Hands On Deep Learning for IoT

22.12.12(Mon) ~ 22.12.16(Fri)

서근태 연구원

Chapter

chapter 1
IoT Ecosystems, Deep Learning Techniques, and Frameworks

chapter 2
Hands-On Deep Learning Application Development for IoT

chapter 3
3-1. Road's Fault Detection / **MobileNet v1**
3-2. Image-based smart solid waste separation / **MobileNet v1**
501 glass / 594 paper / 403 cardboard/ 482 plastic / 410 metal / 137 trash

chapter 4
4-1. Voice Controlled Smart Light / **MobileNet v1**
five commands, namele 'no', 'on,, 'off', 'stop', and 'yes'.
4-2. voice-controlled home access /**MobileNet v1**

chapter 5
5. Indoor_Localization_IoT/ **LSTM**

chapter 6
6-1. Human Activity Recognition (HAR) / **LSTM**
Class Distribution : Walking: 424,400 (38.6%), Jogging: 342,177 (31.2%),
Upstairs: 122,869 (11.2%)
Downstairs: 100,427 (9.1%), Sitting: 59,939 (5.5%), Standing: 48,395 (4.4%)
6-2. Smart Class Room (FER-based) / **CNN**
얼굴 표정에 나타난 감정
(0=화남, 1=혐오, 2=두려움, 3=행복, 4=슬픔, 5=놀람, 6=보통)

chapter 7
7-1. Intelligent Host Intrusion Detection in IoT / **LSTM**
7-2. Network Intrusion Detection / **AutoEncoder, Simple DNN**

chapter 8
8. Predictive_Maintance_IoT/ **RandomForest, LSTM**

chapter 9
9-1. Remote Chronic Disease Management / **CNN, LSTM**
9-2. IoT for Acne detection and Care / **MobileNet V1**

MobileNet v1

Efficient Convolution Neural Networks(CNN)
for Mobile Vision Application

MobileNet v1 등장배경

- 먼저 딥러닝의 상용화를 위하여 필요한 여러가지 제약 사항을 개선시키기 위하여 경량화 네트워크에 대한 연구가 시작되었습니다.

- 딥러닝을 이용한 상품들이 다양한 환경에서 사용되는데 특히, 고성능 컴퓨터가 아닌 상황에서 가벼운 네트워크가 필요하게 됩니다.

- 예를 들어 데이터 센터의 서버나 스마트폰, 자율주행자동차 또는 드론과 같이 가격을 무작정 높일 수 없어서 제한된 하드웨어에 딥러닝 어플리케이션이 들어가는 경우입니다.
  - 이러한 경우에 실시간 처리가 될 정도 성능의 뉴럴넷이 필요하고 또한 얼마나 전력을 사용할 지도 고려를 해야합니다.

- 이러한 제약 사항을 충분히 만족하면서 또한 아래와 같은 성능이 꽤 괜찮아야 어플리케이션에 적용을 할 수 있습니다.
  - 충분히 납득할만한 정확도
  - 낮은 계산 복잡도
  - 저전력 사용
  - 작은 모델 크기

MobileNet v1

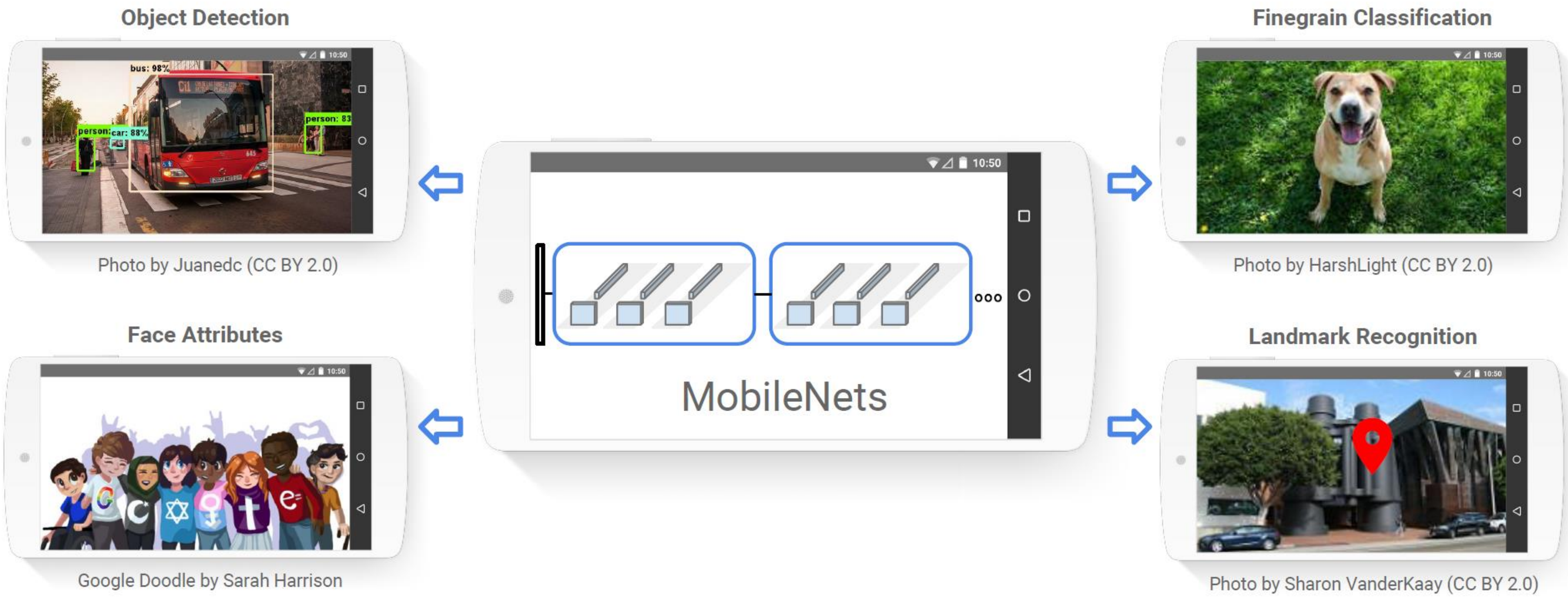


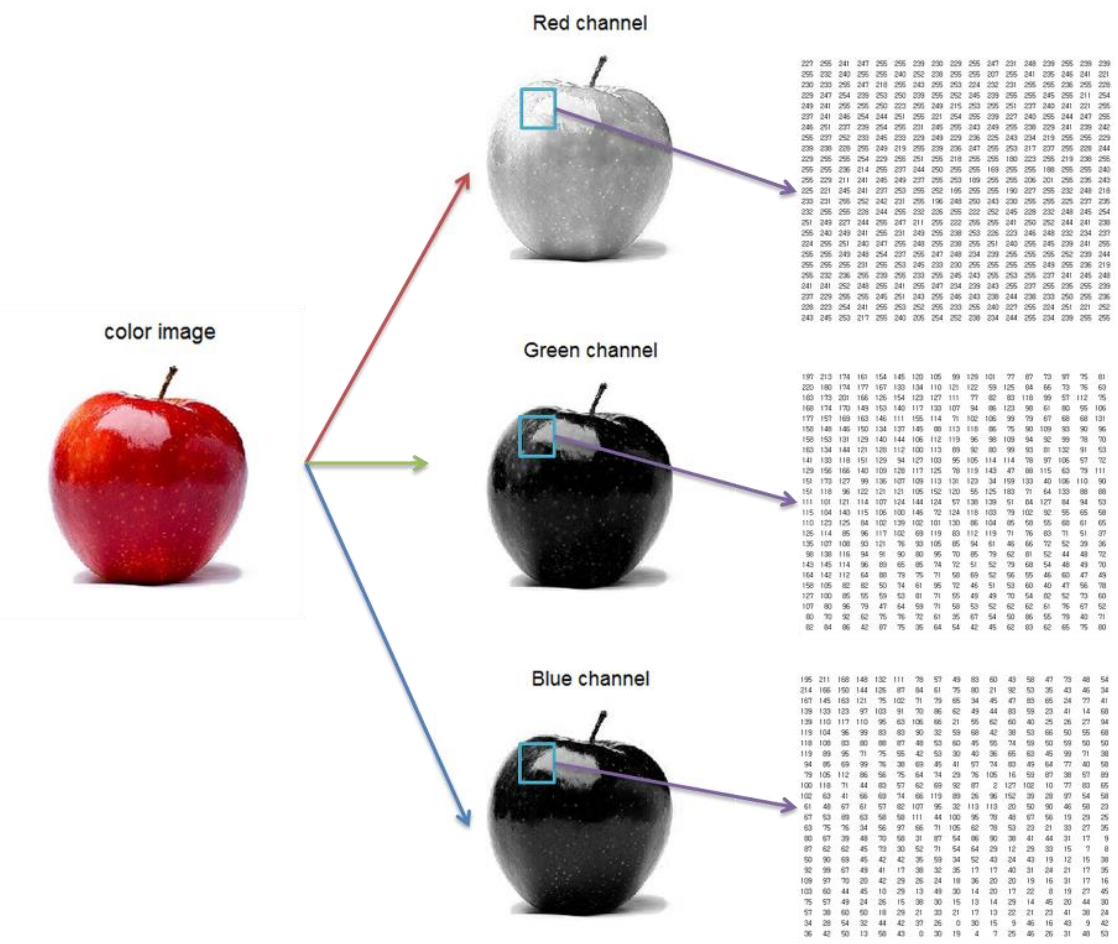
Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

논문(아카이브) : https://arxiv.org/abs/1704.04861
모델 배포(구글 텐서플로) : models/mobilenet_v1.md at master · tensorflow/models · GitHub

Definition of Deep Learning Terms

1. (RGB) Channel
2. Filter
3. Stride/Padding
4. Convolution

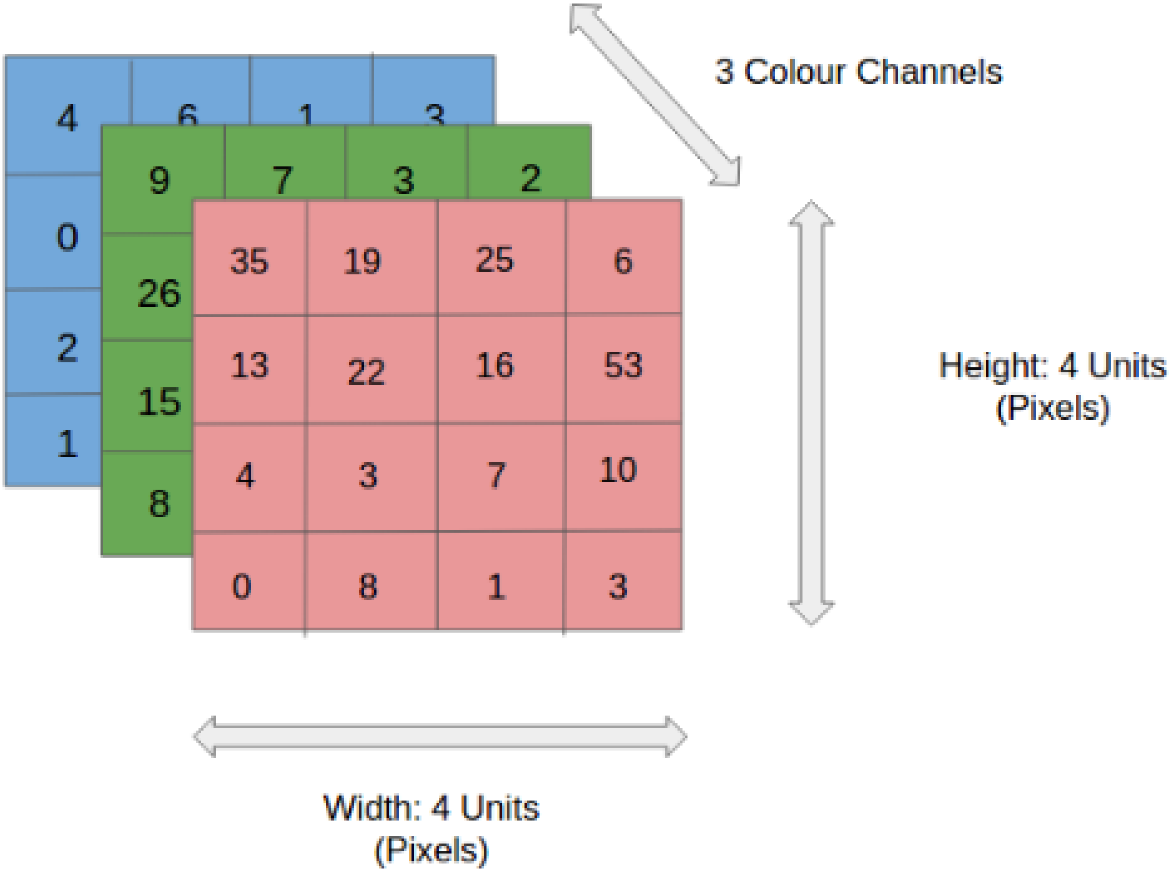
RGB channel



1행 1열값
Red : 227
Green : 197
Blue : 195

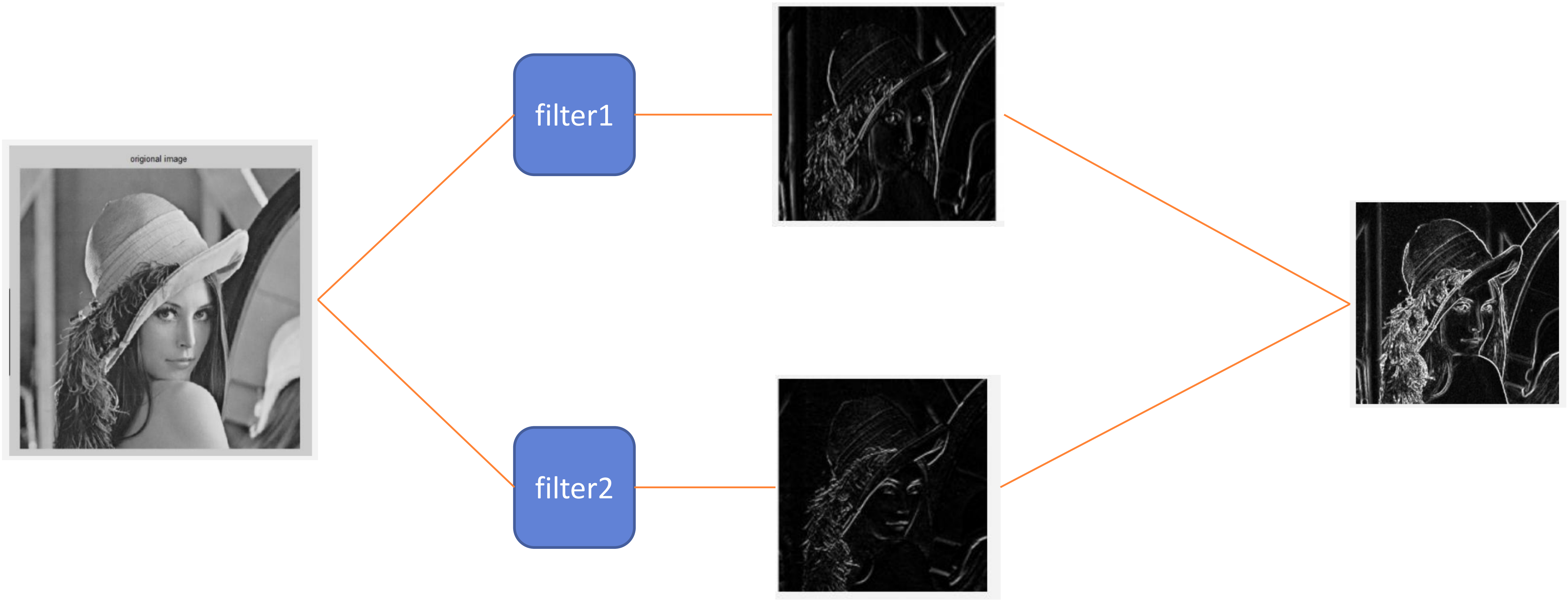
-> Red일 확률이 높다.

RGB channel(계속)



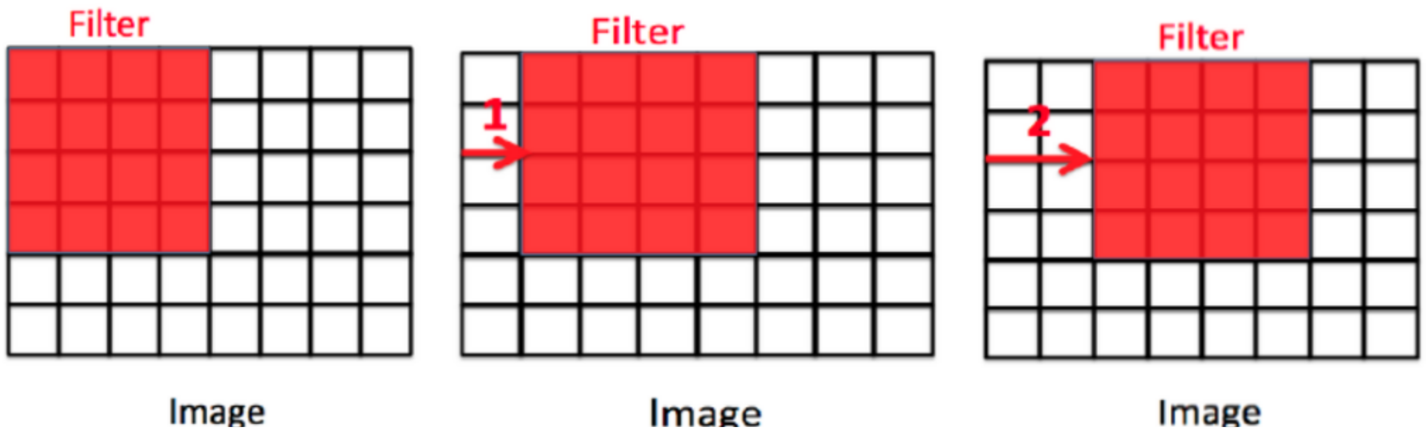
Filter

Purpose of Filter use : 이미지의 세부특징(ex. edge) 추출, 이미지의 크기를 축소 등



Stride/Padding

Stride



Padding

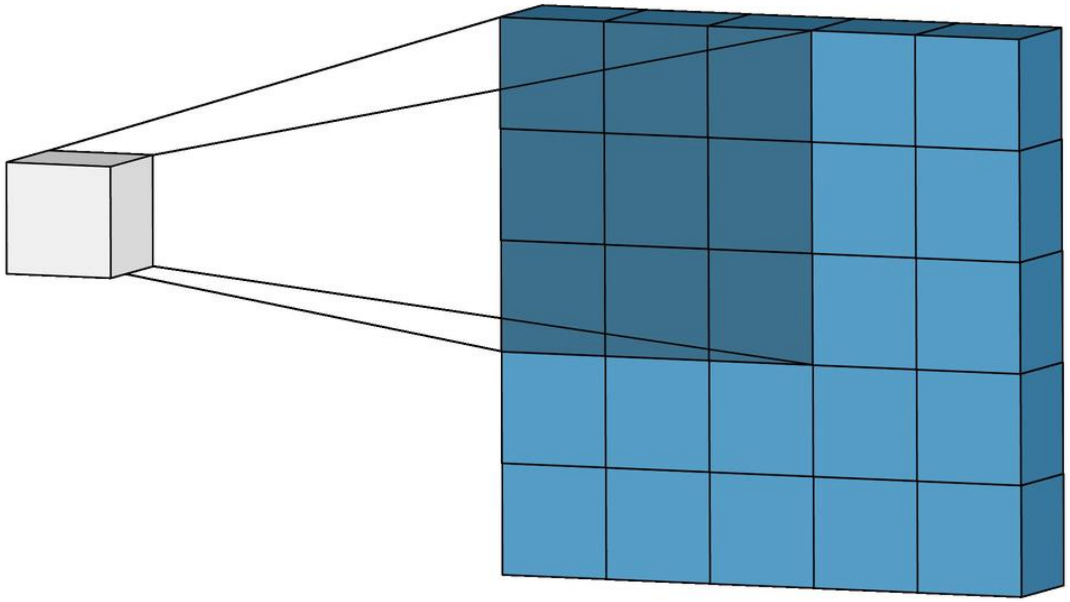
0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

Standard Convolution



ex) 5X5 Image + 3X3 Filter => 3X3 image

MobileNet v1

Body

MobileNet Body

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Depthwise conv + Pointwise conv

```
class MobileNet(nn.Module):
    def __init__(self):
        super(MobileNet, self).__init__()

        def conv_bn(input, output, stride, padding): # 처음 시작할때 convolution
            return nn.Sequential(
                nn.Conv2d(input, output, 3, stride, padding, bias=False),
                nn.BatchNorm2d(out),
                nn.ReLU(inplace=True)
            )

        def conv_dw(input, output, stride, padding): # depthwise separable convolution
            return nn.Sequential( # depthwise convolution
                nn.Conv2d(input, output, 3, stride, padding, groups=in, bias=False),
                nn.BatchNorm2d(input),
                nn.ReLU(inplace=True),

                nn.Conv2d(input, output, 1, 1, 0, bias=False), # pointwise convolution
                nn.BatchNorm2d(output),
                nn.ReLU(inplace=True),
            )

        self.model = nn.Sequential(
            conv_bn(  3,   32, 2),
            conv_dw( 32,   64, 1),
            conv_dw( 64, 128, 2),
            conv_dw(128, 128, 1),
            conv_dw(128, 256, 2),
            conv_dw(256, 256, 1),
            conv_dw(256, 512, 2),
            conv_dw(512, 512, 1),
            conv_dw(512, 512, 1),
            conv_dw(512, 512, 1),
            conv_dw(512, 512, 1),
            conv_dw(512, 512, 1),
            conv_dw(512, 1024, 2),
            conv_dw(1024, 1024, 1),
            nn.AvgPool2d(7),
        )
        self.fc = nn.Linear(1024, 1000)

    def forward(self, x):
        x = self.model(x)
        x = x.view(-1, 1024)
        x = self.fc(x)
        return x
```

MobileNet Body

- Standard Convolution
- Depthwise Separable Convolution

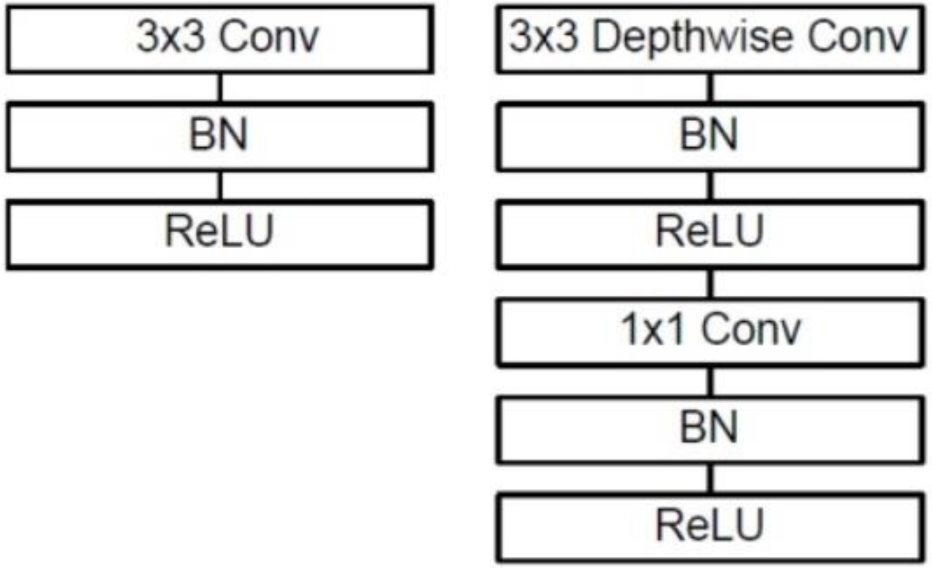
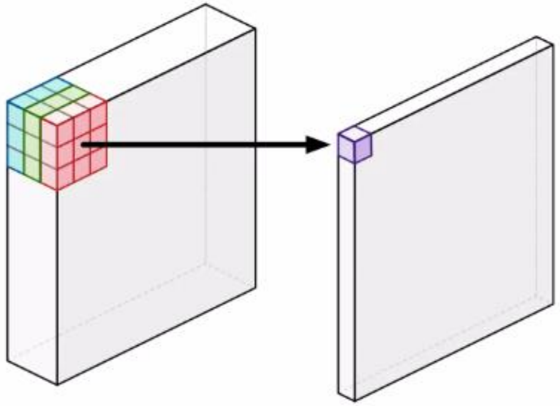
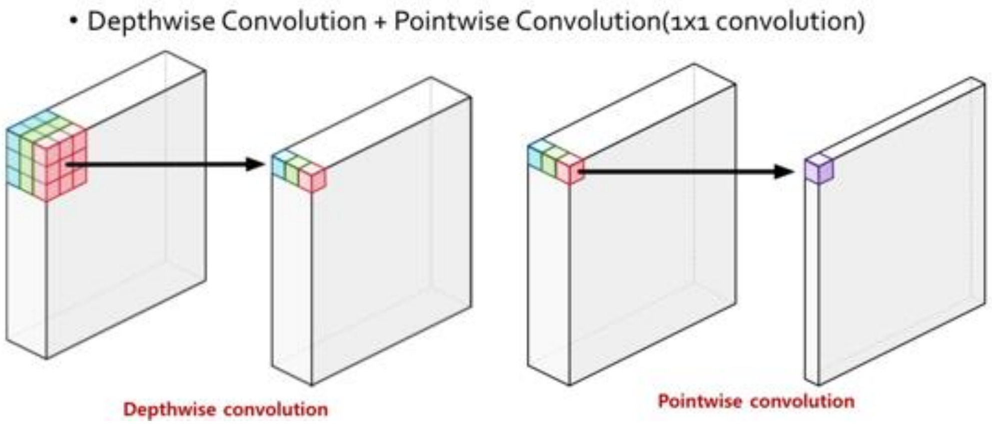


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

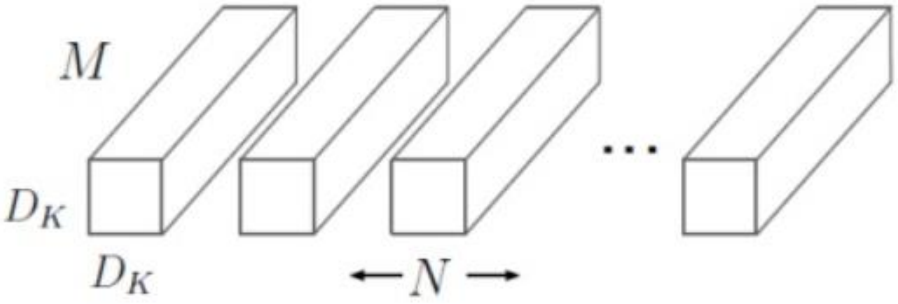
- Standard Convolution



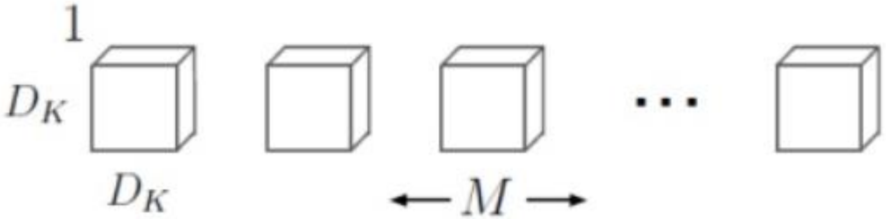
- Depthwise Separable Convolution



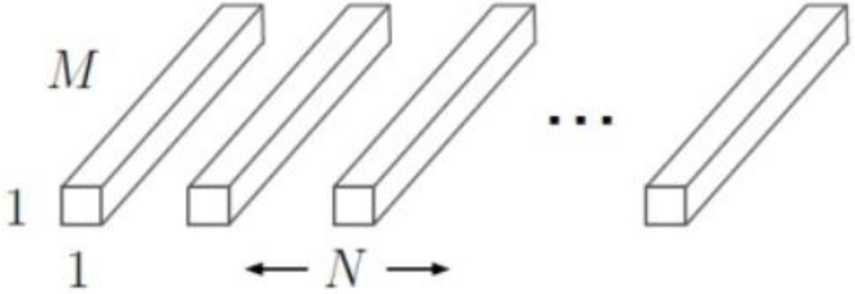
Depthwise Separable Convolution
Filter Shape



(a) Standard Convolution Filters

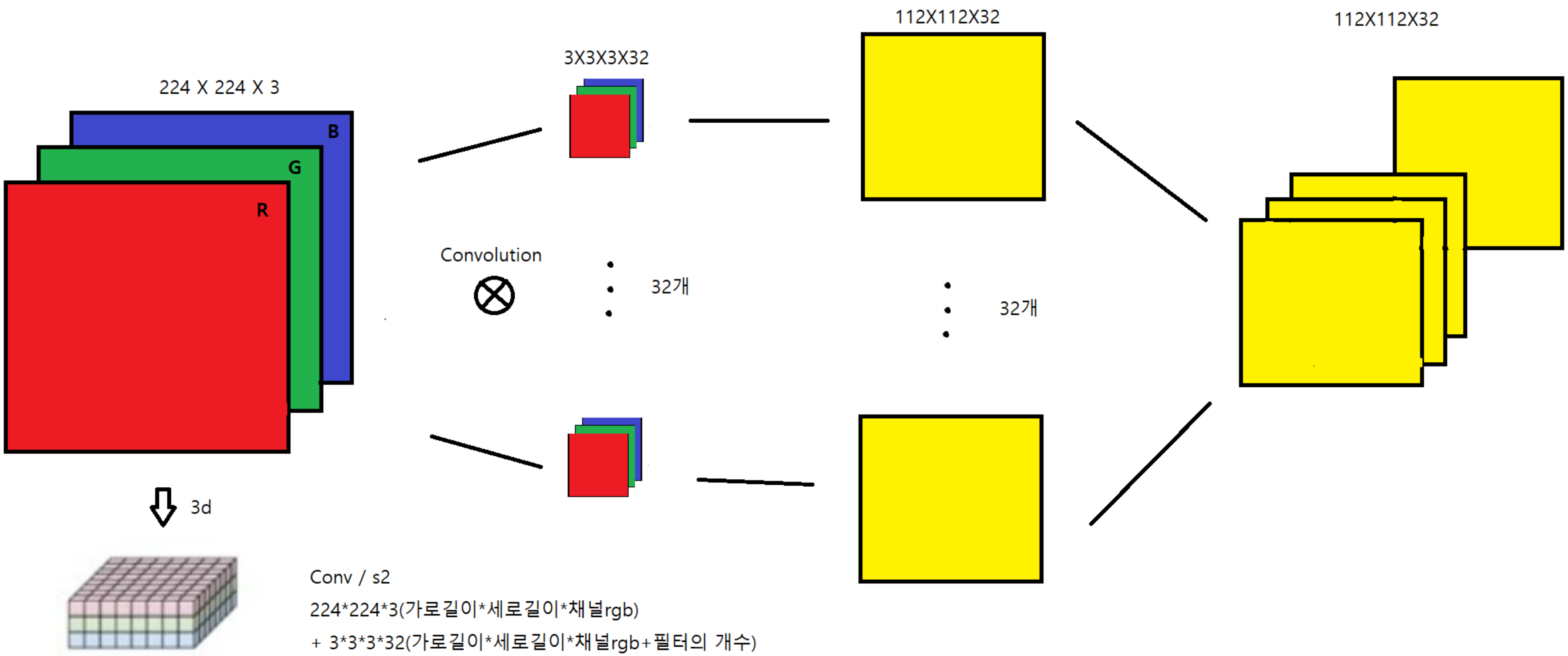


(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Conv / s2



Conv / s2

224*224*3(가로길이*세로길이*채널rgb)

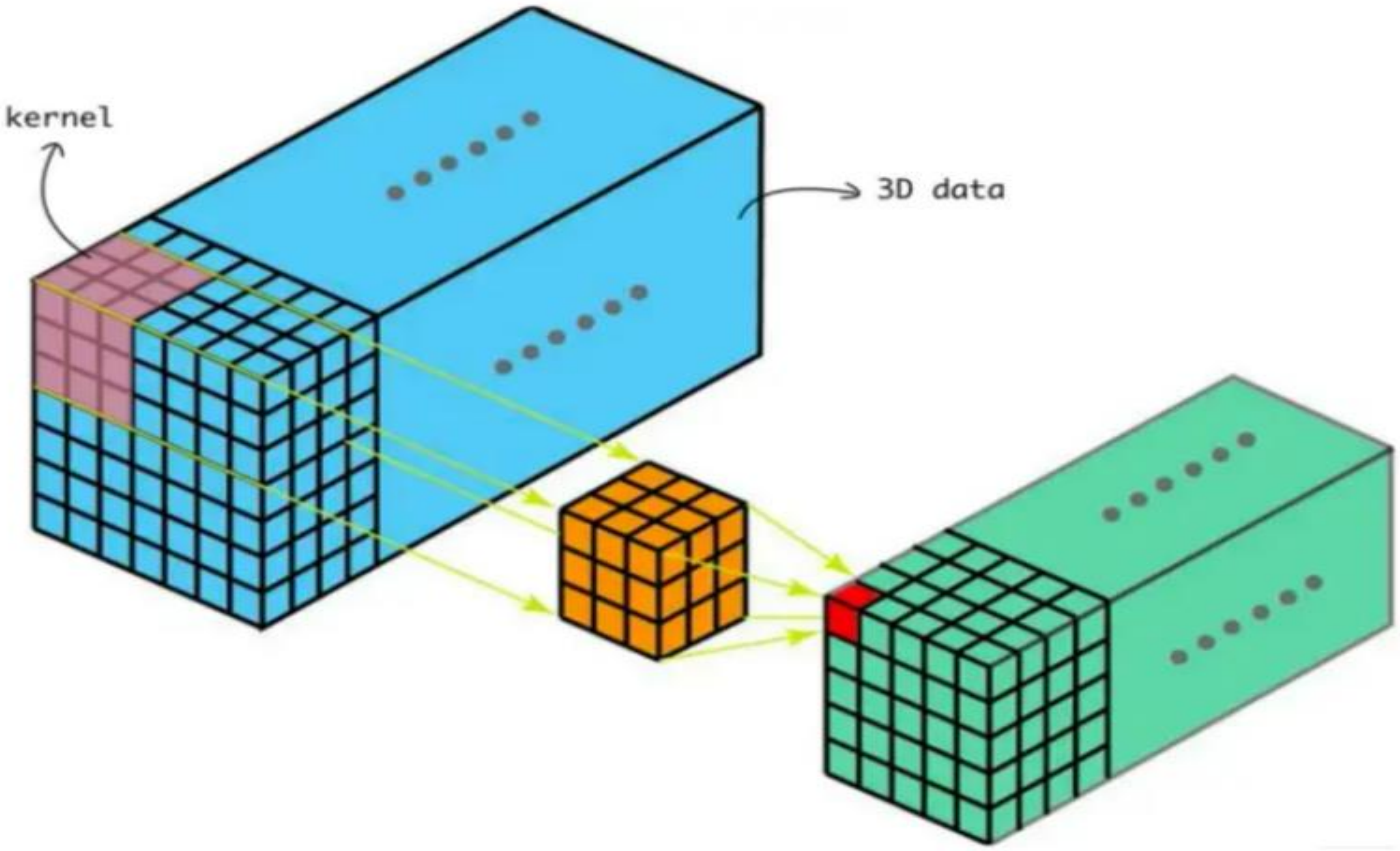
+ 3*3*3*32(가로길이*세로길이*채널rgb+필터의 개수)

-> 112*112*32(가로길이*세로길이*output 채널이자 필터 개수)

공식 : (input size-filter size+2*padding)/stride + 1

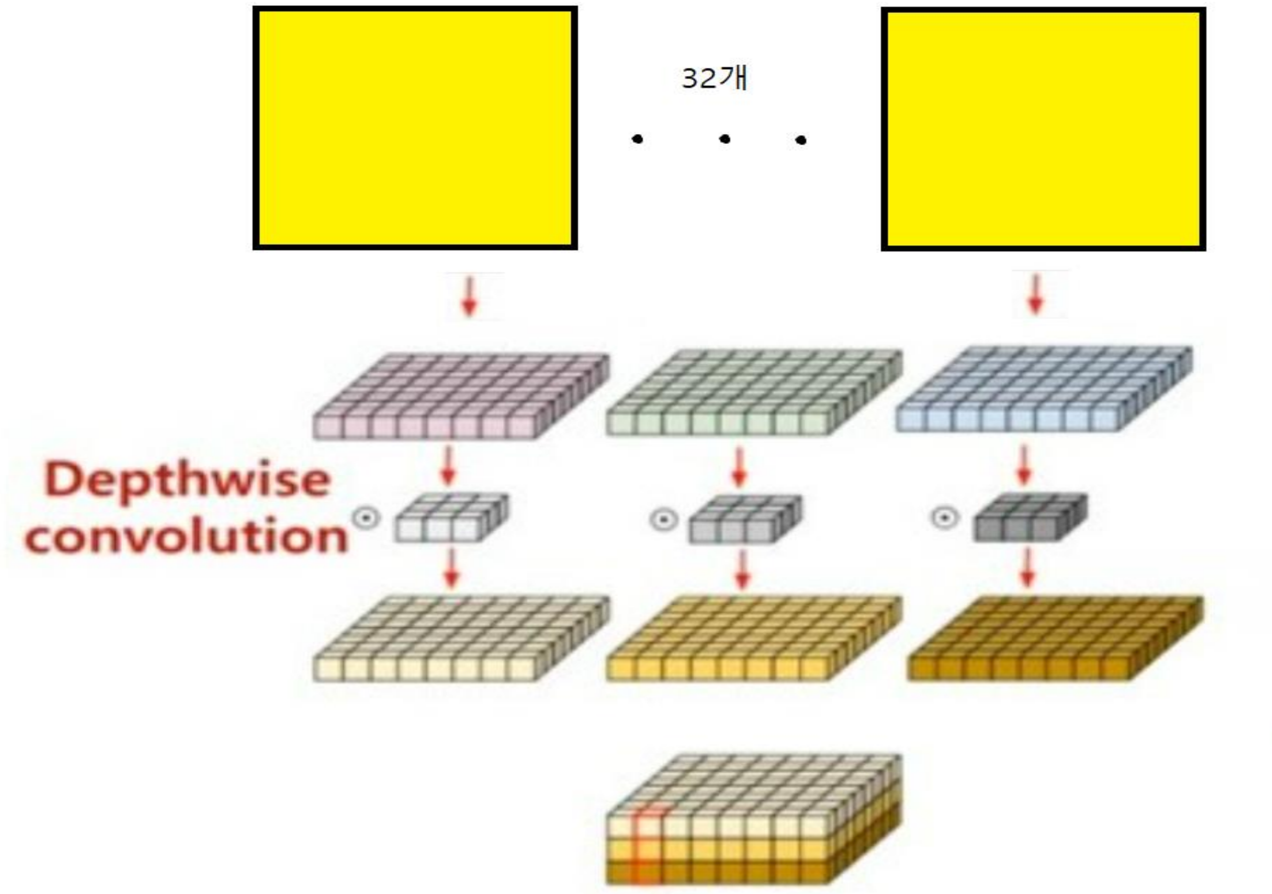
계산식 : (224-3+2*1)/2 +1 = 223/2+1 = 111.5+1 -> 112.5 -> 112(소수점아래는 버림)

Conv / s2 (3D)



Kernel sliding on 3D data

Conv dw / s1 = Depthwise



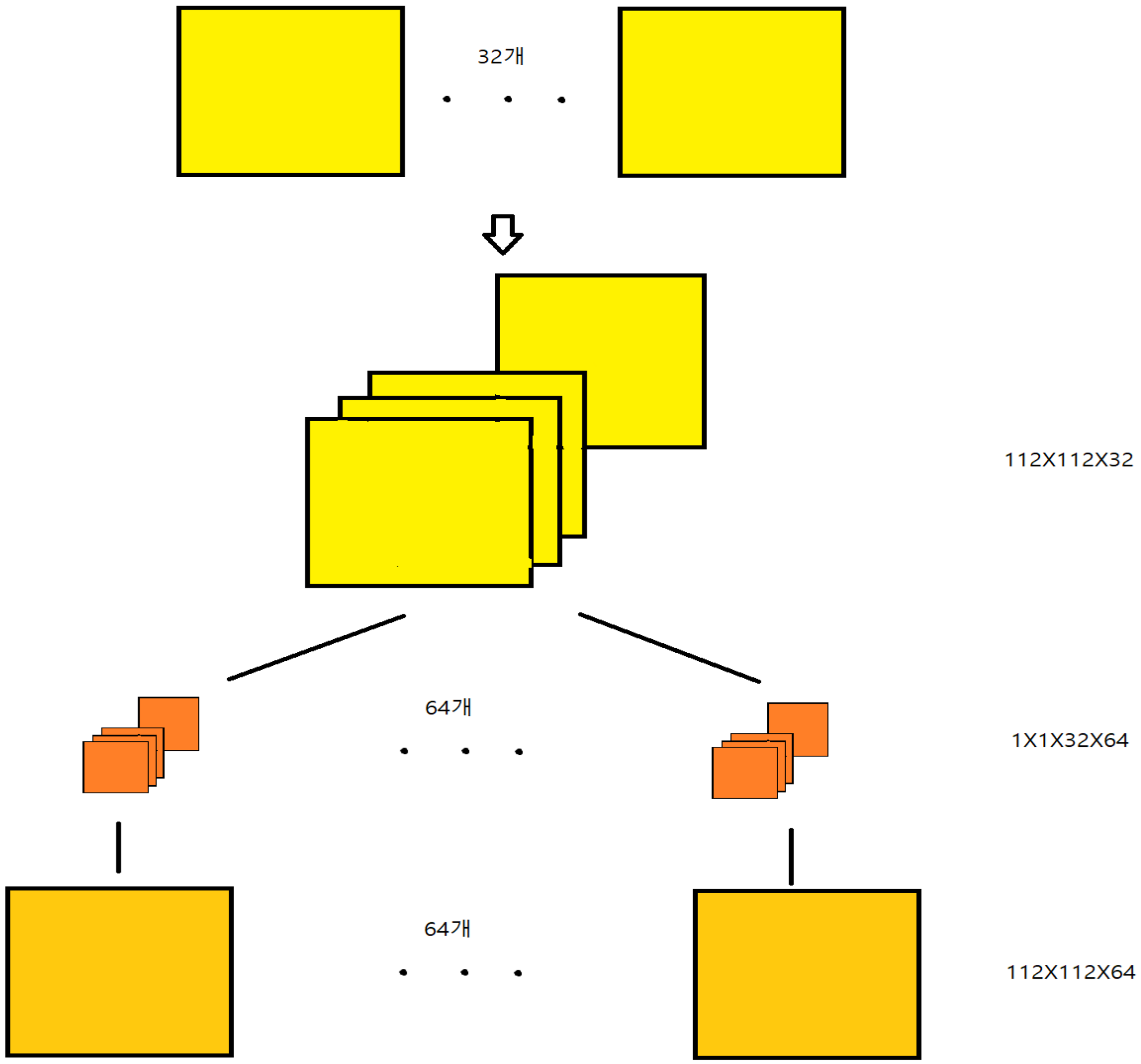
112X112X32

3X3X32

112X112X32

Conv dw / s1
112*112*32(가로길이*세로길이*채널32)
+ 3*3*32(가로길이*세로길이*필터의 개수)
-> 112*112*32
계산식 : (112-3+2*1)/1 +1 = 109+2*1/1+1 = 111+1 -> 112

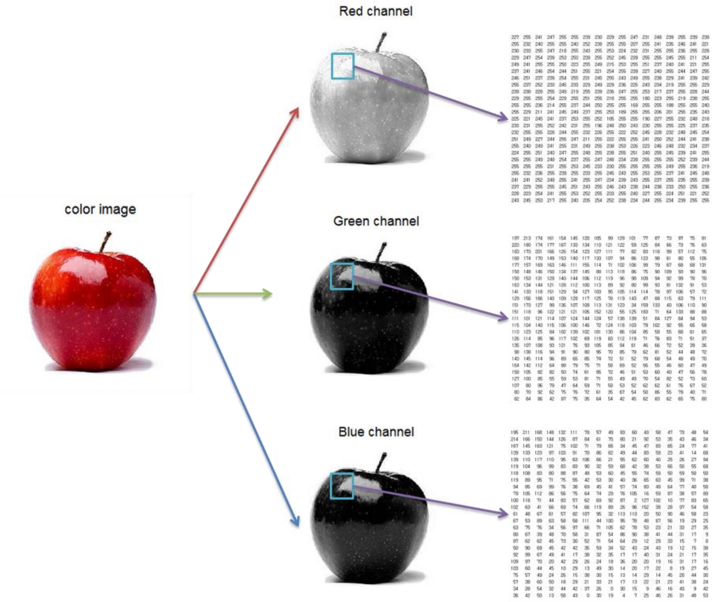
Conv / s1 = Pointwise

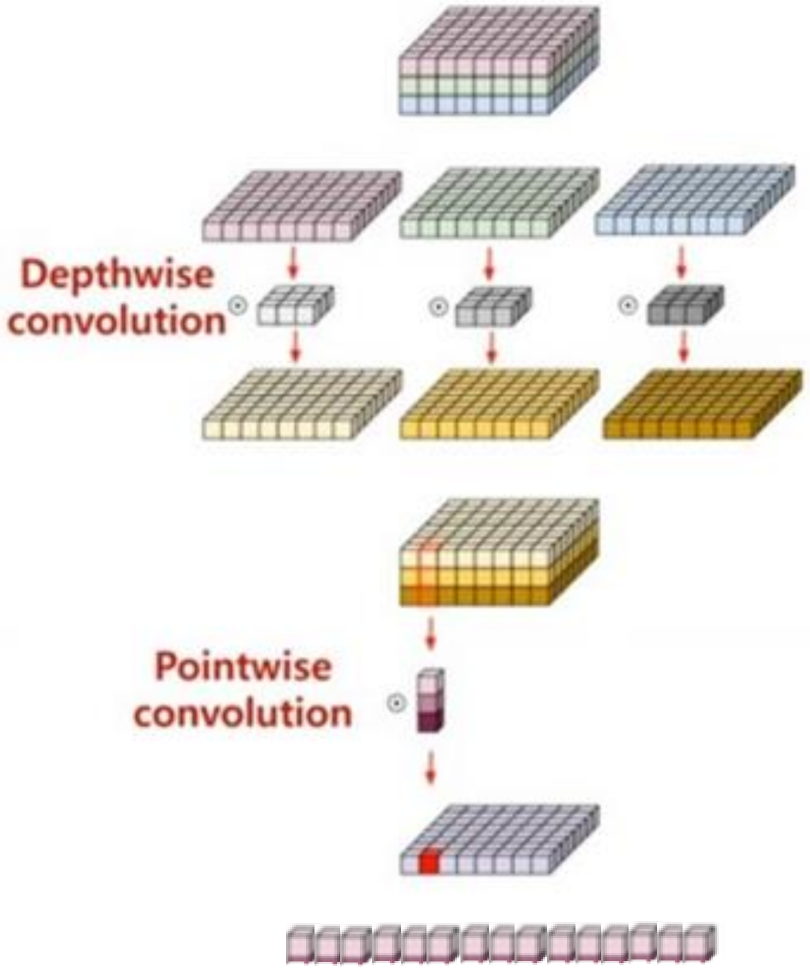


Conv / s1
112*112*32(가로길이*세로길이*채널)
+ 1*1*32*64(가로길이*세로길이*채널+필터의 개수)
-> 112*112*64(가로길이*세로길이*채널)
계산식 : (112-1+2*0)/1 +1 = 111/1+1 = 112

MobileNet Body

I want to know the color of the apple.





→ Apple R,G,B(stack)

→ Apple R,G,B(seperate)

→ Filter

→ 이미지 특징 강조(same size)
이미지 크기/차원 축소(different size)

→ Weighted Apple R,G,B(stack)

→ Pointwise Filter

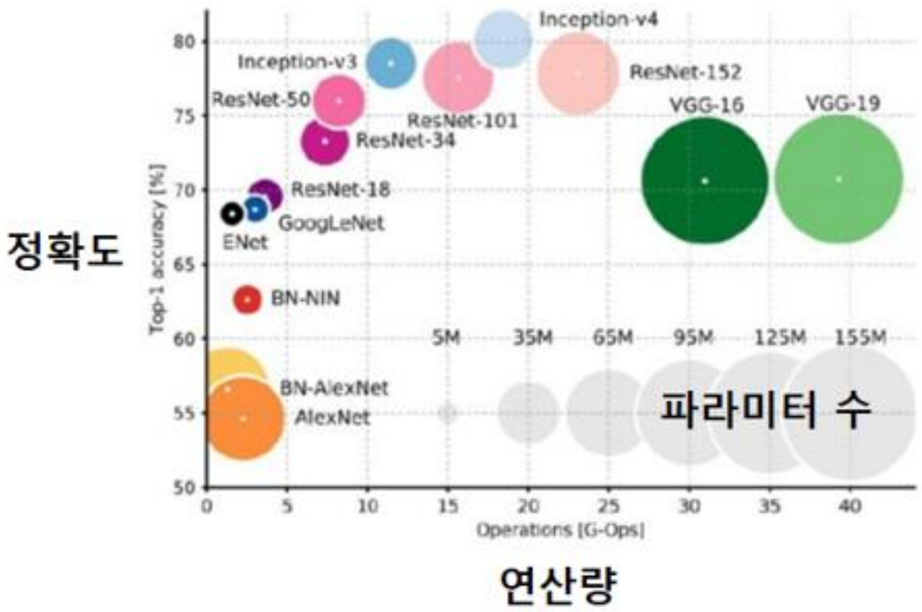
→ New Apple image(stack)

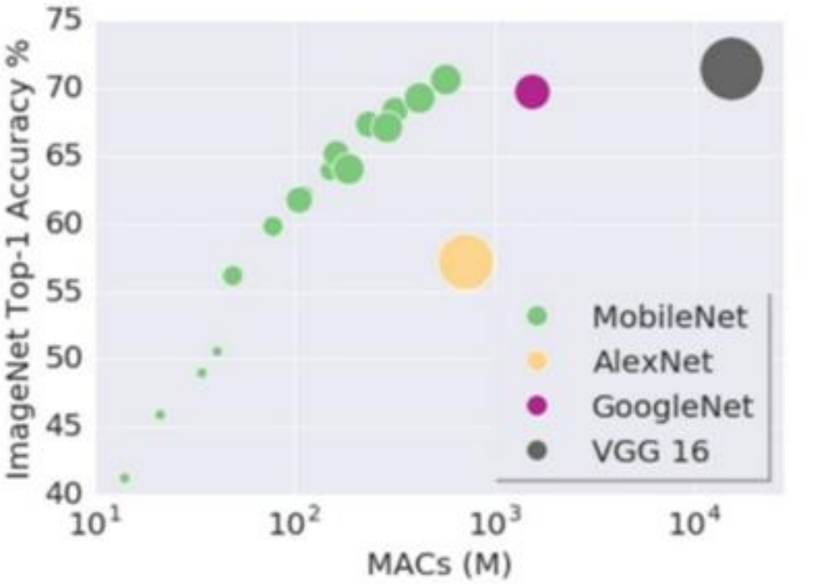
→ New Apple image(seperate)

Pooling/Fully-Connected/Activation Function

Classification → Red

MobileNet Performance





Conv vs Mobilenet 연산량(계산량) 비교(8~9배 차이)

1. Conv
필터 가로*필터 세로*채널 수*필터 개수 * 입력 이미지 가로 *입력 이미지 세로
-> 3*3*3*32*224*224 = **43,352,064**

2. Mobilenet v1
depthwise (필터 가로*필터 세로*채널 수*입력이미지 가로*입력 이미지 세로)
+ pointwise (입력이미지 가로*입력 이미지 세로*채널 수*필터 수)
-> 3*3*3*224*224 + 224*224*3*32 = 1,354,752 + 4,816,896 = **6,171,648**