NUS | Computing

National University of Singapore

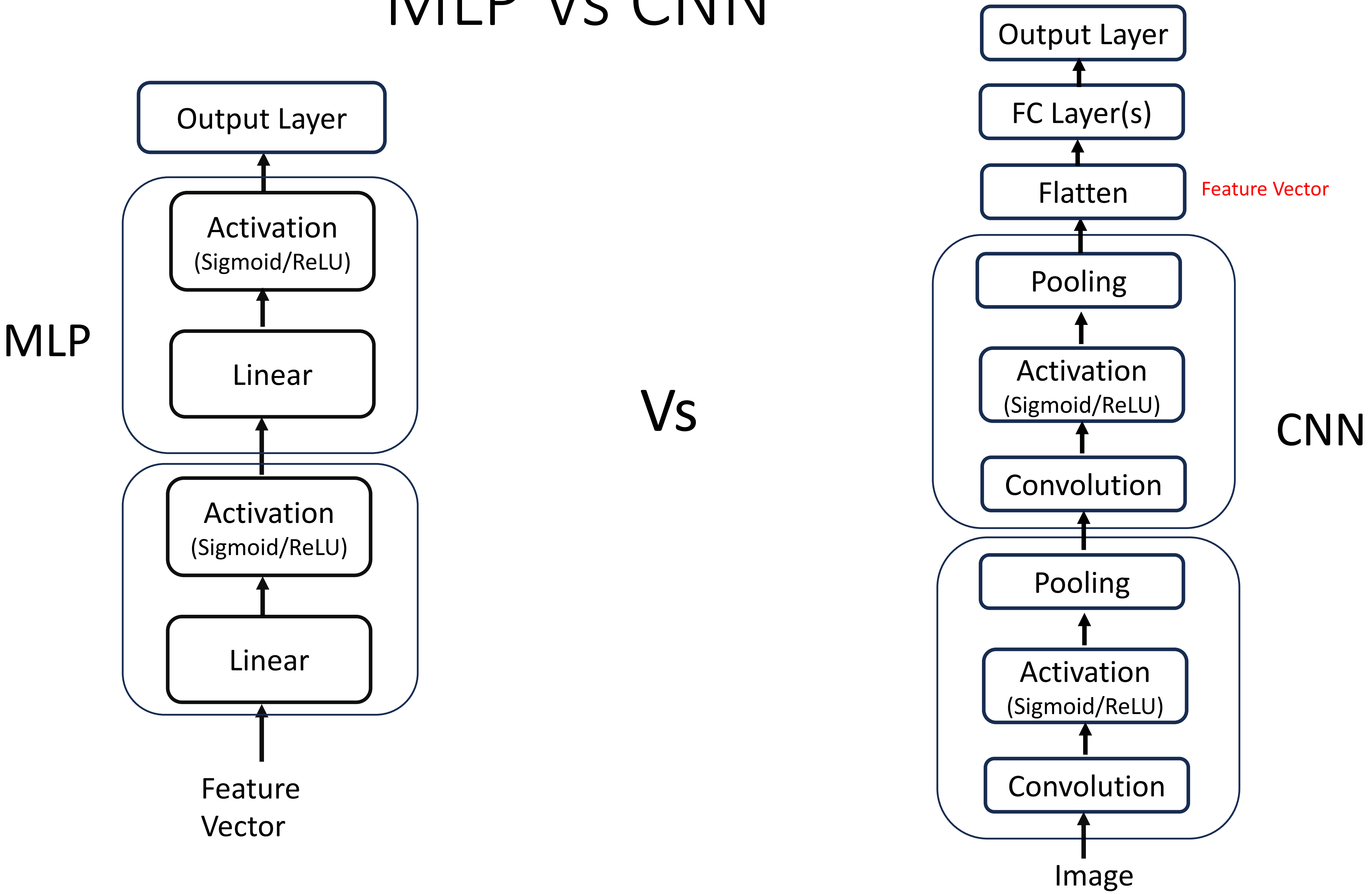
IT5005 Artificial Intelligence

Sirigina Rajendra Prasad

AY2025/2026: Semester 1

CNN Contd.

MLP Vs CNN


Output Layer
Activation
(Sigmoid/ReLU)
MLP
Linear
Activation
(Sigmoid/ReLU)
Linear
Feature
Vector
Vs
Output Layer
FC Layer(s)
Flatten
Feature Vector
Pooling
Activation
(Sigmoid/ReLU)
CNN
Convolution
Pooling
Activation
(Sigmoid/ReLU)
Convolution
Image

A Closer Look at CNN Architecture

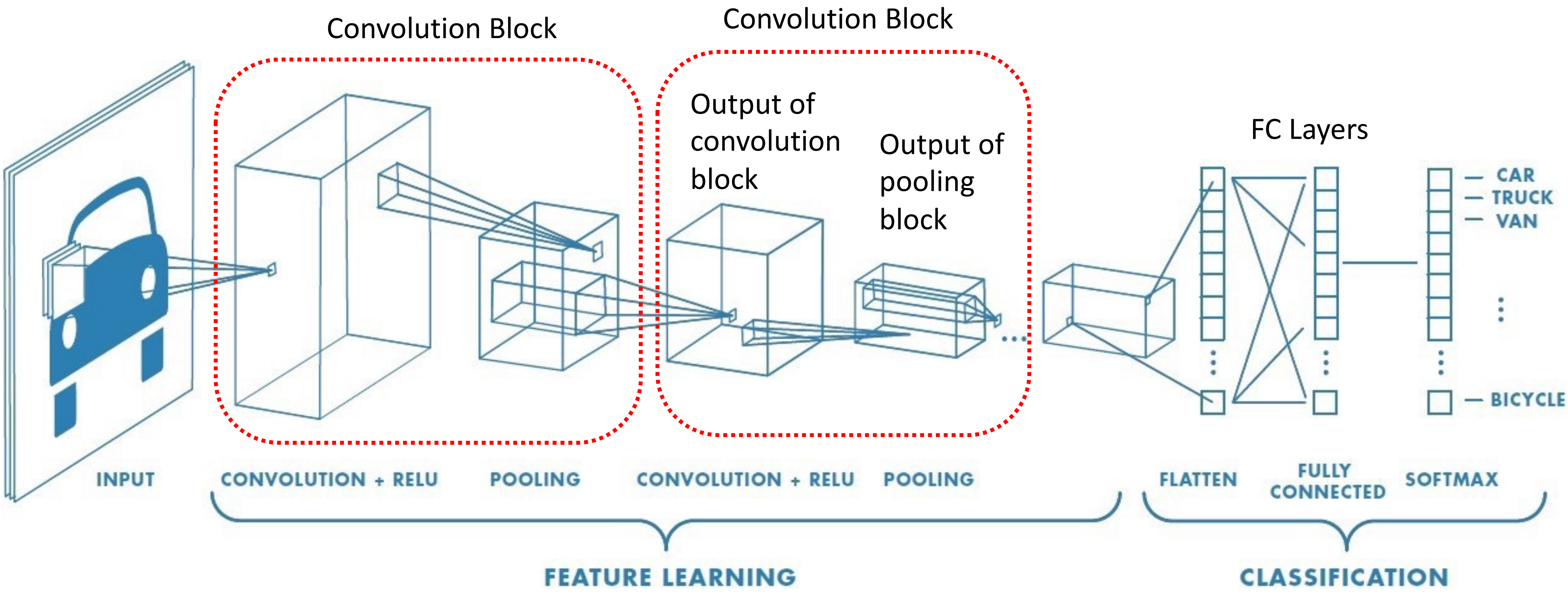
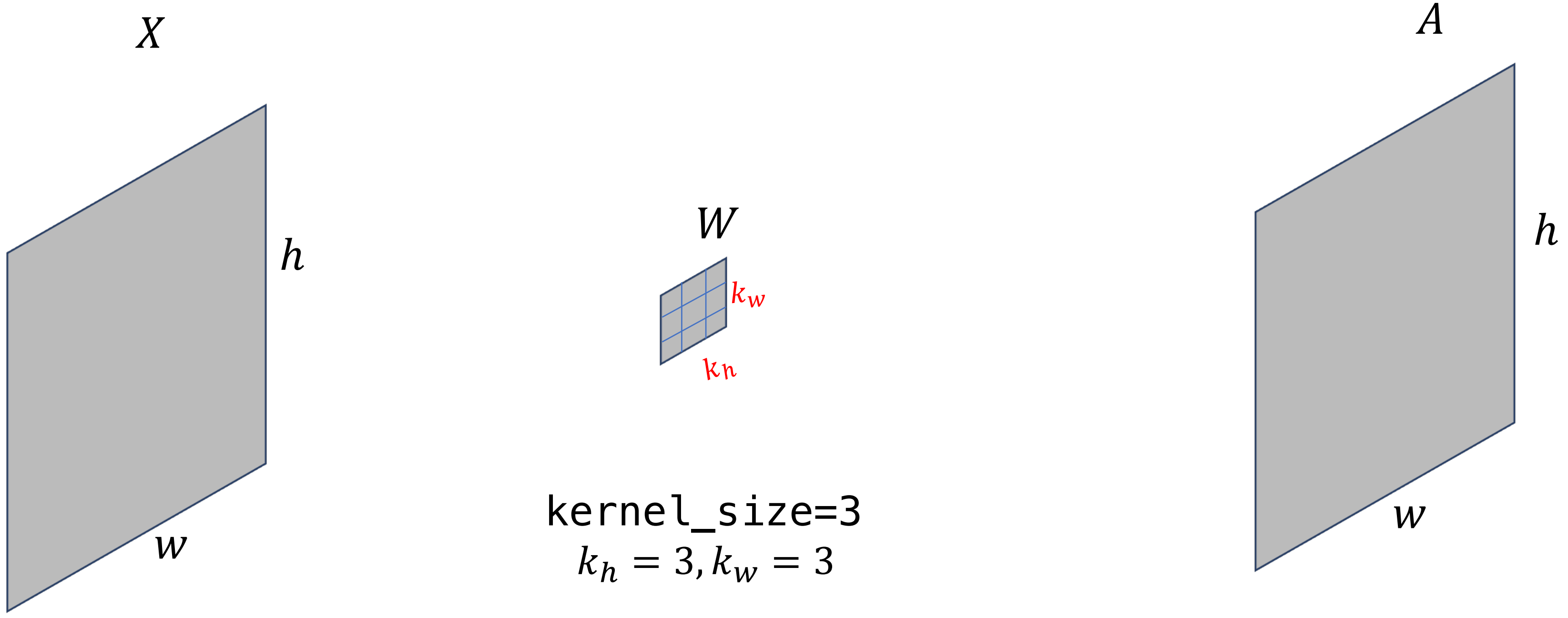


Image credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolution Layer

```
nn.Conv2d(in_channels= , out_channels= , kernel_size=3, stride=1, padding=1)
```

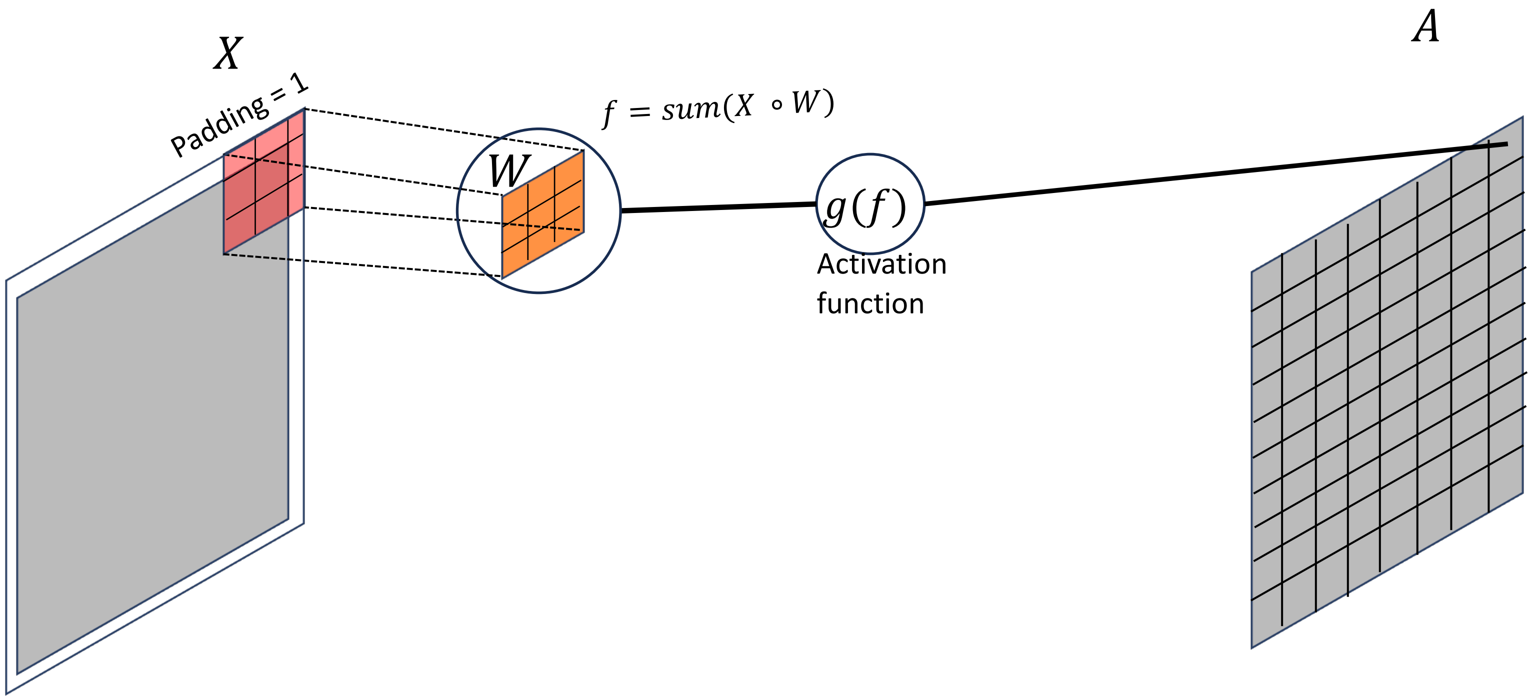


Convolution Layer

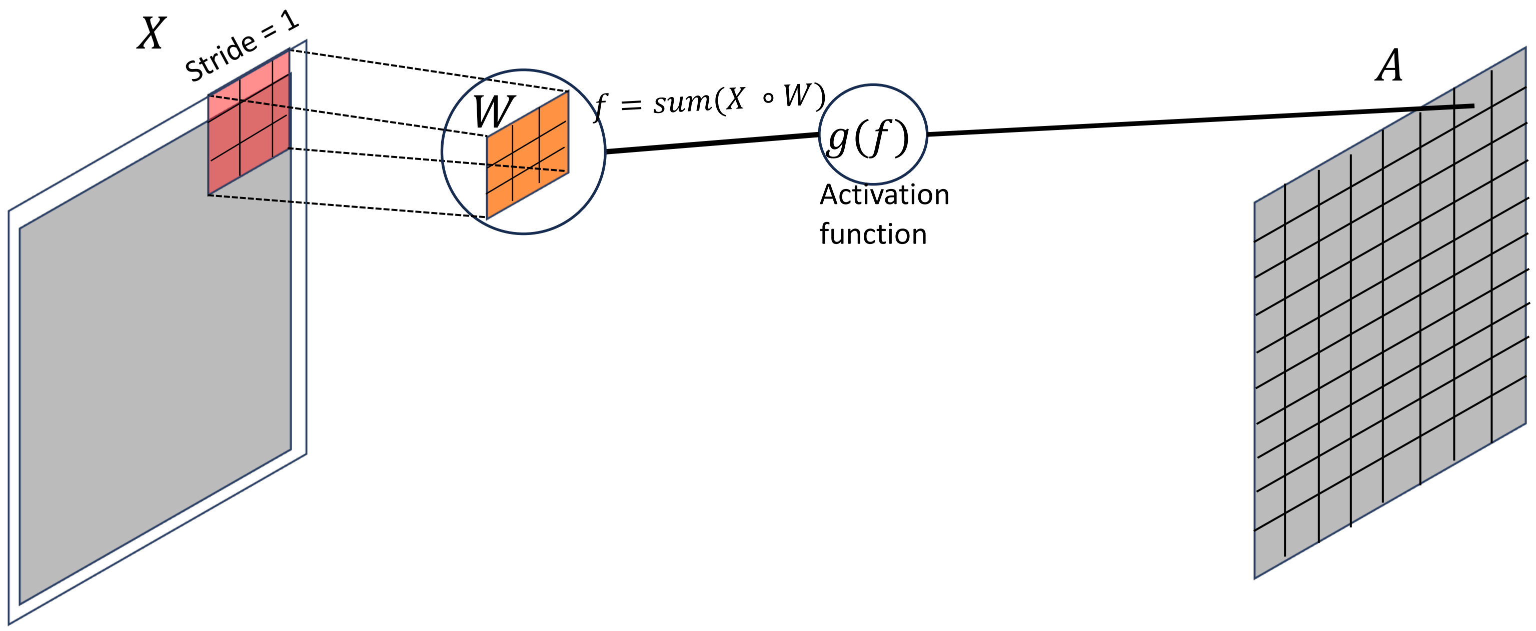
$$f = sum(X \circ W)$$

1. $X \circ W$: Elementwise multiplication of W and X
2. $sum(X \circ W)$ sum of the elements of $X \circ W$

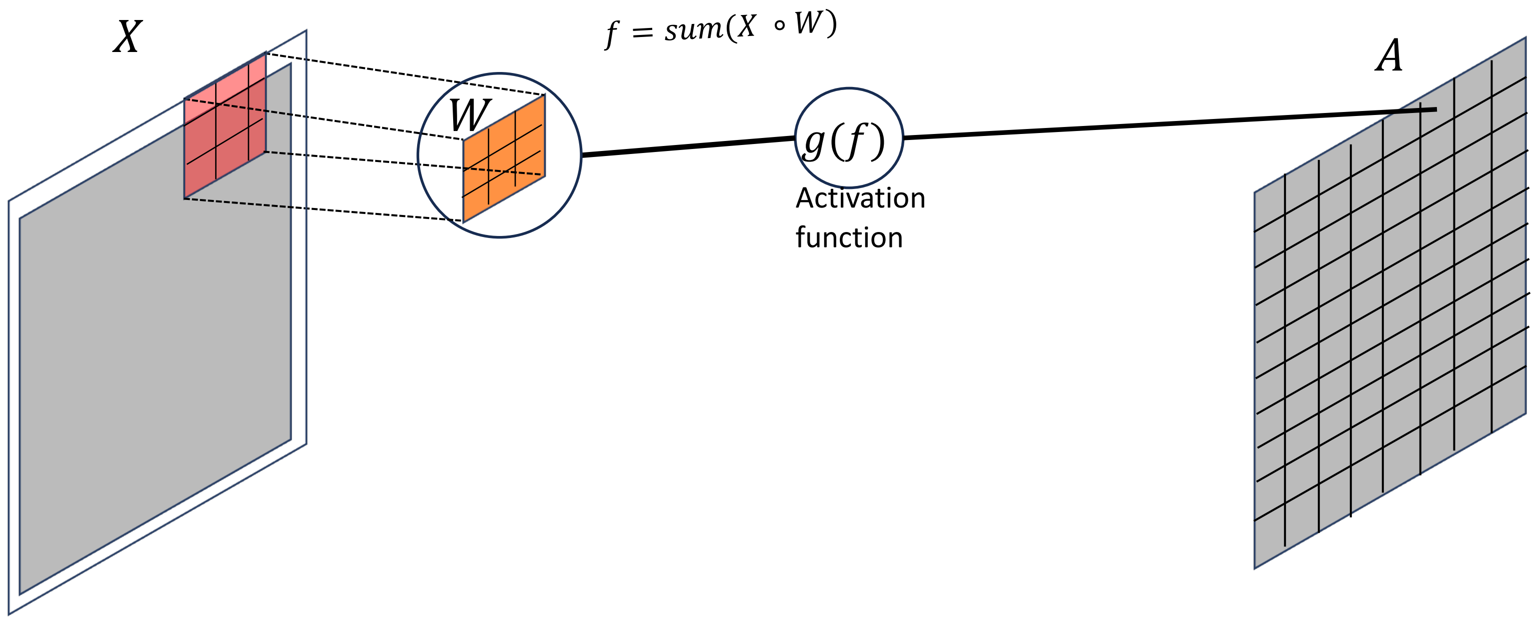
MAC: Multiply-Accumulate



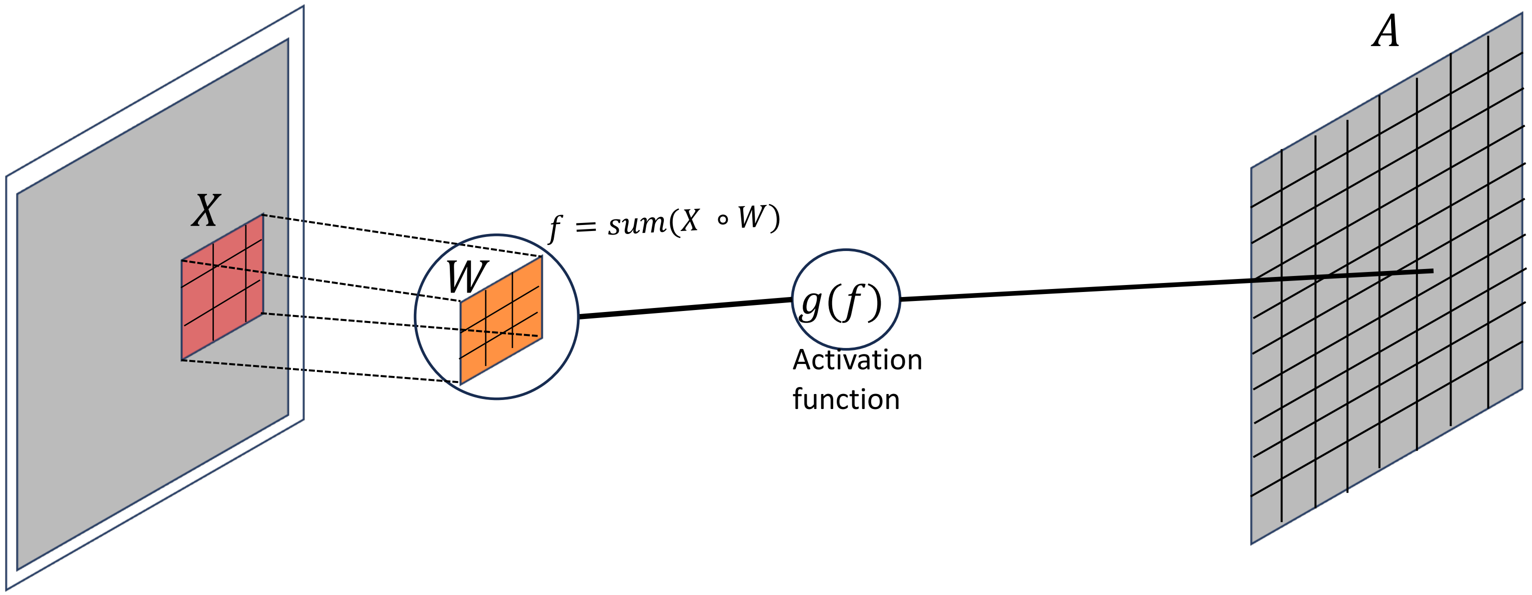
Convolution + Pooling Layer: Gray Scale Image



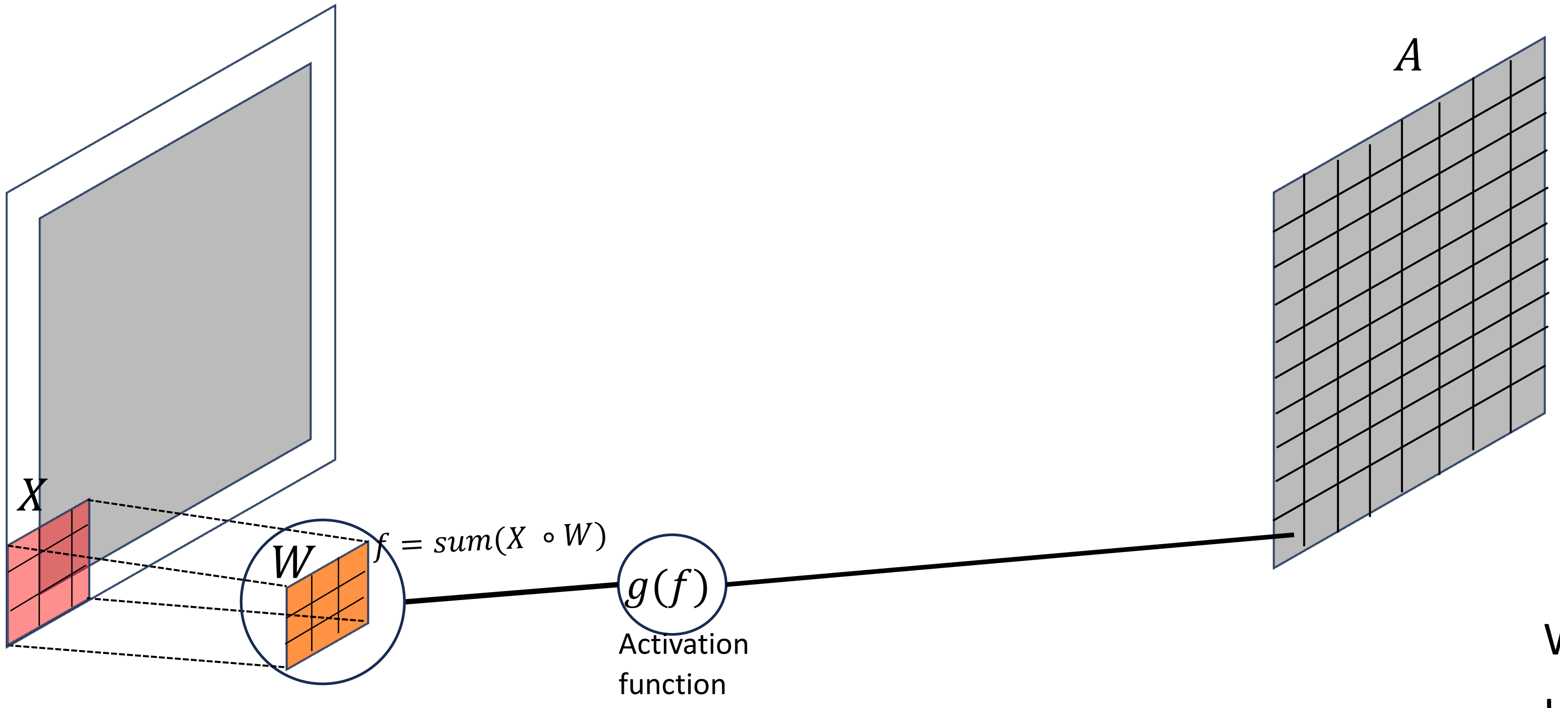
Convolution + Pooling Layer: Gray Scale Image



Convolution + Pooling Layer: Gray Scale Image



Convolution + Pooling Layer: Gray Scale Image



What is the shape of A?

How many parameters (weights)?

Size of Each Kernel’s Output

$$O = \left\lfloor \frac{I + 2P - K}{S} \right\rfloor + 1$$

where

O is the output dimension (either height or width).

I is the output dimension (either height or width).

$P = padding$

$K = kernel\_size$

$S = stride$

Usually, the convolution layer is designed to retain the height and width of input, i.e., $O = I$

Height and width of convolution layer’s output are independent of $in\_channels$ and $out\_channels$

```
nn.Conv2d(in_channels= 1,out_channels= 1, kernel_size=3, stride=1, padding=1)
```

Parameter Count for Convolution Layers

- # of Parameters = ($kernel\_size * kernel\_size * in\_channels * out\_channels$) + # of bias terms

- Each kernel (neuron) has one bias term
  - # of kernels = $out\_channels$
  - # of bias terms = $out\_channels$

- Parameter count is independent of $stride$ and $padding$

```
nn.Conv2d(in_channels= , out_channels= , kernel_size=3, stride=1, padding=1)
```

FLOPS for Convolution Layer

- FLOPS: Floating Point Operations

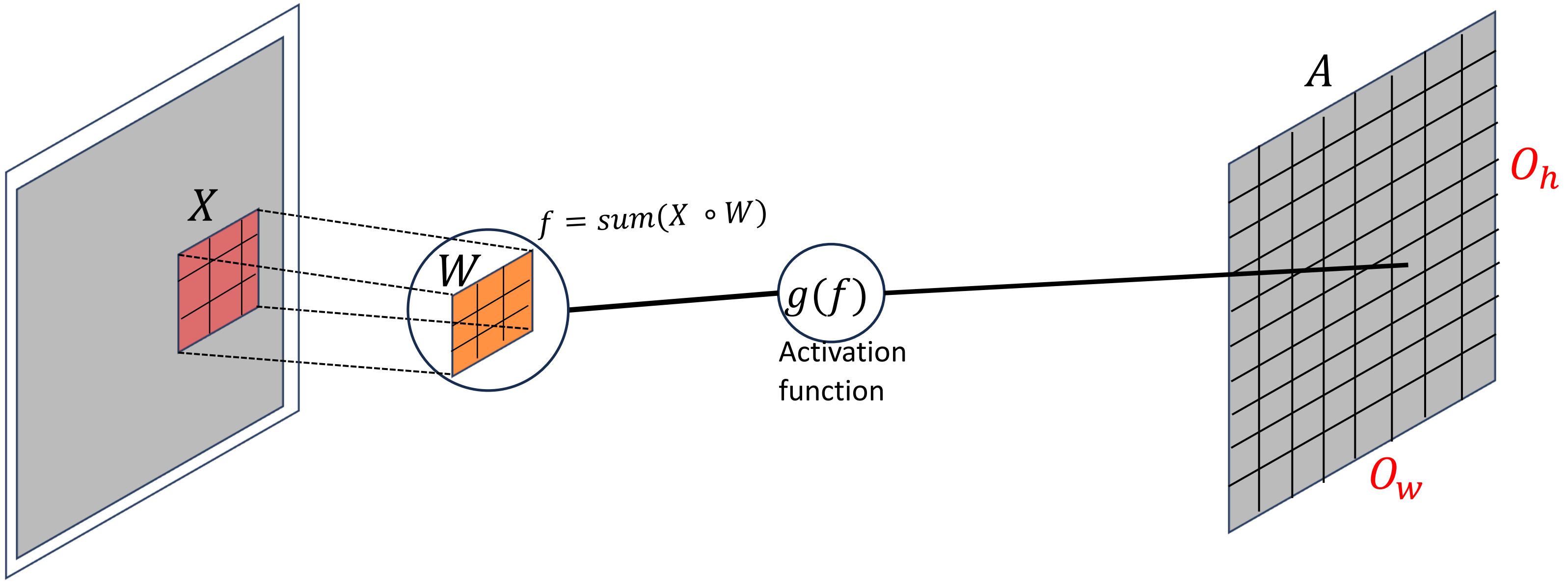
Output height and width

Kernel height and width

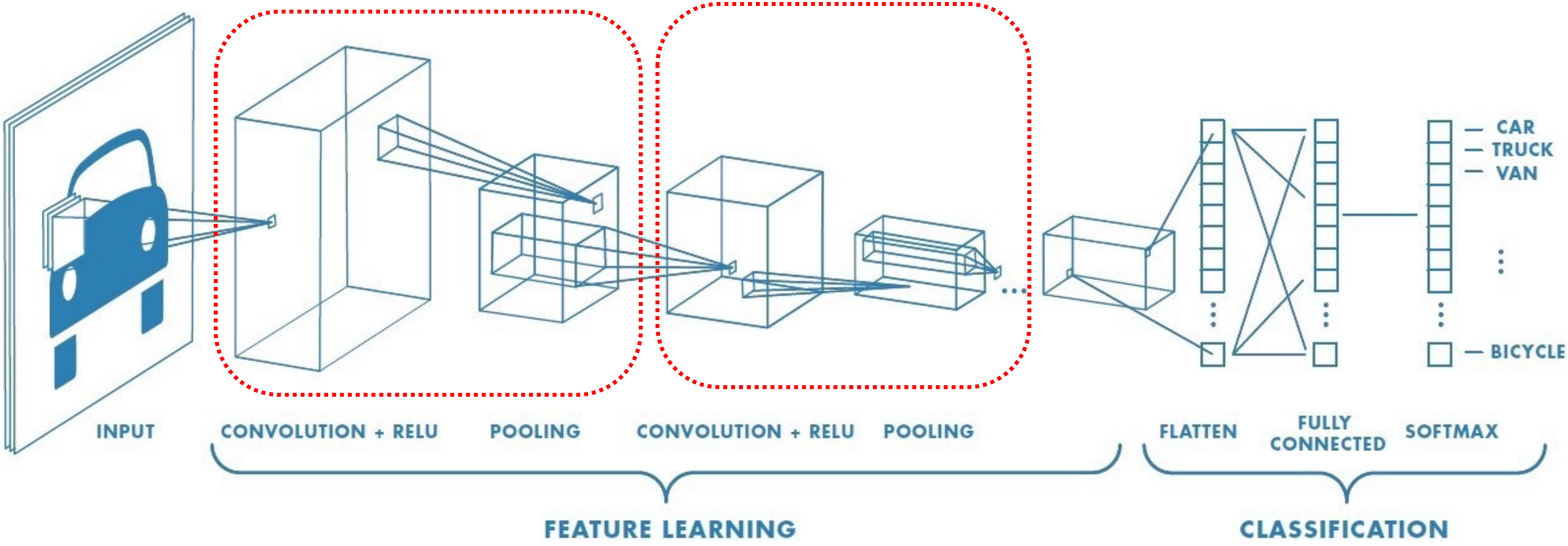
- # of FLOPS = $2 * out\_channels * O_h * O_w * in\_channels * k_h * k_w$

1 Multiplication and 1 Addition

```
nn.Conv2d(in_channels=1, out_channels= 1, kernel_size= 3 , stride=1, padding=1)
```



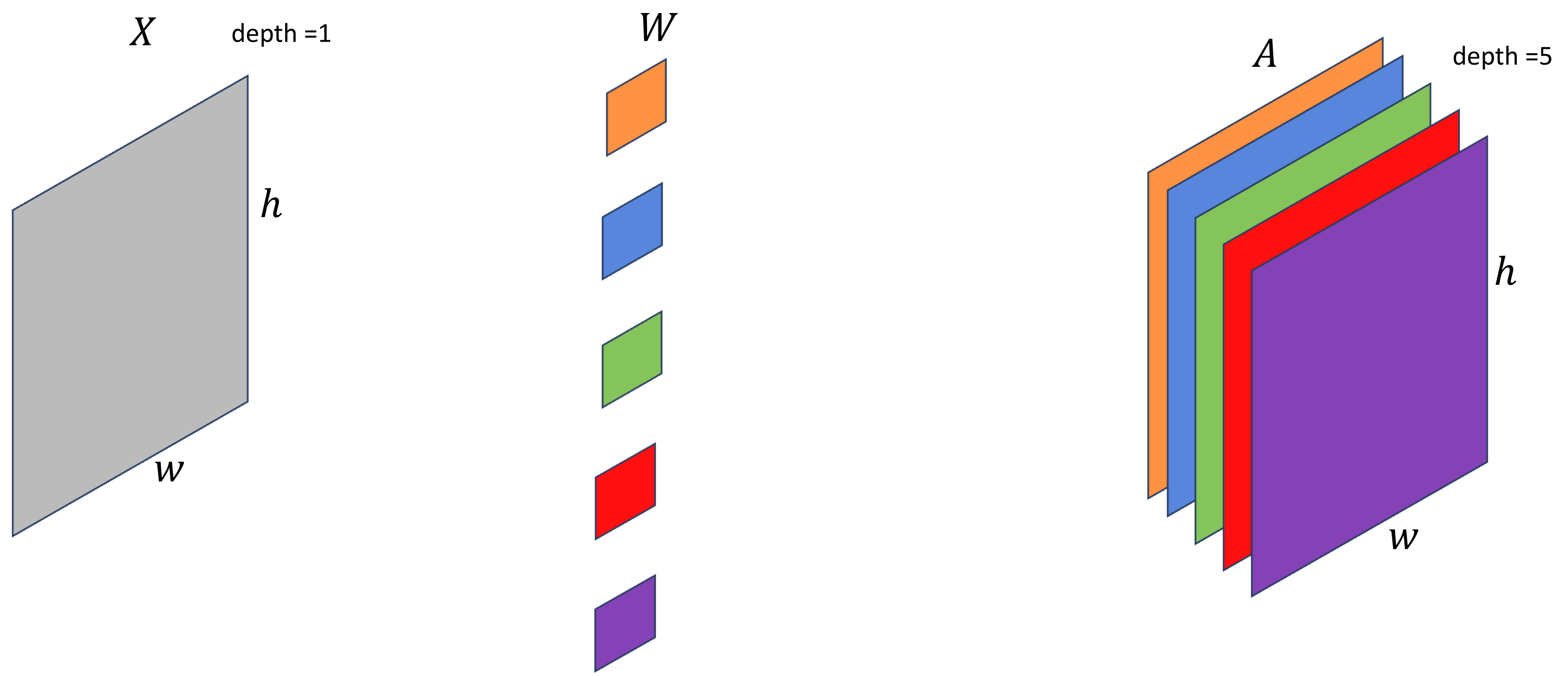
How to increase depth?



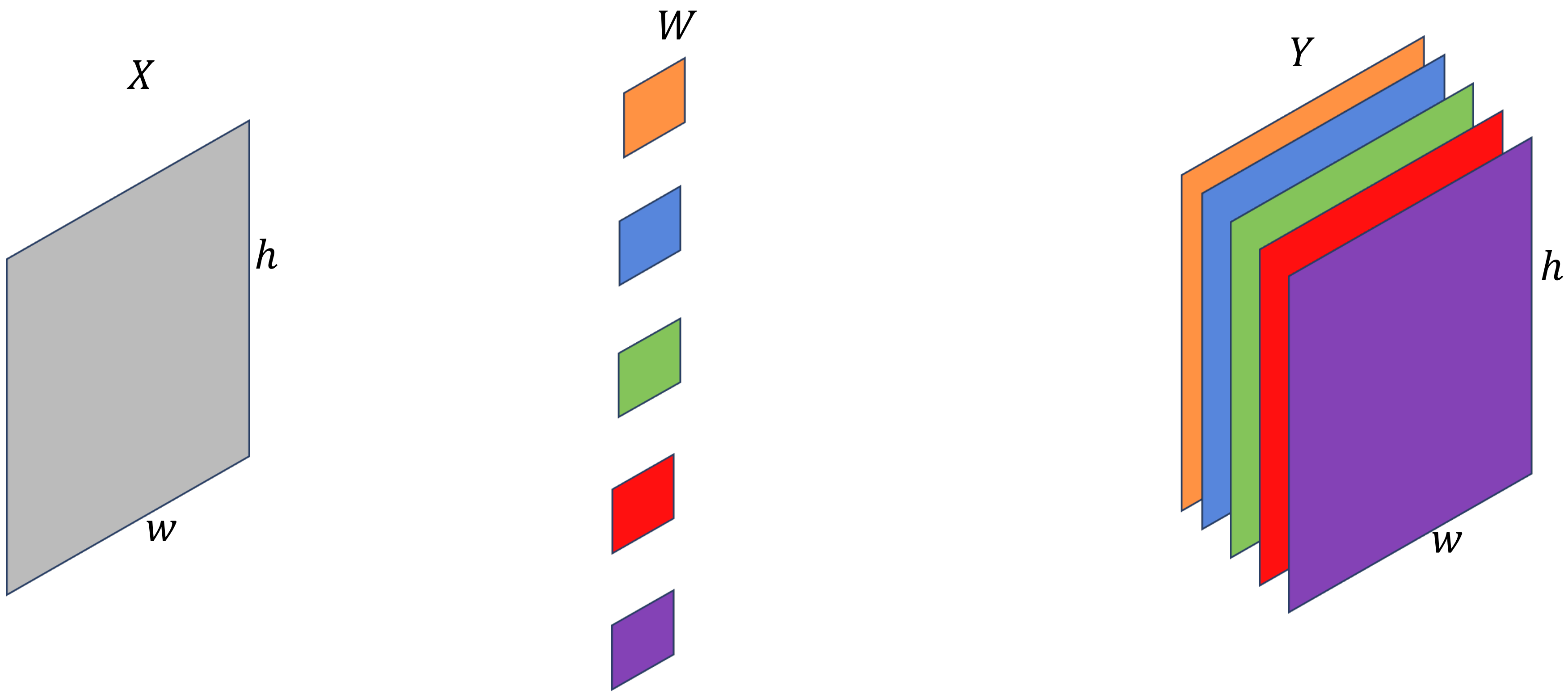
Convolution Layer: Gray Scale Image

How to increase the depth?

Multiple Kernels!!!



Convolution Layer



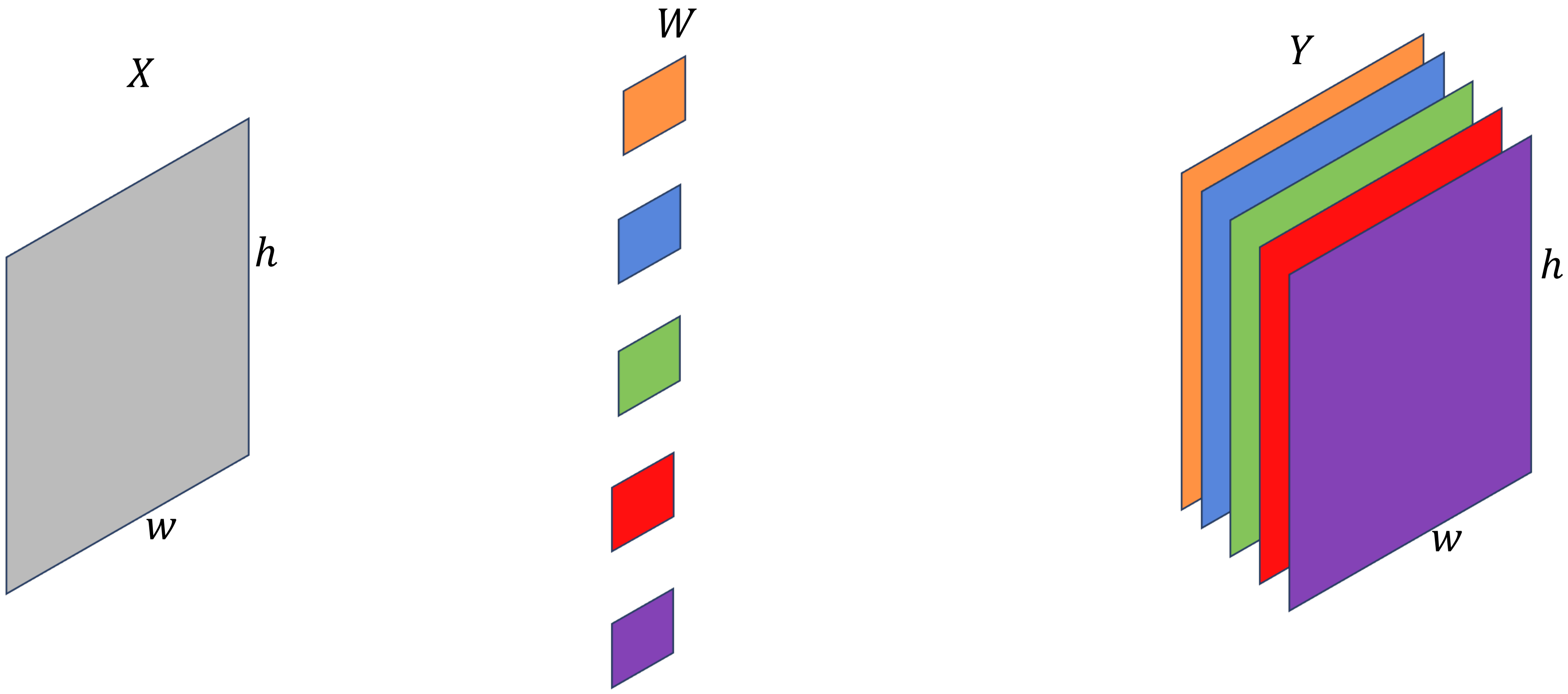

```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```

What is the dimension of each kernel?

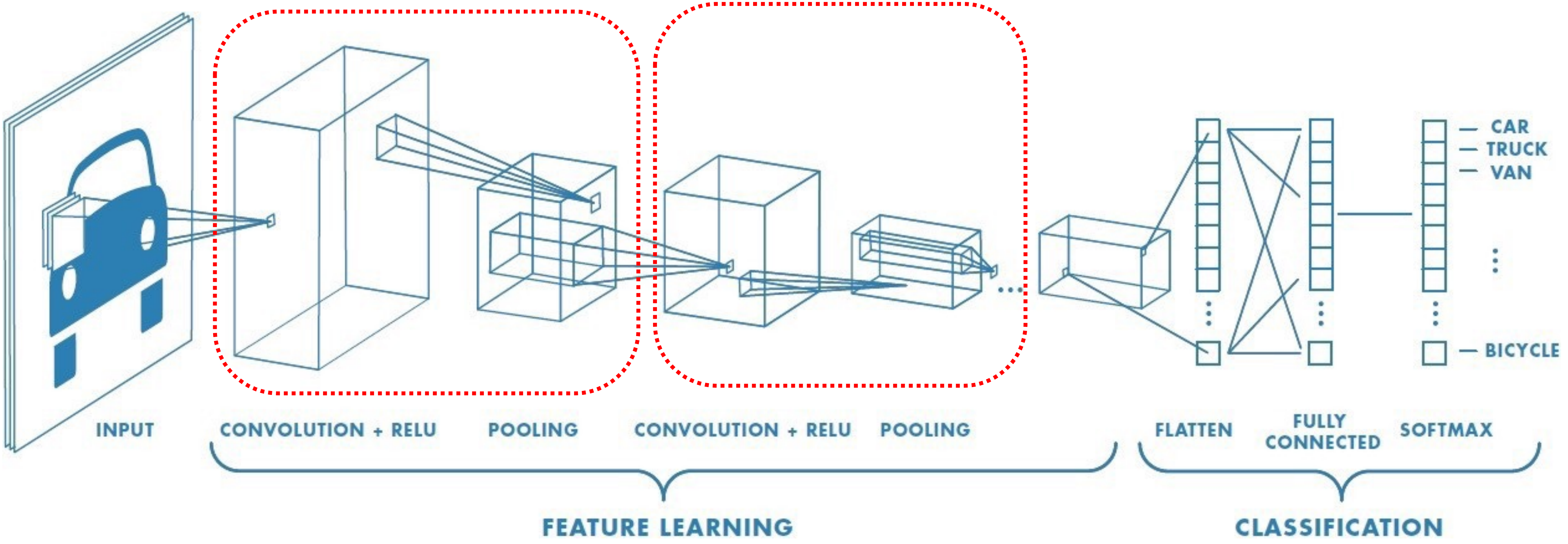
FLOPS for Convolution Layer

```
nn.Conv2d(in_channels=1, out_channels= 5, kernel_size= 3 , stride=1, padding=1)
```

# of FLOPS = $2 * out\_channels * O_h * O_w * in\_channels * k_h * k_w$



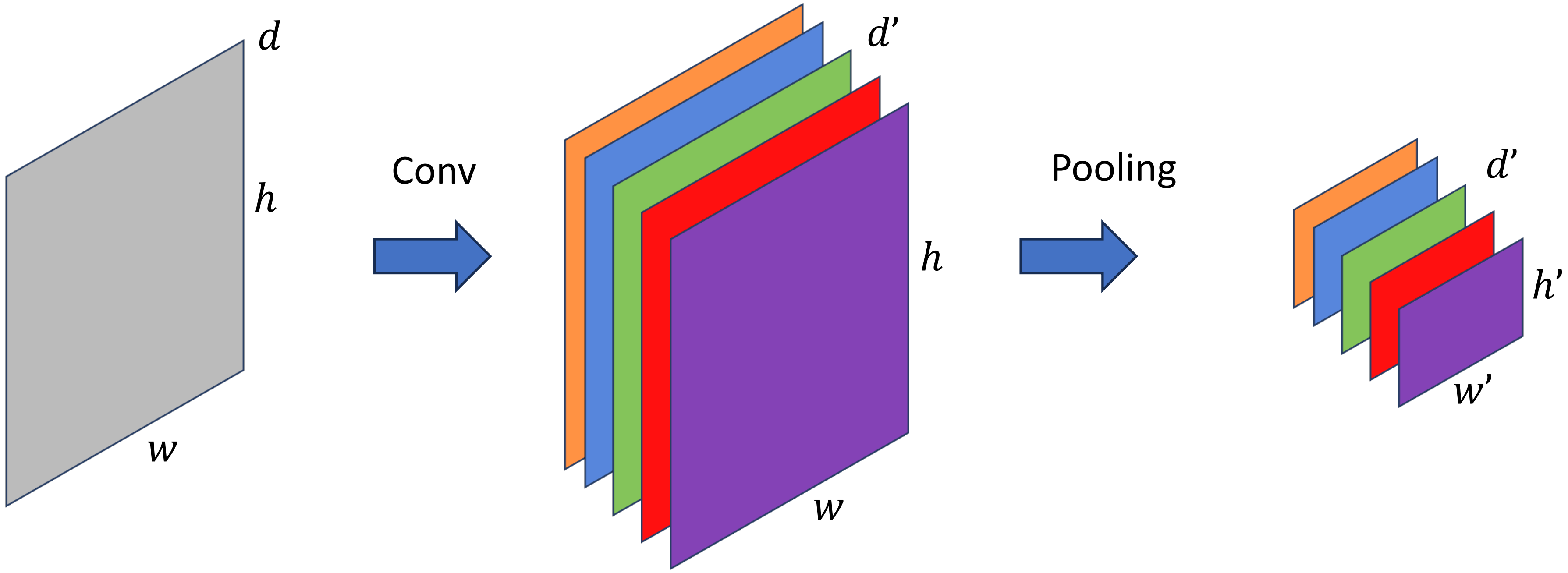
How to reduce height and width?
Why should we reduce height and width?



# of FLOPS $= 2 * out\_channels * O_h * O_w * in\_channels * k_h * k_w$

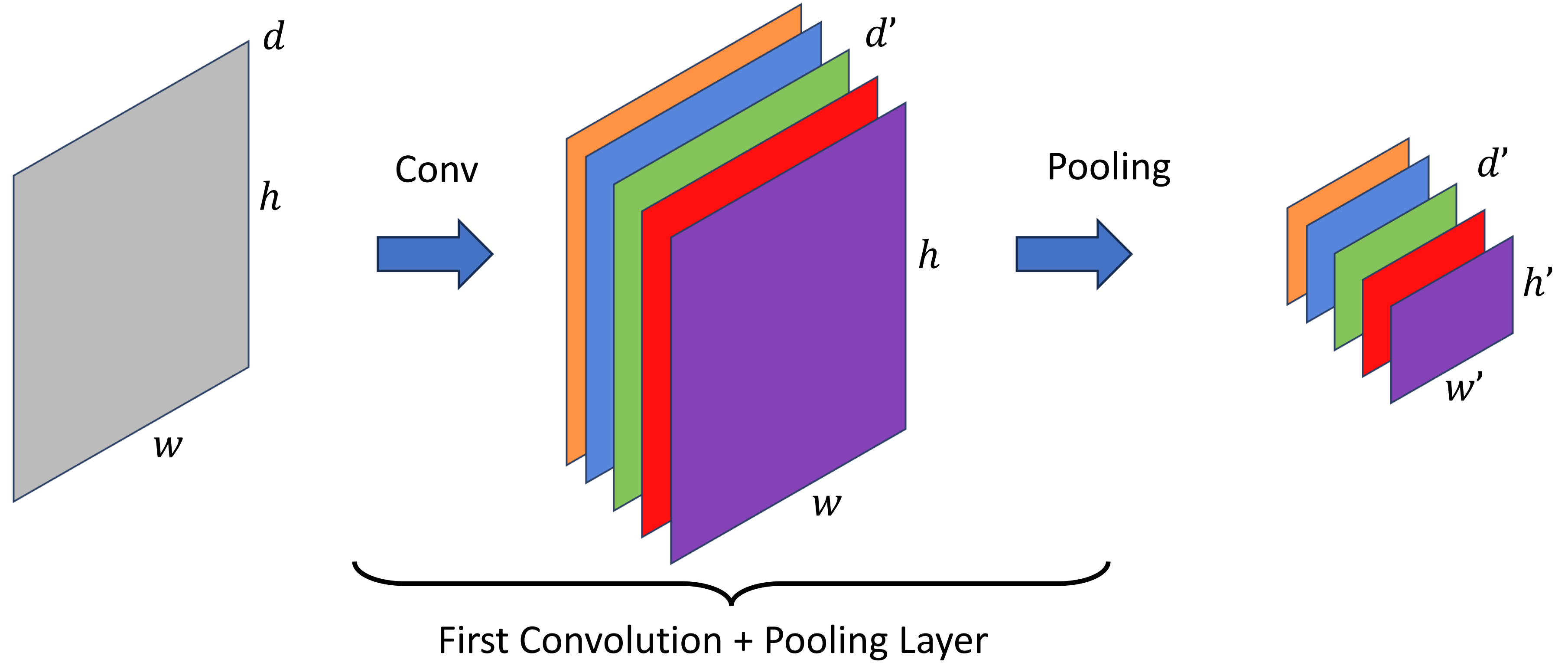
How to reduce height and width?

$$h' = \frac{h}{2} \qquad w' = \frac{w}{2}$$



How to reduce height and width?

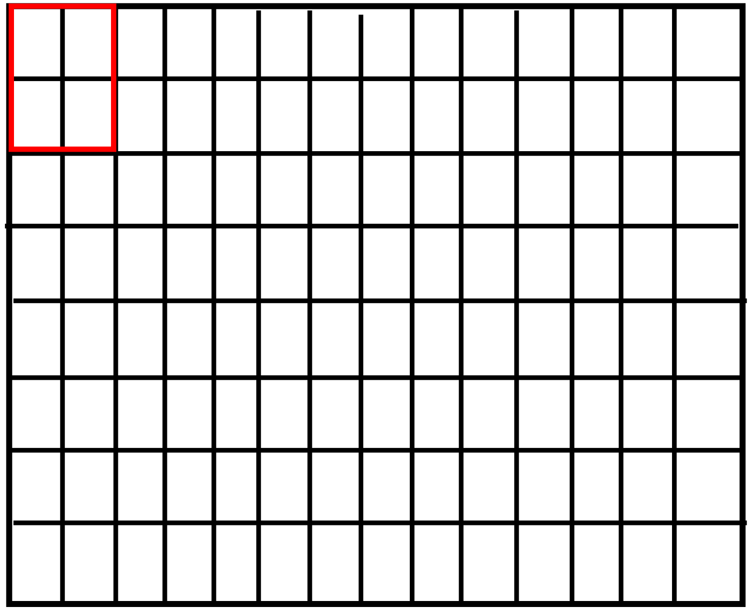
$$h' = \frac{h}{2} \qquad w' = \frac{w}{2}$$

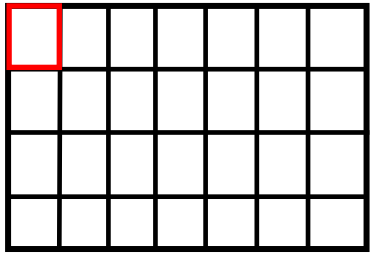



```
nn.Conv2d(in_channels=1, out_channels= 5, kernel_size= 3 , stride=1, padding=1)
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

Max Pooling

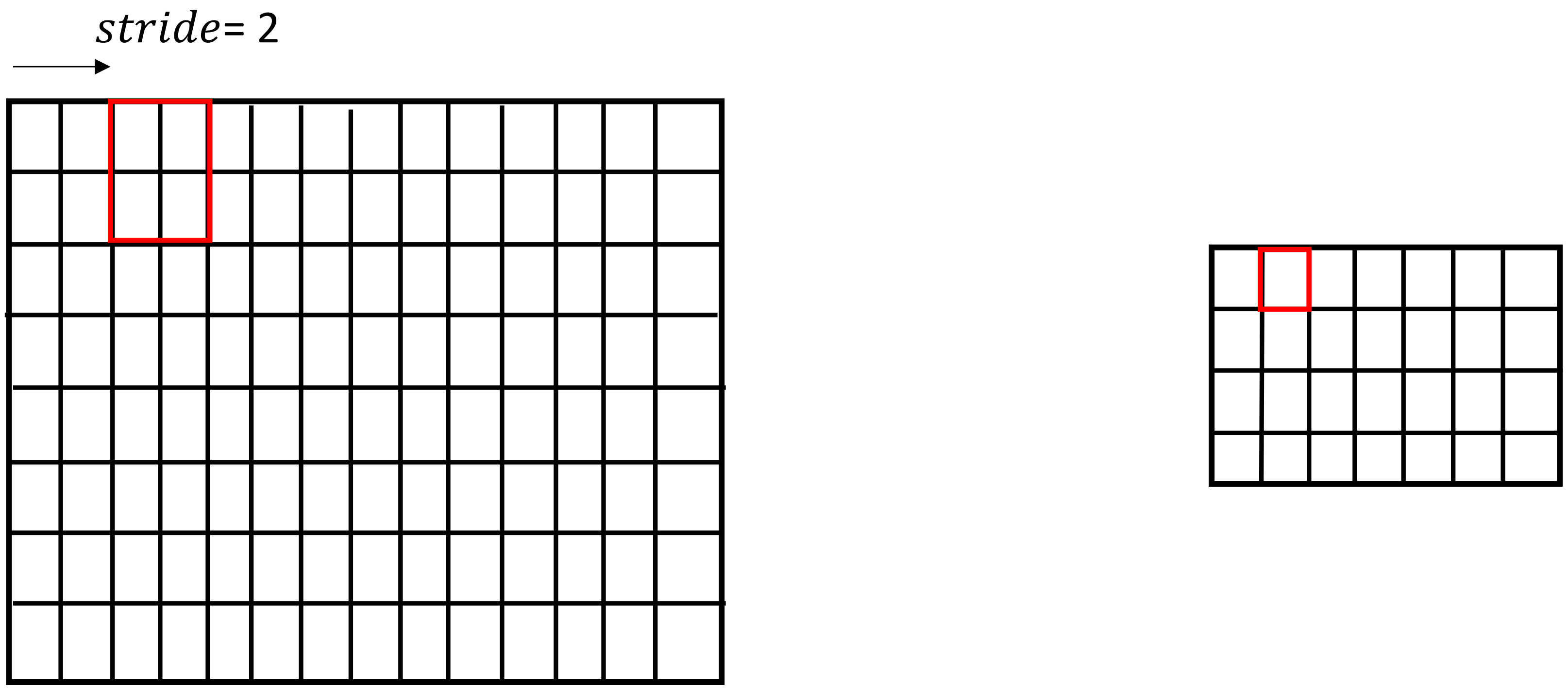
$kernel\_size = 2$





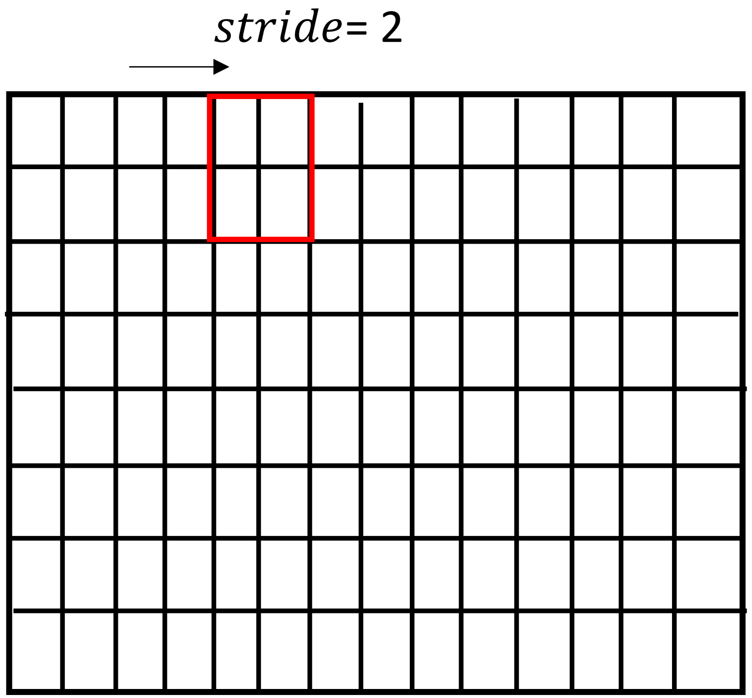
```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

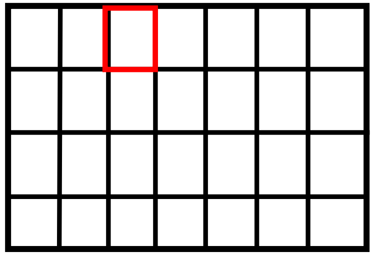
Max Pooling




```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

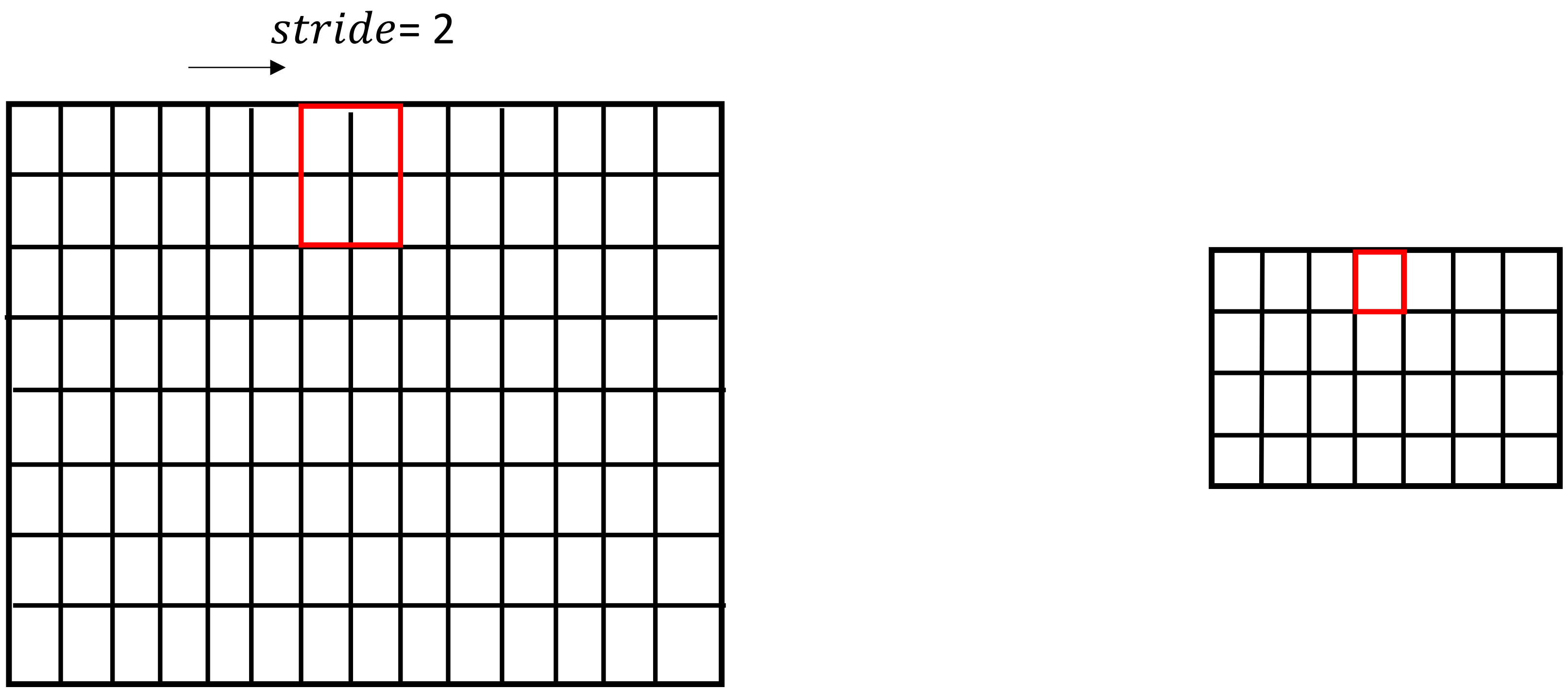
Max Pooling





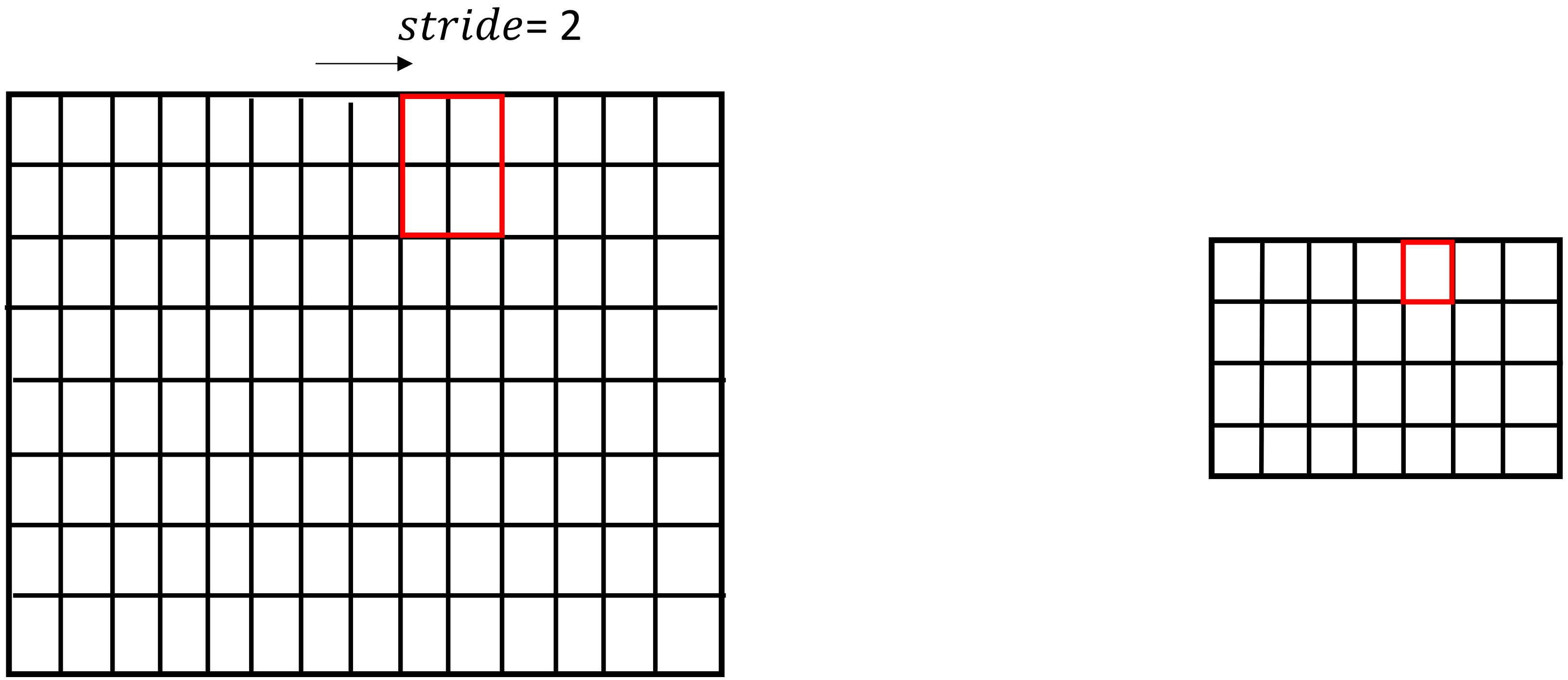
```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

Max Pooling



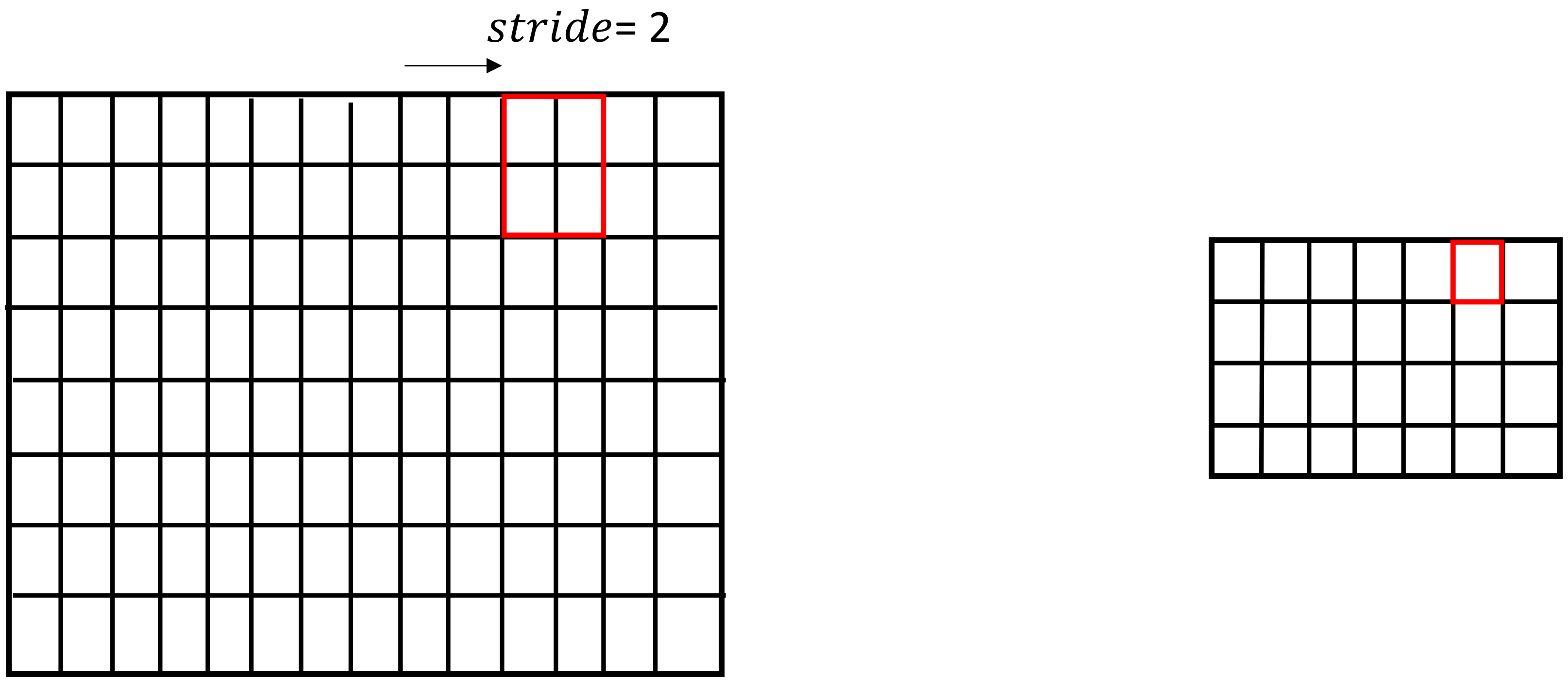

```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

Max Pooling



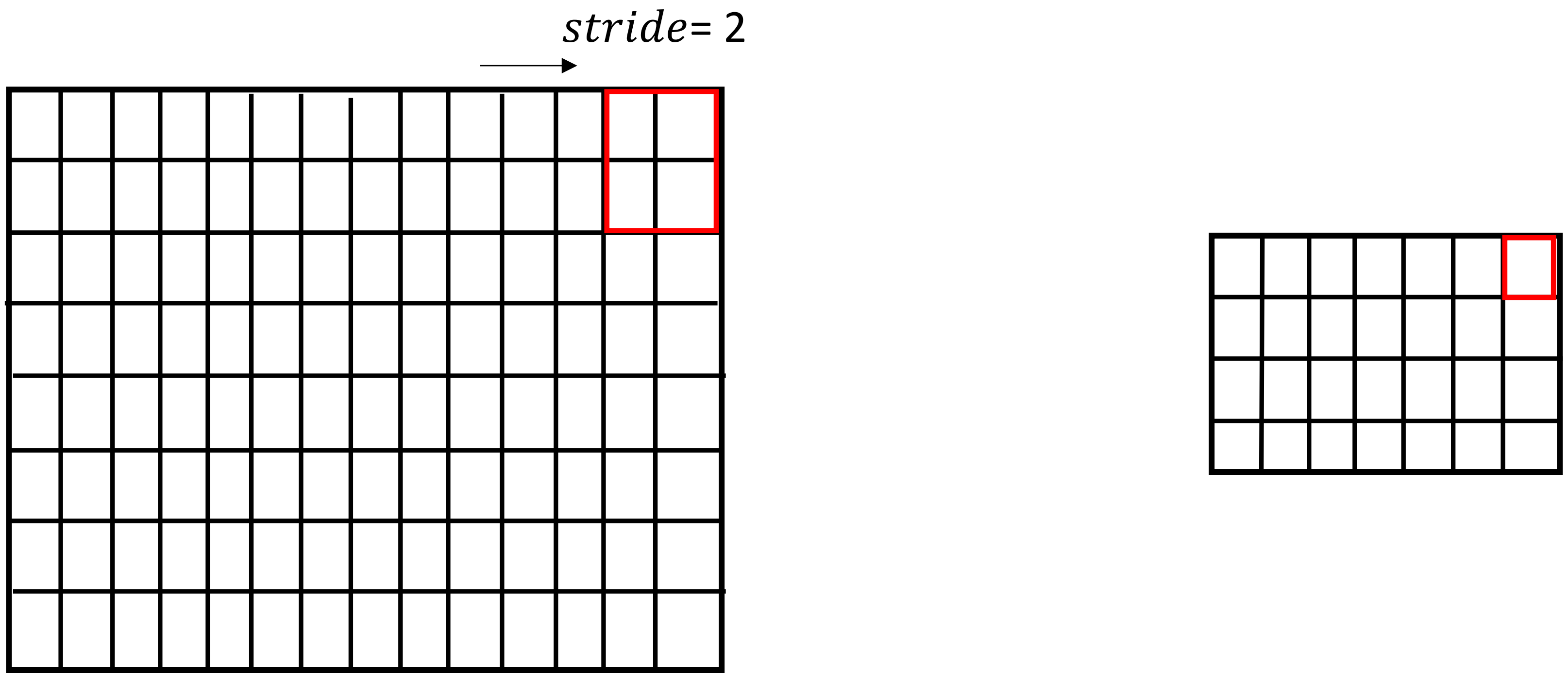

```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

Max Pooling



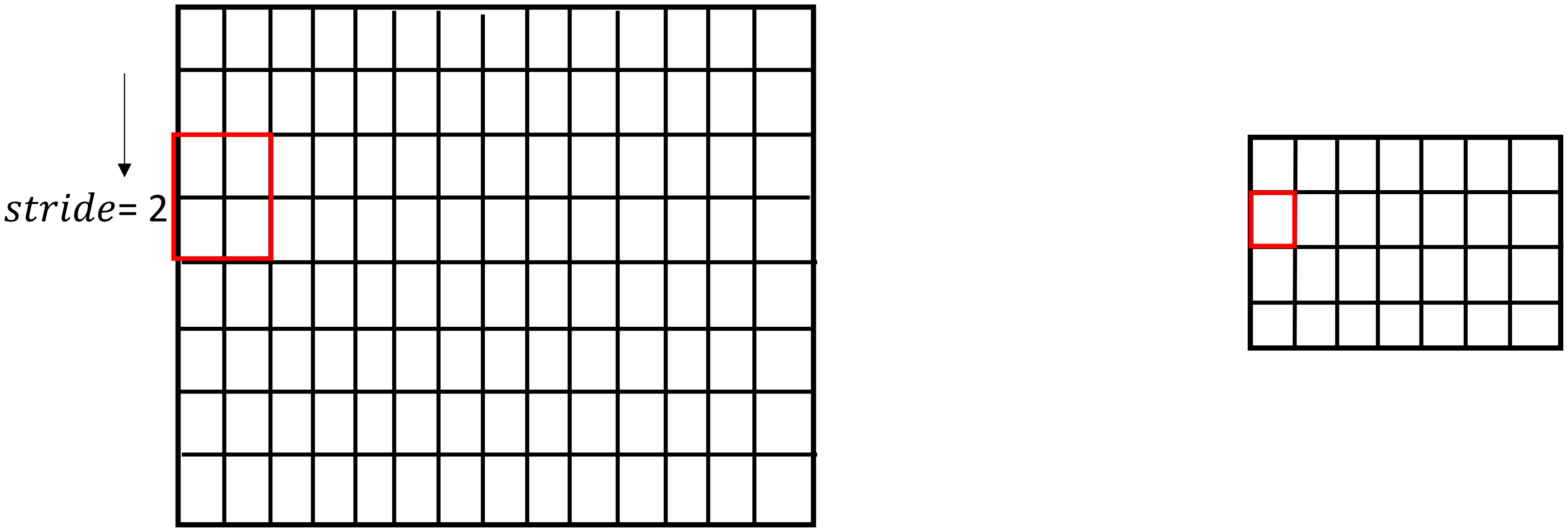

```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

Max Pooling




```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

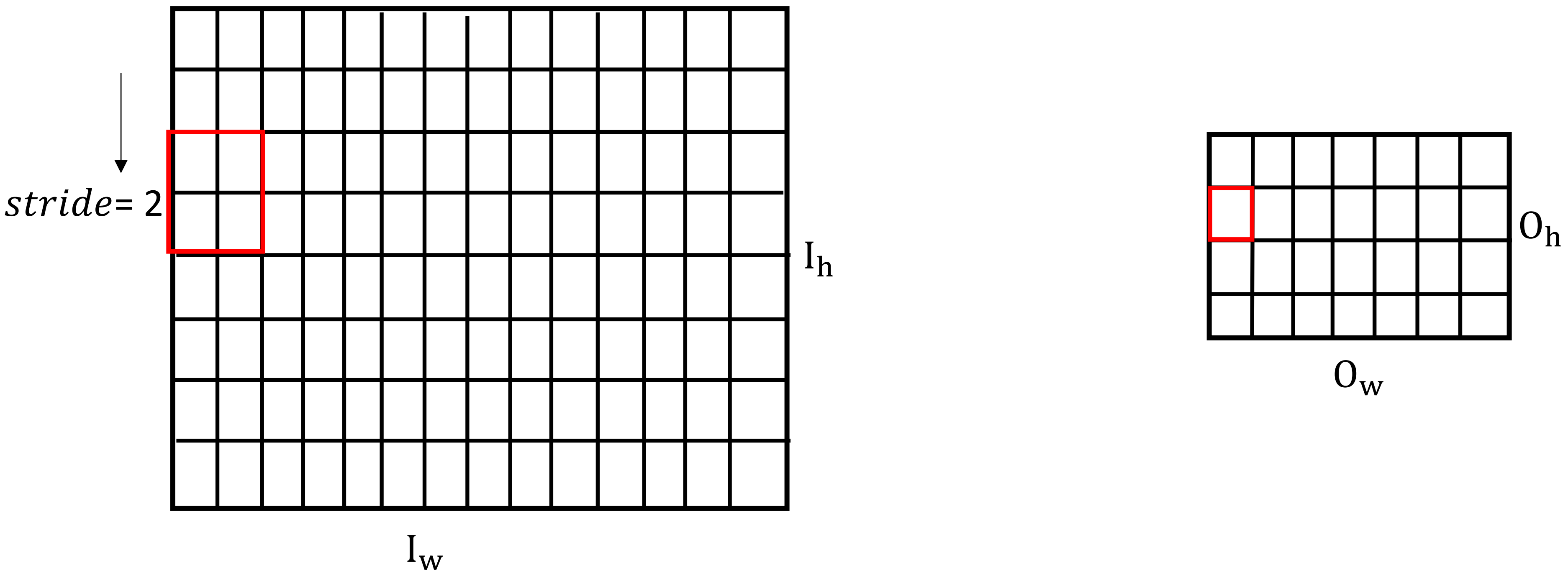
Max Pooling




```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

Output Size of Max Pool layer

- $O_h = \left\lfloor \frac{I_h + 2p - k}{s} \right\rfloor + 1$
- $O_w = \left\lfloor \frac{I_w + 2p - k}{s} \right\rfloor + 1$

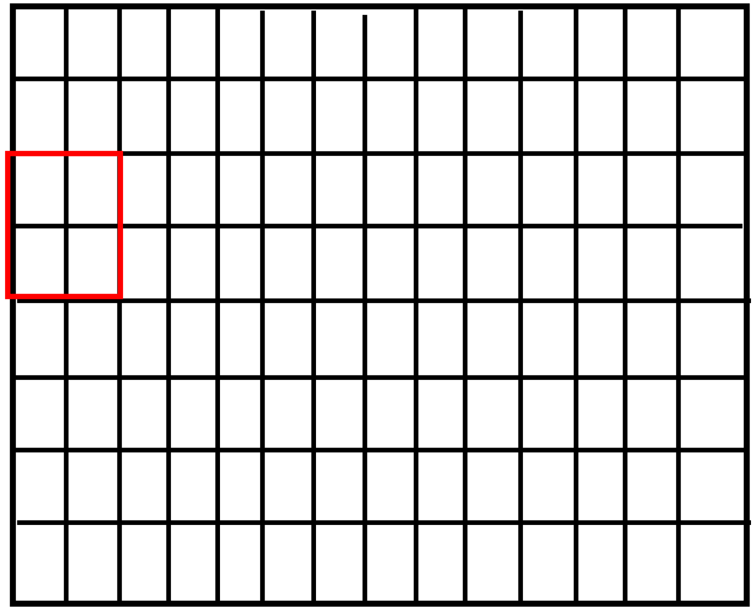


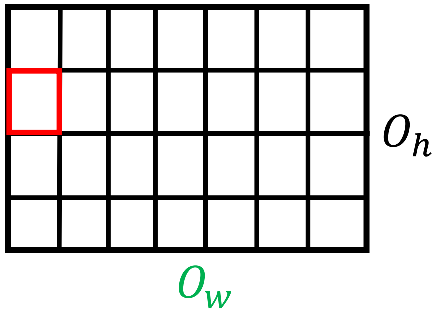

```
max_pool_2D = nn.MaxPool2d(kernel_size = 2,stride = 2)
```

FLOPS for MaxPool Layer

- FLOPS = $\underbrace{in\_channels}_{\text{\# of channels}} * \underbrace{O_h}_{\text{Output height}} * \overbrace{O_w}^{\text{Output width}} * \underbrace{(k^2 - 1)}_{\text{\# of comparisons}}$

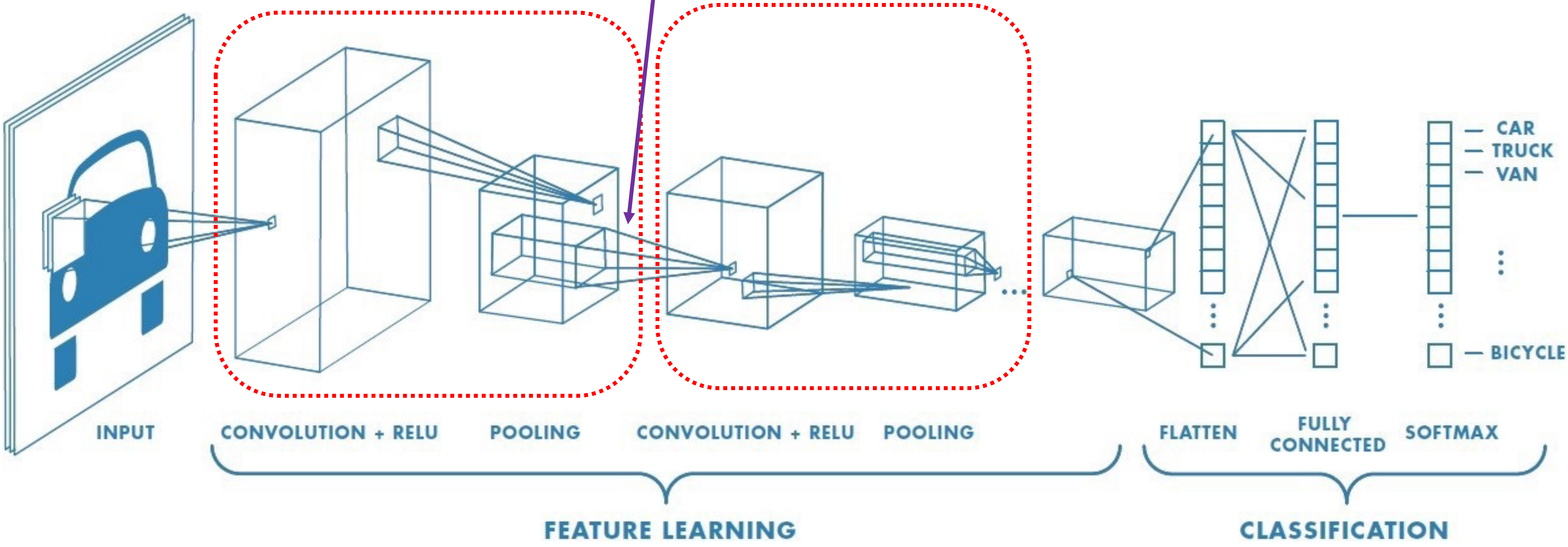
- k is the kernel size





FLOPS: Floating Point Operations

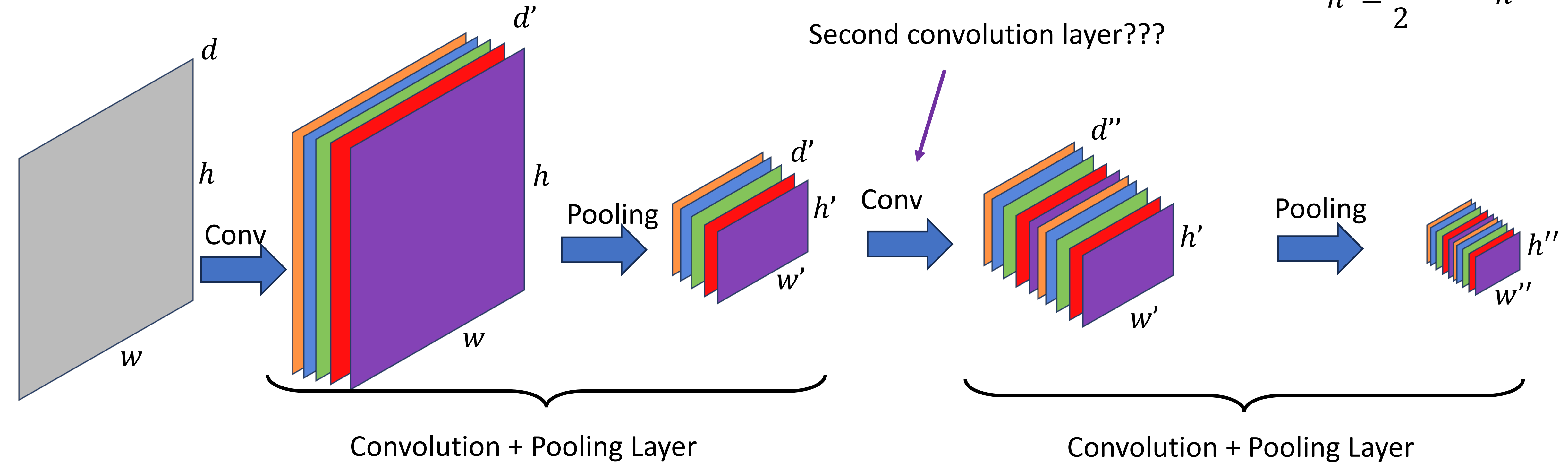
Second convolution layer???



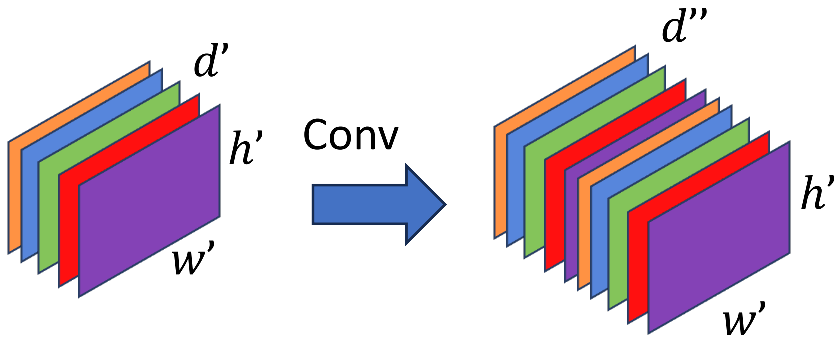
Adding another Conv+Pooling Layer

$$w' = \frac{w}{2} \qquad w'' = \frac{w'}{2}$$

$$h' = \frac{h}{2} \qquad h'' = \frac{w'}{2}$$



Adding another Conv + Pooling Layer



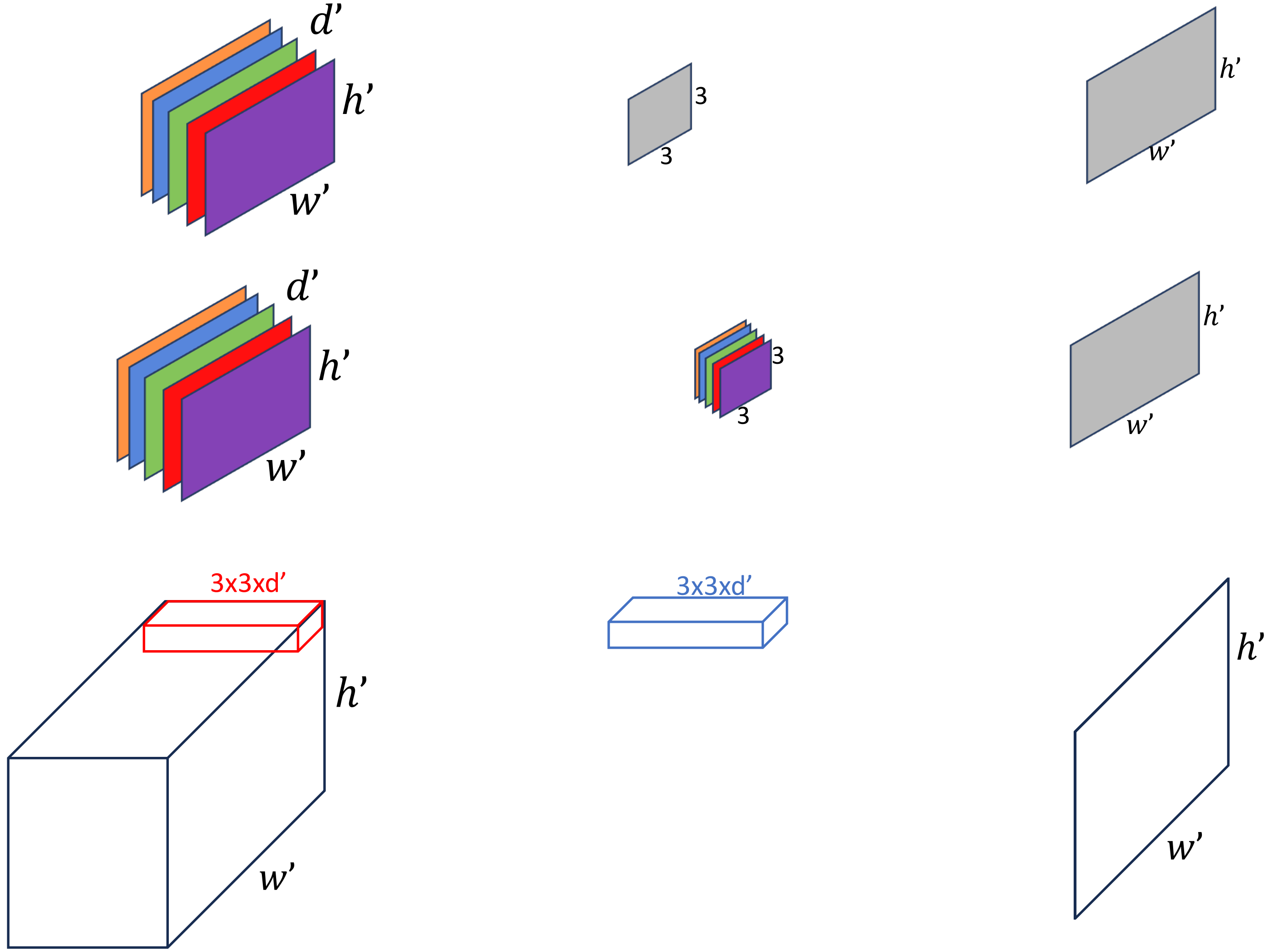
$$w' = \frac{w}{2}$$

$$h' = \frac{h}{2}$$

Questions:
How many kernels?
What is the shape of each kernel?

Adding another Conv+Pooling Layer

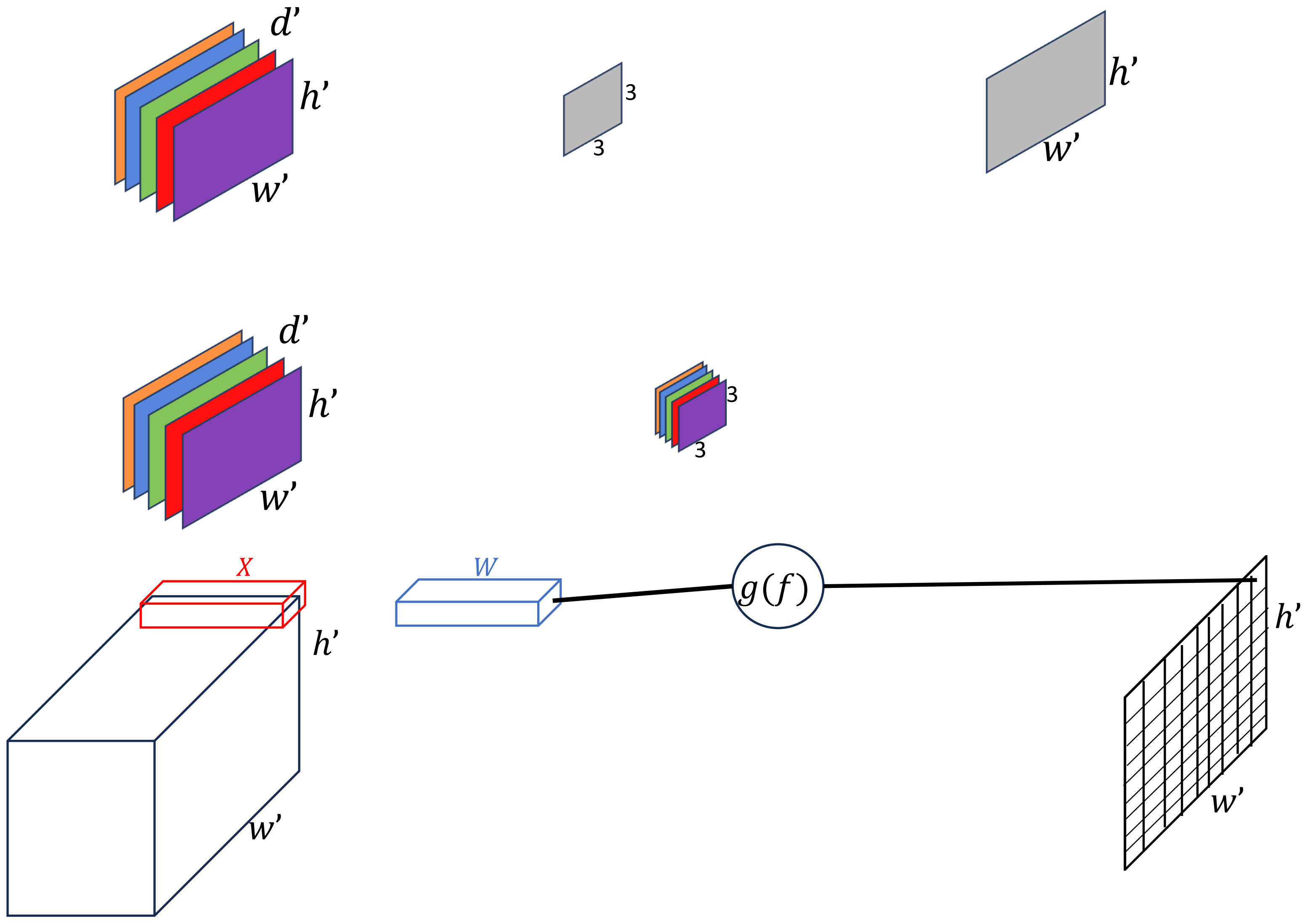
`nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)`



Shape of the kernel:______

Adding another Conv+Pooling Layer

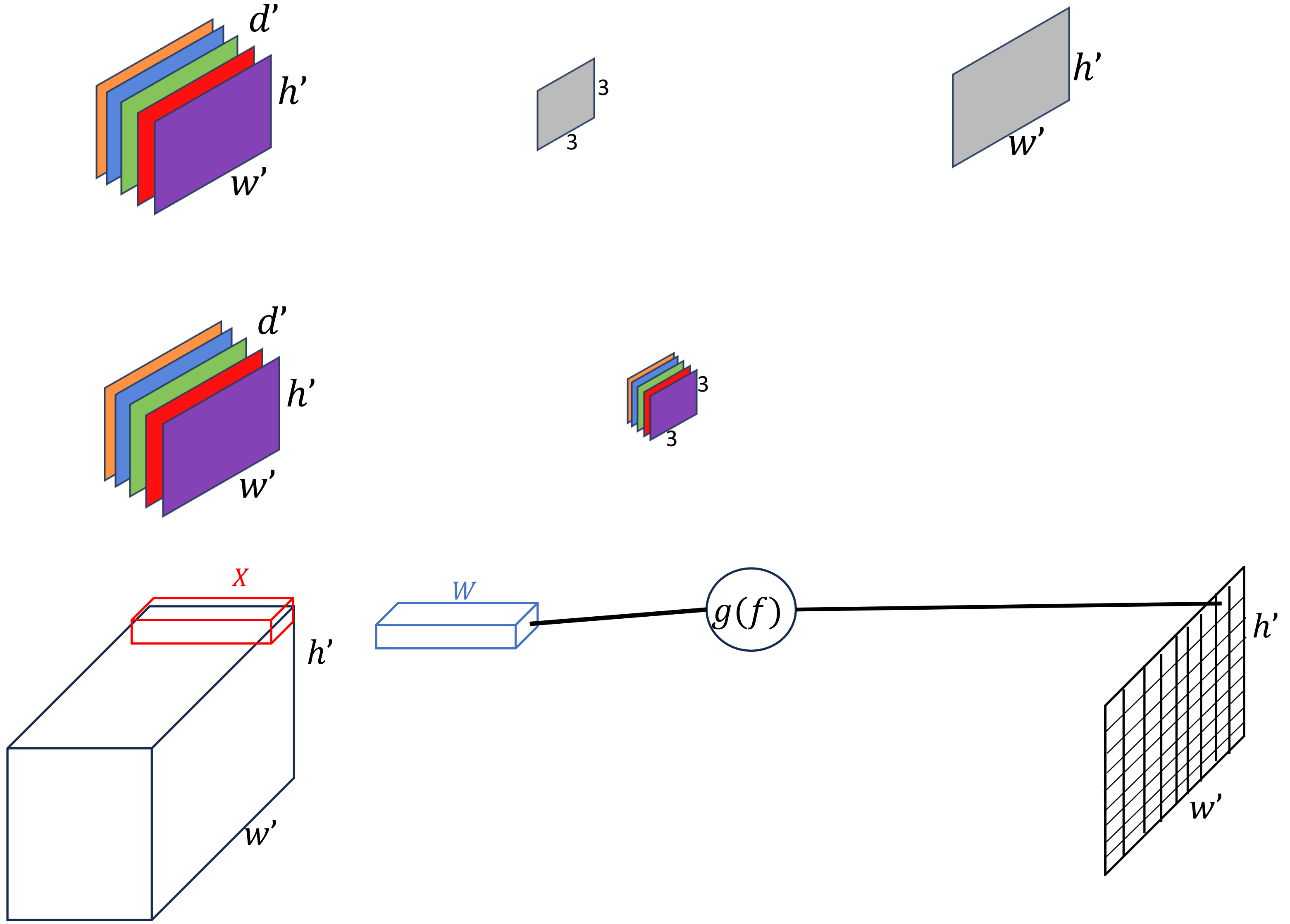
```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```



$f = sum(X \circ W)$

Adding another Conv+Pooling Layer

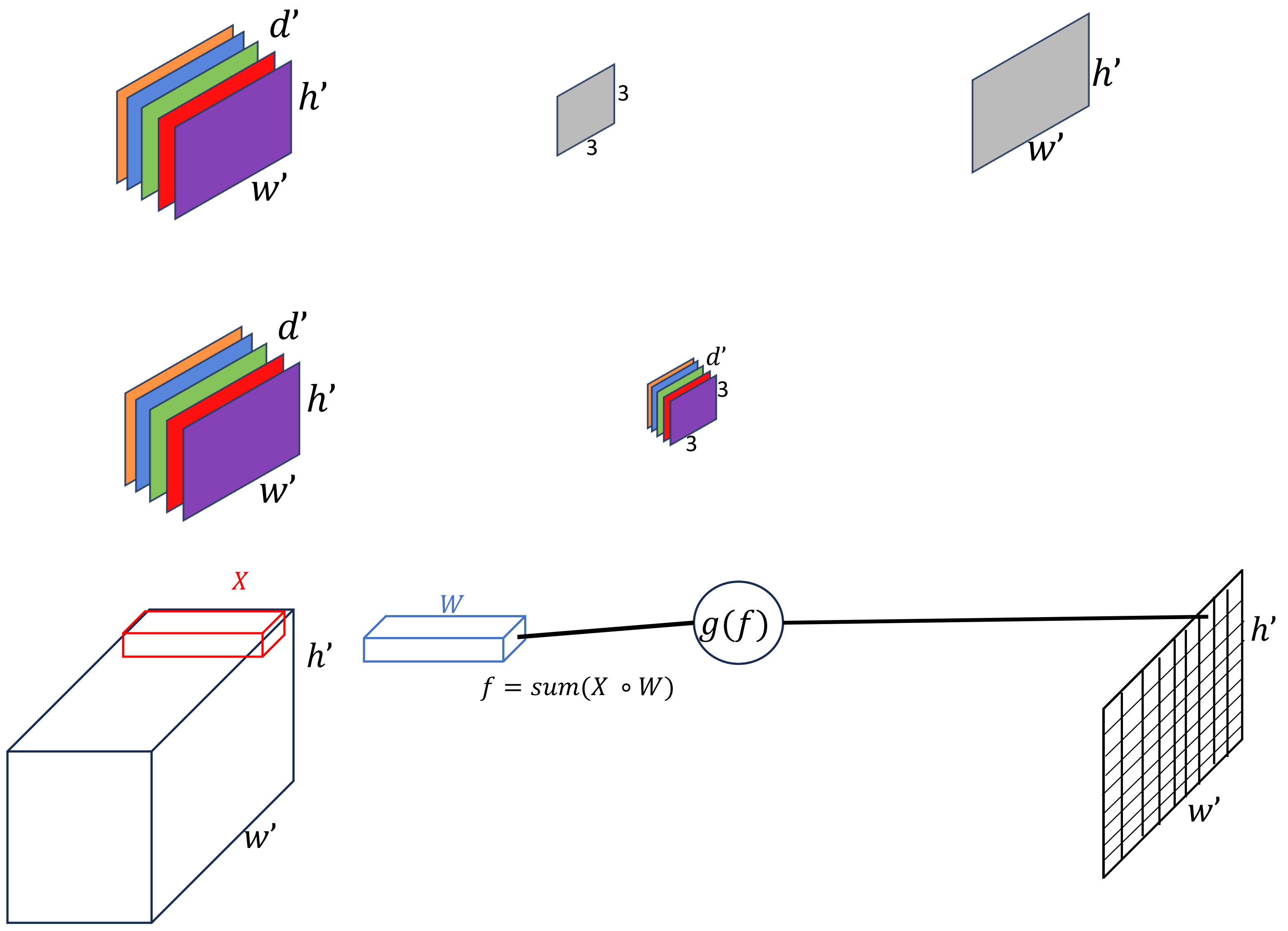
```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```



$f = sum(X \circ W)$

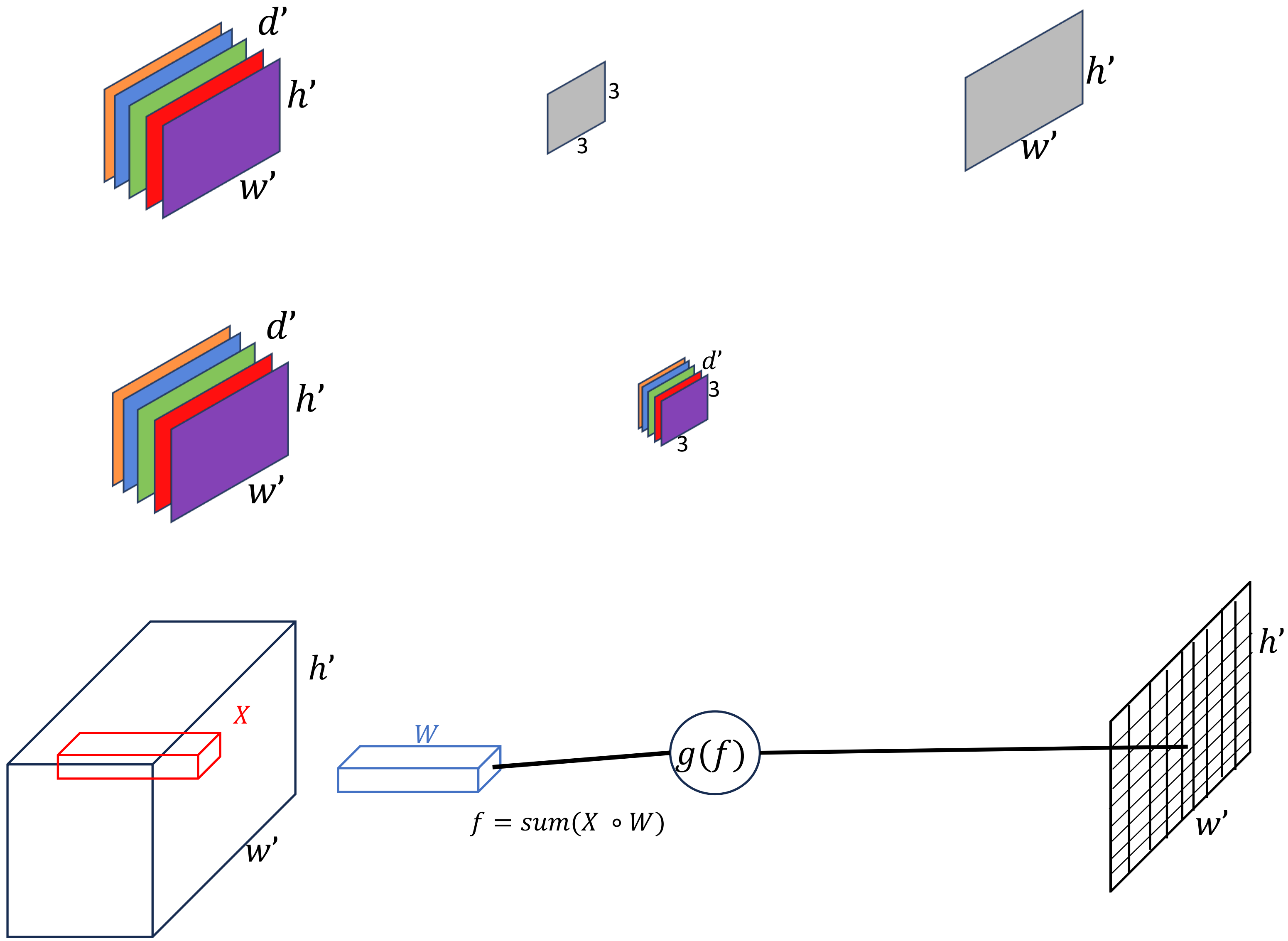
Adding another Conv+Pooling Layer

```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```



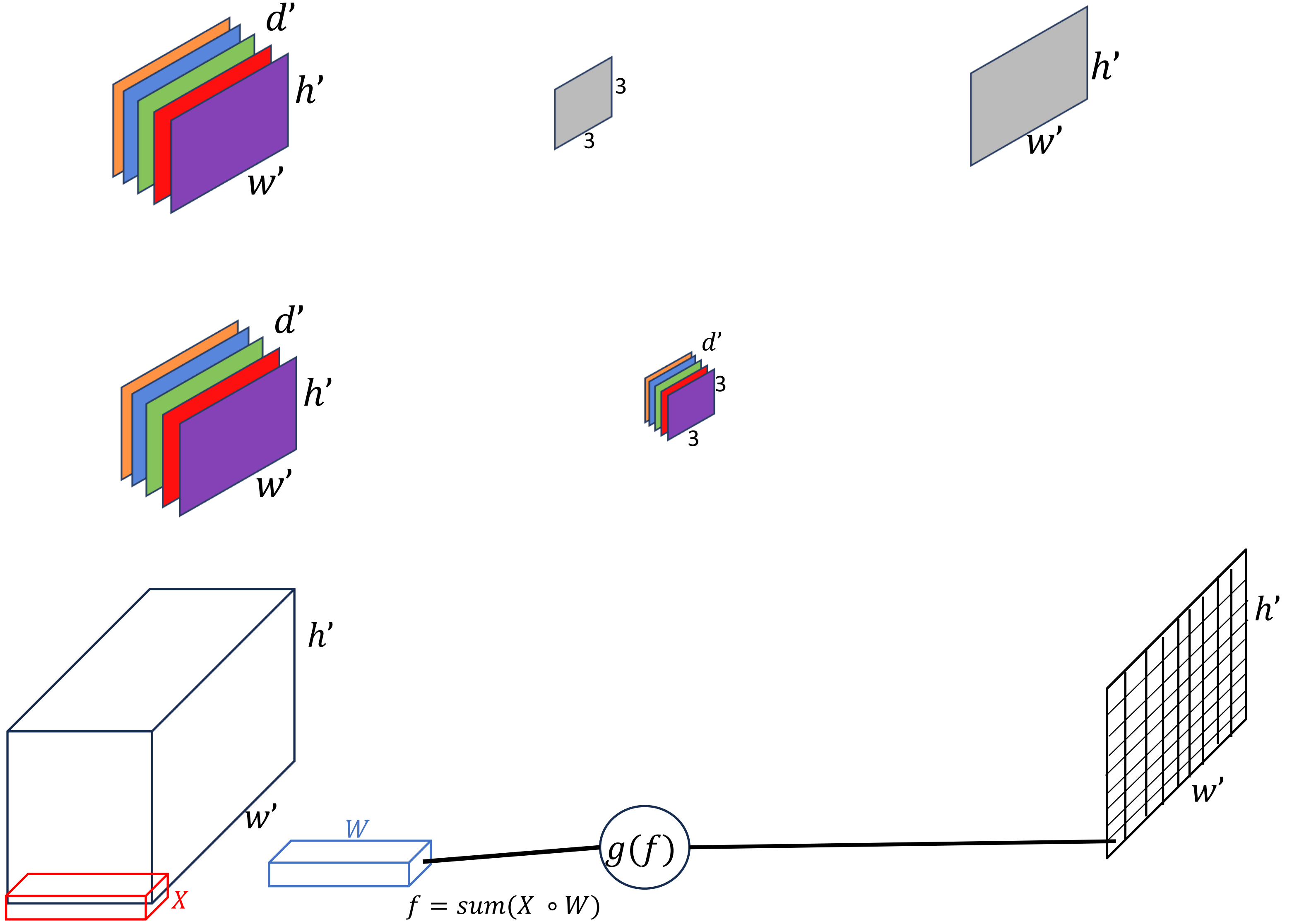
Adding another Conv+Pooling Layer

```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```



Adding another Conv+Pooling Layer

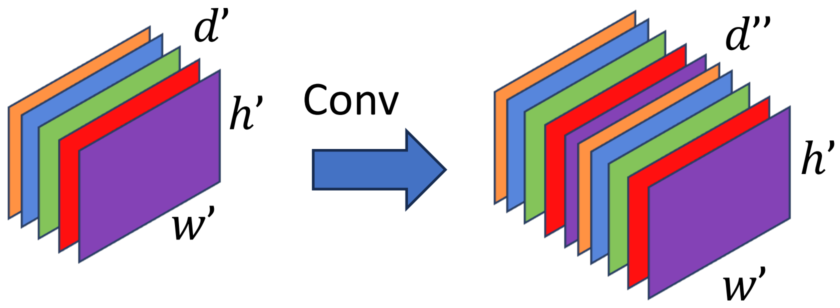
```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```



Adding another Conv+Pooling Layer

$$w' = \frac{w}{2}$$

$$h' = \frac{h}{2}$$

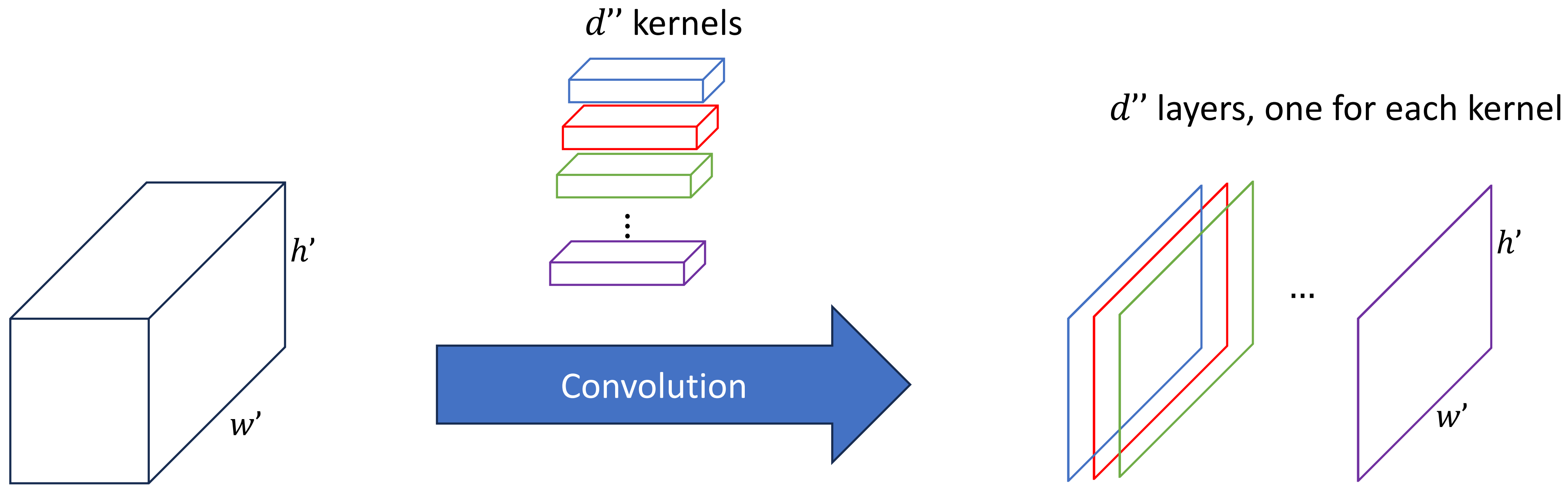



```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```

Questions:
How many kernels?
What is the shape of each kernel?

Adding another Conv+Pooling Layer

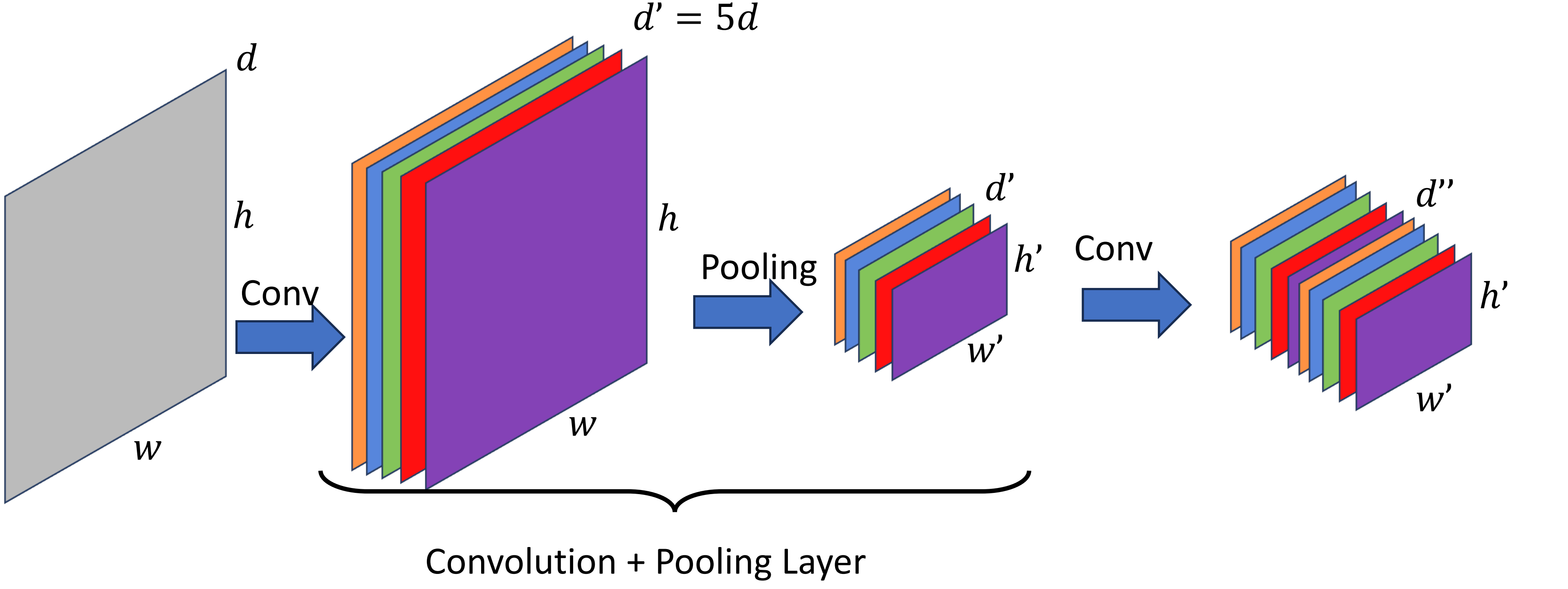
```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)
```



Adding another Conv+Pooling Layers

$$w' = \frac{w}{2}$$

$$h' = \frac{h}{2}$$

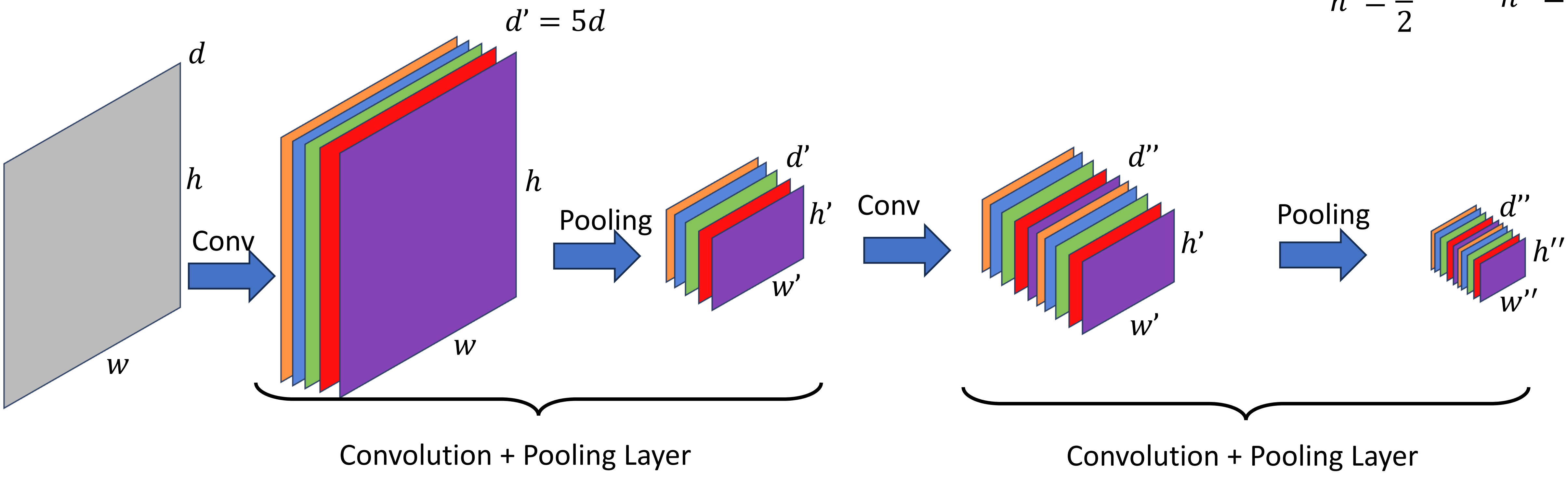


Questions:
How many FLOPS?

Adding another Conv+Pooling Layer

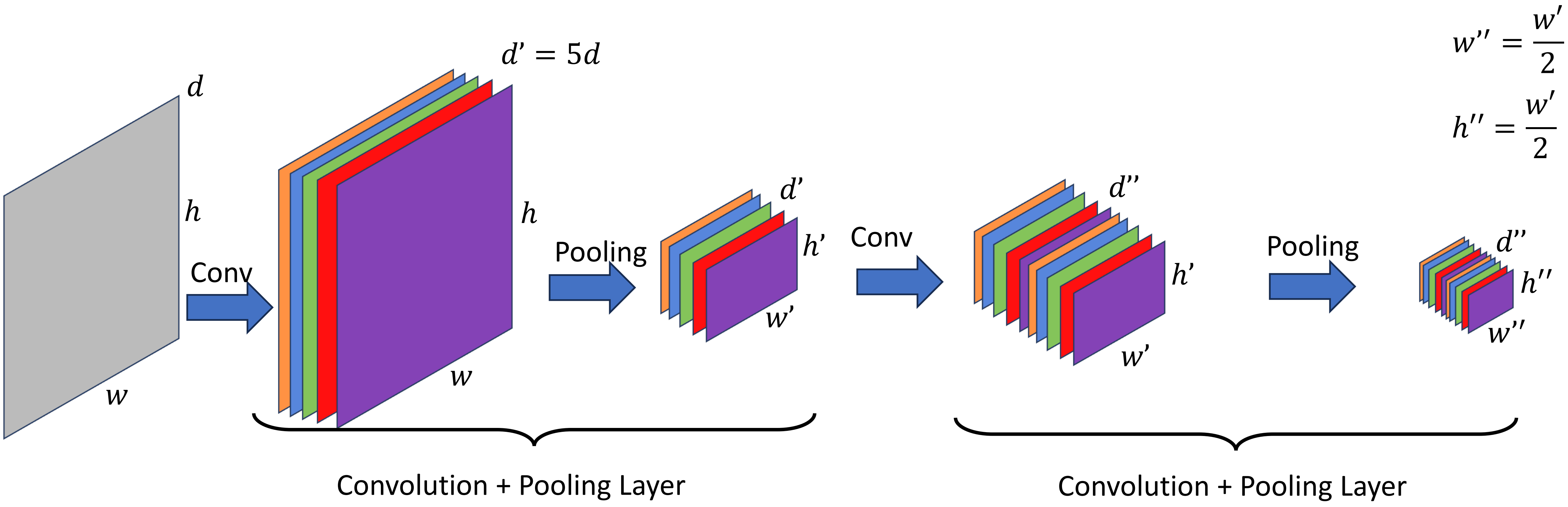
$$w' = \frac{w}{2} \qquad w'' = \frac{w'}{2}$$

$$h' = \frac{h}{2} \qquad h'' = \frac{w'}{2}$$



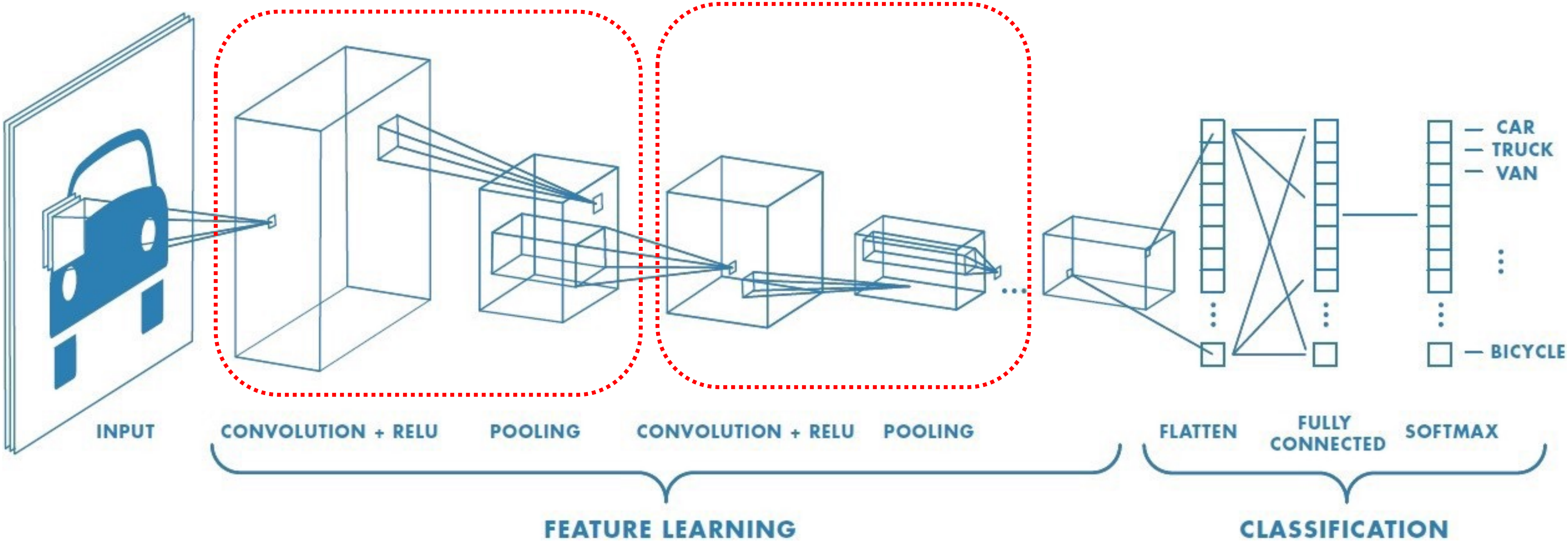

```
nn.MaxPool2d(kernel_size=__, stride=__)
```

Adding another Conv+Pooling Layer

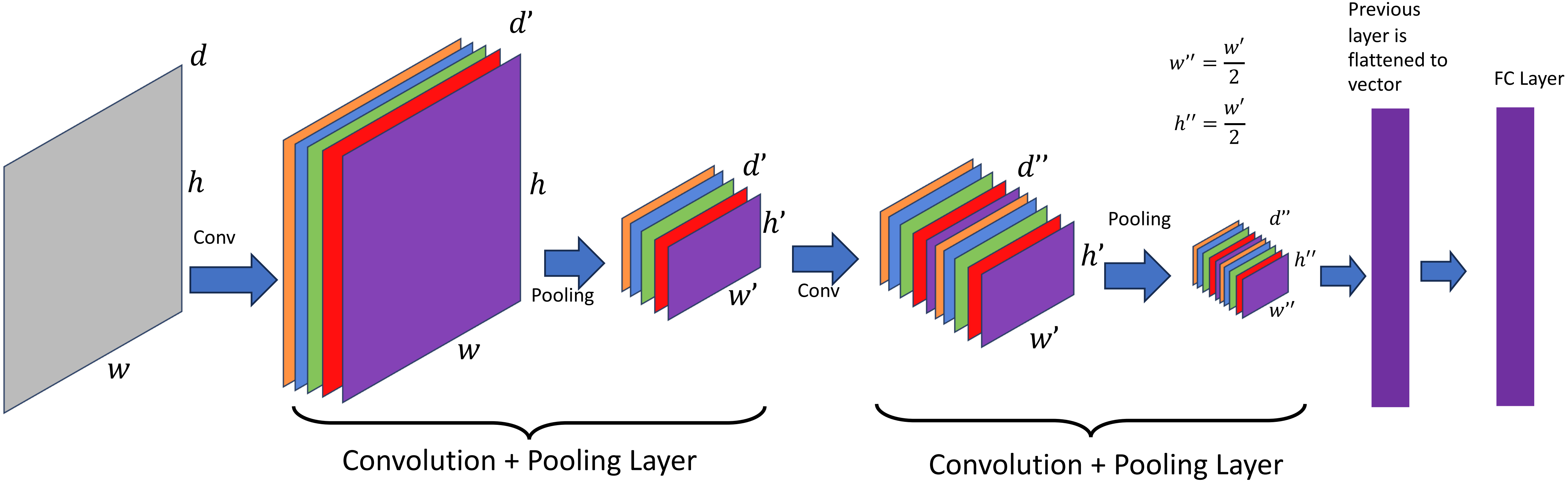



```
nn.Conv2d(in_channels=__, out_channels= __, kernel_size= 3 , stride=1, padding=1)

nn.MaxPool2d(kernel_size=__, stride=__)
```


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Flatten + FC Layers

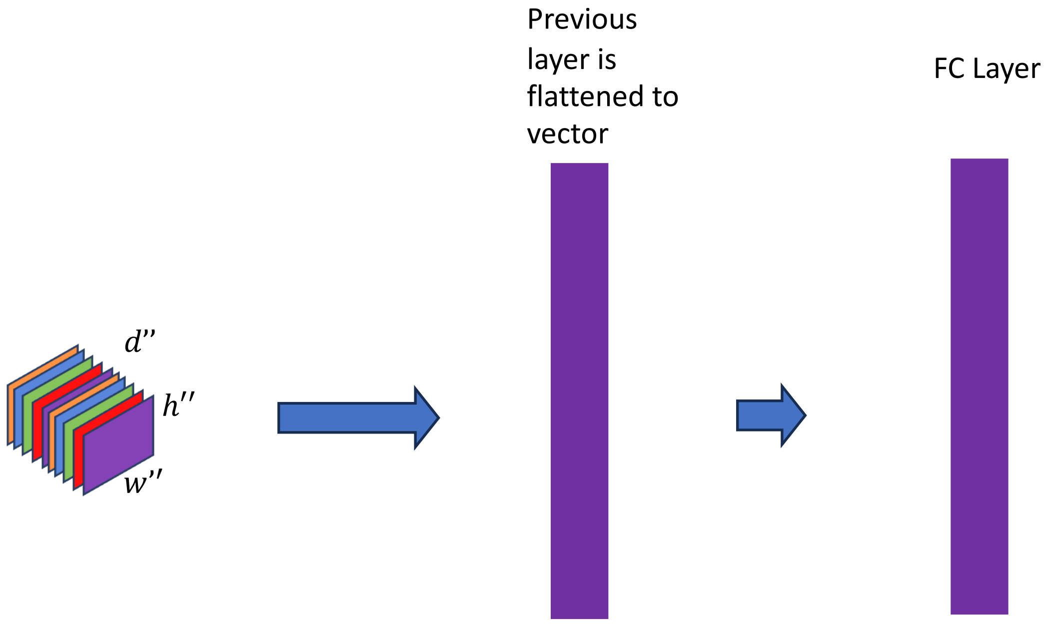


What is the size of flattened vector?

Think of flatten vector as feature vector

Fully Connected (FC) layer is traditional neural network layer with predefined number of neurons

Flattened Layer + FC Layer

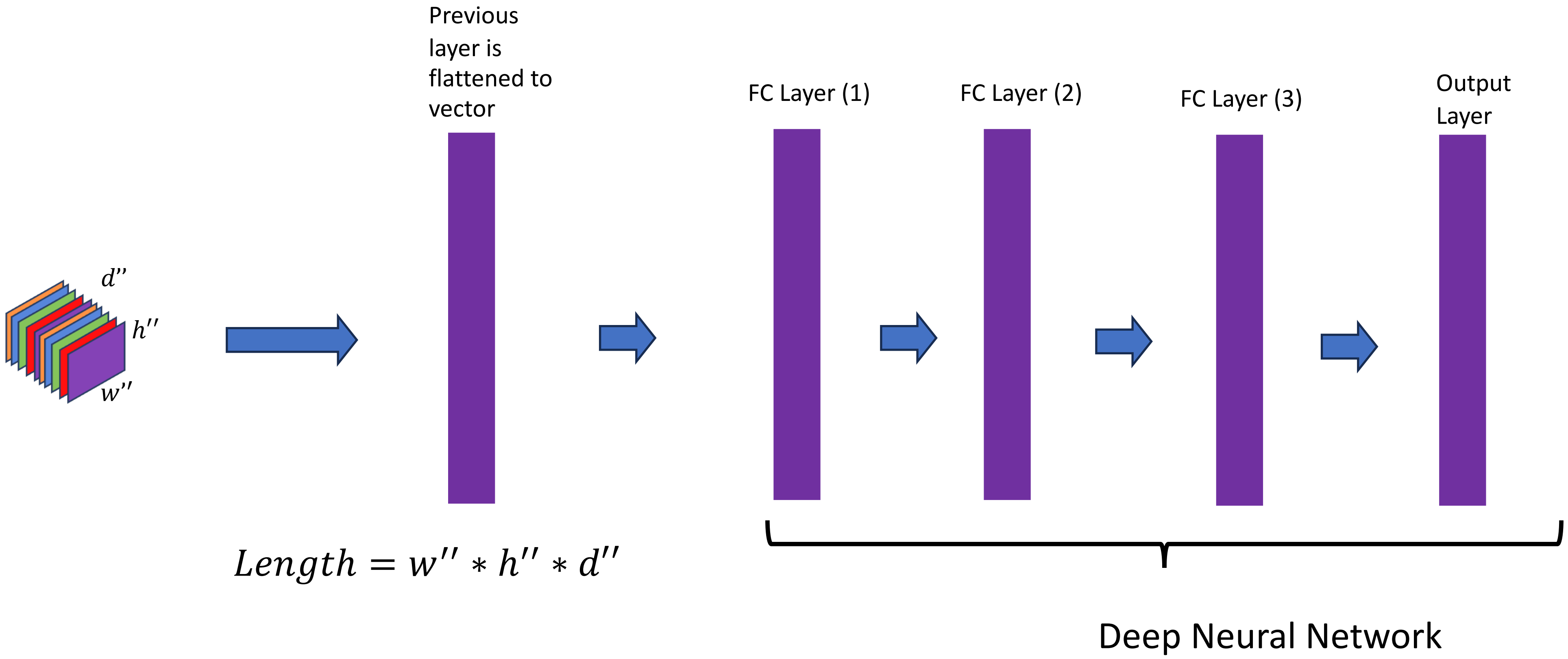


$$Length = w'' * h'' * d''$$

Length of the flattened layer is equal to the size of the input for FC layer

Design of FC layers require the knowledge of size of final Conv+Pooling layer

Flattened Layer + FC Layer(s)



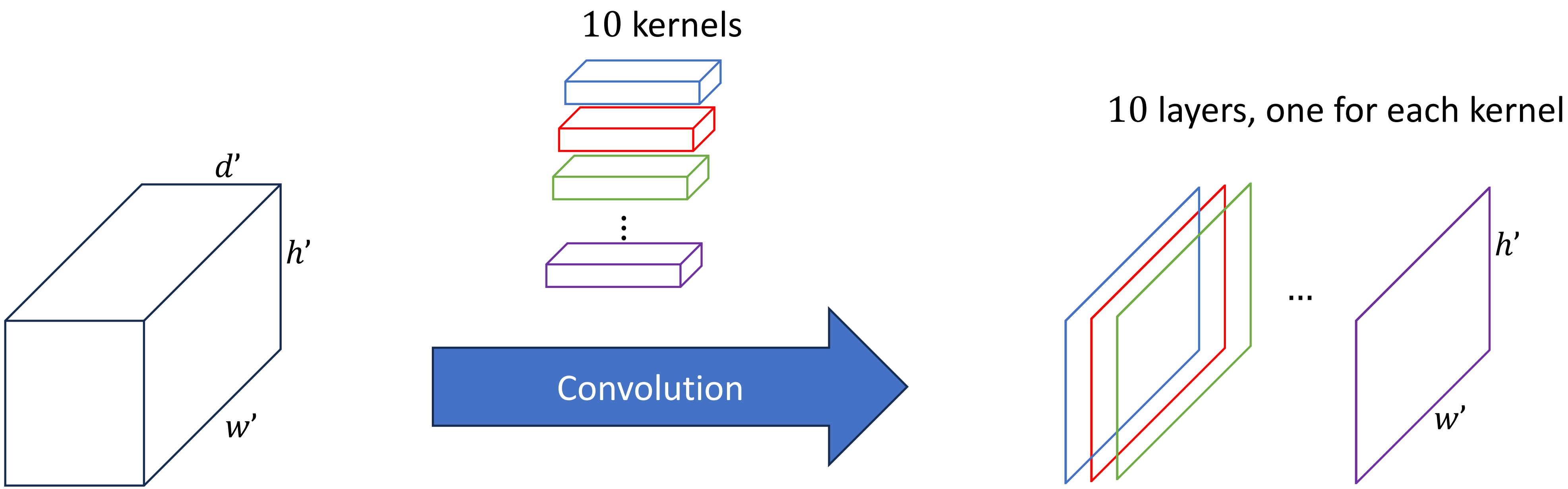
Length of the flattened layer is equal to the size of the input for FC layer

Design of FC layers require the knowledge of size of final Conv+Pooling layer

How many FLOPS per layer?

- FLOPS: Floating Point Operations

```
nn.Conv2d(in_channels=5, out_channels= 10, kernel_size= 3 , stride=1, padding=1)
```

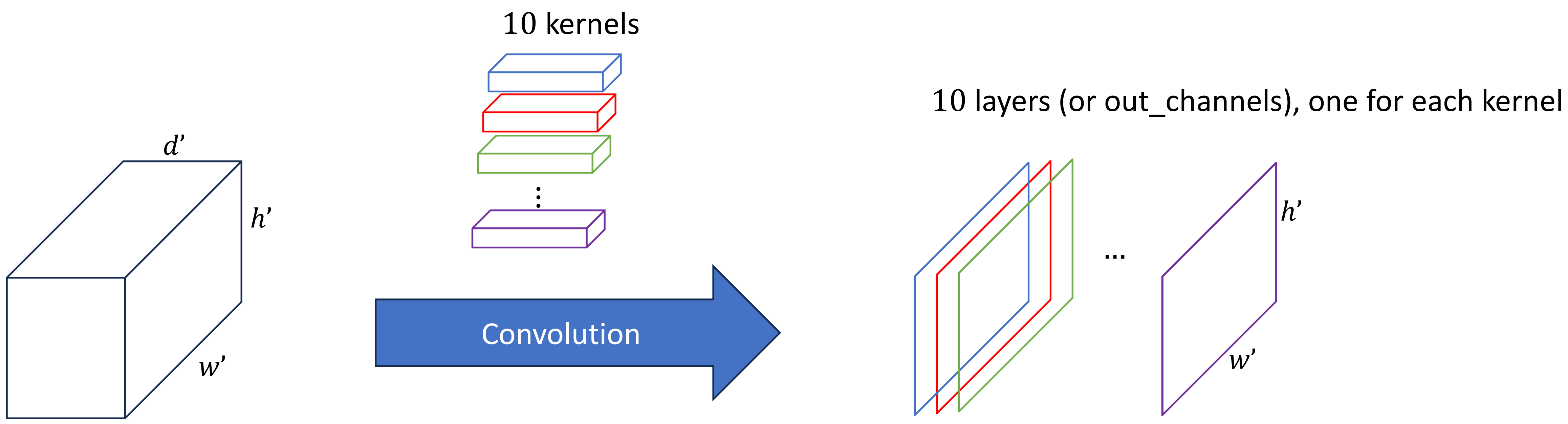


$h' = w' = 16$

$d' = 5$

How many FLOPS per layer?

```
nn.Conv2d(in_channels=5, out_channels= 10, kernel_size= 3 , stride=1, padding=1)
```



$h' = w' = 16 \qquad d' = 5$

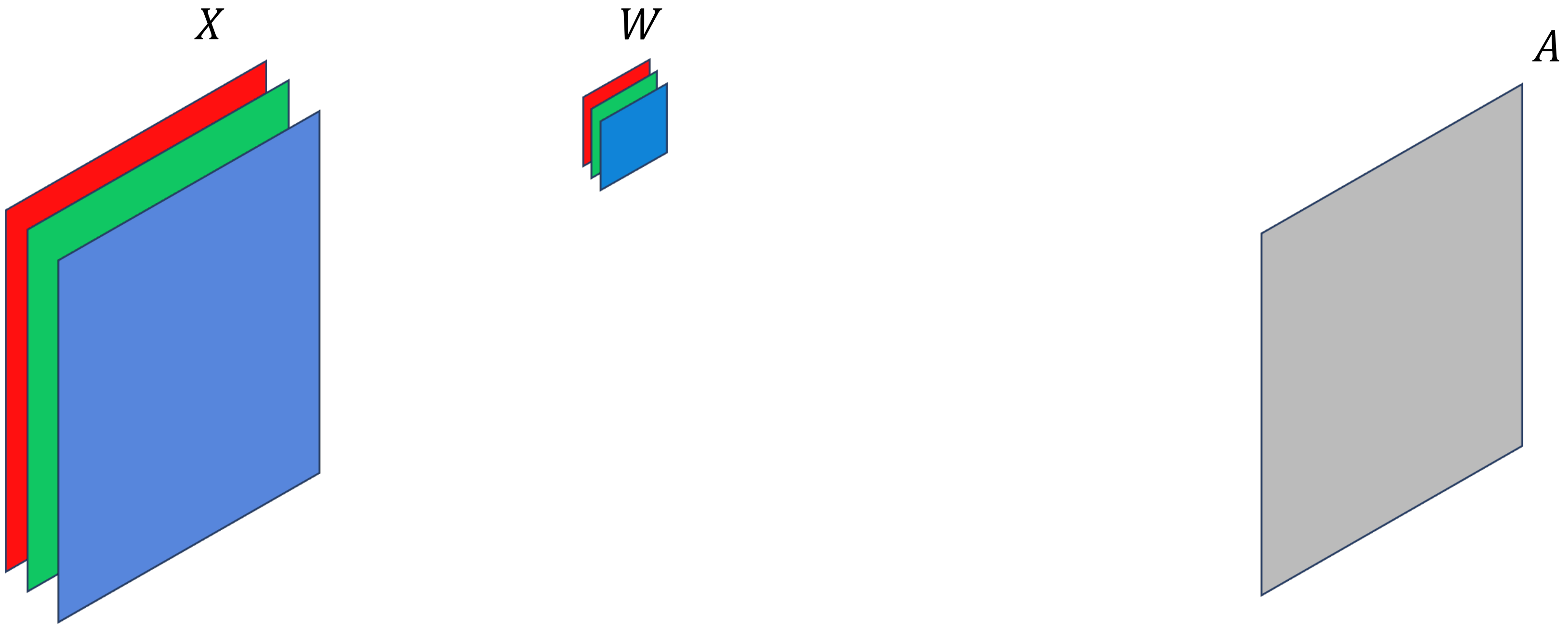
$$(out\_channels * out\_height * out\_width) * 2 * (in\_channels * kernel\_height * kernel\_width)$$

# of pixels in output

(# of multiplications + # of additions) per pixel

Convolution Layer: RGB Input

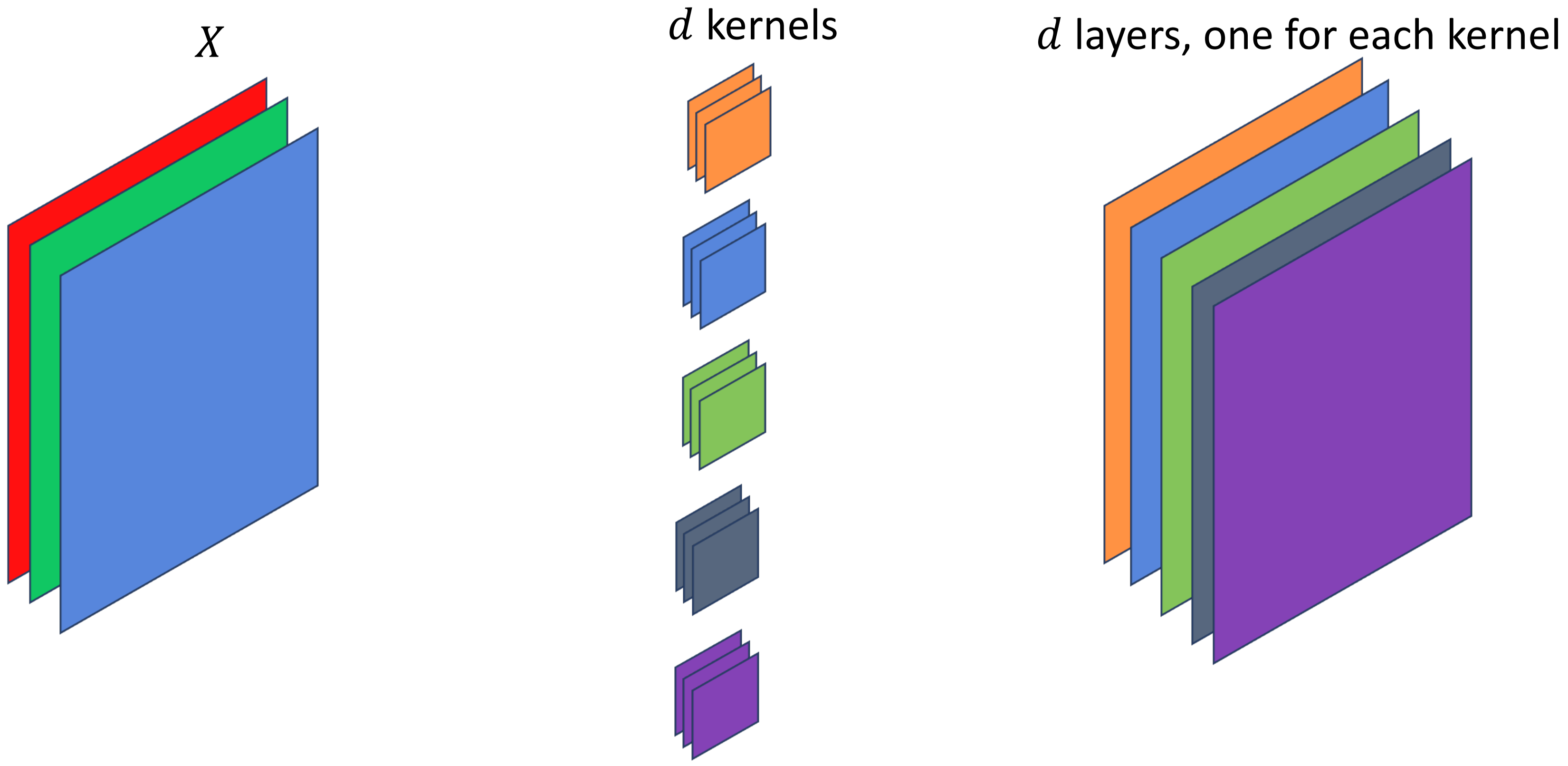
```
nn.Conv2d(in_channels=___, out_channels=__, kernel_size= 3 , stride=1, padding=1)
```




```
Shape of kernel:________
```

Convolution Layer: RGB Input

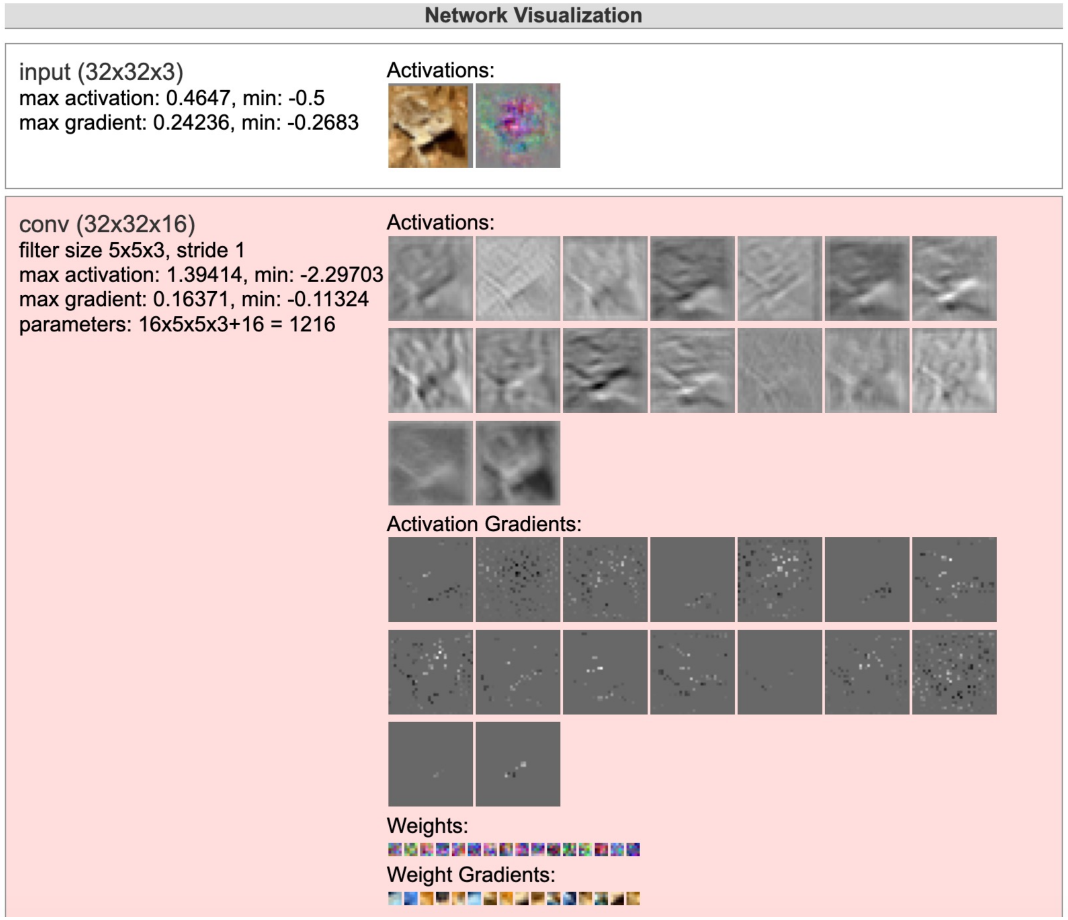
```
nn.Conv2d(in_channels=__, out_channels=__, kernel_size= 3 , stride=1, padding=1)
```



Assuming kernel size of 3 for each kernel, how many parameters in the convolution layer?

CNN Playground

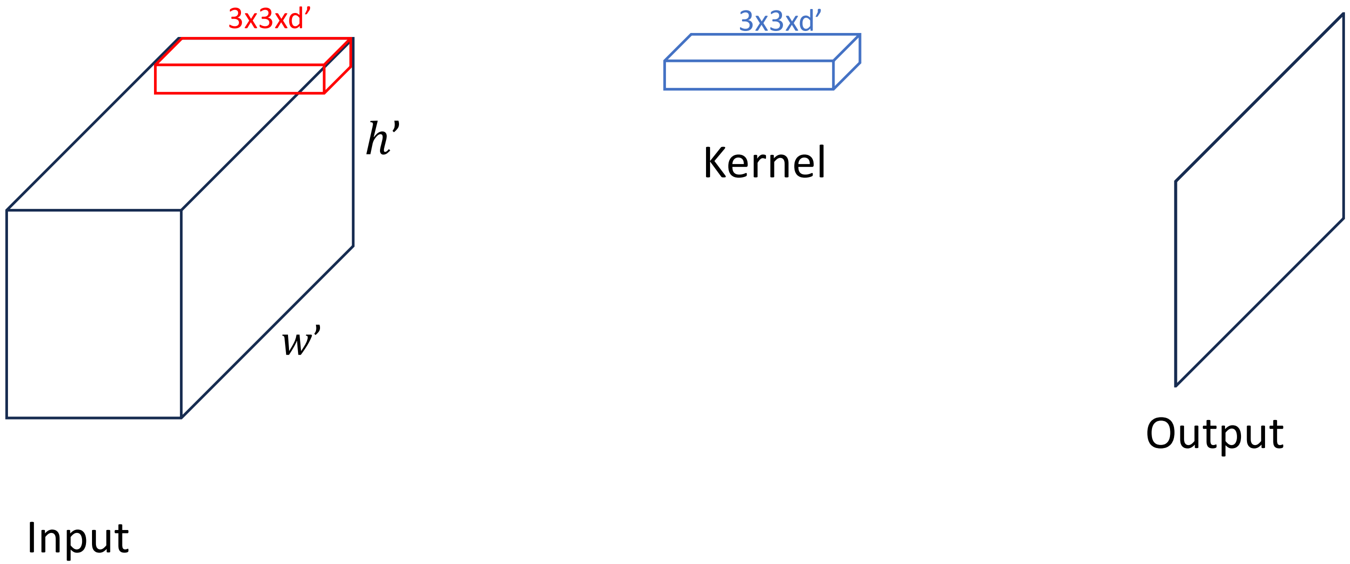
- https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html



Types of Convolution

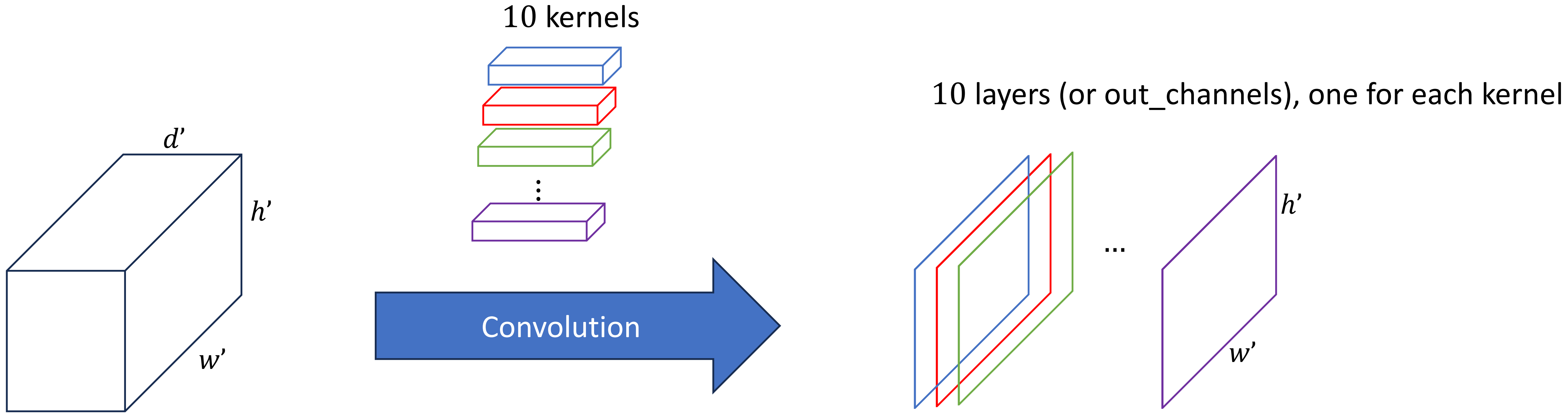
- Standard Convolution
- Depthwise Convolution
- Depthwise separable Convolution

Standard Convolution



Recap: How many FLOPS per layer?

```
nn.Conv2d(in_channels=5, out_channels= 10, kernel_size= 3 , stride=1, padding=1)
```



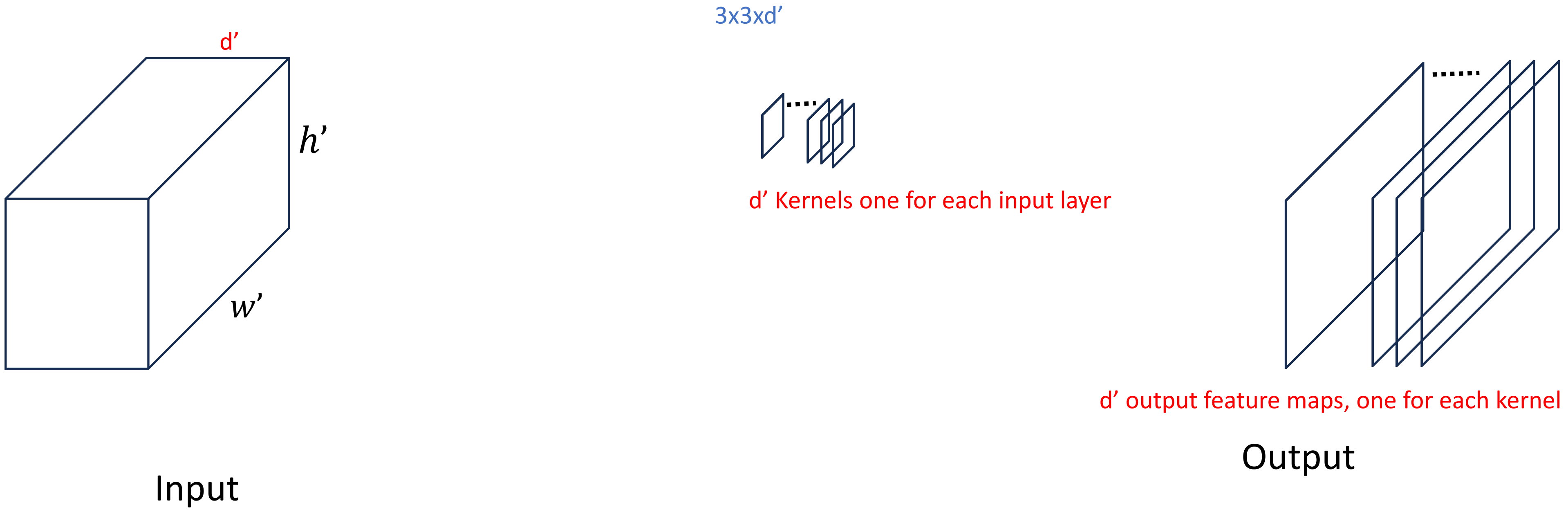
$h' = w' = 16$ $\quad d' = 5$

$(out\_channels * out\_height * out\_width) * 2 * (in\_channels * kernel\_height * kernel\_width)$

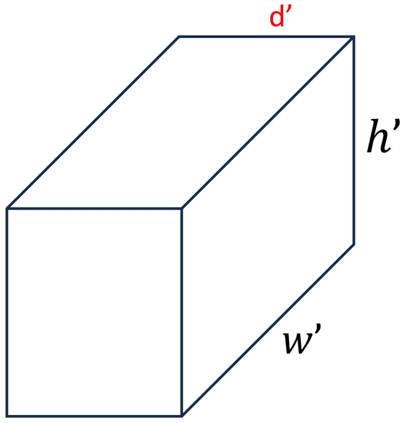
# of pixels in output

(# of multiplications + # of additions) per pixel

Depthwise Convolutions

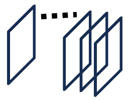


Depthwise Convolutions: # of FLOPs

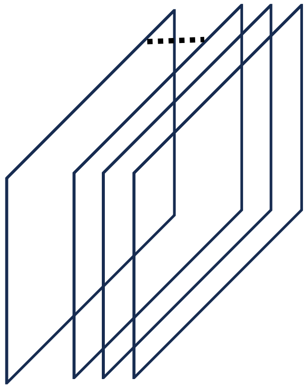


Input

$3{\times}3\ \ kernels\ of\ shape\ 3{\times}3{\times}1$



# of kernels: $d’$
one kernel for each input layer



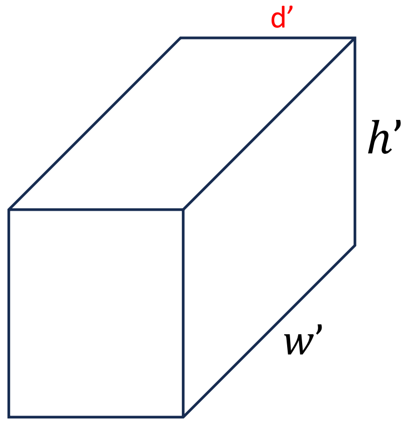
d’ output feature maps, one for each kernel

Output

$$(out\_channels * out\_height * out\_width) * 2 * (kernel\_height * kernel\_width)$$

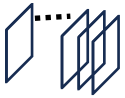
Depthwise Separable Convolutions

No interaction between the layers with depthwise convolutions

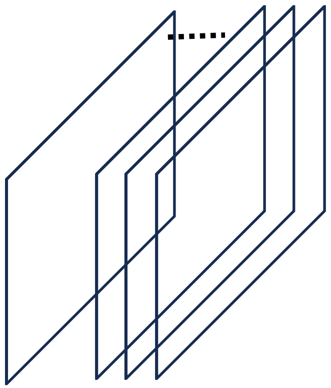


Input

3×3 $kernels\ of\ shape$ $3\times3\times1$



# of kernels: d'
one kernel for each input layer



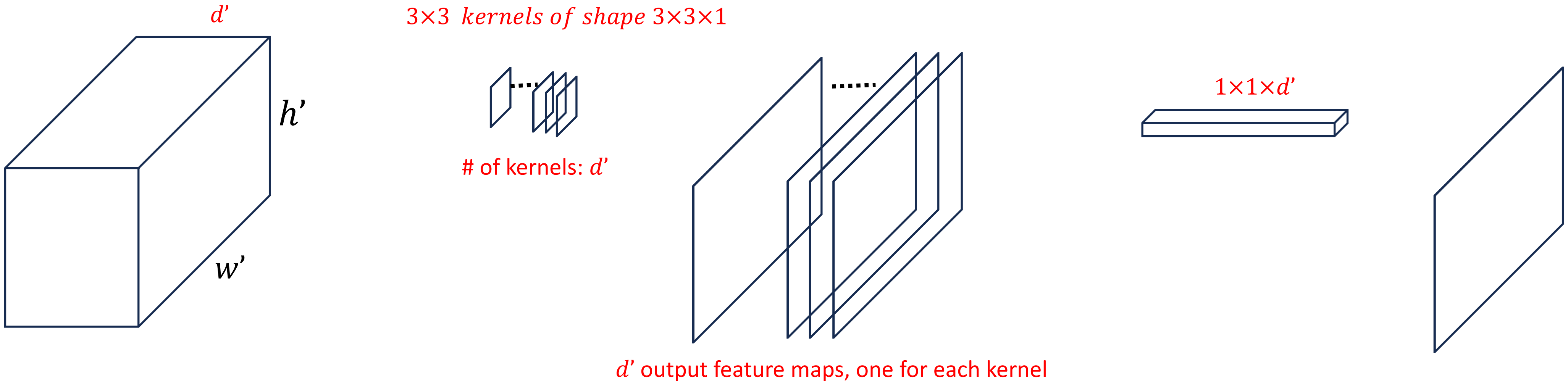
d' output feature maps, one for each kernel

Output

Depthwise Separable Convolutions

depthwise convolutions

pointwise convolutions

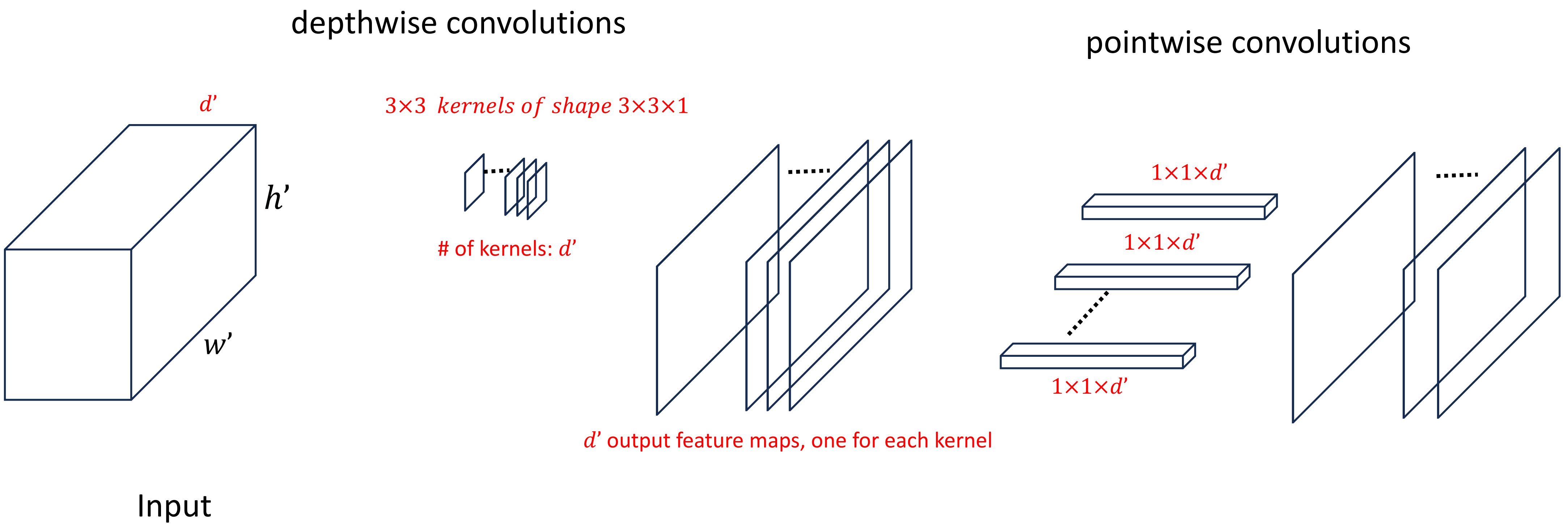


Input

How to get multiple output feature maps at pointwise convolution layer?

$$(out\_channels * out\_height * out\_width) * 2 * (kernel\_height * kernel\_width) + 2*(out\_height * out\_width * in\_channels)$$

Depthwise Separable Convolutions



How to get multiple output feature maps at pointwise convolution layer?

Use multiple 1x1 kernels

$$(out\_channels * out\_height * out\_width) * 2 * (kernel\_height * kernel\_width)$$

Mobilenets

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko
Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam

Google Inc.

{howarda,menglong,bochen,dkalenichenko,weijunw,weyand,anm,hadam}@google.com

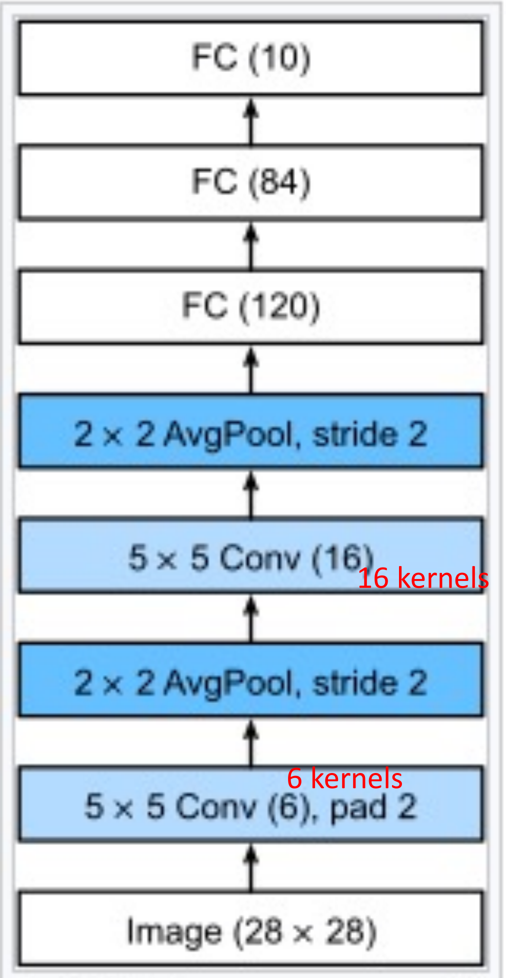
Key Idea: Depthwise Convolutions

https://arxiv.org/pdf/1704.04861

CNN Architectures

- AlexNet
- VGG-16
- ResNet-50
    - Residual connections
    - Bottleneck layer to reduce parameter count
- MobileNetV2
    - Depthwise Convolution

LetNet5 Architecture: Tutorial!!




```
nn.Linear(in_features =__, out_features=__)
nn.Linear(in_features=__, out_features=__)
nn.Linear(in_features=__, out_features=__)
nn.AvgPool2d(kernel_size=__, stride=__)
nn.Conv2d(in_channels=__, out_channels=__, kernel_size=__, stride=2, padding=2)
nn.AvgPool2d(kernel_size=__, stride=__)
nn.Conv2d(in_channels=__, out_channels=__, kernel_size=__, stride=1, padding=2)
```

Image taken from: https://en.wikipedia.org/wiki/LeNet

VGGNet: Homework

- Visual Geometry Group (VGG) at the University of Oxford

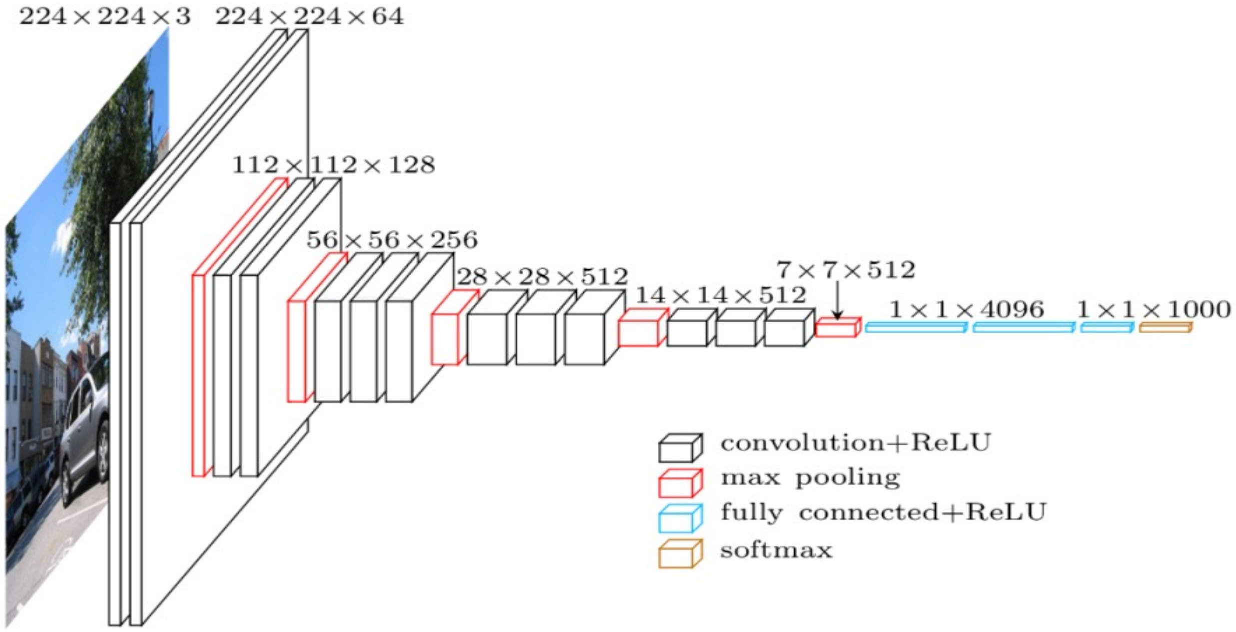


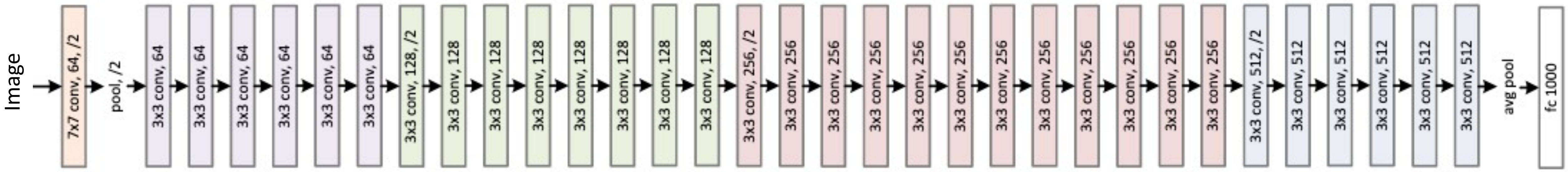
Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

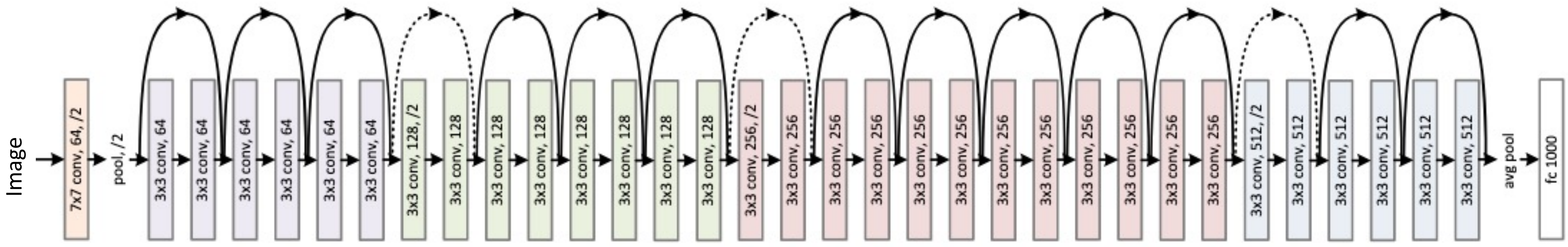
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

https://arxiv.org/pdf/1409.1556v6

ResNet: Homework

Deeeeper Network: How many layers?
Potential Issues with this network!!!

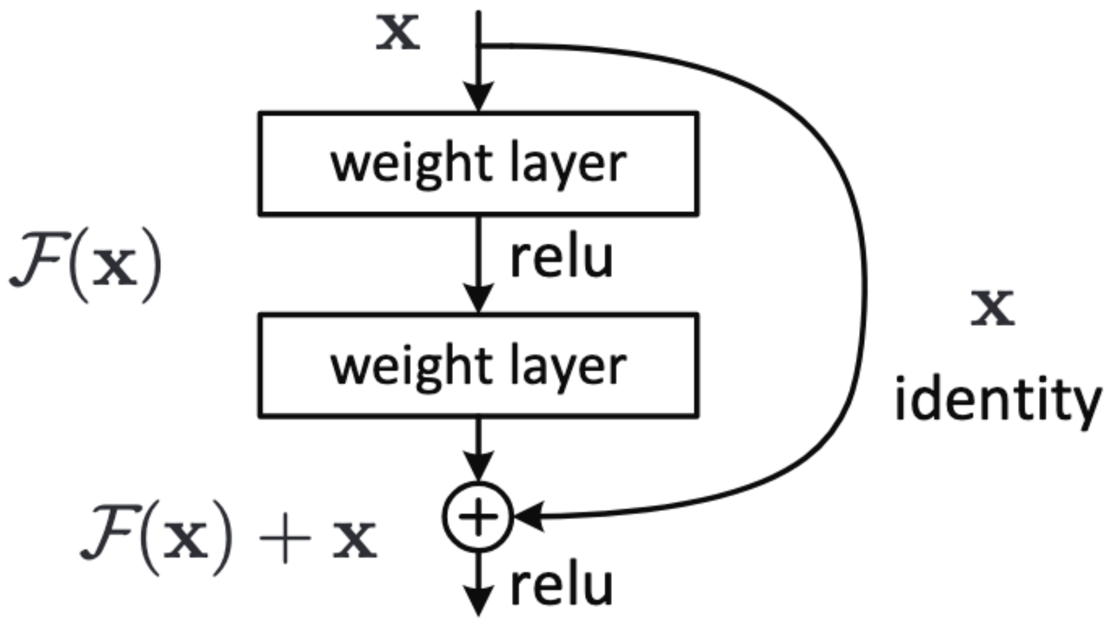




https://arxiv.org/abs/1512.03385

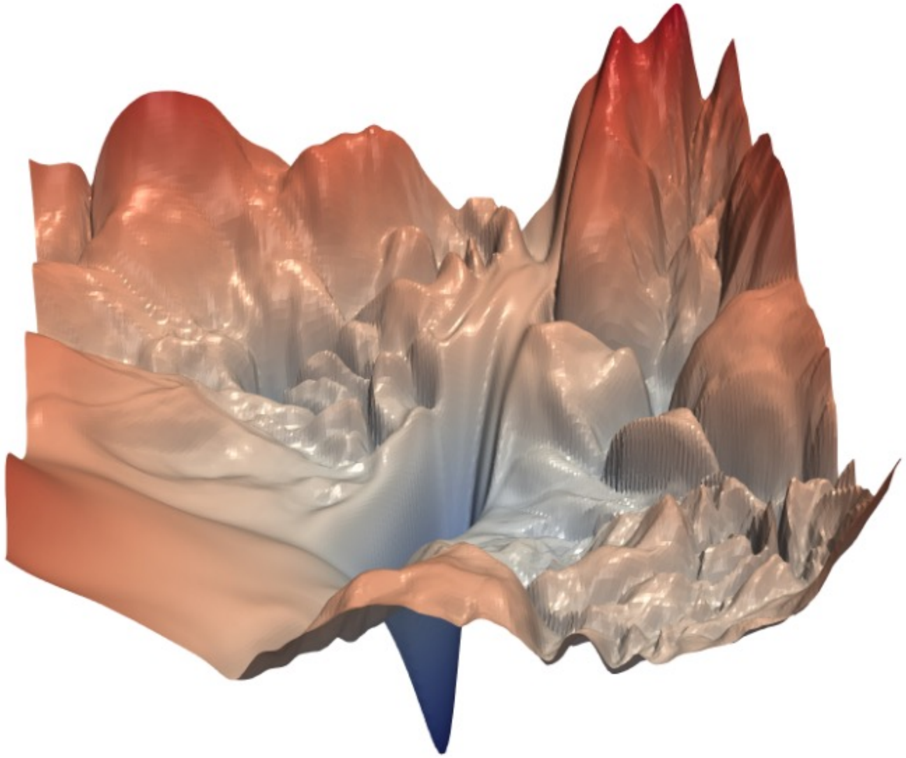
Skip Connection: Tutorial!!

aka Residual Connection

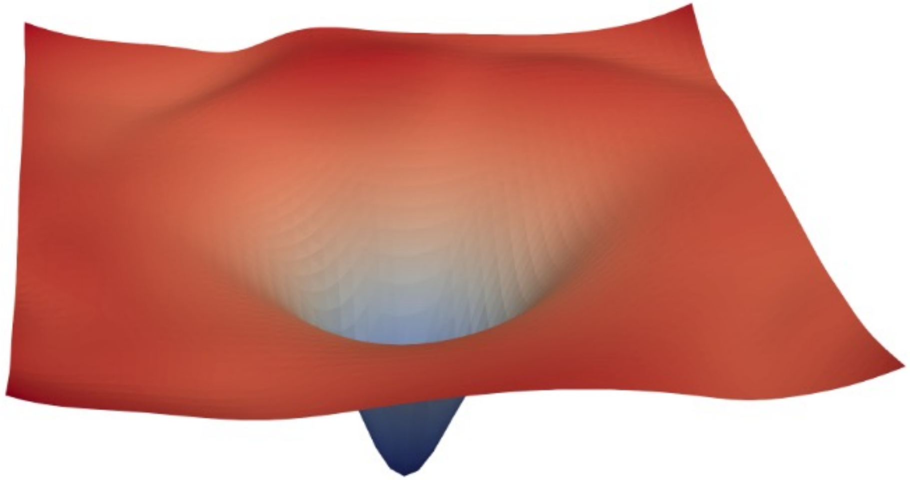


Basic building block of residual learning

Skip Connections



(a) without skip connections



(b) with skip connections

Li, H., Xu, Z., Taylor, G., Studer, C. and Goldstein, T., 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, *31*.