

The slide features decorative geometric shapes in the corners. The top-left corner has several overlapping triangles in shades of blue, green, and red. The bottom-right corner has a cluster of overlapping triangles in various shades of gray.


# Involution

**Involution: Inverting the Inherence of Convolution for Visual Recognition(<https://arxiv.org/abs/2103.06255>)**

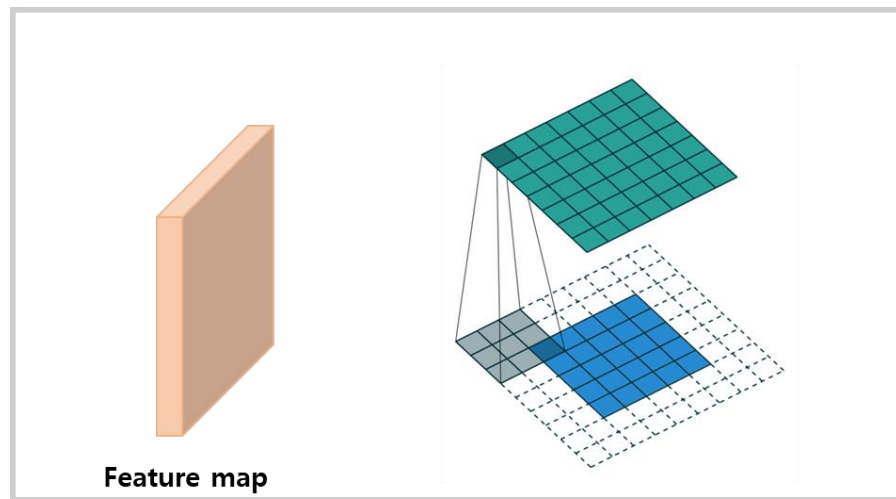
한성대학교 1971336 김태민

A decorative graphic in the top-left corner consisting of several overlapping, semi-transparent geometric shapes in shades of blue, green, and red, resembling a stylized flower or star.

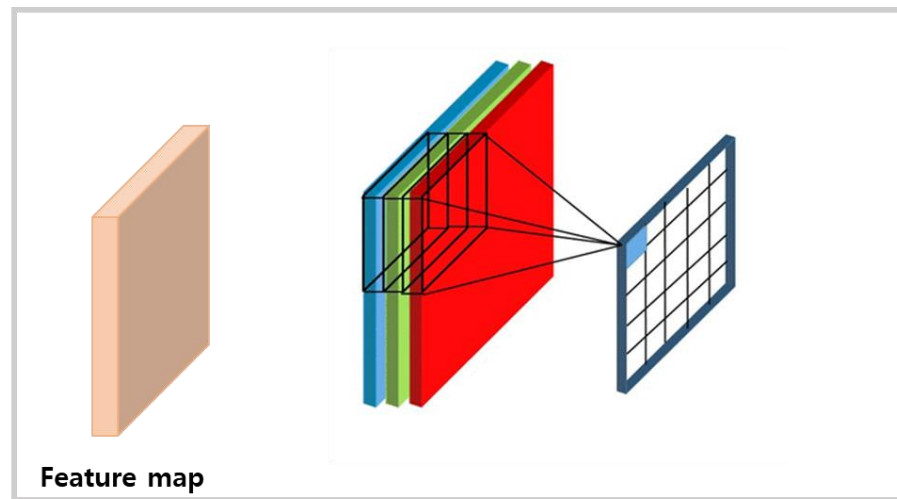
# 목차

- **Convolution**
  - **Problem of Convolution**
  - **Involution Architecture**
  - **Understanding of Involution**
  - **Experiments**
  - **Code**
  - **Involution GAN**
- 
- A decorative graphic in the bottom-left corner consisting of several overlapping, semi-transparent geometric shapes in shades of blue, green, and red, resembling a stylized flower or star.

# Convolution



Spatial-agnostic



Channel-specific

- **Spatial-agnostic**
  - 같은 kernel을 통한 Conv Operation을 모든 위치에서 동일하게 수행한다.
- **Channel-specific**
  - Conv Operation을 수행 시 각각의 다른 채널들의 값을 종합하여 Feature map을 만들어 줌
  - 각 Filter 당 한 개의 새로운 Channel을 만들어 준다.

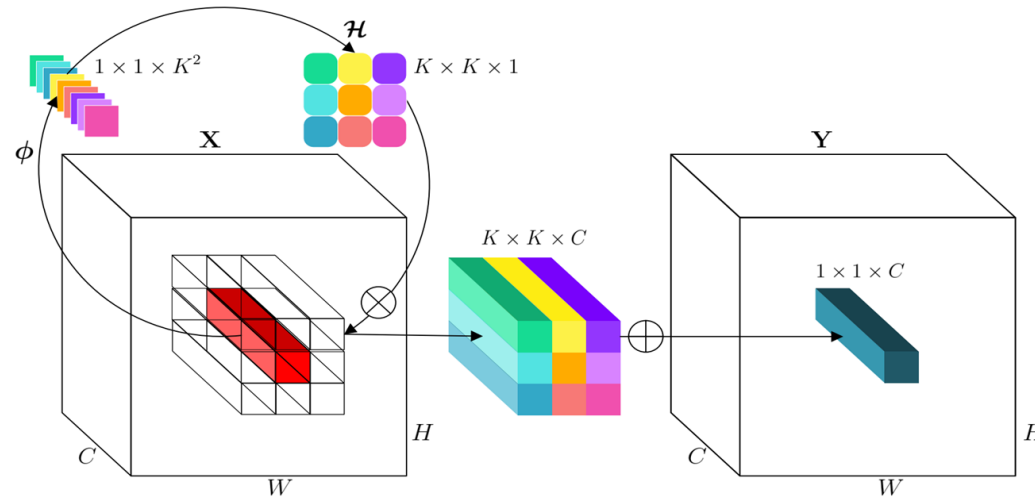
# Problem of Convolution



Input

- Conv Problem으로 여러 연구들이 Spatial Agnostic하지만 Local한 부분만 계산하고 Stride에 따라 움직이므로 모든 면적에 대해 single shot으로는 넓은 범위의 spatial관계를 알기 어렵다고 한다.
- 
- 즉 이미지에서 Conv 수행 시 region 사이의 관계를 알기 위해서 여러 layer의 Conv를 이용 하거나 비정상적으로 Kernel Size를 늘려야 한다는 단점이 있다.
- 또한 하나의 kernel은 하나의 channel을 만드는데 channel이 이전 layer의 정보를 종합해서 만드는데 이러한 과정이 반복되면 Kernel들의 중복성이 높아진다고 한다. 즉 하나의 kernel이 하나의 channel을 만드는 과정에서 중요한 정보와 유사한 kernel만 학습이 될 수가 있어 예시로 커널을 경량화 하는 channel pruning를 진행해도 성능이 많이 떨어지지않는다.

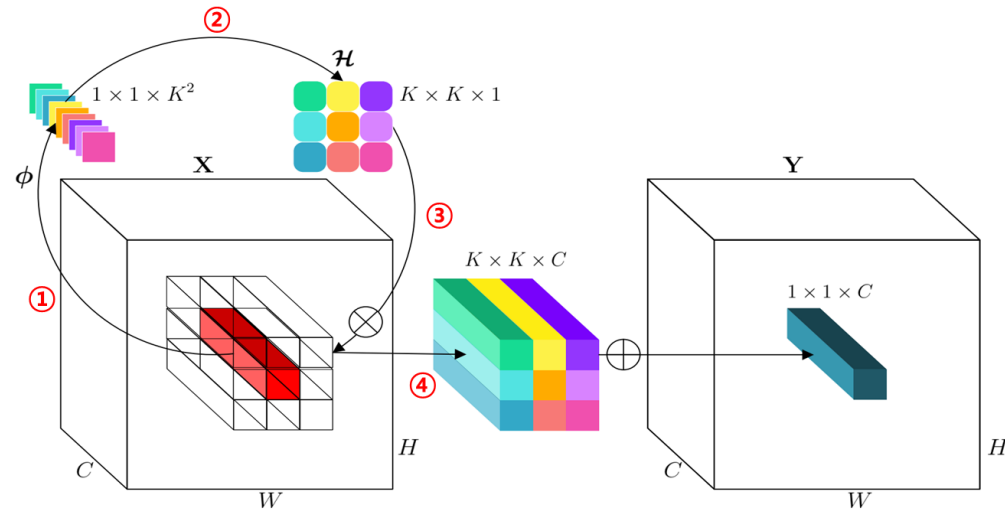
# Involution Architecture



Involution Architecture

- Conv의 특성을 반대로 Spatial-Agnostic & Channel-Specific  $\rightarrow$  Spatial-Specific & Channel-Agnostic
- Spatial-Specific
- 모든 위치에서(각 픽셀) 각각의 다른 Filter를 적용하면 어떨까?
- Channel-Agnostic
- 기존의 Conv는 이미지의 모든 채널에 대해 1개의 필터(채널 수)로 N개를 한번에 종합하였지만
- 모든 채널에 대해 같은 필터로 Operation을 해주면 어떨까?

# Involution Architecture



Involution Architecture

- Spatial-Specific
- 공간 위치에 대해 다른 커널이 생성
- Channel-Agnostic
- 각 커널은 채널을 통해 공유

- 1. 입력 받을 한 개의 픽셀을 추출한다.(1x1xN이용)
- 1.2 다시 한번 1x1xN Conv를 사용한다. Filter의 개수는 매개변수 ( $\text{kernel\_size} * \text{kernel\_size} * \text{group\_number}$ )
- 2.  $K * K * 1$ 로 reshape 변환
- 3. 처음 입력 받은  $k * k * 1$ 의 크기의 patch로 받아온다.
- 커널은  $K * K * C$  ( $C = \text{image channel}$ )로 브로드캐스트
- 4. kerne과 패치를 element-Wise Mutiplication
- 5. reduce sum  $K * K$ 축을 기준으로 reduce\_sum

1. 256x256x3(이미지) -> 256x256x3(특성맵)
- 1.2 256x256x3(특성맵) -> 256x256x9(특성맵)
2. 256x256x9(특성맵) -> 256x256x9(3x3)x1(특성맵)
3. 256x256x9(3x3)x3(패치) (Pixel\_h,Pixel\_w,size,channel)  
256x256x9(3x3)x1(특성맵) -> 256x256x9(3x3)x3 (특성맵)
4. 패치 \*(element-Wise M) 커널 = 256x256x9(3x3)x3
5. output = 256x256x3

# Understanding of Involution

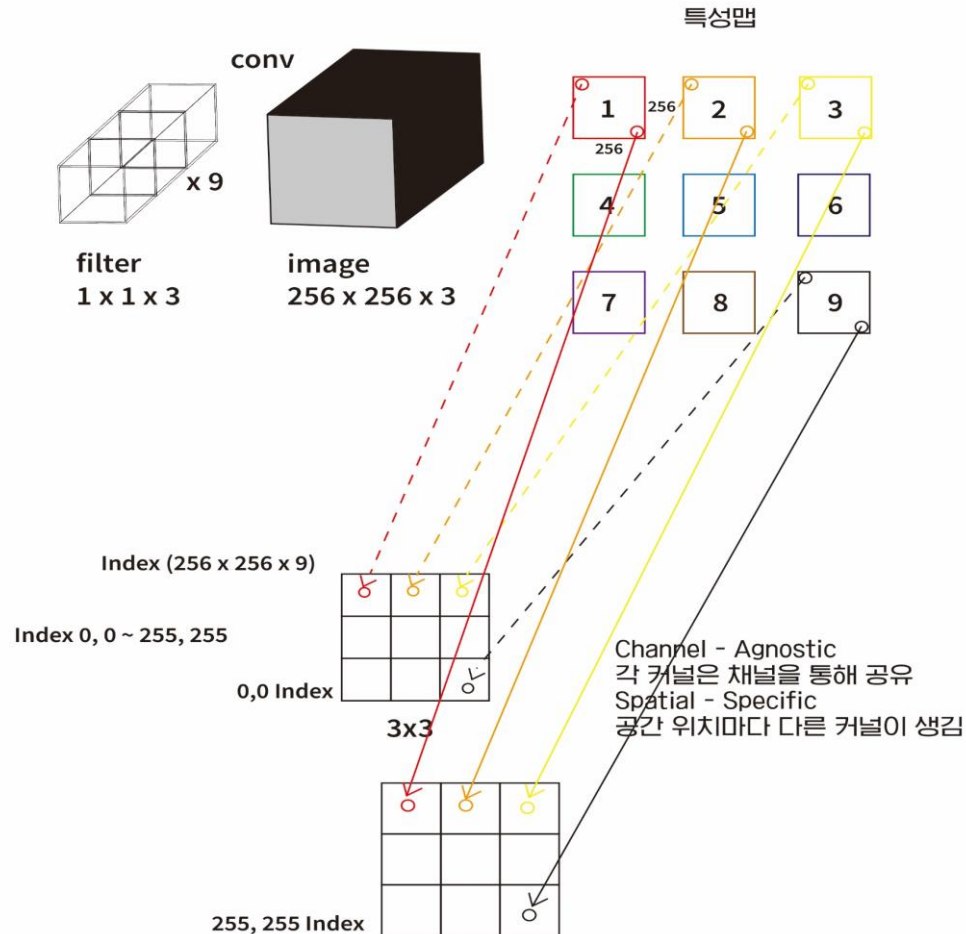


그림 filter에서 2번째 filter의 값이 변하면? (filter 20이라는 가중치 공유)  
모든 픽셀(Index)의 2번째 값이 변하고, 그럼 결국 각 픽셀의 커널은 filter의 채널에 공유  
(0, 0 ~ 255, 255)  
각각의 픽셀 9개의 값은 서로 다른값  
(0,0 ~ 255, 255)

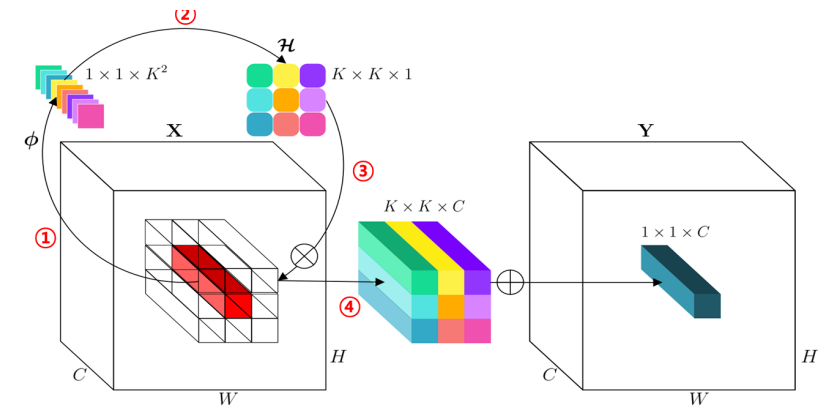
후에 패치랑 픽셀마다의 커널로 연산

- Spatial-Specific
- 공간 위치에 대해 다른 커널이 생성
- Channel-Agnostic
- 각 커널은 채널을 통해 공유

**Algorithm 1** Pseudo code of involution in a PyTorch-like style.

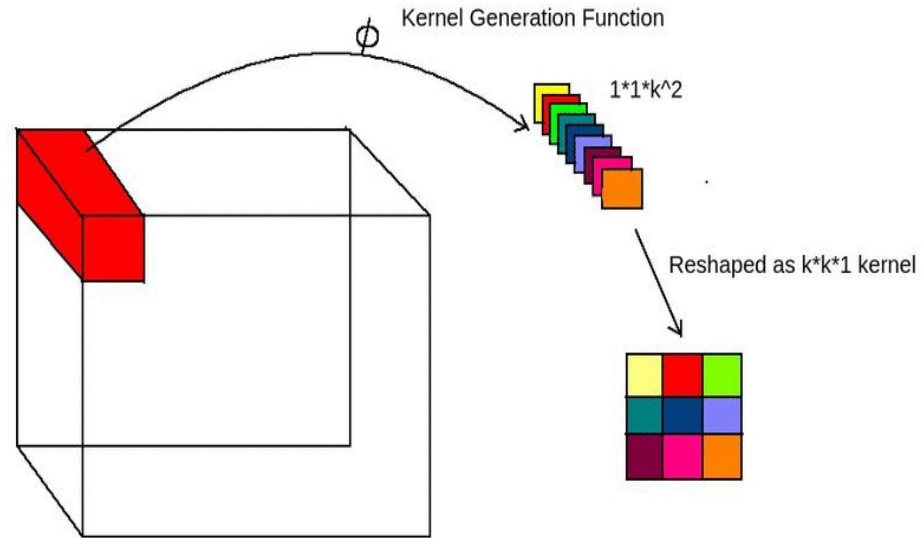
```
# B: batch size, H: height, W: width
# C: channel number, G: group number
# K: kernel size, s: stride, r: reduction ratio

##### initialization #####
o = nn.AvgPool2d(s, s) if s > 1 else nn.Identity()
reduce = nn.Conv2d(C, C//r, 1)
span = nn.Conv2d(C//r, K*K*G, 1)
unfold = nn.Unfold(K, dilation, padding, s)
##### forward pass #####
x_unfolded = unfold(x) # B,CxKxK,HxW
x_unfolded = x_unfolded.view(B, G, C//G, K*K, H, W)
# kernel generation, Eqn. (6)
kernel = span(reduce(o(x))) # B,KxKxG,H,W
kernel = kernel.view(B, G, K*K, H, W).unsqueeze(2)
# Multiply-Add operation, Eqn. (4)
out = mul(kernel, x_unfolded).sum(dim=3) # B,G,C/G,H,W
out = out.view(B, C, H, W)
return out
```



Involution Architecture

# Understanding of Involution



## <kernel Generation Function: 2-layered MLP>

the kernel generation function  $\phi: \mathbb{R}^C \mapsto \mathbb{R}^{K \times K \times G}$  with  $\Psi_{i,j} = \{(i,j)\}$  taking the following form:

$$\mathcal{H}_{i,j} = \phi(\mathbf{X}_{i,j}) = \mathbf{W}_1 \sigma(\mathbf{W}_0 \mathbf{X}_{i,j}). \quad (6)$$

$$\mathbf{W}_0 \in \mathbb{R}^{\frac{C}{r} \times C} \text{ and } \mathbf{W}_1 \in \mathbb{R}^{(K \times K \times G) \times \frac{C}{r}}$$

- r: dimension reduction
- G: the # of groups which share the involution kernels

- 인풋은 C차원의 벡터를 받고
- $\mathbf{X}(i,j)$ 는 C차원의 실수 벡터
- Linear Transformation -> non Linear activation(batchnorm,Relu)-> Linear Transformation->
- 결과값을 커널로 사용



# Experiments

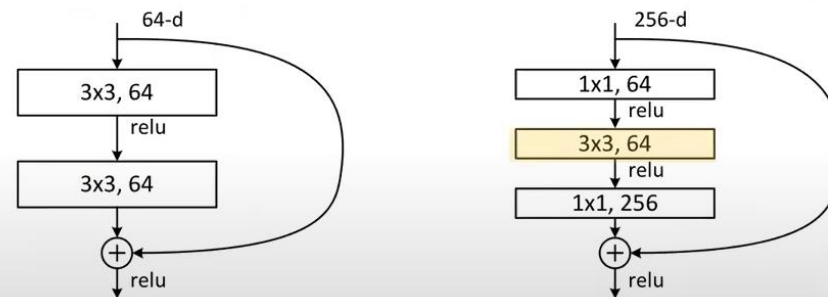


Figure 5. A deeper residual function  $\mathcal{F}$  for ImageNet. Left: a building block (on  $56 \times 56$  feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

- 3x3 Conv은 이제 3x3 Inv 또는 7x7 Inv로 대체 한다.
- 후에 1x1 Conv가 있기 때문에 Inv가 채널방향으로 섞이지 않더라도 1x1 Conv가 다시 채널방향으로 섞어준다.
- An Involution followed by 1x1 convolutions assumes
- feature extraction is conducted with channel-spatial, spatial-alone, and channel-alone interactions for the stream of information propagation

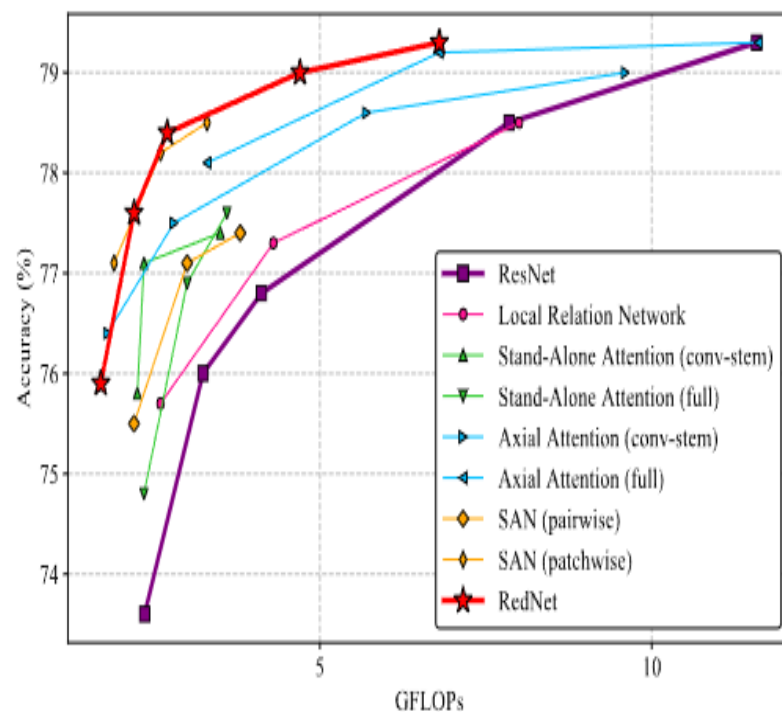
# Experiments

## ImageNet-1K

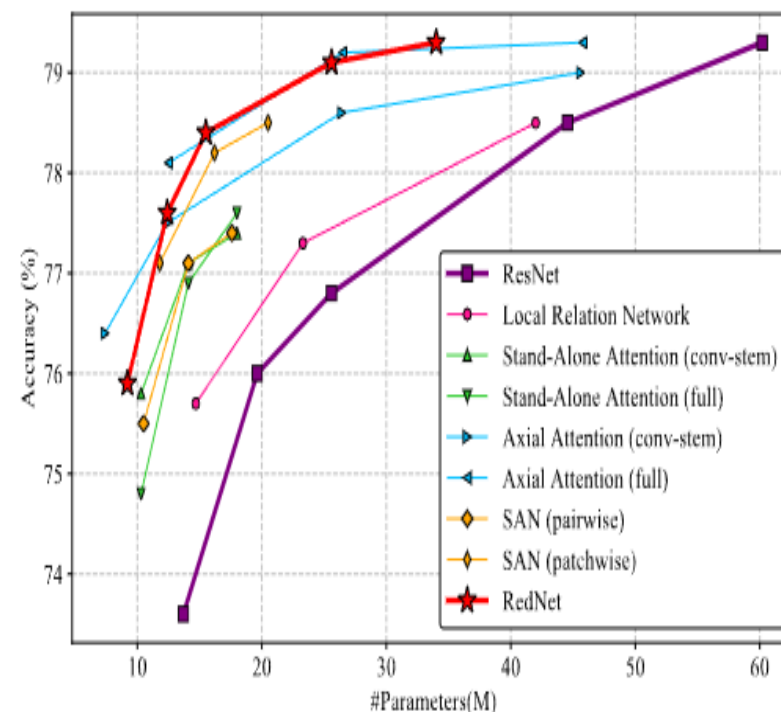
Architecture	#Params (M)	FLOPs (G)	Top-1 Acc. (%)
ResNet-26 [18]	13.7	2.4	73.6
LR-Net-26 [20]	14.7	2.6	75.7
Stand-Alone ResNet-26 [39]	10.3	2.4	74.8
SAN10 <sup>†</sup> [64]	10.5	2.2	75.5
<b>RedNet-26</b>	<b>9.2</b>	<b>1.7</b>	<b>75.9</b>
ResNet-38 [18]	19.6	3.2	76.0
Stand-Alone ResNet-38 [39]	14.1	3.0	76.9
SAN15 <sup>†</sup> [64]	14.1	3.0	77.1
<b>RedNet-38</b>	<b>12.4</b>	<b>2.2</b>	<b>77.6</b>
ResNet-50 [18]	25.6	4.1	76.8
LR-Net-50 [20]	23.3	4.3	77.3
AA-ResNet-50 [2]	25.8	4.2	77.7
Stand-Alone ResNet-50 [39]	18.0	3.6	77.6
SAN19 <sup>†</sup> [64]	17.6	3.8	77.4
Axial ResNet-S <sup>‡</sup> [50]	<b>12.5</b>	3.3	78.1
<b>RedNet-50</b>	<b>15.5</b>	<b>2.7</b>	<b>78.4</b>
ResNet-101 [18]	44.6	7.9	78.5
LR-Net-101 [20]	42.0	8.0	78.5
AA-ResNet-101 [2]	45.4	8.1	78.7
<b>RedNet-101</b>	<b>25.6</b>	<b>4.7</b>	<b>79.1</b>
ResNet-152 [18]	60.2	11.6	<b>79.3</b>
AA-ResNet-152 [2]	61.6	11.9	79.1
Axial ResNet-M <sup>‡</sup> [50]	<b>26.5</b>	<b>6.8</b>	79.2
Axial ResNet-L <sup>‡</sup> [50]	45.8	11.6	<b>79.3</b>
<b>RedNet-152</b>	<b>34.0</b>	<b>6.8</b>	<b>79.3</b>

# Experiments

## ImageNet-1K



(a) The accuracy-complexity envelope on ImageNet.



(b) The accuracy-parameter envelope on ImageNet.

# Experiments

## Detector

Detector	Backbone	Neck	#Params (M)	FLOPs (G)	AP <sup>bbbox</sup>	AP <sup>bbbox</sup> <sub>50</sub>	AP <sup>bbbox</sup> <sub>75</sub>	AP <sup>bbbox</sup> <sub>S</sub>	AP <sup>bbbox</sup> <sub>M</sub>	AP <sup>bbbox</sup> <sub>L</sub>
RetinaNet [30]	ResNet-50	convolution	37.7	239.3	36.6	55.8	39.2	20.9	40.6	47.5
	RedNet-50	convolution	27.8	210.1	38.3 (+1.7)	58.2 (+2.4)	40.5 (+1.3)	21.1 (+0.2)	41.8 (+1.2)	50.9 (+3.4)
	RedNet-50	involution	26.3	199.9	38.2 (+1.6)	58.2 (+2.4)	40.4 (+1.2)	21.8 (+0.9)	41.6 (+1.0)	50.7 (+3.2)

Detector	Backbone	Neck	Head	#Params (M)	FLOPs (G)	AP <sup>bbbox</sup>	AP <sup>bbbox</sup> <sub>50</sub>	AP <sup>bbbox</sup> <sub>75</sub>	AP <sup>bbbox</sup> <sub>S</sub>	AP <sup>bbbox</sup> <sub>M</sub>	AP <sup>bbbox</sup> <sub>L</sub>
Faster R-CNN [40]	ResNet-50	convolution	convolution	41.5	207.1	37.7	58.7	40.8	21.7	41.6	48.4
	RedNet-50	convolution	convolution	31.6	177.9	39.5 (+1.8)	60.9 (+2.2)	42.8 (+2.0)	23.3 (+1.6)	42.9 (+1.3)	52.2 (+3.8)
	RedNet-50	involution	convolution	29.5	135.0	40.2 (+2.5)	62.1 (+3.4)	43.4 (+2.6)	24.2 (+2.5)	43.3 (+1.7)	52.7 (+4.3)
	RedNet-50	involution	involution	29.0	91.5	39.2 (+1.5)	61.0 (+2.3)	42.4 (+1.6)	23.1 (+1.4)	43.0 (+1.4)	50.7 (+2.3)

Detector	Backbone	Neck	Head	#Params (M)	FLOPs (G)	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
Mask R-CNN [17]	ResNet-50	convolution	convolution	44.2	253.4	38.4	59.2	41.9	21.9	42.3	49.7
						35.1	56.3	37.3	18.5	38.6	46.9
	RedNet-50	convolution	convolution	34.2	224.2	40.2 (+1.8)	61.4 (+2.2)	43.7 (+1.8)	24.2 (+2.3)	43.4 (+1.1)	52.5 (+2.8)
						36.1 (+1.0)	58.1 (+1.8)	38.2 (+0.9)	19.9 (+1.4)	39.3 (+0.7)	48.9 (+2.0)
	RedNet-50	involution	convolution	32.2	181.3	40.8 (+2.4)	62.3 (+3.1)	44.3 (+2.4)	24.2 (+2.3)	44.0 (+1.7)	53.0 (+3.3)
						36.4 (+1.3)	59.0 (+2.7)	38.5 (+1.2)	19.9 (+1.4)	39.4 (+0.8)	49.1 (+2.2)
	RedNet-50	involution	involution	29.5	104.6	39.6 (+1.2)	60.7 (+1.5)	42.7 (+0.8)	23.5 (+1.6)	43.1 (+0.8)	51.1 (+1.4)
						35.1 (+0.0)	57.1 (+0.8)	37.3 (+0.0)	19.2 (+0.7)	38.5 (−0.1)	47.3 (+0.4)

Table 3: Performance comparison on COCO detection and segmentation. The bounding box AP is reported for the object detection track in the upper table. The bounding box and mask AP are simultaneously reported for the instance segmentation track in the lower table, listed in the two separate lines following a single detector. In the parentheses are the gaps to the fully convolution-based counterparts. Highlighted in green are the gaps of at least +2.0 points, the same in Table 4 and 5.

# Experiments

Involution Kernel(sum of  $K \times K$  kernels as its representative value)

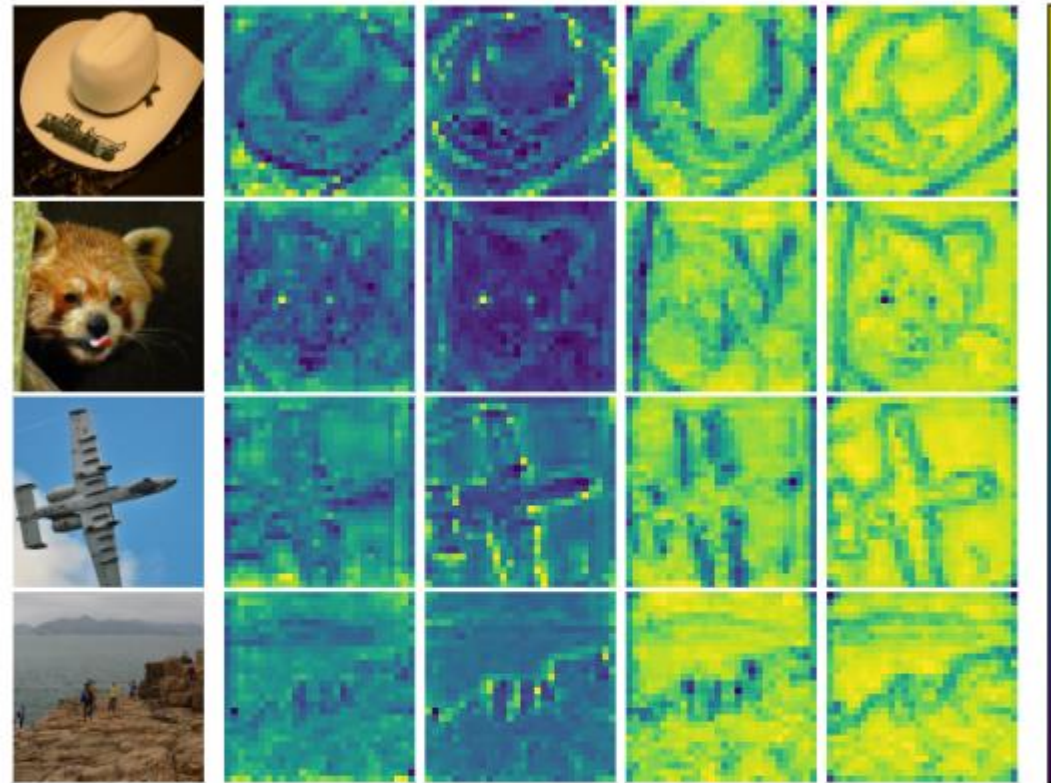


Figure 3: The heat maps in each row interpret the generated kernels for an image instance from the ImageNet validation set, drawn from four different classes, including cowboy hat, lesser panda, warplane, and cliff (from top to bottom).

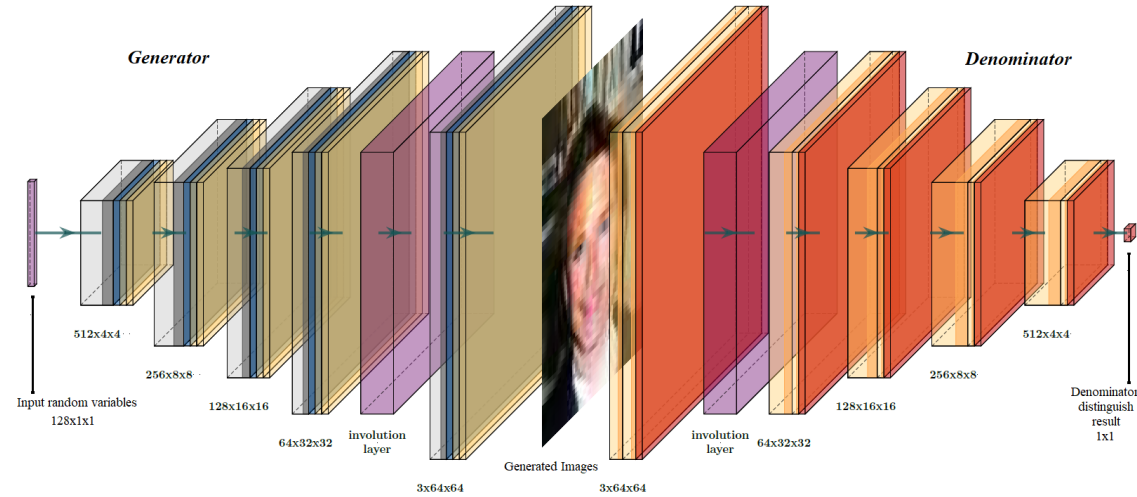


# Code





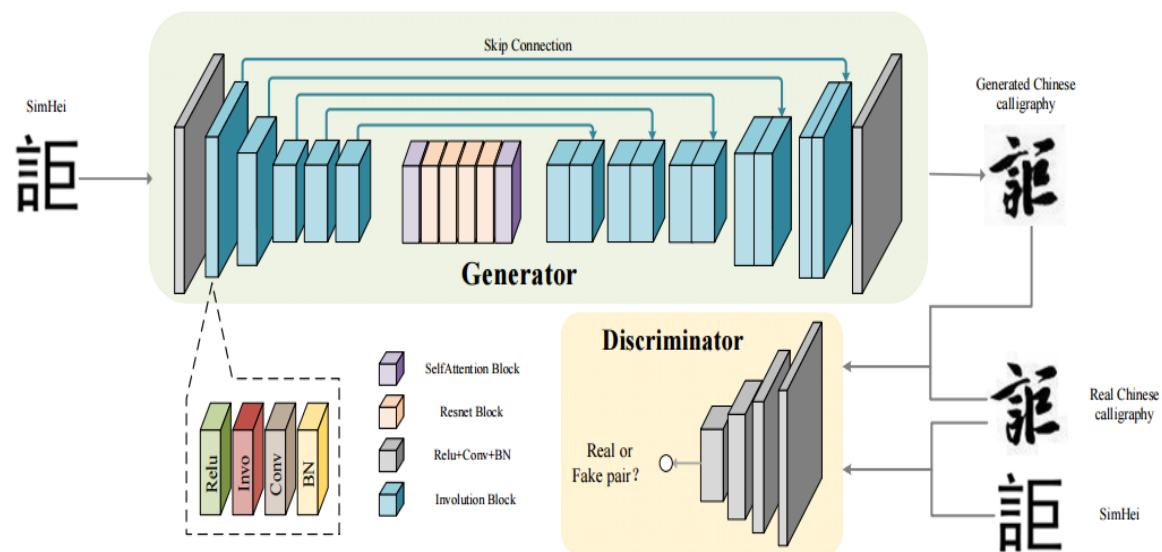
# Involution GAN



iGAN의 이미지는 SAGAN 및 DCGAN보다 채도가 높고 색상이 더 부드럽다.  
(<https://github.com/XiaozhuFang/InvolutionGAN>)

# Involution GAN

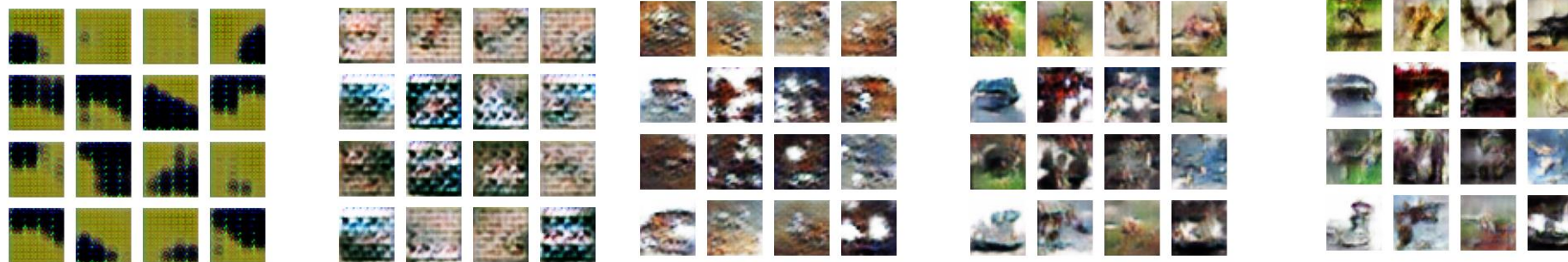
## Chinese Calligraphy Generation Based on Involution





# Involution GAN

Cifar10 DCGAN(0,10,20,30,40)



Cifar10 IDCGAN(0,3,6,9,12)

