



LiVE Group
视觉智能与学习研究中心

机器学习 (第10讲)

主讲：张磊

E-mail: leizhang@cqu.edu.cn
Lab Website: <http://www.leizhang.tk>





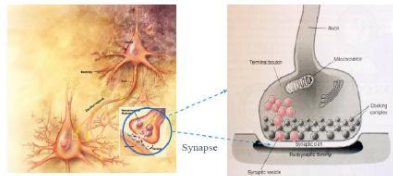
第十章：神经网络与深度学习

第十章：神经网络与深度学习

□ 神经网络发展史

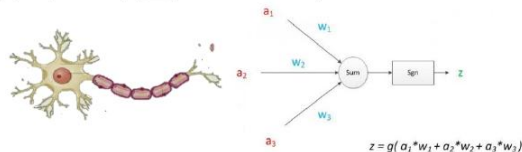
1906

- 神经元学说
- 神经元的连接方式被发现



神经元MP模型, McCulloch, Pitts

1943



Hebb

1949

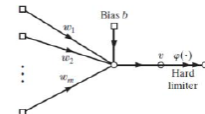
- Hebb学习规则
- $\Delta W_j = k \times f(W_j^T X) X$



1958

- 感知机(Perceptron)模型

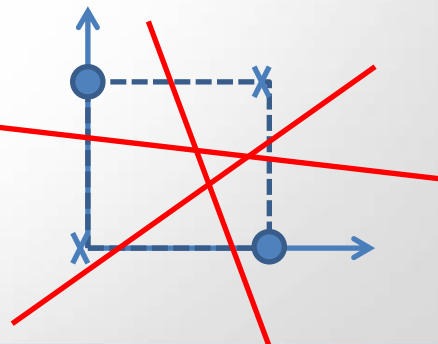
$$w = w + \eta(y - \hat{y})x$$



Marvin Minsky

1969

- XOR (异或) 问题
- 计算能力的限制



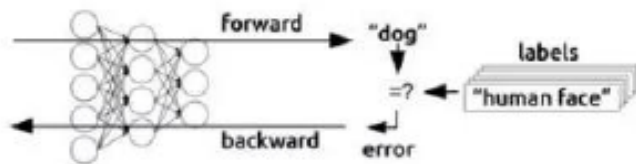
第十章：神经网络与深度学习

□ 神经网络发展史



G.E. Hinton

1986 BP神经网络, Rumelhart, Williams, Hinton



Vladimir Vapnik

1995 ■ 支持向量机(SVM), Vapnik



G.E. Hinton

2006

■ 多层神经网络

■ 深度学习

2012

AlexNet在ImageNet比赛
击败传统机器学习方法

IMAGENET Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)

Held in conjunction with PASCAL Visual Object Classes Challenge 2012 (VOC2012)

2016

AlphaGo

2023

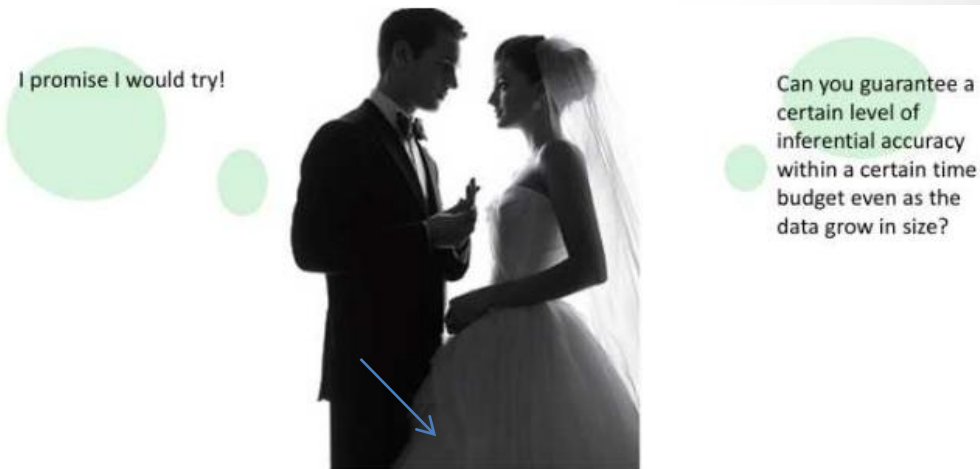
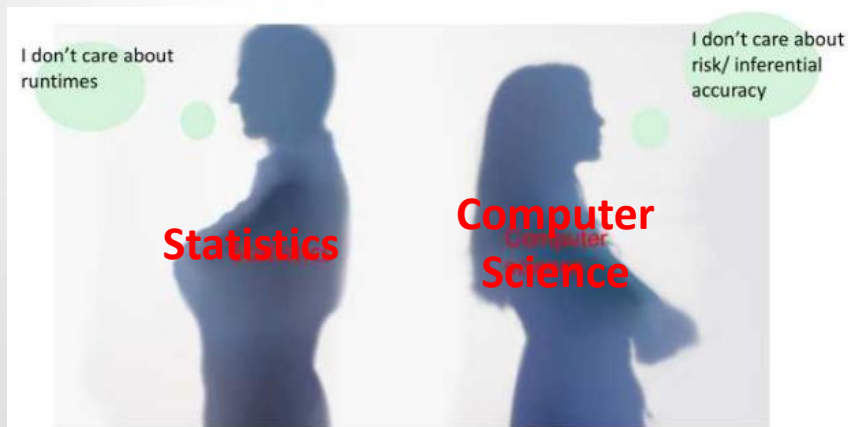
ChatGPT

第四次技术革命

第十章：神经网络与深度学习

□ 神经网络发展史

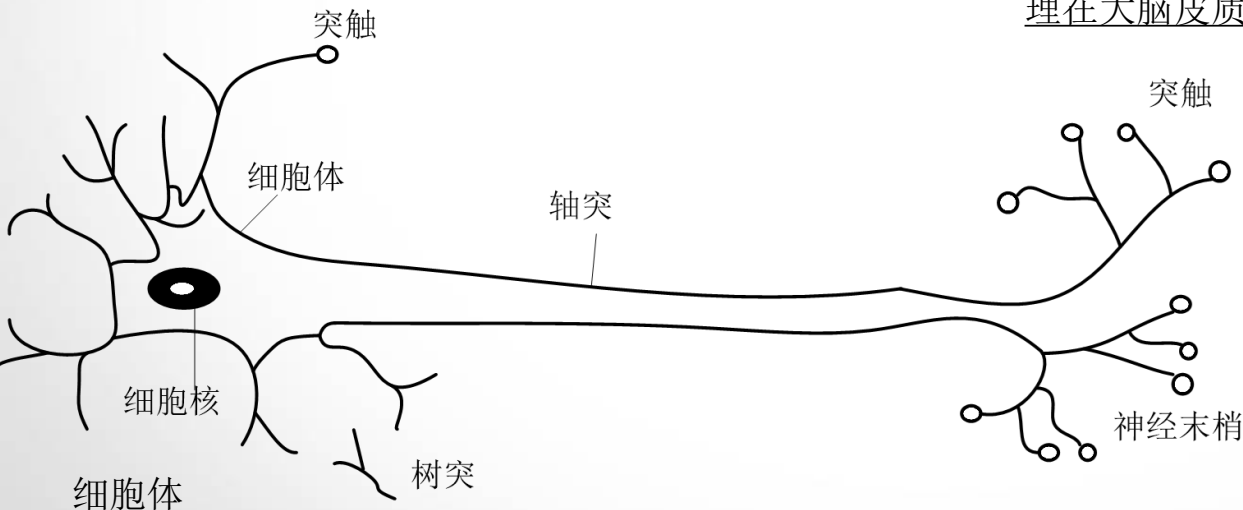
统计学与机器学习的关系



第十章：神经网络与深度学习

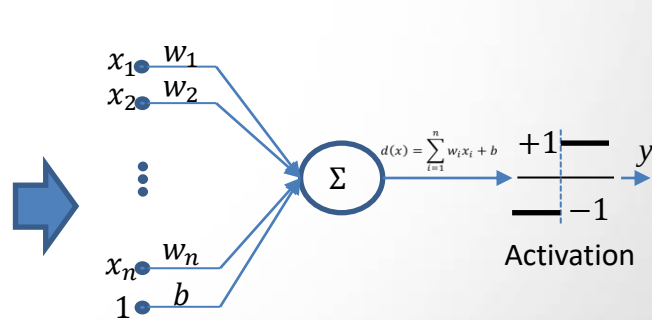
□ 感知器(Frank Rosenblatt, 1958)

生物神经元（神经细胞）结构：



生物神经元结构

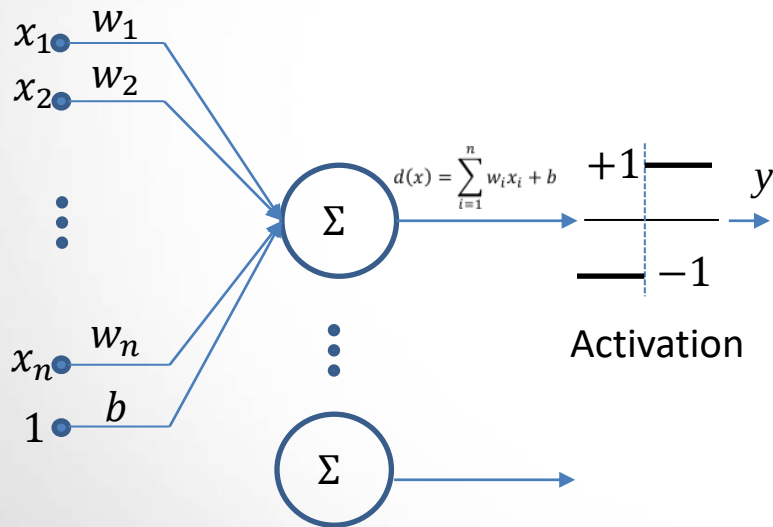
每个神经元由一个细胞体组成，其包含一个细胞核。从细胞体分支扩展出许多为树突的神经纤维和一根长的轴突。一个神经元与10到10万个其他神经元相连，连接处称为突触。信号通过复杂的化学反应从神经元传播到神经元。研究发现，大多数的信息处理在大脑皮质，即大脑外层进行。



人工神经元结构

第十章：神经网络与深度学习

□ 感知器



如何计算权重 w 和 b 呢？

✓ 感知器学习规则

采用类梯度下降方法：

1. 如果第 j 个神经元输出是正确的，则 w_{ij} 和偏差 b_j 不变；
2. 如果第 j 个神经元输出是0，而期望值是1，则 $w_{ij}=w_{ij}+x_i$
 $b_j=b_j+1$ ；
3. 如果第 j 个神经元输出是1，而期望值是0，则 $w_{ij}=w_{ij}-x_i$
 $b_j=b_j-1$ ；



w 和 b 的学习算法：

设第 j 个神经元输出为 y_j ，期望值为 t_j ，则学习规则为

$$w_{ij}=w_{ij}+(t_j-y_j)x_i$$

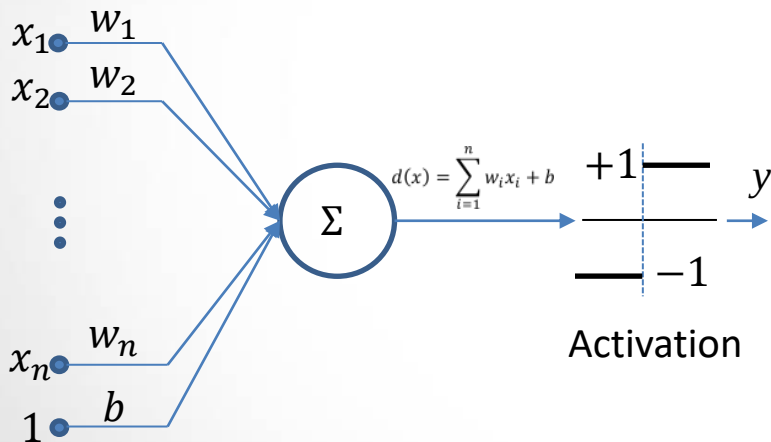
$$b_j=b_j+(t_j-y_j)1$$

$Y=T$

→ 目标任务

第十章：神经网络与深度学习

□ 感知器



单个神经元的感知器结构

✓ 感知器的局限性

1. 感知器的激活函数为阈值函数，输出只能是两个值，因此只能用于简单的分类问题；
2. 感知器是一种线性分类方法，适用于线性可分问题；
3. 收敛速度较慢。

✓ 例：利用感知器，对4个二维样本进行分类：

$$X = \begin{bmatrix} -2 & -1 & 1 & 0 \\ -1 & -2 & 1 & 2 \end{bmatrix}, \text{ 目标为 } T = [0 \ 0 \ 1 \ 1];$$

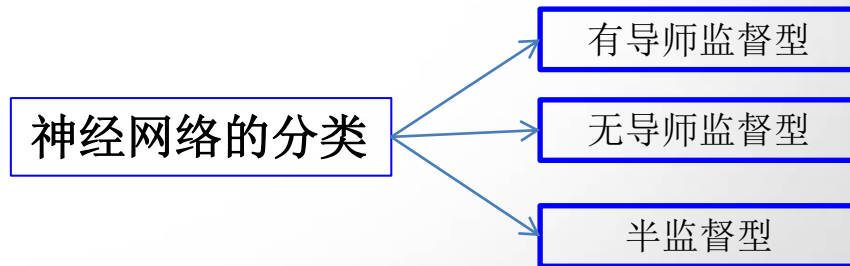
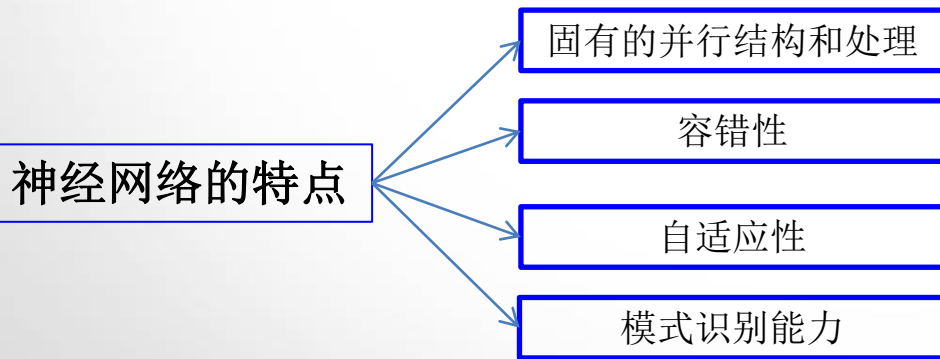
初始权重为 $w = [0, 0], b = 0$



第十章：神经网络与深度学习

□ 人工神经网络（Artificial Neural Network, ANN）

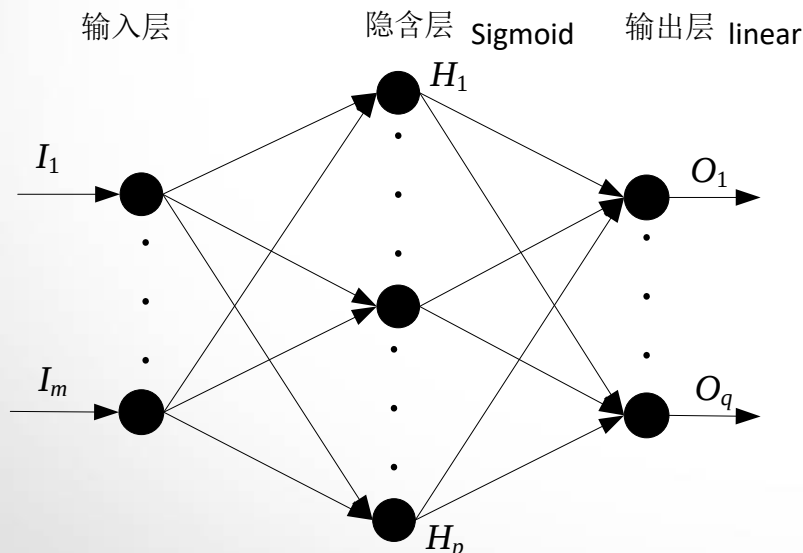
定义：ANN是对人脑或自然神经网络若干基本特性的抽象和模拟，是一种基于连接学说构造的智能仿生模型，是由大量神经元组成的非线性动力系统。神经网络可以看作是输入空间到输出空间的非线性映射，通过调整权重和阈值来“学习”和发现变量间的关系，实现分类或回归任务。



第十章：神经网络与深度学习

□ BP神经网络（Back-propagation Neural Network）

定义：BP网络是一种多层前馈网络，采用最小均方误差为学习函数，梯度下降算法为学习规则，是最广泛使用的网络。



单隐层的BP网络结构

■ 数学理论基础—万能逼近理论



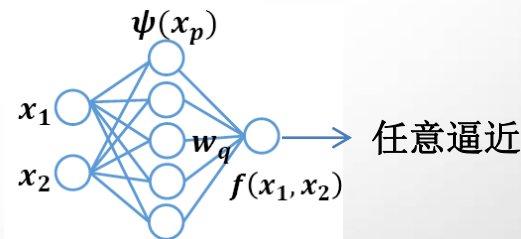
俄国数学家安德雷·柯尔莫哥洛夫, 1956

Kolmogorov定理

$$f(x_1, \dots, x_n) = \sum_{q=1}^{q=2n+1} w_q \left[\sum_{p=1}^n \psi(x_p) \right]$$

st. ψ is a sigmoidal function

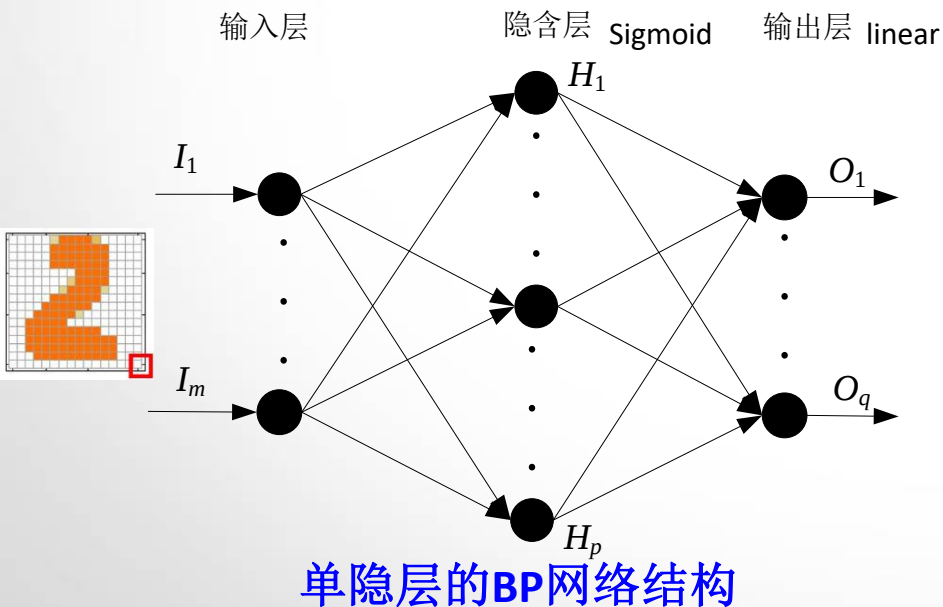
隐层大小为 $2n+1$ ，神经网络以任意精度逼近任意一个 n 元的连续实值函数。



第十章：神经网络与深度学习

□ BP神经网络（Back-propagation Neural Network）

BP训练算法（链式求导法则）：



◆ 输入层到隐含层的权重更新

$$w_{j,i}(t+1) = w_{j,i}(t) + \alpha \delta_j I_i$$

δ_j 为隐含层第 j 个节点的偏差信号；

◆ 隐含层到输出层的权重更新

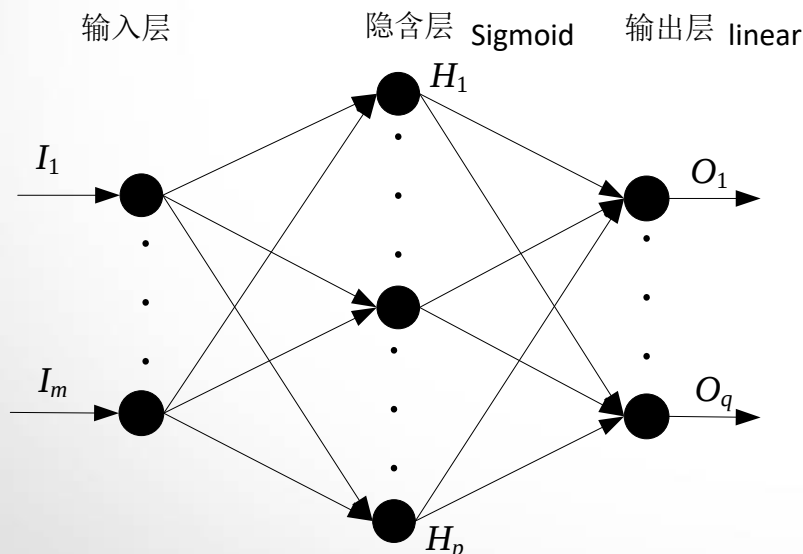
$$w_{k,j}(t+1) = w_{k,j}(t) + \alpha \delta_k H_j$$

H_j 为隐含层第 j 个节点的输出信号；
 δ_k 为输出层第 k 个节点的偏差信号；

第十章：神经网络与深度学习

□ BP神经网络（Back-propagation Neural Network）

BP训练算法：



单隐层的BP网络结构

◆ 隐含层每个节点阈值的更新

$$b_j(t+1) = b_j(t) + \beta \delta_j$$

δ_j 为隐含层第 j 个节点的偏差信号；

$$\delta_j = \sum_k \delta_k w_{k,j} H_j (1 - H_j)$$

◆ 输出层每个节点阈值的更新

$$b_k(t+1) = b_k(t) + \beta \delta_k$$

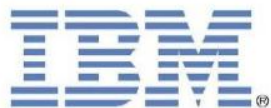
δ_k 为输出层第 k 个节点的偏差信号；

$$\delta_k = (T_k - O_k) O_k (1 - O_k)$$

第十章：神经网络与深度学习

□ 深度学习（Deep learning）

自2012年开始，深度学习的突破性进展



NVIDIA



文本

谷歌NMT翻译
智能问答系统
SyntaxNet
语法分析
...

图像

图像内容分类
图像内容描述
目标追踪检测
图片风格迁移
...

语音

语音内容识别
语音情感分析
音乐自动生成
...

...

...

...

...

第十章：神经网络与深度学习

□ 深度学习（Deep learning）

神经网络是深度学习研究的一部分，典型的网络为**卷积神经网络(CNN)**。

主要应用领域：图像或视频 (局部感受野，端到端学习：End to End Learning)

人类视觉

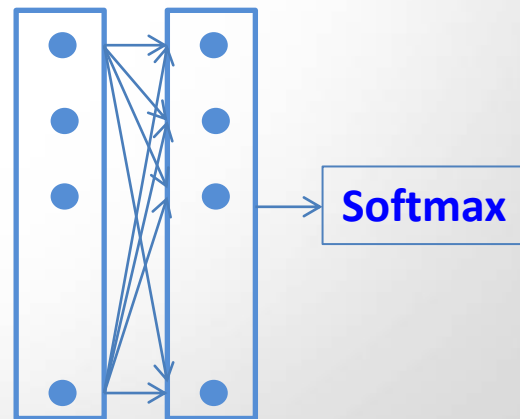
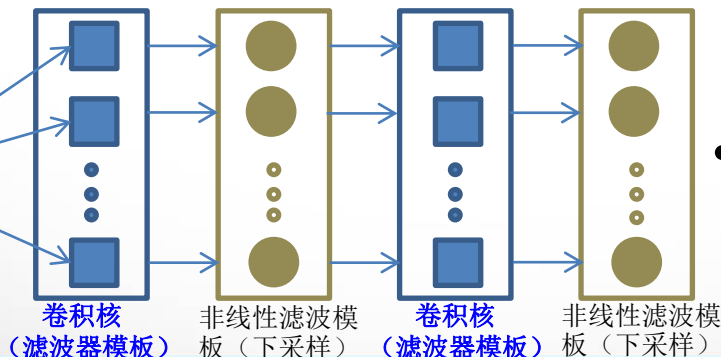


100x100x3



254 255 255 253 249 246 241 241 243 240 239 240 240
255 255 255 251 246 245 242 243 244 240 237 240 241
255 255 255 249 243 243 244 244 240 238 240 241
255 254 253 248 243 243 243 243 242 240 240 239
254 252 251 247 242 242 243 241 243 244 240 240
255 254 250 244 240 240 243 243 242 242 241 241 240
253 254 251 245 242 241 241 241 241 241 242 241 241
251 253 252 245 242 242 239 239 241 241 243 243 241
251 251 249 244 241 243 239 238 240 240 241 242 242
250 247 245 243 240 243 240 240 239 237 237 241 242
247 244 244 243 242 243 241 241 240 237 235 239 242
247 244 244 245 243 242 241 241 239 239 237 237 240
250 245 242 243 242 242 241 241 240 240 240 237 238
252 246 242 242 242 242 241 241 240 240 241 238 237
254 248 244 243 241 242 242 241 240 241 243 241 237
254 248 242 242 241 240 240 241 240 240 241 240
253 249 244 242 241 239 239 240 241 241 240 241 241
252 251 247 243 242 241 240 240 239 240 241 239 240

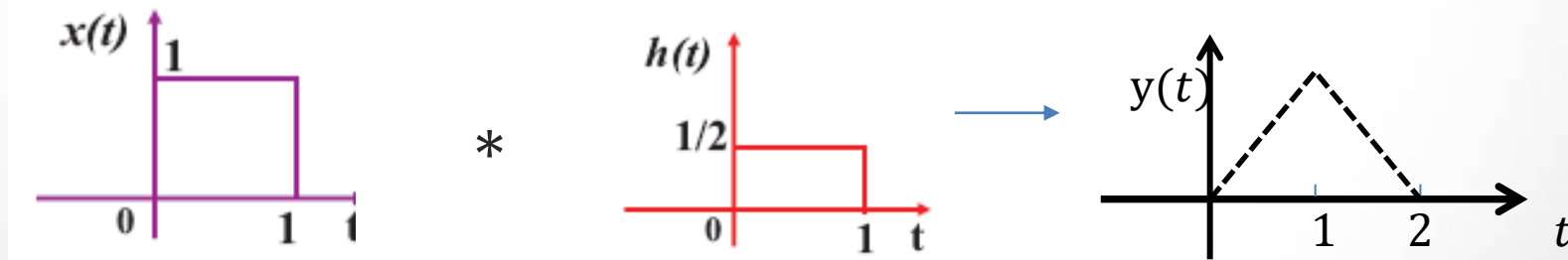
计算机视觉



第十章：神经网络与深度学习

□ 卷积（Convolution）与相关（Correlation）

卷积定义：
$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau$$



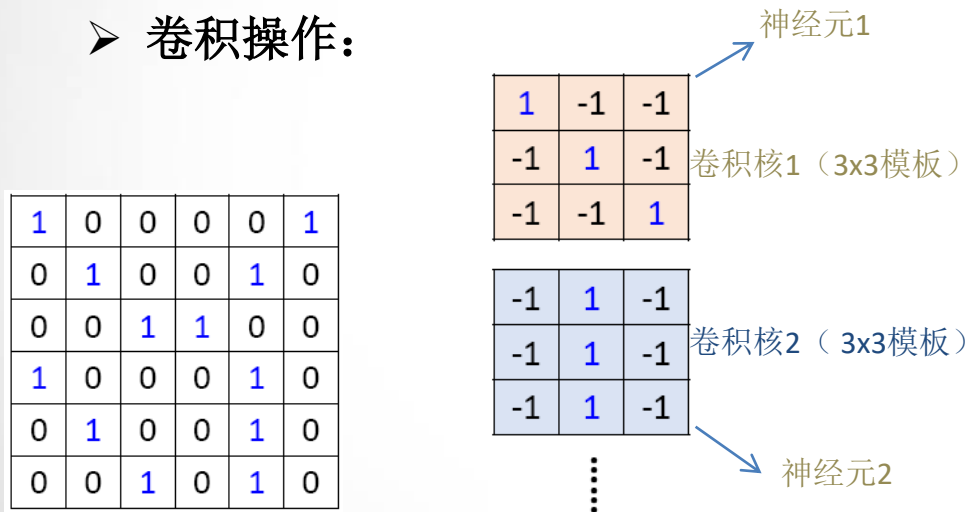
相关定义：
$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(\tau - t)d\tau$$

注：深度学习框架中的“卷积”实际上是“相关”运算，并非真正的卷积。

第十章：神经网络与深度学习

深度学习 (Deep learning)

卷积操作:



先考虑卷积核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

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



卷积核在图像上逐步滑动

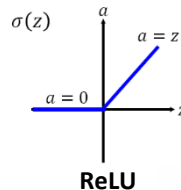
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

卷积核1

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

Feature map

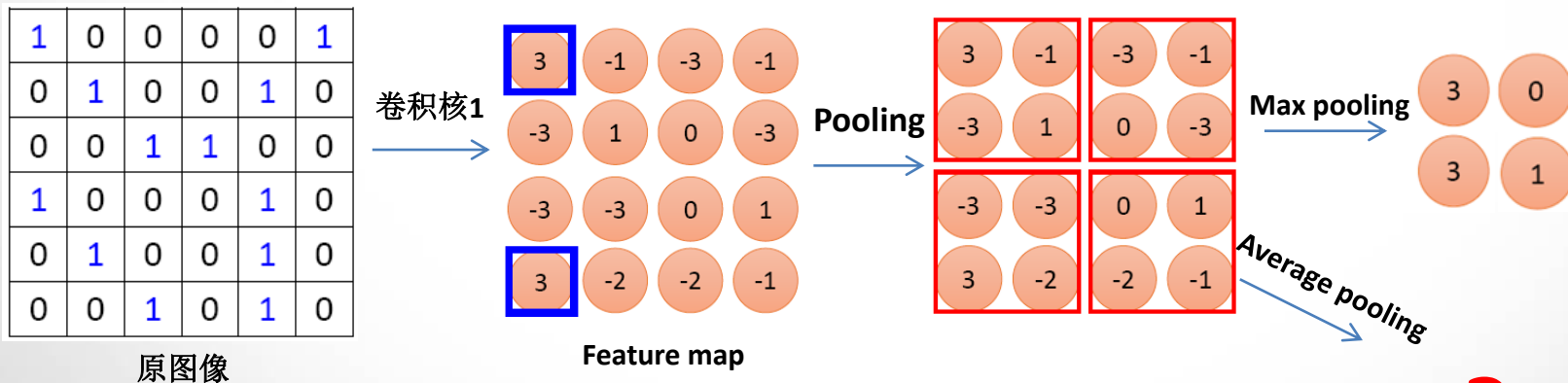
第十章：神经网络与深度学习



深度学习 (Deep learning)

下采样操作(Pooling池化层):

常见方法: **Average pooling**(均值滤波)和**Max pooling**(类似非线性滤波)

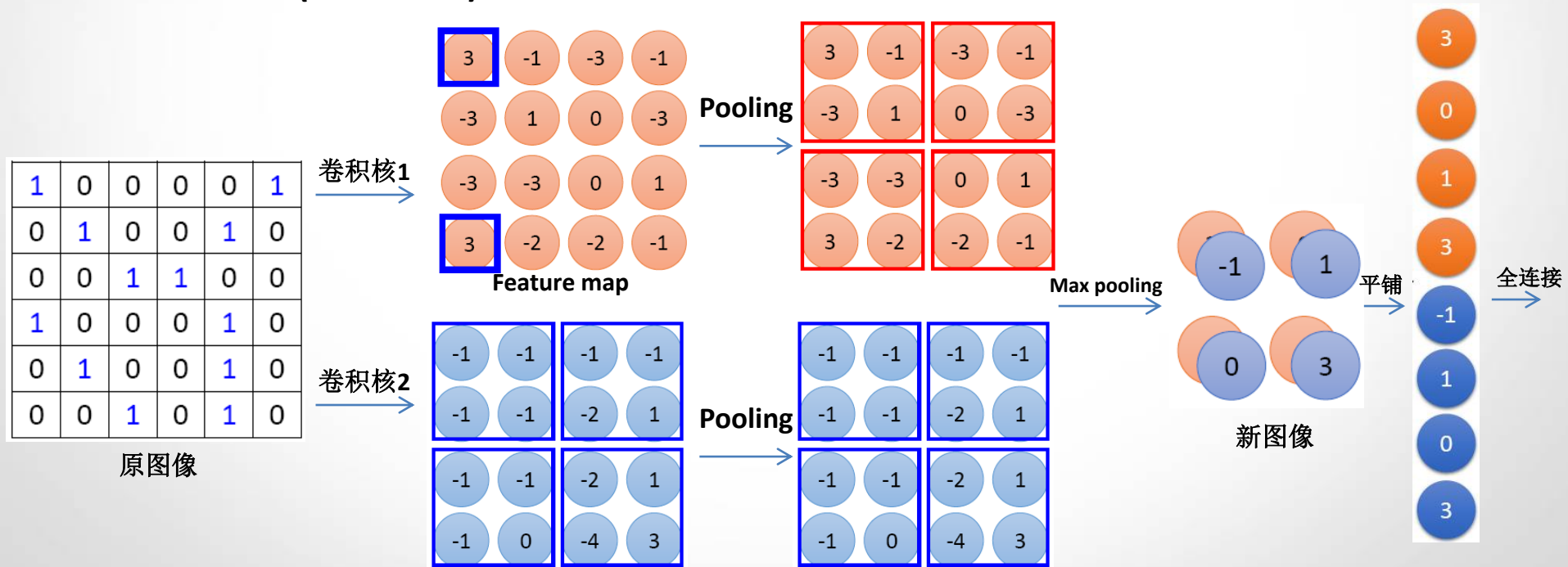


注：每一层输出还需要非线性激活函数映射，通常为ReLU: $\max(0, z)$ 激活函数

第十章：神经网络与深度学习

□ 深度学习（Deep learning）

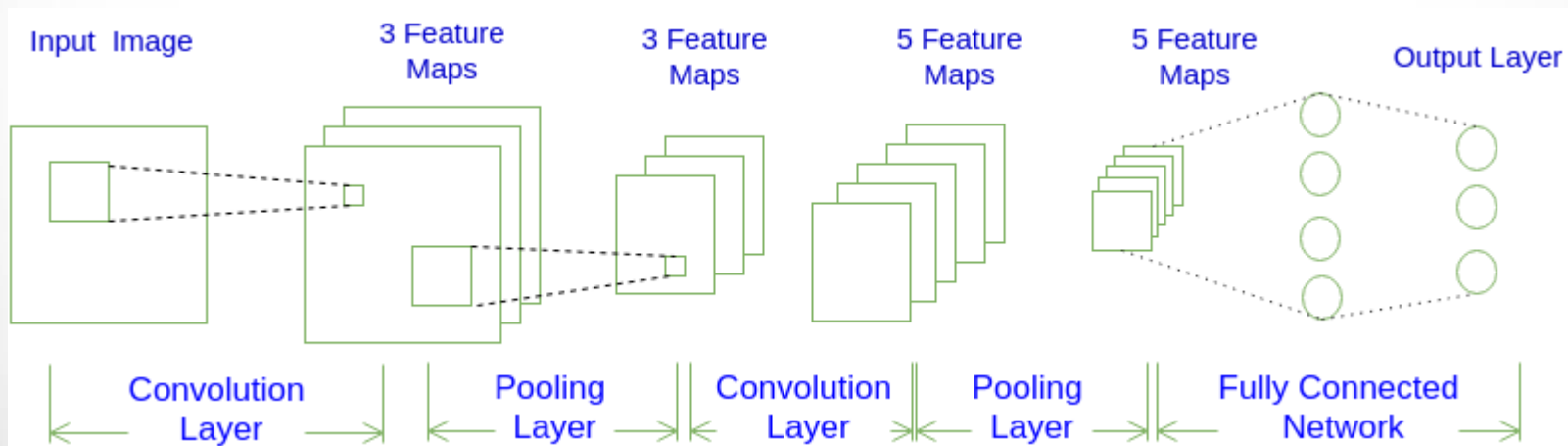
➤ 全连接层(串联平铺):



第十章：神经网络与深度学习

□ 深度学习（Deep learning）

➤ 卷积神经网络完整结构：





第十章：神经网络与深度学习

□ 深度学习（Deep learning）

➤ CNN的训练方法

1. 损失函数：

Softmax交叉熵损失： $L = -\sum_{i=1}^m \log P_i = -\sum_{i=1}^m \log \frac{e^{z_{yi}}}{\sum_{j=1}^K e^{z_j}}$ （交叉熵损失）

Logistic Regression?

2. Mini-batch随机梯度下降

提高计算效率，增强网络泛化能力

局部连接

3. 误差后向传播BP算法

包括三个部分的反传：卷积层、池化层、输出层

CNN优点：

权重共享

4. 深度学习开发程序

Pytorch, TensorFlow, Caffe, Jittor, MegEngine (Python, GPU)

时间/空间
次采样



第十章：神经网络与深度学习

□ 关于交叉熵损失的基础概念回顾

➤ **信息量**：信息消除不确定性的程度。

定义：信息量的大小与信息 x 发生的概率成反比， $I(x) = -\log P(x)$

➤ **信息熵**：信息量的熵，即所有信息量的期望。

定义： $H(X) = -\sum_{i=1}^N P(x_i) \log P(x_i)$

➤ **相对熵**：两个概率分布 $p(x)$ 和 $q(x)$ 之间的差异度量，一般采用KL散度

定义： $D_{KL}(p(X) \parallel q(X)) = \sum_{i=1}^N p(x_i) \log \frac{p(x_i)}{q(x_i)}$ ，KL散度越小，表示两个分布更接近

第十章：神经网络与深度学习

□ 关于交叉熵损失的基础概念回顾

➤ **交叉熵**：相对熵和信息熵的总和。

定义：对KL散度展开， $D_{KL}(p(X) \parallel q(X)) = \sum_{i=1}^N p(x_i) \log \frac{p(x_i)}{q(x_i)} =$
 $\sum_{i=1}^N p(x_i) \log p(x_i) - \sum_{i=1}^N p(x_i) \log q(x_i) = -H(X) + (-\sum_{i=1}^N p(x_i) \log q(x_i))$

其中， $\mathbf{p}(\mathbf{x})$ 为真实分布， $\mathbf{q}(\mathbf{x})$ 为预测分布， $\mathbf{H}(\mathbf{x})$ 可以看作常数。

交叉熵的公式定义为：

$$H(p, q) = - \sum_{i=1}^N p(x_i) \log q(x_i)$$

$$= - \sum_{i=1}^N \sum_{c=1}^C p_c(x_i) \log q_c(x_i) = - \sum_{i=1}^N \log q(x_i)$$



狗	猫	鸟
0	1	0
0.2	0.7	0.1

真实

预测

此处，KL散度和交叉熵等价

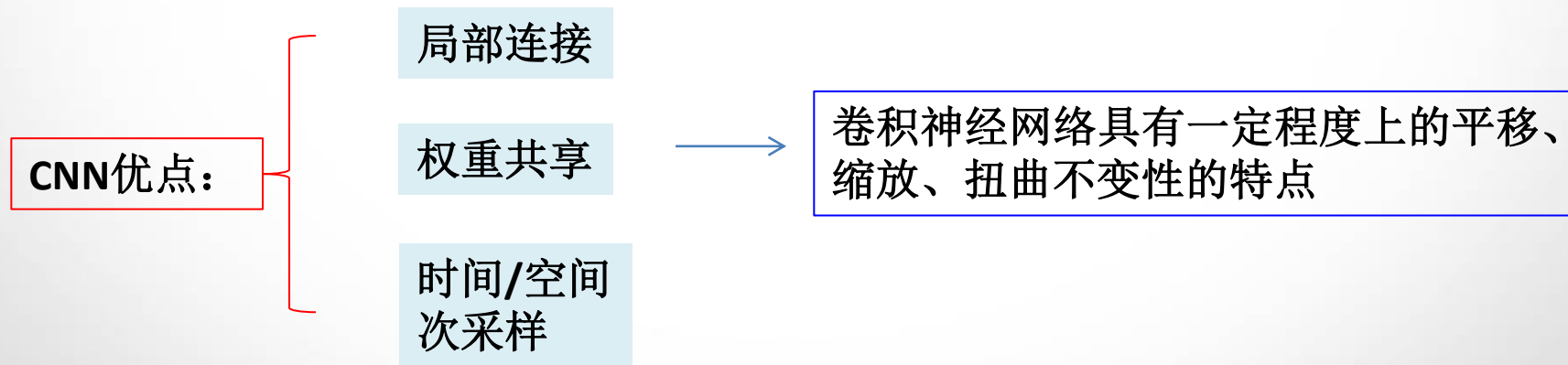


第十章：神经网络与深度学习

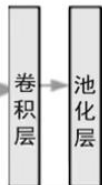
□ 深度学习（Deep learning）

➤ CNN优点

启发于生物上的“感受野”机制，在视觉神经系统中，一个神经元的感受野是指视网膜的特定区域，只有在该区域内的刺激，才能激活该神经元。



输入



output: 56*56*64;

Transition Block

Transition Block

Transition Block

Classification Block

Dense Block 1

Dense Block 2

Dense Block 3

Dense Block 4

预测

“马”

BN
Conv 3*3,32
Relu

cat

BN
Conv 3*3,32
Relu

cat

BN
Conv 3*3,32
Relu

*2

Bottleneck 3

```
# 设置最后一个Dense Block (最后一个Dense Block不需要Transition Block)
feature_list = [x]
for i in range(nb_layers):
    x = Conv2D(growth_rate, (3, 3), padding="same", activation='relu')(x)
    feature_list.append(x)
    if i < (nb_layers-1):
        x = Concatenate()(feature_list)

# 全局池化
x = GlobalAveragePooling2D()(x)
x = Dense(2, activation='softmax')(x)
output_layer = x
model = Model(input_layer, output_layer)
model.summary()
```

设置 [Dense Block + Transition Block] 多个, 此处以1个为例

```
for j in range(3):
```

```
# 1. Dense Block
```

```
# 用一个列表存放提供特征的层
```

```
feature_list = []
```

```
for i in range(nb_layers):
```

```
    x = BatchNormalization()(x)
```

```
    x = Conv2D(growth_rate, (3, 3), padding="same", activation='relu')(x)
```

```
    feature_list.append(x)
```

```
    if i < (nb_layers-1):
```

```
        x = Concatenate()(feature_list)
```

```
# 2. Transition Block
```

```
x = BatchNormalization()(x)
```

```
x = Conv2D(growth_rate, (1, 1), padding="same", activation='relu')(x)
```

```
x = AveragePooling2D((2, 2), strides=(2, 2))(x)
```

#代码示例: 6-16

```
from keras.optimizers import Adam
```

```
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['accuracy'])
```

```
model.fit_generator(train_generator, epochs=5, validation_data=validation_generator)
```



第十章：神经网络与深度学习

□ Transformer（变换器）

背景：2017年由谷歌团队首次提出，论文“Attention is All You Need”发表在机器学习顶会NeurIPS，在自然语言处理的多个任务上取得惊艳的效果。其在结构上完全抛弃了CNN和RNN，完全由注意力机制构成。

贡献：大语言模型均基于Transformer构建，如BERT、GPTs、ChatGPT (Chat Generative Pretraining Transformer)

延伸：视觉Transformer（ViT）的提出，开启了计算机视觉从CNN向Transformer的技术转变，成为视觉大模型的核心技术。

大一统：视觉-语言多模态大模型，如Sora、GPT4o

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, et al. “Attention Is All You Need”, NeurIPS, 2017.

第十章：神经网络与深度学习

Transformer（变换器）

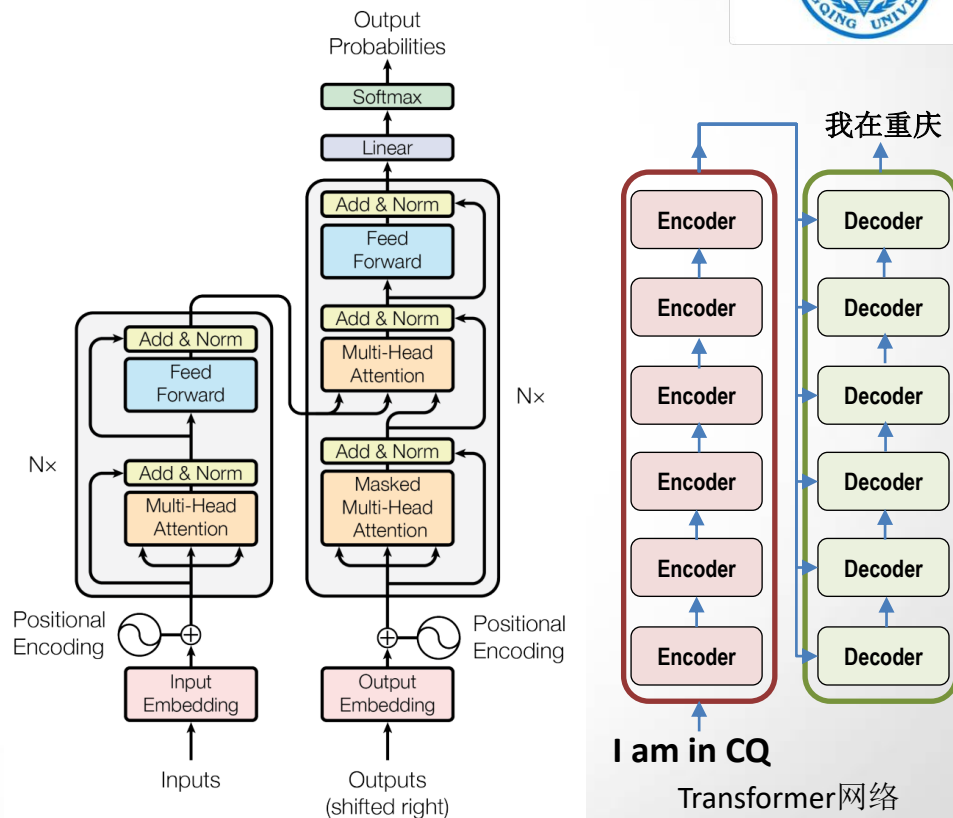
Transformer是一个Encoder-Decoder结构。

输入：一段文本 “I am in CQ”

输出：一段文本 “我在重庆”

Encoder (编码器): 将输入变换到编码空间

Decoder(解码器): 获得预测输出（翻译）



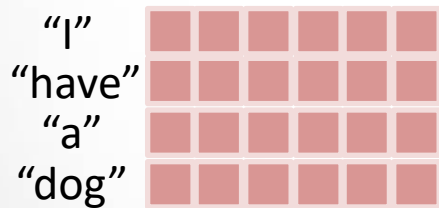
Transformer基本结构 (1个encoder+1个decoder)

第十章：神经网络与深度学习

□ Encoder的输入（embedding）

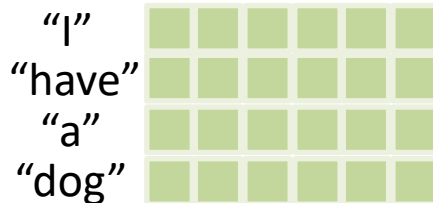
① 输入嵌入 (input embedding)

将每个单词进行word embedding，
形成512维的向量



② 位置编码 (positional encoding)

将每个单词的位置进行编码，
形成512维的向量



$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Encoder的输入 $X_{n \times d} = \text{input embedding} + \text{positional encoding}$

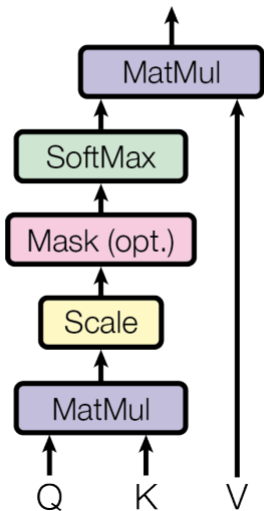
其中，n是单词的数量，d是嵌入维度

第十章：神经网络与深度学习

□ Attention（注意力机制）

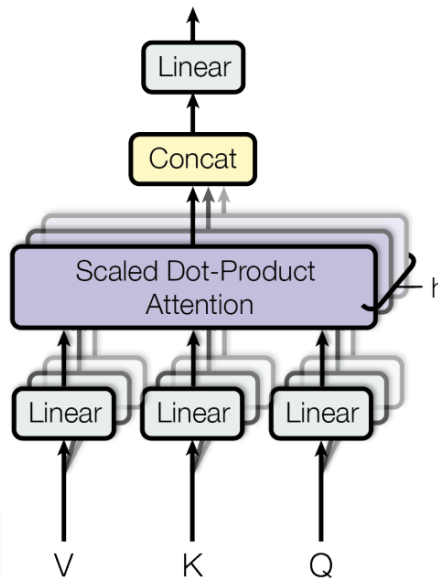
① Self-attention(自注意力)

Scaled Dot-Product Attention



② Multi-head attention (多头注意力)

Multi-Head Attention

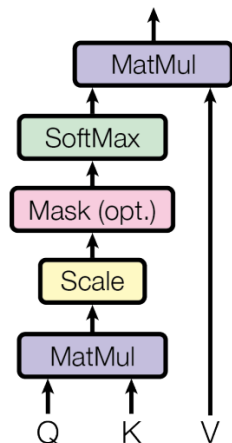


第十章：神经网络与深度学习

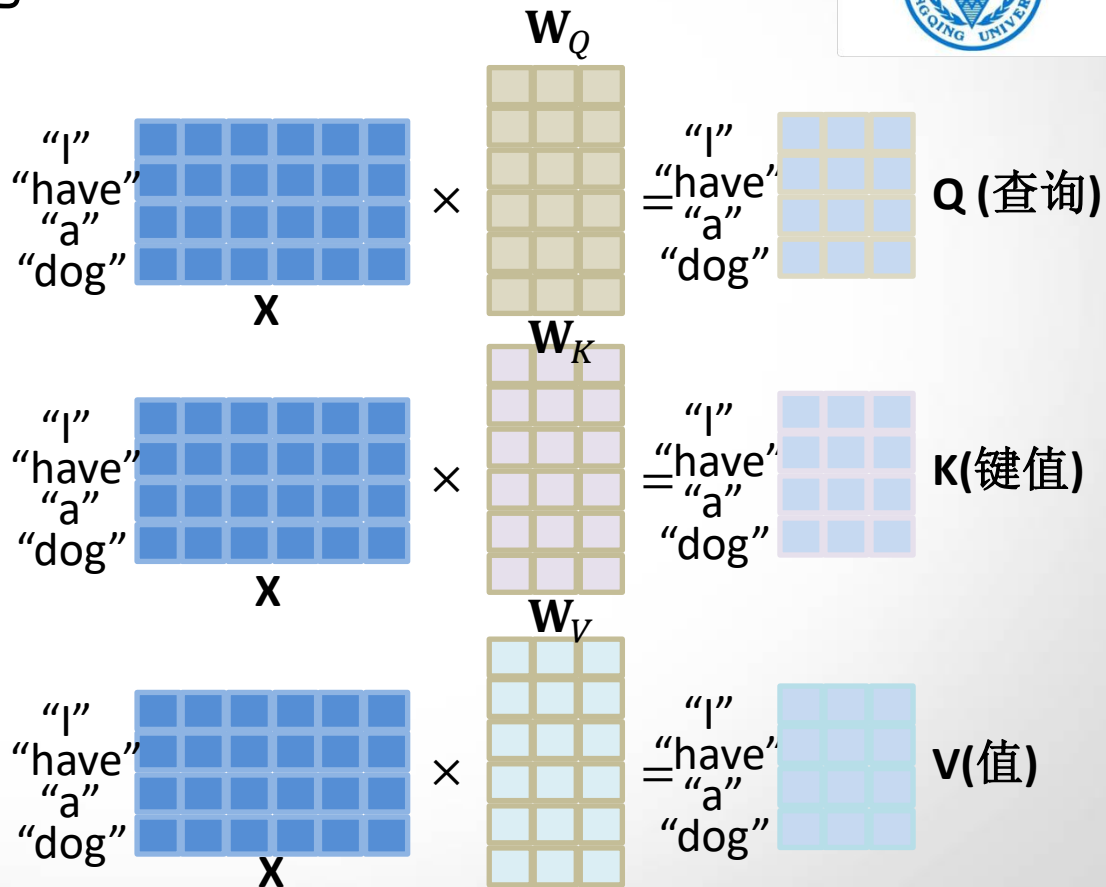
□ Attention（注意力机制）

① Self-attention(自注意力)

Scaled Dot-Product Attention



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

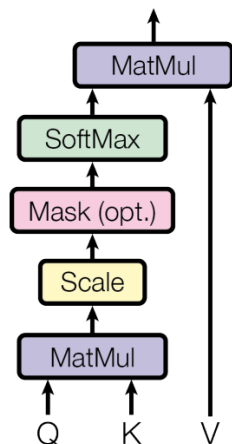


第十章：神经网络与深度学习

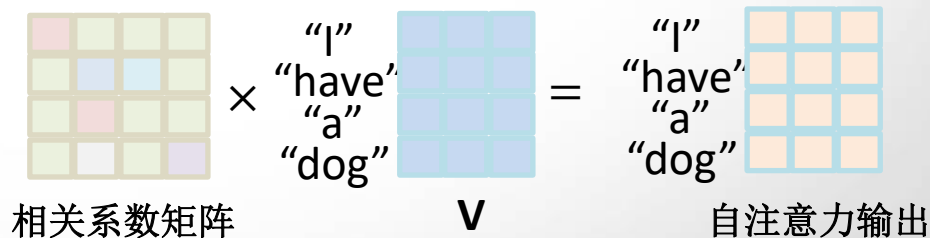
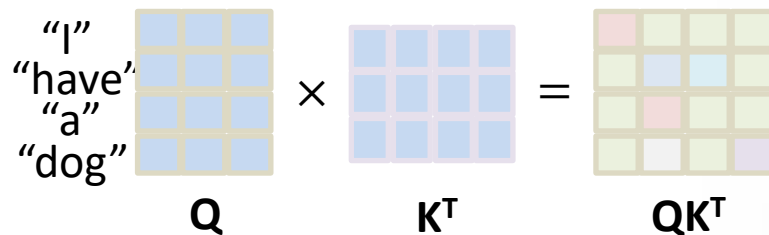
□ Attention（注意力机制）

① Self-attention(自注意力)

Scaled Dot-Product Attention



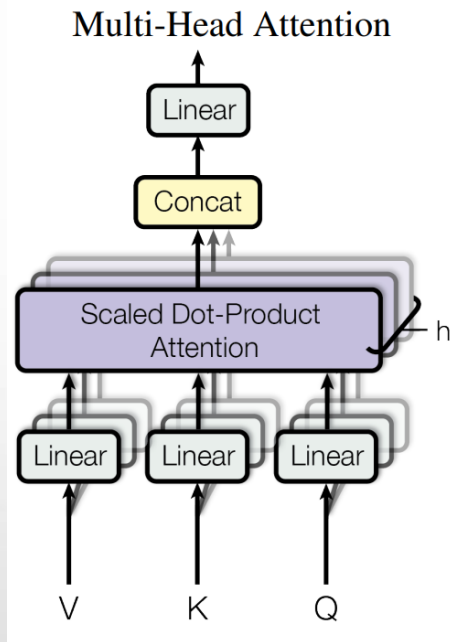
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



第十章：神经网络与深度学习

□ Attention（注意力机制）

② Multi-head attention (多头注意力)

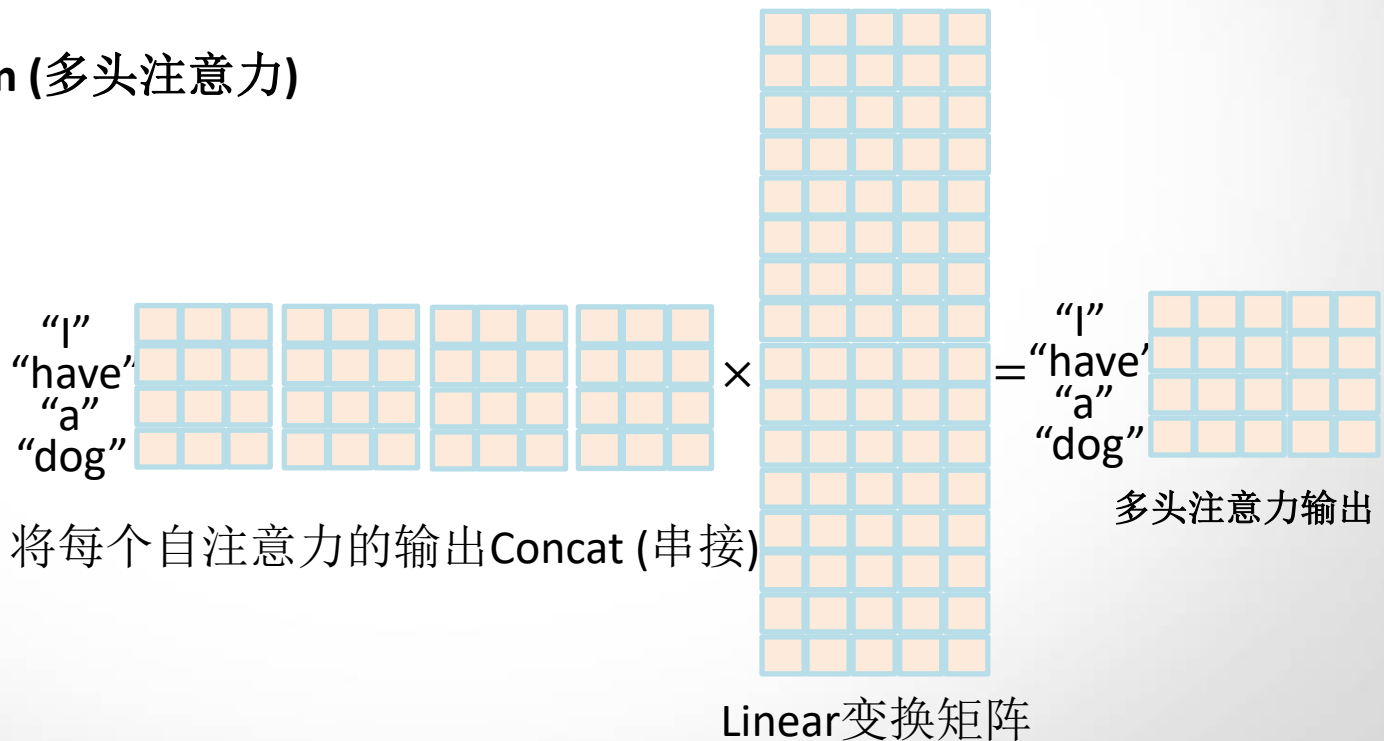
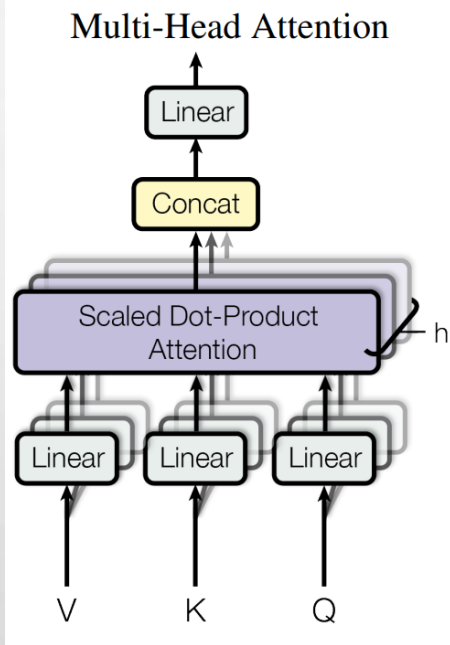


$$\begin{array}{c}
 \begin{array}{ccc}
 \begin{array}{|c|c|c|c|c|} \hline \text{red} & \text{green} & \text{green} & \text{green} & \text{green} \\ \hline \text{green} & \text{blue} & \text{blue} & \text{green} & \text{green} \\ \hline \text{green} & \text{red} & \text{green} & \text{green} & \text{green} \\ \hline \text{green} & \text{green} & \text{green} & \text{green} & \text{purple} \\ \hline \end{array} & \times & \begin{array}{c} \text{"I"} \\ \text{"have"} \\ \text{"a"} \\ \text{"dog"} \end{array} \\
 \text{注意力系数矩阵} & & V \\
 \hline
 \begin{array}{ccc}
 \begin{array}{|c|c|c|c|c|} \hline \text{red} & \text{green} & \text{green} & \text{green} & \text{green} \\ \hline \text{green} & \text{blue} & \text{blue} & \text{green} & \text{green} \\ \hline \text{green} & \text{red} & \text{green} & \text{green} & \text{green} \\ \hline \text{green} & \text{green} & \text{green} & \text{green} & \text{purple} \\ \hline \end{array} & \times & \begin{array}{c} \text{"I"} \\ \text{"have"} \\ \text{"a"} \\ \text{"dog"} \end{array} \\
 \text{注意力系数矩阵} & & V \\
 \hline
 \begin{array}{ccc}
 \begin{array}{|c|c|c|c|c|} \hline \text{red} & \text{green} & \text{green} & \text{green} & \text{green} \\ \hline \text{green} & \text{blue} & \text{blue} & \text{green} & \text{green} \\ \hline \text{green} & \text{red} & \text{green} & \text{green} & \text{green} \\ \hline \text{green} & \text{green} & \text{green} & \text{green} & \text{purple} \\ \hline \end{array} & \times & \begin{array}{c} \text{"I"} \\ \text{"have"} \\ \text{"a"} \\ \text{"dog"} \end{array} \\
 \text{注意力系数矩阵} & & V
 \end{array}
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{ccc}
 \begin{array}{|c|c|c|c|c|} \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \end{array} & \text{"I"} \\ \text{自注意力输出} \\
 \hline
 \begin{array}{ccc}
 \begin{array}{|c|c|c|c|c|} \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \end{array} & \text{"I"} \\ \text{自注意力输出} \\
 \hline
 \begin{array}{ccc}
 \begin{array}{|c|c|c|c|c|} \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \text{orange} & \text{orange} & \text{orange} & \text{orange} & \text{orange} \\ \hline \end{array} & \text{"I"} \\ \text{自注意力输出}
 \end{array}
 \end{array}$$

第十章：神经网络与深度学习

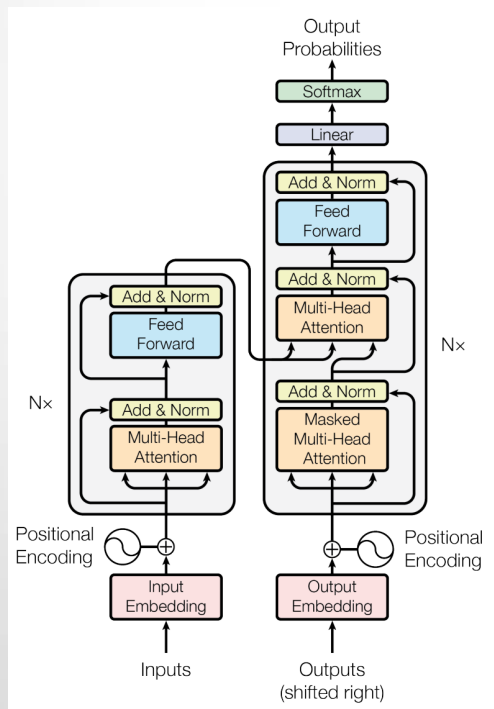
□ Attention（注意力机制）

② Multi-head attention (多头注意力)



第十章：神经网络与深度学习

□ Add、Normalization、Feed Forward（残差连接、归一化和前馈网络）



$$X = \text{LayerNorm}(X + \text{MHA}(X))$$



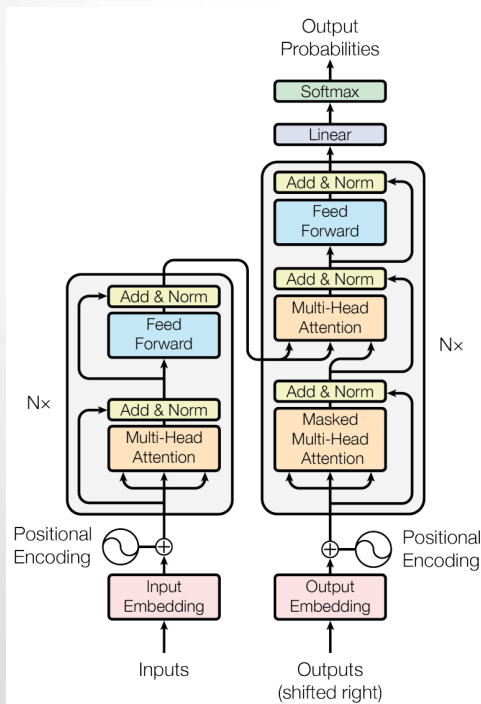
$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



Encoder输出: $Y = \text{LayerNorm}(X + \text{FFN}(X))$

第十章：神经网络与深度学习

□ Decoder



与Encoder类似，区别在于：

1. 包括两个MHA和三个Add&Norm
2. 第一个MHA是masked MHA，即预测某个单词时，需要对该单词之后的词汇掩盖。



《论语·述而》中有言：

“志于道，据于德，依于仁，游于艺”

希望本课程可激发大家学习人工智能的兴趣、了解人工智能的知识、乐于掌握人工智能的技能、形成人工智能伦理品行。

谢谢!

实验室论文、源码和数据发布

<https://www.leizhang.tk>

《机器学习》

主讲：张磊

E-mail: leizhang@cqu.edu.cn

