Convolutional Neural Networks

Outline

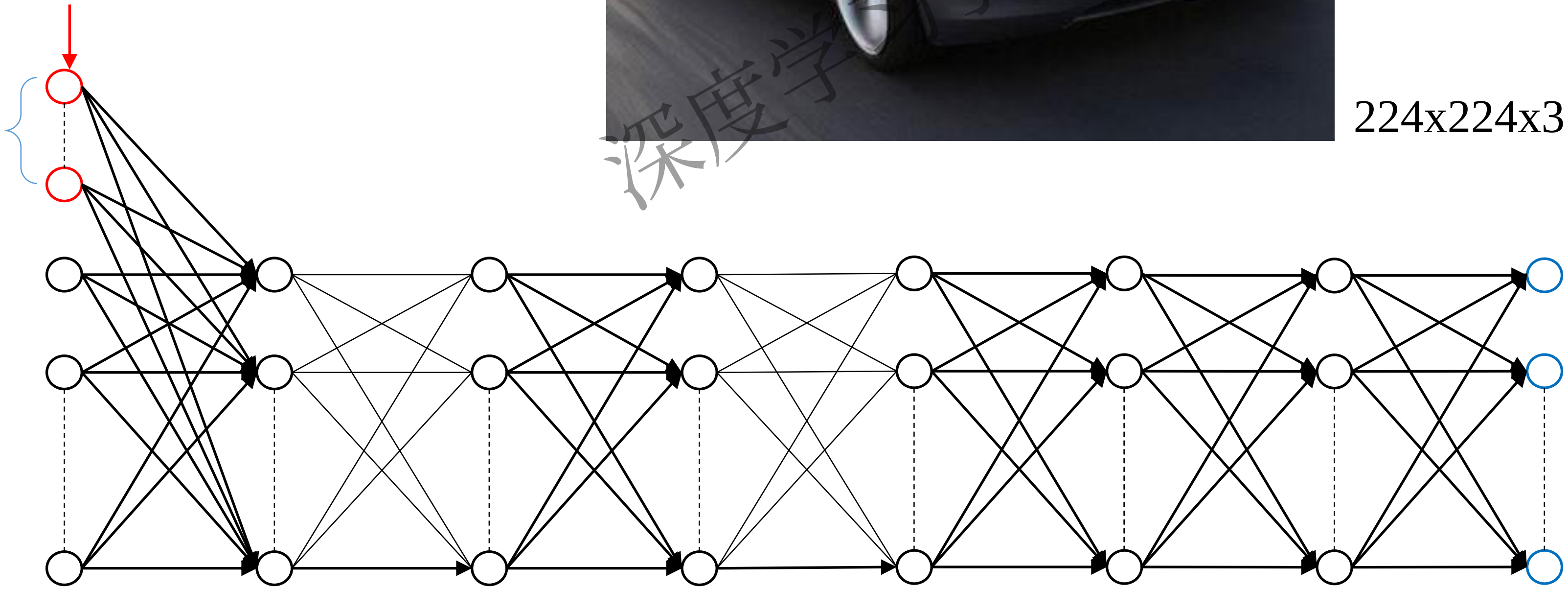
- Motivation
- Components
- Structure
- Learning
- Application

CNN’s Motivation

28 × 28

224x224x3

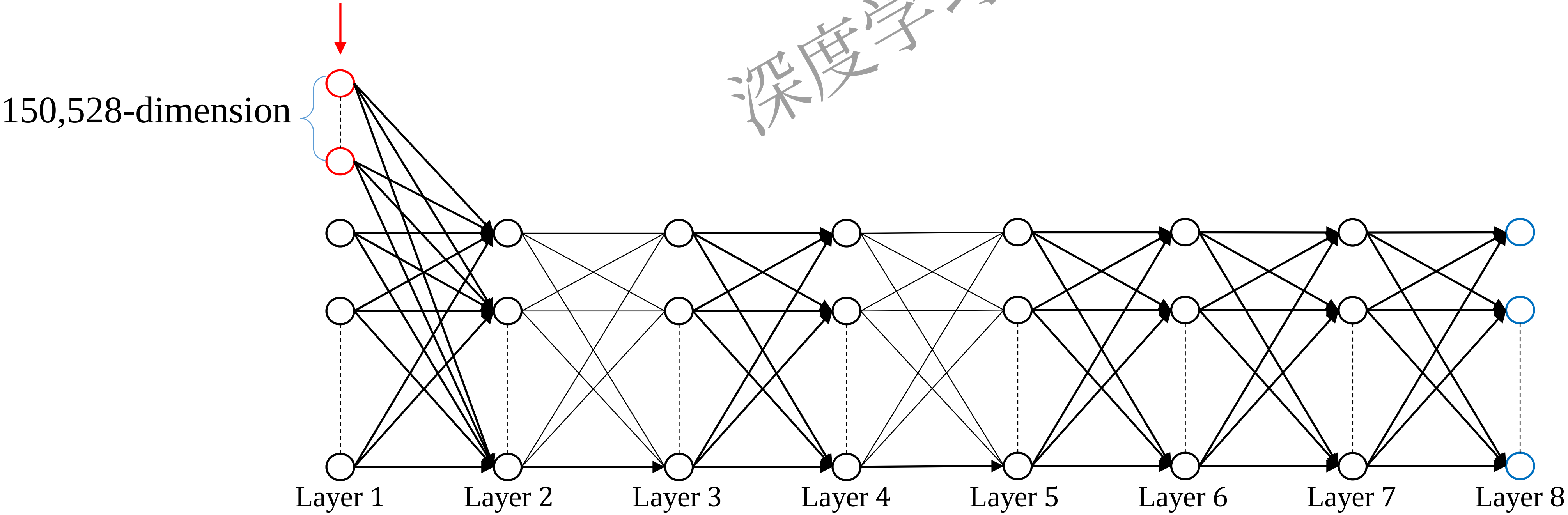
How to handle the very large colorful image in the size of 224x224x3, where the 3 denotes the R,G,B channels.



CNN’s Motivation

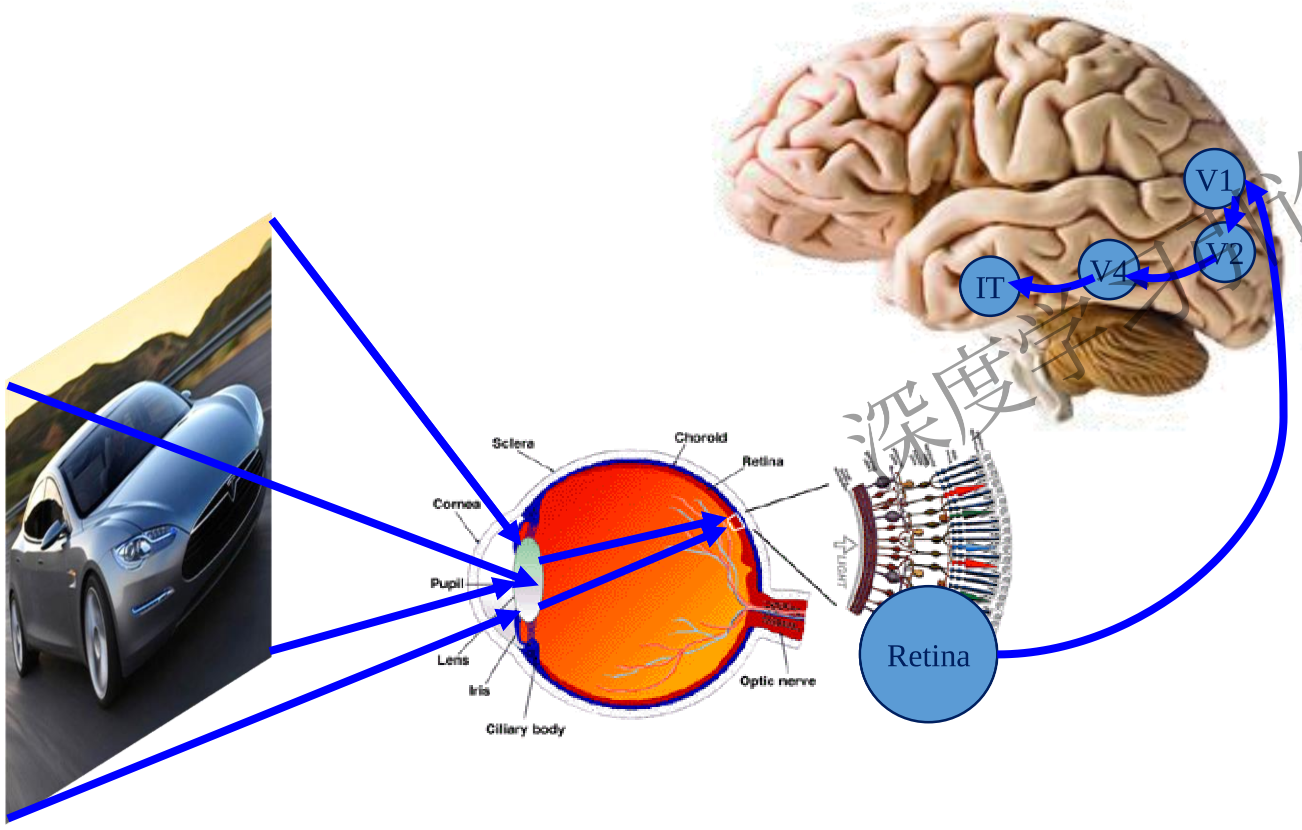
224x224x3

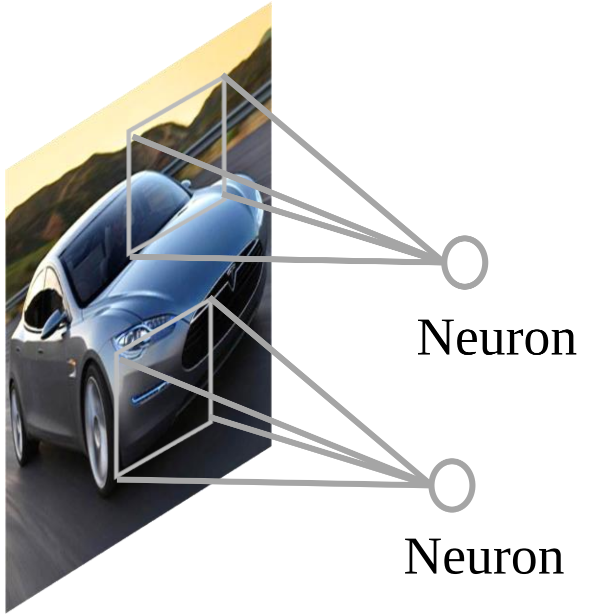
- Suppose the dimension of hidden layer is 100. The number of parameters in the first layer is 15,052,800, unacceptable !
- How does the brain process the image?



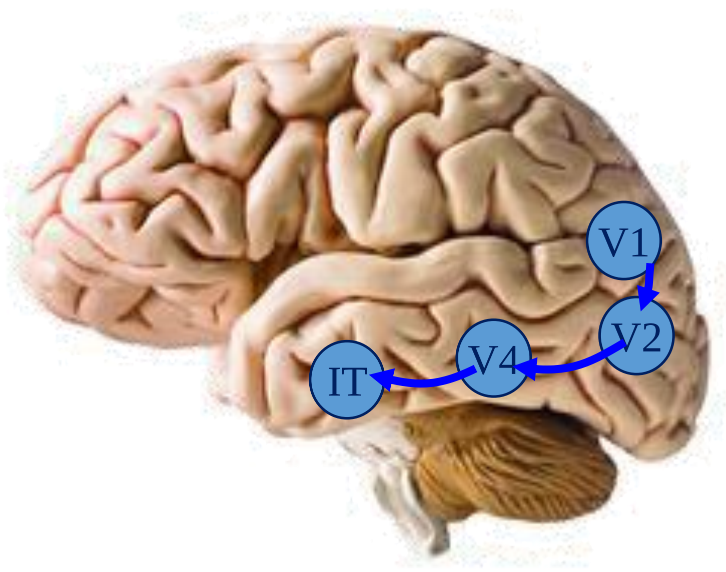
CNN’s Motivation

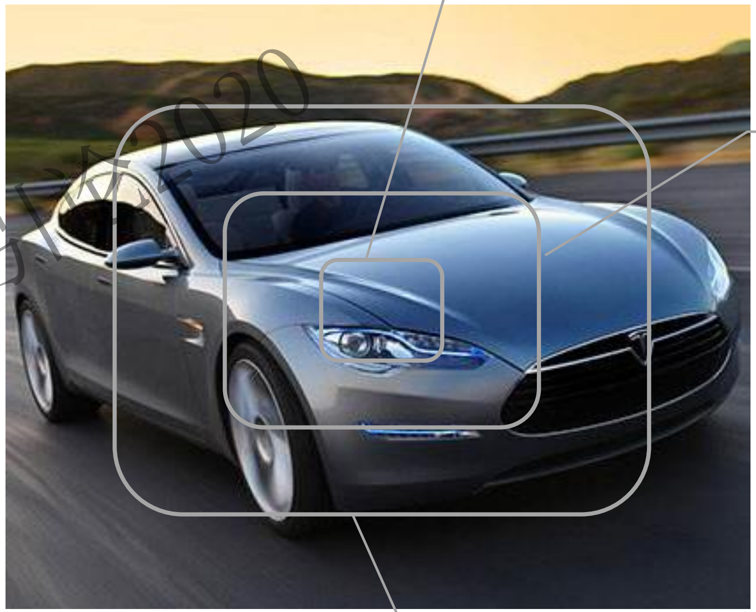
- Each neuron can only perceive a sub-region in the image.
- The sub-region is called receptive field.





CNN’s Motivation





Question:
How to build the model of receptive field?

Outline

- CNNs’ Motivation
- CNNs’ Components
- CNNs’ Structure
- CNNs’ Learning
- CNNs’ Application

CNN's Components

224x224x3

Channel R, 224x224x1

$$\begin{bmatrix} \ddots & 154 & 98 & 152 & 191 & \iddots \\ 110 & 151 & 49 & 215 & 184 & 231 \\ 130 & 141 & 231 & 183 & 140 & 219 \\ 121 & 163 & 195 & 182 & 139 & 216 \\ 142 & 191 & 184 & 173 & 150 & 241 \\ \iddots & 183 & 196 & 150 & 183 & \ddots \end{bmatrix}_{224 \times 224}$$

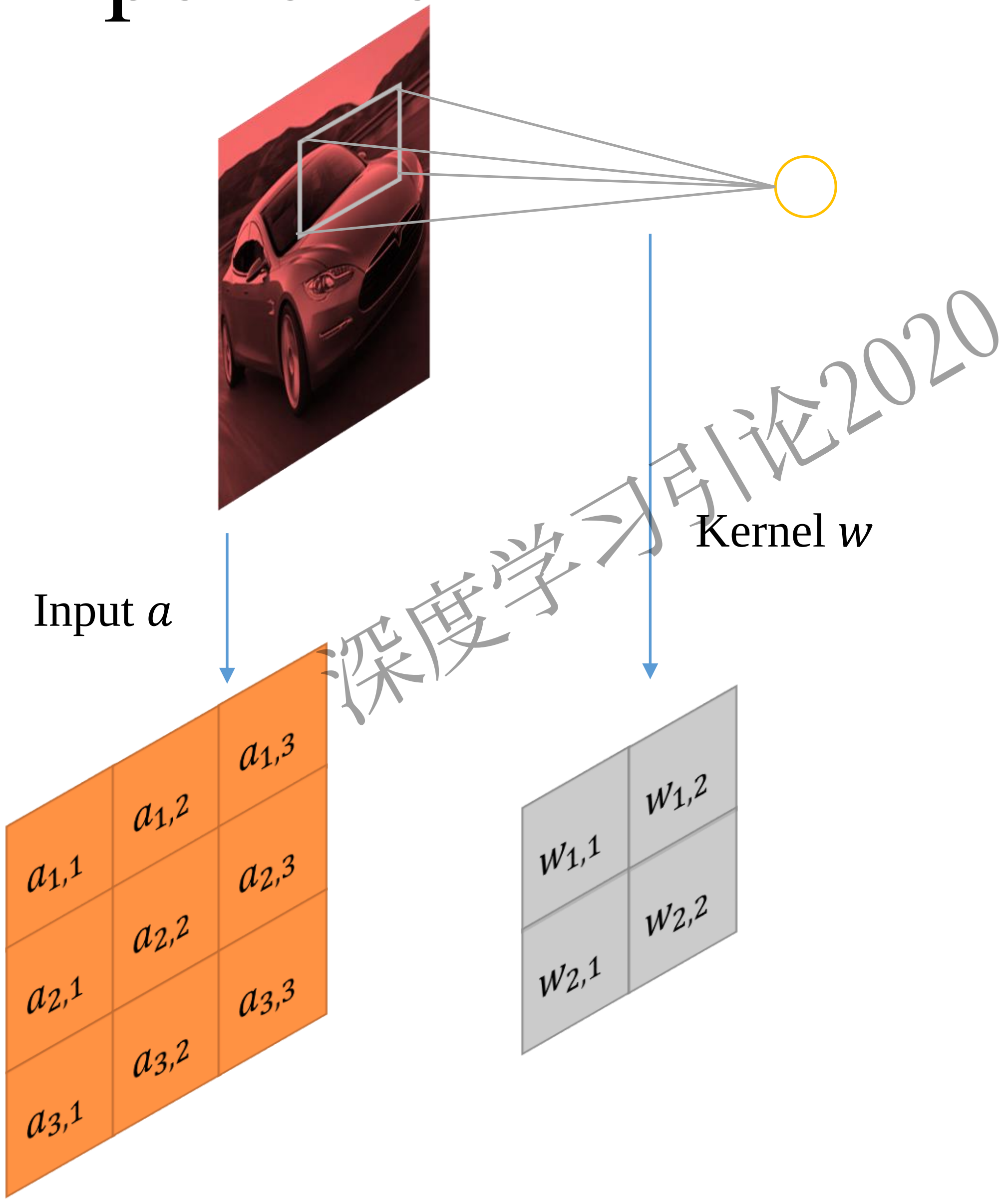
Channel G, 224x224x1

$$\begin{bmatrix} \ddots & 84 & 41 & 112 & 117 & \iddots \\ 80 & 195 & 52 & 172 & 139 & 251 \\ 128 & 131 & 107 & 191 & 147 & 204 \\ 162 & 130 & 218 & 148 & 191 & 196 \\ 171 & 82 & 136 & 122 & 250 & 141 \\ \iddots & 129 & 186 & 154 & 161 & \ddots \end{bmatrix}_{224 \times 224}$$

Channel B, 224x224x1

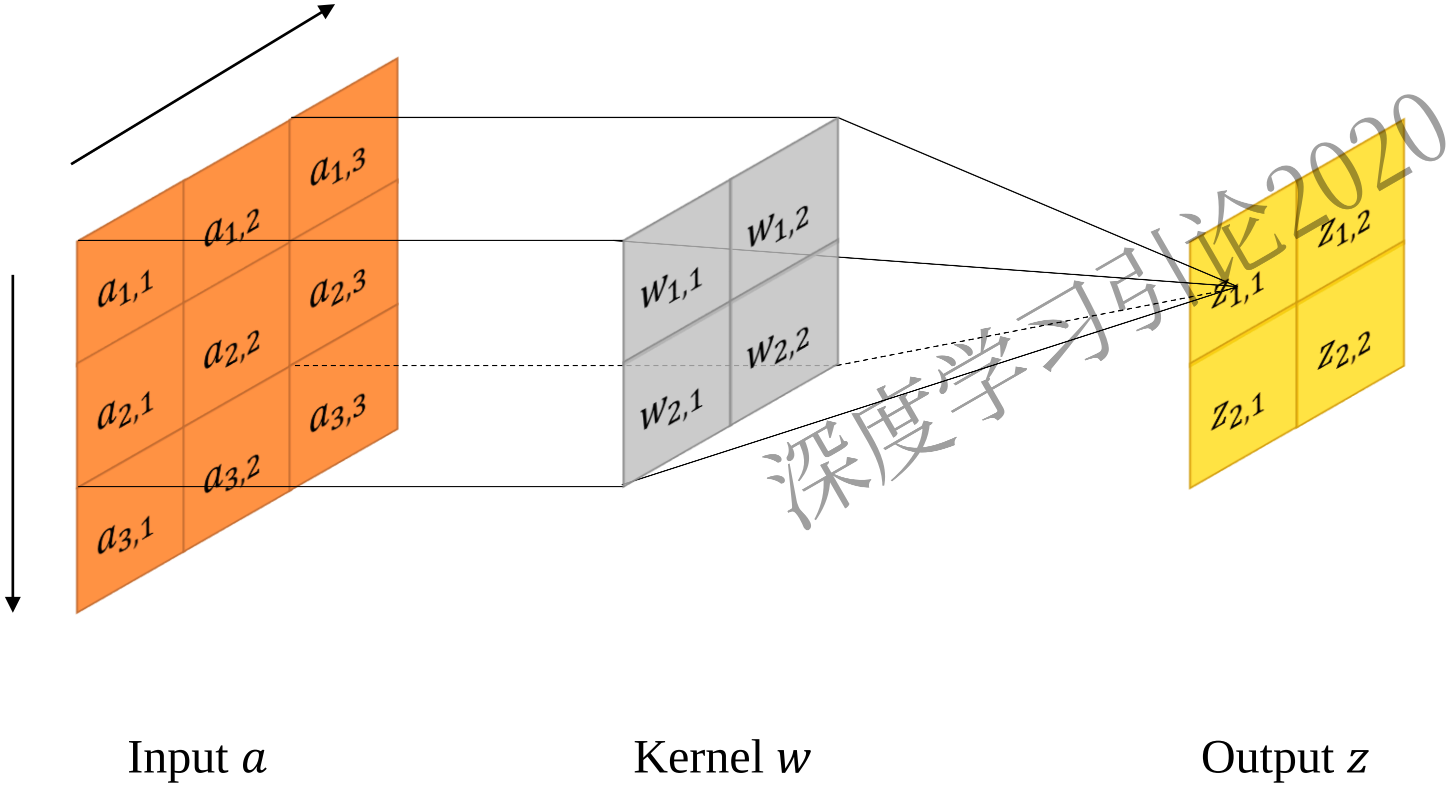
$$\begin{bmatrix} \ddots & 194 & 118 & 141 & 211 & \iddots \\ 190 & 251 & 143 & 218 & 137 & 221 \\ 231 & 162 & 221 & 161 & 129 & 194 \\ 158 & 181 & 171 & 146 & 239 & 186 \\ 172 & 201 & 191 & 131 & 180 & 216 \\ \iddots & 161 & 142 & 182 & 171 & \ddots \end{bmatrix}_{224 \times 224}$$

CNN’s Components



- Kernel is a small matrix used to process image.
- The size of the kernel is equivalent to the receptive field.

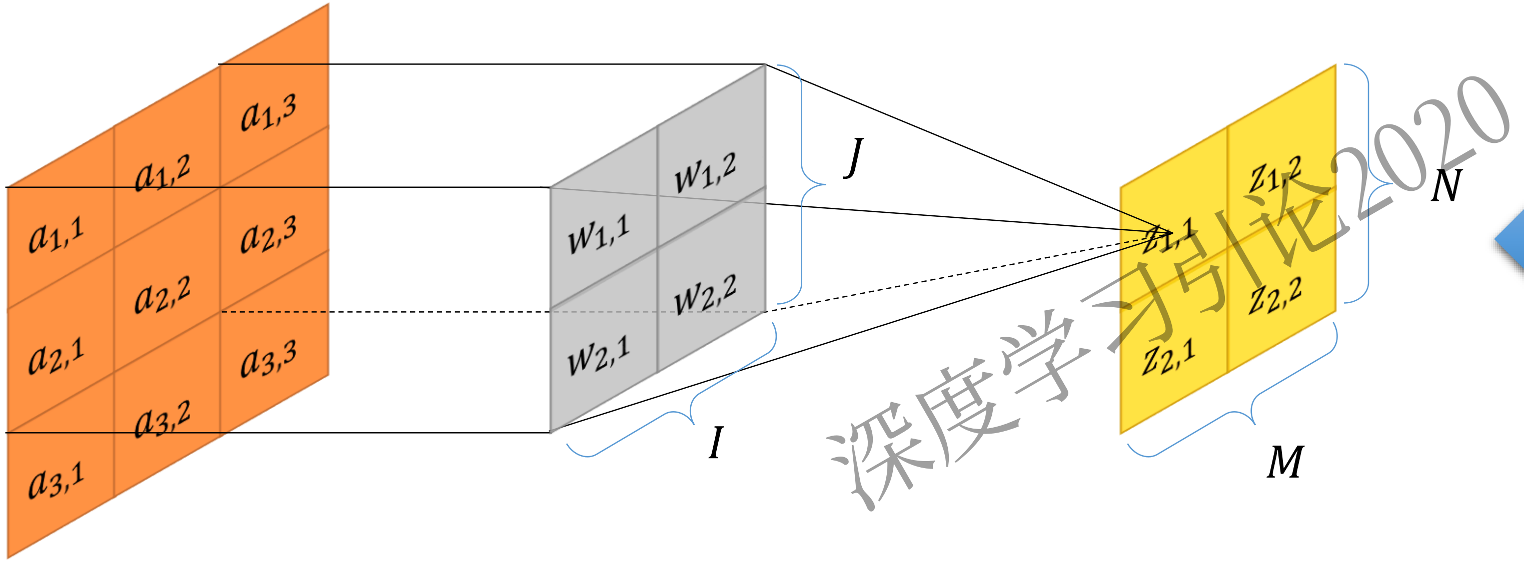
CNN's Components



Convolution

- The kernel w slides over the width and height of input a.
- The output z is computed by the sum of the element-wise product of a and w.

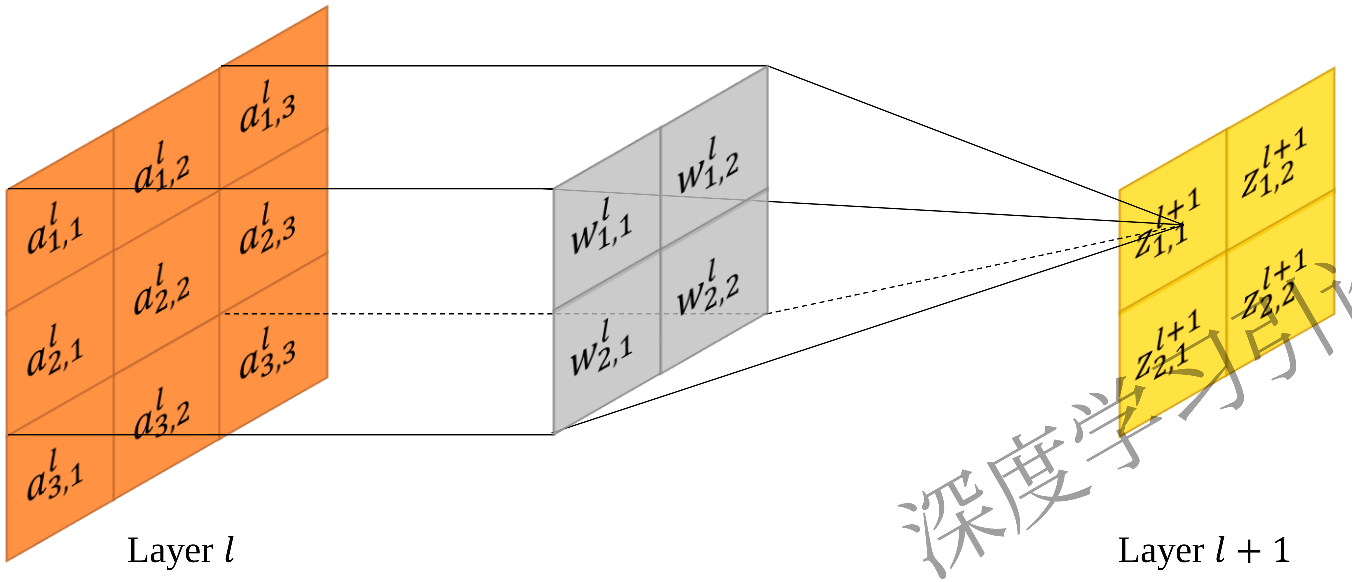
CNN’s Components

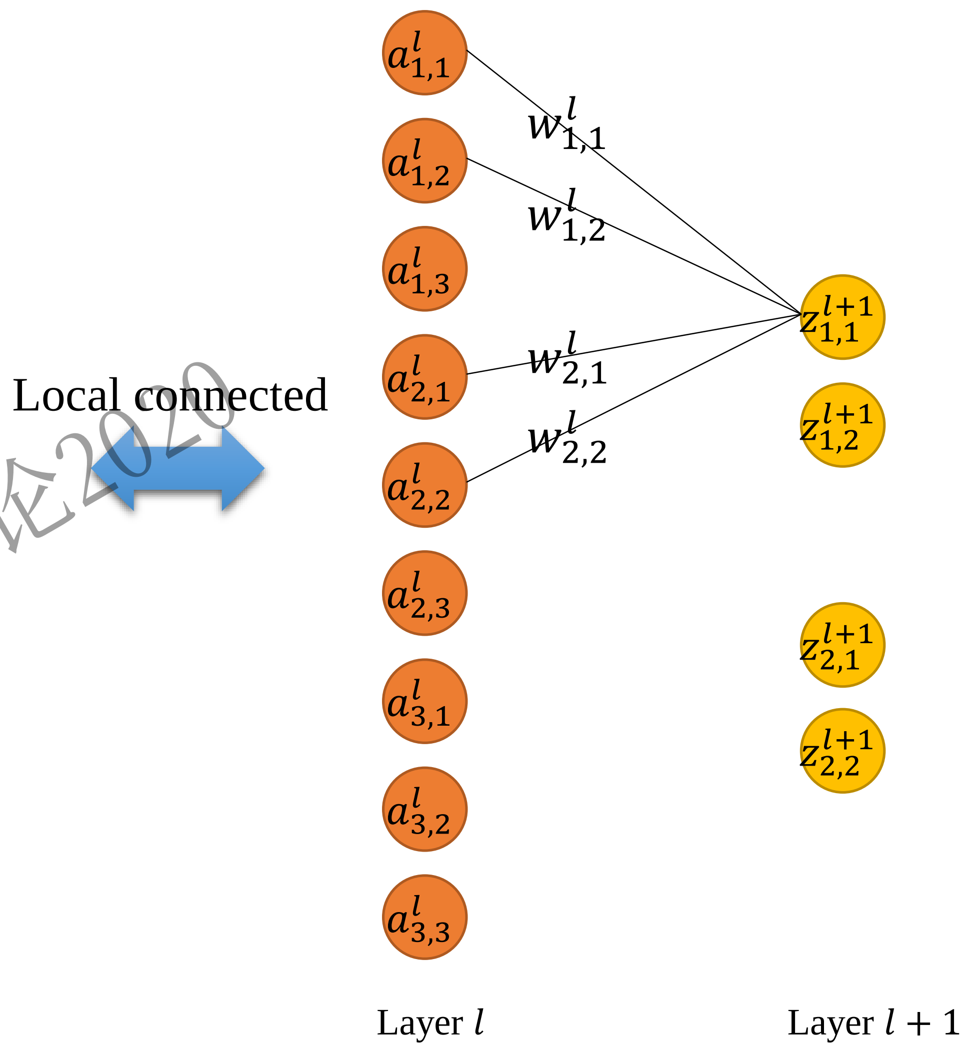


$$z_{n,m} = \sum_{i=1}^{I} \sum_{j=1}^{J} a_{n+j-1,m+i-1} \cdot w_{j,i}$$

- The size of the kernel w is $I \times J$
- The size of the output z is $M \times N$.

CNN's Components

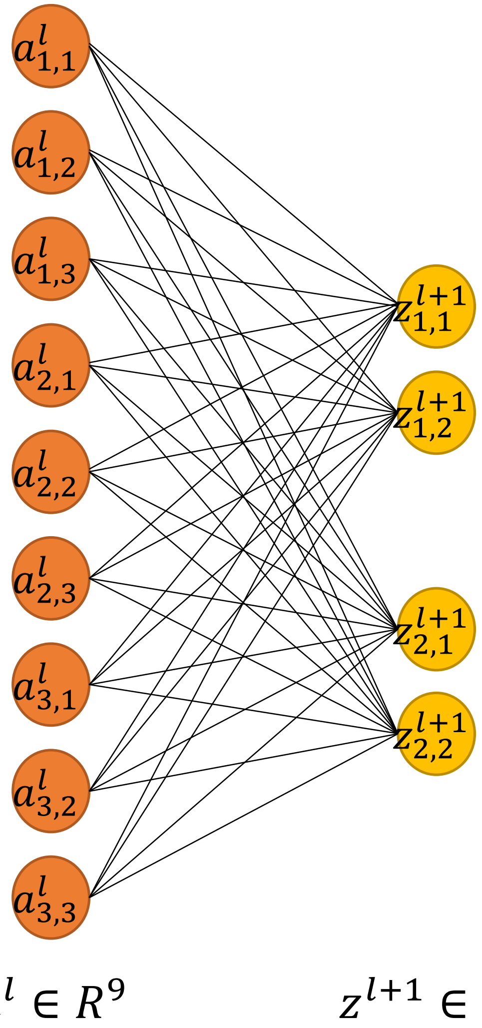




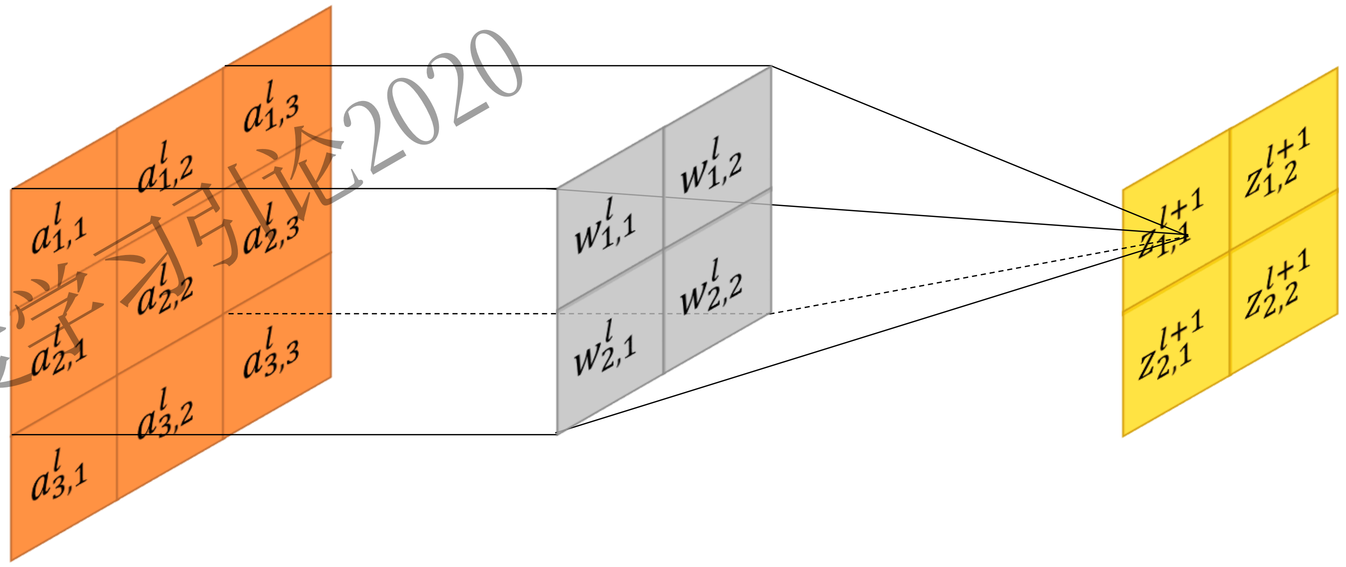
$$z^{l+1}_{n,m} = \sum_{i=1}^{I}\sum_{j=1}^{J} a^l_{n+j-1,m+i-1} \cdot w^l_{j,i}$$

- Each summation is equivalent to the local connected forward computation.
- The kernel is shared in each receptive field.

CNN's Components

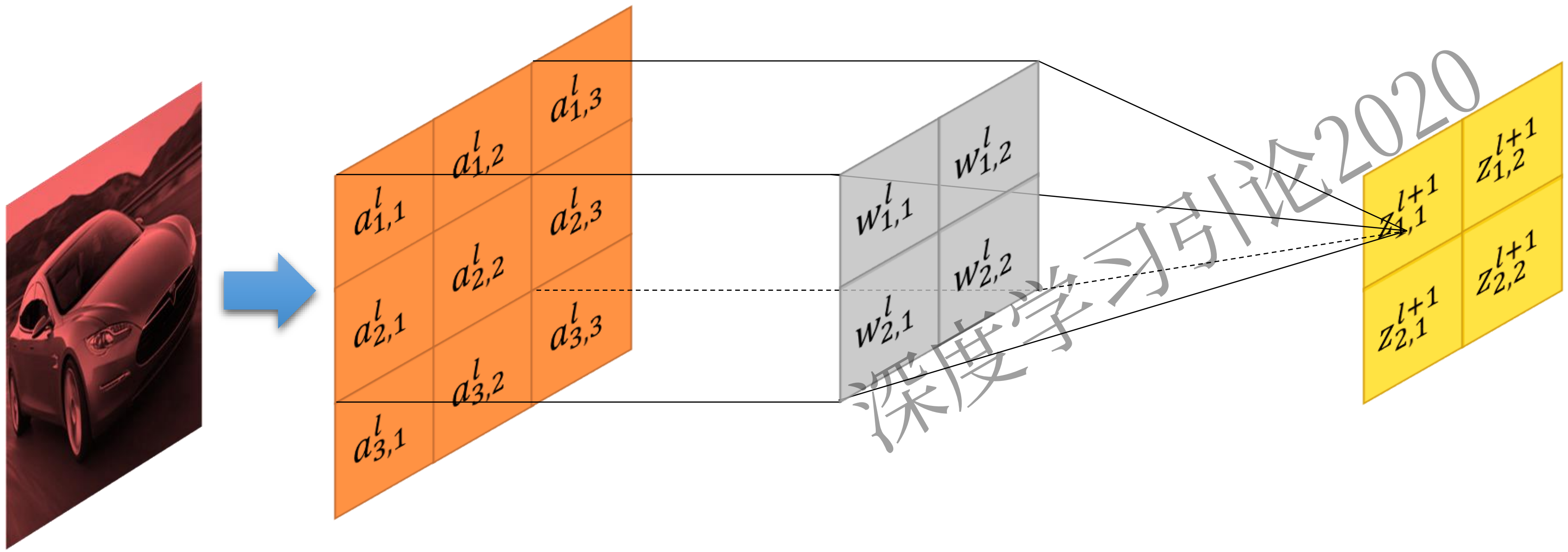


- The number of parameters in fully-connected layer is $9 \times 4 = 36$.



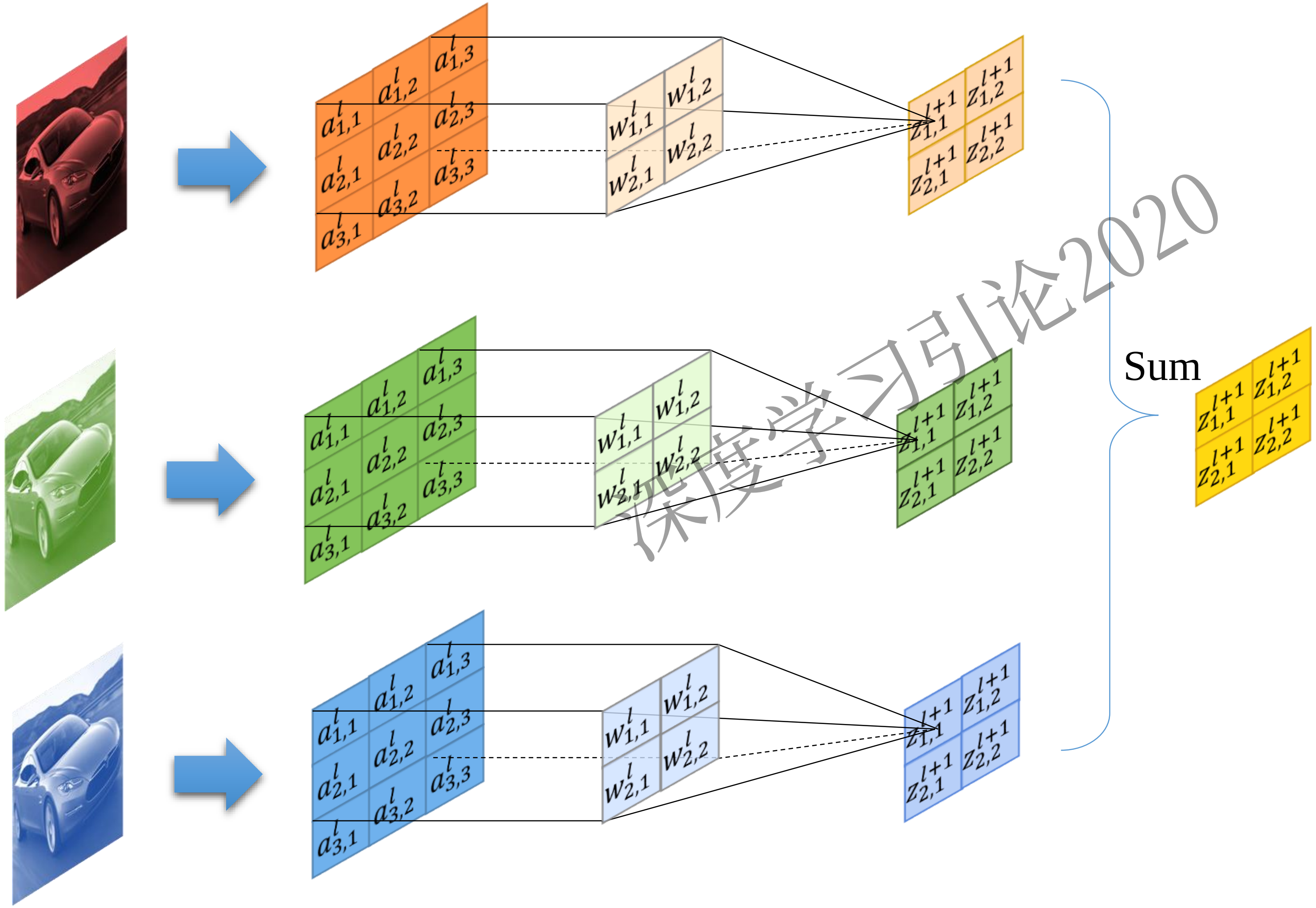
- The number of parameters in convolution is equal to the size of the kernel, which is 2×2 in this case. Much smaller than the fully-connected layer!

CNN’s Components



How to process the input composed of multiple channels?

CNN’s Components

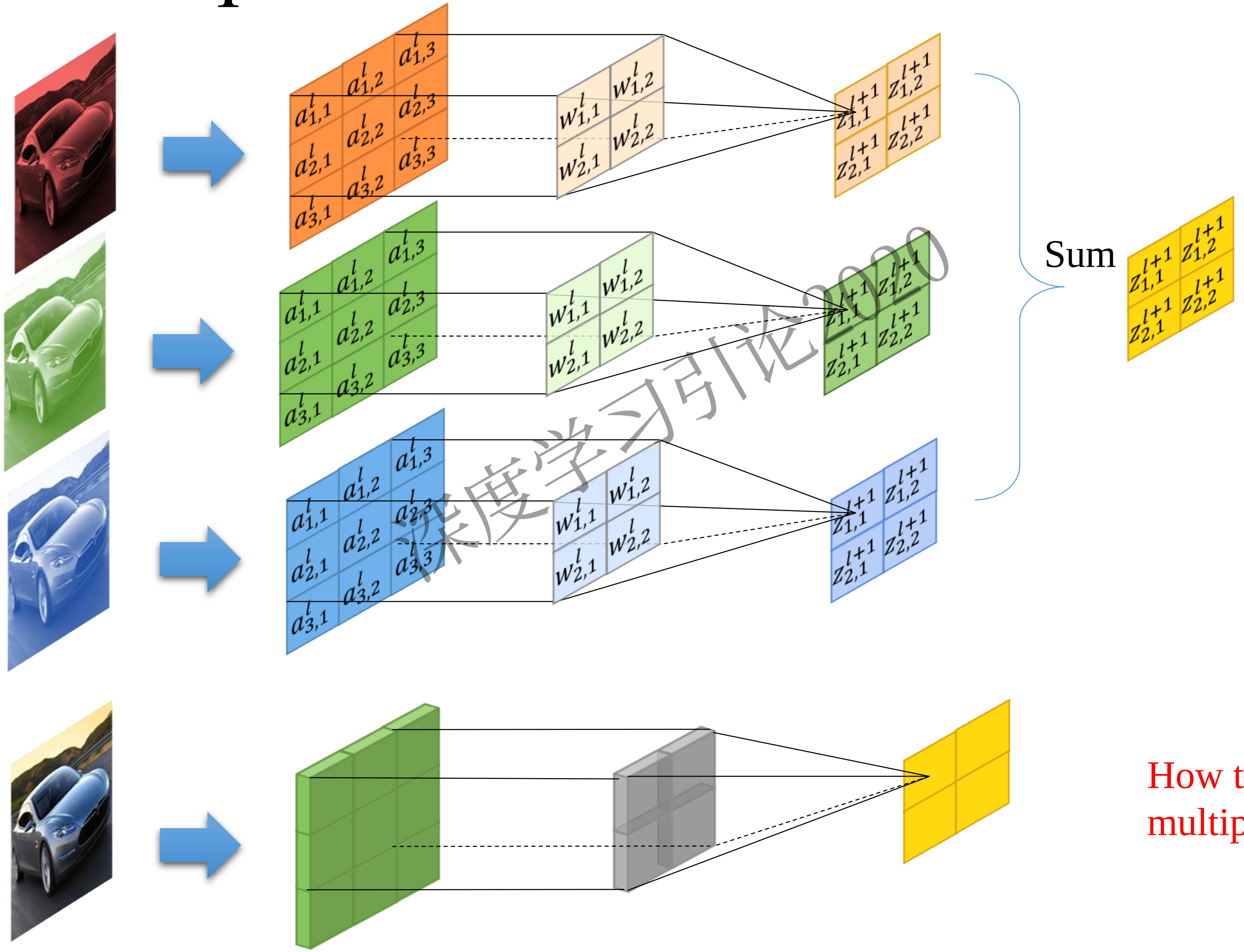


Step 1: Convolve the R, G, B channels independently.

Step 2: Sum the output of each convolution.

CNN’s Components

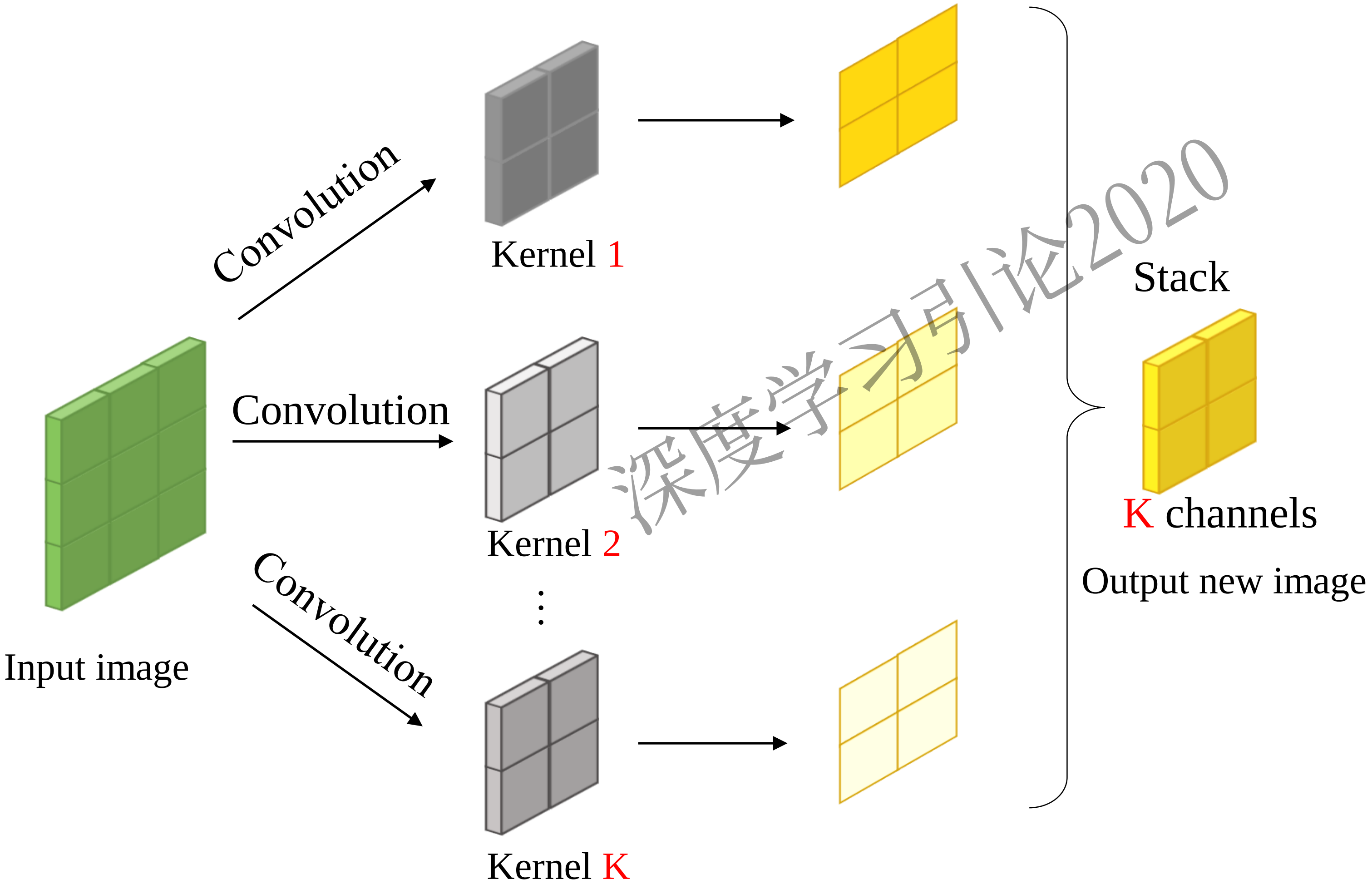
Channel-wise version



Stacked version

How to get the output with multiple channels?

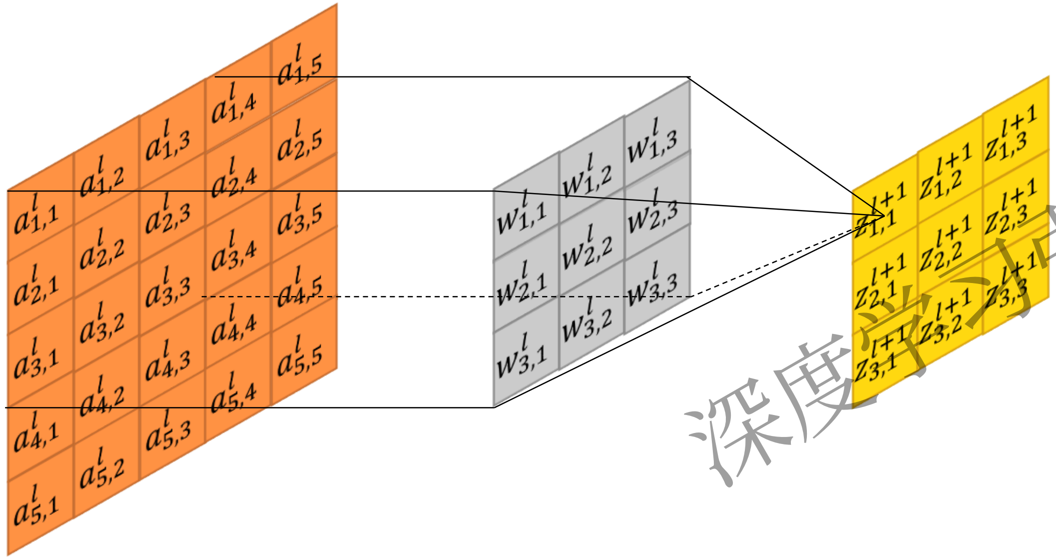
CNN's Components



Step 1: Use multiple kernels to convolve the input.

Step 2: Stack the output of each convolution. The number of kernels is equal to the output channel.

CNN's Components

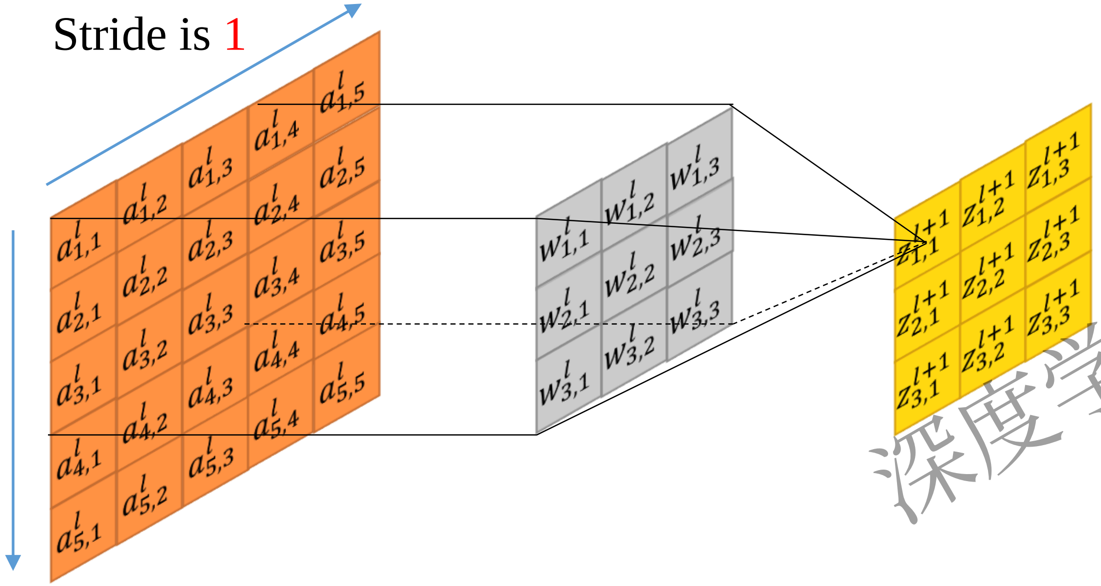


For notational simplicity, we will use the input with a single channel to demonstrate the components of CNNs.

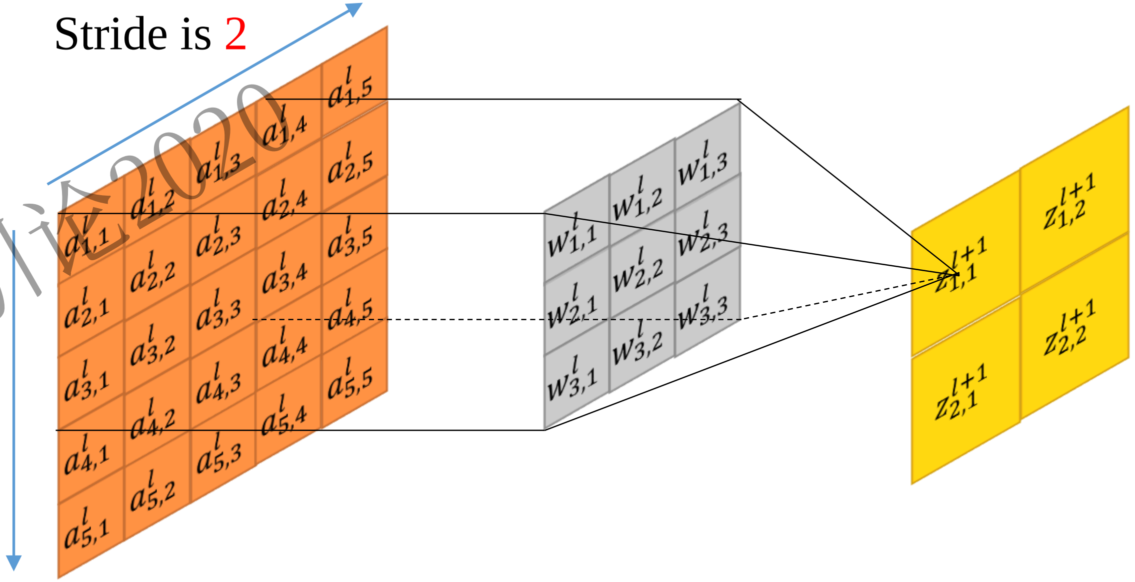
$$z^{l+1}_{n,m} = \sum_{i=1}^{I}\sum_{j=1}^{J} a^{l}_{n+j-1,m+i-1} \cdot w^{l}_{j,i}$$

CNN’s Components

Stride is 1



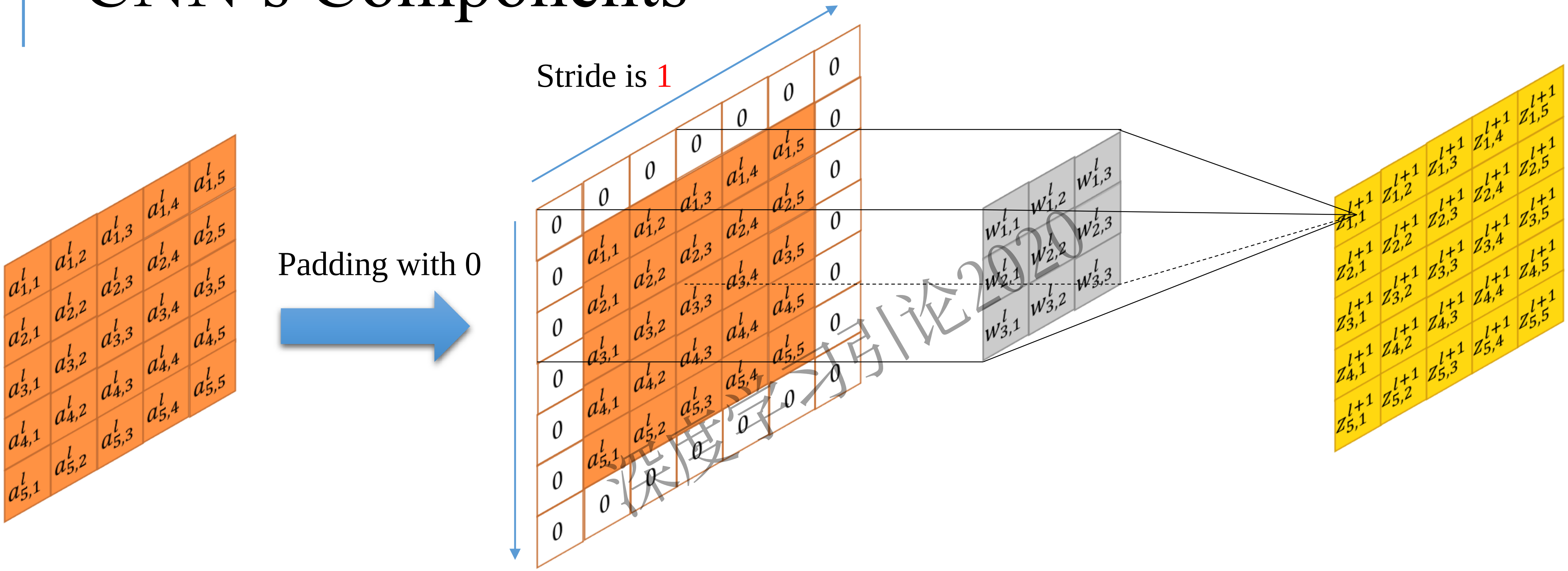
Stride is 2



- Stride controls the step of kernel.
- Note that the size of the output is smaller than the input, even when the stride is 1.

How to keep the size of output the same with that of input, to construct very deep CNNs?

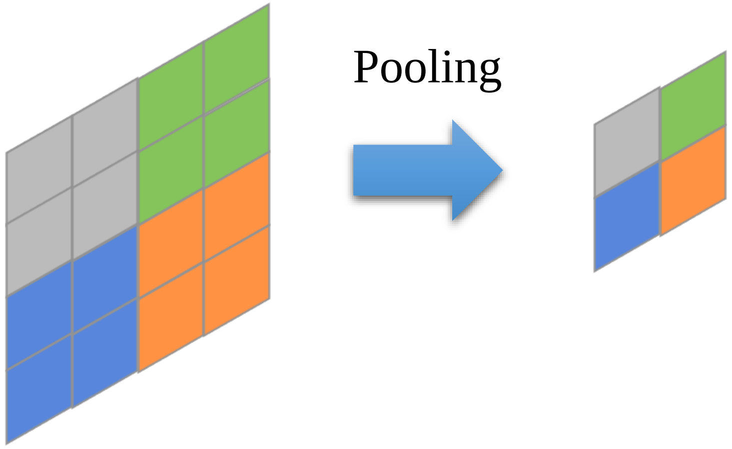
CNN's Components

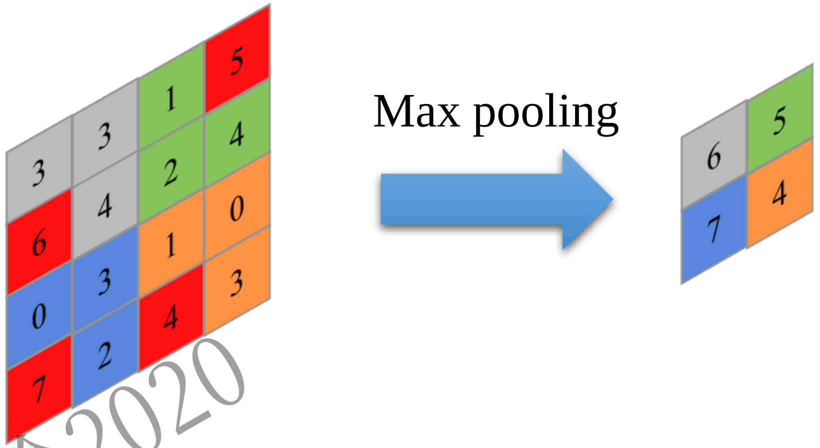


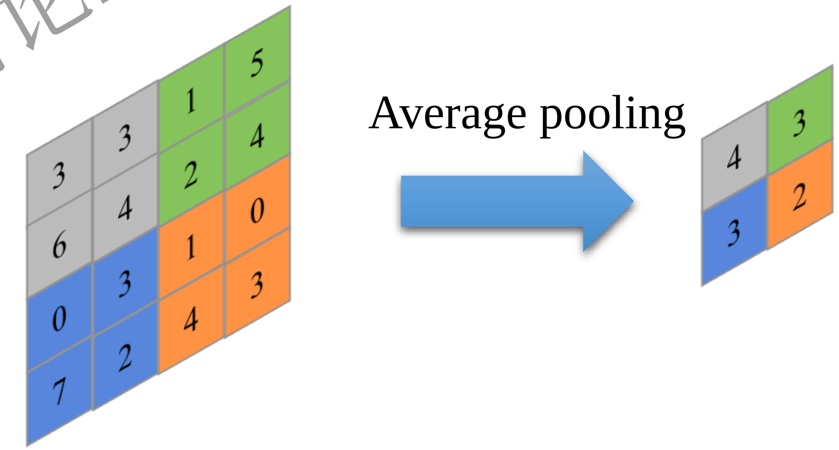
- Padding the input with a certain number of zeros will keep the size of the output unchanged.
- We can downsample the input by using convolution with stride > 1. This is at the cost of introducing more parameter w.

Is there an efficient way to downsample the input?

CNN’s Components



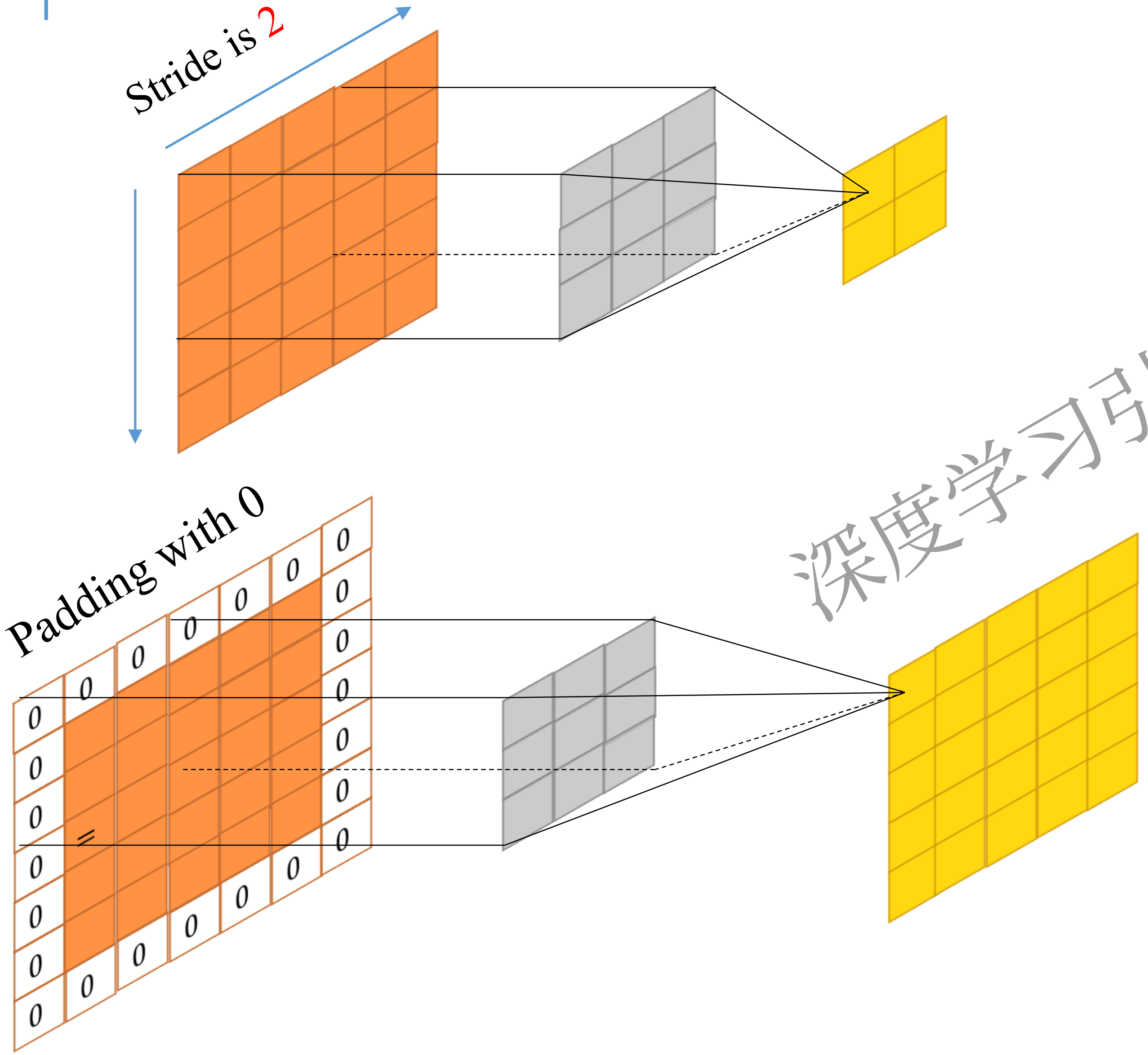


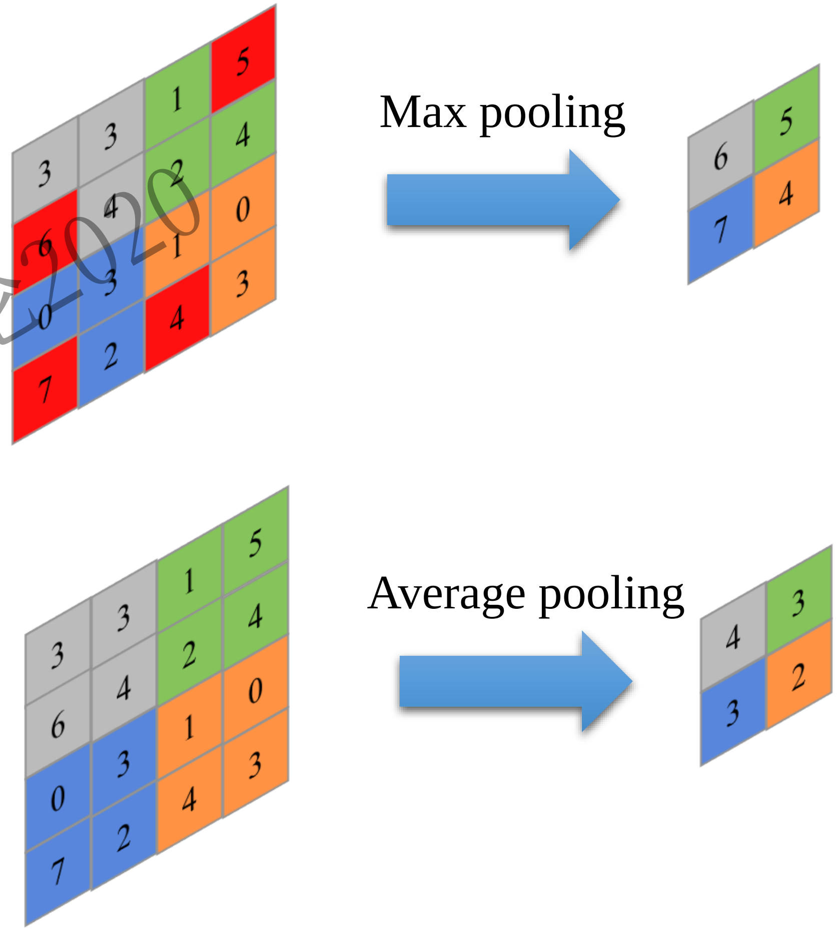


- Pooling aims to reduce the size of input efficiently, without introduce parameters w.

- Max pooling computes the max value in the receptive field.
- Average pooling computes the average value in the receptive field.

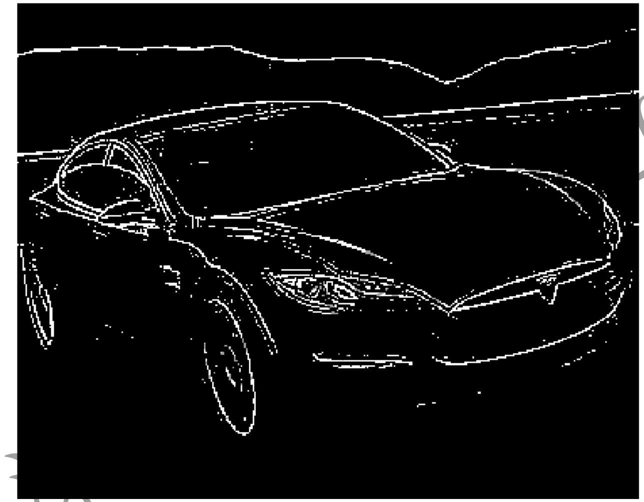
CNN’s Components





The Effectiveness of Kernel

Raw image



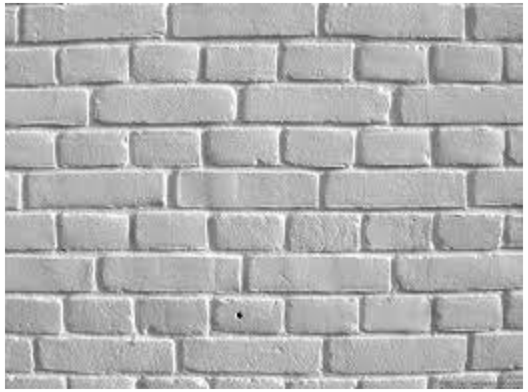
Edge detected by $w1$

Blurred by $w2$

$$w1 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

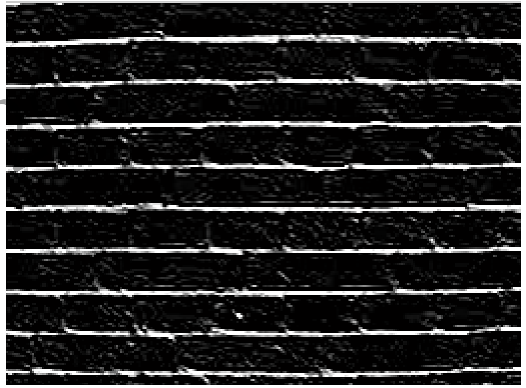
$$\mathrm{w}2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The Effectiveness of Kernel



Raw Image

Vertical edge detected by w1

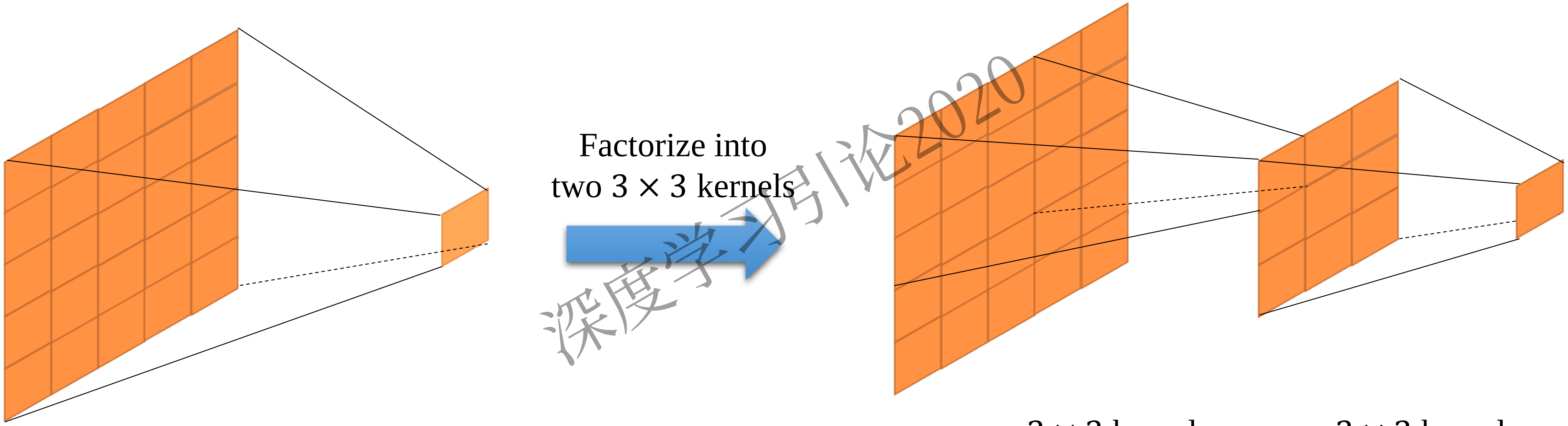


Horizontal edge detected by w2

$$w1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{w2=}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Factorization of Convolution Kernel



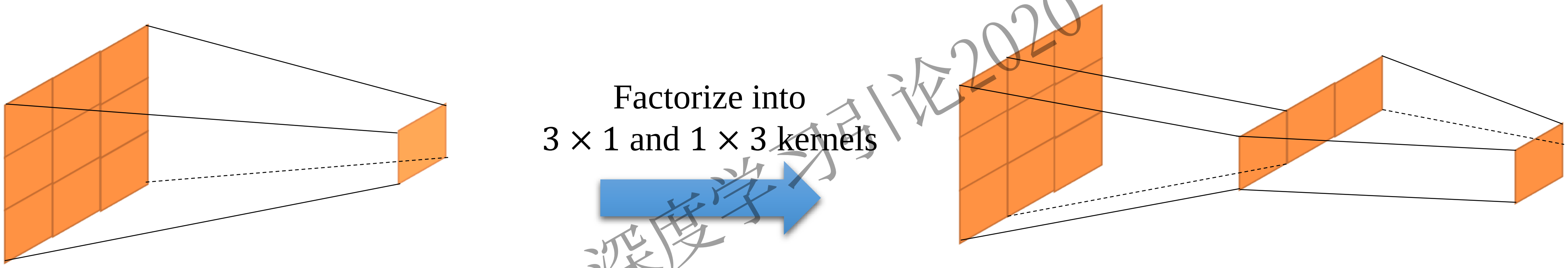
3 × 3 kernel

3 × 3 kernel

5 × 5 kernel
25 parameters

Total of 18 parameters

Factorization of Convolution Kernel



3×3 kernel

9 parameters

3×1 kernel 1×3 kernel

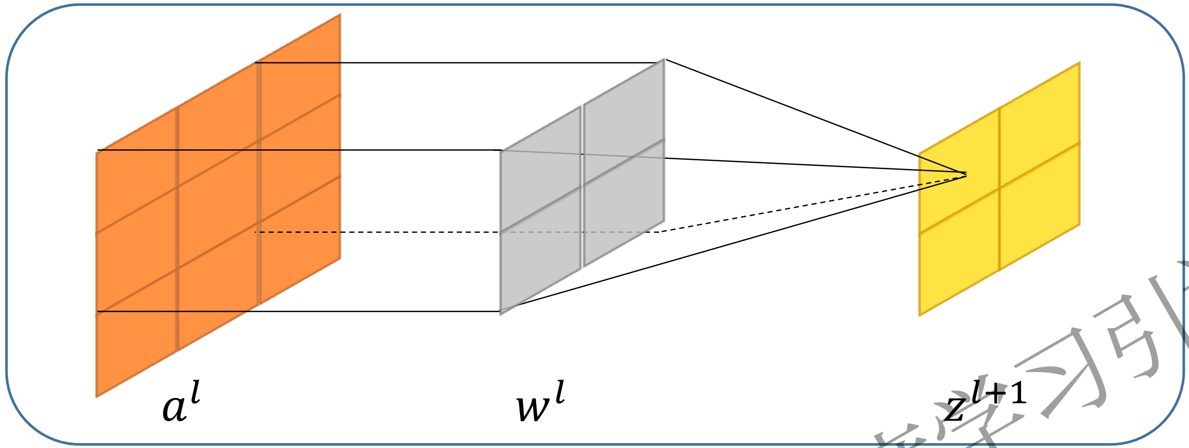
Total of 6 parameters

Outline

- CNNs’ Motivation
- CNNs’ Components
- CNNs’ Structure
- CNNs’ Learning
- CNNs’ Application

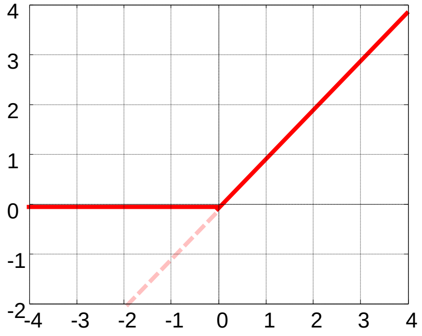
CNN's Components

1. Convolutional layer

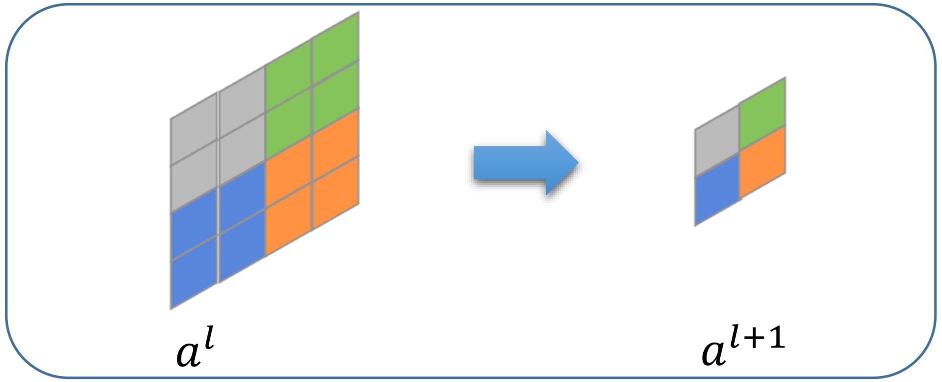


\+ ReLU

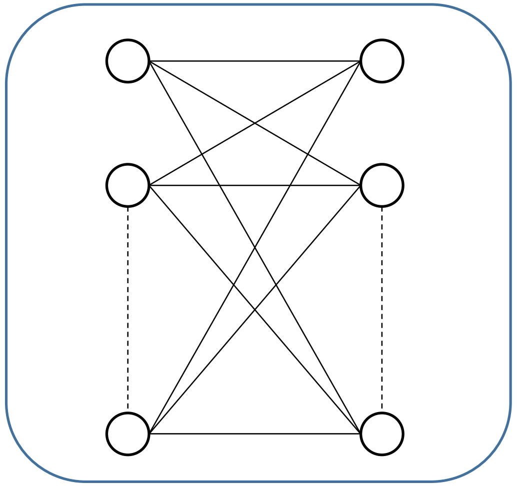
$$f(z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases}$$



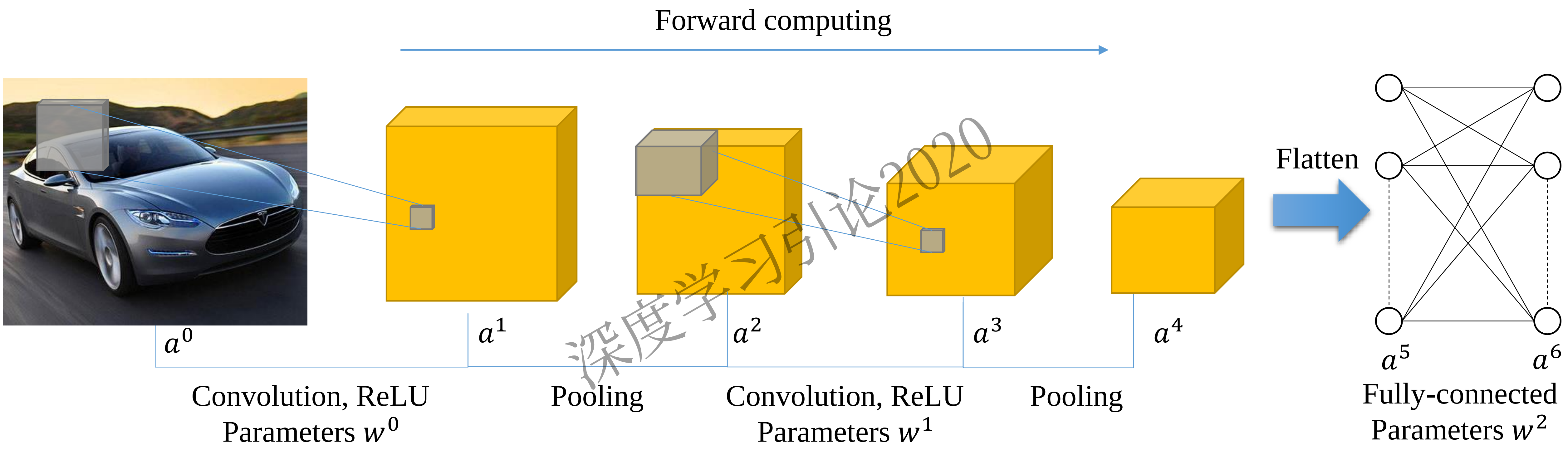
2. Pooling layer



3. Fully-connected layer



CNN's Structure

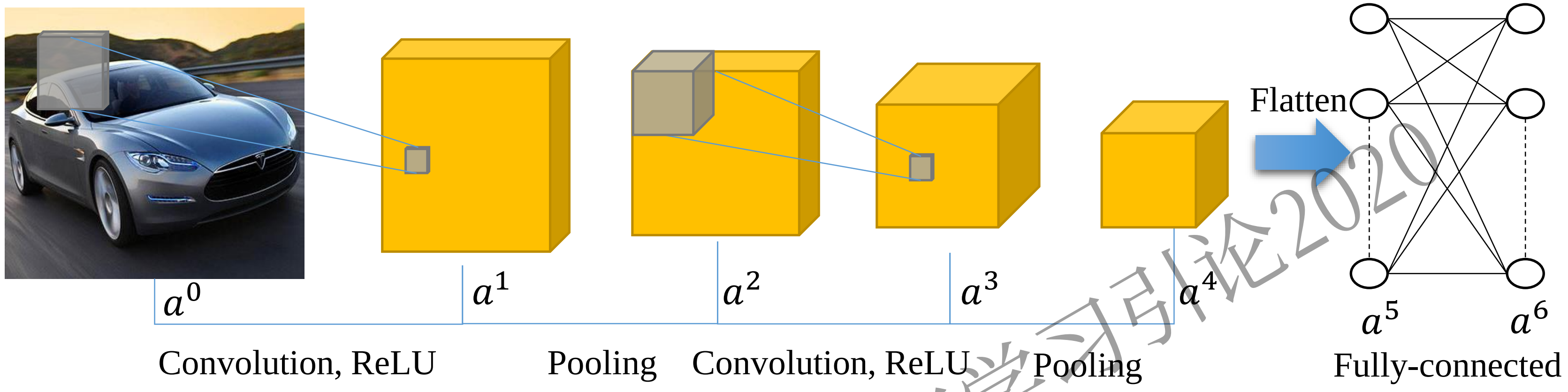


Problem: How to update the parameters in convolutional and fully-connected layers?

Outline

- CNNs’ Motivation
- CNNs’ Components
- CNNs’ Structure
- CNNs’ Learning
- CNNs’ Application

CNN’s Learning

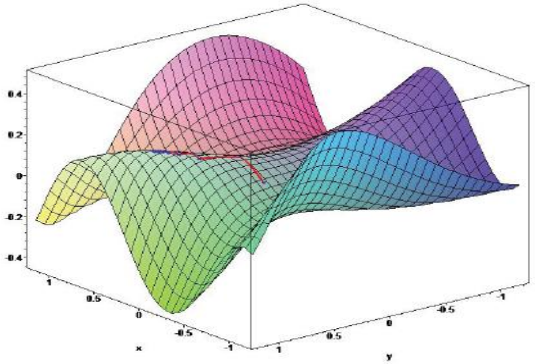


Network Output

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

Target Output

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$



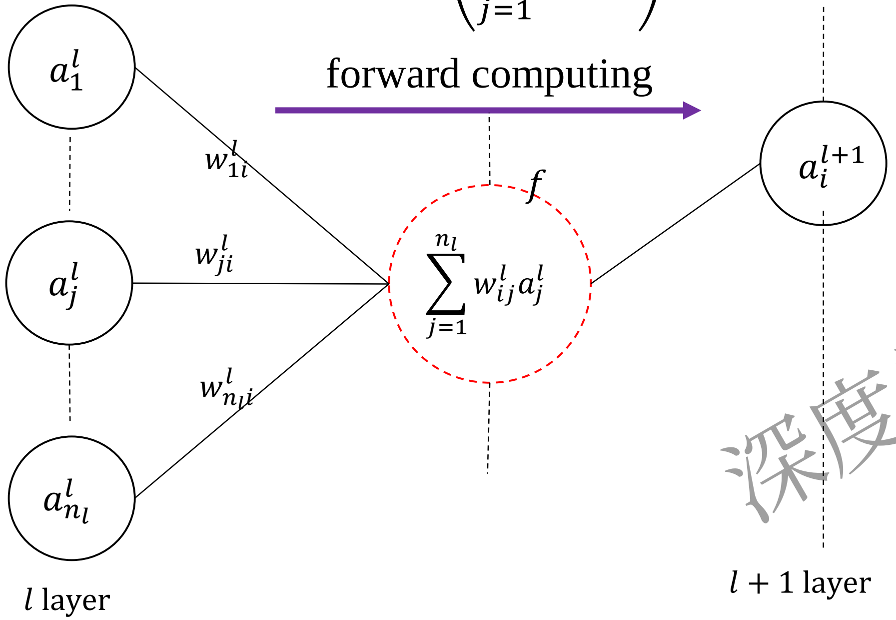
Cost function: $J = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j^L\right)^2 = J(w^1, \cdots, w^L)$

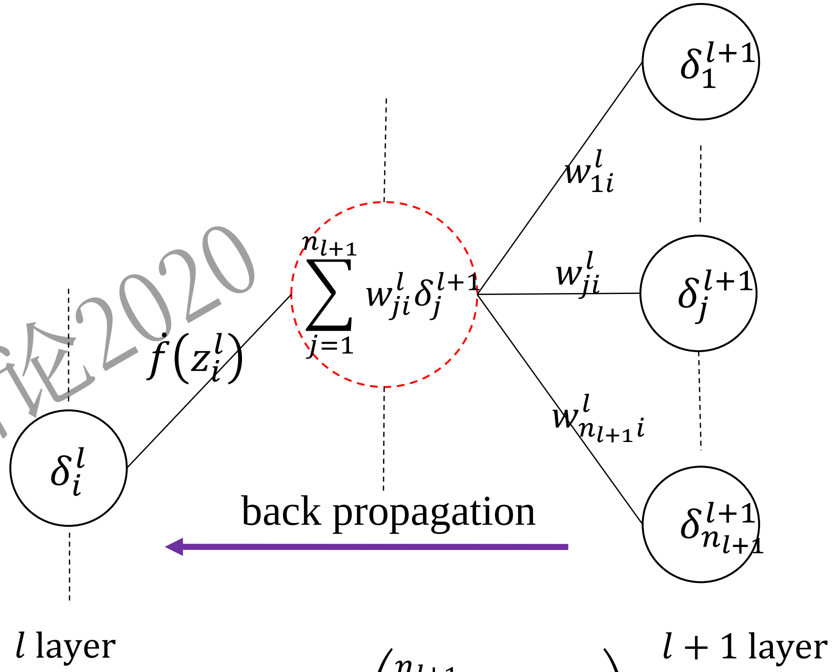
Gradient descend method: $w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$

Use backpropagation to compute $\frac{\partial J}{\partial w_{ji}^l}$ for the fully-connected and convolutional layers.

CNN’s Learning

$$a_i^{l+1} = f\left(\sum_{j=1}^{n_l} w_{ij}^l a_j^l\right)$$





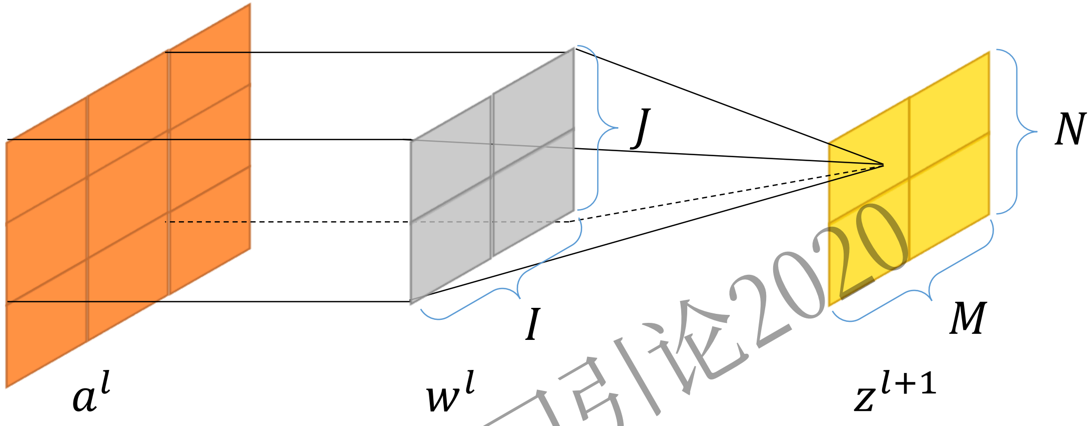
$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} w_{ji}^l \delta_j^{l+1}\right)$$

$$J = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j^L\right)^2$$

$$\delta_i^L = \frac{\partial J}{\partial z_i^L} = \left(a_i^L - y_i^L\right) \cdot \frac{\partial a_j^L}{\partial z_i^L} = \left(a_i^L - y_i^L\right) \cdot \dot{f}(z_i^L)$$

Relationship: $\frac{\partial J}{\partial w_{ji}^l} = \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$

CNN's Learning

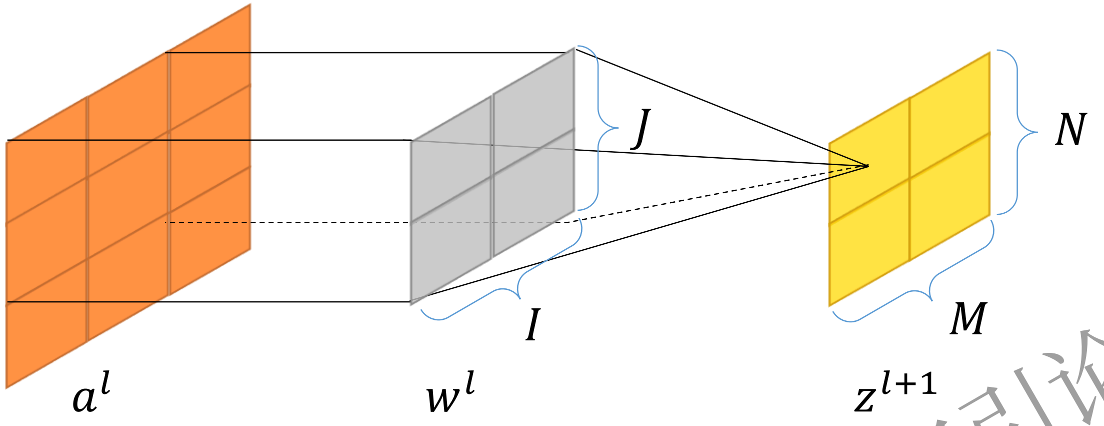


$$z_{n,m}^{l+1} = \sum_{i=1}^{I}\sum_{j=1}^{J} a_{n+j-1,m+i-1} \cdot w_{j,i}^{l}$$

- How to compute $\frac{\partial J}{\partial w_{j,i}^{l}}$ of the convolutional layer ?

The same with the fully-connected layer!

CNN's Learning



$$z_{n,m}^{l+1} = \sum_{i=1}^{I}\sum_{j=1}^{J} a_{n+j-1,m+i-1}^{l} \cdot w_{j,i}^{l}$$

Relationship: $\frac{\partial J}{\partial w_{j',i'}^{l}} = \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\partial J}{\partial z_{n,m}^{l+1}} \cdot \frac{\partial z_{n,m}^{l+1}}{\partial w_{j',i'}^{l}} = \sum_{m=1}^{M}\sum_{n=1}^{N} \delta_{n,m}^{l+1} \cdot \boxed{\frac{\partial z_{n,m}^{l+1}}{\partial w_{j',i'}^{l}}}$

$$\frac{\partial z_{n,m}^{l+1}}{\partial w_{j',i'}^{l}} = \frac{\partial \sum_{i=1}^{I}\sum_{j=1}^{J} a_{n+j-1,m+i-1}^{l} \cdot w_{j,i}^{l}}{\partial w_{j',i'}^{l}} = \begin{cases} a_{n+j'-1,m+i'-1}^{l}, if\ i = i'\ and\ j = j' \\ 0, otherwise \end{cases}$$

$$\frac{\partial J}{\partial w_{j',i'}^{l}} = \sum_{m=1}^{M}\sum_{n=1}^{N} a_{n+j'-1,m+i'-1}^{l} \cdot \delta_{n,m}^{l+1}$$

Convolution!

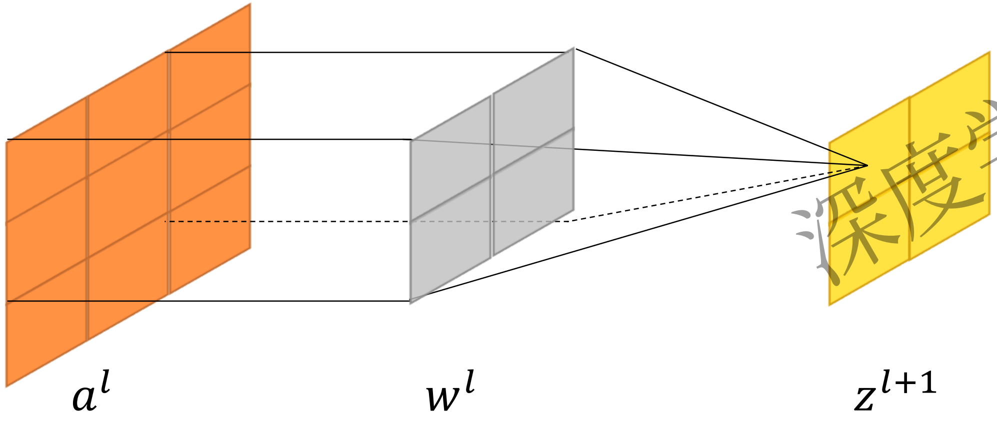
CNN’s Learning

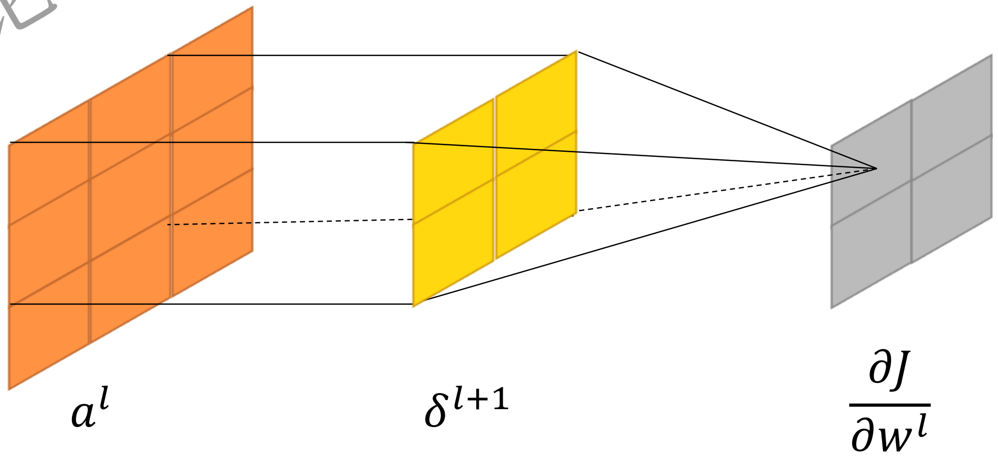
Forward computing

$$z^{l+1}_{n,m} = \sum_{i=1}^{I}\sum_{j=1}^{J} a^{l}_{n+j-1,m+i-1} \cdot w^{l}_{j,i}$$

Relationship

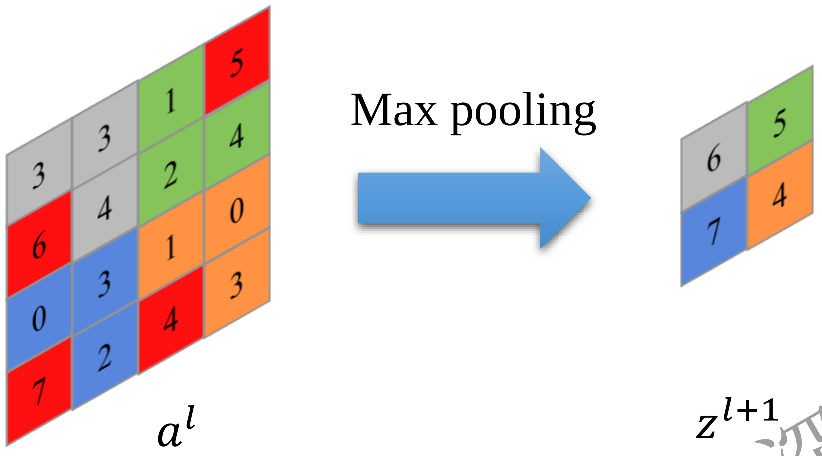
$$\frac{\partial J}{\partial w^{l}_{j,i}} = \sum_{m=1}^{M}\sum_{n=1}^{N} a^{l}_{n+j-1,m+i-1} \cdot \delta^{l+1}_{n,m}$$

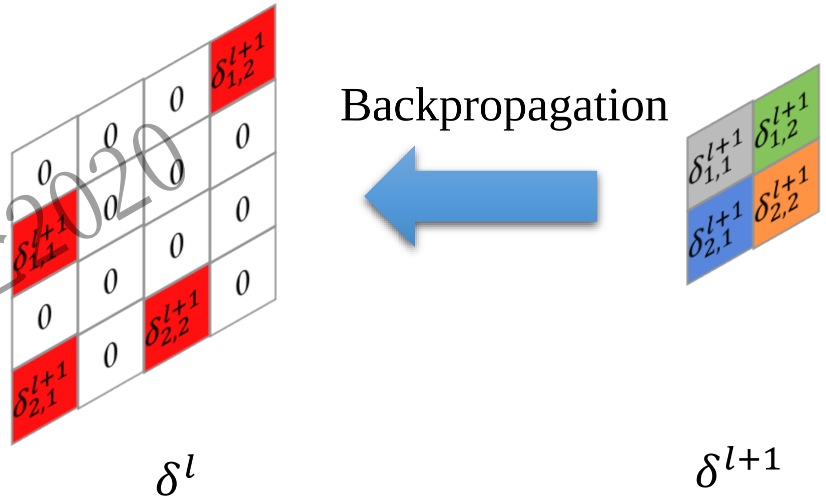




How to backpropagate δ in pooling and convolutional layers?

CNN’s Learning

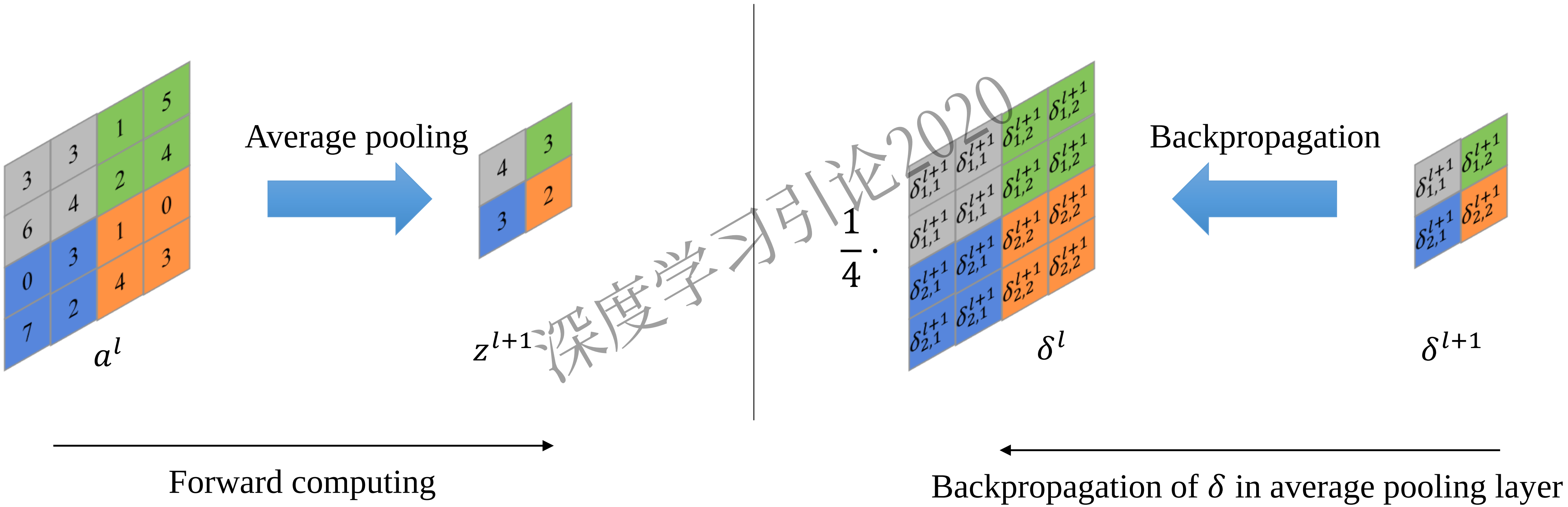




Forward computing

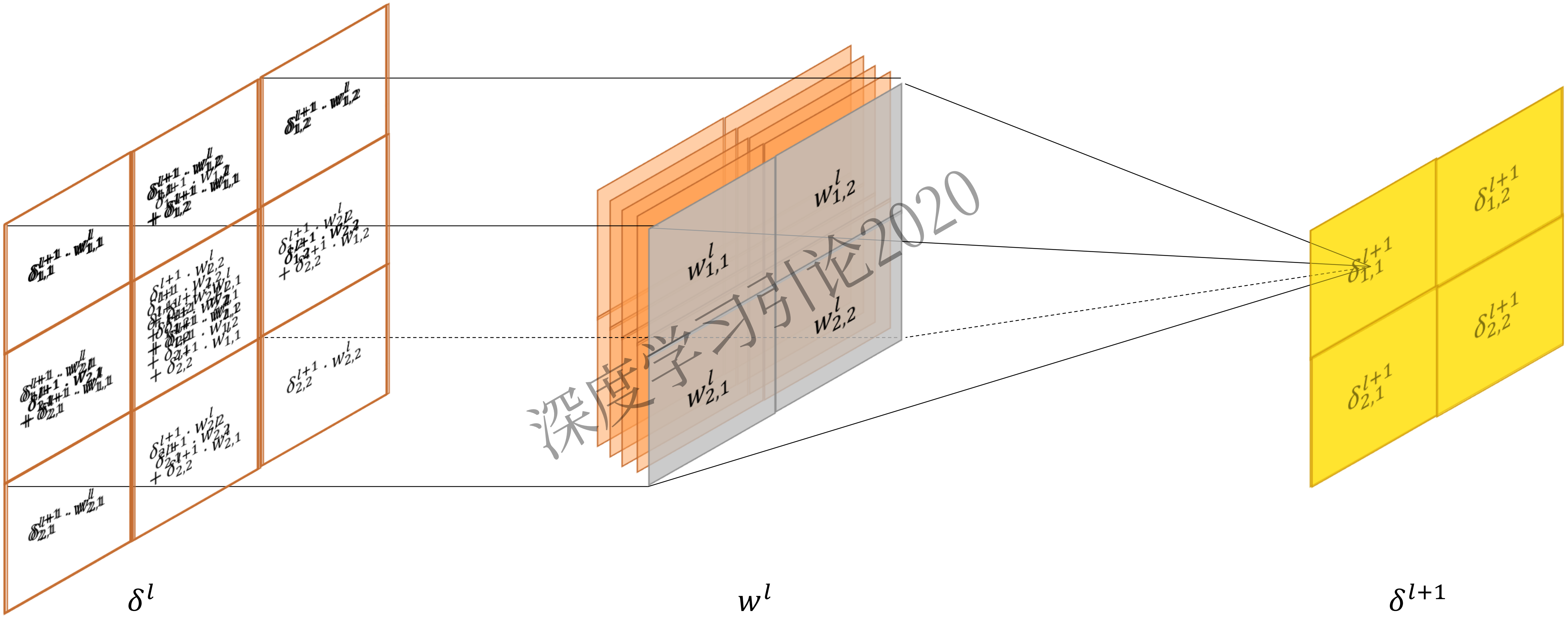
Backpropagation of δ in max pooling layer

CNN's Learning



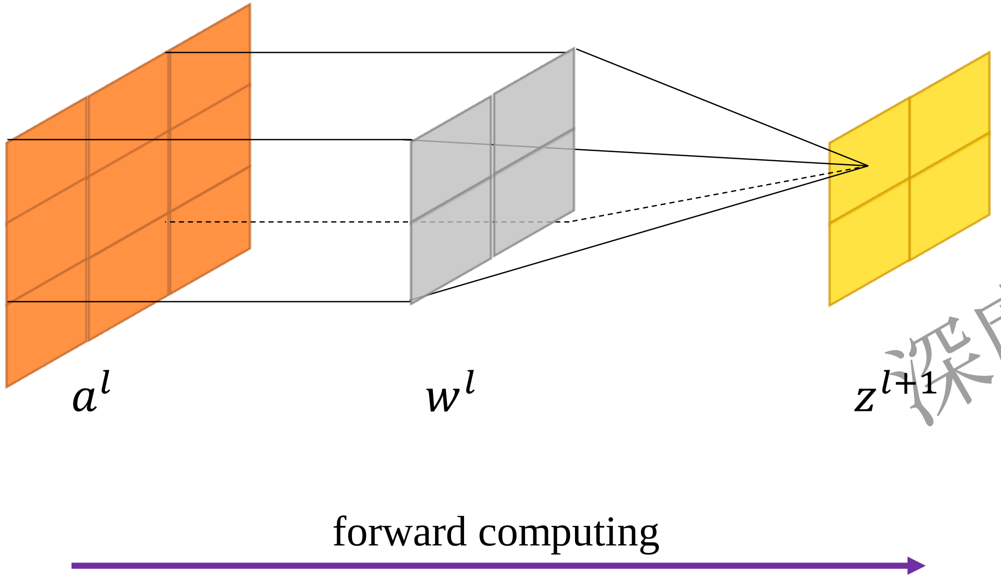
How to backpropagate δ in convolutional layer?

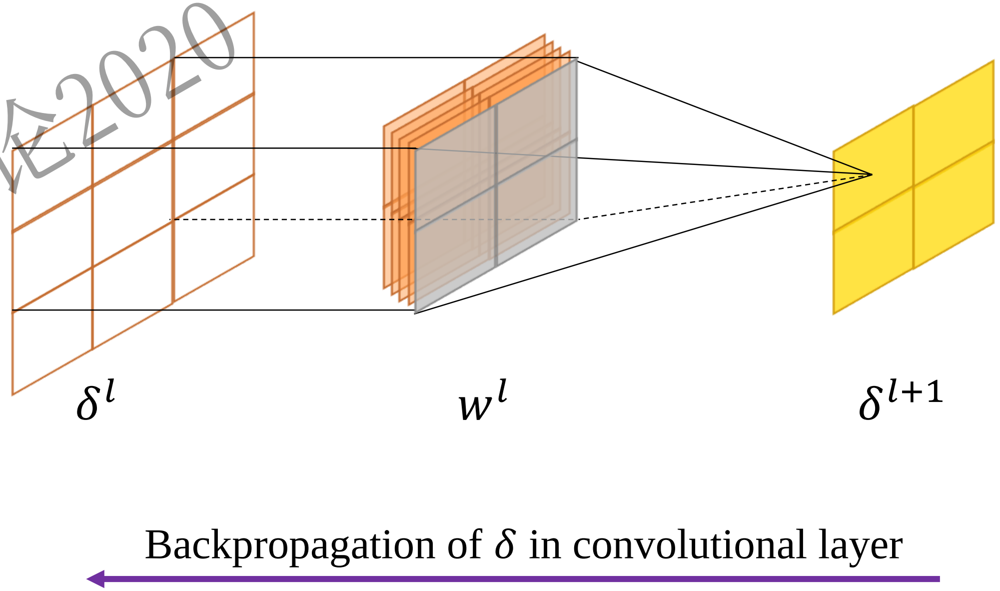
CNN's Learning



Backpropagation of δ in convolutional layer

CNN's Learning

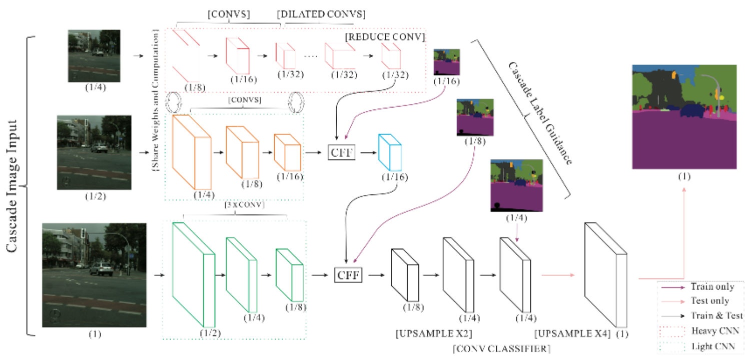




Outline

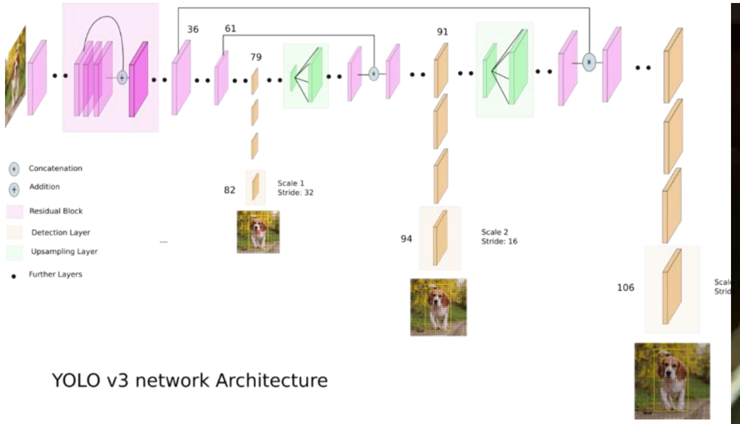
- CNN’s Motivation
- CNN’s Components
- CNN’s Structure
- CNN’s Learning
- CNN’s Application

Real-time Semantic Segmentation



- Real-time semantic segmentation

Real-time Object Detection



YOLO v3 network Architecture

- Real-time object detection

Thanks