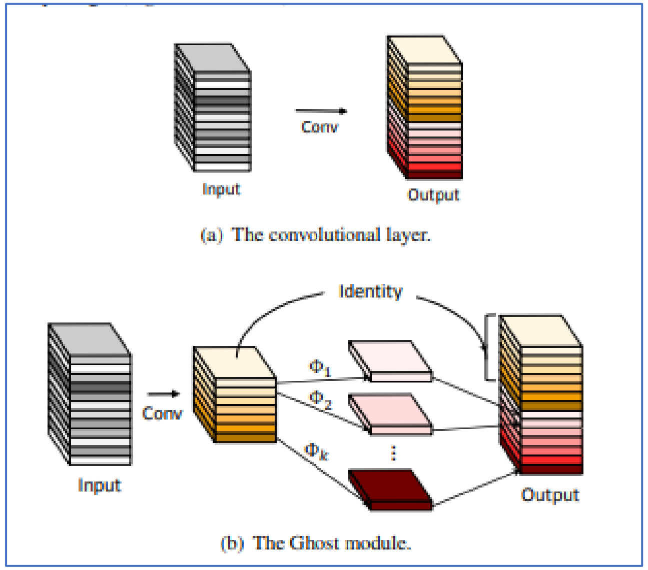
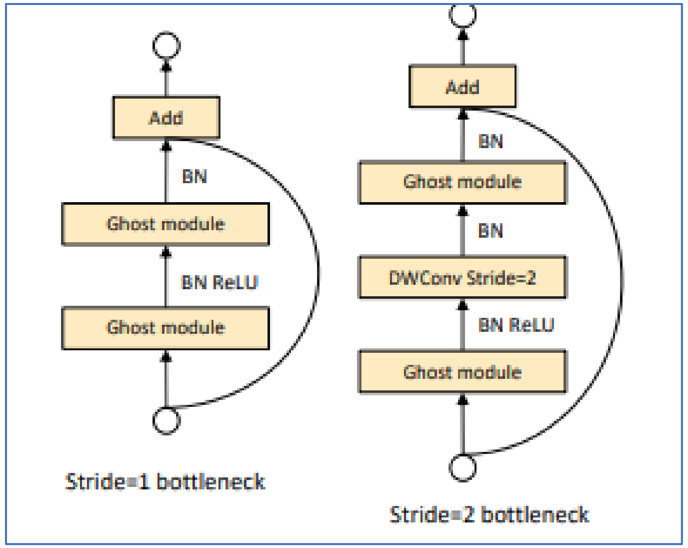
GhostNet: More Features from Cheap Operations

- 핵심내용 : 적은 연산으로 중복되는 feature-map들을 많이 생성해내는 Ghost-Module.
- 목적 : 휴대 가능한 모바일 기기에 네트워크 크기는 작지만 성능이 좋은 아키텍처를 넣기 위함.
- 경향 : 최근에 소형의 딥 뉴럴 네트워크가 제안됨 ex) pruning, knowledge distillation 등.
  마찬가지로 파라미터와 계산량을 줄이고 퍼포먼스를 늘린 mobilnet 등이 출시.
- 제안 : GhostNet.

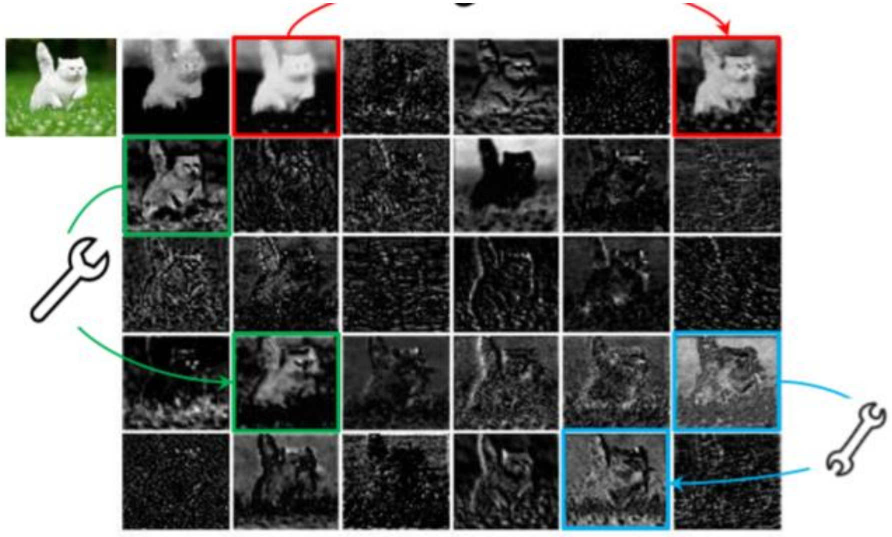
GhostModule



Ghost bottlenecks

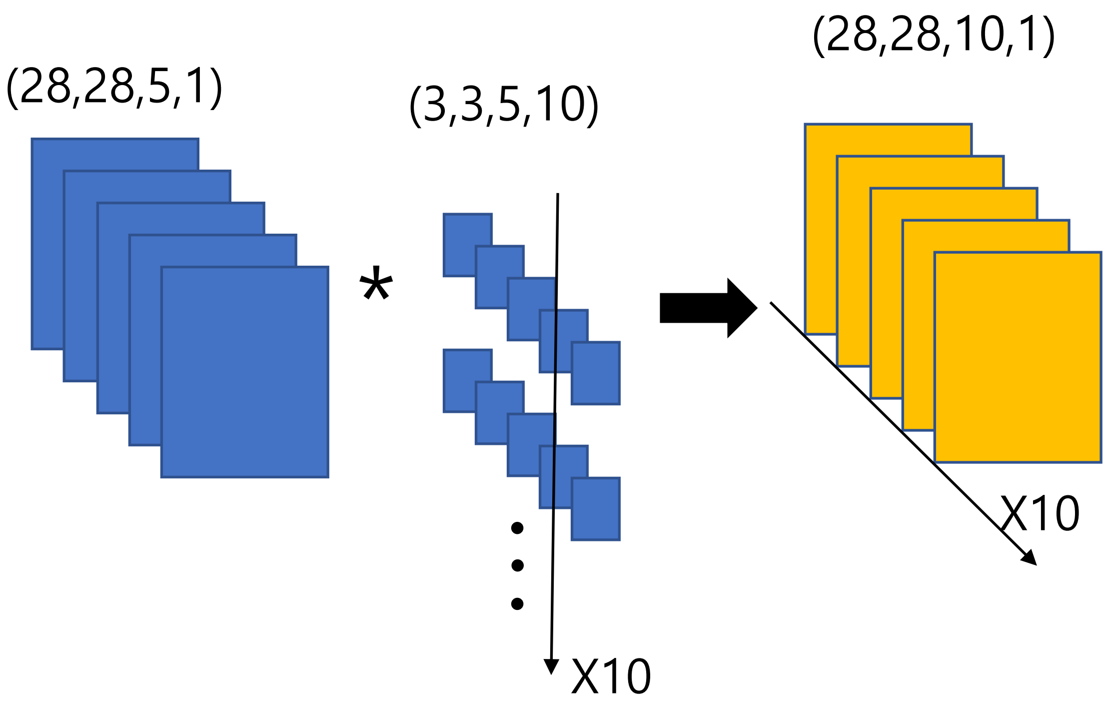


first residual group feature-map in ResNet50



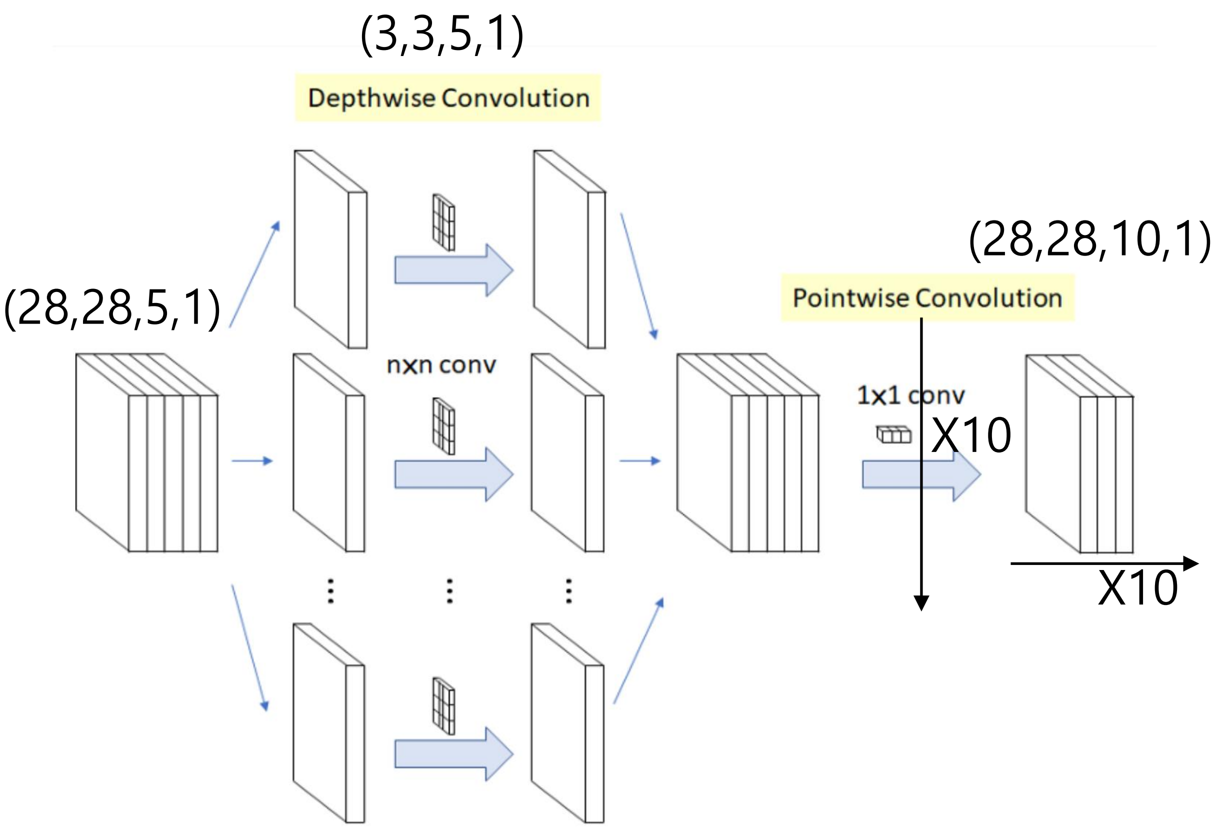
MobileNet conv 연산방법 변화

Original conv



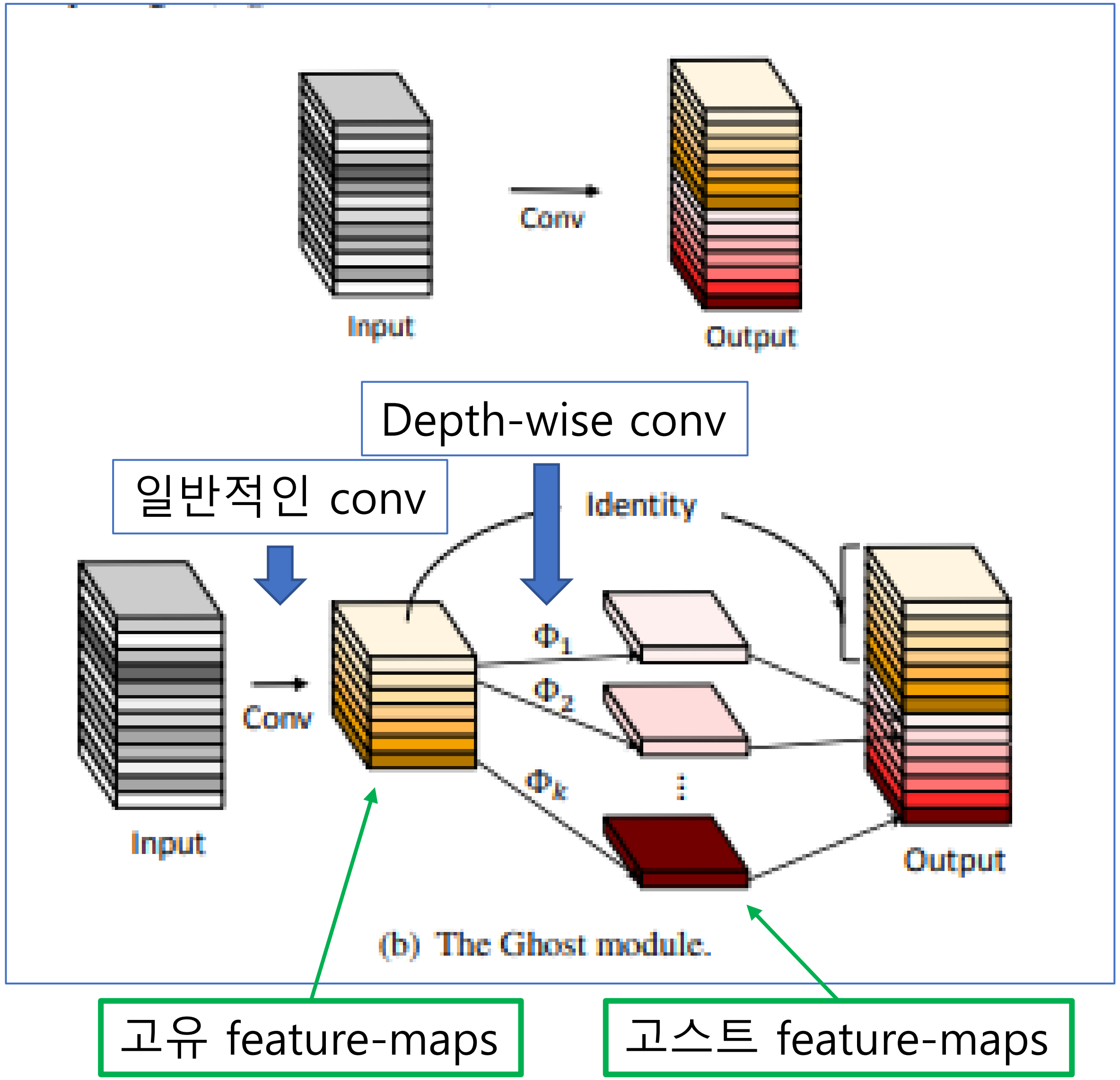
parameters : 3x3x5x10 = 450
FLOPs : 28x28x3x3x5x10 = 405,000

Depthwise separable conv in MobileNet



parameters : 3x3x1x5 + 1x1x5x10 = 95
FLOPs : 28x28x3x3x5 + 28x28x5x10 = 85,500

Ghost Module

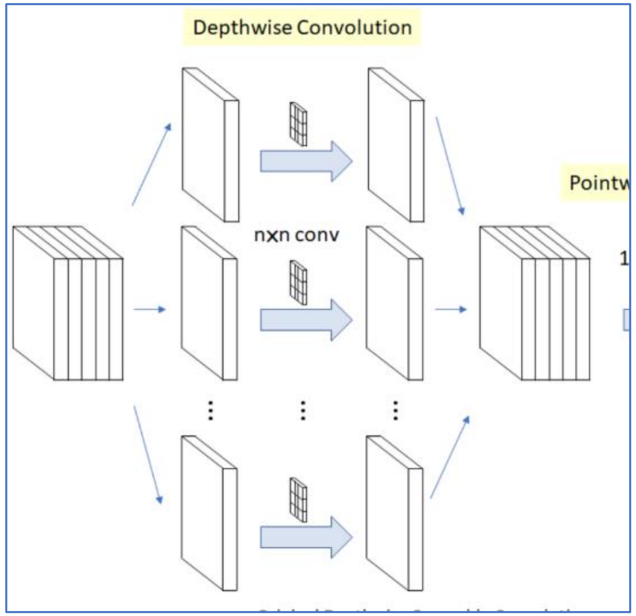


(b) The Ghost module.

일반적인 Conv 과정을 두 파트에 나누어서 진행.

1. 일반적인 conv
2. Depth-wise conv

```python
def forward(self, x):
    x1 = self.primary_conv(x)
    x2 = self.cheap_operation(x1)
    out = torch.cat([x1,x2], dim=1)
    return out[:,:self.oup,:,:]
```



Ghost Module

input = (28,28,5,1)
output =10
k=1, padding=1, stride=1
m=2, s=5, s-1=4

1. 일반적인 conv

$$Y' = X * f',$$

$$Y' \in \mathbb{R}^{h' \times w' \times m}$$

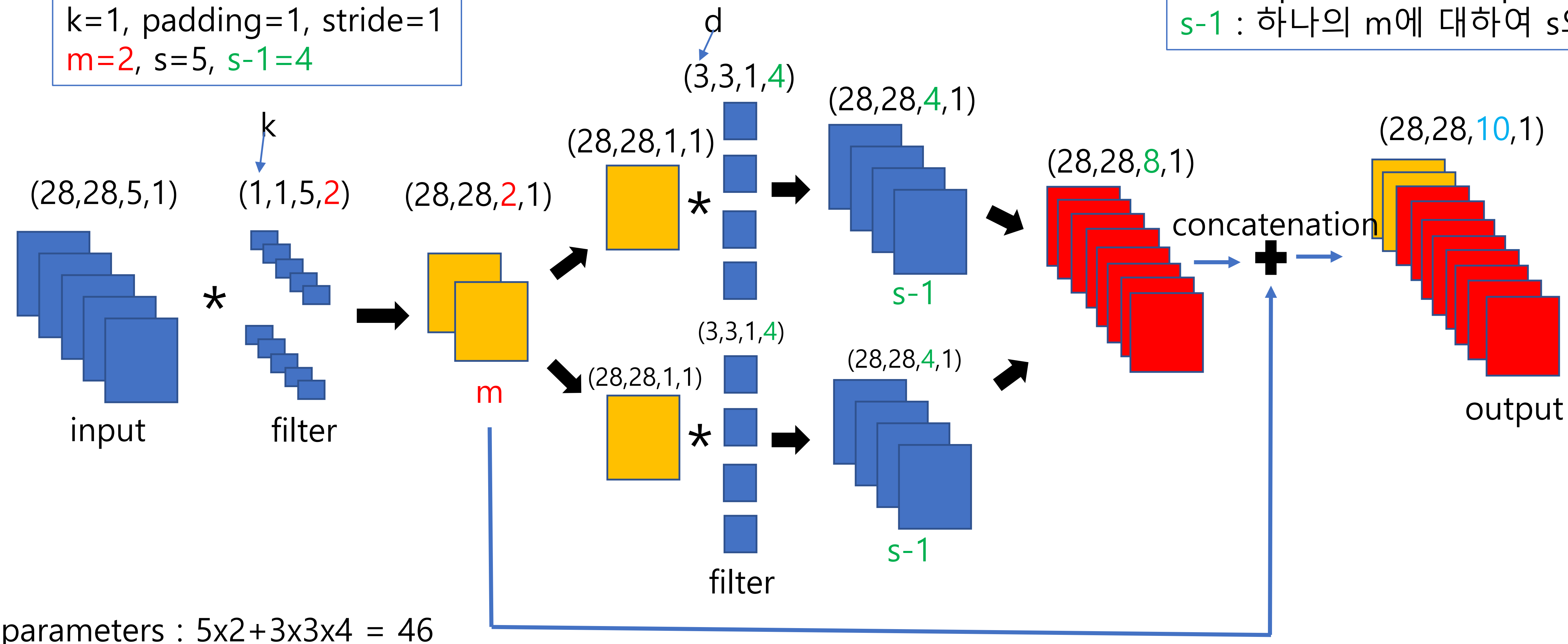
$$f' \in \mathbb{R}^{c \times k \times k \times m}$$

2. depth wise conv

$$y_{ij} = \Phi_{i,j}(y'_i), \quad \forall\, i = 1, ..., m, \quad j = 1, ..., s,$$

$$\boxed{n = m \cdot s}$$

m : 고유한 feature-maps
s : ghost feature-maps
n : output feature-maps 채널 수
s-1 : 하나의 m에 대하여 s의 수



parameters : 5x2+3x3x4 = 46
FLOPs : 28x28x5x2+28x28x3x3x4 = 41,400

Ghost Module

```python
class GhostModule(nn.Module):
    def __init__(self, inp, oup, kernel_size=3, ratio=2, dw_size=3, stride=1, relu=True):
        super(GhostModule, self).__init__()
        self.oup = oup
        init_channels = math.ceil(oup / ratio)
        new_channels = init_channels*(ratio-1)

        self.primary_conv = nn.Sequential(
            nn.Conv2d(inp, init_channels, kernel_size, stride, kernel_size//2, bias=False),
            nn.BatchNorm2d(init_channels),
            nn.ReLU(inplace=True) if relu else nn.Sequential(),
        )

        self.cheap_operation = nn.Sequential(
            nn.Conv2d(init_channels, new_channels, dw_size, 1, dw_size//2, groups=init_channels, bias=False),
            nn.BatchNorm2d(new_channels),
            nn.ReLU(inplace=True) if relu else nn.Sequential(),
        )

    def forward(self, x):
        x1 = self.primary_conv(x)
        x2 = self.cheap_operation(x1)
        out = torch.cat([x1,x2], dim=1)
        return out[:,:self.oup,:,:]

a = torch.randn(1,3,28,28)
net = GhostModule(inp=3,oup=200)
```

다른 모델들과의 차별점

1. 연산량을 줄이기 위한 point-wise Conv 사용하는 모델들과 다르게 kernel_size 조정가능

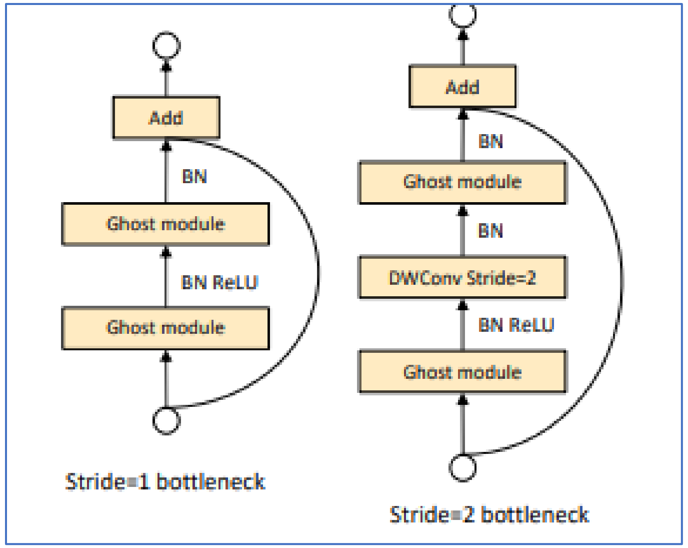
2. 기존 conv의 고유한 피처맵을 이용하여 피처맵을 증가시켰다.

3. 고유한 피처맵을 보존함.

4. 기존의 depth-wise 연산이 제한된 부분이 있었는데 linear operations 는 다양성이 있다.

GhostNet

: Conv → GhonstModule 변경



보통 2개의 ghost module 사용.
첫번째 모듈은 채널의 확장.
두번째 모듈은 shortcut path를 맞추기 위해 축소.

Input	Operator	#exp	#out	SE	Stride
$224^2 \times 3$	Conv2d 3×3	-	16	-	2
$112^2 \times 16$	G-bneck	16	16	-	1
$112^2 \times 16$	G-bneck	48	24	-	2
$56^2 \times 24$	G-bneck	72	24	-	1
$56^2 \times 24$	G-bneck	72	40	1	2
$28^2 \times 40$	G-bneck	120	40	1	1
$28^2 \times 40$	G-bneck	240	80	-	2
$14^2 \times 80$	G-bneck	200	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	480	112	1	1
$14^2 \times 112$	G-bneck	672	112	1	1
$14^2 \times 112$	G-bneck	672	160	1	2
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	Conv2d 1×1	-	960	-	1
$7^2 \times 960$	AvgPool 7×7	-	-	-	-
$1^2 \times 960$	Conv2d 1×1	-	1280	-	1
$1^2 \times 1280$	FC	-	1000	-	-

mobileNetV3의 bottleneck 구조에서
Block을 ghost module로 바꾸기만함.

실험 & 결과

Table 3. The performance of the proposed Ghost module with different d on CIFAR-10.

d	Weights (M)	FLOPs (M)	Acc. (%)
VGG-16	15.0	313	93.6
1	7.6	157	93.5
3	7.7	158	93.7
5	7.7	160	93.4
7	7.7	163	93.1

Kernel size를 바꿔가면서 진행.
3x3일때 공간적인 특징들을 잘 잡는다고 여겨짐.

Table 4. The performance of the proposed Ghost module with different s on CIFAR-10.

s	Weights (M)	FLOPs (M)	Acc. (%)
VGG-16	15.0	313	93.6
2	7.7	158	93.7
3	5.2	107	93.4
4	4.0	80	93.0
5	3.3	65	92.9

고스트 feature-map을 더 많이 생산할수록
네트워크는 가벼워지나 Accuracy는 감소.

Table 5. Comparison of state-of-the-art methods for compressing VGG-16 and ResNet-56 on CIFAR-10. - represents no reported results available.

Model	Weights	FLOPs	Acc. (%)
VGG-16	15M	313M	93.6
ℓ_1-VGG-16 [31, 37]	5.4M	206M	93.4
SBP-VGG-16 [18]	-	136M	92.5
Ghost-VGG-16 (s=2)	7.7M	158M	**93.7**
ResNet-56	0.85M	125M	93.0
CP-ResNet-56 [18]	-	63M	92.0
ℓ_1-ResNet-56 [31, 37]	0.73M	91M	92.5
AMC-ResNet-56 [17]	-	63M	91.9
Ghost-ResNet-56 (s=2)	0.43M	63M	**92.7**

Table 6. Comparison of state-of-the-art methods for compressing ResNet-50 on ImageNet dataset.

Model	Weights (M)	FLOPs (B)	Top-1 Acc. (%)	Top-5 Acc. (%)
ResNet-50 [16]	25.6	4.1	75.3	92.2
Thinet-ResNet-50 [39]	16.9	2.6	72.1	90.3
NISP-ResNet-50-B [59]	14.4	2.3	-	90.8
Versatile-ResNet-50 [49]	11.0	3.0	74.5	91.8
SSS-ResNet-50 [23]	-	2.8	74.2	91.9
Ghost-ResNet-50 (s=2)	13.0	2.2	**75.0**	**92.3**
Shift-ResNet-50 [53]	6.0	-	70.6	90.1
Taylor-FO-BN-ResNet-50 [41]	7.9	1.3	71.7	-
Slimmable-ResNet-50 0.5× [58]	6.9	1.1	72.1	-
MetaPruning-ResNet-50 [36]	-	1.0	73.4	-
Ghost-ResNet-50 (s=4)	6.5	1.2	**74.1**	**91.9**

실험 & 결과

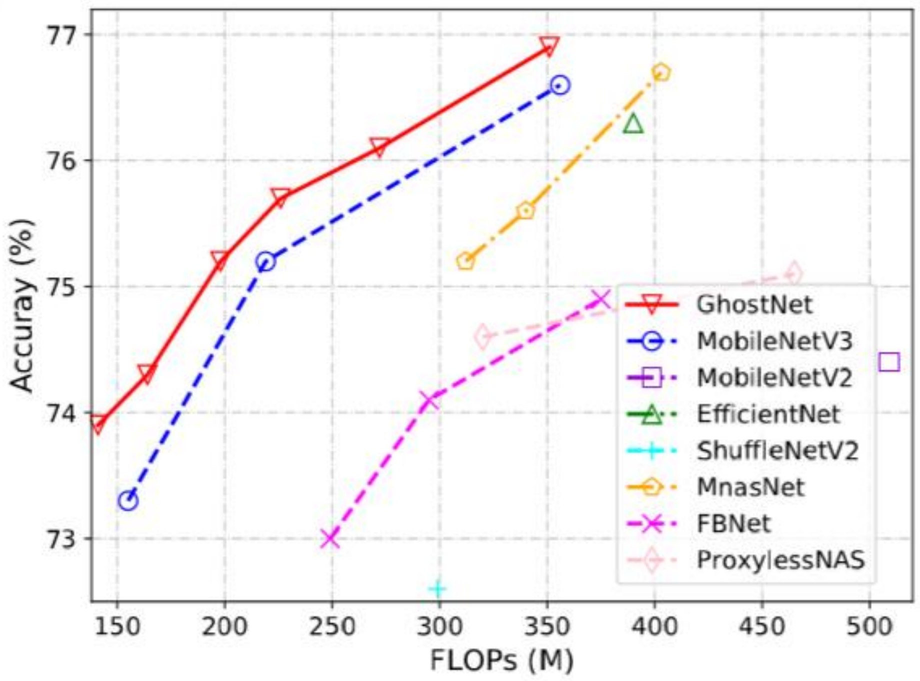


Figure 6. Top-1 accuracy *v.s.* FLOPs on ImageNet dataset.

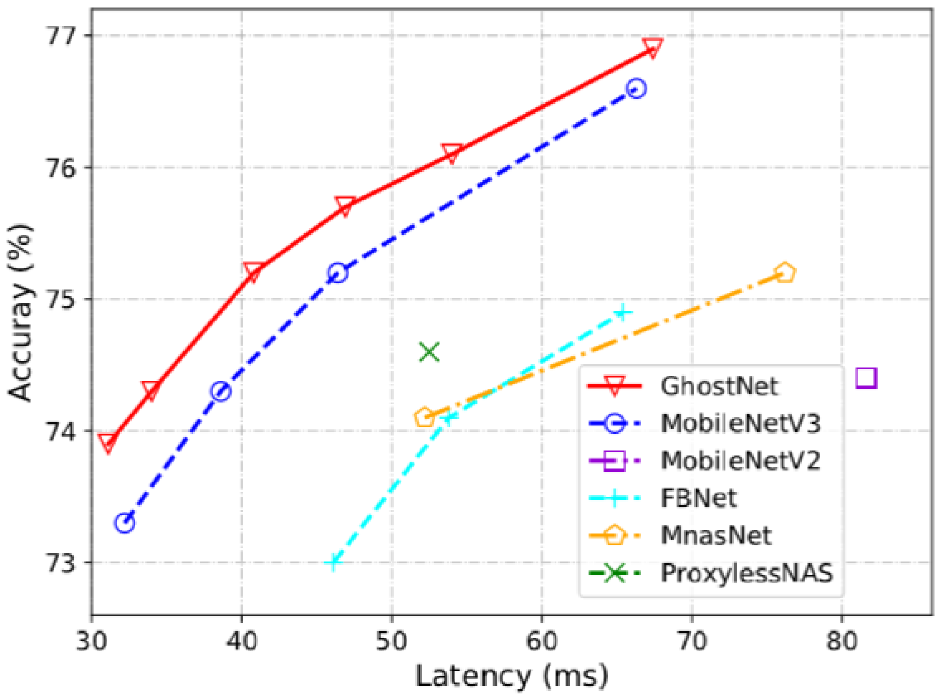


Figure 7. Top-1 accuracy *v.s.* latency on ImageNet dataset.

width multiplier

Table 7. Comparison of state-of-the-art small networks over classification accuracy, the number of weights and FLOPs on ImageNet dataset.

Model	Weights (M)	FLOPs (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
ShuffleNetV1 0.5× (g=8) [61]	1.0	40	58.8	81.0
MobileNetV2 0.35× [44]	1.7	59	60.3	82.9
ShuffleNetV2 0.5× [40]	1.4	41	61.1	82.6
MobileNetV3 Small 0.75× [20]	2.4	44	65.4	-
GhostNet 0.5×	2.6	42	**66.2**	**86.6**
MobileNetV1 0.5× [21]	1.3	150	63.3	84.9
MobileNetV2 0.6× [44, 40]	2.2	141	66.7	-
ShuffleNetV1 1.0× (g=3) [61]	1.9	138	67.8	87.7
ShuffleNetV2 1.0× [40]	2.3	146	69.4	88.9
MobileNetV3 Large 0.75× [20]	4.0	155	73.3	-
GhostNet 1.0×	5.2	141	**73.9**	**91.4**
MobileNetV2 1.0× [44]	3.5	300	71.8	91.0
ShuffleNetV2 1.5× [40]	3.5	299	72.6	90.6
FE-Net 1.0× [6]	3.7	301	72.9	-
FBNet-B [52]	4.5	295	74.1	-
ProxylessNAS [2]	4.1	320	74.6	92.2
MnasNet-A1 [47]	3.9	312	75.2	92.5
MobileNetV3 Large 1.0× [20]	5.4	219	75.2	-
GhostNet 1.3×	7.3	226	**75.7**	**92.7**