北京大学

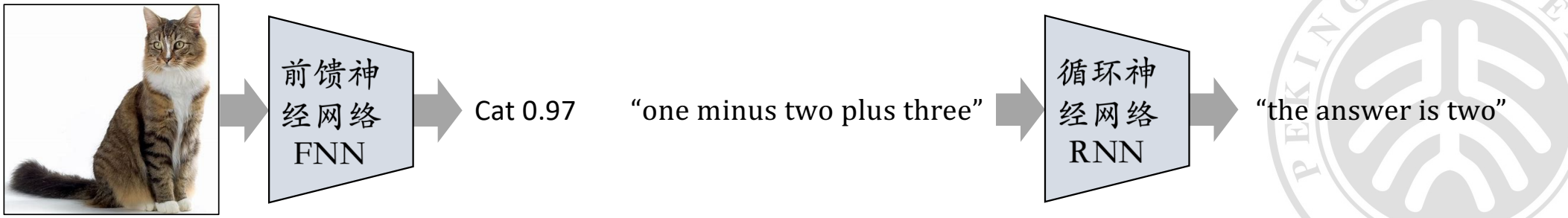
循环神经网络

Recurrent Neural Networks

主讲人：董豪　讲义：董豪

动机

- 多层感知器（MLP）和卷积神经网络（CNN）都是将一个数据样本作为输入，并输出一个结果，被归类为**前馈神经网络（FNN, forward neural network）**，因为它们只是逐层传递数据并为**每个输入获得一个输出**
- 有许多**时间序列（time-series）数据集**，例如语言、视频和生物信号，无法适应这个框架
- **循环神经网络（RNN，recurrent neural network）**是一种专为处理时间序列数据设计的扩展深度学习架构。



动机

Anti-spam
反垃圾邮件

Signal Analysis
信号分析

Language Translation
语言翻译

Image Captioning
图片描述

Chatbot
聊天机器人

Video Analysis
视频分析

Sentence Generation
句子生成

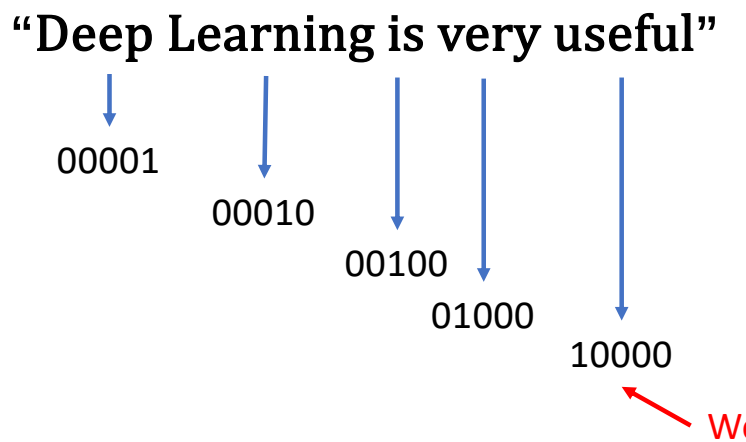
内容提要

- 词的表示 Word Representation
- 序列数据 Sequential Data
- 朴素循环神经网络 Vanilla Recurrent Neural Network
- 长短期记忆网络 LSTM Long Short-Term Memory
- 序列生成模型 RNNs are Generative Models
- 时间序列应用 Time-series Applications

Word Representation

词的表示

- **One-hot Vector**
  - 循环神经网络逐个接收单词，但让我们看看计算机中如何表示一个单词。
  - 简单方法1：One-hot Vector



缺点：

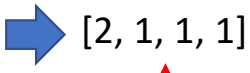
- 大词汇量将导致“维数灾难”
  （字典有多大，一个单词的向量就有多大）
- 所有单词表示都是独立的！太稀疏了！

词的表示

- 词袋模型 Bag of Words
  - 简单方法2：词袋模型
    - 使用单词频率来表示句子

“we like PKU, do we?”

Word	Frequency
we	2
like	1
PKU	1
do	1



[2, 1, 1, 1]

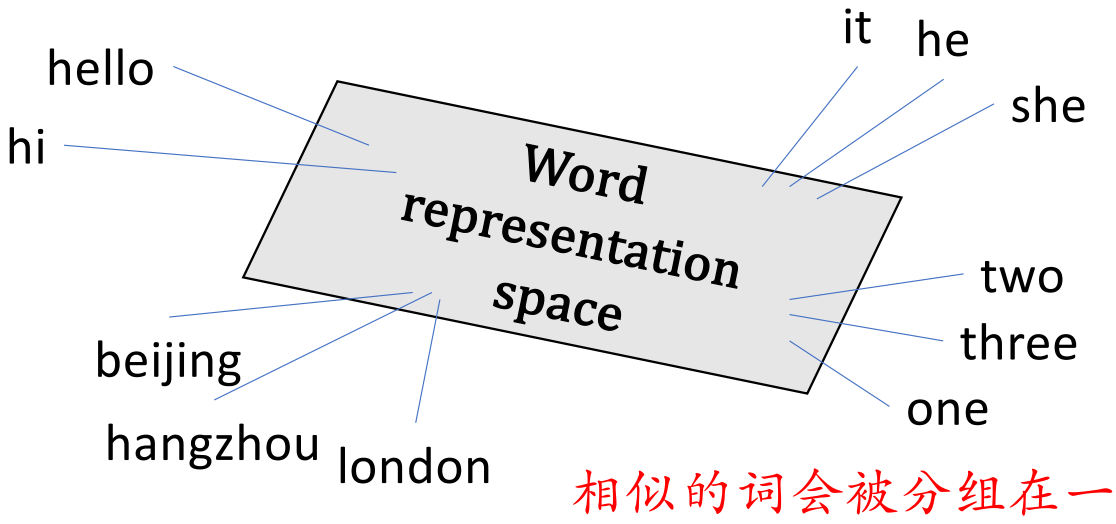
Sentence (text) representation

缺点：

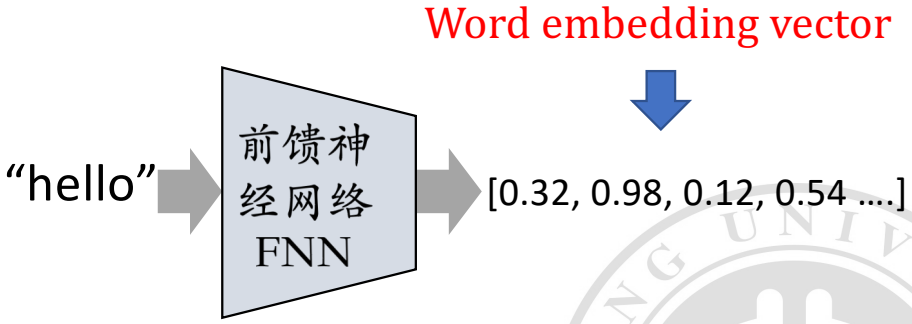
- 大词汇量将导致“维数灾难”
  （字典有多大，一个句子的向量就有多大）
- 丢失了单词在句子中的位置信息

词的表示

- 词嵌入 Word Embedding
  - 用一组浮点数向量来表示一个单词。



相似的词会被分组在一起



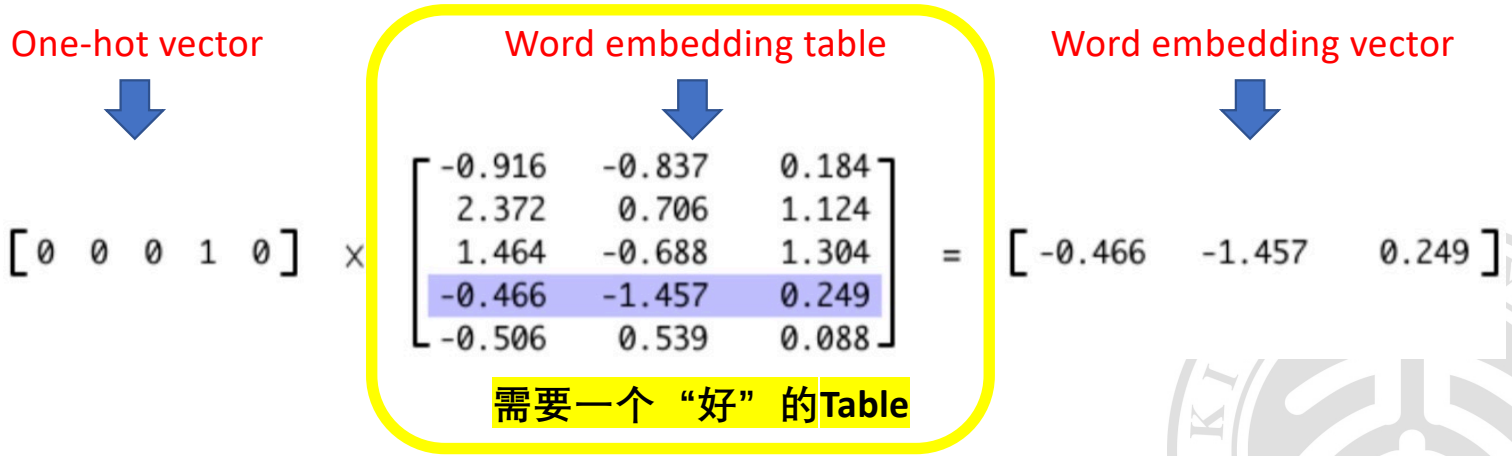
词的表示

- Word Embedding

- 给定一个包含5个单词的词汇表，我们可以创建一个词嵌入表，使得每个单词具有3个特征值。

Each word has a unique ID

Word	ID (Row Index)
---	0
deep	1
learning	2
is	3
popular	4



在实际操作中，为了节省时间，我们不会将独热向量与词嵌入表相乘。而是直接使用单词ID作为行索引，从表中查找相应的嵌入向量。

词的表示

- **理想的词嵌入**
  - 低维度 == 用高级特征表示单词
  - 包含单词的语义信息

相似的单词，如“猫”-“老虎”和“水”-“液体”，应该具有相似的词表示。.

语义信息允许进行语义操作：

$$\text{King} - \text{Man} + \text{Women} = \text{Queen}$$
$$\text{Paris} - \text{France} + \text{Italy} = \text{Rome}$$

词嵌入中的特征包含了诸如“性别”和“位置”等信息。

词的表示

- 习得词嵌入
  - 现有的算法通过阅读大量文本文档来学习词嵌入表，以发现其中的模式，这是一种**自监督**（self-supervised）学习方法。

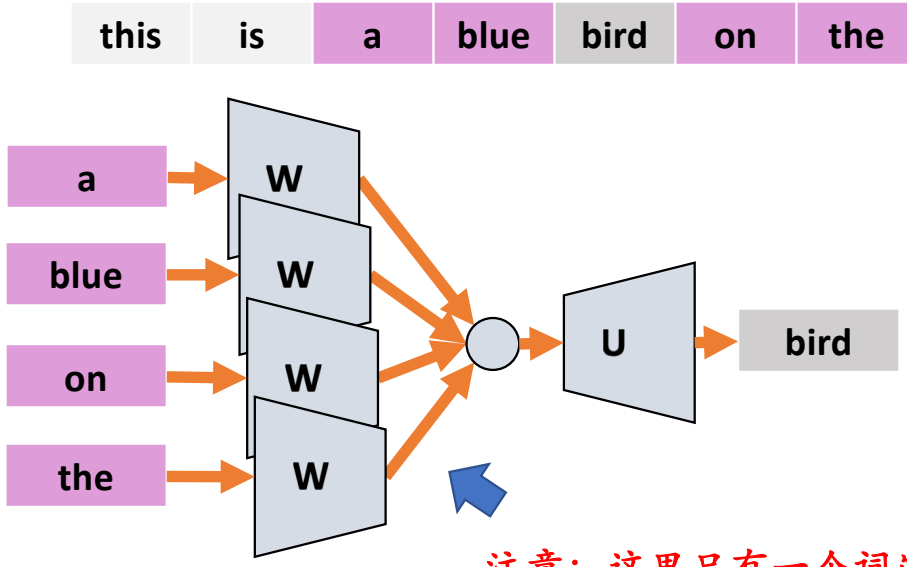
- 使用文本文档作为训练数据，无需任何标签。
- 通过比较上下文来找到相似的单词。

- This is a blue bird
- This is a yellow bird
- This is a red bird
- This is a green bird

由于“颜色”类单词位于句子的相似位置，我们可以将“颜色”类单词分组在一起。

- Word2Vec
  - Google 2013
  - Word2Vec = Skip-Gram/CBOW+ Negative Sampling

- Word2Vec
  - 连续词袋模型(Continuous Bag-of-Words，CBOW): 根据上下文预测中间的词。
    - 在“a blue bird on the tree”这句话里,“bird” 的上下文是 [“a” ,“blue” ,“on” ,“the”,“ tree”]



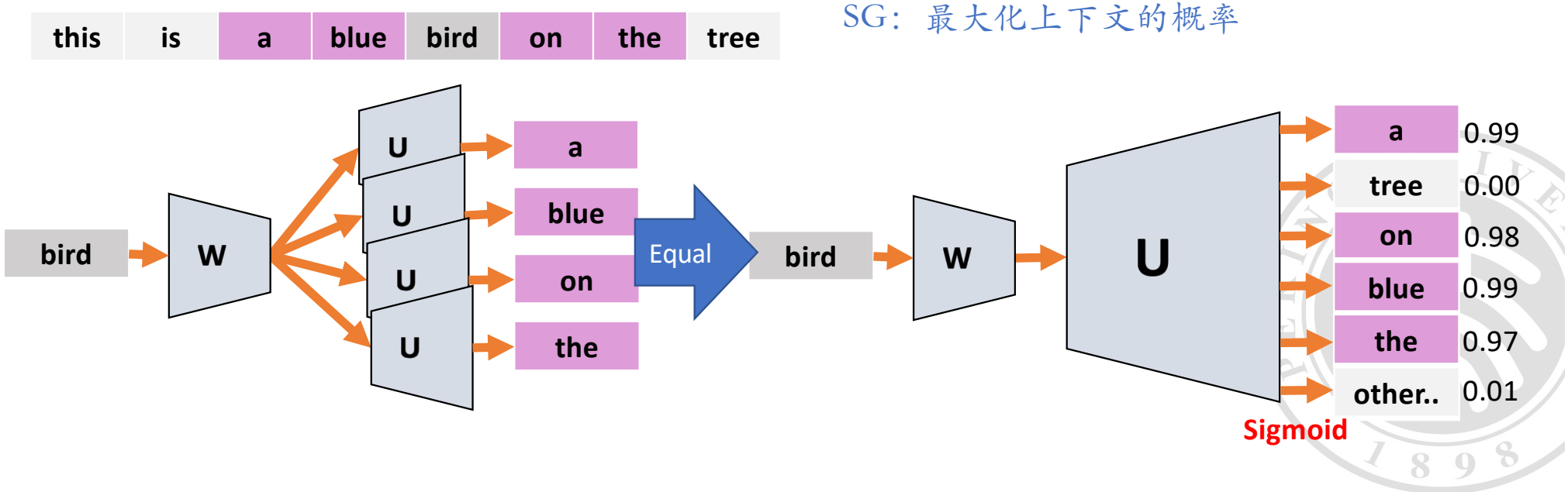
CBOW:最大化中间词的概率

在这个例子中，我们只考虑中间词左右各4个词，即窗口大小为2。

注意：这里只有一个词嵌入表，不同的输入重复使用相同的表。

- Word2Vec
  - Skip-Gram（SG）与CBOW相反，但目的相同
  - CBOW通过上下文预测中间词，而SG则**通过中间词预测上下文。**
  - 在句子“a blue bird on the tree”中，SG 的输入是“bird”，输出是are [“a”, “blue”, “on”, “the”, “tree”]

SG：最大化上下文的概率



- **噪声对比估计（NCE）**

  - Skip-Gram有多个目标输出，因此我们使用Sigmoid而不是Softmax。
    词汇表中的每个单词都被分为**正样本**和**负样本**，并且我们独立地对每个单词进行分类。
  - 大词汇表将导致巨大的计算成本
    我们使用负采样来加速损失函数的计算，从词汇表中随机采样N个负样本。
  - 这种方法被称为**噪声对比估计(Noise-Contrastive Estimation)** ：

$$E = -\Big(\sum_{i \in pos} \log(y_i) + \sum_{j \in neg} \log\big(1 - y_j\big)\Big)$$

Positive samples

Randomly select N negative samples

Sequential Data

序列数据

- 时间步 Time-Step

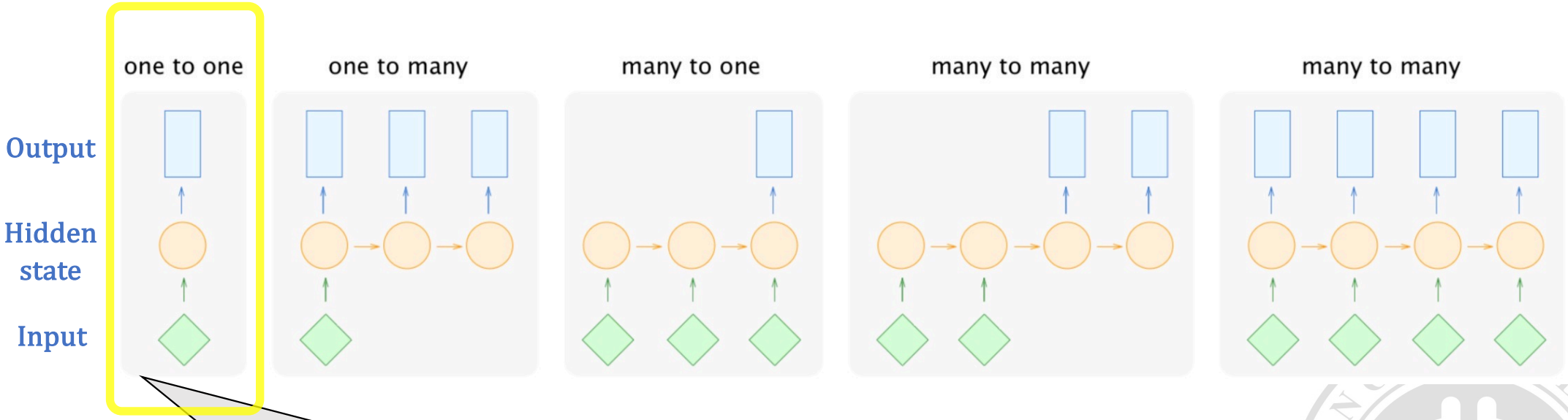
```
import torch

# 定义词汇表
vocab = ['a', 'blue', 'bird', 'on', 'the', 'tree']

# 将文本序列转换为单词id序列
text = 'a blue bird on the tree'
word_ids = [vocab.index(word) for word in text.split()]

# 定义one-hot张量
seq_len = len(word_ids)
vocab_len = len(vocab)
data_seq = torch.zeros(1, seq_len, vocab_len)
```

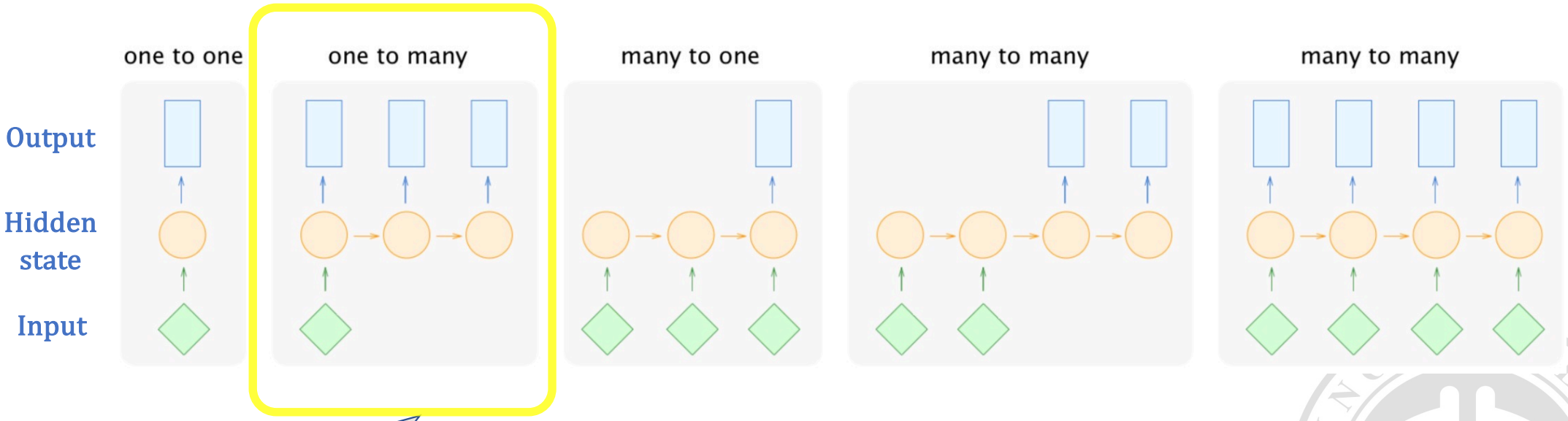
序列数据



Feedforward Neural Networks
前向神经网络

非时间序列问题
比如：图像分类、目标检测

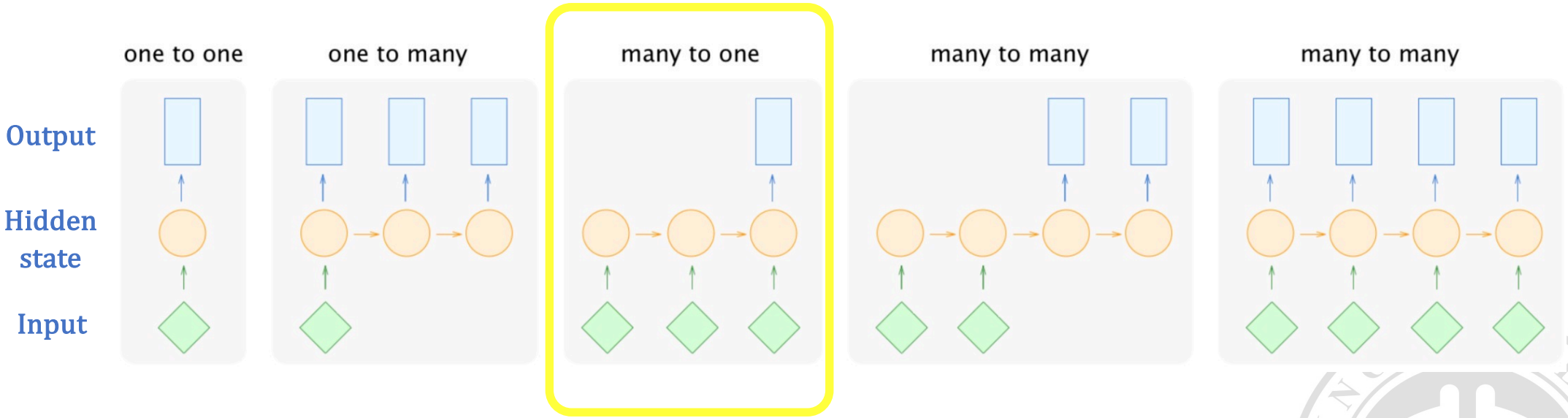
序列数据



输入一个数据，输出多个数据

图片描述：输入一张图片，生成一句话的描述

序列数据

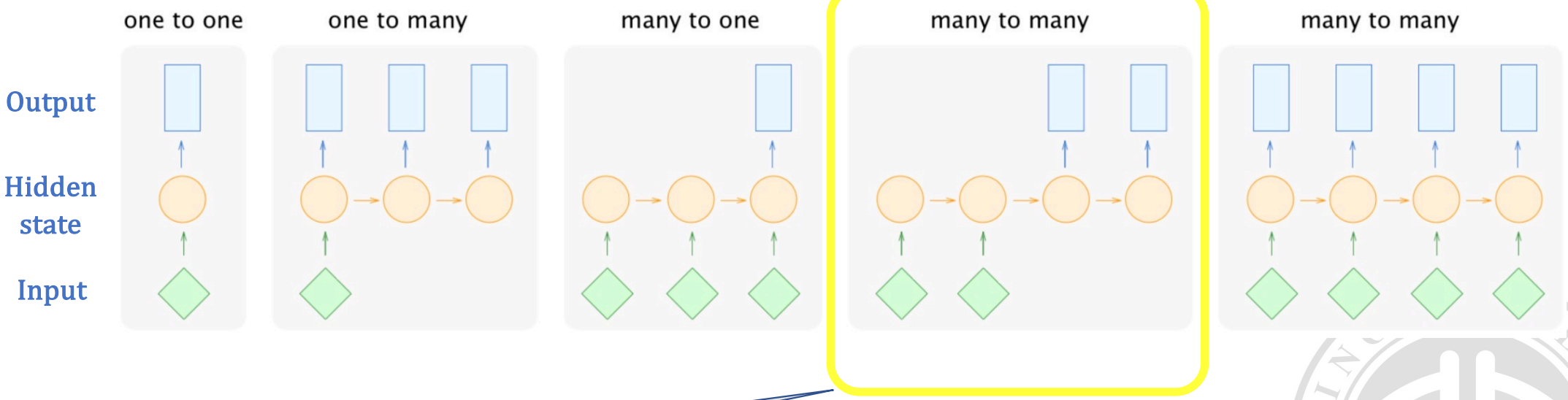


输入多个数据，输出一个数据

情感分类任务：输入一个有序的句子，输出表示幸福概率的数值。

序列数据

异步的
(Seq2Seq)



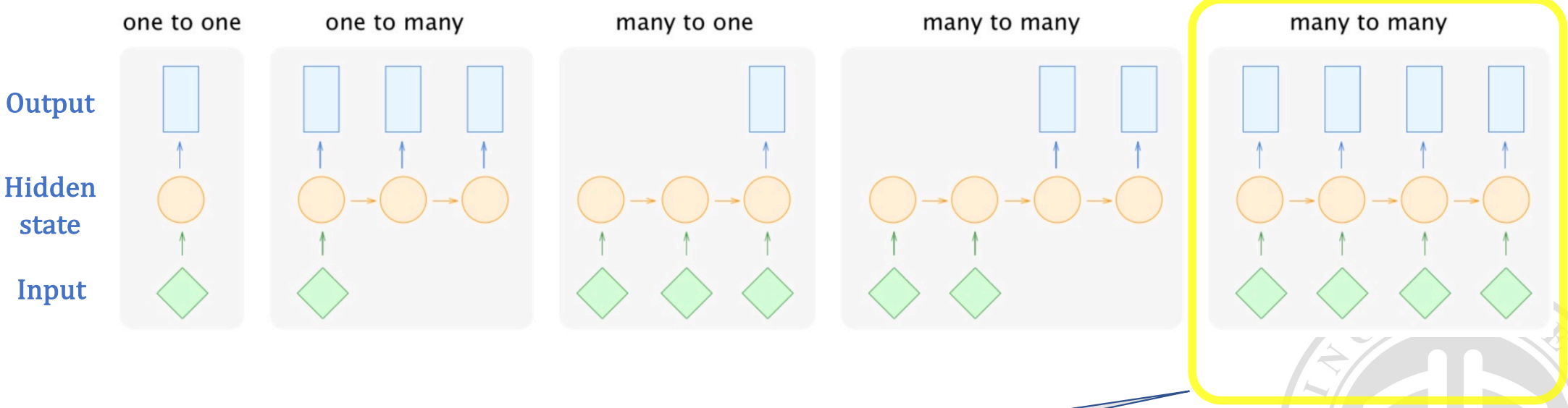
多数据输入和多数据输出

语言翻译：在开始生成翻译句子之前，将整个句子输入到模型中。

序列数据

异步的
(Seq2Seq)

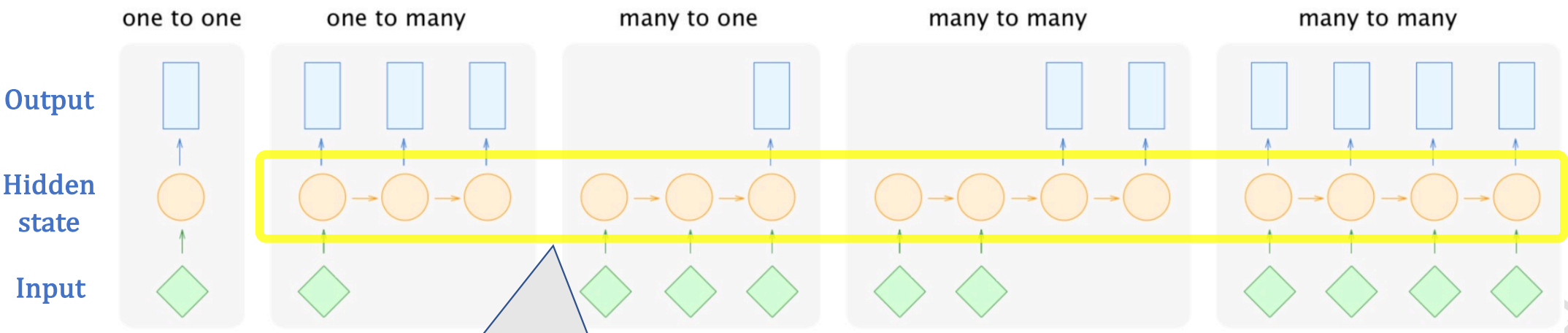
同步的
(simultaneous)



多个数据输入和多个数据输出

天气预测：在每个时间步（time-step）输入信息到模型中，并输出预测的天气状况。

序列数据

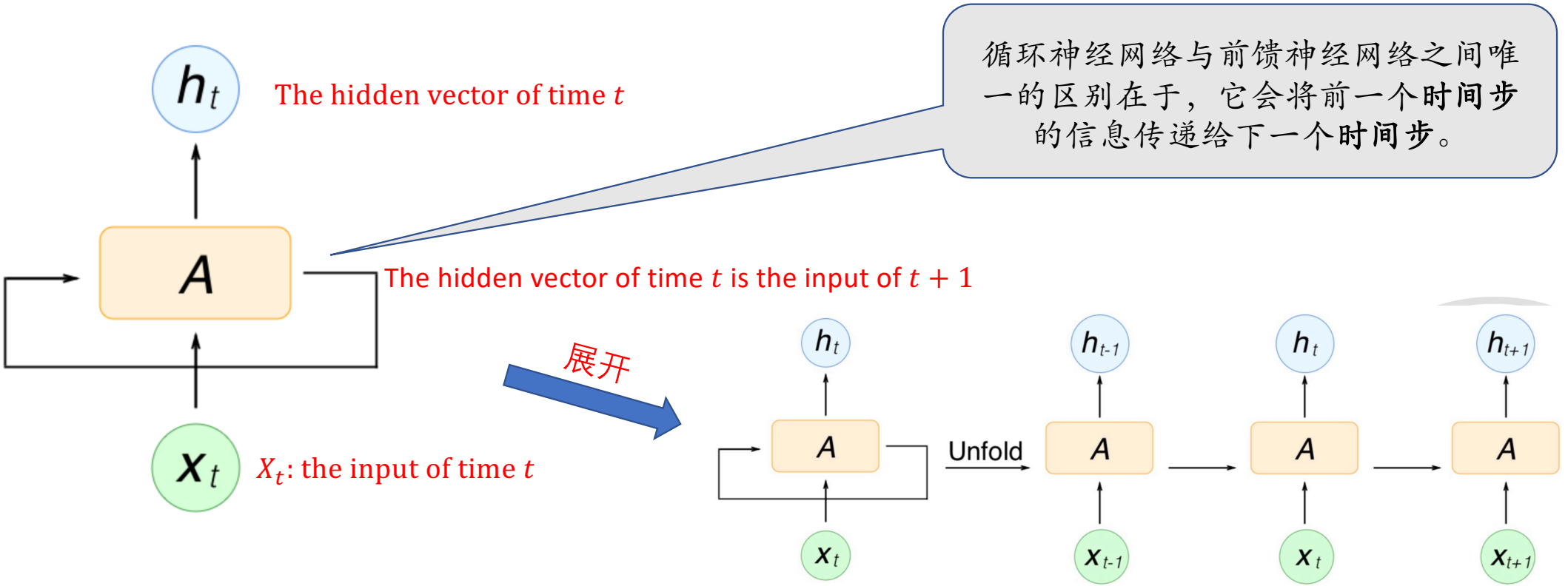


循环神经网络（Recurrent Neural Nets）
存储和处理时间信息。

Vanilla Recurrent Neural Network

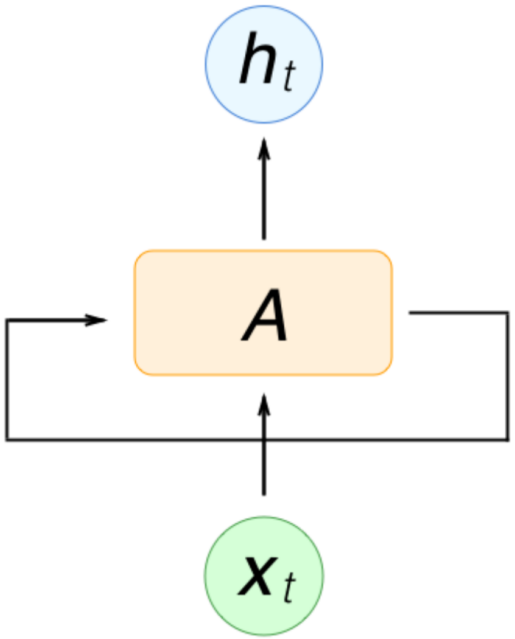
朴素循环神经网络

- Hidden Vector (State)



朴素循环神经网络

- 处理时间信息



Unrolled

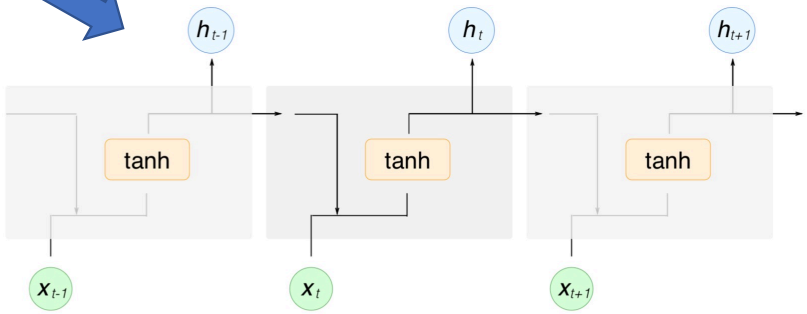
通过隐藏状态，将信息传递到未来

I am from China, so I speak Chinese

X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7

h_0 是一个全0向量，
h_{t-1} 包含了前一个时间步的信息.

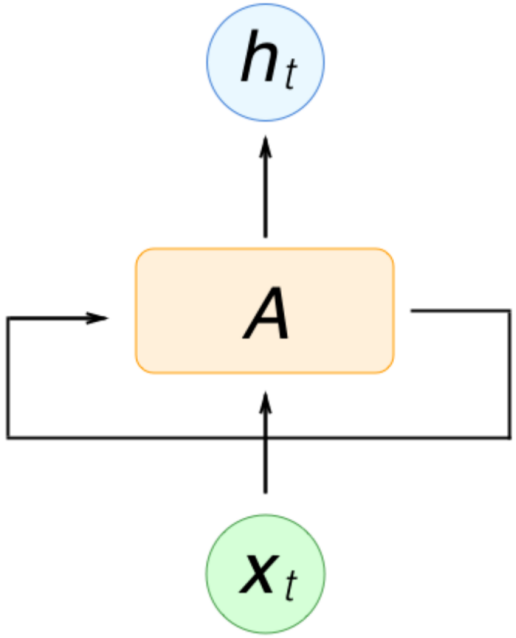
$$h_t = tanh\,(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$



W、U、b：权重矩阵和偏置
σ_h、σ_y：激活函数，σ_h 代表 $tanh$
y_t：时间步t的输出

朴素循环神经网络

- 局限性：长期依赖问题



I am from China, and I live in the UK and US for 10 years, my mother language is ____?____

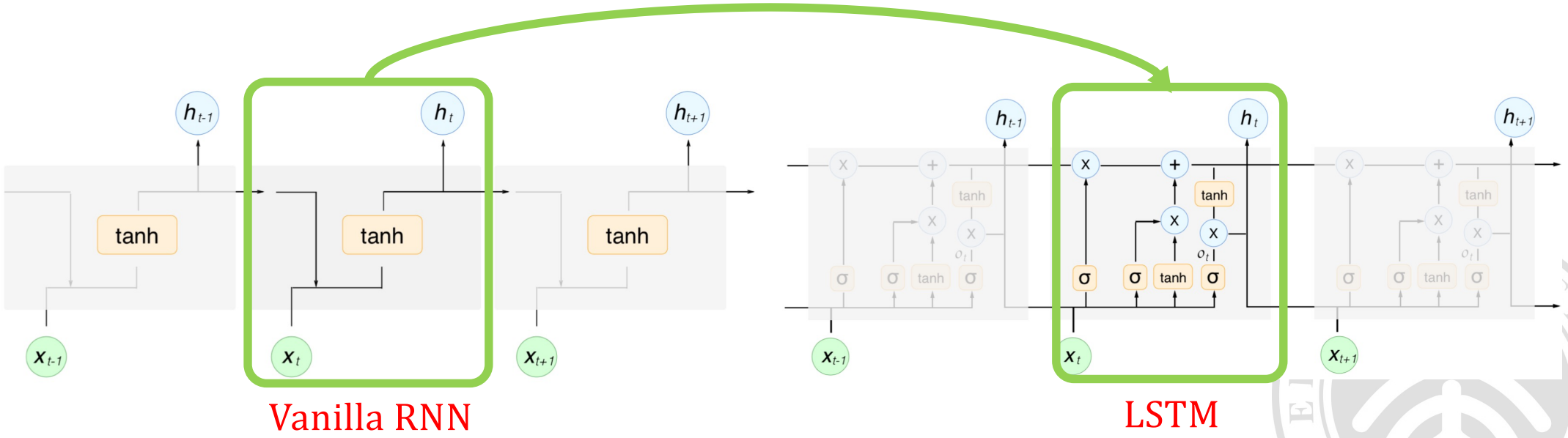
很难维持较为长期的信息

长短期记忆网络LSTM

Long Short-Term Memory

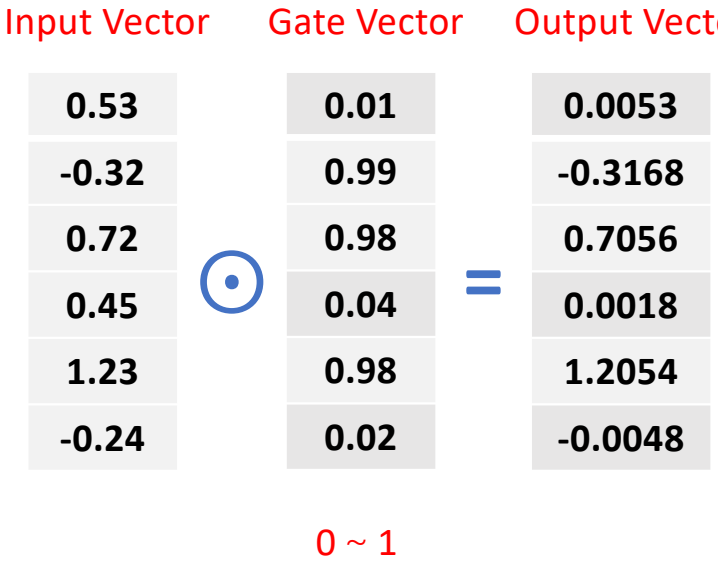
长短期记忆网络LSTM

- 解决长期依赖问题



长短期记忆网络LSTM

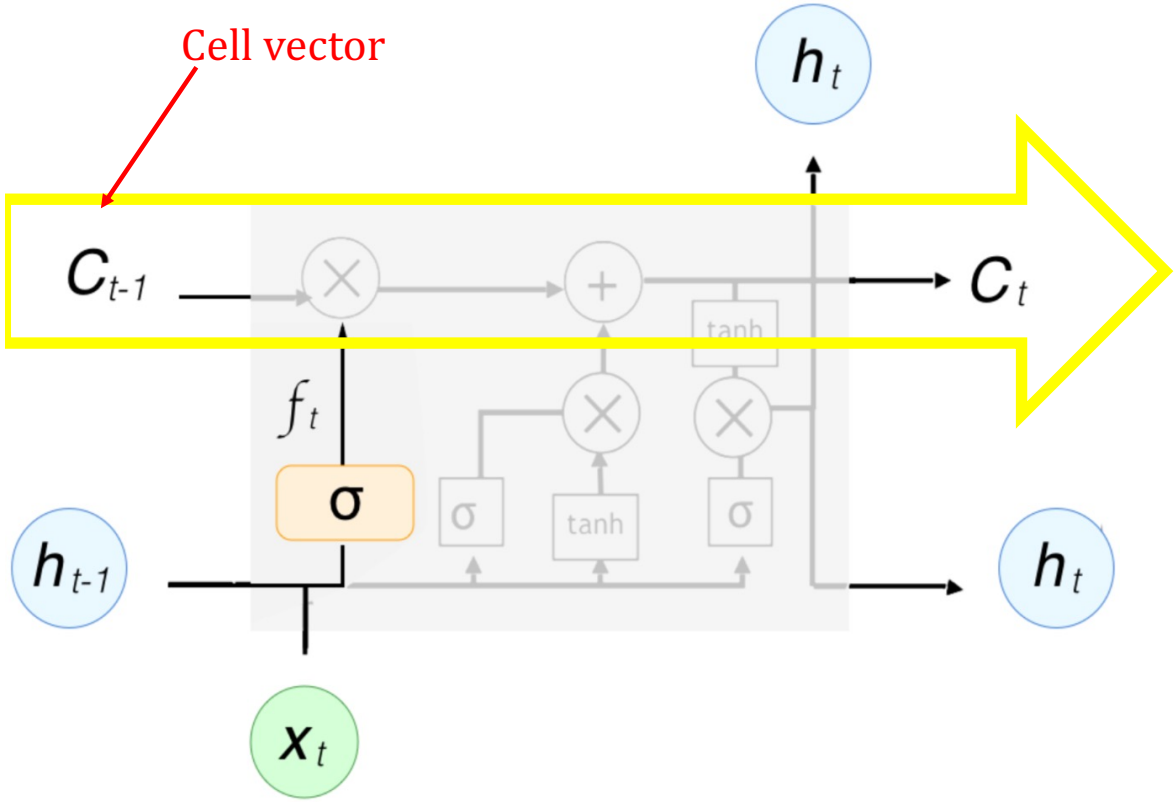
- 门控函数



- 在时间维度上，RNN只传递一个隐藏状态向量h_t，而LSTM则有隐藏状态向量和**细胞向量（cell vector）**。
- 门控向量中的值从0到1变化，0表示“关闭”，1表示“打开”。
- 通过**逐元素（element-wise）**地将输入向量和门控向量相乘来“过滤”信息。

长短期记忆网络LSTM

- Forget Gate



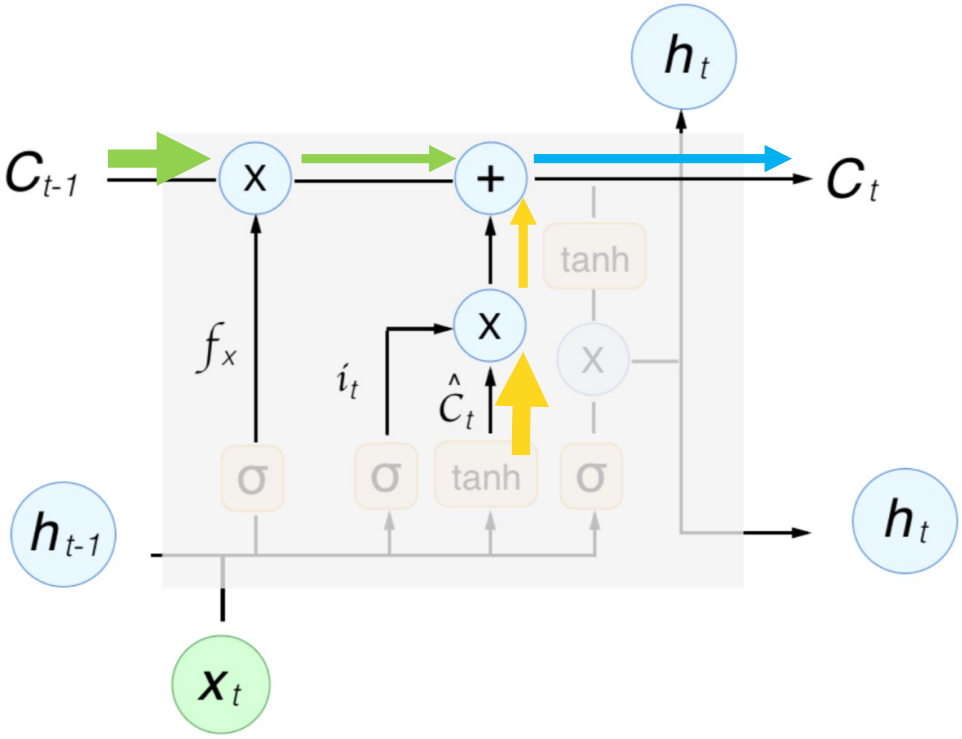
- 计算遗忘门的值：

$$f_t = sigmoid([h_{t-1}, x_t]\boldsymbol{W}_f + b_f)$$

Concatenate two vectors

长短期记忆网络LSTM

- Input Gate



- 计算输入门（input gate ）的输出向量

$$i_t = sigmoid([h_{t-1}, x_t]\boldsymbol{W}_i + b_i)$$

- 计算信息向量information vector

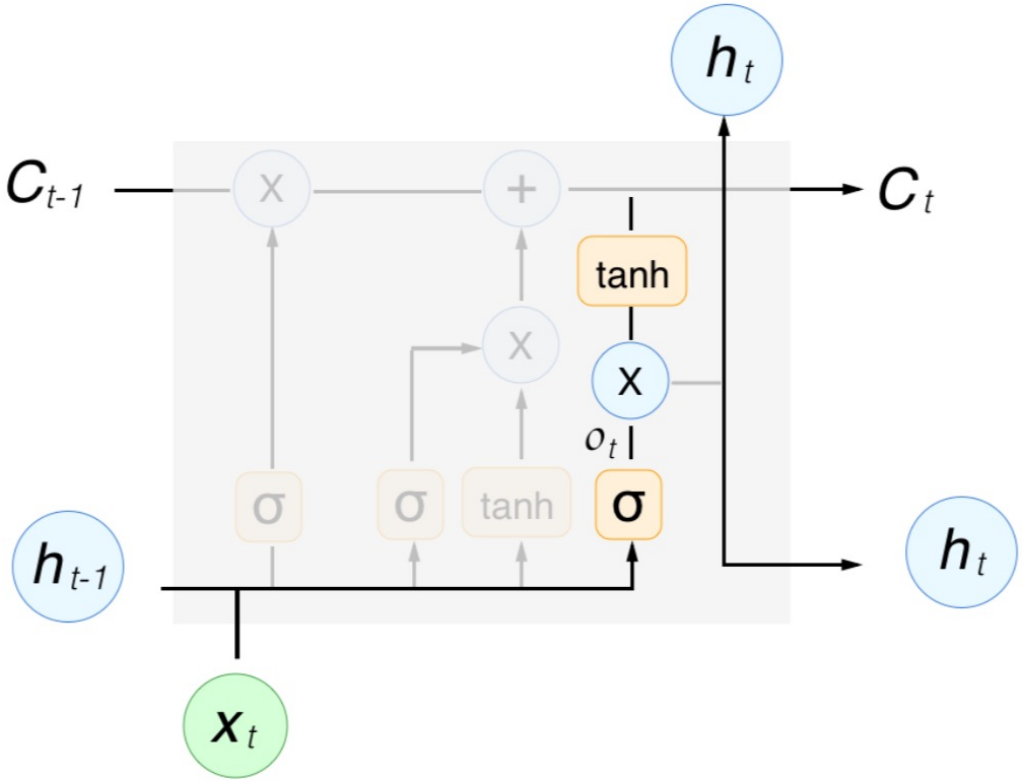
$$\hat{C}_t = \tanh([h_{t-1}, x_t]\boldsymbol{W}_C + b_C)$$

- 计算新的细胞向量

$$C_t = \underbrace{f_t \odot C_{t-1}}_{\text{忘记/保留前面的信息}} + \underbrace{i_t \odot \hat{C}_t}_{\text{输入新的信息}}$$

长短期记忆网络LSTM

- Output Gate



- 计算输出门 output gate 向量

$$o_t = sigmoid([h_{t-1}, x_t]\boldsymbol{W}_o + b_o)$$

- 计算新的隐藏状态

$$h_t = o_t \odot tanh(C_t)$$

长短期记忆网络LSTM

- 问题：
  - 我们是否可以用ReLU来替换门控函数中的Sigmoid函数？
  - 当向向量输入信息时，为什么我们使用tanh而不是Sigmoid？

长短期记忆网络LSTM

- **LSTM的变体**
- LSTM最初是在1997年发明的，目前有许多LSTM的变体，其中包括门控循环单元（GRU）。但是，Greff等人[1]对三个典型任务（语音识别、手写识别和多音乐建模）中的八种LSTM变体进行了分析，并总结了5400次实验（相当于15年的CPU时间）的结果。这项研究表明，没有任何一种LSTM变体在标准LSTM上提供了显著的改进。

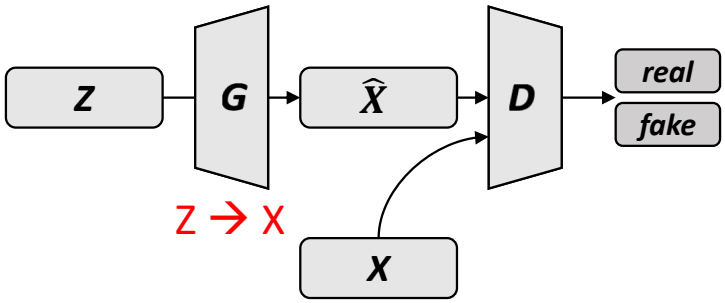
- 门控循环单元（GRU）不具有cell state，并减少了LSTM的计算成本和内存使用。

[1] K. Greff, R. K. Srivastava, J. Koutn´ık, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” IEEE TNNLS, 2017.

RNNs are Generative Models

序列生成模型 RNNs are Generative Models

- 回顾GAN



$$p(\mathrm{X}) = p(X|Z)p(Z)$$

真实数据分布　　生成器　　先验分布

[1] K. Greff, R. K. Srivastava, J. Koutn´ık, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” IEEE TNNLS, 2017.

序列生成模型 RNNs are Generative Models

- RNN

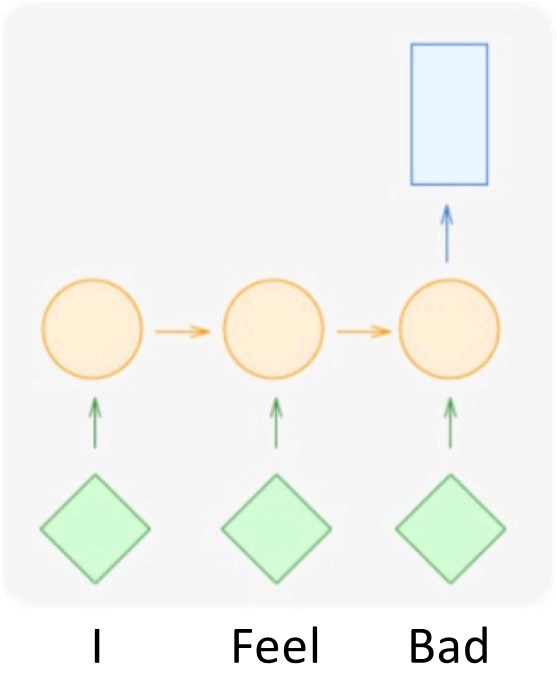
$$p(x_1, x_2, \dots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots p(x_n|x_1, \dots, x_{n-1})$$

逐一生成

Time-series Applications

时间序列应用

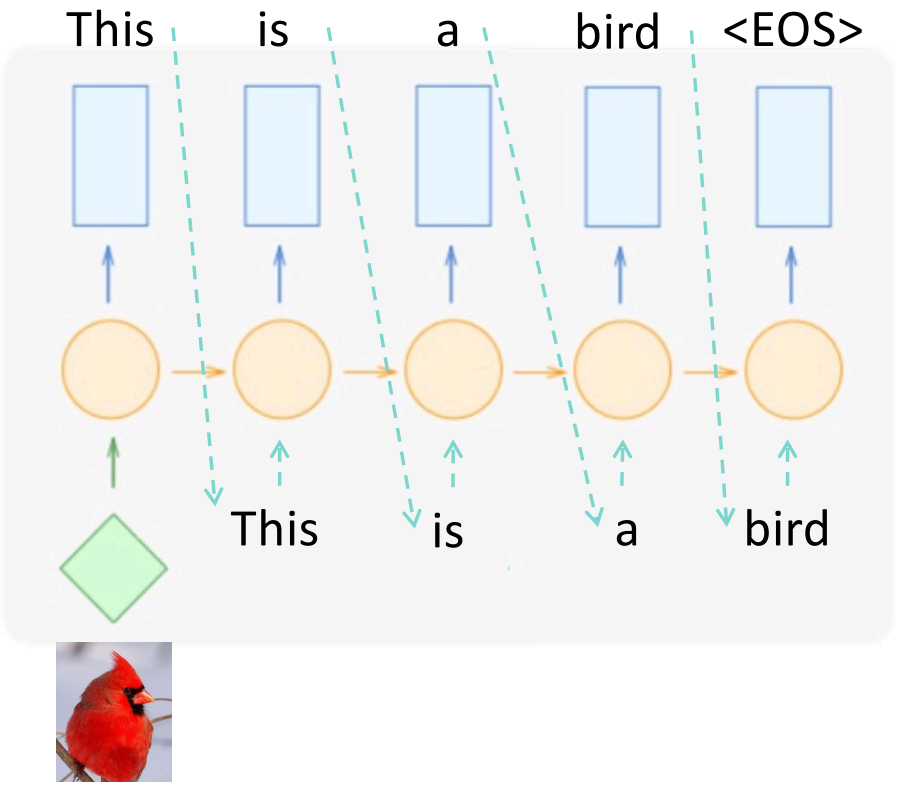
- Many-to-one: 句子情感分类



- 分类任务
- 使用最后一个输出来计算损失。
- 在隐藏向量的顶部堆叠一个全连接层和Softmax。

时间序列应用

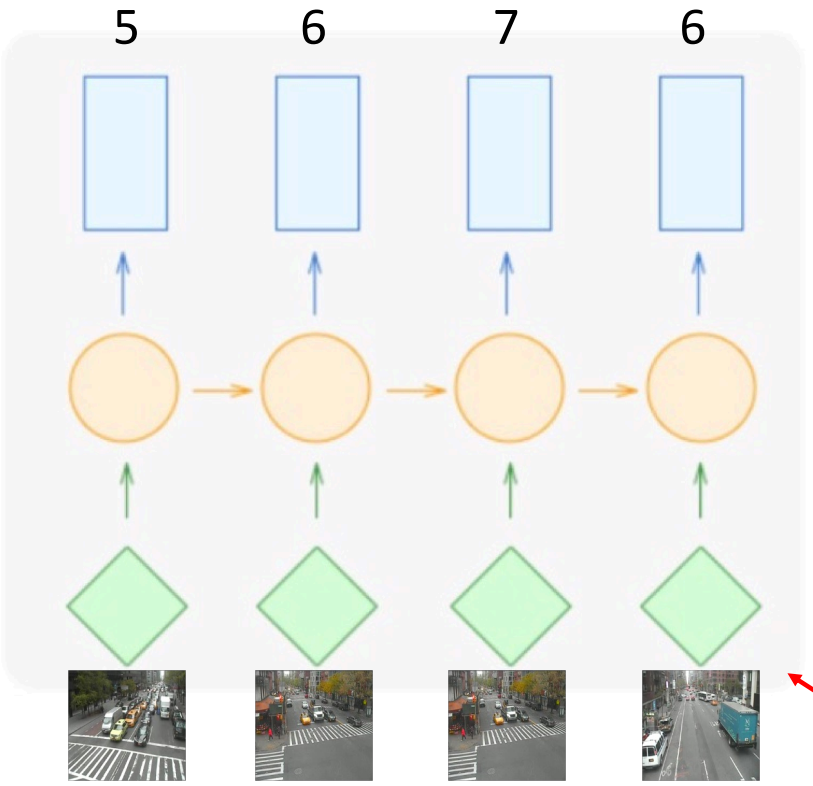
- One-to-many: 图片描述 Image Captioning



- 每个时间步的输出作为下一个时间步的输入。
- 当输出是特殊的结束句子标记（EOS）时，终止该过程。
- 使用所有输出来计算损失，例如，所有输出的平均交叉熵。

时间序列应用

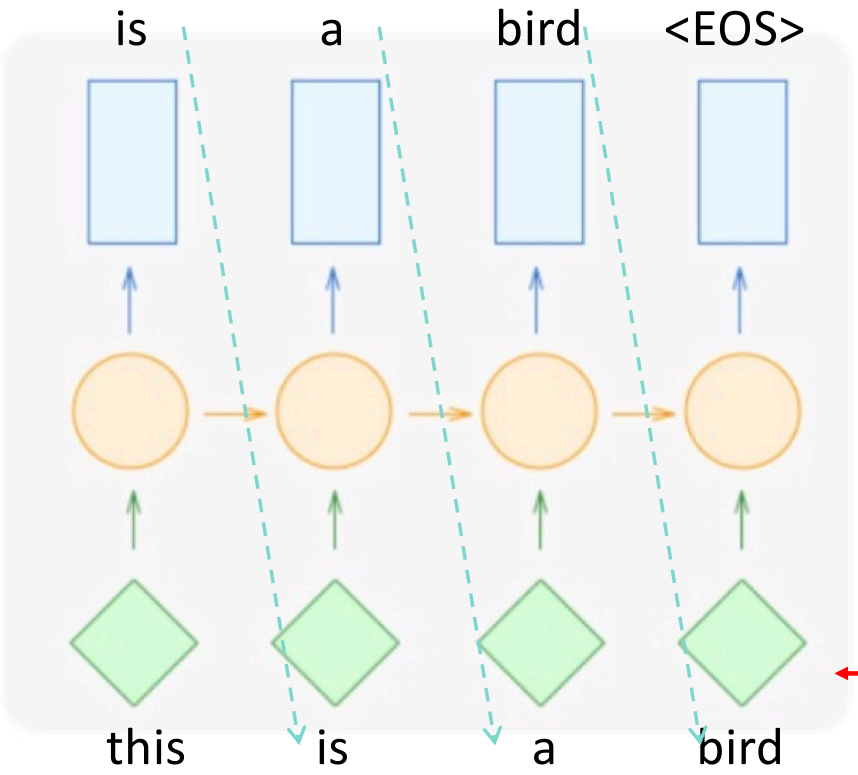
- 同步的Many-to-Many: 交通计数 Traffic Counting



- 训练时需要预先定义一个序列长度来计算损失。
- 测试时，输入的数据一个接一个地进行处理。

时间序列应用

- 同步的 Many-to-Many: 文本生成/语言建模

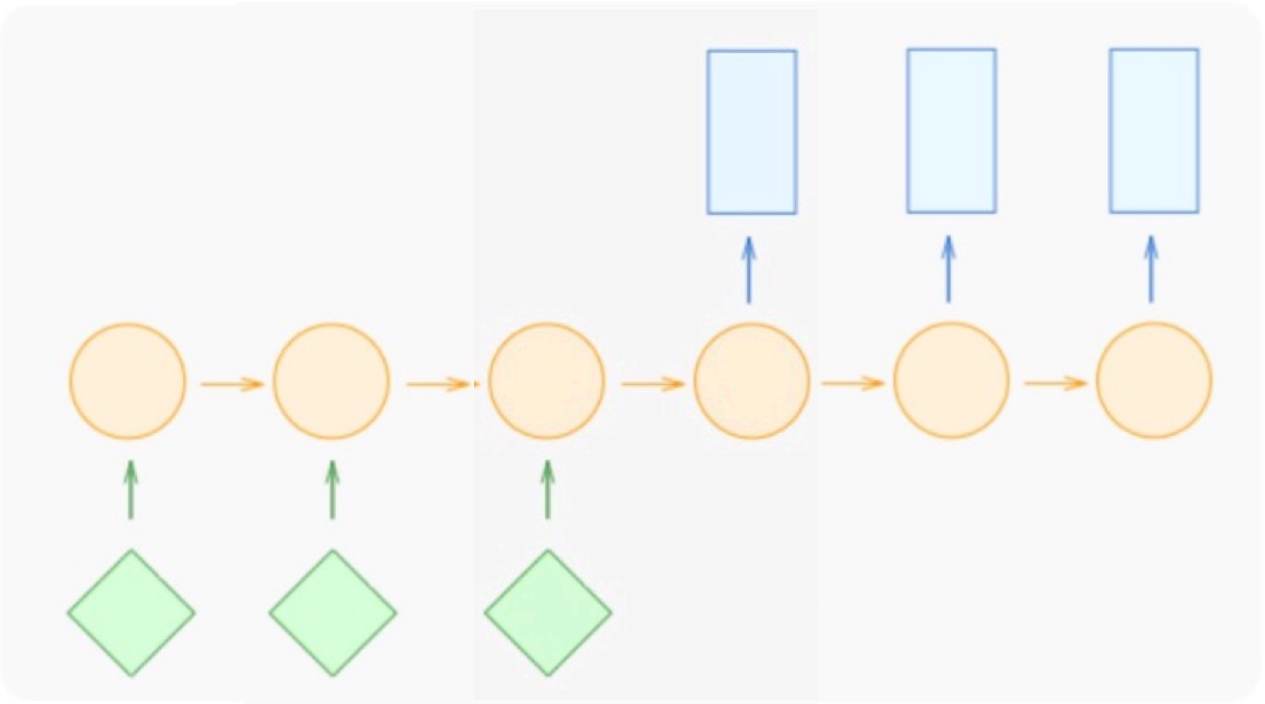


- 每一步的输出作为其下一步的输入。

在测试时，输入“this”，输出整个句子。

时间序列应用

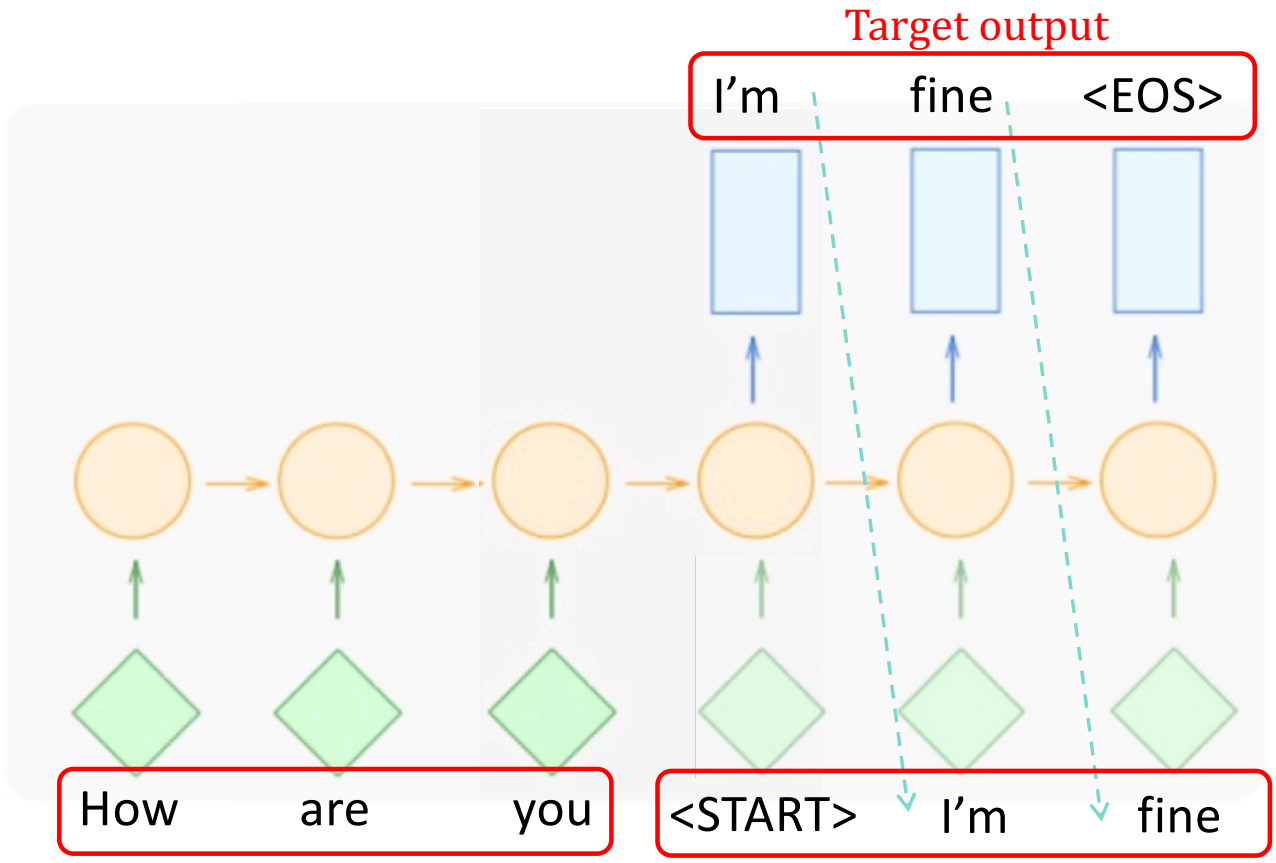
- 异步的 Many-to-Many (Seq2Seq): 聊天机器人 Chatbot



- 将输入的序列数据进行编码处理，然后再开始输出相应的序列结果。

时间序列应用

- 异步的 Many-to-Many (Seq2Seq): 聊天机器人 Chatbot



- 在开始输出结果之前，先对输入的顺序数据进行编码。
- 在训练过程中，在目标输出末尾添加一个EOS（结束）标记，并在解码器输入开头添加一个START标记。

回顾总结

- 动机
  - Time-series data
- 词的表示 Word Representation
  - one-hot vector, BOW, word embedding, Word2Vec, CBOW, Skip-Gram, negative sampling, NCE
- 序列数据 Sequential Data
  - Time-step, one-to-many, many-to-one, asynchronous many-to-many, synchronous many-to-many
- 朴素循环神经网络 Vanilla Recurrent Neural Network
  - Hidden vector (state), long-term dependency problem
- 长短期记忆网络 LSTM Long Short-Term Memory
  - Cell vector (state), forget gate, input gate, output gate
- 序列生成模型 RNNs are Generative Models
  - $p(x_1, x_2, \dots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots p(x_n|x_1, \dots, x_{n-1})$
- 时间序列应用 Time-series Applications
  - one-to-many, many-to-one, asynchronous many-to-many, synchronous many-to-many

作业

- 用循环神经网络在 IMDB 数据集上实现电影评论文本情感识别
- 要求：
  1. 使用 PyTorch 搭建循环神经网络模型
  2. 使用nn.RNN,nn.LSTM,nn.GRU等接口搭建模型，训练后测试集准确率要求不低于85%
  3. 手写实现RNN和LSTM的模型，训练后测试集准确率要求不低于80%
  4. 调整网络结构、损失函数、训练流程，观察对训练效果的影响
  5. 总结实验报告

谢谢