北京航空航天大學

COLLEGE OF SOFTWARE BEIHANG UNIVERSITY 软件学院

人工智能

第7讲：深度学习II

卷积神经网络、自注意力机制与模型训练策略

张晶

2025年春季

- 参考资料： 吴飞，《人工智能导论：模型与算法》，高等教育出版社
- 在线课程： https://www.icourse163.org/course/ZJU-1003377027?from=searchPage
- 本部分参考：李宏毅，《机器学习》课程，台湾大学

提纲

一、线性回归与梯度下降

二、前馈神经网络

三、卷积神经网络

四、序列数据模型

五、深度学习应用

提纲

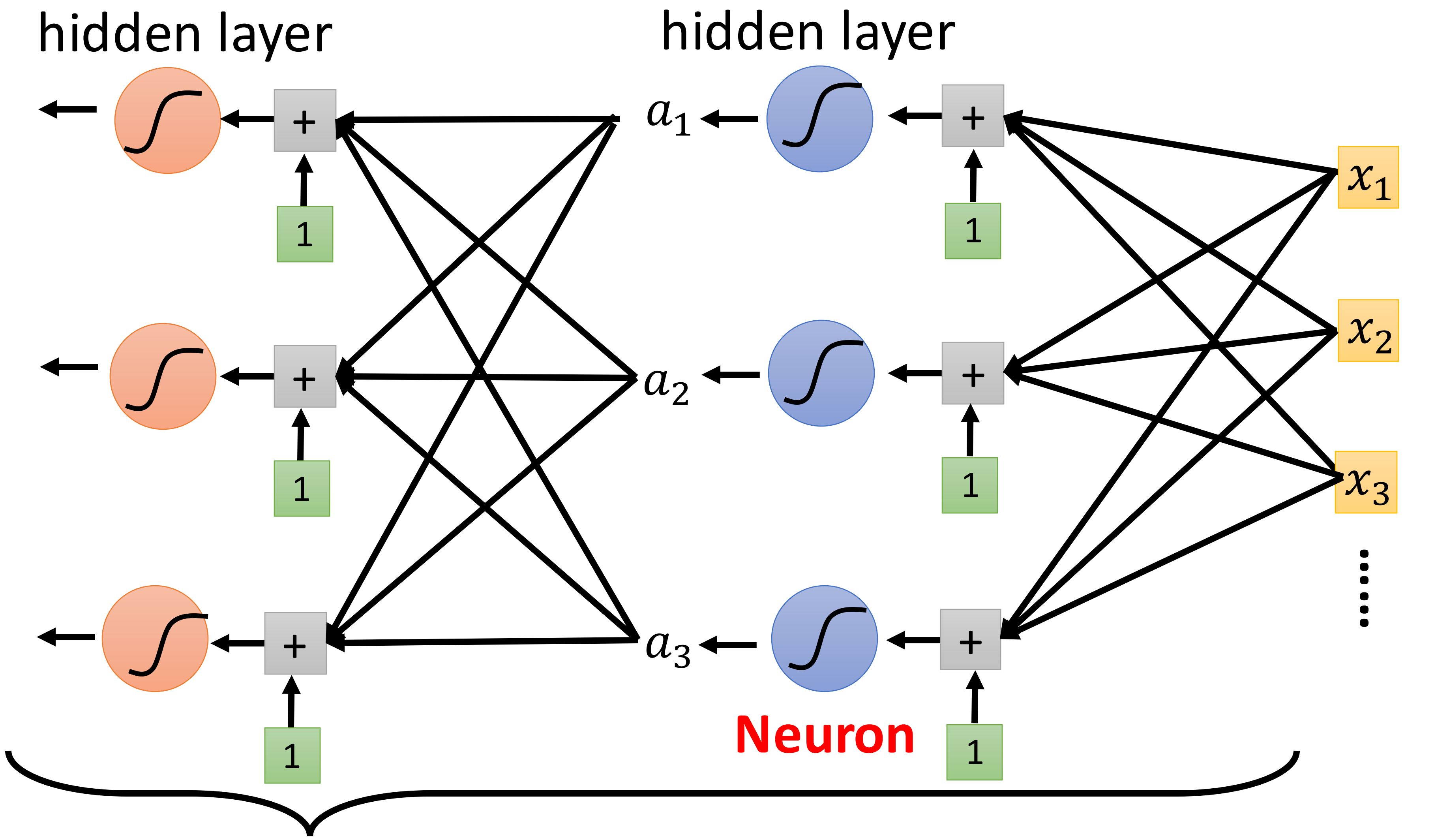
一、线性回归与梯度下降

二、前馈神经网络

三、卷积神经网络

四、序列数据模型

五、深度学习应用



神经网络 Neural Network

Many layers means **Deep** ➡ 深度学习Deep Learning

前馈神经网络

- 输入层、输出层和至少一层的隐藏层构成。网络中各个隐藏层中神经元可接收相邻前序隐藏层中所有神经元传递而来的信息，经过加工处理后将信息输出给相邻后续隐藏层中所有神经元。
- 各个神经元接受前一级的输入，并输出到下一级，模型中没有反馈
- 层与层之间通过“全连接”进行链接，即两个相邻层之间的神经元完全成对连接，但层内的神经元不相互连接。
- 也被称为全连接网络，或多层感知机。

提纲

一、线性回归与梯度下降

二、前馈神经网络

三、卷积神经网络

四、序列数据模型

五、深度学习应用

Convolutional Neural Networks
卷积神经网络

Network architecture designed for image

图像分类

100 x 100

Model

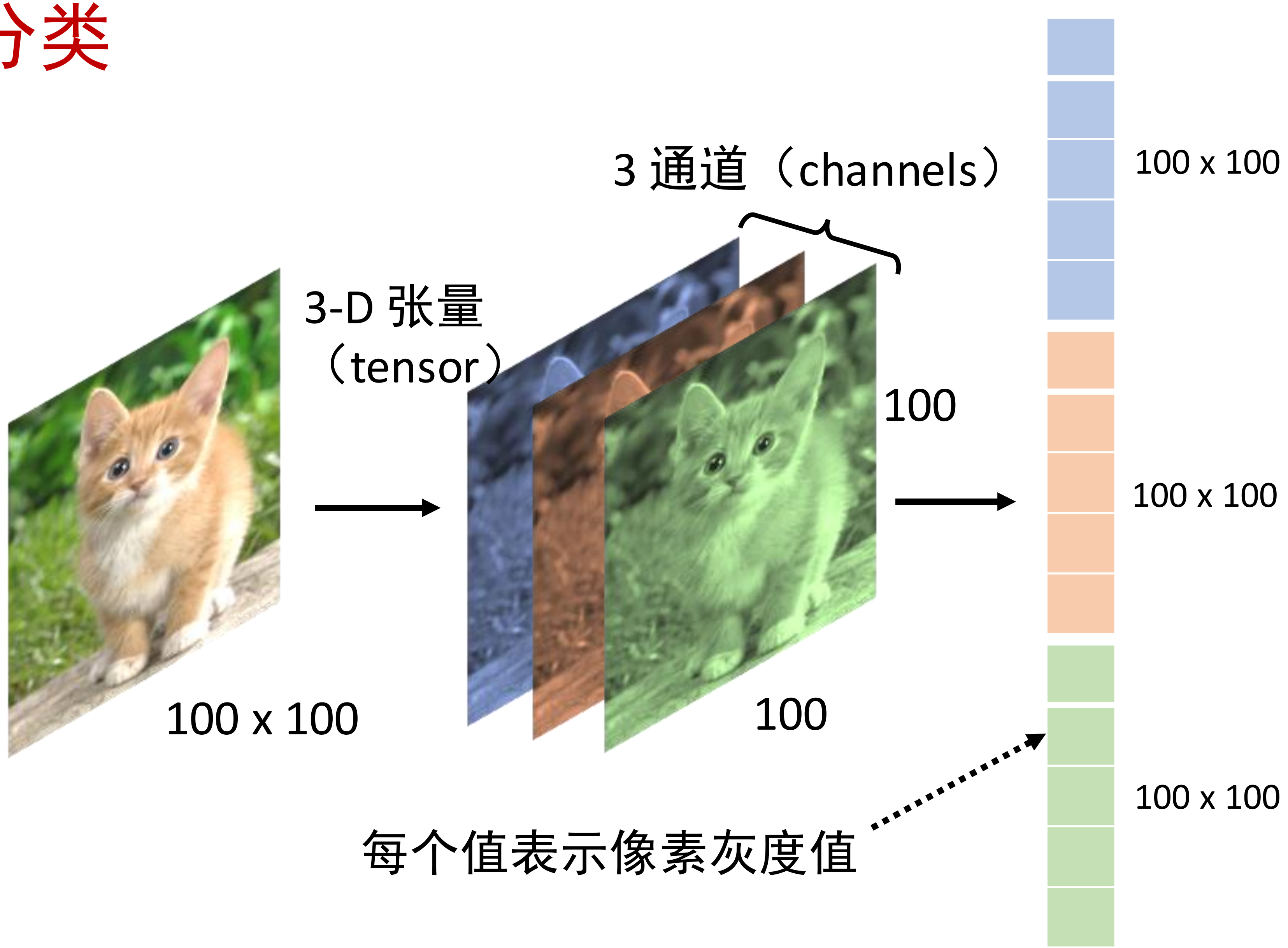
$$\begin{bmatrix} \vdots \\ 0.2 \\ 0.7 \\ 0.1 \\ \vdots \end{bmatrix} \qquad \begin{array}{r} \\ \text{dog} \\ \text{cat} \\ \text{tree} \\ \\ \end{array} \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

$\boldsymbol{y}' \longleftrightarrow \widehat{\boldsymbol{y}}$

交叉熵损失

（待分类图像具有相同尺寸）

图像分类


3-D 张量
（tensor）
3 通道（channels）
100
100 x 100
100
100 x 100
100 x 100
100 x 100
每个值表示像素灰度值

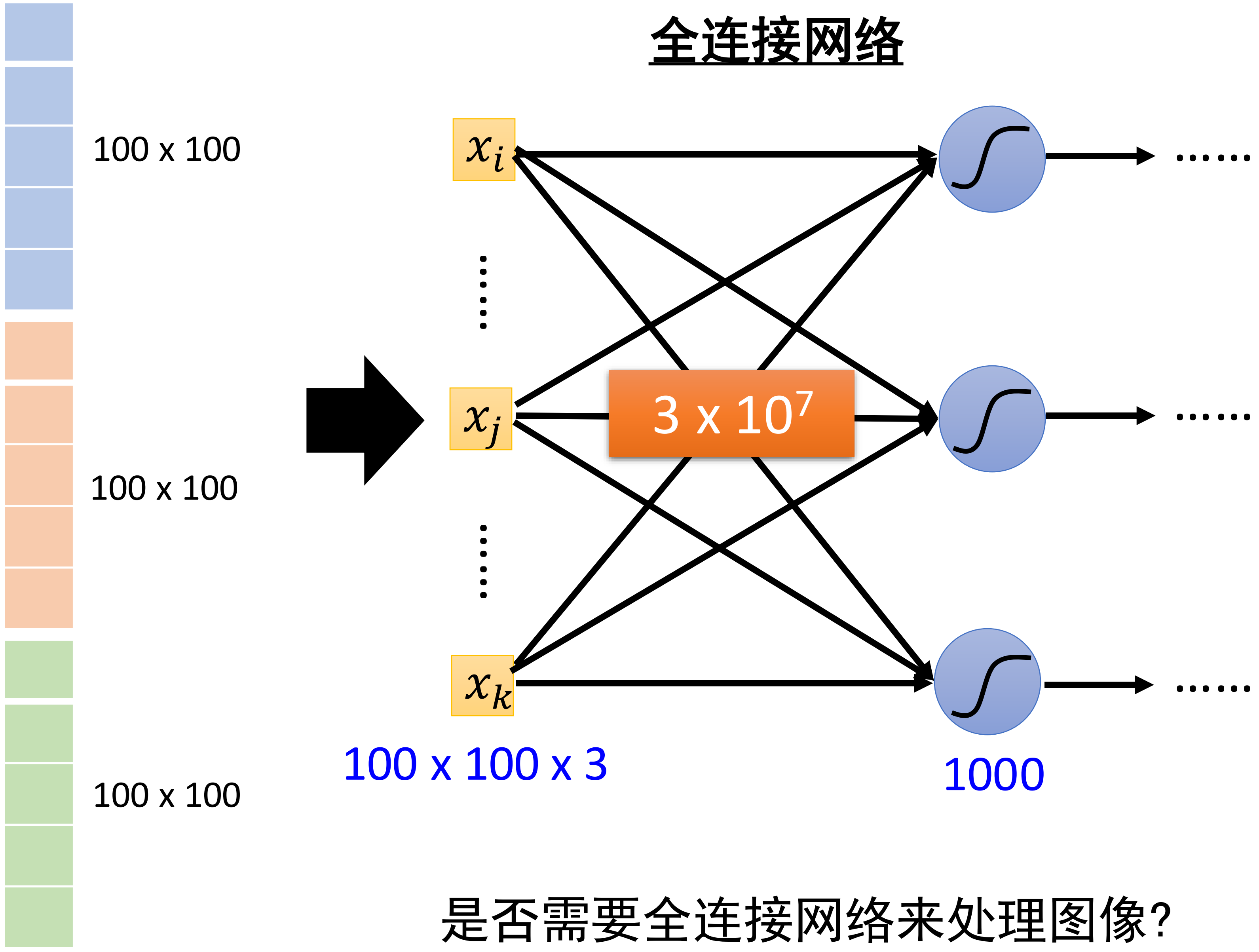
填空题 1分

设置

假设输入图像大小为3 x 100 x 100，分类类别数为1000个， 如果用一层全连接网络实现分类，需要多少参数? [填空1]

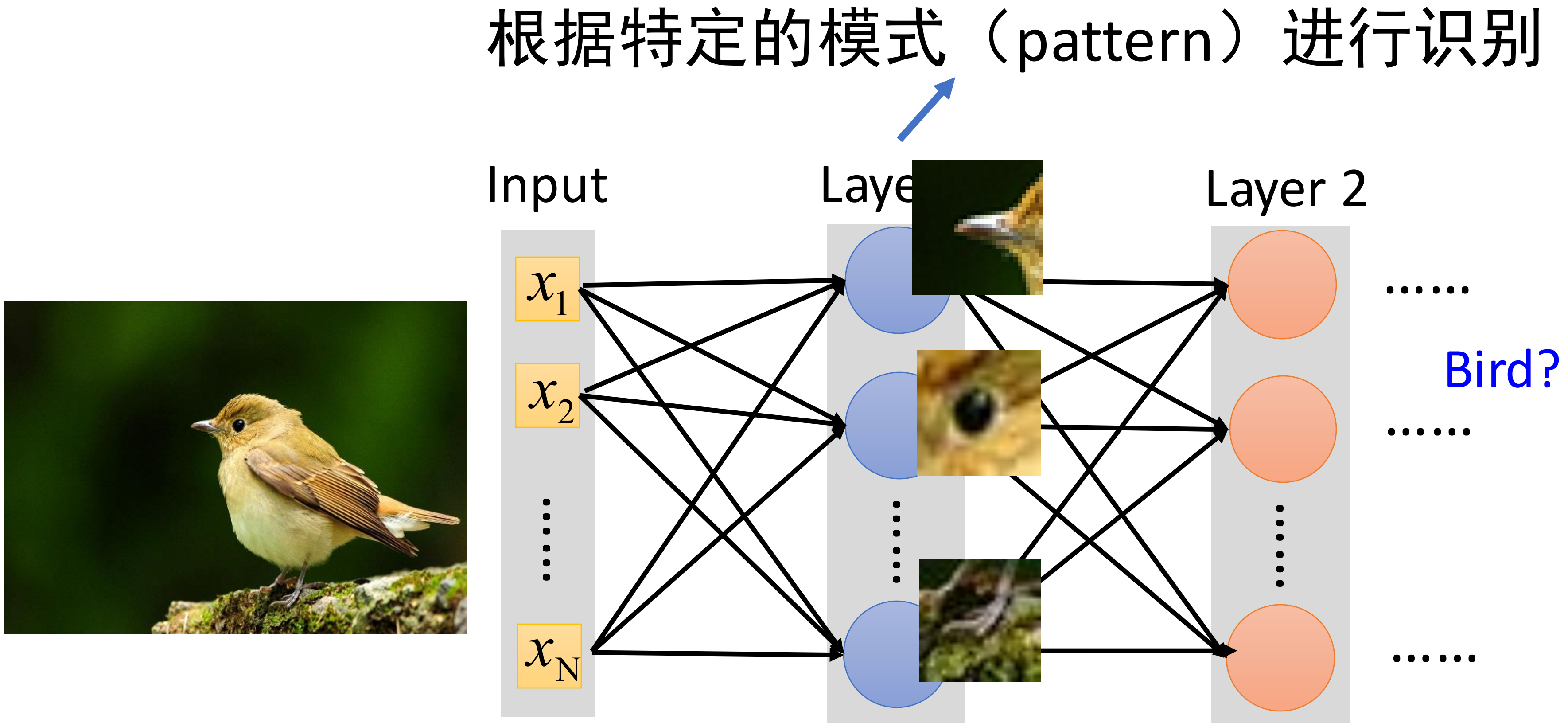
作答

全连接网络



是否需要全连接网络来处理图像?

观察 1

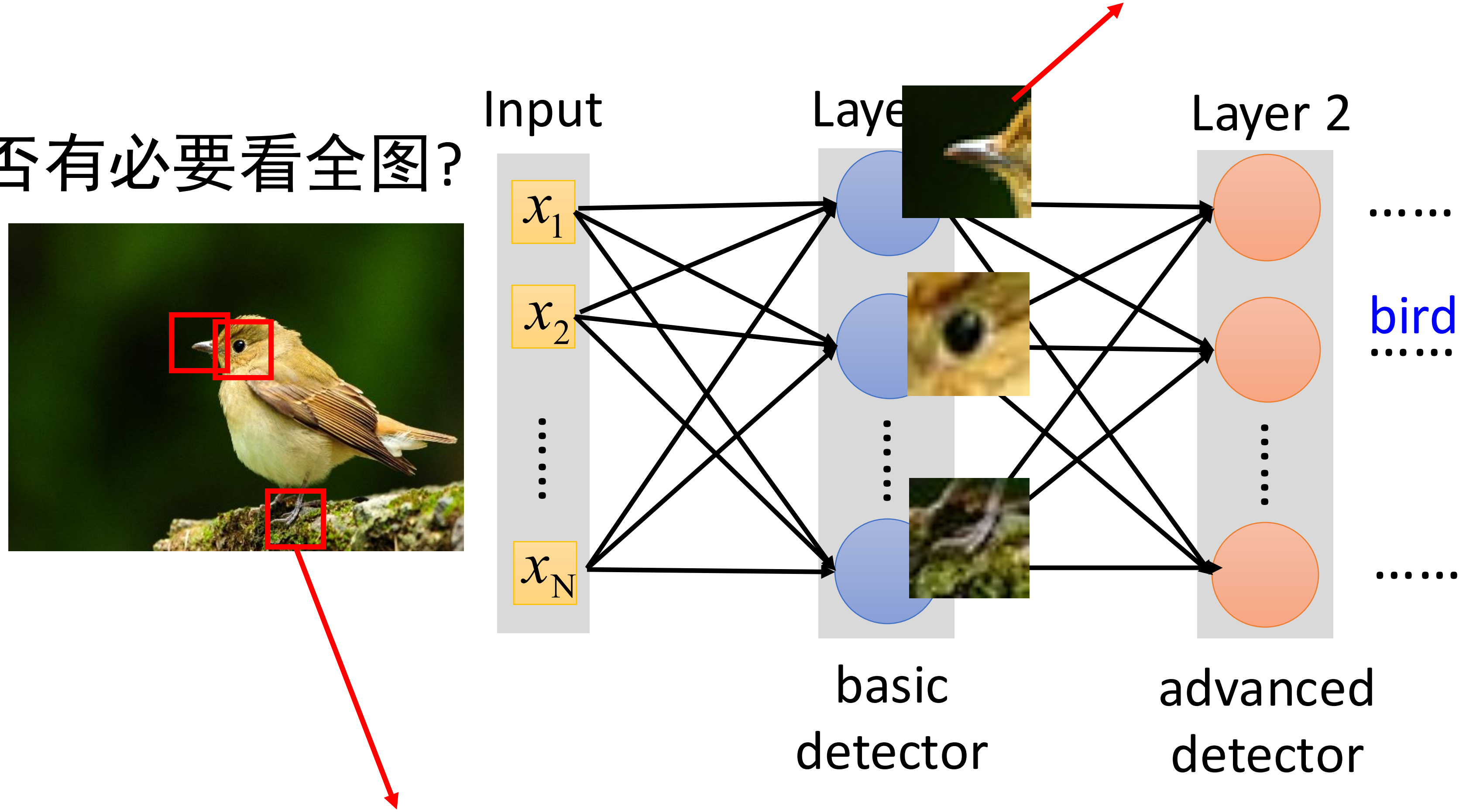


人也许用的是同样的方式实现鸟的识别… ☺

观察 1

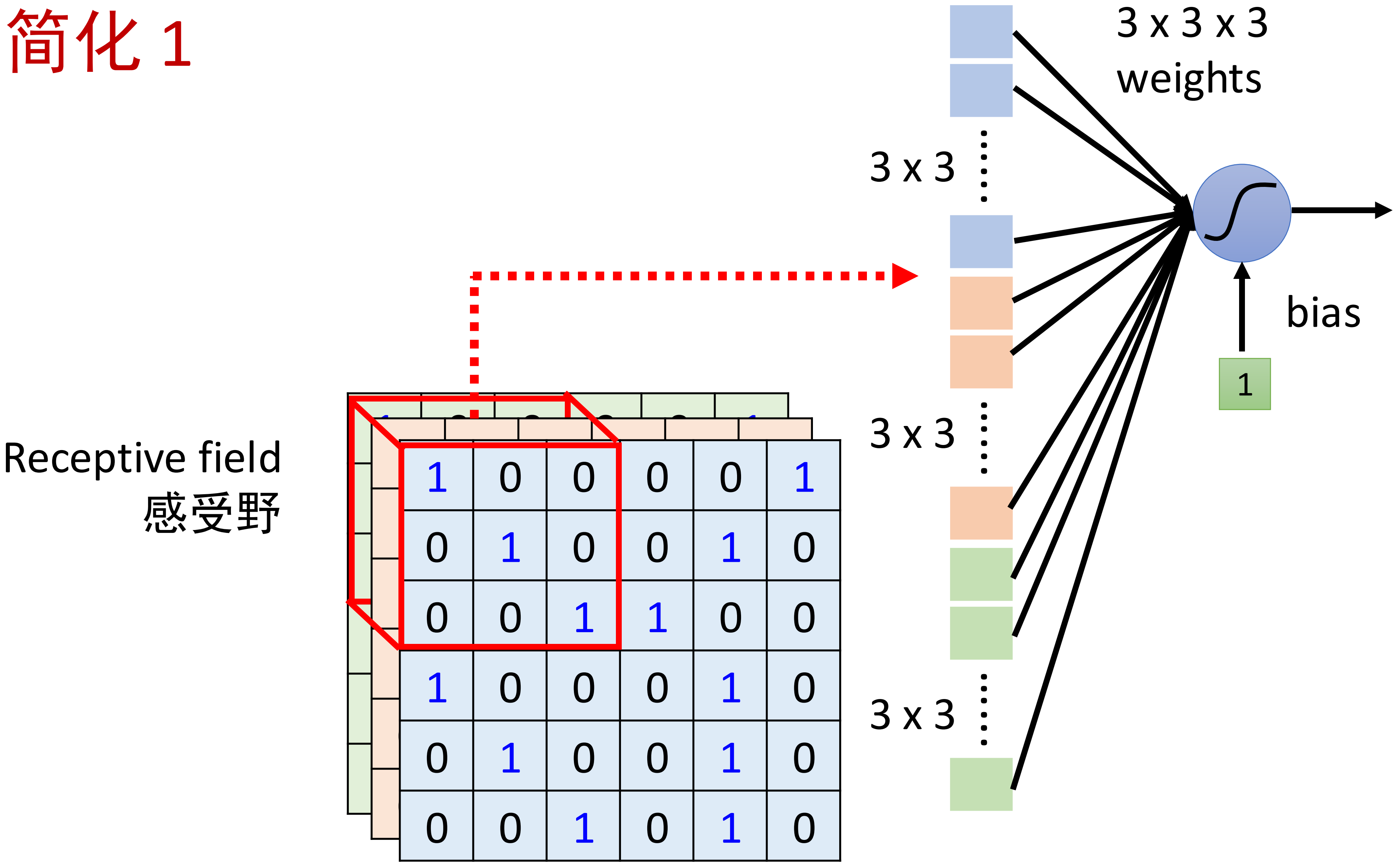
每个神经元无需覆盖整个图像.

是否有必要看全图?



很多patterns只占据图像的一个小区域.

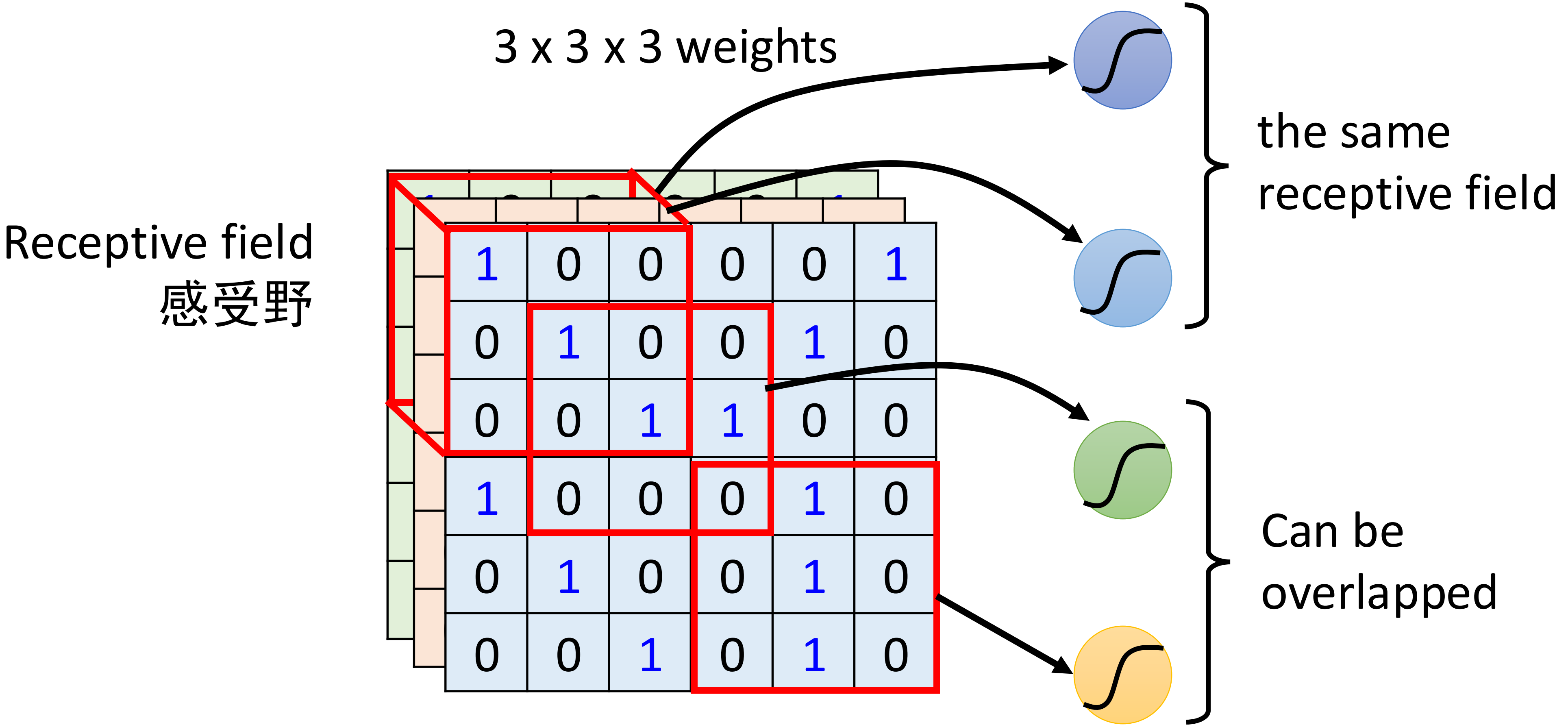
简化 1



Receptive field
感受野
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
3 x 3
3 x 3
3 x 3
3 x 3 x 3
weights
bias
1

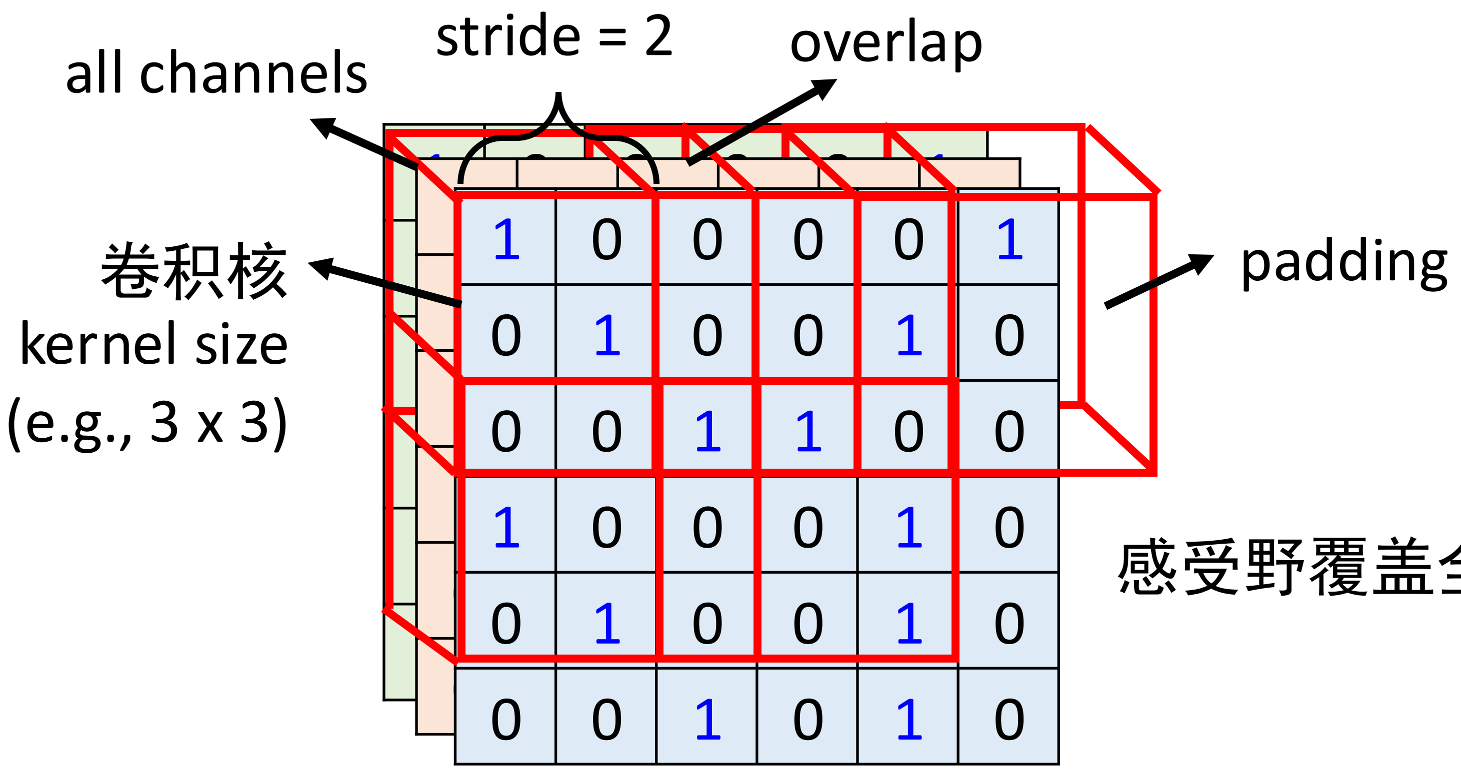
简化 1

- 不同神经元覆盖不同感受野大小?
- 只覆盖部分通道?
- 非正方形感受野?



简化 1 –典型设置

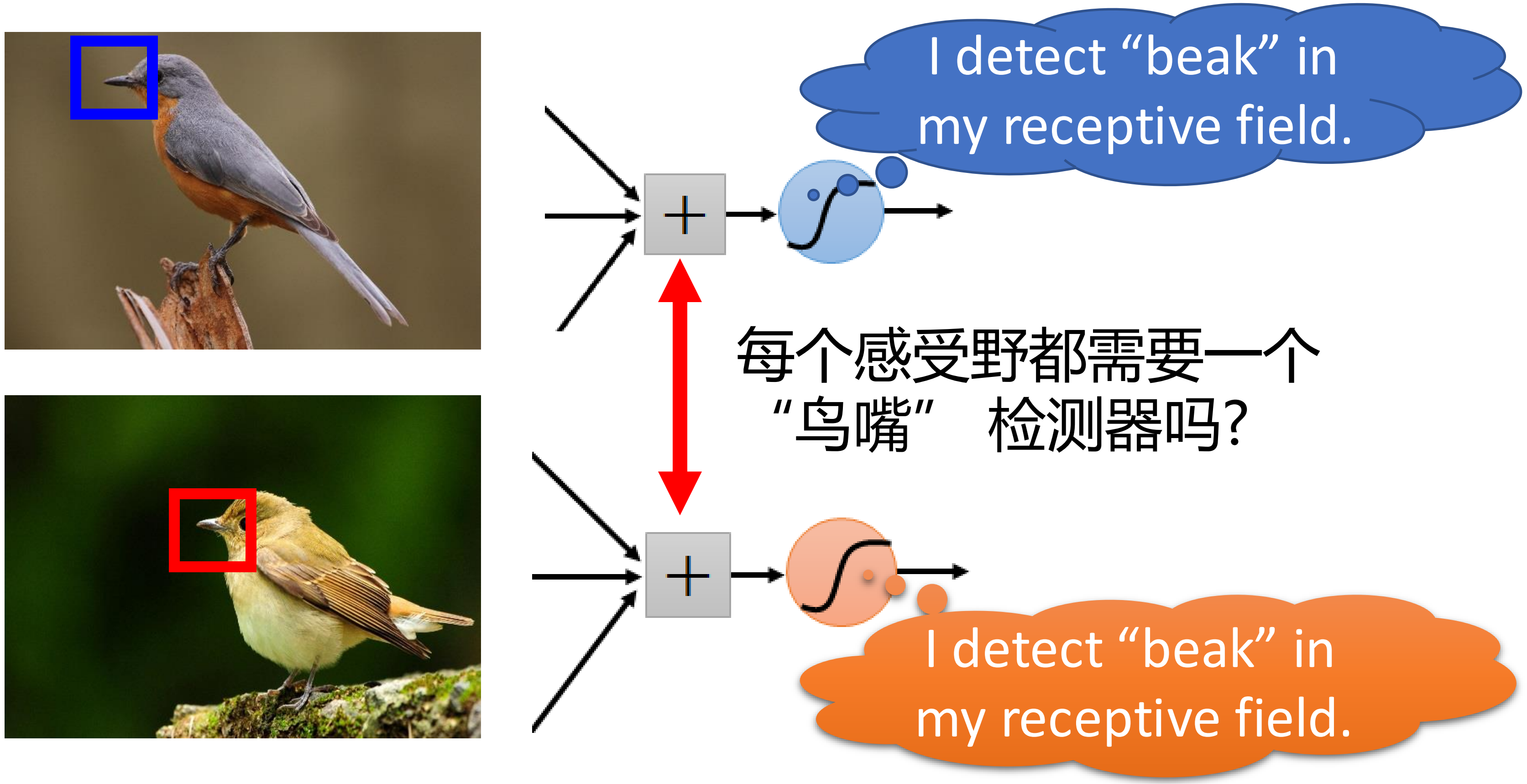
每个感受野覆盖一组神经元 (e.g., 64 neurons).



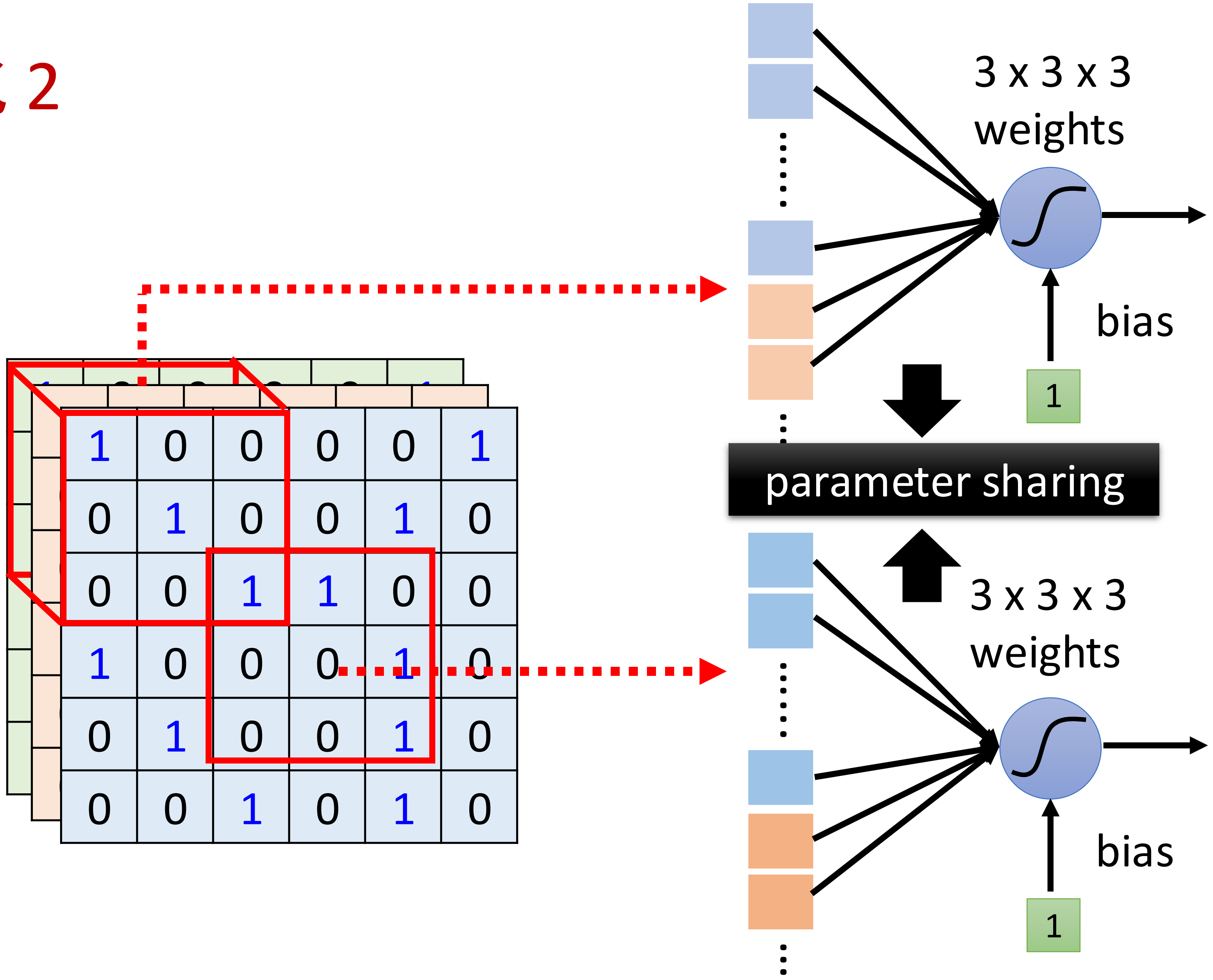
感受野覆盖全图.

观察 2

- 同一pattern可能出现在不同图像的不同区域



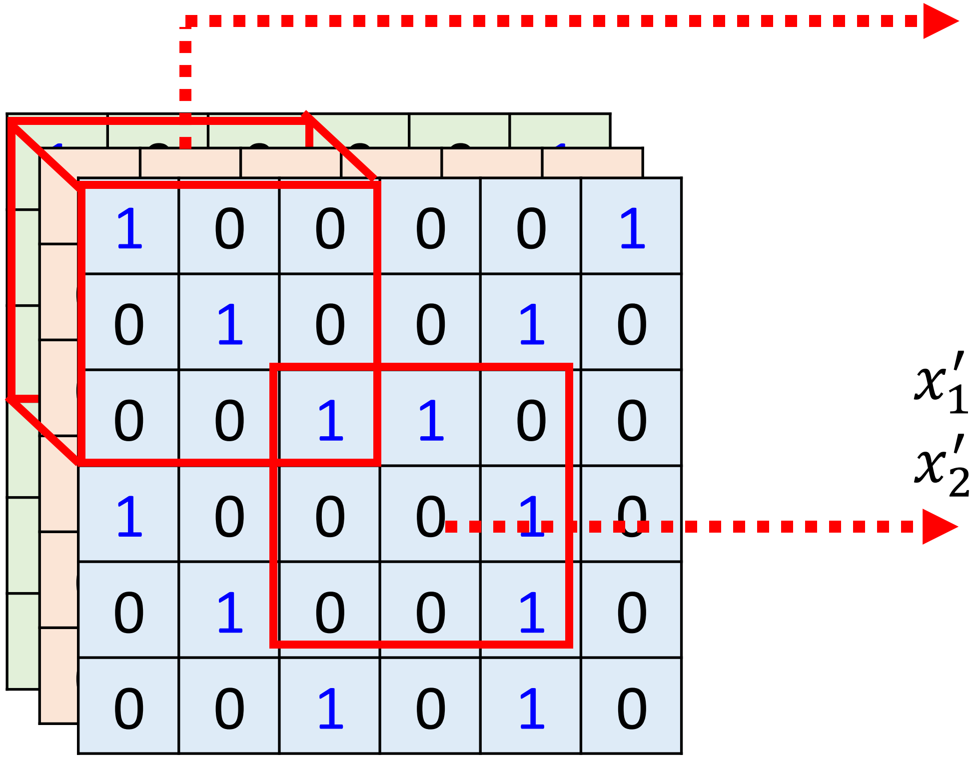
简化 2

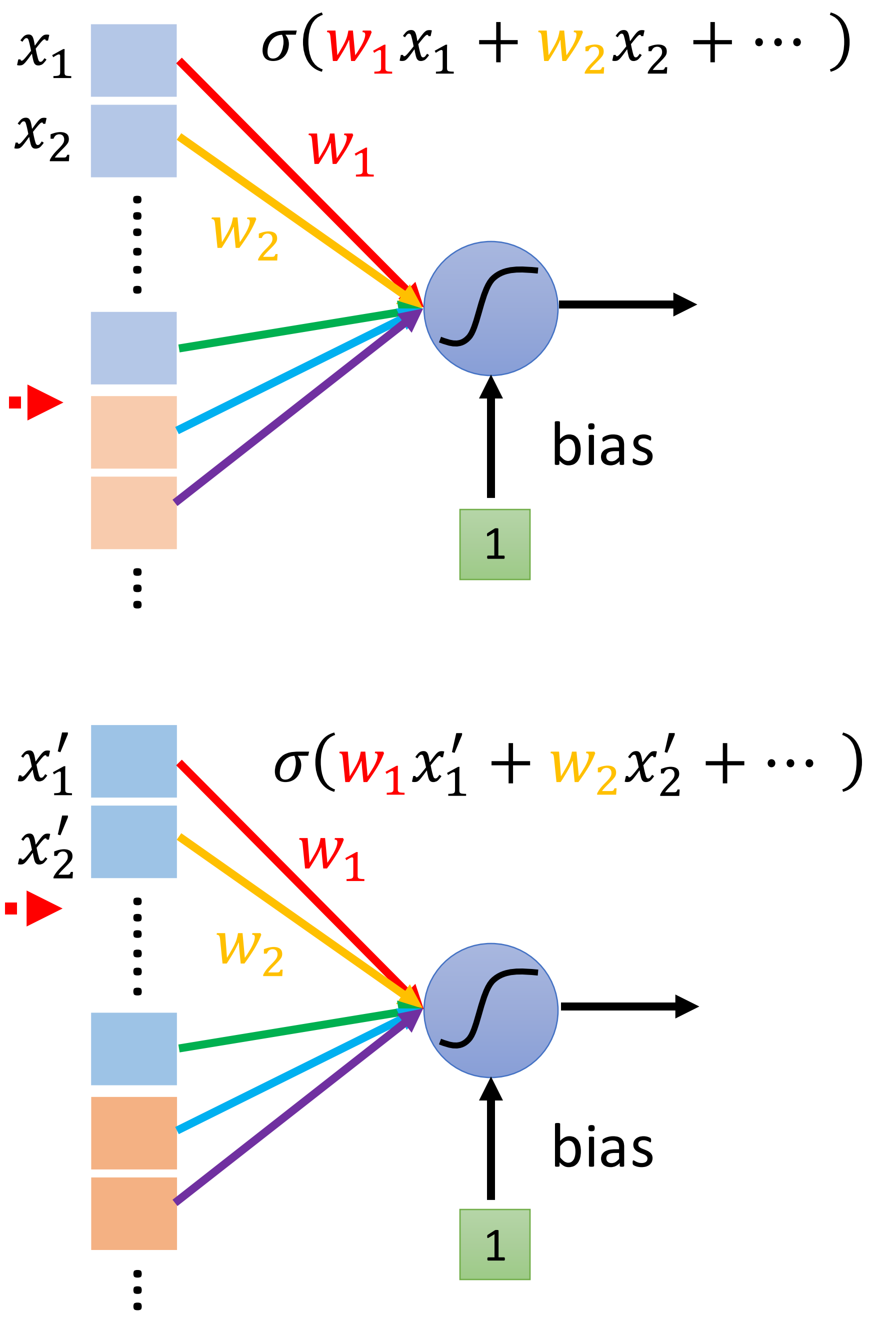


1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
3 x 3 x 3
weights
bias
1
parameter sharing
3 x 3 x 3
weights
bias
1

参数共享

简化 2

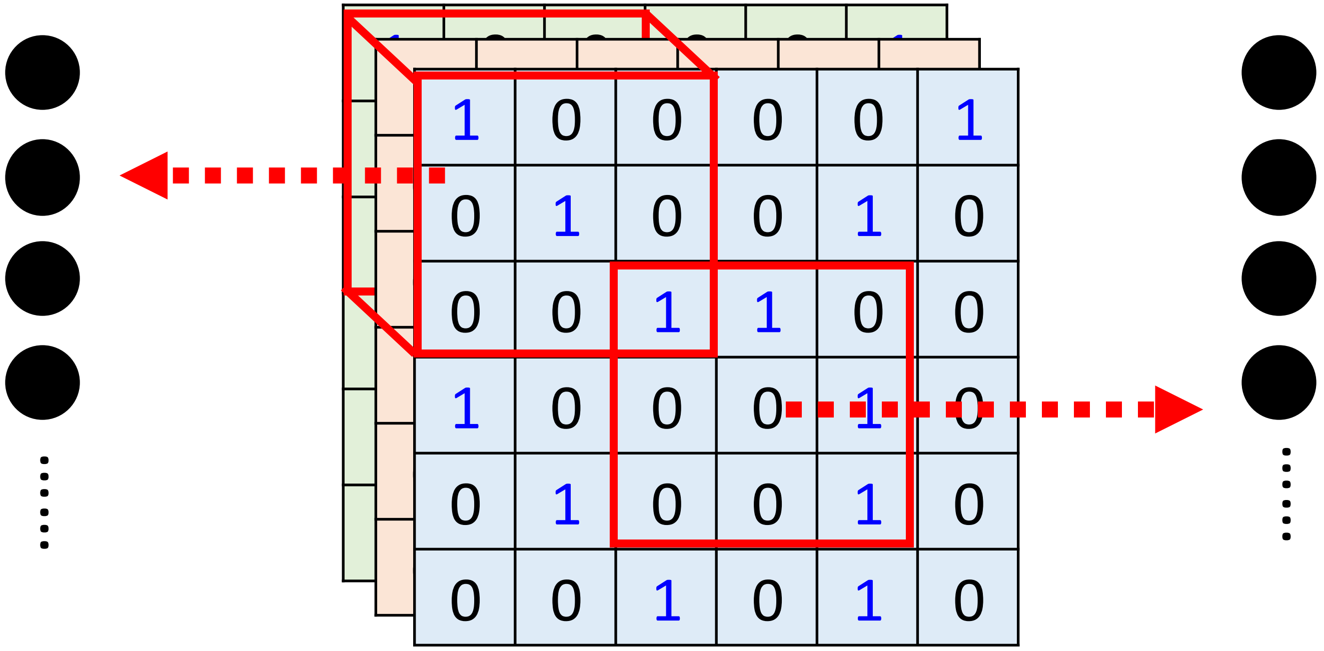




同一感受野的不同神经元不共享参数

简化 2 – 典型设置

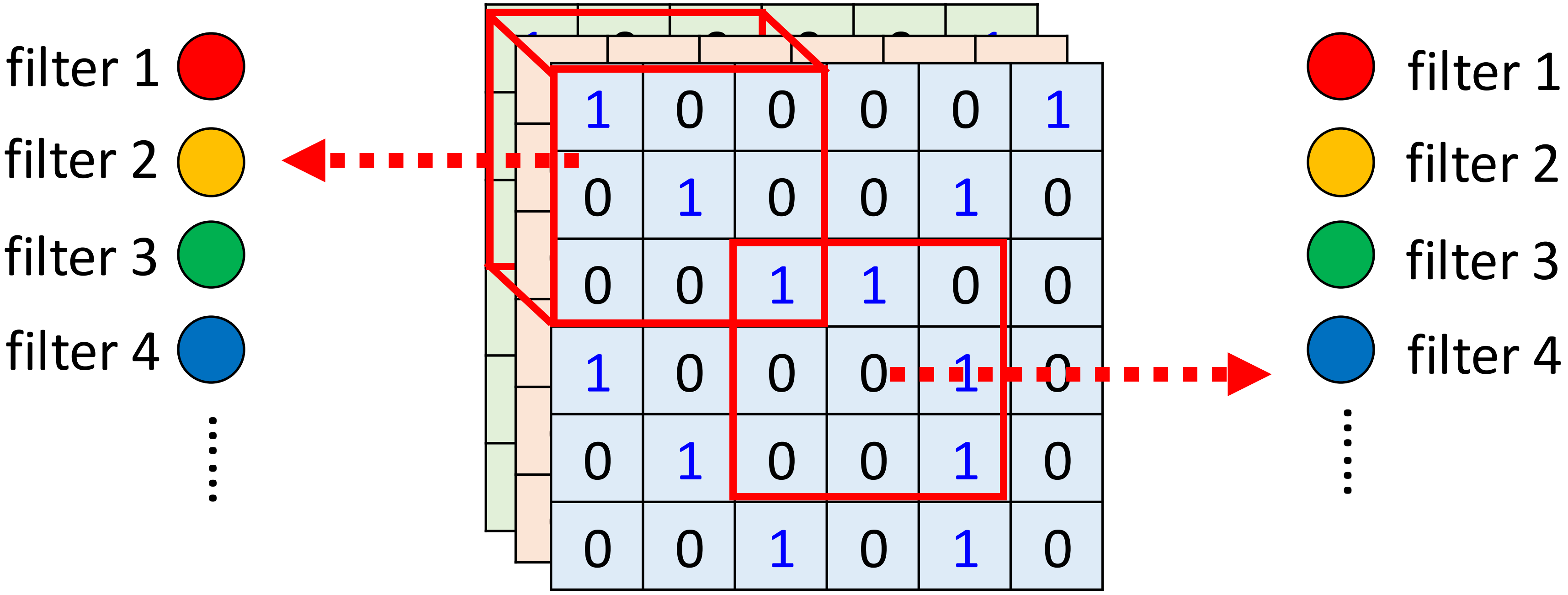
每个感受野有多个神经元 (e.g., 64 个神经元).



简化 2 – 典型设置

每个感受野有多个神经元 (e.g., 64 个神经元).

不同感受野的神经元共享参数（filter，滤波器）



填空题 1分

设置

请思考，一层卷积神经网络的参数与哪些因素有关？ [填空1]

作答

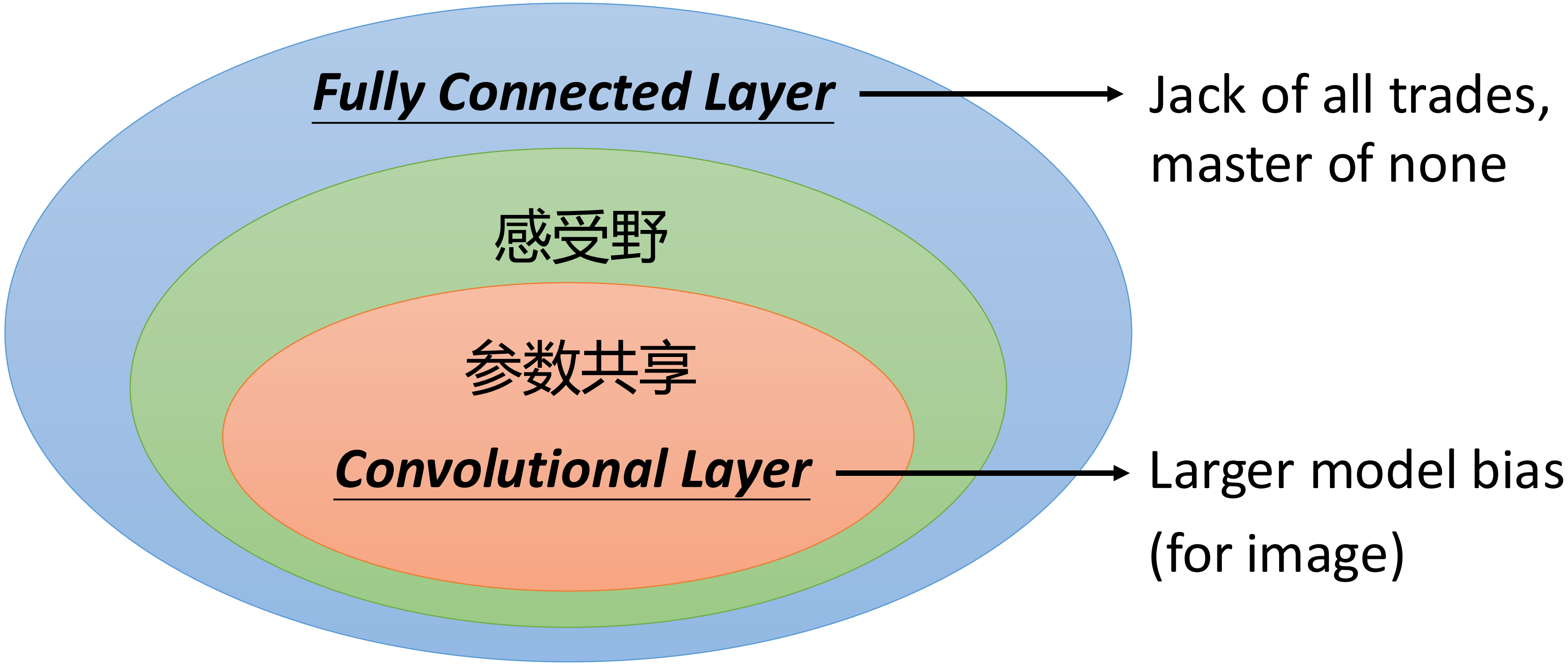
主观题 10分

设置

请思考，卷积神经网络与全连接网络相比，参数量有什么变化？为什么卷积神经网络优于全连接网络？

作答

卷积层的优势

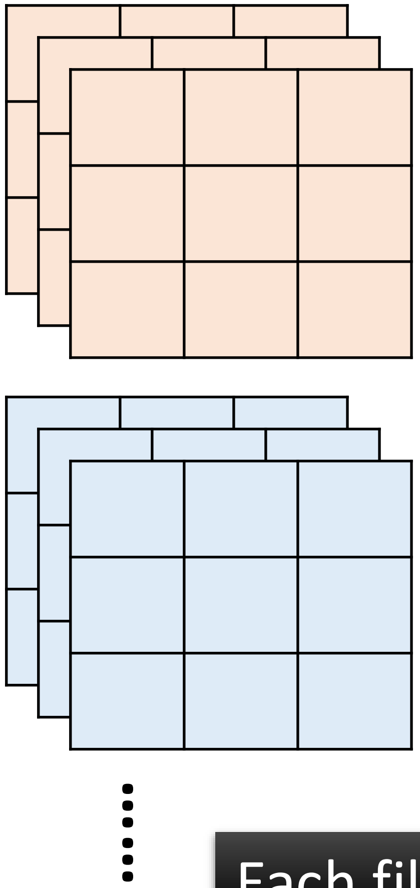


- 通常pattern会远小于整图大小
- 同样的pattern会出现在图上不同区域

卷积层

Another story based on ***filter*** ☺

Convolution



Each filter detects a small pattern (3 x 3 x channel).

通道channel = 3 (colorful)

通道channel = 1 (black and white)

卷积层

Consider channel = 1
（假设黑白图）

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1

⋮

(filters 中的数值即为待学习的未知参数)

卷积层

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

卷积层

-1	1	-1
-1	1	-1
-1	1	-1

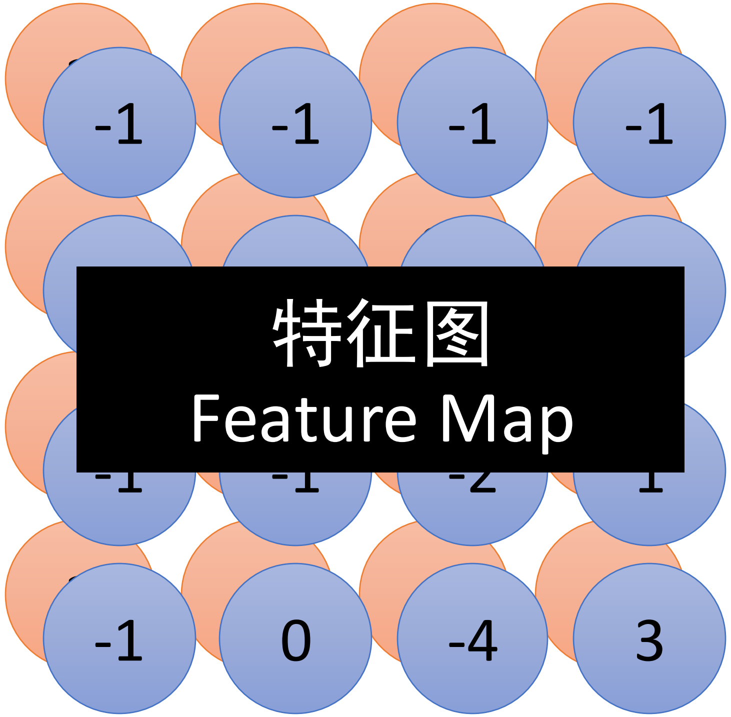
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

对每个filter重复此过程



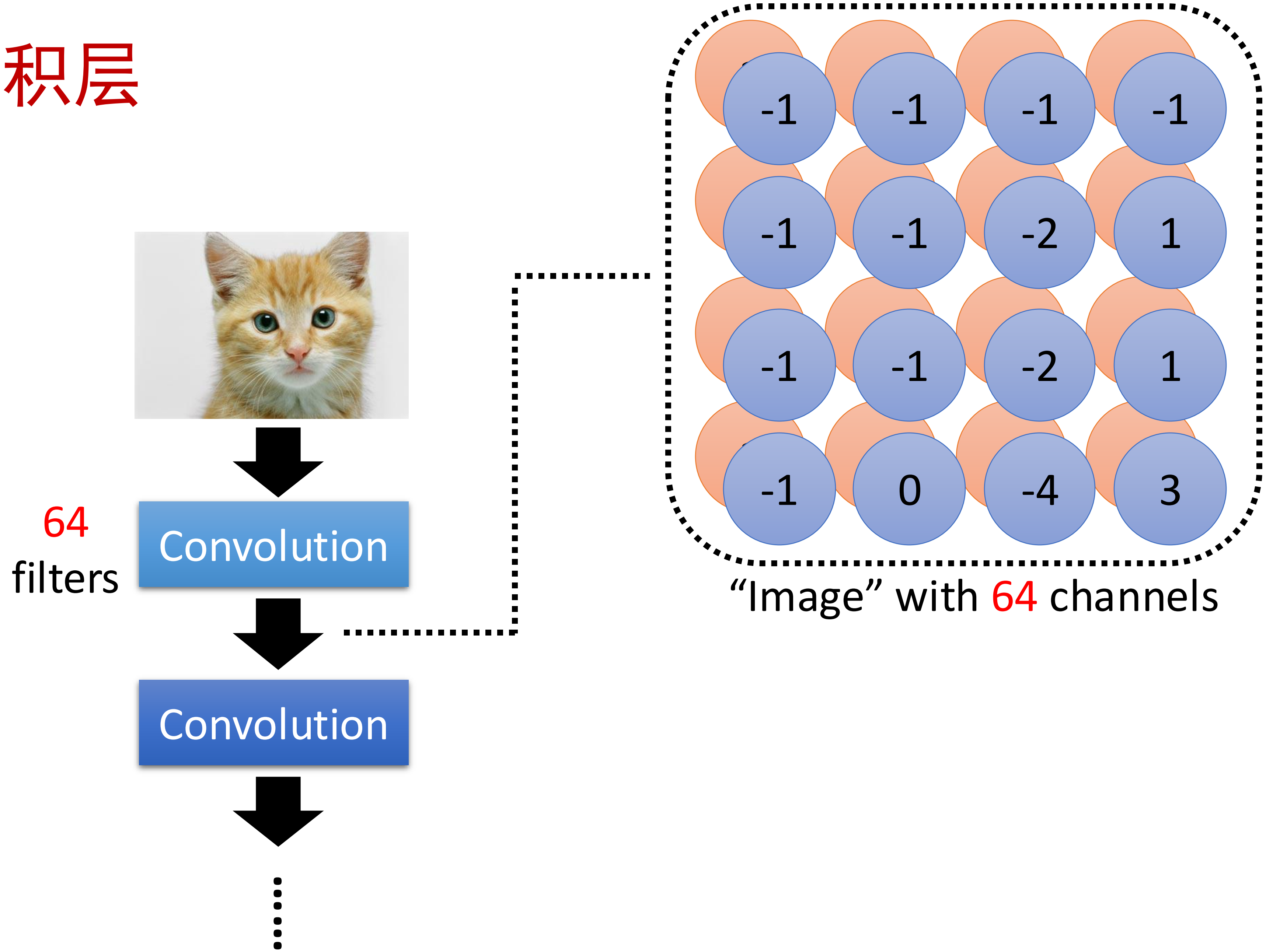
主观题 10分

设置

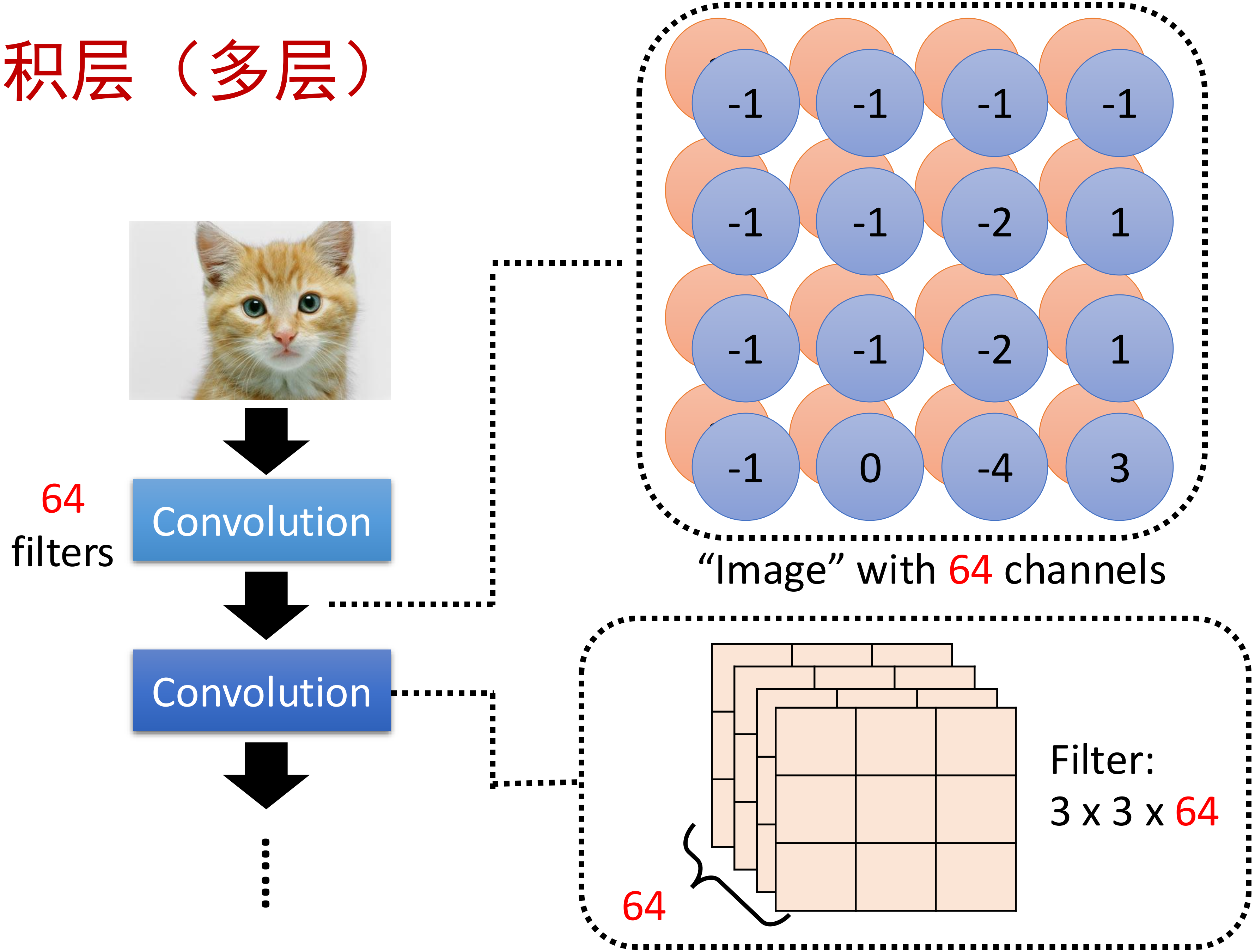
请思考，如果想捕捉图像中较大感受野的pattern，比如鸟嘴，3x3大小的卷积核是否足够？为什么？

作答

卷积层


64
filters
Convolution
Convolution
-1 -1 -1 -1
-1 -1 -2 1
-1 -1 -2 1
-1 0 -4 3
“Image” with 64 channels

卷积层（多层）



64
filters
Convolution
Convolution
-1
-1
-1
-1
-1
-1
-2
1
-1
-1
-2
1
-1
0
-4
3
"Image" with 64 channels
Filter:
3 x 3 x 64
64

卷积层（多层）

64 filters

Convolution

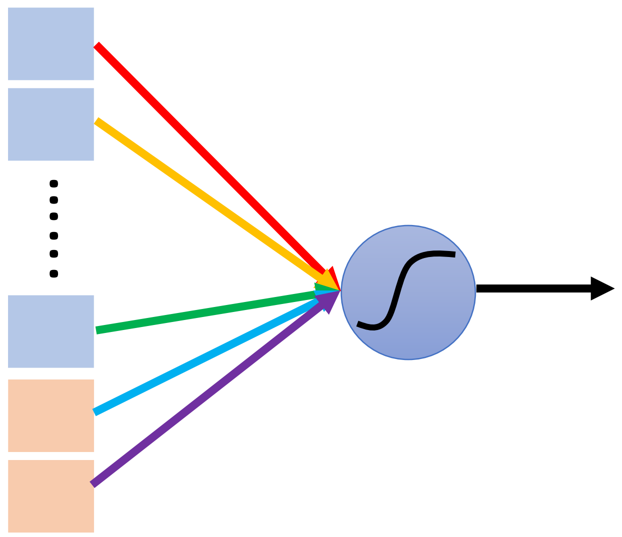
Convolution

⋮

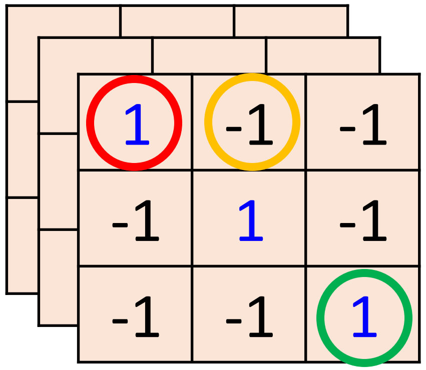
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

两种解释的对比

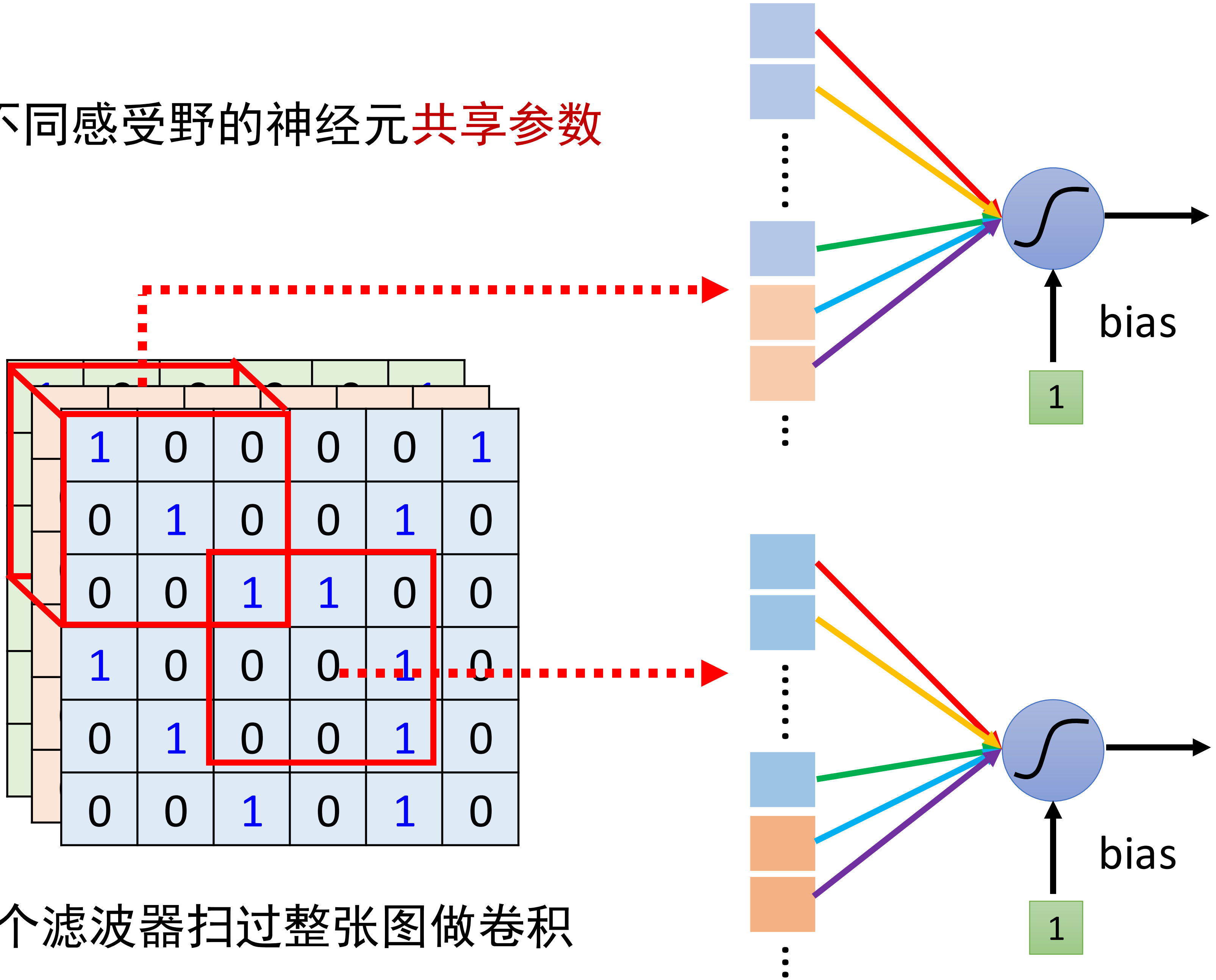


感受野
(Receptive field)



(忽略bias项)

不同感受野的神经元共享参数



每个滤波器扫过整张图做卷积

卷积层

Neuron Version Story	*Filter Version Story*
每个神经元只考虑一个感受野.	一系列的滤波器检测小的图案（pattern）.
不同感受野的神经元共享参数	每个滤波器以卷积的形式扫过整张图

They are the same story.

观察 3

- 对像素降采样不影响物体类别语义

bird

降采样

bird

池化 – 最大池化

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

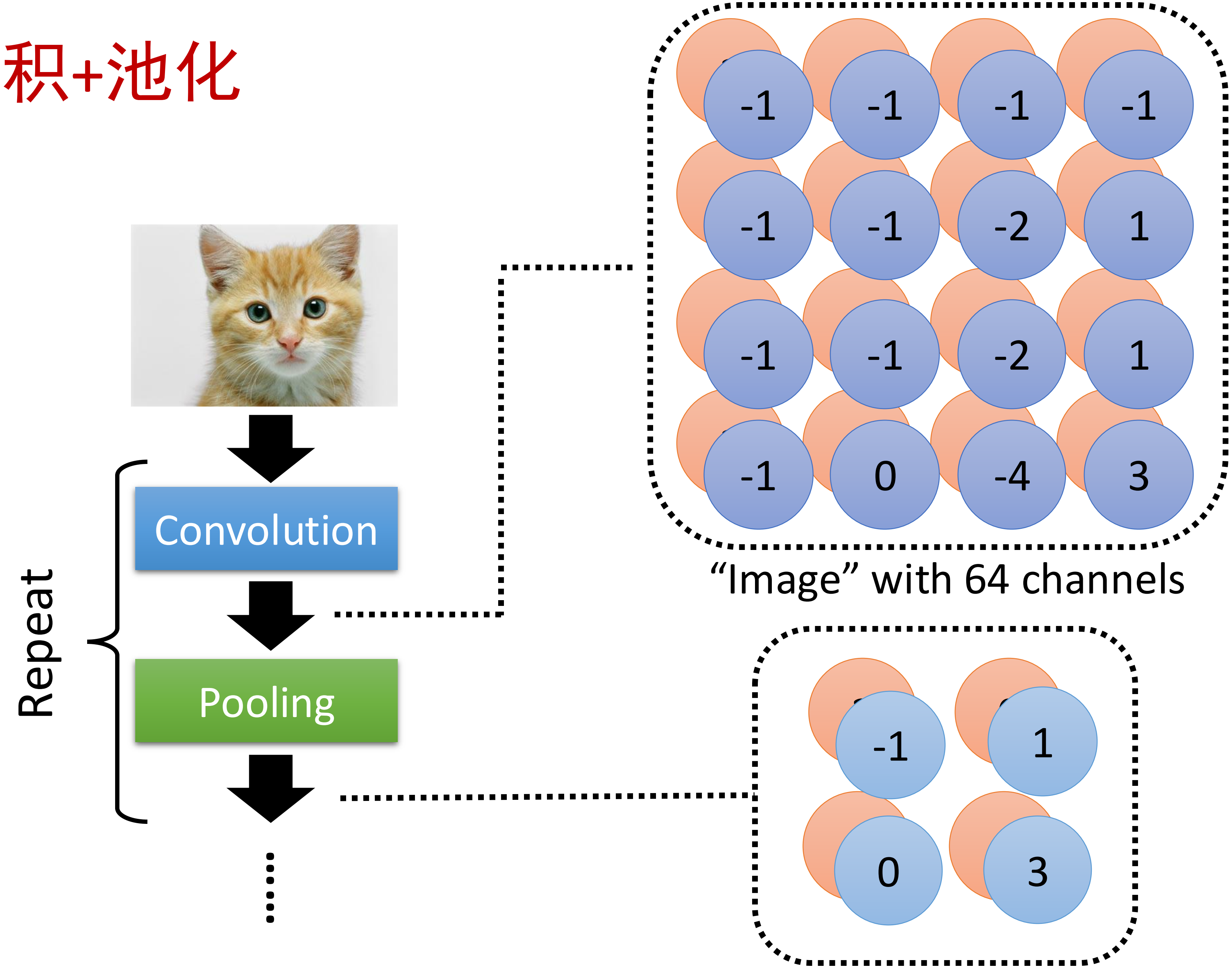
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

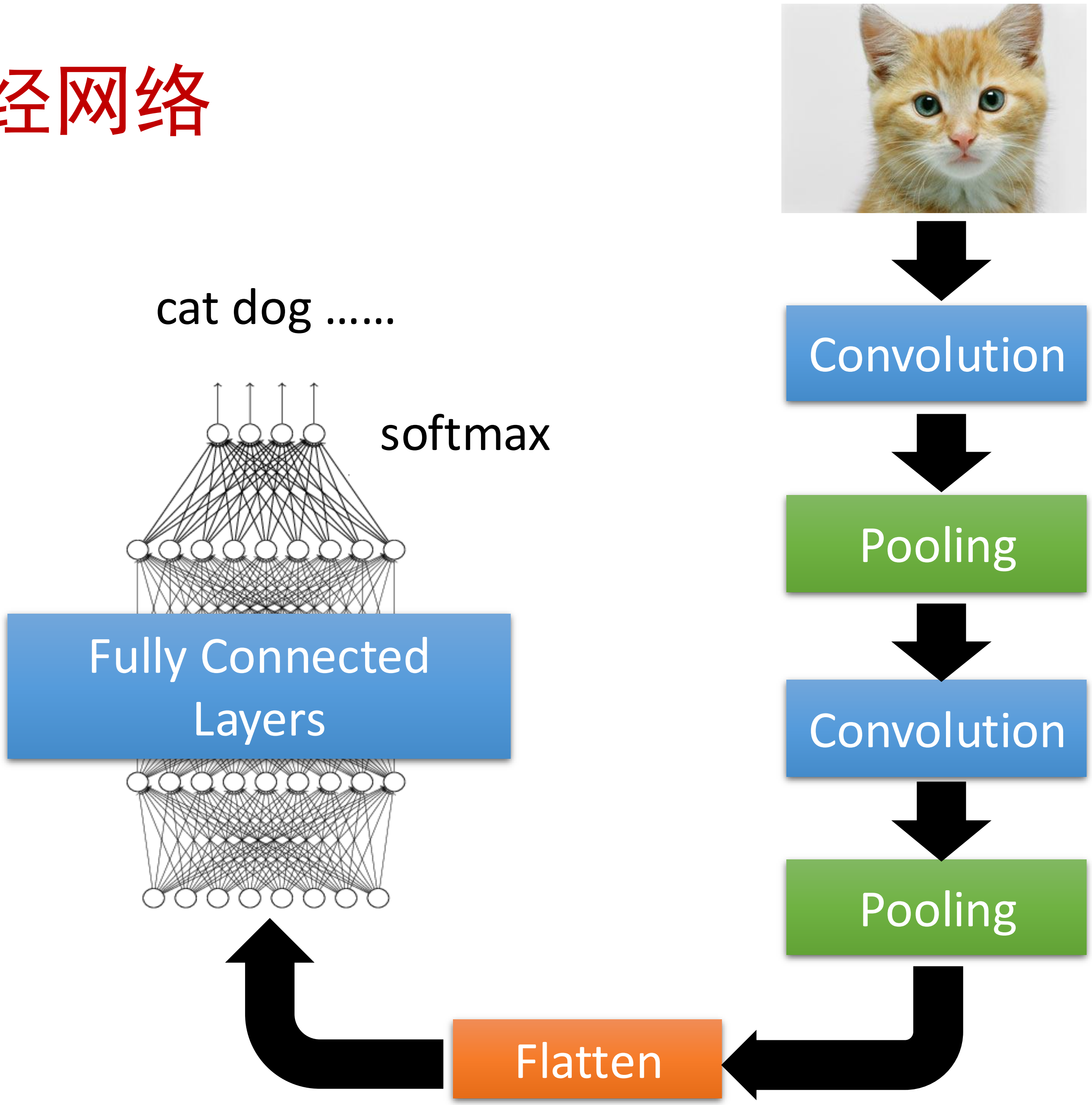
-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

卷积+池化



Convolution
Pooling
Repeat
-1 -1 -1 -1
-1 -1 -2 1
-1 -1 -2 1
-1 0 -4 3
“Image” with 64 channels
-1 1
0 3

卷积神经网络



cat dog ......
softmax
Fully Connected Layers
Convolution
Pooling
Convolution
Pooling
Flatten

问题和不足

- CNN中的卷积操作不满足尺度和旋转不变性 (需要数据增广☺)

北京航空航天大学
COLLEGE OF SOFTWARE BEIHANG UNIVERSITY 软件学院

提纲

一、线性回归与梯度下降

二、前馈神经网络

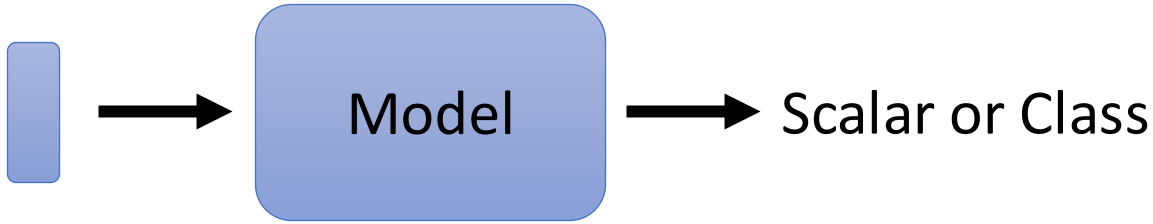
三、卷积神经网络

四、序列数据模型

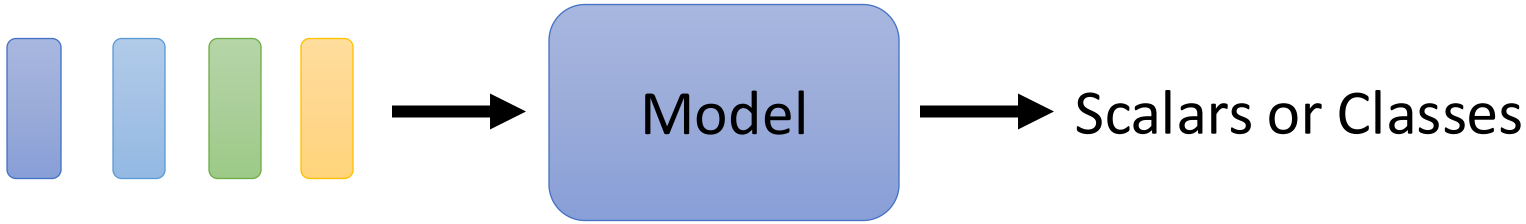
五、深度学习应用

复杂输入数据

- 向量输入（vector input）

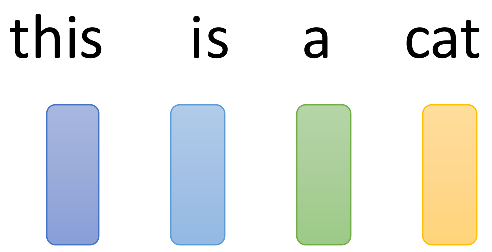


- 向量集输入（vector set input）



(变长序列)

向量集输入



One-hot Encoding

apple = [1 0 0 0 0 ……]

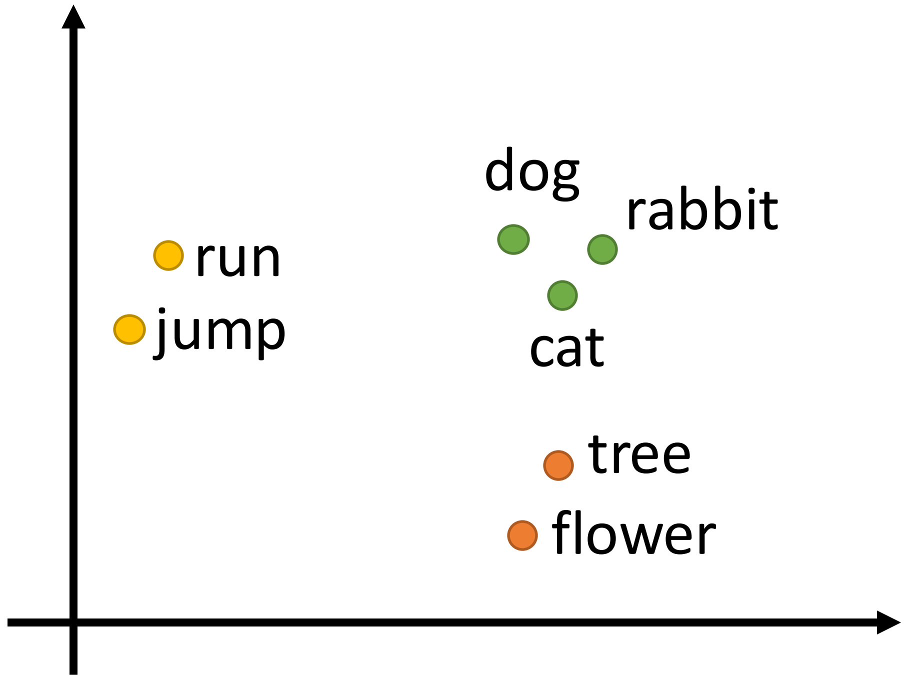
bag = [0 1 0 0 0 ……]

cat = [0 0 1 0 0 ……]

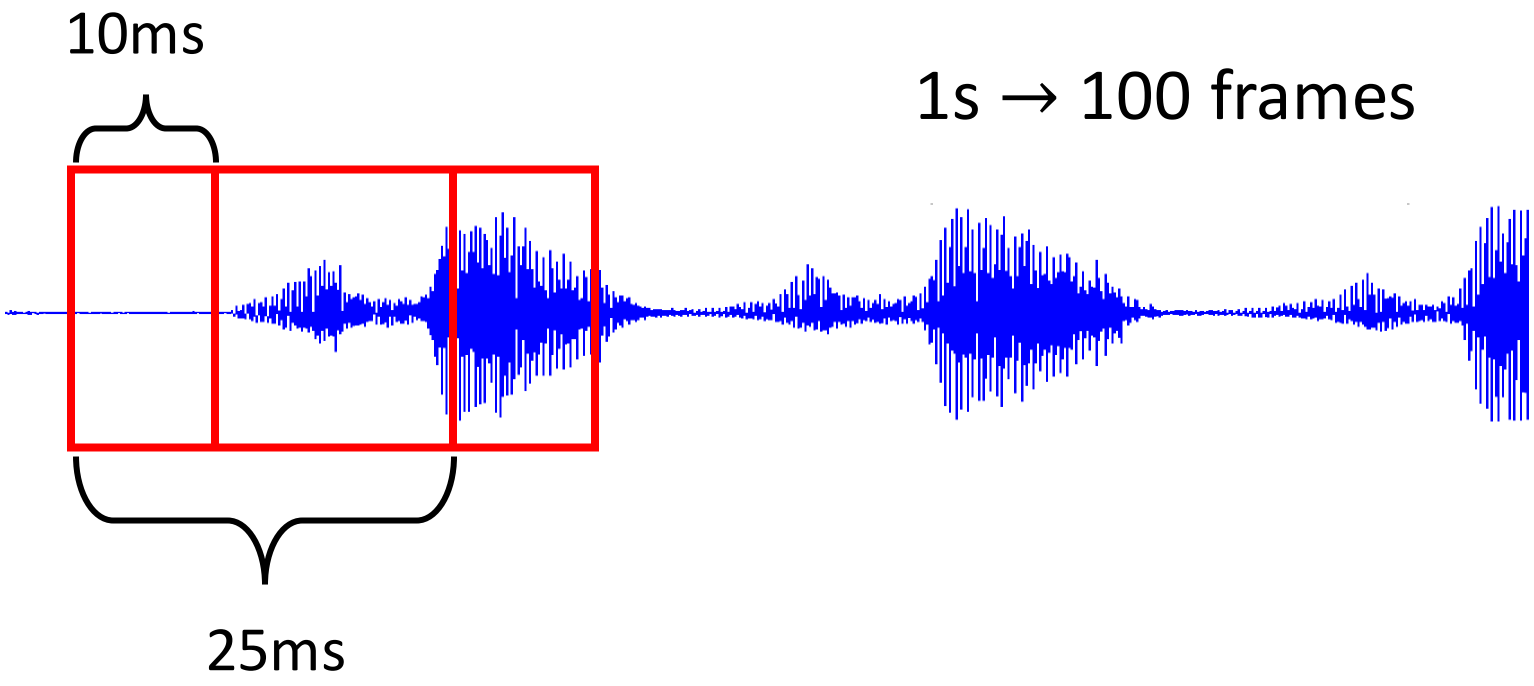
dog = [0 0 0 1 0 ……]

elephant = [0 0 0 0 1 ……]

Word Embedding



向量集输入



frame

400 sample points (16KHz)
39-dim MFCC
80-dim filter bank output

向量集输入

https://medium.com/analytics-vidhya/social-network-analytics-f082f4e21b16

- 图数据(Graph)也可以看做向量集 (每个结点看做一个向量)

Each profile
is a vector

http://www.twword.com/wiki/%E5%88%86%E5%AD%90

向量集输入

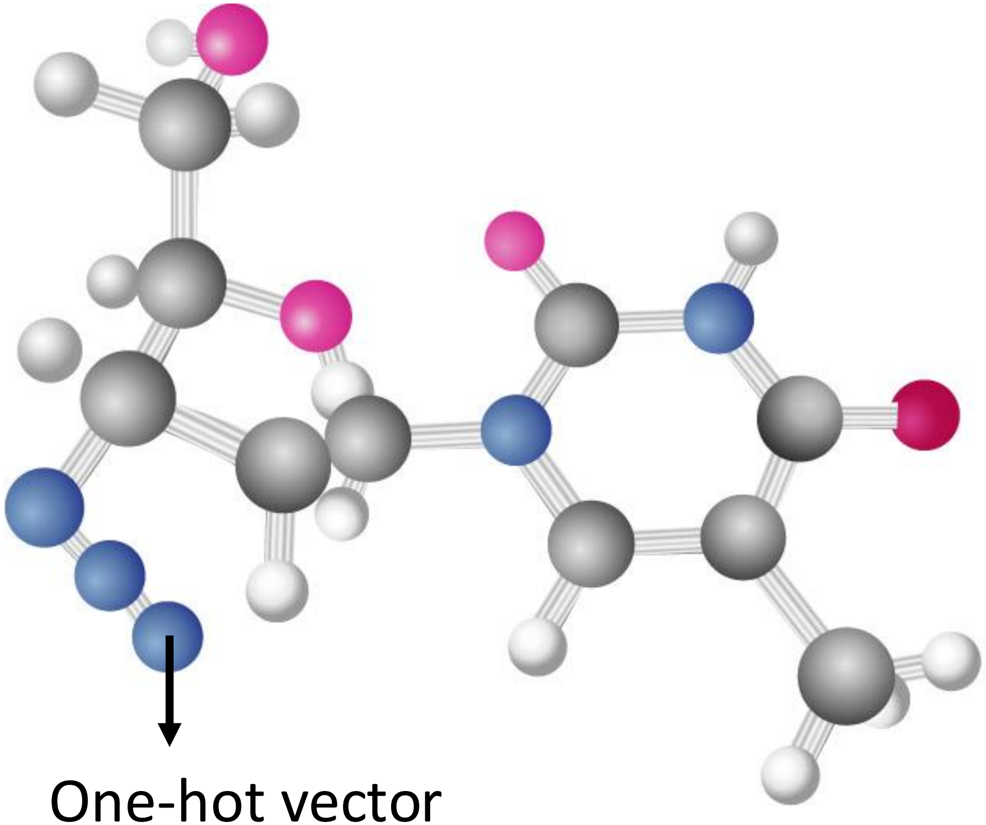
- 图数据(Graph)也可以看做向量集 (每个结点看做一个向量)

H = [1 0 0 0 0 ……]

C = [0 1 0 0 0 ……]

O = [0 0 1 0 0 ……]

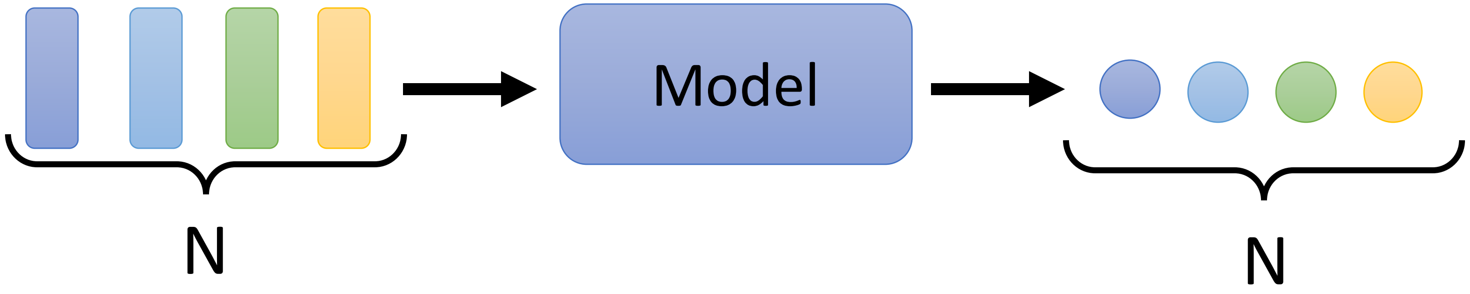
⋮



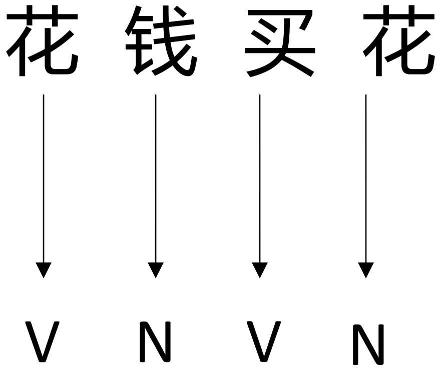
One-hot vector

输出形式

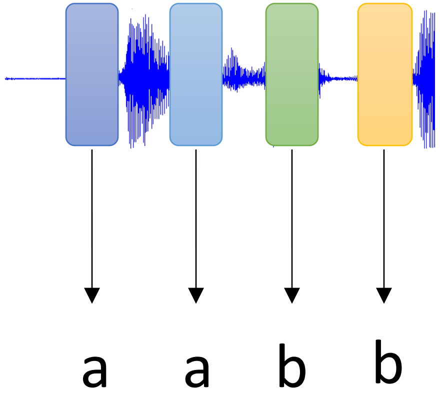
- 一对一输出



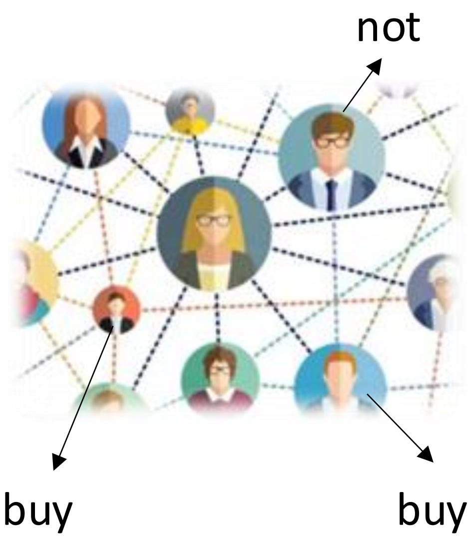
应用



词性标注



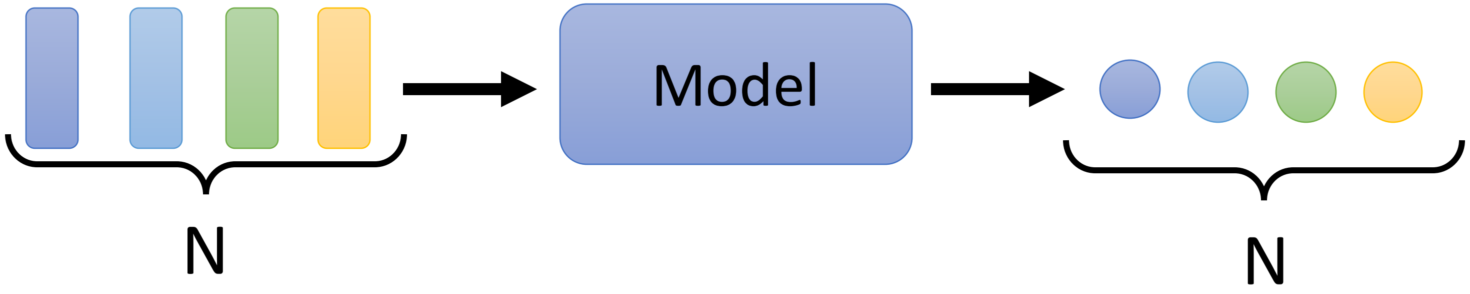
语音识别



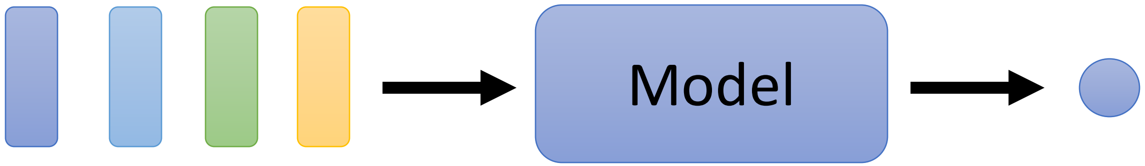
推荐系统

输出形式

- 一对一输出



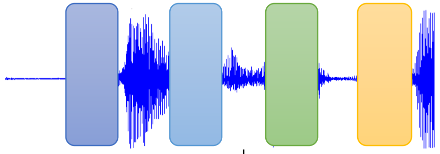
- 多对一输出



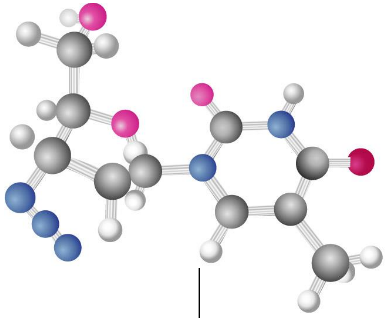
应用

this is good

positive
情感分析



speaker
声纹识别

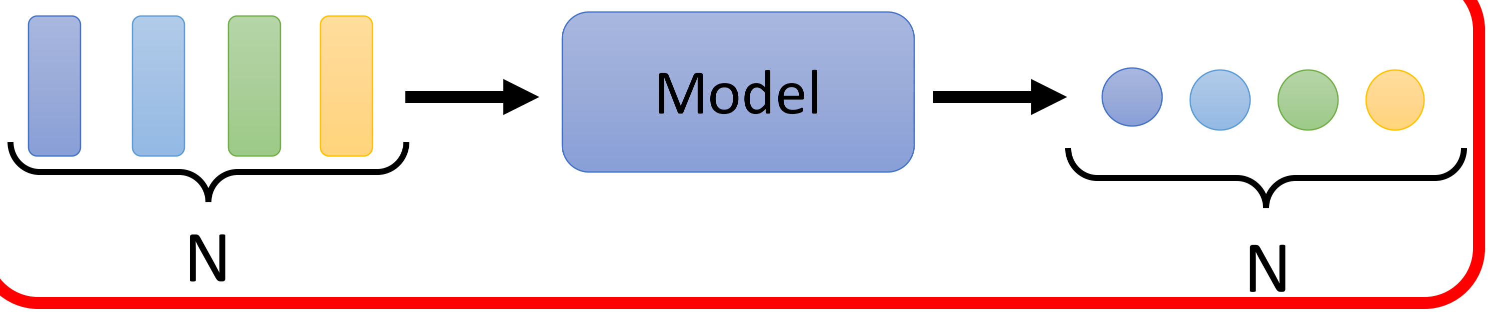


hydrophilicity
分子性质识别

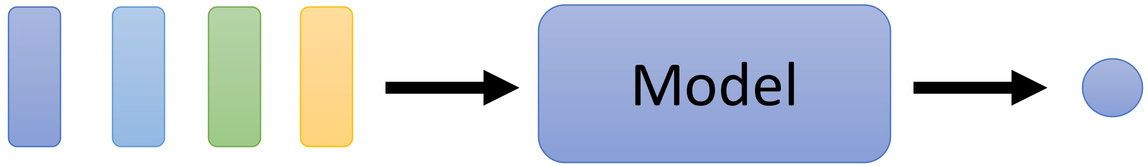
输出形式

- 一对一输出

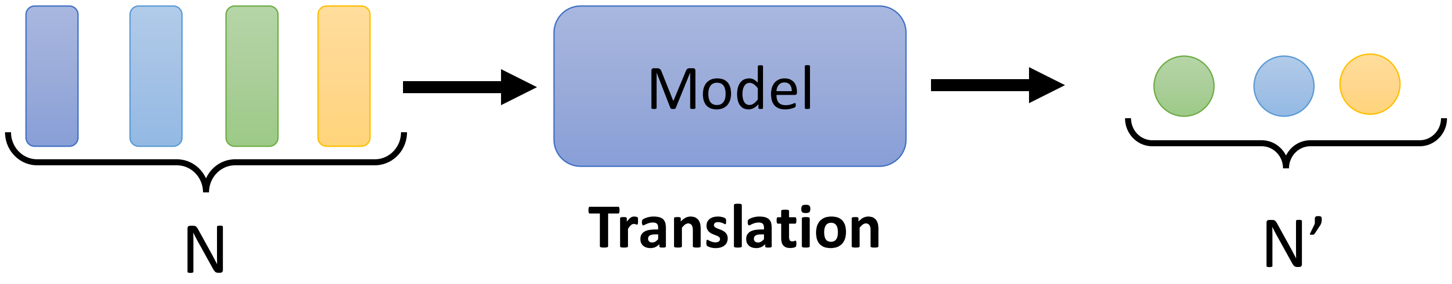
Focus on this first!



- 多对一输出



- 多对多输出 seq2seq



主观题 10分

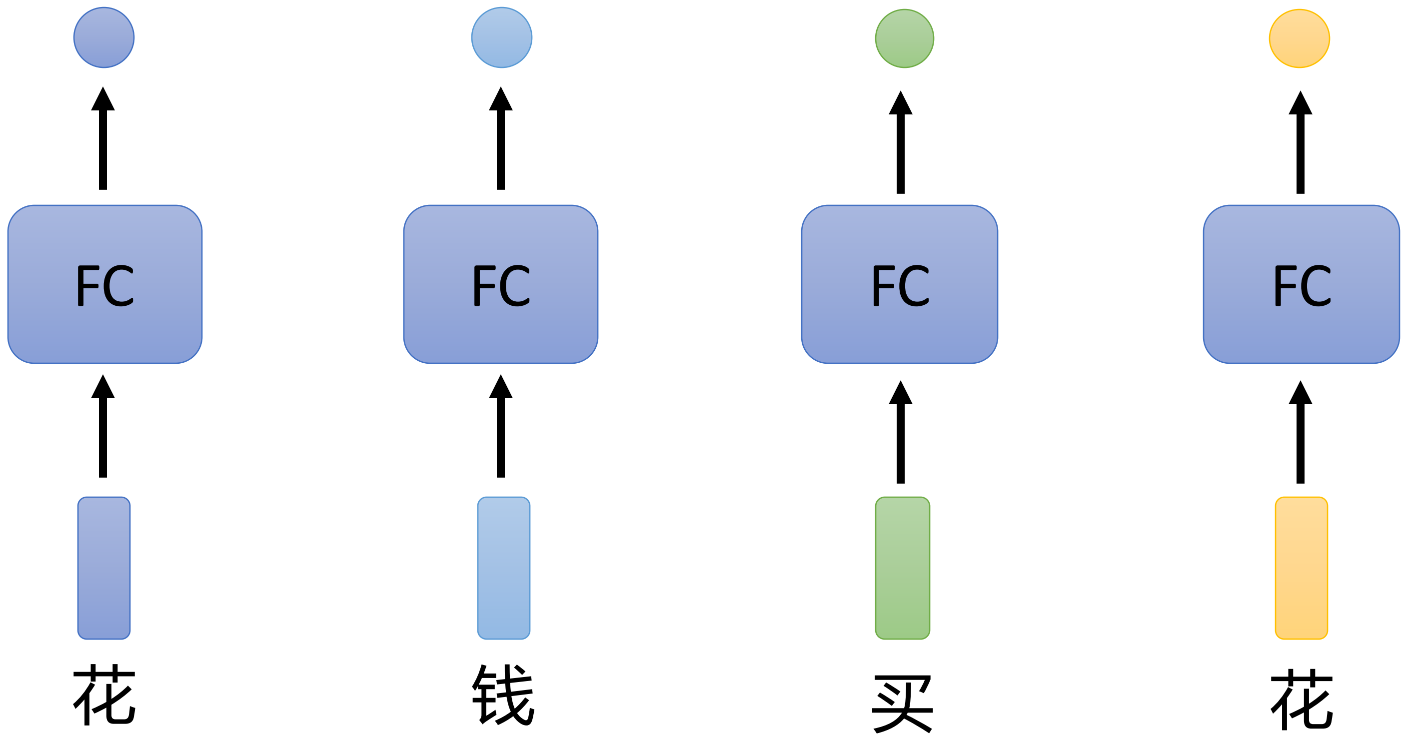
设置

请思考，给定向量集输入，如何建模上下文信息

作答

向量集输入

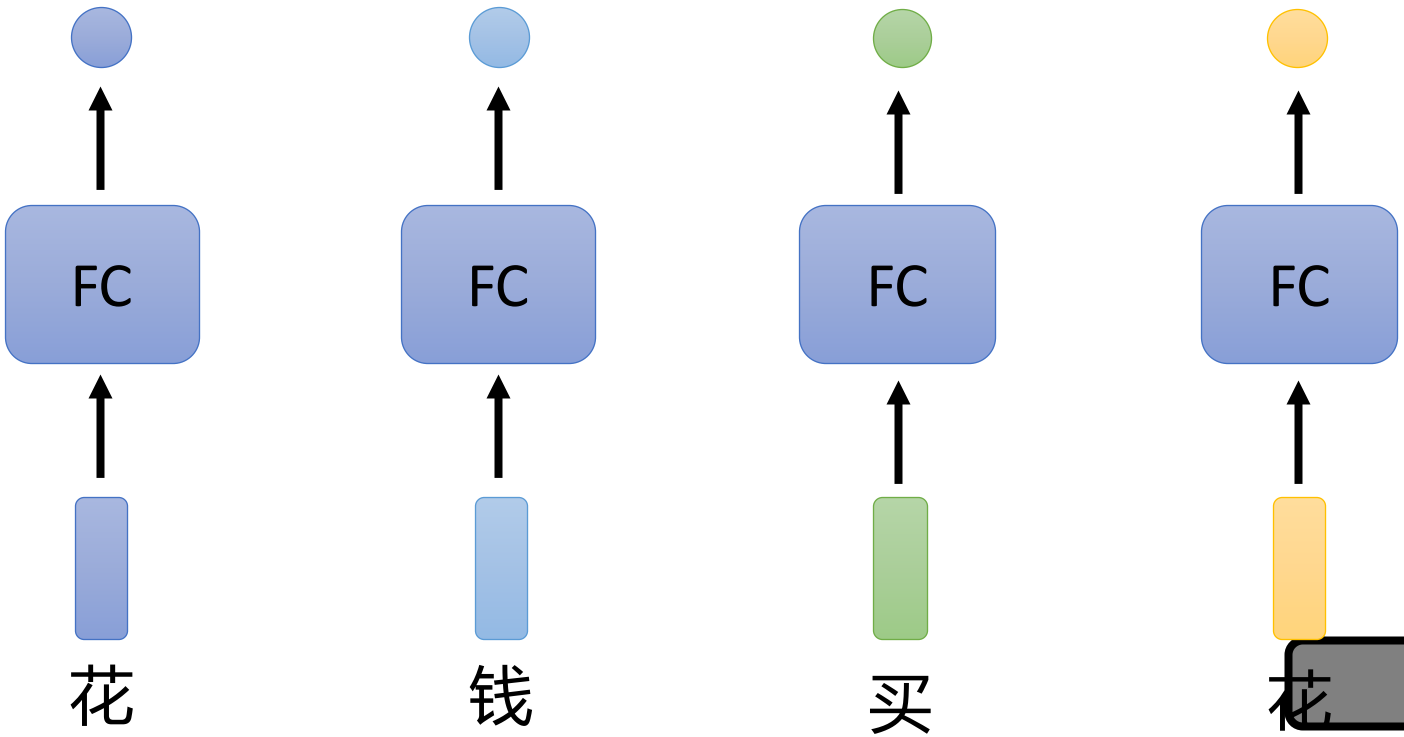
FC Fully-connected



设置

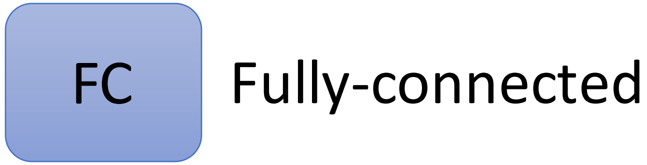
如何考虑上下文信息?

FC Fully-connected

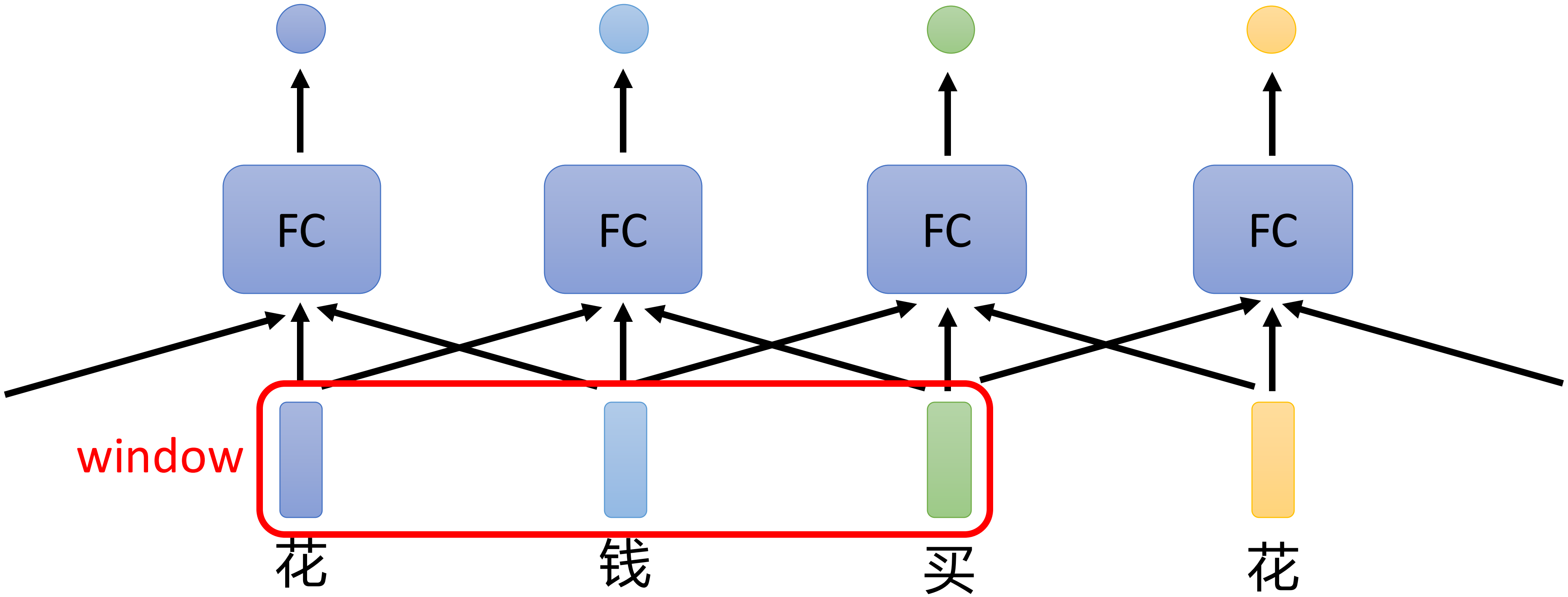


作答

向量集输入

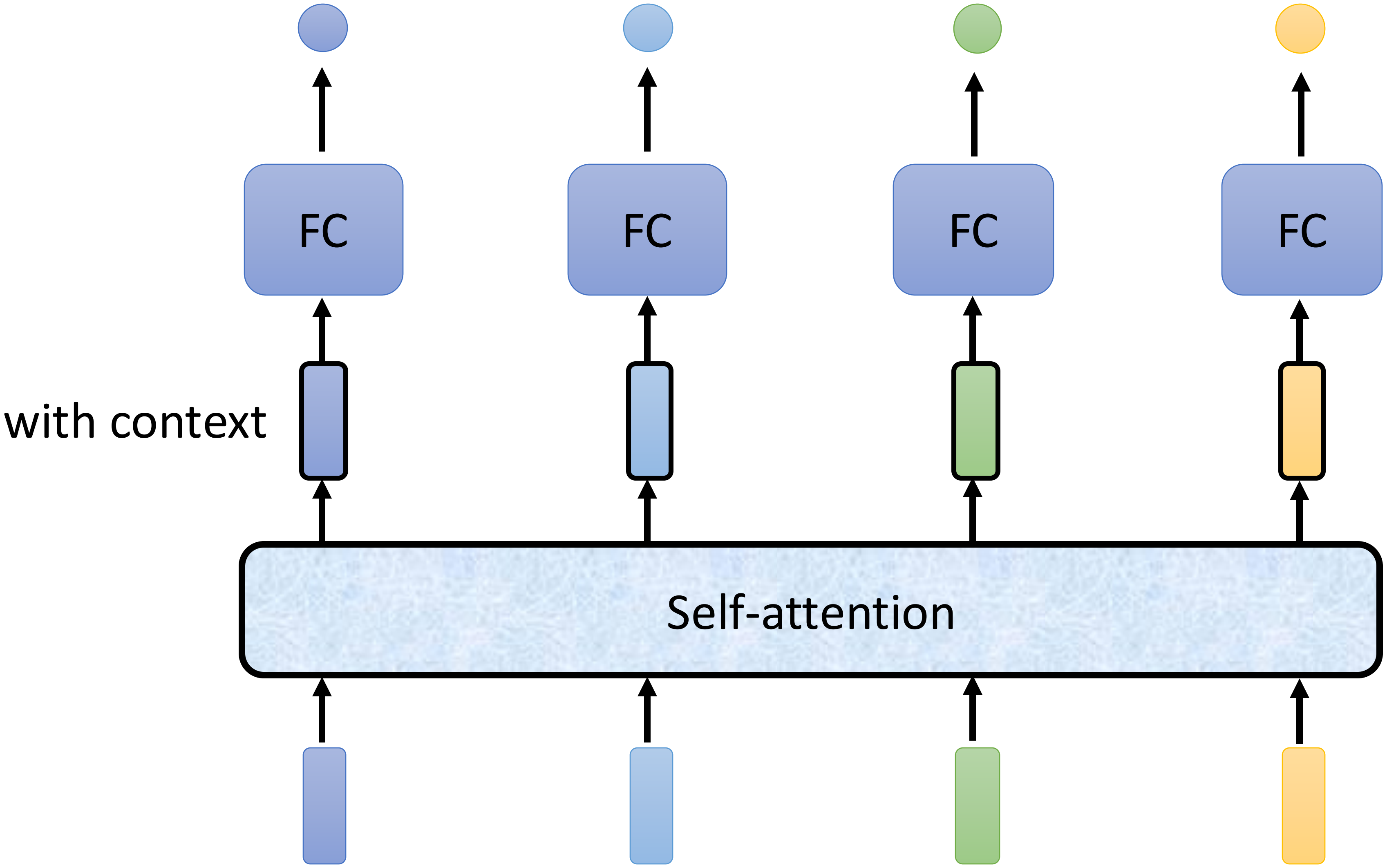


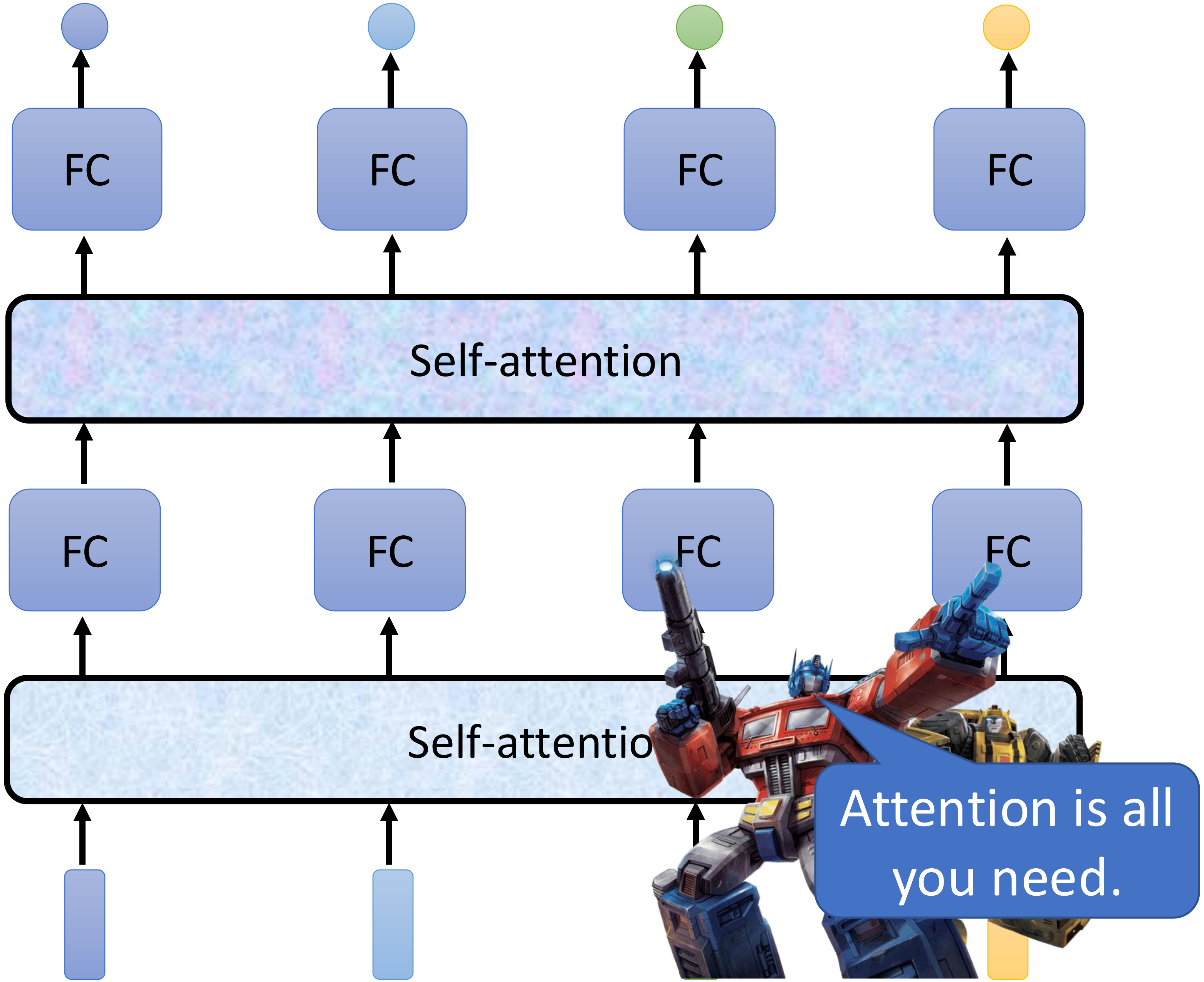
- 如何考虑上下文信息?
  - FC 可以考虑邻域信息
- 如何考虑整个序列的上下文信息?
  - 覆盖整个序列的窗口?



Self-attention
自注意力机制

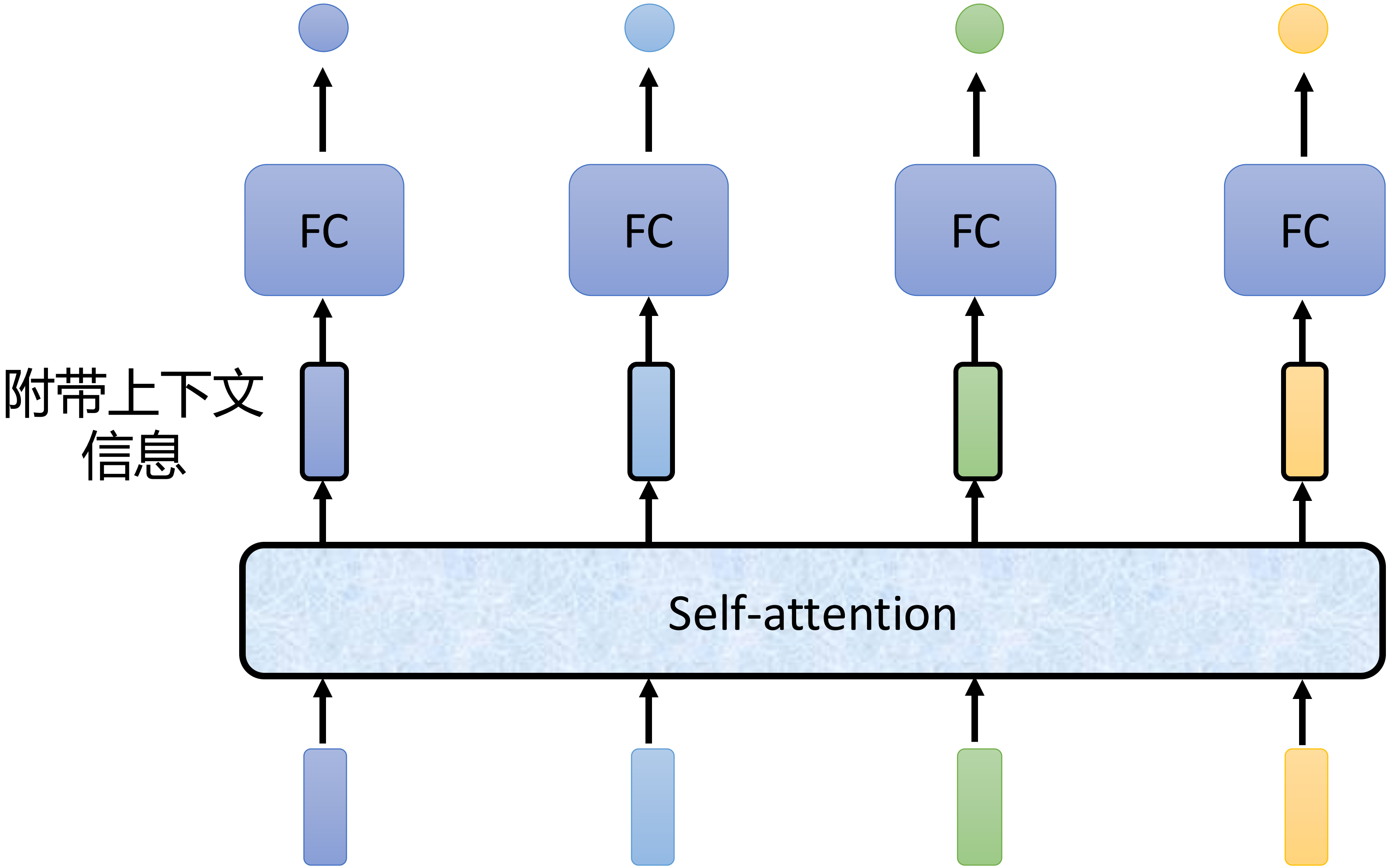
自注意力机制 *Self-attention*

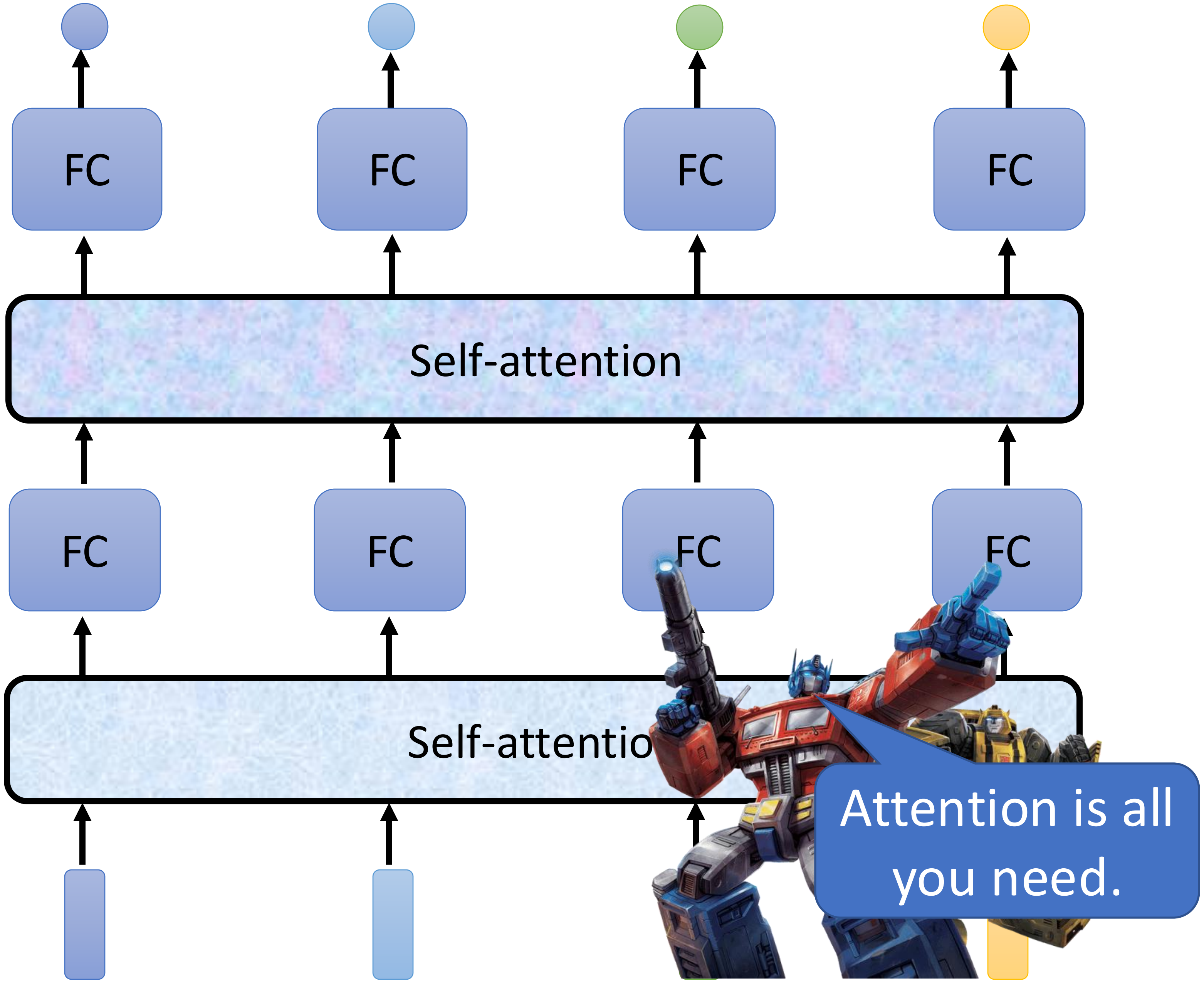




https://arxiv.org/abs/1706.03762

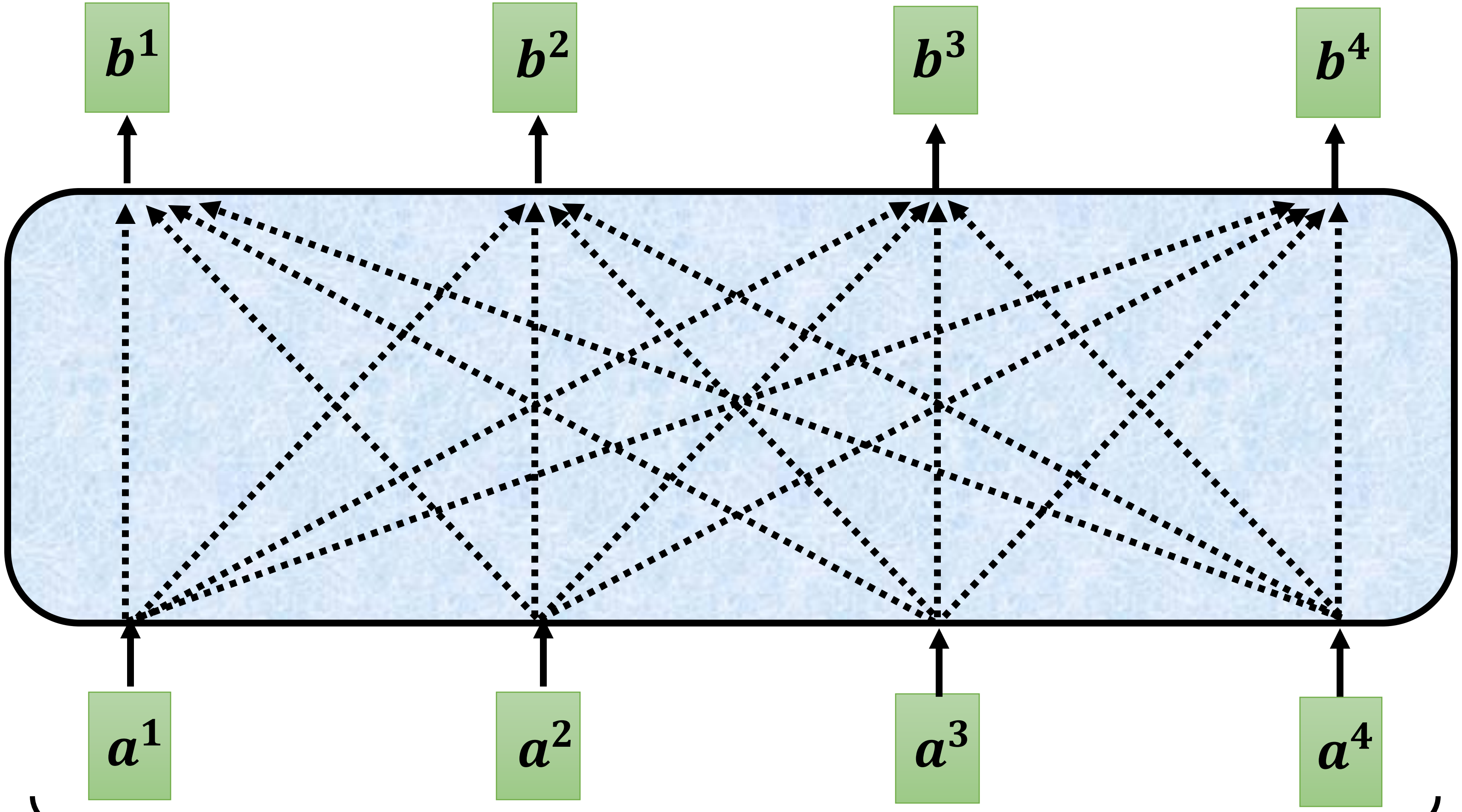
自注意力机制 *Self-attention*





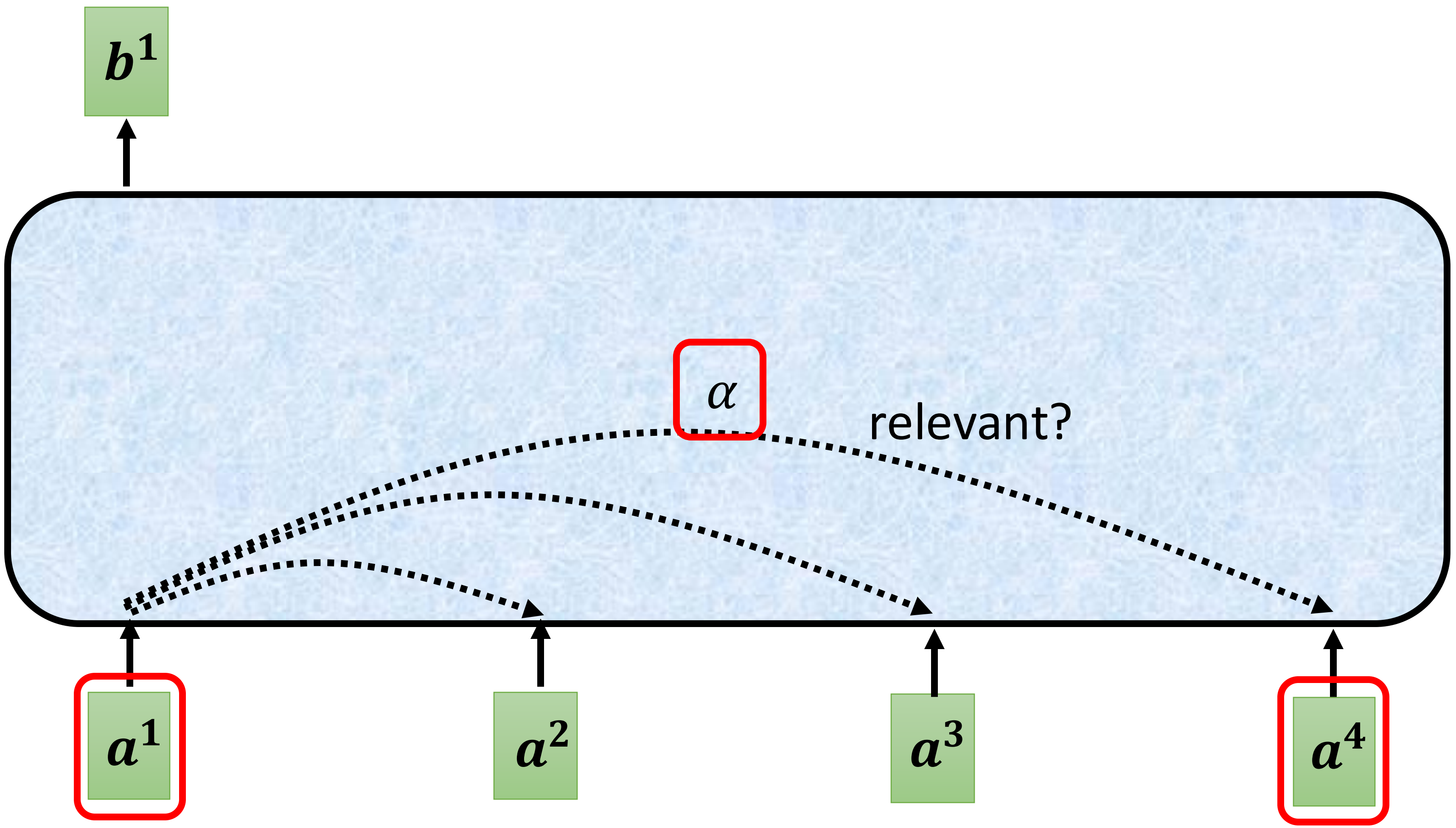
https://arxiv.org/abs/1706.03762

自注意力机制 *Self-attention*



输入特征或隐层特征

自注意力机制 *Self-attention*

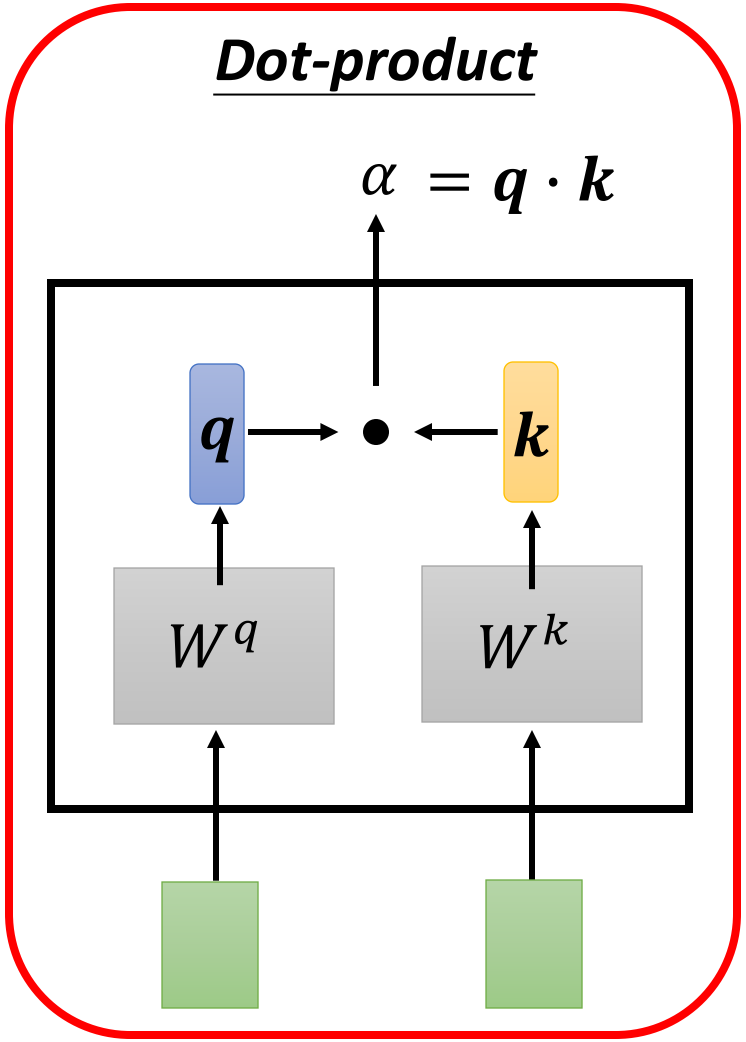


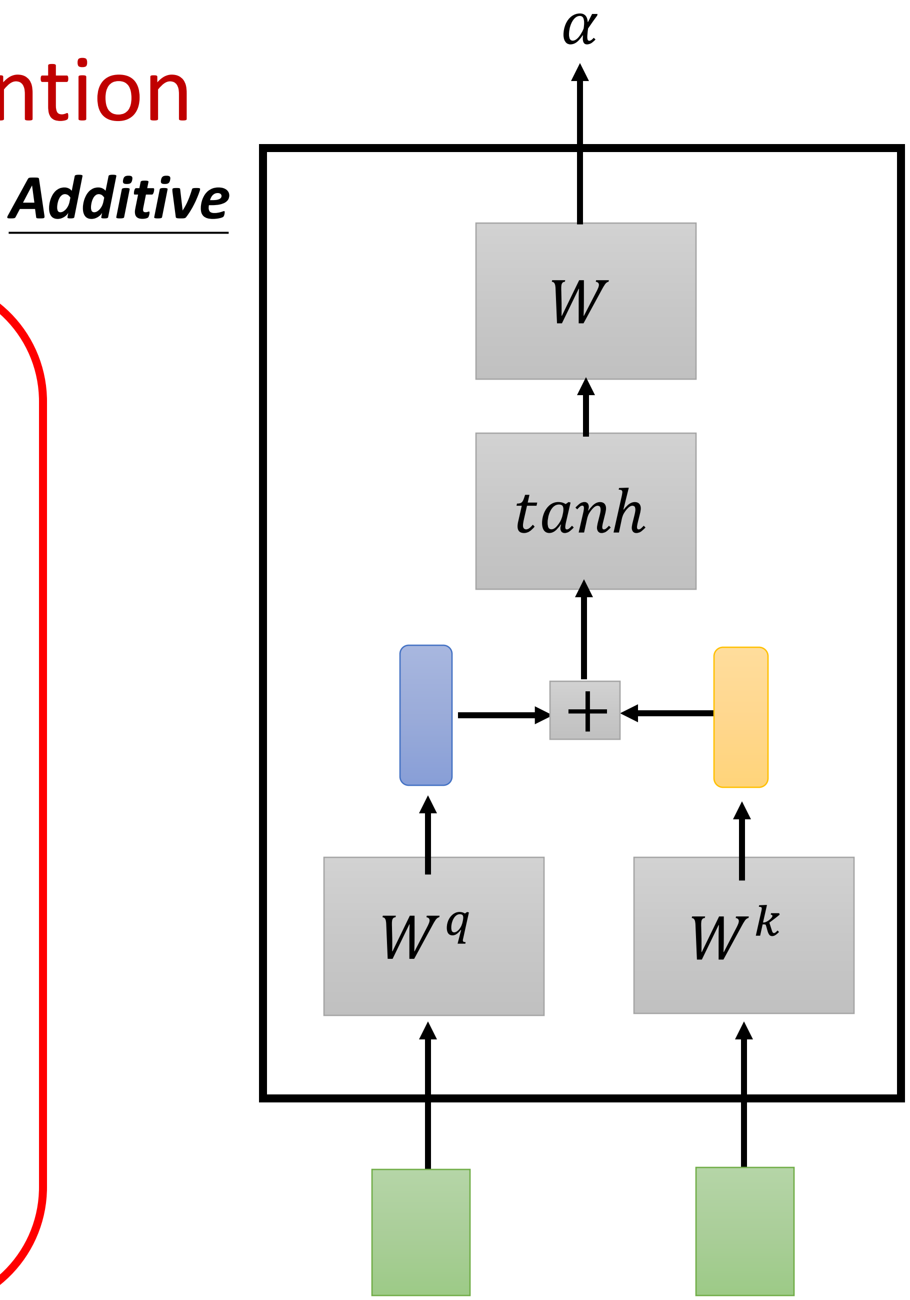
在序列中找到相关向量

引入Q、K、V

- 想象你在图书馆(输入序列)找书：
  - **Query(查询)**：就像你的借书需求("我想找一本关于深度学习的入门书")
  - **Key(键)**：就像每本书的索引标签(书名、分类标签等)
  - **Value(值)**：就是书籍本身的实际内容
- 具体过程
  - 匹配阶段：你的Query(需求)会与所有书的Key(标签)进行比较，计算匹配程度(注意力分数)
  - 加权求和：根据匹配程度，从Value(书籍内容)中提取信息
  - 得到结果：最终你得到的是各种书籍内容的加权组合，其中与你的需求最相关的书贡献最大

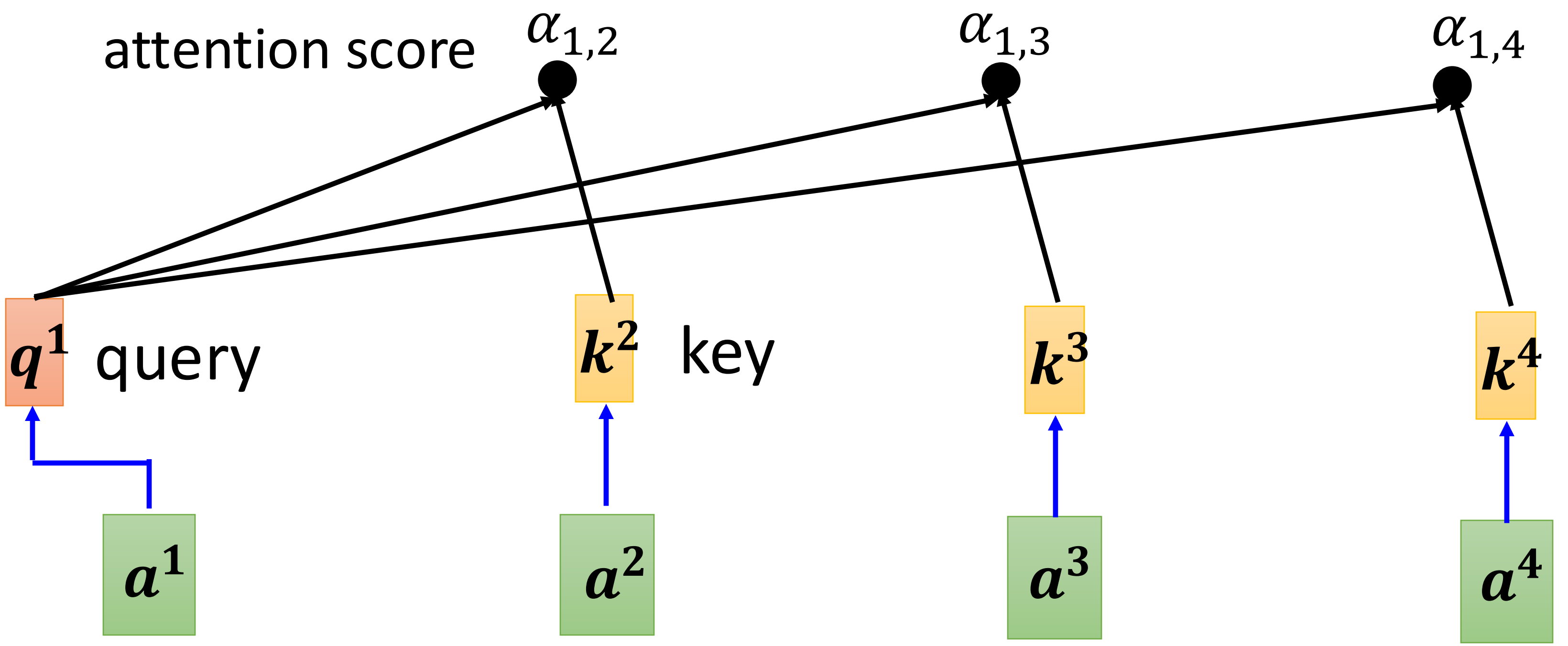
自注意力机制 Self-attention





自注意力机制 *Self-attention*

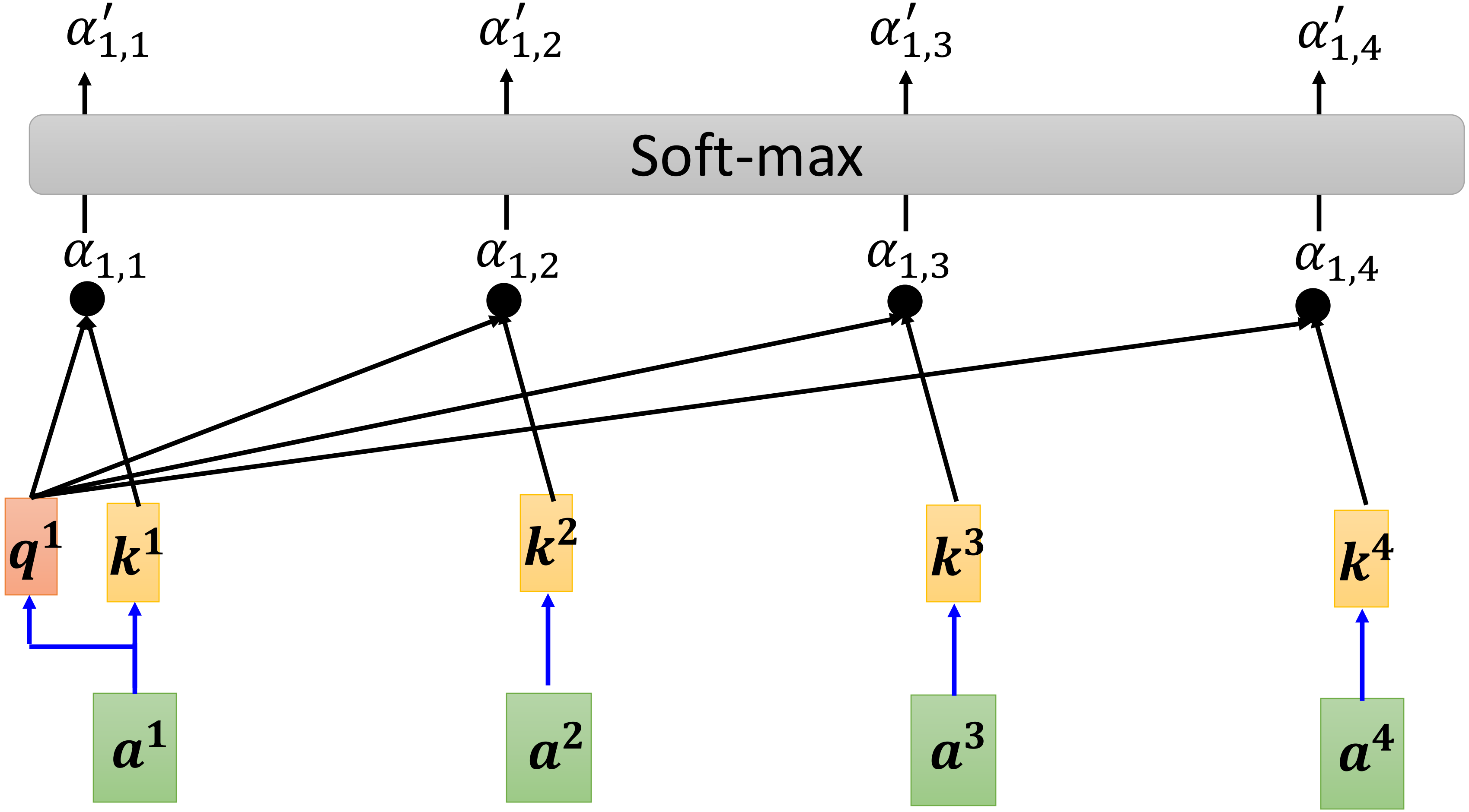
$\alpha_{1,2} = \boldsymbol{q}^1 \cdot \boldsymbol{k}^2 \quad \alpha_{1,3} = \boldsymbol{q}^1 \cdot \boldsymbol{k}^3 \quad \alpha_{1,4} = \boldsymbol{q}^1 \cdot \boldsymbol{k}^4$



$\boldsymbol{q}^1 = W^q \boldsymbol{a}^1 \quad \boldsymbol{k}^2 = W^k \boldsymbol{a}^2 \quad \boldsymbol{k}^3 = W^k \boldsymbol{a}^3 \quad \boldsymbol{k}^4 = W^k \boldsymbol{a}^4$

Self-attention

$$\alpha'_{1,i} = exp(\alpha_{1,i})/\sum_j exp(\alpha_{1,j})$$

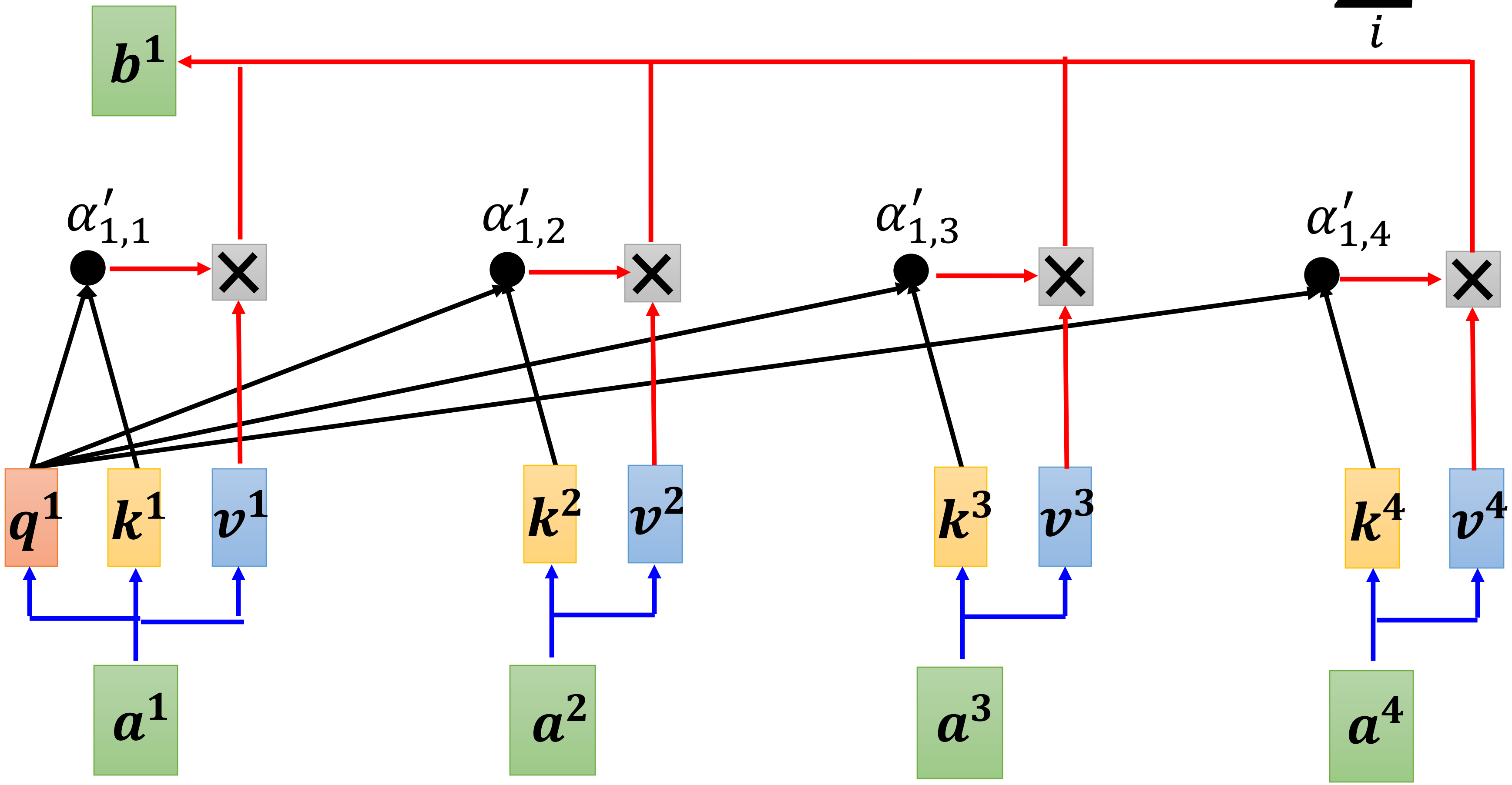


$q^1 = W^q a^1$ $\quad k^2 = W^k a^2$ $\quad k^3 = W^k a^3$ $\quad k^4 = W^k a^4$

$k^1 = W^k a^1$

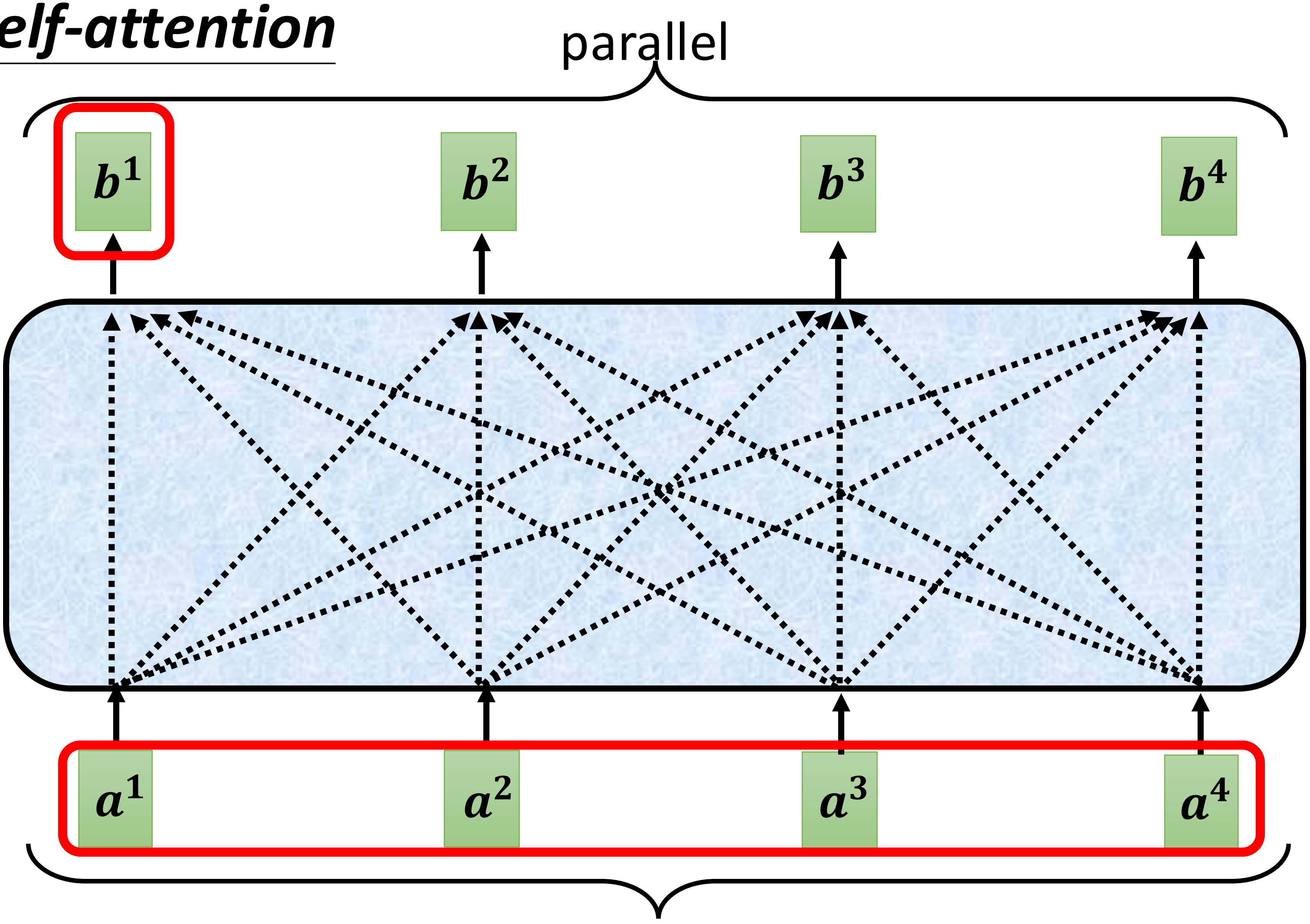
Self-attention

根据attention scores来抽取信息 $\boldsymbol{b}^1 = \sum_i \alpha'_{1,i} \boldsymbol{v}^i$



$\boldsymbol{v}^1 = W^v \boldsymbol{a}^1$ $\boldsymbol{v}^2 = W^v \boldsymbol{a}^2$ $\boldsymbol{v}^3 = W^v \boldsymbol{a}^3$ $\boldsymbol{v}^4 = W^v \boldsymbol{a}^4$

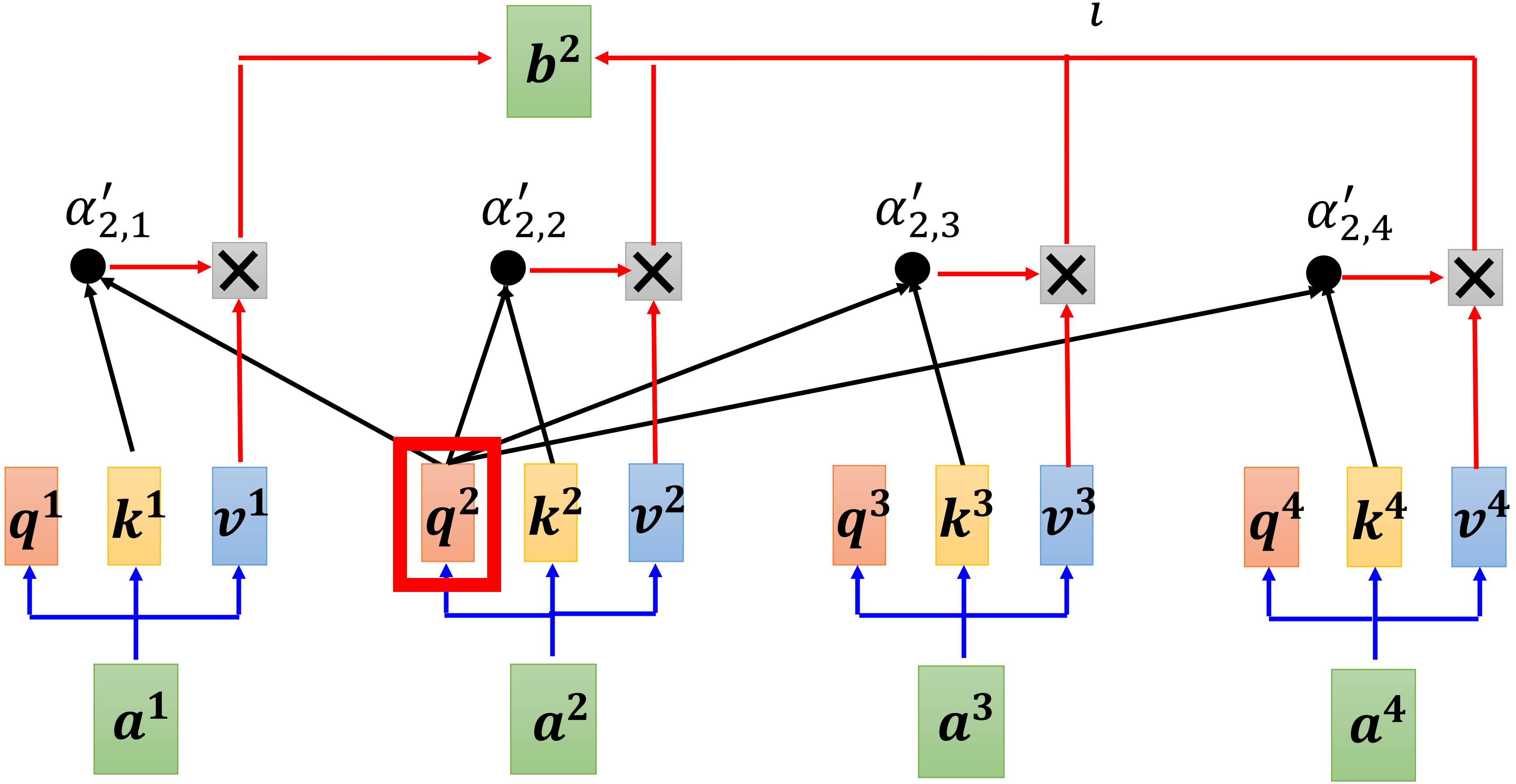
Self-attention



输入特征或隐层特征

Self-attention

$$b^2 = \sum_i \alpha'_{2,i} v^i$$

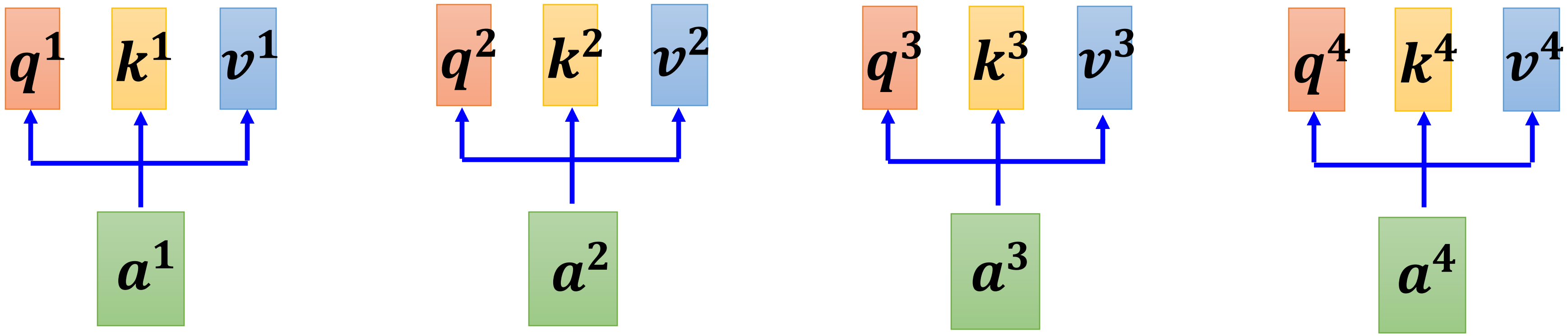


Self-attention

$$q^i = W^q a^i \qquad \underbrace{\begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix}}_{Q} = W^q \underbrace{\begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix}}_{I}$$

$$k^i = W^k a^i \qquad \underbrace{\begin{bmatrix} k^1 & k^2 & k^3 & k^4 \end{bmatrix}}_{K} = W^k \underbrace{\begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix}}_{I}$$

$$v^i = W^v a^i \qquad \underbrace{\begin{bmatrix} v^1 & v^2 & v^3 & v^4 \end{bmatrix}}_{V} = W^v \underbrace{\begin{bmatrix} a^1 & a^2 & a^3 & a^4 \end{bmatrix}}_{I}$$

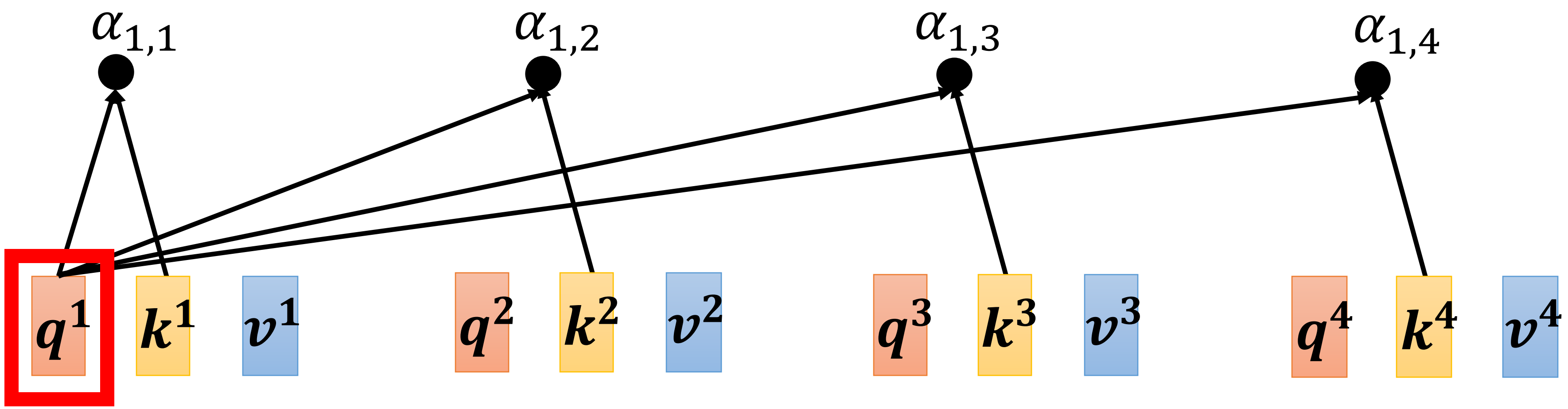


Self-attention

$\alpha_{1,1} = \boxed{k^1}\ \boxed{q^1}$ $\alpha_{1,2} = \boxed{k^2}\ \boxed{q^1}$

$\alpha_{1,3} = \boxed{k^3}\ \boxed{q^1}$ $\alpha_{1,4} = \boxed{k^4}\ \boxed{q^1}$

$$\begin{bmatrix} \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{bmatrix} = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} q^1$$

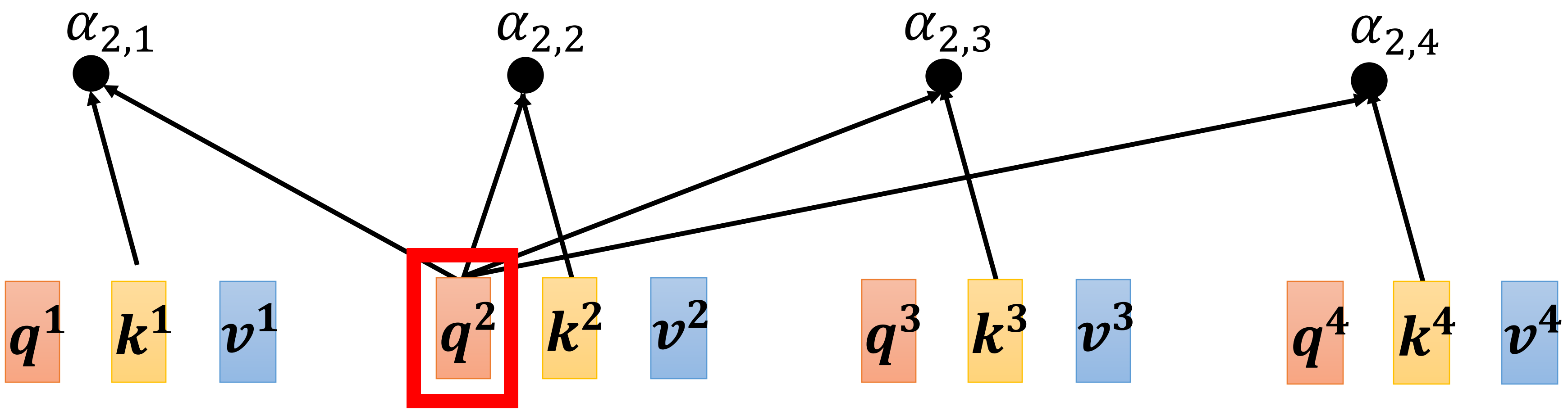


Self-attention

$\alpha_{1,1} = k^1 \, q^1 \qquad \alpha_{1,2} = k^2 \, q^1$

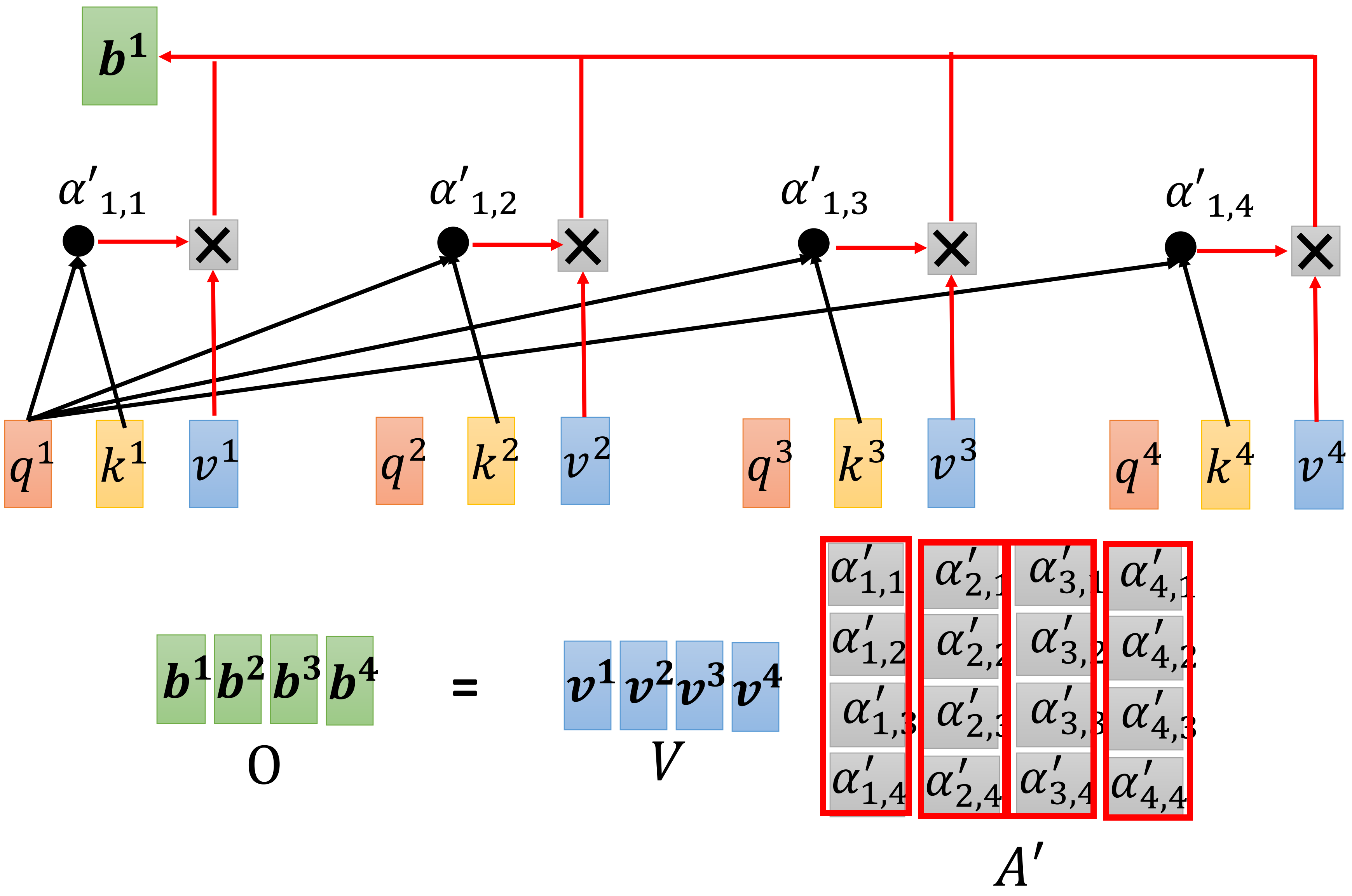
$\alpha_{1,3} = k^3 \, q^1 \qquad \alpha_{1,4} = k^4 \, q^1$

$$\begin{bmatrix} \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{bmatrix} = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} q^1$$



$$\underset{A'}{\begin{bmatrix} \alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1} \\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2} \\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3} \\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4} \end{bmatrix}} \Leftarrow \underset{A}{\begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix}} = \underset{K^T}{\begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix}} \underset{Q}{\begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix}}$$

Self-attention



Self-attention

$$Q = W^q \, I$$

$$K = W^k \, I$$

$$V = W^v \, I$$

Parameters to be learned

$$A' \leftarrow A = K^T \, Q$$

Attention Matrix

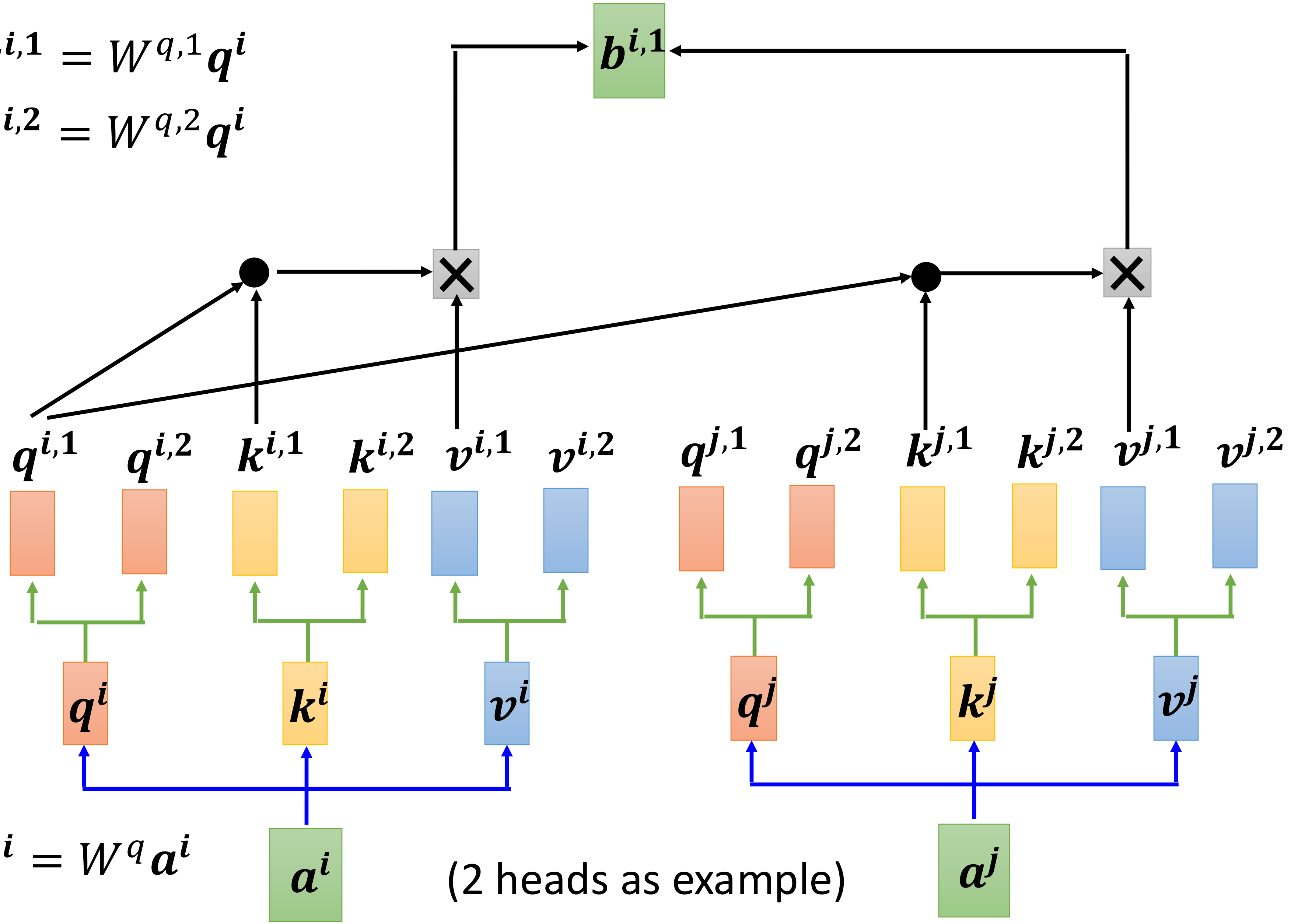
$$O = V \, A'$$

Multi-head Self-attention

Different types of relevance

$$q^{i,1} = W^{q,1} q^i$$

$$q^{i,2} = W^{q,2} q^i$$



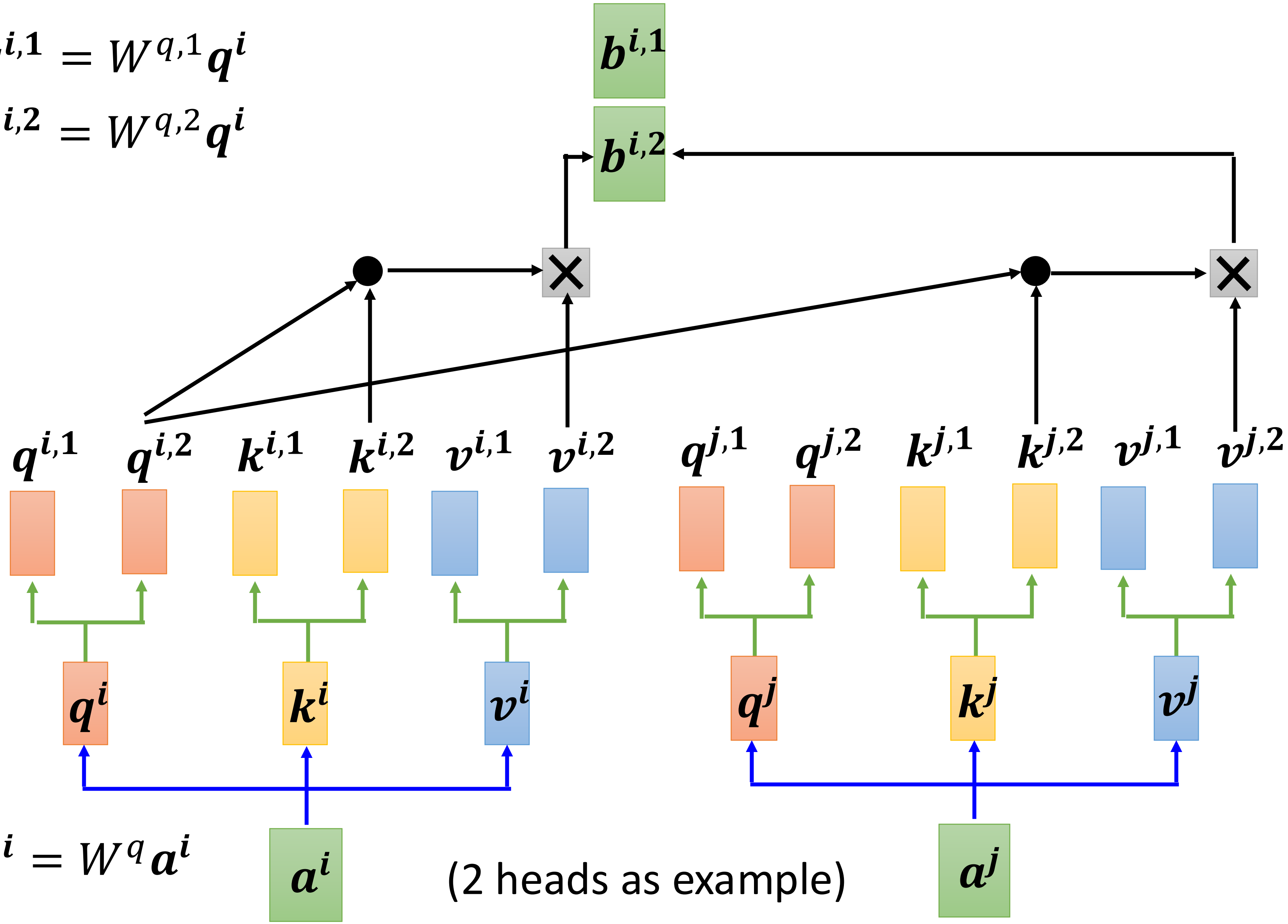
$$q^i = W^q a^i$$

(2 heads as example)

Multi-head Self-attention Different types of relevance

$$q^{i,1} = W^{q,1} q^i$$

$$q^{i,2} = W^{q,2} q^i$$



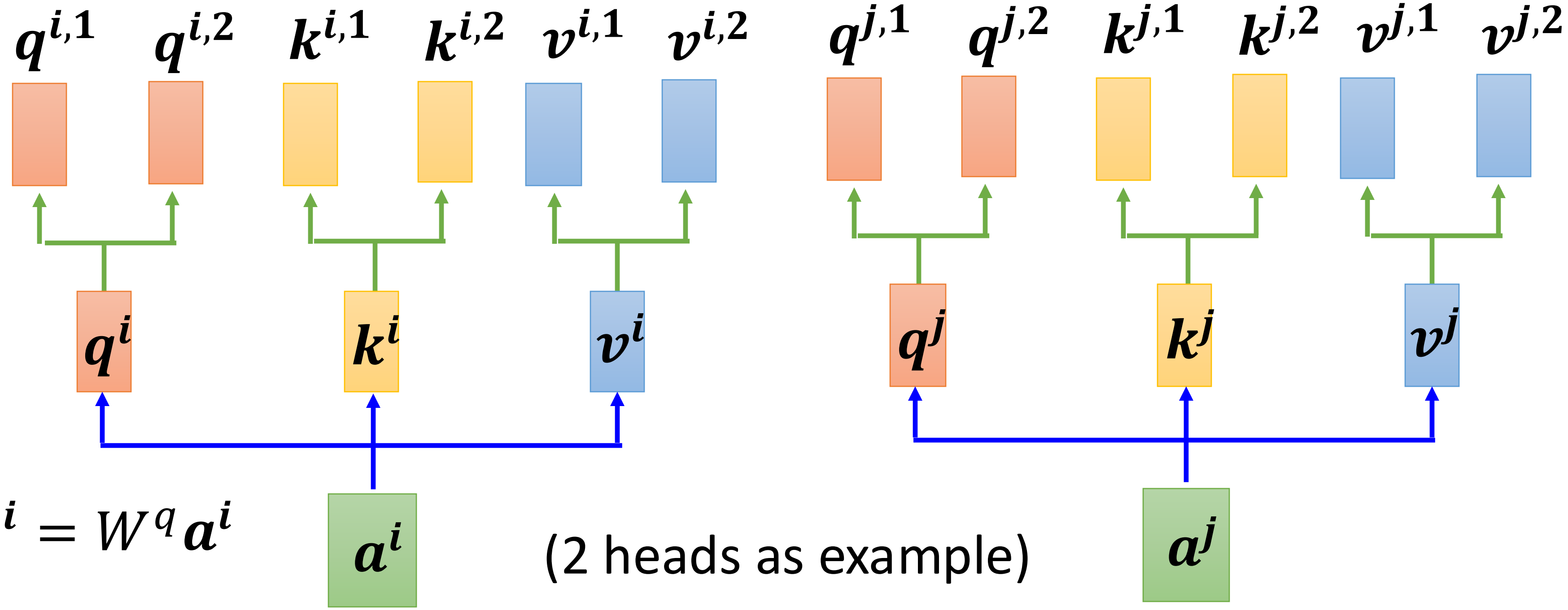
$$q^i = W^q a^i$$

(2 heads as example)

Multi-head Self-attention

Different types of relevance

$$\boldsymbol{b}^i = W^O \begin{bmatrix} \boldsymbol{b}^{i,1} \\ \boldsymbol{b}^{i,2} \end{bmatrix}$$



$$\boldsymbol{q}^i = W^q \boldsymbol{a}^i$$

(2 heads as example)

主观题 10分

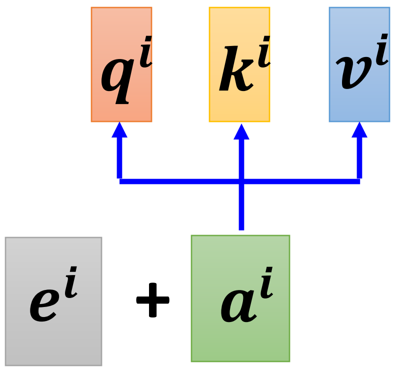
设置

请思考，自注意力机制能否建模序列数据的顺序/位置信息?

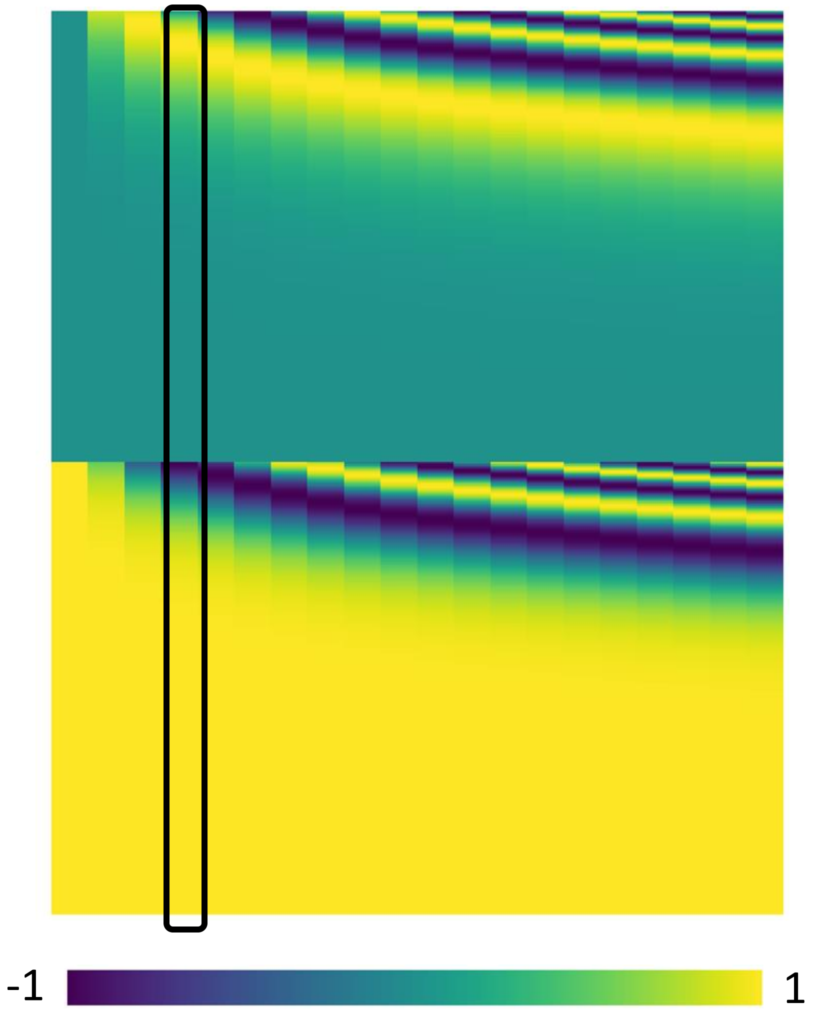
作答

位置编码 positional embedding

- self-attention中缺失了位置信息.
- 可以给每个位置赋予一个唯一的位置编码向量（positional vector） e^i
- 位置编码可以是手工设计的 hand-crafted
- 也可以是从数据中学习的 learned from data



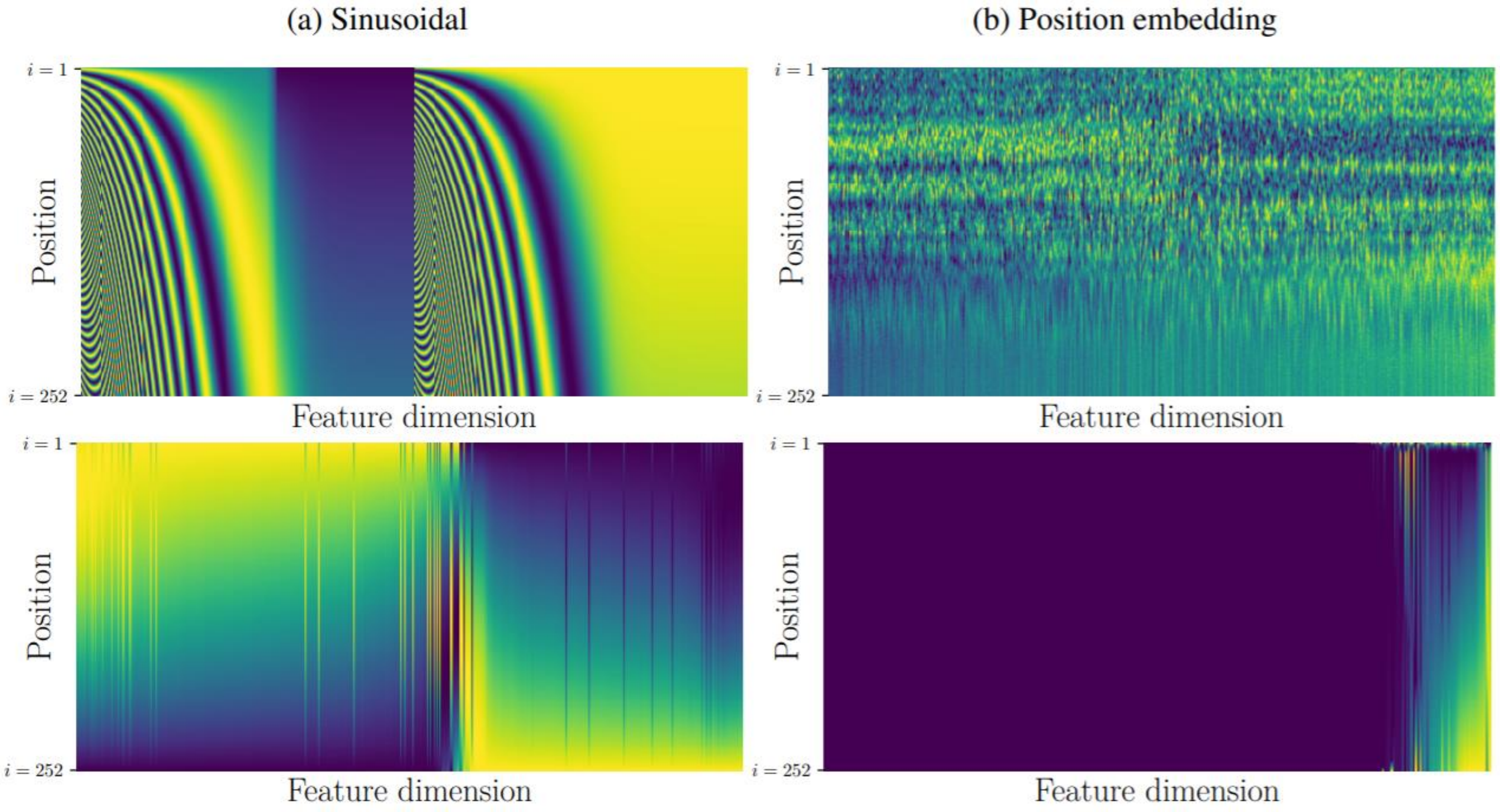
Each column represents a positional vector e^i



https://arxiv.org/abs/2003.09229

Table 1. Comparing position representation methods

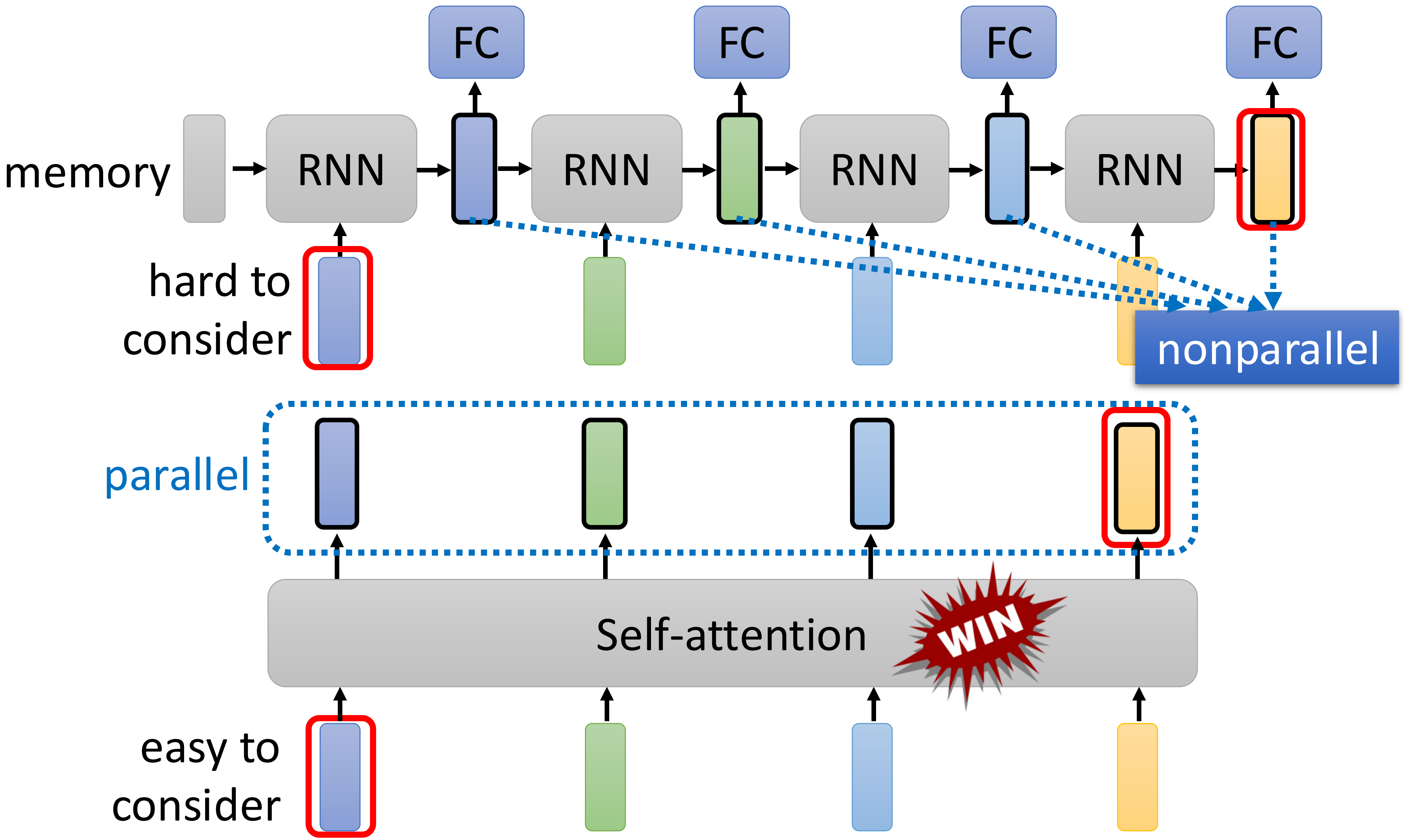
Methods	Inductive	Data-Driven	Parameter Efficient
Sinusoidal (Vaswani et al., 2017)	✓	✗	✓
Embedding (Devlin et al., 2018)	✗	✓	✗
Relative (Shaw et al., 2018)	✗	✓	✓
This paper	✓	✓	✓



(c) FLOATER

(d) RNN

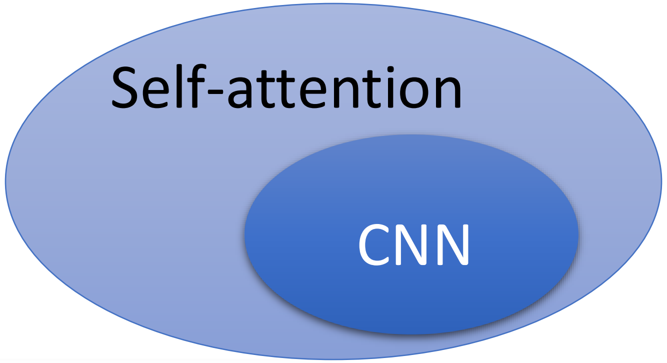
Self-attention v.s. RNN

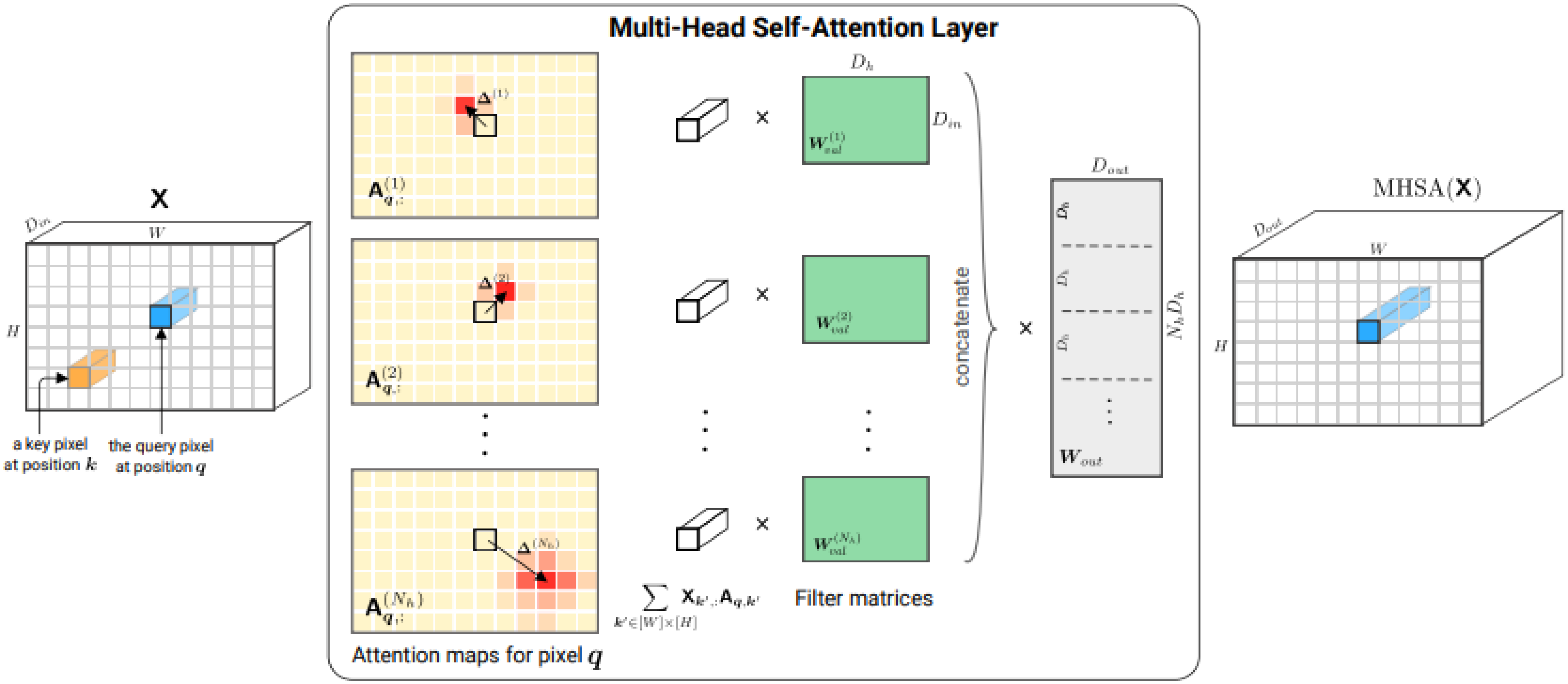


Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

https://arxiv.org/abs/2006.16236

Self-attention v.s. CNN





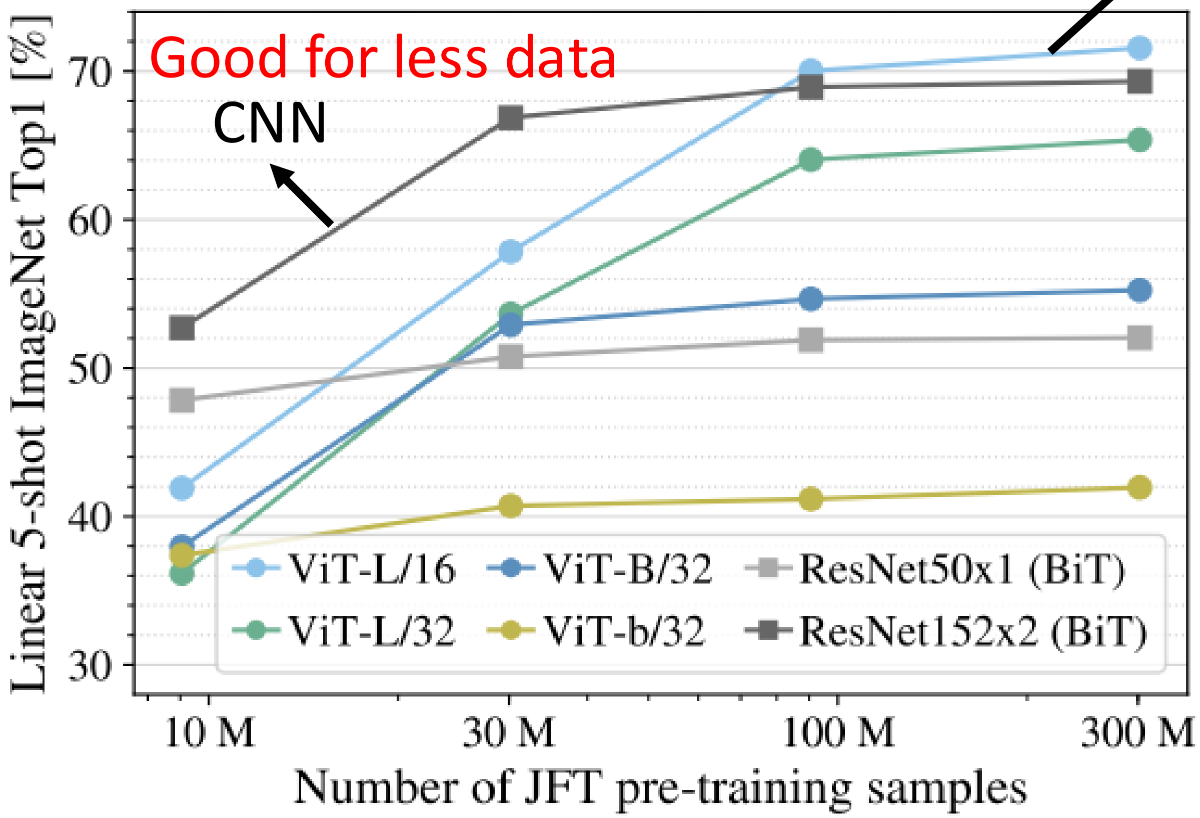
On the Relationship between Self-Attention and Convolutional Layers

https://arxiv.org/abs/1911.03584

Self-attention v.s. CNN

Good for more data

Self-attention

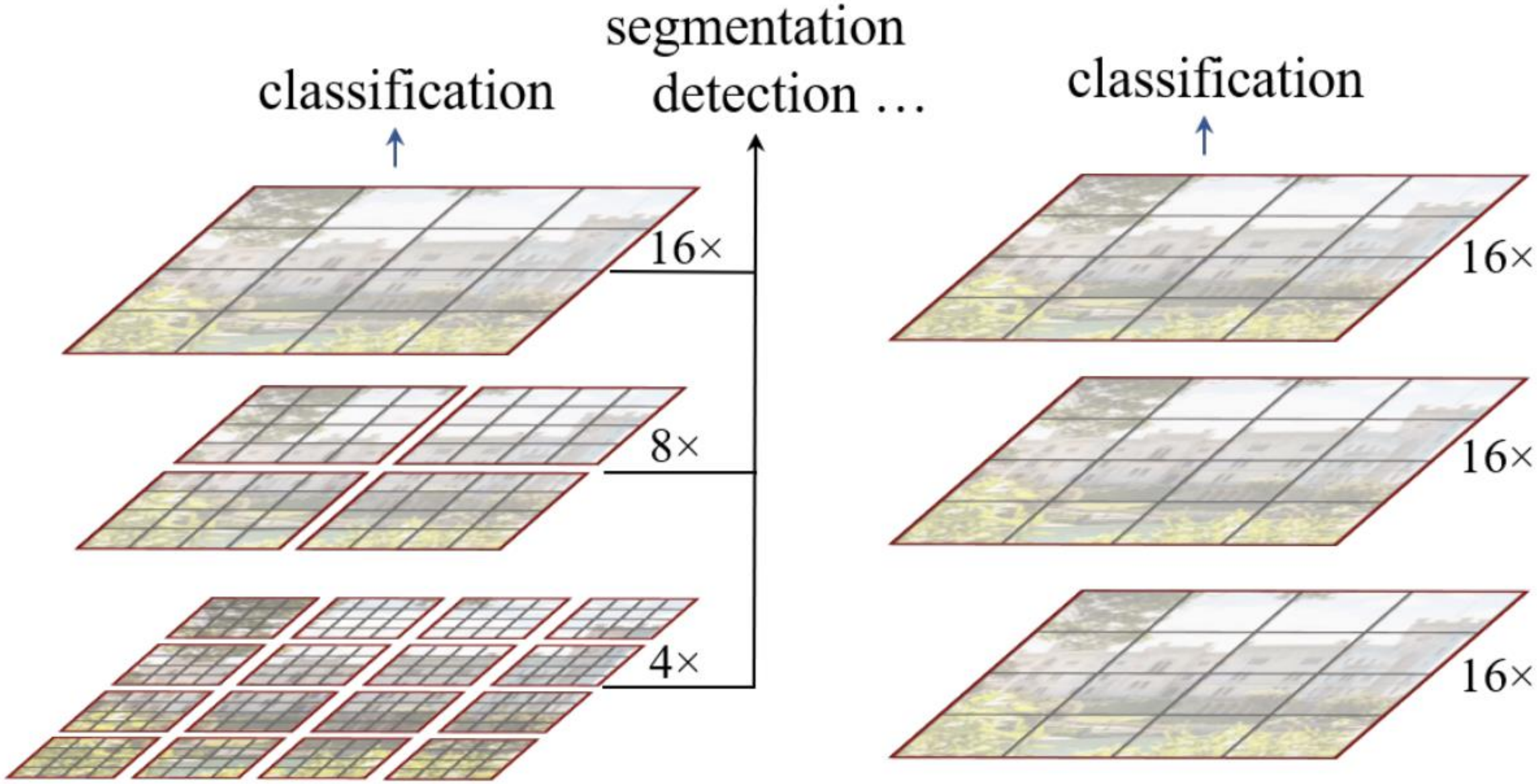


An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

https://arxiv.org/pdf/2010.11929.pdf

Self-attention v.s. CNN

https://arxiv.org/pdf/2103.14030.pdf



(a) Swin Transformer (ours)

(b) ViT

提纲

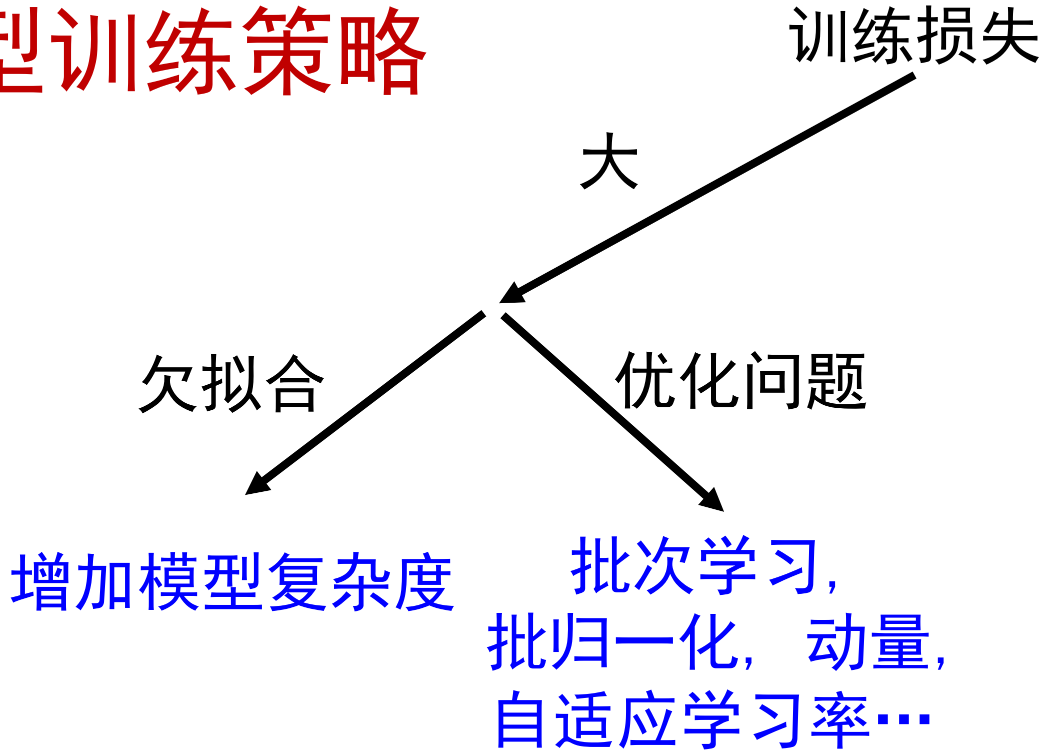
深度学习应用

- 深度学习实践
- 卷积神经网络应用
- 自注意力机制应用

深度学习应用

- 深度学习实践
  - 模型训练策略
- 卷积神经网络应用
- 自注意力机制应用

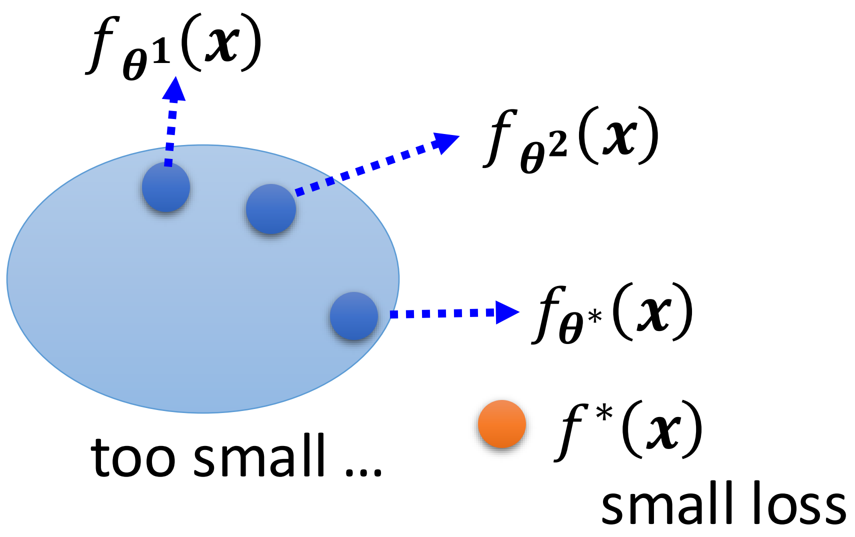
模型训练策略



欠拟合

find a needle in a haystack …

… but there is no needle

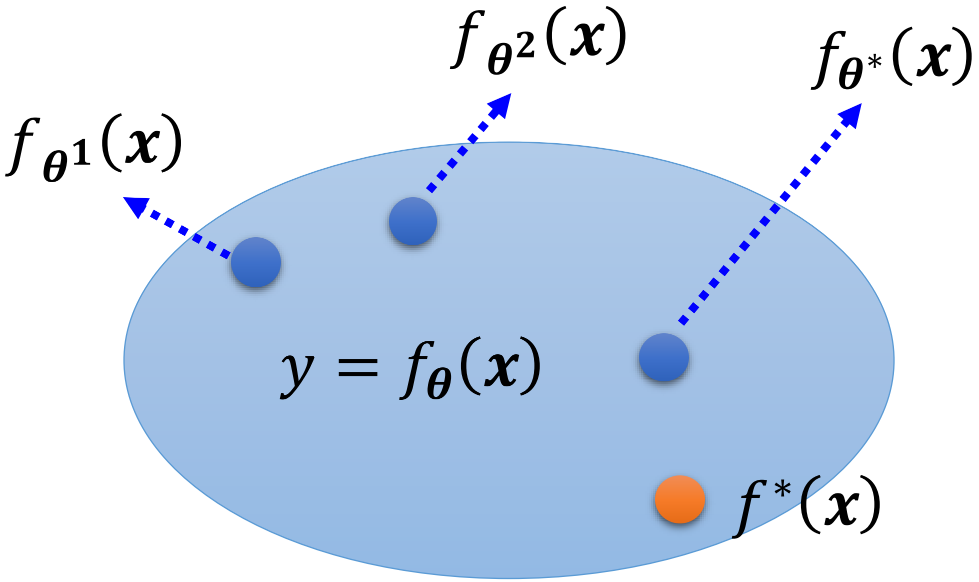


Which one???

优化问题

A needle is in a haystack …

… Just cannot find it.



主观题 10分

设置

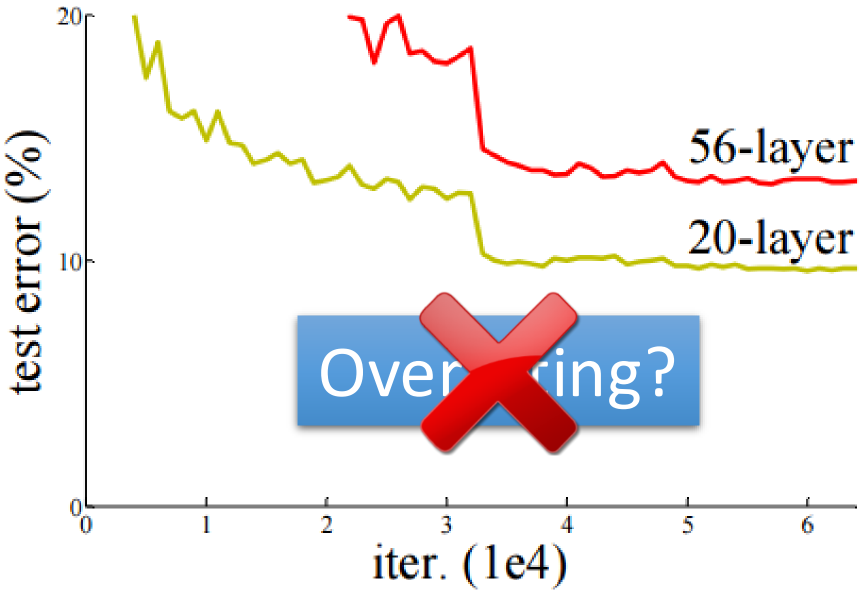
请思考，有哪些方式可以判断模型出现的是欠拟合还是优化问题?

作答

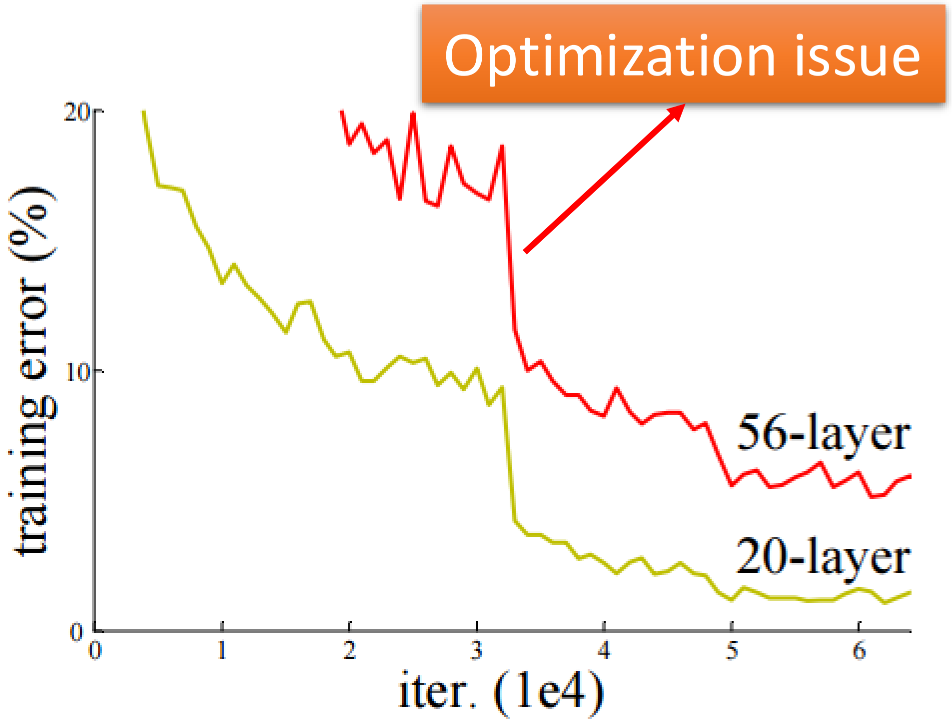
Ref: http://arxiv.org/abs/1512.03385

欠拟合 v.s. 优化问题

- 实验对比分析



Testing Data



Training Data

Ref: http://arxiv.org/abs/1512.03385

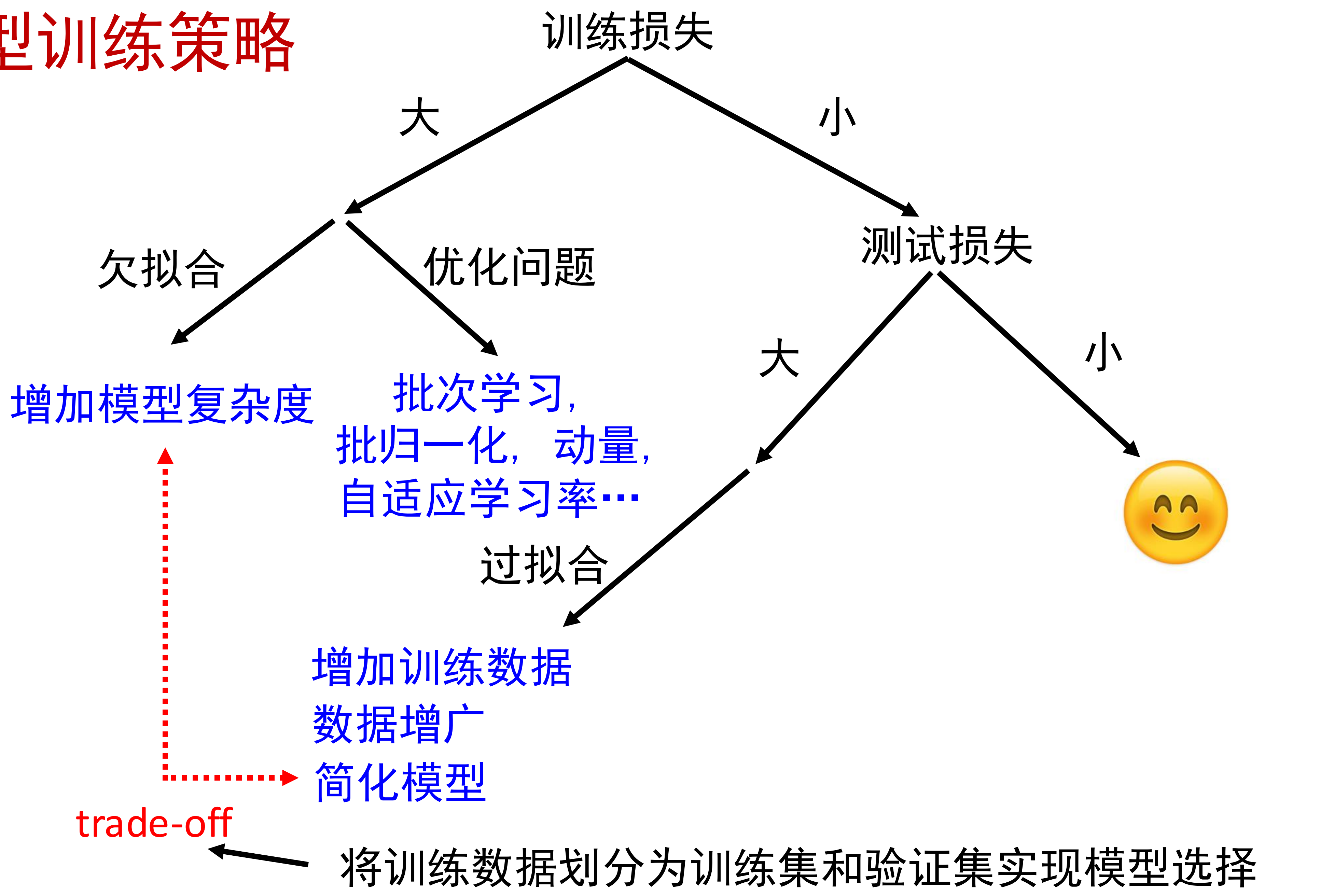
优化问题

- 实验对比分析
- 从易优化的浅层网络（或传统方法）开始尝试
- 如果加深网络无法得到更小的训练损失，那么可以认为是优化问题

	1 layer	2 layer	3 layer	4 layer	5 layer
2017 – 2020	0.28k	0.18k	0.14k	0.10k	0.34k

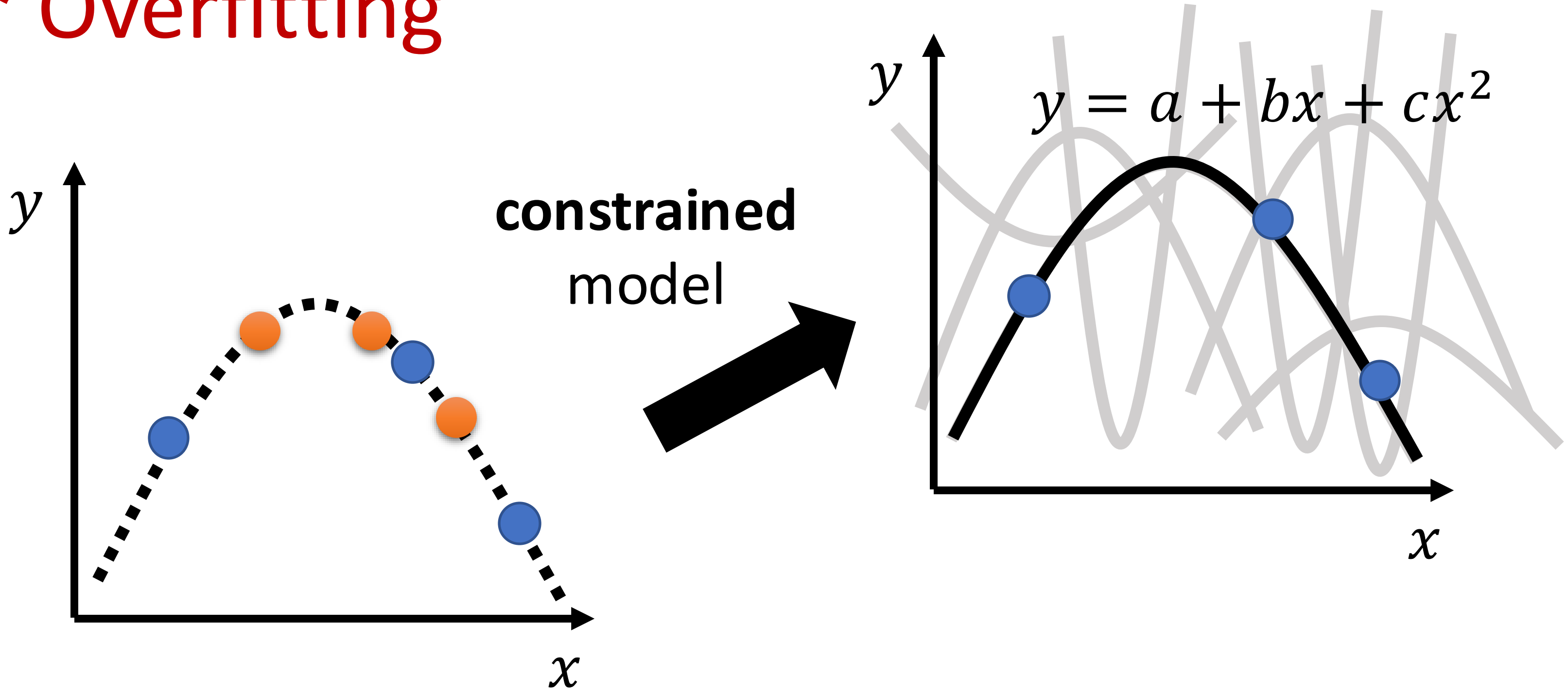
- 解决方案: 使用更好的优化方法

模型训练策略

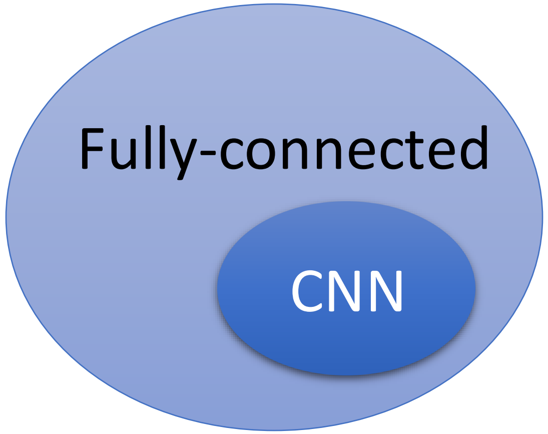


训练损失
大
小
欠拟合
优化问题
测试损失
大
小
增加模型复杂度
批次学习，
批归一化，动量，
自适应学习率…
过拟合
增加训练数据
数据增广
简化模型
trade-off
将训练数据划分为训练集和验证集实现模型选择

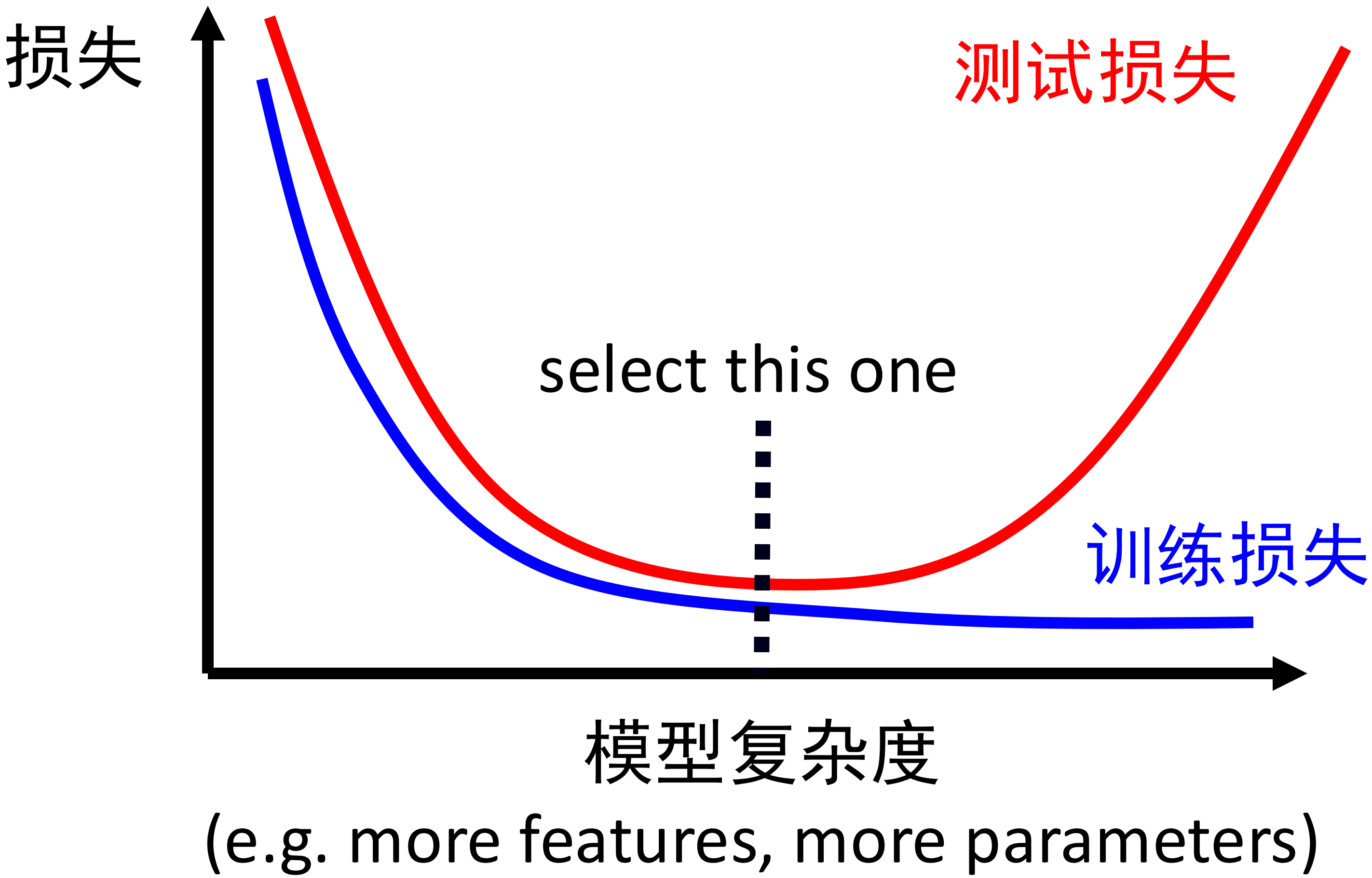
过拟合 Overfitting



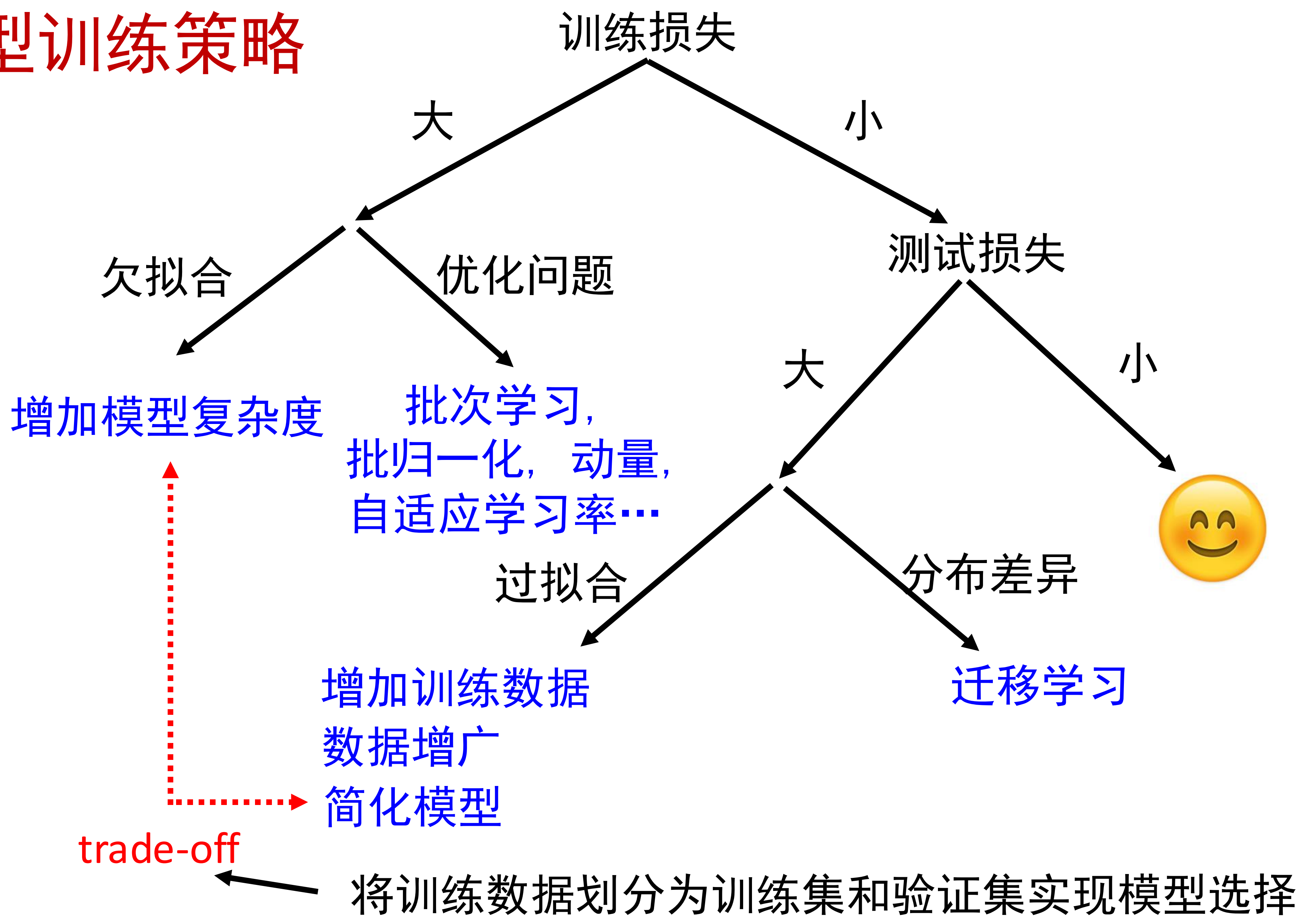
- Less parameters, sharing parameters
- Less features
- Early stopping
- Regularization
- Dropout



Bias-Complexity Trade-off

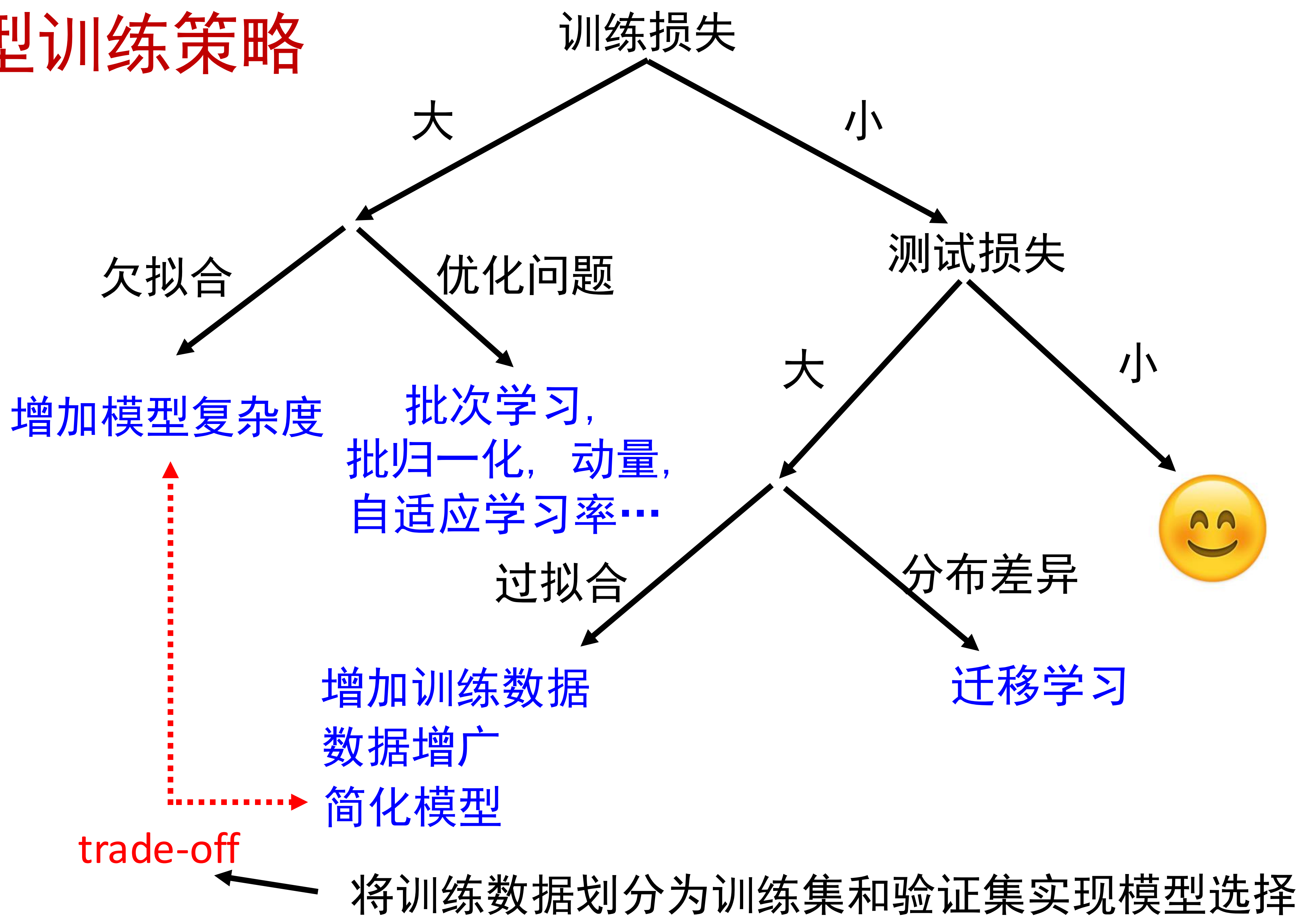


模型训练策略



训练损失
大
小
欠拟合
优化问题
测试损失
大
小
增加模型复杂度
批次学习，
批归一化，动量，
自适应学习率…
过拟合
分布差异
增加训练数据
数据增广
简化模型
迁移学习
trade-off
将训练数据划分为训练集和验证集实现模型选择

模型训练策略


训练损失
大
小
欠拟合
优化问题
测试损失
增加模型复杂度
批次学习，
批归一化， 动量，
自适应学习率…
大
小
过拟合
分布差异
增加训练数据
数据增广
简化模型
迁移学习
trade-off
将训练数据划分为训练集和验证集实现模型选择