📖

[논문리뷰] NASNet [2018]

✨ 해당 논문은 자주 사용되는 데이터에서 **모델 아키텍쳐가 학습하는 방법을 연구**합니다.
transfereability가 가능한 NASNet search space design과 관련된 연구를 설명합니다.

> 💡 작은 데이터로 building block을 찾은 후, **큰 데이터 셋으로 transfer**합니다.
> 즉 **transferability를 가능하게 하는 NASNet search space를 제안합니다.**

해당 연구는 강화학습 방식을 활용한 (NAS) framework의 영감을 받았습니다.

NASNet architecture를 활용하여, **transferrabilty를 달성**하였기에, **이미지의 크기와 NeuralNet의 깊이와 무관하게 사용하는 것이 가능합니다.**

Best cell structure를 **Searching하는 것은 2가지 장점**을 가지고 있습니다.

- Architecture가 **transferrabilty를 찾는 것이 매우 빠릅니다.**
- Cell 자체가 또 **다른 문제로 일반화 될 가능성이 큽니다.**

NASNets에 의하여, **학습된 image features는 유용**하고, **Vision과 관련된 문제를 생성**합니다.

Method

NAS framework는 RNN을 활용하여, 서로 다른 구조의 child network를 활용합니다.
child networks는 validation set에서 Accuracy를 얻고, convergence를 얻기 위해 학습합니다.

이때, Controller를 업데이트하고, 그리하여, better architectures를 얻게 됩니다.

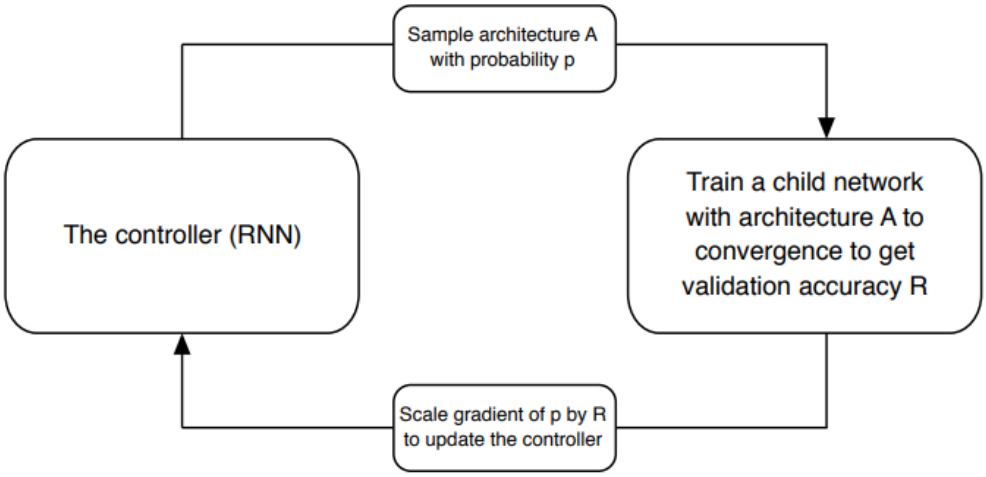


Figure 1. Overview of Neural Architecture Search [71]. A controller RNN predicts architecture A from a search space with probability p. A child network with architecture A is trained to convergence achieving accuracy R. Scale the gradients of p by R to update the RNN controller.

CIFAR-10 dataset이 찾은 **architecture가 더 크고, 높은 해상도를 가진 확장**되도록, **새로운 search space를 설계합니다.**
이러한 search space를 NASNet이라고 부릅니다.

NASNet search space는 **일반적인 CNN의 구성인 filter banks, nonlinearities connection의 조합이 반복되어, 설계**되었습니다.

Controller RNN은 일반적인 **Convolutional cell을 예측하는 것이 가능**해집니다.
Input으로부터 두 가지 feature map을 얻기 위해, 두 가지 **Convolutional Cell을 제안**합니다.

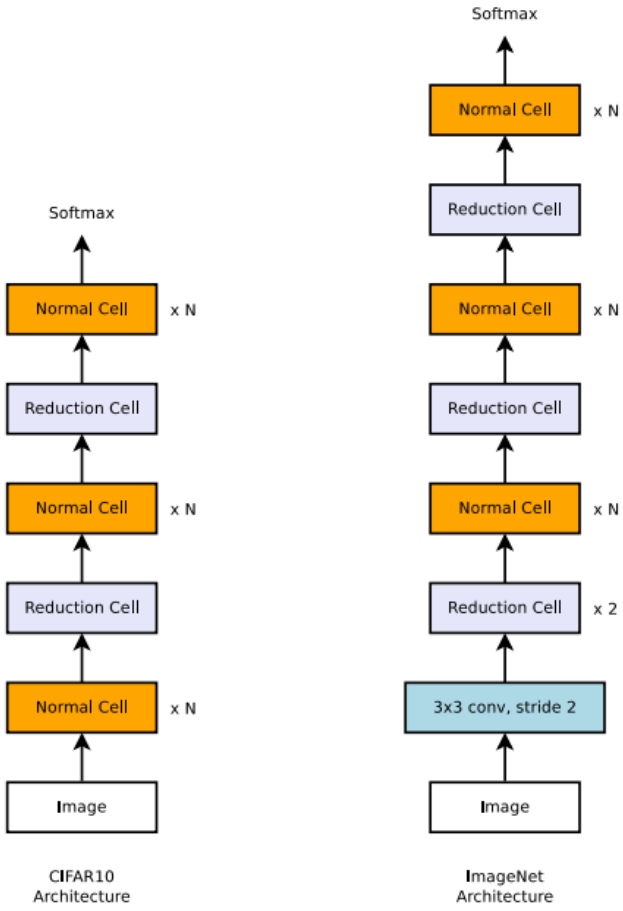


Figure 2. Scalable architectures for image classification consist of two repeated motifs termed *Normal Cell* and *Reduction Cell*. This diagram highlights the model architecture for CIFAR-10 and ImageNet. The choice for the number of times the Normal Cells that gets stacked between reduction cells, N, can vary in our experiments.

🌟 **이미지 분류 문제의 규모를 맞추기 위해 motif 반복 횟수 N과 초기 Convolutional filter의 수**를 **Hyperparameter로 설정**합니다.

[1] Normal Cell : **같은 차원의 stride 1인 형태**로 **feature map으로 반환**합니다.

[2] Reduction Cell : **높이와 너비가 2배 감소된 stride 1 또는 2**인 **feature map으로 반환**합니다.

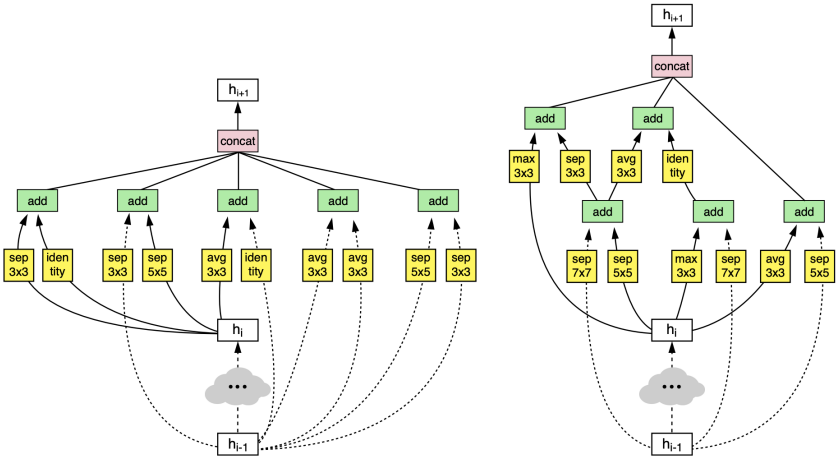


Figure 4. Architecture of the best convolutional cells (NASNet-A) with $B = 5$ blocks identified with CIFAR-10 . The input (white) is the hidden state from previous activations (or input image). The output (pink) is the result of a concatenation operation across all resulting branches. Each convolutional 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.

🌟 Reduction cell은 feature map의 크기를 감소시키기 위해 사용됩니다.
reduction cell과 normal cell을 찾고, 이러한 Cell들을 쌓아, Dataset에 적용하는 것입니다.

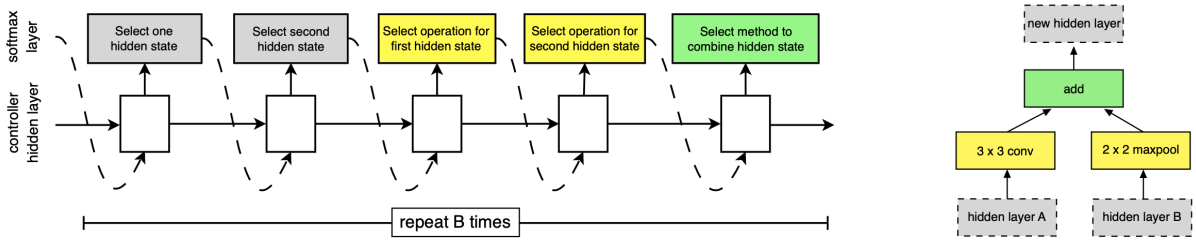


Figure 3. Controller model architecture for recursively constructing one block of a convolutional cell. Each block requires selecting 5 discrete parameters, each of which corresponds to the output of a softmax layer. Example constructed block shown on right. A convolutional cell contains B blocks, hence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

🌟 Reduction Cell과 Normal Cell은 Controller RNN과 강화학습으로 찾은 것입니다.

Block → Cell → Architecture 순으로, **훨씬 효율적인 Search Space를 찾는 것**입니다.

Controller RNN이 Cell을 찾는 과정

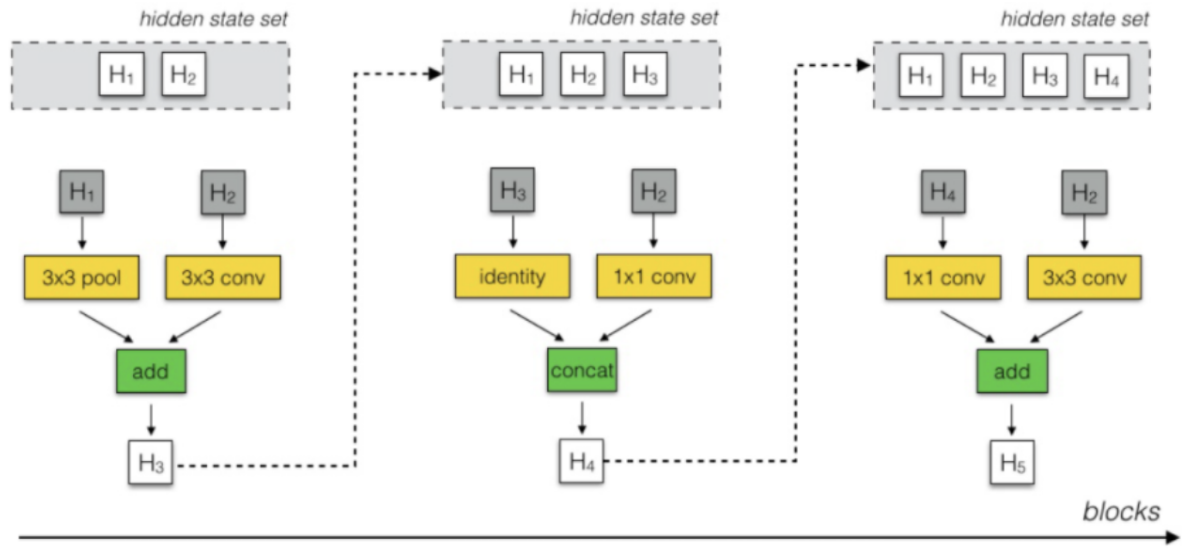
1. h_i, h_{i-1} 로부터 hidden state 하나를 선택합니다.
   h_i, h_{i-1}는 이전 block에서 생성된 hidden state입니다.
2. 1번째와 동일하게 두 번째 hidden state를 선택합니다.
3. 1번째, 2번째에서 선택된 hidden state에 적용할 연산을 선택합니다.

- 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

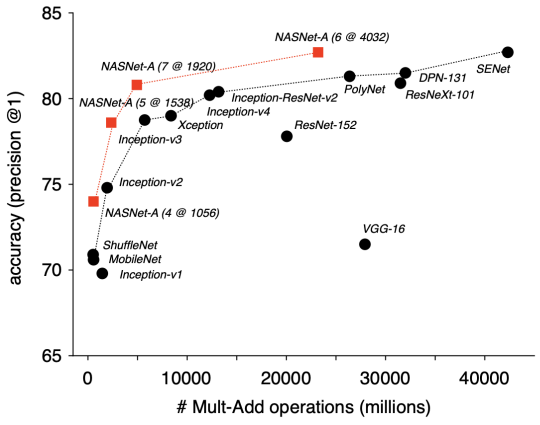
4. Controller RNN은 3번째를 통한 출력된 값들을 결합할 방법을 선택합니다.
    - **element-wise addition betweent two hidden states**
    - **concatenation between two hidden states**

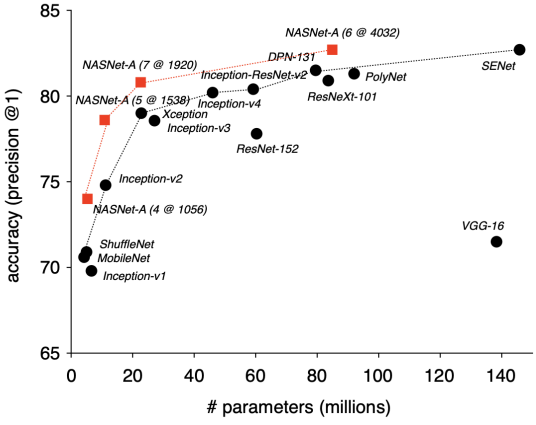
🌟 **마지막으로, Convolutional cell에서 생성된 unused hidden states에서는 finall cell output을 위하여, 깊이와 함께 결합됩니다.**

💡 **Controller RNN의 작동 과정을 시각화한 그림**입니다.



Experiment & Result





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.

Conclusion

학습된 Arcithecture는 computatioanl cost와 parameter측면에서 매우 유연하게 접근이 가능합니다.

해당 연구의 핵심은 깊은 network로부터 복잡한 Architecture를 분리하여,

search space를 설계하는 것입니다.

Reference

Learning Transferable Architectures for Scalable Image Recognition

Developing neural network image classification models often requires significant architecture engineering. In this paper, we study a method to learn the model architectures directly on the dataset of interest. As this approach is expensive when the dataset is large, we propose to search

https://arxiv.org/abs/1707.07012

[논문 읽기] NasNet(2018) 리뷰, Learning Transferable Architectures for Scalable Image Recognition

이번에 읽어볼 논문은 NasNet, Learning Transferable Architectures for Scalabel Image Recognition 입니다. NasNet은 RNN을 활용하여 생성된 convolution block으로 구성되어 있습니다. 이와 대조적으로 ResNet, Inception, MobileNet 등등은 사람이 블락을 설계하고, 블락을 쌓아서 모델을 구축했었습니다. 대표적으로 ResNet은 residual

https://deep-learning-study.tistory.com/543