A6：分析与简答

姓名：陈俊卉　班级：2020219111　学号：2020212256

A6.1 分析卷积神经网络中用1×1的卷积核的作用

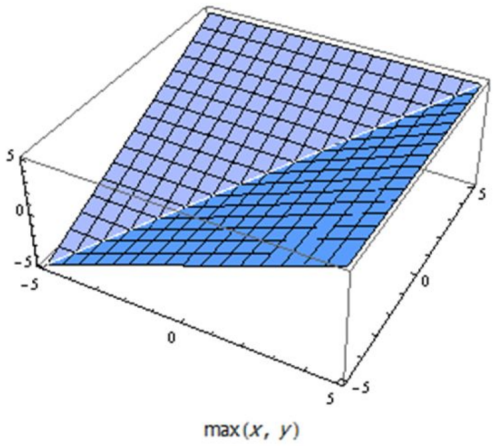
Answer:

- ① 降维或升维，调节通道数
  - 因为1×1的卷积核并不会改变 height 和 width，只改变channels，所以可以将原本的数据量进行增加或减少，称之为升维或降维，实际上是对通道数的调节。
- ② 增加非线性特性
  - 1×1的卷积核可以在保持特征图尺度不变的前提下大幅增加非线性特性（使用后接的非线性激活函数完成这一点）。非线性允许网络学习更加复杂的功能，且使得网络进一步加深。
- ③ 跨通道的信息交互
  - 使用1×1的卷积核实现升维和降维实际上是channel之间的线性组合变化。假设3×3×64的滤波器与1×1×32的滤波器组合，输出层得到3×3×32的滤波器，原来的64个通道跨通道线性组合变成了32个通道。
- ④ 减少参数
  - 降维使得channel变小，特征图变少，参数则变少。相当于在特征图的通道数上进行卷积，压缩特征图，二次提取特征，使得新特征图的特征表达更佳。
- ⑤ 1×1的卷积实际上可以替代全连接层（Yann LeCun）

A6.2 计算函数y = max(x1,⋯,xD)和函数y = argmax(x1,⋯,xD)的梯度。

Answer:

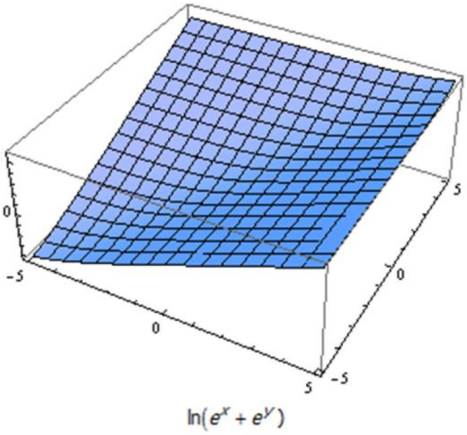
① y = max(x1,⋯,xD)



先考虑下列函数：

$$\mathrm{LSE}(x_1,\ldots,x_n) = \log(\exp(x_1)+\cdots+\exp(x_n)).$$

事实上，LogSumExp (LSE)是对y = max(x1,⋯,xD)的一个平滑逼近。



可以令$m = \max_i x_i$，则显然有

$$\exp(m) \le \sum_{i=1}^{n} \exp(x_i) \le n\exp(m)$$

对上述式子取对数则可以得到

$$\max\{x_1,\ldots,x_n\} \le \mathrm{LSE}(x_1,\ldots,x_n) \le \max\{x_1,\ldots,x_n\} + \log(n).$$

将上述式子的$x_1,\ldots,x_n$替换为$tx_1,\ldots,tx_n$，（$t>0$）可以得到

$$\max\{tx_1,\ldots,tx_n\} < \mathrm{LSE}(tx_1,\ldots,tx_n) \le \max\{tx_1,\ldots,tx_n\} + \log(n). \quad (1)$$

因为$t>0$，所以有

$$t\max\{x_1,\ldots,x_n\} < \mathrm{LSE}(tx_1,\ldots,tx_n) \le t\max\{x_1,\ldots,x_n\} + \log(n).$$

则有

$$\max\{x_1,\ldots,x_n\} < \frac{1}{t}\mathrm{LSE}(tx) \le \max\{x_1,\ldots,x_n\} + \frac{\log(n)}{t}. \quad (2)$$

实际上通过增大t，就可以让LSE逼近max.

使用近似(1)，$LSE(x)[x=(x_1,\dots,x_n)]$关于$x_i$的偏导数是：

$$\frac{\partial}{\partial x_i}\mathrm{LSE}(\mathbf{x})=\frac{\exp x_i}{\sum_j \exp x_j}$$

此时$LSE(x)$的梯度就是$softmax(x_i)$，也即y = max(x1,⋯,xD)的近似梯度；

使用近似(2)，max的梯度近似为：

$$\frac{\partial}{\partial x_i}max(x)\approx\frac{\partial}{\partial x_i}\frac{1}{t}LSE(tx)=\frac{e^{tx_i}}{(e^{tx_1}+\ldots+e^{tx_n})}=softmax(tx_i)\quad t\to+\infty$$

实际上t常常取一个较大的常数。

② y = argmax(x1,⋯,xD)

首先证明：$softmax(x)$是 $one-hot(argmax\ x)$的光滑化：

$$\begin{aligned}
[0,\cdots,1,\cdots,0]&=\text{one-hot}(\underset{i=1,\cdots,N}{\text{argmax}}\ \boldsymbol{x})\\
&=\text{one-hot}(\underset{i=1,\cdots,N}{\text{argmax}}\ \boldsymbol{x}-\max(\boldsymbol{x}))\\
&=\text{one-hot}(\underset{i=1,\cdots,N}{\text{argmax}}\ e^{\boldsymbol{x}-\max(\boldsymbol{x})})\\
&\approx\text{one-hot}(\underset{i=1,\cdots,N}{\text{argmax}}\ e^{\boldsymbol{x}-\log\sum_{k=1}^{N}e^{x_k}})\\
&\approx\left[\frac{e^{x_1}}{\sum_{k=1}^{N}e^{x_k}},\cdots,\cdots,\frac{e^{x_N}}{\sum_{k=1}^{N}e^{x_k}}\right]\\
&=\text{softmax}(\boldsymbol{x})
\end{aligned}$$

所圈的地方用到了①证明的max的光滑化。

于是有：

$$\begin{aligned}
\text{argmax}(\boldsymbol{x})&=\sum_{i=1}^{n}i\times\text{one-hot}(\text{argmax}(\boldsymbol{x}))_i\\
&\approx\sum_{i=1}^{n}i\times\text{softmax}(\boldsymbol{x})\\
&=\frac{\sum_{i=1}^{n}i\times e^{x_i}}{\sum_{i=1}^{n}e^{x_i}}
\end{aligned}$$

其梯度为

$$\frac{\partial argmax(\mathbf{x})}{\partial x_i}=i\cdot softmax(x_i)(1-softmax(x_i))$$

但观察到一个更加接近的近似：

$$\begin{aligned}\operatorname{argmax}(\boldsymbol{x}) &= \sum_{i=1}^{n} i \times \text{one-hot}(\operatorname{argmax}(\boldsymbol{x}))_i \\ &\approx \sum_{i=1}^{n} i \times \operatorname{softmax}(\boldsymbol{x}) \\ &= \frac{\sum_{i=1}^{n} i \times e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}\end{aligned}$$

不妨思考将其替换为

$\sum_{i=1}^{n} i \cdot softmax(t\mathbf{x}) \quad t \to +\infty$

显然，这是一个更接近的近似，尤其当$t \to +\infty$时，其**等价于**$one-hot(argmax\ x)$**！（这显然可以通过对**$softmax(tx_i)$**上下同时除以**$e^t x_{max}$**得到：若**x_i**不为最大值，则为0；否则为1）**

随即求导得

$$\frac{\partial}{\partial x_i} argmax(\mathbf{x}) = \lim_{t \to +\infty} t \cdot i \cdot softmax(tx_i) \cdot (1 - softmax(tx_i))$$

但实际上：

- t不能取无穷，这会导致函数失去梯度（**等价于**$one-hot(argmax\ x)$）；
- t若取很大的值，也会导致函数不太光滑；
- 所以实际上我们还是使用下式来近似

$$\frac{\partial argmax(\mathbf{x})}{\partial x_i} = i \cdot softmax(x_i)(1 - softmax(x_i))$$

$Q.E.D.$

A6.3 推导LSTM网络中参数的梯度，并分析其避免梯度消失的效果

对于普通的RNN：

$$\begin{aligned} h_t &= tanh(W_I x_t + W_R h_{t-1}) \\ y_t &= W_o h_t \end{aligned}$$

$$\frac{\partial E_k}{\partial W_R} = \sum_{i=0}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_i} \frac{\partial h_i}{\partial W_R}$$

其中

$$\frac{\partial h_t}{\partial h_i} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_{i+1}}{\partial h_i} = \prod_{k=i}^{t-1} \frac{\partial h_{k+1}}{\partial h_k}$$

而

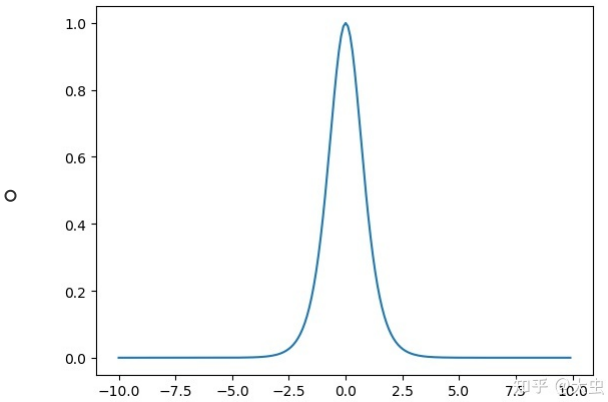
$$\frac{\partial h_{k+1}}{\partial h_k} = diag(tanh^{'}(W_I x_i + W_R h_{i-1}))W_R$$

则梯度为：

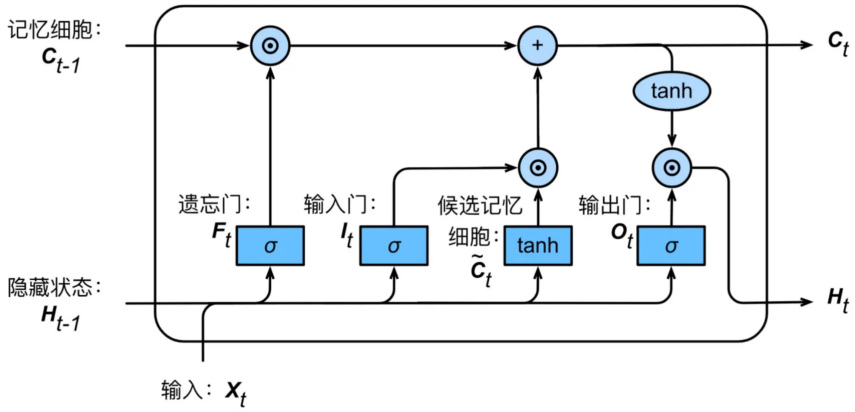
$$\frac{\partial h_t}{\partial h_l} = \prod_{i=l}^{t} diag(tanh^{'}(W_I x_i + W_R h_{i-1}))W_R$$

可以看到：

- 若矩阵W_R的主特征值大于1，则会导致梯度爆炸
- 若矩阵W_R的主特征值小于1，则会导致梯度消失
- $tanh$的导数总是小于等于1，当W_R的主特征值小于1，梯度可能变为0
- **这代表前面的隐藏层对后面的隐藏层造成的影响十分小！是远距离的梯度消失。**
- $tanh$的导数图像：



对于LSTM：



约定：

- E_t：t时刻的损失函数
- y_t：t时刻的输出
- f_t：遗忘门，控制上一个隐藏状态遗忘多少信息
- i_t：输入门，当前状态的候选状态保存的信息
- o_t：输出门，当前隐藏状态输出给外部状态的信息量
- h_t：t时刻的隐藏状态
- $\tilde{C}_t$：候选记忆细胞
- C_t：记忆细胞
- 忽略$bias$（其在求导中不会出现）
- W_R：用于循环的 $weights$
- 由于与循环无关的参数更新比较简单，在此不再枚举
- **下列 · 等价于⊙**

$$
\begin{aligned}
f_t &= \sigma(W_f[x_t, h_{t-1}]) \\
i_t &= \sigma(W_i[x_t, h_{t-1}]) \\
o_t &= \sigma(W_o[x_t, h_{t-1}]) \\
\tilde{C}_t &= tanh(W_C[x_t, h_{t-1}]) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \\
h_t &= o_t \cdot tanh(C_t)
\end{aligned}
$$

则有

$$\frac{\partial E_k}{\partial W_R} = \sum_{i=0}^{t} \frac{\partial E_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial C_{t-1}} \cdot \cdots \cdot \frac{\partial C_{i+1}}{\partial C_i} \cdot \frac{\partial C_i}{\partial W_R}$$

其中

$$\frac{\partial C_t}{\partial C_{t-1}} \cdot \cdots \cdot \frac{\partial C_{i+1}}{\partial C_i} = \prod_{k=i}^{t-1} \frac{\partial C_{k+1}}{\partial C_k}$$

因为有

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

所以对$\frac{\partial C_t}{\partial C_{t-1}}$展开得到

$$\frac{\partial C_t}{\partial C_{t-1}} = \frac{\partial f_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} \cdot C_{t-1} + f_t + \frac{\partial i_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} \cdot \tilde{C}_t + i_t \cdot \frac{\partial \tilde{C}_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}}$$

进一步展开为

$$\frac{\partial C_t}{\partial C_{t-1}} = C_{t-1}\sigma^{'}(\cdot)W_f * o_{t-1}tanh^{'}(C_{t-1}) + f_t + \tilde{C}_t\sigma^{'}(\cdot)W_i * o_{t-1}tanh^{'}(C_{t-1}) + i_t tanh^{'}(\cdot)W_C * o_{t-1}tanh^{'}(C_{t-1})$$

- 对于上述式子，对任意t，不会总是在$(1, +\infty)$或总是在$(0, 1)$.所以当$t \to \infty$时，也不会收敛到0或者∞.
- 而事实上，f_t、o_t、i_t、$\tilde{C}_t$的值是网络学习到的。因此**网络会学会决定梯度。**
- 它为细胞状态更新函数给出了一个更好的导数。
- 门控函数允许网络决定梯度消失的程度，并且可以在每个t取不同的值，**所取的值是从当前输入和隐藏状态学习到的。**

LSTM缓解远距离梯度消失的效果：

- LSTM的梯度传播途径多，在上述求解中一共有四项。而除了f_t这一项的梯度流比较稳定之外，其他三项的梯度流其实与普通的RNN类似，会发生相同的权重矩阵反复连乘。所以可以说在f_t这一条路径上的梯度是LSTM比RNN稳定的最重要因素。**LSTM通过改善一条路径的梯度问题拯救了总体的远距离梯度消失的问题。模型通过学习 f_t 决定什么时候梯度应该降低。**
- 但同样地，由于其他三项仍然为矩阵连乘，所以仍然可能发生梯度爆炸问题。但**由于LSTM相比于RNN多经过了很多次激活函数，所以LSTM梯度爆炸的频率也减少了。一般通过裁剪解决。**

A6.4 当将自注意力模型作为神经网络的一层使用时，分析它和卷积层以及循环层在建模长距离依赖关系的效率和计算复杂度方面的差异

1、自注意力模型：

自注意力模型是注意力机制的一个变体：通过$query$ 与 key 的相似性程度决定 $value$ 的权重分布。与注意力机制的区别是，其减少了对外部信息的依赖，且**其注意力层的权重由输入决定。**

自注意力模型采用QKV模式，计算过程如下图所示：

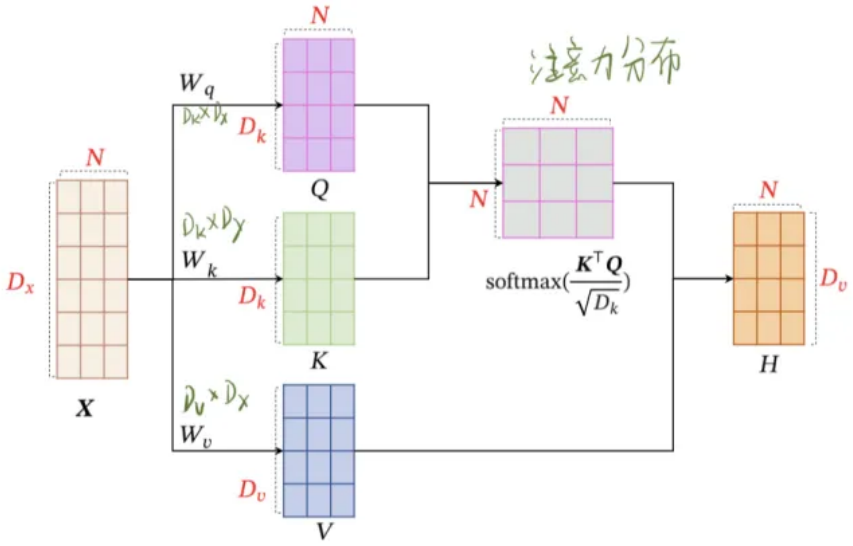


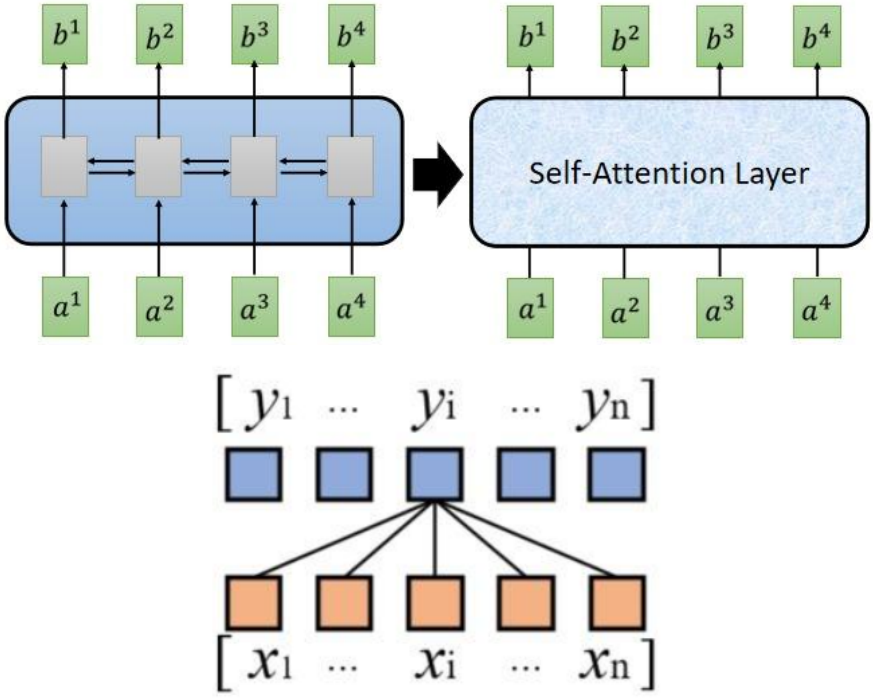
图 8.4　自注意力模型的计算过程

- 首先**输入向量**X分别与W_q,W_k,W_v相乘生成查询向量矩阵 Q 、键向量矩阵 K 和值向量矩阵 V .
- 然后对于每一个查询向量$q_n \in Q$,使用键值对注意力机制可以得到输出向量h_n.

通过以上过程，我们可以看到：**自注意力模型的每个输出都是基于全局信息**的（即是一种直接的对序列的长距离依赖关系），**并且可以并行化计算。**

对于一个输入序列$X = (x_1, x_2, \ldots, x_n)$，$x_i$为第$i$个词的词向量。自注意力机制：

$$y_i = f(x_1, x_2, \ldots, x_n)$$



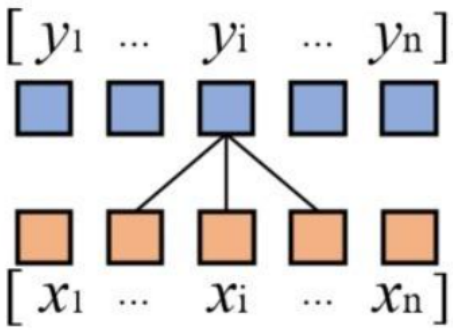
同时，与同样可以直接建立对序列的长距离依赖关系的全连接网络相比，**自注意力模型可以利用注意力机制动态地生成不同连接的权重，从而处理变长的信息序列。**

2、卷积层

众所周知，卷积层使用卷积核进行滑动窗口遍历，**其对于每一个卷积层，卷积核都只能感受局部的信息（即为一种 $N-gram$ 的局部编码）。如果需要感受更大范围的信息，增大感受野，则需要加深层数。从这来看，卷积层在建模长距离依赖的效率是不如自注意力模型的。**

对于一个输入序列$X=(x_1,x_2,\ldots,x_n)$，x_i为第i个词的词向量。卷积层每次只能进行部分处理（以卷积核长度为3为例）：

$$y_i=f(x_{i-1},x_i,x_{i+1})$$



而由于每一次提取相互之间是无关的，所以卷积本身也是可以并行计算的。但相对而言，卷积层计算要慢于自注意力模型：

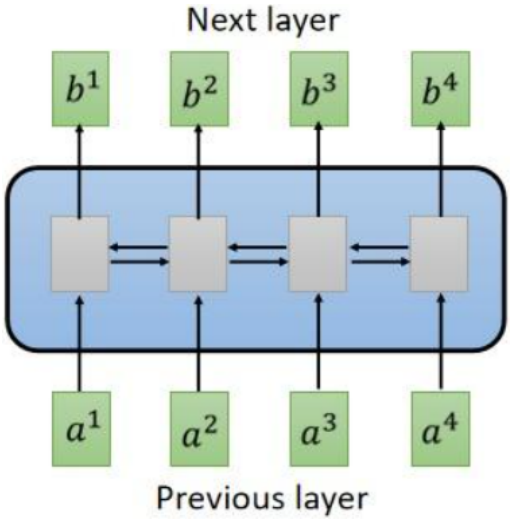
- 对于每一个词向量，进行一次自注意力需要五次向量乘法运算
- 对于每一个词向量，**进行一次完整的卷积提取需要若干层，且若要提取完整的信息，每一层的乘法运算数量最少也与一次自注意力的次数相等（随步长变化）**

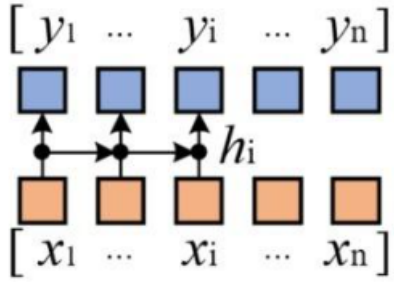
3、循环层

循环层的计算过程是通过递归实现的，在上文A6.3也有介绍到。而由于其递归的特性，导致其计算对于输入序列而言是串行处理的，无法使用并行计算加快计算效率，计算复杂度也随之剧增。

对于一个循环层，其在本层的输出由本层的输入与上层的权重决定：

$$y_i=f(h_{i-1},x_i)$$





同样地，**就像上文A6.3所述，RNN存在梯度爆炸和梯度消失的问题，事实上其对长距离依赖关系的建模效率也很低，可以通过LSTM等方式缓解，但还是不能根本地解决问题。**

4、总结

- **自注意力模型**是对词向量的全局信息直接进行提取与计算，**可以很好地建立长距离依赖的关系，效率很高，一层计算便可实现依赖的建立**，并且其可以根据输入动态生成不同的权重，还**能够进行高效率的高度并行化计算，计算复杂度也很低。**
- **卷积层**对词向量进行部分卷积感受提取，**对于一层卷积层来说，事实上只能获得局部信息，需要通过堆叠更多层数增大感受野，直到提取所有信息，建立长距离依赖关系的效率一般**；卷积神经网络的不同卷积核之间**能够进行并行计算**，但**计算次数要比自注意力模型多。**
- **循环层**通过递归获得全局信息，但**事实上会出现梯度爆炸和梯度消失等问题，难以建立真正长距离依赖的关系；**且由于是递归，同时无法进行并行计算，**计算复杂度相对最高，计算效率也很低。**