

□ 为什么有BP神经网络，CNN，还要RNN？

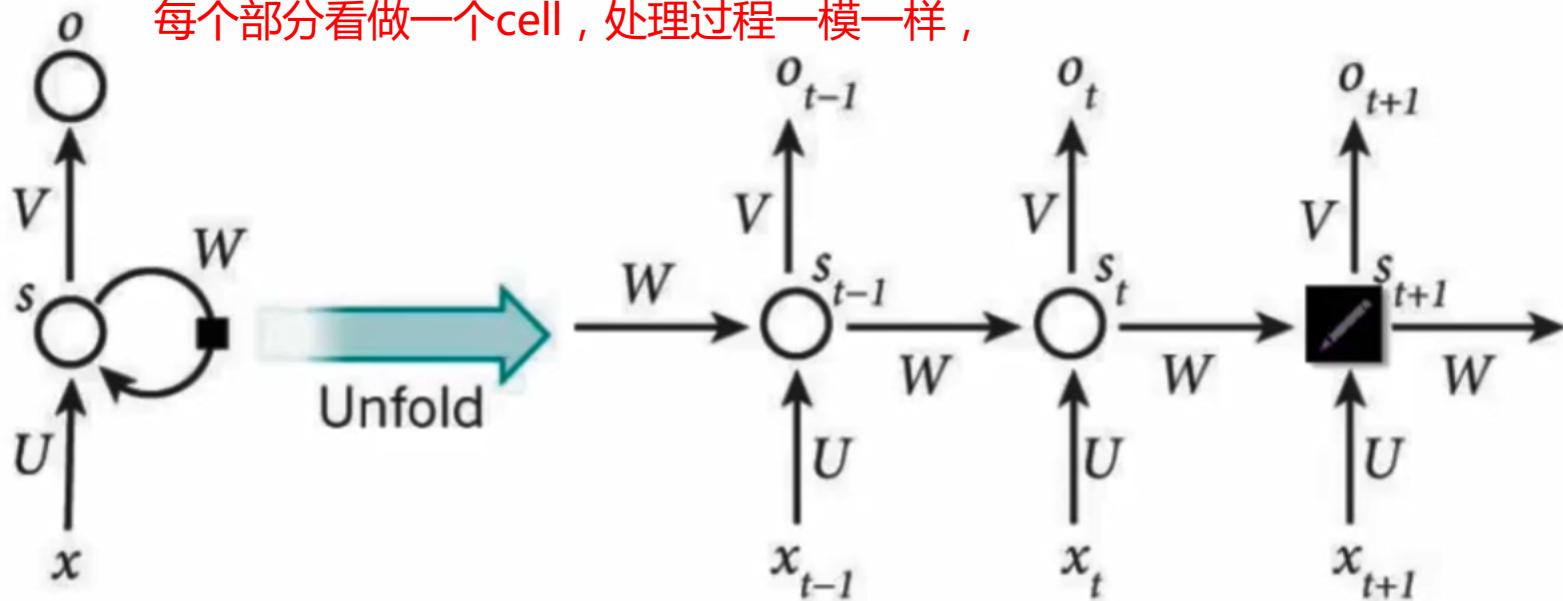
- 传统神经网络(包括CNN)，输入和输出都是互相独立的。
有些任务用人工神经网络、CNN解决不了
 - 图像上的猫和狗是分隔开的，但有些任务，后续的输出和之前的内容是相关的。
 - “我是中国人，我的母语是_____”
自然语言处理任务中，输入和输出之间不独立
- RNN引入“记忆”的概念
变更的只是输入或者memory
 - 循环2字来源于其每个元素都执行相同任务。
 - 但是输出依赖于 输入 和 “记忆”

做预测不只依赖于input，
还依赖于之前的一部分信息，会把它存在memory

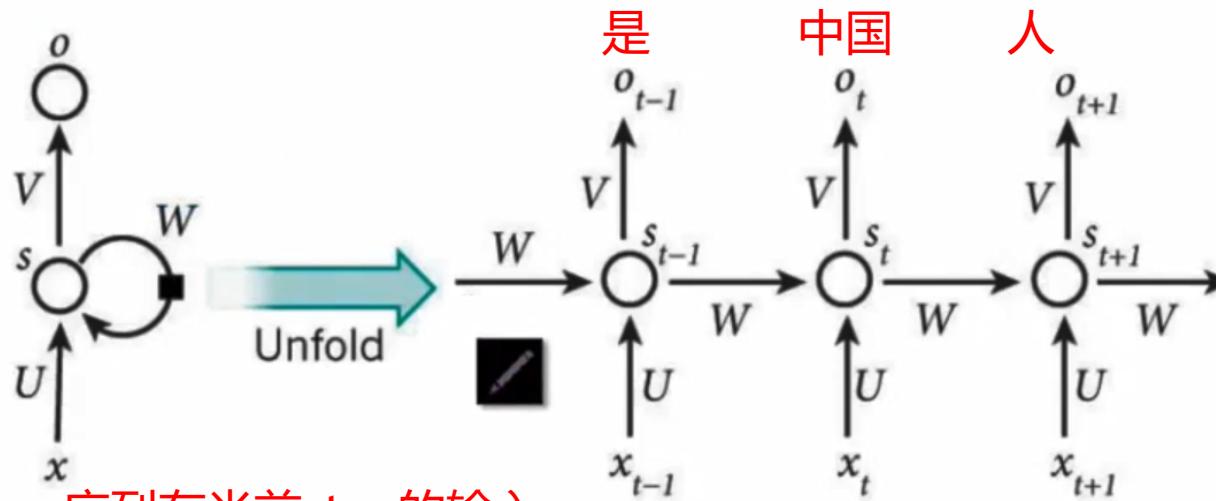
RNN的结构

□ 简单来看，把序列按时间展开

为了体现RNN的循环性，可以将多层fold起来，
每个部分看做一个cell，处理过程一模一样，



背景：要完成一个任务(Language model)：一句话知道出现的若干个词情况下，
出现下一个词会出现什么。
(有一个序列，在这个序列前n个element知道情况下去推断下一个element)



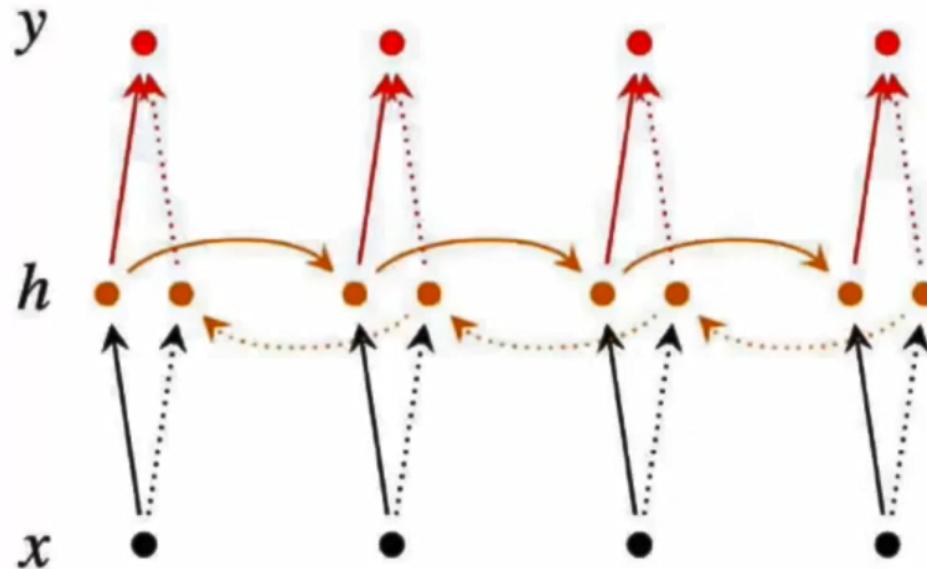
- 序列在当前step的输入 我 是 中国 人
- X_t是时间t处的输入
- S_t是时间t处的“记忆”， S_t=f(UX_t+WS_{t-1})， f可以是tanh等
- O_t是时间t处的输出，比如是预测下个词的话，可能是softmax输出的属于每个候选词的概率， O_t = softmax(VS_t)

- 可以把隐状态 S_t 视作“记忆体”，捕捉了之前时间点上的信息。
- 输出 O_t 由当前时间及之前所有的“记忆”共同计算得到。
- 很可惜，实际应用中， S_t 并不能捕捉和保留之前所有信息（记忆有限？）
矩阵 S_t 维度 容量有限，
- 不同于 CNN，这里的 RNN 其实整个神经网络都共享一组参数 (U, V, W) ，极大减小了需要训练和预估的参数量
- 图中的 O_t 在有些任务下是不存在的，比如文本情感分析，其实只需要最后的 output 结果就行

文本分类，在当前任何部分都可以做总结，输出结果，但是不一定准确。任务是需要看完整个文本给出一个结果。

□ 双向RNN

- 有些情况下，当前的输出不只依赖于之前的序列元素，还可能依赖之后的序列元素
- 比如从一段话踢掉部分词，让你补全
- 直观理解：双向RNN叠加



不一样的W和V让它捕捉更多信息

$$\vec{h}_t = f(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b})$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\vec{h}_t; \overleftarrow{h}_t] + c)$$

两个memory做拼接

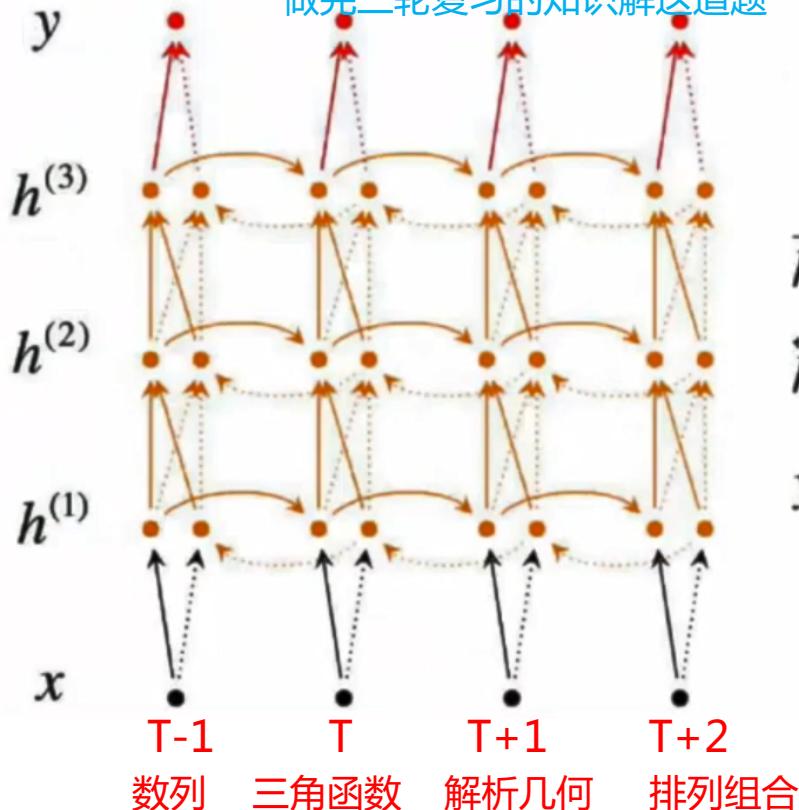
深层双向RNN

Deep : 五年级的知识变难了，一周内掌握不了 → 多思考一下



和双向RNN的区别是每一步/每个时间点我们设定多层
结构

做完三轮复习的知识解这道题



同一章前一轮复习

同一轮复习前一章

$$\vec{h}_t = f(\vec{W} \vec{h}_t^{(i-1)} + \vec{V} \vec{h}_{t-1}^{(i)} + \vec{b})$$

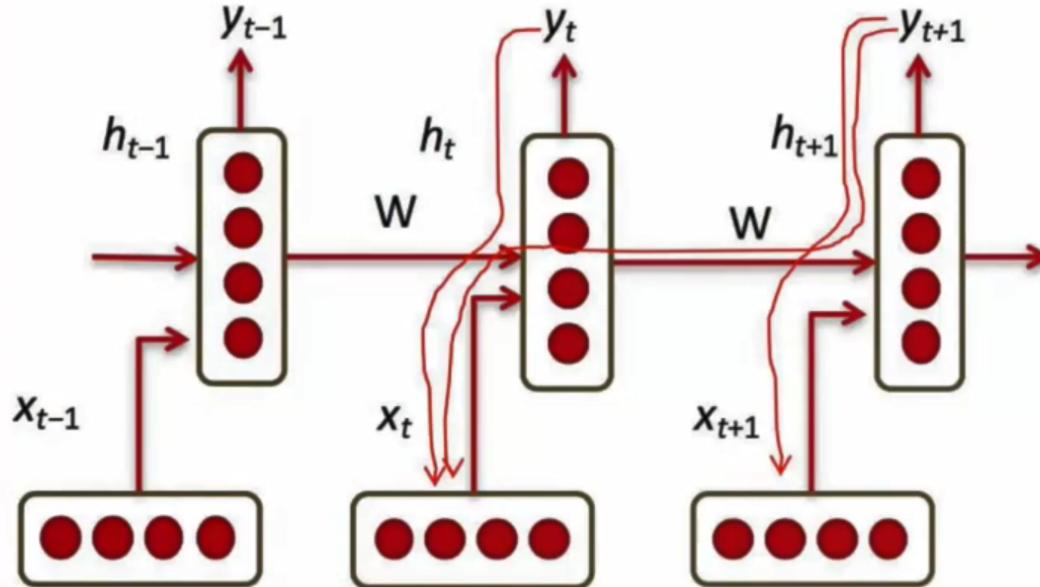
$$\overset{\leftarrow}{h}_t = f(\overset{\leftarrow}{W} \overset{\leftarrow}{h}_t^{(i-1)} + \overset{\leftarrow}{V} \overset{\leftarrow}{h}_{t+1}^{(i)} + \overset{\leftarrow}{b})$$

$$y_t = g(U[\vec{h}_t; \overset{\leftarrow}{h}_t] + c)$$

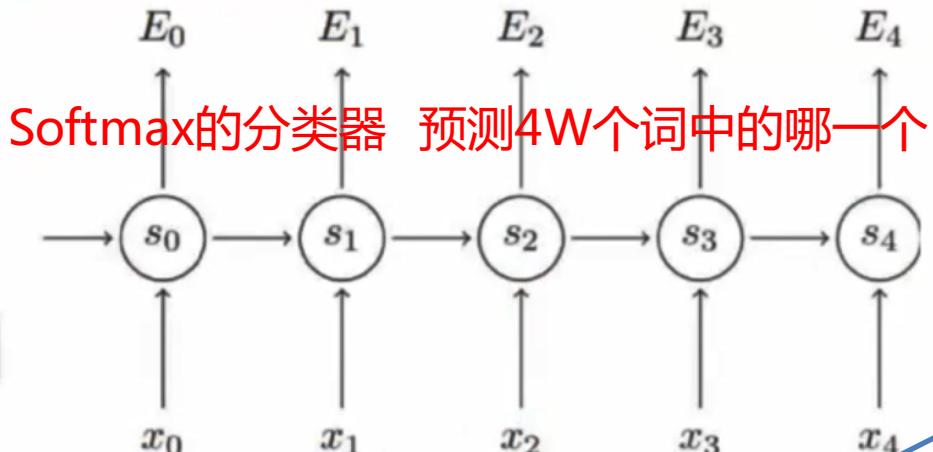
RNN与BPTT算法



- MLP(DNN)与CNN用BP算法求偏导
- BPTT和BP是一个思路，只不过既然有step，就和时间t有关系



BPTT(BackPropagation Through Time) 基于时间的反向传播 (调参)



我们的目标是计算误差关于参数U、V和W的梯度，然后使用梯度下降法学习出好的参数。

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t \quad \text{交叉熵损失 cross entropy}$$

每一个时间点都有一个输出 每个输出都可以去计算loss

$$\begin{aligned} E(y, \hat{y}) &= \sum_t E_t(y_t, \hat{y}_t) \\ &= -\sum_t y_t \log \hat{y}_t \end{aligned}$$

完成整个句子预测：沿着时间轴把每个位置的 loss加在一起

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}.$$

但是 $s_3 = \tanh(Ux_t + Ws_2)$ 依赖于 s_2

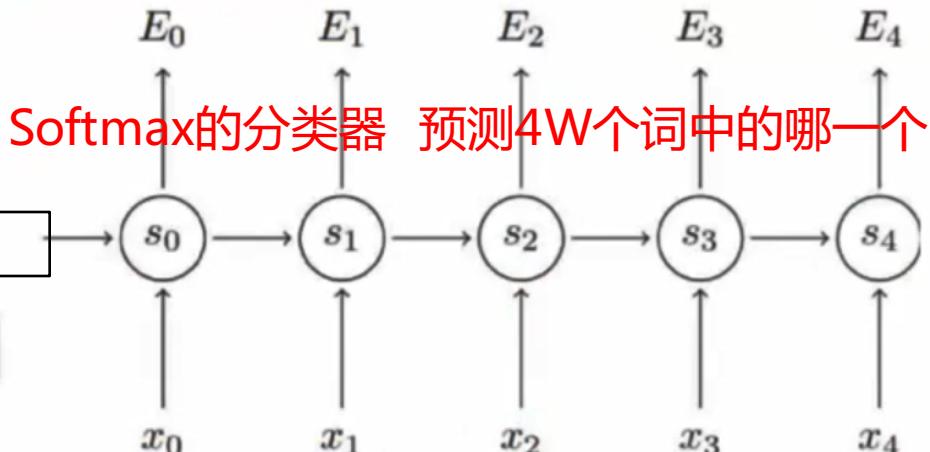
↓
链式法则

$$\frac{\partial E}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

梯度下降，使loss值最小
要求我们去求
lossfunction对于W的偏导

求偏导的过程在这一层完成不了，必须用
BPTT，不能用BP

BPTT：沿着时间轴往前追溯



$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t \quad \text{交叉熵损失 cross entropy loss}$$

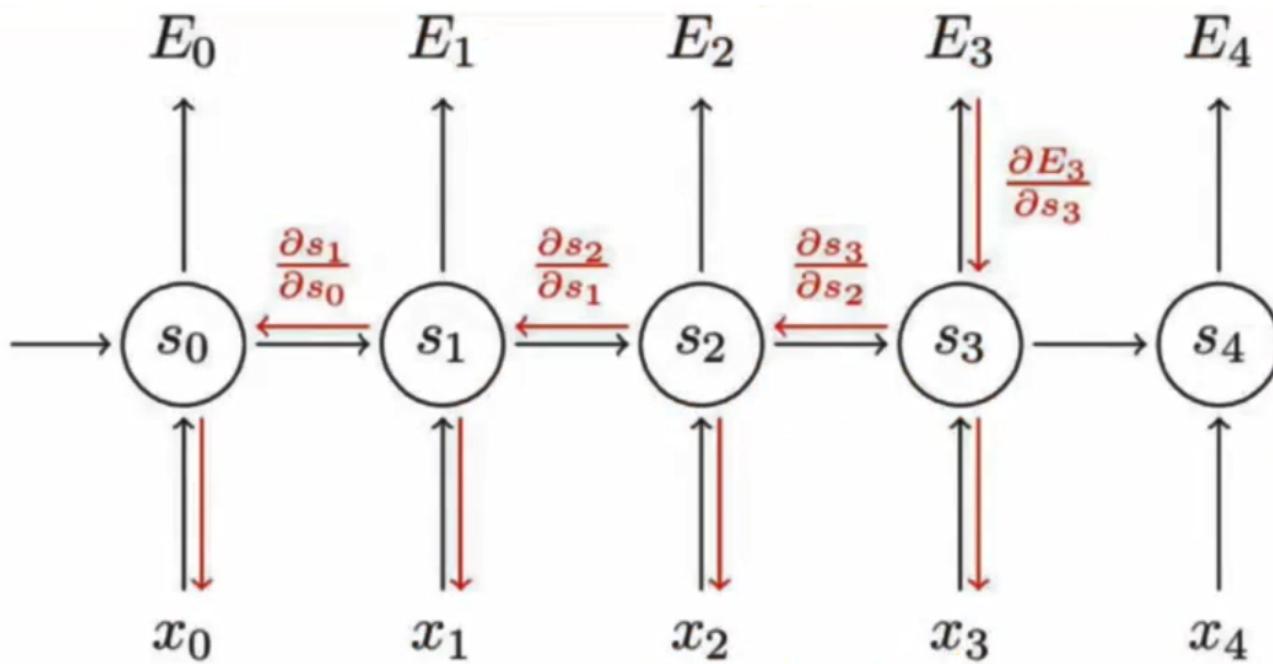
$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}$$

但是 $s_3 = \tanh(Ux_t + Ws_2)$ 依赖于 s_2

↓ 链式法则

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$



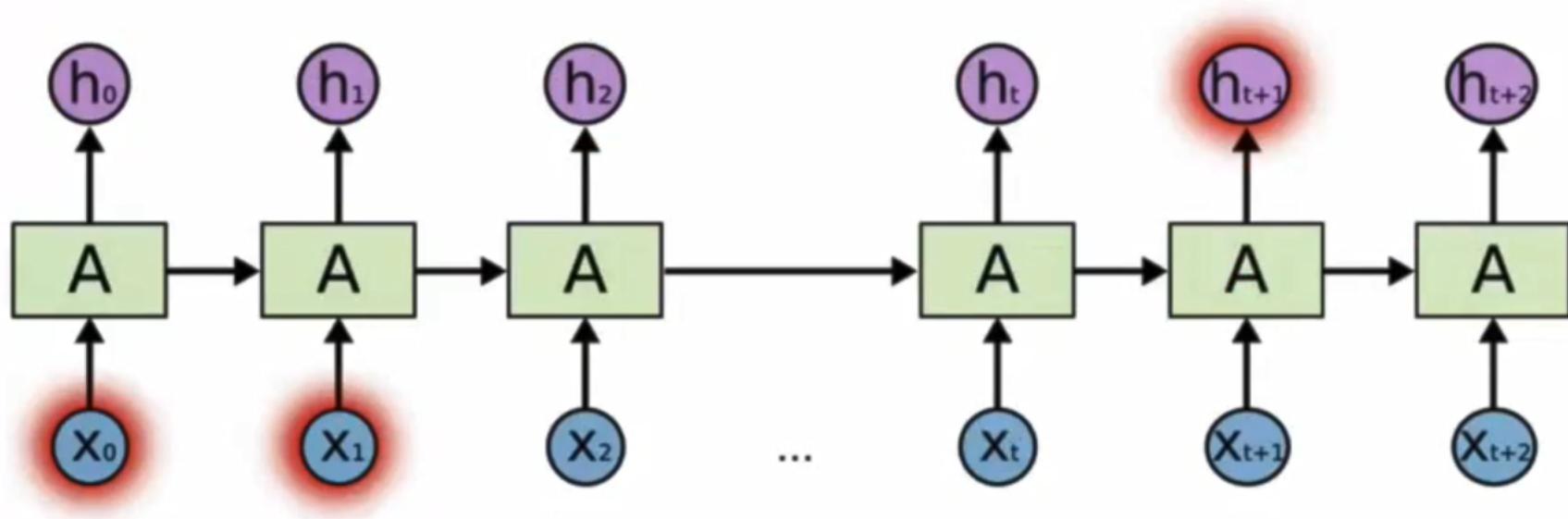
$$\frac{\partial E_3}{\partial W} = \sum_{k=0} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

链式法则

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^3 \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$



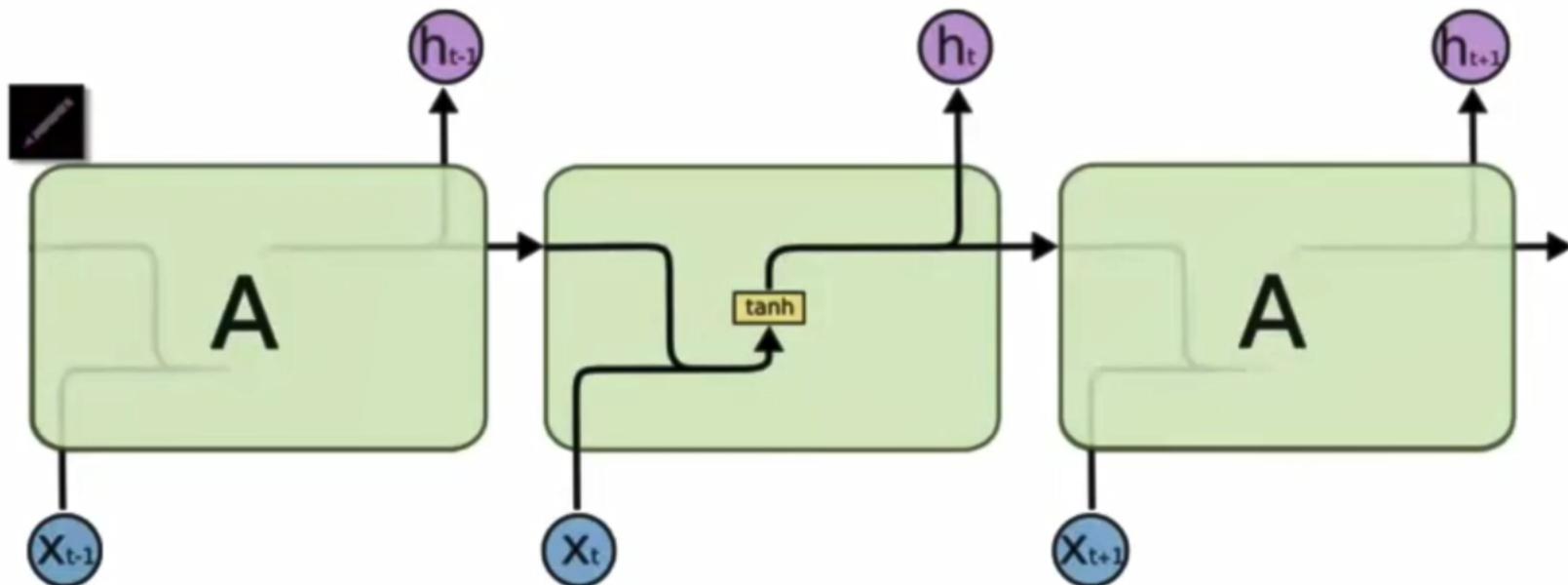
- 前面提到的RNN解决了，对之前的信息保存的问题
- 但是！存在长期依赖的问题。
 - 看电影的时候，某些情节的推断需要依赖很久以前的一些细节。
 - 很多其他的任务也一样。
 - 很可惜随着时间间隔不断增大时，RNN 会丧失学习到连接如此远的信息的能力。
- LSTM是RNN一种，大体结构几乎一样。区别是？
 - 它的“记忆细胞”改造过。
 - 该记的信息会一直传递，不该记的会被“门”截断。

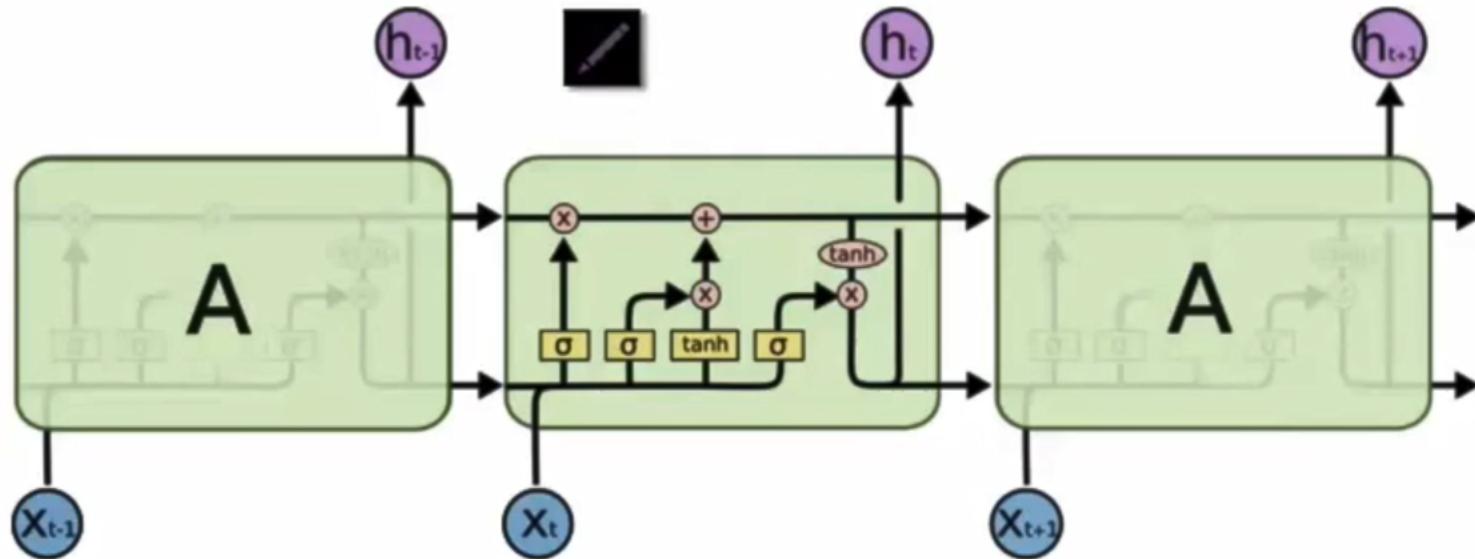


RNN可以被训练来，通过前面的单词来预测接下来的单词。

实际上，相关信息和需要该信息的位置之间的距离可能非常的远。

不幸的是，随着距离的增大，RNN对于如何将这样的信息连接起来无能为力。





Neural Network
Layer

非线性处理模块

Pointwise
Operation

逐点运算

Vector
Transfer

信息传
播方向

Concatenate

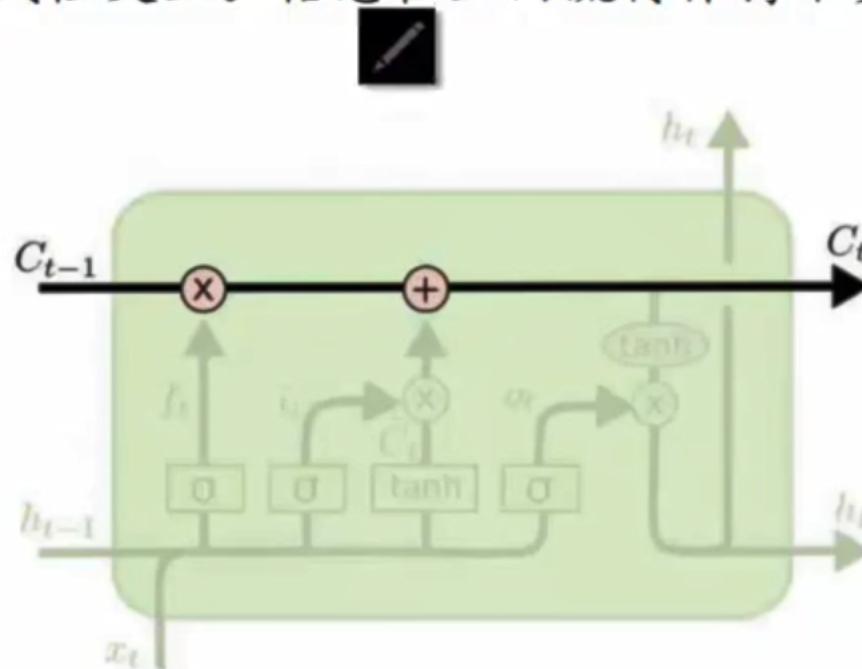
信息拼接

Copy

信息复制

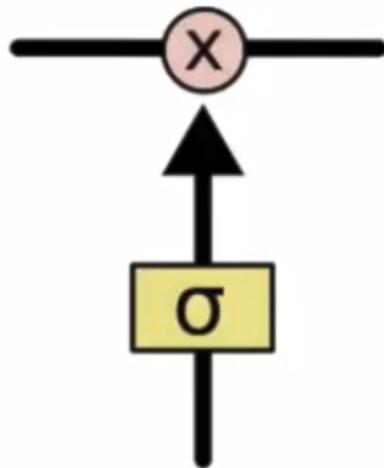
□ LSTM关键：“细胞状态” Cell State

□ 细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。



T-1时刻的记忆到现在的记忆 在传送带上往前传 发生信息的交互 可以取东西 也可以放东西上去

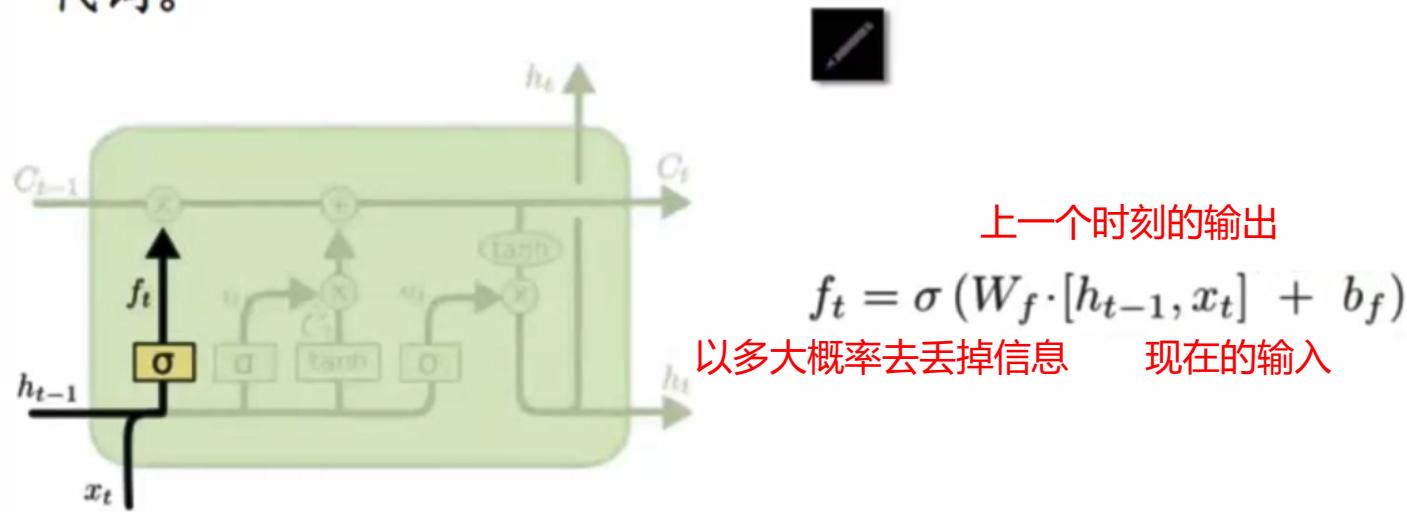
- LSTM怎么控制“细胞状态”？
 - 通过“门”让信息选择性通过，来去除或者增加信息到细胞状态
 - 包含一个sigmoid神经网络层 和 一个pointwise乘法操作
 - Sigmoid 层输出0到1之间的概率值，描述每个部分有多少量可以通过。0代表“不许任何量通过”，1就指“允许任意量通过”



做记忆的变更：要把这部分
记忆存下去

LSTM的几个关键“门”与操作

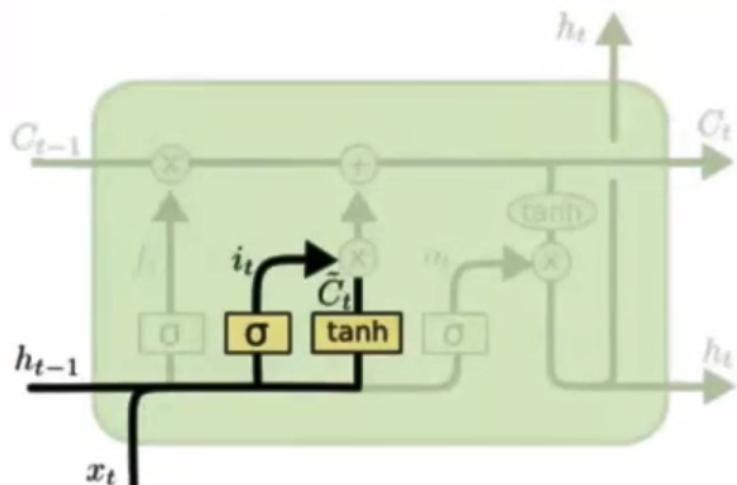
- 第1步：决定从“细胞状态”中丢弃什么信息 => “忘记门”
- 比如完形填空中填“他”或者“她”的问题，细胞状态可能包含当前主语的类别，当我们看到新的代词，我们希望忘记旧的代词。



LSTM的几个关键“门”与操作

□ 第2步：决定放什么新信息到“细胞状态”中

- ① Sigmoid层决定什么值需要更新 
- ② Tanh层创建一个新的候选值向量 \tilde{C}_t
- ③ 上述2步是为状态更新做准备



(产生一个概率值，以多少值去更新信息) 用 i_t 对 C_t 做过滤，哪一部分知识能够补充到我之前的知识体系中

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

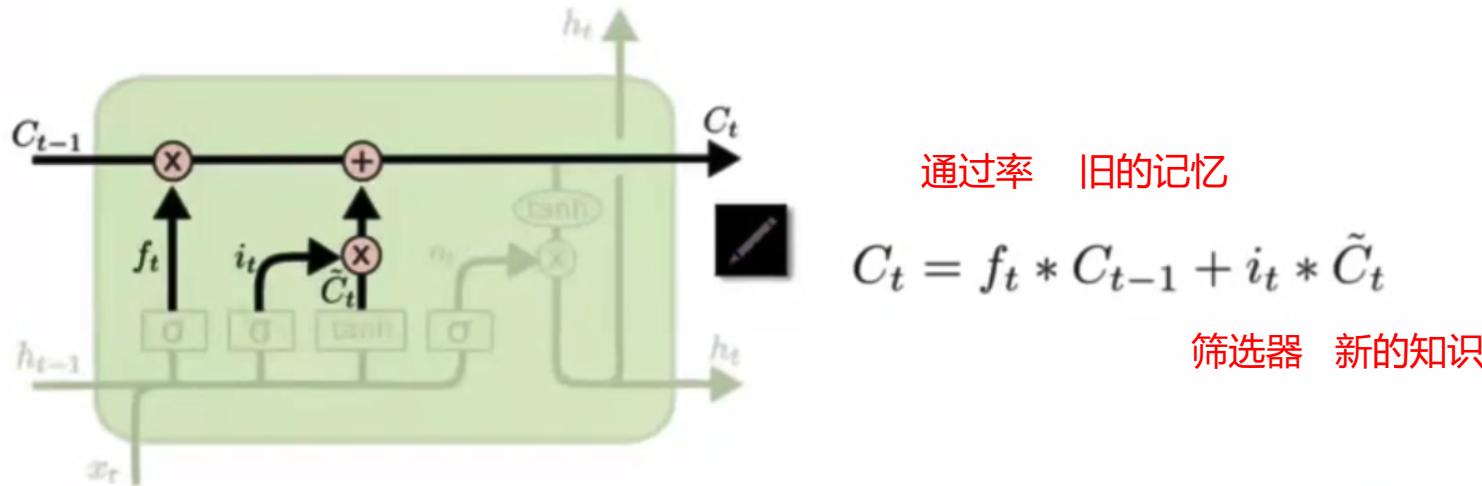
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

C_t :六年级这一年学到了什么

LSTM的几个关键“门”与操作

□ 第3步：更新“细胞状态”

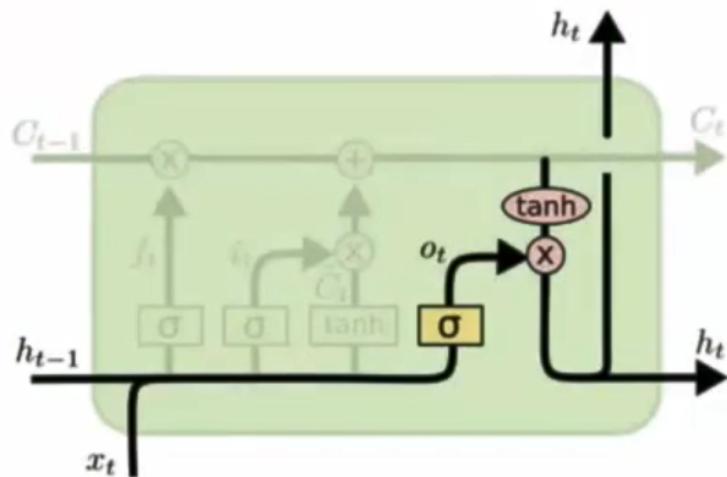
- ① 更新 C_{t-1} 为 C_t
- ② 把旧状态与 f_t 相乘，丢弃掉我们确定需要丢弃的信息
- ③ 加上 $i_t * \tilde{C}_t$ 。这就是新的候选值，根据我们决定更新每个状态的程度进行变化。



LSTM的几个关键“门”与操作

□ 第4步：基于“细胞状态”得到输出

- ① 首先运行一个sigmoid层来确定细胞状态的哪个部分将输出
- ② 接着用tanh处理细胞状态(得到一个在-1到1之间的值)，再将它和sigmoid门的输出相乘，输出我们确定输出的那部分。
- ③ 比如我们可能需要单复数信息来确定输出“他”还是“他们”



P(0,1) 从前六年的知识中筛选出来解决当前题目的知识
$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

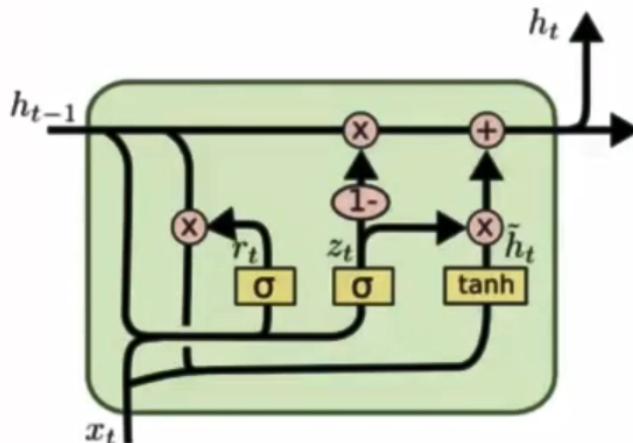
$$h_t = o_t * \tanh (C_t)$$
 小学全六年知识
解决了给出答案

LSTM的变种：GRU只有两个门，分别为更新门和重置门，即图中的 z_t 和 r_t

更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越少。重置门用于控制忽略前一时刻的状态信息的程度，重置门的值越小说明忽略得越多。

□ Gated Recurrent Unit (GRU), 2014年提出

- 将忘记门和输入门合成了一个单一的 更新门
- 同样还混合了细胞状态和隐藏状态，和其他一些改动。
- 比标准LSTM简单。



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

更新记忆

h' 主要包含了当前输入的 x^t 数据。把 h' 添加到当前的隐藏状态，相当于“记忆了当前时刻的状态”，类似于LSTM的选择记忆阶段。

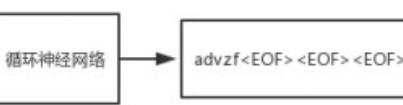
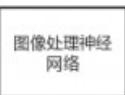
h_t : 忘记传递下来的 h_{t-1} 中的某些维度信息，并加入当前节点输入的某些维度信息

LSTM识别验证码

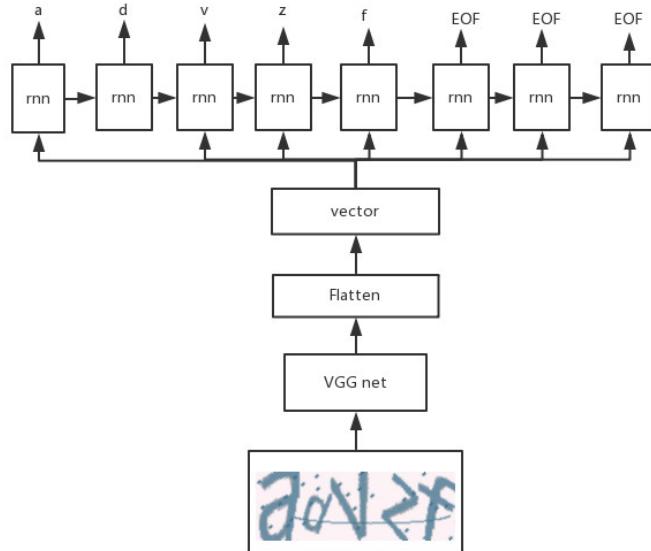
N=7+1 验证码最长为7位+1位表示结束



验证码尺寸: H.W.C = 60.250.



advzf <EOF> <EOF> <EOF>



encoder是Image, decoder是验证码序列。这里把decoder部分的输入用 encoder (image) 的最后一层复制N份作为decoder部分的每个cell的输入。

