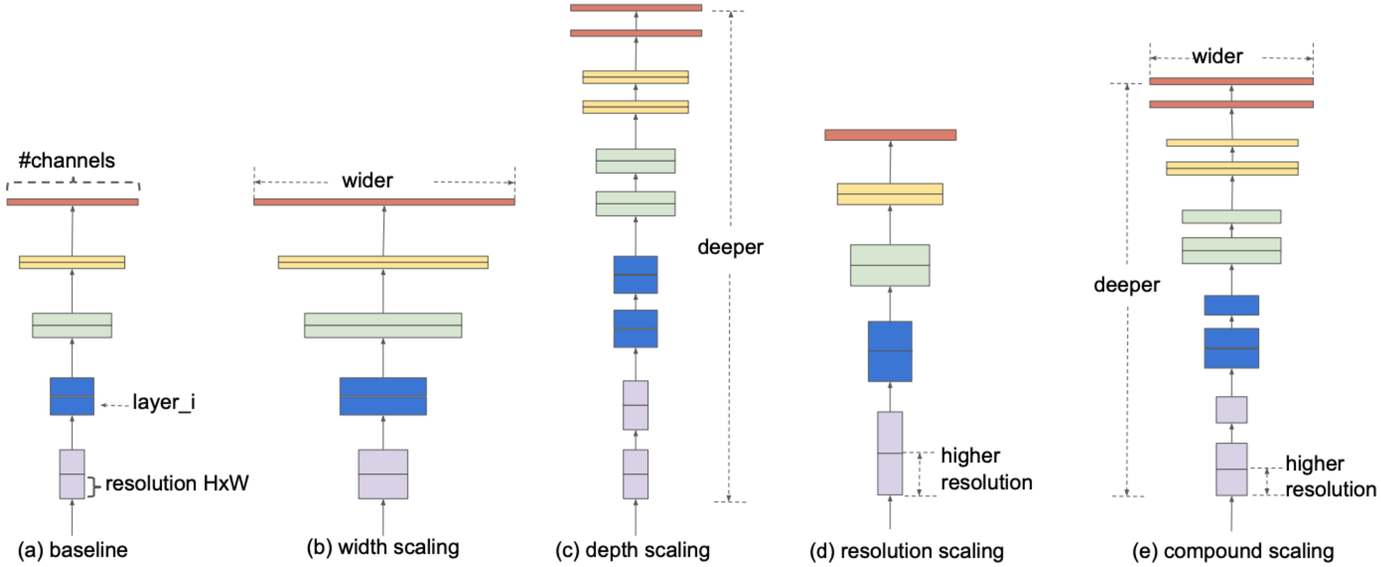
Architecture	Perplexity valid	Perplexity test	Params (M)	Search Cost (GPU days)	Search Method
LSTM + augmented loss (Inan et al., 2017)	91.5	87.0	28	–	manual
LSTM + continuous cache pointer (Grave et al., 2016)	–	68.9	–	–	manual
LSTM (Merity et al., 2018)	69.1	66.0	33	–	manual
LSTM + skip connections (Melis et al., 2018)	69.1	65.9	24	–	manual
LSTM + 15 softmax experts (Yang et al., 2018)	66.0	63.3	33	–	manual
ENAS (Pham et al., 2018b)† (searched on PTB)	72.4	70.4	33	0.5	RL
DARTS (searched on PTB)	71.2	69.6	33	1	gradient-based

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

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Boulder

YouTube Video



$d \rightarrow$ depth scaling coefficient
$w \rightarrow$ width scaling coefficient
$r \rightarrow$ resolution scaling coefficient

$$\max_{d,w,r} \quad \text{accuracy}$$

$$\text{s.t.} \quad \text{memory} \leq \text{target\_memory}$$
$$\text{flops} \leq \text{target\_flops}$$

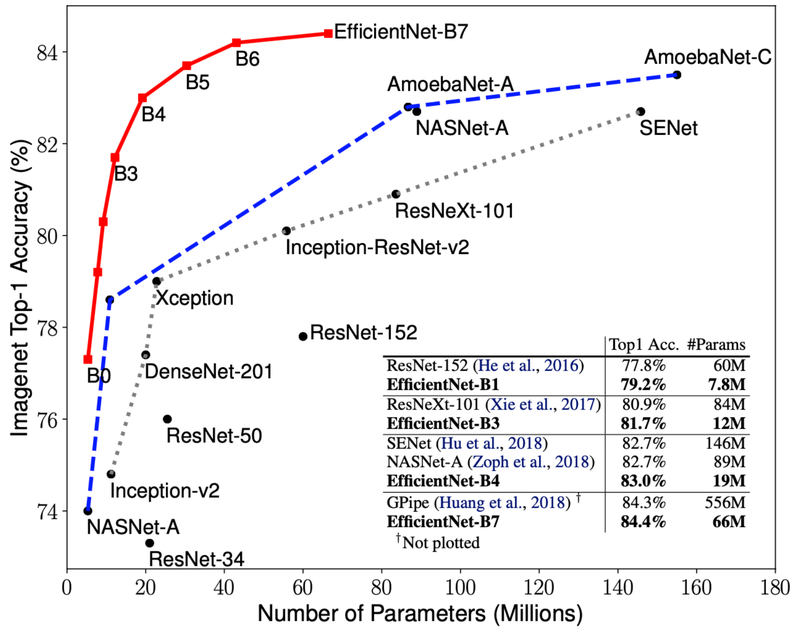
Compound Scaling

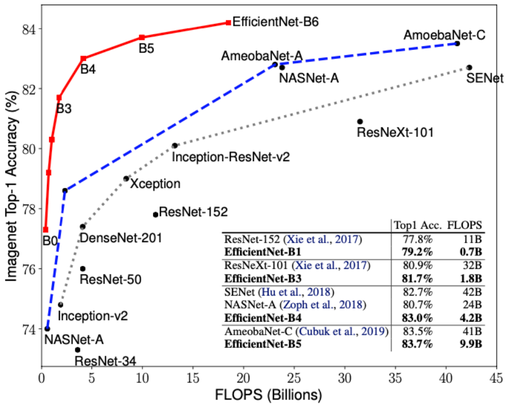
$\phi \rightarrow$ compound coefficient
$d = \alpha^\phi, w = \beta^\phi, r = \gamma^\phi$
$\alpha\beta^2\gamma^2 \approx 2, \underbrace{\alpha \geq 1, \beta \geq 1, \gamma \geq 1}_{\text{small grid search}}$

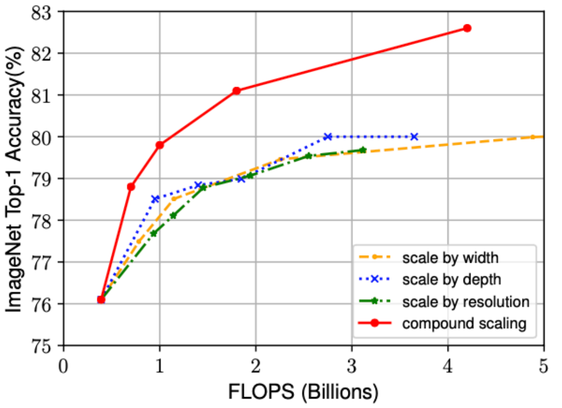
EfficientNet

$\max \text{accuracy} \cdot (\text{flops}/\text{target\_flops})^\omega$
$\omega = -0.07 \rightarrow$ controls the tradeoff btw accuracy & flops

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

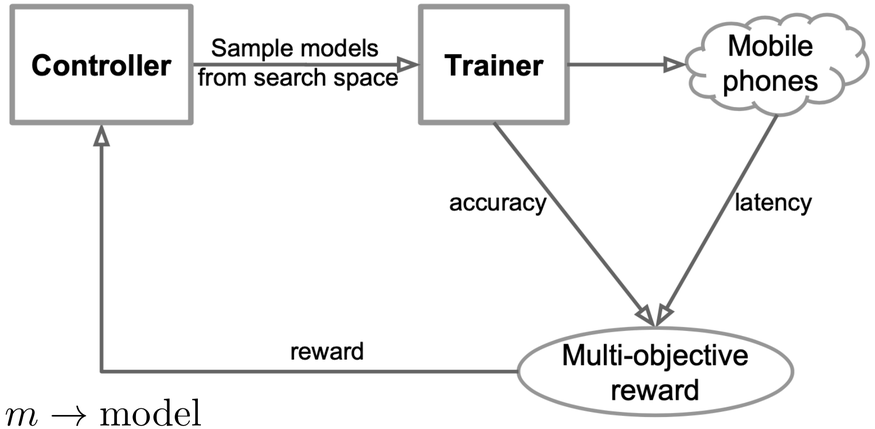






Tan, Mingxing, and Quoc V. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *arXiv preprint arXiv:1905.11946* (2019).

MnasNet: Platform-Aware Neural Architecture Search for Mobile



$m \rightarrow$ model

$\text{ACC}(m) \rightarrow$ accuracy of the model on the target task

$\text{LAT}(m) \rightarrow$ inference latency on the target mobile platform

$T \rightarrow$ target latency

Hard Constraint

$$\underset{m}{\text{maximize}} \quad ACC(m)$$

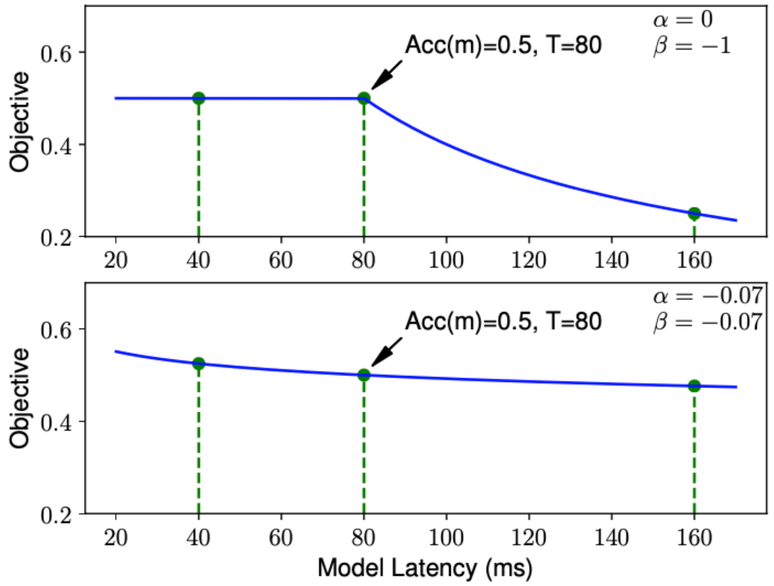
$$\text{subject to} \quad LAT(m) \leq T$$

Soft Constraint

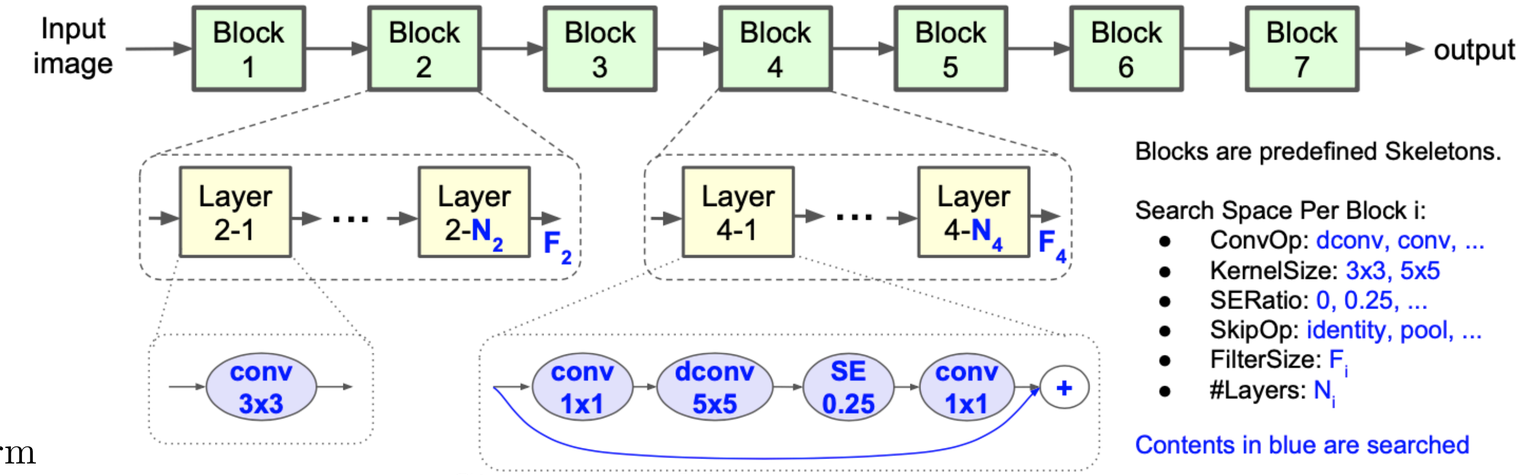
$$\underset{m}{\text{maximize}} \quad ACC(m) \times \left[\frac{LAT(m)}{T}\right]^w$$

$$w = \begin{cases} \alpha, & \text{if } LAT(m) \leq T \\ \beta, & \text{otherwise} \end{cases}$$

Empirical observation: doubling the latency usually brings about 5% relative accuracy gain.



Factorized Hierarchical Search Space



Blocks are predefined Skeletons.

Search Space Per Block i:

- ConvOp: dconv, conv, ...
- KernelSize: 3x3, 5x5
- SERatio: 0, 0.25, ...
- SkipOp: identity, pool, ...
- FilterSize: F_i
- #Layers: N_i

Contents in blue are searched

Given two models:

(1) m_1 has latency l and accuracy a

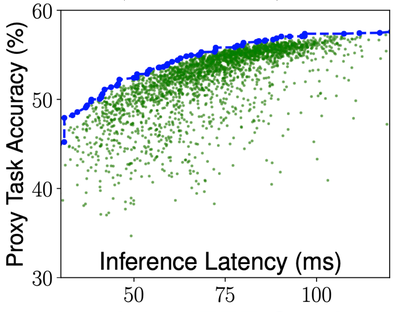
(2) m_2 has latency $2l$ and 5% higher accuracy $1.05a$

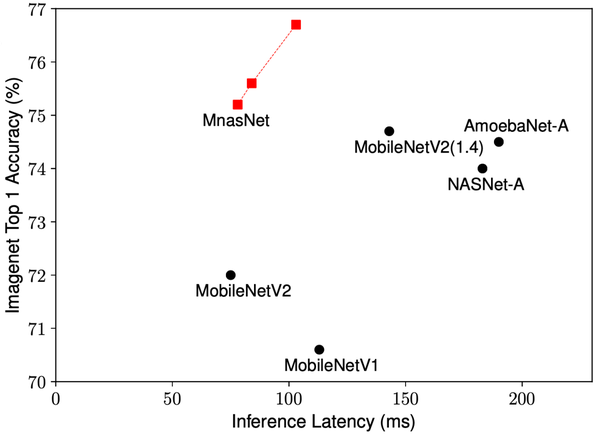
They should have similar rewards:

$R(m_2) = 1.05a(2l/T)^\beta \approx R(m_1) = a(l/T)^\beta$.

Solving this gives $\beta \approx -0.07$.

Therefore, use $\alpha = \beta = -0.07$.



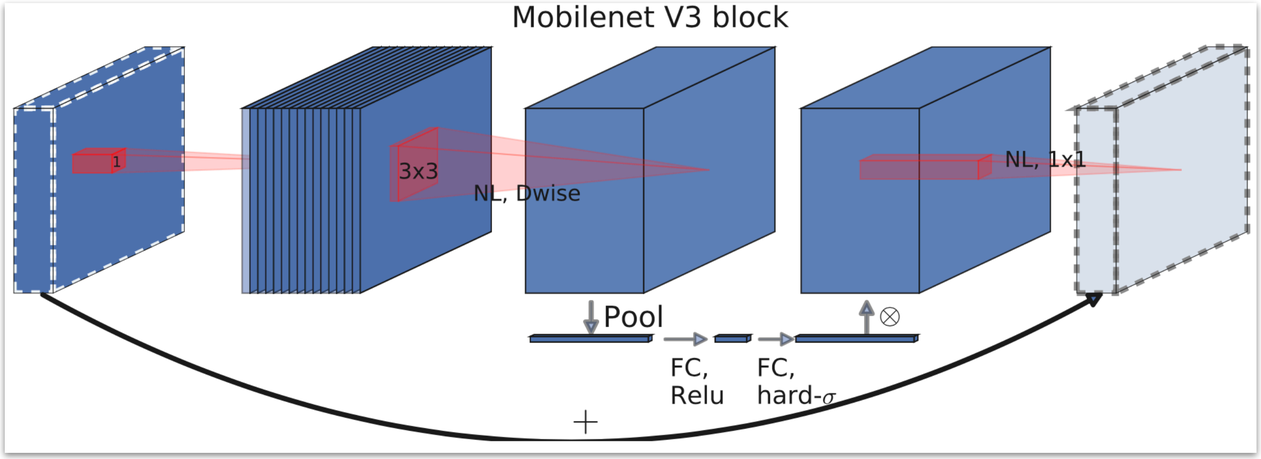


Tan, Mingxing, et al. "Mnasnet: Platform-aware neural architecture search for mobile." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

Searching for MobileNetV3

Efficient Mobile Building Blocks

MobileNetV1 (Depth-wise separable convolutions):
– spatial filtering: light weight depth-wise convolution
– feature generation: heavier 1×1 point-wise convolution
MobileNetV2 (linear bottleneck and inverted residual structure)
MnasNet (lightweight attention modules based on squeeze and excitation)

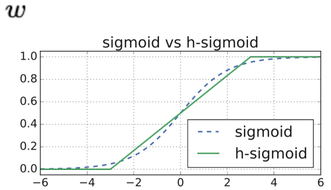


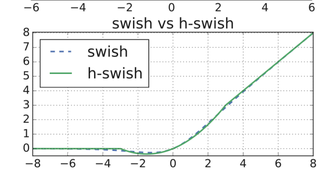
Platform-Aware NAS for Block-wise Search

MnasNet-A1 $\rightarrow$ large model

$$\underset{m}{\text{maximize}} \quad ACC(m) \times \left[\frac{LAT(m)}{T}\right]^w$$

$m \rightarrow$ model
$T \rightarrow$ target latency
$w = -0.15 < -0.07 \rightarrow$ small model





Nonlinearities

$$\text{h-swish}[x] = x\frac{\text{ReLU6}(x+3)}{6}$$

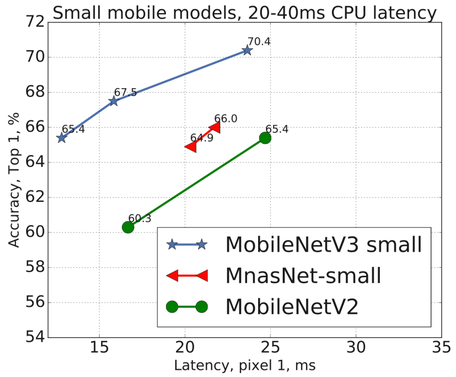
NetAdapt for Layer-wise Search

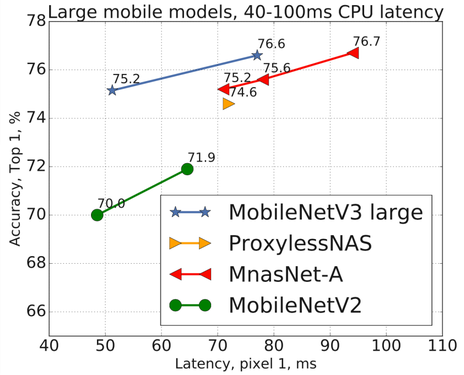
1. Starts with a seed network architecture found by platform-aware NAS.
2. For each step:
   (a) Generate a set of new *proposals*. Each proposal represents a modification of an architecture that generates at least δ reduction in latency compared to the previous step.
   (b) For each proposal we use the pre-trained model from the previous step and populate the new proposed architecture, truncating and randomly initializing missing weights as appropriate. Fine-tune each proposal for T steps to get a coarse estimate of the accuracy.
   (c) Selected best proposal according to some metric.
3. Iterate previous step until target latency is reached.

types of proposals:

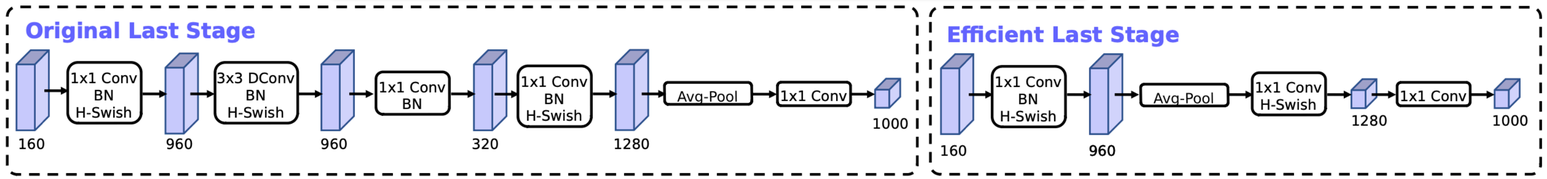
1. Reduce the size of any expansion layer;
2. Reduce bottleneck in all blocks that share the same bottleneck size - to maintain residual connections.

$L \rightarrow$ latency of seed model
$T = 10000$
metric: $\max \frac{\Delta \text{Acc}}{|\Delta \text{latency}|}$





Redesigning Expensive Layers



Howard, Andrew, et al. "Searching for mobilenetv3." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.

Boulder

Questions?

YouTube Playlist