循环神经网络

RNN

RNN: RECURRENT NEURAL NETWORK

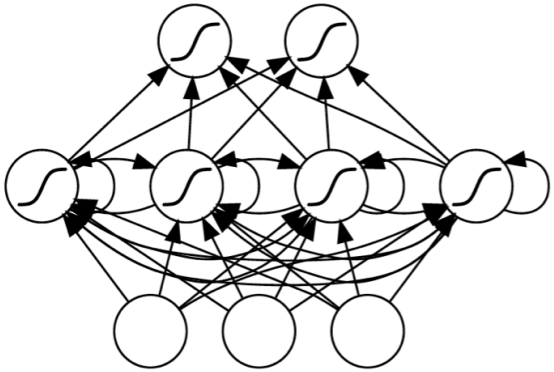
- 循环神经网络基本原理、运算及运行过程
- 语言模型
- 结构类别及应用
- 训练方法 BPTT（back-propagation through time）
- LSTM （long short-term memory）
- LSTM的输入输出和参数个数（应用）
- 注意力机制

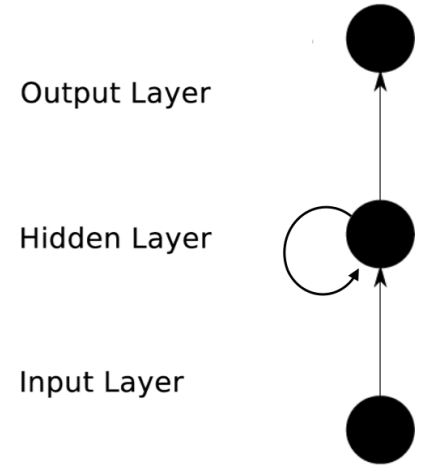
RNN

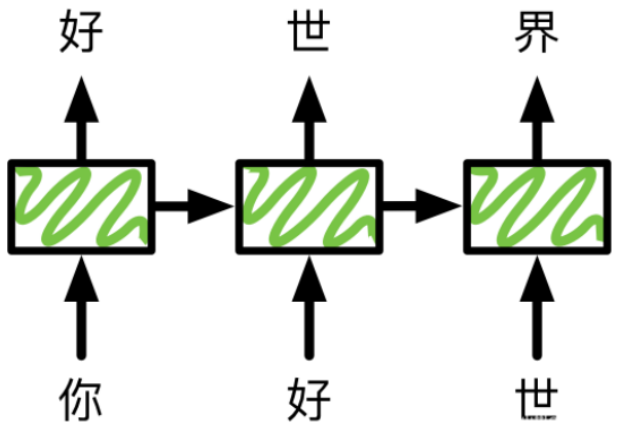
- **循环神经网络 Recurrent NN**
- **区别于Recursive NN，递归神经网络**
- **时间展开　空间结构展开**

RNN（Recurrent Neural Network）是一类用于处理序列数据的神经网络。

基础的神经网络只在层与层之间建立了权连接，RNN最大的不同之处就是在层之间的神经元之间也建立的权连接。







基本运算

- **每次迭代使用相同的参数和函数**

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

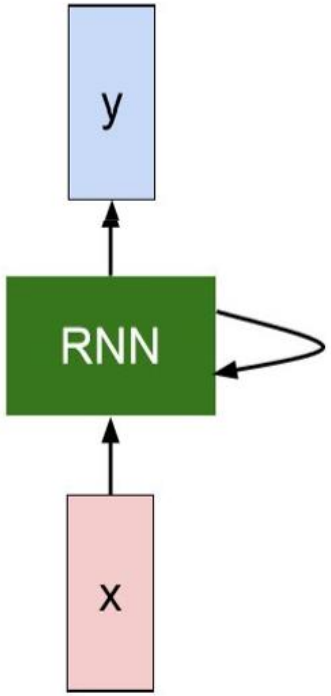
$$h_t = f_W(h_{t-1}, x_t)$$

new state

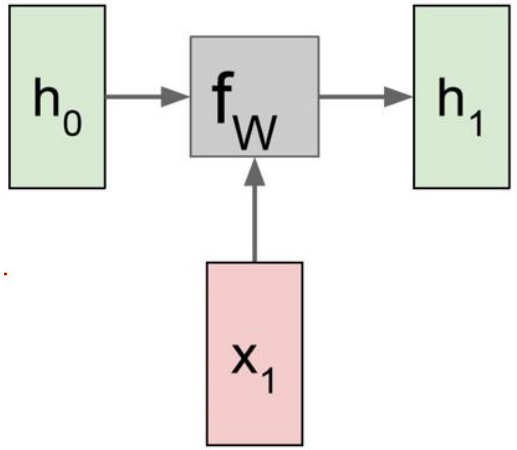
some function with parameters W

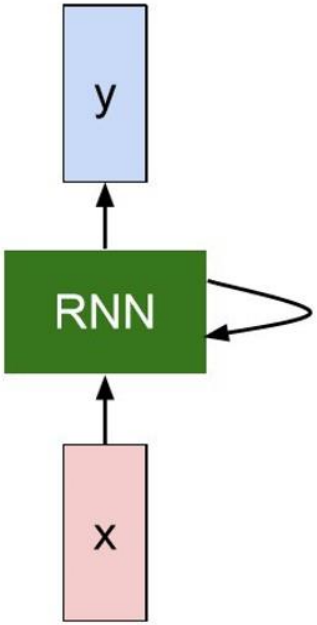
old state

input vector at some time step



RNN的计算





$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

语言模型

语言模型（language model）是自然语言处理的重要技术。自然语言处理中最常见的数据是文本数据。我们可以把一段自然语言文本看作一段离散的时间序列。假设一段长度为T的文本中的词依次为$w_1, w_2, \ldots, w_T$，那么在离散的时间序列中，w_t（$1 \leq t \leq T$）可看作在时间步（time step）t的输出或标签。给定一个长度为T的词的序列$w_1, w_2, \ldots, w_T$，语言模型将计算该序列的概率：

$$P(w_1, w_2, \ldots, w_T).$$

语言模型可用于提升语音识别和机器翻译的性能。例如，在语音识别中，给定一段“厨房里食油用完了”的语音，有可能会输出“厨房里食油用完了”和“厨房里石油用完了”这两个读音完全一样的文本序列。如果语言模型判断出前者的概率大于后者的概率，我们就可以根据相同读音的语音输出“厨房里食油用完了”的文本序列。在机器翻译中，如果对英文“you go first”逐词翻译成中文的话，可能得到“你走先”“你先走”等排列方式的文本序列。如果语言模型判断出“你先走”的概率大于其他排列方式的文本序列的概率，我们就可以把“you go first”翻译成“你先走”。

语言模型的计算

既然语言模型很有用，那该如何计算它呢？假设序列$w_1, w_2, \ldots, w_T$中的每个词是依次生成的，我们有

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1}).$$

例如，一段含有4个词的文本序列的概率

$$P(w_1, w_2, w_3, w_4) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)P(w_4 \mid w_1, w_2, w_3).$$

为了计算语言模型，我们需要计算词的概率，以及一个词在给定前几个词的情况下的条件概率，即语言模型参数。设训练数据集为一个大型文本语料库，如维基百科的所有条目。词的概率可以通过该词在训练数据集中的相对词频来计算。例如，$P(w_1)$可以计算为w_1在训练数据集中的词频（词出现的次数）与训练数据集的总词数之比。因此，根据条件概率定义，一个词在给定前几个词的情况下的条件概率也可以通过训练数据集中的相对词频计算。例如，$P(w_2 \mid w_1)$可以计算为w_1, w_2两词相邻的频率与w_1词频的比值，因为该比值即$P(w_1, w_2)$与$P(w_1)$之比；而$P(w_3 \mid w_1, w_2)$同理可以计算为w_1、w_2和w_3这3个词相邻的频率与w_1和w_2这2个词相邻的频率的比值。以此类推。

语言模型的计算

$P(w2/w1) = \#(w1,w2)/\#w1$

$P(w3/w1,w2) = \#(w1,w2,w3)/\#(w1,w2)$

例：学校 学校好

N元语法

当序列长度增加时，计算和存储多个词共同出现的概率的复杂度会呈指数级增加。n元语法通过马尔可夫假设（虽然并不一定成立）简化了语言模型的计算。这里的马尔可夫假设是指一个词的出现只与前面n个词相关，即n阶马尔可夫链（Markov chain of order n）。如果$n = 1$，那么有$P(w_3 \mid w_1, w_2) = P(w_3 \mid w_2)$。如果基于$n-1$阶马尔可夫链，我们可以将语言模型改写为

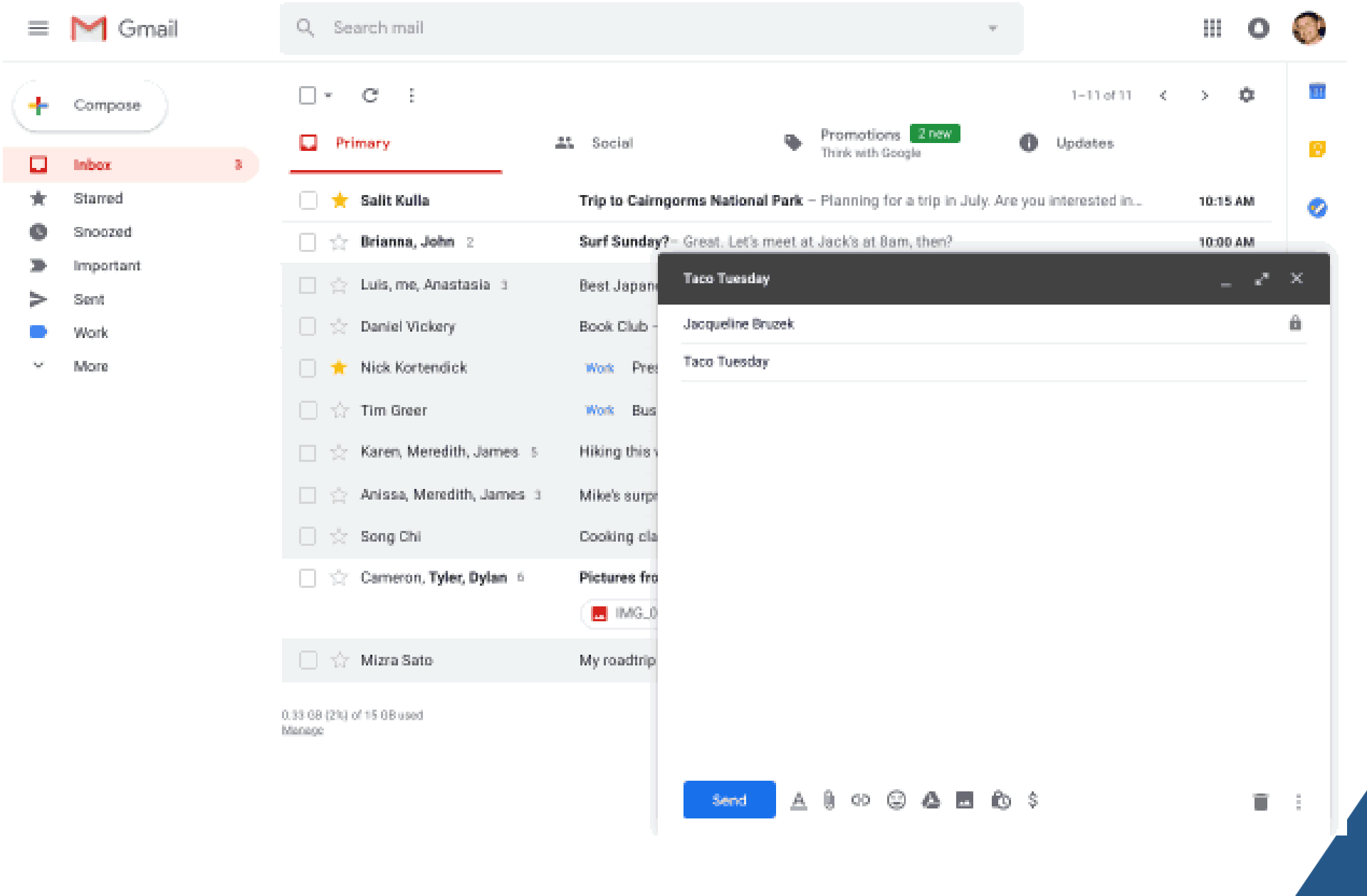
$$P(w_1, w_2, \ldots, w_T) \approx \prod_{t=1}^{T} P(w_t \mid w_{t-(n-1)}, \ldots, w_{t-1}).$$

以上也叫n元语法（n-grams）。它是基于$n-1$阶马尔可夫链的概率语言模型。当n分别为1、2和3时，我们将其分别称作一元语法（unigram）、二元语法（bigram）和三元语法（trigram）。例如，长度为4的序列w_1, w_2, w_3, w_4在一元语法、二元语法和三元语法中的概率分别为

$$\begin{aligned}
P(w_1, w_2, w_3, w_4) &= P(w_1)P(w_2)P(w_3)P(w_4),\\
P(w_1, w_2, w_3, w_4) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2)P(w_4 \mid w_3),\\
P(w_1, w_2, w_3, w_4) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)P(w_4 \mid w_2, w_3).
\end{aligned}$$

例:
he eats pizza与he drinks pizza

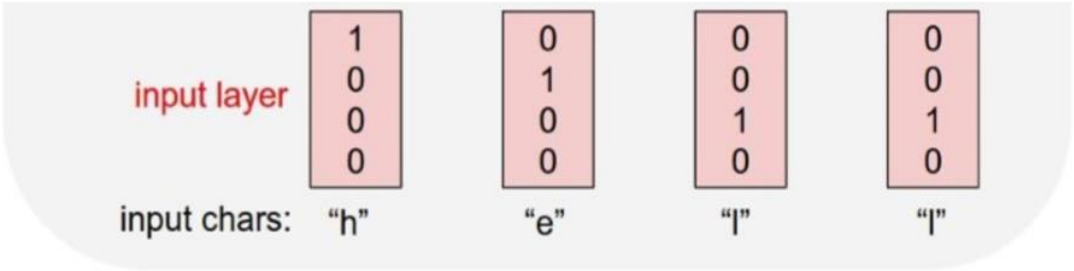
迅雷不及掩耳盗铃


Gmail
Search mail
Compose
Inbox 3
Starred
Snoozed
Important
Sent
Work
More
1–11 of 11
Primary
Social
Promotions 2 new
Think with Google
Updates
Salit Kulla
Trip to Cairngorms National Park – Planning for a trip in July. Are you interested in...
10:15 AM
Brianna, John 2
Surf Sunday? – Great. Let's meet at Jack's at 8am, then?
10:00 AM
Luis, me, Anastasia 3
Daniel Vickery
Nick Kortendick
Work
Tim Greer
Work
Karen, Meredith, James 5
Hiking this
Anissa, Meredith, James 3
Song Chi
Cooking cla
Cameron, Tyler, Dylan 6
Pictures fro
Mizra Sato
My roadtrip
0.33 GB (2%) of 15 GB used
Manage
Taco Tuesday
Jacqueline Bruzek
Taco Tuesday
Send

字符级的语言模型

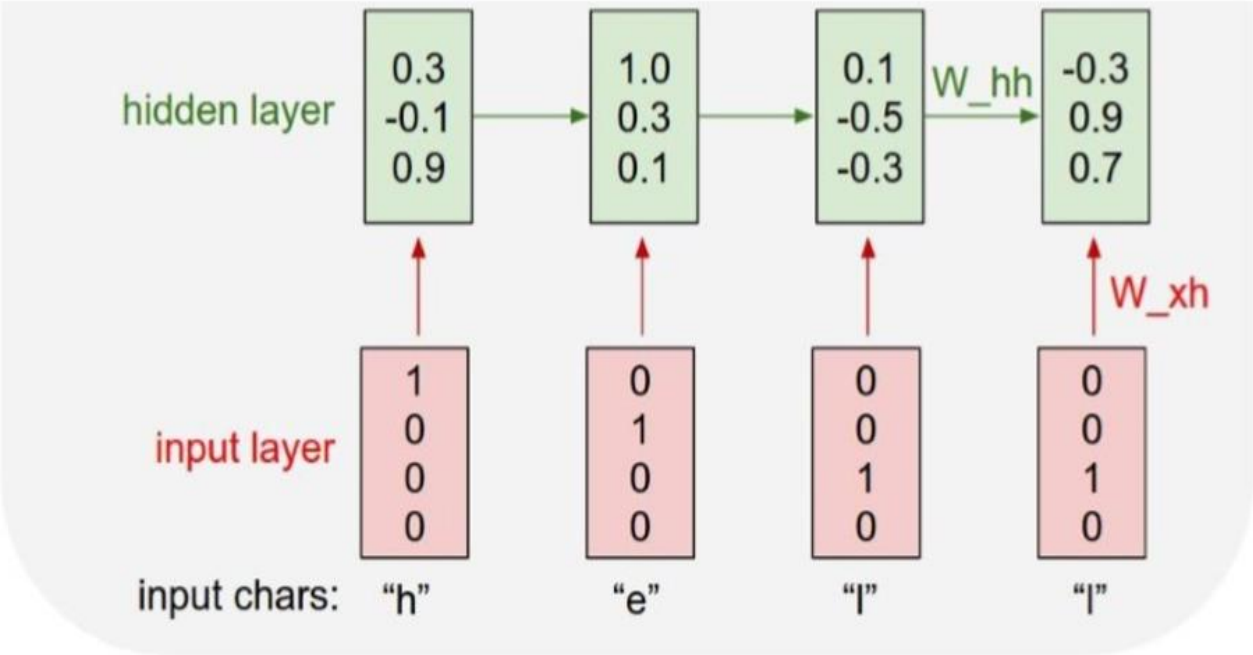
Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

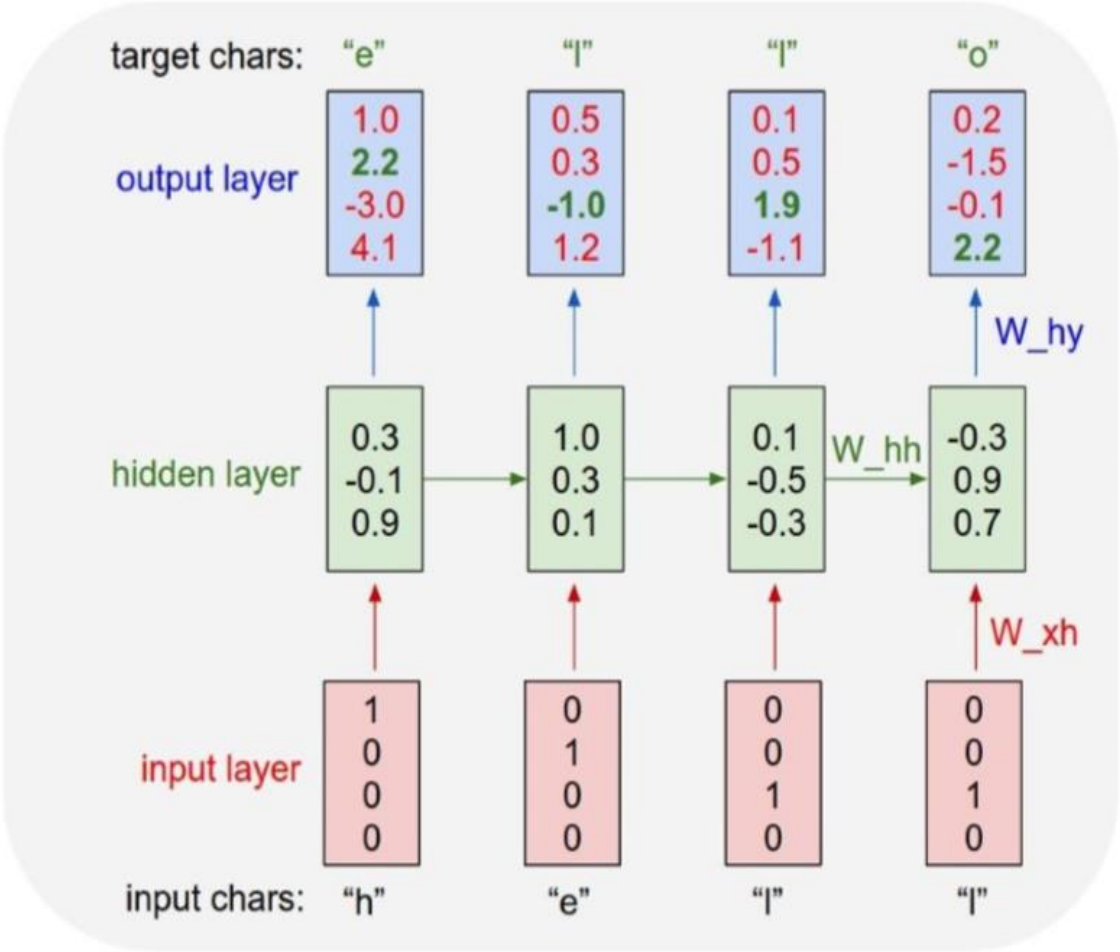


字符级的语言模型

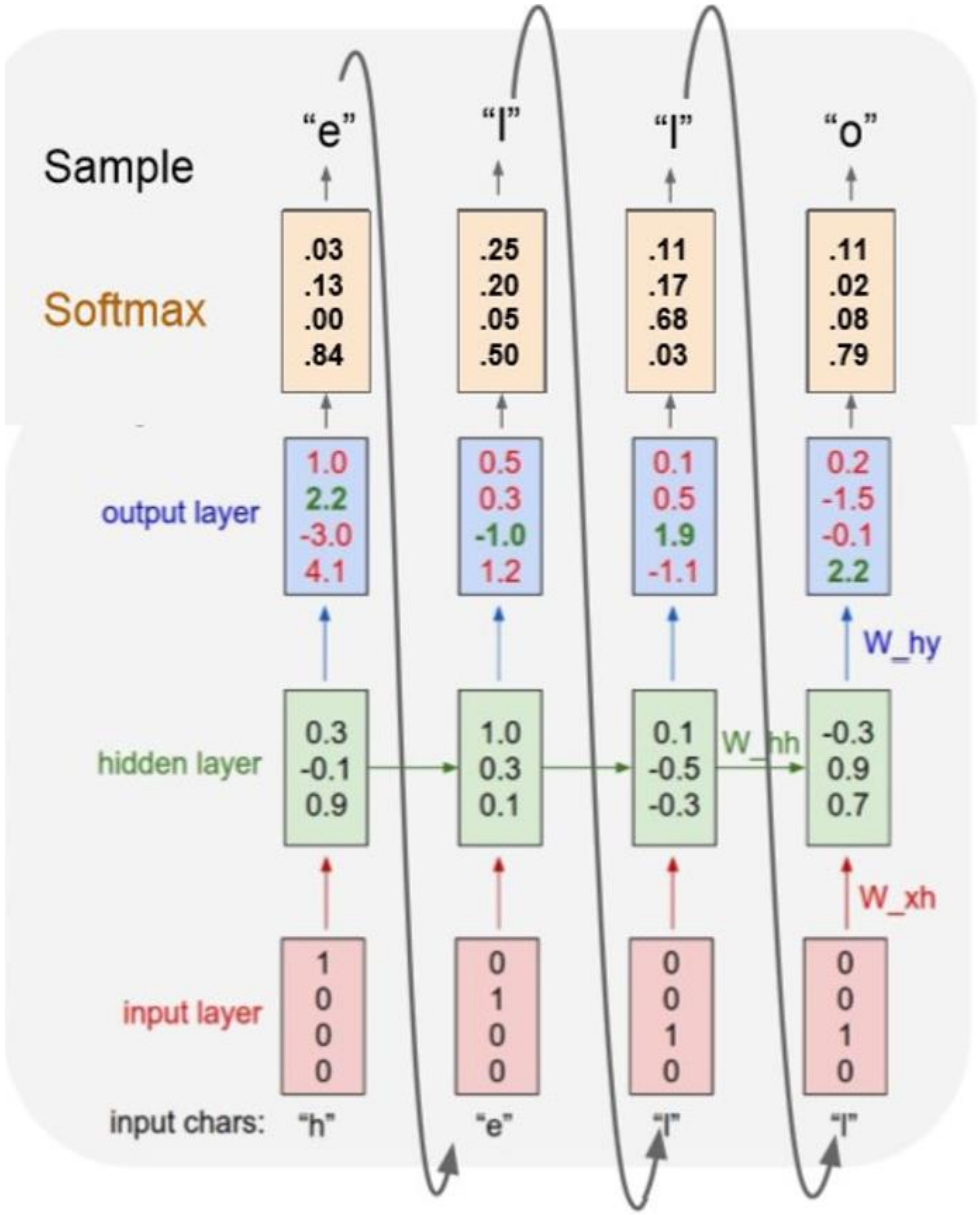
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



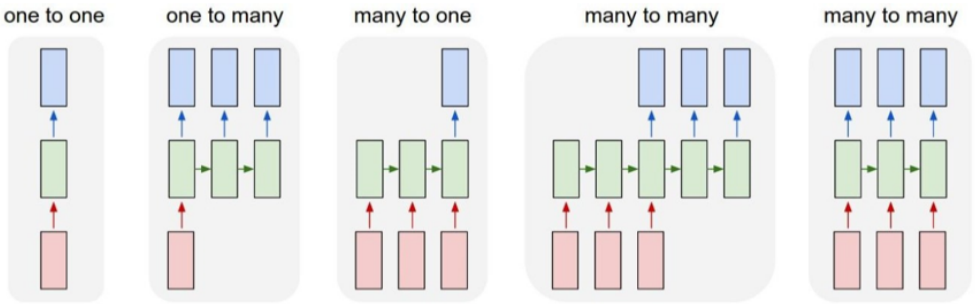
字符级的语言模型


target chars:
"e"
"l"
"l"
"o"
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
"h"
"e"
"l"
"l"

RNN

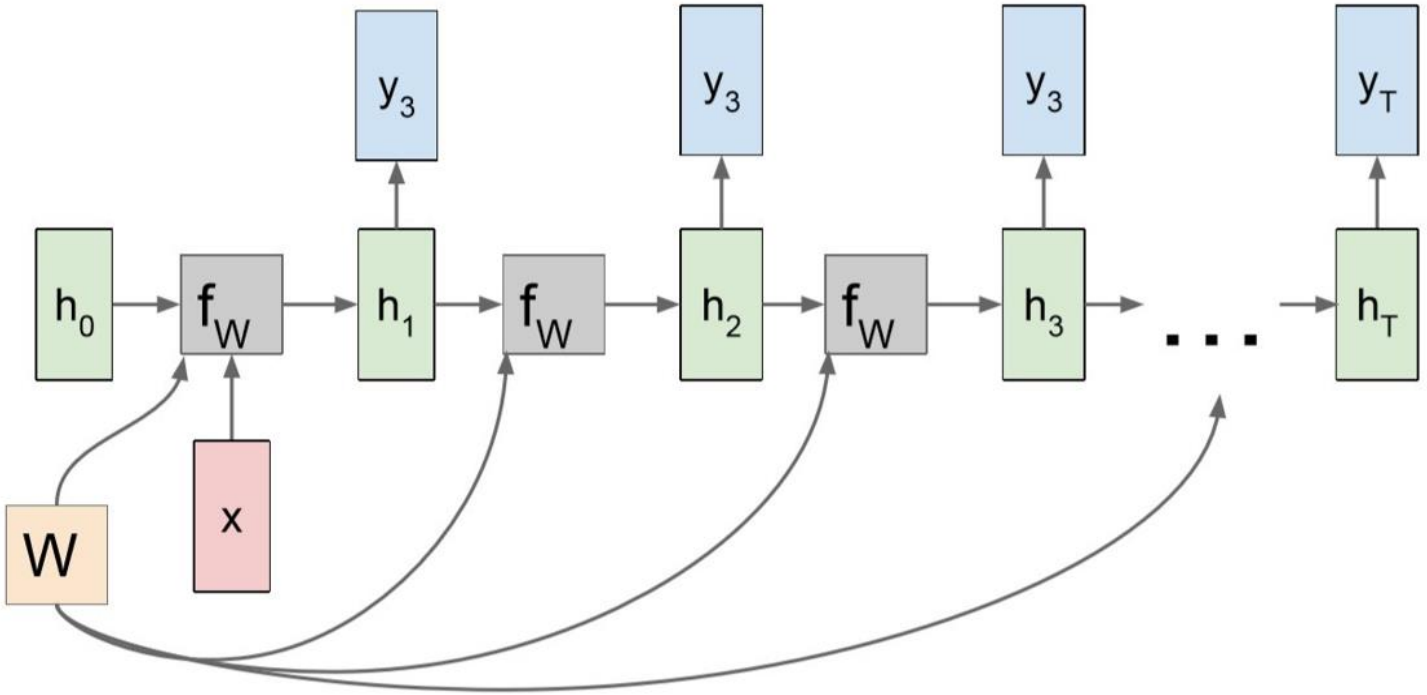

Sample
"e"
"l"
"l"
"o"
Softmax
.03
.13
.00
.84
.25
.20
.05
.50
.11
.17
.68
.03
.11
.02
.08
.79
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
"h"
"e"
"l"
"l"

RNN的类别

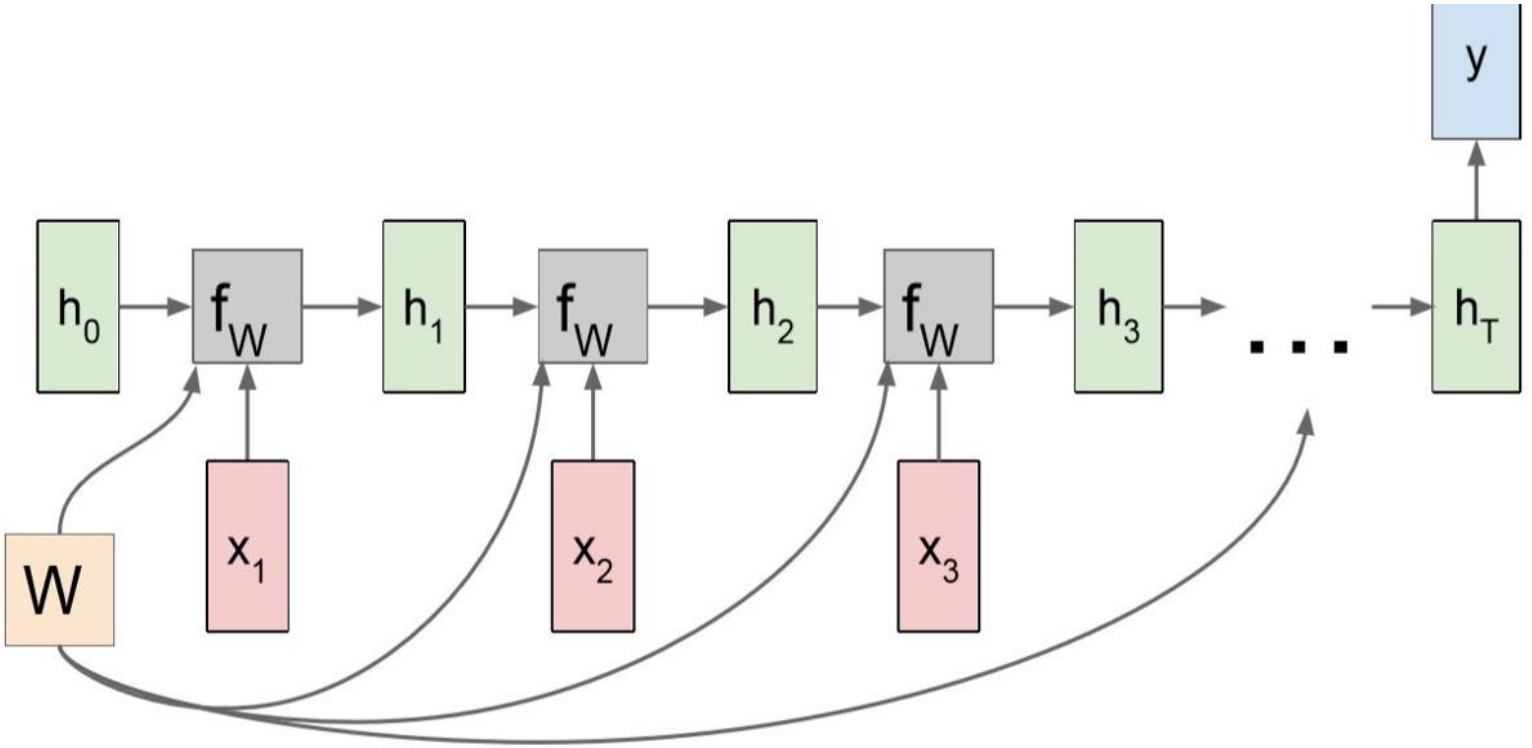


- Image Captioning
  - image → sequence of words
- Sentiment Classification
  - sequence of words → sentiment
- Machine Translation
  - seq of words → seq of words
- Video classification on frame level

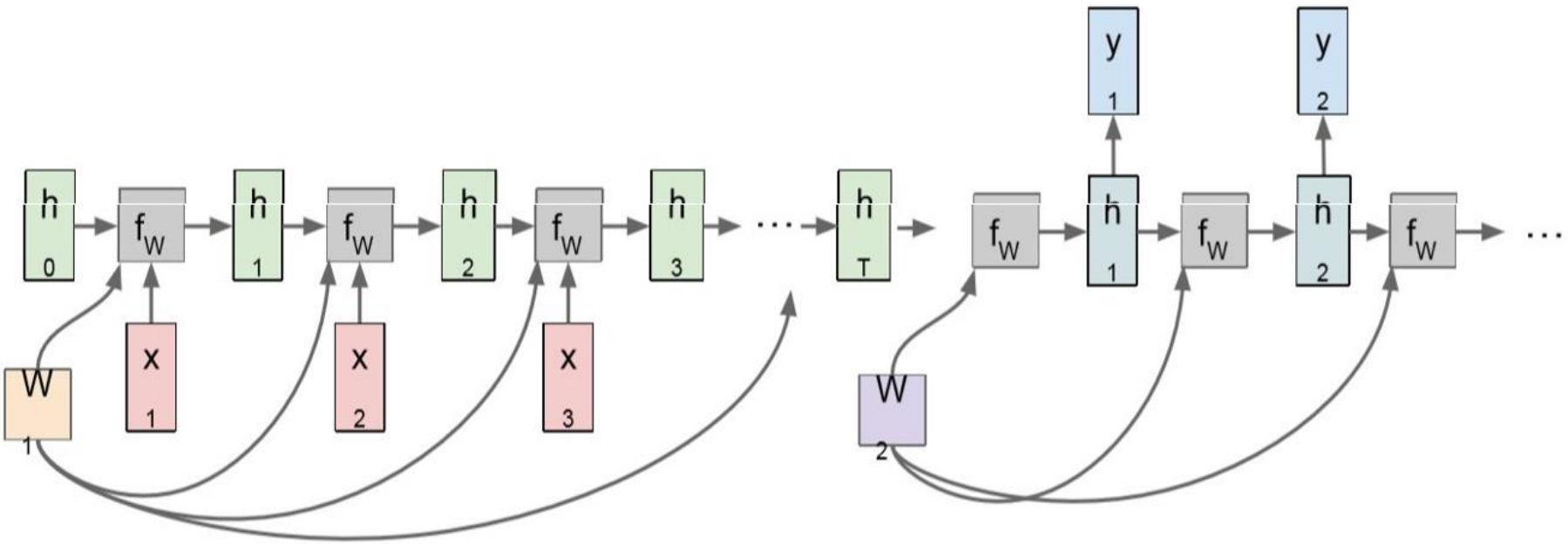
One to Many



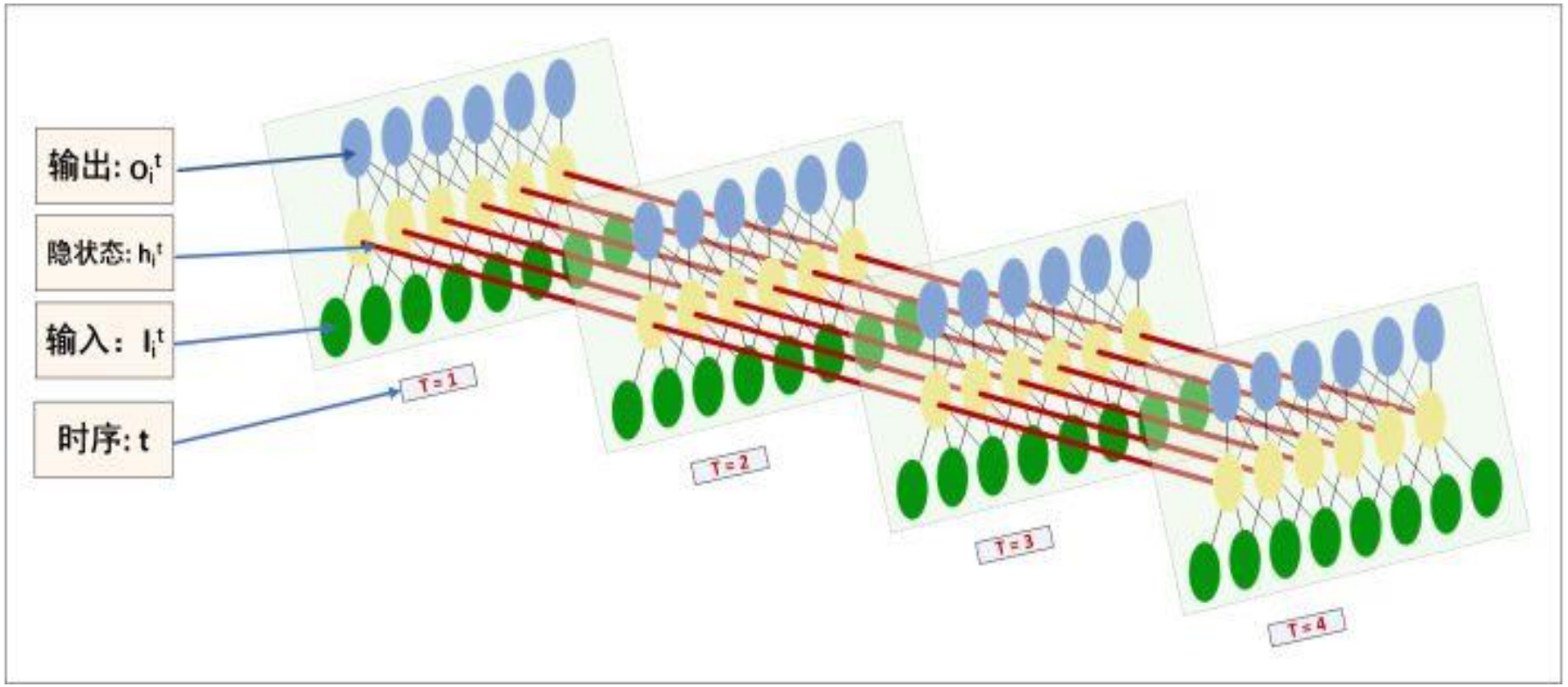
Many to One



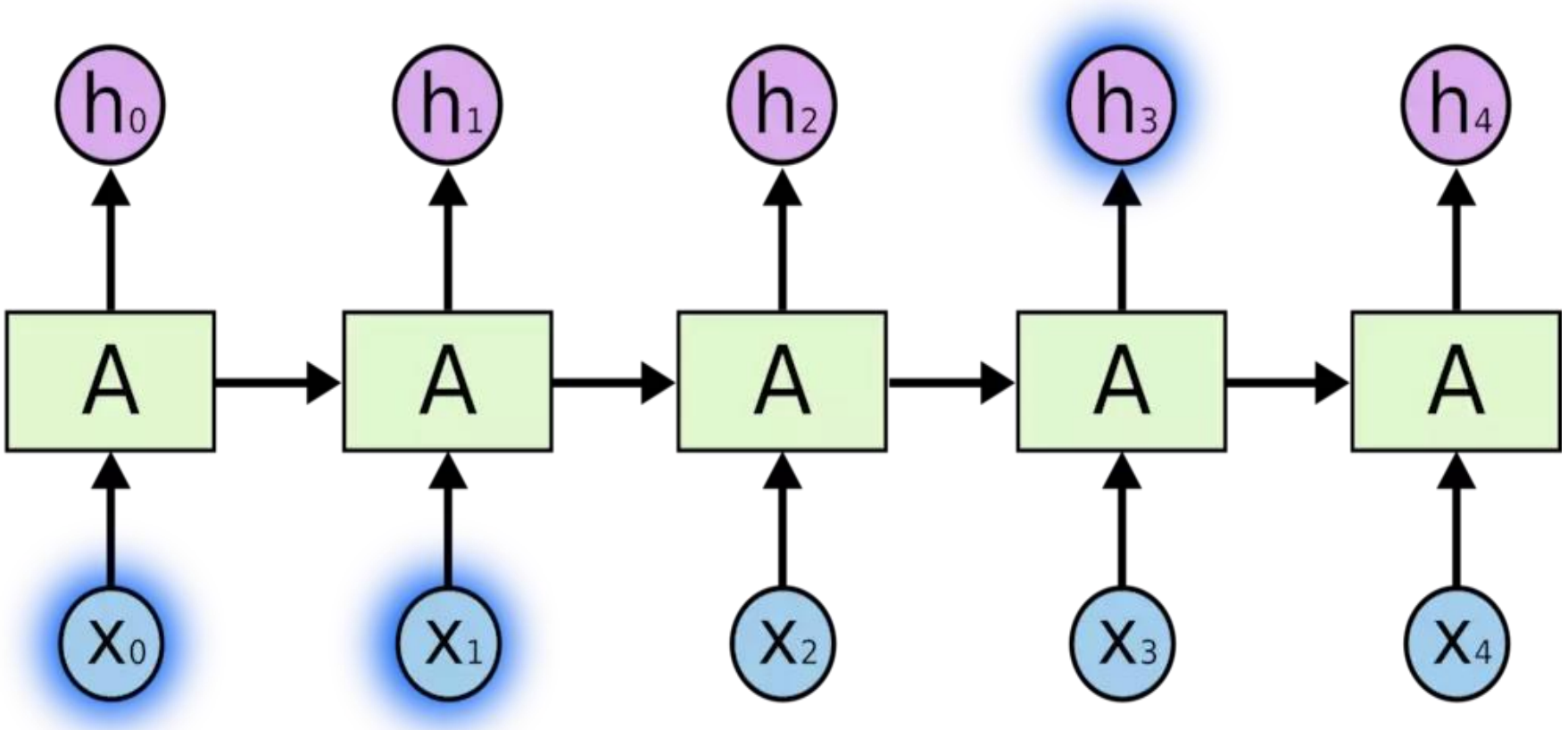
Sequence to Sequence



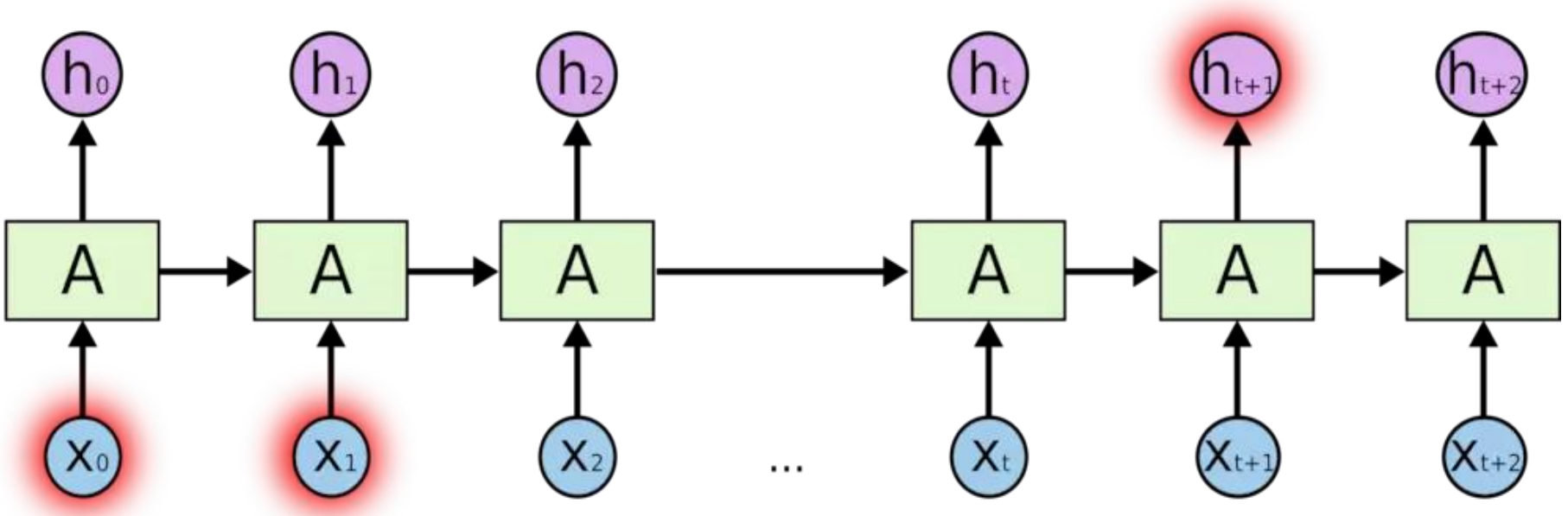
其它表示形式



长期依赖问题 (LONG-TERM DEPENDENCIES)

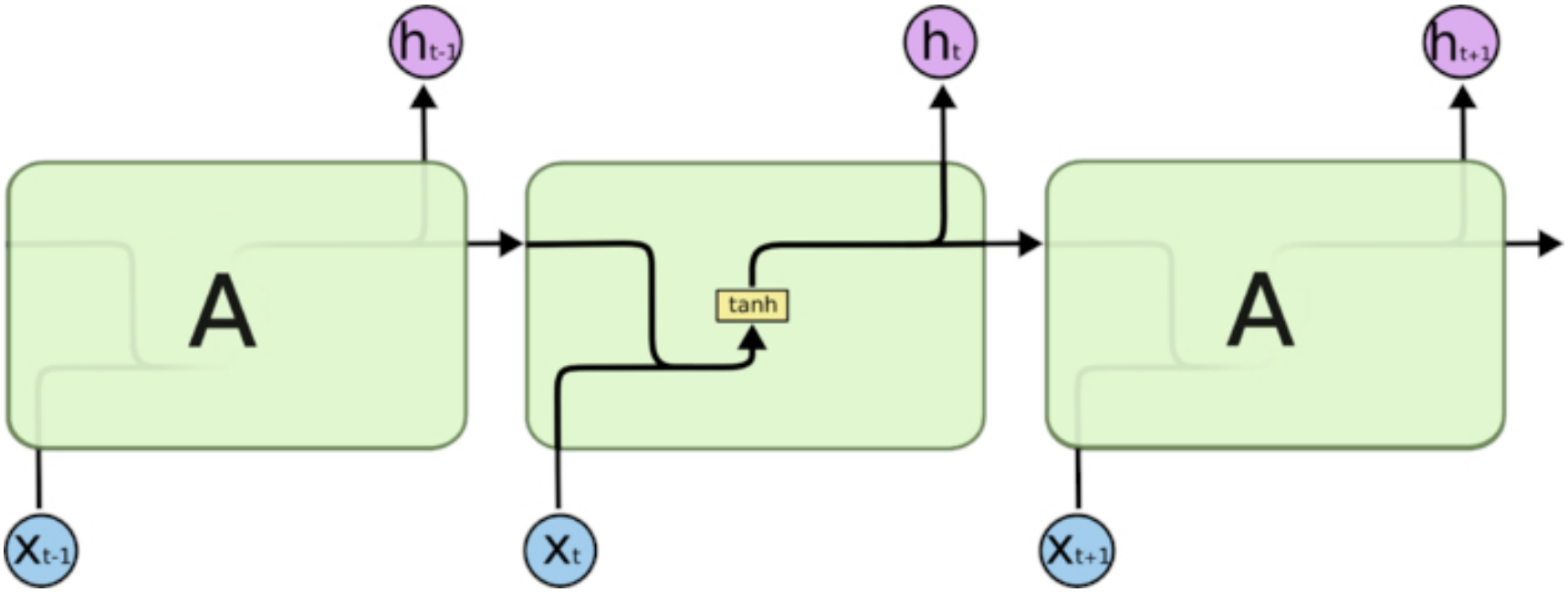


不太长的相关信息和位置间隔

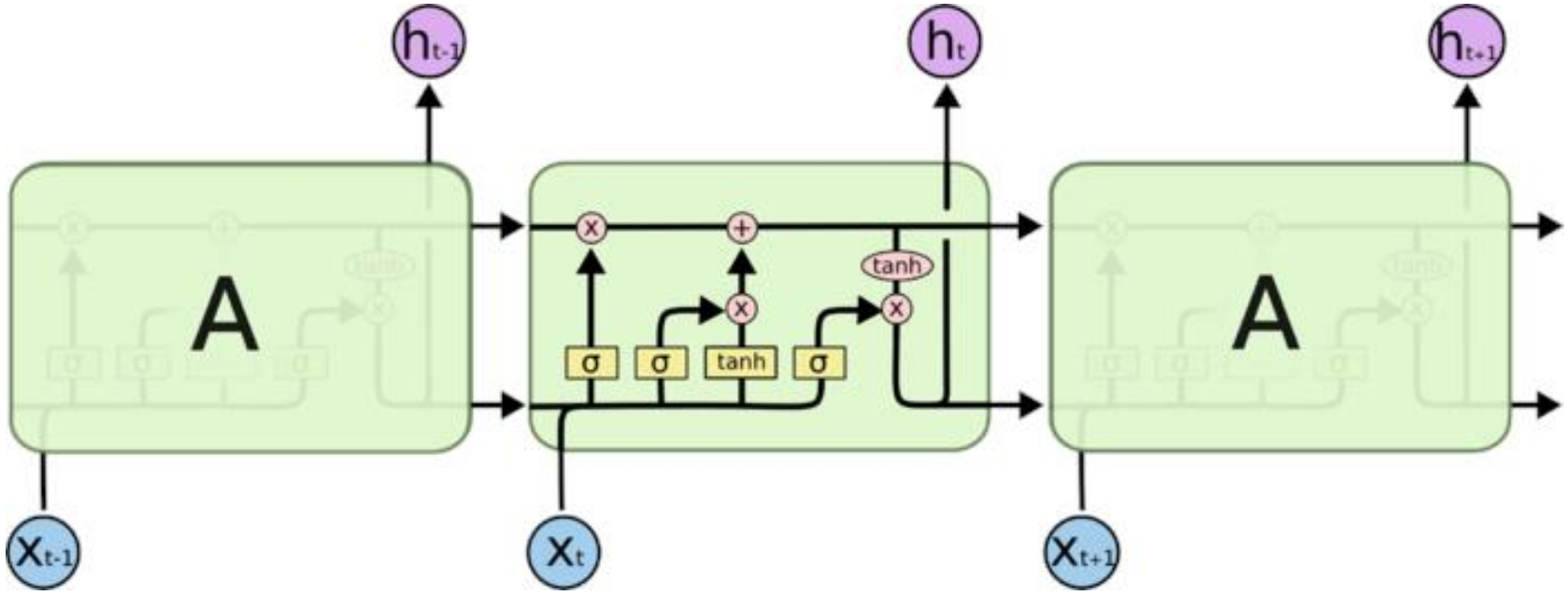


相当长的相关信息和位置间隔

RNN

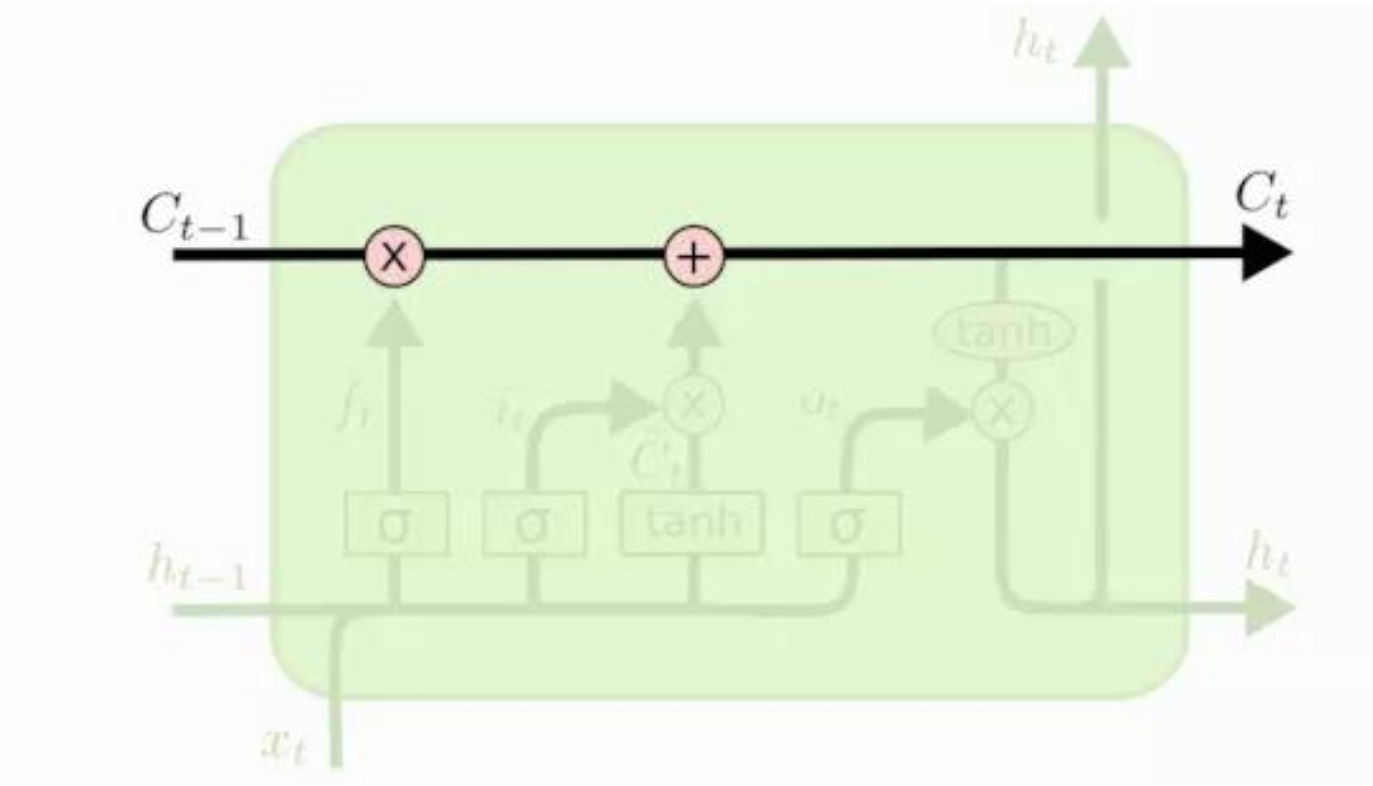


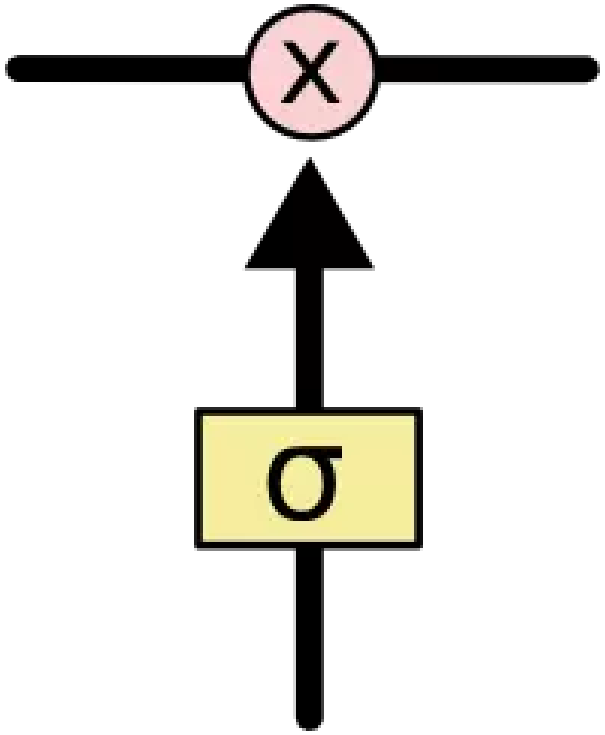
LSTM（LONG SHORT-TERM MEMORY）



- 遗忘门/输入门/输出门
- cell 最上面的一条线的状态即 s(t) 代表了长时记忆，而下面的 h(t)则代表了工作记忆或短时记忆

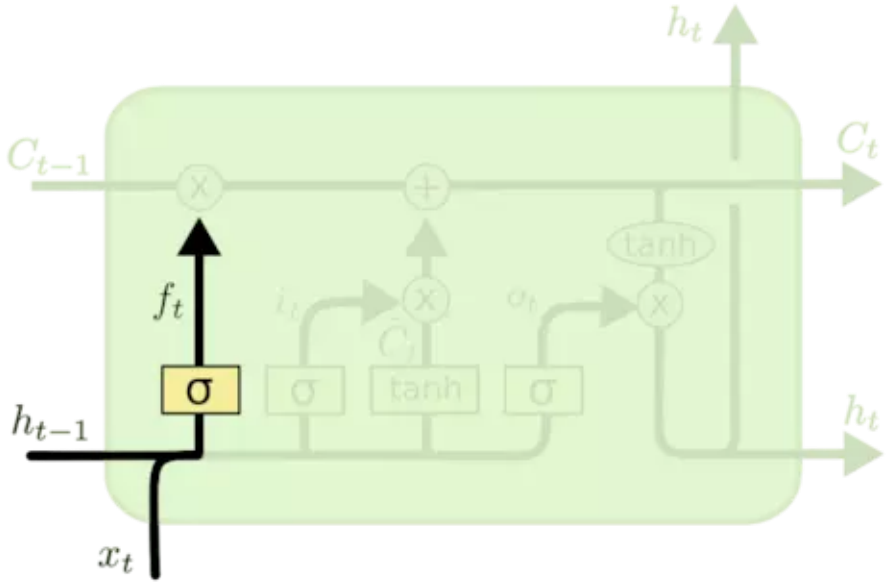
CELL状态传递





包含一个 sigmoid 神经网络层
和一个 pointwise 乘法操作

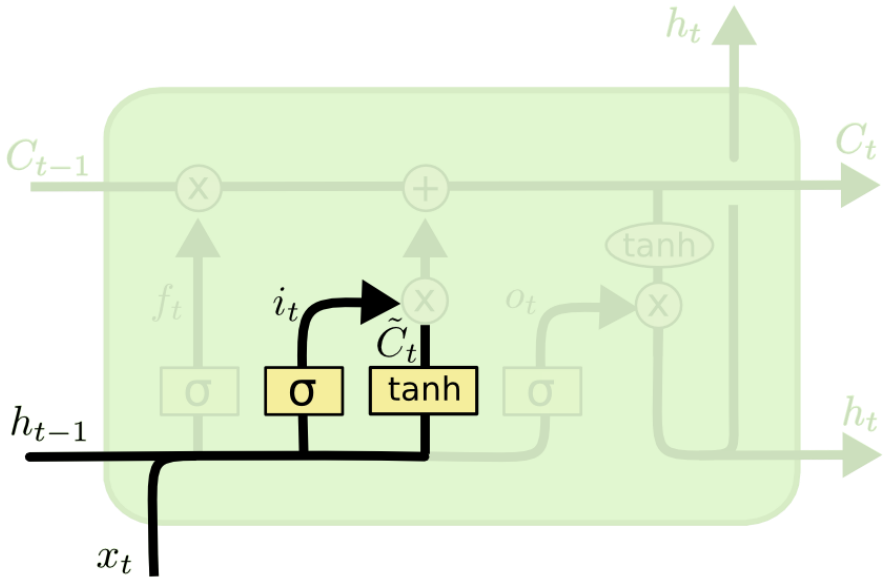
遗忘门



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

concatenate

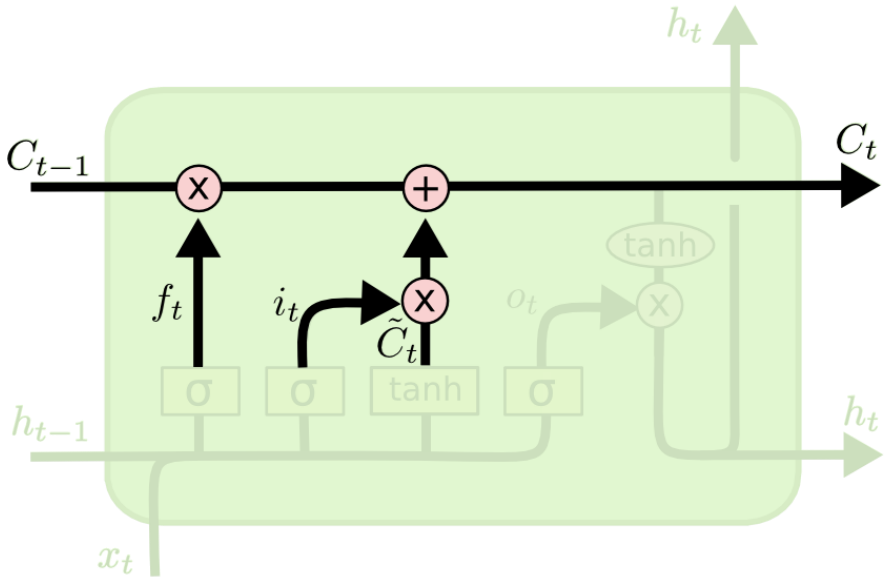
输入门



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

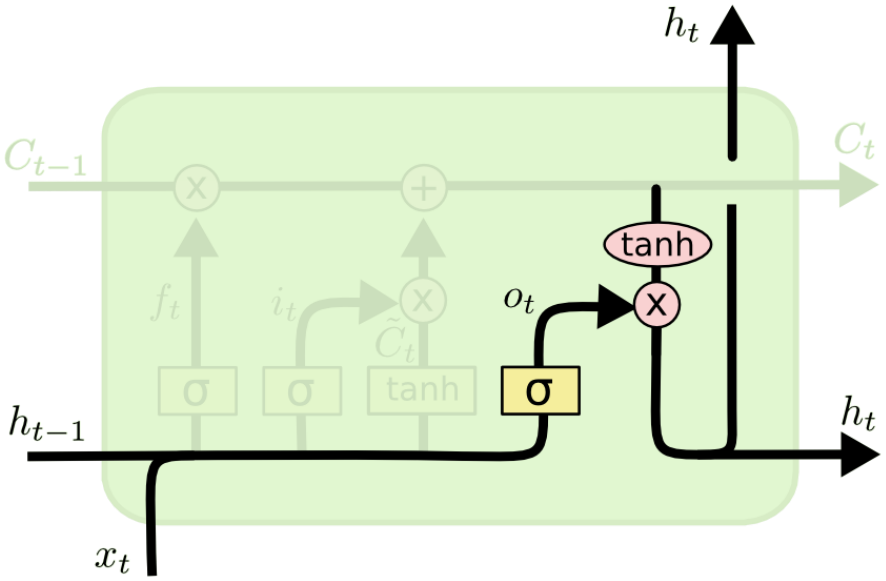
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

更新过程



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

输出门



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

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

LSTM层参数个数

- m是输入（位数）向量维数，n是隐向量维数（隐单元个数）
- 参数个数?

LSTM层参数个数

- m是输入（位数）向量维数，n是隐向量维数（隐单元个数）
- 参数个数（不包括输出层): 4*[(m+n)*n+n]

LSTM的输入输出

step1, raw text:

接触LSTM模型不久，简单看了一些相关的论文，还没有动手实现过。然而至今仍然想不通LSTM神经网络究竟是怎么工作的。……

step2, tokenize (中文得分词):

sentence1: 接触 LSTM 模型 不久 ， 简单 看了 一些 相关的 论文 ， 还 没有 动手 实现过 。

sentence2: 然而 至今 仍然 想不通 LSTM 神经网络 究竟是 怎么 工作的。

……

step3, dictionarize:

sentence1: 1 34 21 98 10 23 9 23

sentence2: 17 12 21 12 8 10 13 79 31 44 9 23

……

step4, padding every sentence to fixed length:

sentence1: 1 34 21 98 10 23 9 23 0 0 0 0 0

sentence2: 17 12 21 12 8 10 13 79 31 44 9 23 0

……

step4, padding every sentence to fixed length:

sentence1: 1 34 21 98 10 23 9 23 0 0 0 0 0

sentence2: 17 12 21 12 8 10 13 79 31 44 9 23 0

.....

step5, mapping token to an embeddings:

sentence1:

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

,每一列代表一个词向量，词向量维度自行确定；矩阵列数固定为time_step length。

sentence2:

.....

step6, feed into RNNs as input:

假设 一个RNN的time_step 确定为 l ,则padded sentence length(step5中矩阵列数)固定为 l 。一次RNNs的run只处理一条sentence。每个sentence的每个token的embedding对应了每个时序 t 的输入 I_i^t 。一次RNNs的run，连续地将整个sentence处理完。

step7, get output:

看图，每个time_step都是可以输出当前时序 t 的隐状态 h_i^t ；但整体RNN的输出 o_i^t 是在最后一个time_step $t = l$ 时获取，才是完整的最终结果。

step4, padding every sentence to fixed length:

sentence1: 1 34 21 98 10 23 9 23 0 0 0 0 0

sentence2: 17 12 21 12 8 10 13 79 31 44 9 23 0

.....

step5, mapping token to an embeddings:

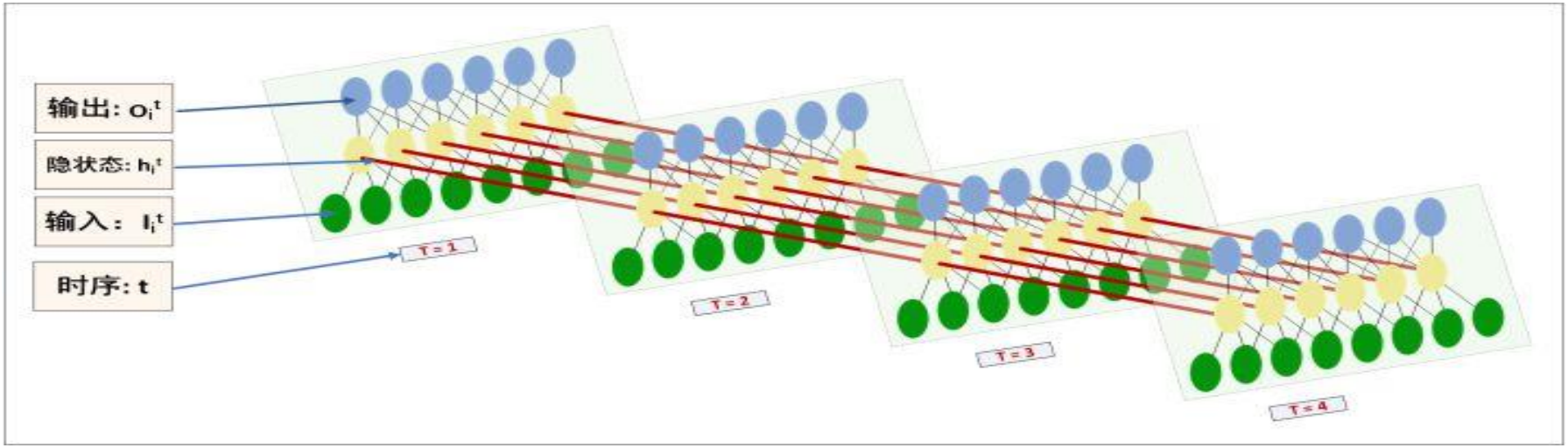
sentence1:

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

,每一列代表一个词向量，词向量维度自行确定；矩阵

列数固定为time_step length。

sentence2:



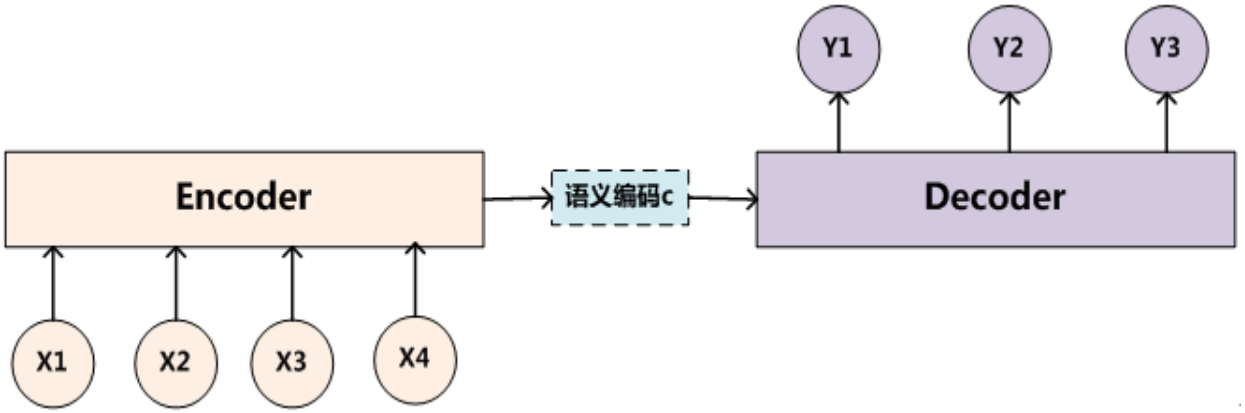
step7, get output:

看图，每个time_step都是可以输出当前时序 t 的隐状态 h_i^t ；但整体RNN的输出 o_i^t 是在最后一个time_step $t=l$ 时获取，才是完整的最终结果。

step8, further processing with the output:

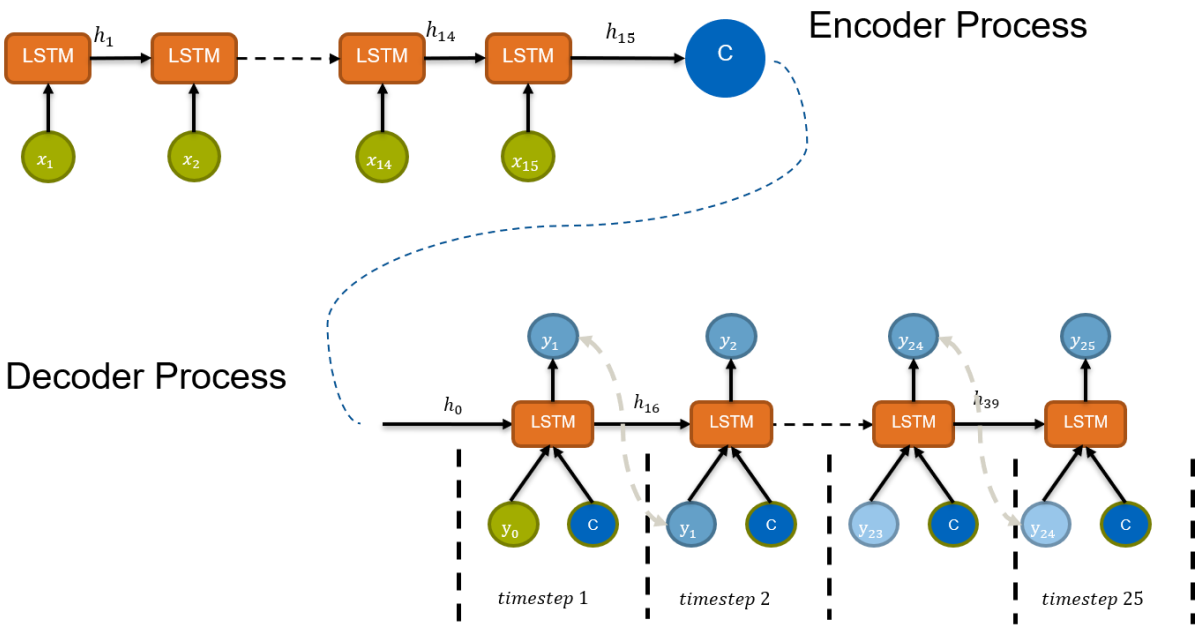
我们可以将output根据分类任务或回归拟合任务的不同，分别进一步处理。比如，传给cross_entropy&softmax进行分类……或者获取每个time_step对应的隐状态 h_i^t ，做seq2seq网络……或者搞创新……

ENCODER-DECODER结构

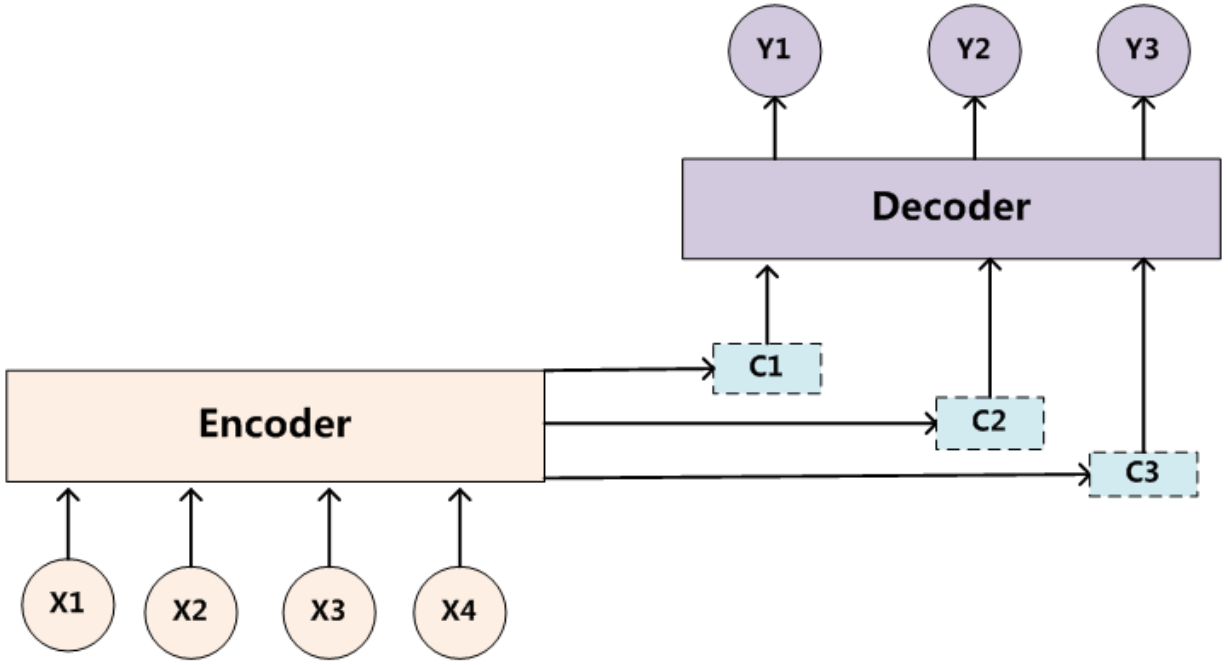


$$\mathbf{C} = \mathcal{F}(\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_m)$$

$$\mathbf{y}_i = \mathcal{G}(\mathbf{C}, \mathbf{y}_1, \mathbf{y}_2 \ldots \mathbf{y}_{i-1})$$



注意力机制



$$y_1 = f1(C_1)$$
$$y_2 = f1(C_2, y_1)$$
$$y_3 = f1(C_3, y_1, y_2)$$

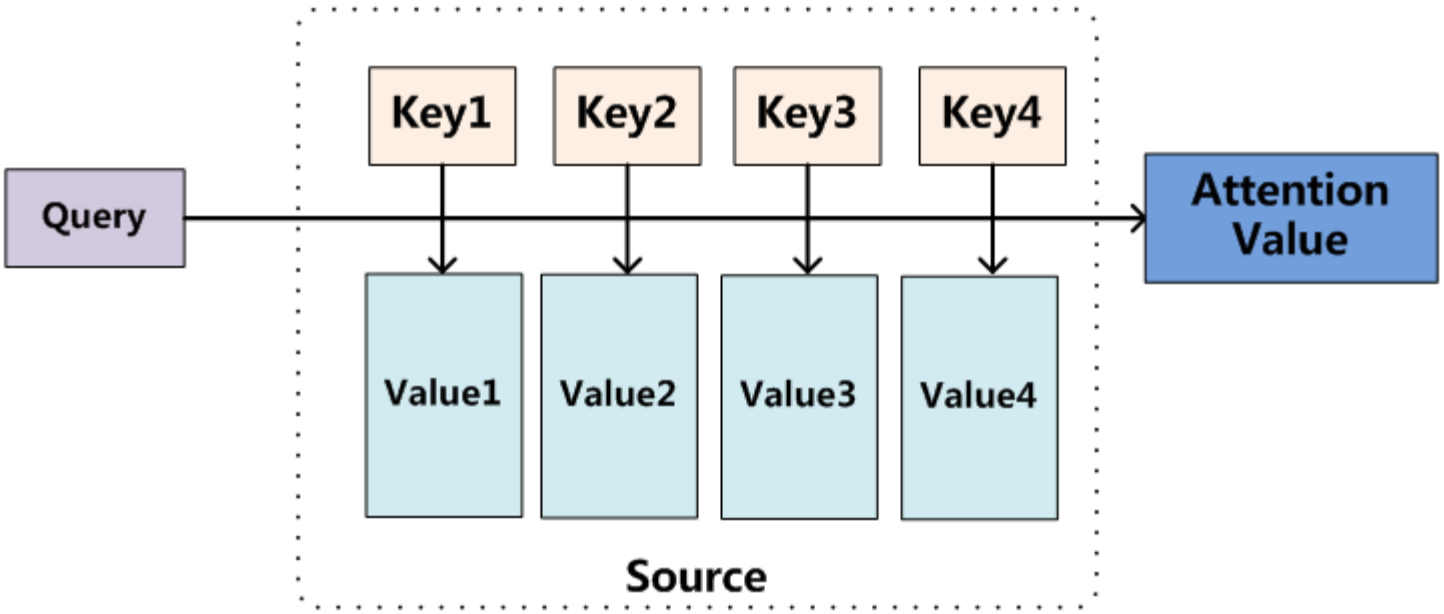
$$C_{汤姆} = g(0.6 \times f2("Tom"), 0.2 \times f2("Chase"), 0.2 \times f2("Jerry"))$$

$$C_{追逐} = g(0.2 \times f2("Tom"), 0.7 \times f2("Chase"), 0.1 \times f2("Jerry"))$$

$$C_{杰瑞} = g(0.3 \times f2("Tom"), 0.2 \times f2("Chase"), 0.5 \times f2("Jerry"))$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

注意力机制的本质



$$\mathbf{Attention(Query, Source)} = \sum_{i=1}^{L_x} \boldsymbol{Similarity(Query, Key_i)} * \boldsymbol{Value_i}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

TRANSFORMER

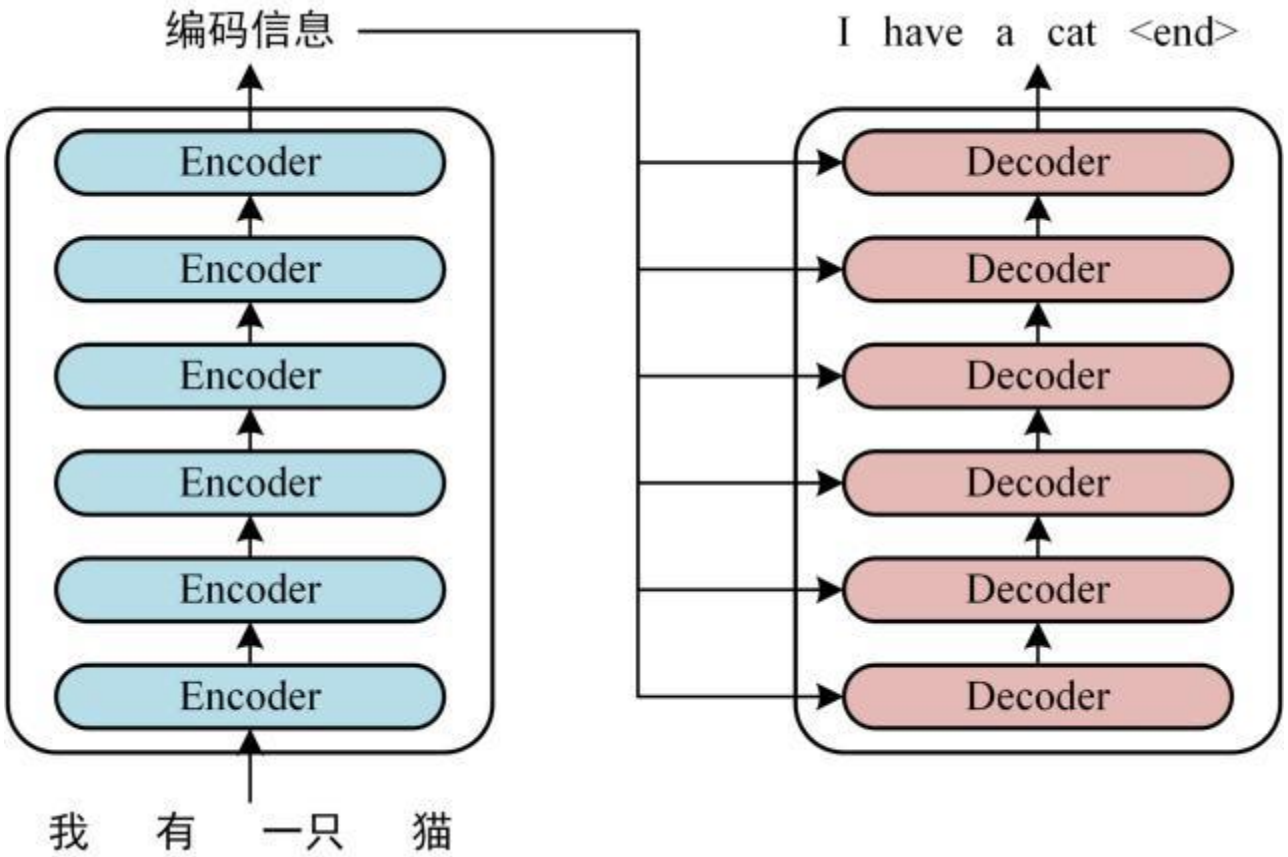
1. 背景
2. 整体结构（解析三步)

   TRM-EnDecoder-MH ATTN-Self ATTN
3. （宏观）工作流程
4. Self attention
5. Multi-head attention
6. Encoder内部结构
7. Decoder内部结构
8. 位置编码，teacher forcing等

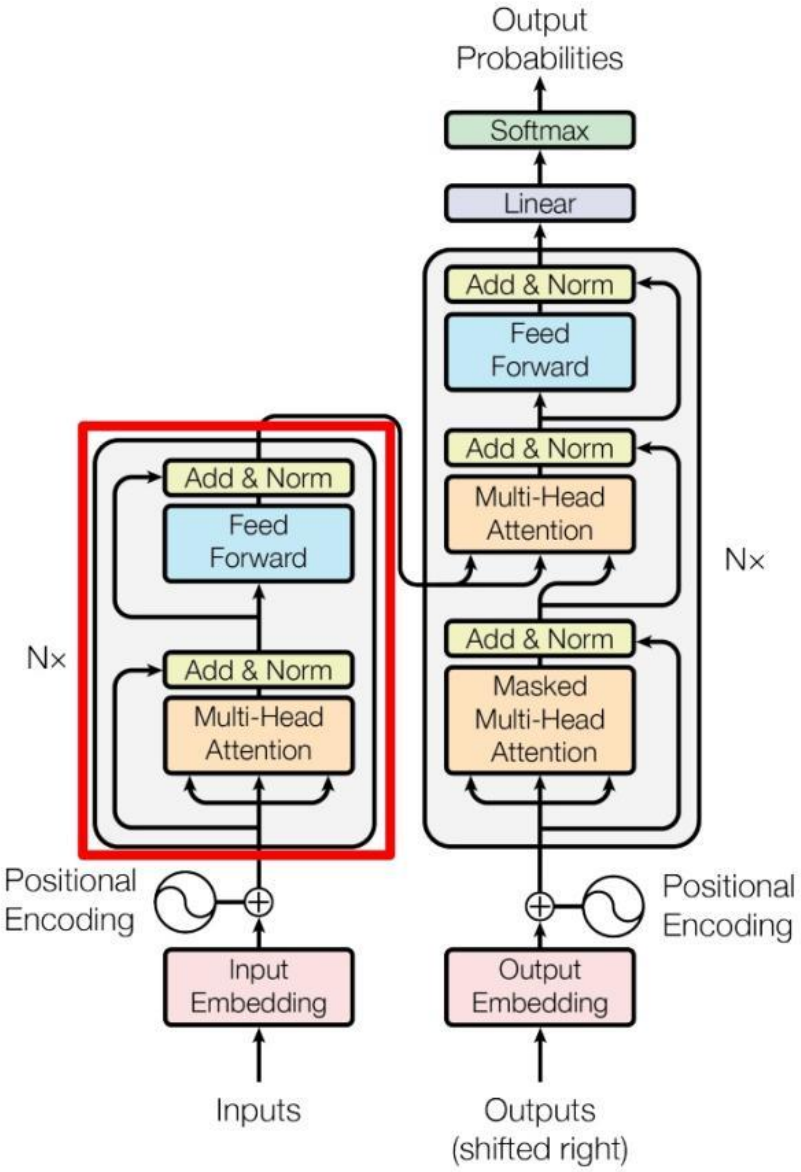
1. TRANSFORMER背景

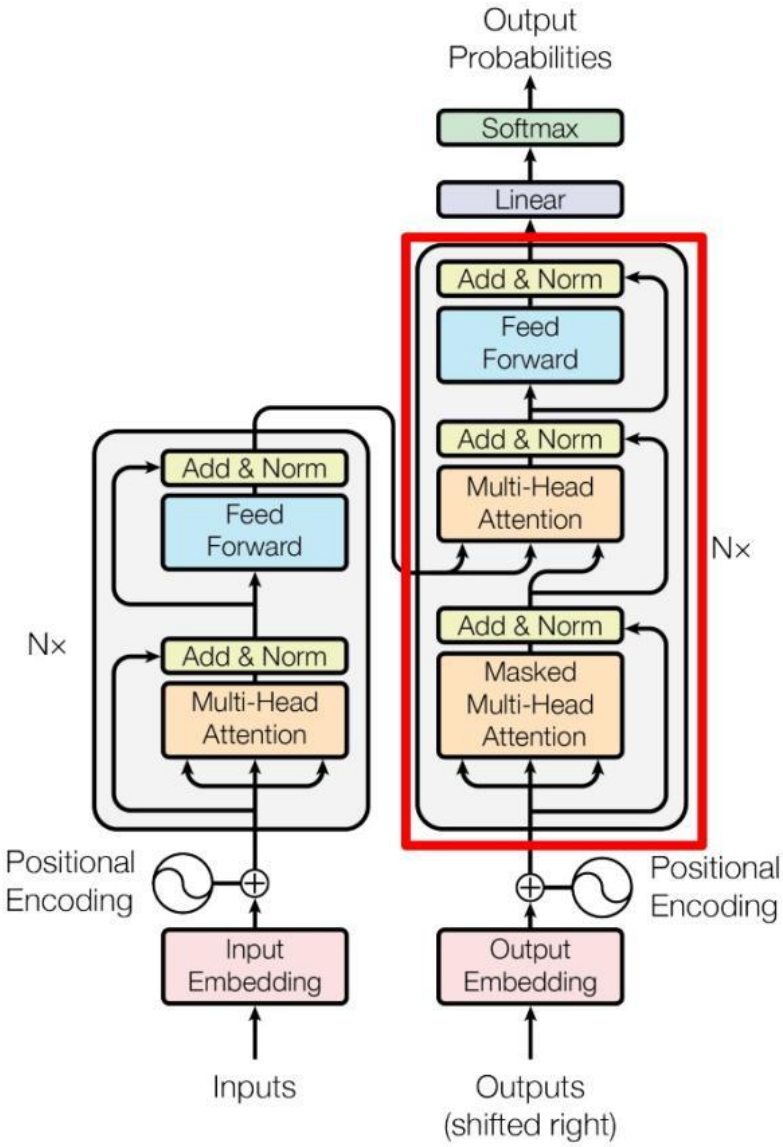
- Transformer出自于Google于2017年发表的论文《Attention is all you need》
- 最开始是用于机器翻译，取得了非常好的效果。
- 在CV、RS等领域也取得了非常不错的表现。

2. TRANSFORMER整体结构

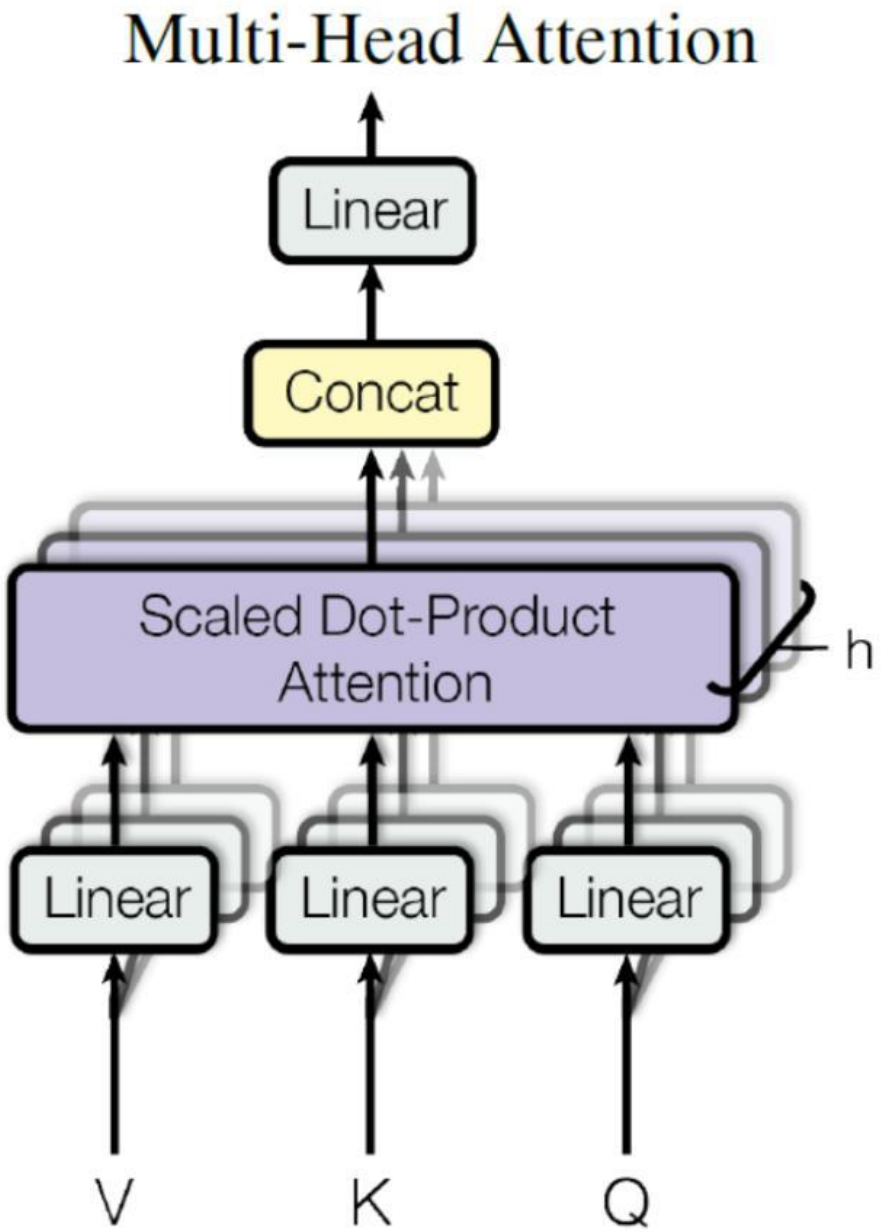


2. 结构解析一步：ENCODER-DECODER



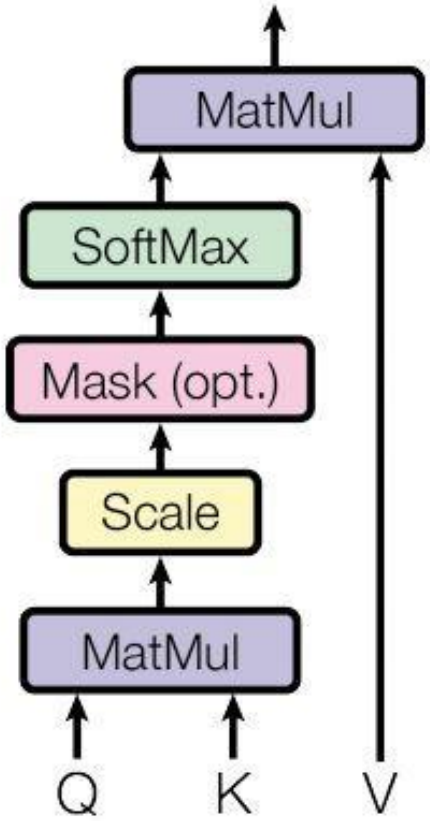


2. 结构解析二步


Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

2. 结构解析三步：SELF-ATTENTION

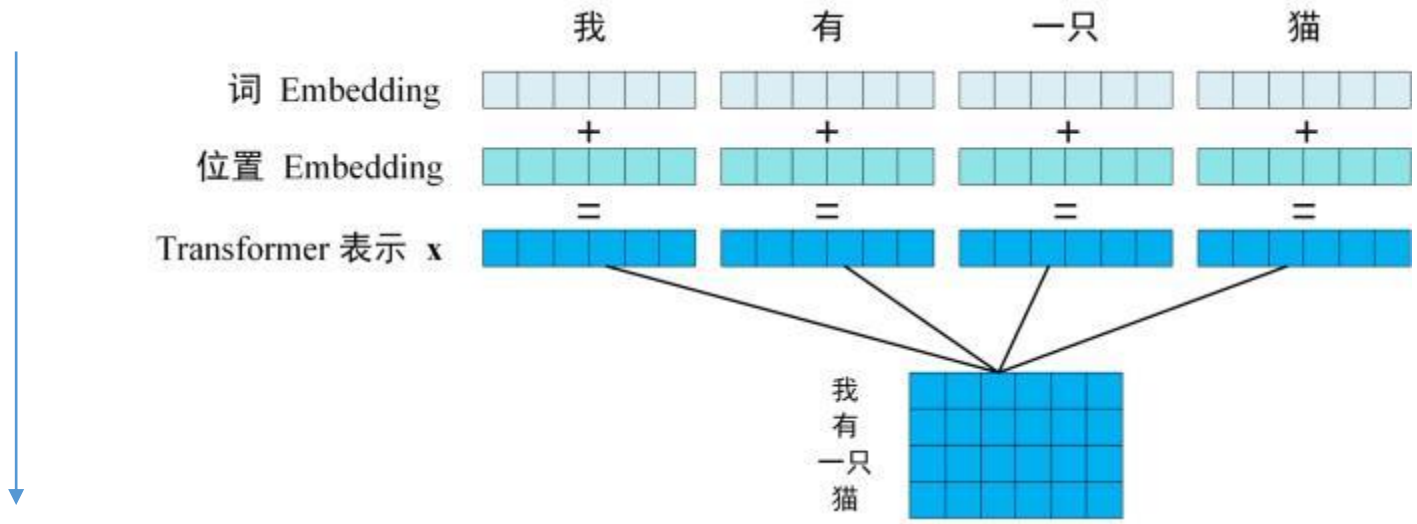
Scaled Dot-Product Attention



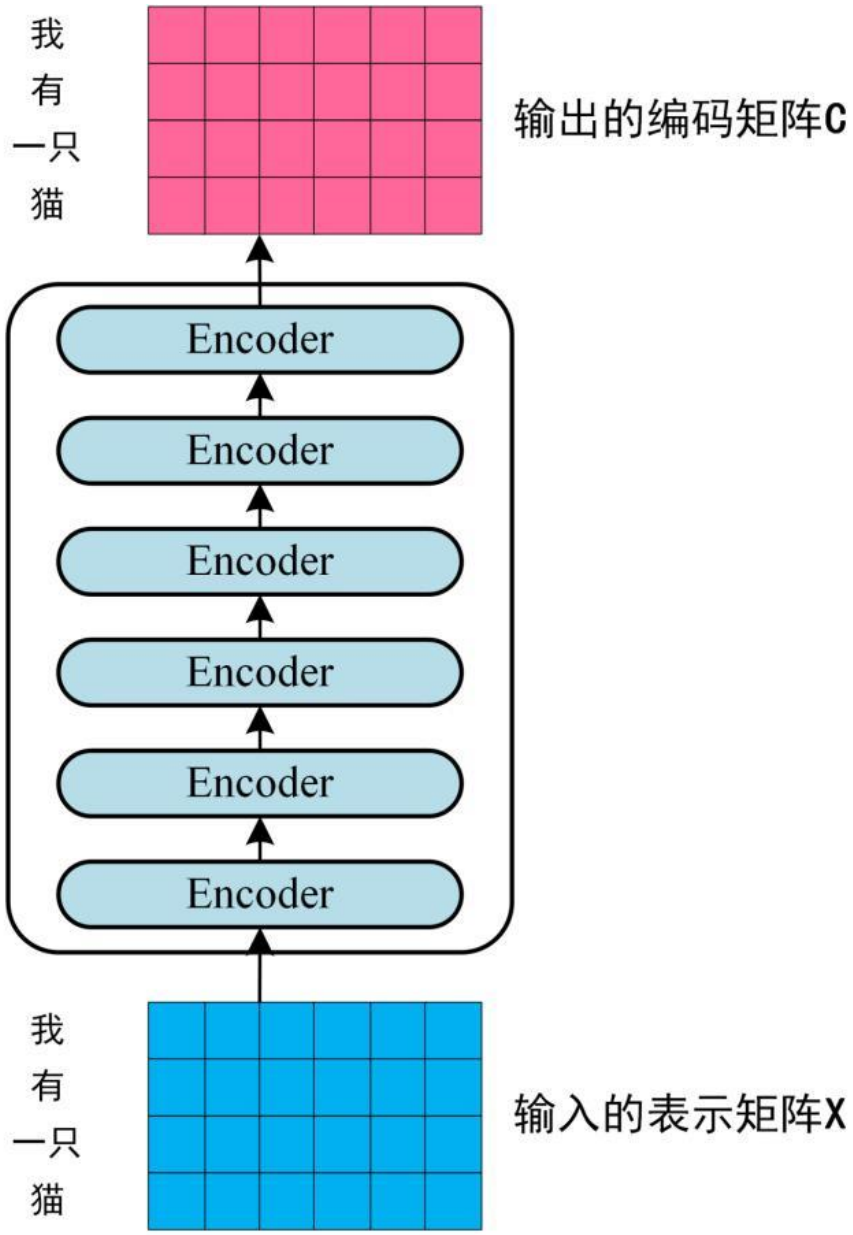
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

d_k是Q, K矩阵的列数，即向量维度

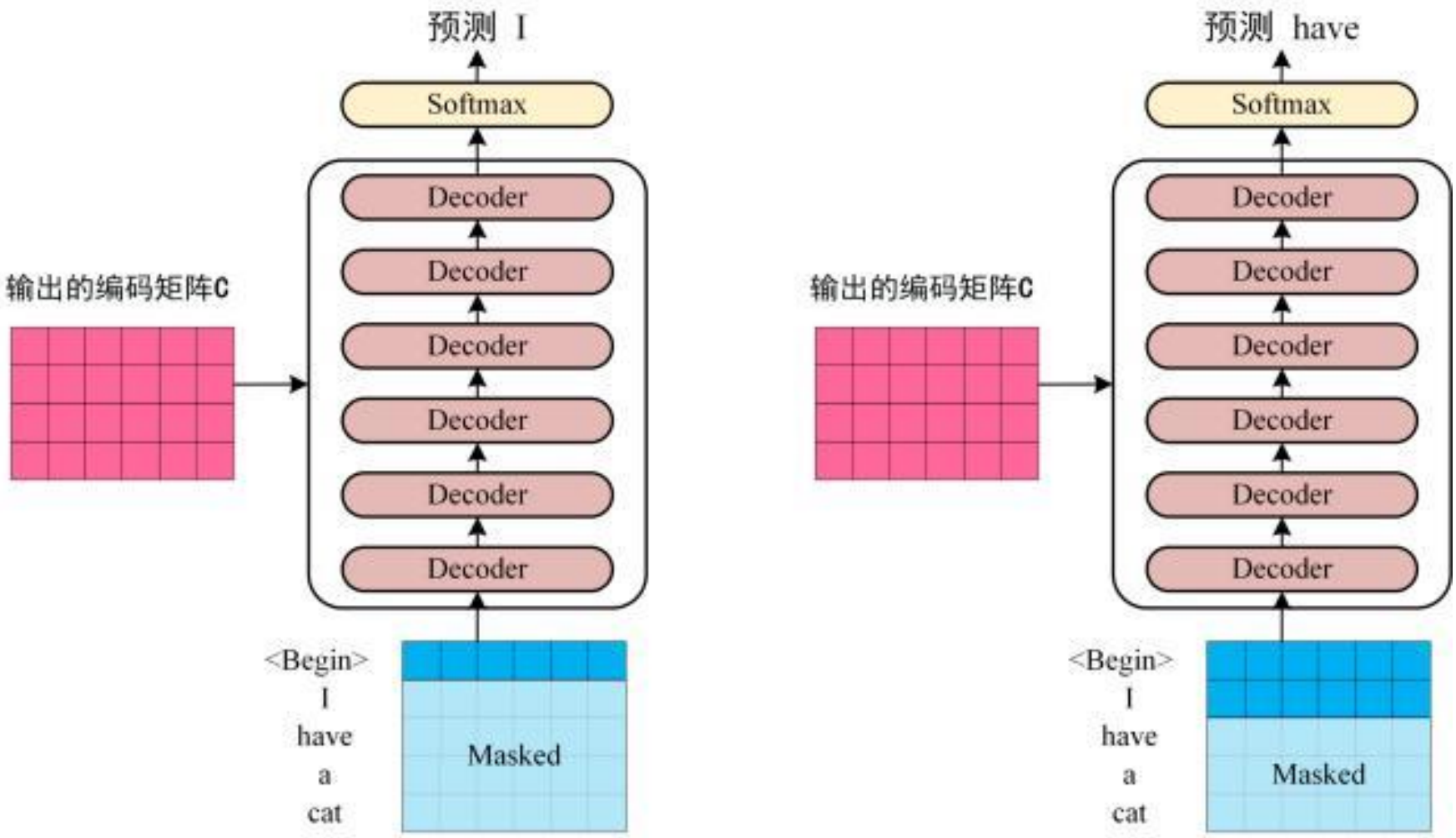
3. (宏观)工作流程



3. (宏观)工作流程

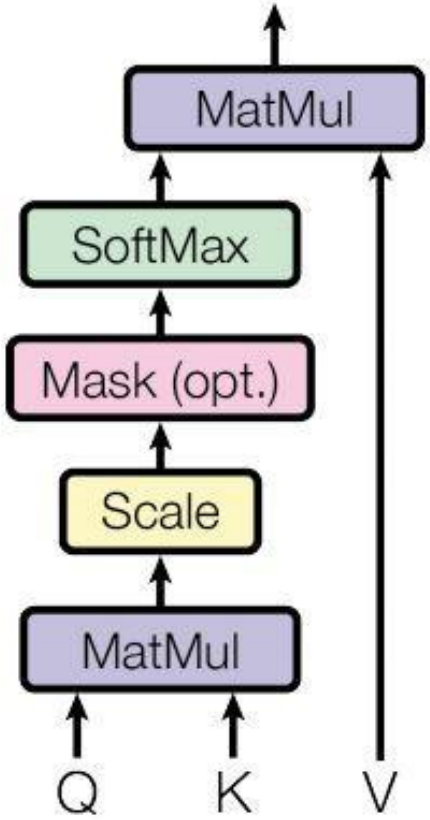

我
有
一只
猫
输出的编码矩阵C
Encoder
Encoder
Encoder
Encoder
Encoder
Encoder
我
有
一只
猫
输入的表示矩阵X

3. (宏观)工作流程


预测 I
Softmax
Decoder
Decoder
Decoder
Decoder
Decoder
Decoder
输出的编码矩阵C
<Begin>
I
have
a
cat
Masked
预测 have
Softmax
Decoder
Decoder
Decoder
Decoder
Decoder
Decoder
输出的编码矩阵C
<Begin>
I
have
a
cat
Masked

4. SELF-ATTENTION

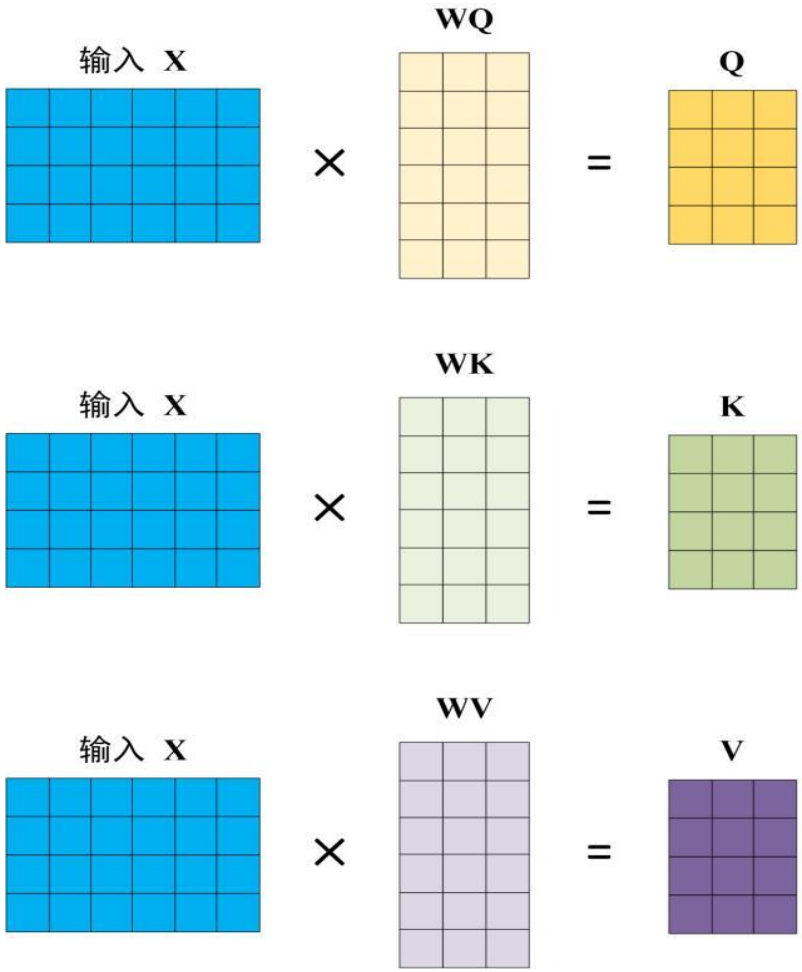
Scaled Dot-Product Attention



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

d_k是Q, K矩阵的列数，即向量维度

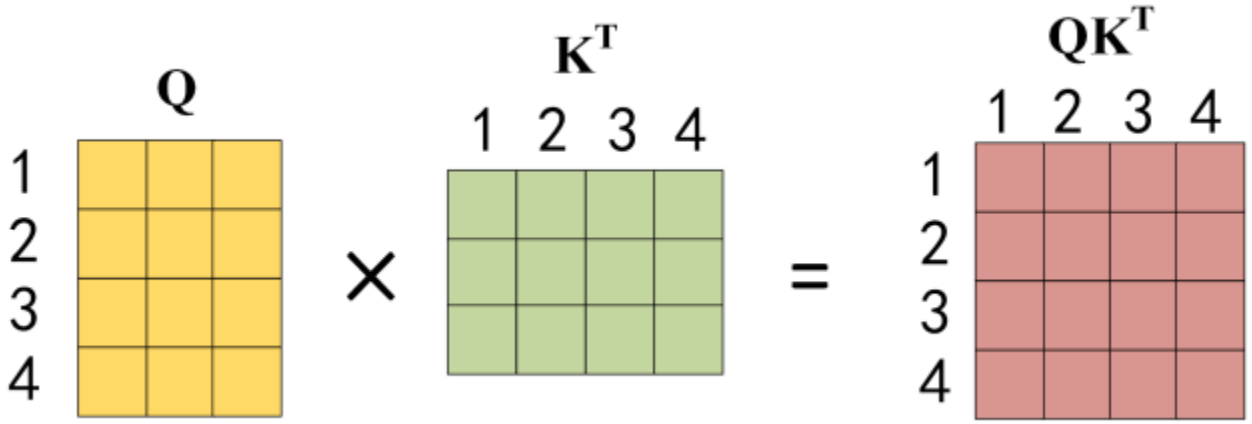
4. SELF-ATTENTION



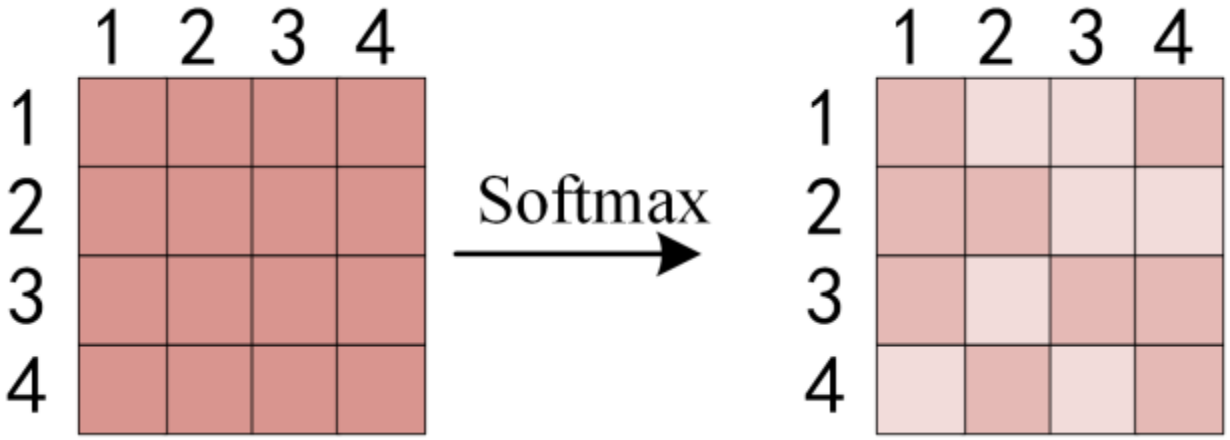
$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

d_k是Q,K矩阵的列数，即向量维度

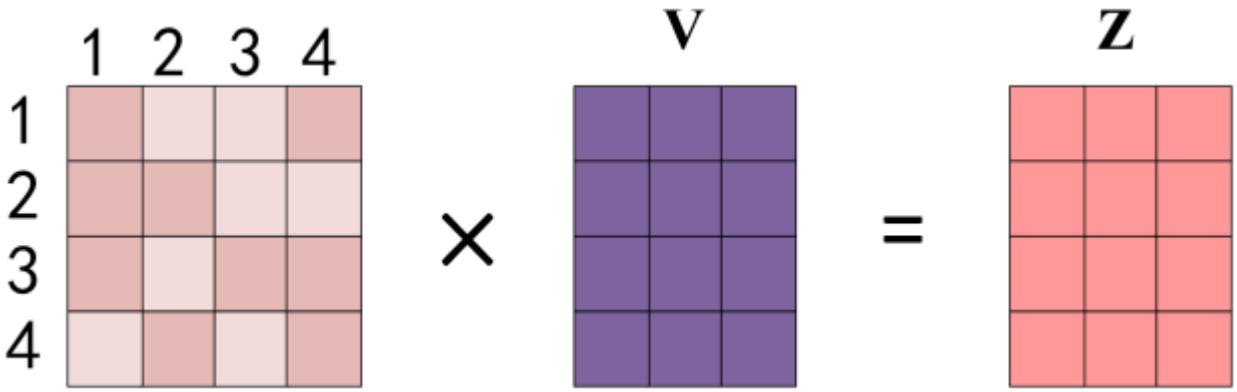
4. SELF-ATTENTION



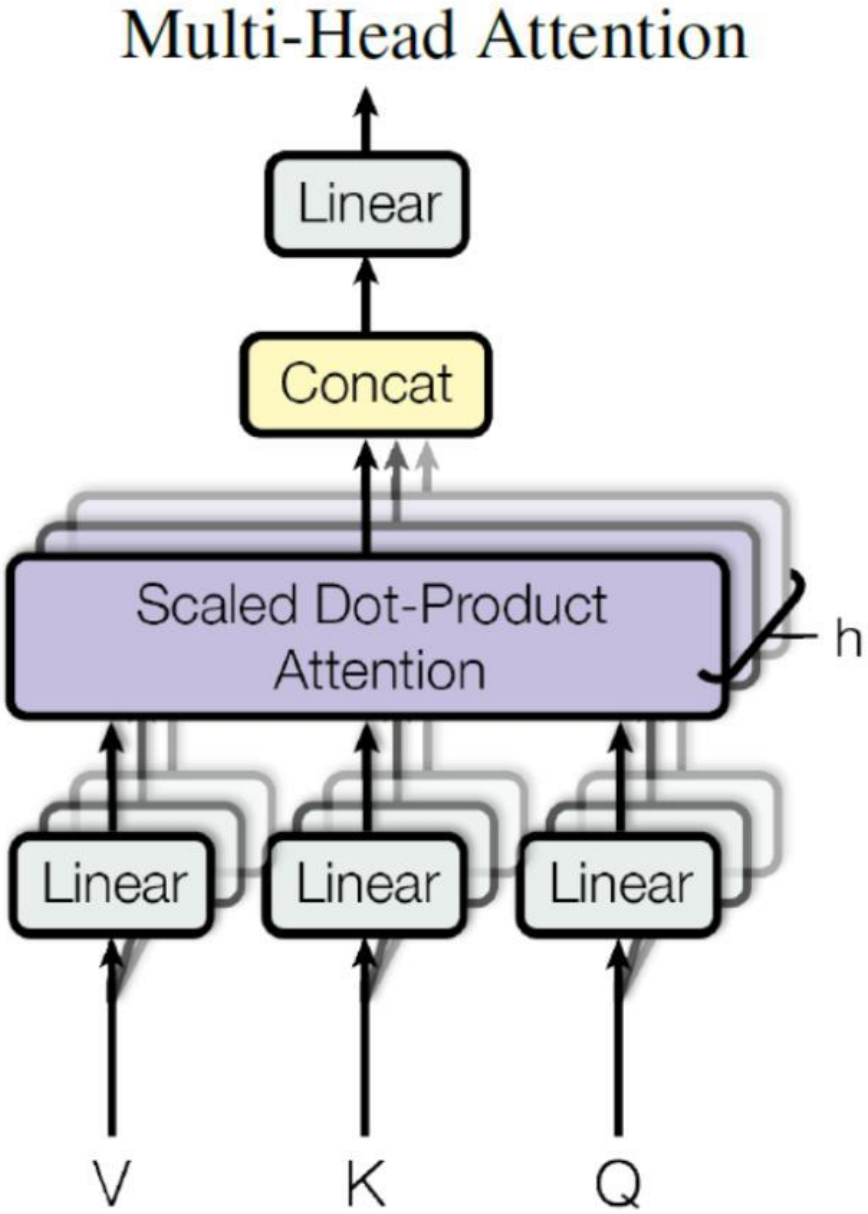
4. SELF-ATTENTION

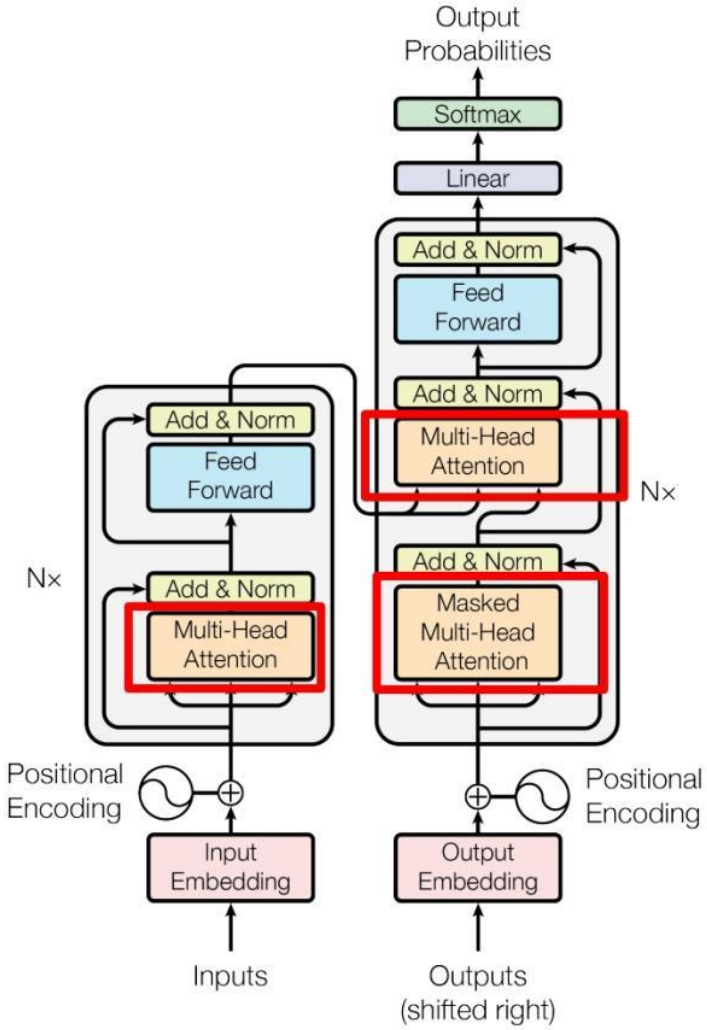


4. SELF-ATTENTION

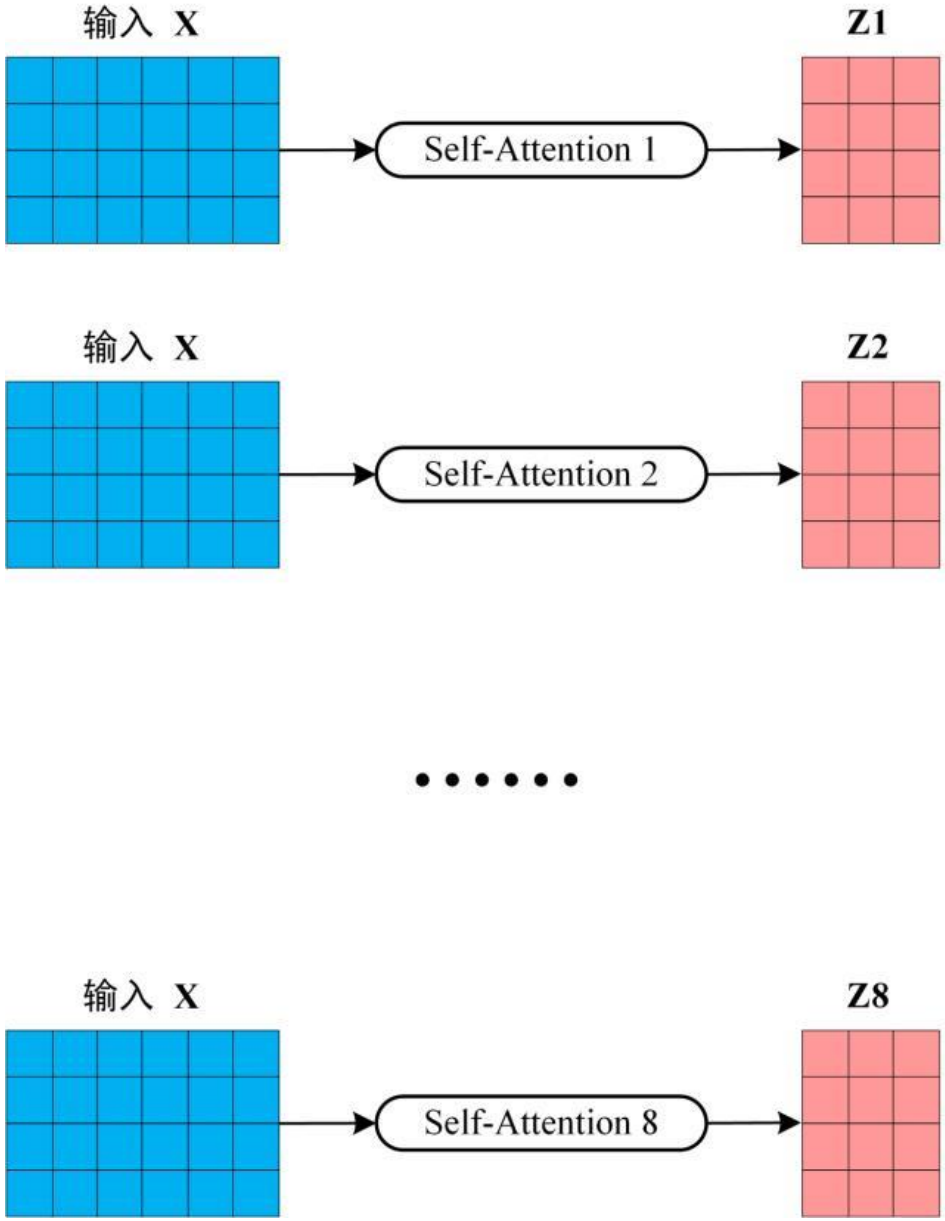


5. MULTI-HEAD ATTENTION

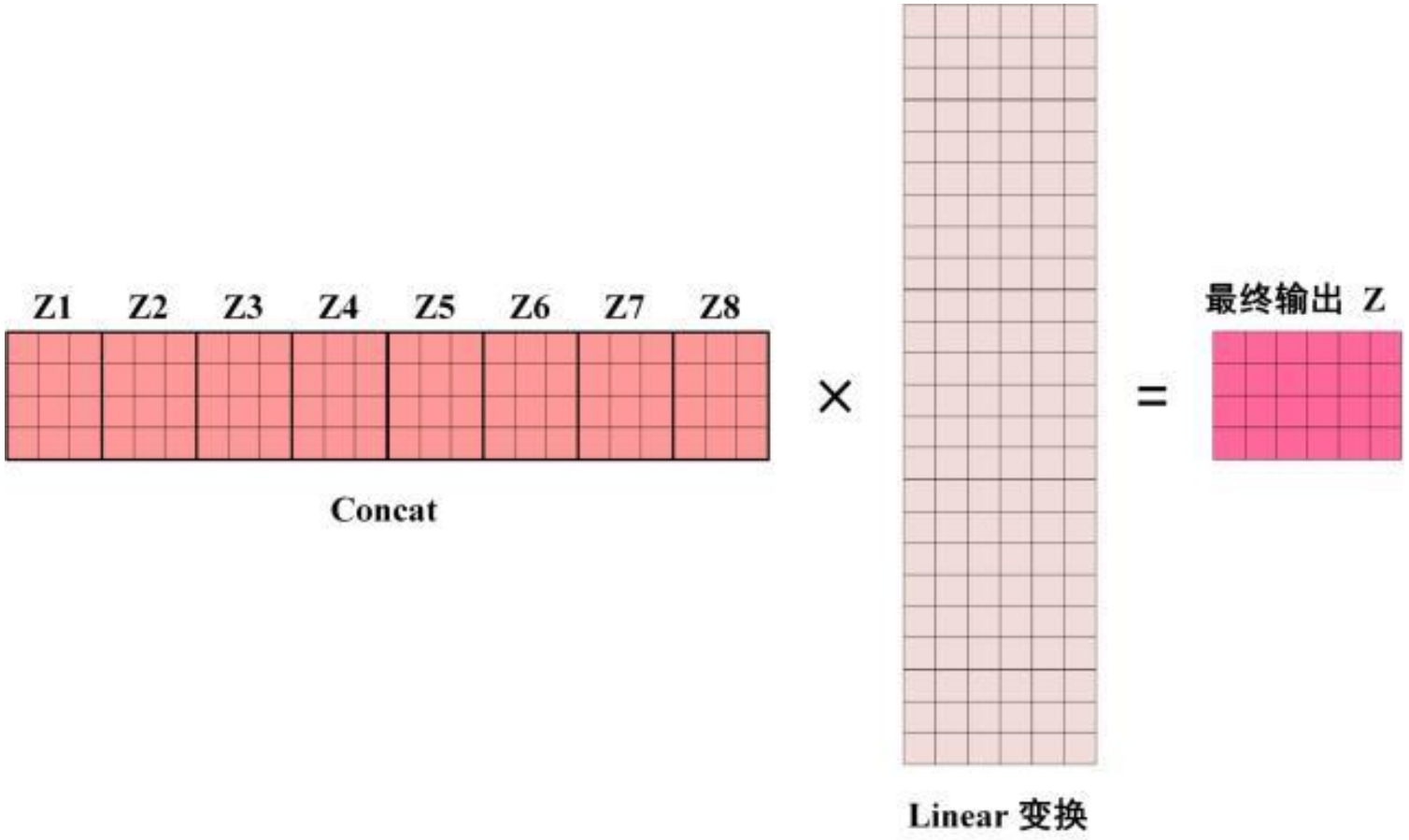




5. MULTI-HEAD ATTENTION



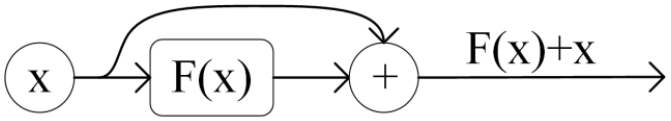
5. MULTI-HEAD ATTENTION

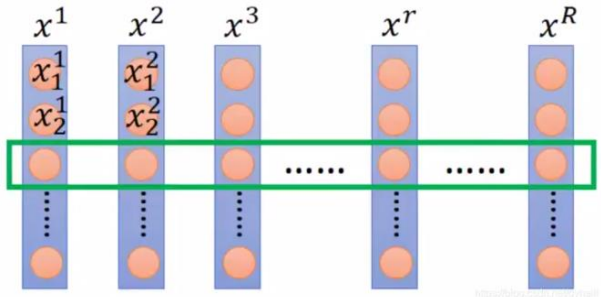


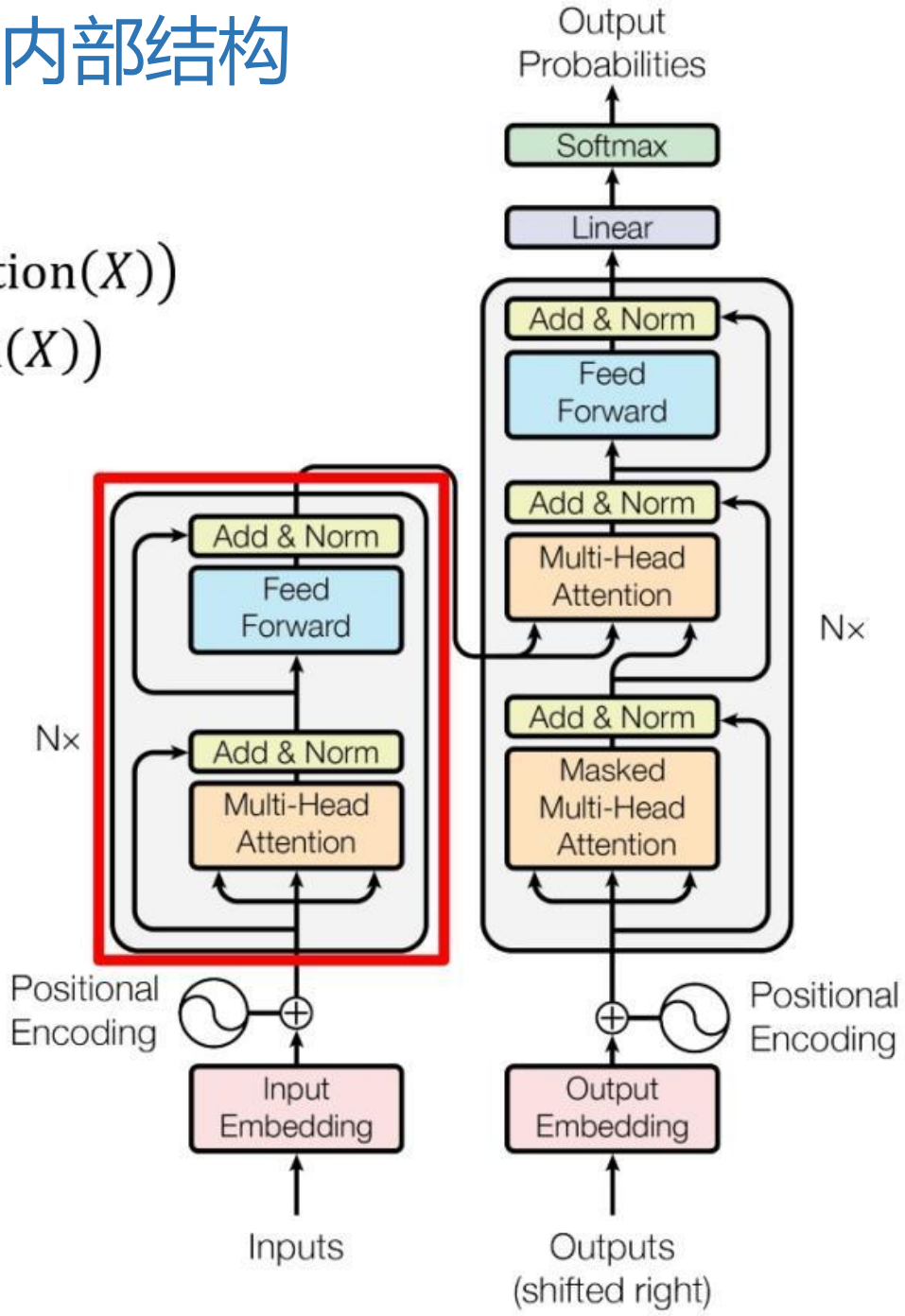
6. TRM里的ENCODER内部结构

$$\text{LayerNorm}\left(X + \text{MultiHeadAttention}(X)\right)$$

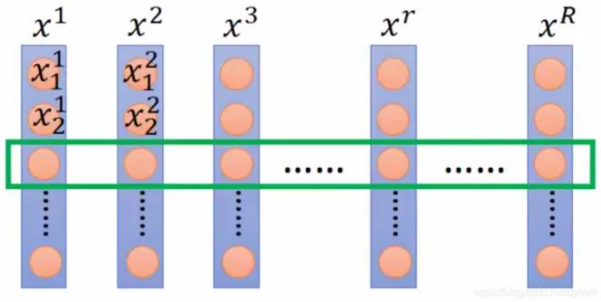
$$\text{LayerNorm}\left(X + \text{FeedForward}(X)\right)$$

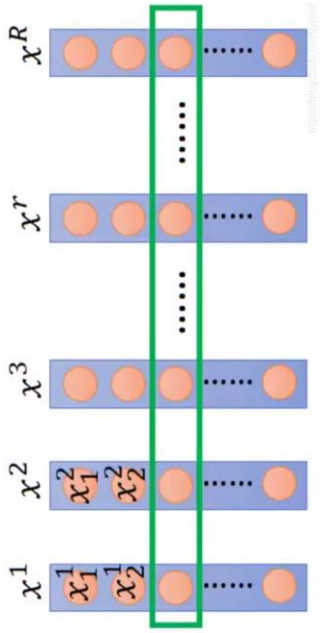


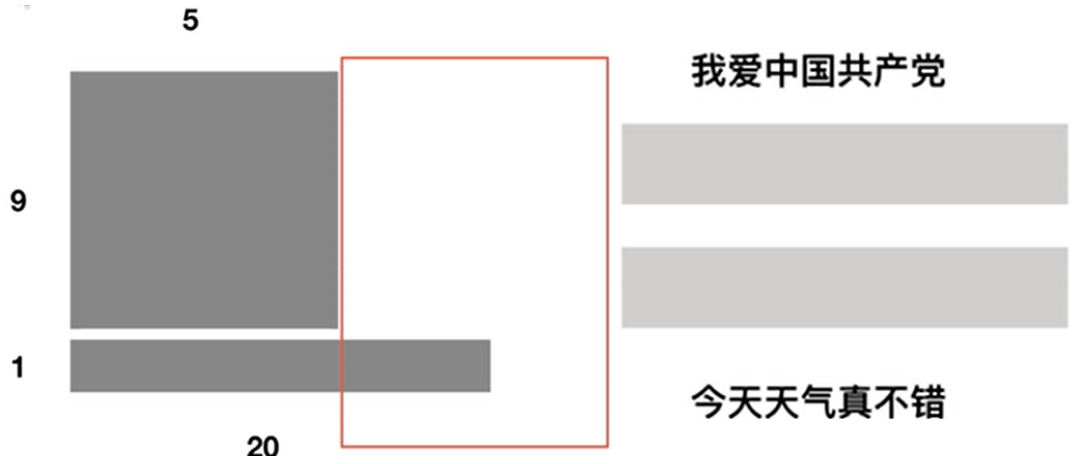




6. LAYER NORMALIZATION







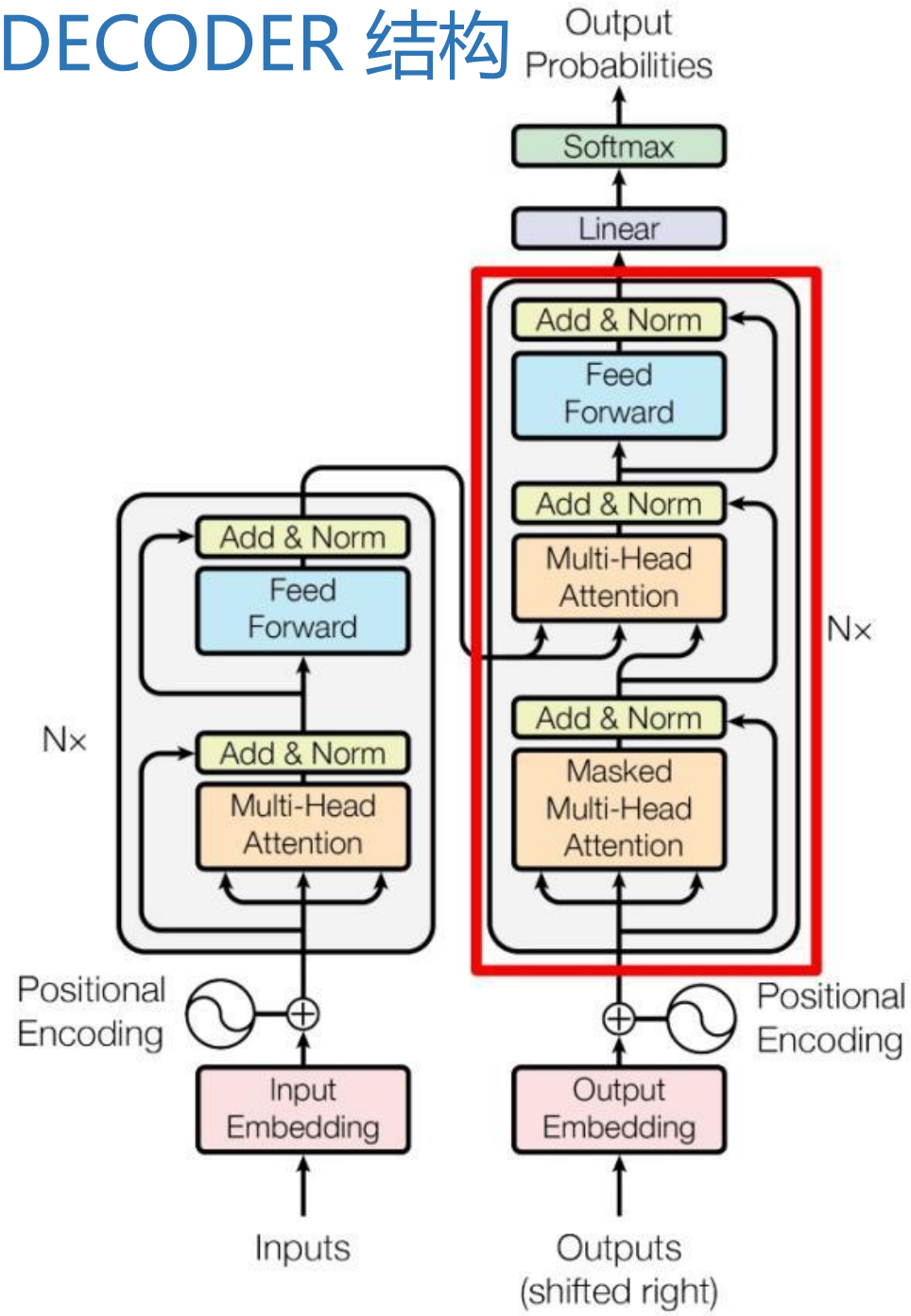
6. ENCODER里的 FEED FORWARD

两层：

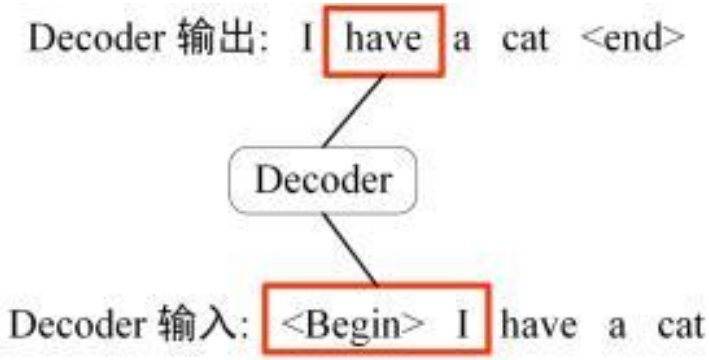
$$\max(0, XW_1 + b_1)W_2 + b_2$$

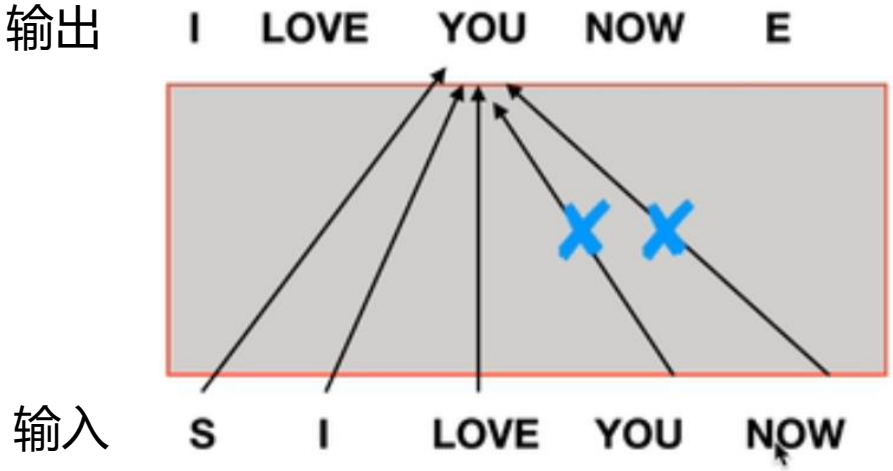
X是输入，Feed Forward 最终得到的输出矩阵的维度与 **X** 一致。

7. TRM里的DECODER 结构



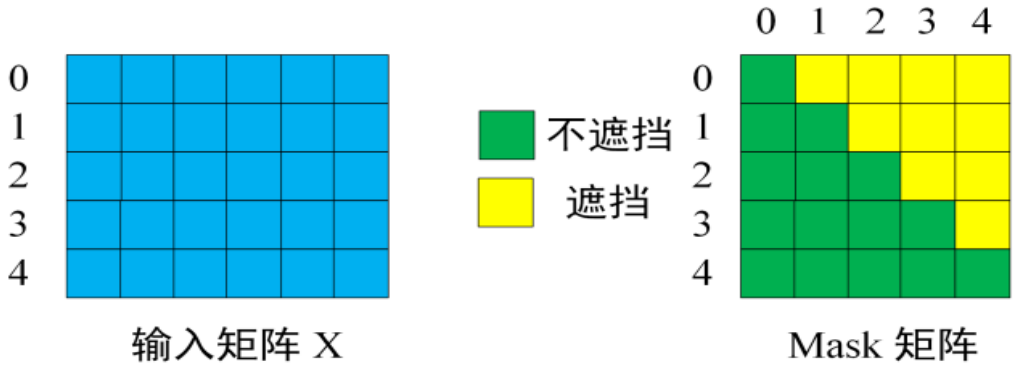
7. DECODER MASK操作



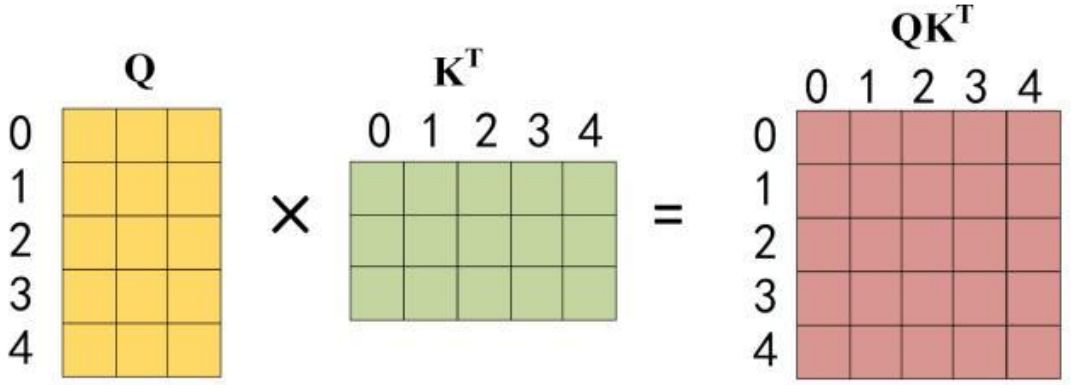


7. DECODER MASK操作

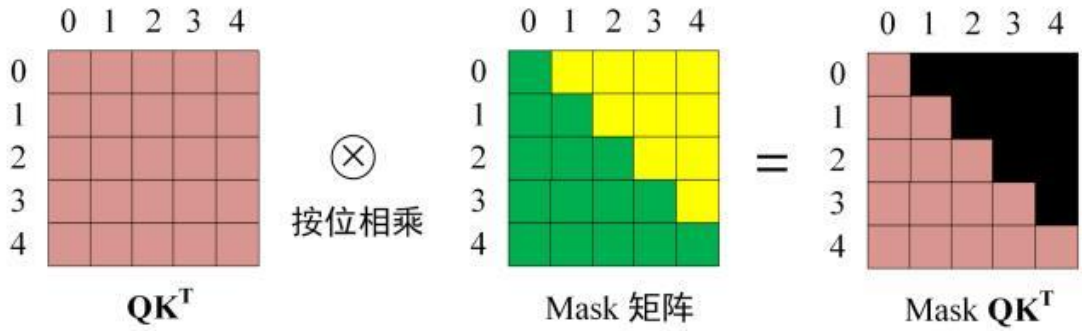
第一步



第二步

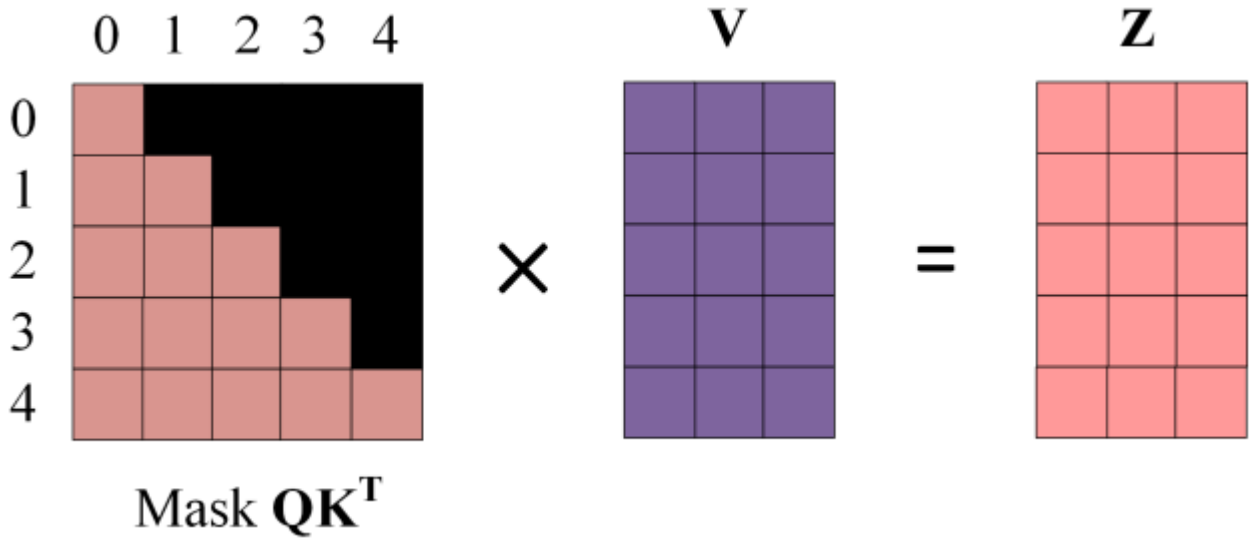


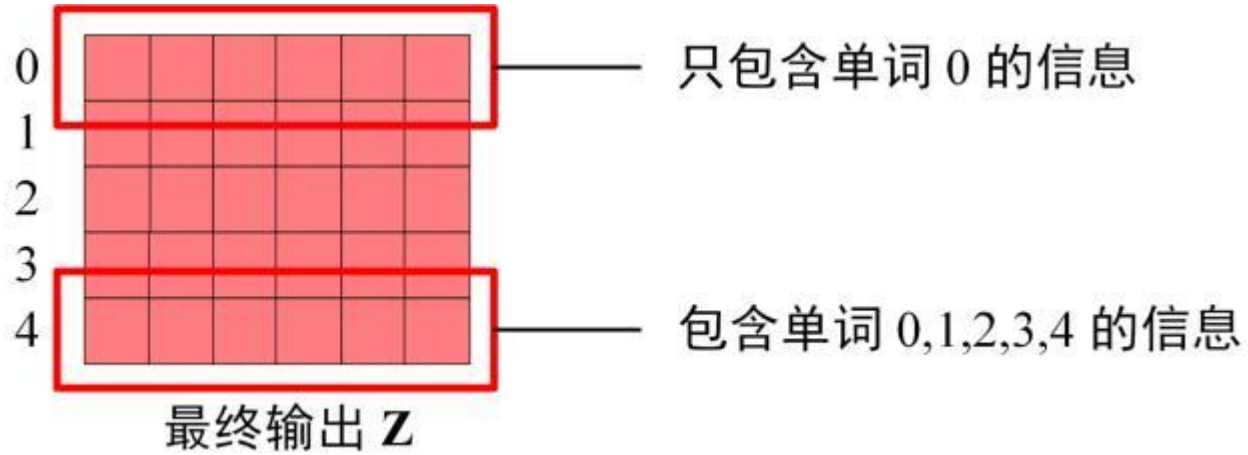
第三步



7. DECODER MASK操作

第四步





8. 位置编码 WHY

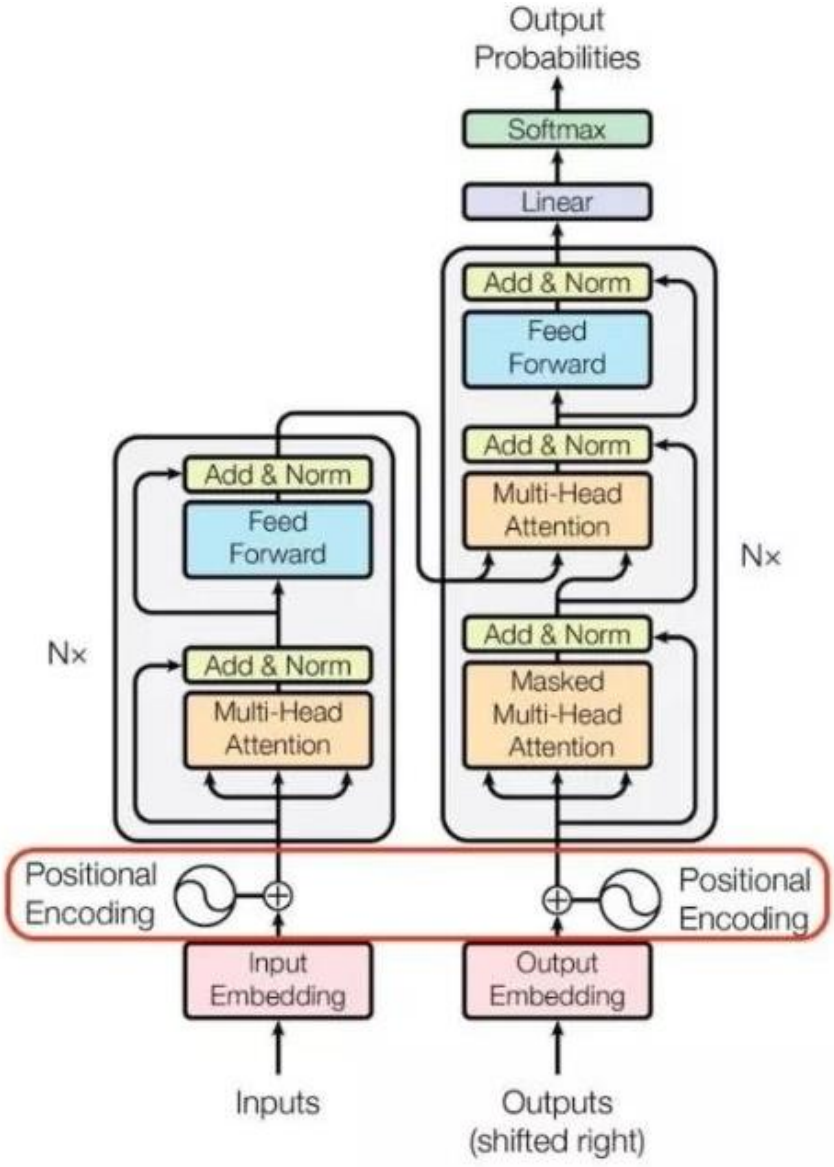
- 位置和顺序重要：

I **do not** like the story of the movie, but I **do** like the cast.
I **do** like the story of the movie, but I **do not** like the cast.

- Transformer模型并行训练，抛弃了RNN作为序列学习的基本模型。词序信息就会丢失，模型就没有办法知道每个词在句子中的相对和绝对的位置信息。

8. 位置编码 WHAT

模型原先的输入是不含词序信息的词向量，位置编码需要将词序信息和词向量结合起来形成一种新的表示输入给模型，这样模型就具备了学习词序信息的能力。



8. 位置编码 HOW

- 根据公式:

$$\mathrm{PE}(pos, 2i) = sin\left(\frac{pos}{10000^{2i/d_{model}}}\right),$$

$$\mathrm{PE}(pos, 2i+1) = cos\left(\frac{pos}{10000^{2i/d_{model}}}\right).$$

- 对于一个单词 w 位于 $pos \in [0, L-1]$, 假设使用 encoding 的维度为 4, 那么这个单词的 encoding 为:

$$\begin{aligned} e_w &= \left[\sin\left(\frac{\text{pos}}{10000^{\circ}}\right), \cos\left(\frac{\text{pos}}{10000^{\circ}}\right), \sin\left(\frac{\text{pos}}{10000^{2/4}}\right), \cos\left(\frac{\text{pos}}{10000^{2/4}}\right)\right] \\ &= \left[\sin(\text{pos}), \cos(\text{pos}), \sin\left(\frac{\text{pos}}{100}\right), \cos\left(\frac{\text{pos}}{100}\right)\right] \end{aligned}$$

比如20个单词，每个单词编码为512维度。
一共20行，那么每一行就表示一个词向量，包含512个值，每个值在-1到1之间。

8. DECODER的输入输出和执行步骤

整体右移一位（Shifted Right）

- 0-</s>【起始符】
- 1-“I”
- 2-“Love”
- 3-“China”

Time Step 1

- 初始输入：起始符</s> + Positional Encoding
- 中间输入：（我爱中国）Encoder Embedding
- 最终输出：产生预测“I”

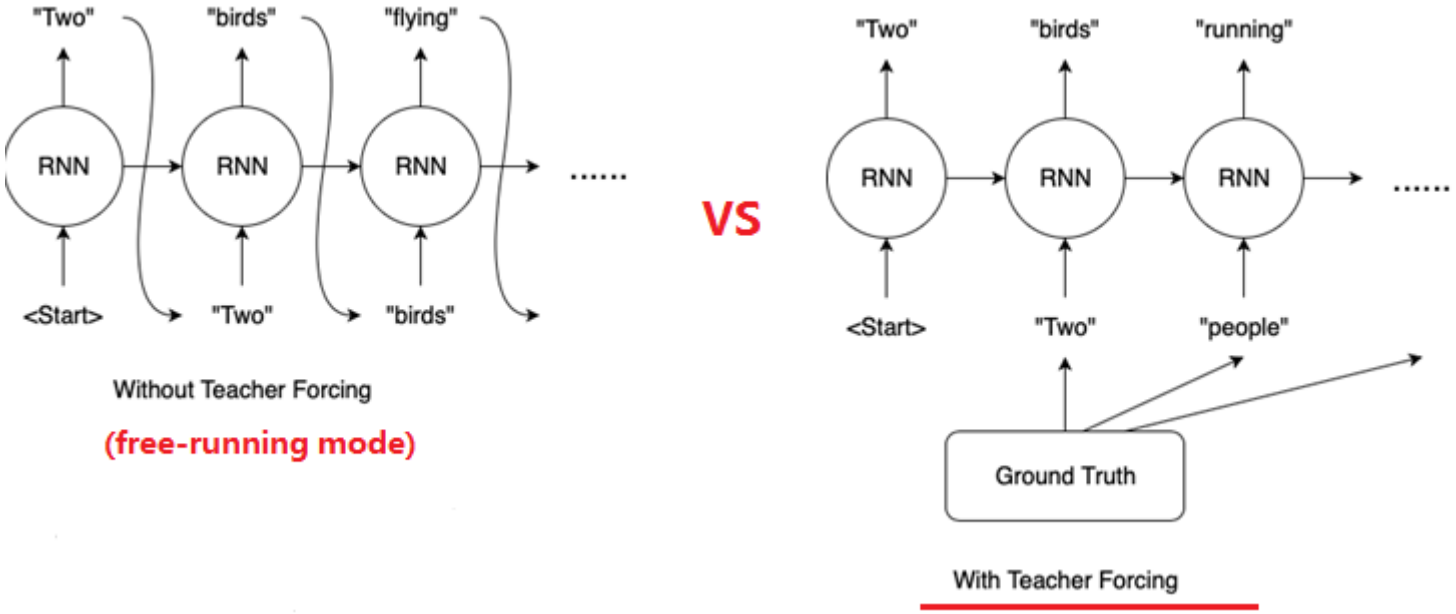
Time Step 2

- 初始输入：起始符</s> + “I”+ Positonal Encoding
- 中间输入：（我爱中国）Encoder Embedding
- 最终输出：产生预测“Love”

Time Step 3

- 初始输入：起始符</s> + “I”+ “Love”+ Positonal Encoding
- 中间输入：（我爱中国）Encoder Embedding
- 最终输出：产生预测“China”

8. TEACHER FORCING（RNN）



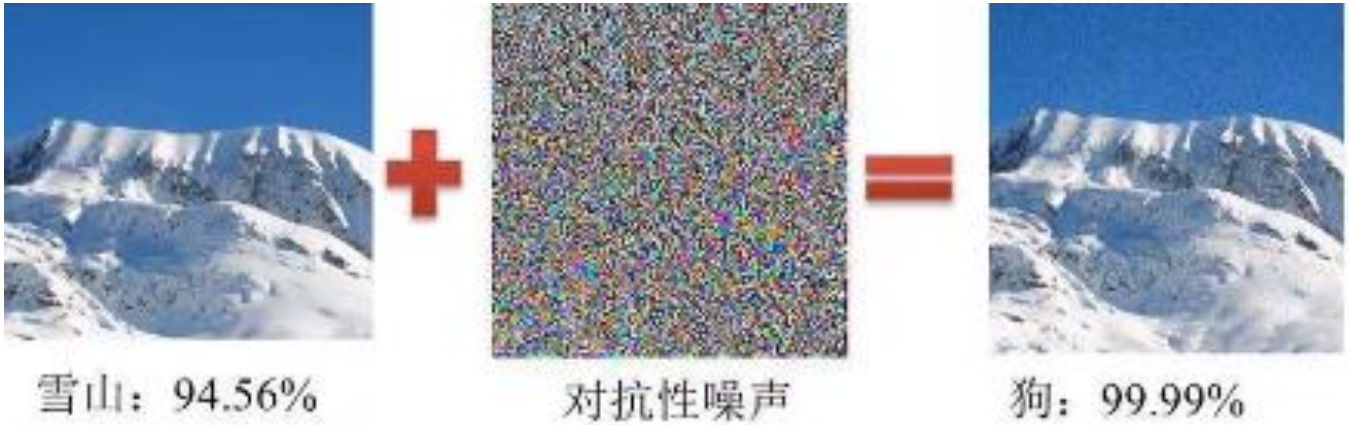
训练迭代过程早期的RNN预测能力非常弱，几乎不能给出好的生成结果。如果某一个unit产生了垃圾结果，必然会影响后面一片unit的学习。错误结果会导致后续的学习都受到不好的影响，导致学习速度变慢，难以收敛。

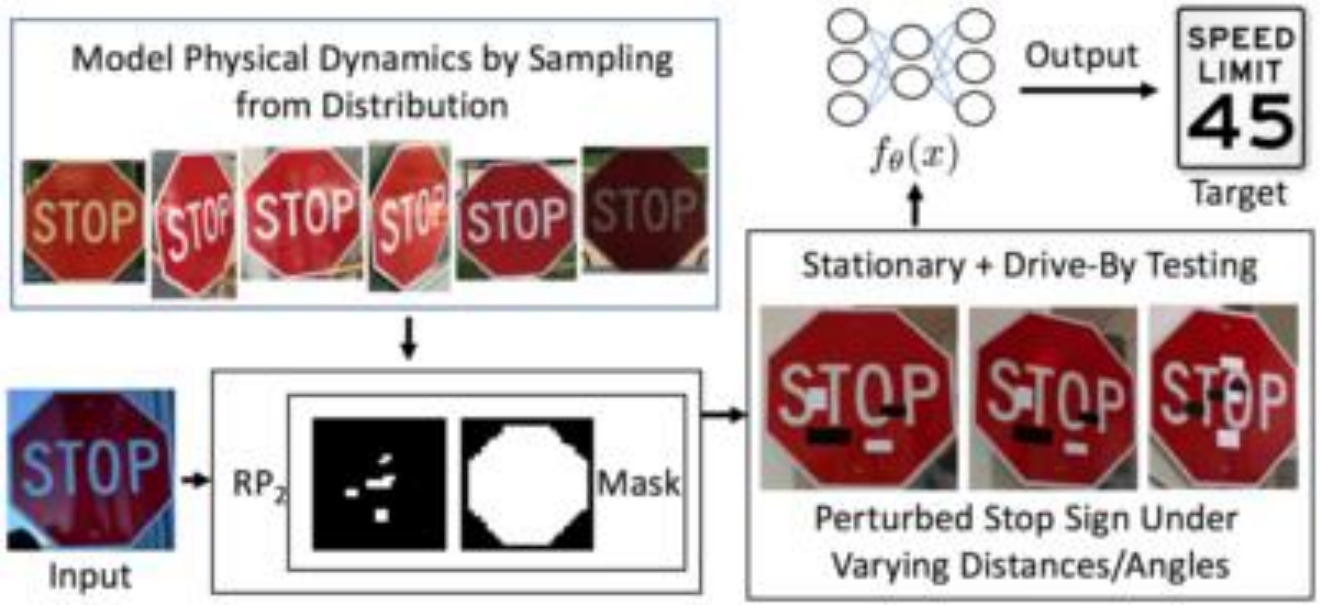
teacher-forcing过于依赖ground truth数据，在训练过程中，模型会有较好的效果，但是在测试的时候因为不能得到ground truth的支持，所以如果目前生成的序列在训练过程中有很大不同，模型就会变得脆弱。换言之，这种模型的cross-domain能力会更差，即如果测试数据集与训练数据集来自不同的领域，模型的performance就会变差。

TRANSFORMER用于目标检测

DETR AND VIT

课题


雪山：94.56%
对抗性噪声
狗：99.99%


Model Physical Dynamics by Sampling from Distribution
STOP
Input
RP2
Mask
$f_\theta(x)$
Output
SPEED LIMIT 45
Target
Stationary + Drive-By Testing
Perturbed Stop Sign Under Varying Distances/Angles

课题

工作

1. 参与AI Security（人工智能安全）方向的研究项目；
2. 协助团队进行模型与算法测试并进行优化和改进提升，参与进行中的Working Paper工作；
3. 撰写技术开发文档和文档管理；
4. 在完成上述工作的情况下，发现新的科学问题可以撰写新的研究论文。

要求

1. 具备一定的机器学习与深度学习基础；
2. 熟练掌握一种或多种编程语言，如Python；
3. 能够独立工作，同时也是优秀的团队合作者。具备较强的沟通能力。