Survey on Deep Learning

—— in a view of NLP

费豪

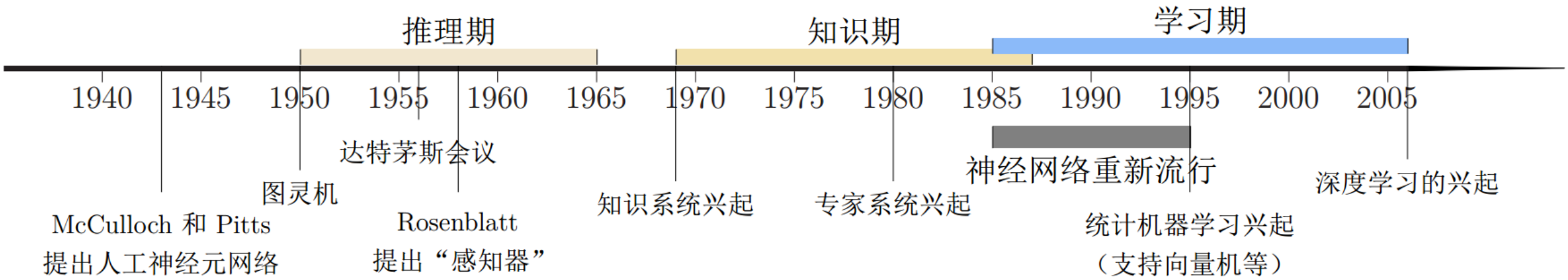
2020/8/12

Content

- Preliminary
- Starting from neural nets
- Neural models for NLP
- Neural nets for other domain applications
- Deep learning programming framework

○、Preliminary

1. 人工智能（神经网络）简史



第一阶段：模型提出，1943～1969

第二阶段：冰河期，1969～1983

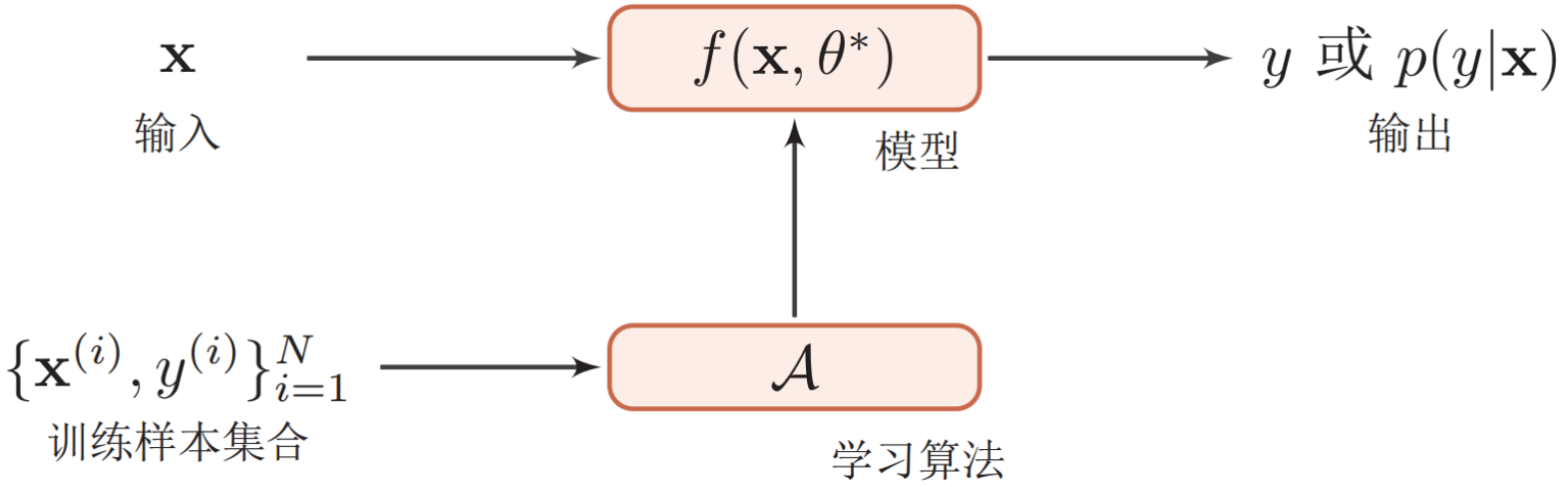
第三阶段：反向传播算法引起的复兴，1983～1995

第四阶段：流行度降低， 1995～2006

第五阶段：深度学习的崛起，2006-

○、Preliminary

2. 机器学习系统

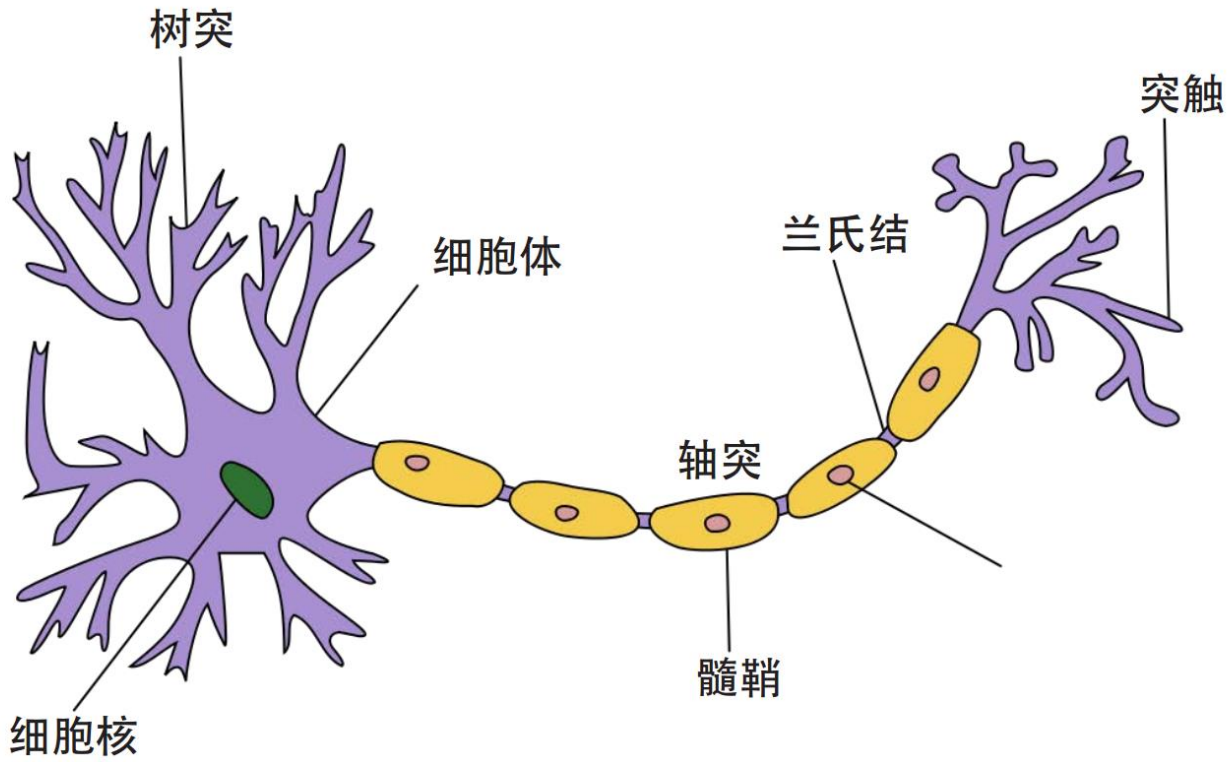


基于统计机器学习系统的本质：基于归纳偏置的函数拟合

一、Starting from neural nets

1. Neuron -> Perceptron -> MLP

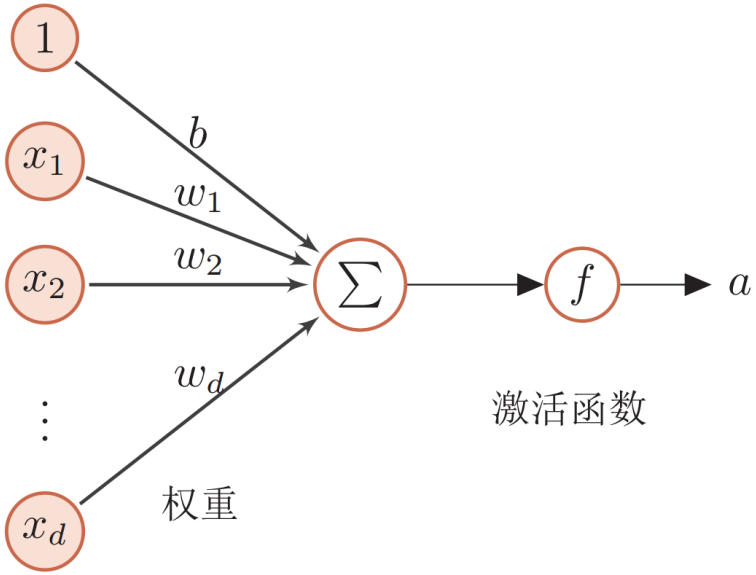
一个神经元结构：

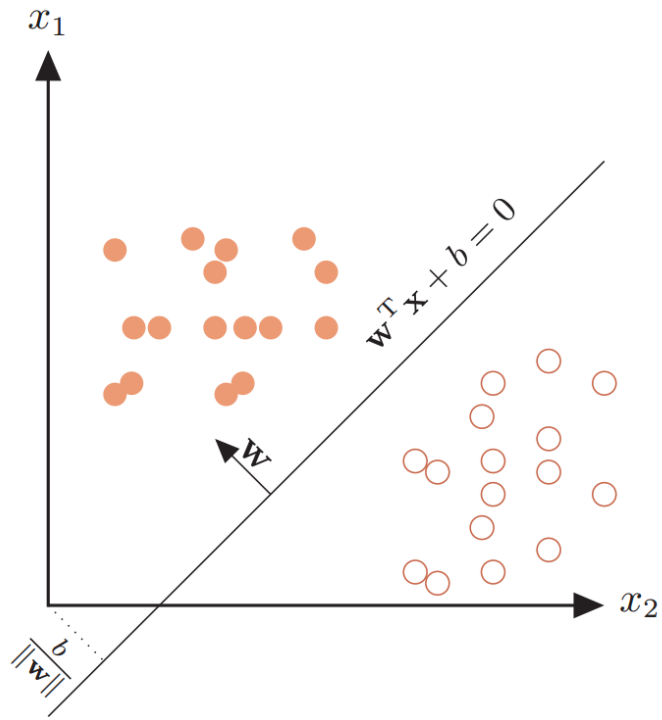


一、Starting from neural nets

1. Neuron -> Perceptron -> MLP

Perceptron, 感知机：二类分类的线性分类模型





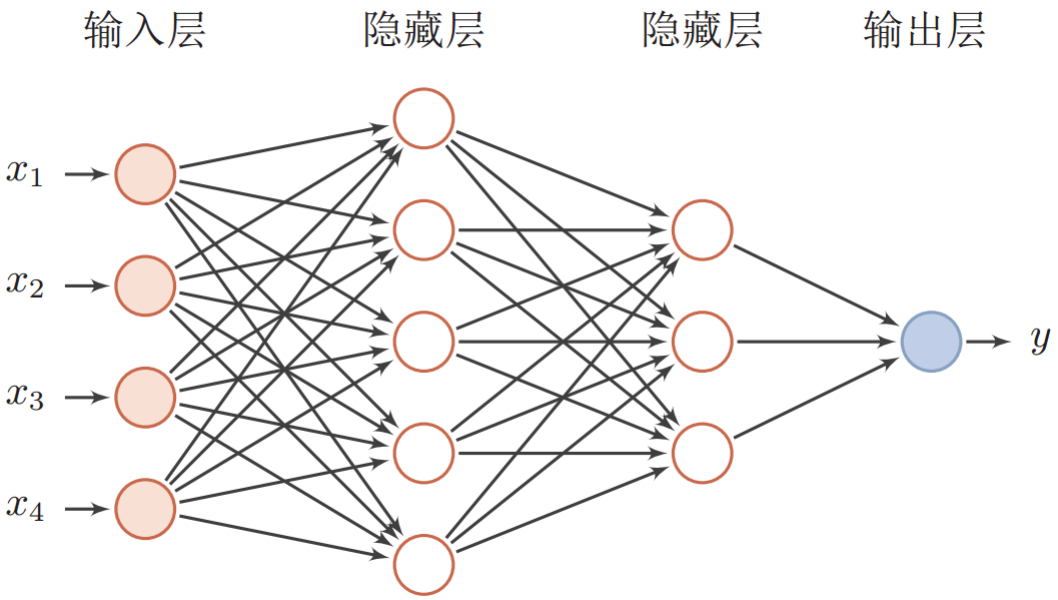
$f(x) = sign(w \cdot x + b)$ 非线性激活函数

$$sign(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

一、Starting from neural nets

1. Neuron -> Perceptron -> MLP

MLP, Multi-Layer Perceptron， 多层感知器
Dense Layer, 带隐层的神经网络
FCNN, Fully Connected Neural Network, 全连接神经网络
FNN, Feedforward Neural Network, 前馈神经网络



$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}).$$

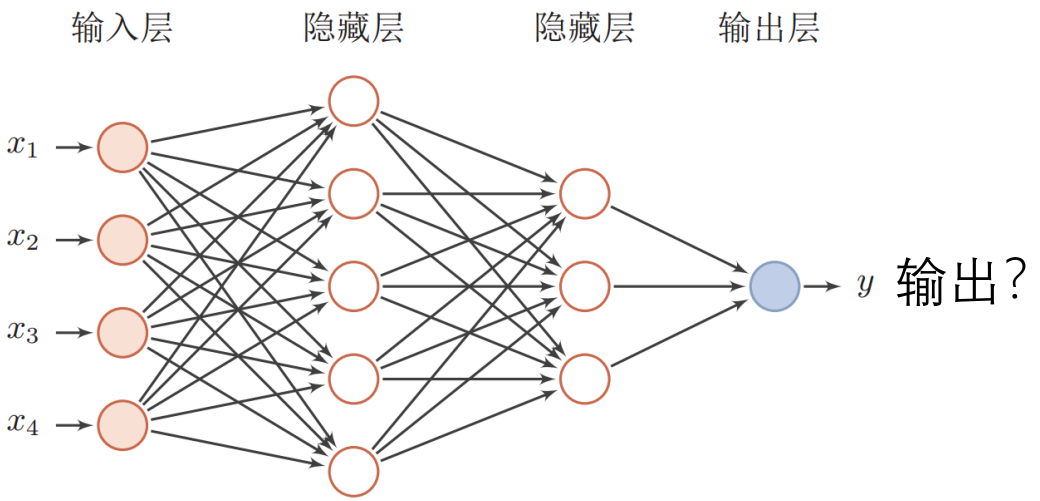
或者合并:

$$\mathbf{a}^{(l)} = f_l(W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})$$

$f_l(\cdot)$: l 层的非线性激活函数

一、Starting from neural nets

1. Neuron -> Perceptron -> MLP



- 回归问题，连续值输出
  - 直接输出

- 分类问题，离散值输出
  - Logistic 二分类

$$p(y=1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \triangleq \frac{1}{1+\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})},$$

  - Softmax 多类

$$p(y=c|\mathbf{x}) = \mathrm{softmax}(\mathbf{w}_c^{\mathrm{T}}\mathbf{x}) = \frac{\exp(\mathbf{w}_c^{\top}\mathbf{x})}{\sum_{c=1}^{C}\exp(\mathbf{w}_c^{\top}\mathbf{x})},$$

一、Starting from neural nets

2. Backpropagation

神经网络的学习方式：反向传播算法，链式求导

核心： 梯度下降, 计算参数(*W, b*)的偏导

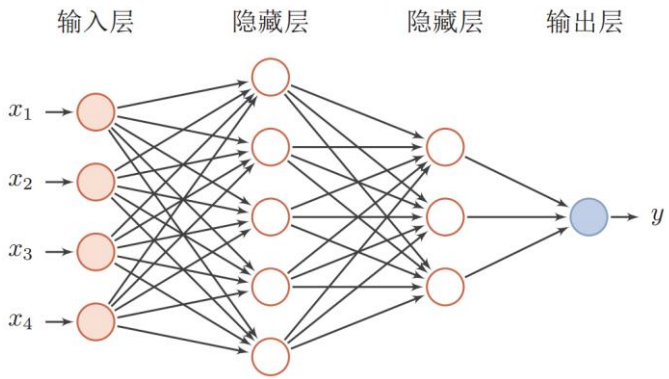
- 输入训练样本：$(\mathbf{x}, \mathbf{y})$
- 网络输出：$\hat{\mathbf{y}}$
- 真实结果与预测输出的误差，via损失函数：$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$
- 计算每一层中每一个感知机中的各个参数，via反向传播：

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial W_{ij}^{(l)}} = \left(\frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}} \right)^{\mathrm{T}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}},$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \left(\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \right)^{\mathrm{T}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}.$$

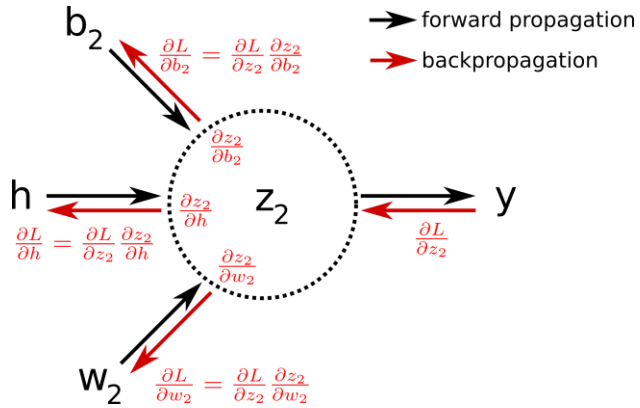
即计算这三项偏导：

$$\frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}} \quad \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \quad \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$



$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$

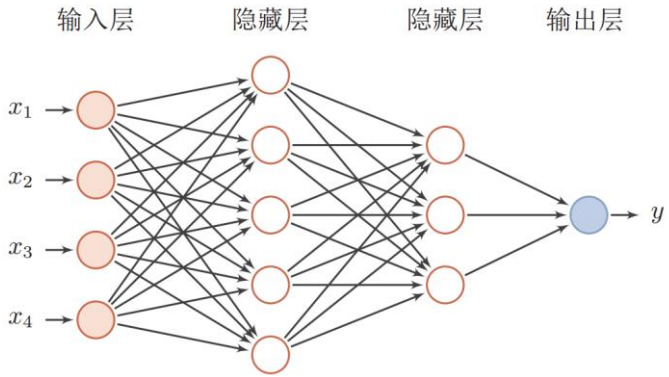
$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}).$$



一、Starting from neural nets

2. Backpropagation

（1）z与W $\dfrac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}}$



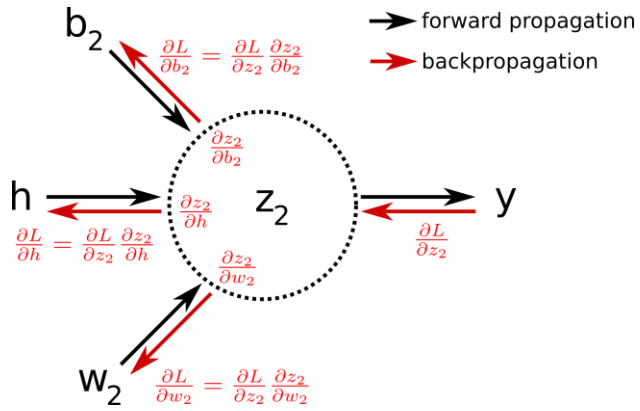
$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}).$$

$$\frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}} = \frac{\partial (W^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})}{\partial W_{ij}^{(l)}}$$

$$= \begin{bmatrix} \frac{\partial (W_{1:}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})}{\partial W_{ij}^{(l)}} \\ \vdots \\ \frac{\partial (W_{i:}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})}{\partial W_{ij}^{(l)}} \\ \vdots \\ \frac{\partial (W_{m^{(l)}:}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})}{\partial W_{ij}^{(l)}} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ a_j^{(l-1)} \\ \vdots \\ 0 \end{bmatrix}$$

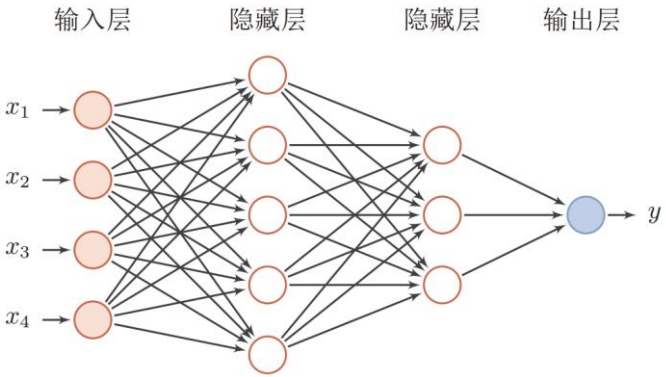
$$\triangleq \mathbb{I}_i(a_j^{(l-1)}),$$



一、Starting from neural nets

2. Backpropagation

（2）z与b $\dfrac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}}$

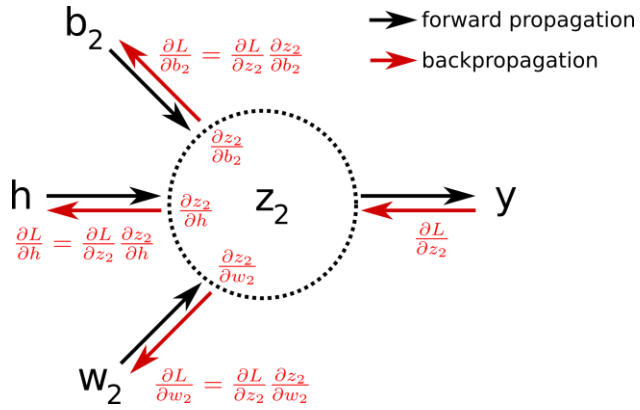


$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}).$$

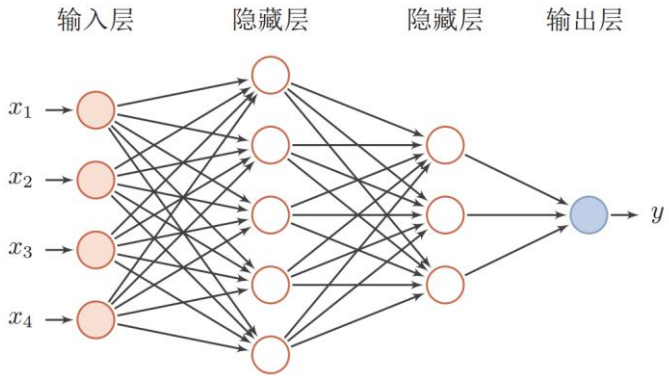
$$\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} = \mathbf{I}_{m^{(l)}}$$

$m \times m$的单位矩阵



一、Starting from neural nets

2. Backpropagation

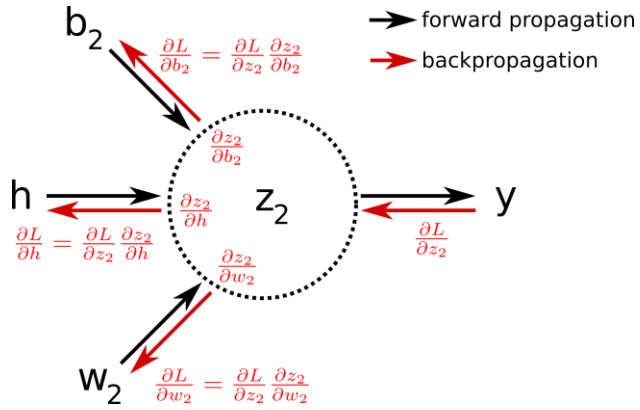


$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}).$$

（3）误差项 z与L, 记作 $\delta^{(l)}$ 代替 $\dfrac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$

表示第 l 层神经元对最终损失的影响 .



根据链式法则：

$$\delta^{(l)} \triangleq \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l+1)}}$$

$$= \mathrm{diag}(f_l'(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^{\mathrm{T}} \cdot \delta^{(l+1)}$$

$$= f_l'(\mathbf{z}^{(l)}) \odot \left((W^{(l+1)})^{\mathrm{T}} \delta^{(l+1)}\right),$$

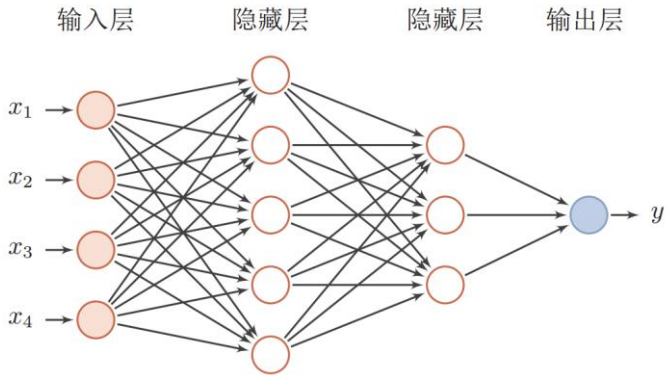
一、Starting from neural nets

2. Backpropagation

（4）归纳

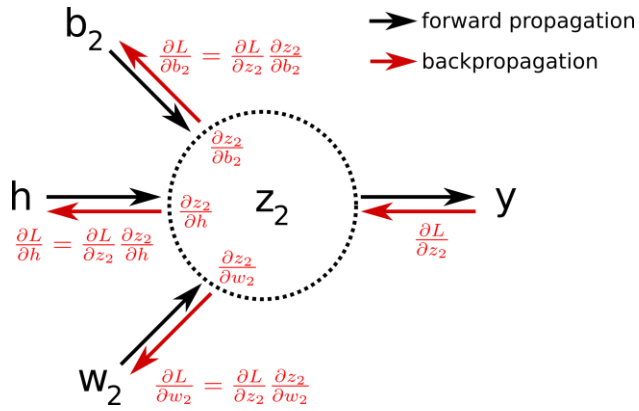
$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial W_{ij}^{(l)}} = \mathbb{I}_i (a_j^{(l-1)})^{\mathrm{T}} \delta^{(l)} = \delta_i^{(l)} a_j^{(l-1)}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$



$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}).$$



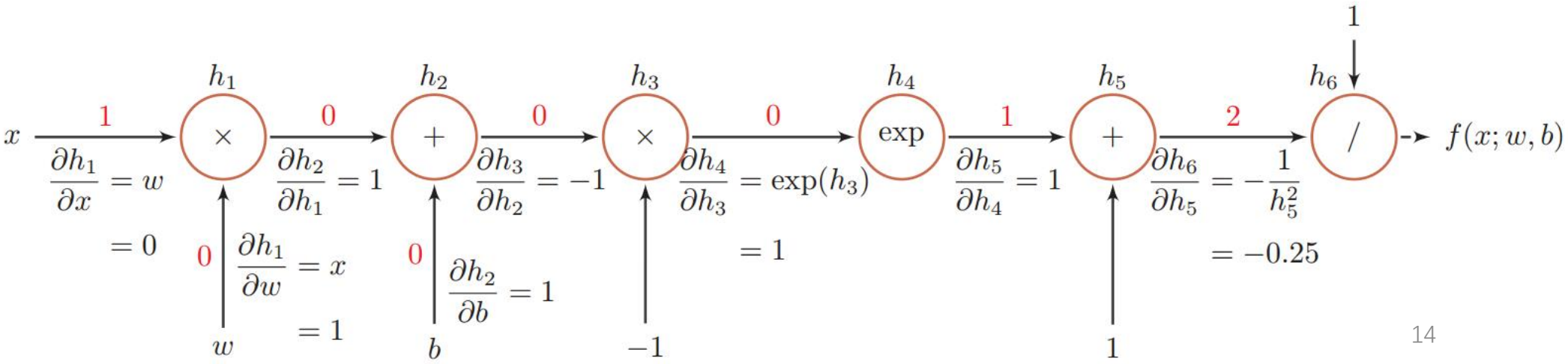
一、Starting from neural nets

2. Backpropagation

自动微分：让深度学习框架自动计算梯度

基本原理：所有的数值计算可以分解为一些基本操作，
+, -, ×, /, 初等函数exp, log, sin, cos等

- Example: $$f(x;w,b)=\frac{1}{\exp\big(-(wx+b)\big)+1}$$
- 将其分解为一系列的基本操作，并构成一个计算图（Computational Graph）



一、Starting from neural nets

2. Backpropagation

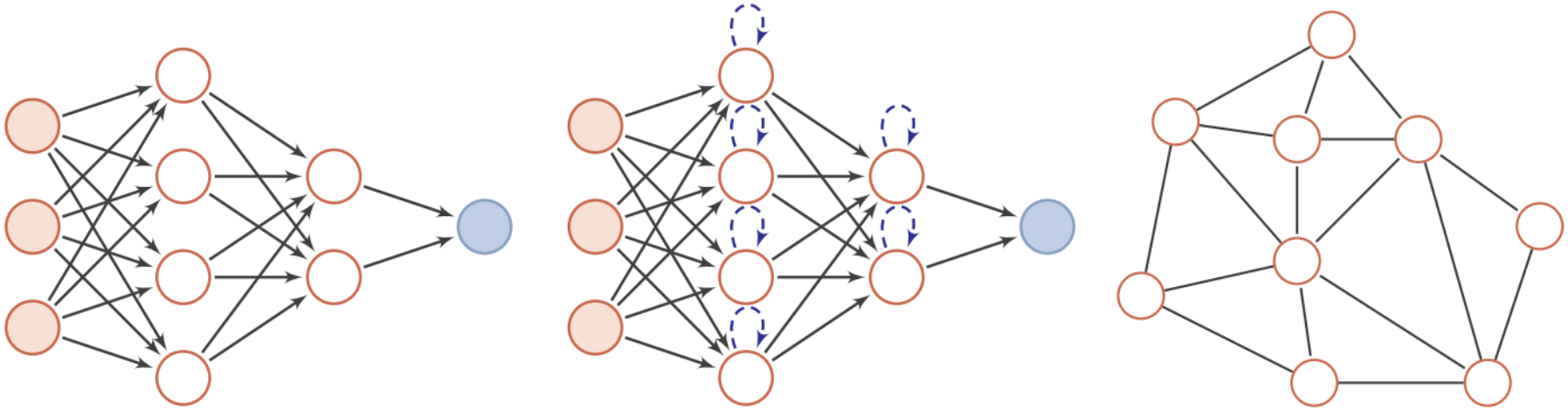
Every thing in computing graph should be differentiable.

- 基本数值计算操作：+, -, ×, /, 初等函数exp, log, sin, cos等
- 其他基本操作
- 激活函数&输出函数

激活函数	函数	导数
Logistic 函数	$f(x) = \frac{1}{1+\exp(-x)}$	$f'(x) = f(x)\big(1 - f(x)\big)$
Tanh 函数	$f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \max(0, x)$	$f'(x) = I(x > 0)$
ELU	$f(x) = \max(0, x) + \min\big(0, \gamma(\exp(x) - 1)\big)$	$f'(x) = I(x > 0) + I(x \leq 0) \cdot \gamma \exp(x)$
SoftPlus 函数	$f(x) = \log\big(1 + \exp(x)\big)$	$f'(x) = \frac{1}{1+\exp(-x)}$

一、Starting from neural nets

3. Last generation of neural nets



(a) 前馈网络　　(b) 反馈网络　　(c) 图网络

大部分的神经网络模型都属于前馈网络.

一、Starting from neural nets

3. Last generation of neural nets

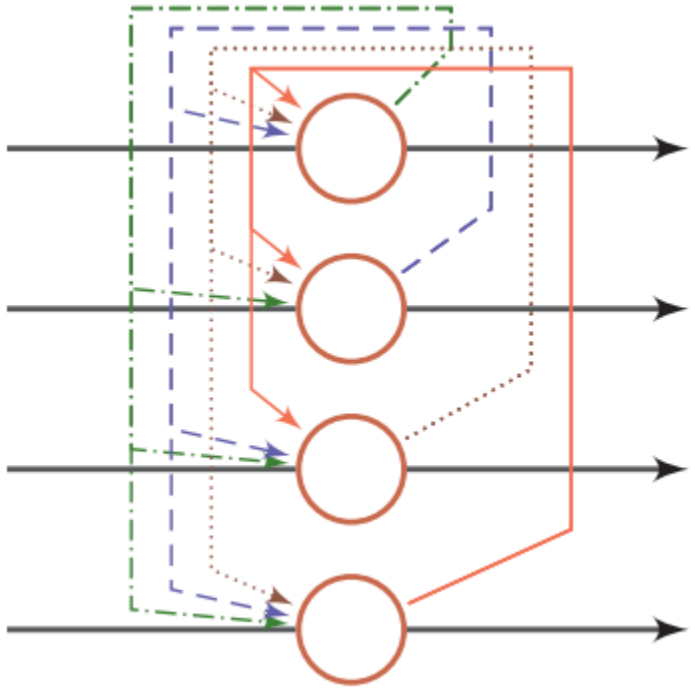
- 前馈神经网络
  - RBF, Radial Basis Function,径向基函数
  - Boltzman 玻尔兹曼机
  - RBM, Restricted Boltzmann Machine,受限玻尔兹曼机
  - DBM , Deep Boltzmann Machine,深度波尔茨曼机
  - BN, Belief Network, 信念网络
  - DBN, Deep Belief Network,深度置信网络

一、Starting from neural nets

3. Last generation of neural nets

- 反馈神经网络

  - Elman神经网络
  - Hopfield网络
  - RNN循环神经网络



二、Neural models for NLP

- CNN
- RNN
- Attention mechanism
- Sequence-to-sequence
- AutoEncoder
- GAN
- Graph models
- Embeddings
- Reinforcement learning
- Other networks

二、Neural models for NLP

1. Convolutional Neural Network， CNN或ConvNet,卷积神经网络

一维卷积: 序列型数据，文本，声音信号
二维卷积：二维图像，二维结构数据
三维卷积：三位图像…

- 可定义卷积操作： $\mathbf{y} = \mathbf{w} \otimes \mathbf{x},$

$$y_t = \sum_{k=1}^{m} w_k \cdot x_{t-k+1}$$

W即为滤波器（Filter）或卷积核（Convolution Kernel）

W=[w_1, …. , w_k , …]

二、Neural models for NLP

1. Convolutional Neural Network， CNN或ConvNet,卷积神经网络

一维卷积

- Example 1

  假设有卷积核W=[w_1， w_2， w_3], 其中 w_1= 1, w_2= 1/2, w_3= 1/4

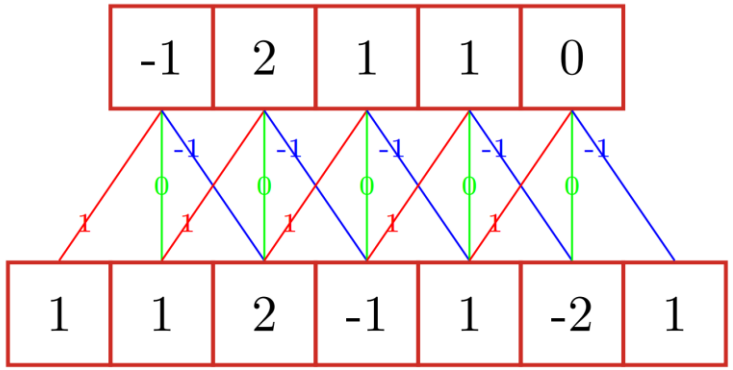
  则时刻t收到的信号 $y_t = 1 \times x_t + 1/2 \times x_{t-1} + 1/4 \times x_{t-2}$

  $$= w_1 \times x_t + w_2 \times x_{t-1} + w_3 \times x_{t-2}$$

  $$= \sum_{k=1}^{3} w_k \cdot x_{t-k+1}.$$

- Example 2

  假设卷积核W=[-1,0,1], 则卷积过程：



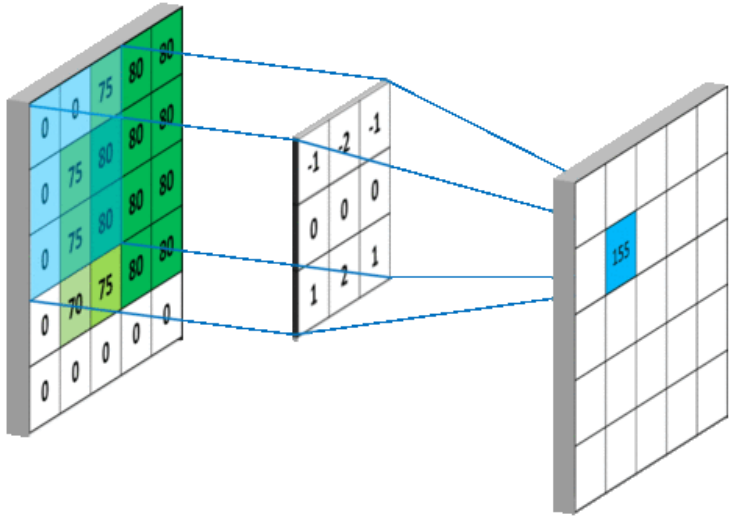
二、Neural models for NLP

1. Convolutional Neural Network， CNN或ConvNet,卷积神经网络

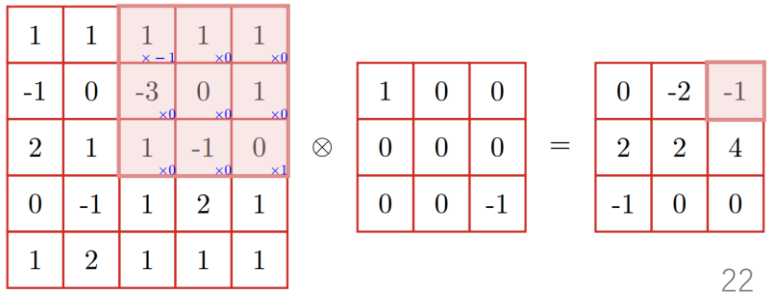
二维卷积

- 给定一个图像：$X \in \mathbb{R}^{M \times N}$
- 滤波器：$W \in \mathbb{R}^{m \times n}$
- 则卷积为：$$y_{ij} = \sum_{u=1}^{m} \sum_{v=1}^{n} w_{uv} \cdot x_{i-u+1, j-v+1}$$

➢ Example



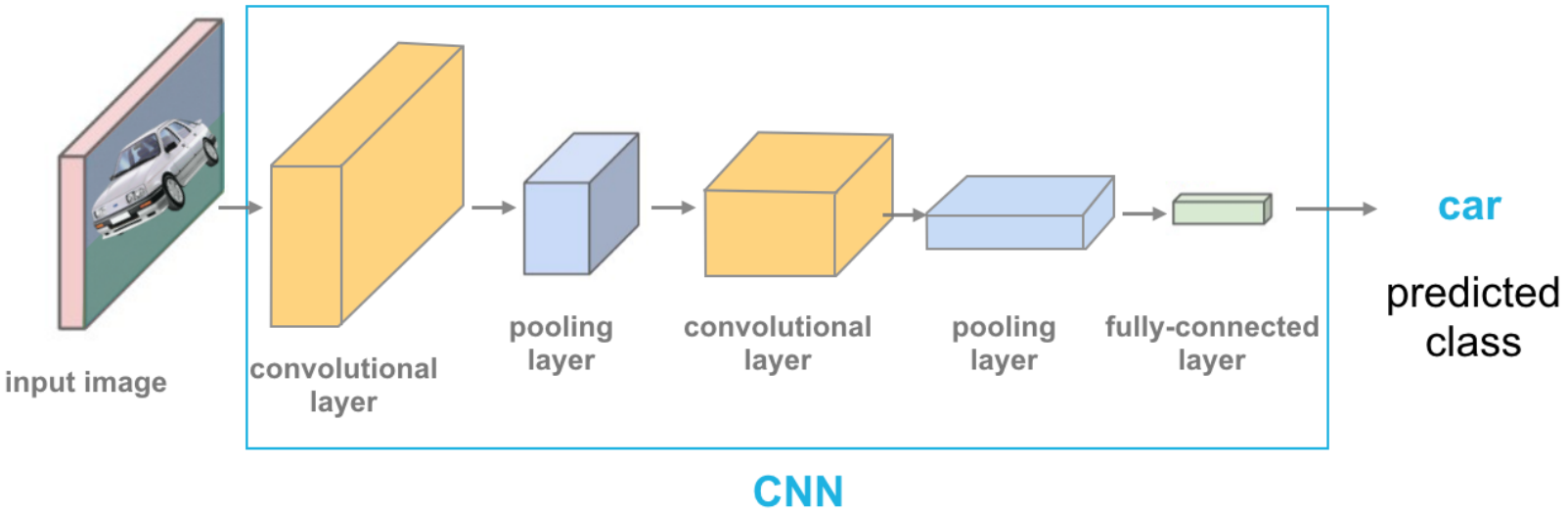
卷积操作：类似于生物学上感受野的机制



二、Neural models for NLP

1. Convolutional Neural Network， CNN或ConvNet,卷积神经网络

完整的CNN由卷积层、汇聚层和全连接层构成



其中汇聚层（Pooling Layer）/子采样层（Subsampling Layer）：

进行特征选择，降低特征数量，并从而减少参数数量。

二、Neural models for NLP

2. Recurrent Neural Network， RNN，循环神经网络

时序数据：当前时刻的信息和其过去一段时间的信息关联紧密，故需要一种记忆能力.

- 普通前馈网络为静态网络，不具备记忆能力.
- RNN具有短期记忆能力

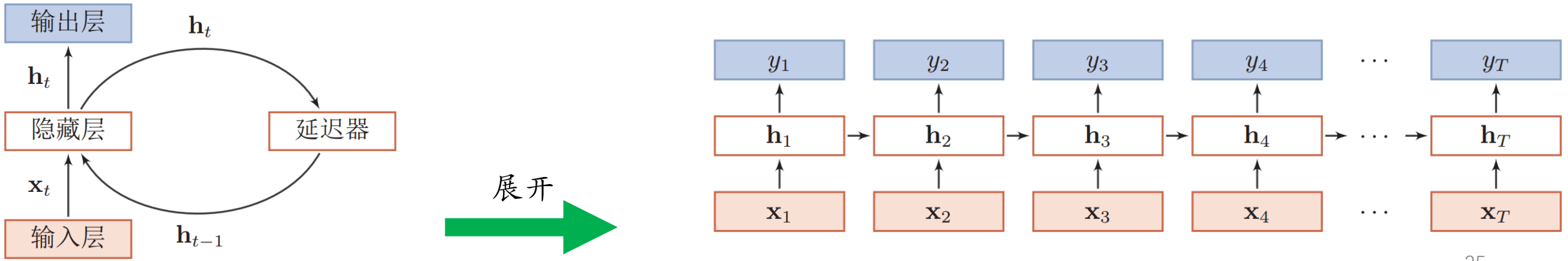
➢ 一句话：RNN的当前输出不仅仅与当前输入有关，还与过去状态有关。

二、Neural models for NLP

2. Recurrent Neural Network

- Vanilla RNN
  - 给定一个输入序列 $\mathbf{x}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$
  - RNN工作原理：

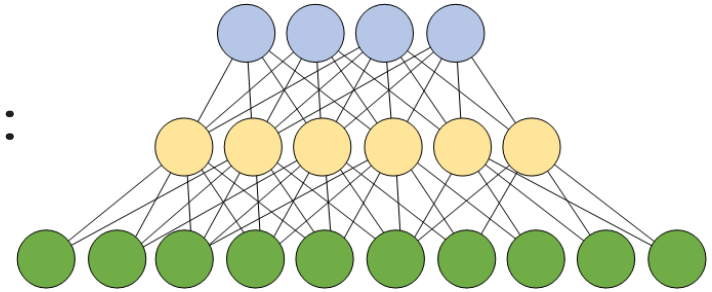
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$



二、Neural models for NLP

2. Recurrent Neural Network

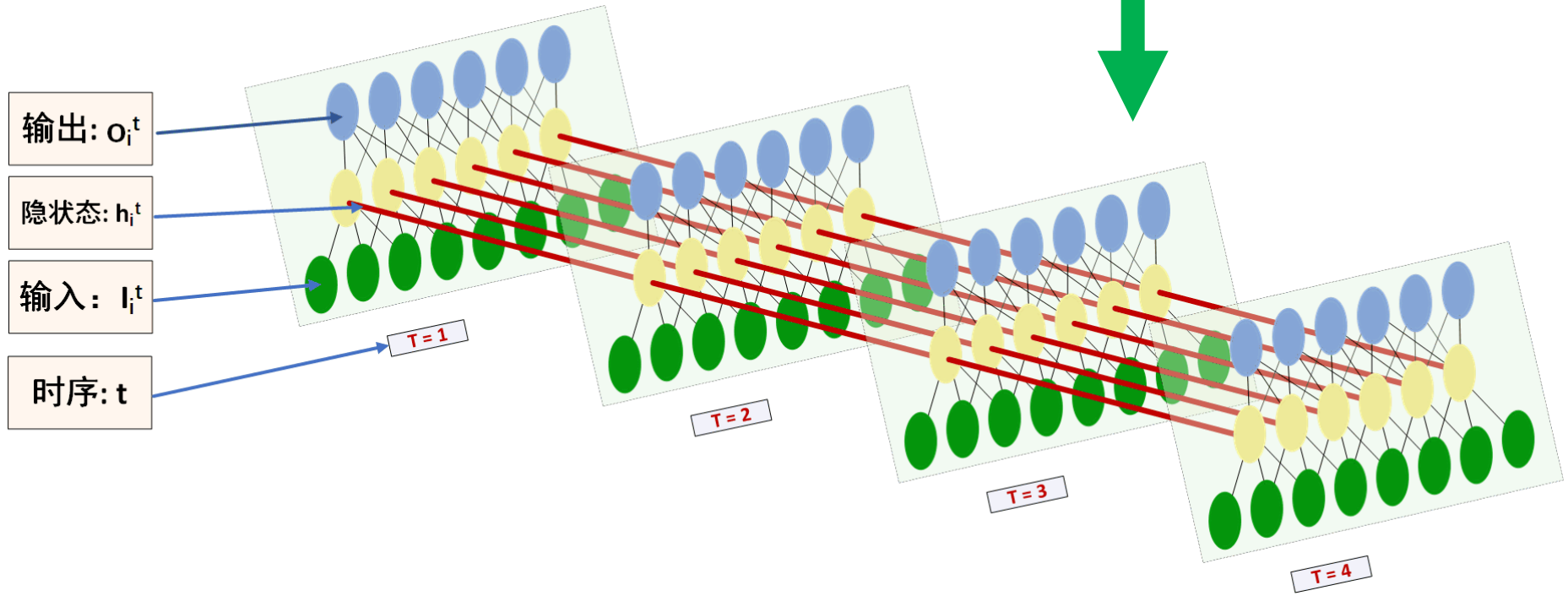
MLP：



依然难以理解？ 与MLP什么关系？

扩展到多个时序上

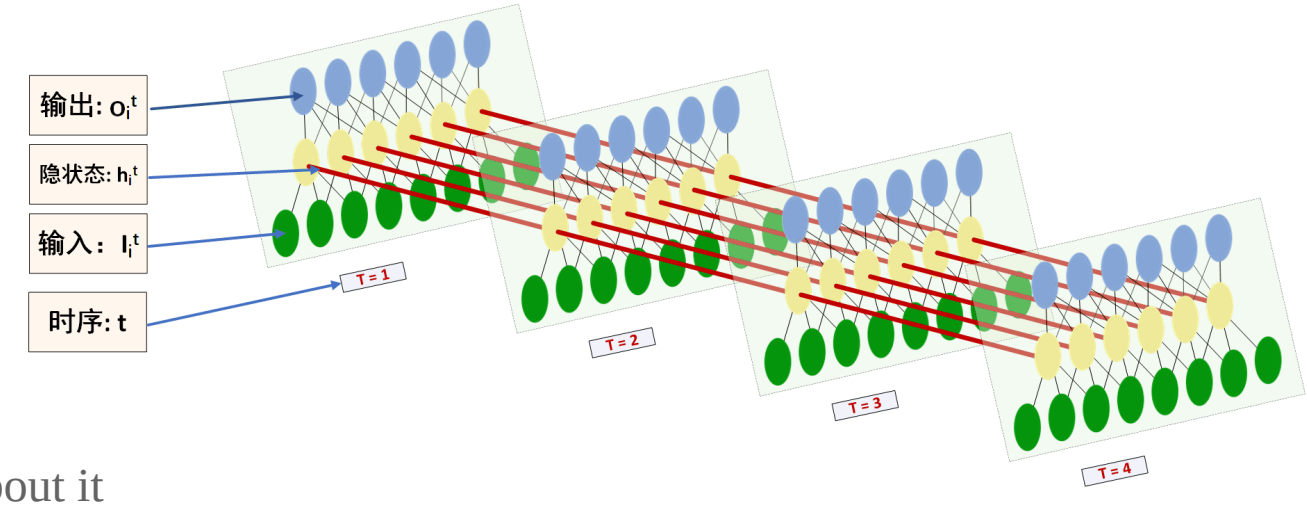
RNN：



一个时序下就是一个基本的MLP，只不过t个MLP共享了参数. 我们称这个MLP为RNN cell.

二、Neural models for NLP

2. Recurrent Neural Network



➢ Example

- 输入句子: I am feeling quite sick about it
  共7个词，所以RNN的时序长度t为7.

- 句子对应的embeddings：

$$\begin{bmatrix} 0.341 & 0.133 & 0.011 & \dots \\ 0.435 & 0.081 & 0.501 & \dots \\ 0.013 & 0.958 & 0.121 & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

  共7列，其中每一列对应某个词的词向量，每列的长度(词向量的维度)代表MLP的输入size.

- 运行RNN：每个时序t下，MLP以对应的t-th 词向量为输入，输出对应的表征向量**o**_i，同时其隐状态h_t将会传递到下一个时序(t+1)进一步进行运算。每一步都能输出一个表征向量**o**_i，所以对于本句子，共能输出7个表征向量，以及最终时序下的隐状态向量h_t。

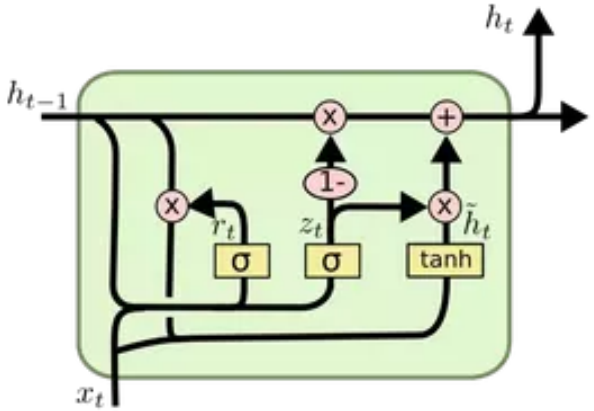
- 后续：得到t个表征向量**o**_i和一个隐状态向量h_t，进一步通过分类器可以得到分类结果.

二、Neural models for NLP

2. Recurrent Neural Network

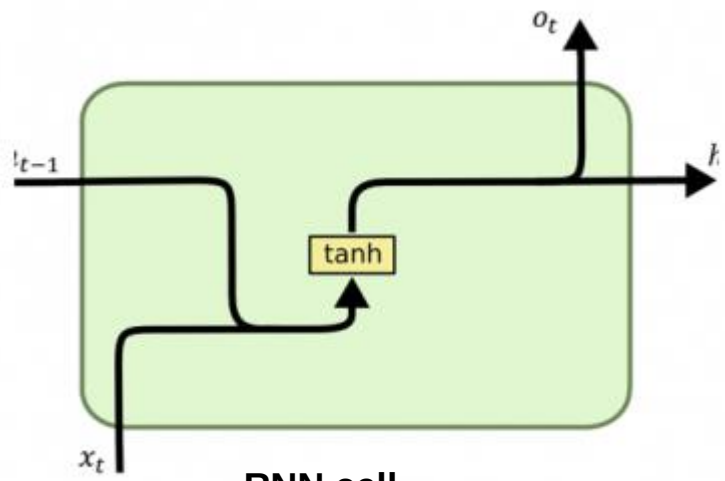
常见RNN模型：

- Vanilla RNN
- RNN + gate mechanism:
  - ✓ GRU: Gated Recurrent Unit
  - ✓ LSTM: Long Short Term Memory



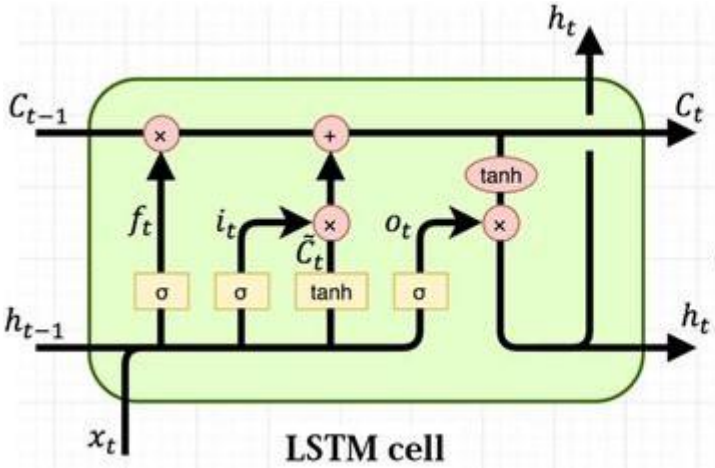
GRU cell

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



RNN cell

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



LSTM cell

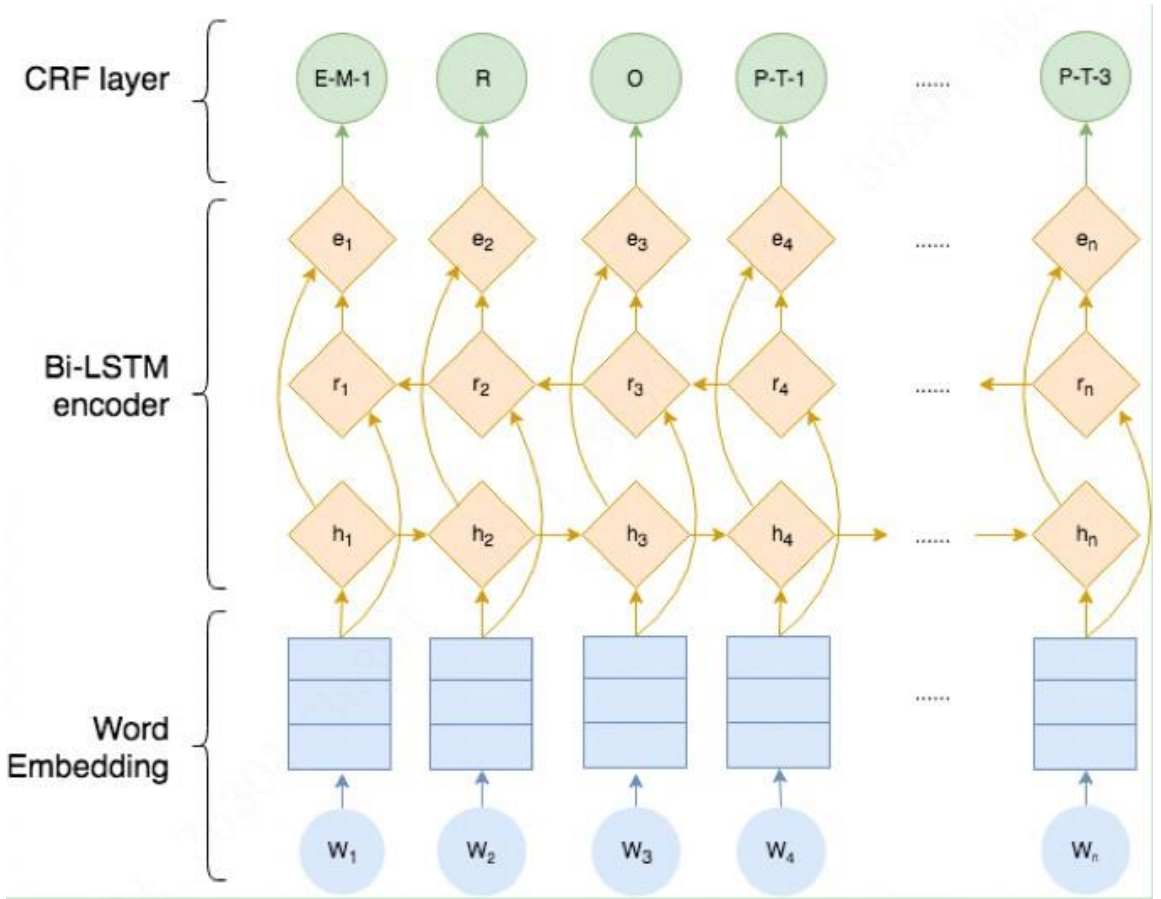
$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

二、Neural models for NLP

2. Recurrent Neural Network

De facto RNN模型：
Multi-layer BiLSTM

- 直接用最终output representation
  sentence-level classification
- 每一步的output representation
  token-level classification

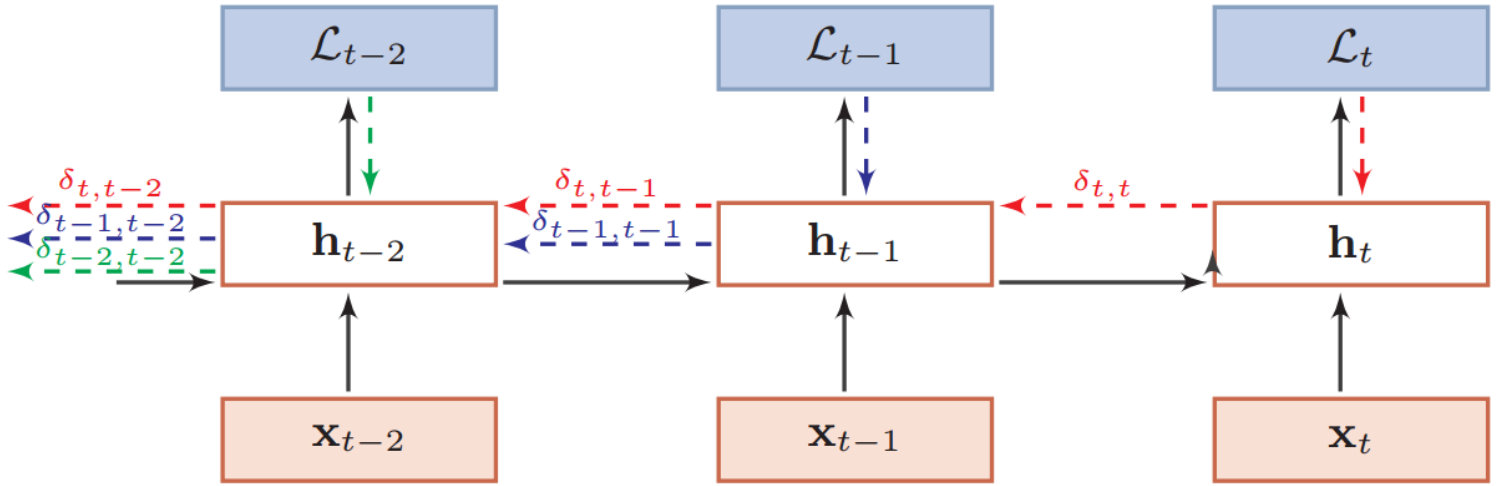


二、Neural models for NLP

2. Recurrent Neural Network

随时间反向传播（Backpropagation Through Time， BPTT）

将循环神经网络看作是一个展开的多层前馈网络，其中“每一层”对应循环网络中的“每个时刻”



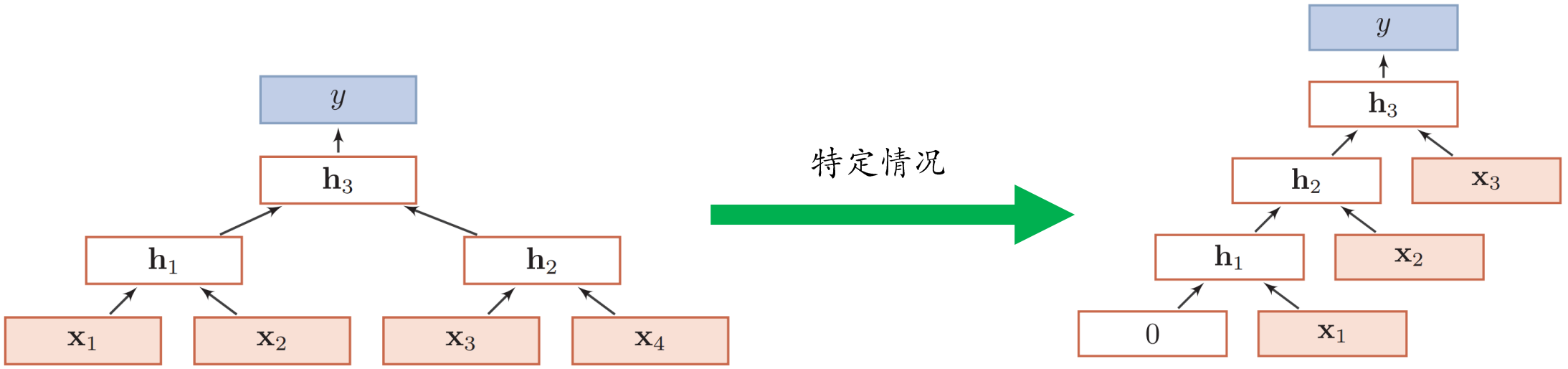
二、Neural models for NLP

2. Recurrent Neural Network

只有时序？Nope.

Recursive Neural Network,　RecurNN,递归神经网络

- Recurrent NN 的一般性扩展网络.



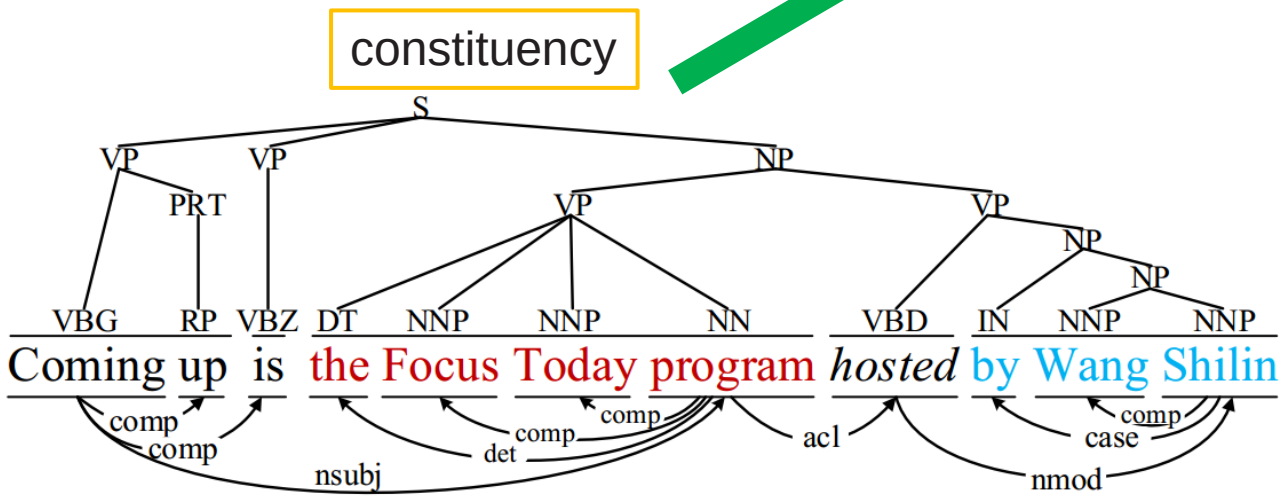
Recursive NN　　Recurrent NN

二、Neural models for NLP

2. Recurrent Neural Network

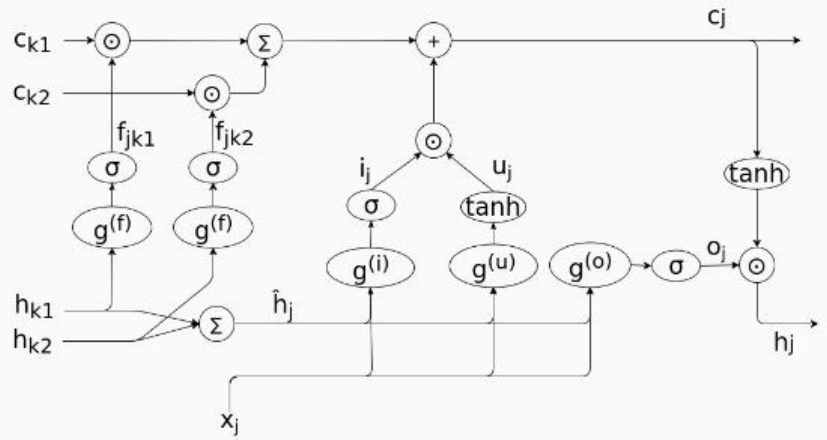
Tree structures in NLP

- Syntactic dependency
- Phrasal constituency



Child-sum tree LSTM

Children outputs and memory cells are summed

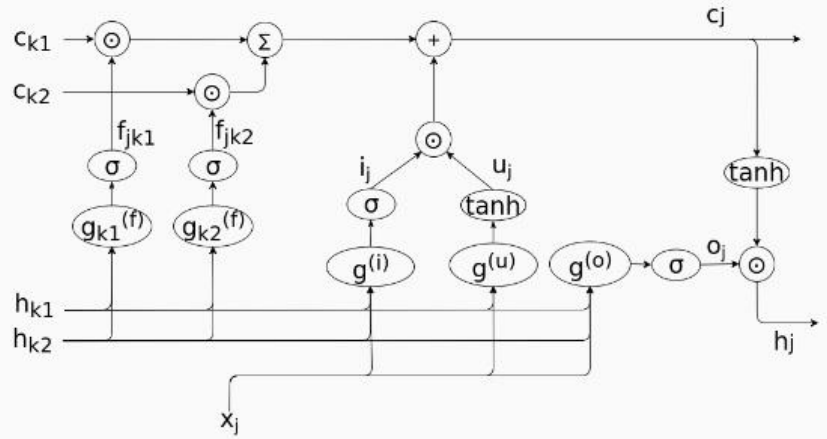


Child-sum tree LSTM at node j with children k_1 and k_2

10

N-ary tree LSTM

Given $g_k^{(n)}(x_t, h_{l_1}, \cdots, h_{l_N}) = W^{(n)}x_t + \sum_{l=1}^{N} U_{kl}^{(n)} h_{jl} + b^{(n)}$



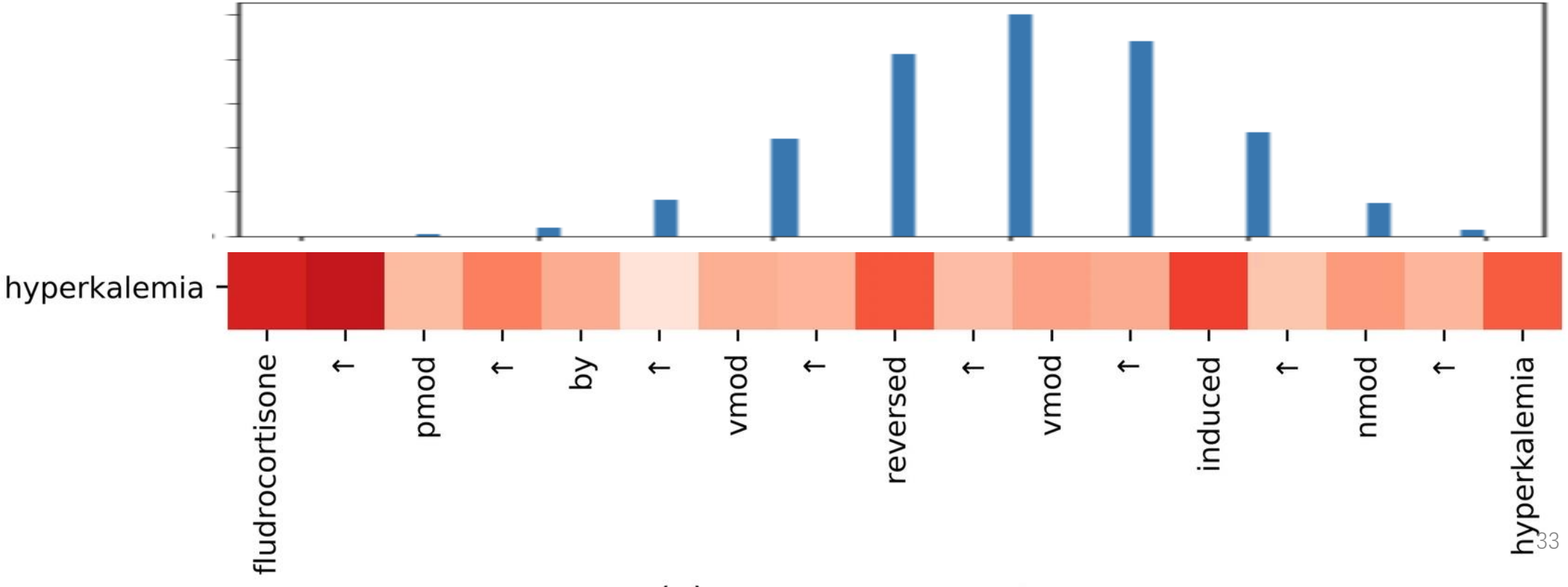
Binary tree LSTM at node j with children k_1 and k_2

12

二、Neural models for NLP

3. Attention mechanism

动机：不是所有的输入都有相同的作用，我们只让模型更关注于与任务关系最为紧密的元素即可，从而节省计算资源并且提升信息利用率。



二、Neural models for NLP

3. Attention mechanism

- 给定N 个输入信息 $X = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$，以及和任务相关的查询向量$\mathbf{q}$.
- 计算出N个元素的“任务贡献度分布”或“注意力分布”：

$$\begin{aligned}\alpha_i &= p(z = i|X, \mathbf{q}) \\ &= \text{softmax}\left(s(\mathbf{x}_i, \mathbf{q})\right) \\ &= \frac{\exp\left(s(\mathbf{x}_i, \mathbf{q})\right)}{\sum_{j=1}^{N} \exp\left(s(\mathbf{x}_j, \mathbf{q})\right)},\end{aligned}$$

$s(\mathbf{x}_i, \mathbf{q})$：注意力打分函数，衡量x与q的“匹配程度”

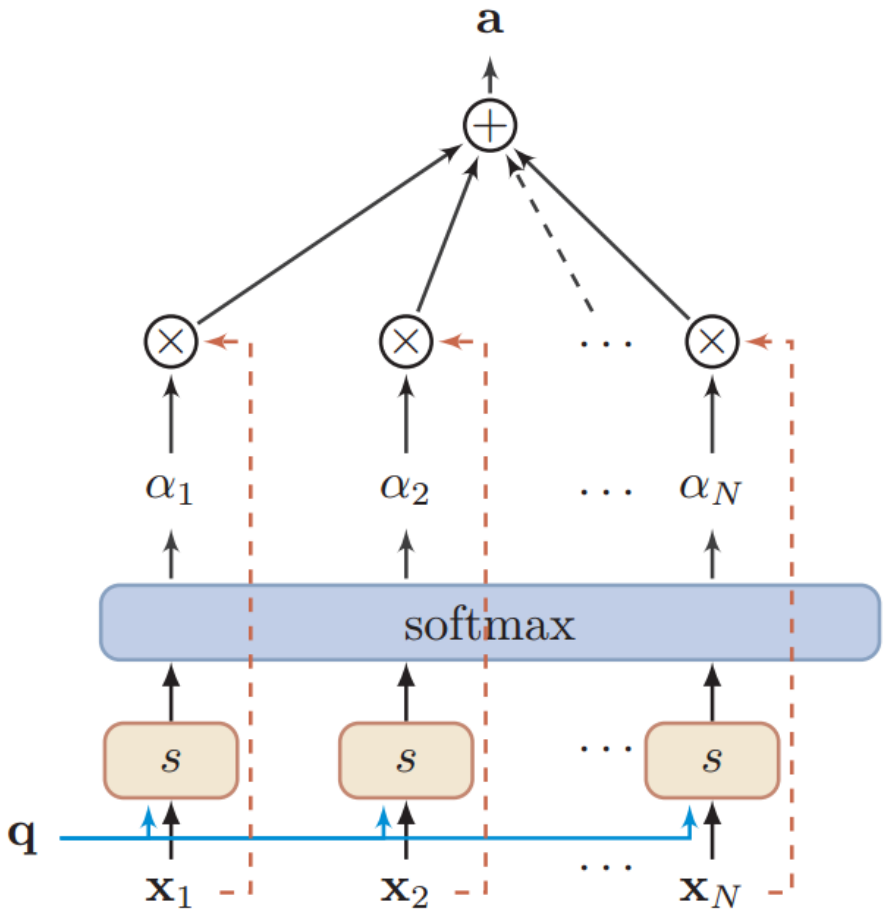
加性模型	$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{v}^{\mathrm{T}} \tanh(W\mathbf{x}_i + U\mathbf{q}),$
点积模型	$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{x}_i^{\mathrm{T}}\mathbf{q},$
缩放点积模型	$s(\mathbf{x}_i, \mathbf{q}) = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{q}}{\sqrt{d}},$
双线性模型	$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{x}_i^{\mathrm{T}} W\mathbf{q},$

二、Neural models for NLP

3. Attention mechanism

- 计算出N个元素表征的加权平均，为最终表征：

$$\begin{aligned}\mathbf{att}(X, \mathbf{q}) &= \sum_{i=1}^{N} \alpha_i \mathbf{x}_i, \\ &= \mathbb{E}_{z \sim p(z|X,\mathbf{q})}[\mathbf{x}].\end{aligned}$$



二、Neural models for NLP

3. Attention mechanism

- Attention类型

按信息选择：

- Hard Attention
- Soft Attention (Local)
- Sparse Attention
- Atrous Attention
- …

按结构：

- Task-specific Attention
- Self Attention
- Hierarchical Attention
- Multi-head Attention
- Attention over Attention
- …

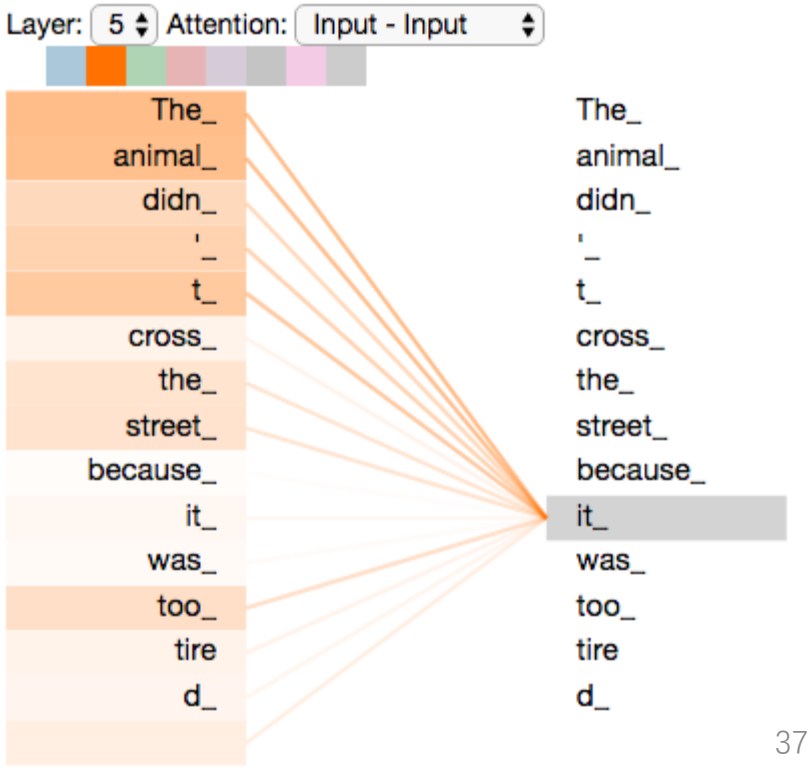
- Attention应用

- Seq-to-seq nets
- Memory nets
- Pointer Net
- …

二、Neural models for NLP

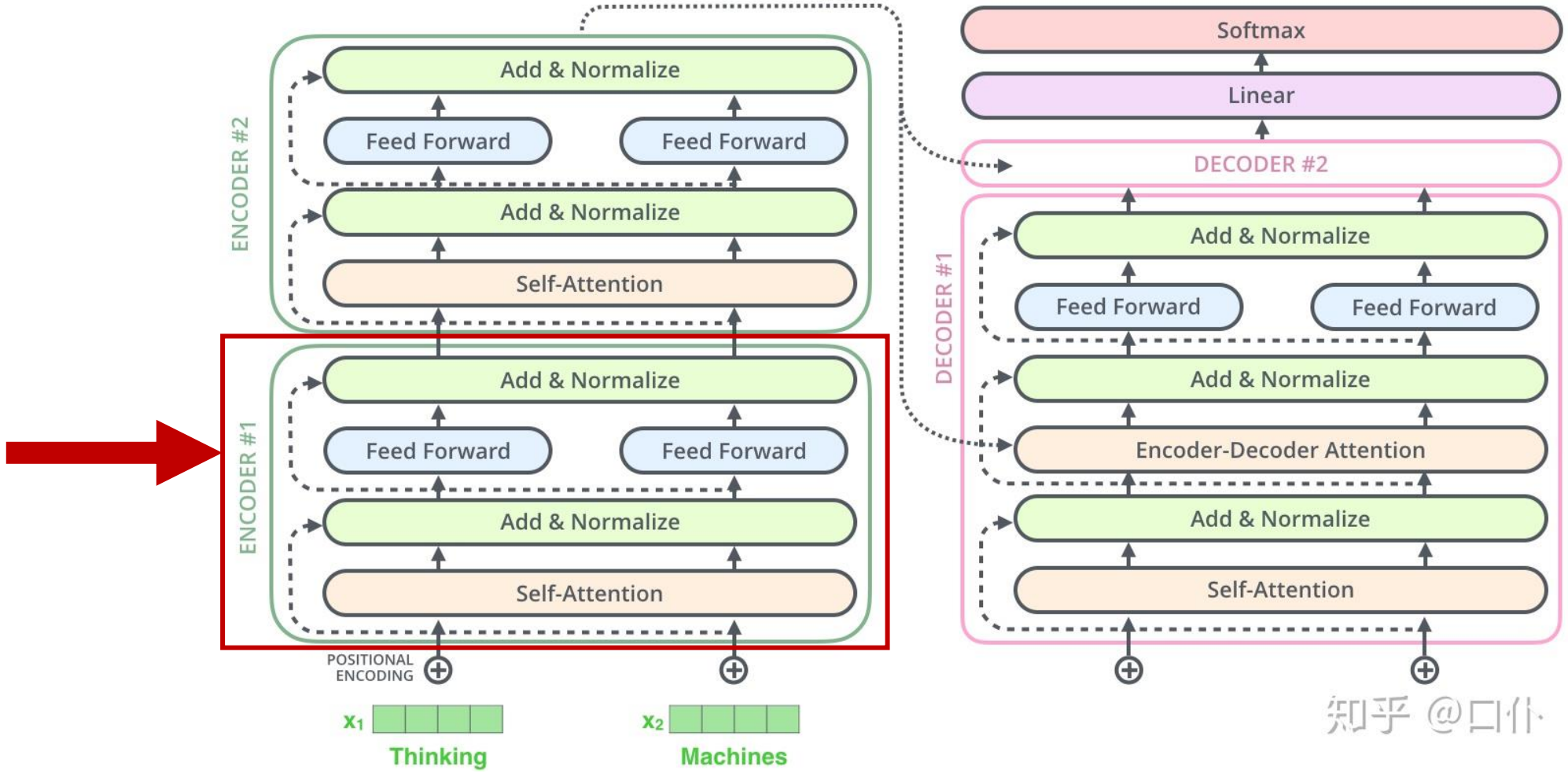
3. Attention mechanism

- Self-attention based **Transformer**
  - Self-attention： “动态”地对序列内部元素生成不同连接的权重，无需外部query参与。
  - Multi-head attention
  - Residuals connection
  - Positional Encoding
  - Deep Stacked
  - 相比RNN，有绝对的并行计算优势



二、Neural models for NLP

3. Attention mechanism

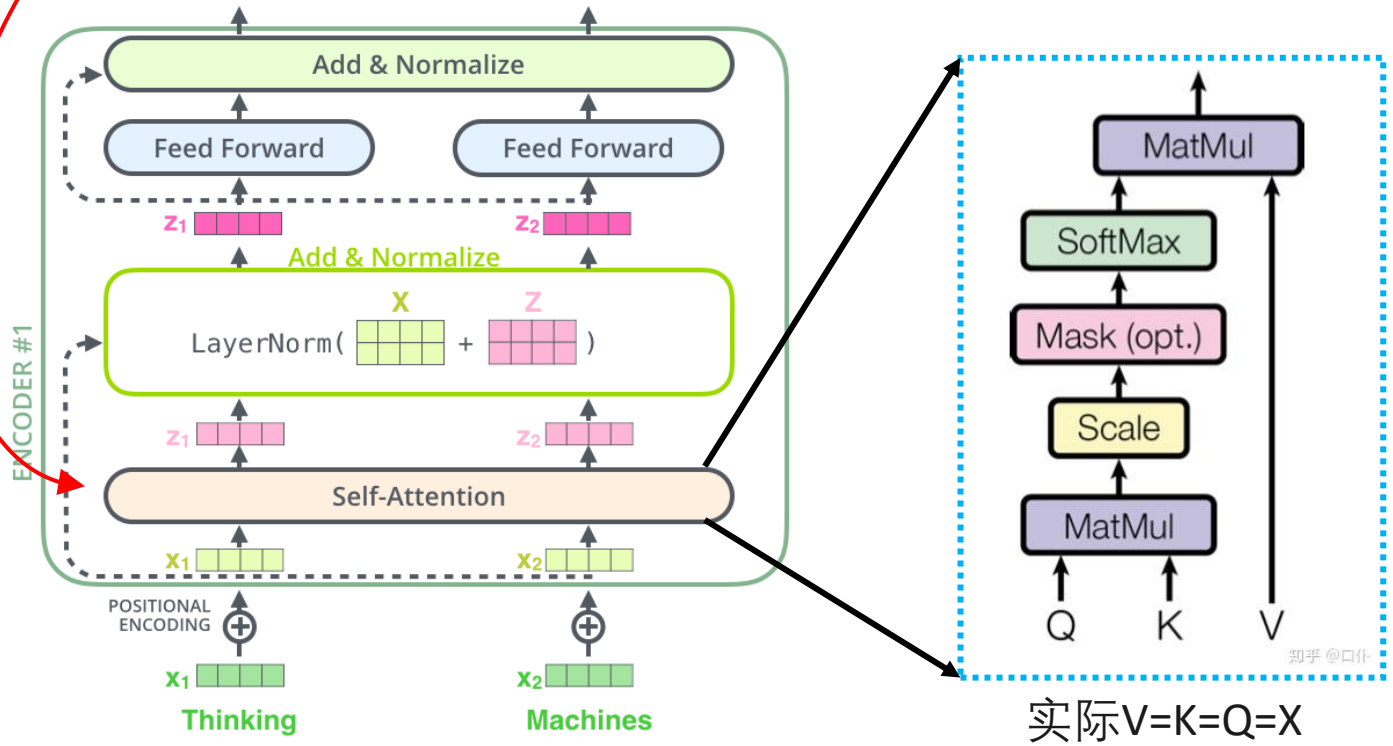


二、Neural models for NLP

3. Attention mechanism

Multi-Head Attention：利用多个查询$Q = [\mathbf{q}_1, \cdots, \mathbf{q}_M]$，来平行地计算从输入信息中选取多个信息，每个注意力关注输入信息的不同部分。

$$\mathbf{att}\Big((K,V),Q\Big) = \mathbf{att}\Big((K,V),\mathbf{q}_1\Big) \oplus \cdots \oplus \mathbf{att}\Big((K,V),\mathbf{q}_M\Big),$$



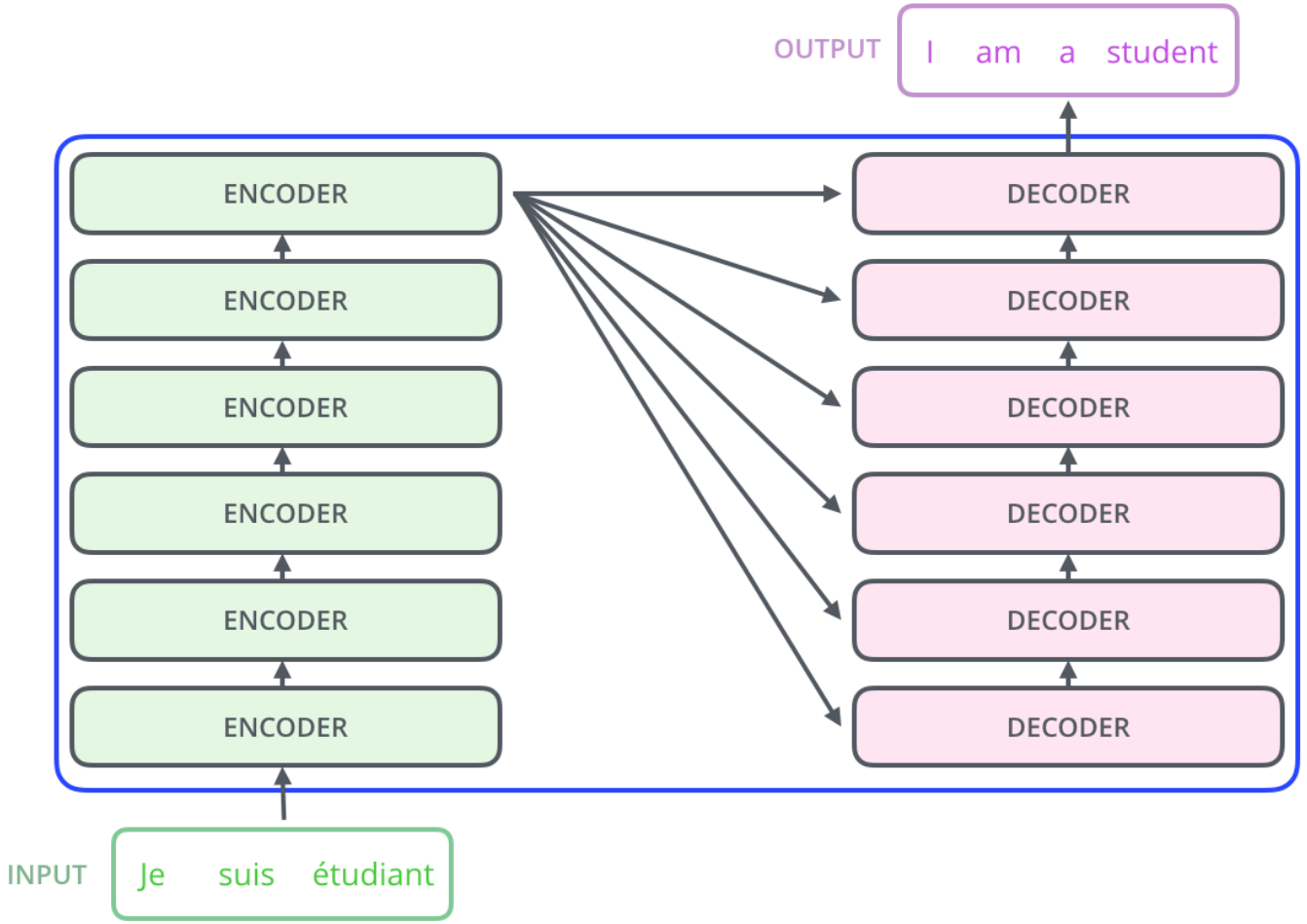
$$\text{Attention}\,(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

实际V=K=Q=X

二、Neural models for NLP

3. Attention mechanism

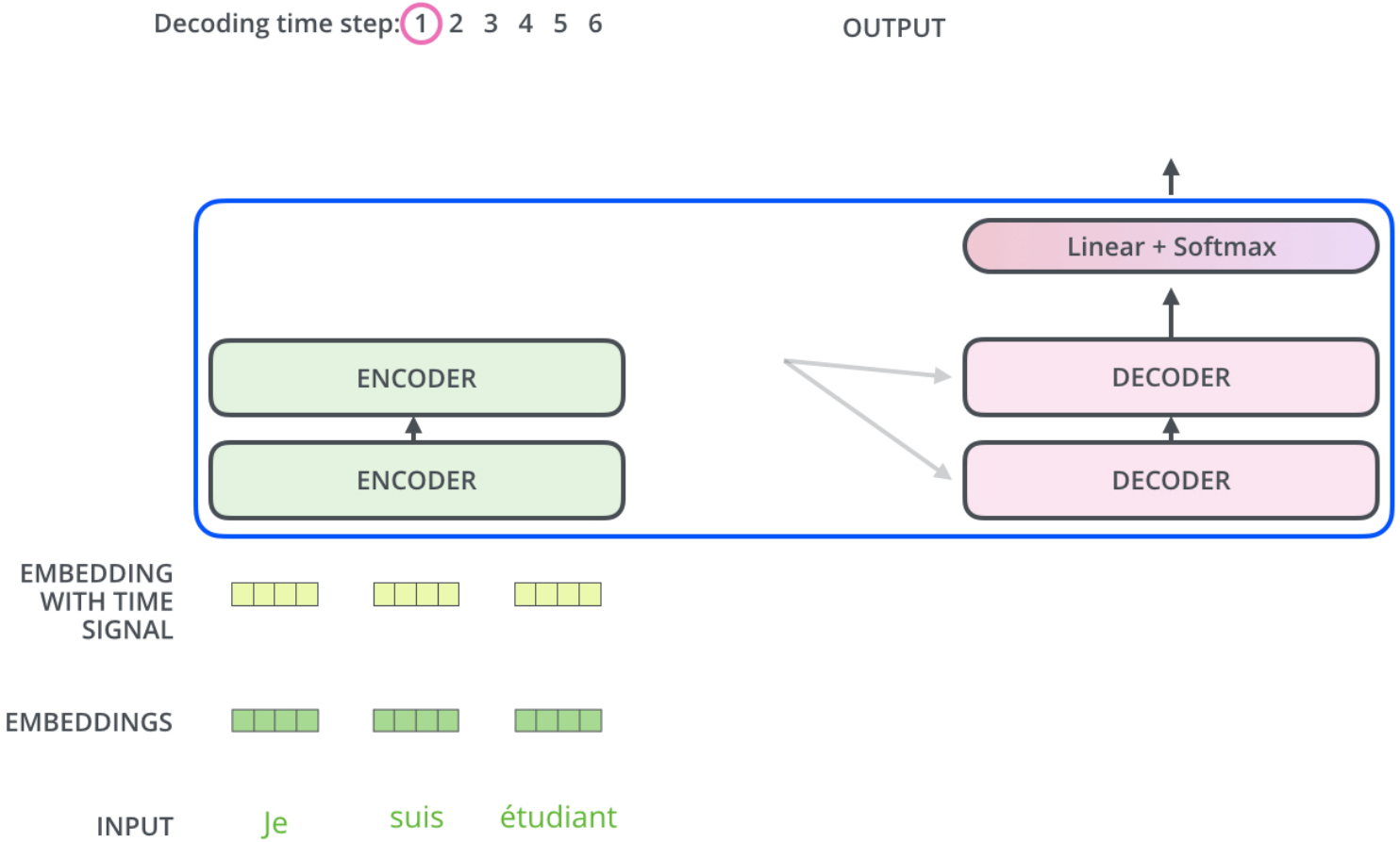
Transformer in encoder-decoder architecture



二、Neural models for NLP

3. Attention mechanism

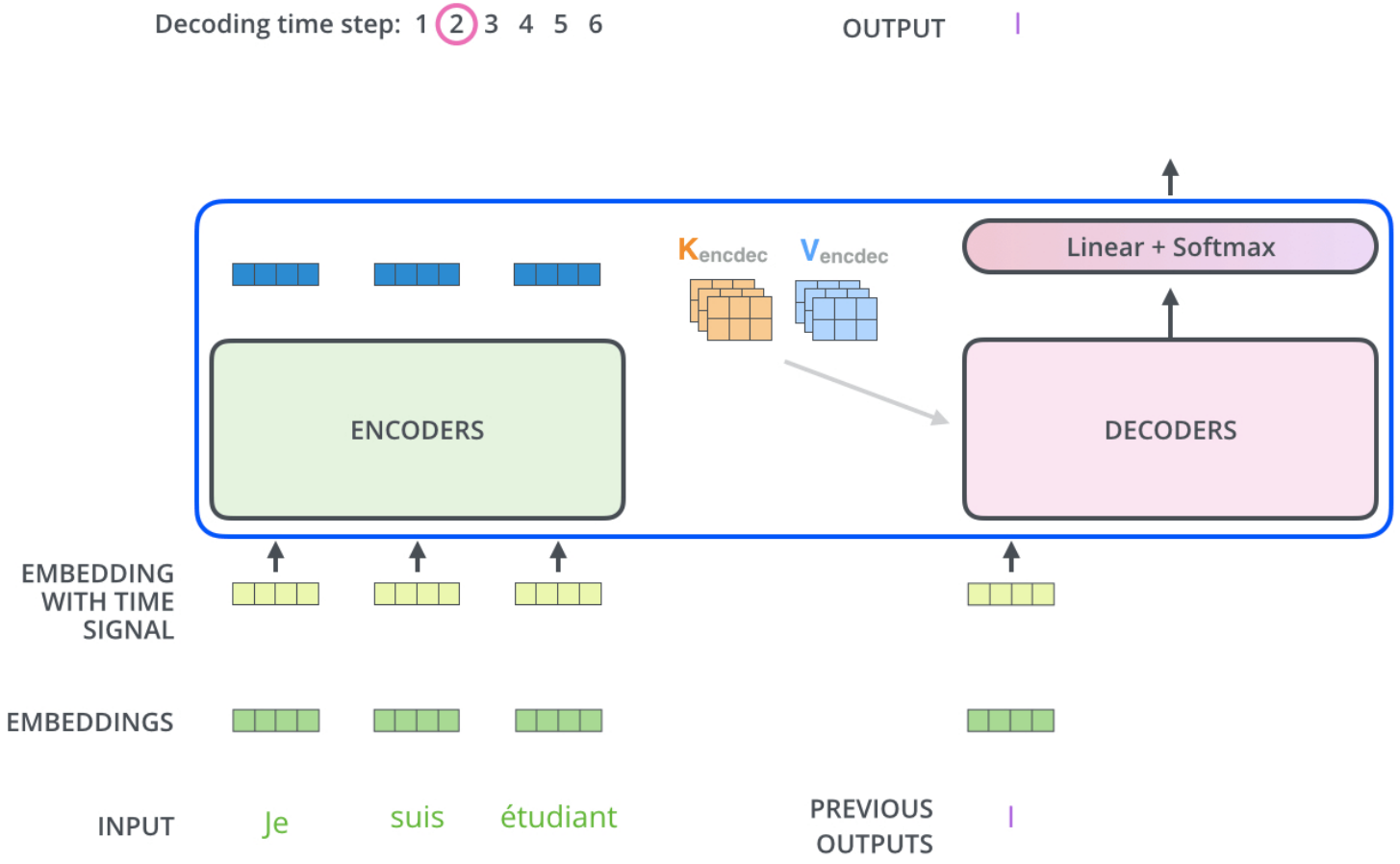
Transformer in encoder-decoder architecture



二、Neural models for NLP

3. Attention mechanism

Transformer in encoder-decoder architecture

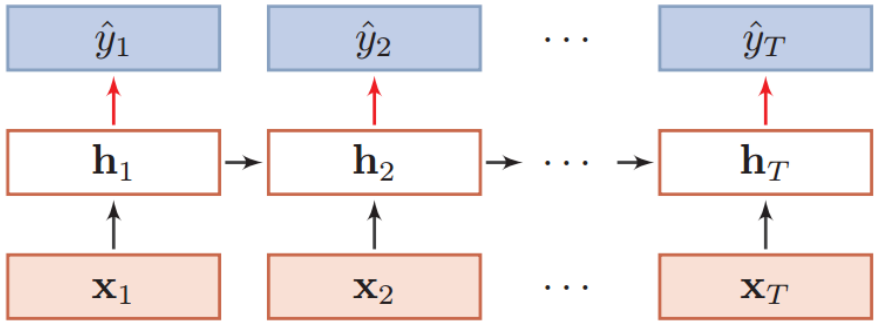


二、Neural models for NLP

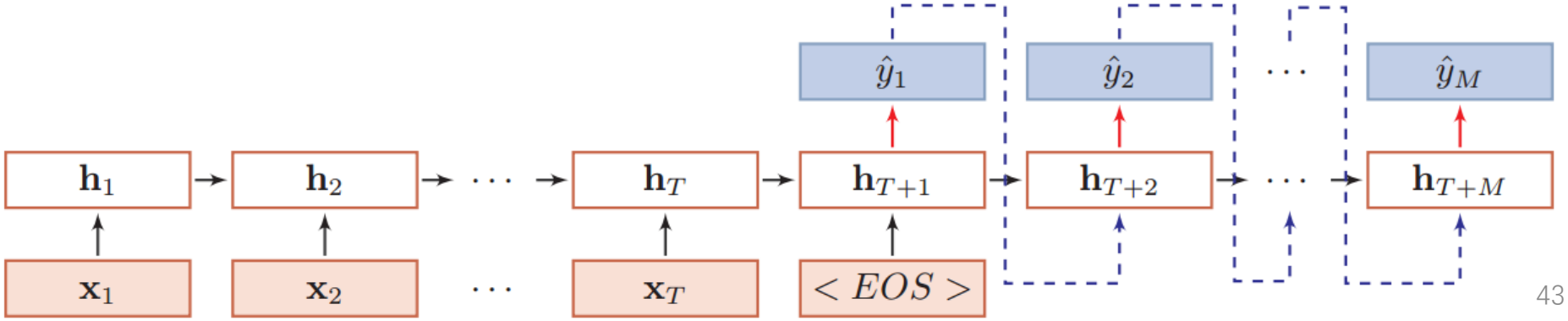
4. Sequence-to-sequence

Sequence-to-sequence (seq2seq)即一种异步序列学习模式，或编码器-解码器（Encoder-Decoder）结构.

- In RNN: 输入序列和输出序列有严格的对应关系，保持相同的长度, 即同步序列到序列学习



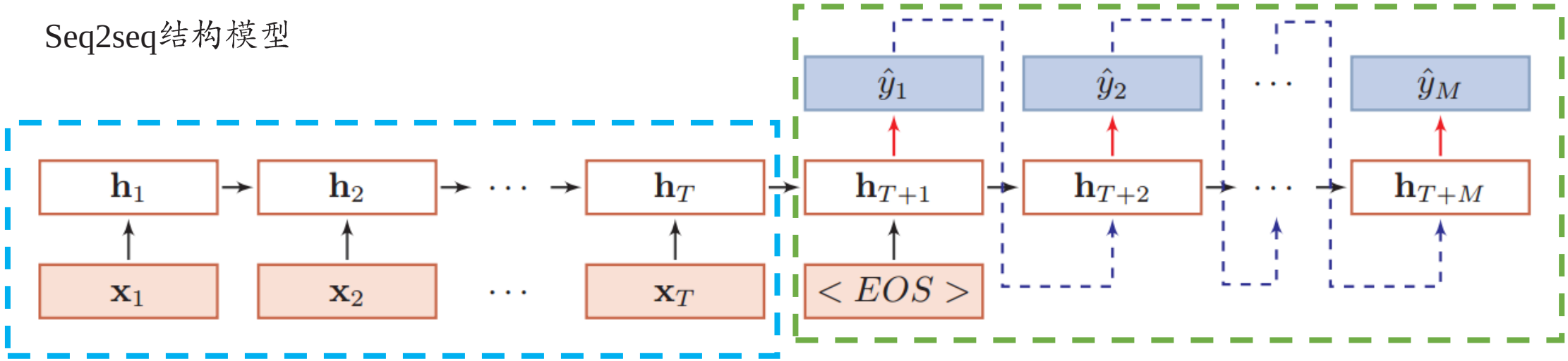
- In Sequence-to-sequence: 输入序列和输出序列不需要有严格的对应关系，也不需要保持相同的长度



二、Neural models for NLP

4. Sequence-to-sequence

Seq2seq结构模型



编码器，Encoder

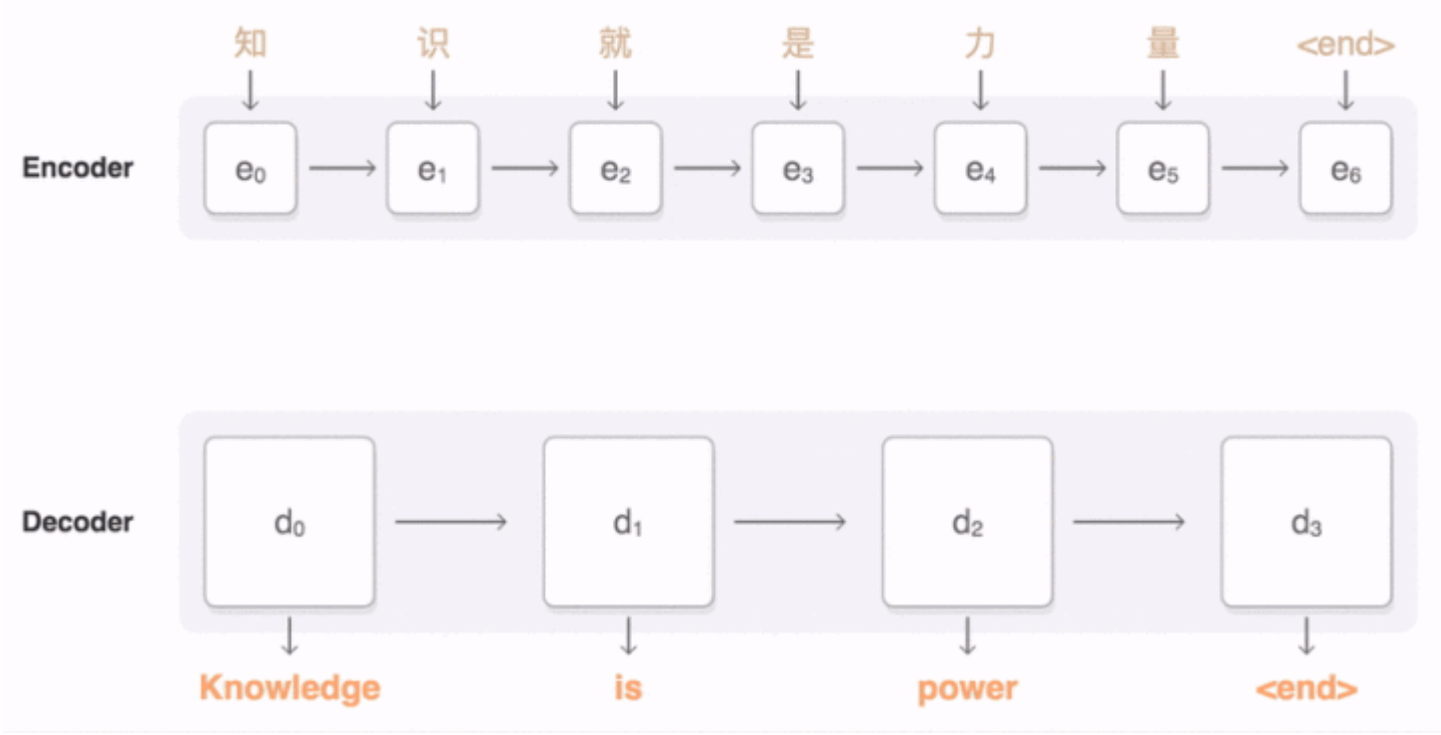
解码器，Decoder: 通常采用非线性的自回归模型

- ✓ RNN, GRU, LSTM
- ✓ Transformer
- ✓ …

二、Neural models for NLP

4. Sequence-to-sequence

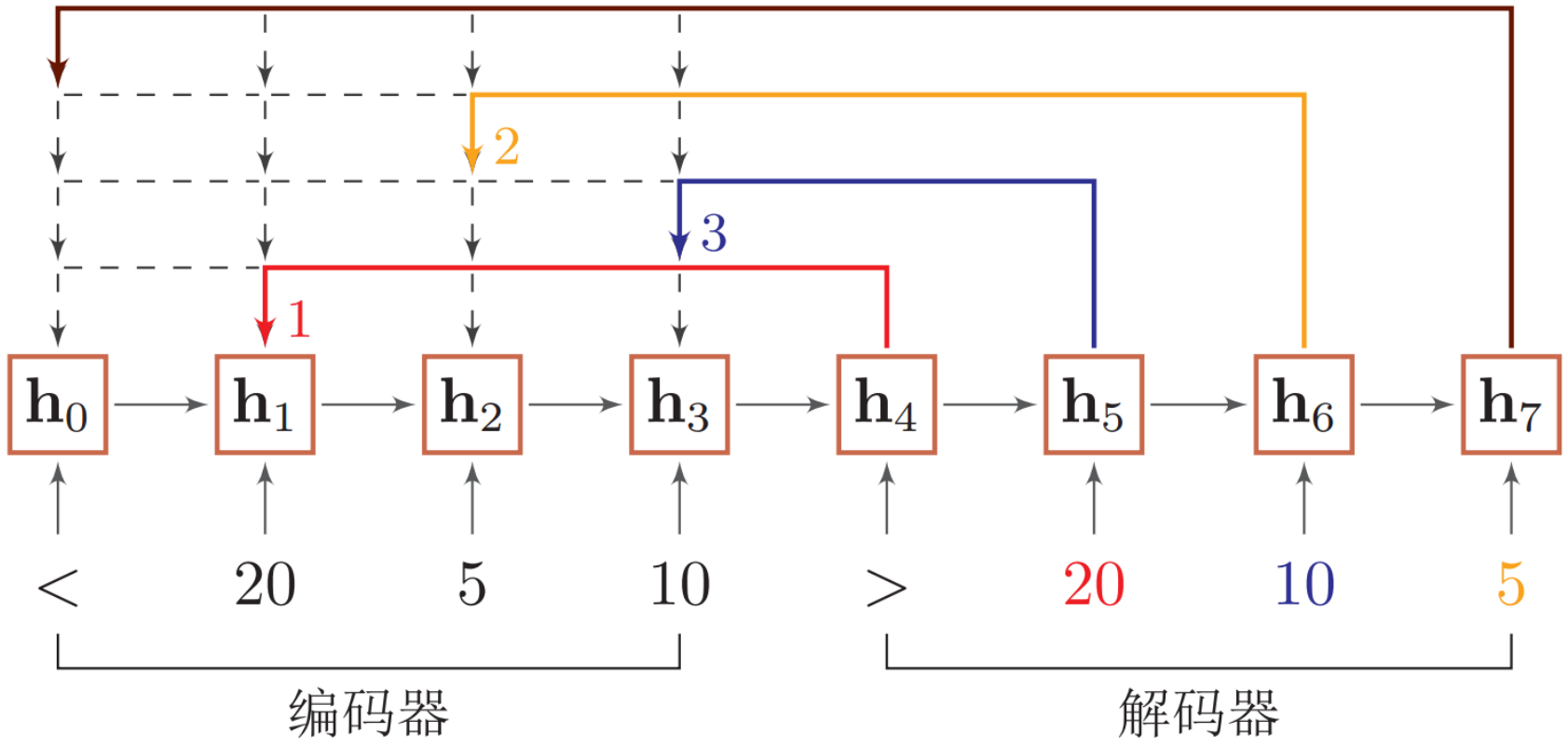
Attention-based Seq2seq model



二、Neural models for NLP

4. Sequence-to-sequence

Pointer networks

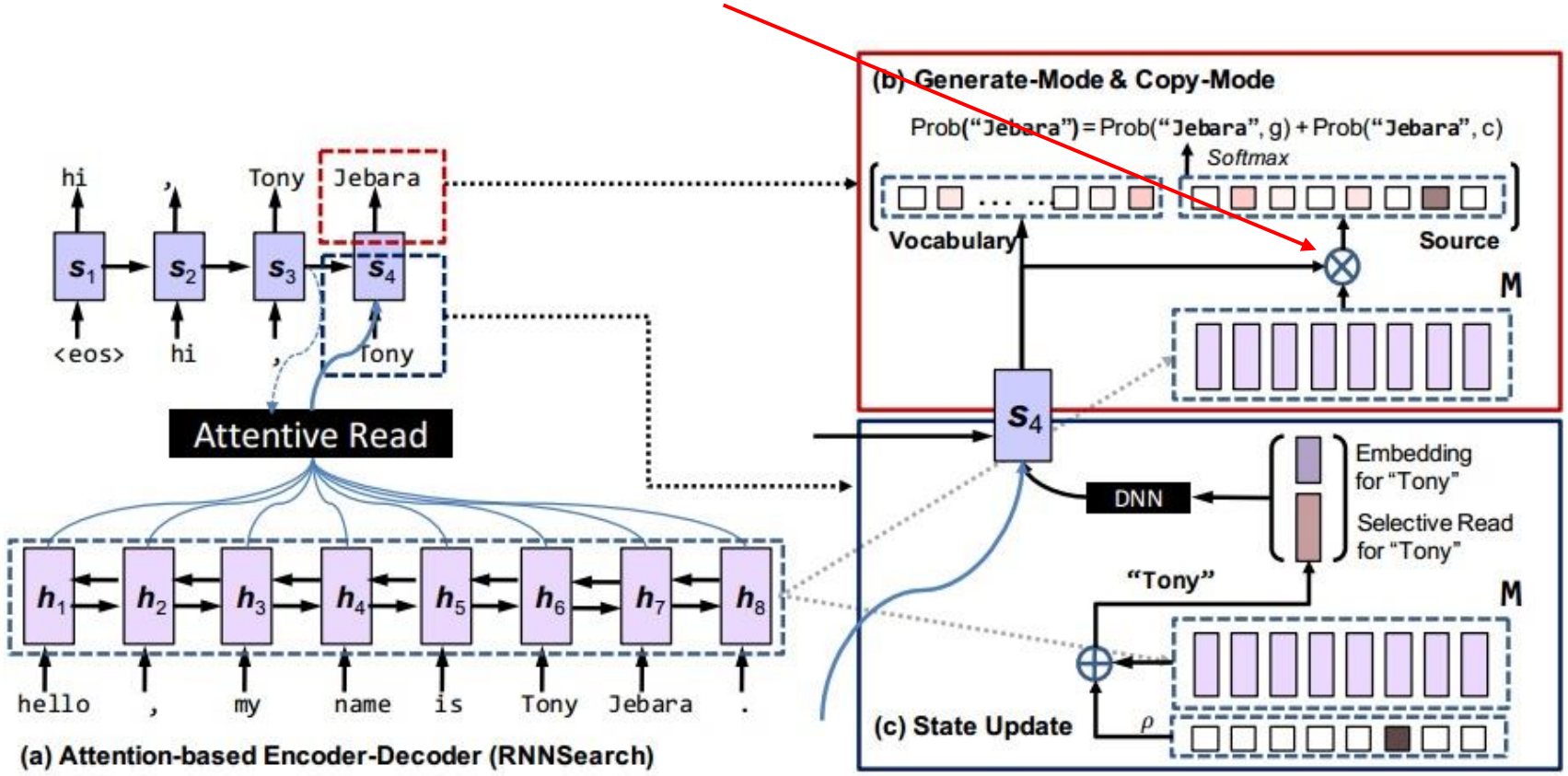


二、Neural models for NLP

4. Sequence-to-sequence

Copy networks: copy from source

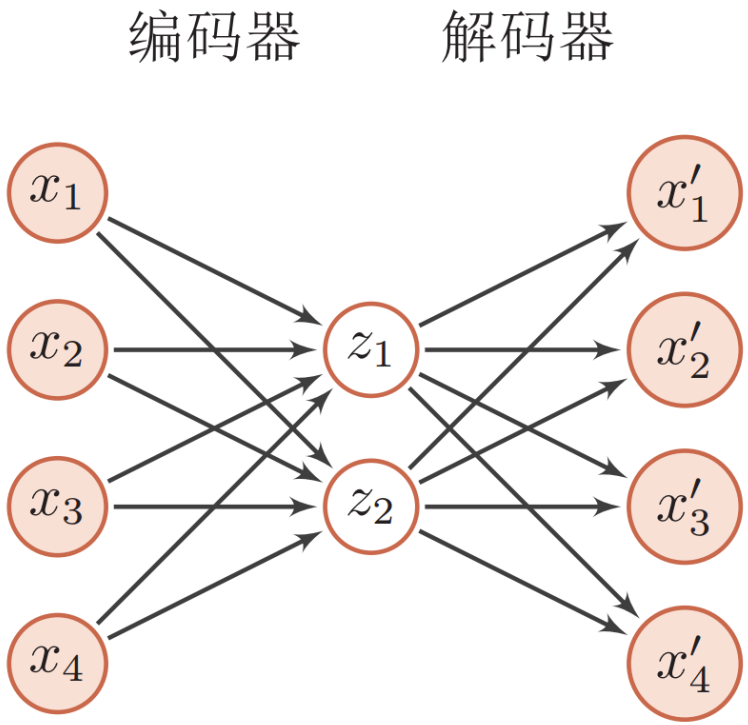
Gate 决定是“粘贴”还是选择“生成”， **generate-mode** or **copy-mode**



二、Neural models for NLP

5. AutoEncoder

AutoEncoder, 自编码器，是通过无监督的方式来学习一组数据的有效编码（或表示）的神经网络模型。



编码阶段：$\mathbf{z} = s(W^{(1)}\mathbf{x} + b^{(1)})$

解码阶段：$\mathbf{x}' = s(W^{(2)}\mathbf{z} + b^{(2)})$

学习目标：最小化重构误差：

$$\mathcal{L} = \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - g(f(\mathbf{x}^{(n)}))\|^2$$
$$= \sum_{n=1}^{N} \|\mathbf{x}^{(n)} - f \circ g(\mathbf{x}^{(n)})\|^2.$$

二、Neural models for NLP

5. AutoEncoder

AutoEncoder

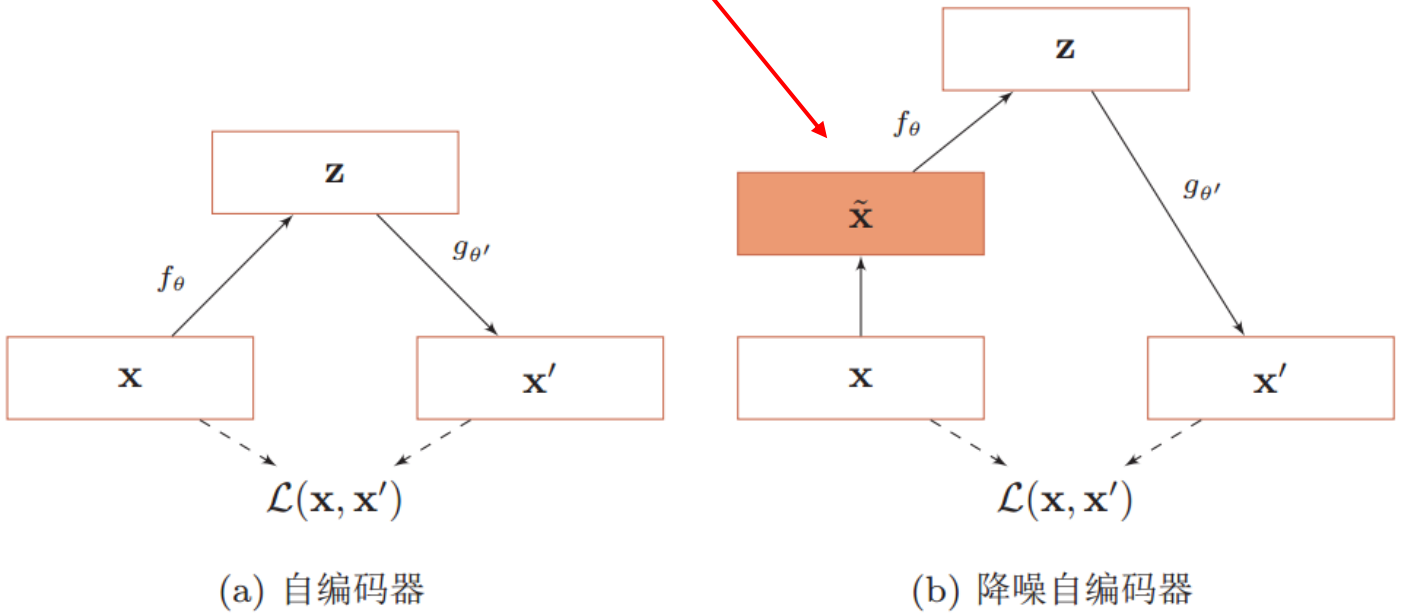
- Stacked AutoEncoder
- Denoising AutoEncoder
- Sparse AutoEncoder
- Variational AutoEncoder
- Convolutional AutoEncoder

二、Neural models for NLP

5. AutoEncoder

Denoising AutoEncoder

引入噪声来增加编码鲁棒性：根据比例 μ 随机将 x的一些维度的值设置为 0.



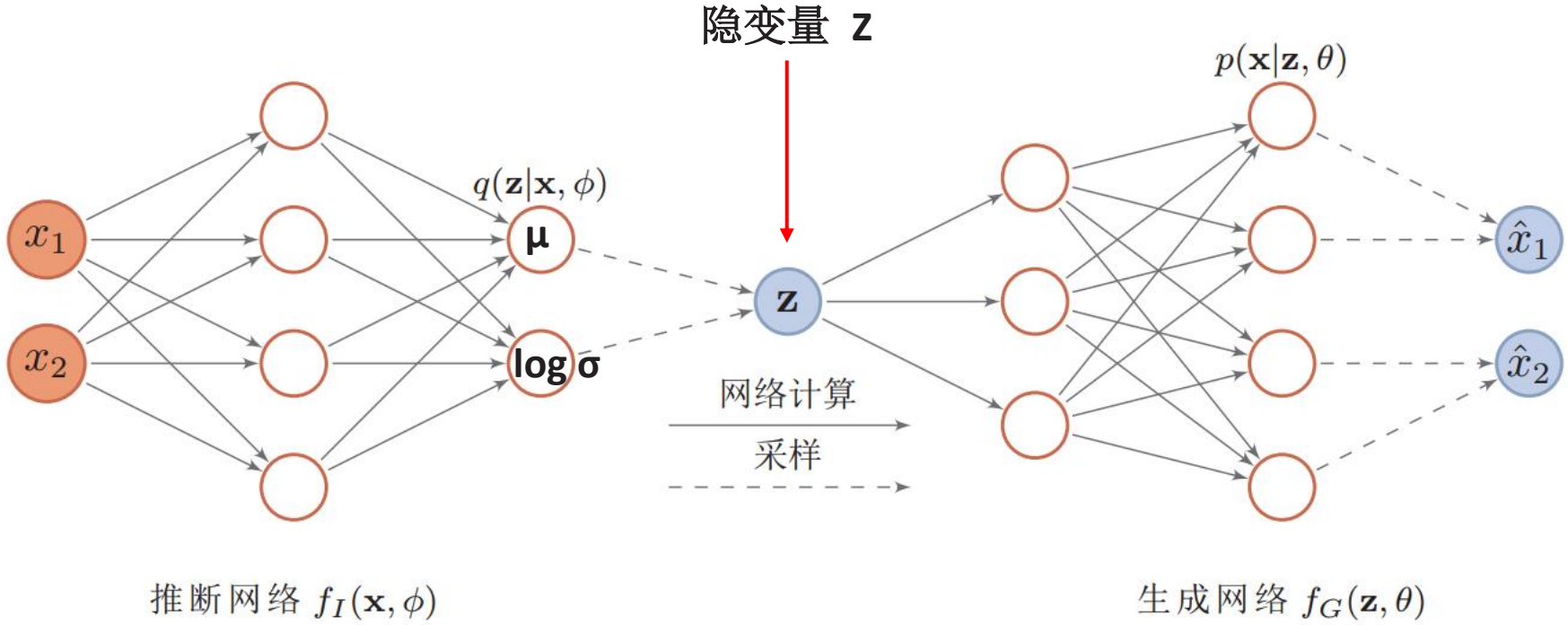
(a) 自编码器　　(b) 降噪自编码器

二、Neural models for NLP

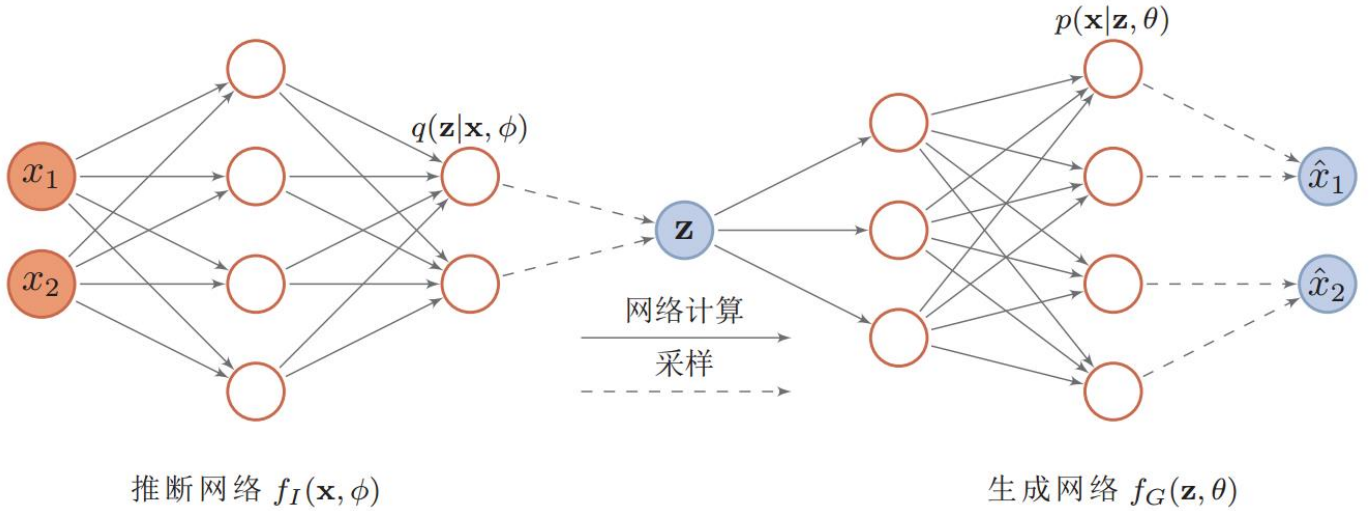
5. AutoEncoder

Variational AutoEncoder

- 生成模型
- 贝叶斯网络模型



Variational AutoEncoder



- 生成模型：联合概率

$$p(\mathbf{x},\mathbf{z}|\theta) = p(\mathbf{x}|\mathbf{z},\theta)p(\mathbf{z}|\theta)$$

已知 **z**时观测变量**x**的条件概率密度函数

隐变量 **z**先验分布的概率密度函数

- 学习目标：最大化生成概率(最大化对数边际似然)

$$\log p(\mathbf{x}|\theta) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\phi)}\left[\log \frac{p(\mathbf{x},\mathbf{z}|\theta)}{q(\mathbf{z}|\phi)}\right] + D_{\mathrm{KL}}(q(\mathbf{z}|\phi)\|p(\mathbf{z}|\mathbf{x},\theta))$$

decoding part

encoding part

EM算法来求解：

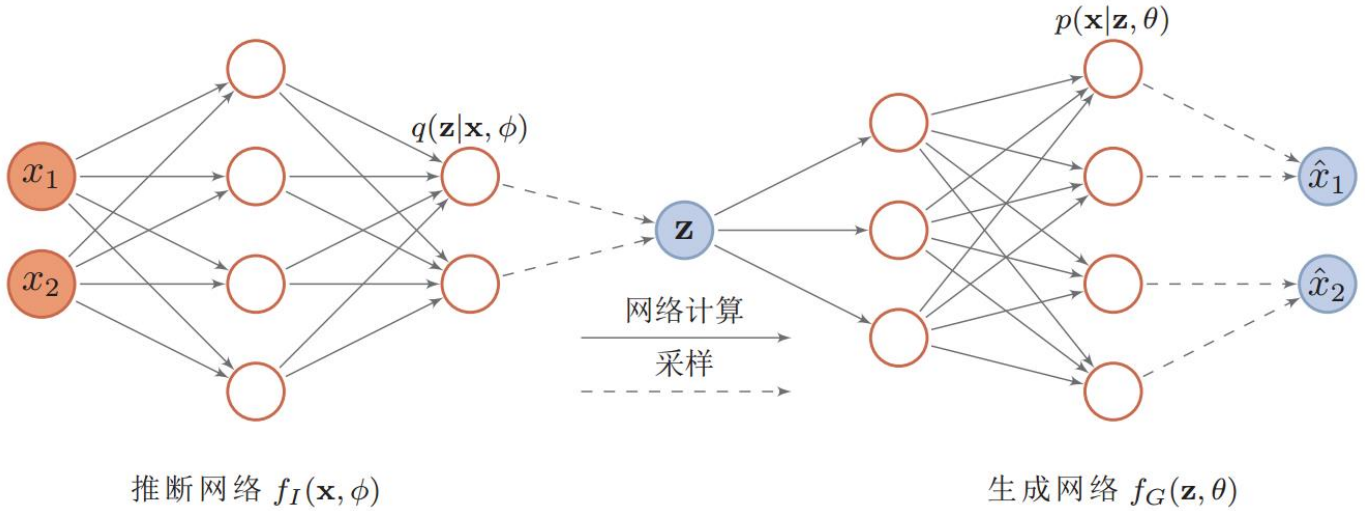
- E-step: 寻找一个密度函数$q(\mathbf{z}|\phi)$使其等于或接近于后验密度函数 $p(\mathbf{z}|\mathbf{x},\theta)$
- M-step: 保持$q(\mathbf{z}|\phi)$固定，寻找θ来最大化 $\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\phi)}\left[\log \frac{p(\mathbf{x},\mathbf{z}|\theta)}{q(\mathbf{z}|\phi)}\right]$

Variational AutoEncoder

$$D_{\mathrm{KL}}(q(\mathbf{z}|\phi)\|p(\mathbf{z}|\mathbf{x},\theta))$$

隐变量后验概率密度函数:

$$p(\mathbf{z}|\mathbf{x},\theta)=\frac{p(\mathbf{x}|\mathbf{z},\theta)p(\mathbf{z}|\theta)}{\int_z p(\mathbf{x}|\mathbf{z},\theta)p(\mathbf{z}|\theta)d\mathbf{z}}.$$



- 实质计算是一个**统计变分推断**问题 -> **变分**自编码器

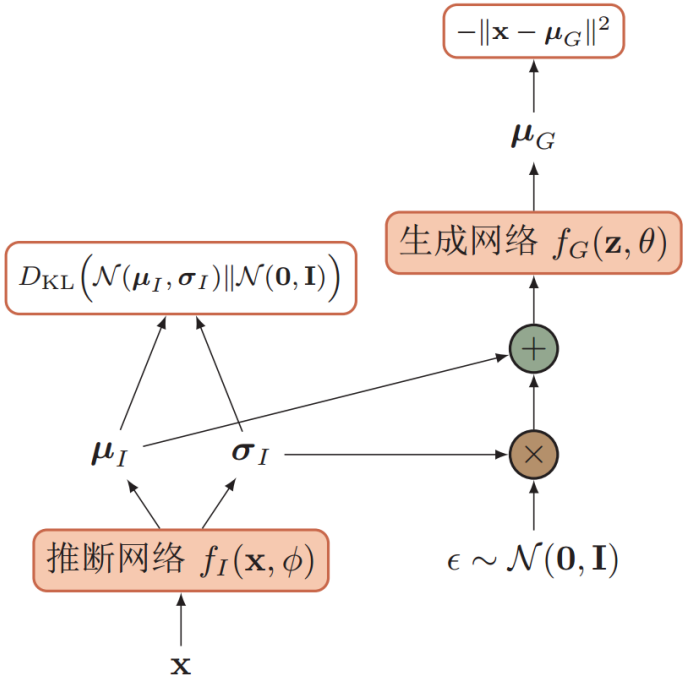
➢ 采样是一个离散的操作，不可微分。那么神经网络模型如何实现反向传播？

- **重参法**

假设 $q(\mathbf{z}|\mathbf{x},\phi)$ 为正态分布 $N(\boldsymbol{\mu}_I,\boldsymbol{\sigma}_I^2 I)$，可以这样采样 **z**：

$$\mathbf{z}=\boldsymbol{\mu}_I+\boldsymbol{\sigma}_I\odot\boldsymbol{\epsilon}$$

$$\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})$$

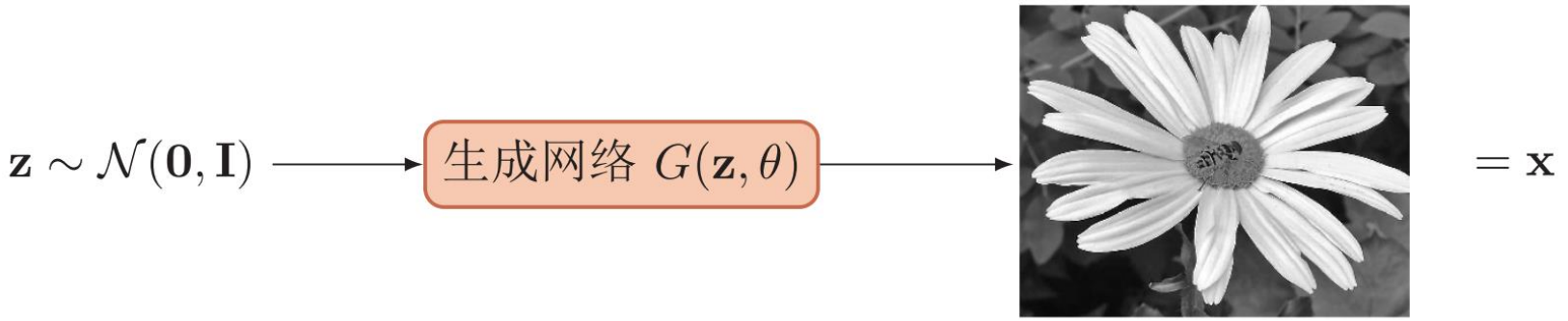


二、Neural models for NLP

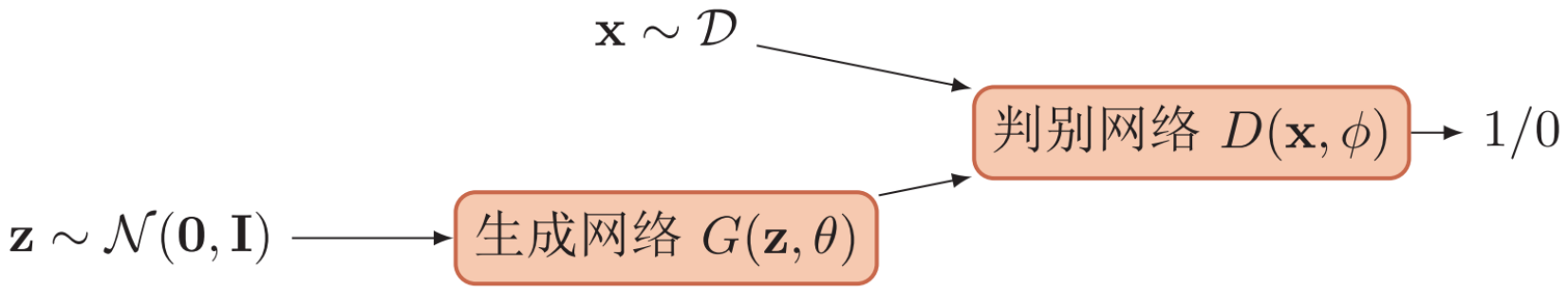
6. GAN

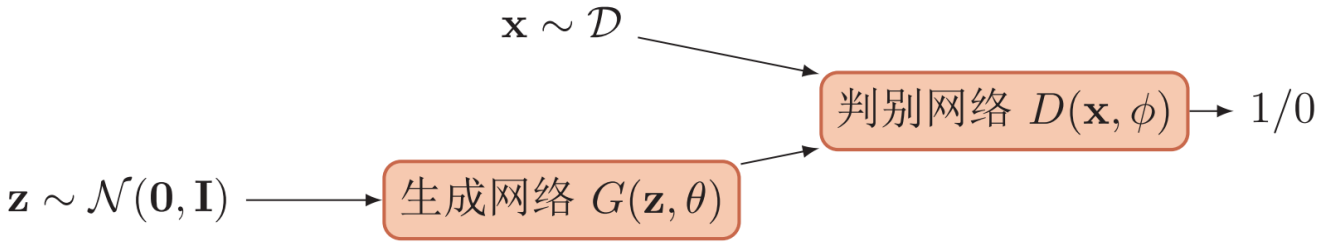
Generative Adversarial Nets, 生成对抗网络

- 隐式分布模型，不同于前面的VAE显示地建模分布，GAN模型直接用模型拟合分布，提升生成能力上限.



- GAN通过对抗训练的方式来使得生成网络产生的样本服从真实数据分布





- 判别网络（Discriminator Network）

  区分出一个样本$\mathbf{x}$是来自于真实分布$p_r(\mathbf{x})$还是来自于生成模型$p_\theta(\mathbf{x})$，为二分类器.

$$\begin{aligned}
&\min_{\phi} -\Big(\mathbb{E}_{\mathbf{x}}\Big[y \log p(y=1|\mathbf{x}) + (1-y)\log p(y=0|\mathbf{x})\Big]\Big) \\
&= \max_{\phi}\Big(\mathbb{E}_{\mathbf{x}\sim p_r(\mathbf{x})}\Big[\log D(\mathbf{x},\phi)\Big] + \mathbb{E}_{\mathbf{x}'\sim p_\theta(\mathbf{x}')}\Big[\log(1-D(\mathbf{x}',\phi))\Big]\Big) \\
&= \max_{\phi}\Big(\mathbb{E}_{\mathbf{x}\sim p_r(\mathbf{x})}\Big[\log D(\mathbf{x},\phi)\Big] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\Big[\log(1-D\Big(G(\mathbf{z},\theta),\phi\Big)\Big]\Big),
\end{aligned}$$

- 生成网络（Generator Network）

  生成样本，并尽可能让判别网络将自己生成的样本判别为真实样本.

$$\begin{aligned}
&\max_{\theta}\Big(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\Big[\log D\Big(G(\mathbf{z},\theta),\phi\Big)\Big]\Big) \\
&= \min_{\theta}\Big(\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\Big[\log\Big(1-D\Big(G(\mathbf{z},\theta),\phi\Big)\Big)\Big]\Big).
\end{aligned}$$

等价，但一般选上面组

二、Neural models for NLP

6. GAN

Generative Adversarial Nets

- **DCGAN**
- **WGAN**
- **SeqGAN**
- BEGAN
- LAPGAN
- PROGAN
- SAGAN
- BigGAN
- WGAN-GP
- LSGAN
- F-GAN
- UGAN
- LS-GAN
- MRGAN
- RGAN
- SN-GAN
- …

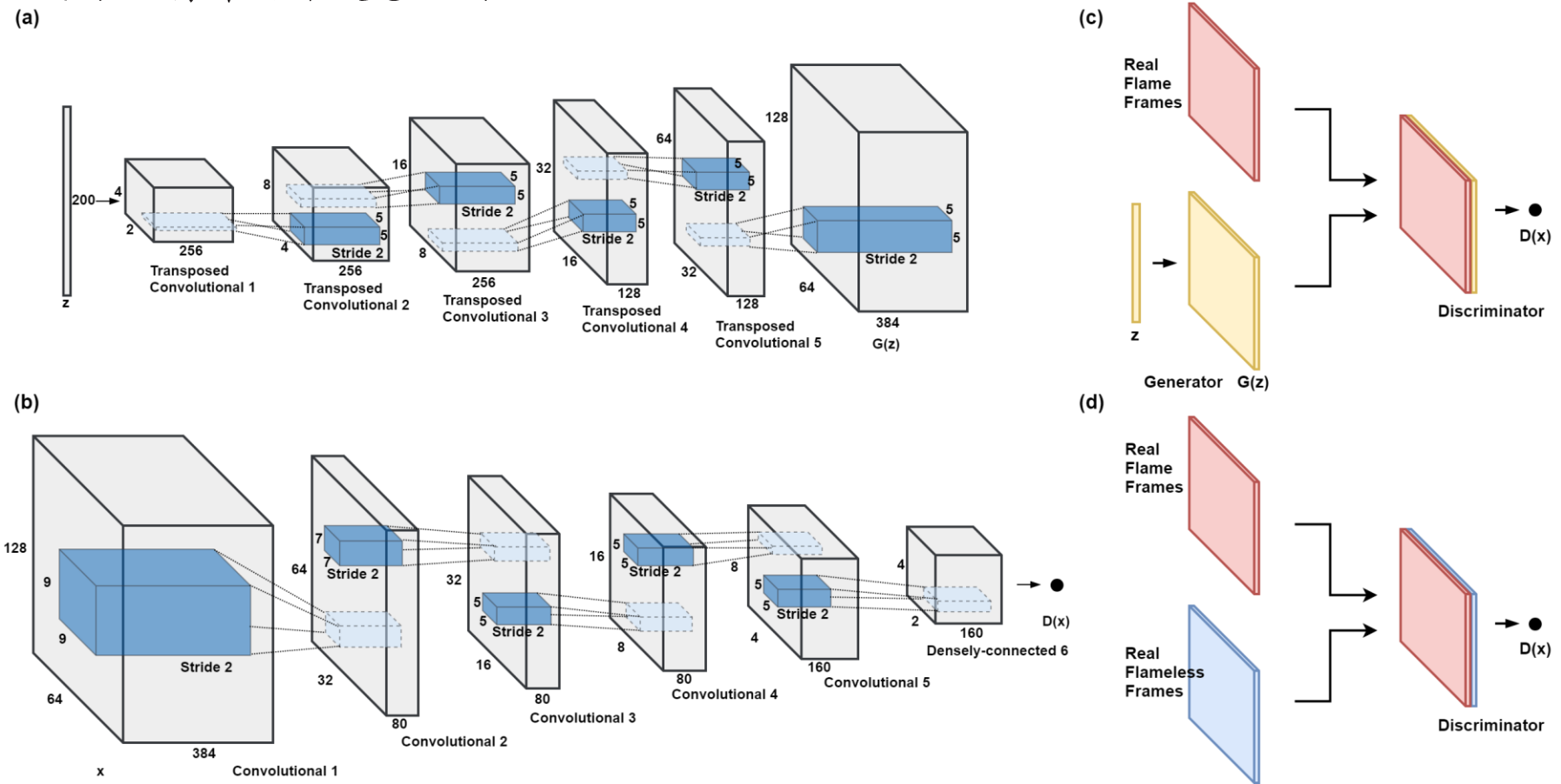
二、Neural models for NLP

6. GAN

DCGAN, Deep Convolutional Generative Adversarial Networks

- 判别网络: 传统的深度卷积网络，但使用了带步长的卷积来实现下采样操作，不用max pooling操作
- 生成网络:特殊的深度卷积网络

Application:
图像生成



二、Neural models for NLP

6. GAN

WGAN, Wassertein Adversarial Networks

- GAN 中交叉熵（**JS 散度**）不适合衡量生成数据分布和真实数据分布的距离，如果通过优化 JS散度训练 GAN会导致找不到正确的优化目标.
- WGAN是一种通过用 **Wassertein距离**替代 **JS散度**来优化训练的生成对抗网络.

说白点：

- 当两个分布没有重叠或者重叠非常少时，它们之间的 KL 散度为 $+\infty$， JS散度为 $\log 2$，并不随着两个分布之间的距离而变化，这时候GAN无法学习。
- 而 Wasserstein距离可以依然衡量两个没有重叠分布之间的距离，使得生成网络参数 θ 的梯度不会消失。

GAN:

$$-\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1-D(x))]$$

WGAN:

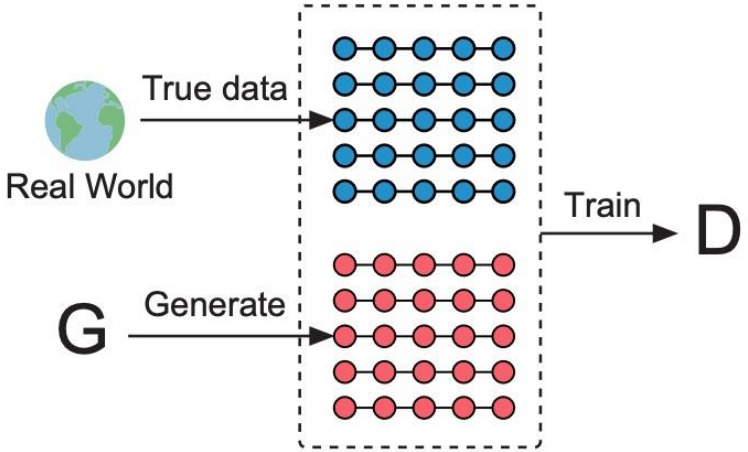
$$\min_D V_{WGAN}(D) = -E_{x\sim p_{data}(x)}[D(x)] + E_{z\sim p_z(z)}[D(G(z))]$$
$$\min_G V_{WGAN}(G) = -E_{z\sim p_z(z)}[D(G(z))]$$

二、Neural models for NLP

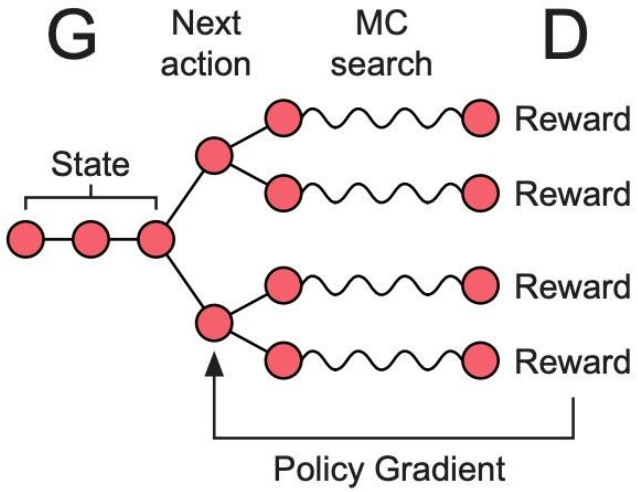
6. GAN

SeqGAN, Sequence Adversarial Networks

- GAN只能评估出整个生成序列的score/loss，不能够细化到去评估当前生成的局部token的好坏和对后面生成的影响，即评价不够细粒度.
- SeqGAN：把整个GAN看作一个**强化学习系统**，状态是生成的token，动作为下一个生成的token，利用蒙特卡洛搜索去估计状态行为值，用Policy Gradient算法更新Generator的参数.



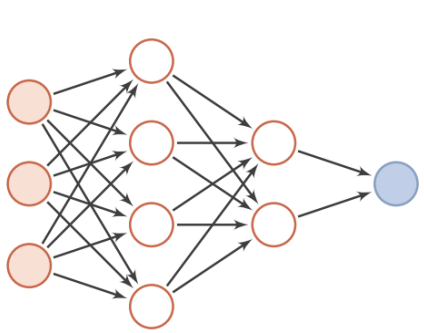
- 生成模型采用LSTM模型
- 判别模型采用CNN模型，和highway net模型



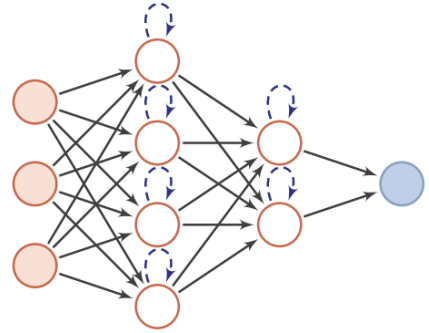
- 具体生成器的学习方式：生成过程中每一步都进行行为采样并估值，得到反馈reward用进行policy gradient进行参数更新.

二、Neural models for NLP

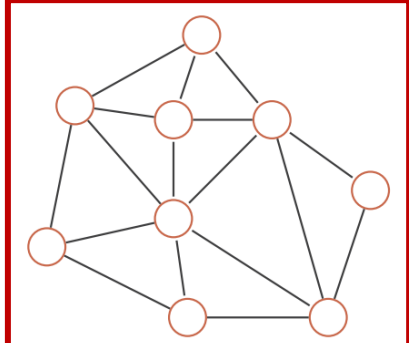
7. Graph neural models



(a) 前馈网络



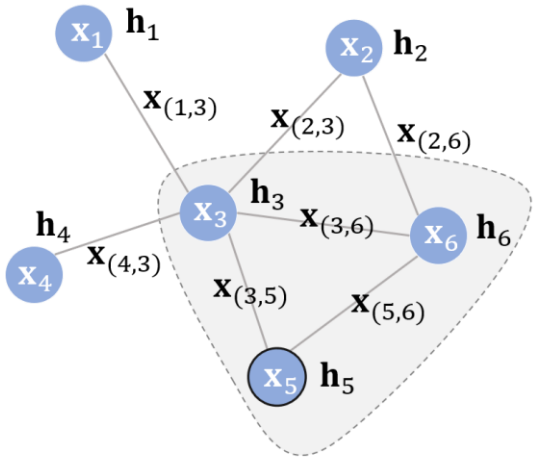
(b) 反馈网络



(c) 图网络

- 图神经网络将网络结构从线性扩展到图结构，属于前馈网络与反馈网络的泛化类。
- 支持图结构数据，如无环图，圈，有向和无向图等.

- 给定一张图 G = (V, E), V = {v_1 , ..., v_N } 是节点集合，N = |V |为节点数量，E ⊆ V × V 即节点之间的边的集合，M = |E| 为边的数量.
- 每个结点都有其对应的特征表征向量, **x**_v表示结点v的表征；连接两个结点的边E也有自己的特征，**x**_(v,u)表示结点v与结点u之间边的特征
- GNN的学习目标: 获得每个结点的隐藏状态表征向量**h**_v来作为节点的最终表征embedding. 具体方式为从邻接点中进行传播迭代式学习.



$$\mathbf{h}_5 = f(\mathbf{x}_5, \mathbf{x}_{(3,5)}, \mathbf{x}_{(5,6)}, \mathbf{h}_3, \mathbf{h}_6, \mathbf{x}_3, \mathbf{x}_6)$$

图网络模型隐藏状态更新函数:

$$\mathbf{h}_v^{t+1} = f(\mathbf{x}_v, \mathbf{x}_c o[v], \mathbf{h}_n^t e[v], \mathbf{x}_n e[v])$$

$$= \sum_{u \in ne[v]} FNN([\mathbf{x}_v; \mathbf{x}_{(u,v)}; \mathbf{h}_u^t; \mathbf{x}_u])$$

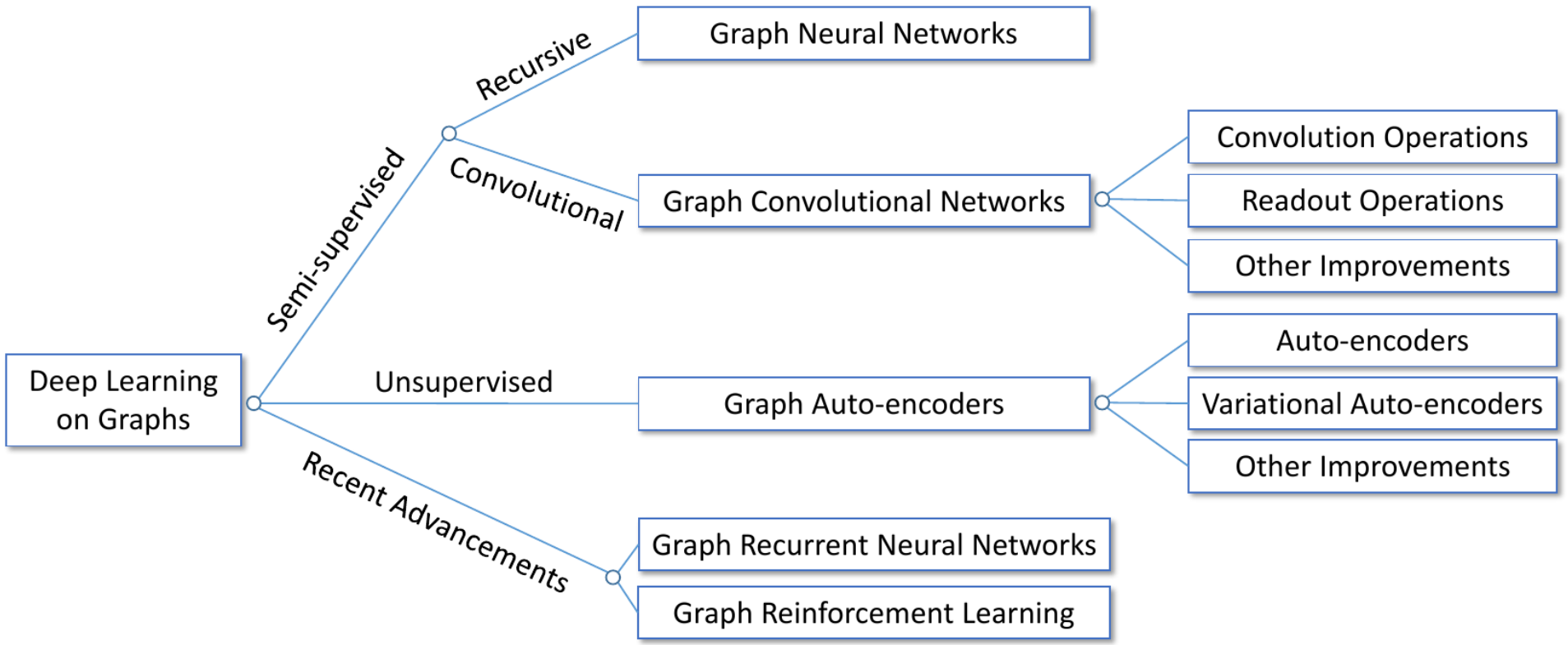
利用神经网络(Neural Network)来拟合该函数

- $\mathbf{x}_c o[v]$ ： 结点v相邻的边的特征,
- $\mathbf{x}_n e[v]$ ： 结点v的邻居结点的特征,
- $\mathbf{h}_n^t e[v]$ ： 邻居结点在t时刻的隐藏状态

二、Neural models for NLP

7. Graph neural models

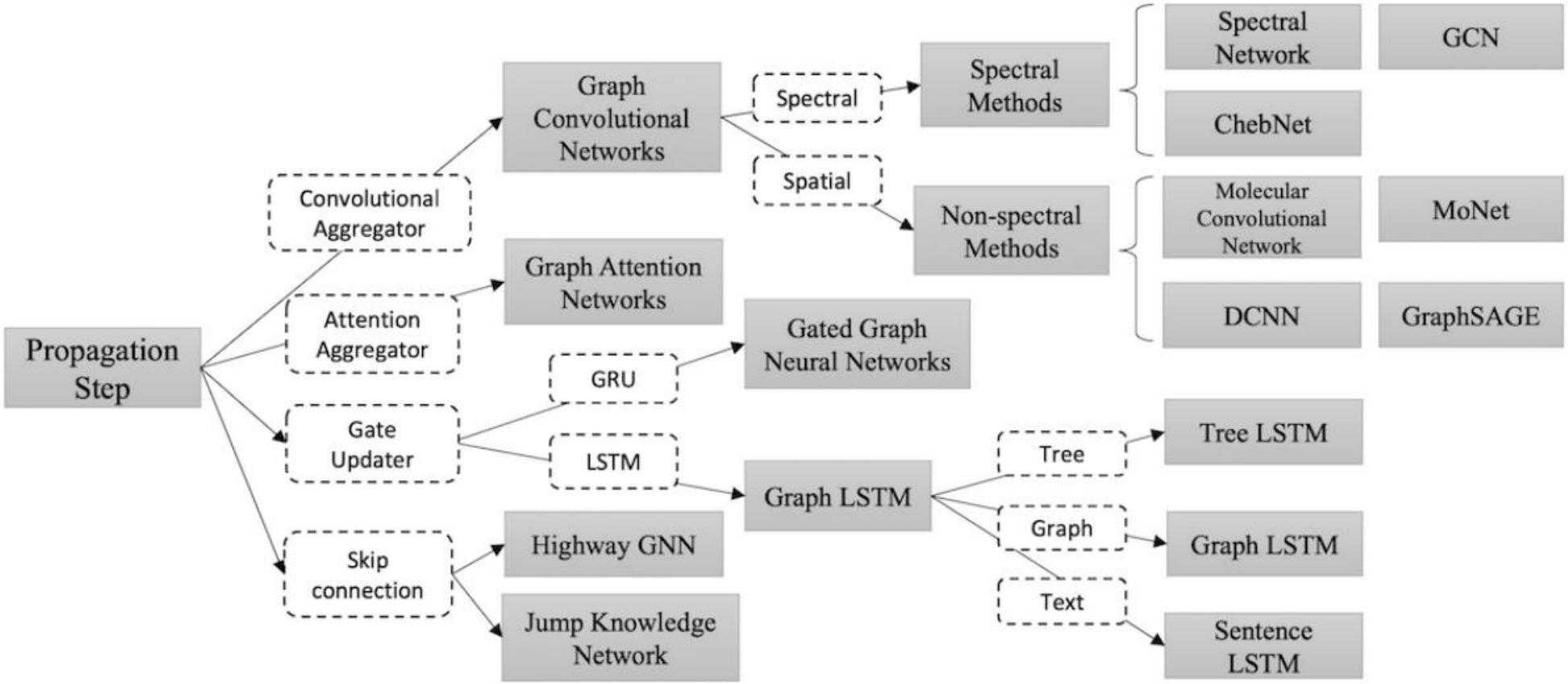
图神经网络分类



二、Neural models for NLP

7. Graph neural models

图神经网络分类



二、Neural models for NLP

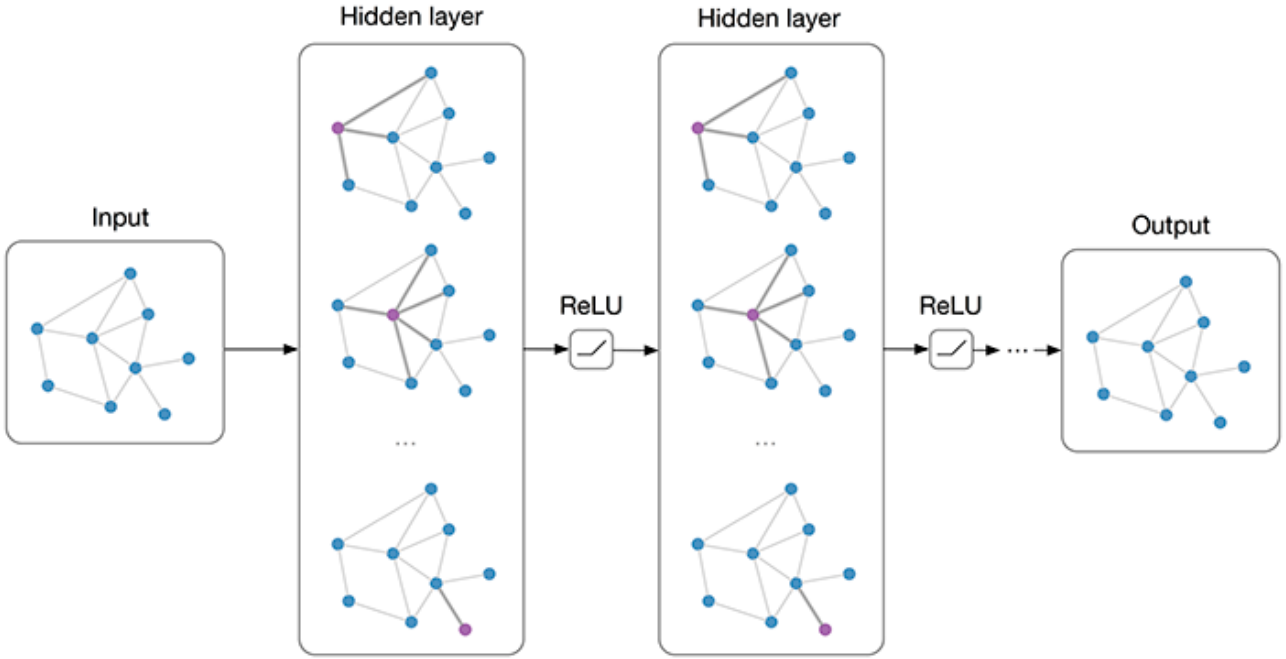
7. Graph neural models

GCN，Graph Convolutional Network, 图卷积神经网络

- GNN通过结点信息的传播使整张图达到收敛，然而特征学习能力不够充分，收敛速度慢.
- GCN将卷积操作扩展到图网络结构上，类似于前馈网络中的CNN，保持更强的图中局部特征学习能力.

➢ 问题：邻居结点数量不固定，如何进行卷积操作?

一句话理解：根据傅里叶变换将节点操作转化到"频域"中进行，即频域卷积(Spectral Convolution).



$$f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

$\hat{D}$ ： $\hat{A}$ 的degree matrix

W: 卷积核

H: 节点隐状态

$$\hat{A} = A + I$$

$\hat{A}$: 增加self-loop

A: 邻接矩阵

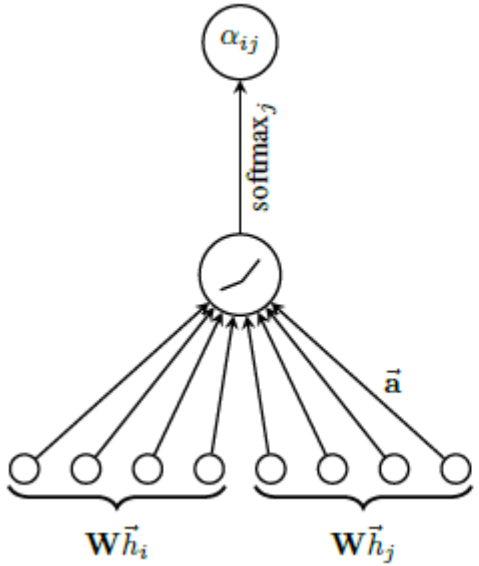
I: 单位矩阵

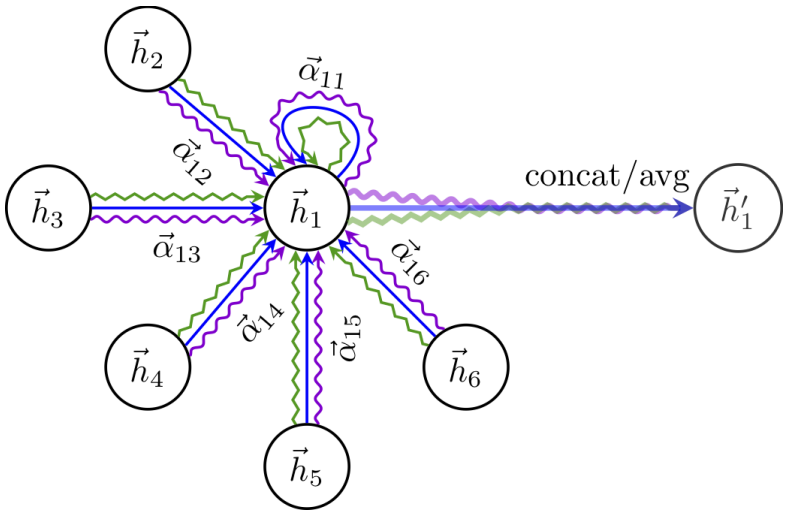
二、Neural models for NLP

7. Graph neural models

GAT，Graph Attention Network, 图注意力神经网络

- GCN的边是简单固定的， Convolution **加权平均**过程中邻居节点的权值也是简单固定的.
- GAT将注意力机制扩展到图网络结构上，让模型学得邻居节点的权值，使学习到“不同边的贡献不同”.





➢ 度量节点v_i对其邻节点的相关度分布：

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})}.$$

$$= \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

➢ GNN with attention weights：

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right)$$

二、Neural models for NLP

8. Embeddings

词向量是神经网络语言模型的副产品，那么有哪些种类的神经网络语言模型?

- 神经网络语言模型for静态词向量
  - Neural Network Language Model， NNLM
  - Log-Bilinear Language Model， LBL
  - Recurrent Neural Network based Language Model， RNNLM
  - C&W Language Model
  - CBOW(Continuous Bag-of-Words) and Skip-gram

- 神经网络语言模型for动态词向量(预训练上下文词表征)，又称新一代词表征
  - ELMo
  - GPT
  - BERT
  - ERNIE
  - XLNET
  - ALBERT
  - RoBERTa
  - ELECTRA
  - Big Bird
  - …

二、Neural models for NLP

8. Embeddings

Neural Network Language Model， NNLM

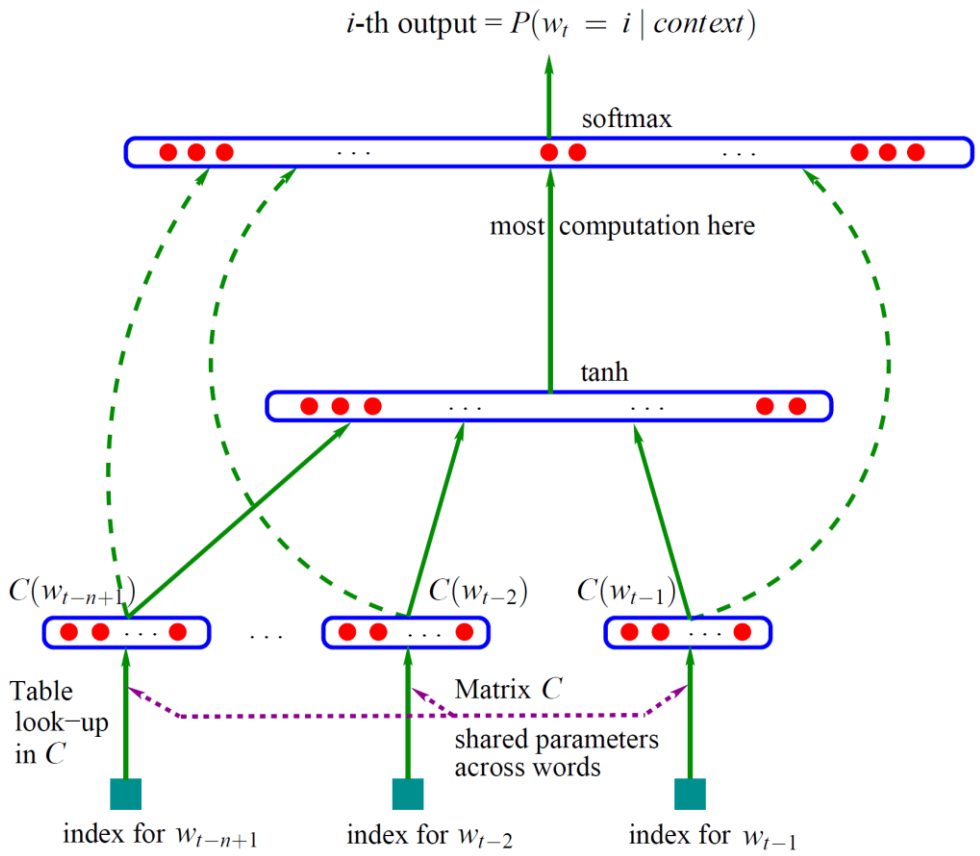
（自回归）神经网络语言模型学习目标？最大化预测词的似然概率：

$$max_\theta \ logp_\theta(x) = \sum_{t=1}^{T} log\ p_\theta(x_t|x_{<t})$$

其中：

$$P(w_i \mid w_1, w_2, ..., w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

马尔科夫假设



二、Neural models for NLP

8. Embeddings

C&W Language Model

- 主要任务:
  - 词性标注 (POS)、
  - 短语识别(CHUNK)、
  - 命名实体识别(NER) 、
  - 语义角色标注 (SRL)
- 得到副产品：词向量

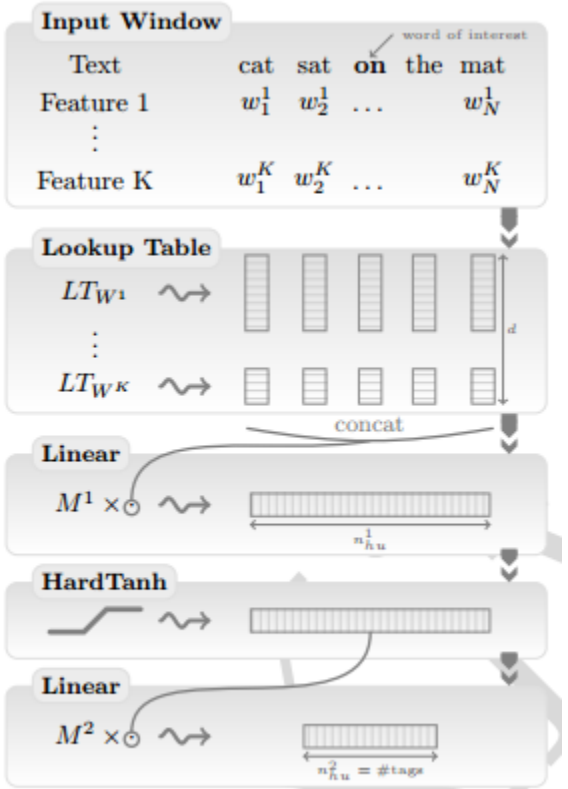
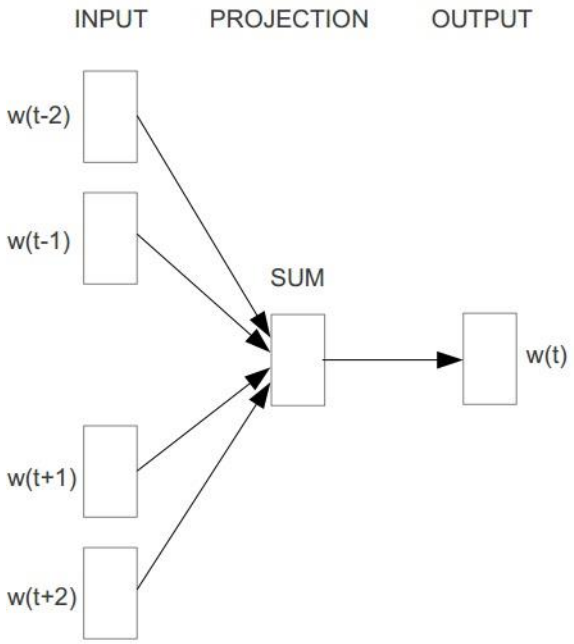


Figure 1: Window approach network.

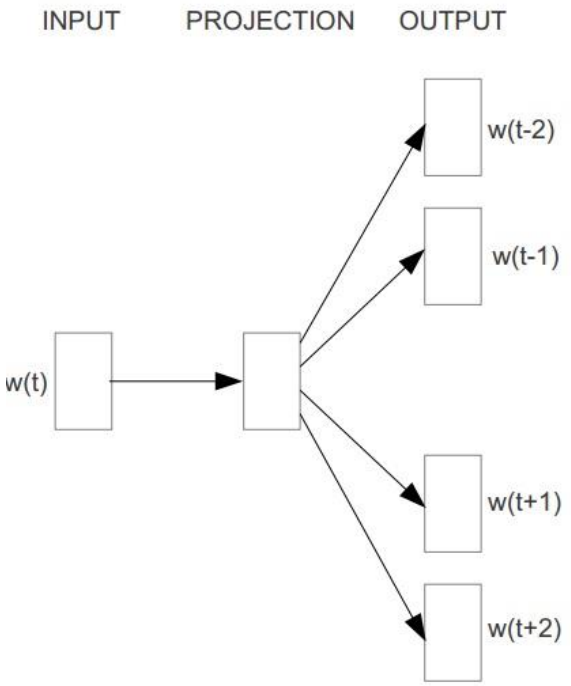
二、Neural models for NLP

8. Embeddings

CBOW and Skip-gram



CBOW



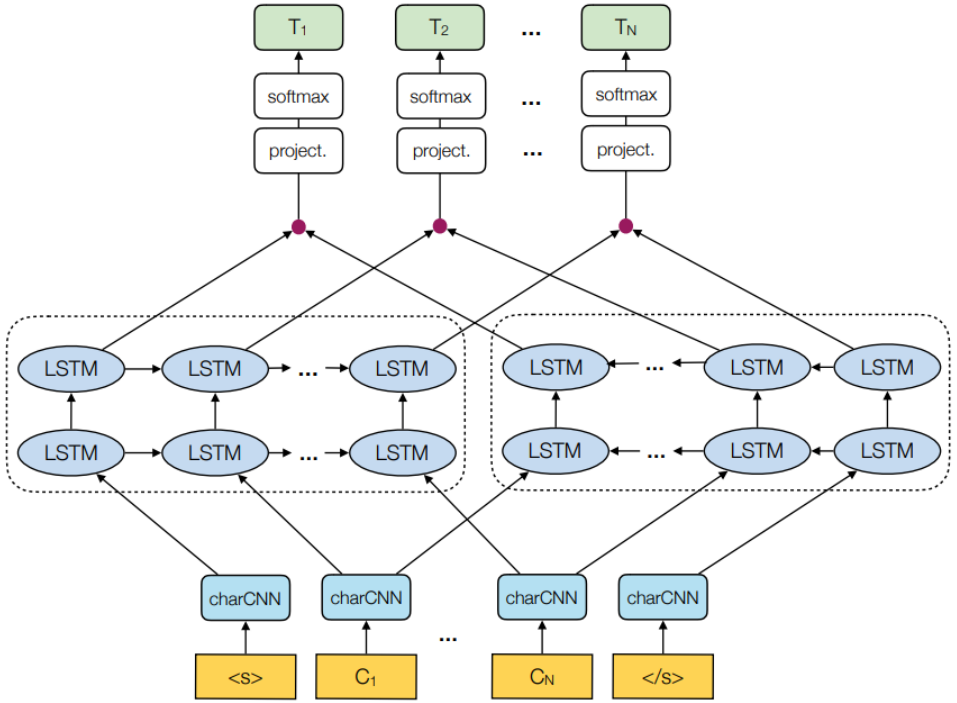
Skip-gram

常用的word2vec预训练词向量的出处.

二、Neural models for NLP

8. Embeddings

ELMo



- 1层字级别的CNN
- 2层词级别的LSTM,
- **每一层LSTM都有分别前向和后向2组，但不是像普通BiLSTM直接对应拼接，而是两个独立的单向LSTM实现的单向语言模型进行自回归预训练.**

What makes ELMo more prominent than RNNLM?

- 双向语言模型，其实是2个单向语言模型（前向和后向）的集成.
- 预训练语言庞大，比静态词向量都要大.

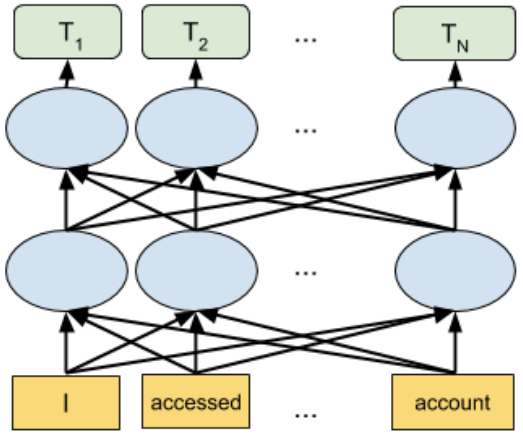
二、Neural models for NLP

8. Embeddings

BERT：Bidirectional Transformers

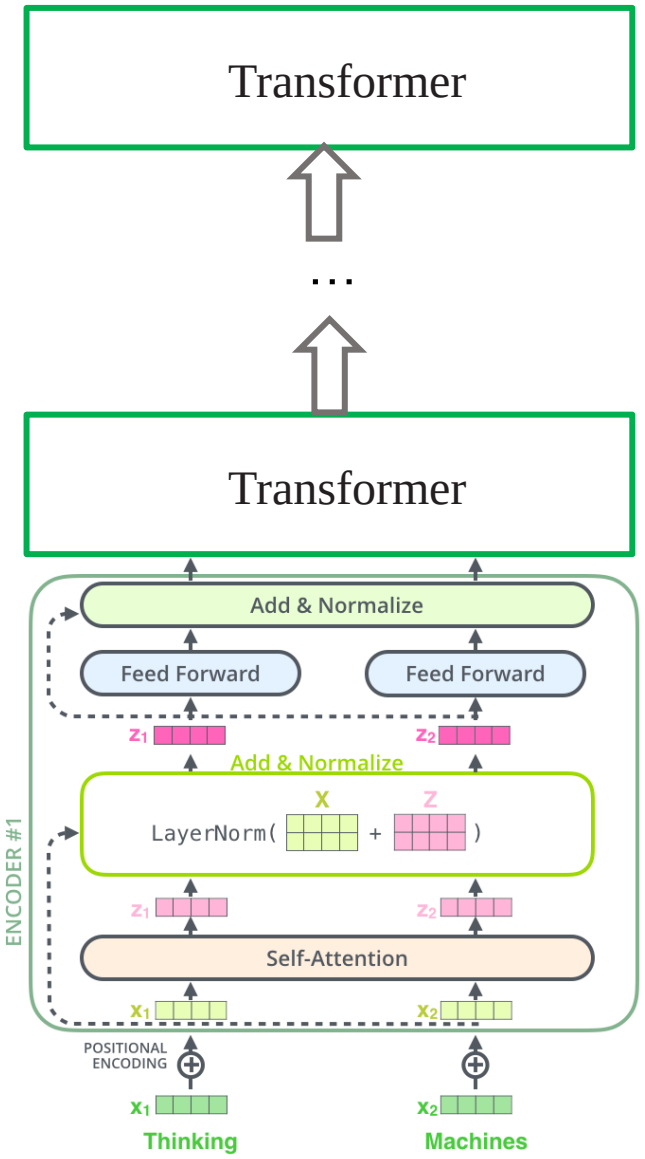
将若干层Transformer encoder进行堆叠，deep Transformer.

- 词表征表达能力更强.
- 基于句子级别的词表征，解决一词多义.
- 能捕捉到丰富的 语义和语法 信息.
- 可适用于无监督跨语言特征.
- …



等价

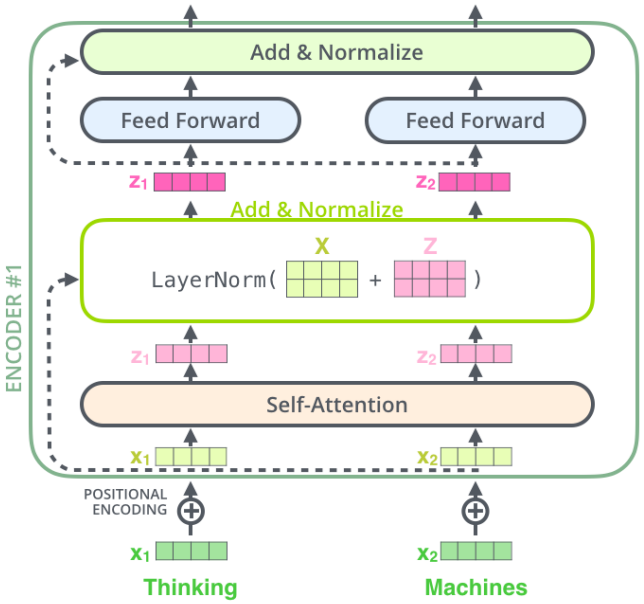
=



二、Neural models for NLP

8. Embeddings

BERT



What makes BERT so prominent?

- 自编码语言模型, 随机mask掉一些单词，在训练过程中根据上下文对masked词进行预测，使预测概率最大化: Masked Language Model, MLM. 从而迫使其充分利用双向的上下文信息.

$$max_\theta \ logp_\theta(\bar{x}|\hat{x}) \approx \sum_{t=1}^{T} log \ m_t p_\theta(x_t|\hat{x}) = \sum_{t=1}^{T} log \ m_t log \frac{exp(H_\theta(\hat{x})_t^T e(x_t))}{\sum_{x'} exp(H_\theta(\hat{x})_t^T e(x'))}$$

- Next Sequence Prediction，NSP. 判断句子B是否是句子A的下文.
- Deep structure of Transformer (self-attention).
- 预训练语言庞大.

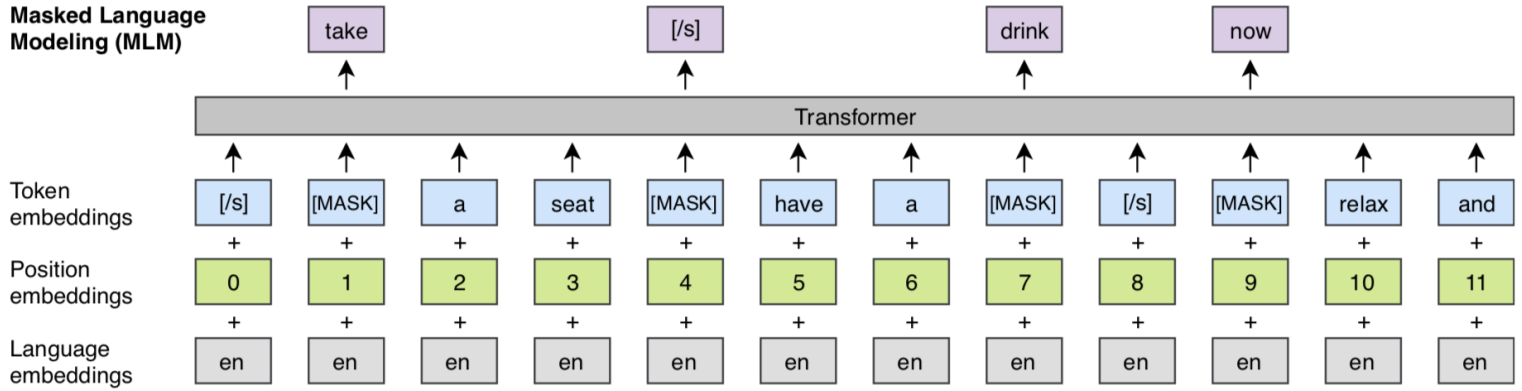
二、Neural models for NLP

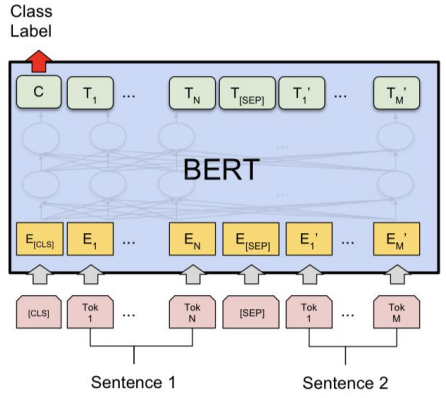
8. Embeddings

BERT

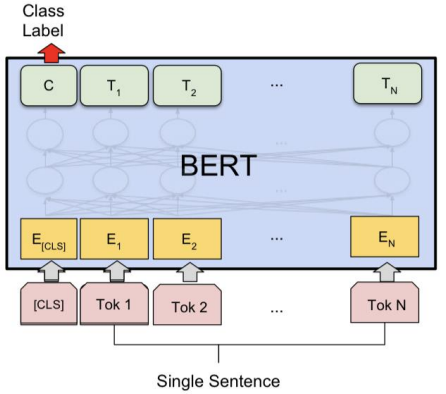
How to leverage BERT?

- Pre-train
  - MLM, NSP
  - 基于10B级别的语料进行无监督的训练
- Fine-tune
  - 针对不同任务与其对应的训练数据进行有监督训练

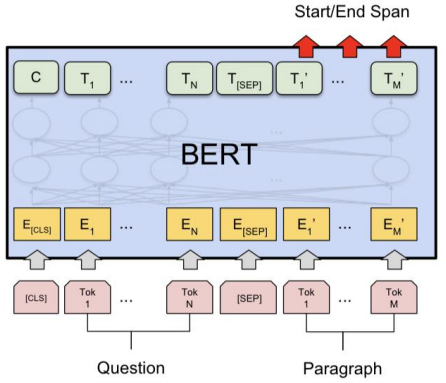




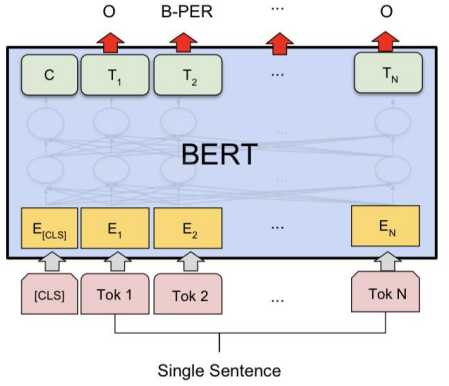
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1

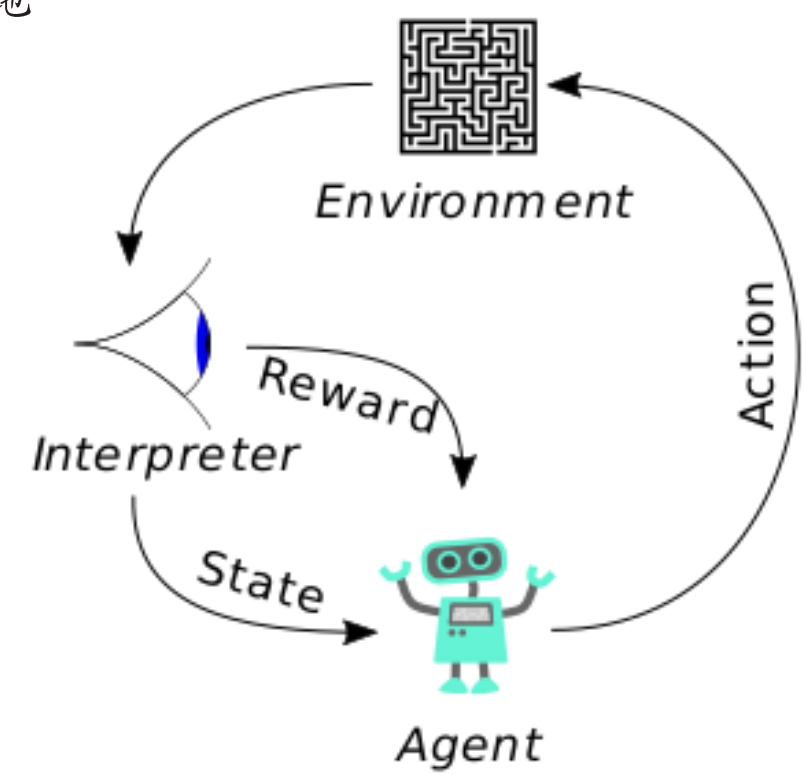


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

二、Neural models for NLP

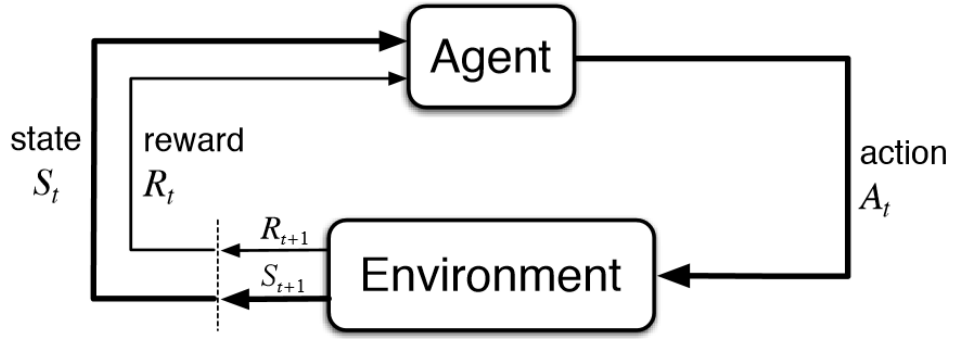
9. Reinforcement learning

- 强化学习核心思想：通过设计合适的**奖惩机制**启发式地让Agent学习到正确的环境应对能力.
- 使用环境：一切决策过程优化.
- 基本5要素：
  - Agent
  - Environment
  - Action
  - State
  - Reword

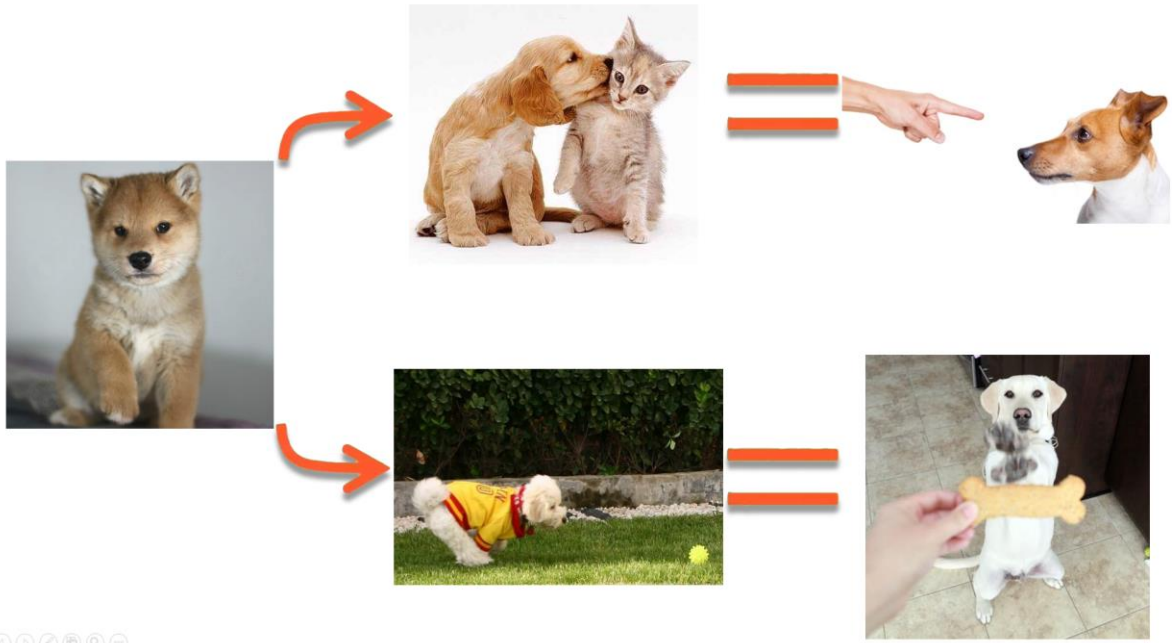


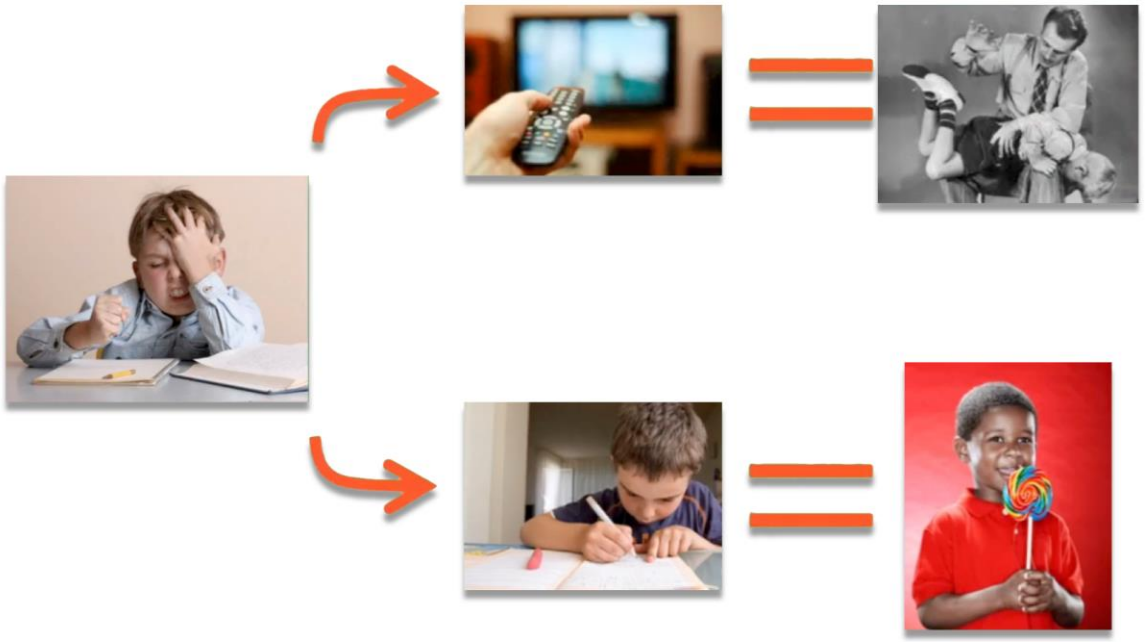
二、Neural models for NLP

9. Reinforcement learning

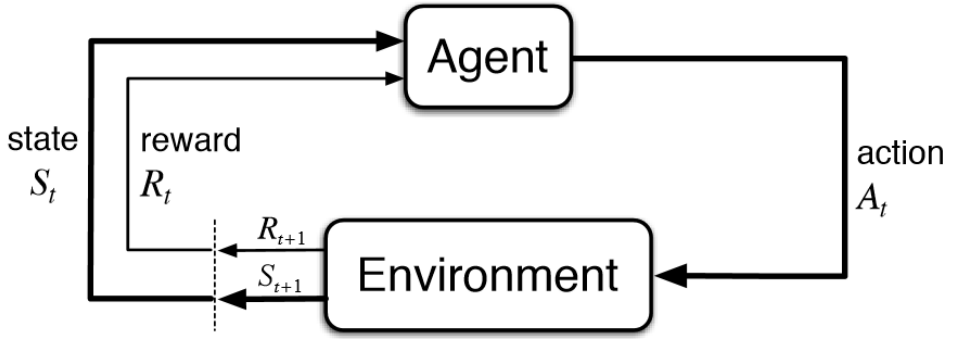


核心思想：通过设计合适的**奖惩机制**启发式地让Agent学习到正确的环境应对能力.





二、Neural models for NLP



9. Reinforcement learning

种类

角度一：根据学习方式

- Value-based: Q-learning Sarsa Deep Q Net
- Action-based: Policy Gradient
- Value&Action-based: Actor-Critic
- Model-based: Model-based RL

角度二：根据是否连续

- Continuous action
- Discrete action

角度三：根据采样方式

角度...

二、Neural models for NLP

9. Reinforcement learning

- 了解了强化学习是什么，有什么用，以及怎么工作的后，我们更关心一个强化学习是**如何进行学习**的.

策略用神经网络模型来拟合，即策略网络.

- Policy Gradient + Deep Learning
  - 目标是学习到一个策略: $\pi(a|s)$ (假设可微分)来最大化期望回报.
  - 策略梯度（policy gradient）采用梯度上升的方法来优化参数 θ 使得目标函数 J (θ)最大。

$$
\begin{aligned}
\frac{\partial \mathcal{J}(\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \int p_\theta(\tau) G(\tau) d\tau \\
&= \int \left(\frac{\partial}{\partial \theta} p_\theta(\tau) \right) G(\tau) d\tau \\
&= \int p_\theta(\tau) \left(\frac{1}{p_\theta(\tau)} \frac{\partial}{\partial \theta} p_\theta(\tau) \right) G(\tau) d\tau \\
&= \int p_\theta(\tau) \left(\frac{\partial}{\partial \theta} \log p_\theta(\tau) \right) G(\tau) d\tau \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\frac{\partial}{\partial \theta} \log p_\theta(\tau) G(\tau) \right], \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-1} \left(\frac{\partial}{\partial \theta} \log \pi_\theta(a_t|s_t) \gamma^t G(\tau_{t:T}) \right) \right],
\end{aligned}
$$

具体实现：REINFORCE 算法，每次采集一条轨迹，计算每个时刻的梯度并更新参数.

  - 所以总的回报(reward):

$$
G(\tau_{t:T}) = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'+1}
$$

二、Neural models for NLP

9. Reinforcement learning

- Policy Gradient + Deep Learning

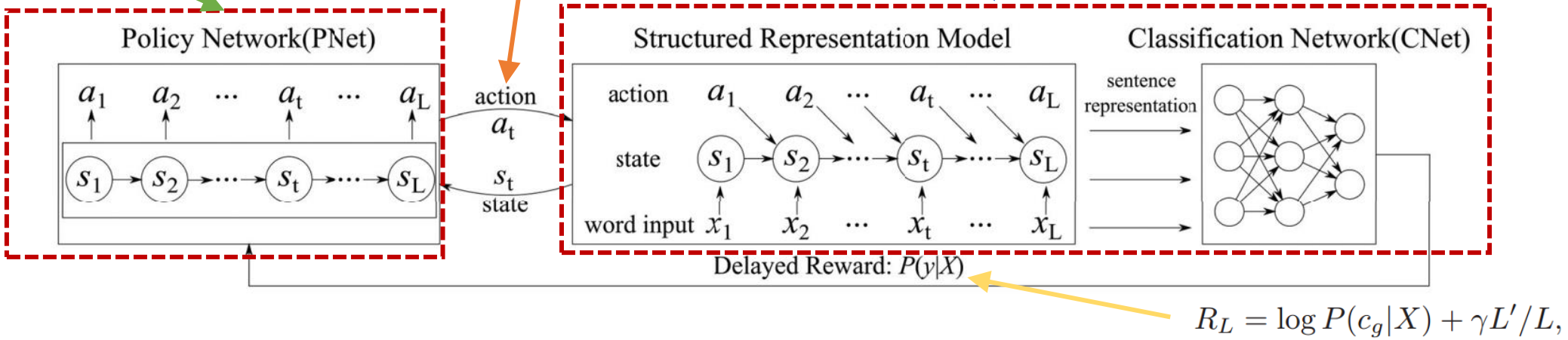
  Example：Learning Structured Representation for Text Classification via RL, AAAI2018

Agent(Policy Net)

$\pi(a_t|\mathbf{s_t};\Theta) = \sigma(\mathbf{W} * \mathbf{s_t} + \mathbf{b})$

$\{Retain, Delete\}$

Environment



$R_L = \log P(c_g|X) + \gamma L'/L,$

Learning the Policy Net:

$$J(\Theta) = \mathbb{E}_{(\mathbf{s_t},a_t)\sim P_\Theta(\mathbf{s_t},a_t)} r(\mathbf{s_1}a_1 \cdots \mathbf{s_L}a_L)$$

$$= \sum_{\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L} \prod_t \pi_\Theta(a_t|\mathbf{s_t}) R_L$$

Policy Gradient

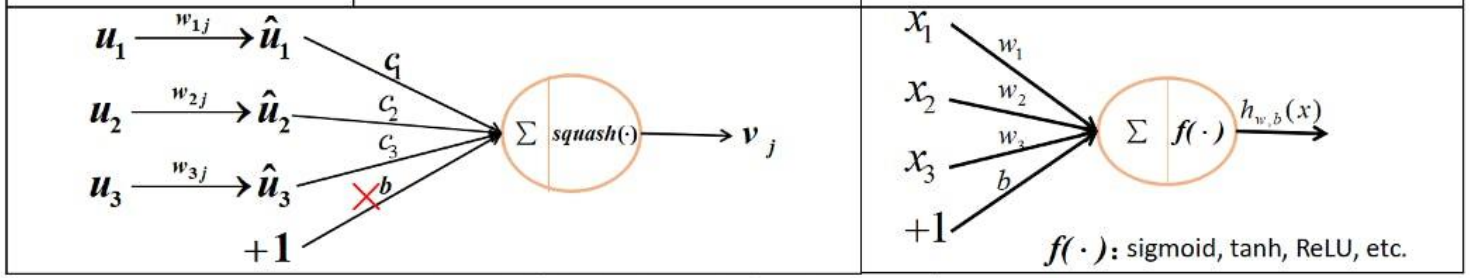
$$\nabla_\Theta J(\Theta) = \sum_{t=1}^{L} R_L \nabla_\Theta \log \pi_\Theta(a_t|\mathbf{s_t})$$

二、Neural models for NLP

10. Other types of networks

- Capsule Nets

		capsule	VS.	traditional neuron
Input from low-level neuron/capsule		vector(u_i)		scalar(x_i)
Operation	Affine Transformation	$\hat{u}_{j\|i} = W_{ij} u_i$ (Eq. 2)		–
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{j\|i}$ (Eq. 2)		$a_j = \sum_{i=1}^{3} W_i x_i + b$
	Sum			
	Non-linearity activation fun	$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2} \frac{s_j}{\|s_j\|}$ (Eq. 1)		$h_{w,b}(x) = f(a_j)$
output		vector(v_j)		scalar(h)



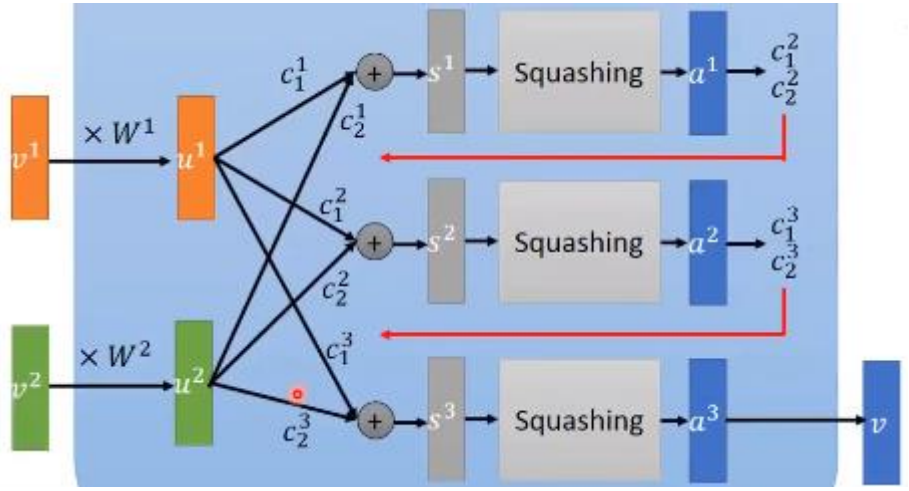
Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

特性：

- Capsule取代了神经元，各自的输出是矢量特征封装性, encapsulated features.
- 每个输出的Capsule大小表示表示“特征”的存在概率，其方向表示“特征”的属性（物体的方向、尺度等特征的状态）.
- 在明确表示各特征的属性的基础上，通过向更高层的Capsule传递，可以更准确地学习高阶特征，包括不同特征之间的相对位置关系等.

二、Neural models for NLP

10. Other types of networks



计算一层Capsule流程：

1: $b_{ij} = 0.$
2: $\overline{\boldsymbol{P}^{\boldsymbol{2}}_{\boldsymbol{j}|\boldsymbol{i}}} = \boldsymbol{W}_{\boldsymbol{ij}}\boldsymbol{P}^{\boldsymbol{2}}_{\boldsymbol{i}}.$
3: **for** each iteration in r **do**
4: $c_i = softmax(b_i).$
5: $\boldsymbol{S}_j = \sum_i c_{ij}\overline{\boldsymbol{P}^{\boldsymbol{2}}_{\boldsymbol{j}|\boldsymbol{i}}}.$
6: $\overline{\boldsymbol{P}^{\boldsymbol{3}}_{\boldsymbol{j}}} = g(\boldsymbol{S}_j).$
7: $a_{ij} = \overline{\boldsymbol{P}^{\boldsymbol{2}}_{\boldsymbol{j}|\boldsymbol{i}}} \cdot \boldsymbol{P}^{3}.$
8: $b_{ij} = b_{ij} + a_{ij}.$
9: **end for**

Dynamic Routing

Squash function： $\boldsymbol{P}^1_j = g(\boldsymbol{f}^1),$

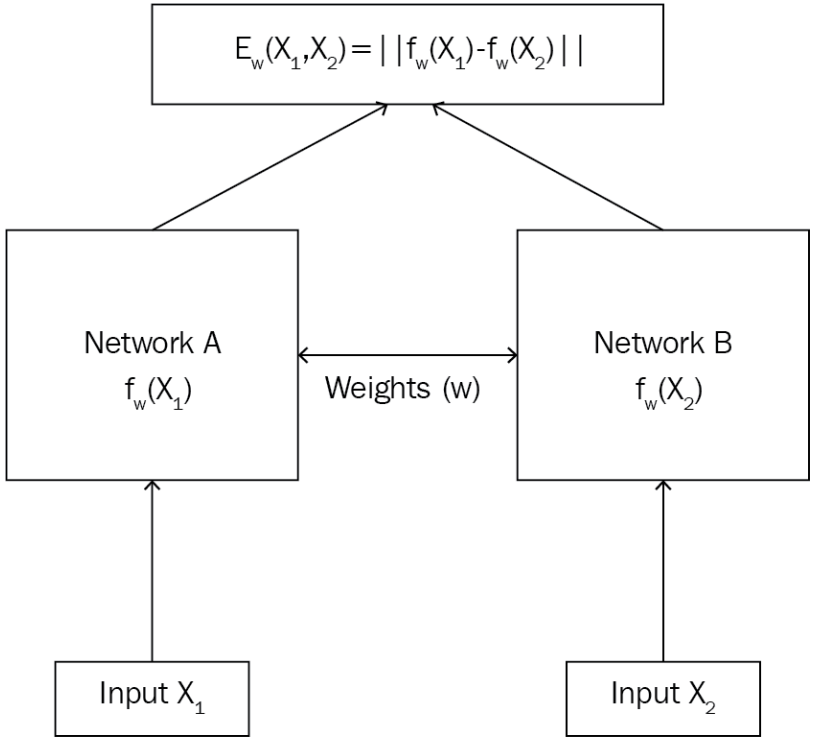
$$g(\mathbf{x}) = \frac{\|x\|^2}{0.5 + \|x\|^2}\frac{x}{\|x\|}.$$

二、Neural models for NLP

10. Other types of networks

- Siamese network

  度量两个输入的相似度.



特性：

- 左右网络结构保持一致.
- 共享权值.
- 损失函数是contrastive loss：

$$L(W,(Y,X_1,X_2)) = \frac{1}{2N}\sum_{n=1}^{N} Y D_W^2 + (1-Y)max(m-D_W,0)^2$$

$$D_W(X_1,X_2) = ||X_1 - X_2||_2 = (\sum_{i=1}^{P}(X_1^i - X_2^i)^2)^{\frac{1}{2}}$$

三、Neural nets for other domain applications

1. Computer Vision

- CNN (2d/3d kernel)
- LeNet
- PixelCNN
- AlexNet
- GoogLeNet
- resNet, 残差网络
- VGGNet
- Inception V1-V4
- R-CNN
- Faster R-CNN
- SSD
- YOLO

- NLP：以RNN为主导
- CV：以CNN为主导

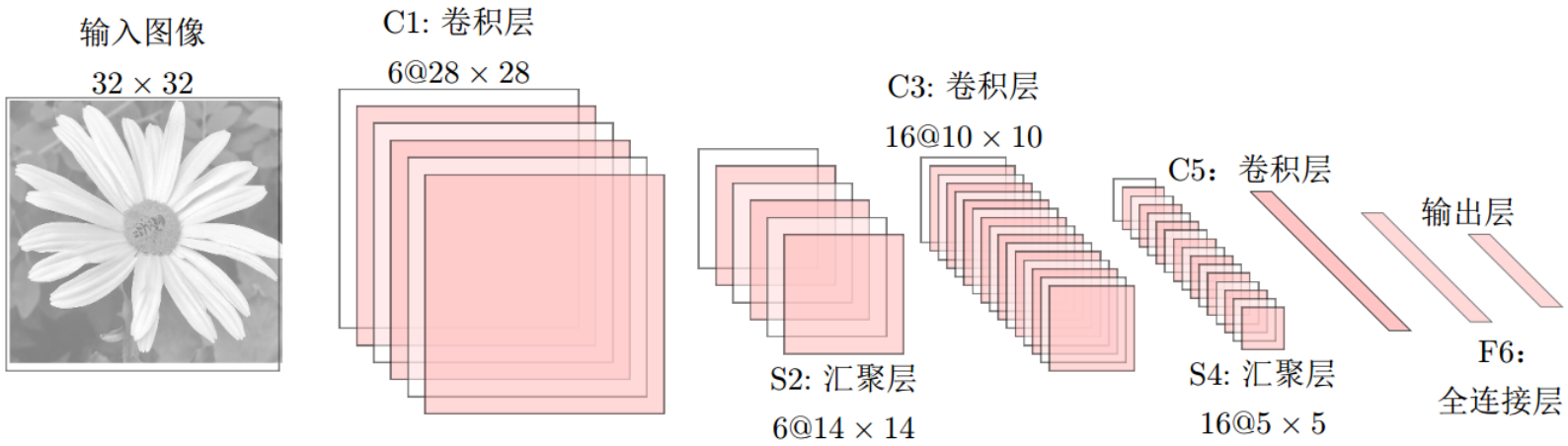
相比之下，CV的网络模型非常之深！

三、Neural nets for other domain applications

1. Computer Vision

LeNet-5

图像分类：手写数字识别系统

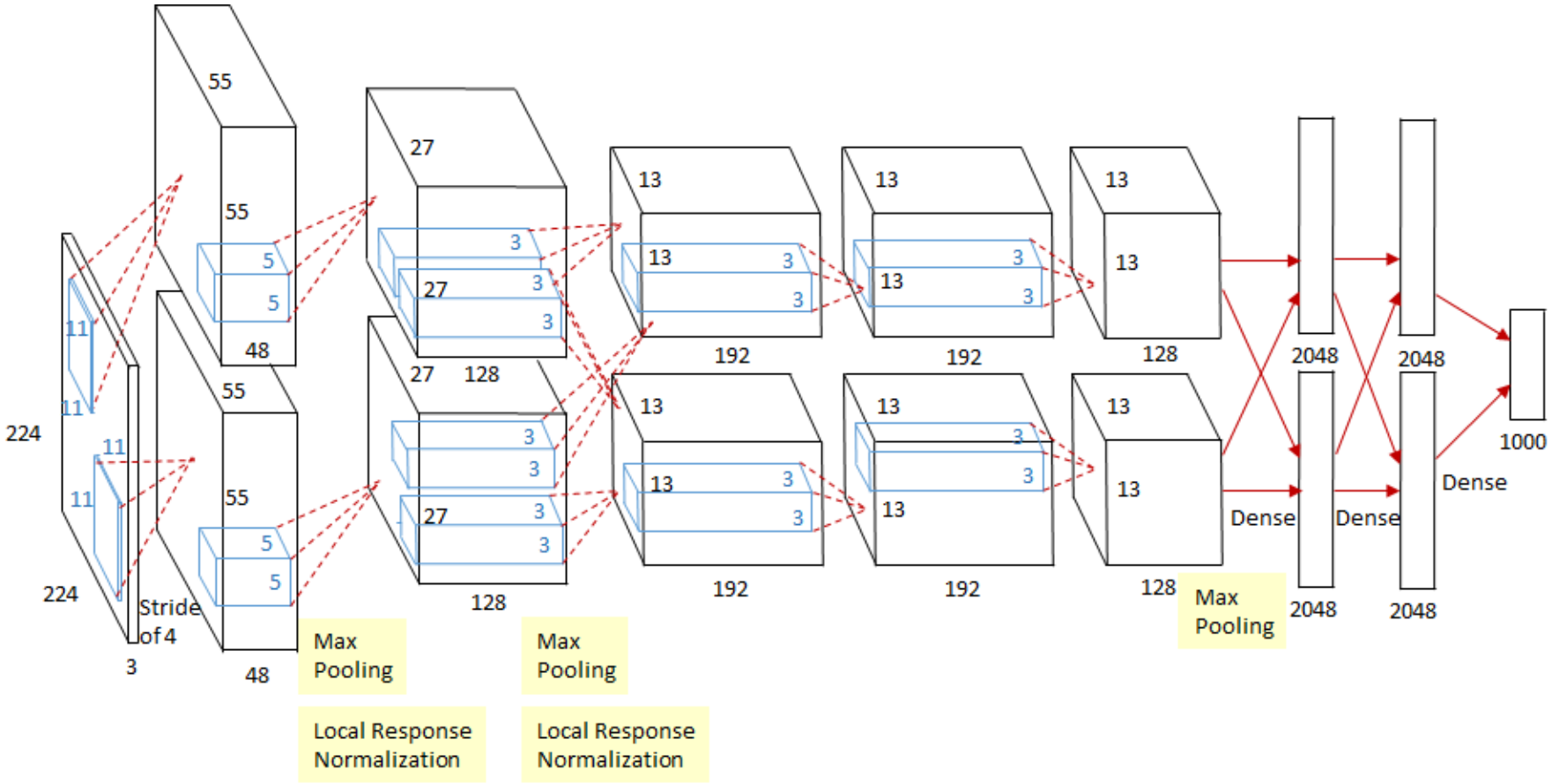


三、Neural nets for other domain applications

1. Computer Vision

AlexNet

图像分类

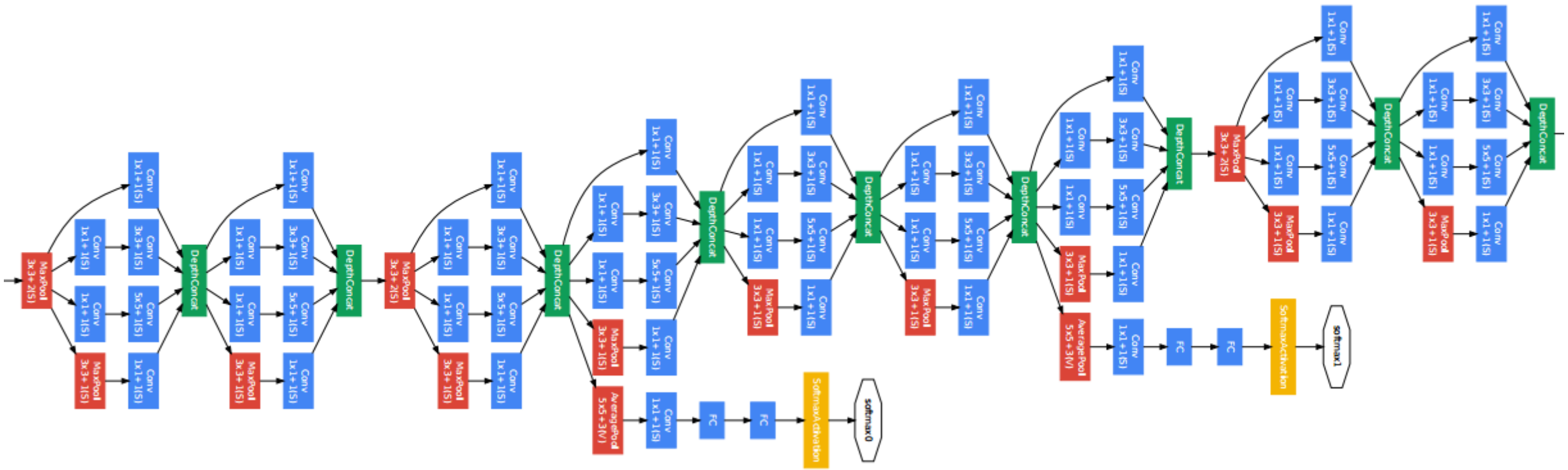


三、Neural nets for other domain applications

1. Computer Vision

Inception v1-v4

图像分类

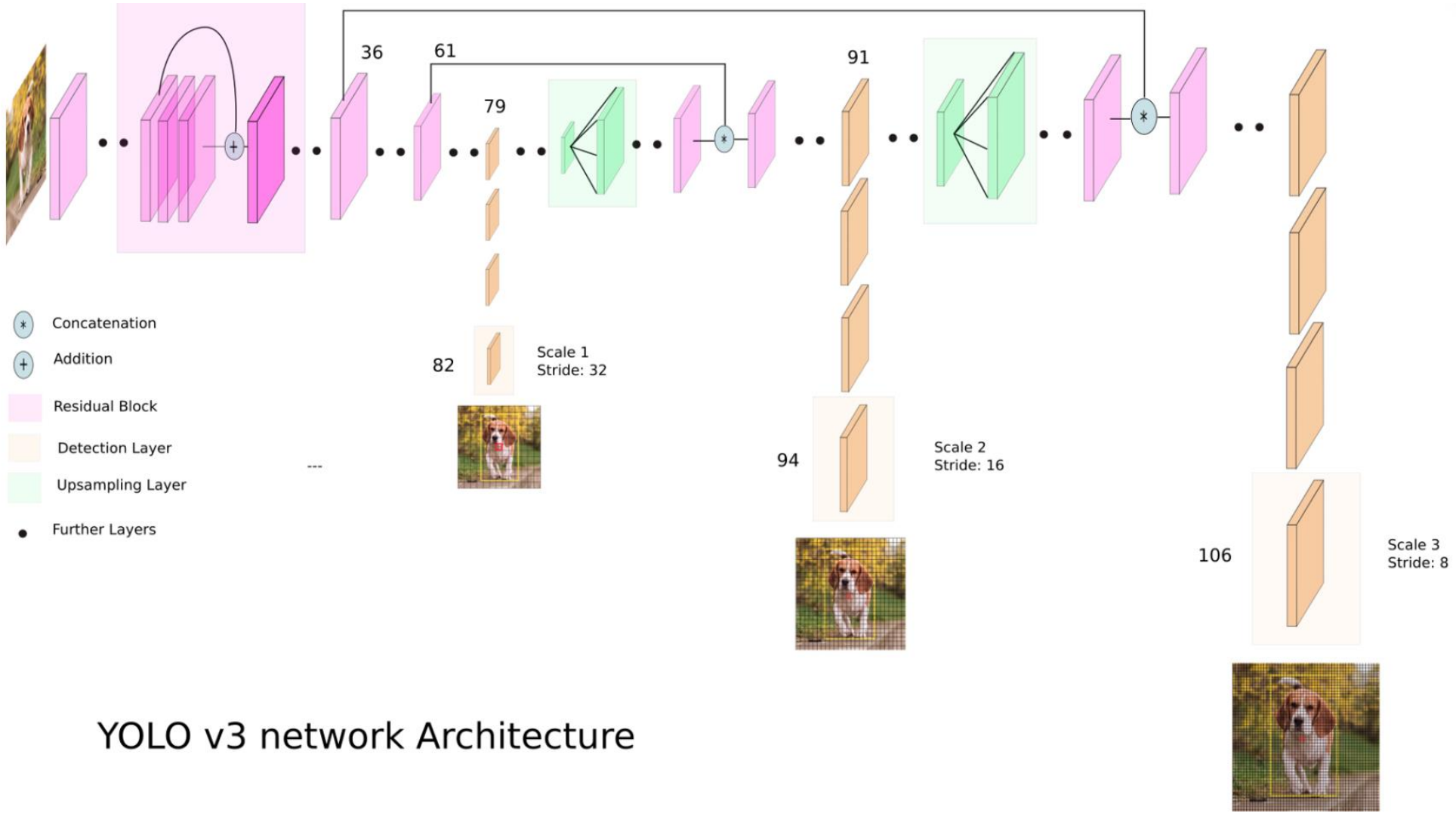


三、Neural nets for other domain applications

1. Computer Vision

YOLO

目标检测、
图像分割



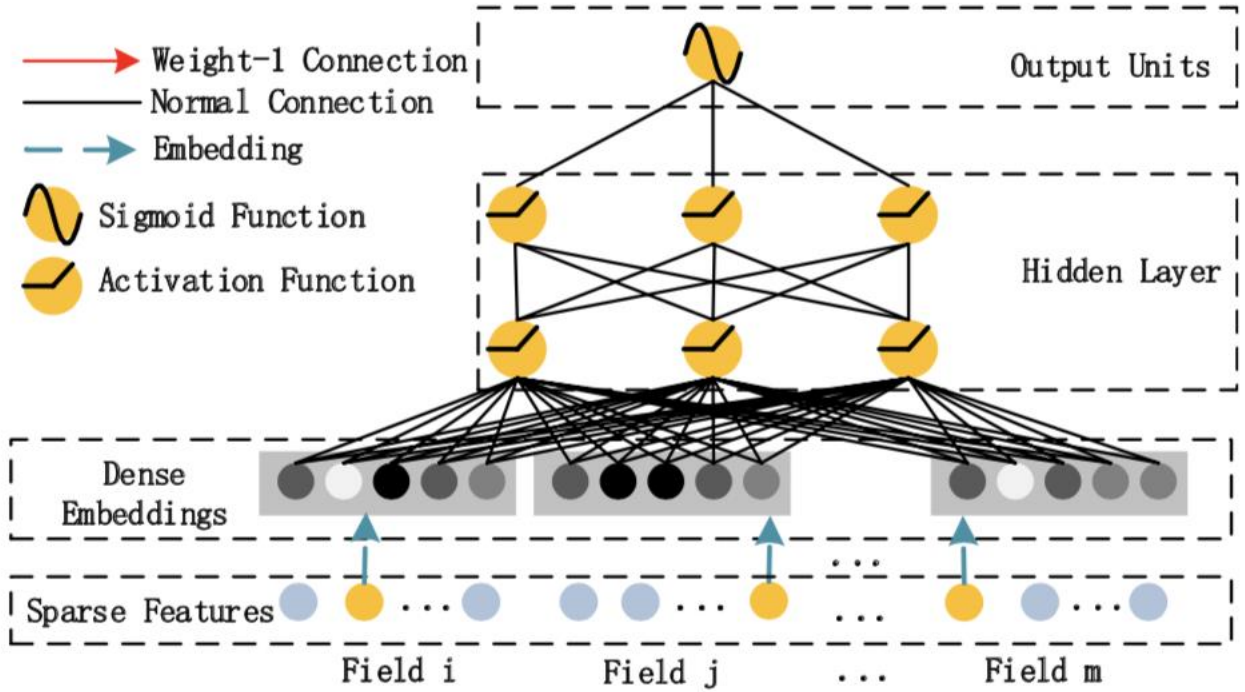
YOLO v3 network Architecture

三、Neural nets for other domain applications

2. Data Mining

DeepFM

CTR



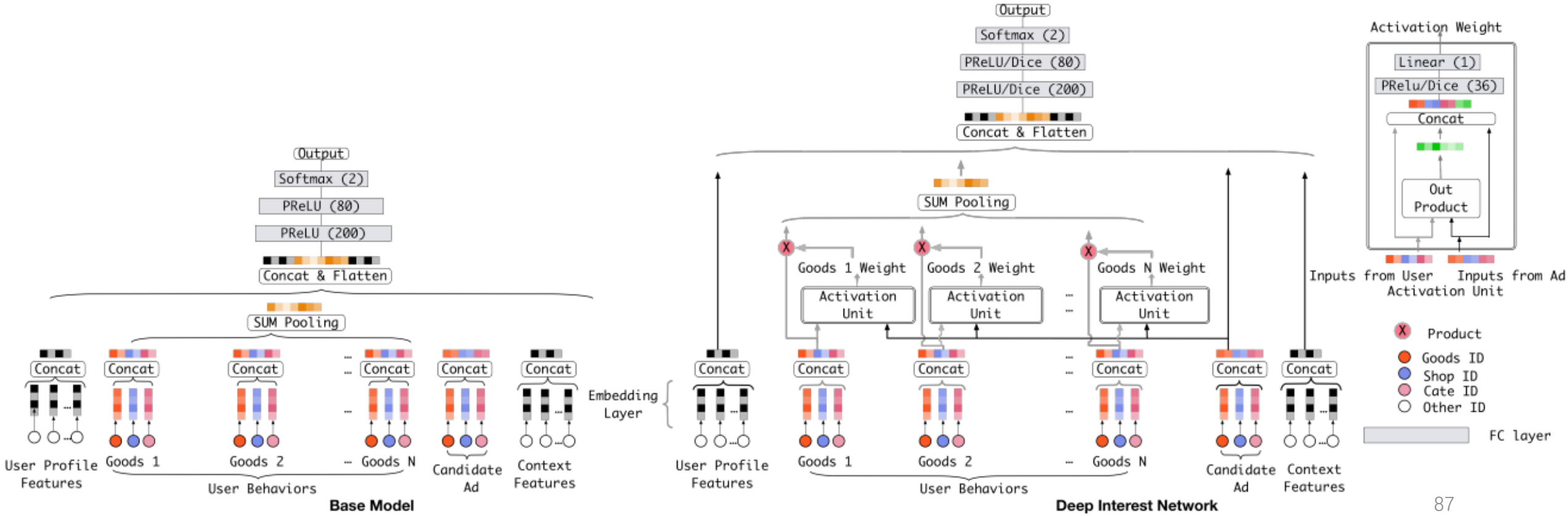
$$y_{FM} = \langle w, x \rangle + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} \langle V_i, V_j \rangle x_{j_1} \cdot x_{j_2},$$

三、Neural nets for other domain applications

2. Data Mining

DIN，Deep Interest Network，深度兴趣网络

用户历史行为挖掘

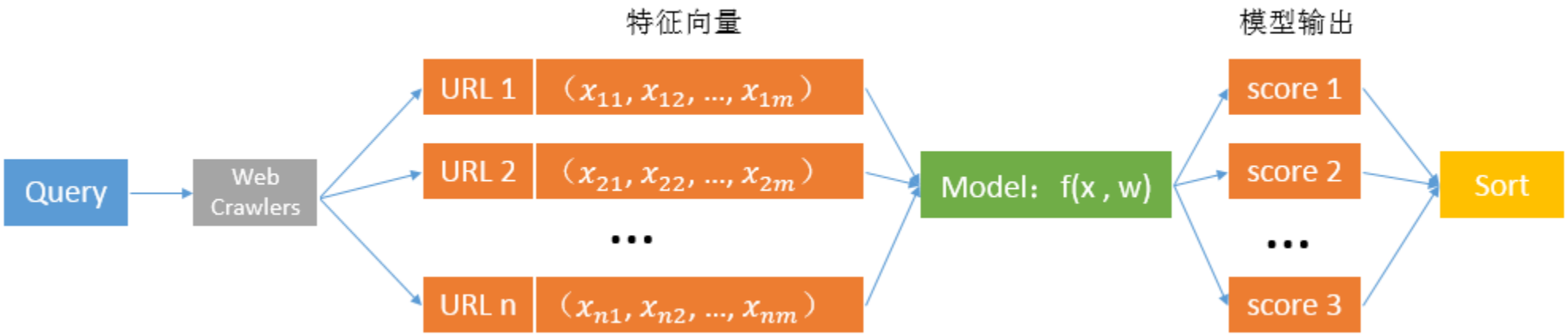


三、Neural nets for other domain applications

2. Data Mining

RankNet, LambdaRank, LambdaMart

排序模型



预测相关性概率：$$P_{ij} = P(U_i > U_j) = \frac{1}{1 + e^{-\sigma(s_i - s_j)}}$$

真实相关性概率：$$\overline{p}_{ij} = \frac{1}{2}(1 + S_{ij})$$ 如果U_i比U_j更相关，那么$S_{ij} = 1$；如果U_i不如U_j相关，那么$S_{ij} = -1$

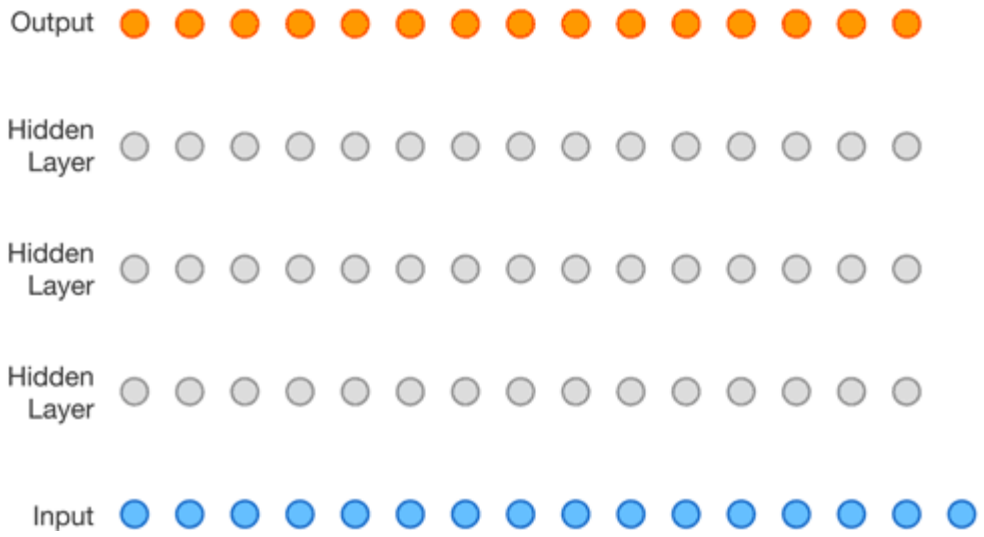
损失函数：$$C_{ij} = -\overline{P}_{ij} log P_{ij} - (1 - \overline{P}_{ij}) log(1 - P_{ij})$$

三、Neural nets for other domain applications

3. Audio and Speech

WaveNet

音频生成



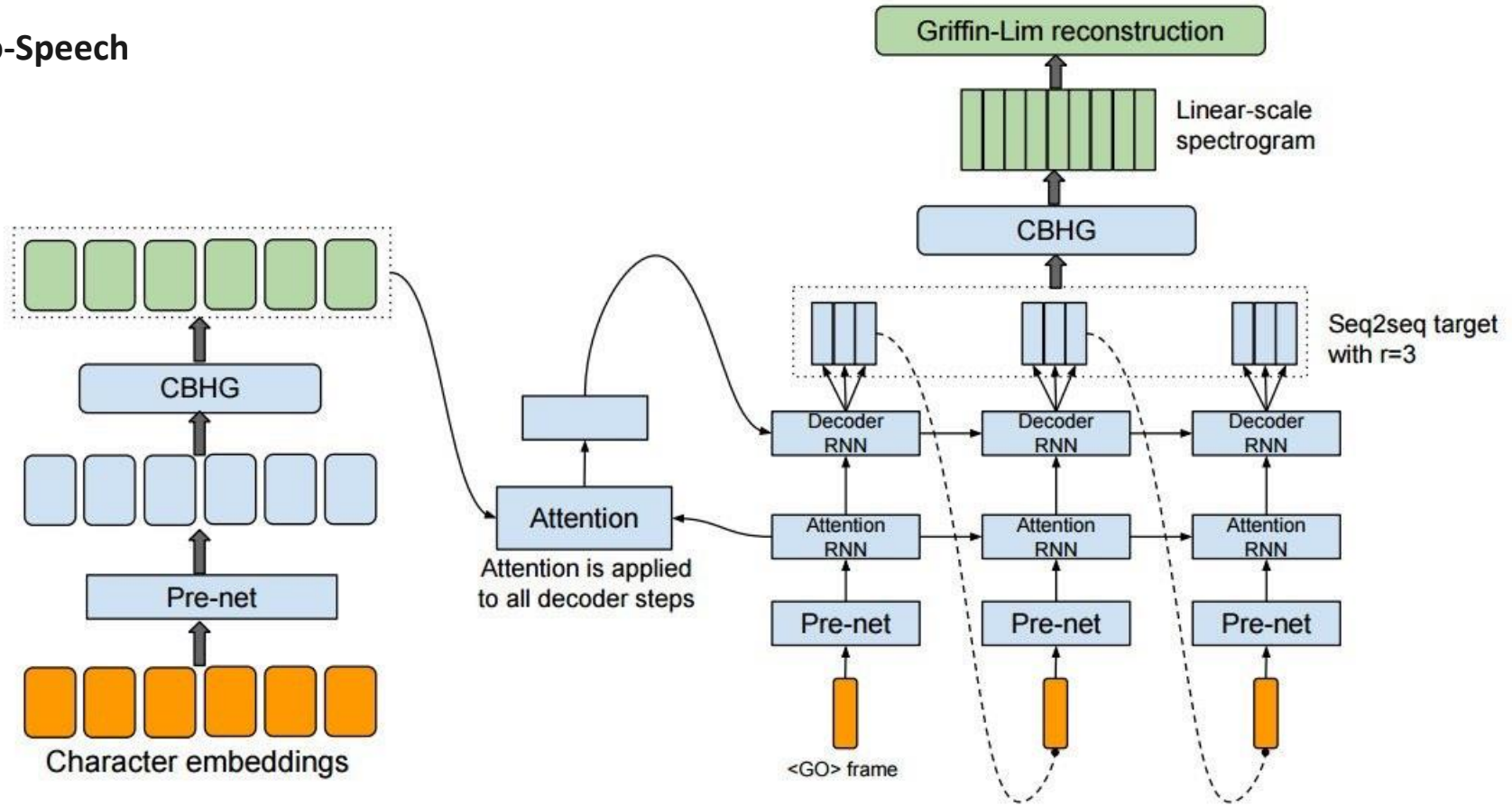
学习类似语言模型：$p(x) = \prod_{t=1}^{T} p(x_t | x_1, x_2, \ldots, x_{t-1})$

三、Neural nets for other domain applications

3. Audio and Speech

Tacotron2

音频生成, Text-to-Speech



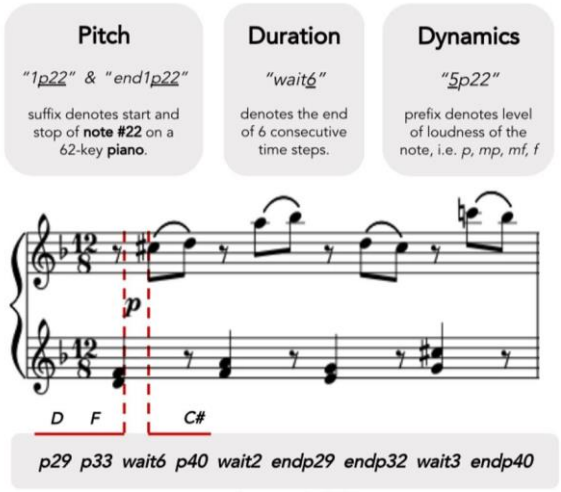
三、Neural nets for other domain applications

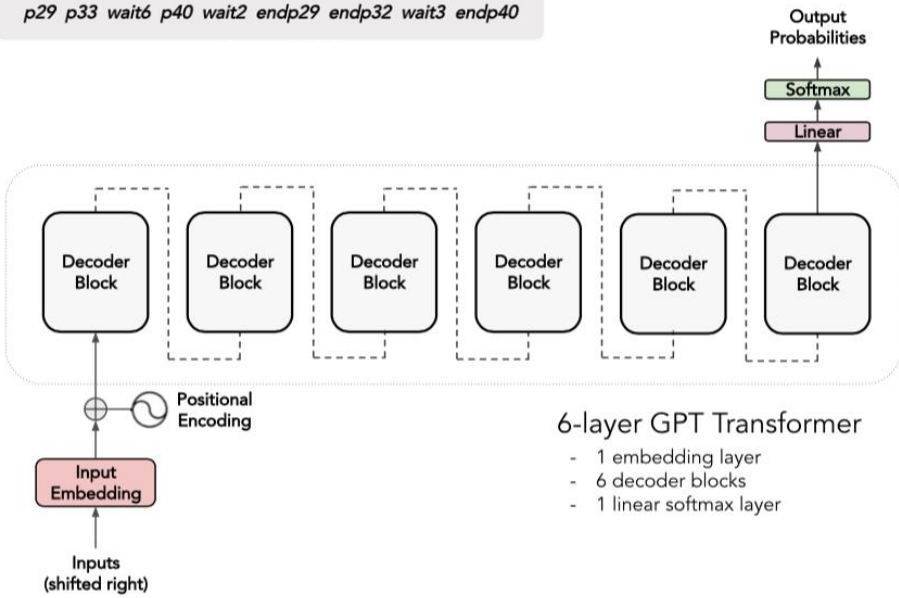
3. Audio and Speech

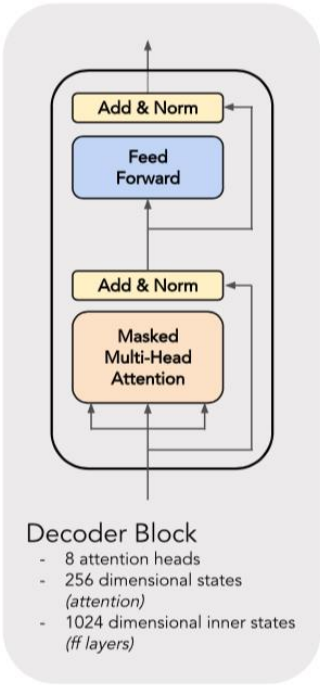
MuseNet ——OpenAI

生成音乐

- Pre-encoding raw data
- Tokenizing and sequencing
- Encoding







四、Deep learning programming framework

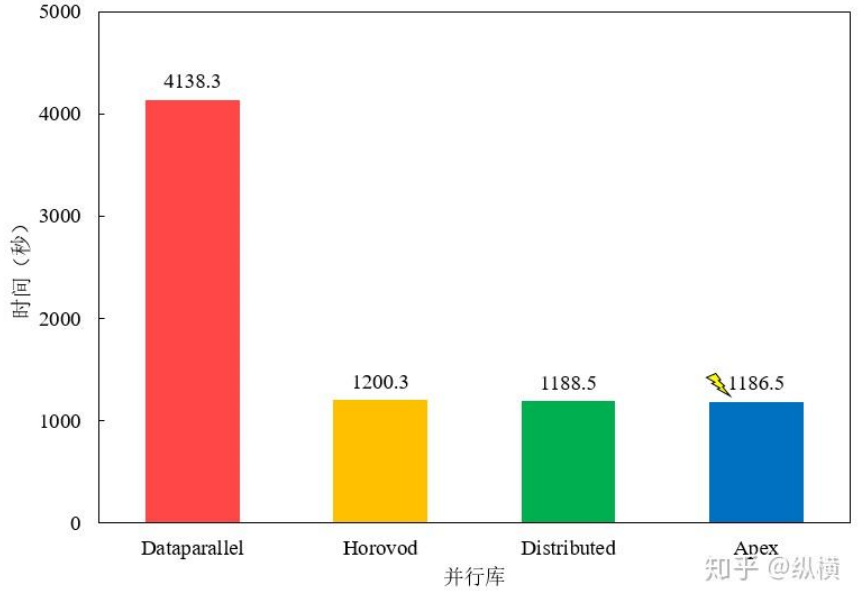
- 总体分为2大类
  - 底层基础框架
    - TensorFlow
    - Pytorch
    - Caffe
    - Theano
    - CNTK
    - Mxnet
    - Dynet
    - Paddle paddle
    - Chainer
    - Deeplearning4j
    - ...
  - 高层再加工/应用框架
    - Keras
    - Tensor2tensor
    - AllenNLP
    - OpenNMT
    - FastAI
    - ParlAI
    - ...

四、Deep learning programming framework

- 底层基础框架

	Language	CUDA support	Release year	Recommendation
TensorFlow	Python, C/C++, Java, Go, JavaScript, R, Julia, Swift	Yes	2015	93
Pytorch(Torch)	Python, C++, Julia	Yes	2002	100
Caffe	Python, MATLAB, C++	Yes	2013	89
Theano	Python	Yes	2007	60
CNTK	Python, C++	Yes	2016	76
Mxnet	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, Clojure	Yes	2015	80
Dynet	Python, C, Scala, Java	Yes	2017	90
Paddle paddle	Python	Yes	2016	90
Chainer	Python	Yes	2015	85
Deeplearning4j	Java, Scala, Clojure, Python, Kotlin	Yes	2014	73

四、Deep learning programming framework



- Examples in Pytorch
  - LSTM
    - (h, c) = LSTM(xs)
    - hs, (h, c) = LSTMCell(x, (h,c))
  - 并行训练方法
    - DataParallel

      model = nn.DataParallel(model.cuda(), device_ids=gpus, output_device=gpus[0])
    - torch.distributed

      dist.init_process_group(backend='nccl')
      model = torch.nn.parallel.DistributedDataParallel(model, device_ids=[args.local_rank])
      …
    - torch. multiprocessing
    - NVIDIA **Apex**
    - Uber **Horovod**

四、Deep learning programming framework

- 高层再加工/应用框架

	Language	Field	Underlying Framework	Recommendation
Keras	Python,	NLP/CV	TensorFlow/Theano	92
Tensor2tensor	Python, C++, Julia	NLP/CV	TensorFlow	85
AllenNLP	Python, C++	NLP	PyTorch	95
OpenNMT	Python	NLP-NMT	PyTorch/TensorFlow	90
FastAI	Python, C++	NLP	PyTorch	86
ParlAI	Python,	NLP-Dialog	PyTorch	80

四、Deep learning programming framework

please notice

- 所有的计算操作在框架中都是batch-wise的.
- 很多step-wise的计算操作都被并行化.
- 尽可能地使用CUDA进行计算.
- 多学习优秀的开源代码，模仿风格和用法.
- 熟练掌握某一个框架下尽可能多的API.

MISC

- 本tutorial只是神经网络模型的一个浅层的介绍，没涉及到具体领域下的具体任务。若与不同领域下的不同的任务结合起来将会有非常多的组合。
- 还有还有各种学习范式而结合的各种机器学习的细节，这些才是发论文搞科研的关键。
- 学习DL，最推荐的方式就是确定任务后快速上手代码，跑通demo。