Learning Transferable Architectures for Scalable Image Recognition

(CVPR 2018)

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le
Google Brain

https://qdata.github.io/deep2Read

Presenter: Arshdeep Sekhon

Fall 2018

Neural Architecture Search Categories

- Search Space: which architectures can be represented in principle.
- Search Method: exploration-exploitation trade-off
- Estimating performance: reduction of cost

Motivation

- inspired by NAS by RL method
- computationally expensive for large datasets
- search for a good architecture on a proxy smaller dataset (CIFAR-10), and then transfer the learned architecture to ImageNet

Motivation

- inspired by NAS by RL method
- computationally expensive for large datasets
- search for a good architecture on a proxy smaller dataset (CIFAR-10), and then transfer the learned architecture to ImageNet
- Design a new search space: NASNet Search Space
- complexity of the architecture is independent of the depth of the network and the size of input images

NASNet Search Space

- all convolutional networks are composed of convolutional layers (or cells) with identical structure but different weights.
- Searching for the best convolutional architectures: reduced to searching for the best cell structure.
- Searching for the best cell structure has two benefits:
    - faster than searching for an entire network architecture
    - the cell itself is more likely to generalize to other problems

Method: Search Method

- Search method: NAS

Method: Search Method

- Search method: NAS
- a controller recurrent neural network (RNN) samples child networks with different architectures.
- The child networks are trained to convergence to obtain some accuracy on a held-out validation set.
- The resulting accuracies are used to update the controller so that the controller will generate better architectures over time.
- The controller weights are updated with policy gradient

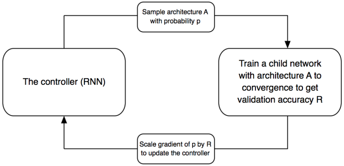


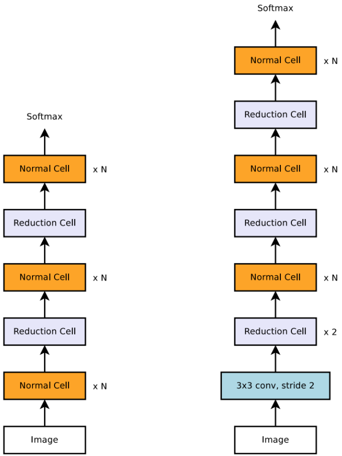
Figure 1. Overview of Neural Architecture Search [71]. A controller RNN predicts architecture A from a search space with probability p. A child network with architecture A is trained to convergence achieving accuracy R. Scale the gradients of p by R to update the RNN controller.

Method: Search Space

- the overall architectures of the convolutional nets are manually predetermined
- convolutional cells repeated many times where each convolutional cell has the same architecture, but different weights.
- learn two types of cells:
  - 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

Method

- fixed architecture for CIFAR 10 and ImageNet
- consider the number of motif repetitions N and the number of initial convolutional filters as free parameters



Method: combining search space and search method

- each cell receives as input two initial hidden states h_i and h_{i-1} which are the outputs of two cells in previous two lower layers or the input image.
- The controller RNN recursively predicts the rest of the structure of the convolutional cell, given these two initial hidden states.
- The predictions of the controller for each cell are grouped into B blocks,
- where each block has 5 prediction steps made by 5 distinct softmax classifiers

Method

1. Select a hidden state from h_i, h_{i-1} or from the set of hidden states created in previous blocks.
2. Select a second hidden state from the same options as in Step 1.
3. Select an operation to apply to the hidden state selected in Step 1.
4. Select an operation to apply to the hidden state selected in Step 2.
5. Select a method to combine the outputs of Step 3 and 4 to create a new hidden state

Method

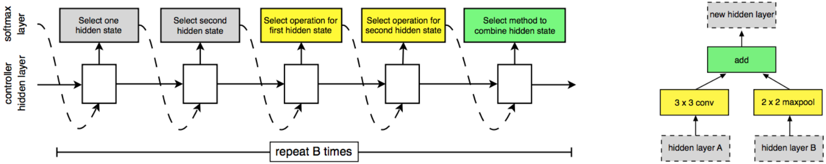


Figure 3. Controller model architecture for recursively constructing one block of a convolutional cell. Each block requires selecting 5 discrete parameters, each of which corresponds to the output of a softmax layer. Example constructed block shown on right. A convolutional cell contains B blocks, hence the controller contains $5B$ softmax layers for predicting the architecture of a convolutional cell. In our experiments, the number of blocks B is 5.

Method: operation choices

- 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

For the combining in Step 5:

- element wise addition
- concatenation along filter dimension

Experiments: Top performing Reduction and Normal Cells

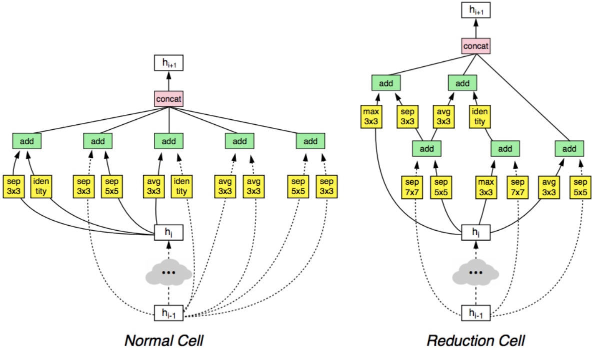


Figure 4. Architecture of the best convolutional cells (NASNet-A) with $B = 5$ blocks identified with CIFAR-10 . The input (white) is the hidden state from previous activations (or input image). The output (pink) is the result of a concatenation operation across all resulting branches. Each convolutional cell is the result of B blocks. A single block is corresponds to two primitive operations (yellow) and a combination operation (green). Note that colors correspond to operations in Figure 3.

Controller

- one-layer LSTM with 100 hidden units at each layer
- $2 \times 5B (= 5)$ softmax predictions for the two convolutional cells
- Each of the 10B predictions of the controller RNN is associated with a probability.
- joint probability of a child network is the product of all probabilities at these 10B softmaxes.
- joint probability is used to compute the gradient for the controller RNN.
- The gradient is scaled by the validation accuracy of the child network to update the controller RNN
- so that controller assigns low probabilities for bad child networks and high probabilities for good child.
- use Proximal Policy Optimization policy gradient optimization

Results on CIFAR10

model	depth	# params	error rate (%)
DenseNet ($L = 40, k = 12$) [26]	40	1.0M	5.24
DenseNet($L = 100, k = 12$) [26]	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) [26]	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) [26]	190	25.6M	3.46
Shake-Shake 26 2x32d [18]	26	2.9M	3.55
Shake-Shake 26 2x96d [18]	26	26.2M	2.86
Shake-Shake 26 2x96d + cutout [12]	26	26.2M	2.56
NAS v3 [71]	39	7.1M	4.47
NAS v3 [71]	39	37.4M	3.65
NASNet-A (6 @ 768)	-	3.3M	3.41
NASNet-A (6 @ 768) + cutout	-	3.3M	2.65
NASNet-A (7 @ 2304)	-	27.6M	2.97
NASNet-A (7 @ 2304) + cutout	-	27.6M	2.40
NASNet-B (4 @ 1152)	-	2.6M	3.73
NASNet-C (4 @ 640)	-	3.1M	3.59

Table 1. Performance of Neural Architecture Search and other state-of-the-art models on CIFAR-10. All results for NASNet are the mean accuracy across 5 runs.

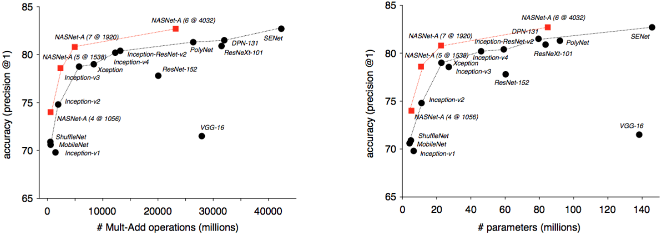


Figure 5. Accuracy versus computational demand (left) and number of paramet rs (right) across top performing published CNN architec-

Results on ImageNet

Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [29]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (5 @ 1538)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [60]	299×299	23.8 M	5.72 B	78.8	94.4
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [58]	299×299	55.8 M	13.2 B	80.1	95.1
NASNet-A (7 @ 1920)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [68]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [69]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
SENet [25]	**320×320**	**145.8 M**	**42.3 B**	**82.7**	**96.2**
NASNet-A (6 @ 4032)	**331×331**	**88.9 M**	**23.8 B**	**82.7**	**96.2**

Table 2. Performance of architecture search and other published state-of-the-art models on ImageNet classification. Mult-Adds indicate the number of composite multiply-accumulate operations for a single image. Note that the composite multiple-accumulate operations are calculated for the image size reported in the table. Model size for [25] calculated from open-source implementation.

Model	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V1 [59]	6.6M	1,448 M	69.8 †	89.9
MobileNet-224 [24]	4.2 M	569 M	70.6	89.5
ShuffleNet (2x) [70]	~ 5M	524 M	70.9	89.8
NASNet-A (4 @ 1056)	**5.3 M**	**564 M**	**74.0**	**91.6**
NASNet-B (4 @ 1536)	5.3M	488 M	72.8	91.3
NASNet-C (3 @ 960)	4.9M	558 M	72.5	91.0

Table 3. Performance on ImageNet classification on a subset of models operating in a constrained computational setting, i.e., $<$ 1.5 B multiply-accumulate operations per image. All models use 224x224 images. † indicates top-1 accuracy not reported in [59] but from open-source implementation.

Efficiency of architecture search methods

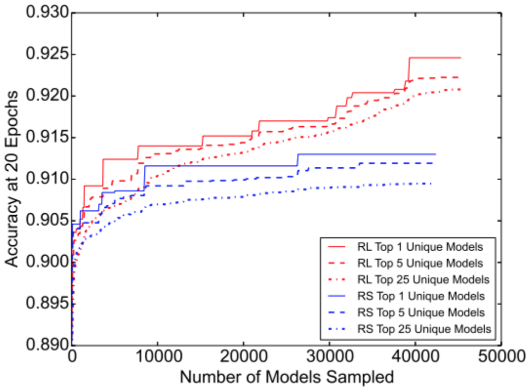


Figure 6. Comparing the efficiency of random search (RS) to reinforcement learning (RL) for learning neural architectures. The x-axis measures the total number of model architectures sampled, and the y-axis is the validation performance on CIFAR-10 after 20 epochs of training.