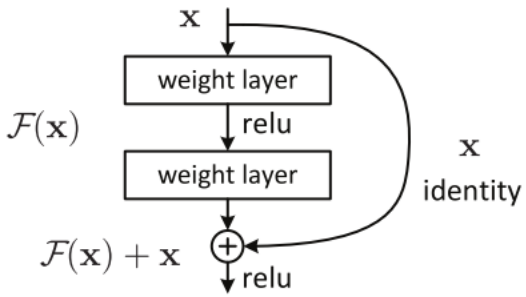
# Exploring Randomly Wired Neural Networks for Image Recognition

1기 박소영

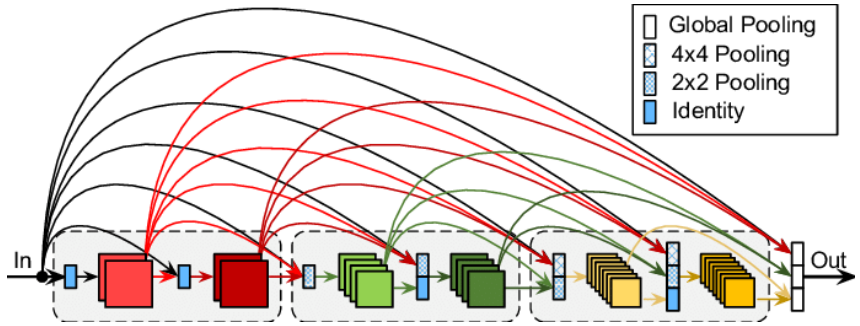
# Introduction

# 01. INTRO

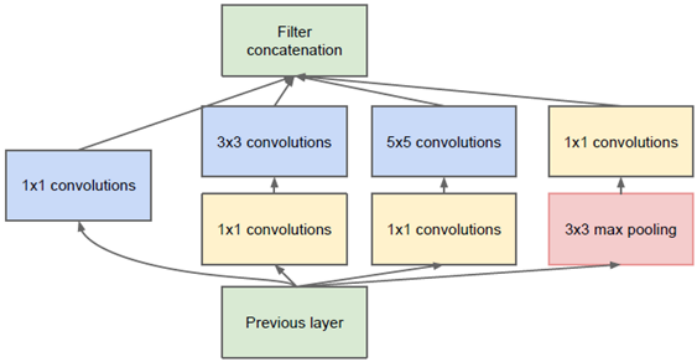
## 오늘 날의 네트워크 트렌드




ResNet




DenseNet




Inception

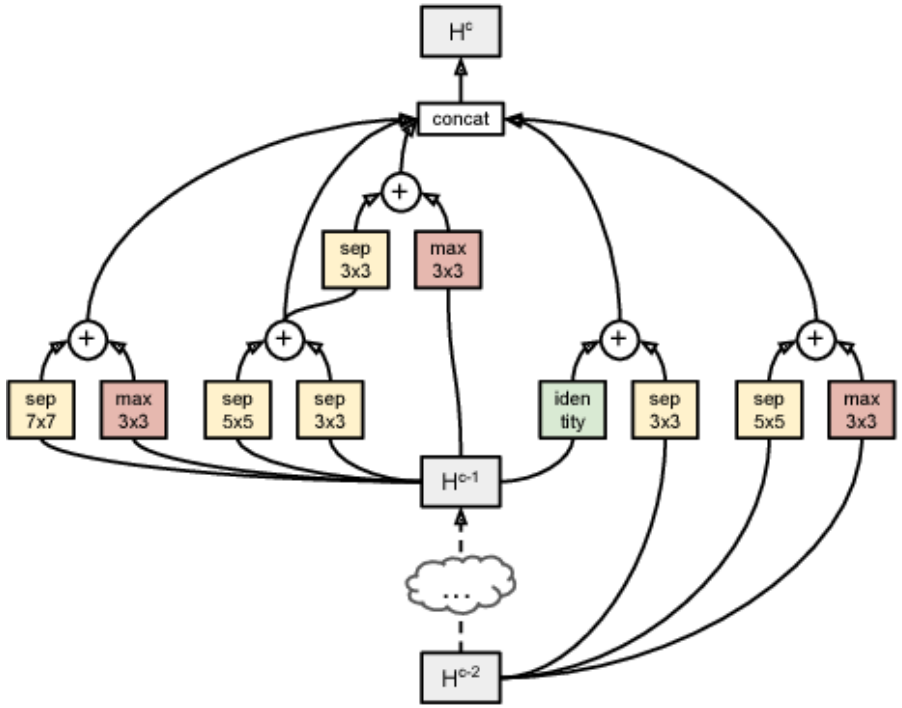
# 01. INTRO

## NAS : Neural Architecture Search

## Neural Architecture Search (NAS)

: 사람이 뉴럴 네트워크의 구조를 직접 찾지 않고 자동으로 찾게 해주는 것

- 강화학습 기반
- 많은 뉴럴 네트워크를 학습해야 한다



# 01. INTRO

## 생각의 발단

딥러닝에서 feature를 뽑는다

# 01. INTRO

## 생각의 발단

네트워크를 디자인해서

딥러닝에서 feature를 뽑는다

# 01. INTRO

## 생각의 발단

네트워크를 생성하는 제네레이터를 만들어서

네트워크를 디자인해서

딥러닝에서 feature를 뽑는다

# 01. INTRO

## 그로 인한 결과

이를 위해 randomly하게 연결된 뉴럴 네트워크를 고안
3가지 랜덤 그래프로부터 뉴럴 네트워크를 생성

# Methodology

# 02. METHOD

## Stochastic network generator

Network generator

$$g : \Theta \longmapsto \mathcal{N}$$

# 02. METHOD

## Stochastic network generator

Network generator $g: \Theta \longmapsto \mathcal{N}$

Stochastic network generator $g(\theta, s)$

# 02. METHOD

## Randomly wired neural networks

NAS

Network generator

# 02. METHOD

## How to generate a Random wired network?

### 1) Generating general graphs

뉴럴 네트워크의 기반이 될 그래프를 생성

# 02. METHOD

How to generate a Random wired network?

## 2) Mapping

### Edge operation

그래프에서의 edge : 노드에서 노드로 정보를 보내는 것

### Node operation

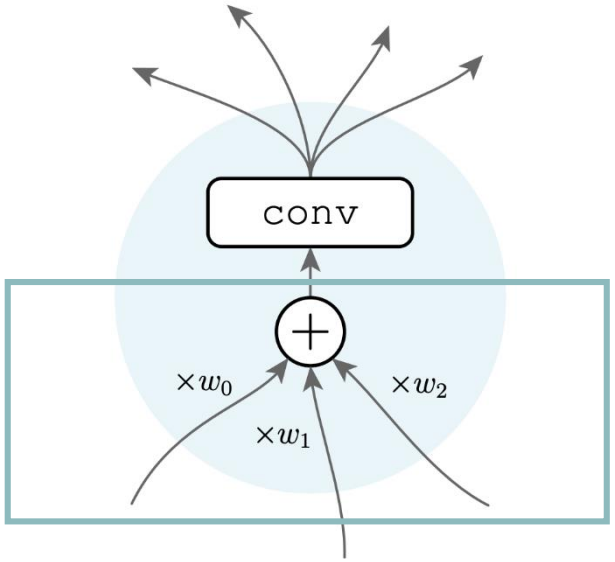
3가지 operation이 순차적으로 일어난다

# 02. METHOD

## Mapping - Node operation

### 1)Aggregation

Input을 받고 Weight들을 sum

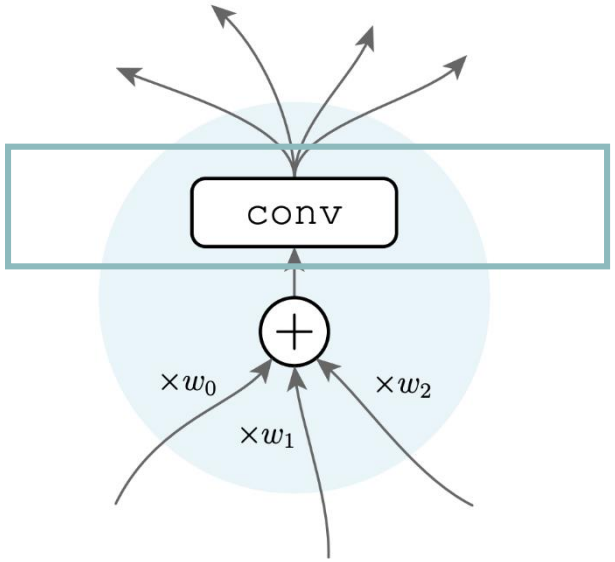


# 02. METHOD

## Mapping - Node operation

### 2)Transformation

Sum 후에 conv를 통과
(ReLU – Convolution – BN triplet)
Conv 3*3, 1*1 사용 : 똑같은 operation

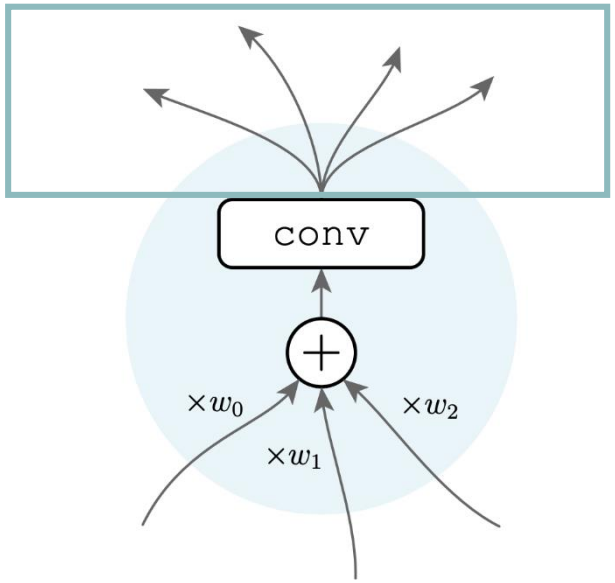


# 02. METHOD

## Mapping - Node operation

### 3)Distribution

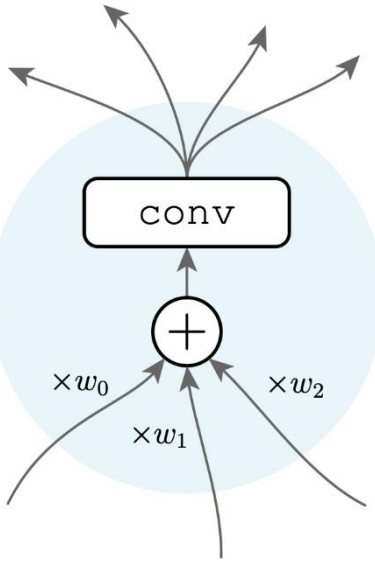
Transformation된 정보는
Output으로 나간다



# 02. METHOD

## Mapping - Node operation에 의한 영향

1. Input node와 Output node의 채널 수가 같다
-> 어떤 노드와도 연결할 수 있다
-> convolution 계산이 너무 커지지 않는다




2. Transformation은 FLOP, 파라미터 수도 일정

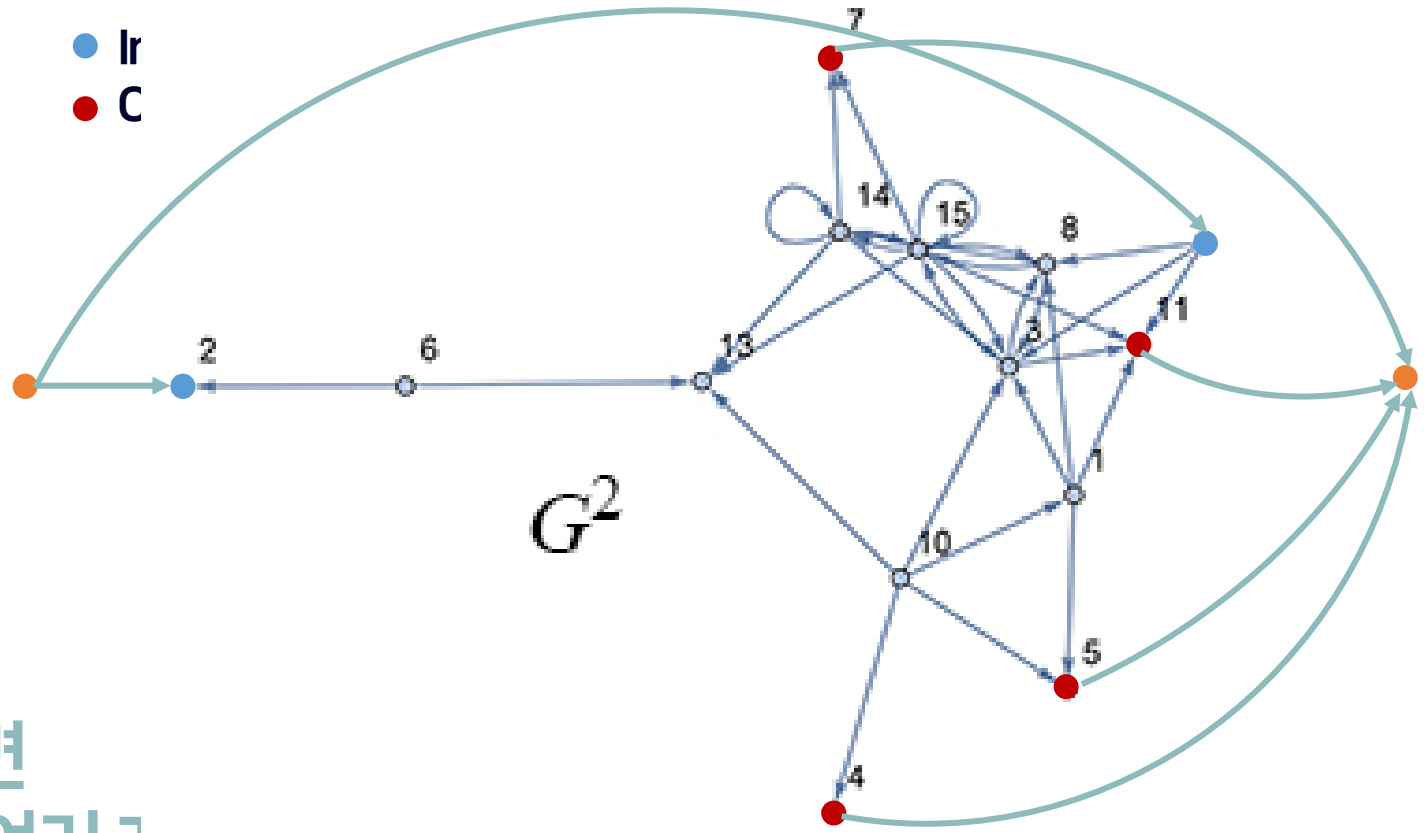
3. Aggregation, distribution모두 파라미터가 없다

4. 전체 FLOP은 노드 수에 비례.

5. wiring만 설정하여 이게 어떻게 영향을 끼치는지를 볼 수 있게 된다

# 02. METHOD

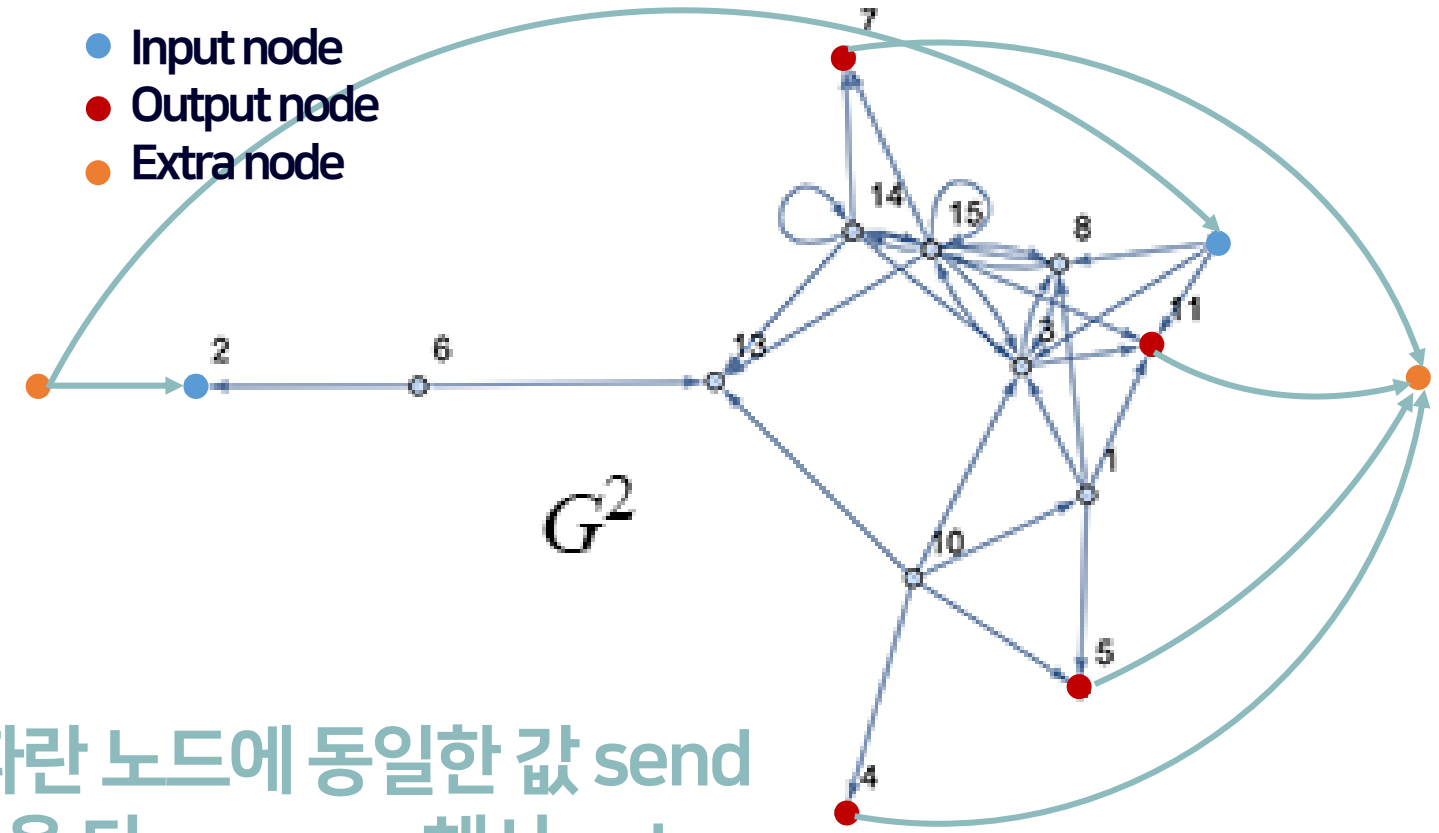
## Mapping - Input and Output nodes




일반 랜덤 그래프를 만들면
Input, Output node가 여러 개 생길 수 있나

# 02. METHOD

## Mapping - Input and Output nodes



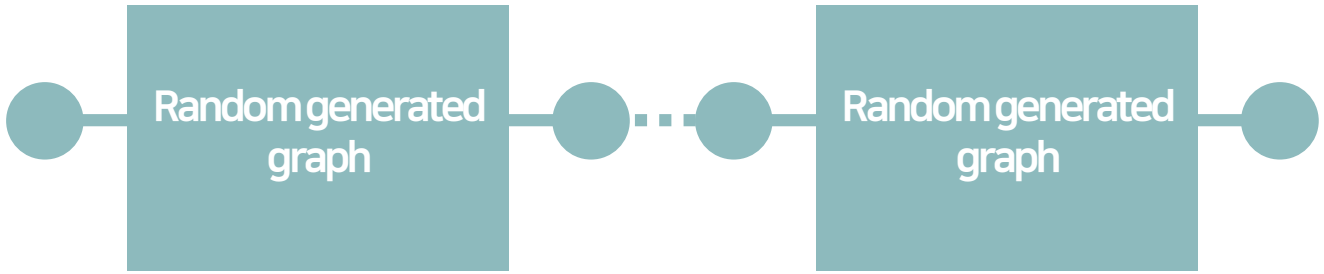

Input Extra Node : 모든 파란 노드에 동일한 값 send

Output Extra Node : 값들을 다 average해서 get

# 02. METHOD

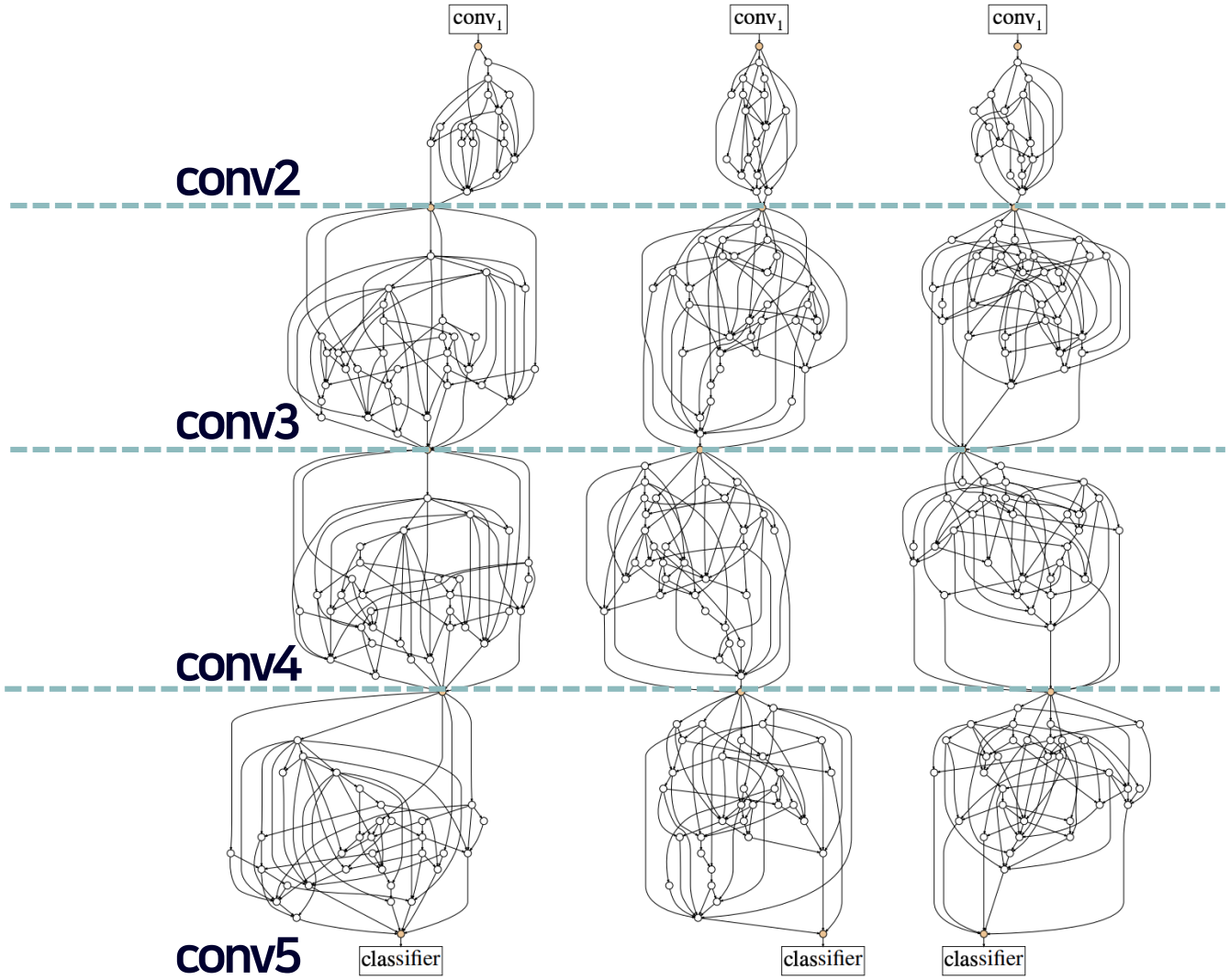
## Mapping - Stage

이미지 분류에서 해상도 유지는 좋지 않다
-> feature map을 점진적으로 축소하는 stage로 나누는게 일반적



# 02. METHOD

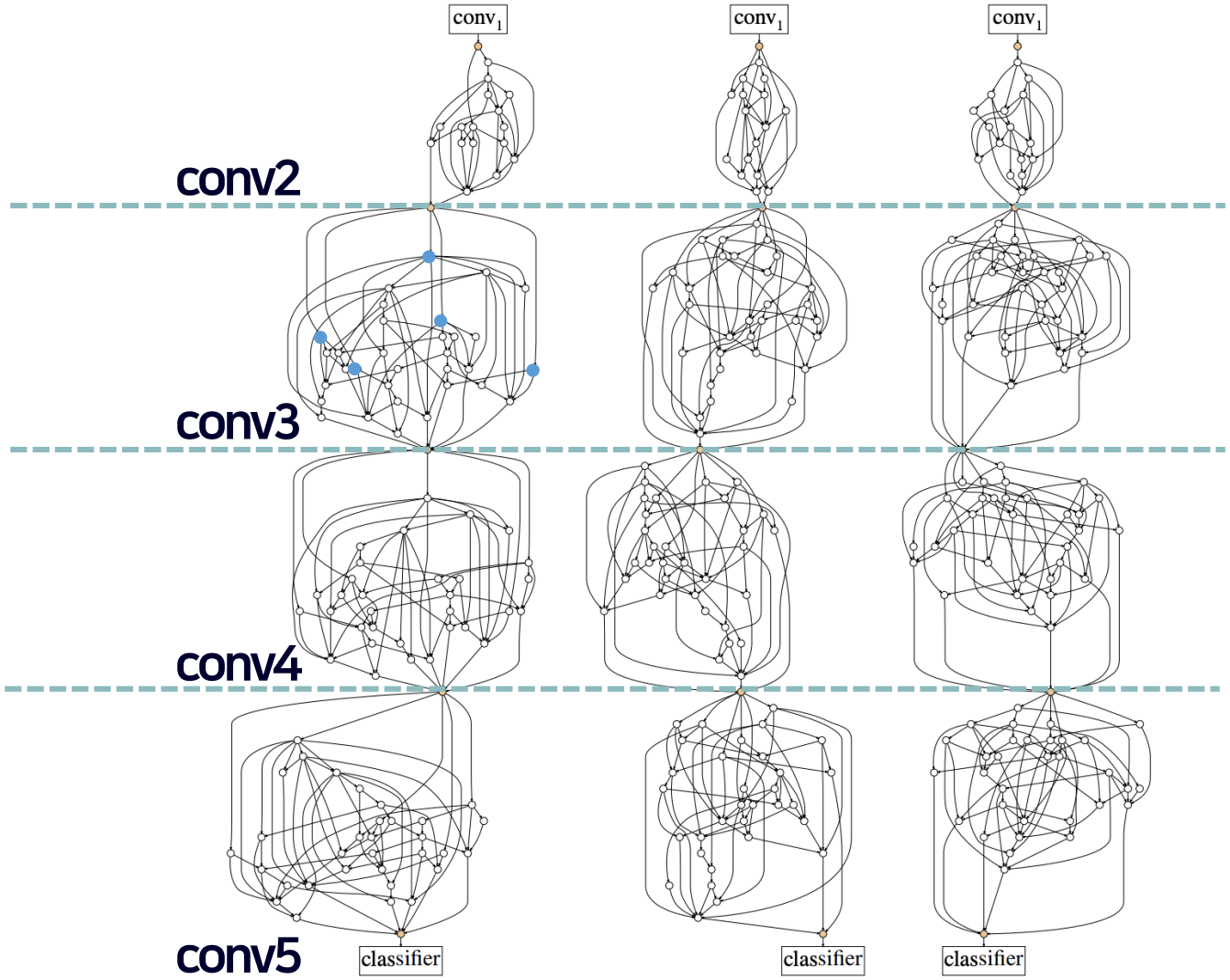
## Mapping - Stage



# 02. METHOD

## Mapping - Stage

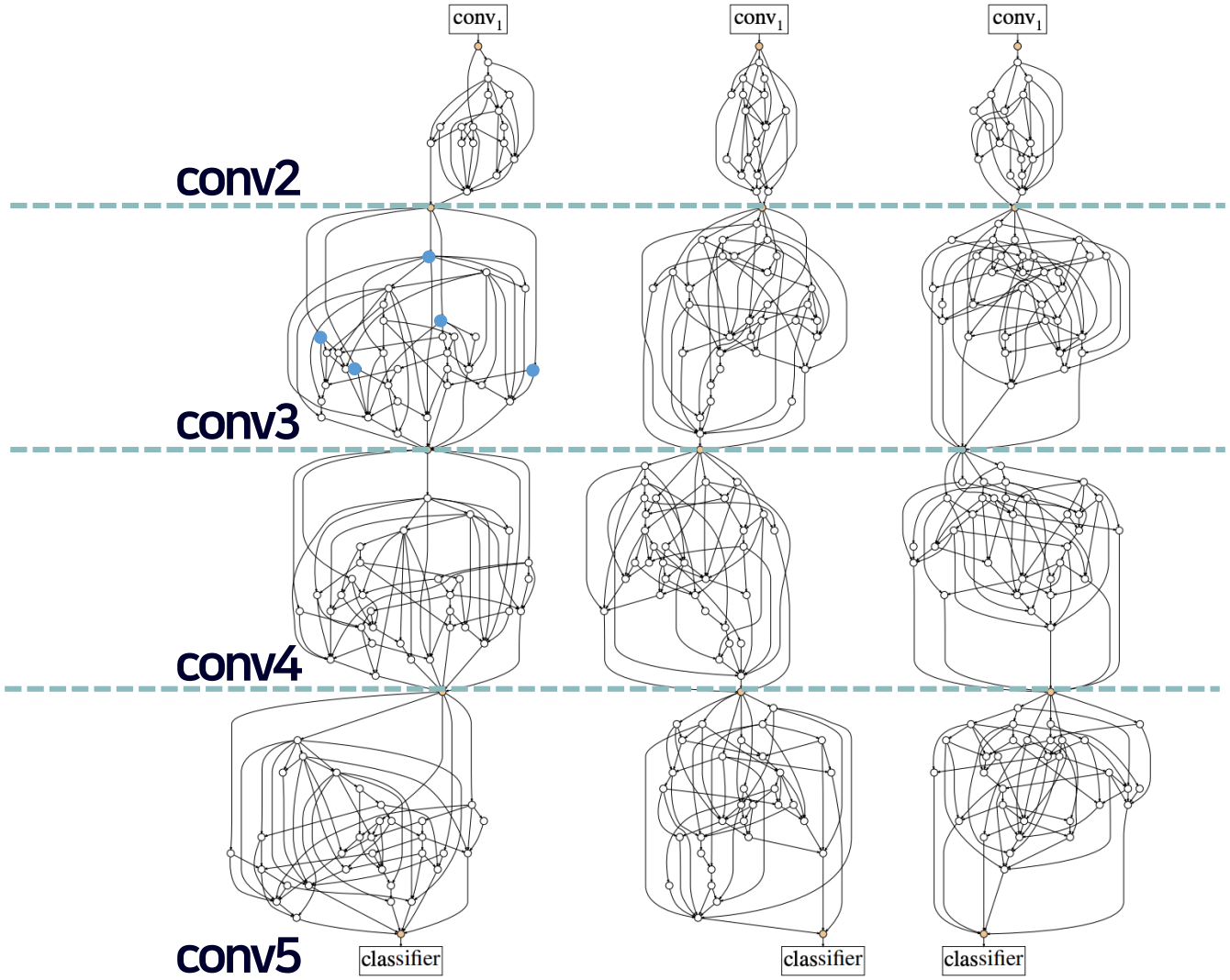
- Stride2 conv



# 02. METHOD

## Mapping - Stage

- Stride2 conv



# 02. METHOD

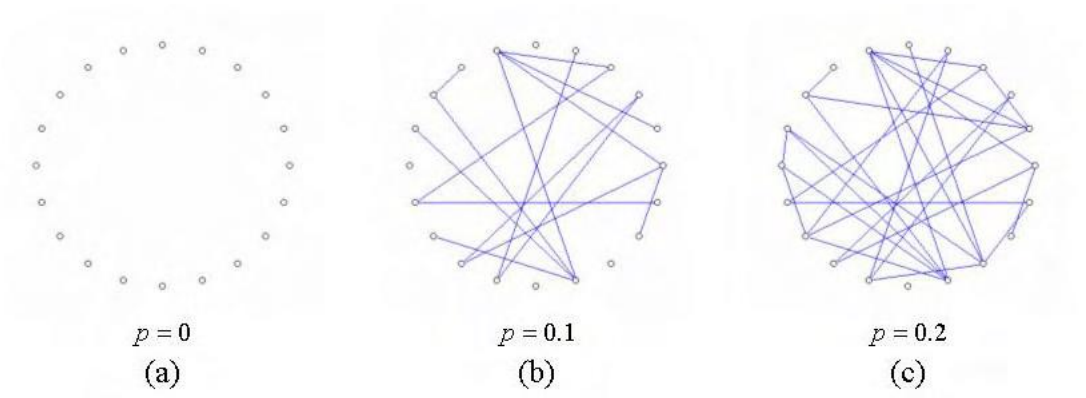
## Random Graph Models

# 02. METHOD

## Random Graph Models

### Erdős–Rényi (ER)

* ER(P)
* N개의 노드를 가진 그래프를 **뭐든지** 만들 수 있다
* 어떤 그래프도 만들어질 확률이 0이 아니다

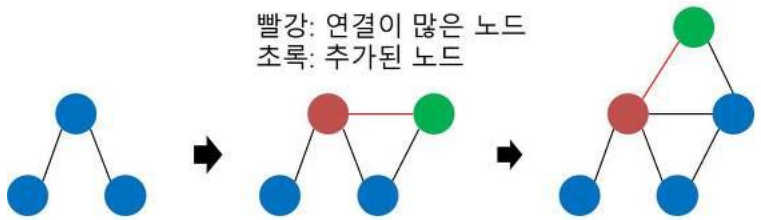


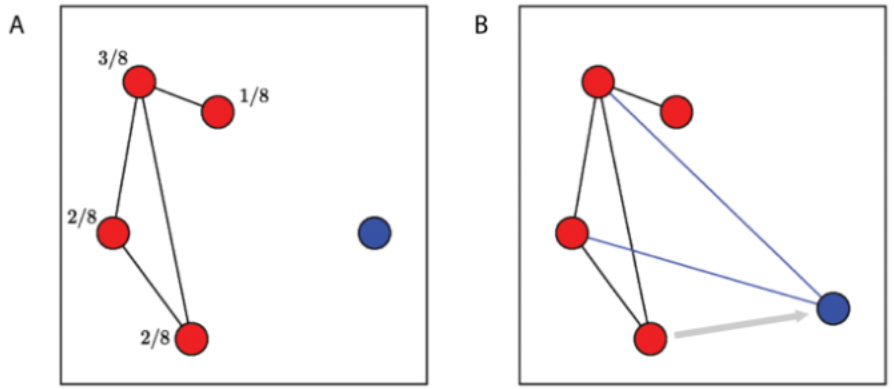
# 02. METHOD

## Random Graph Models

### Barabási-Albert (BA)

* BA(M)
* Edge = M(N-M)개
* 모든 N노드 그래프의 부분집합



# 02. METHOD

## Random Graph Models

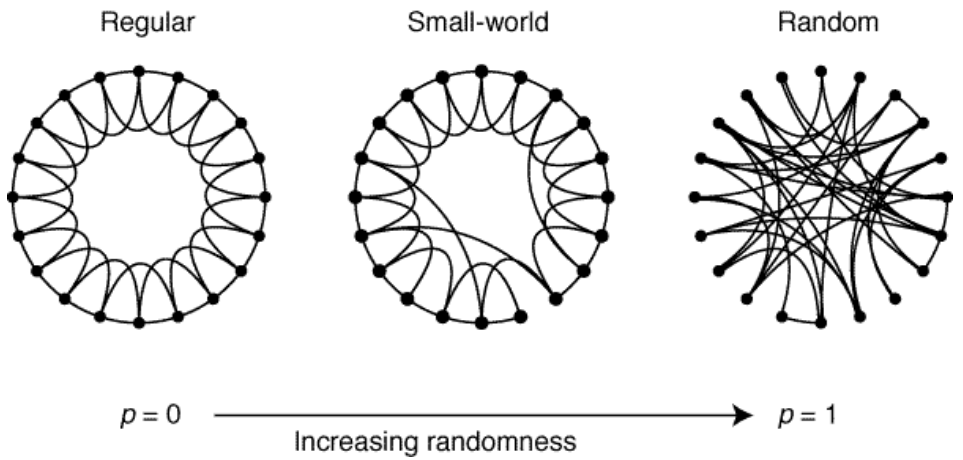
### Watts–Strogatz (WS)

# 02. METHOD

## Random Graph Models

### Watts-Strogatz (WS)

* WS(K,P)
* Edge : N*K
* N노드 그래프의 부분집합



# 02. METHOD

## Random Graph Models

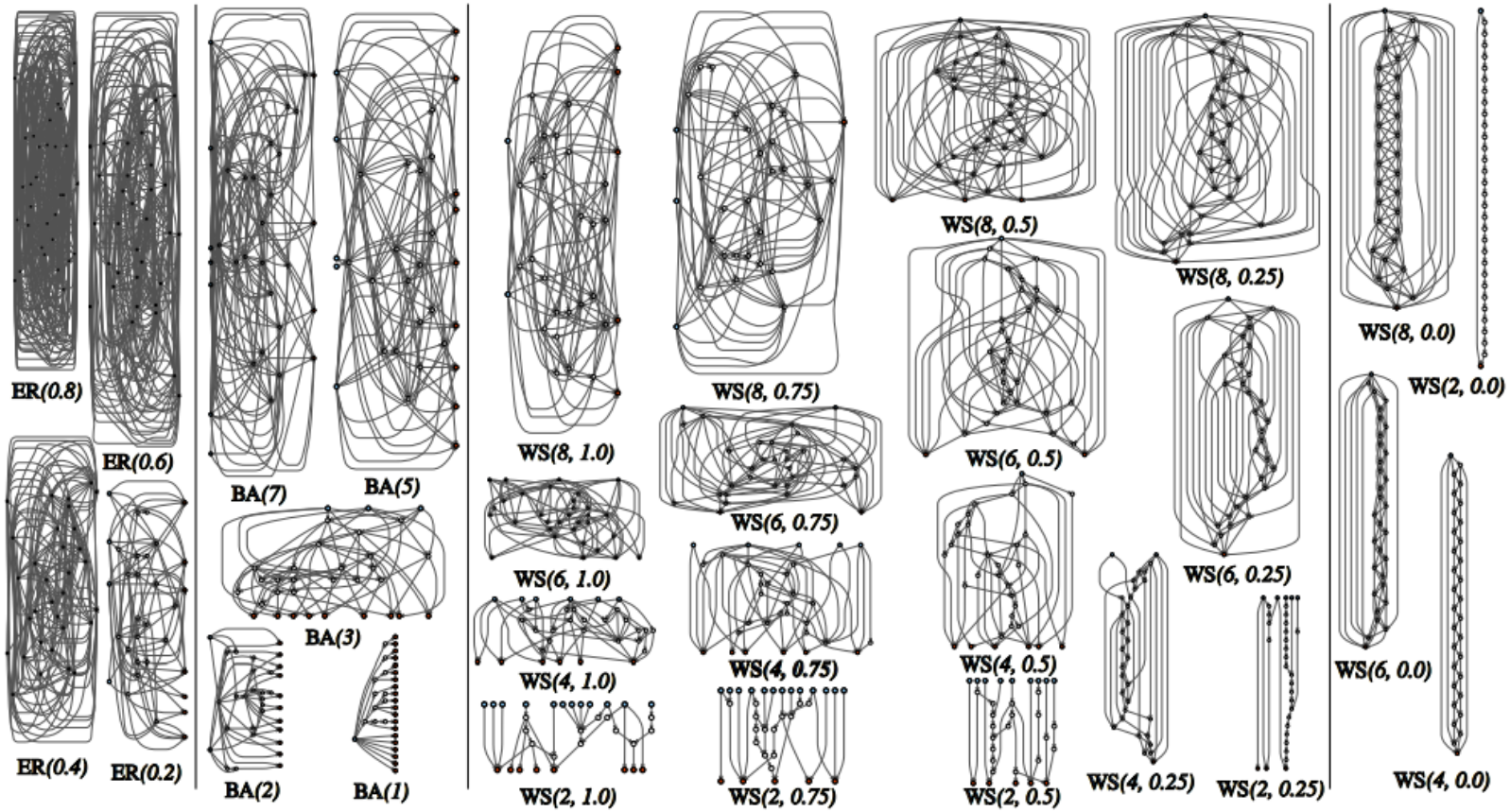



Figure 4. **Visualization of the random graphs generated by ER, BA, and WS**. Each plot represents one random graph instance sampled by the specified generator. The generators are those in Figure 3. The node count is $N$=32 for each graph. A blue/red node denotes an input/output node, to which an extra unique input/output node (not shown) will be added (see §3.2).

ER(P) BA(M) WS(K,P) Regular

# 02. METHOD

## Design and Optimization

$$g(\theta, s) \qquad \theta : P, M, (K, P) \in ER, BA, WS$$

# Experiments

# 03. EXPERIMENTS

## ImageNet 1000 classification

* 5개의 랜덤 seed를 가지고 5개의 네트워크를 사용
* **평균+-std** 값으로 분류 정확도 측정
* 모든 실험에 동일한 seed 사용

| stage | output | *small regime* | *regular regime* |
|---|---|---|---|
| $\text{conv}_1$ | 112×112 | 3×3 conv, $C/2$ | |
| $\text{conv}_2$ | 56×56 | 3×3 conv, $C$ | ***random wiring*** $N/2, C$ |
| $\text{conv}_3$ | 28×28 | ***random wiring*** $N, C$ | ***random wiring*** $N, 2C$ |
| $\text{conv}_4$ | 14×14 | ***random wiring*** $N, 2C$ | ***random wiring*** $N, 4C$ |
| $\text{conv}_5$ | 7×7 | ***random wiring*** $N, 4C$ | ***random wiring*** $N, 8C$ |
| classifier | 1×1 | 1×1 conv, 1280-d<br>global average pool, 1000-d *fc*, softmax | |

# 03. EXPERIMENTS

## Random graph generators

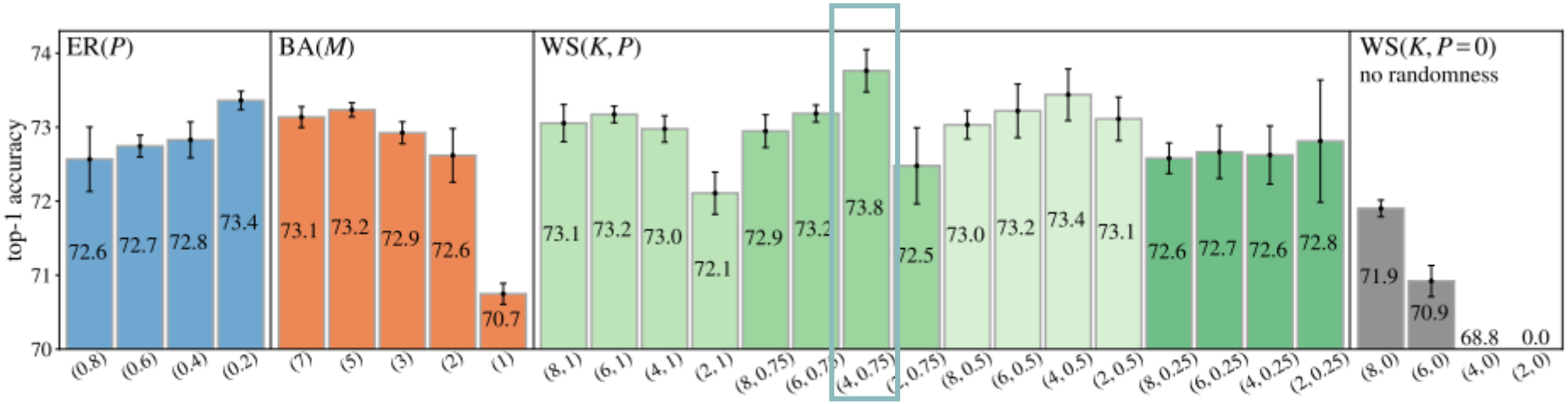



Figure 3. **Comparison on random graph generators: ER, BA, and WS** in the small computation regime. Each bar represents the results of a generator under a parameter setting for $P$, $M$, or $(K, P)$ (tagged in x-axis). The results are ImageNet top-1 accuracy, shown as mean and standard deviation (std) over 5 random network instances sampled by a generator. At the rightmost, WS$(K, P{=}0)$ has no randomness.

# 03. EXPERIMENTS

## Graph Damage

**Random하게 하나의 node/Edge를 삭제**
**추가적인 학습 없이 정확도를 측정**

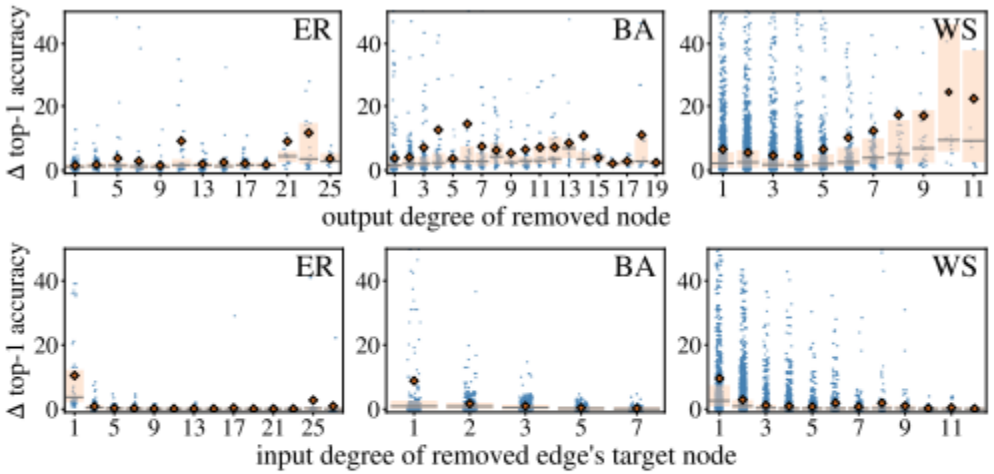



Figure 5. **Graph damage ablation**. We randomly ***remove one node*** (top) or ***remove one edge*** (bottom) from a graph after the network is trained, and evaluate the loss (Δ) in accuracy on ImageNet. From left to right are ER, BA, and WS generators. Red circle: *mean*; gray bar: *median*; orange box: *interquartile range*; blue dot: *an individual damaged instance*.

# 03. EXPERIMENTS

## Node operation

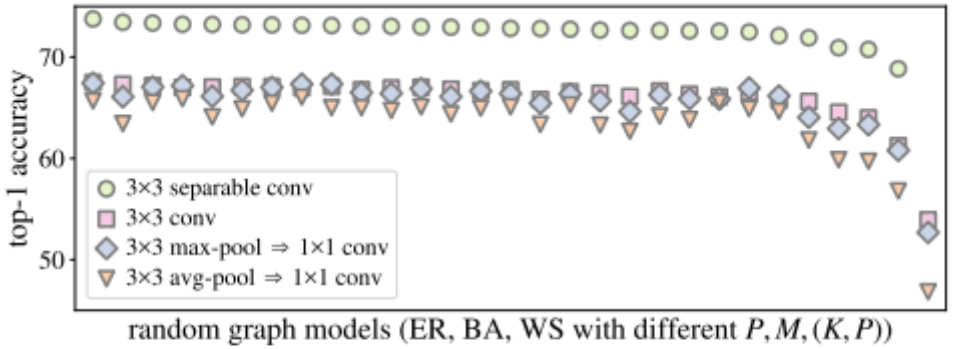



Figure 6. **Alternative node operations**. Each column is the mean accuracy of the same set of 5 random graphs equipped with different node operations, sorted by "3×3 separable conv" (from Figure 3). The generators roughly maintain their orders of accuracy.

# 03. EXPERIMENTS

## 진짜 결과

| network | top-1 acc. | top-5 acc. | FLOPs (M) | params (M) |
|---|---|---|---|---|
| MobileNet [15] | 70.6 | 89.5 | 569 | 4.2 |
| MobileNet v2 [40] | 74.7 | - | 585 | 6.9 |
| ShuffleNet [54] | 73.7 | 91.5 | 524 | 5.4 |
| ShuffleNet v2 [30] | **74.9** | 92.2 | 591 | 7.4 |
| NASNet-A [56] | 74.0 | 91.6 | 564 | 5.3 |
| NASNet-B [56] | 72.8 | 91.3 | 488 | 5.3 |
| NASNet-C [56] | 72.5 | 91.0 | 558 | 4.9 |
| Amoeba-A [34] | 74.5 | 92.0 | 555 | 5.1 |
| Amoeba-B [34] | 74.0 | 91.5 | 555 | 5.3 |
| Amoeba-C [34] | **75.7** | **92.4** | 570 | 6.4 |
| PNAS [26] | 74.2 | 91.9 | 588 | 5.1 |
| DARTS [27] | 73.1 | 91.0 | 595 | 4.9 |
| **RandWire-WS** | **74.7**$_{\pm 0.25}$ | **92.2**$_{\pm 0.15}$ | 583$_{\pm 6.2}$ | 5.6$_{\pm 0.1}$ |

Table 2. **ImageNet: small computation regime** (*i.e.*, <600M FLOPs). RandWire results are the mean accuracy (±std) of 5 random network instances, with WS$(4, 0.75)$. Here we train for 250 epochs similar to [56, 34, 26, 27], for fair comparisons.

| network | top-1 acc. | top-5 acc. | FLOPs (B) | params (M) |
|---|---|---|---|---|
| ResNet-50 [11] | 77.1 | 93.5 | 4.1 | 25.6 |
| ResNeXt-50 [52] | 78.4 | 94.0 | 4.2 | 25.0 |
| **RandWire-WS**, $C$=109 | **79.0**$_{\pm 0.17}$ | **94.4**$_{\pm 0.11}$ | 4.0$_{\pm 0.09}$ | 31.9$_{\pm 0.66}$ |
| ResNet-101 [11] | 78.8 | 94.4 | 7.8 | 44.6 |
| ResNeXt-101 [52] | 79.5 | 94.6 | 8.0 | 44.2 |
| **RandWire-WS**, $C$=154 | **80.1**$_{\pm 0.19}$ | **94.8**$_{\pm 0.18}$ | 7.9$_{\pm 0.18}$ | 61.5$_{\pm 1.32}$ |

Table 3. **ImageNet: regular computation regime** with FLOPs comparable to ResNet-50 (top) and to ResNet-101 (bottom). ResNeXt is the 32×4 version [52]. RandWire is WS$(4, 0.75)$.

# 03. EXPERIMENTS

## 진짜 결과

| 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 | $81.6_{\pm 0.13}$ | $95.6_{\pm 0.07}$ | $16.0_{\pm 0.36}$ | $61.5_{\pm 1.32}$ |

Table 4. **ImageNet: large computation regime.** Our networks are the same as in Table 3 ($C$=154), but we evaluate on 320×320 images instead of 224×224. Ours are only trained for 100 epochs.

| backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| ResNet-50 [11] | 37.1 | 58.8 | 39.7 | 21.9 | 40.8 | 47.6 |
| ResNeXt-50 [52] | 38.2 | 60.5 | 41.3 | 23.0 | 41.5 | 48.8 |
| **RandWire-WS**, $C$=109 | **39.9** | **61.9** | **43.3** | **23.6** | **43.5** | **52.7** |
| ResNet-101 [11] | 39.8 | 61.7 | 43.3 | 23.7 | 43.9 | 51.7 |
| ResNeXt-101 [52] | 40.7 | 62.9 | 44.5 | 24.4 | 44.8 | 52.7 |
| **RandWire-WS**, $C$=154 | **41.1** | **63.1** | **44.6** | **24.6** | **45.1** | **53.0** |

Table 5. **COCO object detection** results fine-tuned from the networks in Table 3, reported on the `val2017` set. The backbone networks have comparable FLOPs to ResNet-50 or ResNet-101.

# 04. CONCLUSION

## 의의

3개의 고전적인 랜덤 그래프 모델로 만들어지는 랜덤 유선 nn을 탐구했다
결과적으로 평균 정확도가 최근의 뉴럴 아키텍쳐 연구에 근접했다
네트워크 생성기는 노벨성을 가지고 있다~