Generative Models: Fundamentals and Applications

Lecture 6: Auto-Regressive Networks

Shuigeng Zhou, Yuxi Mi
College of CSAI

November 3, 2025

目录

- 线性自回归网络
- 神经自回归网络
- 循环神经网络
- Transformer

自回归

- 自回归 (Auto-Regressive)
  - 一种处理时间序列的统计学方法
    - 用变量之前各个时刻的观测值预测当前时刻的观测值
      - 线性自回归模型：假设它们为线性关系

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

其中，ε_t表示t时刻的随机误差

观测数据的有序排列

- 链式法则

$$p(x_1, x_2, \dots, x_D) = p(x_1)p(x_2|x_1) \dots p(x_D|x_1, \dots, x_{D-1})$$

  - 观测数据往往是有序的
    - “顺序”可以是时间，也可以被人为定义
      - 例如：对于图像，定义——左侧的像素位于右侧的像素之前，或顶部的像素在底部的像素之前

按行排序

x_1	x_2	x_3	x_4
x_5	x_6		

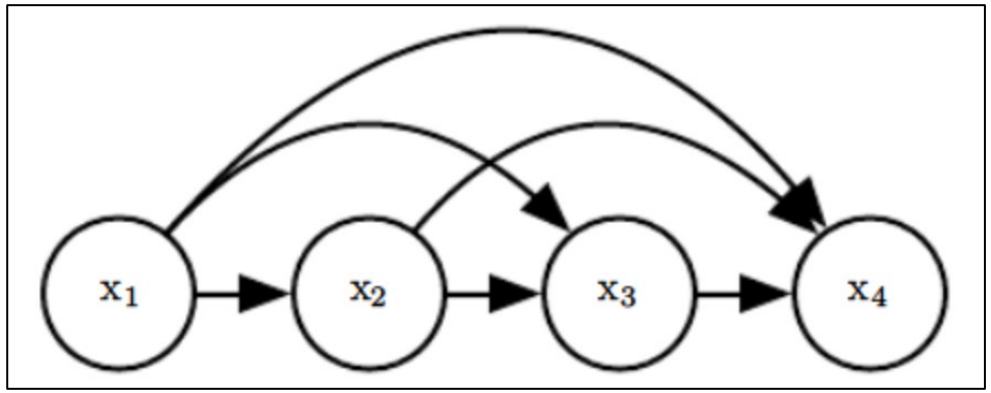
按列排序

x_1	x_5		
x_2	x_6		
x_3			
x_4			

观测数据的有序排列

- 概率图模型
  - E.g. 4维数据

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)$$



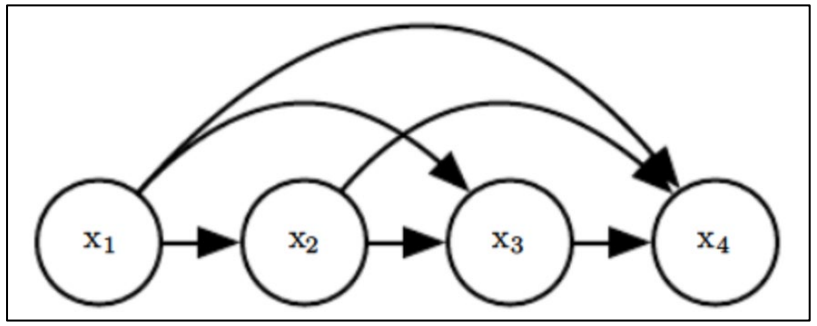
自回归网络

- 自回归网络：没有潜在随机变量的有向概率模型
  - 用条件概率 $p(x_i|x_{<i})$ 表示可见层相邻元素的关系
  - 用条件概率乘积表示联合概率分布

$$p(x_1, x_2, \dots, x_D) = \prod_{i=1}^{D} p(x_i|x_{<i})$$

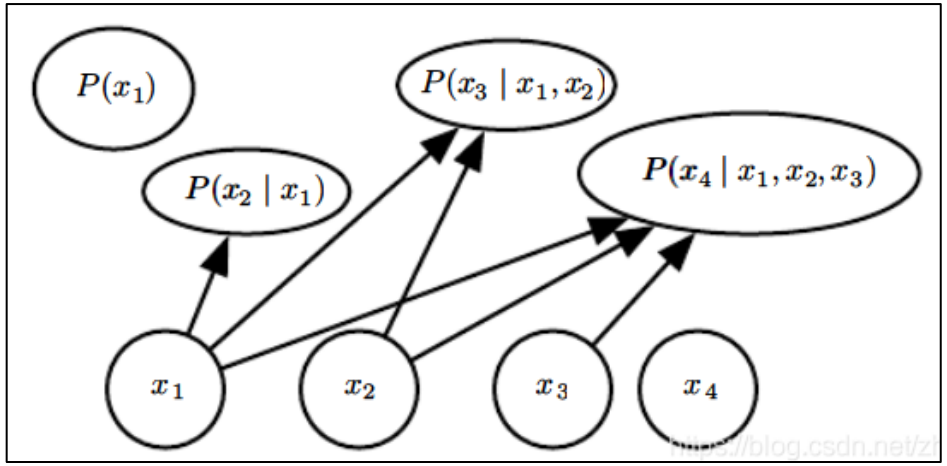
  - $x_{<i} = x_1, \dots, x_{i-1}$

x_1	x_2	x_3	x_4
x_5	x_6		



自回归网络

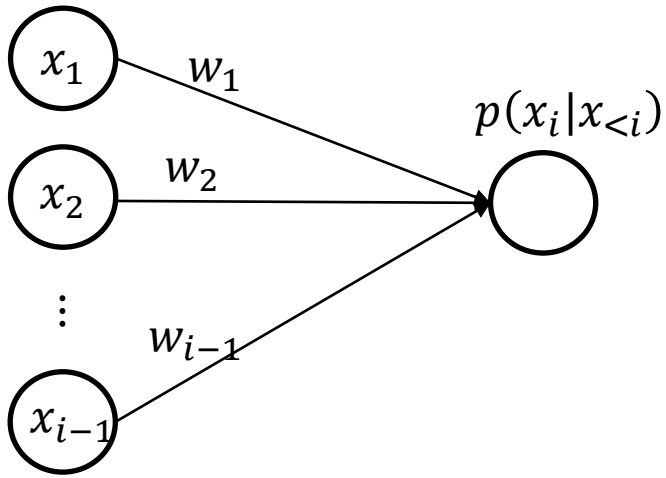
- 关键：求 $p(x_i|x_{<i})$
  - 构造神经网络来逼近条件概率$p(x_i|x_{<i})$



线性自回归网络

- 每个条件概率$p(x_i|x_{<i})$均被定义为线性模型
  - x_i是实值数据
    - 线性回归

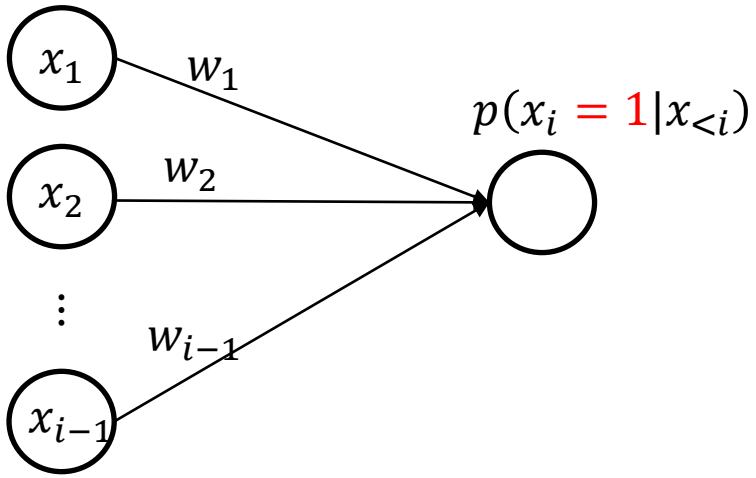
$$p(x_i|x_{<i}) = b + w_1 x_1 + \cdots + w_{i-1} x_{i-1}$$



线性自回归网络

- 每个条件概率$p(x_i|x_{<i})$均被定义为线性模型
  - $x_i = 0,1$
    - Logistic 回归

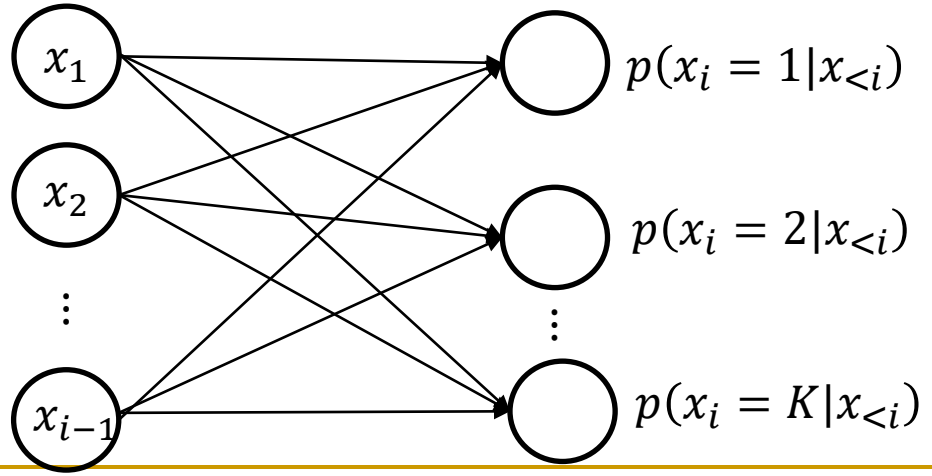
$$p(x_i = 1|x_{<i}) = \sigma\left(b + w_1 x_1 + \cdots + w_{i-1} x_{i-1}\right)$$



线性自回归网络

- 每个条件概率 $p(x_i|x_{<i})$ 均被定义为线性模型
  - $x_i = 1,2,\ldots,K$
    - Softmax 回归

$$p(x_i = k|x_{<i}) = \frac{\exp\{b_k + w_{k,1}x_1 + \cdots + w_{k,i-1}x_{i-1}\}}{\sum_{k'=1}^{K} \exp\{b_{k'} + w_{k',1}x_1 + \cdots + w_{k',i-1}x_{i-1}\}}$$



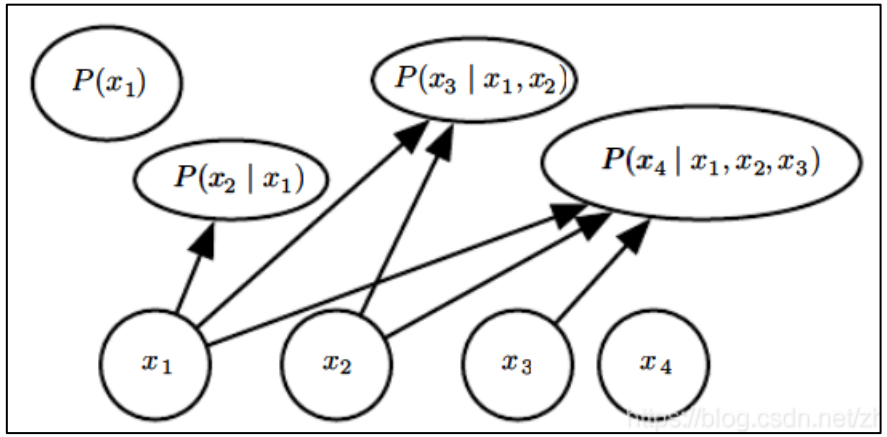
实例：MNIST数据集

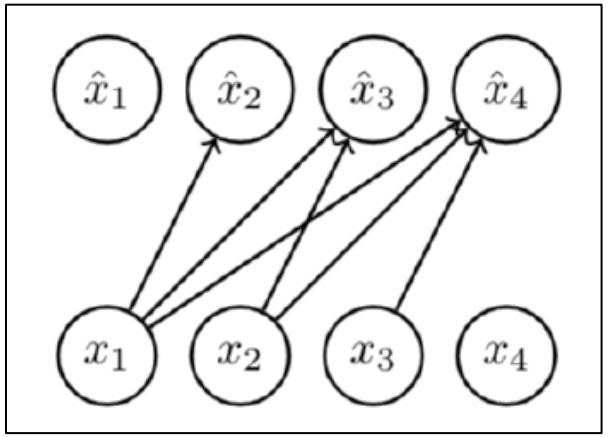
- 每张图片维度D=28×28=784
- 每个像素点值为$x_i = \{0,1\}$
- 如何计算$p(x) = p(x_1, x_2, \ldots, x_D)$?

$$p(x_1, x_2, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | x_{<i})$$

实例：MNIST数据集

- 计算 $p(x_i|x_{<i})$
  - 令$\hat{x}_1 = p(x_1 = 1) = w^1$ ← 伯努利分布
  - 令$\hat{x}_i = p(x_i = 1|x_{<i}) = \sigma(w_0^i + \sum_{j=1}^{i-1} w_j^i x_j)$ ← Logistic回归





实例：MNIST数据集

- 计算 $p(x) = p(x_1, x_2, \ldots, x_D)$
  - E.g.: 计算$p(x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0)$

$$p(x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0) = (1 - \hat{x}_1) \times \hat{x}_2 \times \hat{x}_3 \times (1 - \hat{x}_4)$$
$$= (1 - \hat{x}_1) \times \hat{x}_2(x_1 = 0) \times \hat{x}_3(x_1 = 0, x_2 = 1) \times (1 - \hat{x}_4)(x_1 = 0, x_2 = 1, x_3 = 1)$$

  - E.g.: 计算$p(x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0)$

$$p(x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0) = \hat{x}_1 \times \hat{x}_2 \times (1 - \hat{x}_3) \times (1 - \hat{x}_4)$$
$$= \hat{x}_1 \times \hat{x}_2(x_1 = 1) \times (1 - \hat{x}_3)(x_1 = 1, x_2 = 1) \times (1 - \hat{x}_4)(x_1 = 1, x_2 = 1, x_3 = 1)$$

  - 只有神经网络参数未知
- 通过极大似然估计来评估模型参数

实例：MNIST数据集

- 如何从 $p(x_1, x_2, \ldots, x_D)$ 采样新数据？
  - 采样$\bar{x}_1 \sim p(x_1)$　随机采样[1,0]，p=$[\hat{x}_1, 1-\hat{x}_1]$
  - 采样$\bar{x}_2 \sim p(x_2|x_1 = \bar{x}_1)$
  - 采样$\bar{x}_3 \sim p(x_3|x_1 = \bar{x}_1, x_2 = \bar{x}_2)$
  - ⋮
  - 采样$\bar{x}_D \sim p(x_D|x_1 = \bar{x}_1, x_2 = \bar{x}_2, \ldots, x_{D-1} = \bar{x}_{D-1})$

- 注意：采样顺序不可交换

线性自回归网络

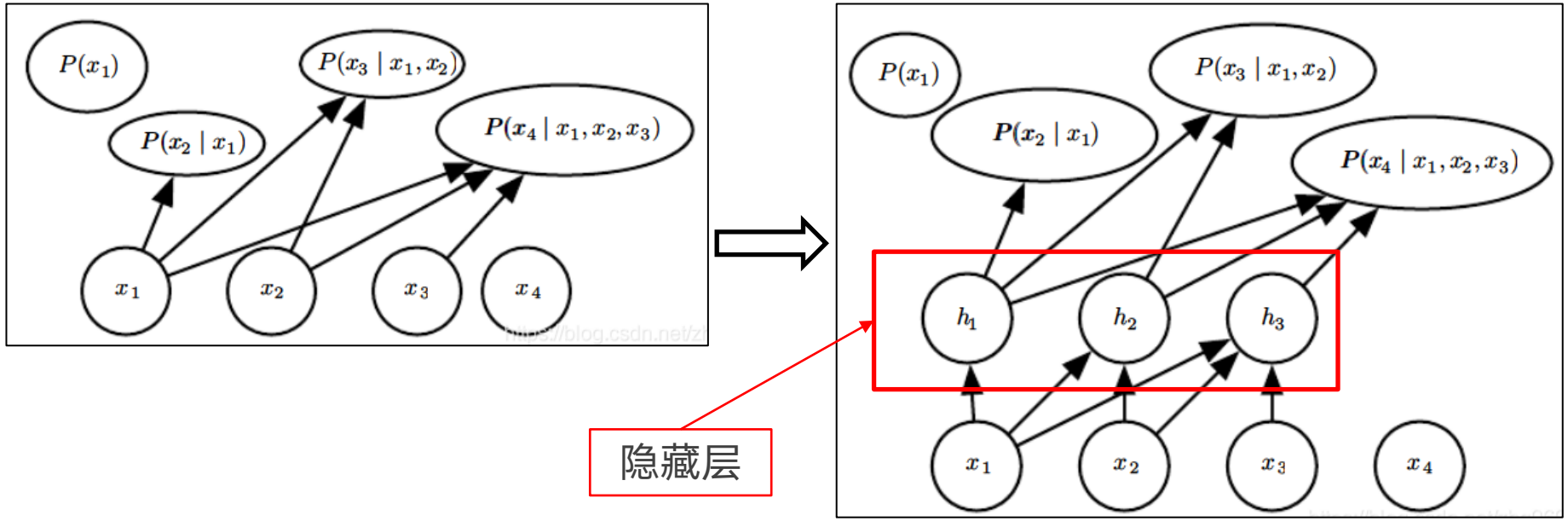
- 参数个数 $O(D^2)$
  - x_i是实值数据
    - $p(x_i|x_{<i}) = b + w_1 x_1 + \cdots + w_{i-1} x_{i-1}$
  - $x_i = 0,1$
    - $p(x_i|x_{<i}) = \sigma\,(b + w_1 x_1 + \cdots + w_{i-1} x_{i-1})$
  - $x_i = 1,2,\ldots,K$
    - $p(x_i = k|x_{<i}) = \frac{\exp\{b_k + w_{k,1} x_1 + \cdots + w_{k,i-1} x_{i-1}\}}{\sum_{k'=1}^{K} \exp\{b_{k'} + w_{k',1} x_1 + \cdots + w_{k',i-1} x_{i-1}\}}$
- 缺点：
  - 线性模型容量有限，拟合函数的能力不足

目录

- 线性自回归网络
- 神经自回归网络
- 循环神经网络
- Transformer

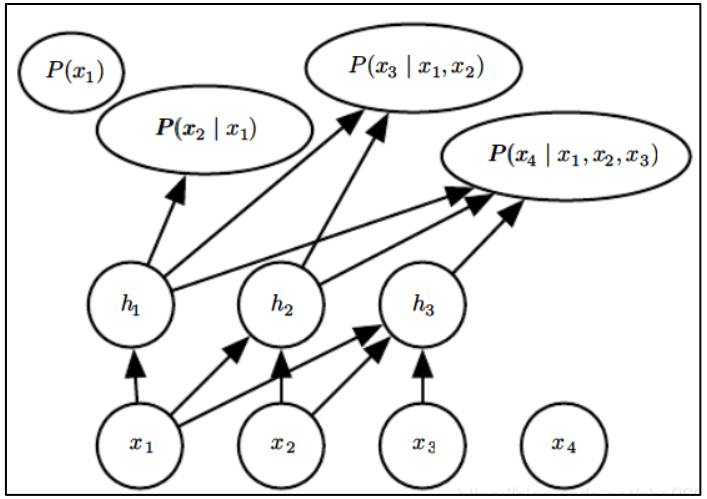
神经自回归网络

- 神经自回归网络：用神经网络建模 $p(x_i|x_{<i})$
  - 可以任意增加容量，理论上可以拟合任意联合分布



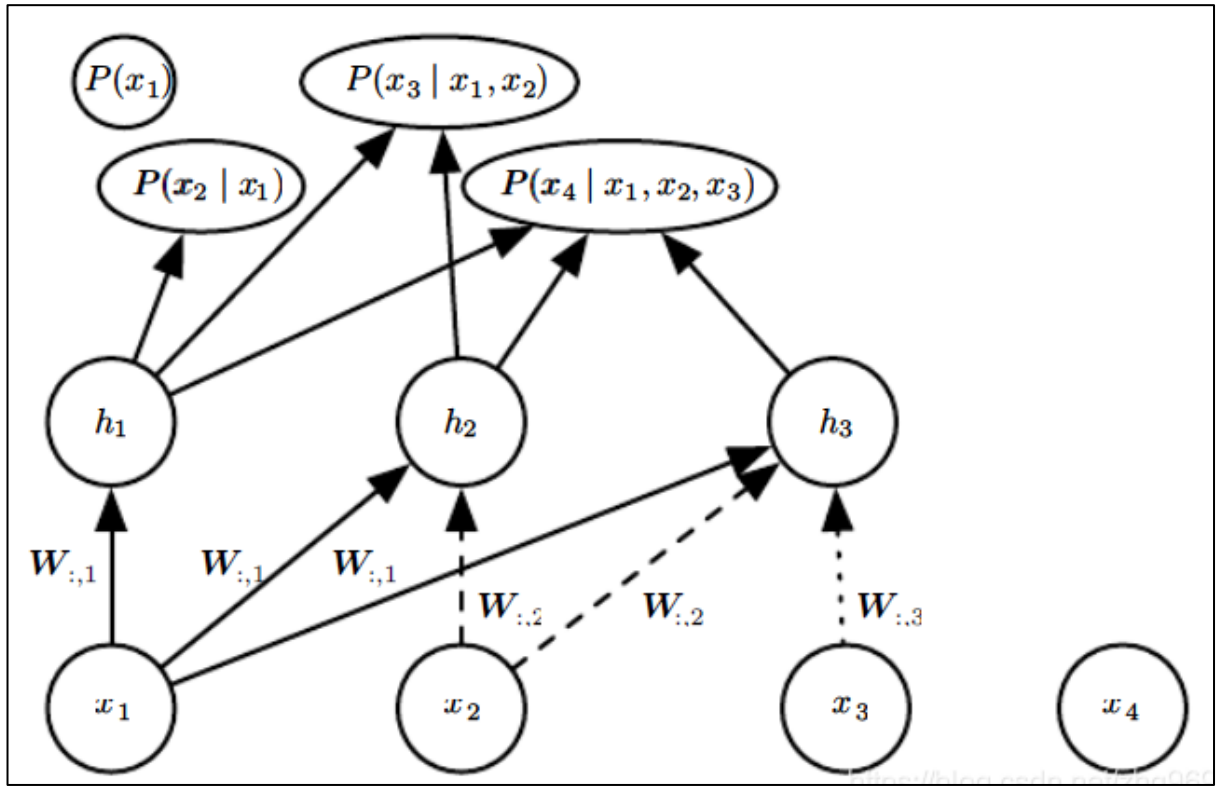
神经自回归网络

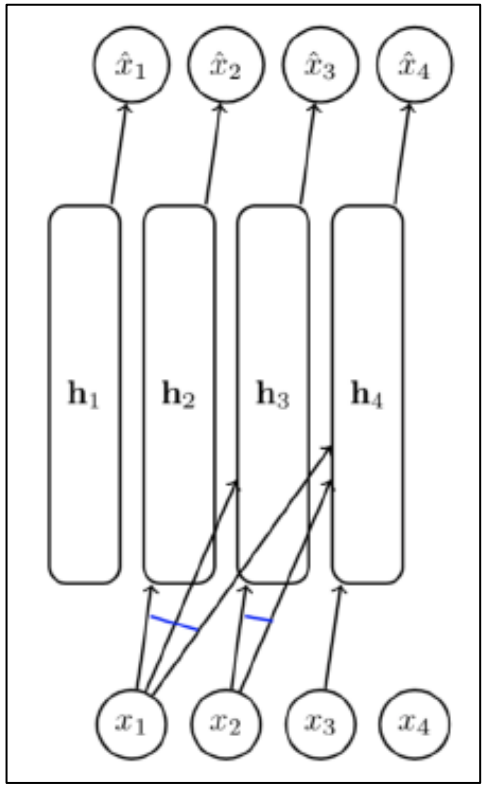
- 关键：求 $p(x_i|x_{<i})$
  - 假设 $x_i = 1,2,\ldots,K$
  - 神经网络具有 $(i-1)$ 个输入和 K 个输出
  - 网络的前向计算可以共享
    - 不需要对每个 x_i 使用不同的神经网络
    - 意味着预测x_i所计算的隐藏层特征可以复用于预测x_j $(j > i)$



神经自回归分布估计(NADE)

- 不同组 j 的隐藏单元：参数共享





神经自回归分布估计(NADE)

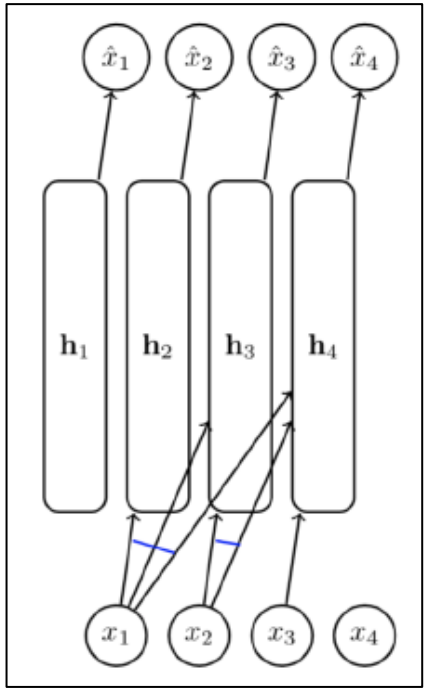
- $x_i = 0,1$
  - 计算隐藏层

$$\mathbf{h}_i = \sigma(A_i \mathbf{x}_{<i} + \mathbf{c}_i)$$

  - 计算 $p(x_i|x_{<i})$

模型参数

$$\hat{x}_i = p(x_i|x_1, \cdots, x_{i-1}; A_i, \mathbf{c}_i, \boldsymbol{\alpha}_i, b_i)$$
$$= \sigma(\boldsymbol{\alpha}_i \mathbf{h}_i + b_i)$$



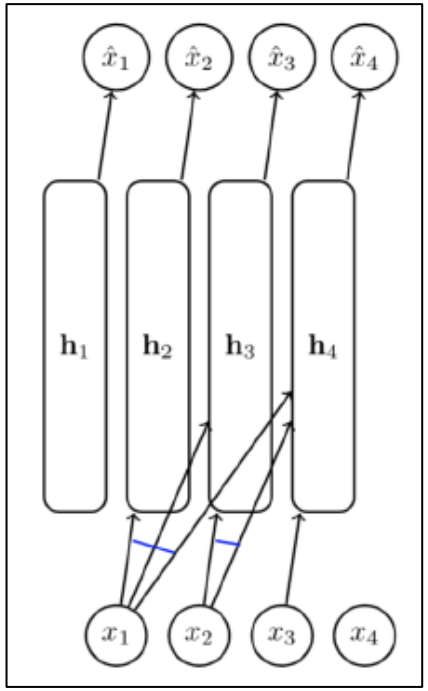
神经自回归分布估计(NADE)

- $x_i = 0,1$
  - 参数共享

$$\mathbf{h}_i = \sigma(A_i \mathbf{x}_{<i} + \mathbf{c}_i) \Longrightarrow \mathbf{h}_i = \sigma(W_{\cdot,<i} \mathbf{x}_{<i} + \mathbf{c})$$

$$\mathbf{h}_2 = \sigma\left(\underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{A_2} x_1 + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c_2}\right) \Longrightarrow \mathbf{h}_2 = \sigma\left(\underbrace{\begin{pmatrix}\vdots \\ w_1 \\ \vdots\end{pmatrix}}_{A_2} x_1 + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c}\right)$$

$$\mathbf{h}_3 = \sigma\left(\underbrace{\begin{pmatrix}\vdots & \vdots\end{pmatrix}}_{A_3} \begin{pmatrix}x_1 \\ x_2\end{pmatrix} + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c_3}\right) \Longrightarrow \mathbf{h}_3 = \sigma\left(\underbrace{\begin{pmatrix}\vdots & \vdots \\ w_1 & w_2 \\ \vdots & \vdots\end{pmatrix}}_{A_3} \begin{pmatrix}x_1 \\ x_2\end{pmatrix} + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c}\right)$$



神经自回归分布估计(NADE)

- 参数共享

$$\mathbf{h}_i = \sigma(W_{\cdot,<i}\mathbf{x}_{<i} + \mathbf{c})$$

  - 减少了模型参数，加快模型计算

$$\begin{aligned}
&(W_{:,<i+1}x_{<i+1} + \boldsymbol{c}) - (W_{:,<i}x_{<i} + \boldsymbol{c}) \\
&= (W_{:,1} \quad \cdots \quad W_{:,i-1} \quad W_{:,i}) \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_i \end{pmatrix} - (W_{:,1} \quad \cdots \quad W_{:,i-1}) \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \end{pmatrix} \\
&= W_{:,i}x_i
\end{aligned}$$

    - 计算$\mathbf{h}_{i+1} \in \mathbb{R}^H$的时间复杂度为$O(H)$
    - 计算$p(x)$的时间复杂度为$O(DH)$

神经自回归分布估计(NADE)

- $x_i = 1,2,..,K$
  - 参数共享

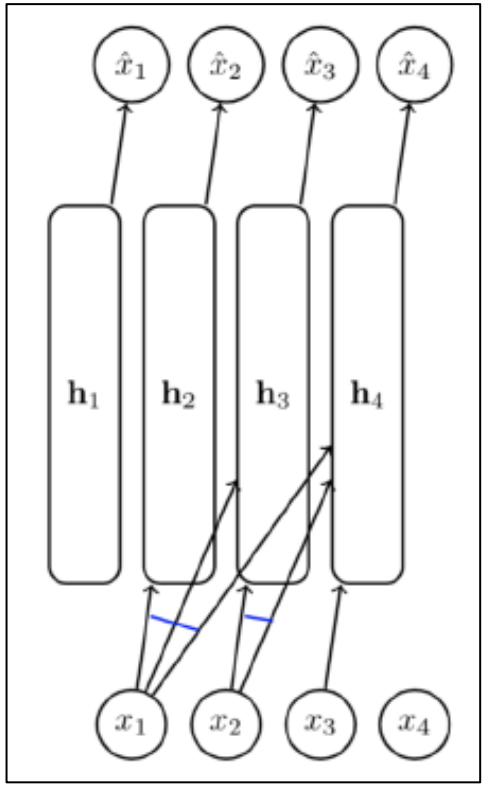
$$\mathbf{h}_i = \sigma(W_{.,<i}\mathbf{x}_{<i} + \mathbf{c})$$

  - 类别分布

$$p(x_i|x_1,\cdots,x_{i-1}) = Cat(p_i^1,\cdots,p_i^K)$$

$$\hat{\mathbf{x}}_i = (p_i^1,\cdots,p_i^K) = softmax(X_i\mathbf{h}_i + \mathbf{b}_i)$$

    - $X_i \in K \times H$：系数矩阵

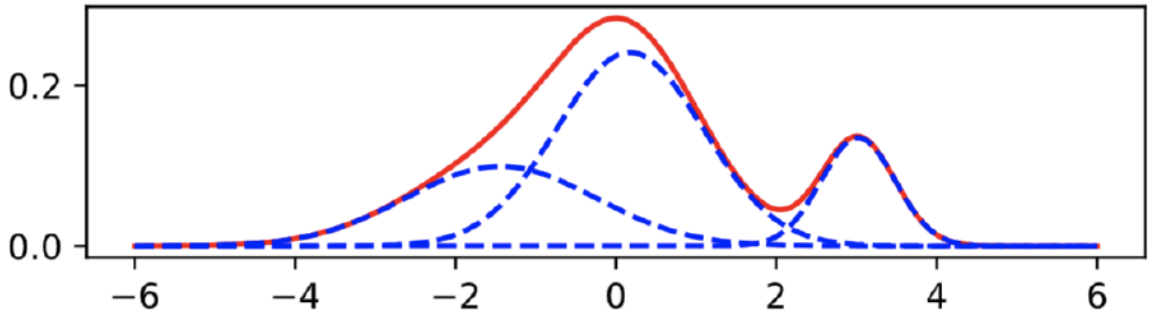


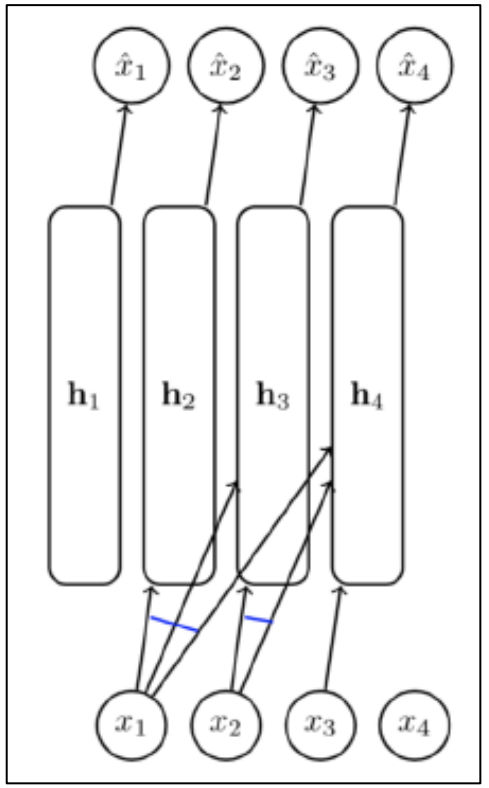
实值神经自回归密度估计(RNADE)

- $x_i \in \mathbb{R}$
- 概率密度函数

$$p(\boldsymbol{x}) = \prod_{d=1}^{D} p(x_d \,|\, \boldsymbol{x}_{<d})$$

  - 关键：$p(x_d|x_{<d})$
    - 高斯混合模型





实值神经自回归密度估计(RNADE)

- 参数共享

$$\boldsymbol{a}_d = \boldsymbol{W}_{\cdot,<d}\boldsymbol{x}_{<d} + \boldsymbol{c}$$

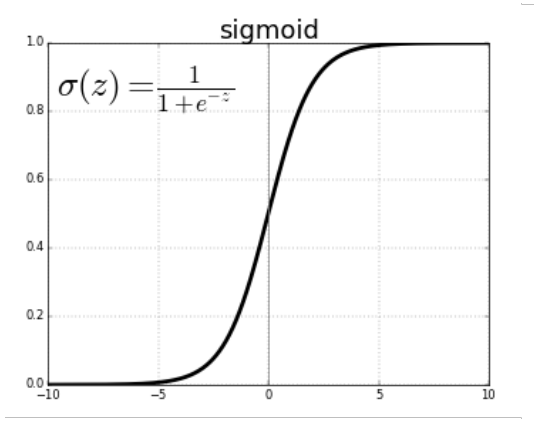
  - 激活函数
    - 例如，Sigmoid

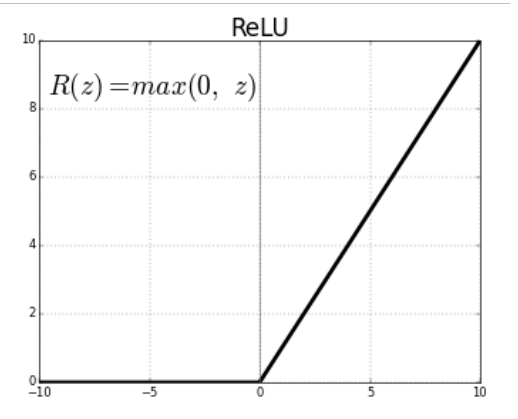
$$\boldsymbol{h}_d = \text{sigm}\left(\rho_d \boldsymbol{a}_d\right)$$

    - 例如，ReLU

$$\boldsymbol{h}_d = \begin{cases} \rho_d \boldsymbol{a}_d & \text{if } \rho_d \boldsymbol{a}_d > 0 \\ 0 & \text{otherwise.} \end{cases}$$

      - ρ_d：缩放系数





实值神经自回归密度估计(RNADE)

- 条件概率密度函数

$$p(x_d|x_{<d}) = \sum_{j=1}^{K} \alpha_{d,j}\, \mathcal{N}(x_d; \mu_{d,j}, \sigma_{d,j})$$

  - α_d：各个单高斯分布的混合权重——归一化

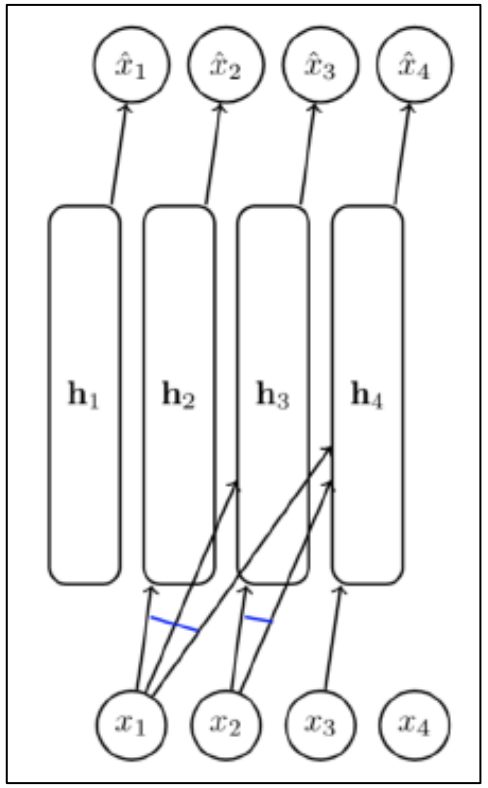
$$\boldsymbol{\alpha}_d = \text{softmax}\left(\boldsymbol{V}_d^{\alpha\top}\boldsymbol{h}_d + \boldsymbol{b}_d^{\alpha}\right)$$

  - μ_d：混合高斯分布的均值

$$\boldsymbol{\mu}_d = \boldsymbol{V}_d^{\mu\top}\boldsymbol{h}_d + \boldsymbol{b}_d^{\mu}$$

  - σ_d：混合高斯分布的标准差——确保为正

$$\boldsymbol{\sigma}_d = \exp\left(\boldsymbol{V}_d^{\sigma\top}\boldsymbol{h}_d + \boldsymbol{b}_d^{\sigma}\right)$$



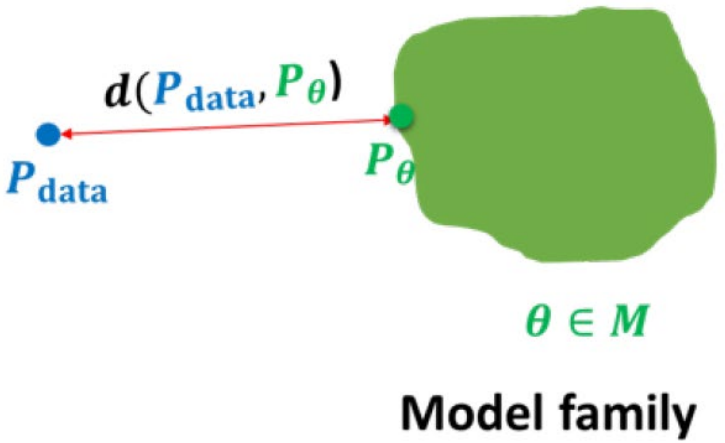
模型估计

- 给定n个独立同分布的数据

$$x_1, x_2, \ldots, x_n \sim p_{data}(x)$$

- 令$p_\theta(x)$是估计出得概率模型

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



模型估计

- 最小化两个分布间的距离

$$\min D_{KL}(p_{data}(x) \| p_\theta(x)) = \mathbb{E}_{p_{data}}\left[\log \frac{p_{data}(x)}{p_\theta(x)}\right]$$

$$= \mathbb{E}_{p_{data}} \log p_{data}(x) - \mathbb{E}_{p_{data}} p_\theta(x)$$

$$= constant - L(\theta)$$

  - 等价于极大似然估计

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i) \rightarrow \mathbb{E}_{p_{\text{data}}}[\log p_\theta(x)]$$

模型估计

- 最小化负对数似然函数

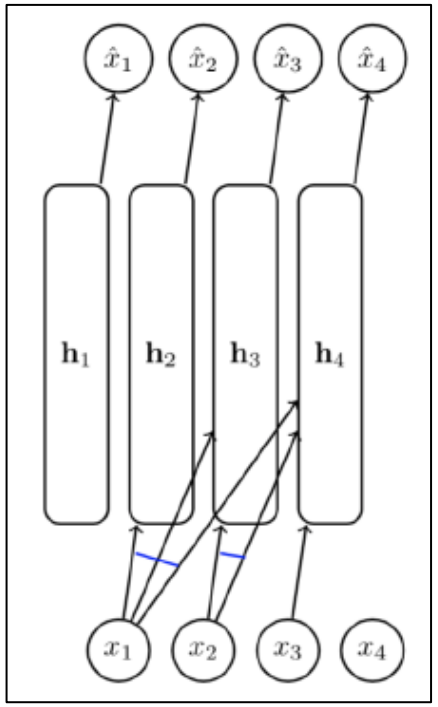
$$\min \frac{1}{n}\sum_{i=1}^{n} -\log p(\mathbf{x}_i) = \frac{1}{n}\sum_{i=1}^{n}\sum_{d=1}^{D} -\log p(x_{i,d}|x_{i,<d})$$

  - 求解：随机梯度下降法 (SGD)

$$\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_\theta \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i)$$

模型估计

- 离散NADE算法：伪代码



Algorithm 1 Computation of $p(\mathbf{v})$ and learning gradients for NADE

Input: training observation vector $\mathbf{v}$

Output: $p(\mathbf{v})$ and gradients of $-\log p(\mathbf{v})$ on parameters

Computing $p(\mathbf{v})$
$\mathbf{a} \leftarrow \mathbf{c}$
$p(\mathbf{v}) \leftarrow 0$
for i from 1 to D **do**
  $\mathbf{h}_i \leftarrow \text{sigm}(\mathbf{a})$
  $p(v_i = 1|\mathbf{v}_{<i}) \leftarrow \text{sigm}(b_i + \mathbf{V}_{i,.}\mathbf{h}_i)$
  $p(\mathbf{v}) \leftarrow p(\mathbf{v})\,(p(v_i = 1|\mathbf{v}_{<i})^{v_i} + (1 - p(v_i = 1|\mathbf{v}_{<i}))^{1-v_i})$
  $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{W}_{.,i}v_i$
end for

前向计算

Computing gradients of $-\log p(\mathbf{v})$
$\delta\mathbf{a} \leftarrow 0$
$\delta\mathbf{c} \leftarrow 0$
for i from D to 1 **do**
  $\delta b_i \leftarrow (p(v_i = 1|\mathbf{v}_{<i}) - v_i)$
  $\delta\mathbf{V}_{i,.} \leftarrow (p(v_i = 1|\mathbf{v}_{<i}) - v_i)\,\mathbf{h}_i^\top$
  $\delta\mathbf{h}_i \leftarrow (p(v_i = 1|\mathbf{v}_{<i}) - v_i)\,\mathbf{V}_{i,.}^\top$
  $\delta\mathbf{c} \leftarrow \delta\mathbf{c} + (\delta\mathbf{h}_i)\mathbf{h}_i(1 - \mathbf{h}_i)$
  $\delta\mathbf{W}_{.,i} \leftarrow (\delta\mathbf{a})v_i$
  $\delta\mathbf{a} \leftarrow \delta\mathbf{a} + (\delta\mathbf{h}_i)\mathbf{h}_i(1 - \mathbf{h}_i)$
end for

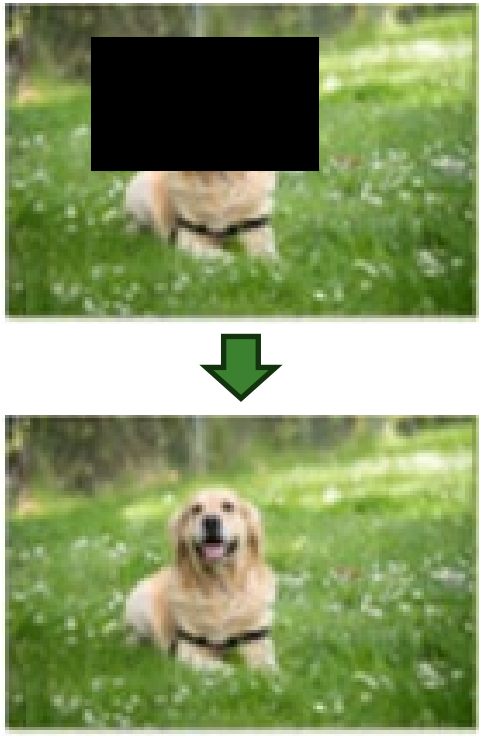
反向传播

return $p(\mathbf{v}), \delta\mathbf{b}, \delta\mathbf{V}, \delta\mathbf{c}, \delta\mathbf{W}$

NADE：扩展

- NADE-k
  - 推断数据中的缺失数据
    - 因式化条件分布

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}}) = \prod_{i \in \text{mis}} p_{\boldsymbol{\theta}}(x_i \mid \mathbf{x}_{\text{obs}})$$



    - 自回归模型

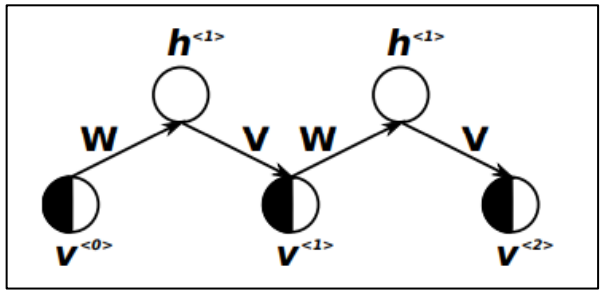
$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid o) = \prod_{d=1}^{D} p_{\boldsymbol{\theta}}(x_{o_d} \mid \mathbf{x}_{o_{<d}}),$$

    - 训练时，不再遵循固定的索引顺序

NADE：扩展

- NADE-k
  - 条件模型

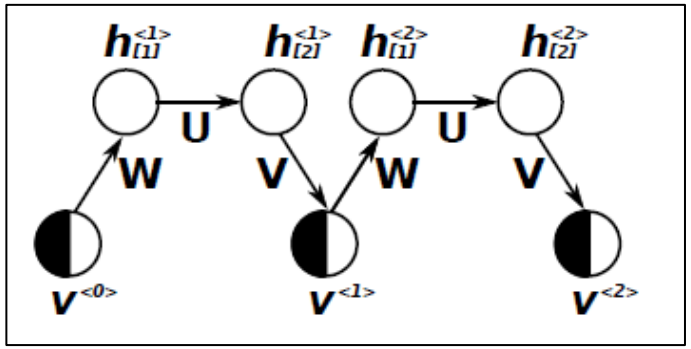
$$p_{\boldsymbol{\theta}}(\mathbf{x}_{\text{mis}} \mid \mathbf{x}_{\text{obs}}) = \prod_{i \in \text{mis}} p_{\boldsymbol{\theta}}(x_i \mid \mathbf{x}_{\text{obs}})$$



    - 深度前馈神经网络：nk层
      - 可见层和隐藏层间反复迭代

$$\mathbf{v}^{\langle 0 \rangle} = \mathbf{m} \odot \mathbb{E}_{\mathbf{x} \in \text{data}}[\mathbf{x}] + (1 - \mathbf{m}) \odot \mathbf{x}$$

缺失部分　　观测部分

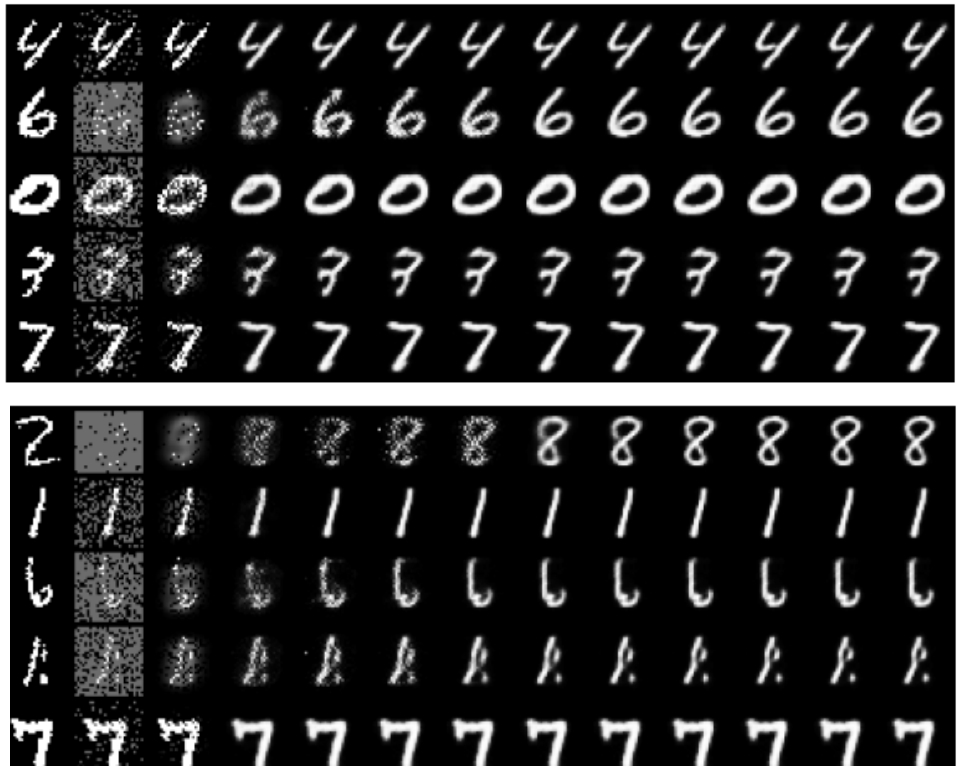


$$\mathbf{h}^{\langle t \rangle} = \phi(\mathbf{W}\mathbf{v}^{\langle t-1 \rangle} + \mathbf{c})$$
$$\mathbf{v}^{\langle t \rangle} = \mathbf{m} \odot \sigma(\mathbf{V}\mathbf{h}^{\langle t \rangle} + \mathbf{b}) + (1 - \mathbf{m}) \odot \mathbf{x}$$

更新后的可见层

NADE：扩展

- NADE-k
  - MNIST上的实验结果



NADE：扩展

- 深度NADE
  - 多个隐藏层
  - 能够通过随机抽样处理任意顺序的数据，将信息提供给指定用于观测某些输入的隐藏单元并预测缺失信息

$$p_{\text{ensemble}}(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} p(\mathbf{x} \mid o^{(i)})$$

目录

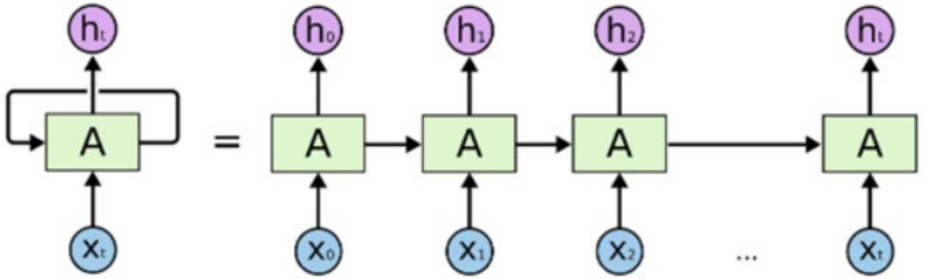
- 线性自回归网络
- 神经自回归网络
- 循环神经网络
- Transformer

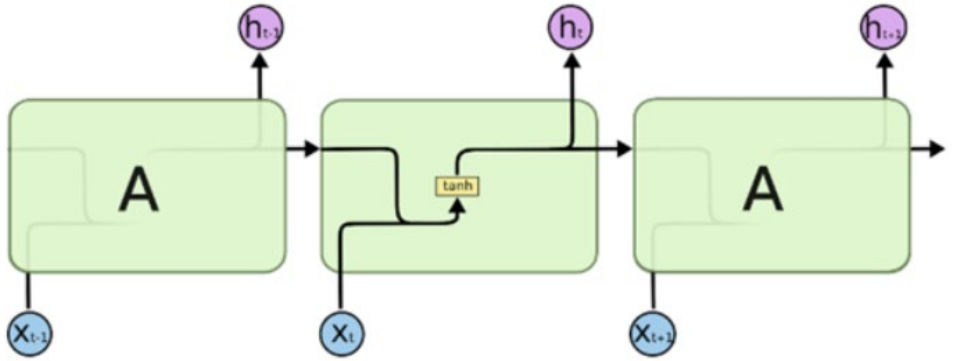
循环神经网络(RNN)

- 循环神经网络（Recurrent Neural Network，RNN）
  - 一类具有短期记忆能力的神经网络
  - 网络会记忆和传递前一时刻隐藏状态
  - 在网络结构上
    - 隐藏层间节点间相互连接
    - 隐藏层接收当前输入的结果、上一时刻隐藏层的输出

循环神经网络(RNN)

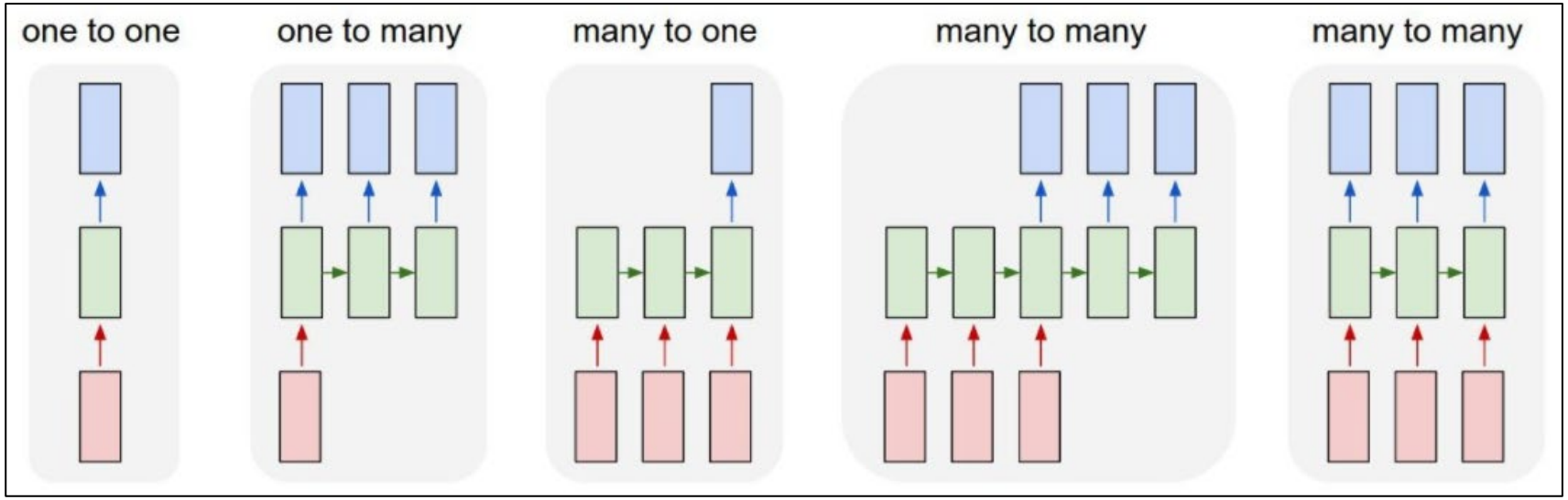
- RNN网络结构
  - 具有短期记忆网络的结构





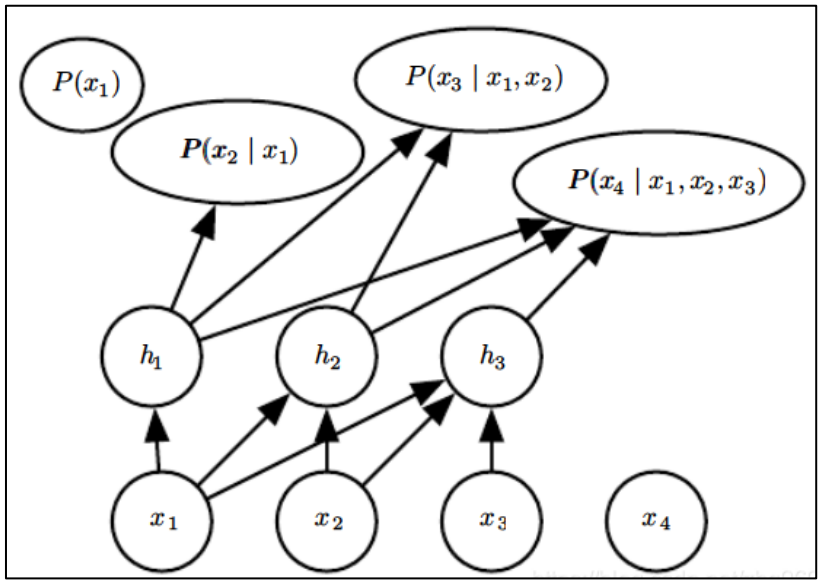
循环神经网络(RNN)

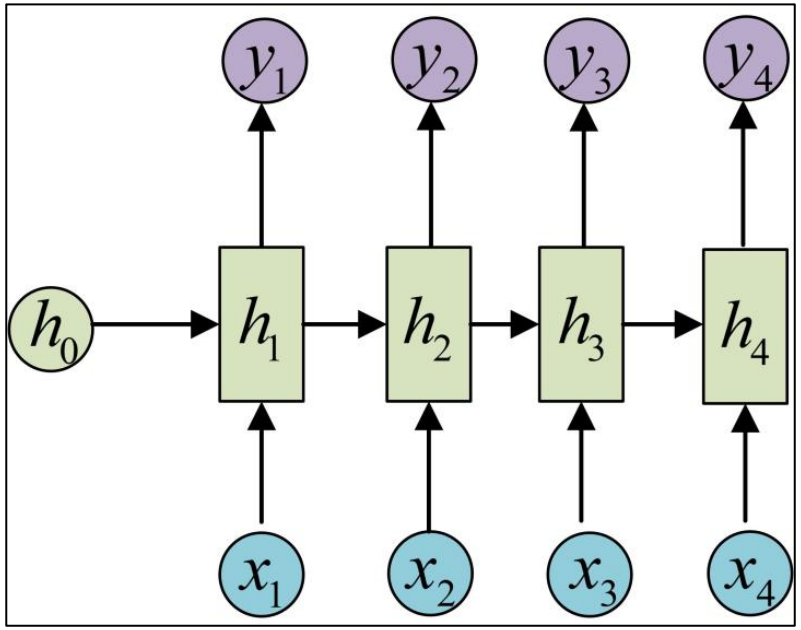
- 定长输入和输出 (e.g. 图像分类)
- 输入定长向量，输出序列 (e.g.图像生成字幕)
- 输入序列，输出定长向量 (e.g. 情感分类)
- 异步的序列输入和输出 (e.g.文本翻译)
- 同步的序列输入和输出 (e.g. 语音识别、视频逐帧标注)



循环神经网络(RNN)

- 同步的序列输入和输出



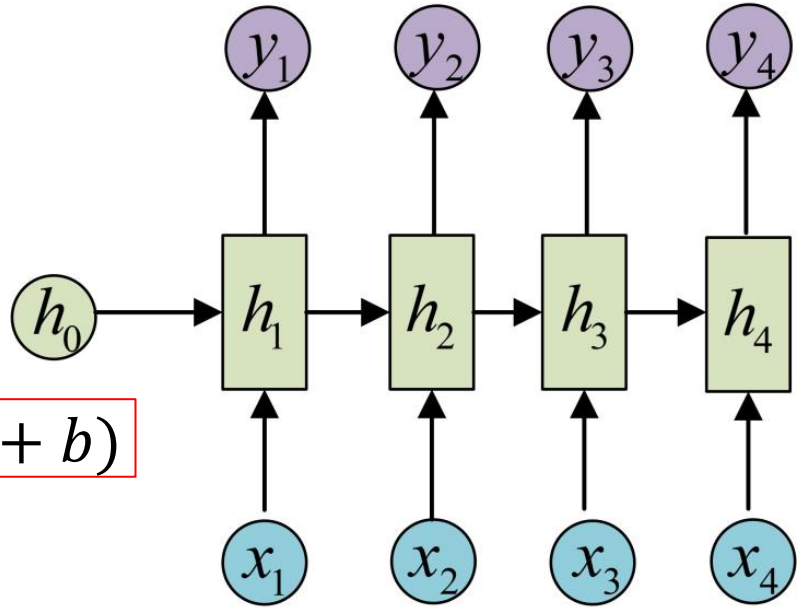


循环神经网络(RNN)

- 如何计算 $p(x_i|x_{<i})$
  - 隐藏层

$$h_0 = b_0$$

$$h_i = \tanh(W_{hh}h_{i-1} + W_{xh}x_i + b)$$

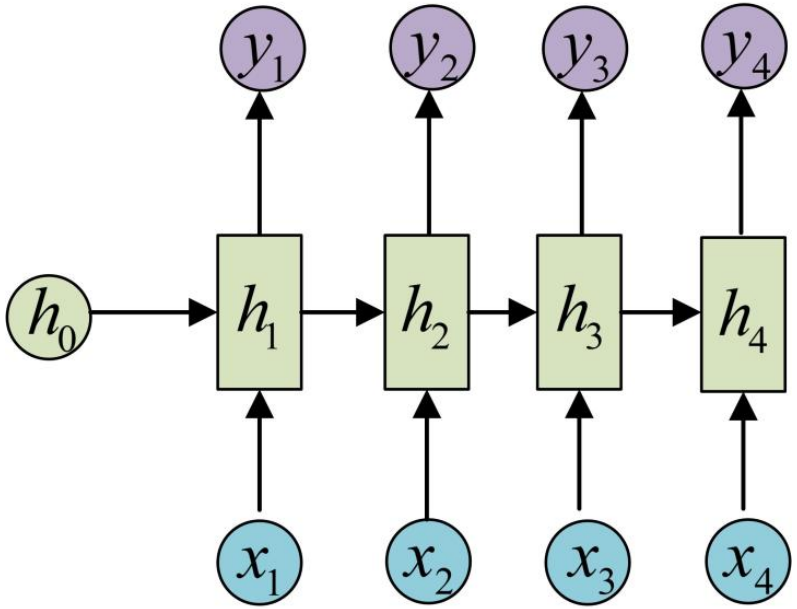


  - 参数W_{hh}, W_{xh}, b共享
    - 说明RNN每一步都在做相同的事，只不过是输入不同
      大大降低了网络中需要学习的参数，从而提高效率

循环神经网络(RNN)

- 如何计算 $p(x_i|x_{<i})$
  - $x_i = 1,2,\ldots,K$
  - 输出

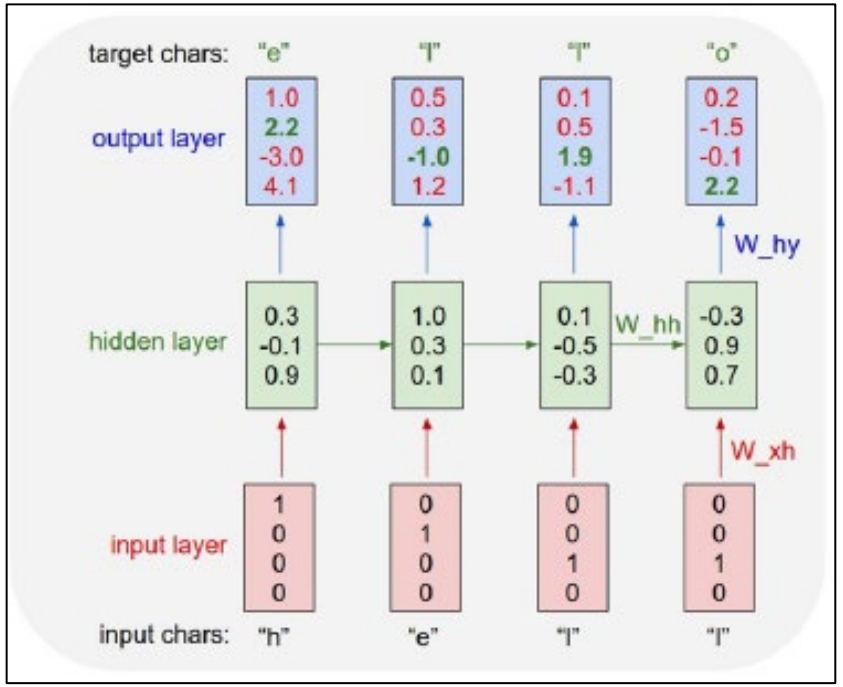
$$p(x_i|x_{<i}) = \text{softmax}(W_{hy}h_i + c)$$



循环神经网络(RNN)

- 实例
  - 假设$x_i \in \{\mathrm{h}, \mathrm{e}, \mathrm{l}, \mathrm{o}\}$
  - 求生成单词“hello”的概率

$$
\begin{aligned}
& p(x = hello) \\
= & \, p(x_1 = h) \\
& * p(x_2 = e | x_1 = h) \\
& * p(x_3 = l | x_1 = h, x_2 = e) \\
& * p(x_4 = l | x_1 = h, x_2 = e, x_3 = l) \\
& * p(x_5 = o | x_1 = h, x_2 = e, x_3 = l, x_4 = l)
\end{aligned}
$$



循环神经网络(RNN)

- 缺点：

$$h_i = \tanh(W_{hh}h_{i-1} + W_{xh}x_i + b)$$

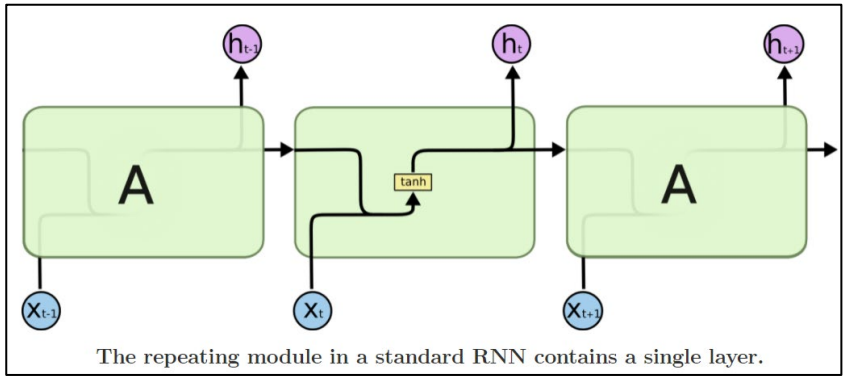
$$\Rightarrow \quad \frac{\partial h_i}{\partial h_{i-1}} = \tanh'(W_{hh}h_{i-1} + W_{xh}x_i + b)\, W_{hh} = (1 - \tanh^2(W_{hh}h_{i-1} + W_{xh}x_i + b))\, W_{hh}$$

$$\Rightarrow \quad \frac{\partial h_i}{\partial h_1} = W_{hh}^{i-1}\,\Lambda$$

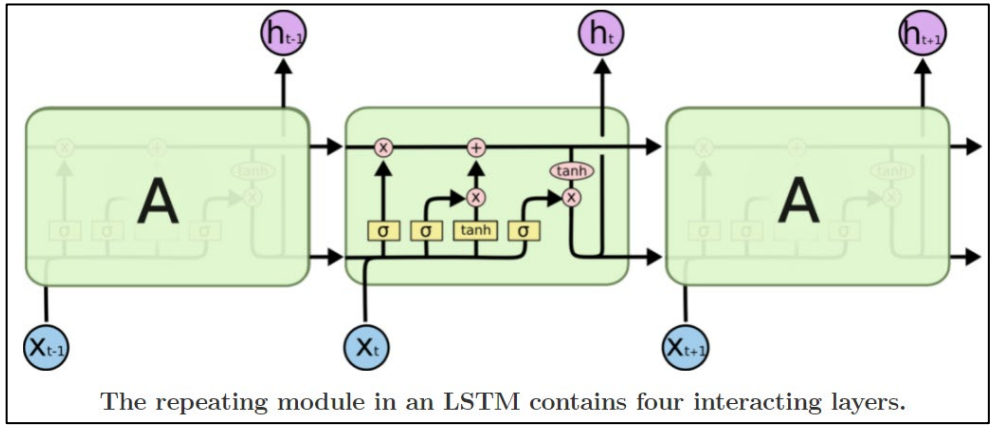
  - 梯度消失：如果$W_{hh} < 1, \frac{\partial h_i}{\partial h_1} \to 0$
  - 梯度爆炸：如果$W_{hh} > 1, \frac{\partial h_i}{\partial h_1} \to \infty$

长短时记忆网络（LSTM）

- 长短时记忆网络 LSTM
  - Hochreiter & Schmidhuber 于 1997 年发表



The repeating module in a standard RNN contains a single layer.

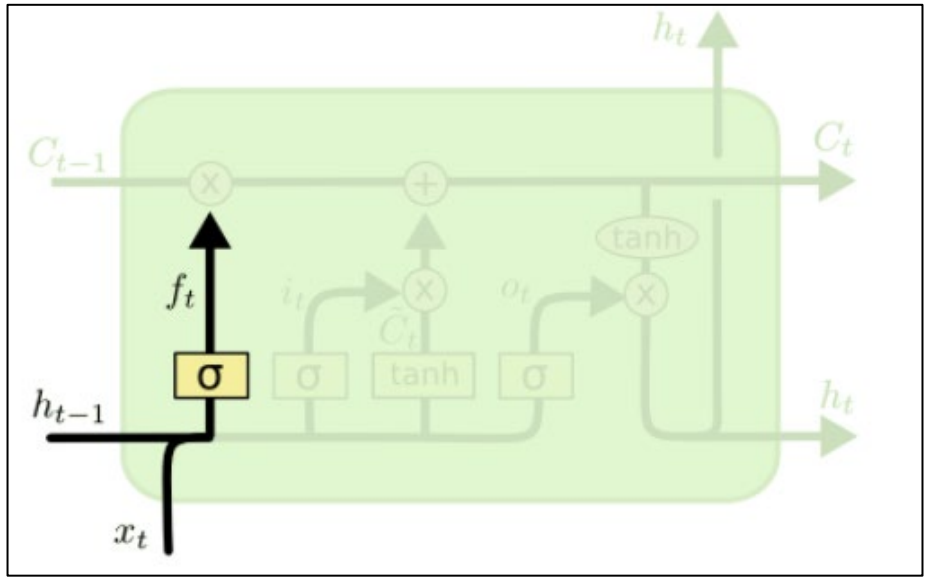


The repeating module in an LSTM contains four interacting layers.

长短时记忆网络（LSTM）

- 第一步：选择忘记过去某些信息
  - 遗忘门 f_t

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



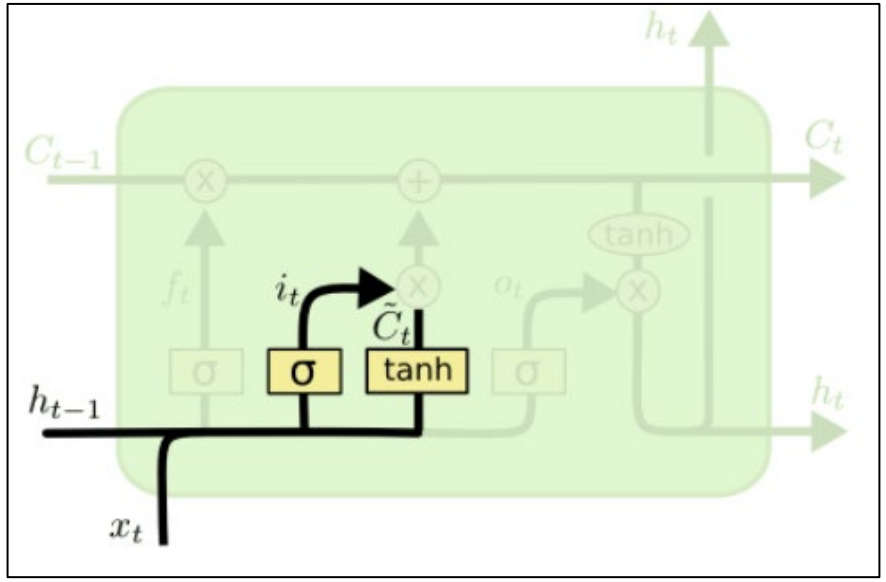
长短时记忆网络（LSTM）

- 第二步：记忆现在的某些信息
  - 输入门 i_t

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

  - 可能候选记忆细胞

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

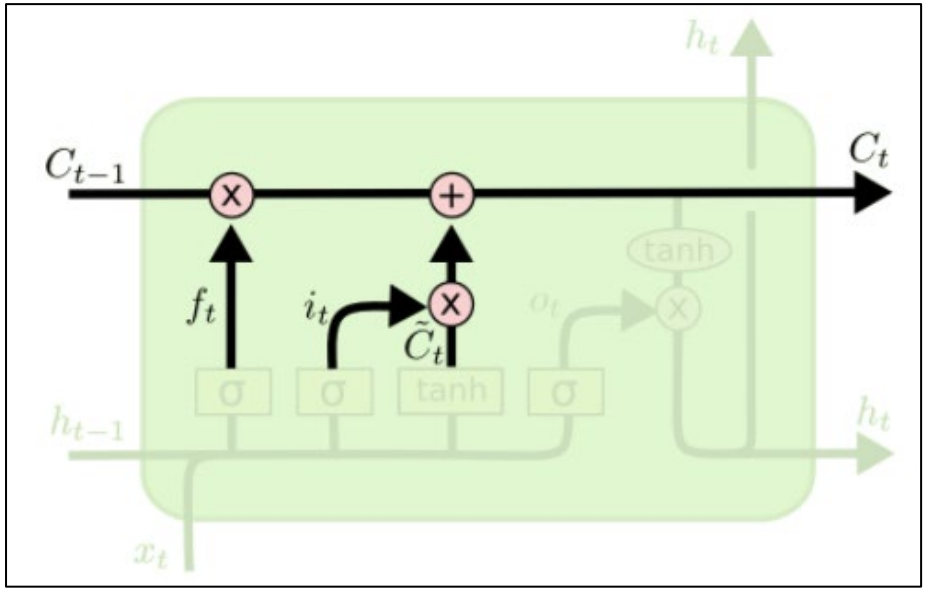


长短时记忆网络（LSTM）

- 第三步：将过去与现在的记忆合并

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

  - 如果遗忘门一直近似1且输入门一直近似0，过去的记忆细胞将一直通过时间保存并传递至当前时间步
  - 线性和：避免梯度消失和梯度爆炸

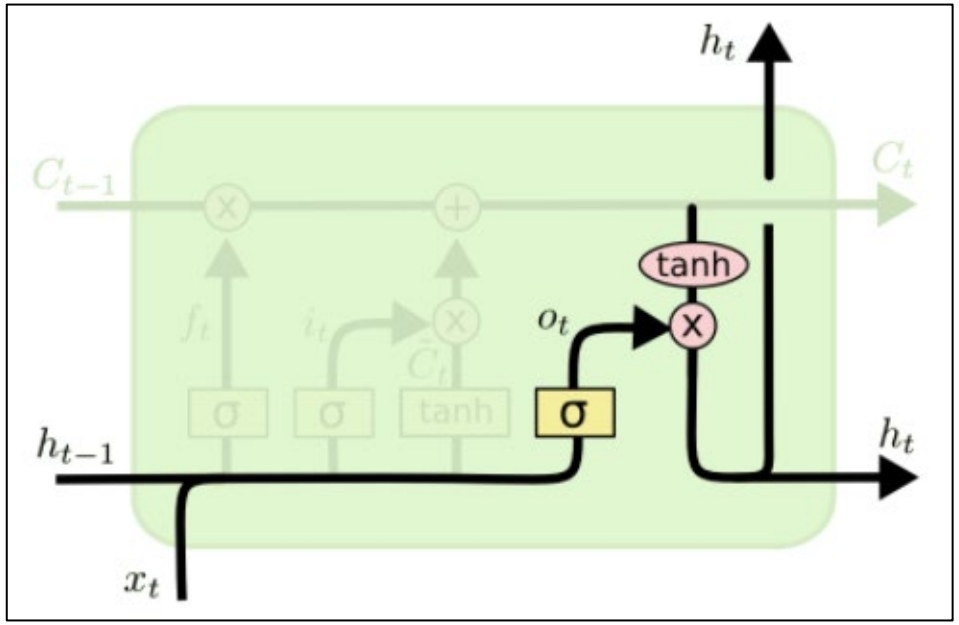


长短时记忆网络（LSTM）

- 第四步：输出
  - 当输出门近似1时，记忆细胞信息将传递到隐藏状态供输出层使用
  - 当输出门近似0时，记忆细胞信息只自己保留

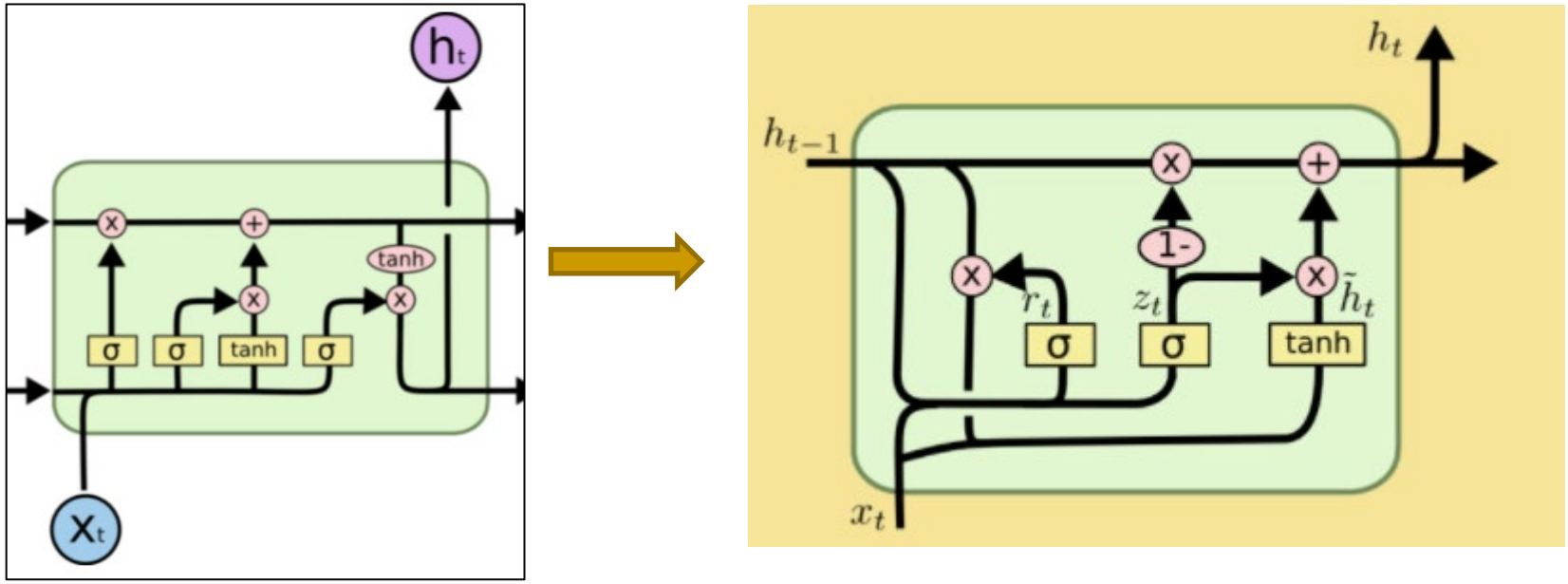
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



Gated Recurrent Unit (GRU)

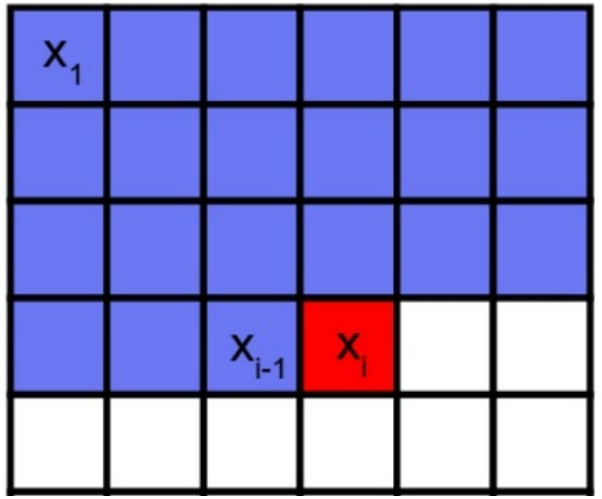
- 对LSTM优化
  - 把遗忘门和输入门合并成为“更新门”
  - 把细胞状态和隐含状态合并

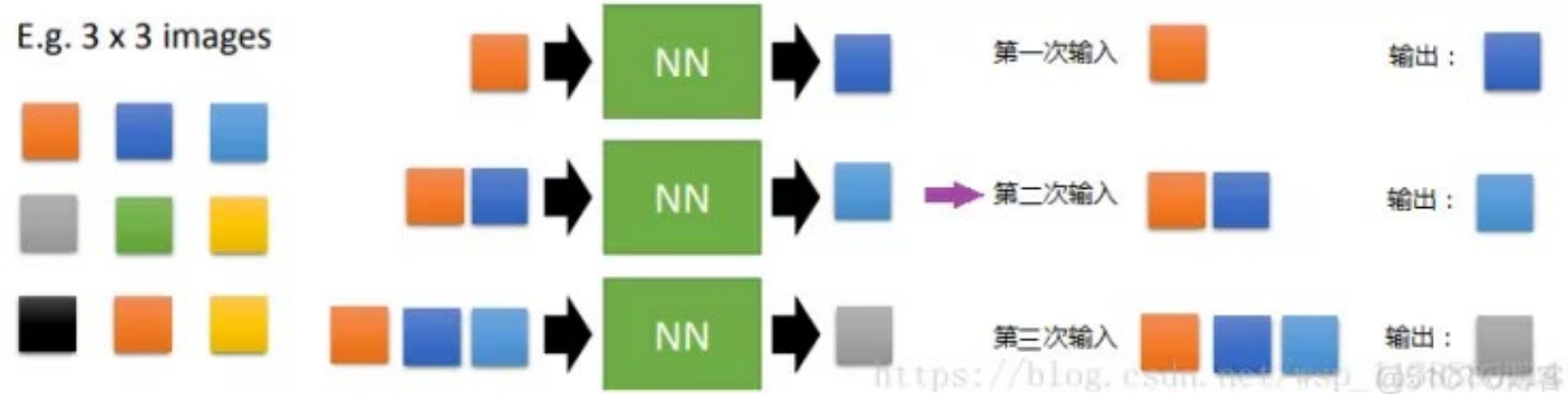


像素循环神经网络（Pixel RNN）

- 生成概率

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

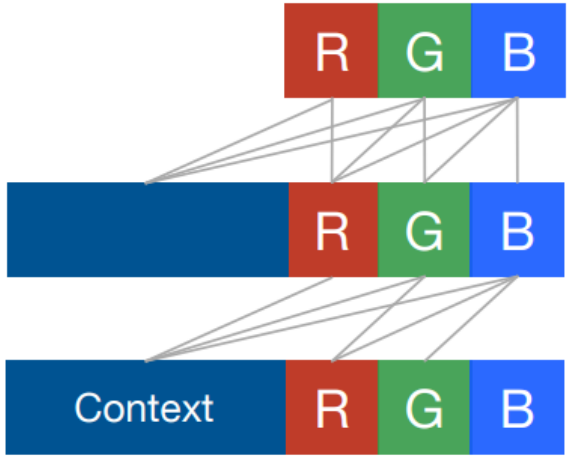




像素循环神经网络（Pixel RNN）

- 顺序提取像素特征
  - 每个像素点有三个特征通道

$$p(x_i|x_{<i}) = p(x_{i,R}|x_{<i})\ p(x_{i,G}|x_{<i}, x_{i,R})\ p(x_{i,B}|x_{<i}, x_{i,R}, x_{i,G})$$

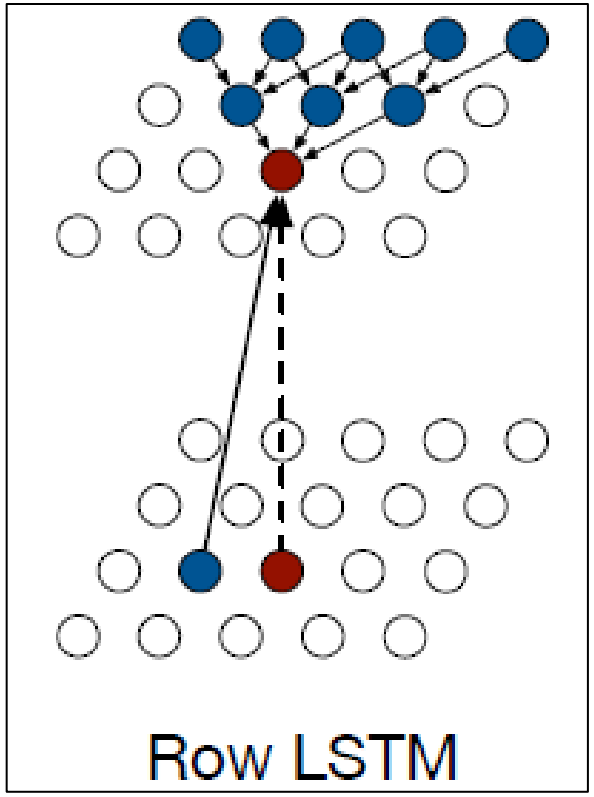


像素循环神经网络（Pixel RNN）

- 像素值的建模
  - 看成离散变量
  - 条件概率 $p(x_i|x_{<i})$ 是一个多项分布
- 局限
  - 只捕捉一维序列
  - 改进
    - Row LSTM
    - Diagonal BiLSTM
    - PixelCNN

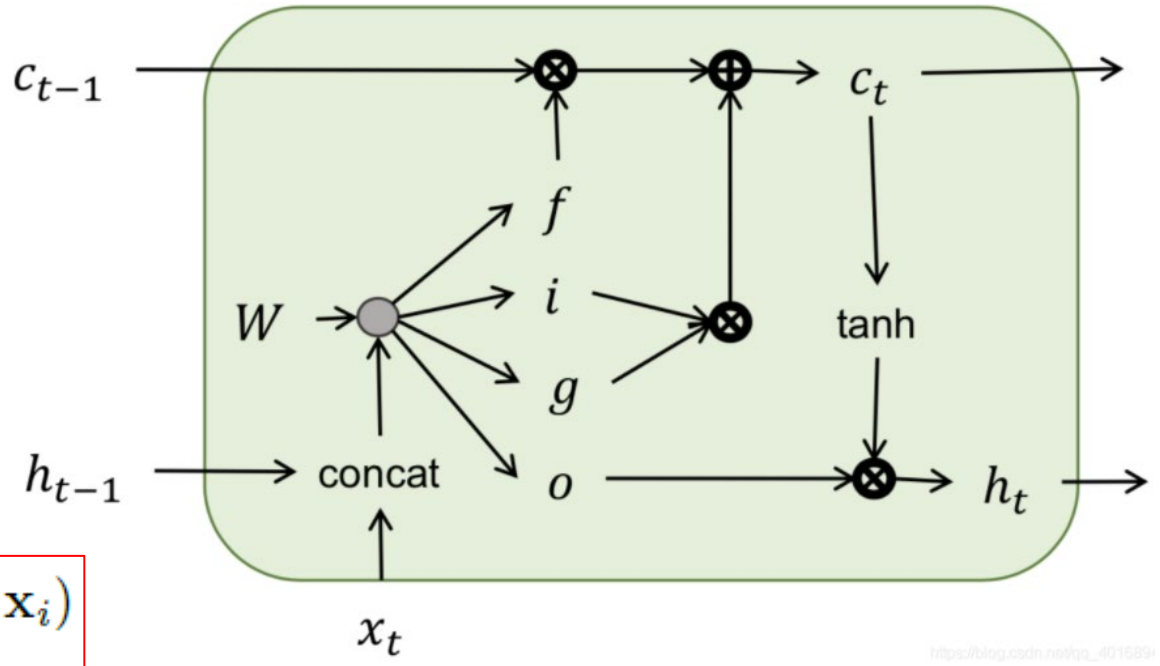
像素循环神经网络（Pixel RNN）

- Row LSTM
  - 按行处理、逐行扫描图像：LSTM
  - 跨行扫描： k× 1 (k≥3) 卷积



像素循环神经网络（Pixel RNN）

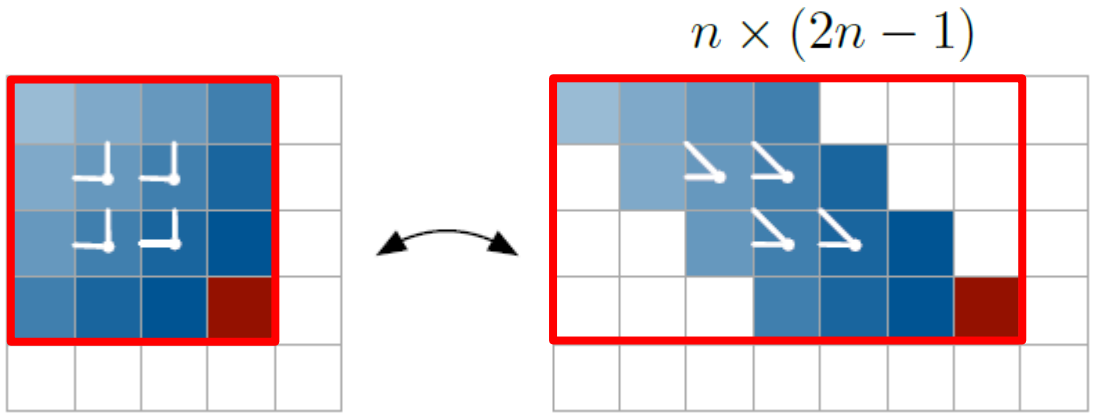
- 4个门
  - f：遗忘门
  - i：输入门
  - o：输出门
  - g：上下文门

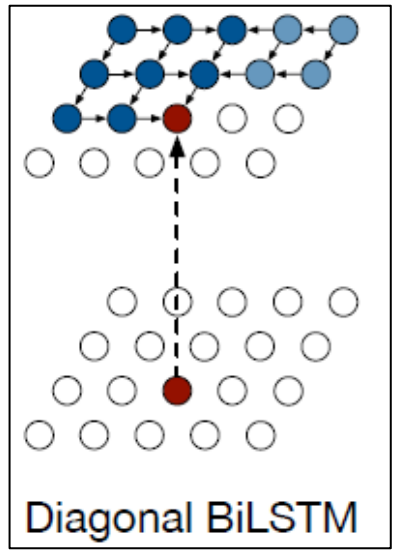


$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma(\mathbf{K}^{ss} \circledast \mathbf{h}_{i-1} + \mathbf{K}^{is} \circledast \mathbf{x}_i)$$
$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

像素循环神经网络（Pixel RNN）

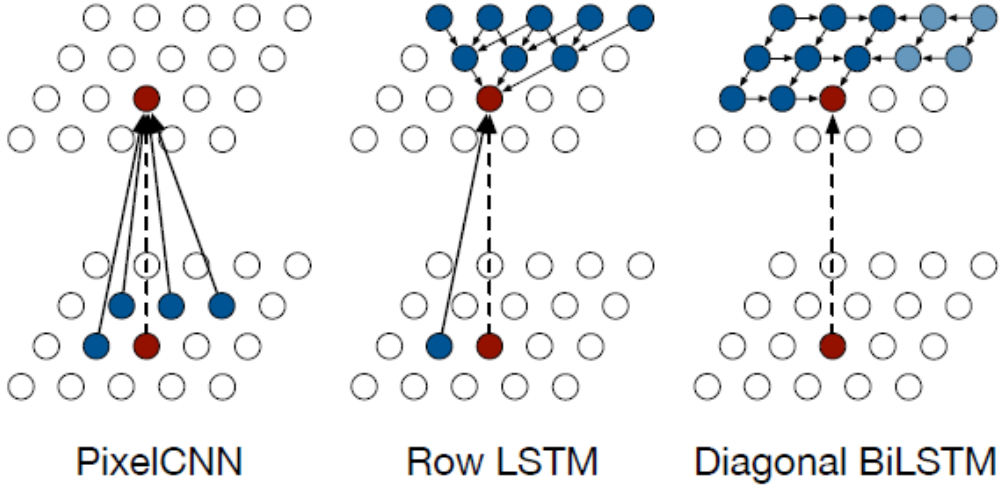
- Diagonal BiLSTM
  - 并行计算和捕获任何图像大小的整个可用上下文
  - 首先将输入图倾斜映射到另一个空间，使它很容易沿对角线应用卷积





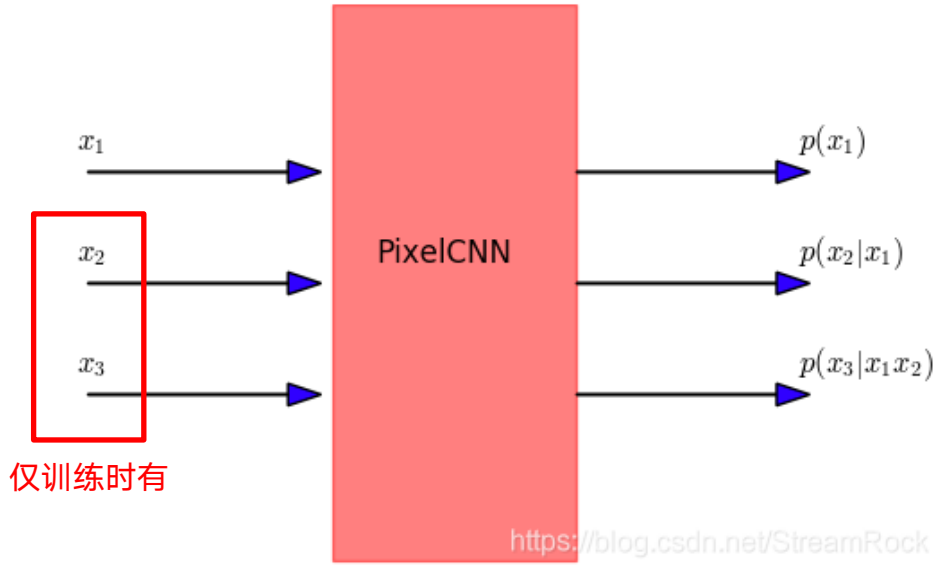
像素循环神经网络（Pixel RNN）

- PixelCNN
  - Row LSTM 和 Diagonal BiLSTM 具有理论上无限的感受野
  - PixelCNN 以卷积代替循环，获得局部感受野
    - 如何保证因果依赖？遮罩卷积



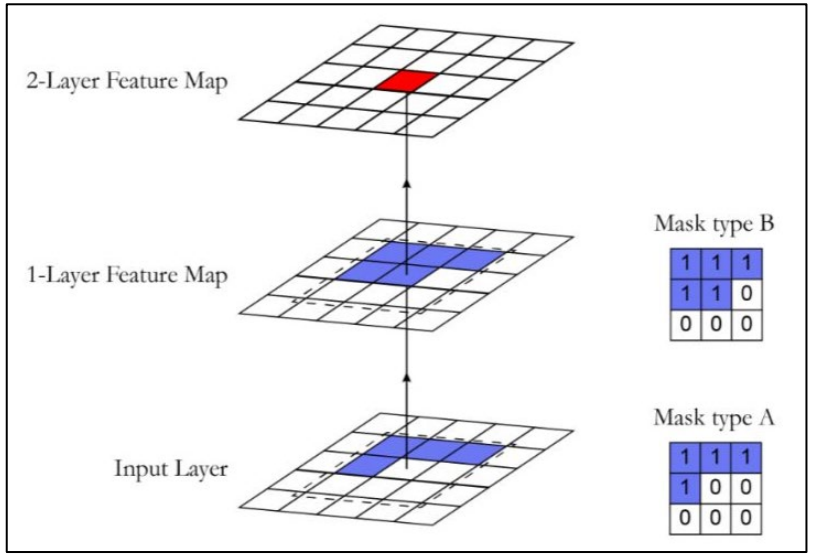
像素循环神经网络（Pixel RNN）

- PixelCNN
  - 利用了卷积的并行性，提升训练效率
  - 但，生成时仍需逐像素采样



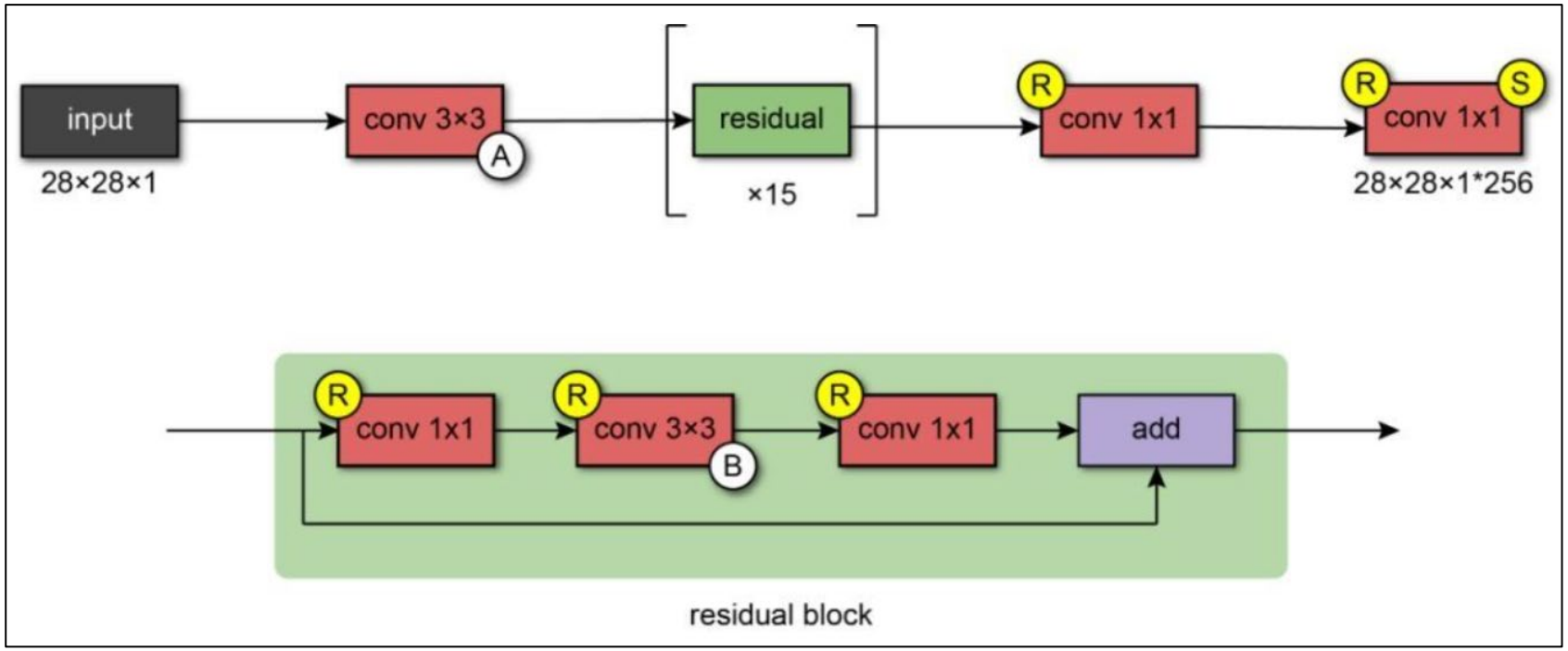
像素递归神经网络（Pixel RNN）

- PixelCNN
  - Mask机制：使模型只能访问符合顺序约束的像素
  - Mask A
    - 应用于第一卷积层；屏蔽当前像素和下文
  - Mask B
    - 应用于所有后续卷积层；屏蔽下文



像素循环神经网络（Pixel RNN）

- PixelCNN
  - 模型结构

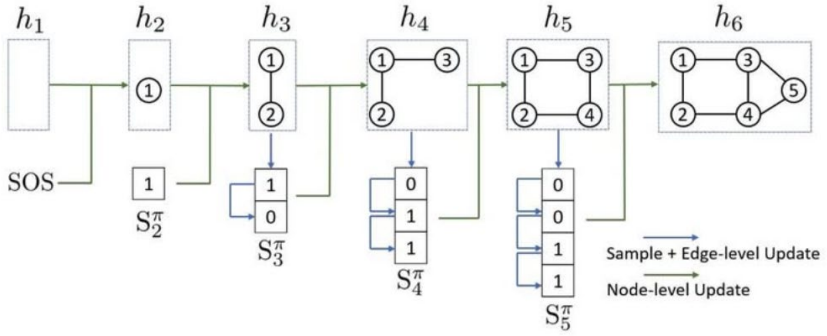


像素循环神经网络（Pixel RNN）

- PixelCNN
  - 生成过程
    - 创建一个与输入图像相同形状的空张量，用 1 填充，获得 $p(x_1)$
    - 从 $p(x_1)$ 采样，将采样值分配给像素 x_1
    - 再次将输入提供给网络，对下一个像素进行采样，并将采样值分配给该像素
    - 重复上一步，直到生成整张图片

其他一些经典模型

- GraphRNN
  - 从像素到图



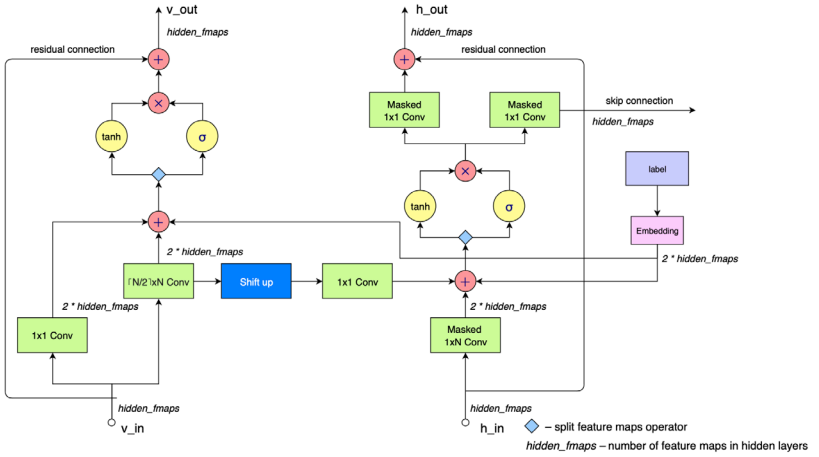
- Gated PixelCNN
  - 增强非线性表达能力

PixelCNN

$$h = \mathrm{ReLU}(W * x)$$

Gated PixelCNN

$$h = \tanh(W_f * x) \odot \sigma(W_g * x)$$

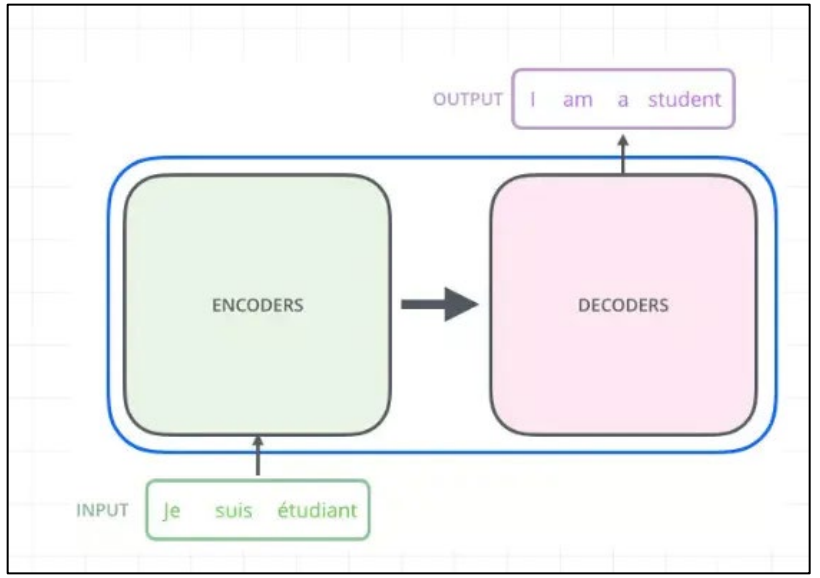


目录

- 线性自回归网络
- 神经自回归网络
- 递归神经网络
- Transformer

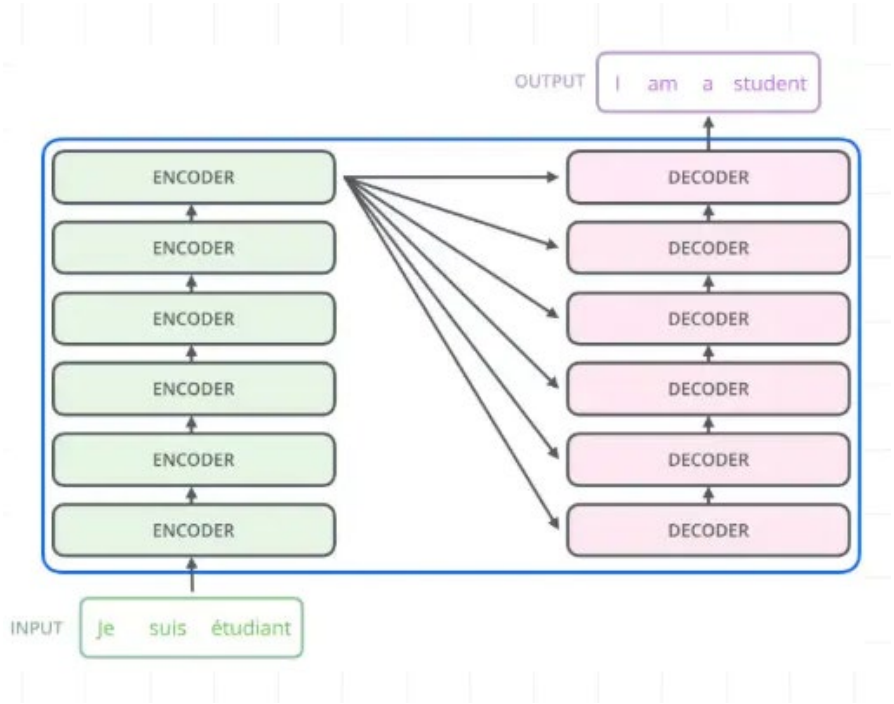
Transformer

- RNN
  - 效率低、难以并行
- Transformer
  - 采用注意力机制 (Attention)
    易于并行计算
    - 自注意力、交叉注意力
  - Encoder-Decoder 结构



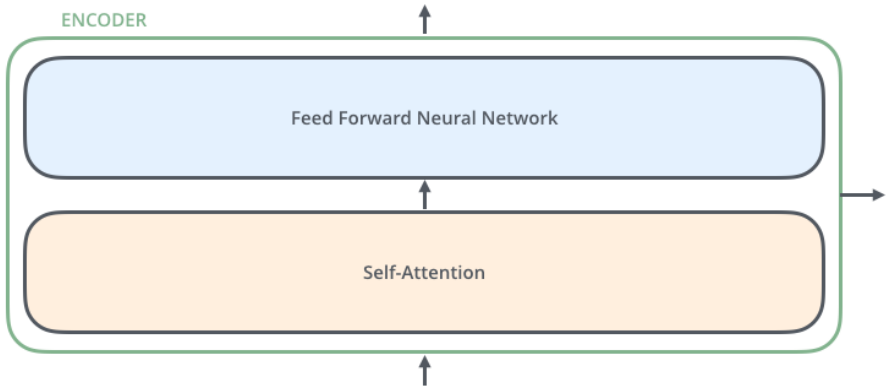
Transformer

- Encoder-Decoder
  - 编码器：多层堆叠的编码器模块
  - 解码器：数量相同的解码器模块
  - 交互：
    - 每一层解码器通过交叉注意力机制利用编码器最后一层的输出



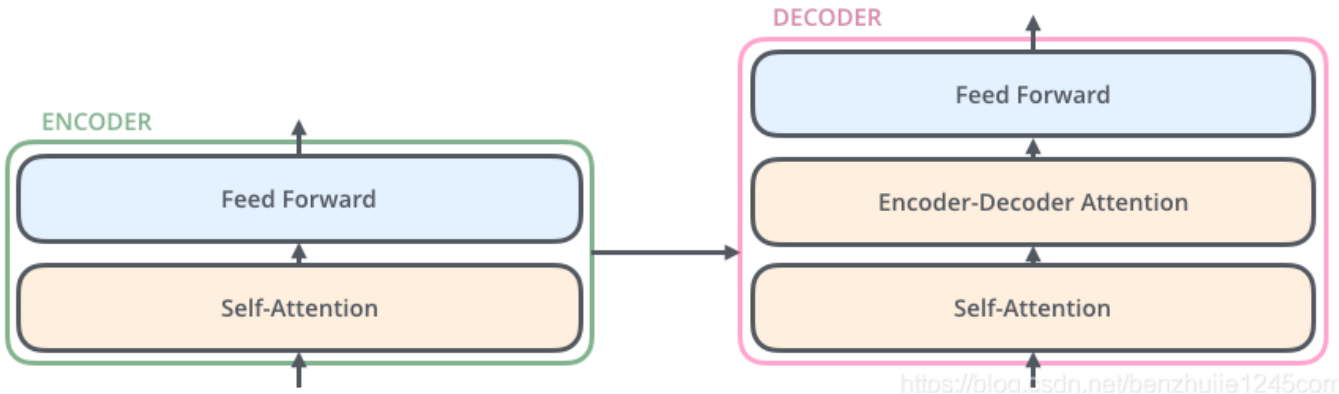
Transformer

- 编码器模块
  - 自注意力机制 (Self-Attention)
    - 信息聚合：在处理输入序列中某个位置时聚合序列中其它位置的特征信息
  - 逐点前馈网络 (Feed Forward)
    - 特征变换
  - 每个编码器层结构相同
    但权重独立



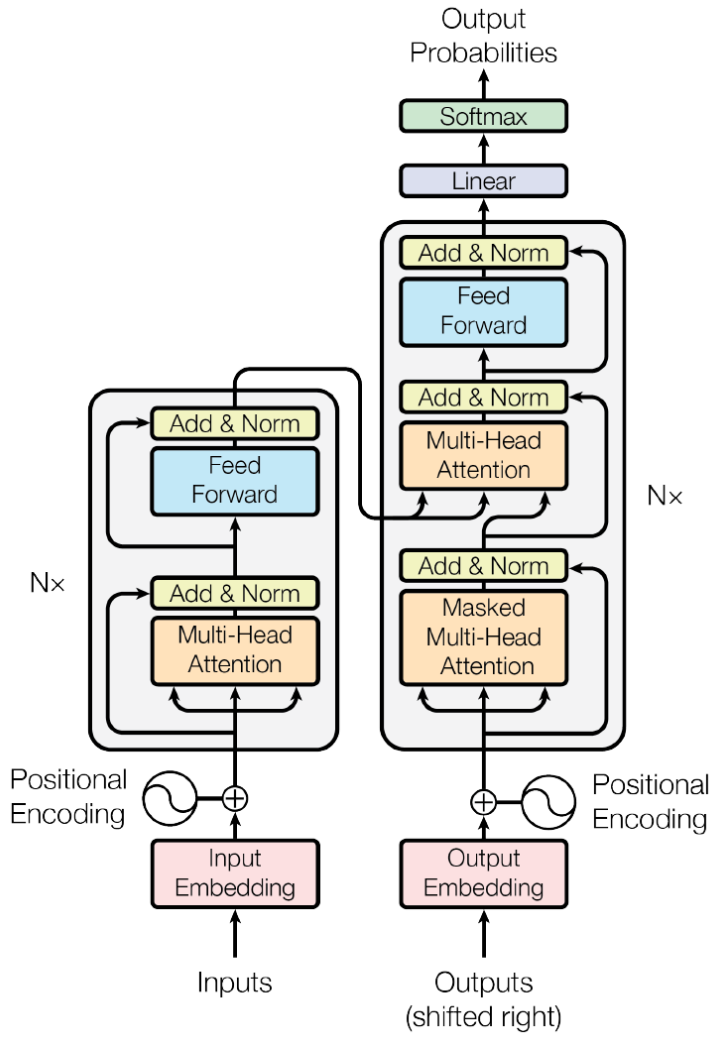
Transformer

- 解码器模块
  - 带掩码的自注意力机制
  - 交叉注意力机制
    - 在生成每个位置的表示时，以编码器输出为键和值
      以解码器当前隐状态为查询，计算注意力得分，融合输入序列的信息
  - 逐点前馈网络



Transformer

- 整体结构
  - 编码器、解码器
  - 词向量嵌入 (Input Embedding)
  - 位置编码 (Positional Encoding)



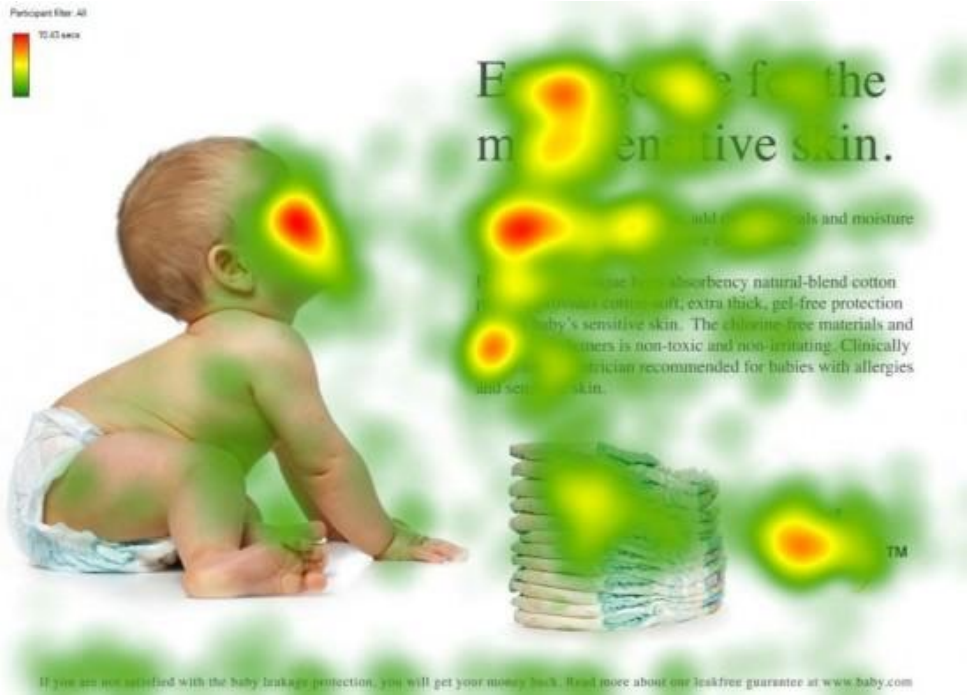
NIPS papers
https://papers.neurips.cc › paper › 7181-attention... PDF

Attention is All you Need

by A Vaswani Cited by 202949 — We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely.

Transformer

- 注意力机制
  - 人类：认知注意力
  - 模型
    - 在处理序列时，不是所有位置的信息都同样重要
    - 在处理某个位置时，有选择地关注序列中与它最相关的位置

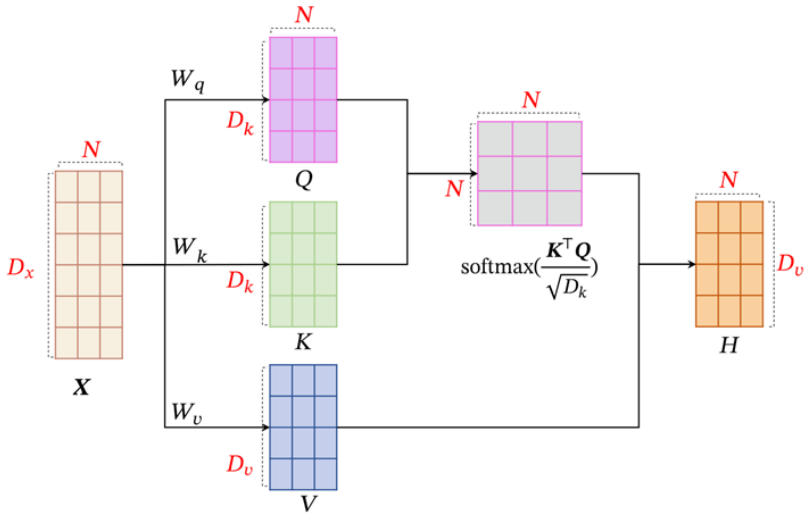


Transformer

- 缩放点积注意力 (Scaled Dot-Product Attention)
  - 注意力机制的基本计算单元
  - 输入 X 经权重矩阵得到词向量 Q, K, V

$$Q_i = W_Q x_i, \quad K_i = W_K x_i, \quad V_i = W_V x_i$$

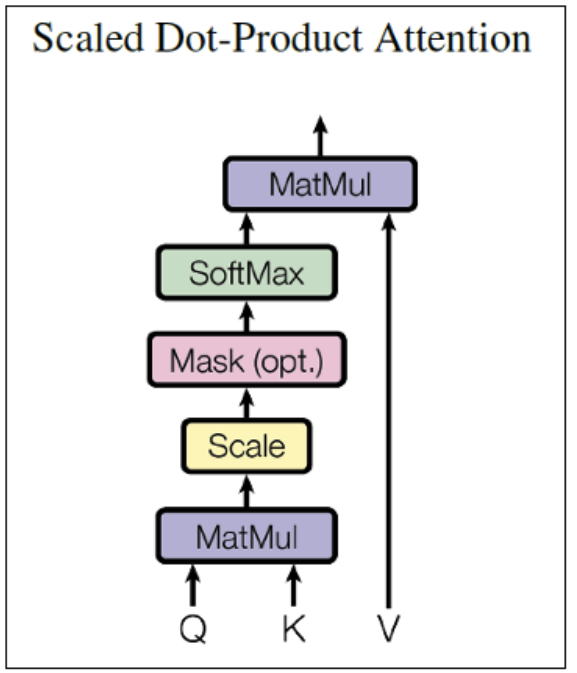
  - Q：希望更新的查询向量
  - K：键向量，候选信息的特征，提供相关性权重
  - V：值向量，候选信息的内容，提供信息



Transformer

- 缩放点积注意力 (Scaled Dot-Product Attention)
  - 计算注意力得分 QK^T：查询向量 Q 与 键向量 K 的相关性
  - 转换为概率分布 $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$
  - 用概率分布加权值向量 V，输出： $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

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



Transformer

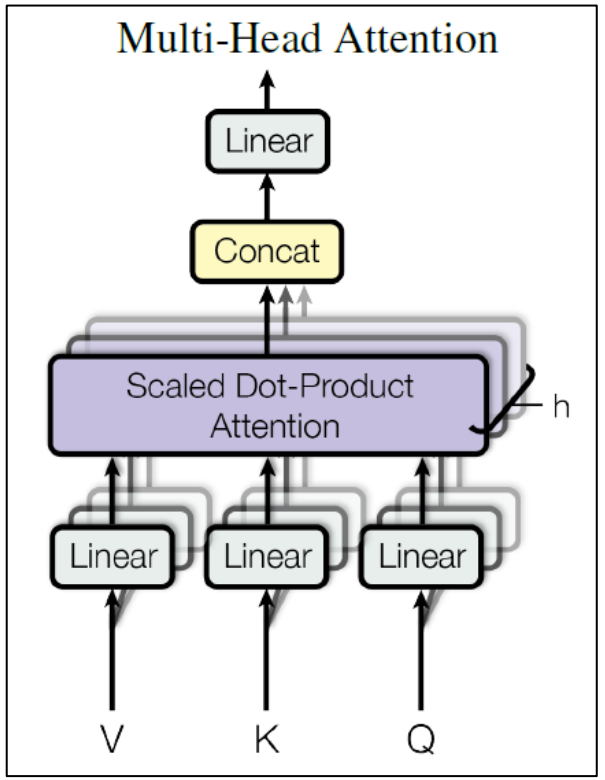
- 缩放点积注意力：以自注意力为例
  - 输入：我 喜欢 吃 苹果
    - 查询向量 Query: [喜欢]
    - 键向量 Key: [我, 喜欢, 吃, 苹果]
    - 值向量 Value: [我, 喜欢, 吃, 苹果]
  - 计算：
    - 注意力得分 QK^T：喜欢 * 我 = 0.6；喜欢 * 喜欢 = 1.0；喜欢 * 吃 = 0.8；喜欢 * 苹果 = 0.3
    - 归一化为概率分布 $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$：0.22, 0.37, 0.30, 0.11
    - 加权聚合 $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$：喜欢’ = 0.22 × 我 + 0.37 × 喜欢 + 0.30 × 吃 + 0.11 × 苹果
  - 效果：
    - 新的“喜欢” 融合了整个上下文的信息

Transformer

- 多头注意力（Multi-Head Attention）
  - 单个注意力机制
    - 在一次计算中，只能学到一种相关性
    - 词与词间可能有多种相关性关系（如语法、语义、情感）
  - 多头注意力
    - 每个“头”独立关注一种相关性
    - 对 Q, K, V 做不同线性映射，得到多个注意力头
    - 各自计算缩放点积注意力，再拼接

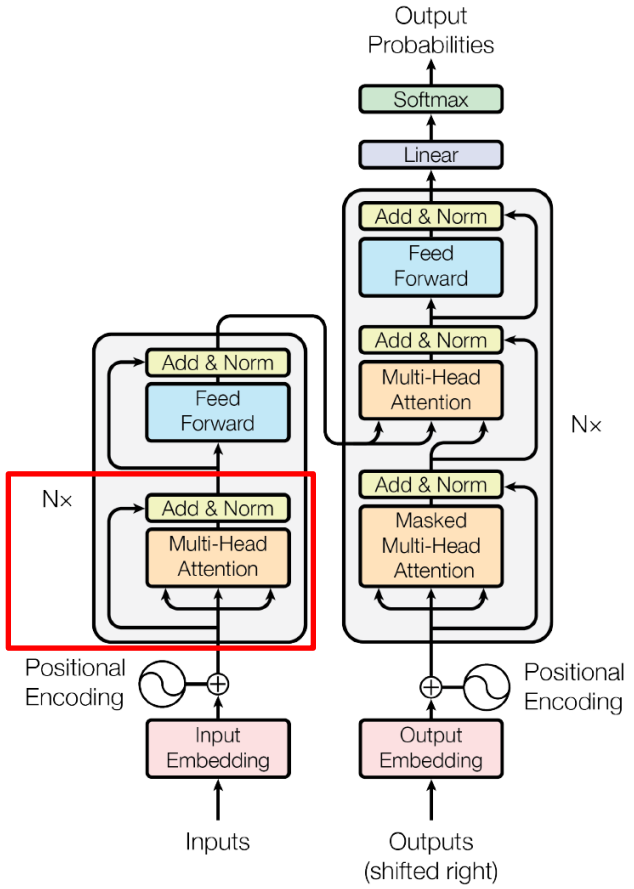
$$Q \in R^{m \times d_k} \ , \ K \in R^{m \times d_k} \ , \ V \in R^{m \times d_v}$$

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_\mathrm{h}) W^O$$



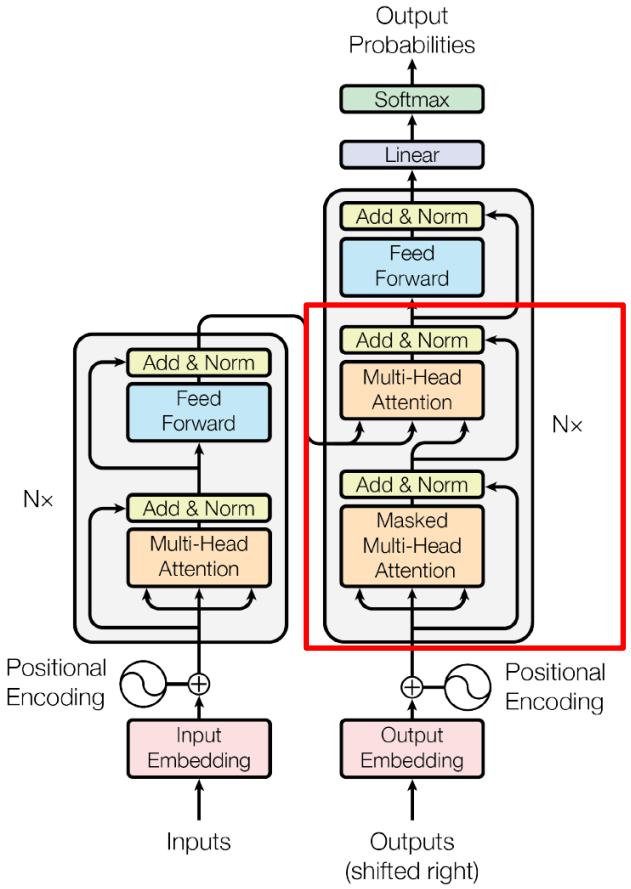
Transformer

- Encoder
  - 理解输入序列的语义
    - 每个位置都关注整个序列，形成“上下文感知表示”
  - 实现：多头注意力机制
    - Encoder只处理序列本身，$Q = K = V$ 来自上一层输出



Transformer

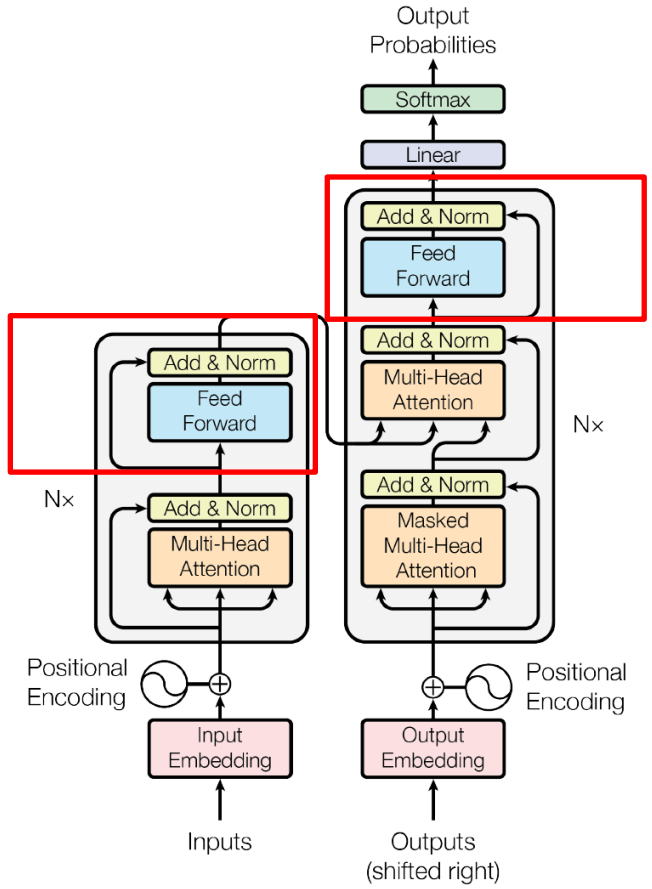
- Decoder
  - 在理解输入的基础上生成输出序列
  - 掩码多头注意力机制
    - 掩码：遵循顺序性——当前位置只依赖已生成的位置
    - 多头注意力：
      - Q 来自上一 decoder 层输出
        K, V 来自于 encoder 最终输出
      - Decoder 的每个位置都能获取
        输入序列的所有位置信息



Transformer

- 逐点前馈网络
  - 加工聚合的信息：
    - 对每个位置进行单独进行非线性变换
  - 结构：
    - 映射到高维空间：线性变换
    - 提供非线性语义：ReLU 激活输出
    - 投影至原维度：第二个线性变换

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



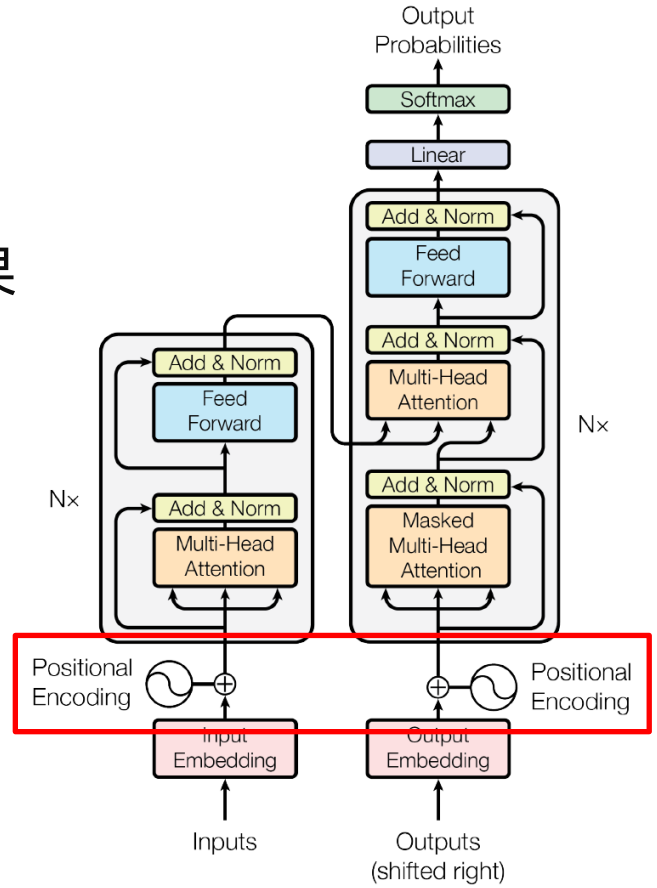
Transformer

- 位置编码
  - Transformer 默认无法捕捉序列顺序信息
    - 例如，将 K, V 按行打乱，不会改变 Attention 结果

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

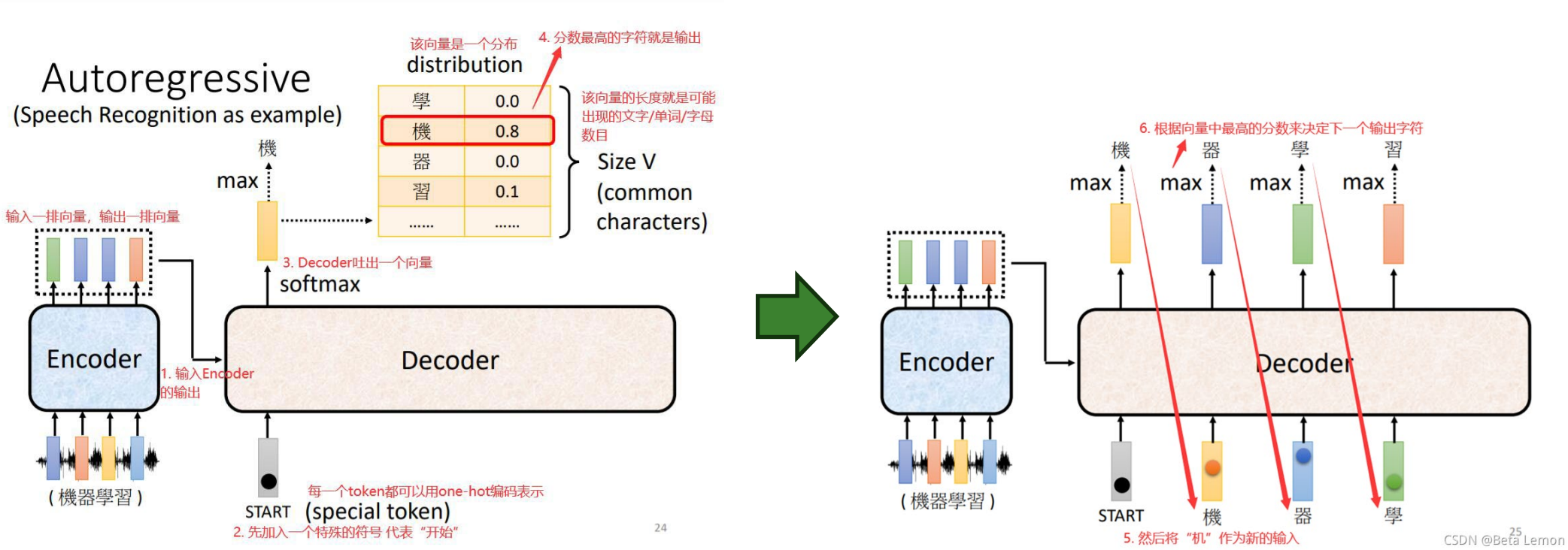
  - 改进：编码显式位置信息

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



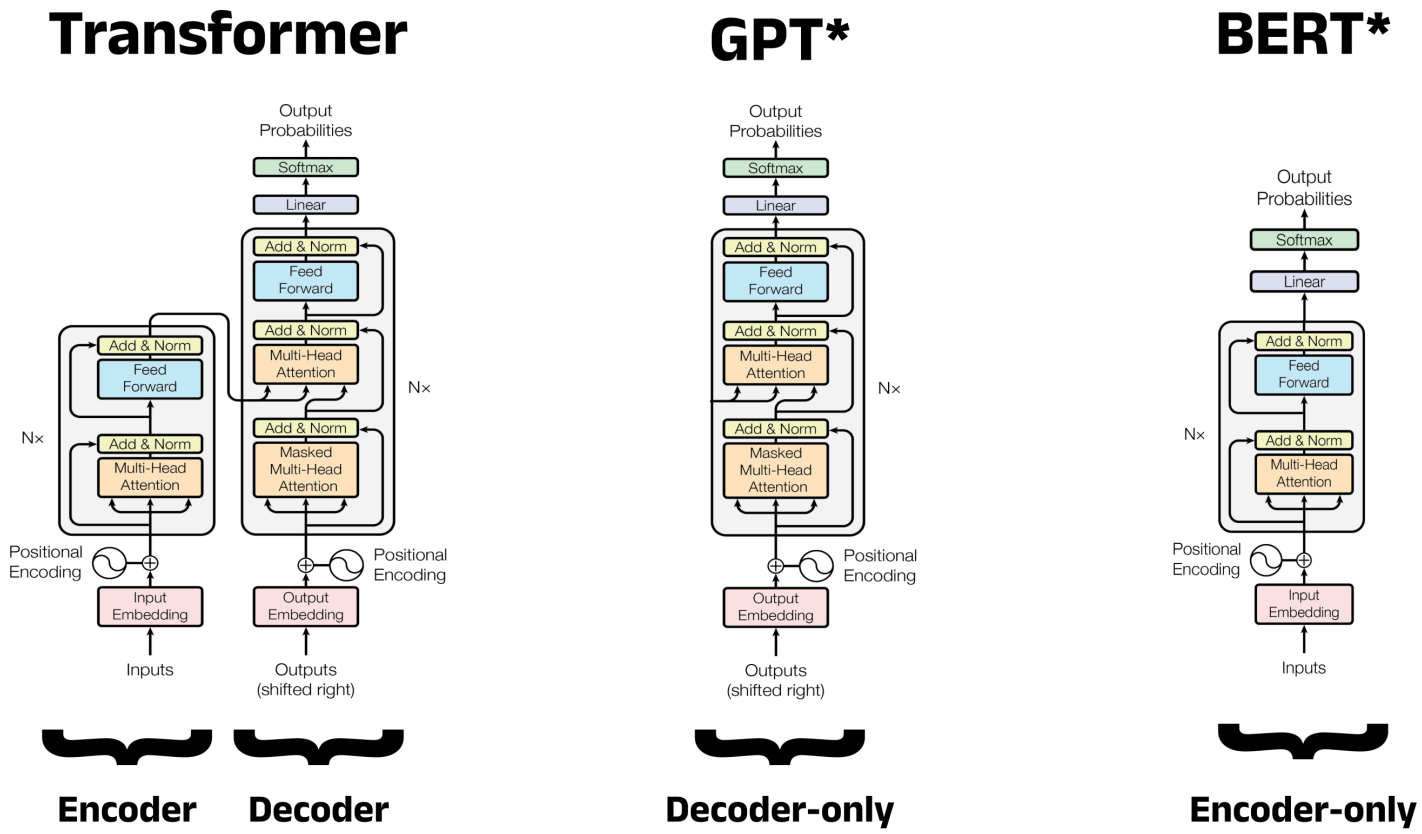
Transformer

- 基于Transformer的自回归：以语音识别为例



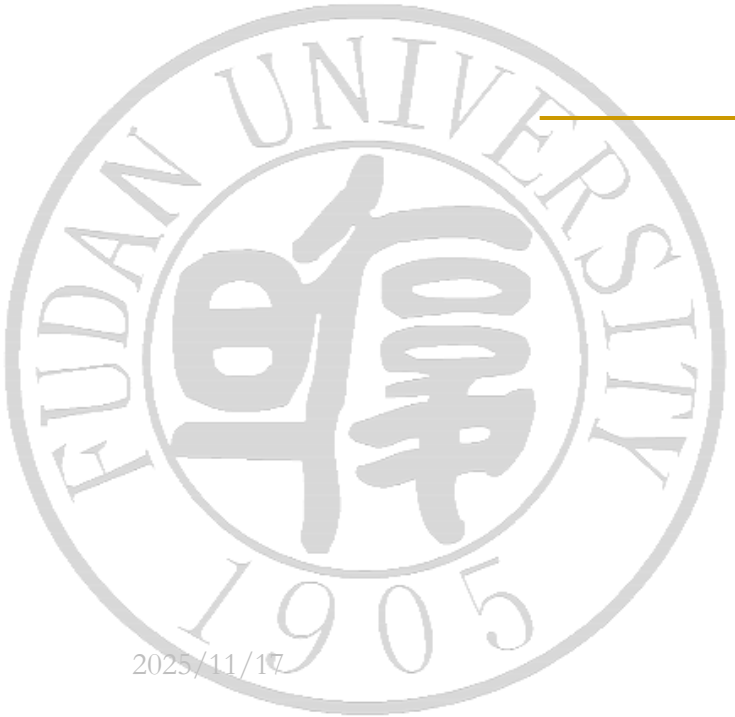
Transformer

- 代表性变体
  - GPT
  - BERT



*Illustrative example, exact model architecture may vary slightly

Thanks!



Questions?