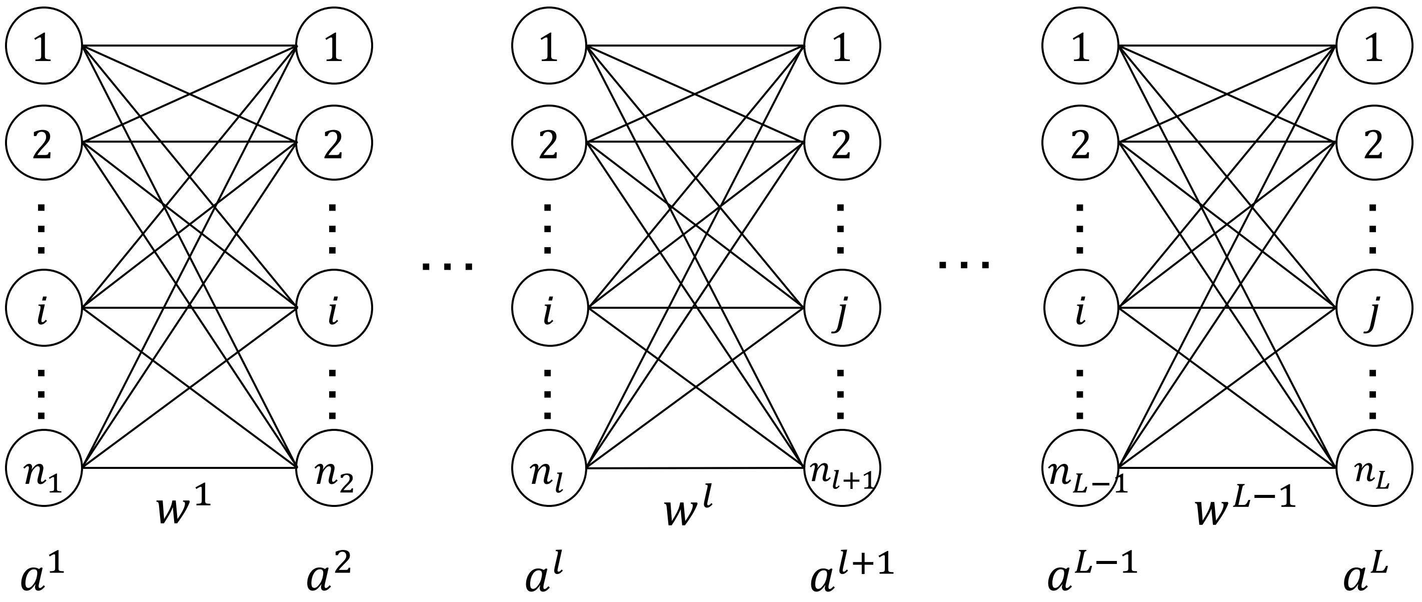
机器学习-第八章 卷积神经网络简介

教师：胡俊杰 助理研究员

邮箱：hujunjie@scu.edu.cn

3. 反向传播



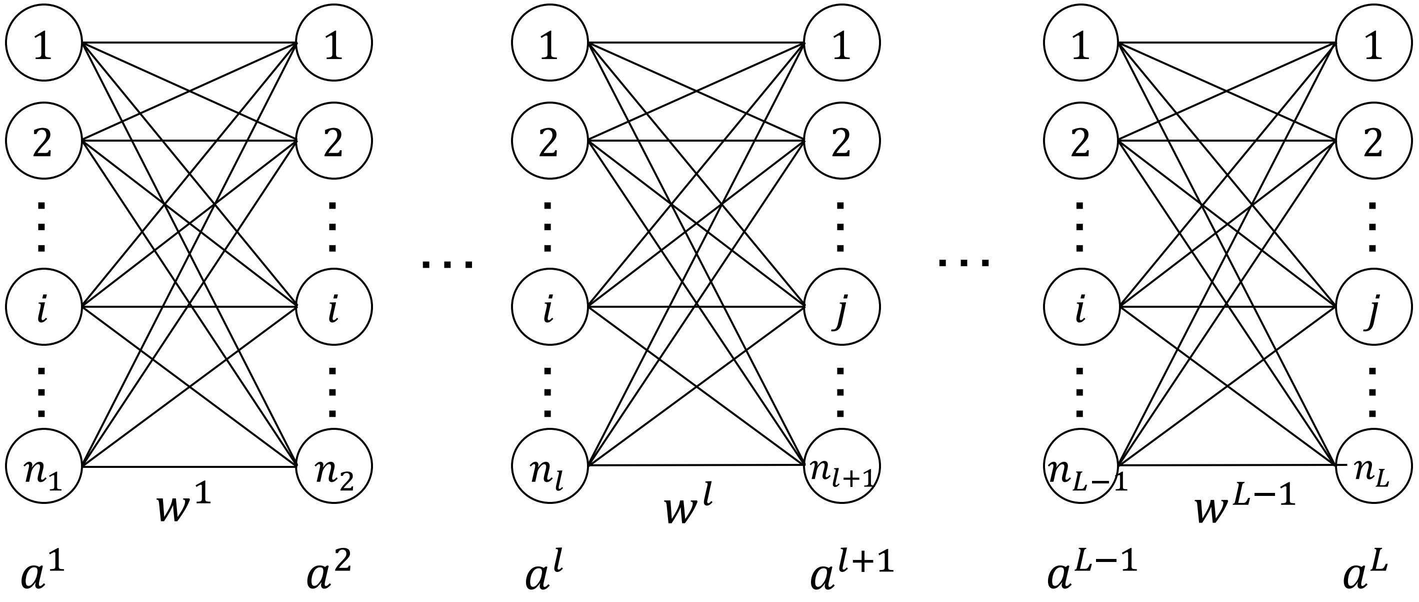
- a^1: 输入，a^L: 输出

- 考虑训练数据样本(x,y)，其中a^1即对应x，二者维度相等
- 标签y采用one-hot编码，假设有y共包含10个类别，则各类别的标签表达形式如下

$$
\begin{array}{cccccccccc}
\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix} \\
\text{类}1 & \text{类}2 & \text{类}3 & \text{类}4 & \text{类}5 & \text{类}6 & \text{类}7 & \text{类}8 & \text{类}9 & \text{类}10
\end{array}
$$

- a^L的维度与y的维度相等

3. 反向传播



- a^1: 输入，a^L: 输出

$\boxed{\text{输出：} a^L} \xrightarrow{\text{距离}} \boxed{\text{标签：} y}$

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_L} \end{bmatrix}$$

- 损失函数（代价函数/性能函数）J
- J度量输出a^L与标签y的距离
- J是关于$(w^1, \cdots, w^{L-1})$的函数，$J = J(w^1, \cdots, w^{L-1})$

均方误差
(Mean Squared Error ,MSE)

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

目标：最小化J

3. 反向传播

随机梯度下降算法

第1步. 准备训练数据集 $D=\{(x,y)\}$

第2步. 随机初始化神经网络各层参数$(w^1,w^2,\dots,w^{L-1})$, 设置学习率 α.

第3步. 随机选择b个样本（一个batch），计算并累积各样本对各层参数的梯度$\frac{\partial J}{\partial w_{ji}^l}$

第4步. 更新参数

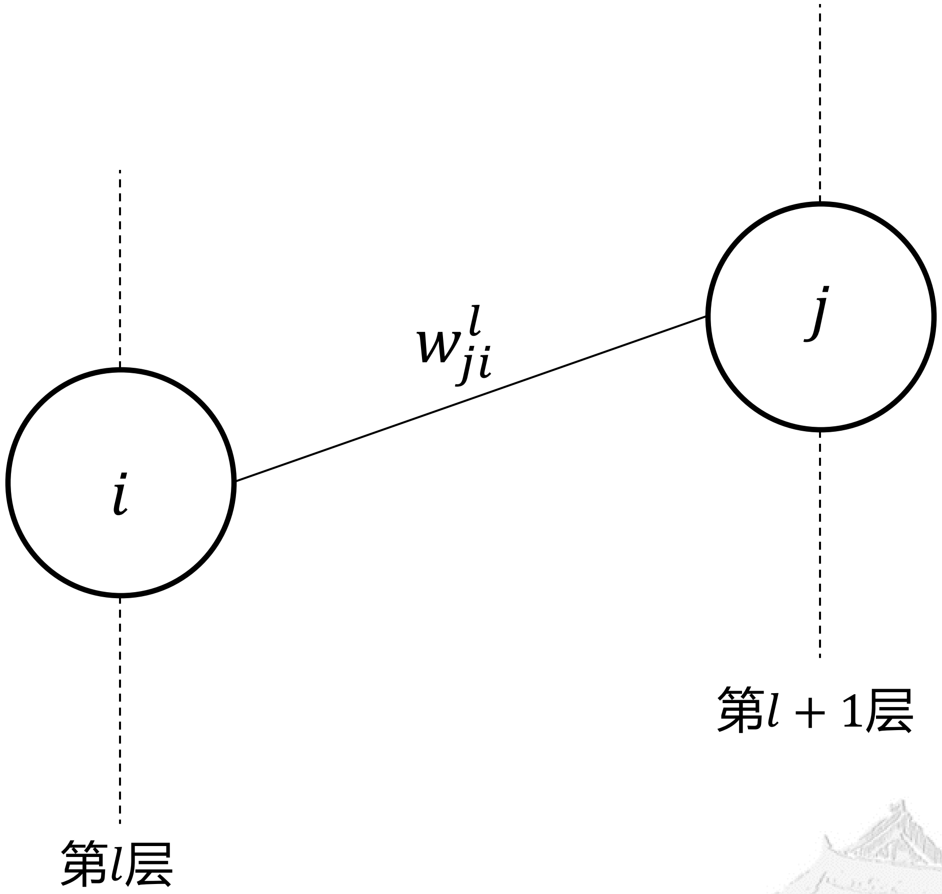
$$w_{ji}^l := w_{ji}^l - \alpha \frac{1}{b}\frac{\partial J}{\partial w_{ji}^l}$$

第5步. 继续第3步，直到模型收敛.

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

$\boxed{\text{反向传播（backpropagation）算法求}\frac{\partial J}{\partial w_{ji}^l}}$

3. 反向传播



前向计算
$$\begin{cases} z_j^{l+1} = \sum_{i=1}^{n_l} w_{ji}^{l} a_i^{l} \\ a_j^{l+1} = f\left(z_j^{l+1}\right) \end{cases}$$

- 定义敏感系数$\delta_i^l = \frac{\partial J}{\partial z_i^l}$
- 链式法则:

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

- a_i^l由前向计算可求得，如何求δ_j^{l+1}，即各层神经元的敏感系数

3. 反向传播

- 最后一层神经元的敏感系数δ_i^L

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

$$\delta_i^L=\frac{\partial J}{z_i^L}=?$$

3. 反向传播

- 最后一层神经元的敏感系数δ_i^L

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

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

3. 反向传播

- 隐藏层神经元的敏感系数δ_i^l

$$z_j^{l+1}=\sum_{i=1}^{n_l} w_{ji}^l a_i^l=\sum_{i=1}^{n_l} w_{ji}^l f\left(z_i^l\right)$$

$$\delta_i^l=\frac{\partial J}{\partial z_i^l}=\sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial z_i^l}=?$$

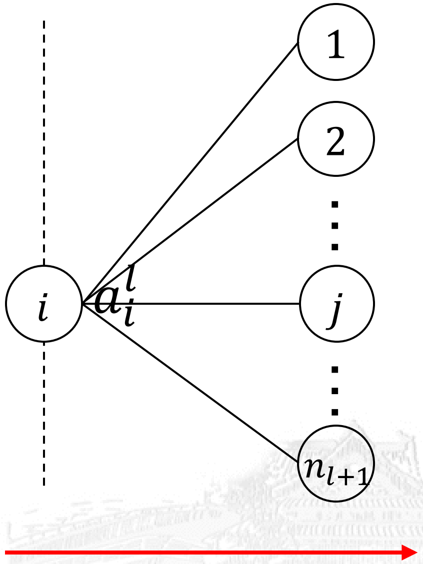
链式法则

3. 反向传播

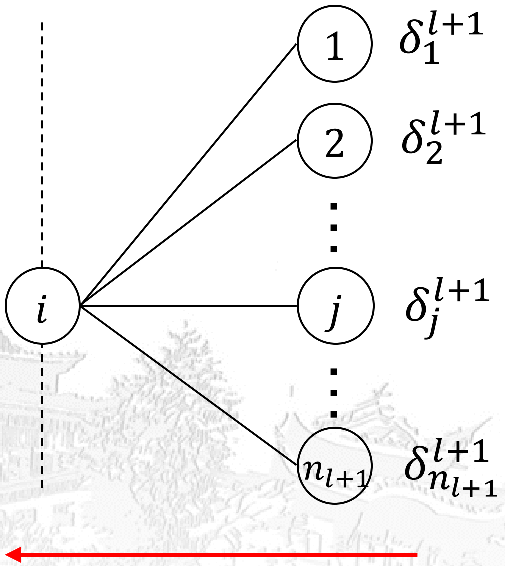
■ 隐藏层神经元的敏感系数δ_i^l

$$z_j^{l+1}=\sum_{i=1}^{n_l} w_{ji}^l a_i^l=\sum_{i=1}^{n_l} w_{ji}^l f\left(z_i^l\right)$$

$$\delta_i^l=\frac{\partial J}{\partial z_i^l}=\underbrace{\sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial z_i^l}}_{\text{链式法则}}=\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l}=\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^l\, \dot{f}\left(z_i^l\right)=\dot{f}\left(z_i^l\right)\left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^l\right)$$



前向计算



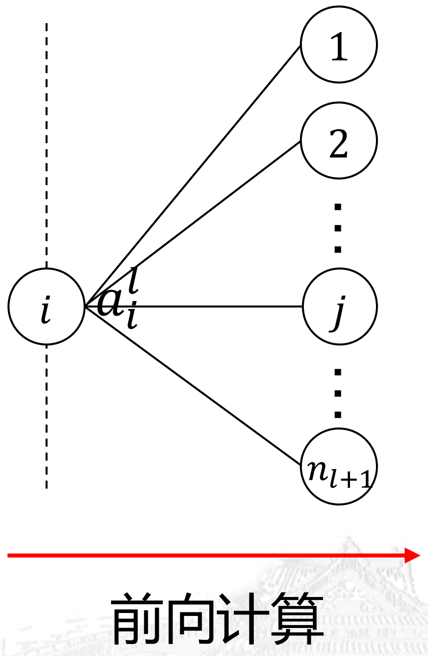
反向传播

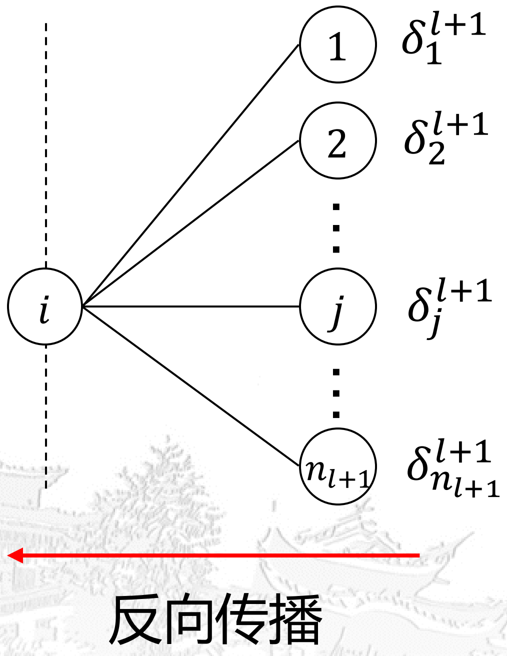
3. 反向传播

■ 前向计算与反向传播对比

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

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





3. 反向传播

总结

- 反向传播是一种用于计算$\frac{\partial J}{\partial w_{ji}^{l}}$的算法

- δ_j^{l+1}与$\frac{\partial J}{\partial w_{ji}^{l}}$的关系： $\frac{\partial J}{\partial w_{ji}^{l}} = \frac{\partial J}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial w_{ji}^{l}} = \delta_j^{l+1} a_i^{l}$

通过计算δ_j^{l+1}从而得到$\frac{\partial J}{\partial w_{ji}^{l}}$

- δ_j^{l+1}与δ_i^{l}的关系： $\delta_i^{l} = \dot{f}(z_i^{l}) \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^{l} \right)$

δ_j^{l+1}的反向传播

3. 反向传播

标量形式

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

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

向量形式

- $\frac{\partial J}{\partial w^{l}} \in R^{n_{l+1} \times n_l}, a^{l} \in R^{n_l \times 1}, \delta^{l} \in R^{n_l \times 1}$

请写出左侧两个公式的向量形式

3. 反向传播

■ 标量形式

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

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

■ 向量形式

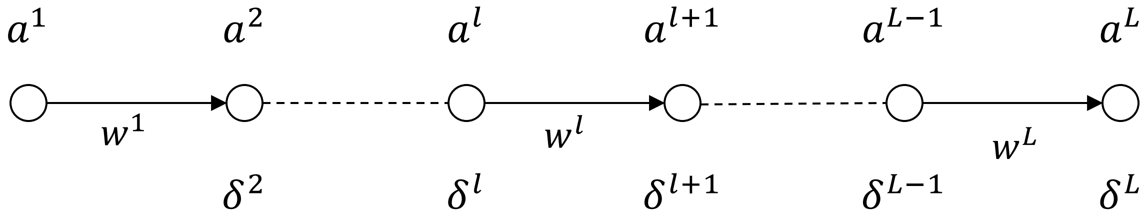
- $\frac{\partial J}{\partial w^l} \in R^{n_{l+1} \times n_l}, a^l \in R^{n_l \times 1}, \delta^l \in R^{n_l \times 1}$

$$\frac{\partial J}{\partial w^l} = \delta^{l+1} (a^l)^T$$

$$\delta^l = \dot{f}(z^l) \circ (w^l)^T \delta^{l+1}$$

$\circ$: element-wise product

4. 梯度消失问题

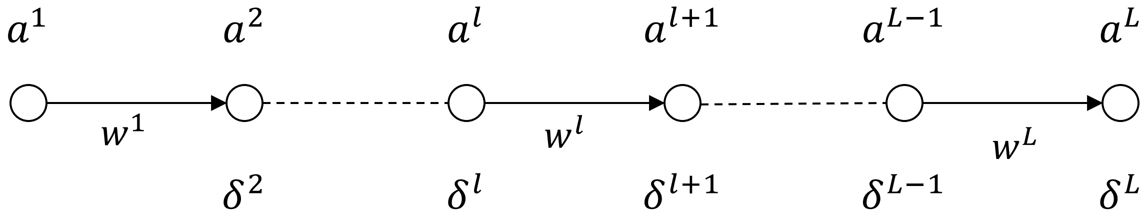


考虑每层仅有单个神经元

- 前向计算： $a^L = f\left(W^{L-1} f\left(W^{L-2} f\left(W^{L-3} \cdots f\left(W^1 a^1\right)\right)\right)\right)$

- 反向传播： $$\begin{aligned}\delta^l &= \dot{f}(z^l) w^l \delta^{l+1} \\ &= \dot{f}(z^l) w^l \dot{f}(z^{l+1}) w^{l+1} \delta^{l+2} \\ &= \dot{f}(z^l) w^l \dot{f}(z^{l+1}) w^{l+1} \cdots \dot{f}(z^{L-1}) w^{L-1} \delta^L\end{aligned}$$

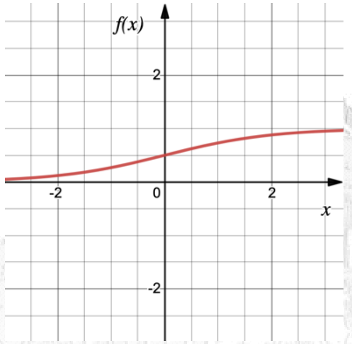
4. 梯度消失问题



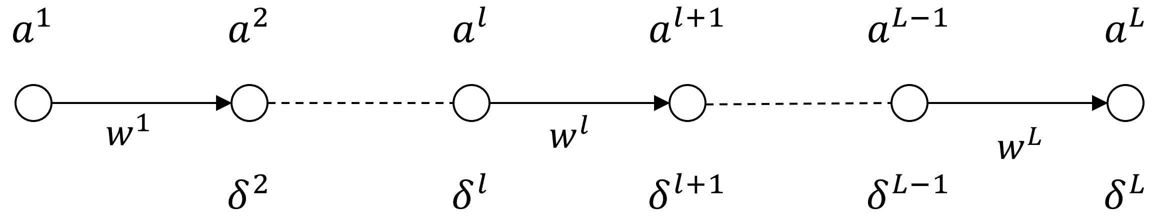
考虑每层单个神经元

- 反向传播： $\delta^l = \dot{f}(z^l)w^l\dot{f}(z^{l+1})w^{l+1}\cdots\dot{f}(z^{L-1})w^{L-1}\delta^L$ 梯度消失

- 当激活函数选择sigmoid $f(x) = \dfrac{1}{1+e^{-x}}$ $0 < \dot{f}(x) < 1$



4. 梯度消失问题



考虑每层单个神经元

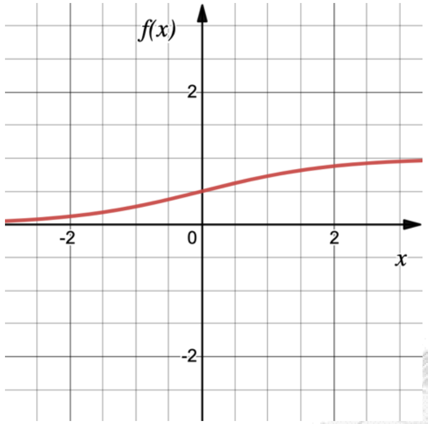
$$\delta^l = \dot{f}(z^l)w^l\dot{f}(z^{l+1})w^{l+1}\cdots\dot{f}(z^{L-1})w^{L-1}\delta^L$$

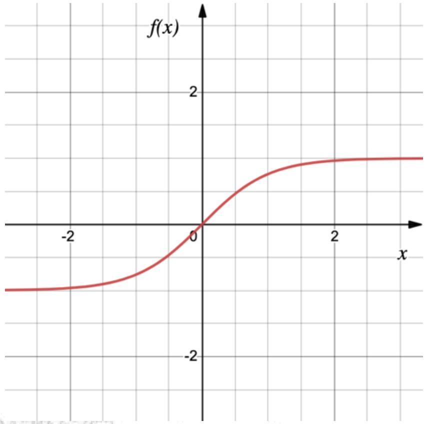
- $\dot{f}(x) = 1$，线性激活函数，得到的是线性模型，模型拟合能力弱
- $\dot{f}(x) \neq 1$，非线性激活函数，模型拟合能力强，但梯度消失/爆炸，训练困难

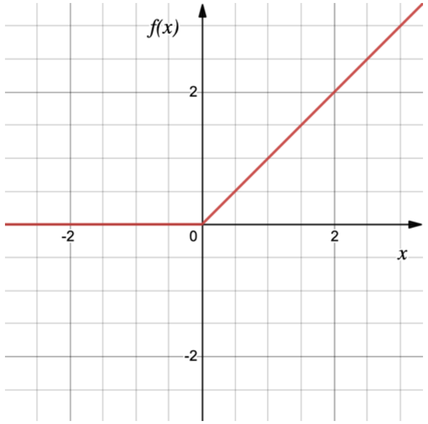
4. 梯度消失问题

■ 激活函数的角度

Sigmoid	Tanh	ReLU
$f(x)=\dfrac{1}{1+e^{-x}}$	$f(x)=\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$f(x)=\max(0,x)$

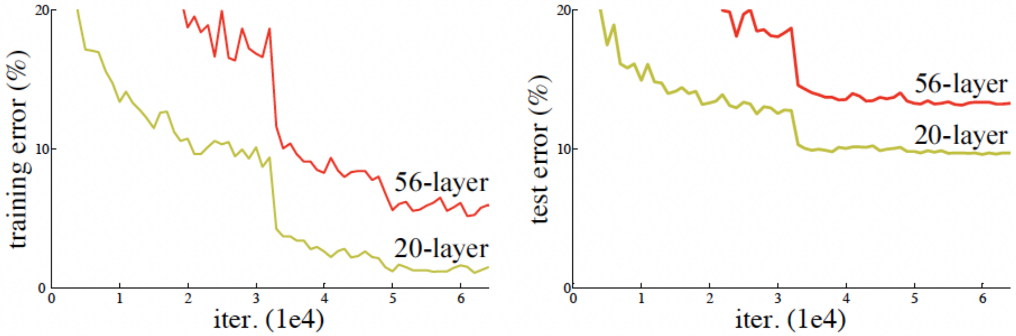




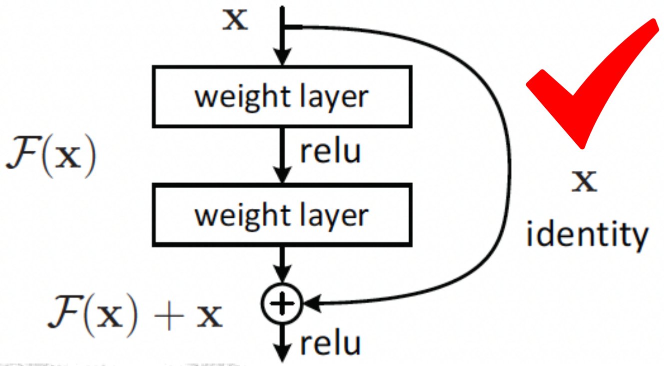


4. 梯度消失问题

■ 模型结构的角度



■ 模型深度并非越大越好



■ 传统深度神经网络模型：$F(x)$

■ 深度残差神经网络模型：$F(x) + x$

大纲

CNNs的起源

CNNs的组件

CNNs的结构

CNNs的学习

CNNs的应用

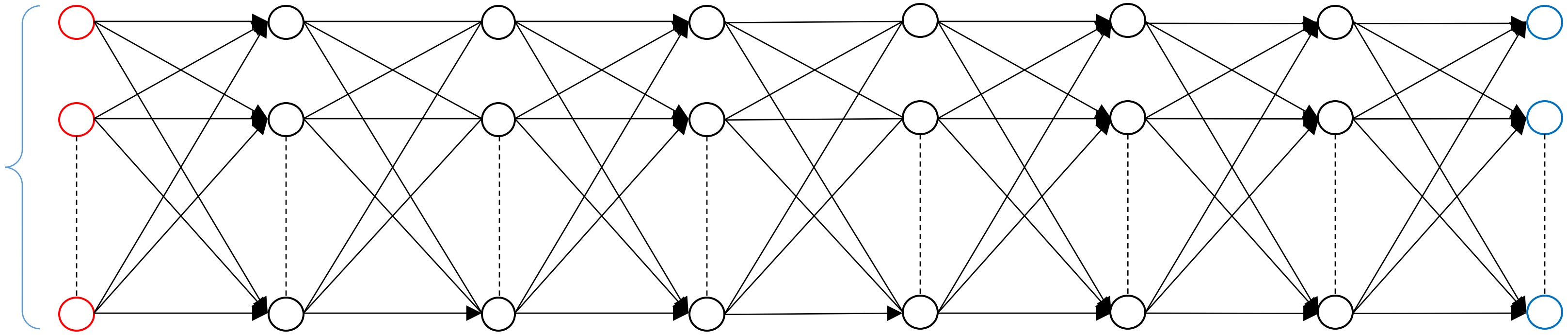
CNNs的起源

如何处理更大的图像（如常用的224x224x3），其中3代表RGB 3个通道

224x224x3

28 × 28

784维

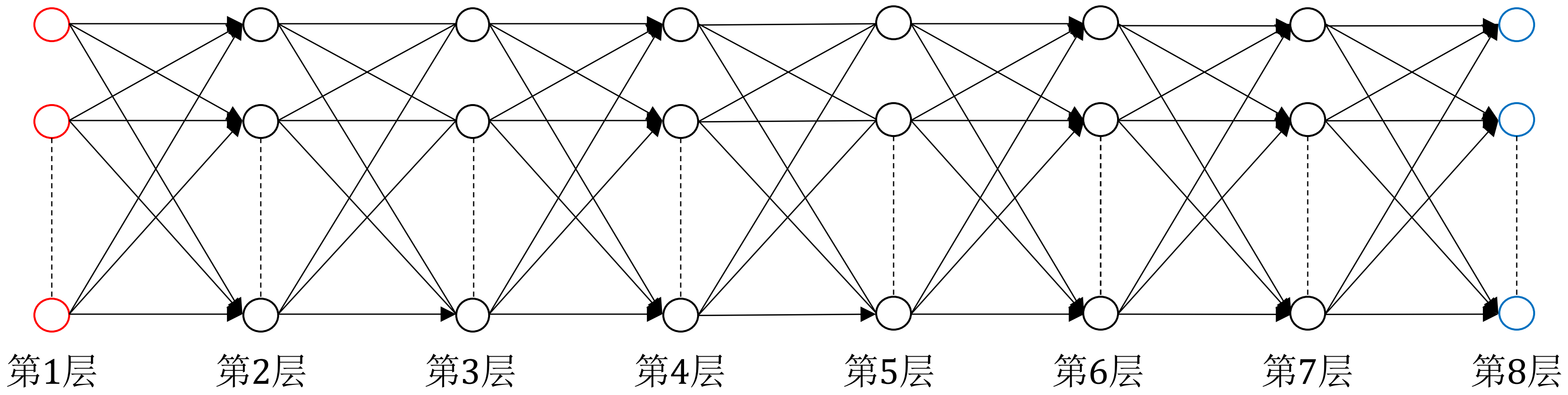


CNNs的起源

224x224x3

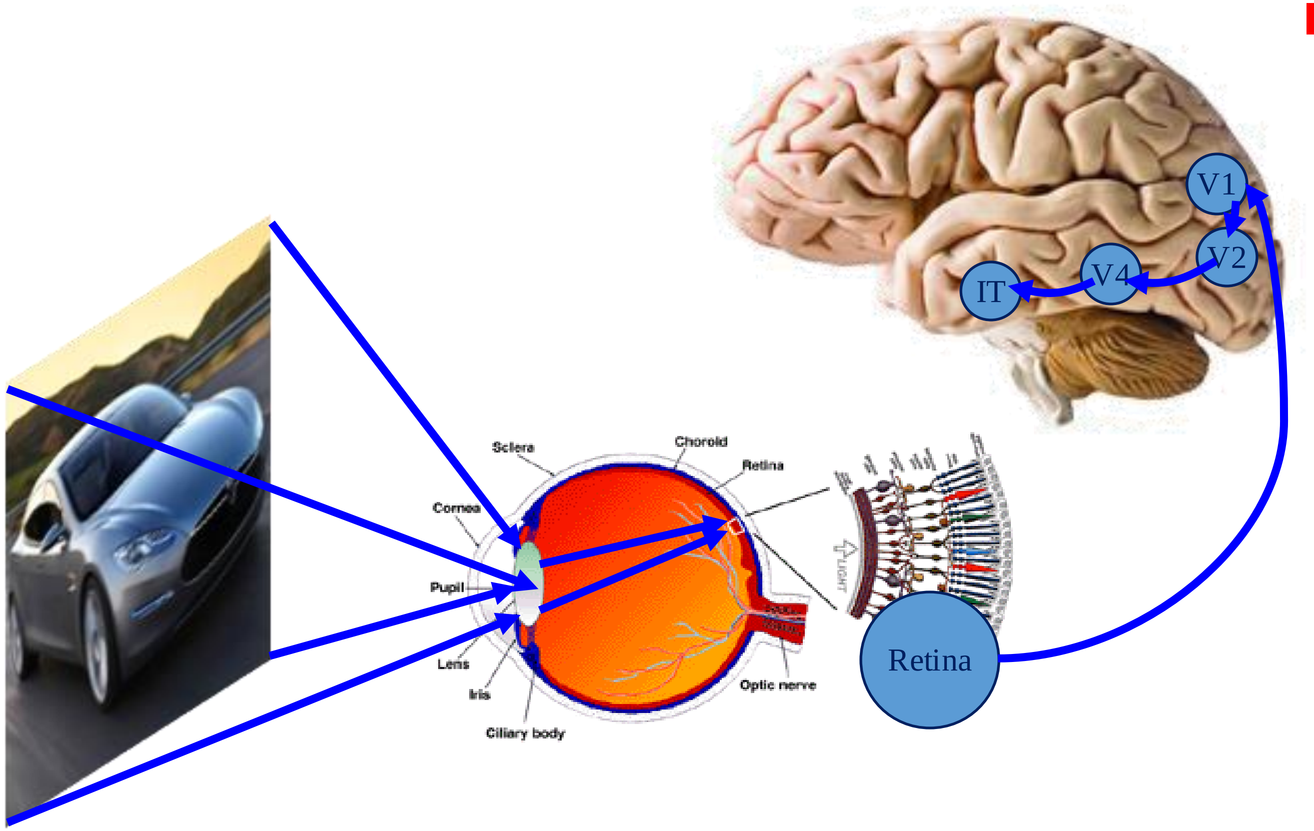
150,528维

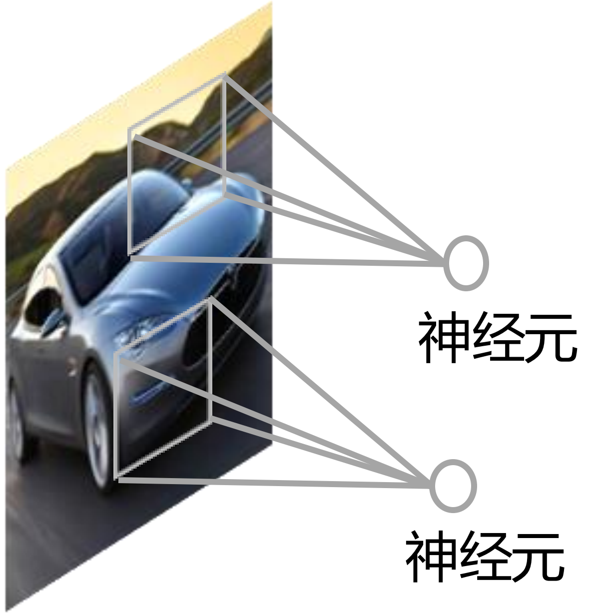
- 假设第1个隐藏层的维度为100，则第一层即包含15,052,800个可学习参数，显然该参数量过大，将导致严重的过拟合
- 人脑是如何处理图像的?



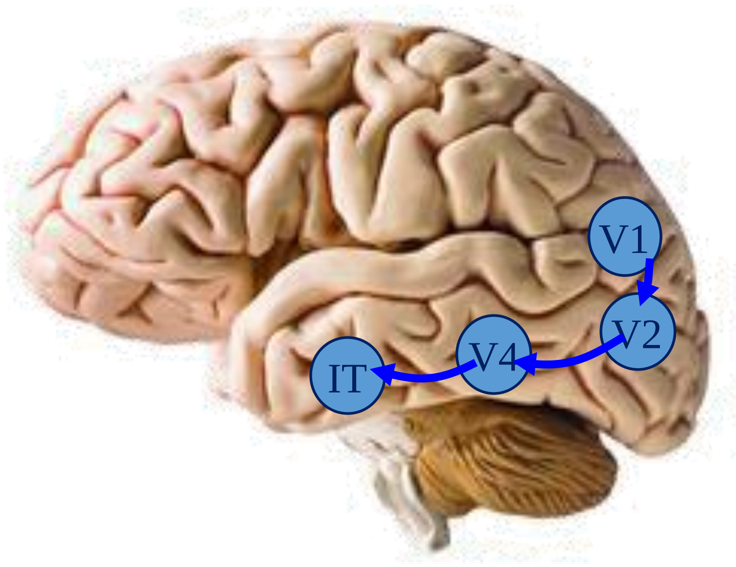
CNNs的起源

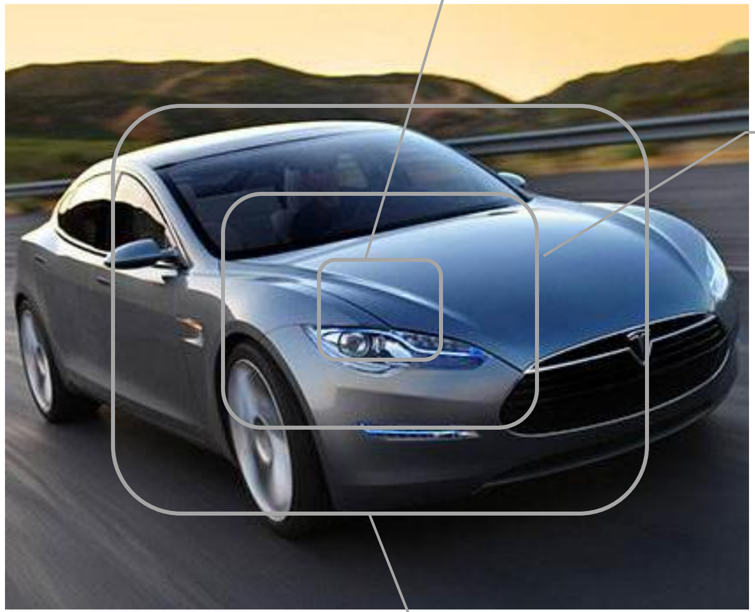
- 人脑中的神经元只会处理图像中的部分区域
- 该区域被称为感受野 (receptive field)





CNNs的起源





V1区域神经元的感受野

V2区域神经元的感受野

V4区域神经元的感受野

问题:
如何仿照人脑结构，建立关于感受野的模型？

大纲

- CNNs的起源
- CNNs的组件
- CNNs的结构
- CNNs的学习
- CNNs的应用

CNNs的组件

224x224x3

红色通道, 224x224x1

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

绿色通道, 224x224x1

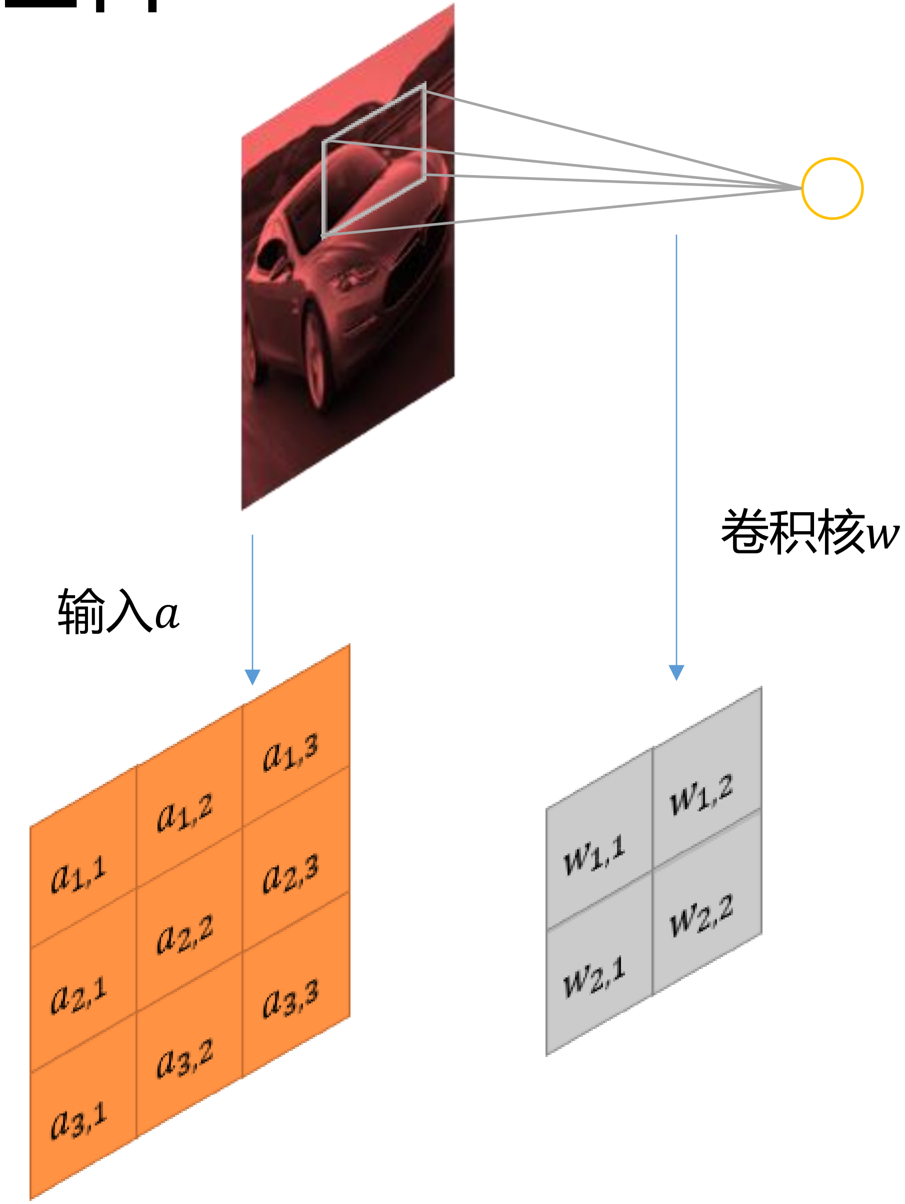
$$\begin{bmatrix} \ddots & 84 & 41 & 112 & 117 & \iddots \\ 80 & 195 & 52 & 172 & 139 & 251 \\ 128 & 131 & 107 & 191 & 147 & 204 \\ 162 & 130 & 218 & 148 & 191 & 196 \\ 171 & 82 & 136 & 122 & 250 & 141 \\ \iddots & 129 & 186 & 154 & 161 & \ddots \end{bmatrix}_{224 \times 224}$$

蓝色通道, 224x224x1

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

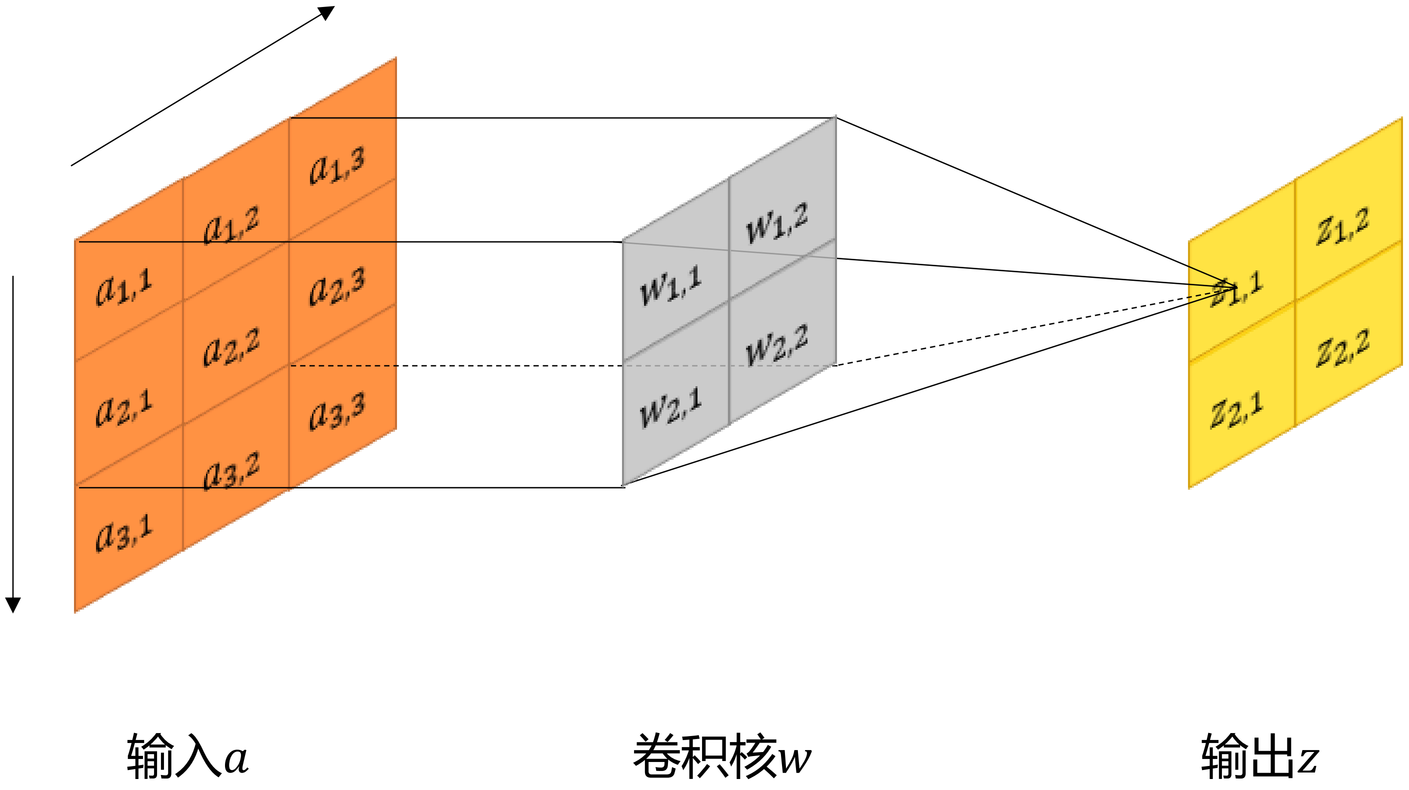
注：此处不同颜色仅为了直观理解，真实情况中各通道均为灰度图像

CNNs的组件



- 卷积核是用于提取图像中抽象特征的可学习参数矩阵
- 卷积核的大小等价于感受野的大小

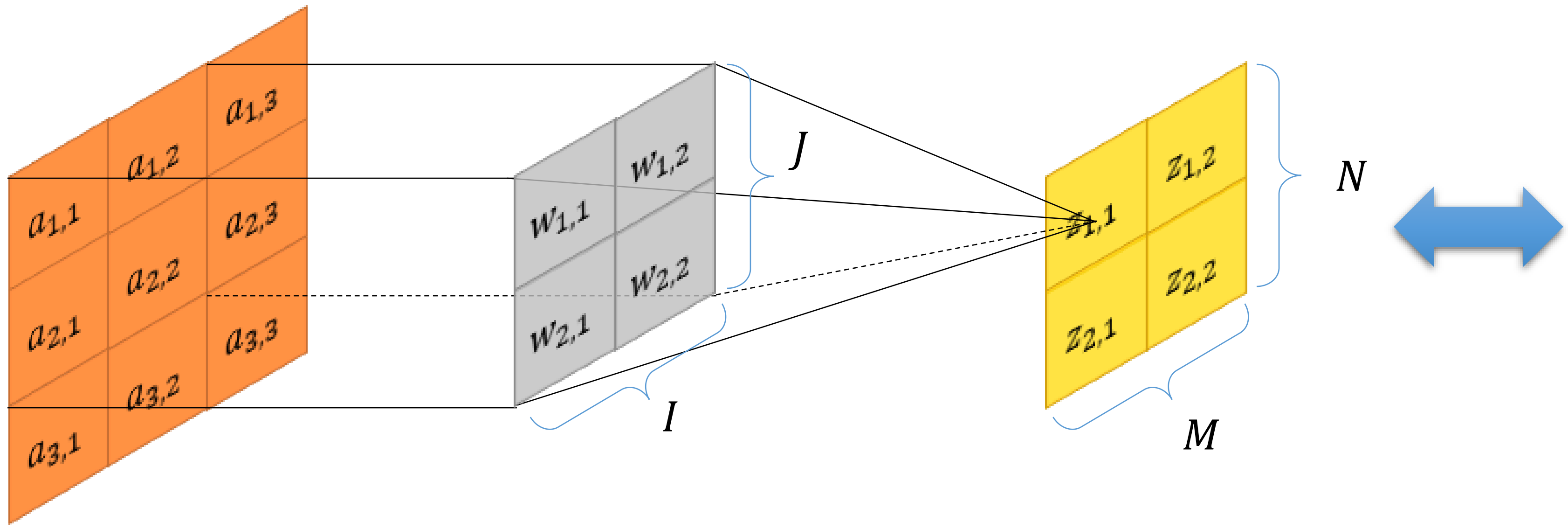
CNNs的组件



卷积

- 卷积核w沿着输入a 的宽和高的方向进行滑动
- 输出z由w和a对位元素相乘并求和得到

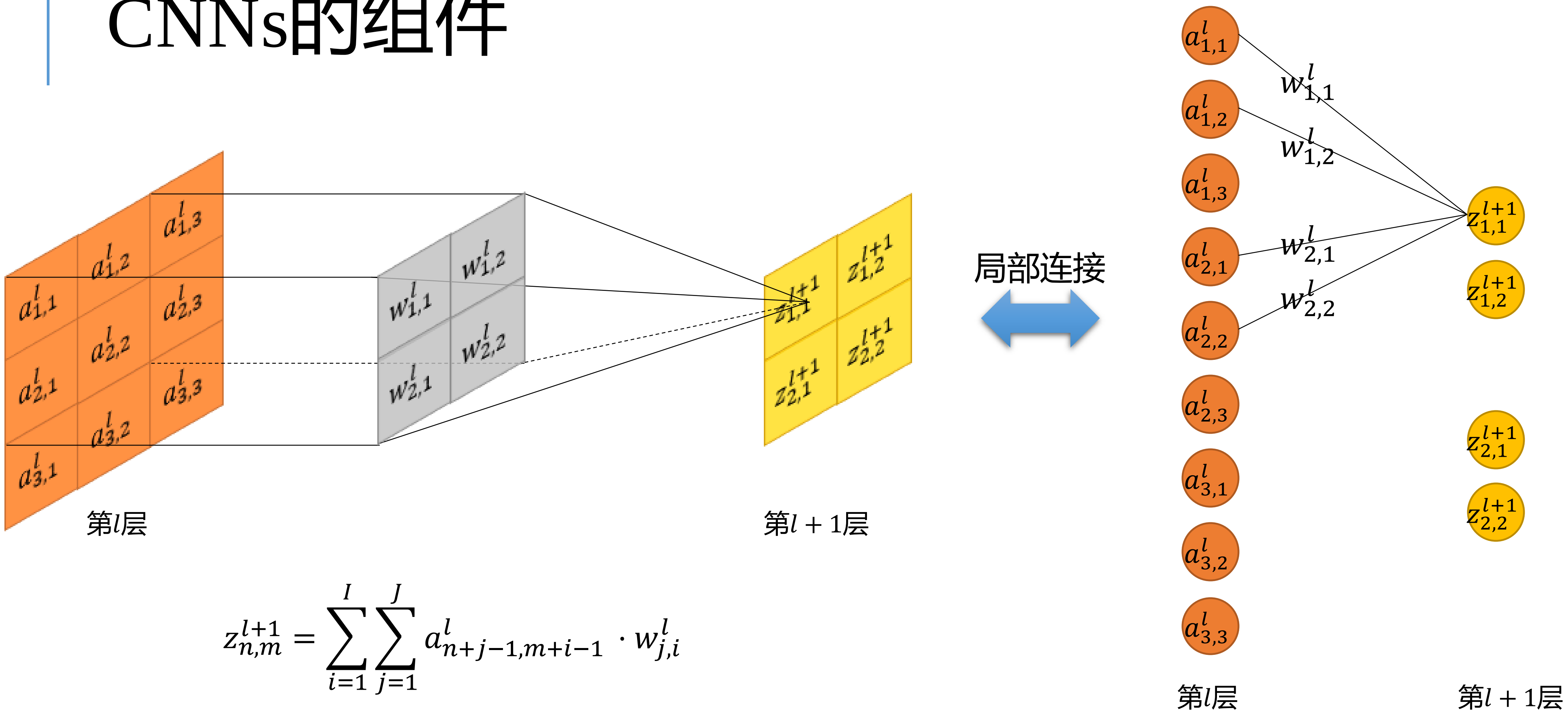
CNNs的组件



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

- 假设卷积核w 的大小是$I \times J$
- 假设输出 z 的大小是 $M \times N$

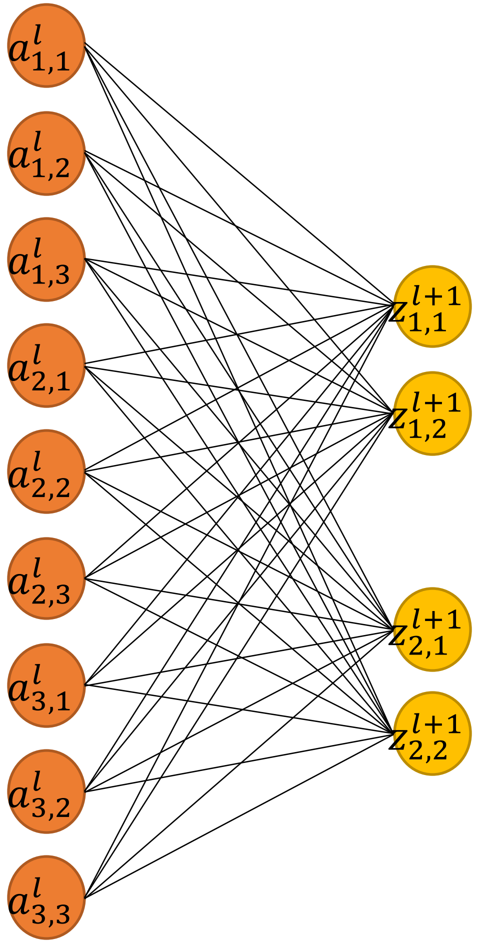
CNNs的组件



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

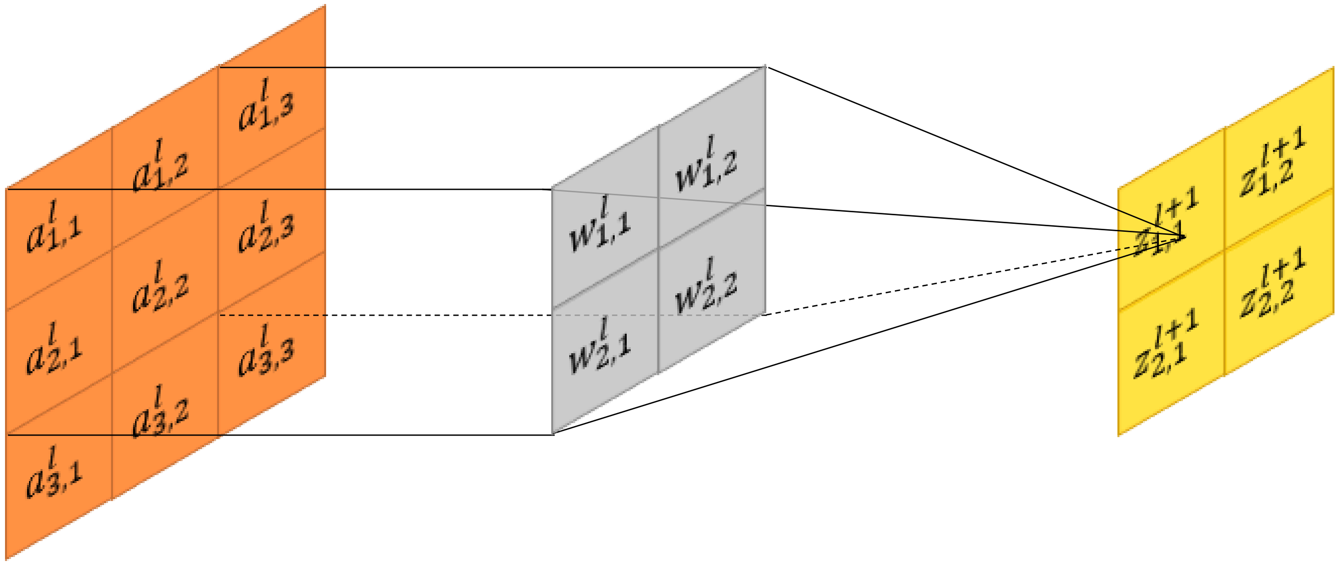
- 对位元素相乘并求和等价于全连接中的局部连接
- 每次运算共享卷积核

CNNs的组件



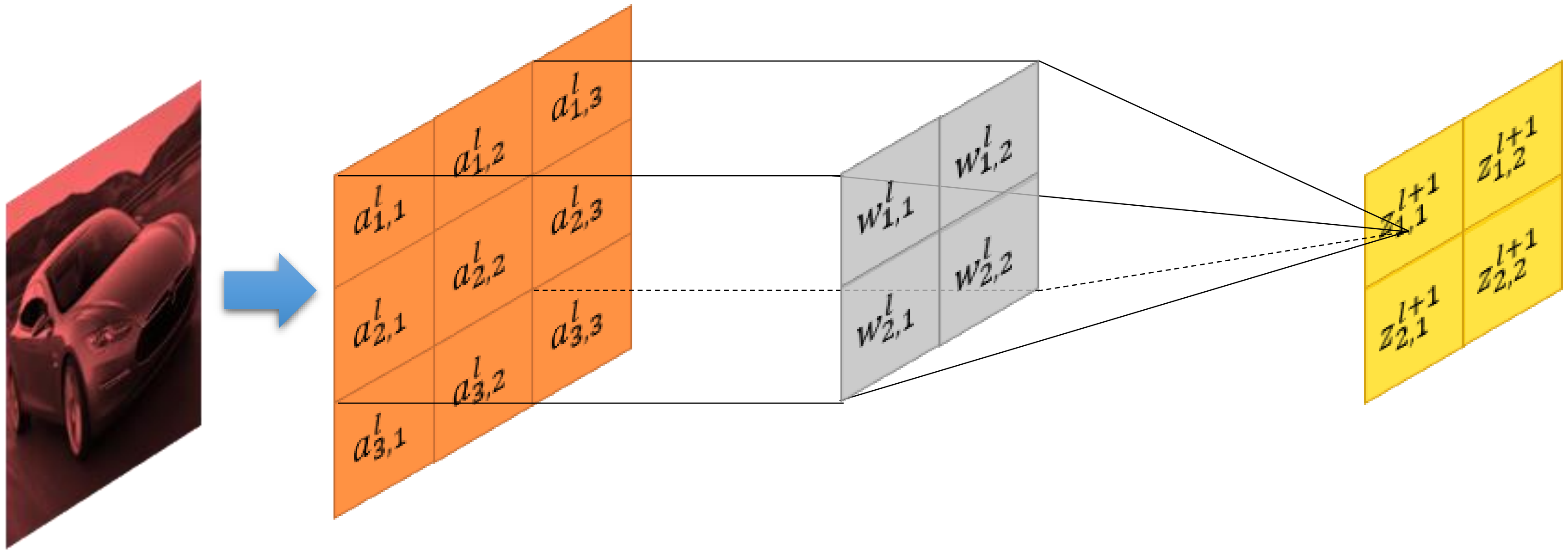
$a^l \in R^9$ $\quad$ $z^{l+1} \in R^4$

- 全连接中的参数为$9 \times 4 = 36$



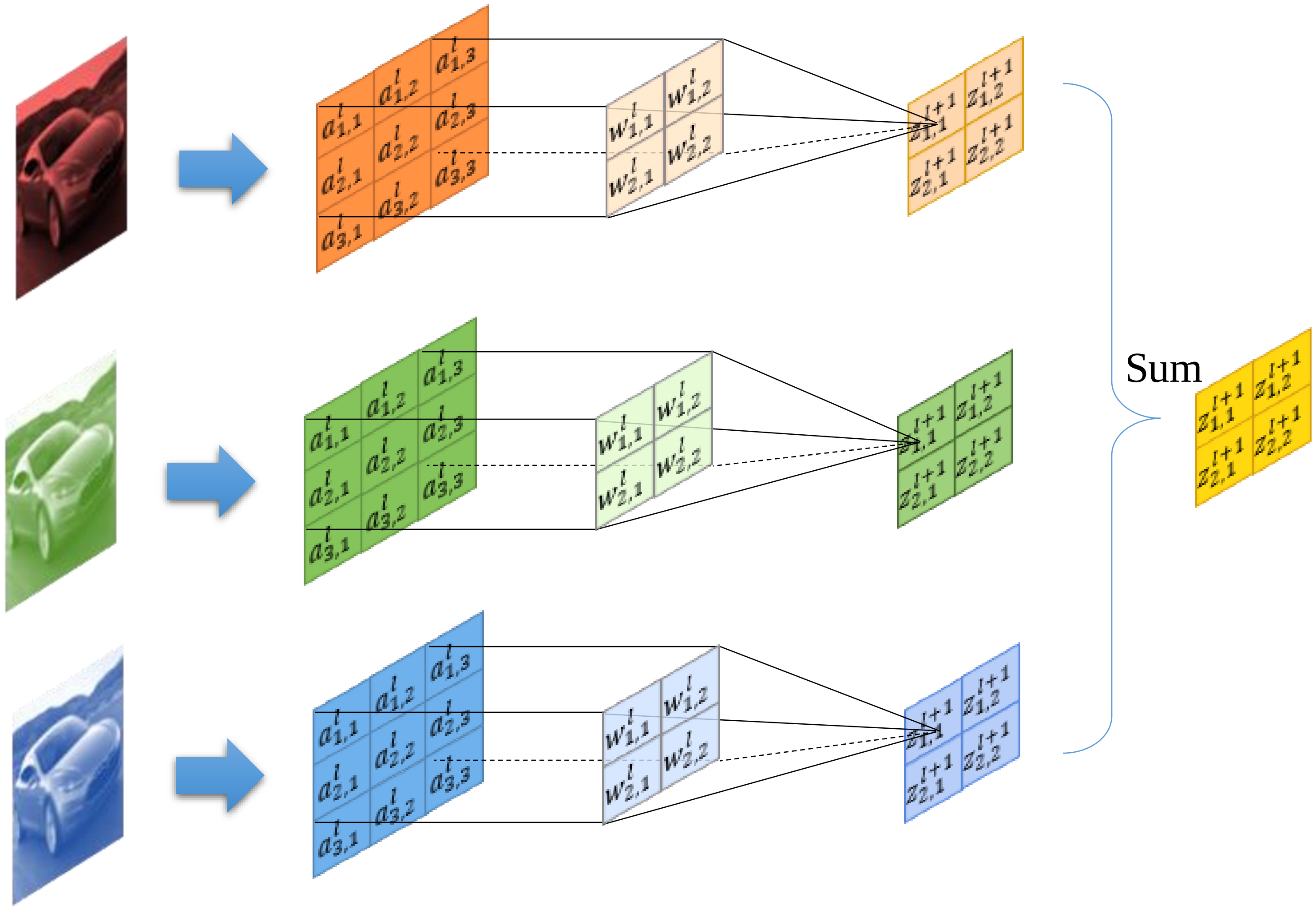
- 卷积操作中的参数量等于卷积核的大小，即2×2，远小于全连接中的参数量

CNNs的组件



如何处理包含多通道的输入?

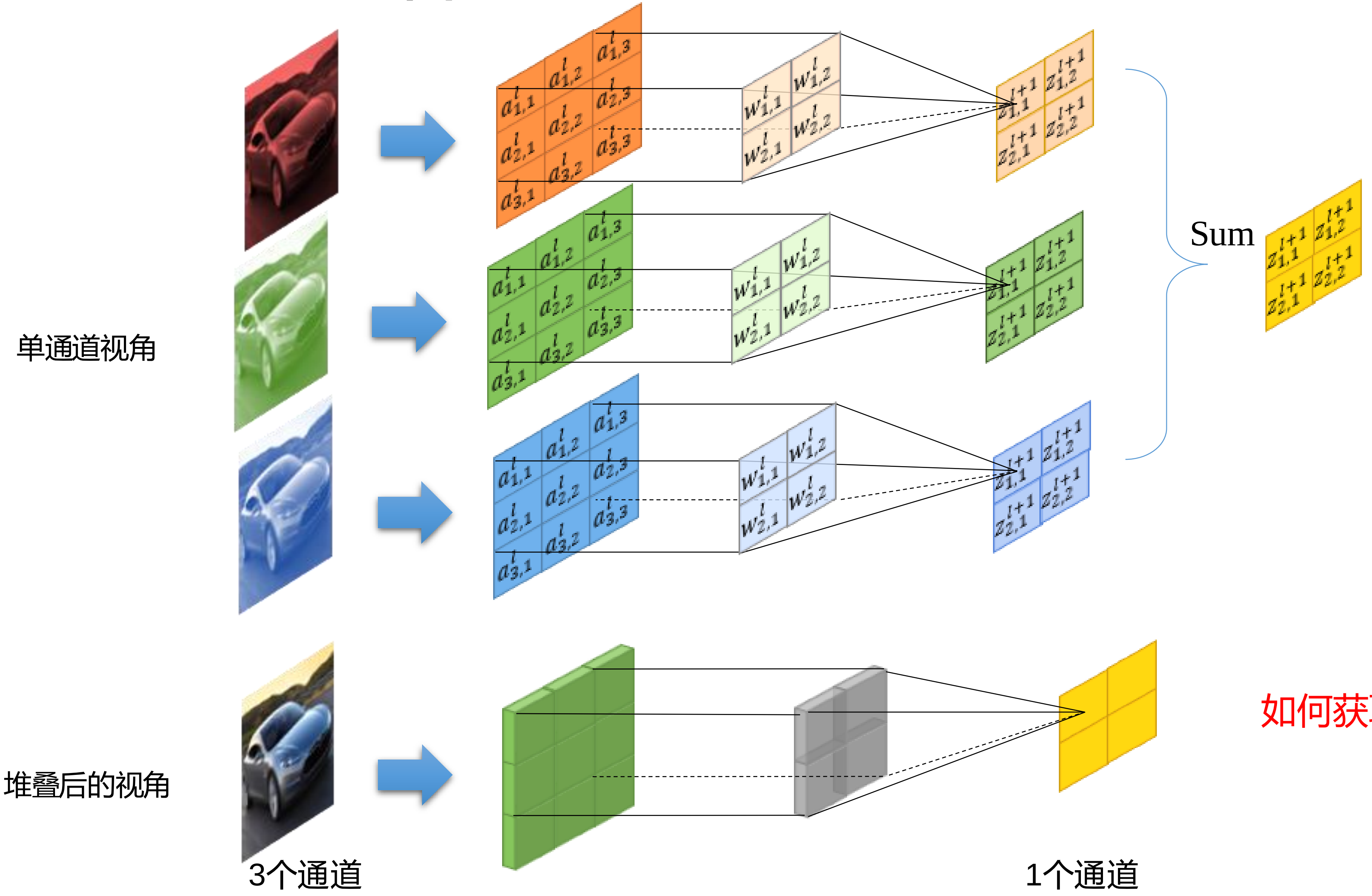
CNNs的组件



第1步: 分别对R, G, B通道进行卷积运算

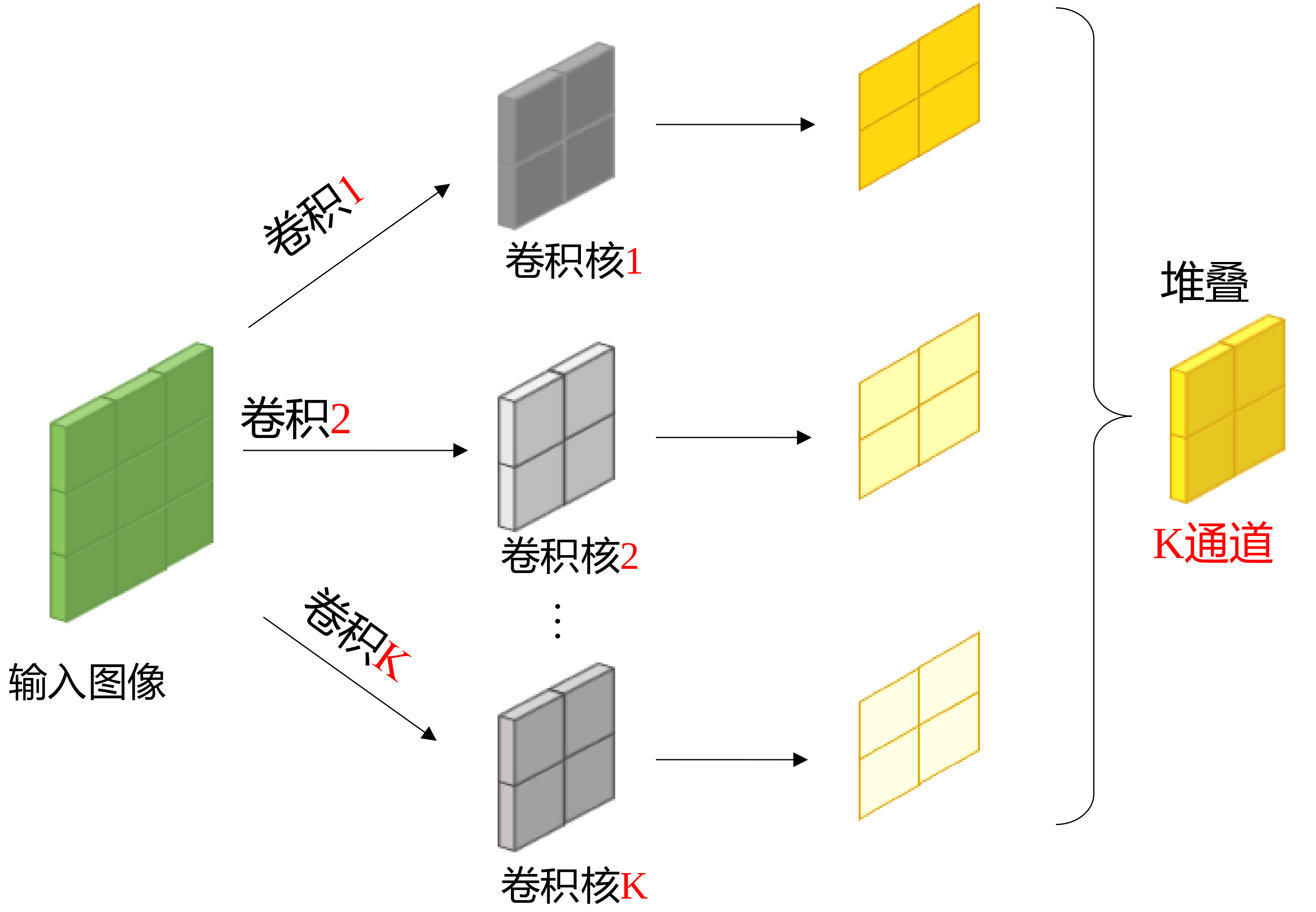
第2步: 对每个卷积运算的输出求和

CNNs的组件



如何获取具有多个通道的输出?

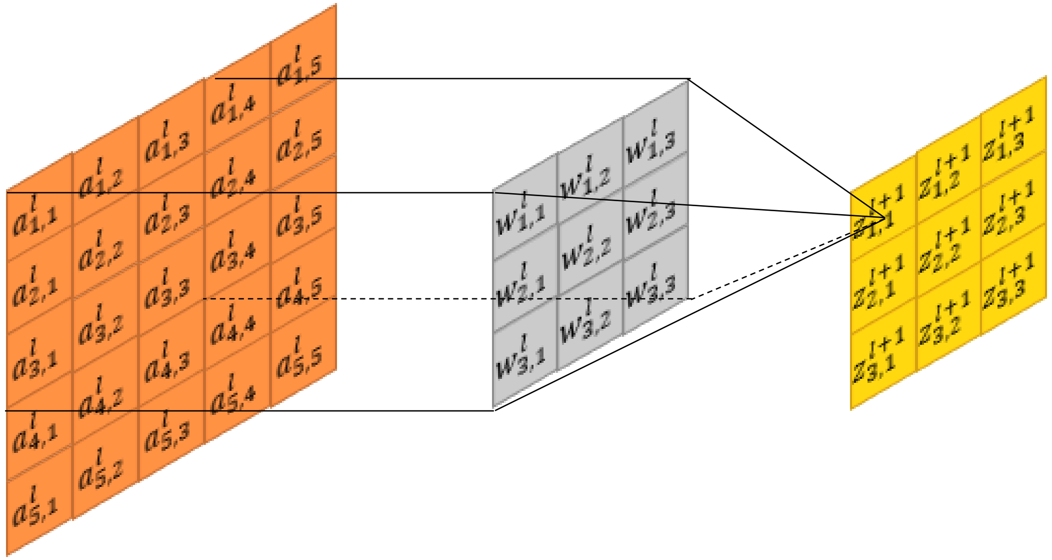
CNNs的组件



第1步: 使用多套卷积核对输入进行卷积

第2步: 堆叠每个卷积的输出，其中输出的通道数等于卷积核的数目

CNNs的组件

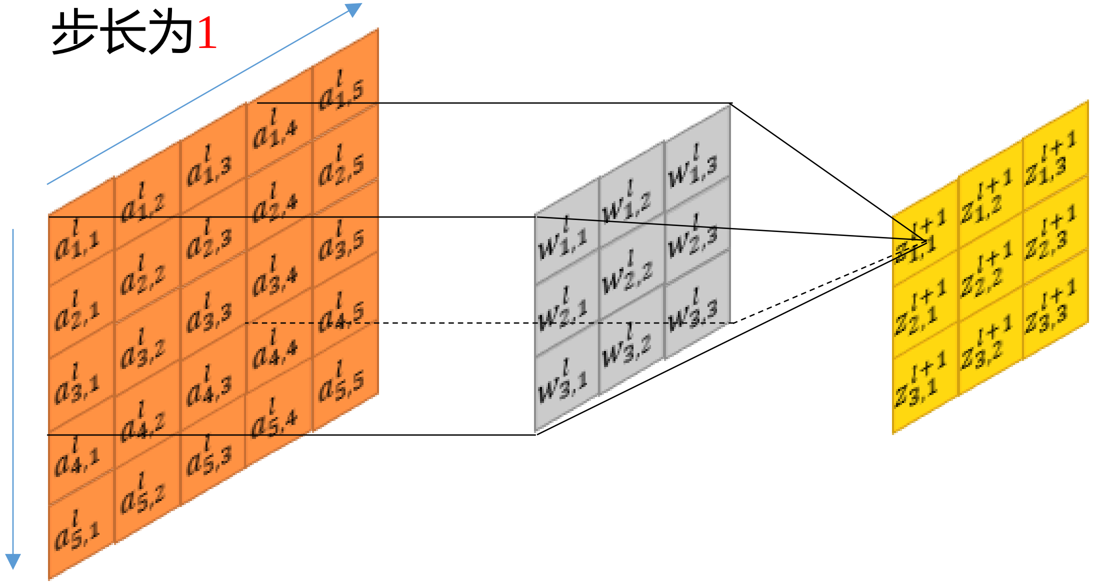


为了表述的简便，后续将以单通道输入为例进行讲解

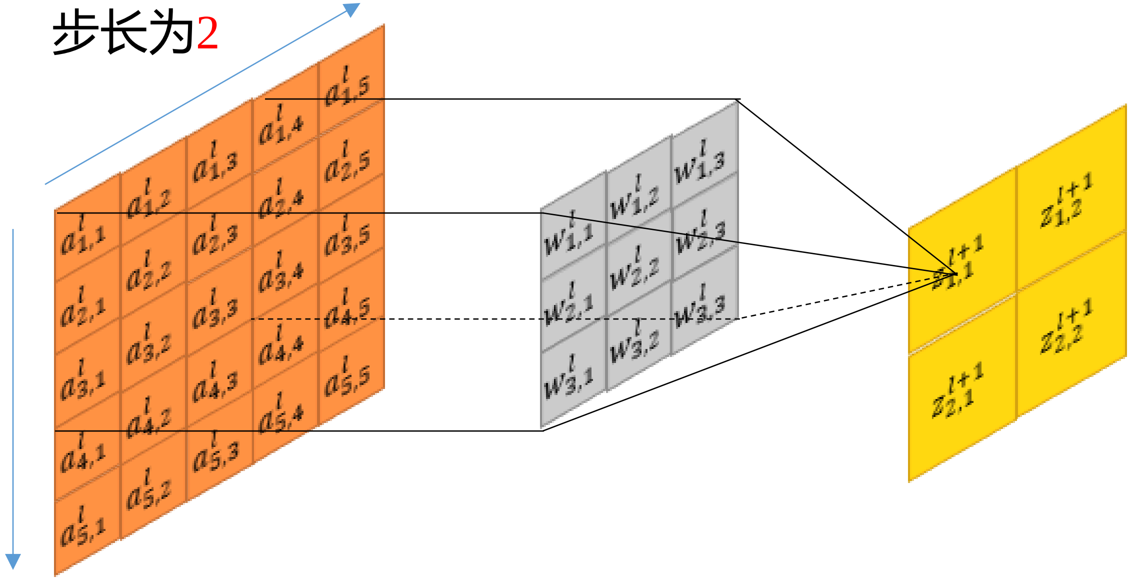
$$z^{l+1}_{n,m} = \sum_{i=1}^{I} \sum_{j=1}^{J} a^{l}_{n+j-1,m+i-1} \cdot w^{l}_{j,i}$$

CNNs的组件

步长为1



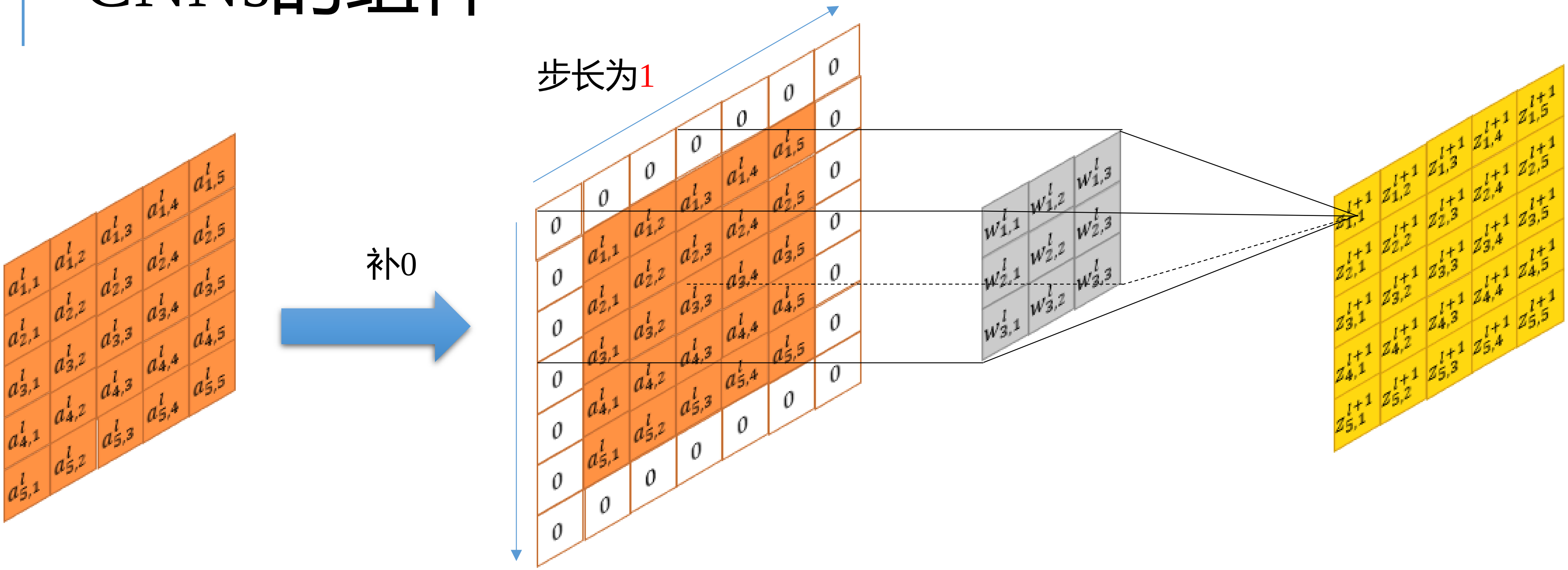
步长为2



- 步长控制卷积核的移动幅度
- 注意到即使步长为1，输出的大小也将小于输入的大小

如何保持输出的大小与输入的大小一致，从而构建深度CNNs?

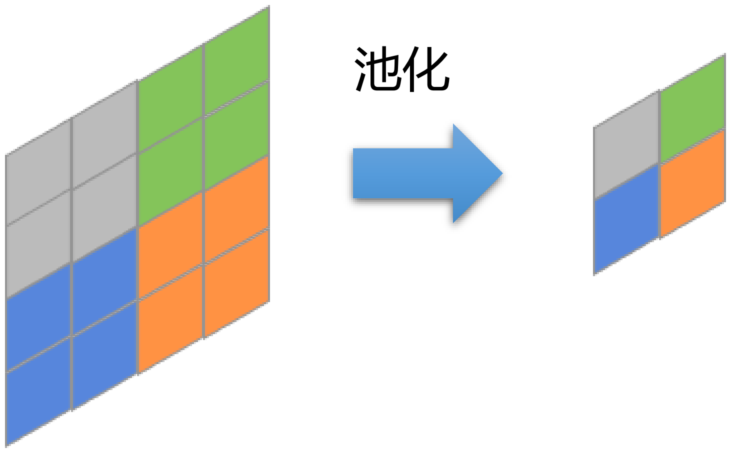
CNNs的组件

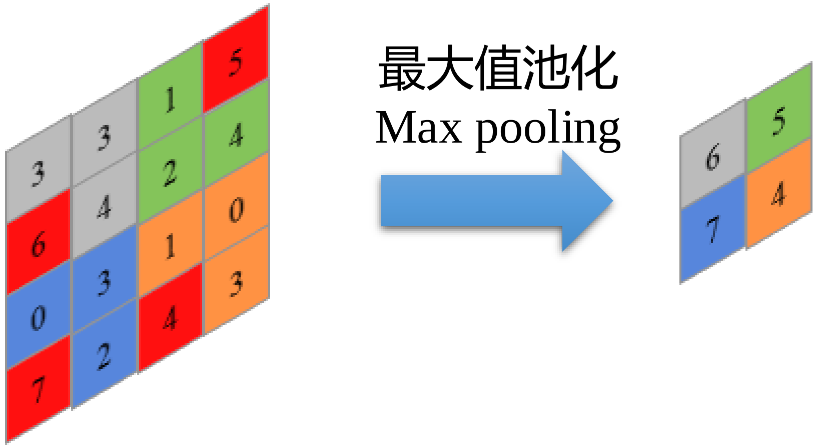


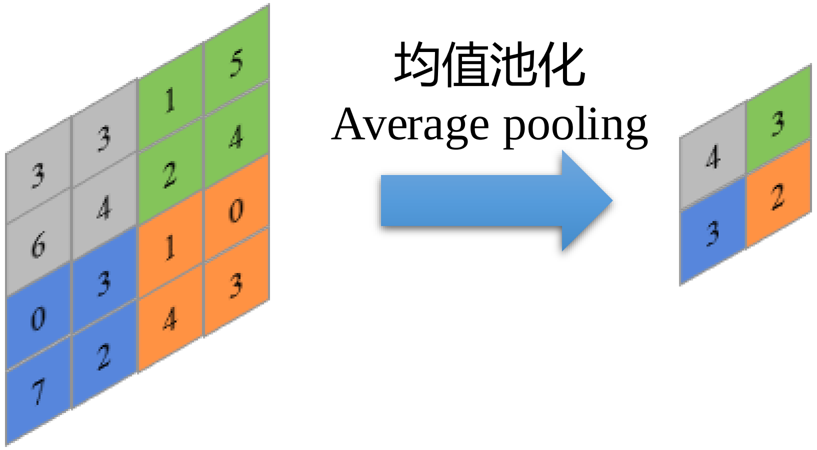
- 对输入输入的边缘补0，从而保持输出大小与输入大小一致
- 当步长大于1时，将实现对输入下采样的效果，然而这种下采样将引入可学习参数w

是否存在更高效的下采样方式?

CNNs的组件



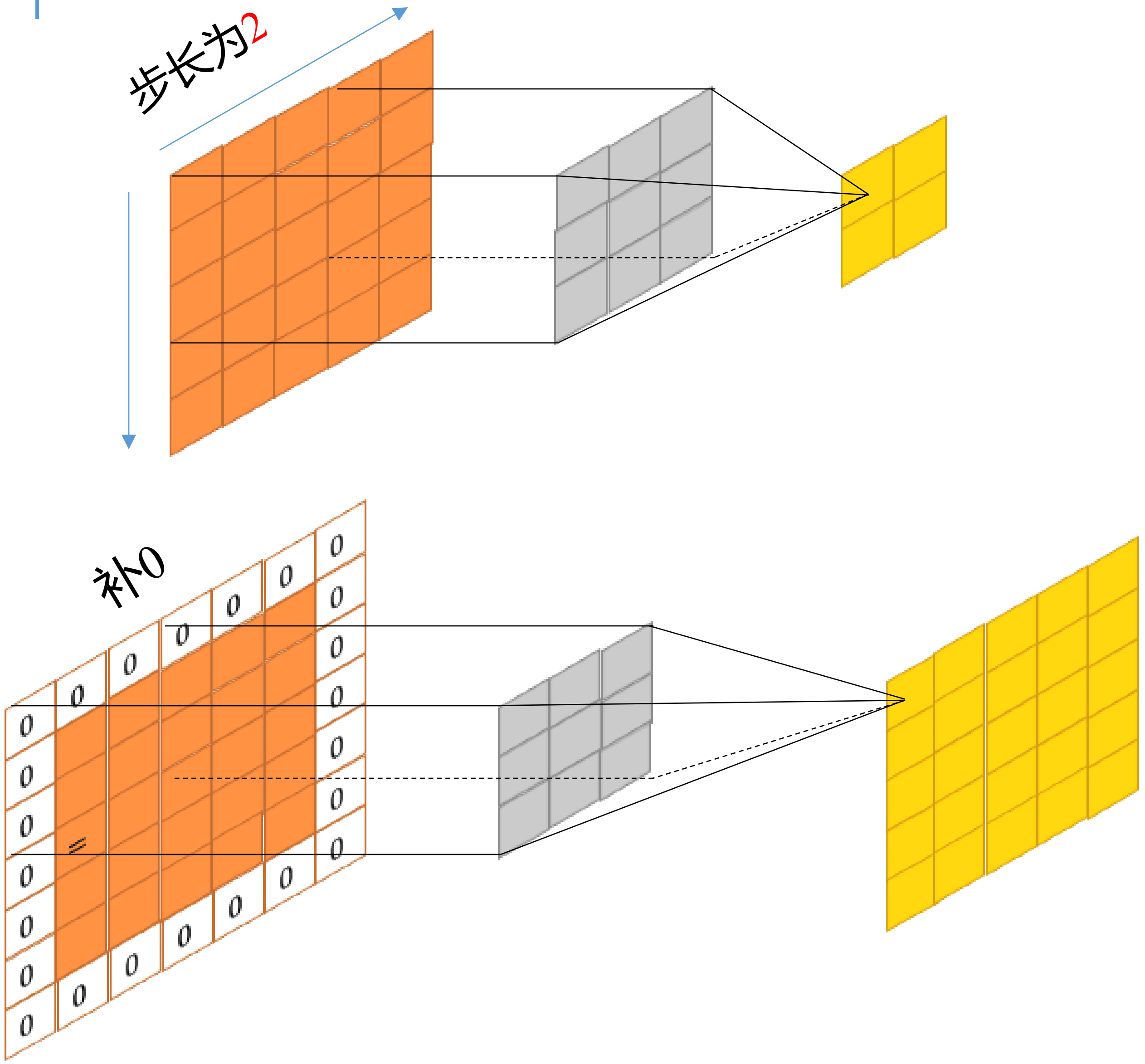


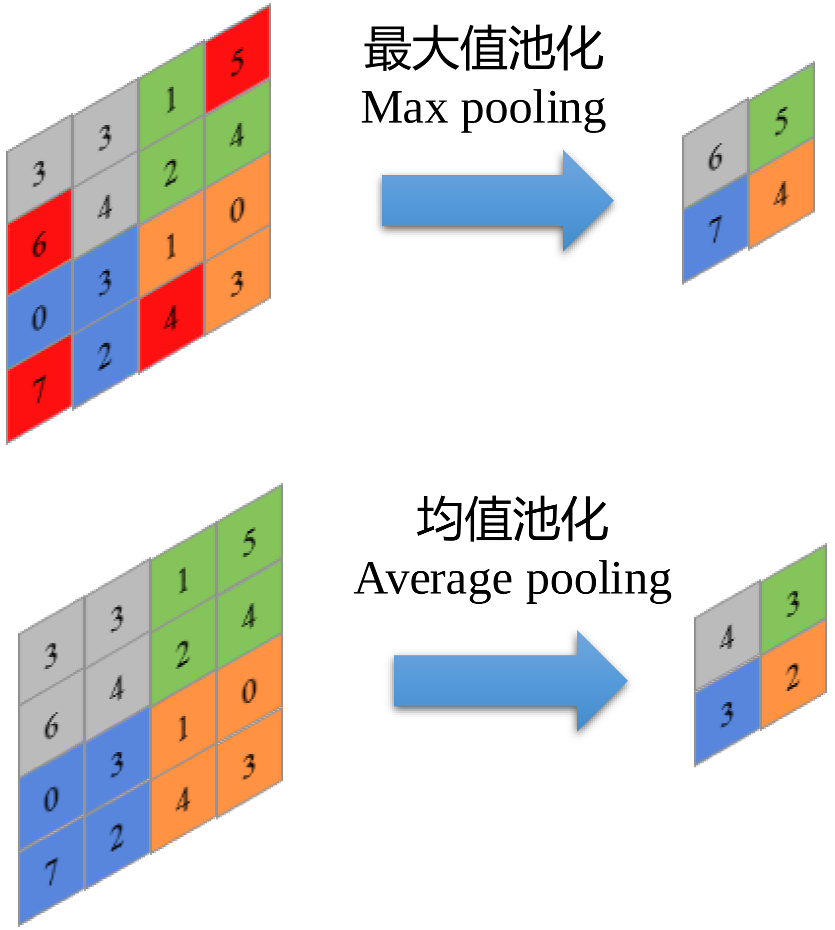


- 池化的目的在于不引入可学习参数w前提下，高效地对输入下采样，降低输入的维度

- 最大值池化输出感受野中的最大值
- 均值池化输出感受野中的均值

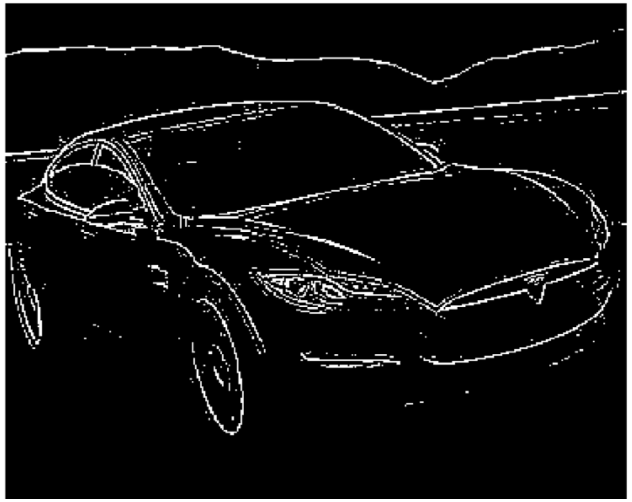
CNNs的组件





卷积核的作用

原始图像



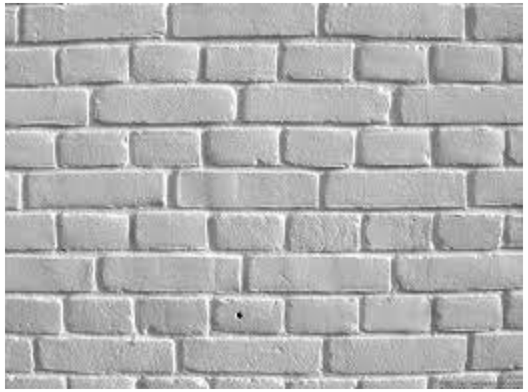
$w1$提取图像中的边缘

$w2$实现了图像的虚化效果

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

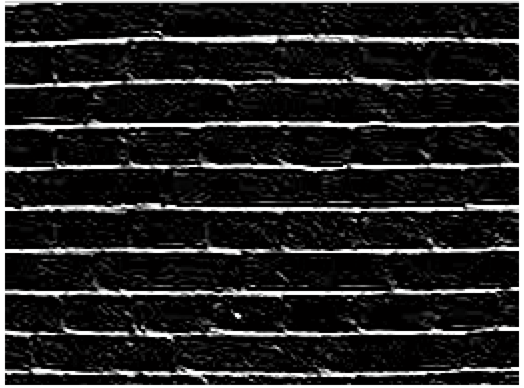
$$w2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

卷积核的作用



原始图像

$w1$提取垂直边缘

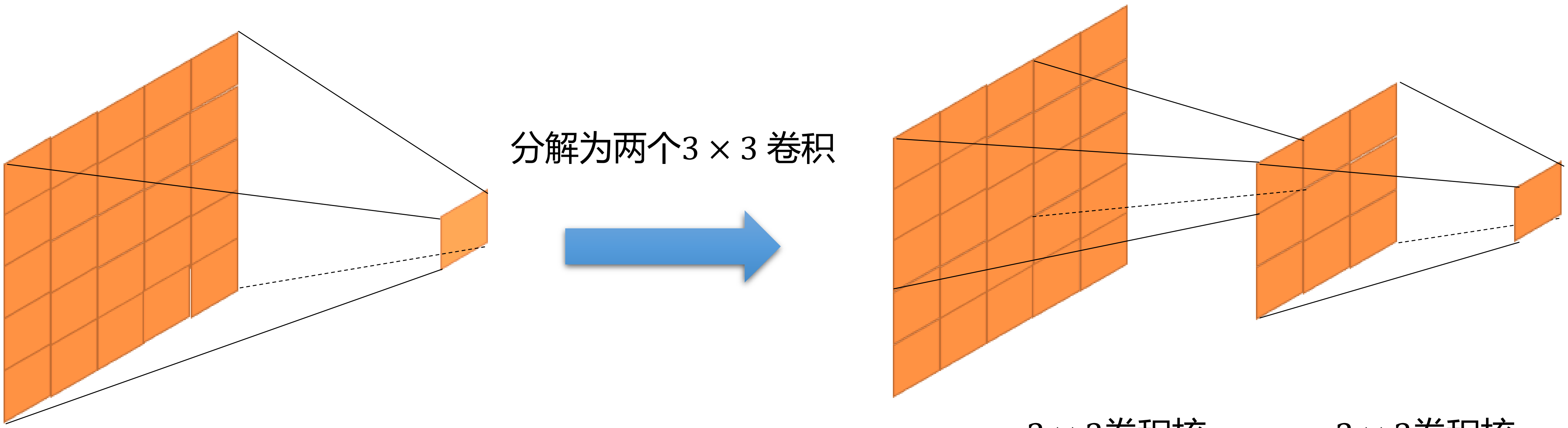


$w2$提取水平边缘

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

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

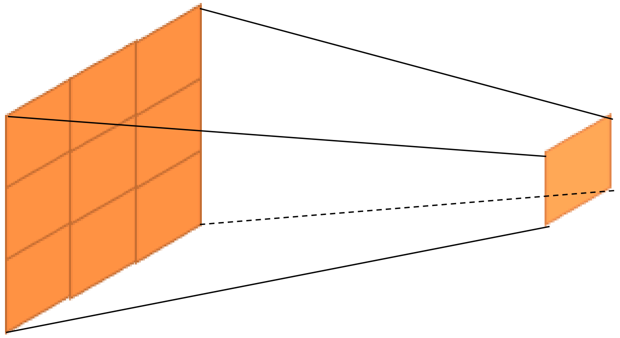
卷积核的分解



5 × 5 卷积核
25 个参数

3 × 3卷积核　　3 × 3卷积核
共18个参数

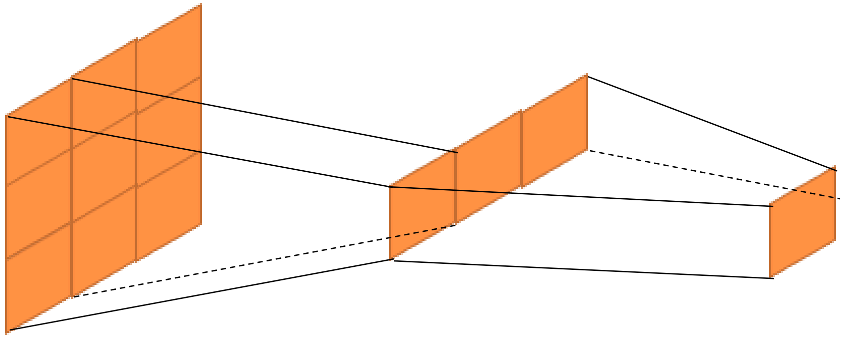
卷积核的分解



3×3卷积核

9 个参数

分解为3×1和1×3卷积



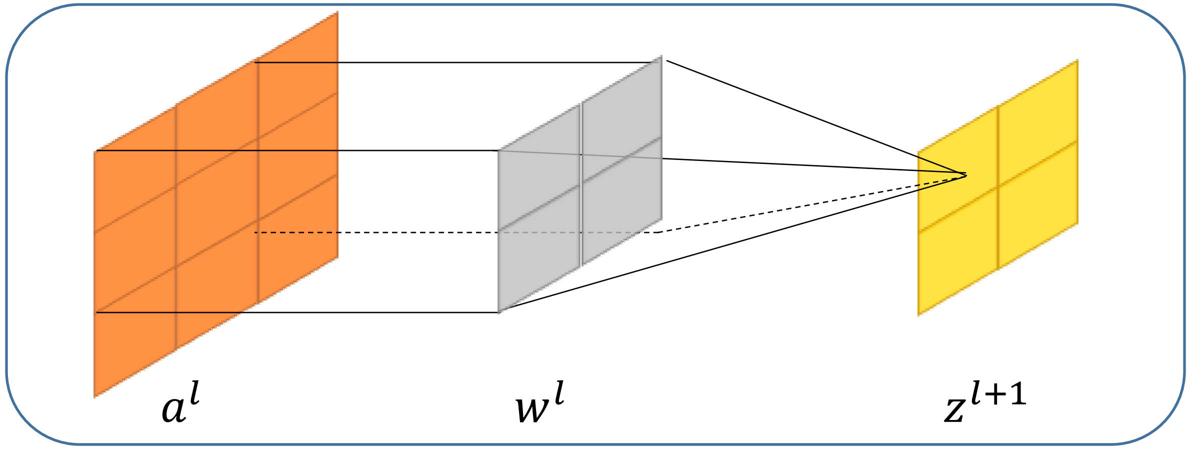
3×1卷积核　　1×3 卷积核

共6 个参数

大纲

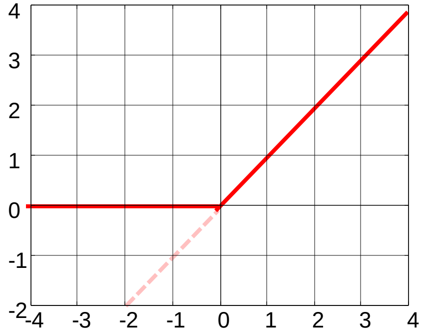
CNNs的结构

1. 卷积层

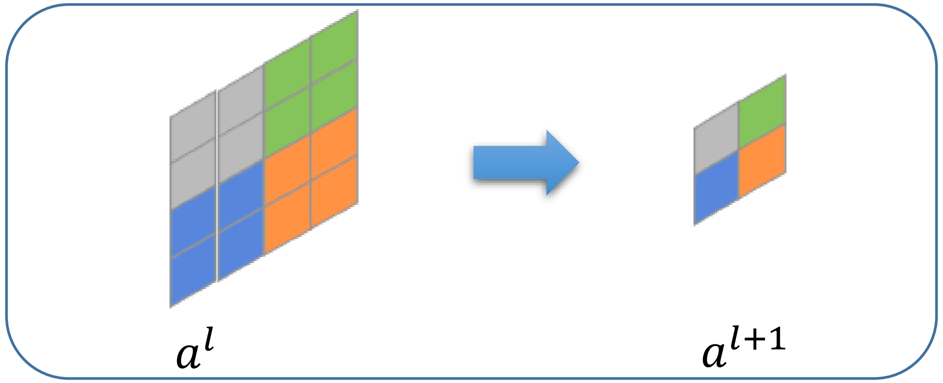


$+$ ReLU

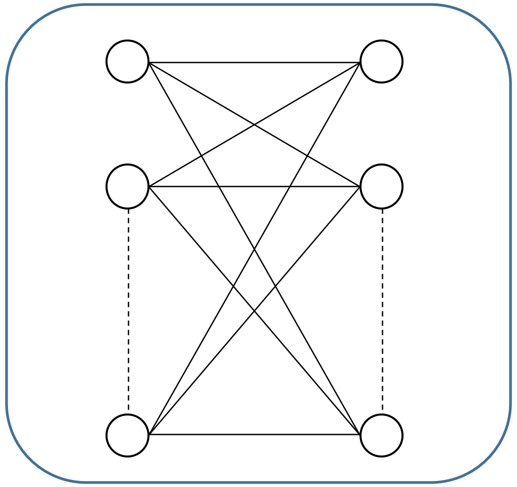
$$f(z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases}$$



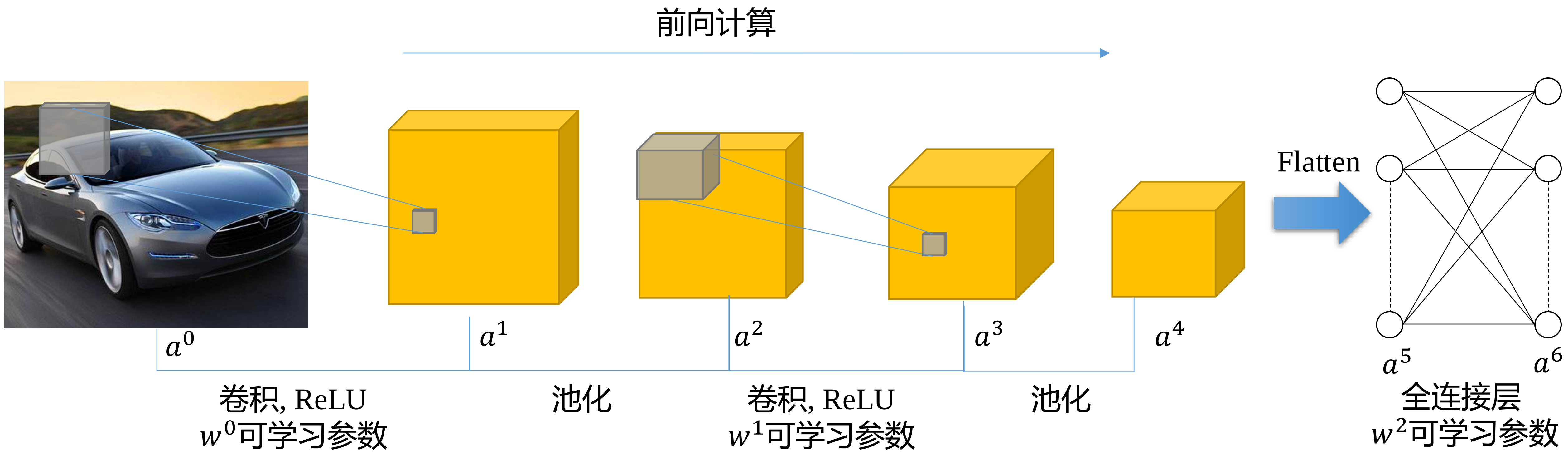
2. 池化层



3. 全连接层



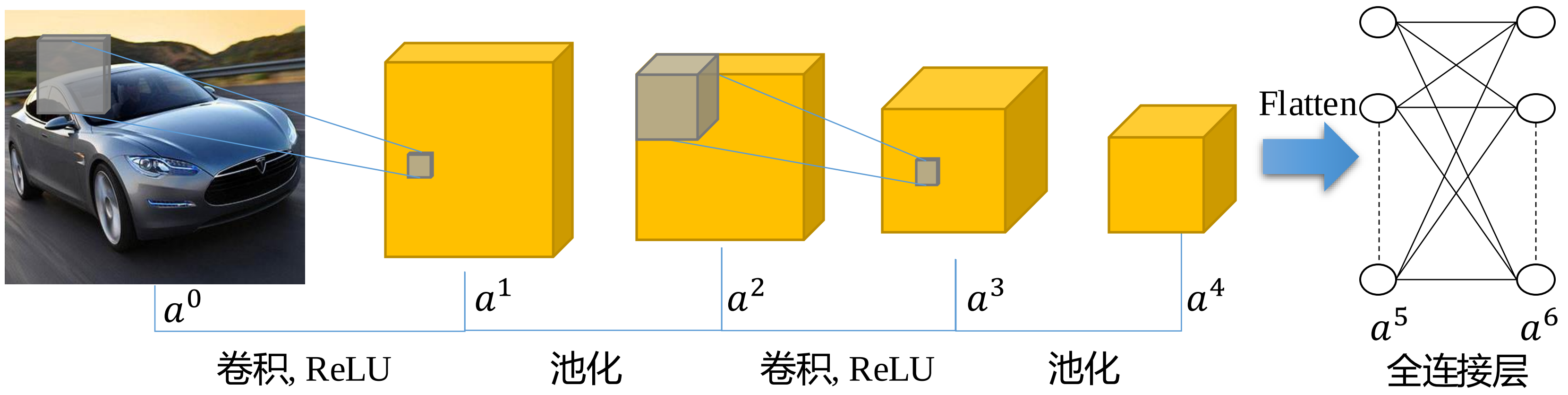
CNNs的结构



如何更新卷积和全连接层中的可学习参数

大纲

CNNs的学习

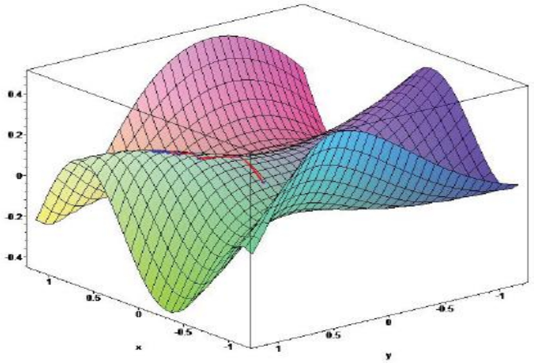


网络输出

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

标签

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



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

随机梯度下降: $w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$

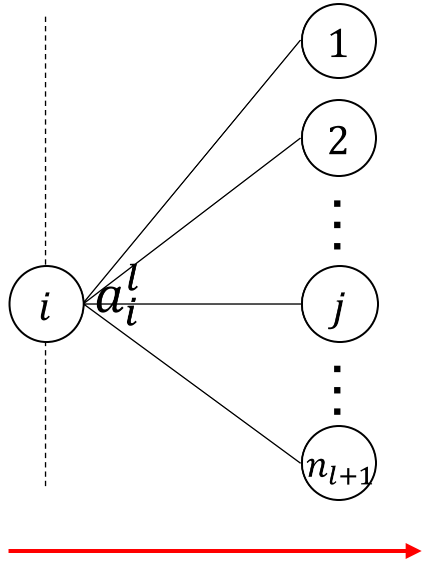
使用反向传播算法计算卷积和全连接中的 $\frac{\partial J}{\partial w_{ji}^l}$

注：实际应用中一般使用交叉熵作为代价函数，且对最后一层的输出作用softmax函数

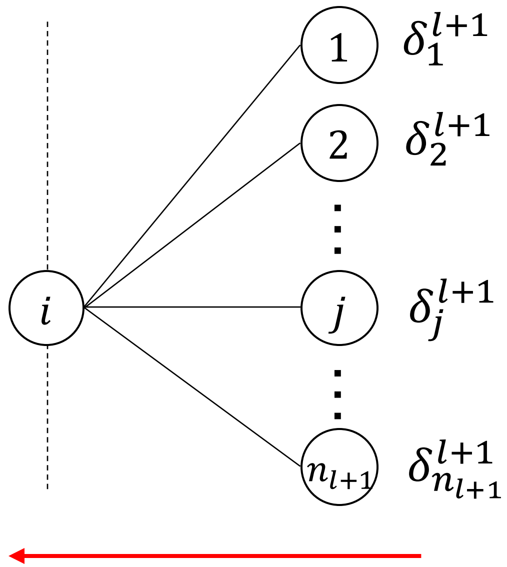
CNNs的学习

■ 全连接层的前向计算与反向传播

$$a_j^{l+1} = f\left(\sum_{i=1}^{n_l} w_{ji}^l a_i^l\right) \qquad \delta_i^l = \dot{f}(z_i^l)\left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} w_{ji}^l\right)$$



前向计算

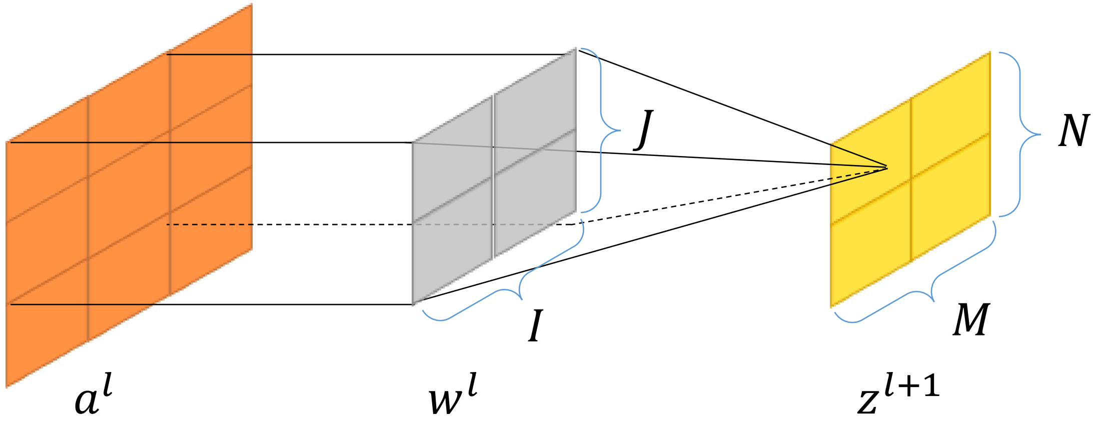


反向传播

- δ_j^{l+1}与$\frac{\partial J}{\partial w_{ji}^l}$的关系：

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

CNNs的学习

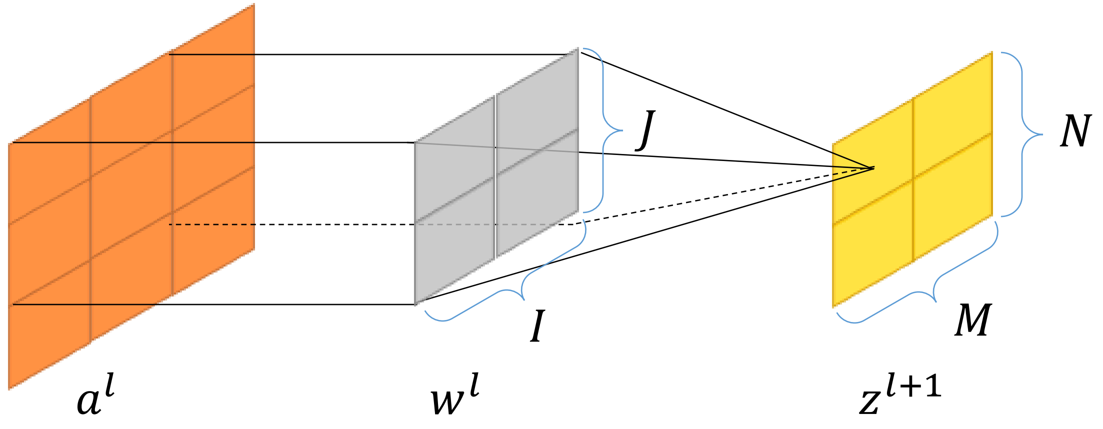


$$z_{n,m}^{l+1} = \sum_{i=1}^{I} \sum_{j=1}^{J} a_{n+j-1,m+i-1} \cdot w_{j,i}^{l}$$

- 如何计算卷积层中的 $\frac{\partial J}{\partial w_{j,i}^{l}}$?

与全连接层计算方式一致!

CNNs的学习



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

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

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

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

仍是卷积!

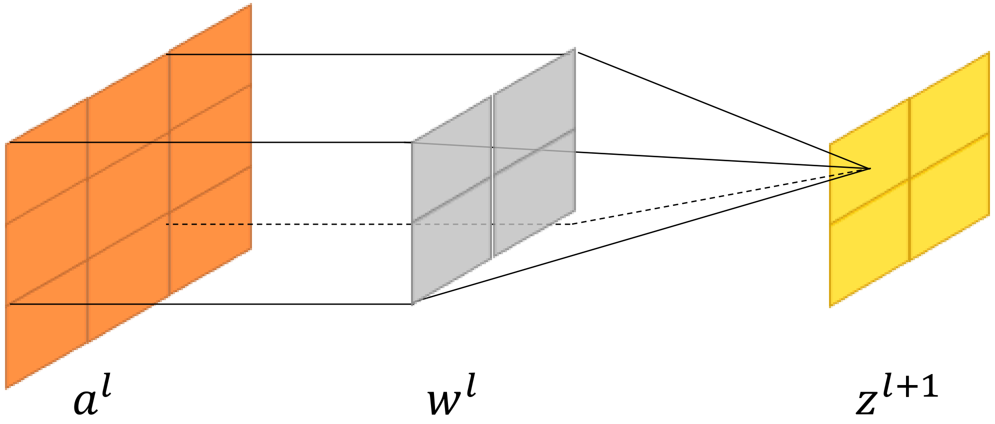
CNNs的学习

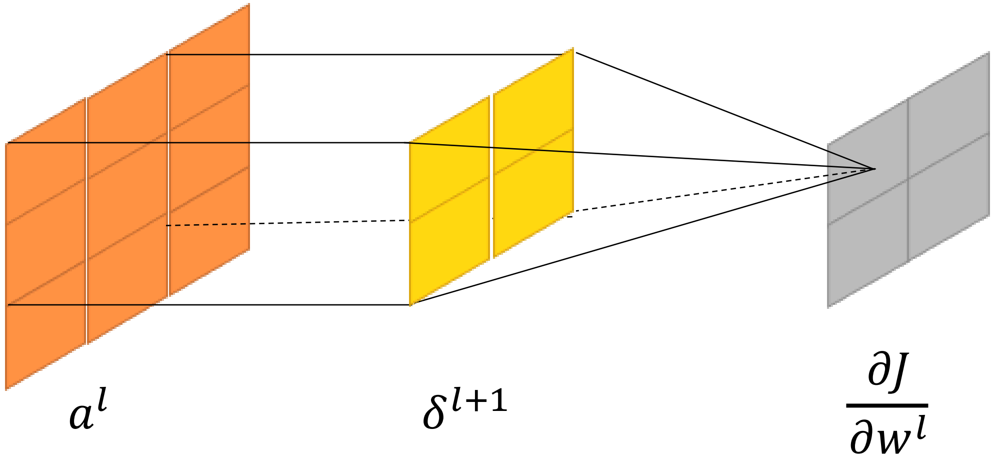
前向计算

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

关系

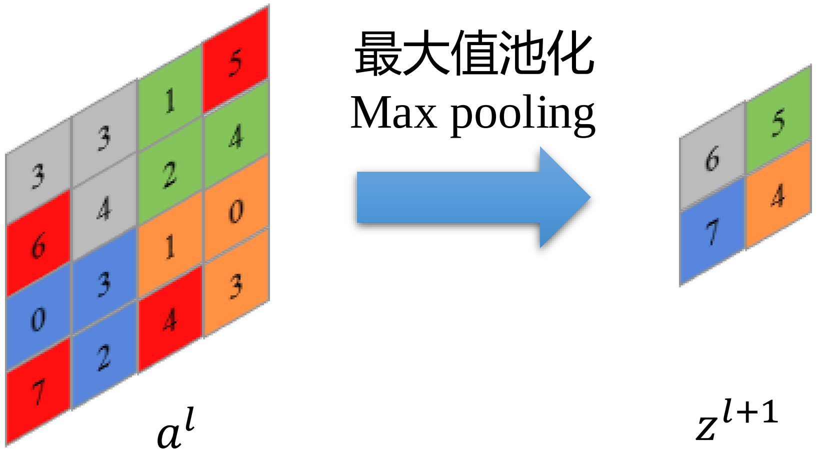
$$\frac{\partial J}{\partial w_{j,i}^{l}} = \sum_{m=1}^{M} \sum_{n=1}^{N} a_{n+j-1,m+i-1}^{l} \cdot \delta_{n,m}^{l+1}$$



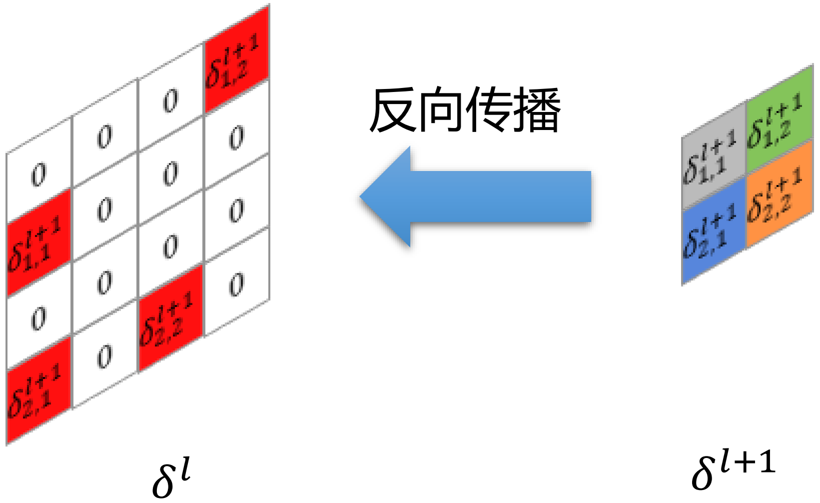


如何反向传播池化层与卷积层中的δ?

CNNs的学习

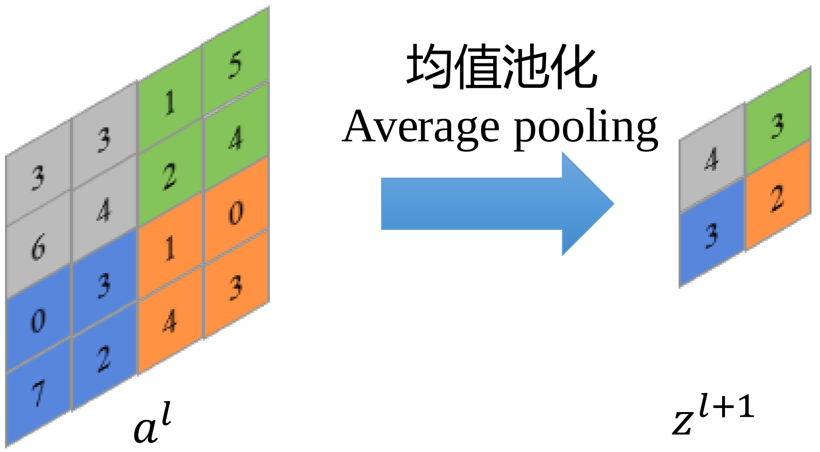


前向计算

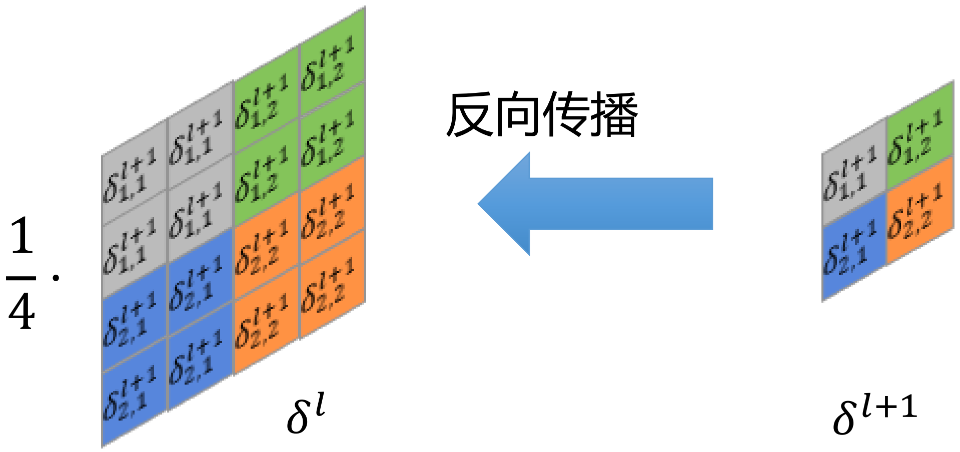


反向传播最大值池化中的δ

CNNs的学习



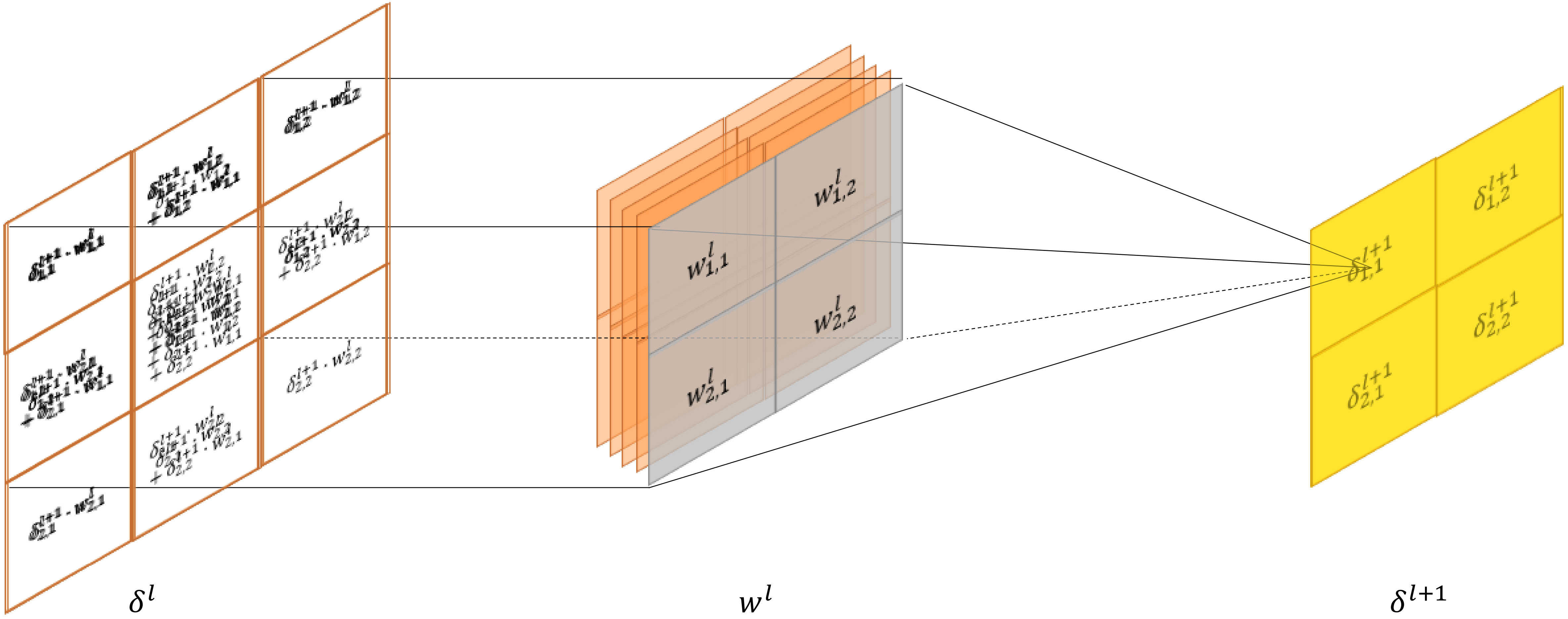
前向计算



反向传播均值池化中的δ

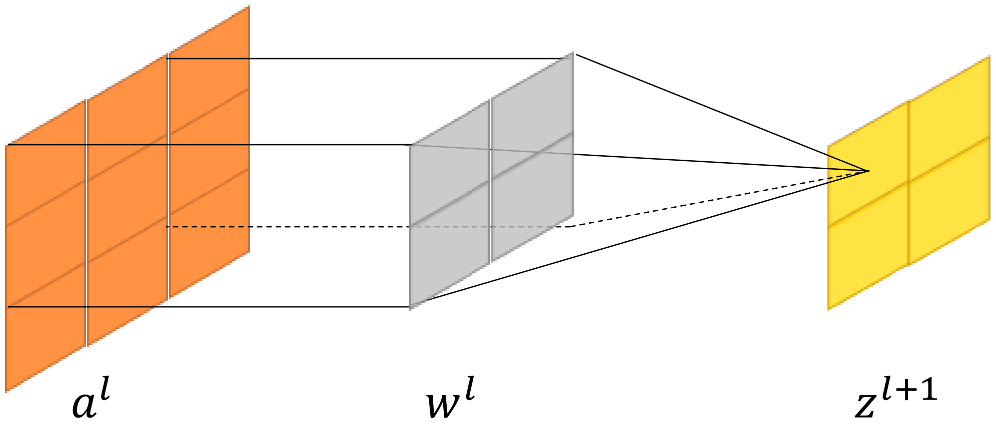
如何反向传播卷积层中的δ?

CNNs的学习

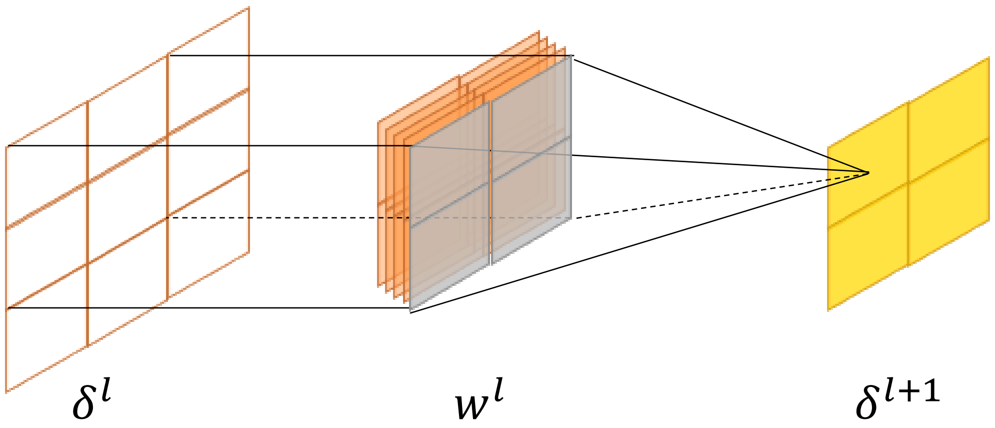


反向传播卷积层中的δ

CNNs的学习



前向计算

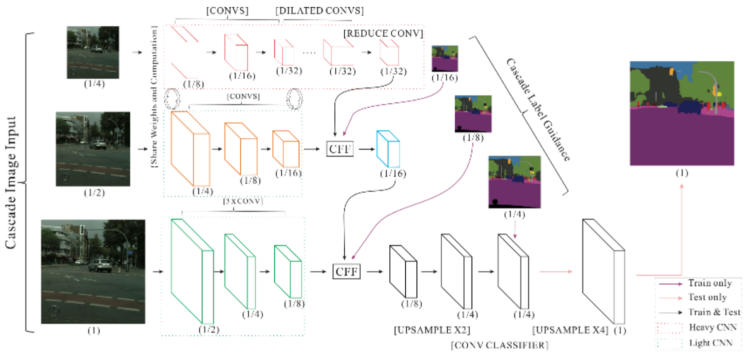


反向传播卷积层中的δ

大纲

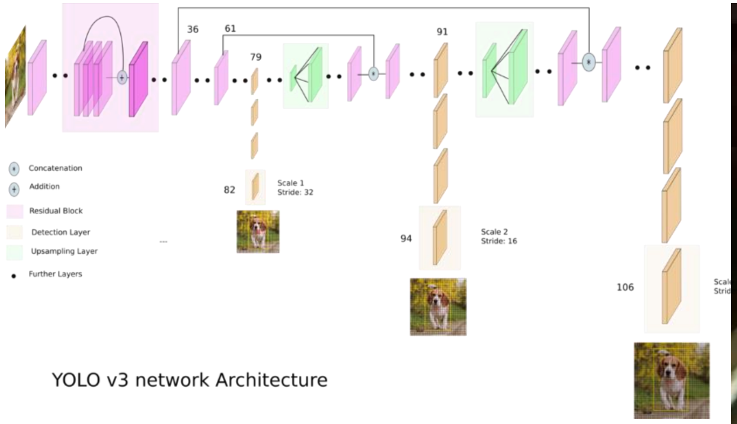
- CNNs的起源
- CNNs的组件
- CNNs的结构
- CNNs的学习
- CNNs的应用

CNNs的应用



- 实时语义分割

CNNs的应用

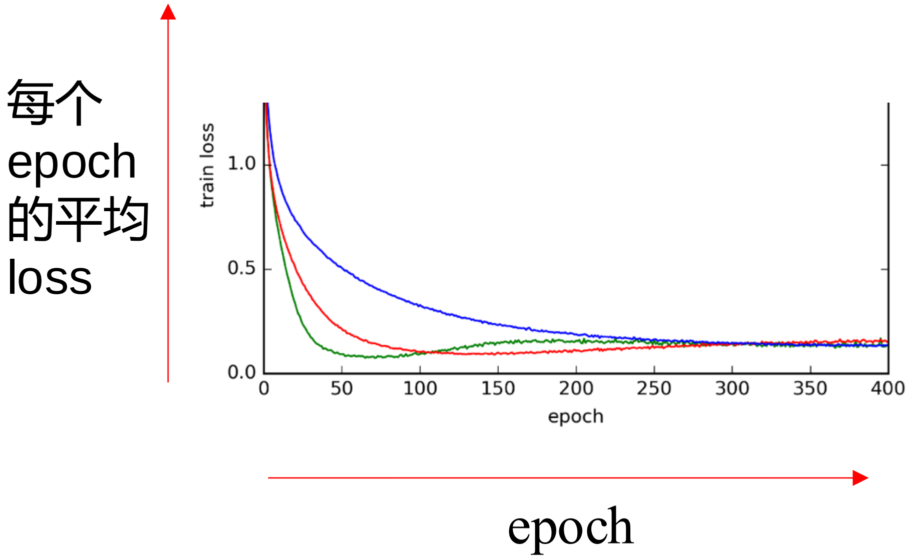


YOLO v3 network Architecture

■ 实时目标检测

课后作业

- 针对CIFAR-10数据集，训练一个5-8层的卷积神经网络
  - 对比全连接神经网络与卷积神经网络的训练集和验证集loss曲线
  - 汇报卷积神经网络的测试集准确率



- epoch: 所有训练样本迭代完一次称为一个epoch
- 选择验证集上准确率最高的模型，汇报其在测试集上的准确率

谢 谢!