循环神经网络

概览

1. 简介

2. 循环神经网络

3. LSTM及其变种

4. 双向循环神经网络

5. 深度循环神经网络

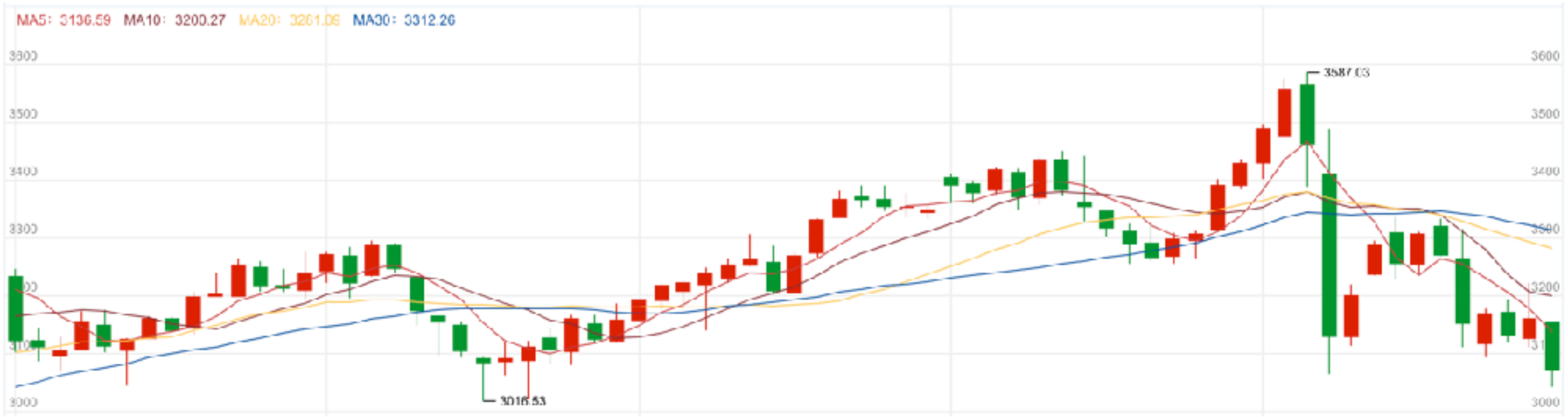
6. 循环神经网络的应用实例

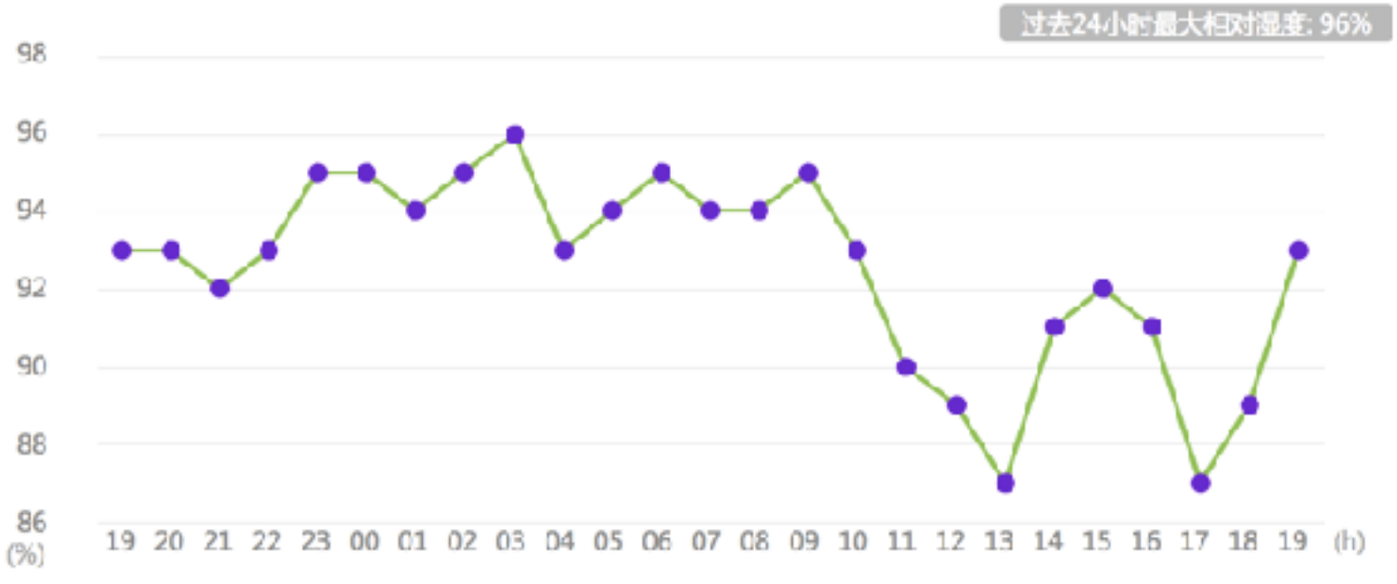
1. 简介

循环神经网络

循环神经网络（Recurrent Neural Network，RNN）是一类可以处理时间序列问题的人工神经网络。RNN与前馈神经网络不同，可利用其“记忆”功能处理序列间的问题。RNN可用于语音识别、构建语言模型等。

时间序列


MA5: 3136.59
MA10: 3200.27
MA20: 3261.06
MA30: 3312.26
3600
3500
3400
3300
3200
3000
3587.03
3016.53


过去24小时最大相对湿度: 96%
98
96
94
92
90
88
86
(%)
19 20 21 22 23 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 (h)


中国移动
11:51
92%
“Siri我不想和你玩儿了”
轻点以编辑
那么好吧。

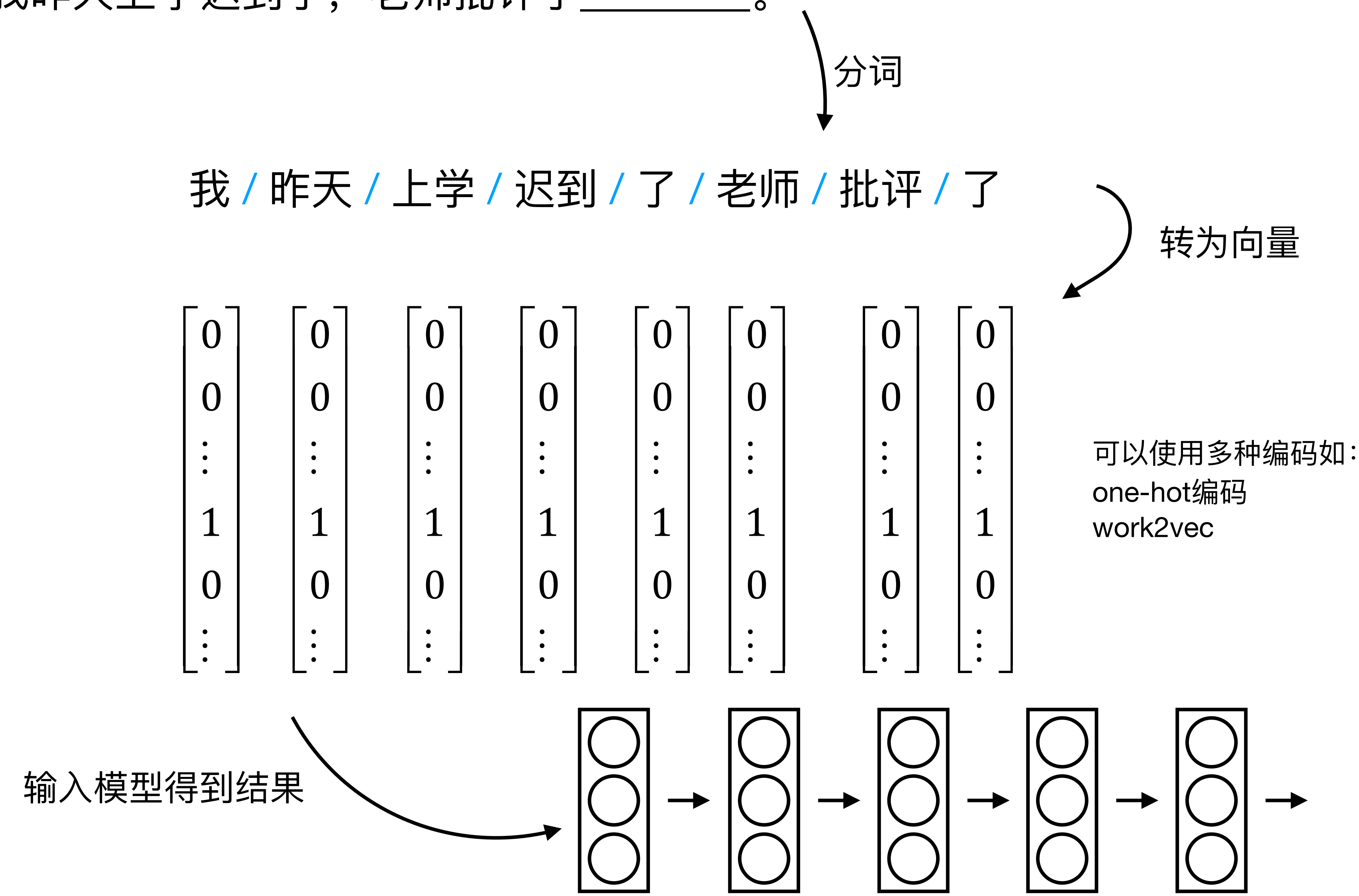
RNN用途

1. 语音识别：将音频转化为文本，例如Siri。
2. 机器翻译：将一种语言翻译为另一种语言，例如Google翻译。
3. DNA序列分析：分析DNA序列得出DNA功能。
4. 艺术作品生成：生成诗歌、音乐等。
5. 视频动作识别：根据视频，判断视频中对象的动作。
6. 关键词识别：识别句子中的关键词语。
7. 其它。

2. 循环神经网络

使用前馈网络预测缺失词

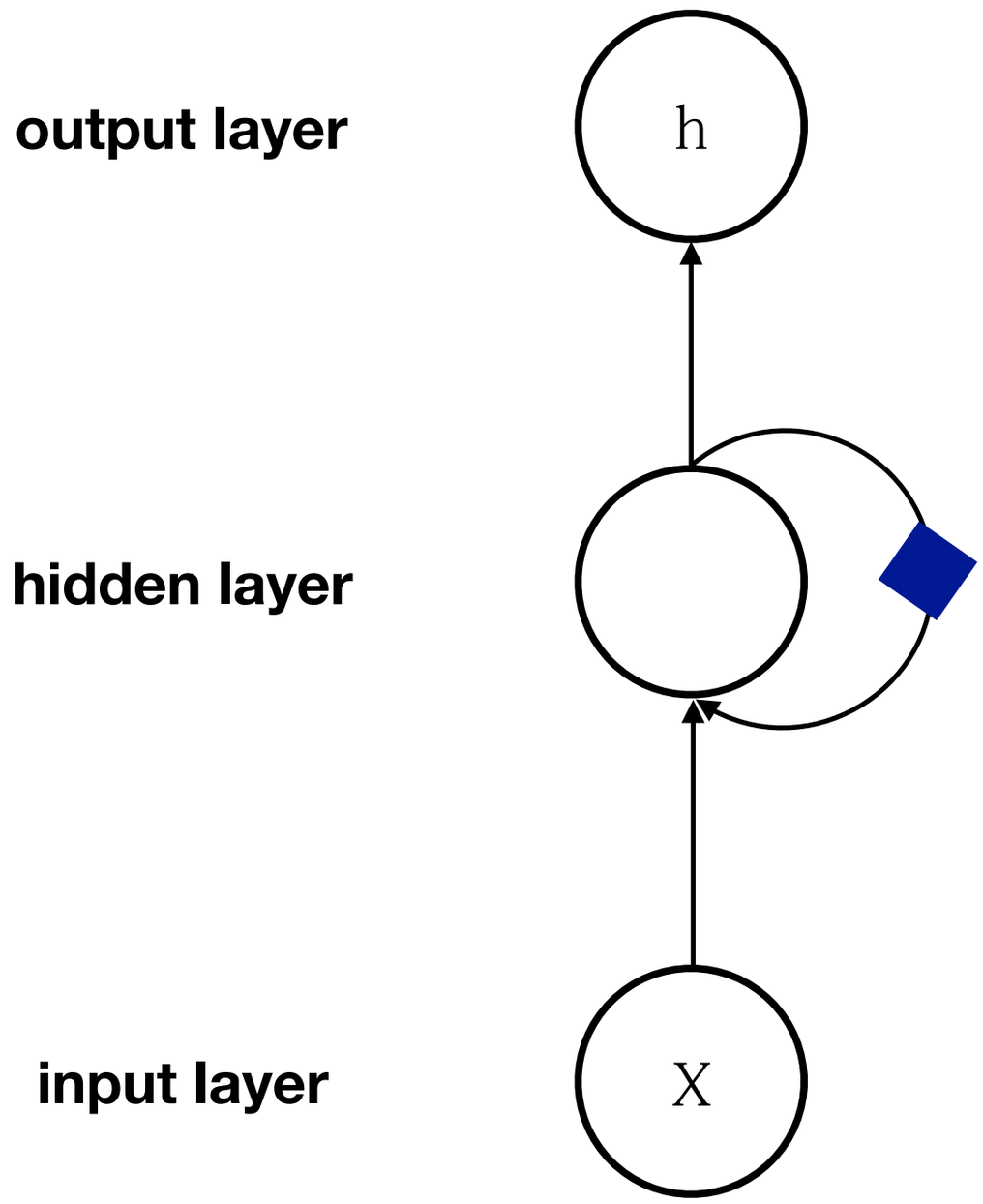
句子：我昨天上学迟到了，老师批评了________。



使用前馈网络的问题

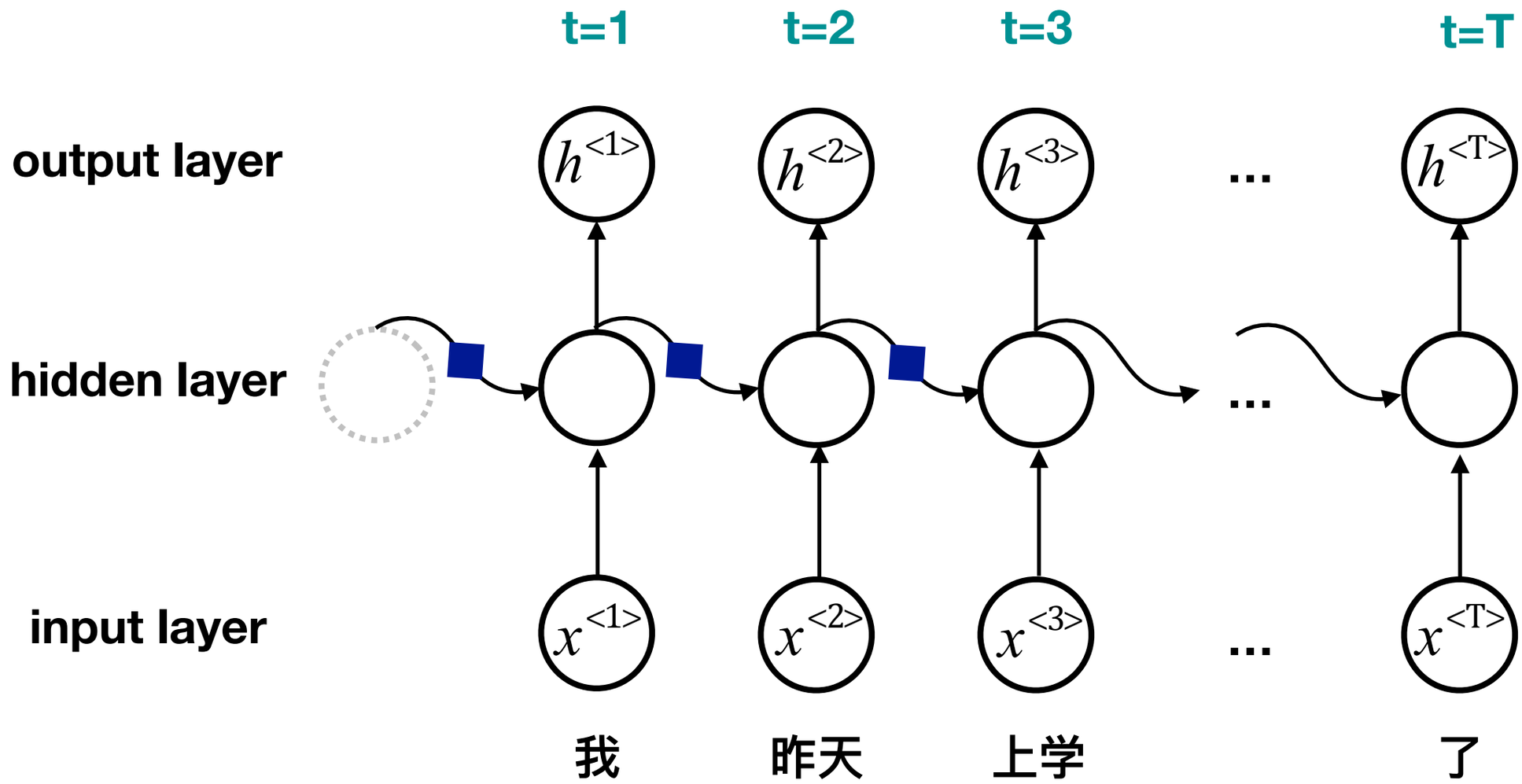
- 数据序列长度不固定，难以输入前馈网络；
- 可以通过边界处理解决上述问题，但有弊端。
- 模型无法共享从文本不同位置学习到的特征。

RNN

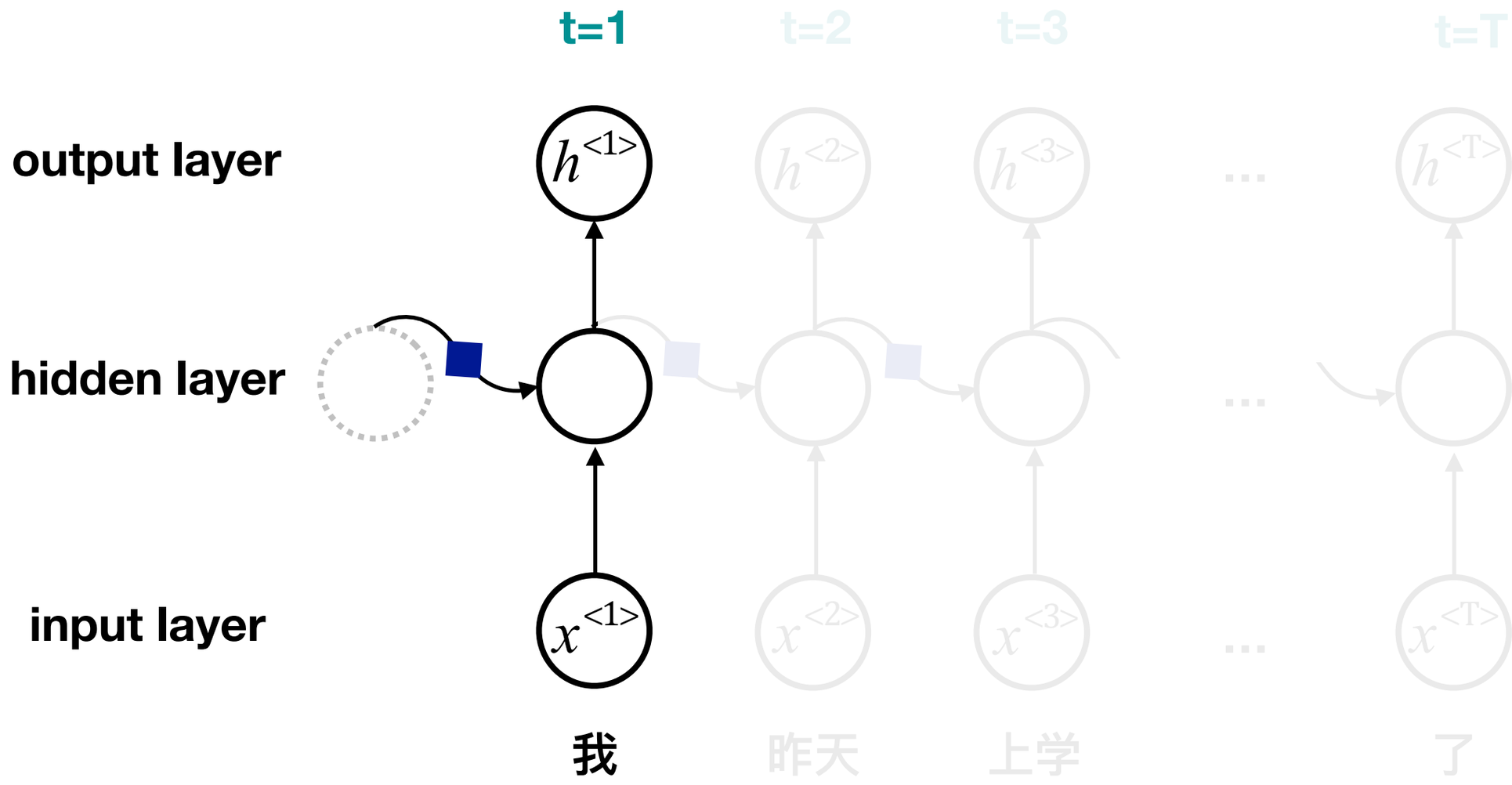


一个三层的RNN模型。其中隐藏层的输出同时作为输出层的输入与下一个时间序列的部分输入。

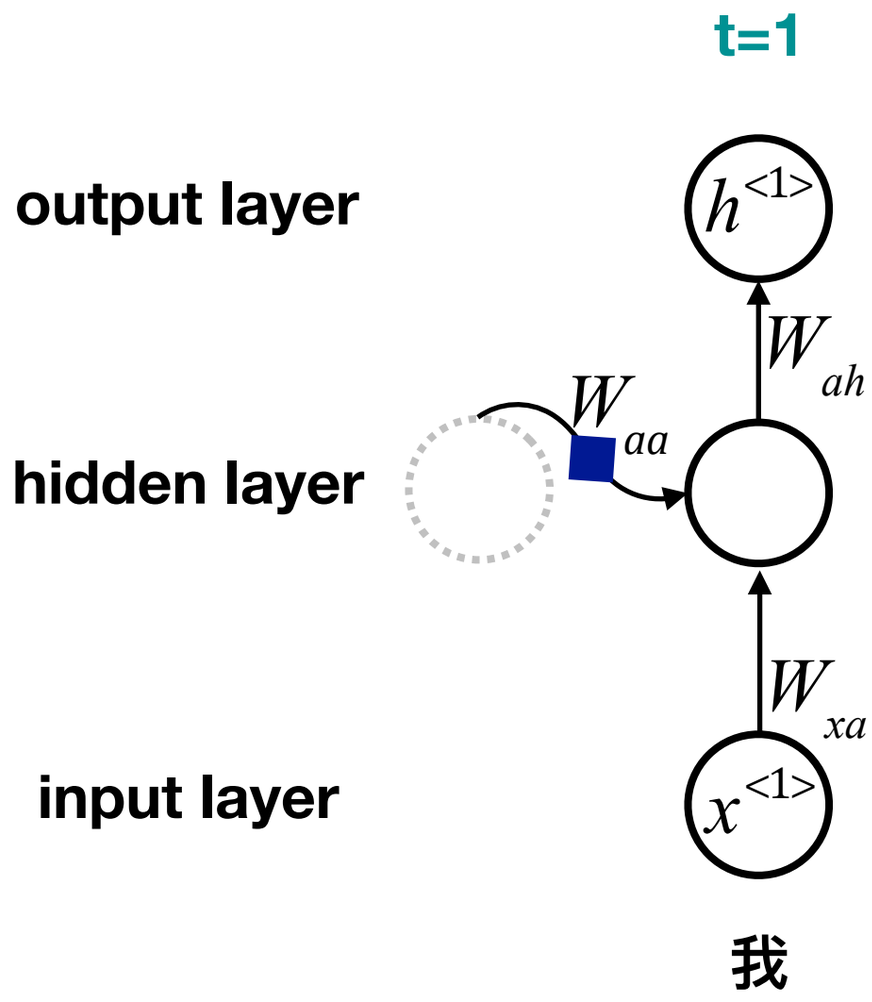
RNN按照时间序列展开



RNN运算过程


t=1
t=2
t=3
t=T
output layer
hidden layer
input layer
$h^{<1>}$
$h^{<2>}$
$h^{<3>}$
...
$h^{<T>}$
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
...
$x^{<T>}$
我
昨天
上学
了

RNN运算过程



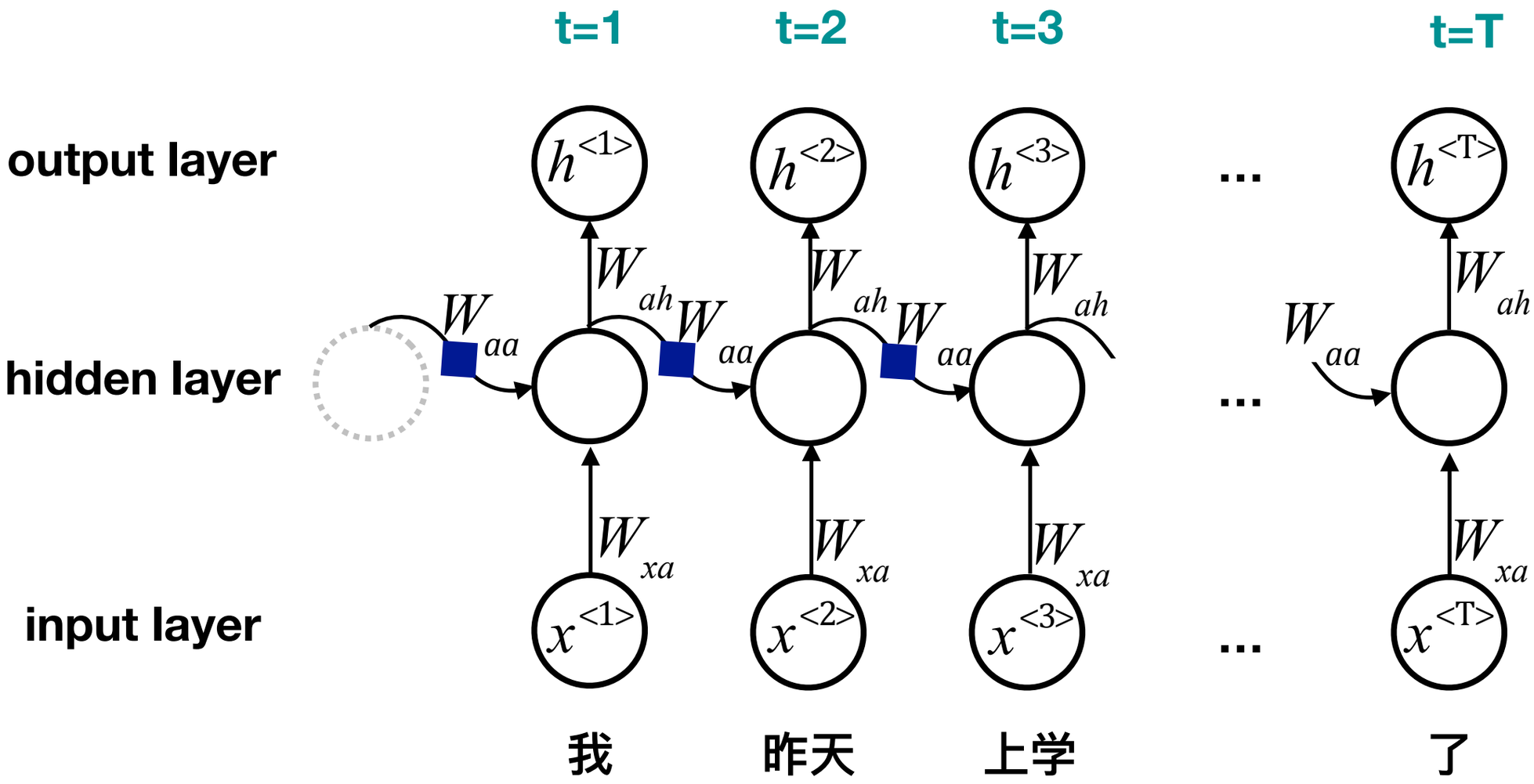
每个箭头曲线（包括直线）都代表一个全连接结构，可以看到：

1) 输入层与隐藏层之间全连接；

2) 隐藏层自身全连接（在相连两个序列上）；

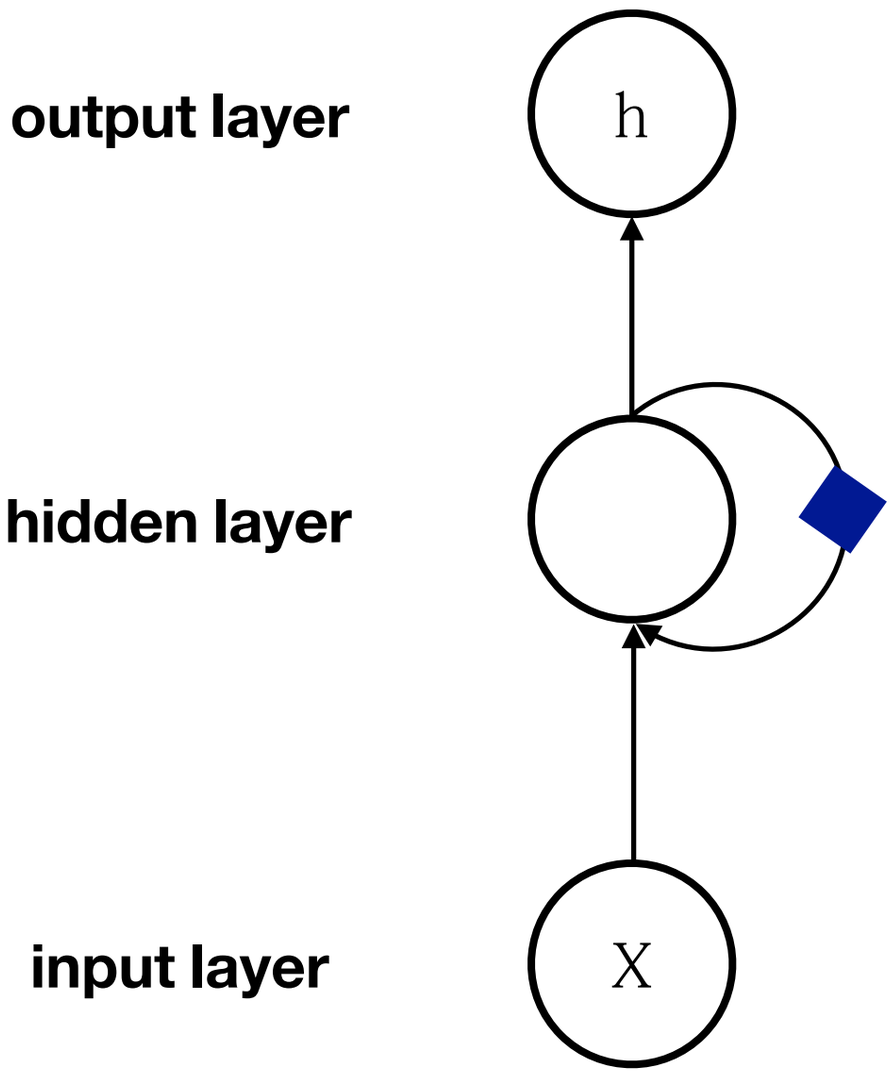
3) 隐藏层与输出层全连接。

RNN运算过程



可以看到：前一个时间序列的隐藏层输出给后一个序列的隐藏层，模型在时间序列上共享参数。

RNN运算过程



隐藏层：

$$a^{<0>} = \mathbf{0}$$

一般的为Tanh / ReLU激活函数

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$

输出层：

根据任务类型确定激活函数

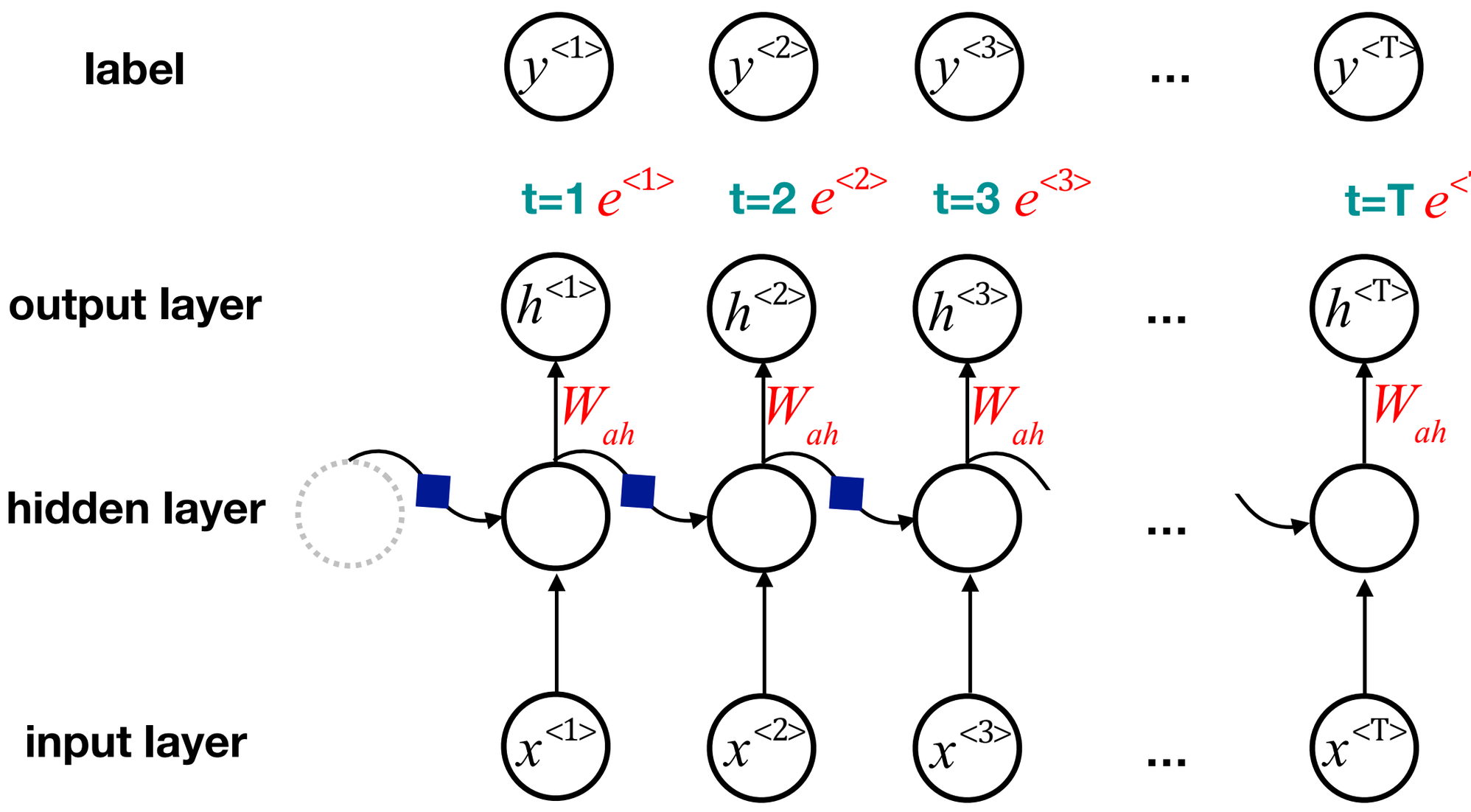
$$h^{<t>} = g(W_{ah}a^{<t>} + b_h)$$

2.1 基于时间的反向传播算法

基于时间的反向传播

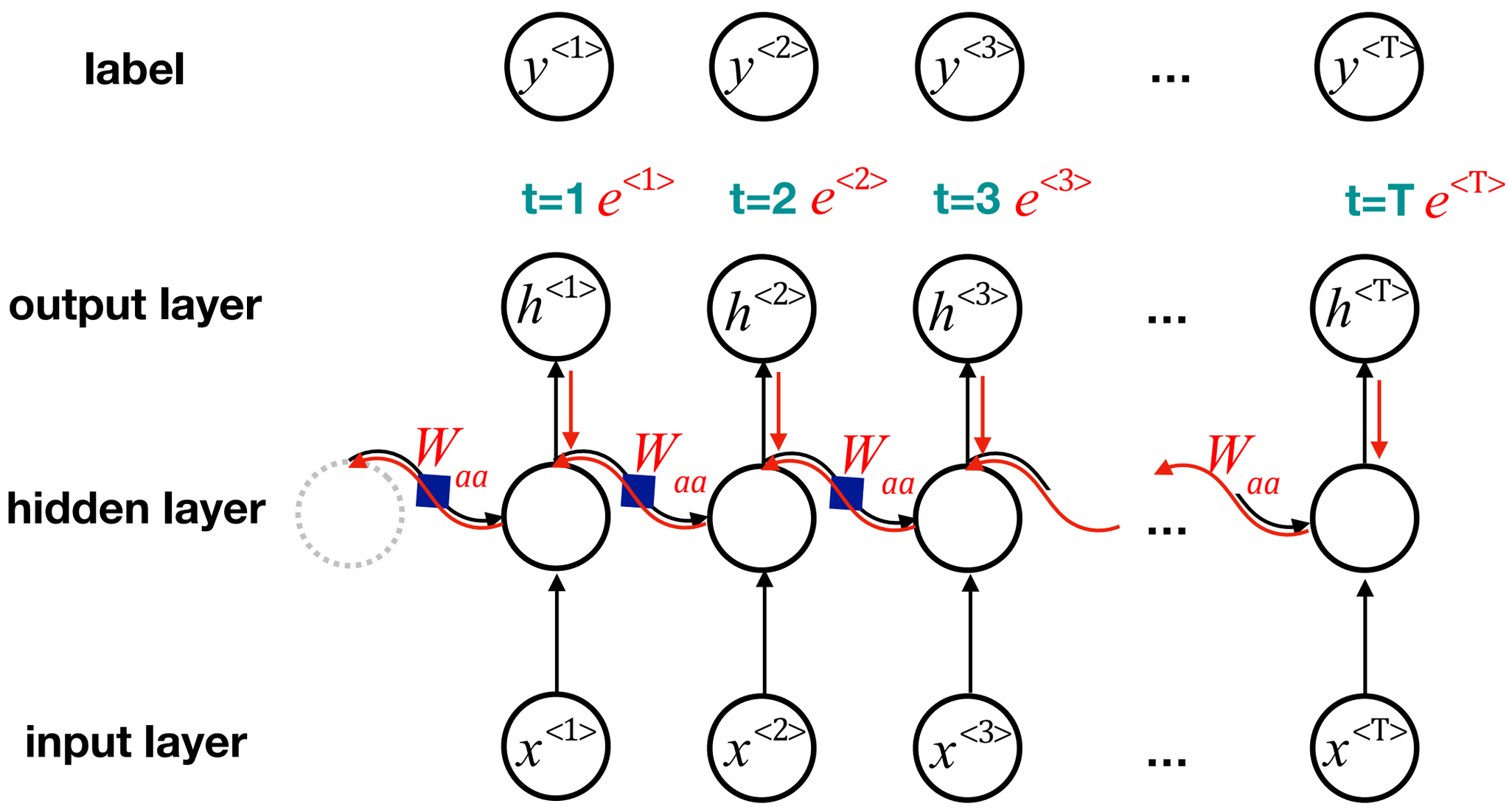
基于时间的反向传播（Back Propagation Through Time, BPTT）算法，是RNN系列模型的训练方法。其本质上就是反向传播算法，与前馈网络不同的是，其反向传播时的“通路”多了一条时间序列通路。

基于时间的反向传播



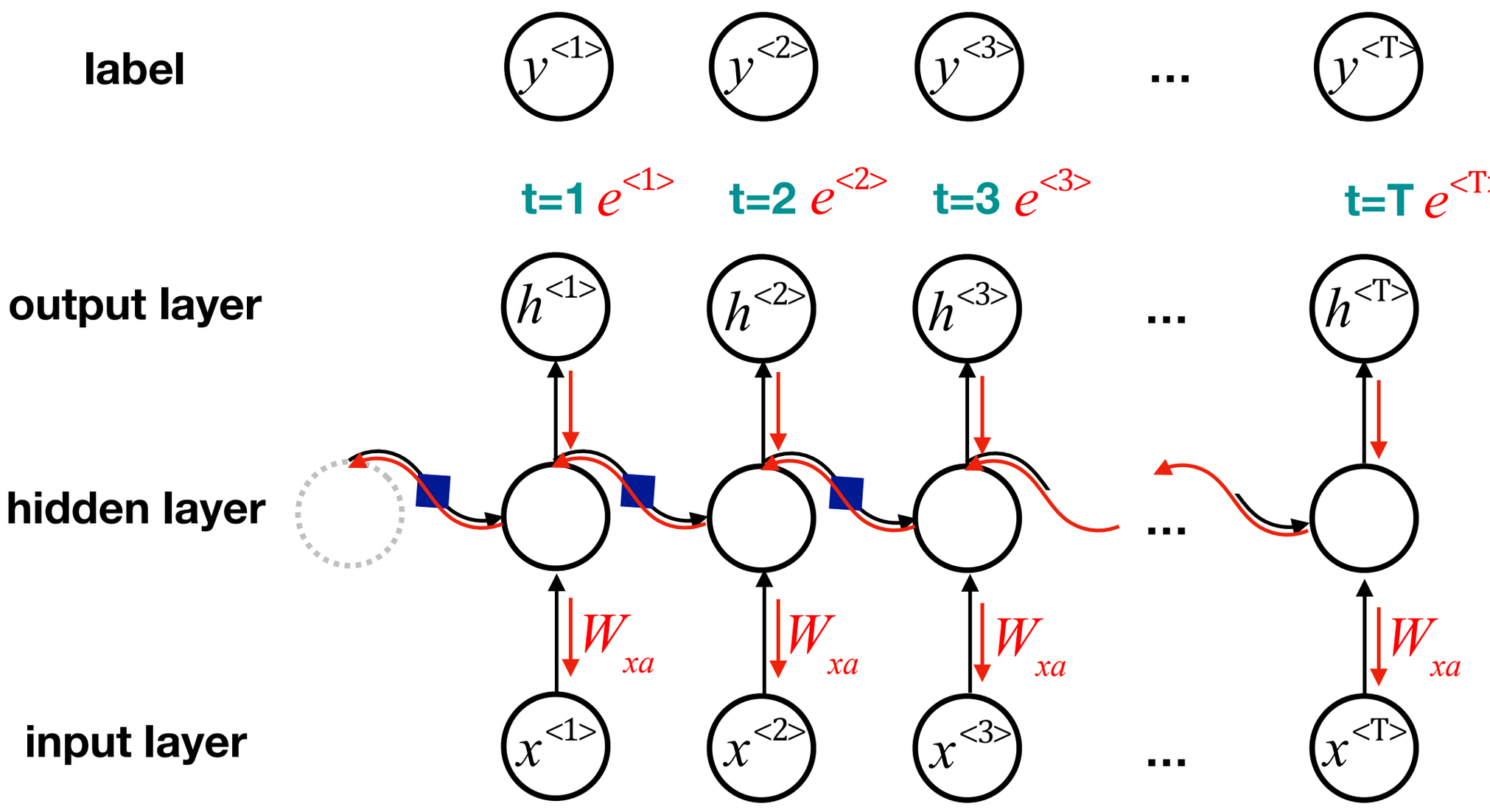
可以看到：对 W_{ah} 的误差来自于每一个时间步的误差之和。

基于时间的反向传播



可以看到：对 W_{aa} 的误差来自于每一个时间步与之后时间步的误差之和。

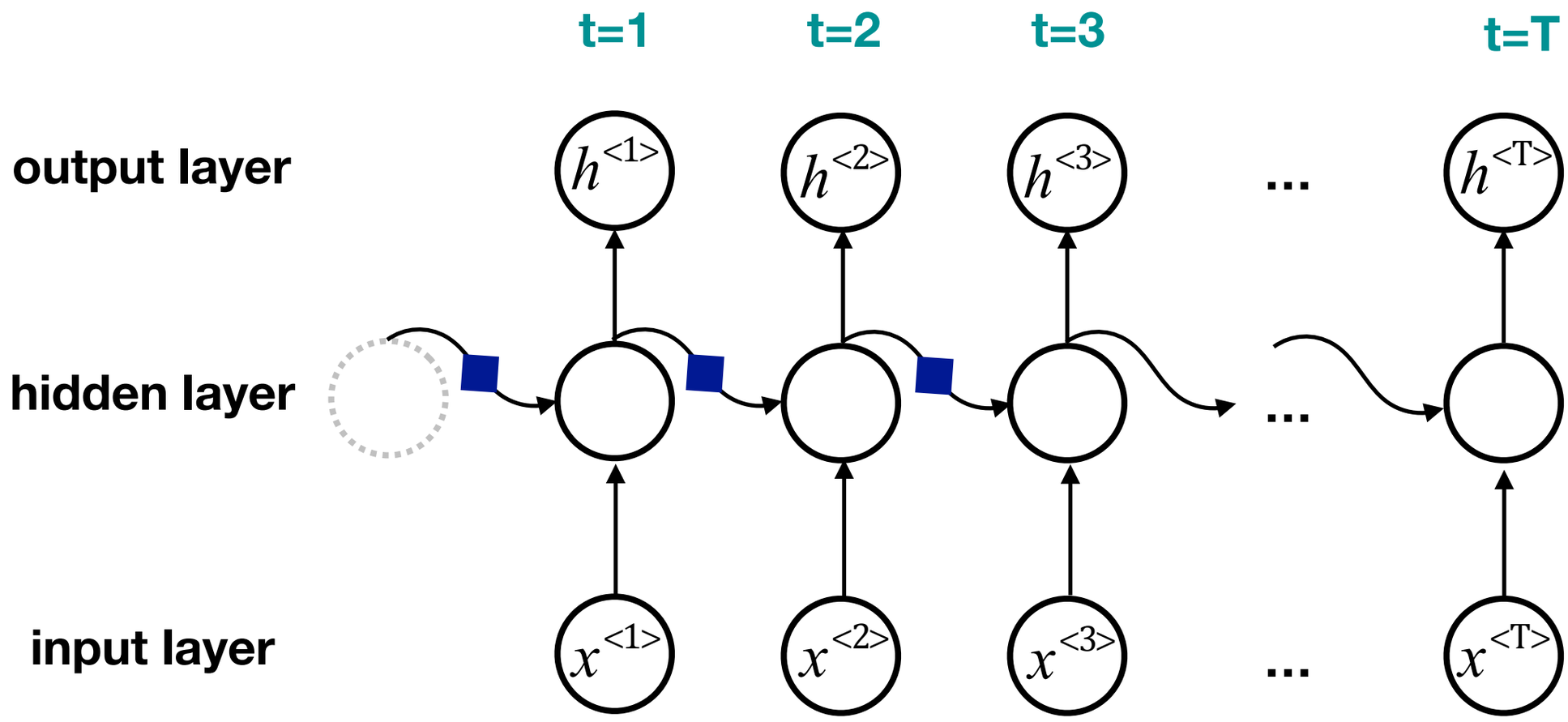
基于时间的反向传播



可以看到：对 W_{xa} 的误差来自于每一个时间步与之后时间步的误差之和。

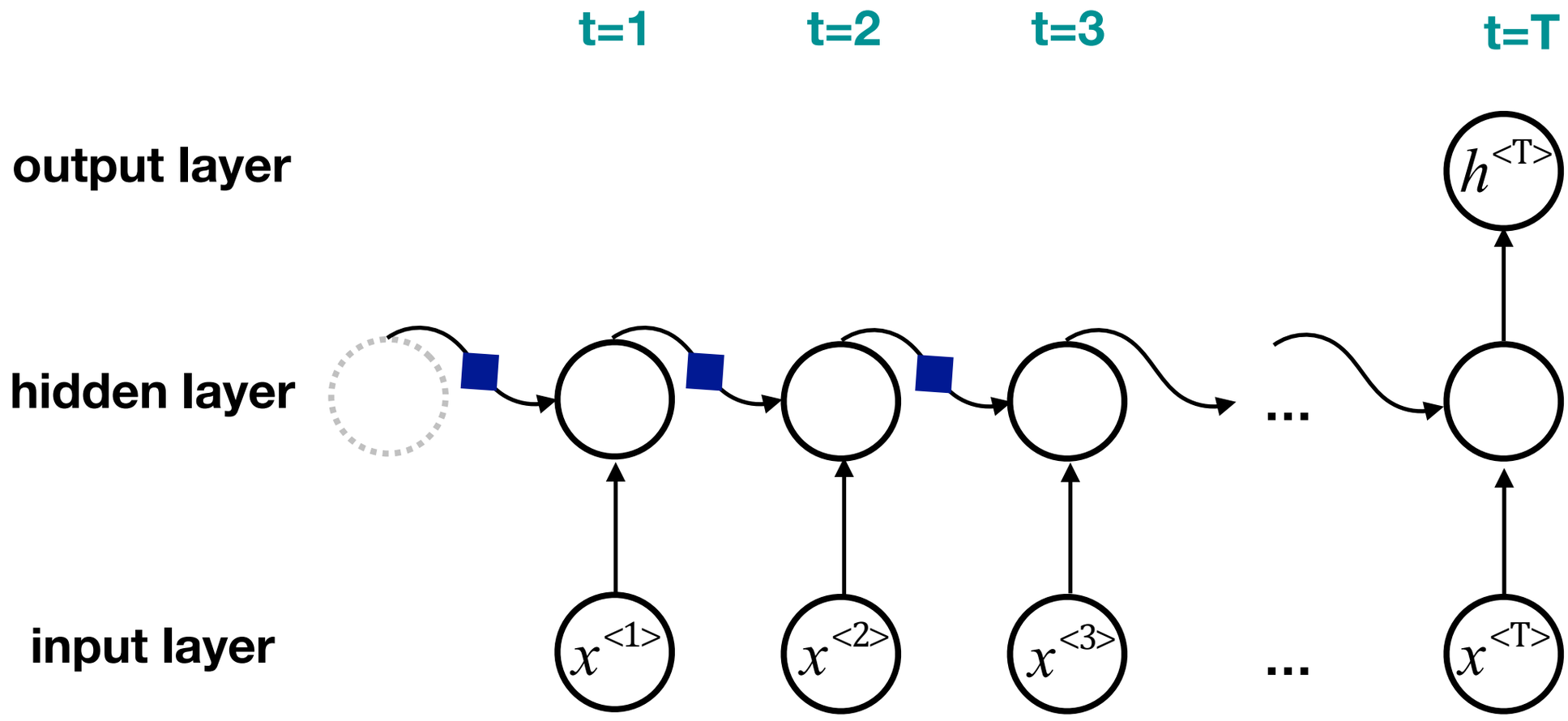
2.2 多种多样的RNN结构

N-N



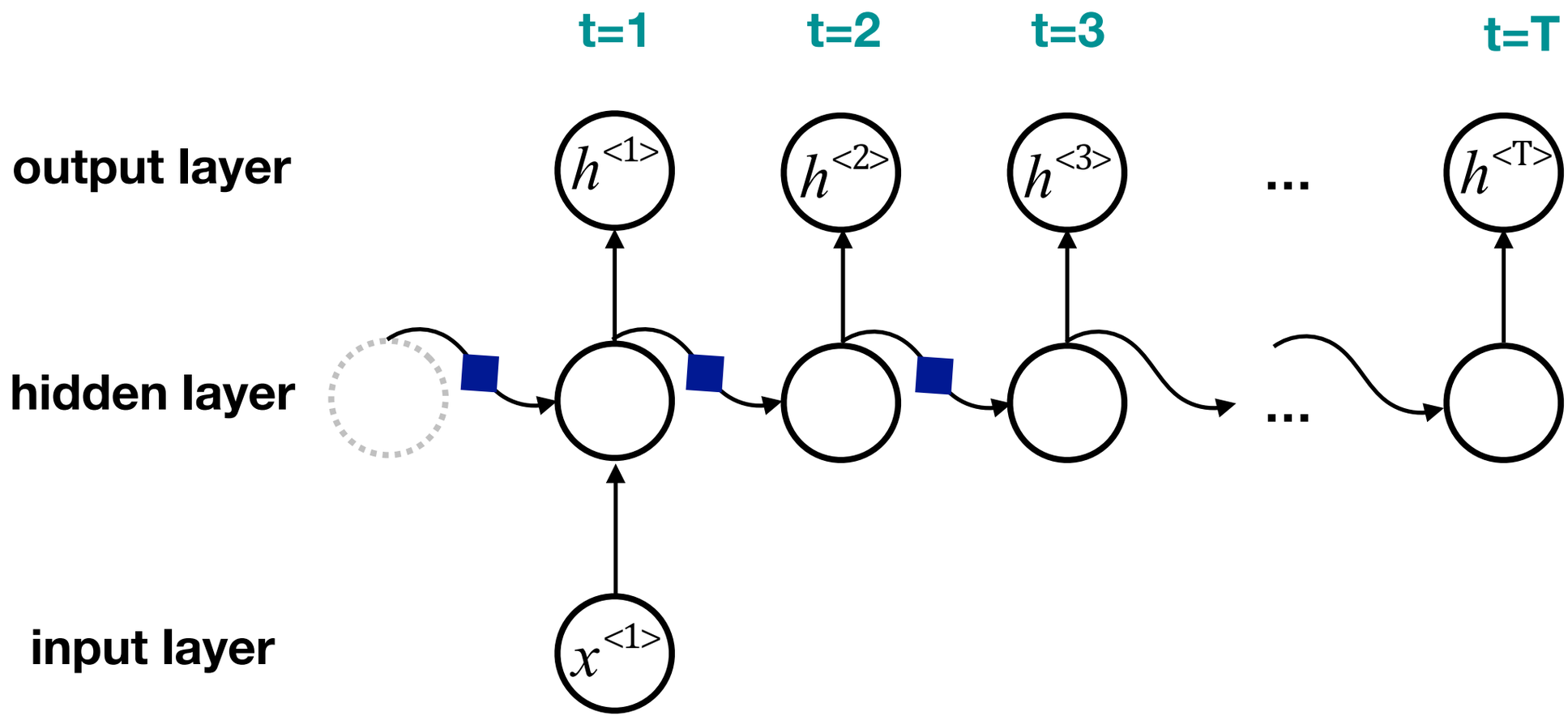
N-N是经典的RNN模型结构，即输入序列数量等于输入序列数量。

N-1



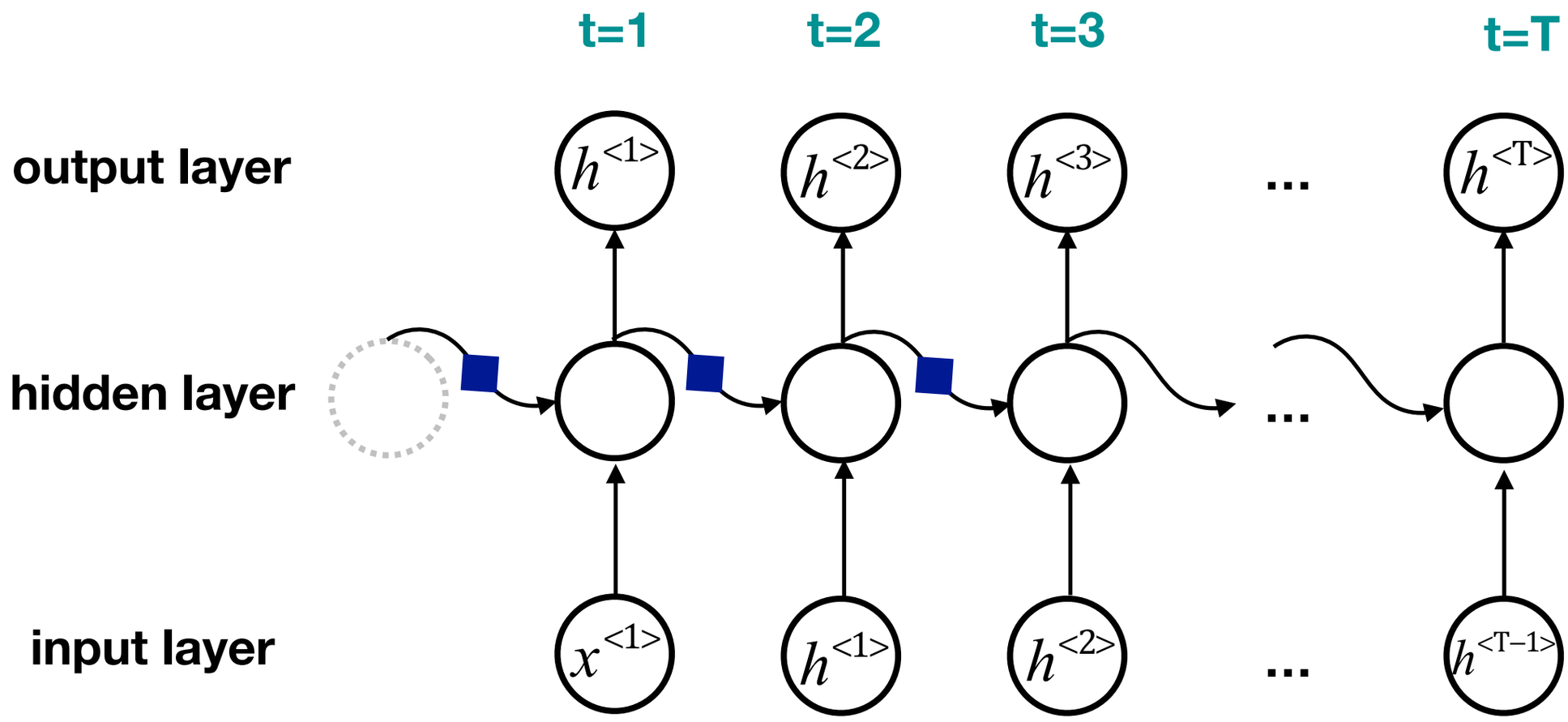
N-1即输入序列数量不固定，输出序列数量等于1。例如输入一段文本，输出是否包含违法信息。

1-N



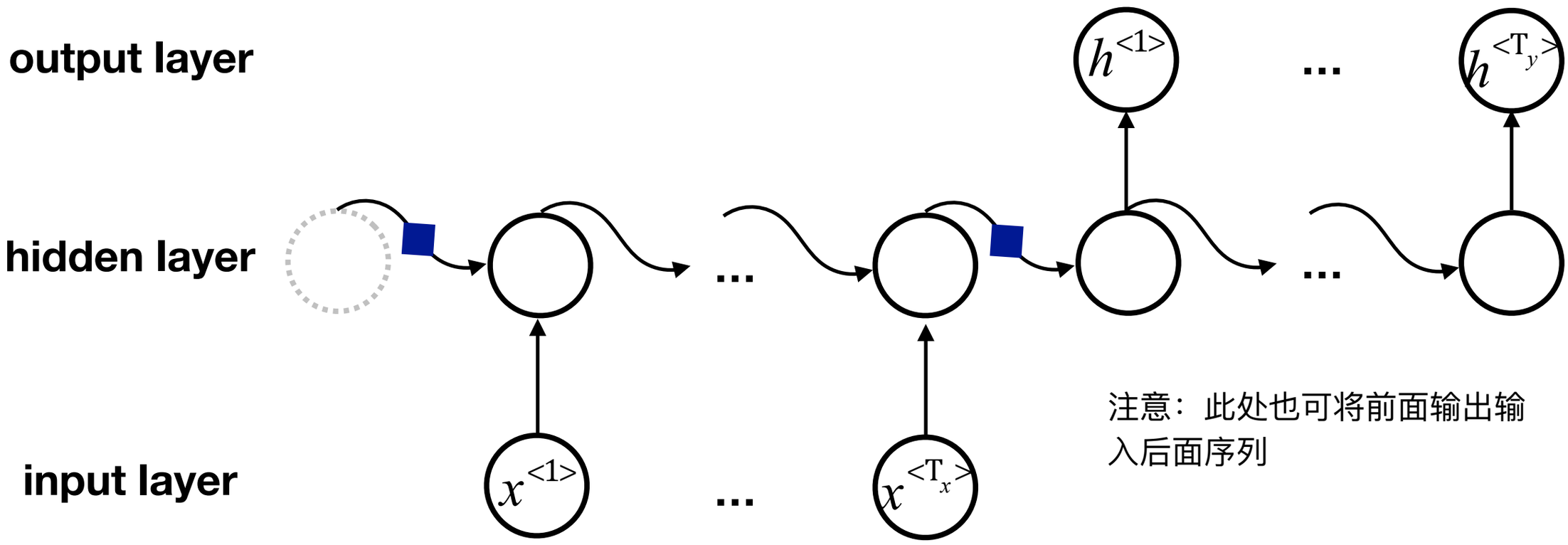
1-N即输入1个序列，生成N个序列。例如根据类别生成音乐，根据图像生成文本等。

1-N



1-N模型通常会将上一个序列的输出作为新的序列的输入来处理。

N-M



N-M即输入序列长度与输出序列长度不相等，例如机器翻译中将英文翻译为中文。此时需要用到多对多的RNN序列。RNN序列的终止可以根据输出特殊符号来确定。还可以完成语音识别、文本摘要等很多功能。

其它结构

- 1-1结构即输入一个序列，输出一个序列，等价于前馈神经网络；

- 注意力机制。

TensorFlow实现

1. 了解TensorFlow实现RNN的方法；

2. 使用多对一结构完成MNIST识别。

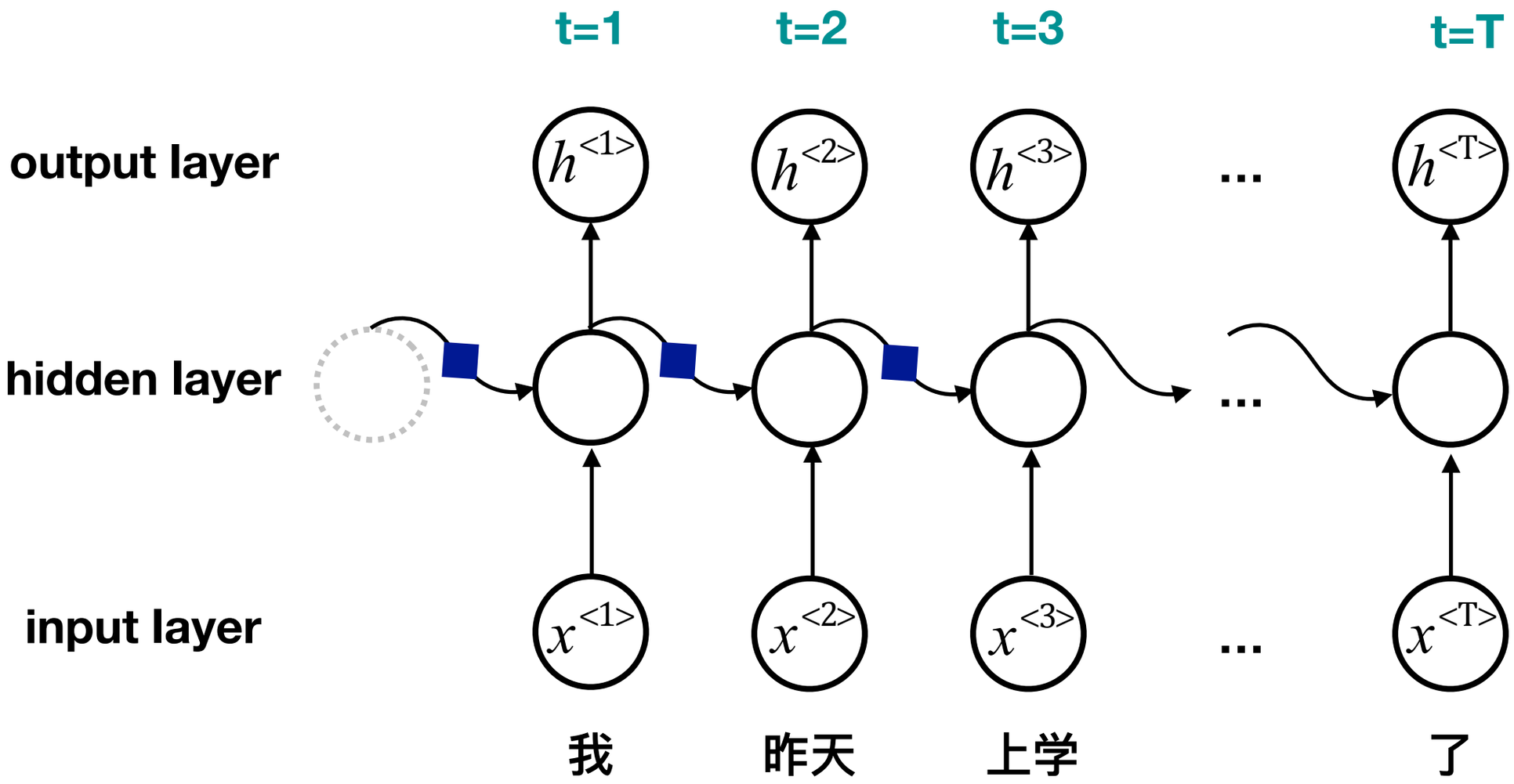
3. LSTM及其变种

3.1 LSTM

LSTM

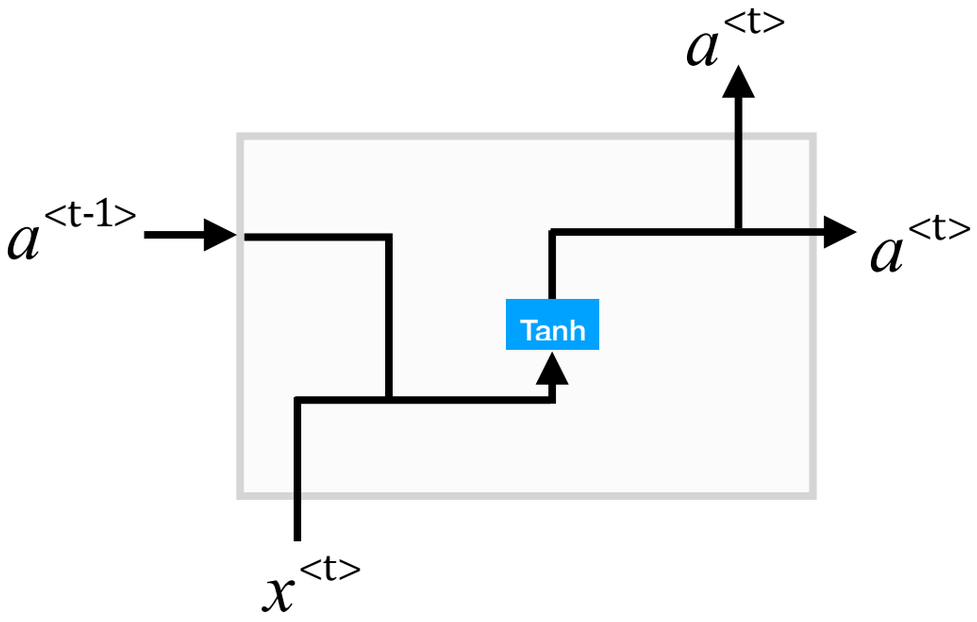
长短时记忆网络（Long Short Term Memory unit，LSTM）是RNN的一种变种，其引入“门控单元”来减缓梯度消失问题，使得模型更容易学习到长期依赖。换句话说记住长期信息是LSTM的默认行为，不是需要付出很大代价才能获得的能力。

RNN的梯度不稳定问题



这里，我们期望模型输出“我”，这个需要根据上文的第一个字符来进行确定。然而在训练模型时，由于最后一个时间序列上的误差回传到第一个序列时已经变得很小（也可能很大），模型会出现梯度消失问题，最终模型很难学习到长期依赖。

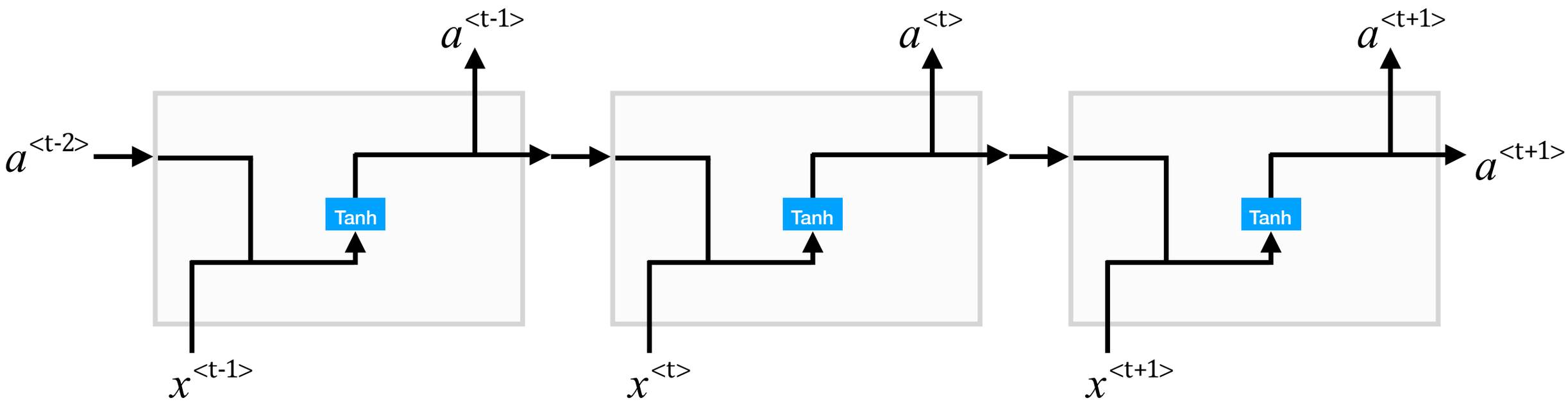
RNN隐藏层的结构



$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$

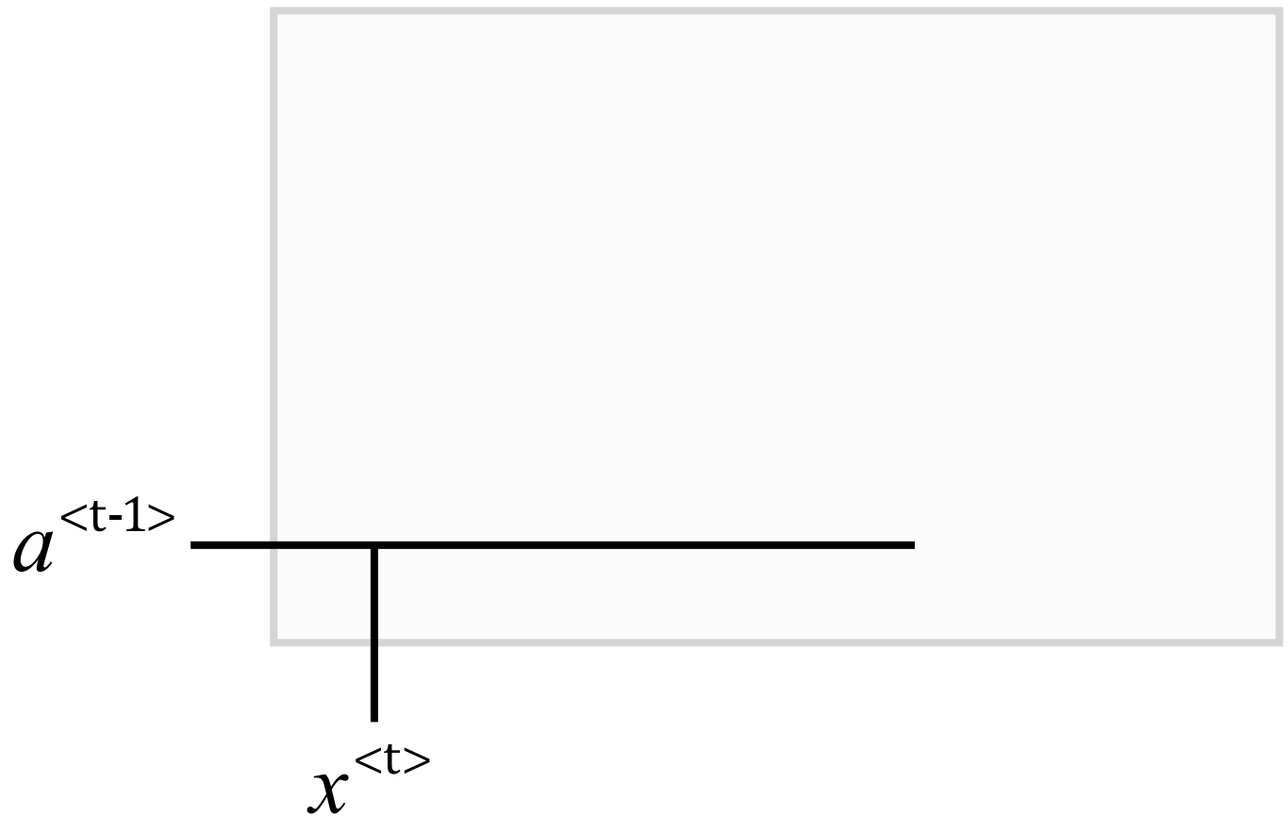
等价于： $a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$

其中，W_a是W_{aa}与W_{xa}在第二个维度上的拼接，a与x是向量的拼接。



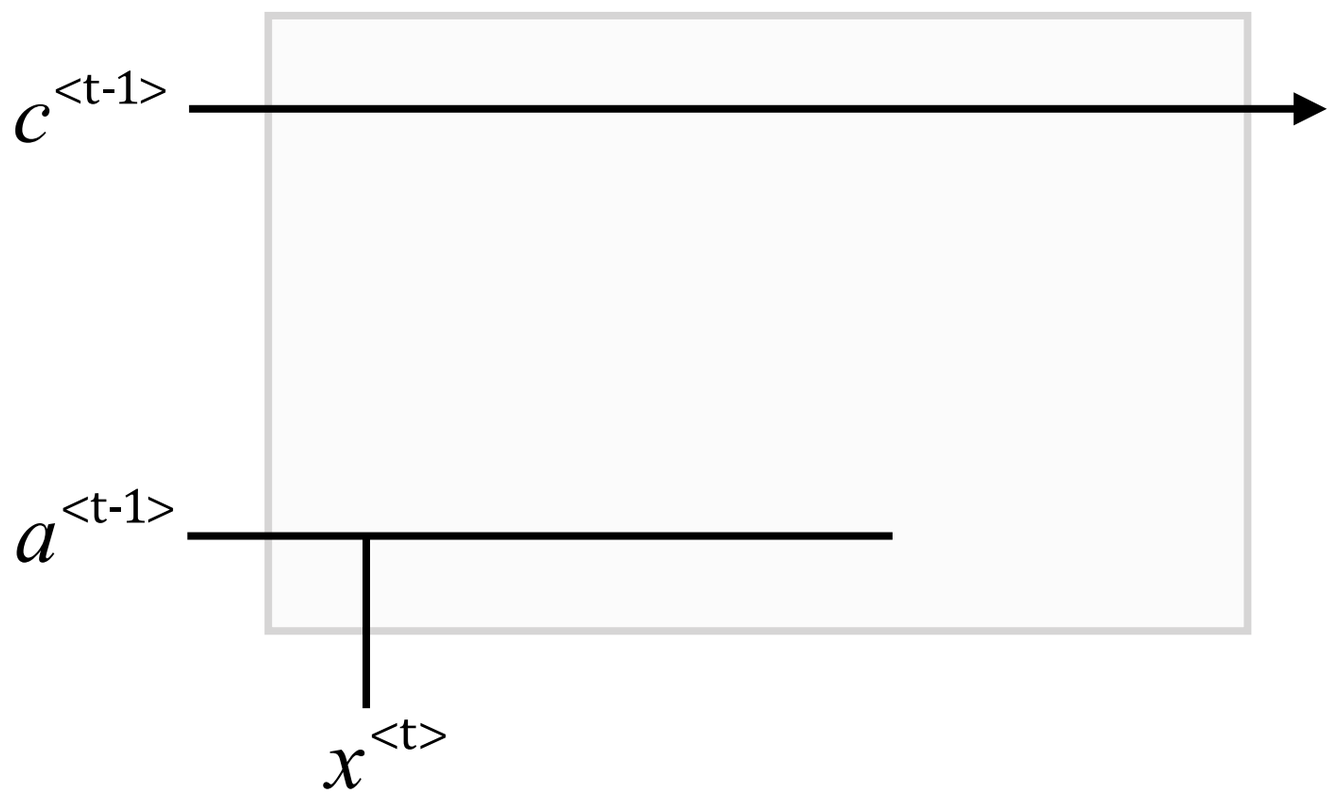
LSTM隐藏层结构

LSTM输入：类似于RNN，隐藏层的输入也可以写成将上一个隐藏层输出与当前输入的拼接。



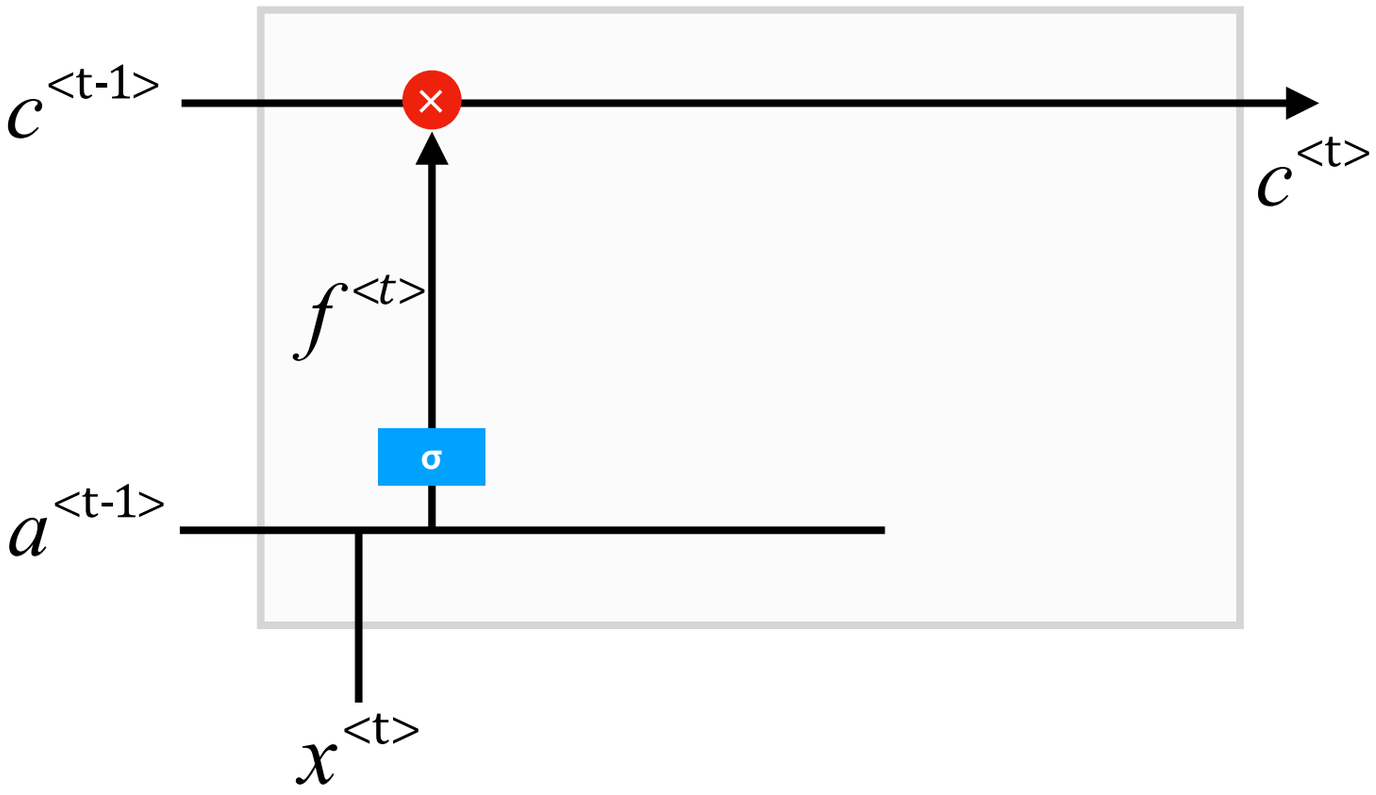
LSTM隐藏层结构

LSTM状态：LSTM比RNN多了状态信息，也可以认为是LSTM的记忆模块，存储了之前的状态信息。



LSTM隐藏层结构

LSTM状态更新：LSTM可以根据输入数据**丢弃**部分无用的“记忆”。例如当输入“我的作业本忘带了…”这时候信息“作业本”可能就会成为无用信息，可以丢弃了。



遗忘门：控制状态信息的去留。

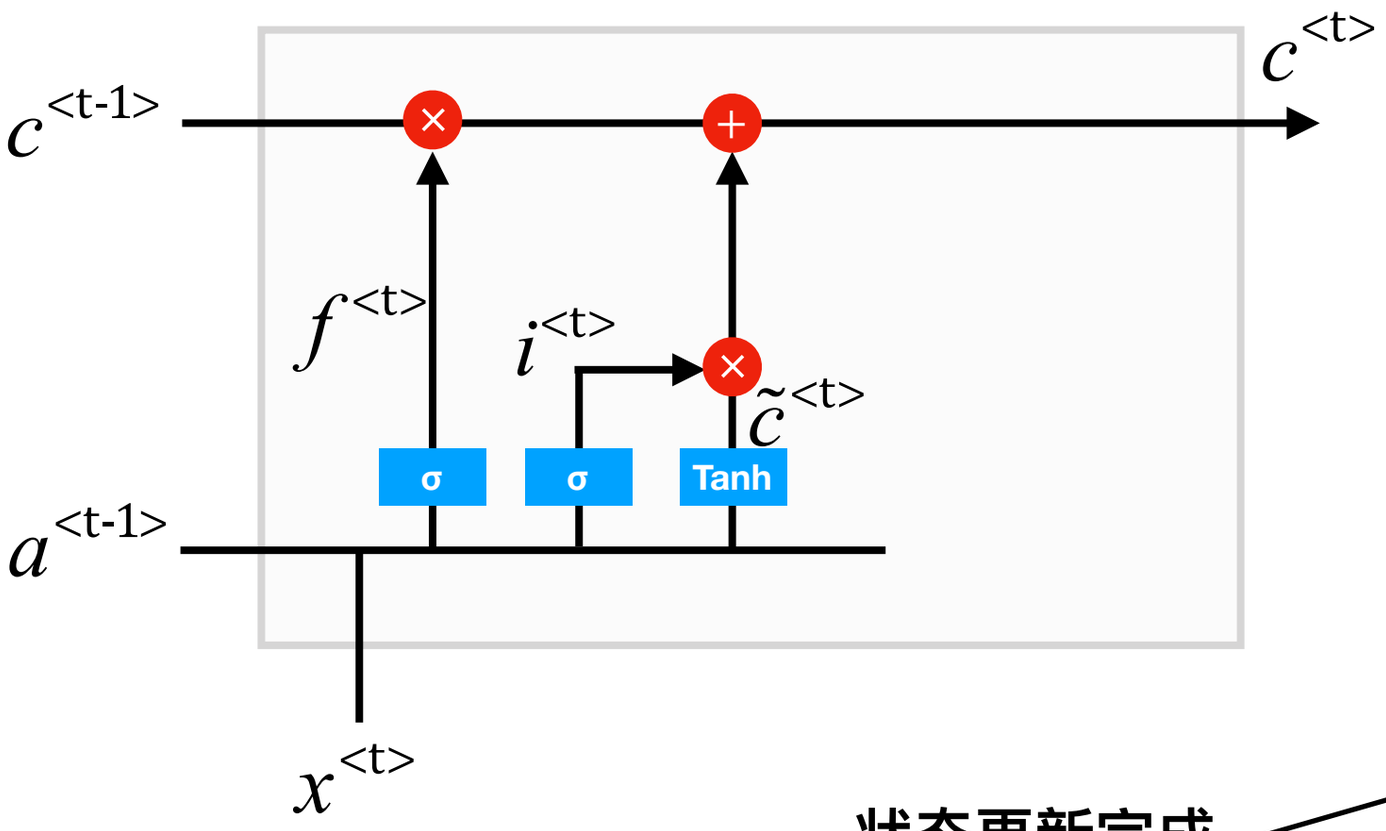
$$f^{<t>}=\sigma(W_f[a^{<t-1>},x^{<t>}]+b_f)$$

遗忘后的状态：

$$f^{<t>}\odot c^{<t-1>}$$

LSTM隐藏层结构

LSTM状态更新：LSTM可以根据输入数据**更新**部分“记忆”。例如当输入“我的作业本忘带了，需要再写一份”这时候丢弃了“作业本”信息，又加入了“再写”的信息。



输入门：控制输入信息的哪部分加入状态。

$$i^{<t>} = \sigma(W_i[a^{<t-1>}, x^{<t>}] + b_i)$$

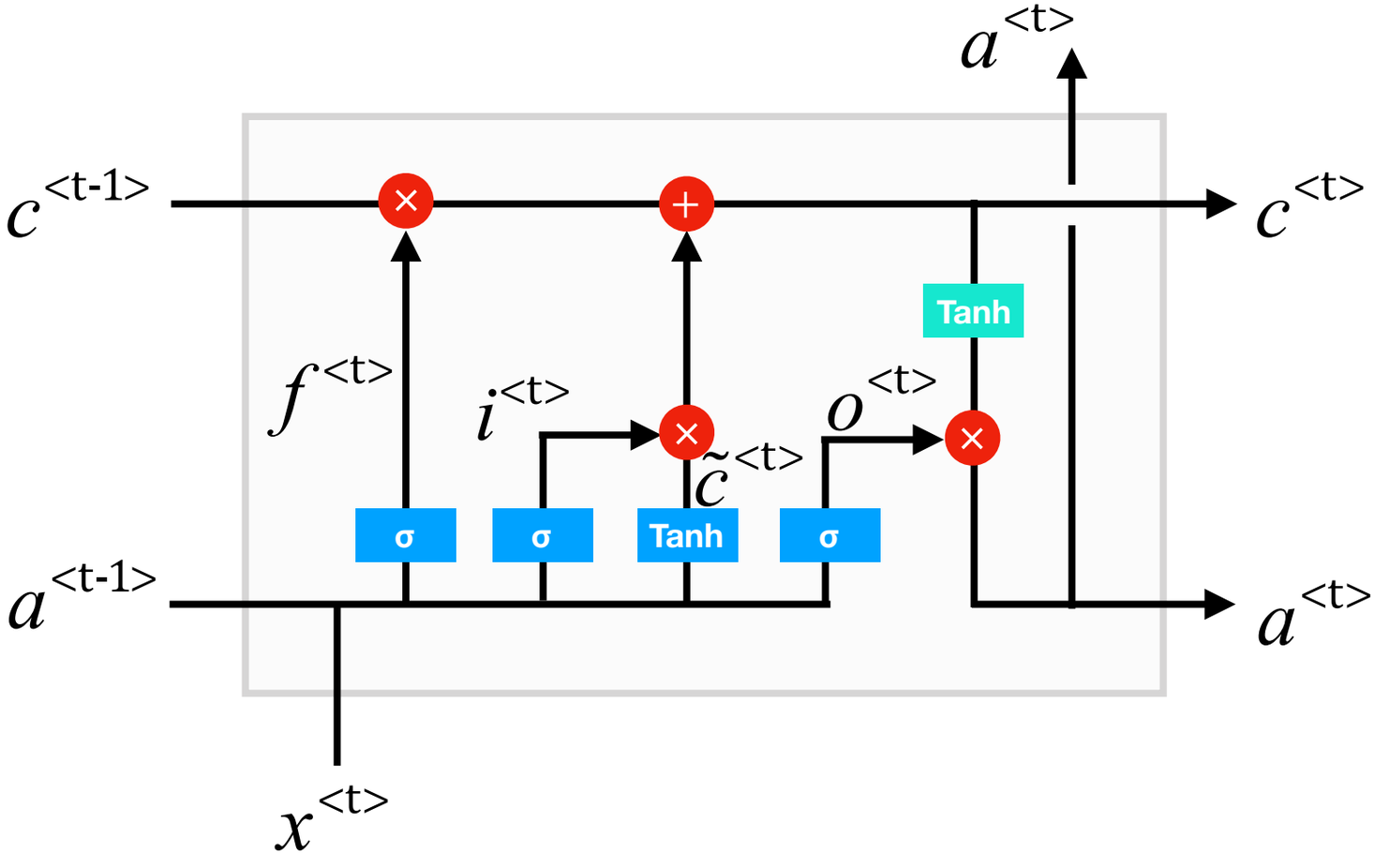
$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

输入新信息后的状态：

状态更新完成 → $$c^{<t>} = f^{<t>} \odot c^{<t-1>} + i^{<t>} \odot \tilde{c}^{<t>}$$

LSTM隐藏层结构

LSTM输出：根据输入信息选择输出隐藏层状态的内容（并不是所有的内容都是需要在当前输出）。



输出门：控制从状态信息中输出哪些信息：

$$o^{<t>}=\sigma(W_o[a^{<t-1>},x^{<t>}]+b_o)$$

LSTM输出为：

$$a^{<t>}=o^{<t>}\odot\tanh(c^{<t>})$$

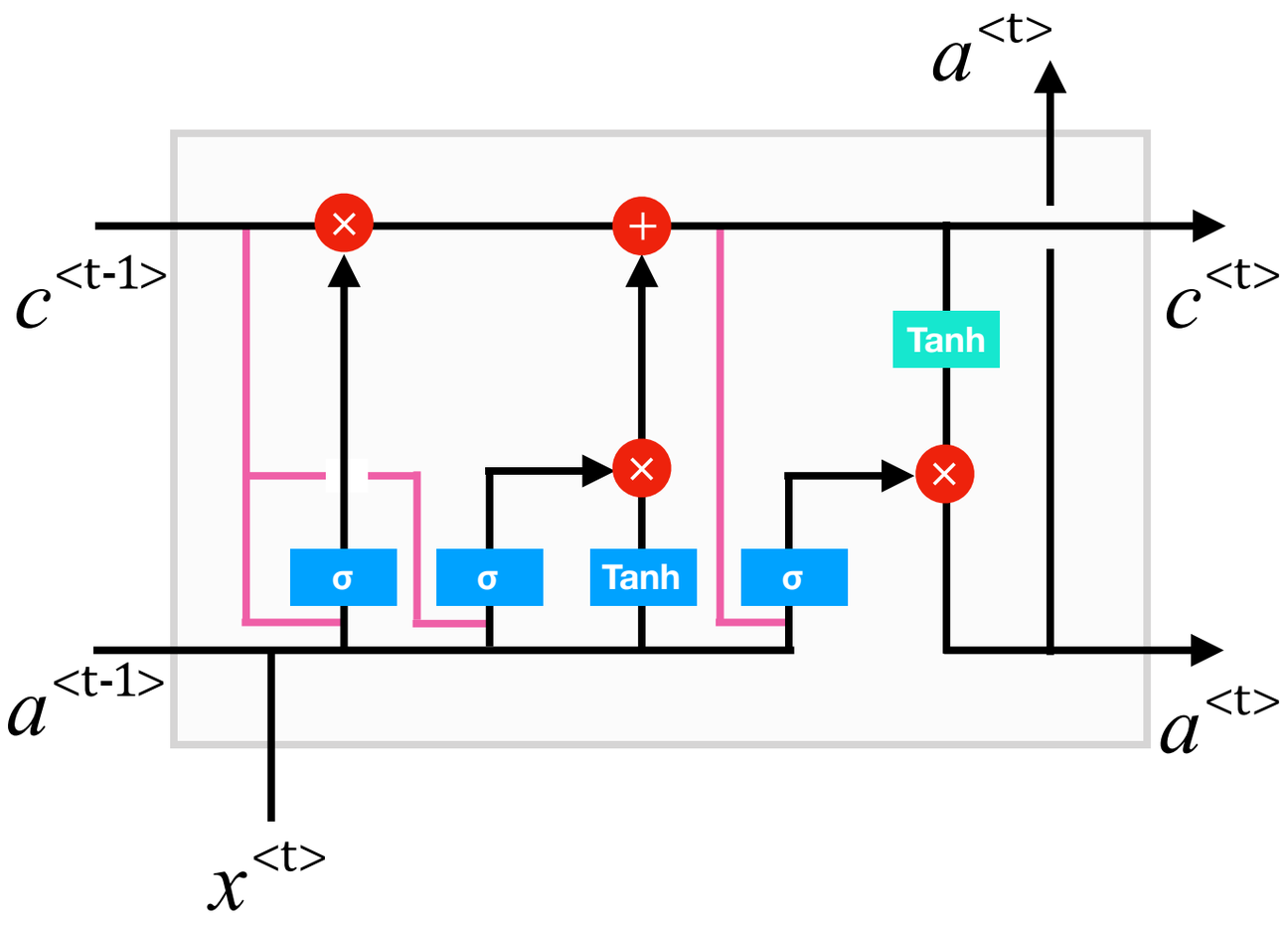
LSTM特点

- 通过加入状态结构，使得LSTM天然具备了“记忆”能力；
- LSTM使用了遗忘门、输入门、输出门控制数据的更新与输出；
- LSTM减缓了梯度消失问题，使得模型更容易训练与更容易获得长程依赖。

3.2 LSTM变种

带窥孔连接的LSTM

窥孔连接（Peephole Connection）是一种流行的LSTM变种，具体做法是在LSTM的各个门接收状态作为输入。



3个门如下：

$$f^{<t>} = \sigma(W_f[c^{<t-1>}, a^{<t-1>}, x^{<t>}] + b_f)$$

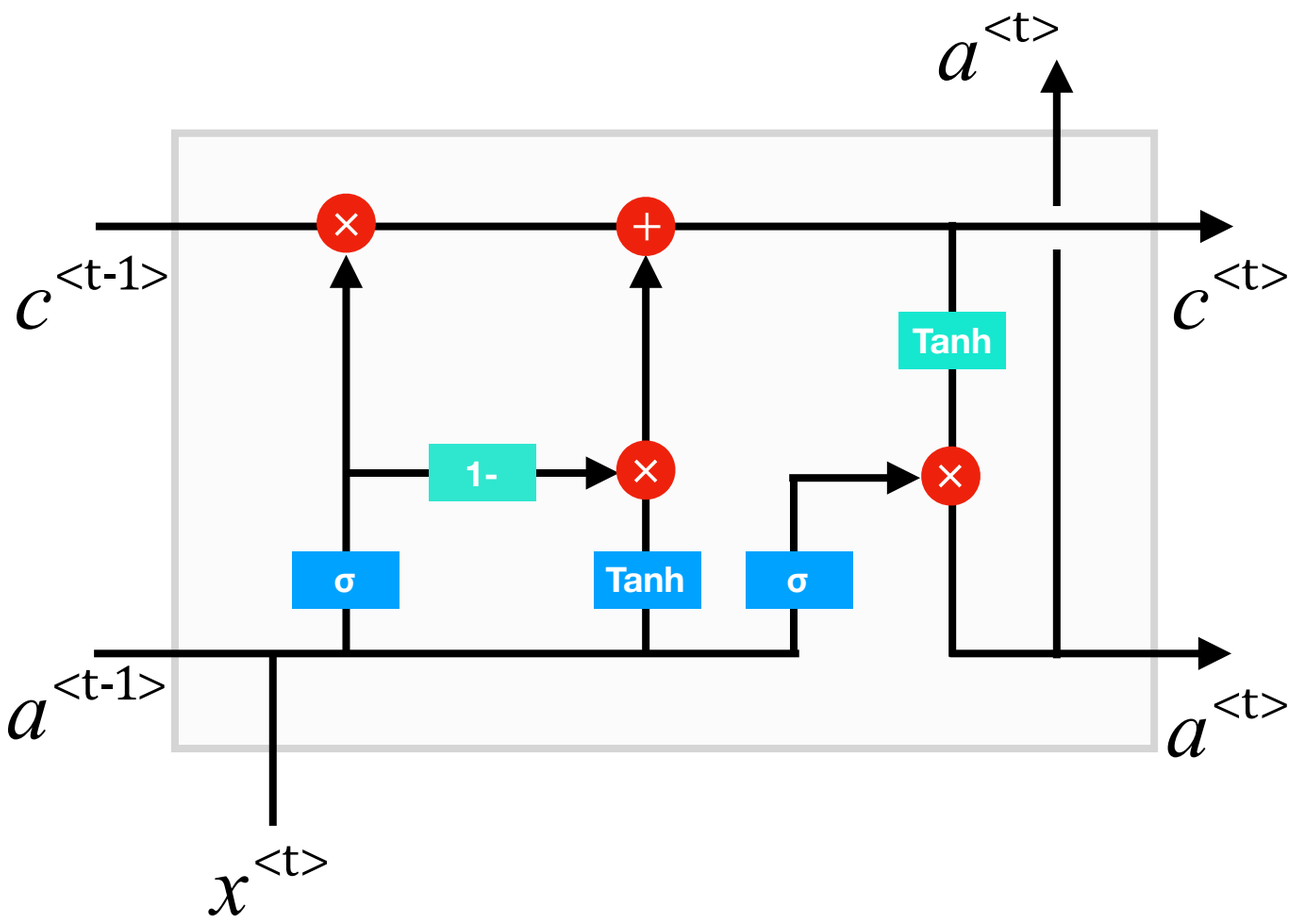
$$i^{<t>} = \sigma(W_i[c^{<t-1>}, a^{<t-1>}, x^{<t>}] + b_i)$$

$$o^{<t>} = \sigma(W_o[c^{<t>}, a^{<t-1>}, x^{<t>}] + b_o)$$

实际中，不一定必须使用3个窥孔，也可仅使用2个或一个，用法比较多样。

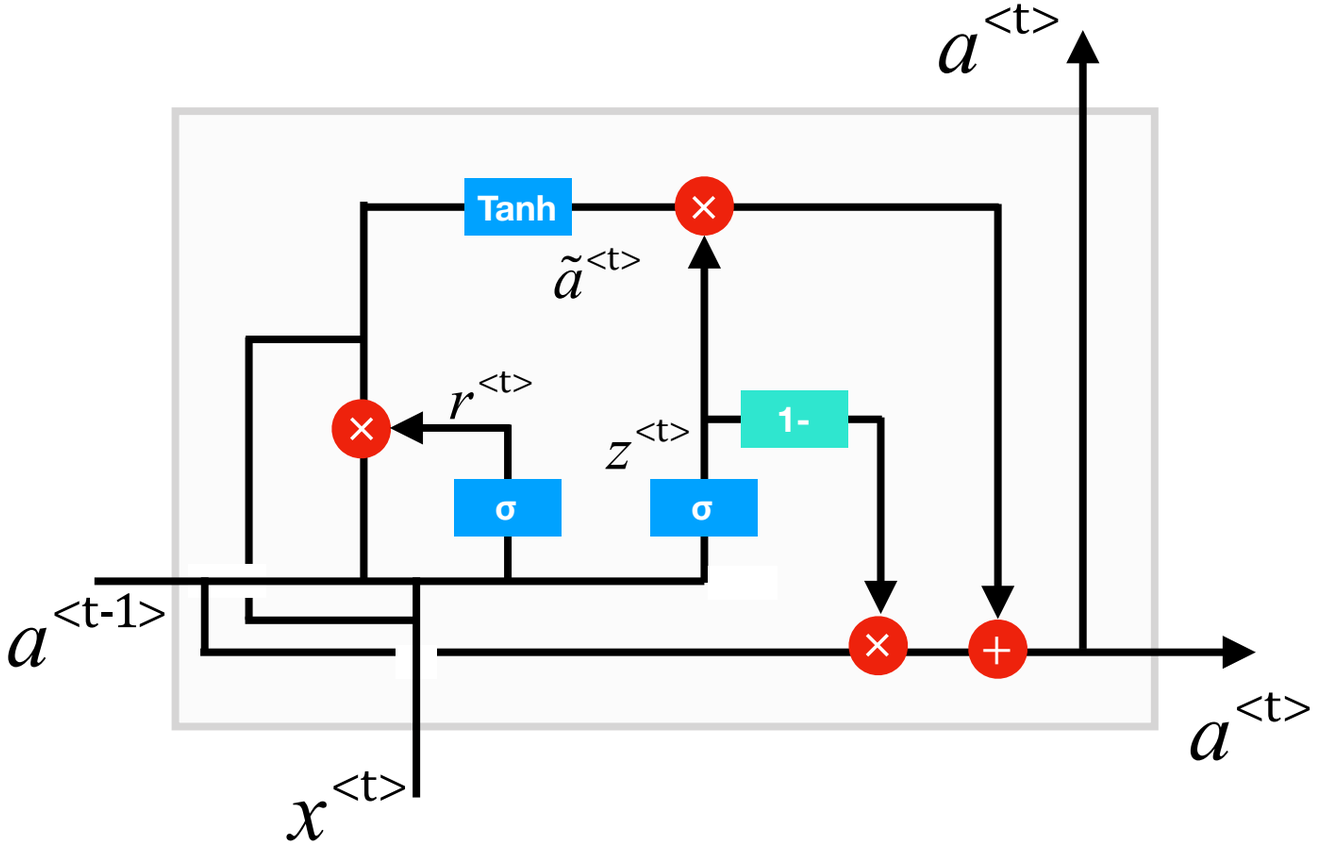
耦合遗忘门与输入门

LSTM中遗忘门与输入门是独立，事实上我们可以使用一个门代替两个门，即在状态中遗忘的位置加入新信息。这样可以减少参数数量。 $c^{<t>} = f^{<t>} \odot c^{<t-1>} + (1- f^{<t>}) \odot \tilde{c}^{<t>}$



GRU

Gated Recurrent Unit(GRU)，是LSTM的一种简化版，GRU去除了状态c。与LSTM相比，GRU表现出了相近的性能，但其参数规模更小，是最常使用的一种RNN模型。



$$r^{<t>} = \sigma(W_r[a^{<t-1>}, x^{<t>}] + b_r)$$

$$z^{<t>} = \sigma(W_z[a^{<t-1>}, x^{<t>}] + b_z)$$

$$\tilde{a}^{<t>} = \tanh(W_a[r^{<t>} \odot a^{<t-1>}, x^{<t>}] + b_a)$$

$$a^{<t>} = (1 - z^{<t>}) * a^{<t-1>} + z^{<t>} * \tilde{a}^{<t>}$$

TensorFlow实现

- 了解使用TensorFlow实现LSTM以及其变种GRU等的方法；

- 使用LSTM完成MNIST训练。

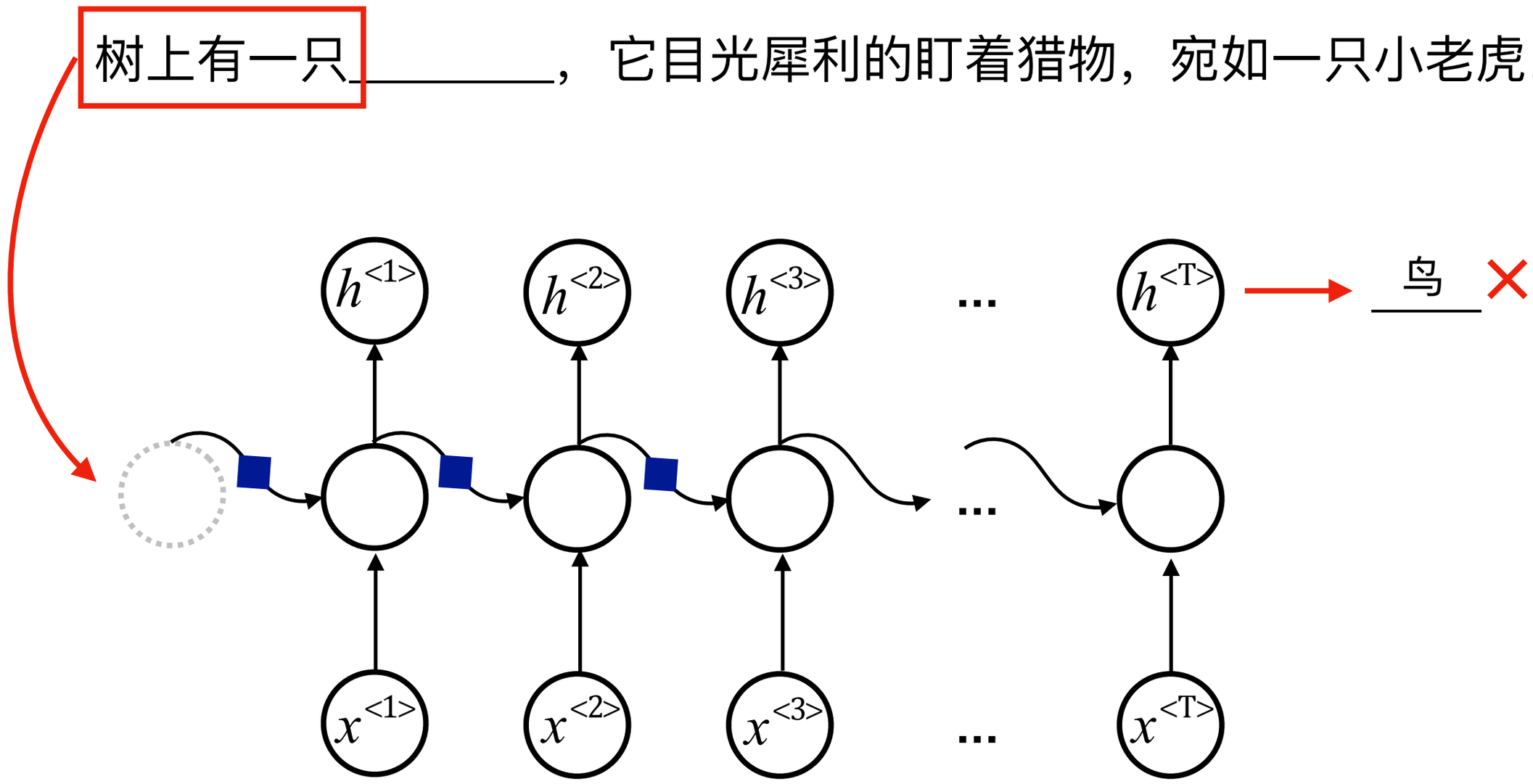
4. 双向循环神经网络

双向循环神经网络

双向循环神经网络（Bidirectional RNN）是一种可以利用上下文信息的循环神经网络。BiRNN在结构上类似于使用了2个一正一反的子循环神经网络组成，子网络也可以是LSTM以及其变种。

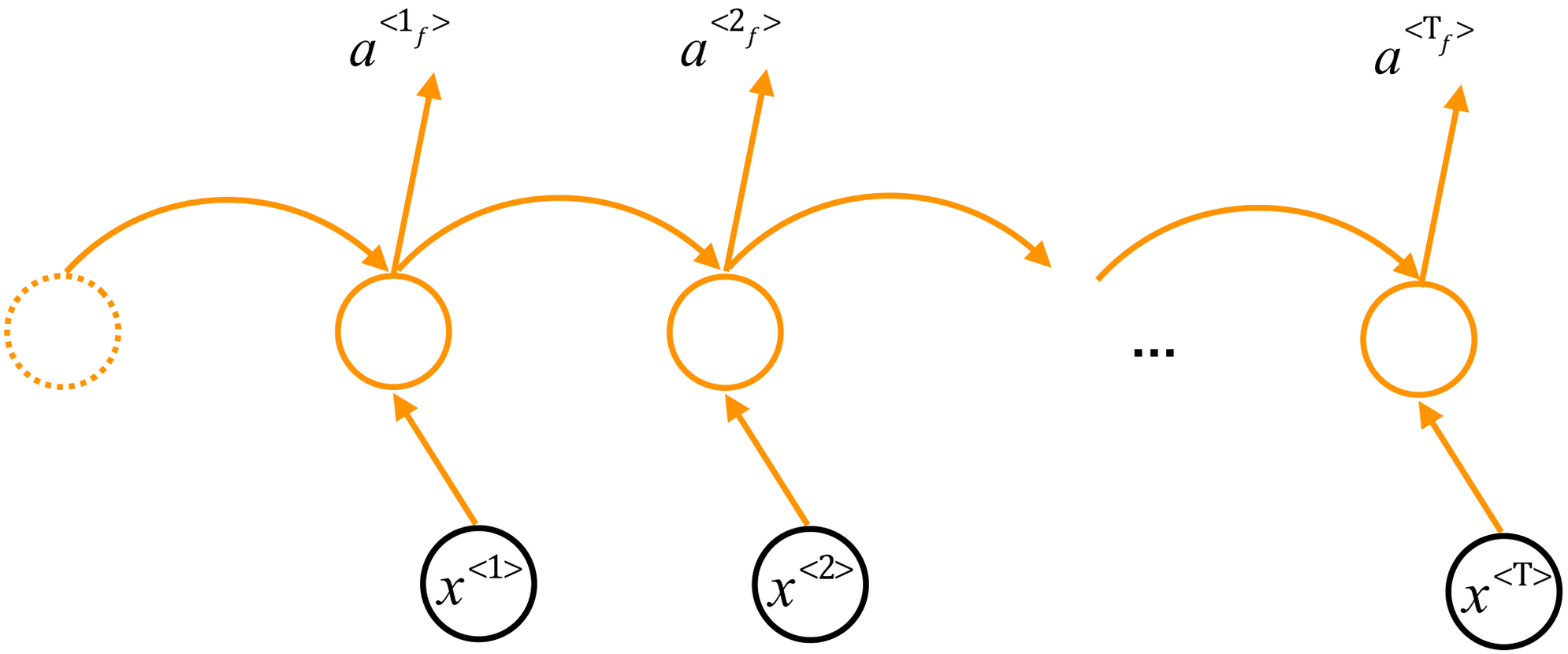
使用RNN预测缺失词

树上有一只_____，它目光犀利的盯着猎物，宛如一只小老虎。



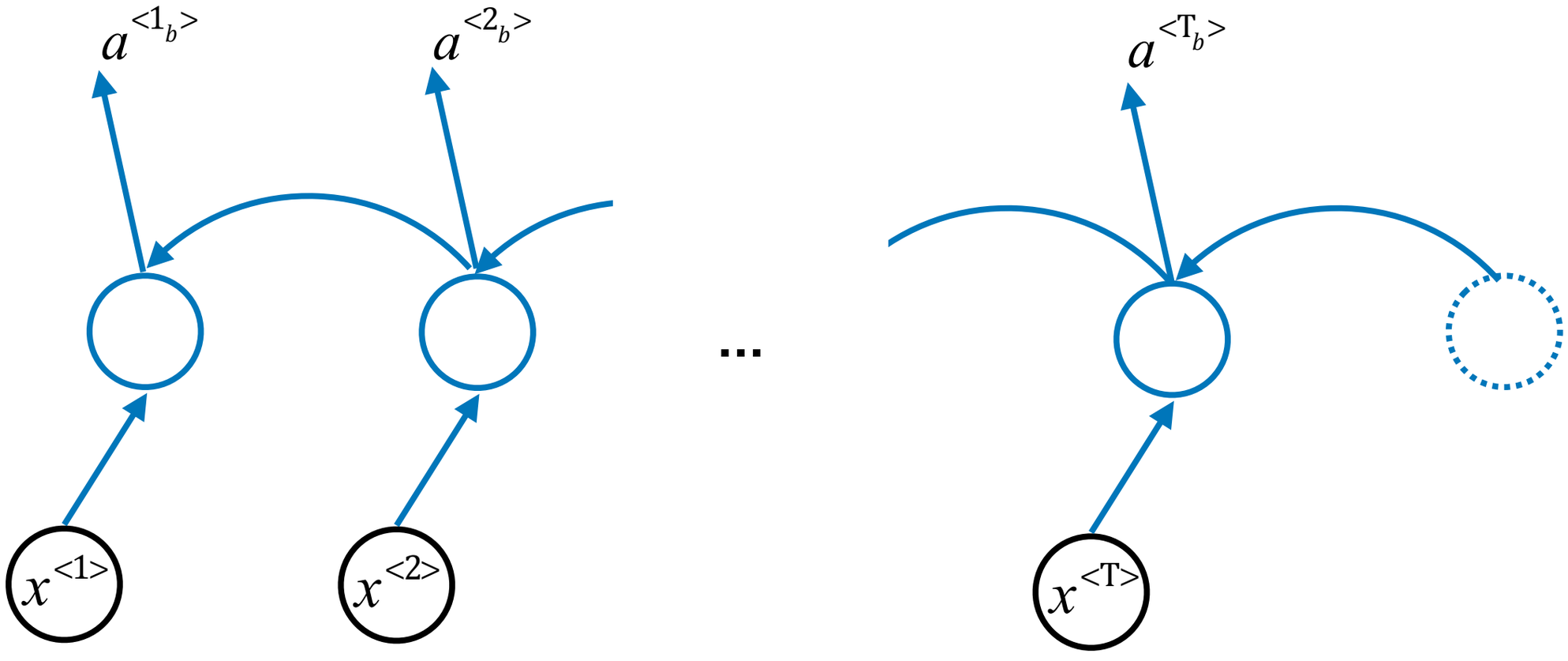
使用RNN的问题：RNN只能利用上文信息，无法利用下文信息。

双向循环神经网络



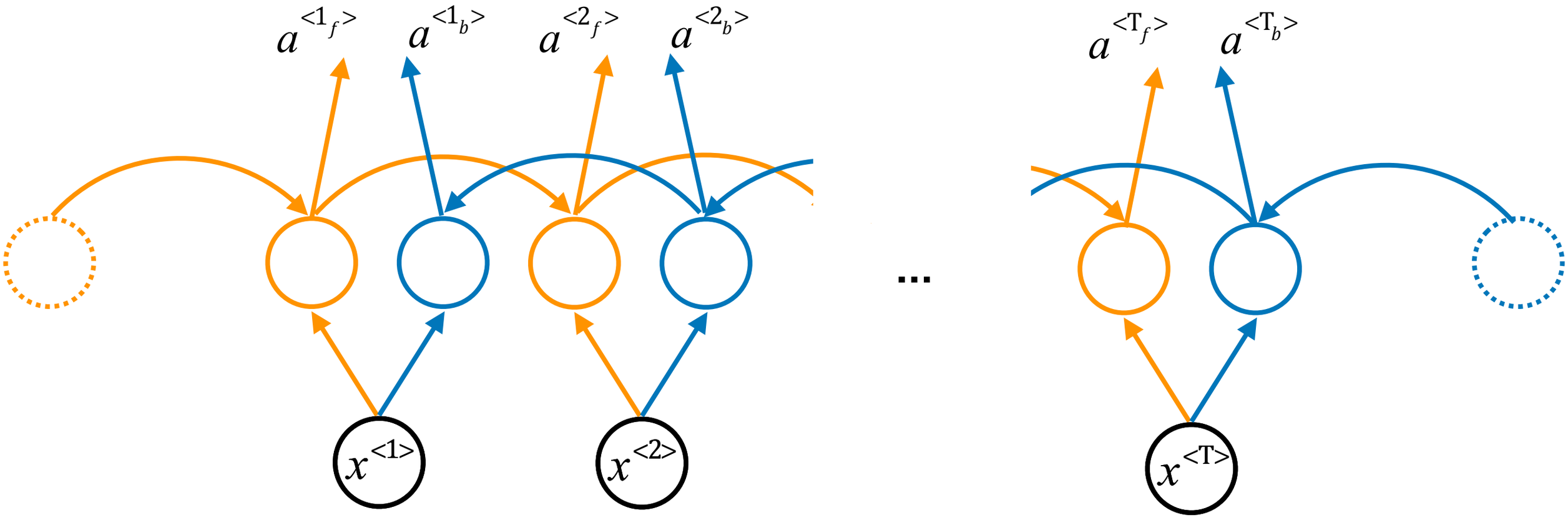
双向循环神经网络包含两个子网络，第一个网络的序列**顺序输入**循环神经网络。

双向循环神经网络



双向循环神经网络包含两个子网络，第二个网络的序列**倒序输入**循环神经网络。

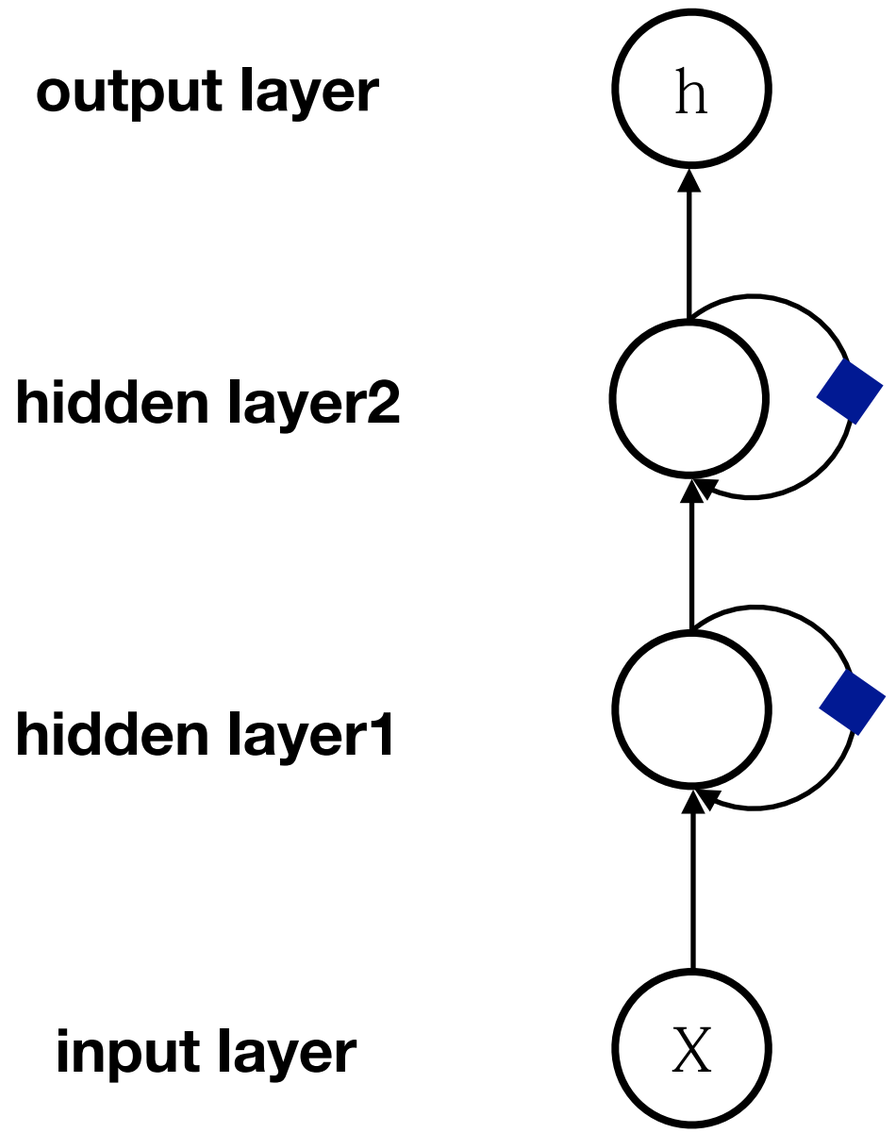
双向循环神经网络



双向循环神经网络每一时刻的输出都包含两个子模型输出，这样模型能够利用上下文信息。

5. 深度循环神经网络

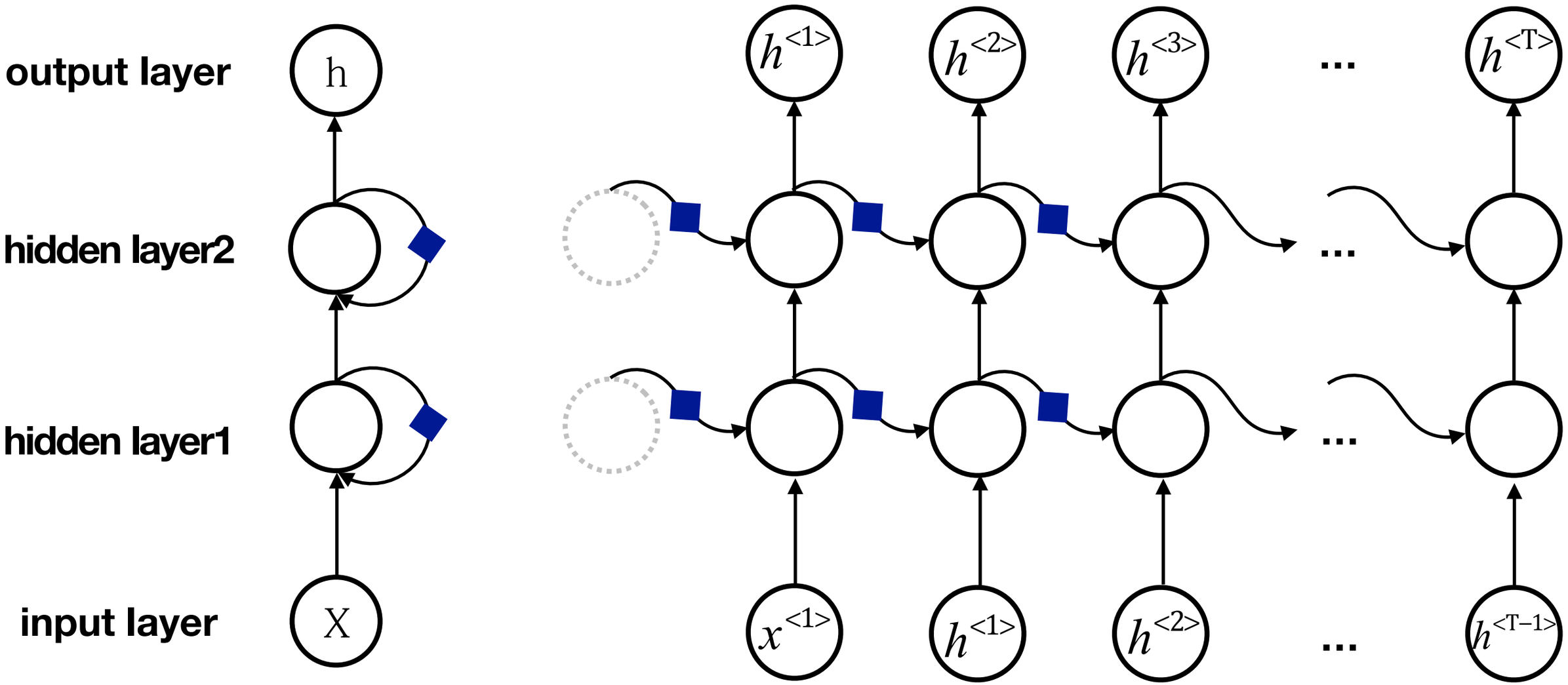
深度循环神经网络



特点：

1. Deep RNN即包含多个隐藏层RNN的模型；
2. 每个RNN层接受各自对应的上个时间序列的输出；
3. 由于时间序列往往较长，浅层模型的复杂度也会很高，所以RNN深度不宜太深。

深度循环神经网络


output layer
h
hidden layer2
hidden layer1
input layer
X
$h^{<1>}$
$h^{<2>}$
$h^{<3>}$
...
$h^{<T>}$
$x^{<1>}$
$h^{<1>}$
$h^{<2>}$
$h^{<T-1>}$

使用TensorFlow实现

- 了解TensorFlow实现双向循环神经网络的方法；
- 了解TensorFlow实现深度循环神经网络的方法；
- 使用深度循环神经网络实现MNIST识别

6.循环神经网络的应用实例

6.1 word2vec

word2vec

Word2vec是一种在自然语言处理中将字词表示为较（**One-Hot Representation**）低维度的稠密**词向量**表示。word2vec比one-hot encoding表示的维度更低，词与词之间的语义关系更强。

One-Hot Representation

Whatever is worth doing is worth doing well.

1. 以“单词”为个体表示文本（数量多）；
2. 以“字母”为个体表示文本（数量少，组合多）。

任何值得做的，就把它做好。

1. 以“字”为个体对文本表示（数量少，组合多）；
2. 以“词”为个体对文本表示（数量多）。

One-Hot Representation

使用字母表示：

Whatever is worth doing is worth doing well.

[w, h, a, t, e, …, e, l, l]

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\cdots\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

[a, b, c, d, e, f, g, h, i, g, k ,l ,…, x, y, z]

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\cdots\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

共26个独热编码向量

One-Hot Representation

使用单词表示：

Whatever is worth doing is worth doing well.

[whatever, is, …., well]

[a, aah, above,…., zweig, zymogen,]

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\cdots\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\qquad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

独热编码向量

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\qquad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\cdots\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

One-Hot Representation

使用字表示：

任何值得做的，就把它做好。

[一, 二,丁,三, …, 龖, 靐, 龘,]

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\cdots\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

[任, 何, 值, 得 …, 好]

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\cdots\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

一般的只使用常用字，非常用字使用<UNK>符号代替

One-Hot Representation

使用词表示：

任何值得做的，就把它做好。

分词

[任何, 值得 …, 它, 做好]

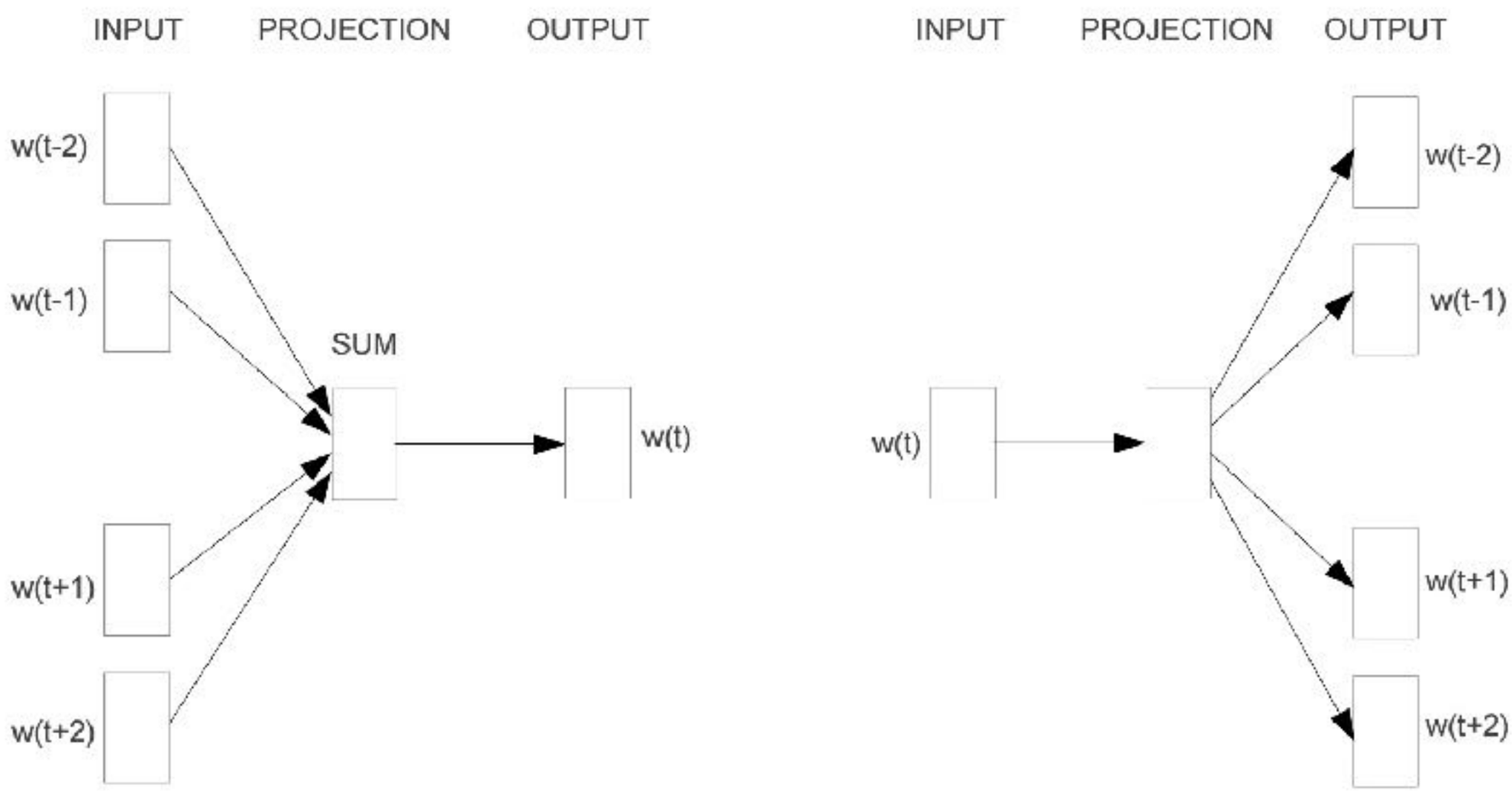
$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\cdots\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

[上学, 上课, 石家庄，…, 学习]

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\cdots\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

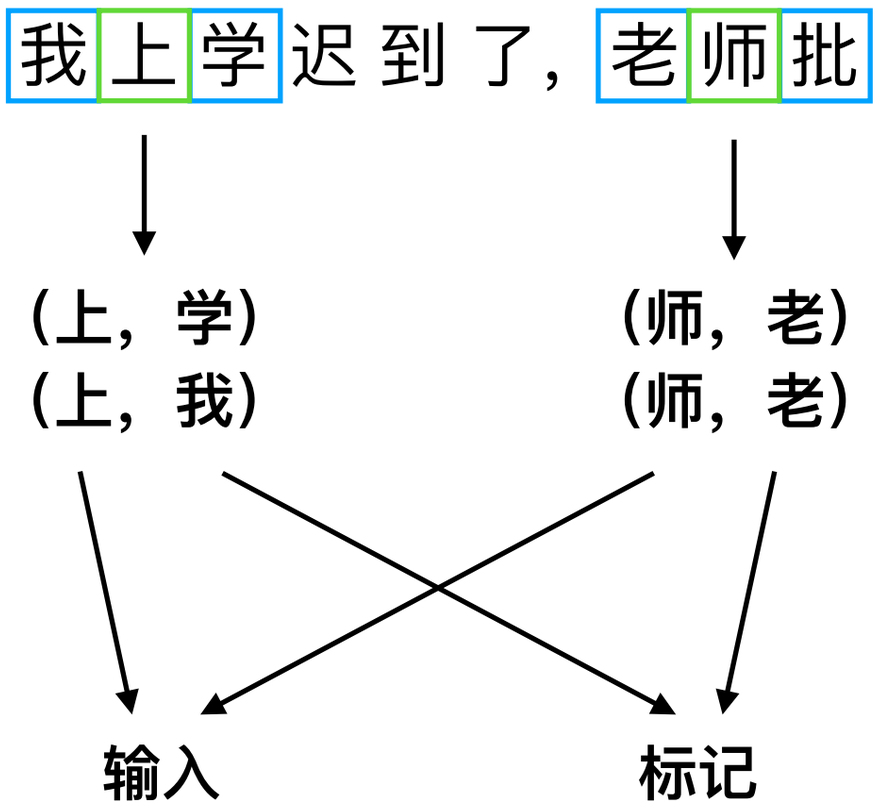
一般的会有数万甚至更多的的常见词

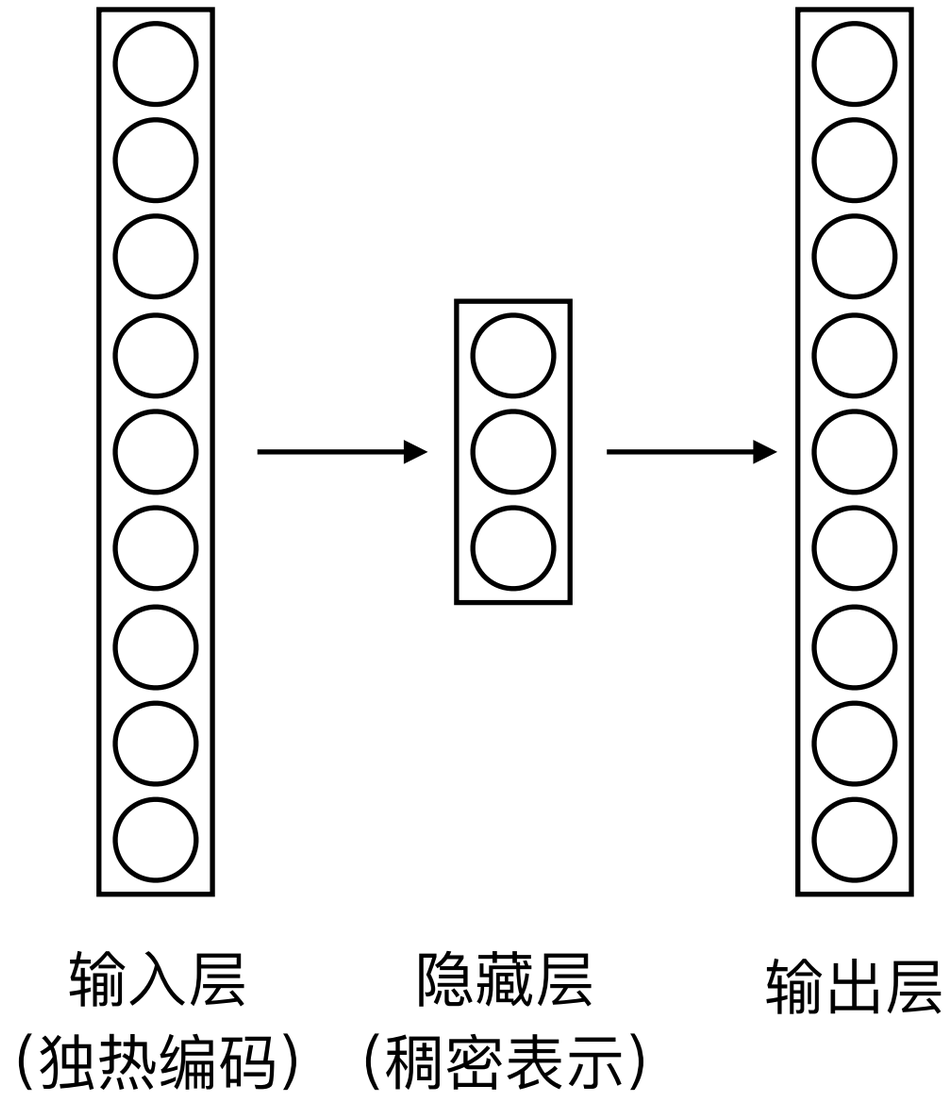
word2vec模型结构



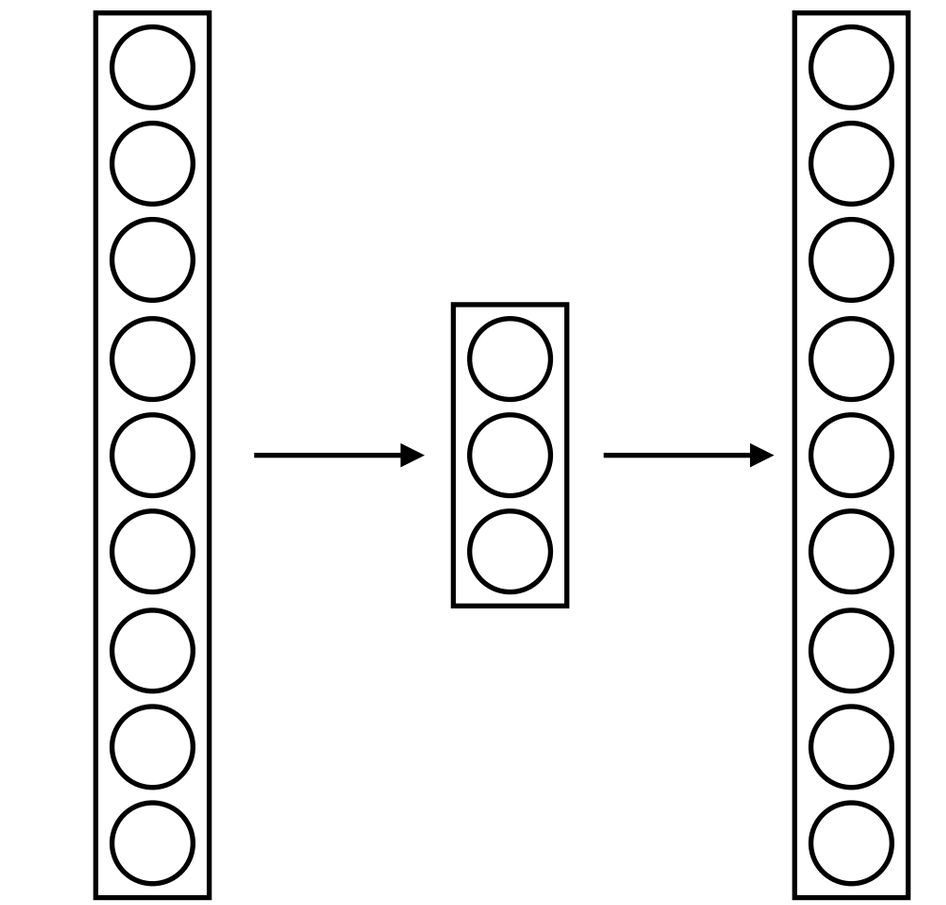
CBOW **Skip-gram**

skip-gram



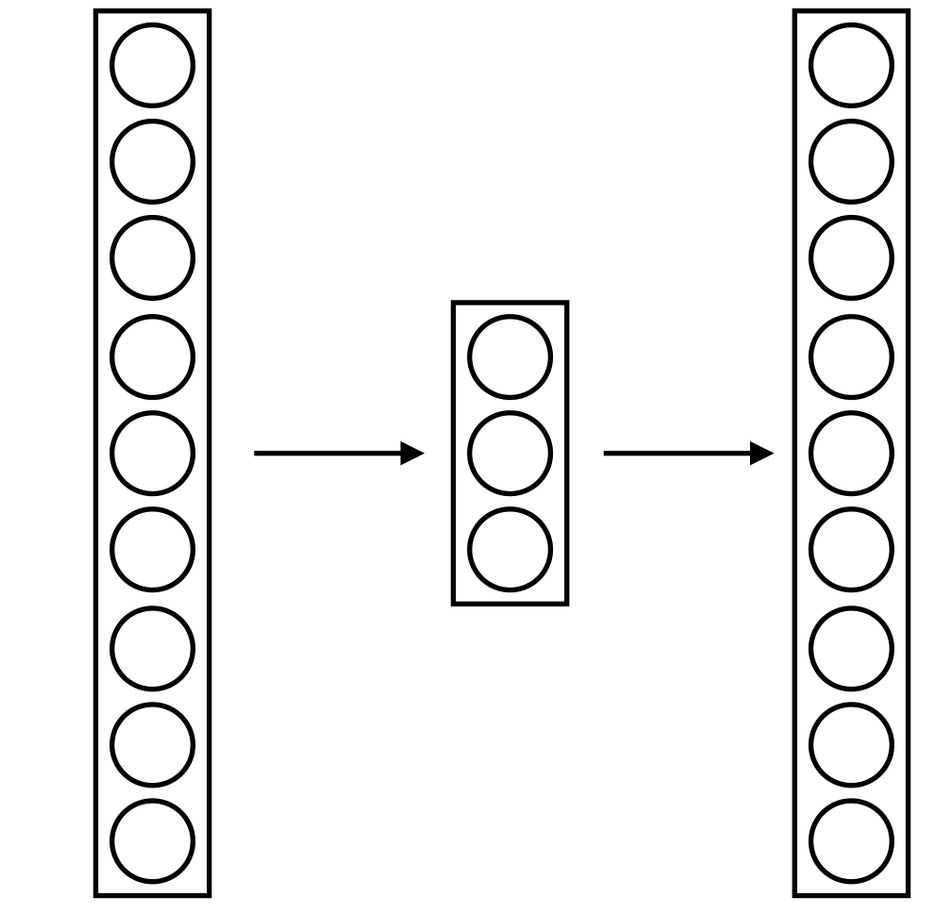


skip-gram



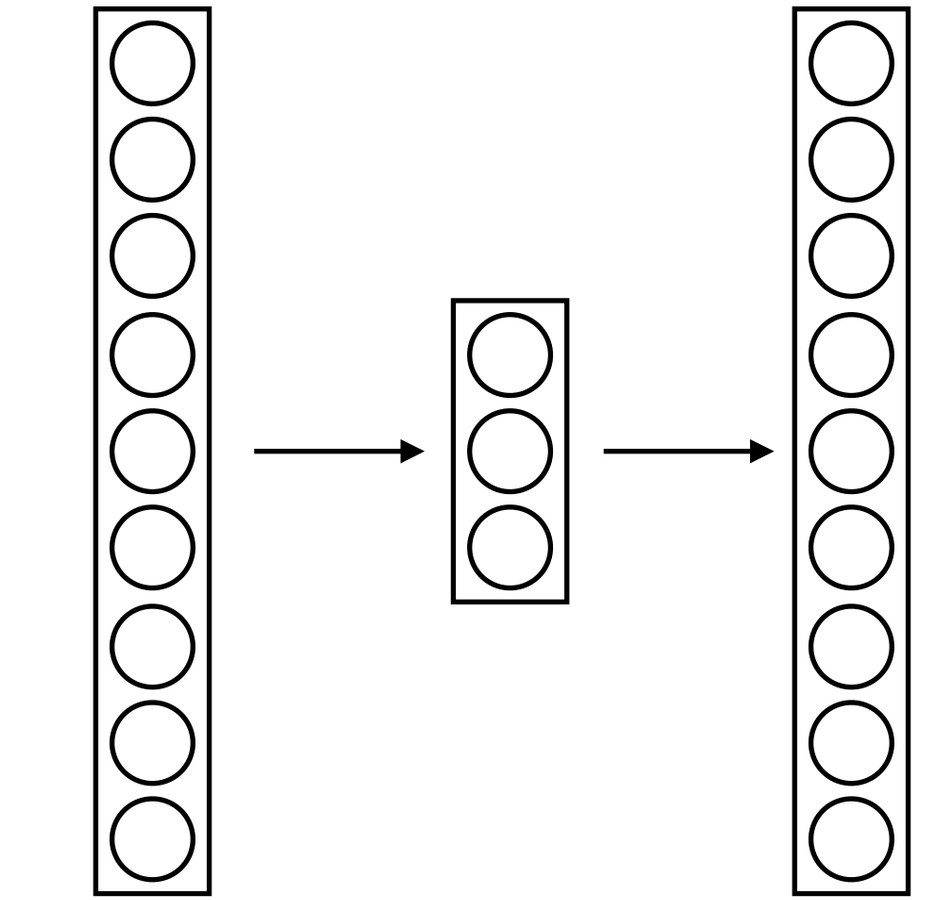
词（或字、字母）在句子中的位置越接近，在语义关系上也越接近，我们使用句子中的某个词作为输入，希望模型预测与此词关系最近的此，这样模型就可以学习到词与词之间的语义关系。同时利用神经网络对输入进行了压缩，使得表示的维度大大降低。

skip-gram



1. 隐藏层没有激活函数；
2. 输出层激活函数为softmax；
3. 由于输入层、输出层的规模往往很大，直接使用softmax时计算效率很低，往往采用一些替代方法（分层softmax或负采样）。

负采样



输入层
（独热编码）

隐藏层
（稠密表示）

输出层

我 上 学 迟 到 了，老 师 批 评 了 我。

（上，学） → 1

（上，其） → 0

（上，下） → 0　随机生成的标记结果

（上，了） → 0

1. 将输出层的softmax激活函数转换为logistic激活函数，即将多分类转化为多个二分类。
2. 每个正样本生成对应的N个负样本。
3. 训练时仅对正负样本对应的输出单元求大年并进行训练。

word2vec

- 实际中由于word2vec输入是规模很大的稀疏张量，所以一般并不采用矩阵运算。

- word2vec不仅可以用来查找邻近词，还可以根据两个向量的差查找对应词。

- word2vec的使用有两种方式，一种是无监督学习即前面所述方法；另一种是监督学习，即将其作为模型输入层随着具体任务学习。

6.2 语言模型

语言模型

语言模型（language model）也被称为统计语言模型，是一种对字词等组合出的序列的可能性进行描述的模型。广泛应用于语音识别、机器翻译、中文分词、拼写纠错等任务中。语言模型可形式化表示为：$p(S)=p(w_1,w_2,...,w_n)=\prod_{i=1}^{n}p(w_i|w_1,w_2,...,w_{i\text{-}1})$

语言模型

输入音频

识别结果

1. 做作业
2. 做做业
3. 坐坐业
4. …

音频识别结果中，每一个发音都相同，但只有第一个识别结果是我们想要的，这里就需要用到语言模型判断哪个组合出现的概率更高。

语言模型

语言模型 $p(S)=p(w_1,w_2,\ldots,w_m)=\prod_{i=1}^{m}p(w_i\mid w_1,w_2,\ldots,w_{i\text{-}1})$

其中 $p(w_i\mid w_1,w_2,\ldots,w_{i\text{-}1})=\dfrac{\text{count}(w_1,w_2,\ldots,w_i)}{\text{count}(w_1,w_2,\ldots,w_{i\text{-}1})}$

缺点：

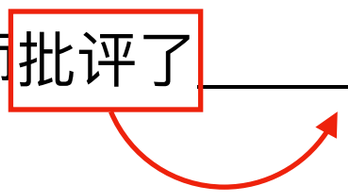
1. 参数空间大，计算量大，实际中难以实现；
2. 数据集中有大量的稀疏组合，难以获取足够了的样本。

N-Gram

N-Gram模型是上述语言模型的具体实现。其采用马尔科夫假设，即认为语言中每个单词只与其前面长度N-1的上下文有关。

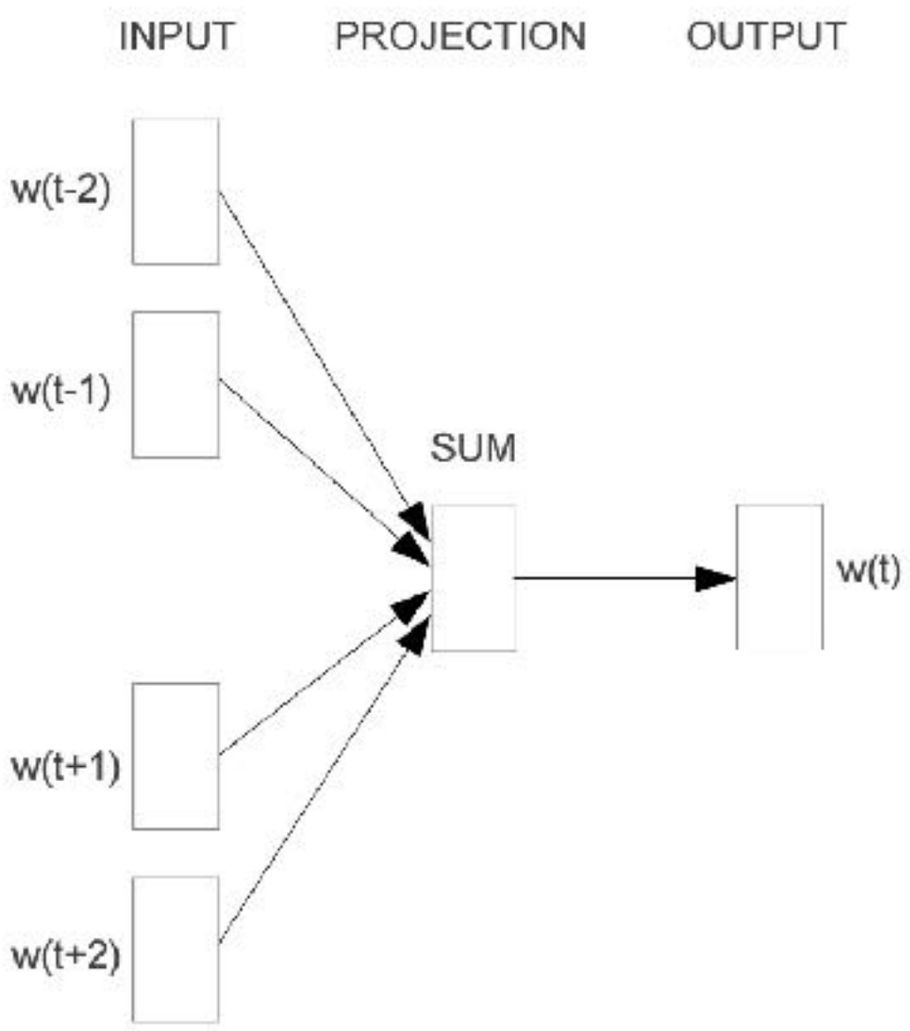
$$p(S)=p(w_1,\ldots,w_m)=\prod_{i=1}^{m}p(w_i \mid w_{1-(n-1)},\ldots,w_{i-1})$$

预测缺失词：我今天上课迟到了，老师批评了_____。

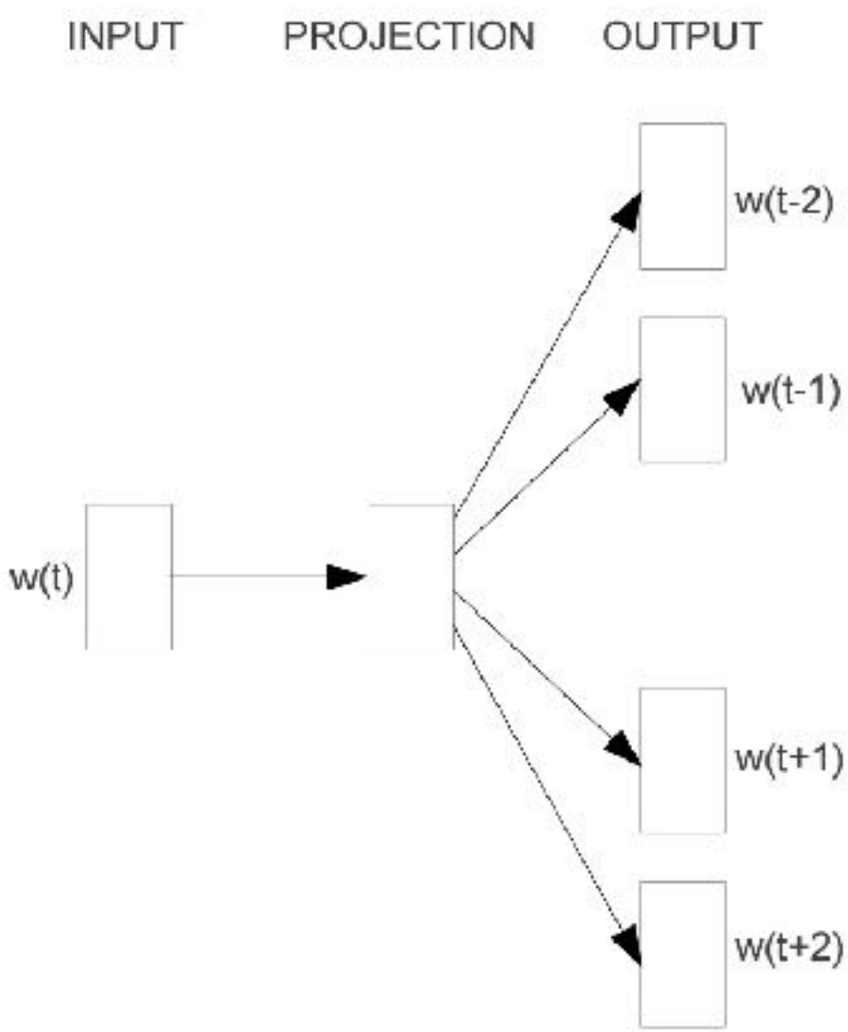


实际中N越大越好，但受限于复杂度，一般的N取2（bigram）或者3（trigram）。Google层使用过5-Gram模型，这是目前使用过几乎最大的N-Gram模型。

word2vec

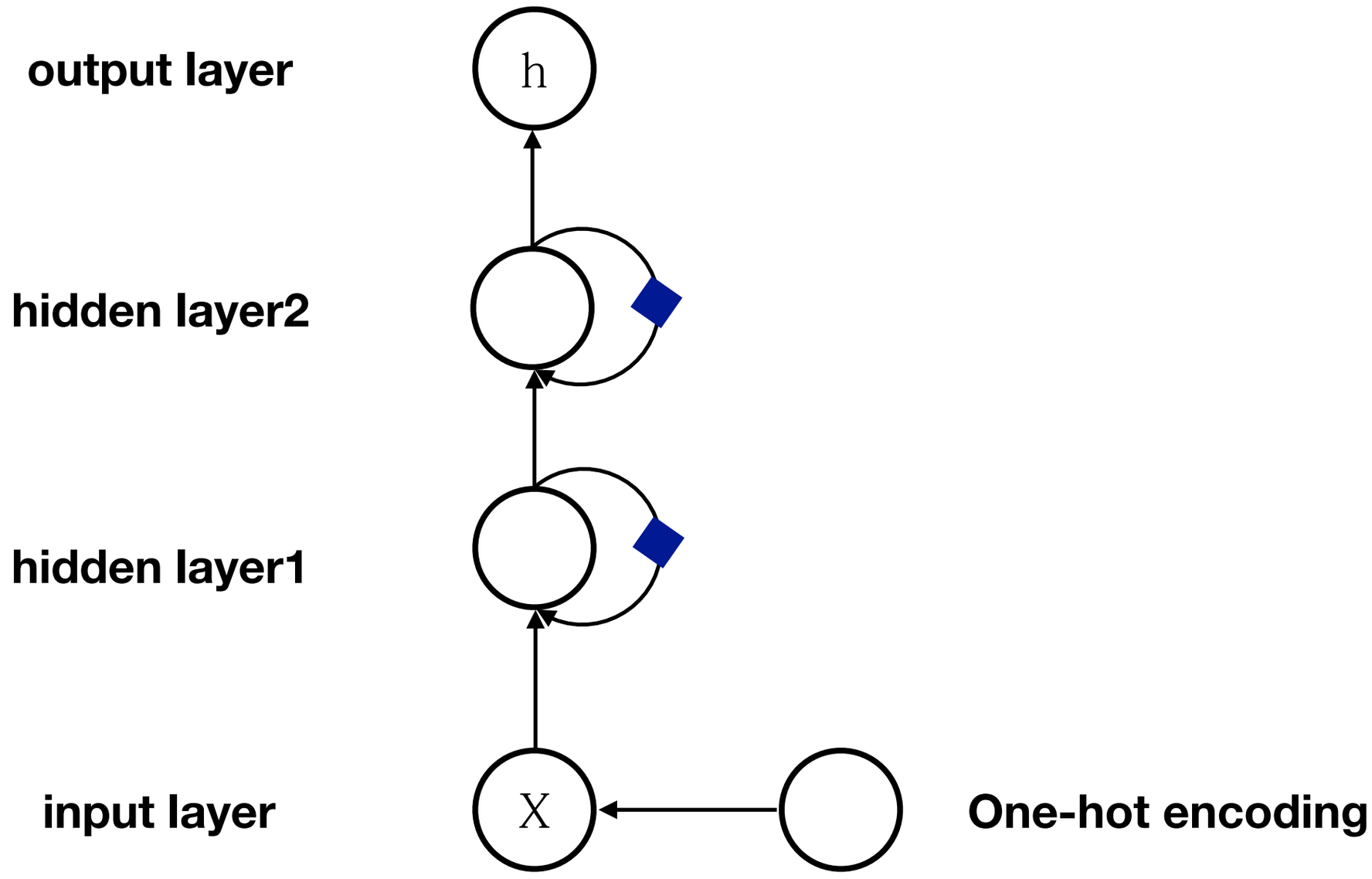


CBOW



Skip-gram

使用RNN构建语言模型



使用RNN构建语言模型

训练集：规模很大的语料库。 ⟶ 生成**词汇表** [今天、明天、批评、...]

我今天上学迟到了，老师批评了我。

↓ **分词**

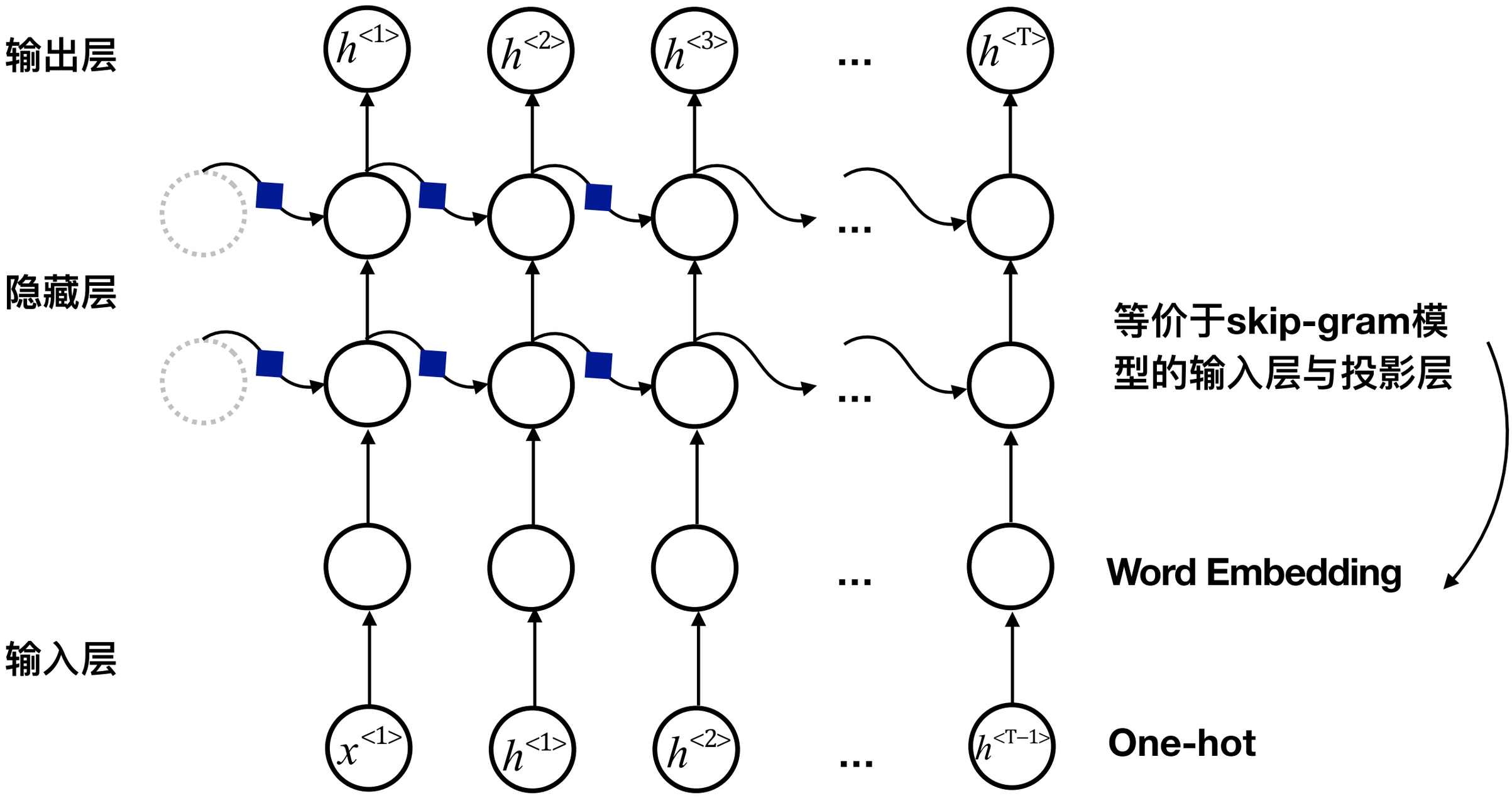
我 / 今天 / 上学 / 迟到 / 了/，老师 / 批评 / 了 / 我 / <EOS>。

↓ **转为one-hot编码**

$$\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\quad\cdots\quad\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}\begin{bmatrix}0\\0\\\vdots\\1\\0\\\vdots\end{bmatrix}$$

根据词汇表进行转换，词汇表中不存在的此使用<UNK>对应的编码替换，结尾加上<EOS>编码表示句子结束。

使用RNN构建语言模型



TensorFlow实现

RNN语言模型

6.3 使用LSTM

生成诗歌