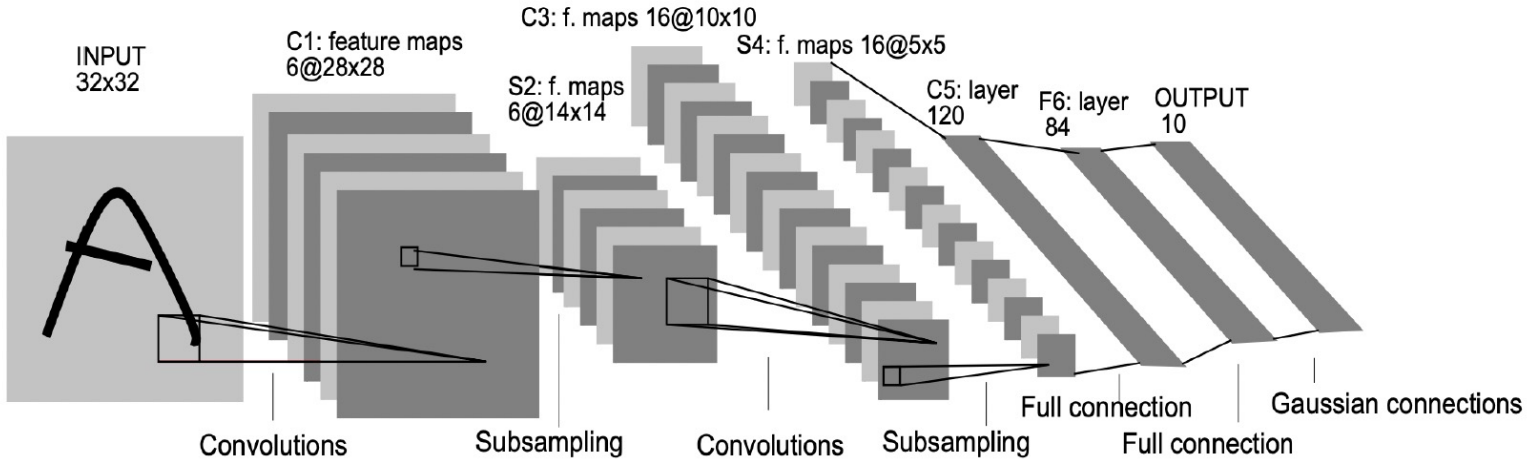
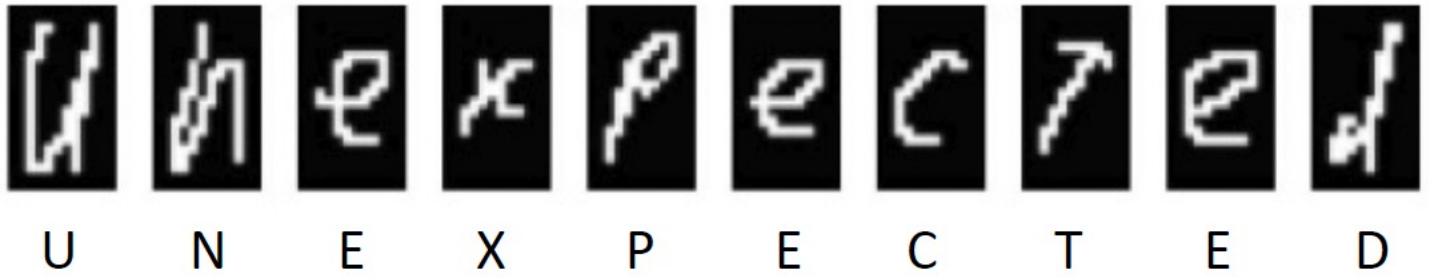
模式识别
Pattern Recognition

李泽榉，复旦大学 生物医学工程与技术创新学院

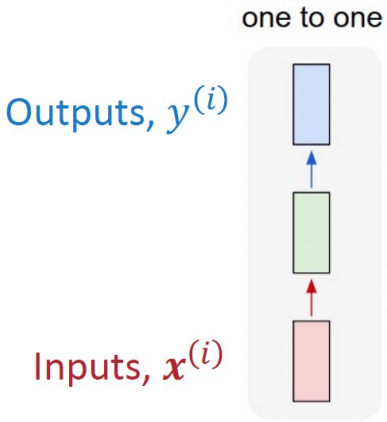
目录

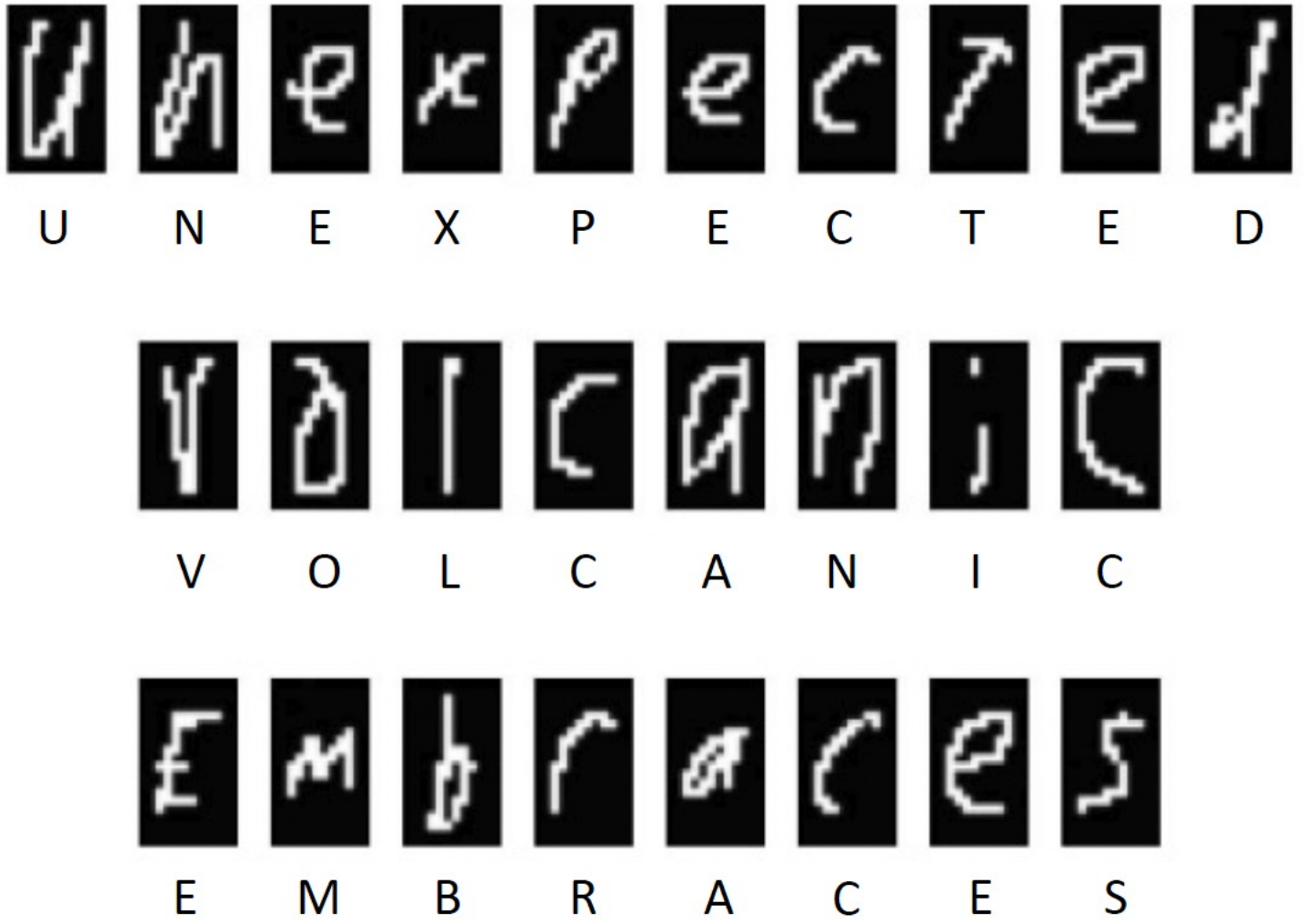
回顾：卷积神经网络


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections


U N E X P E C T E D

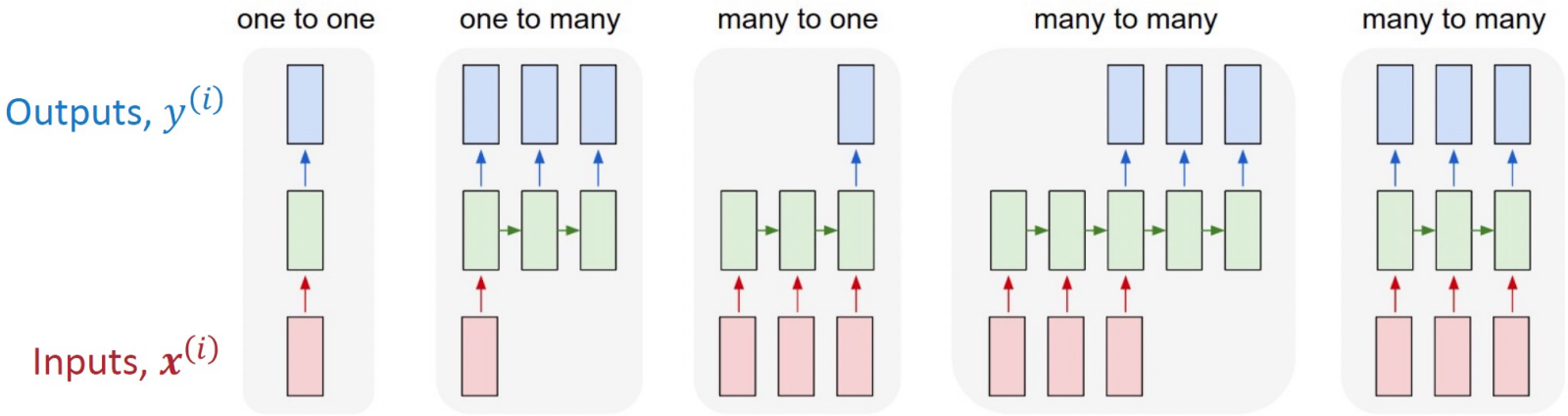
时序数据建模


one to one
Outputs, $y^{(i)}$
Inputs, $x^{(i)}$


U N E X P E C T E D
V O L C A N I C
E M B R A C E S

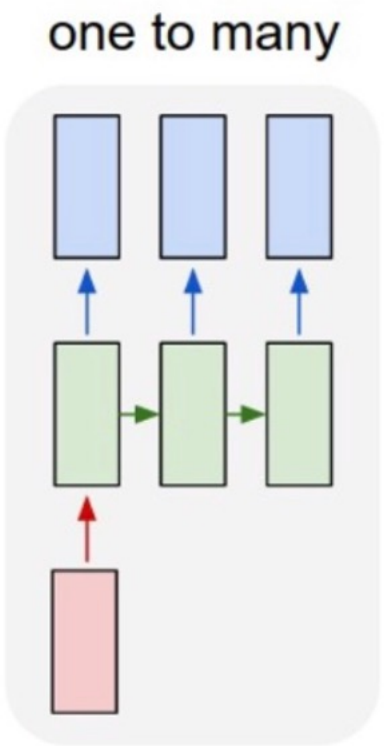
时序数据建模

$$\boldsymbol{y}^{(i)} = \left[\boldsymbol{y}_1^{(i)}, \boldsymbol{y}_2^{(i)}, \ldots, \boldsymbol{y}_{T_i}^{(i)}\right]$$



$$\boldsymbol{x}^{(i)} = \left[\boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, \ldots, \boldsymbol{x}_{T_i}^{(i)}\right]$$

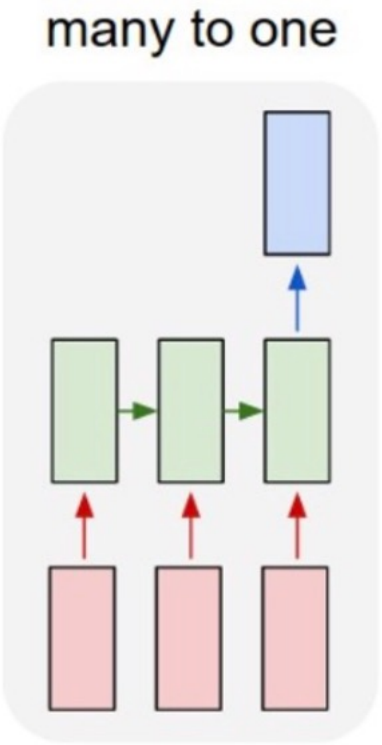
时序数据建模



特点: 单个输入通过RNN生成一系列输出，可以连续生成多个输出

适用任务: 能够从单个数据点生成序列，可以适用于音乐生成和图像描述等任务

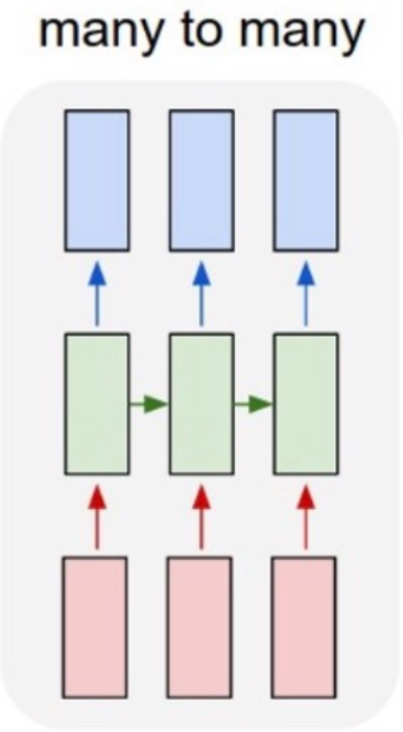
时序数据建模



特点: 序列输入转化为单个输出，经过RNN处理后输出一个结果

适用任务: 需要理解整个序列后做出判断或分类的任务，例如情感分析和垃圾邮件分类检测

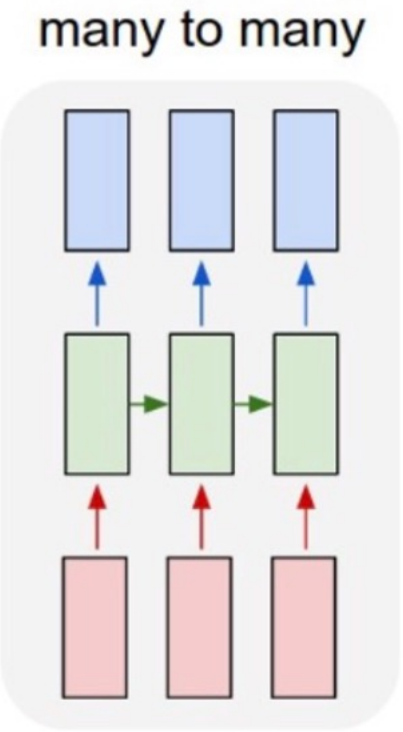
时序数据建模



特点: (同步类型)序列输入和序列输出，输入和输出长度相同

适用任务: 在每个时间步都有输入和输出，适用于需要逐步处理的复杂任务，如词性标注

时序数据建模

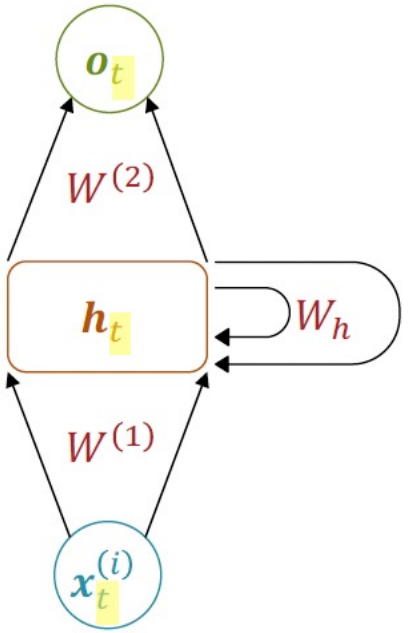


特点: (异步类型)序列输入和序列输出，输入和输出长度不相同

适用任务: 用来处理输入输出长度不匹配的情况，允许网络结构更加灵活地处理信息，如机器翻译和语音识别任务

RNN

$$\boldsymbol{h}_t = \left[1, \theta\left(W^{(1)}\boldsymbol{x}_t^{(i)} + W_h\boldsymbol{h}_{t-1}\right)\right]^T \text{ and } \boldsymbol{o}_t = \hat{y}_t^{(i)} = \theta\left(W^{(2)}\boldsymbol{h}_t\right)$$



- Training dataset consists of (input sequence, label sequence) pairs, potentially of varying lengths

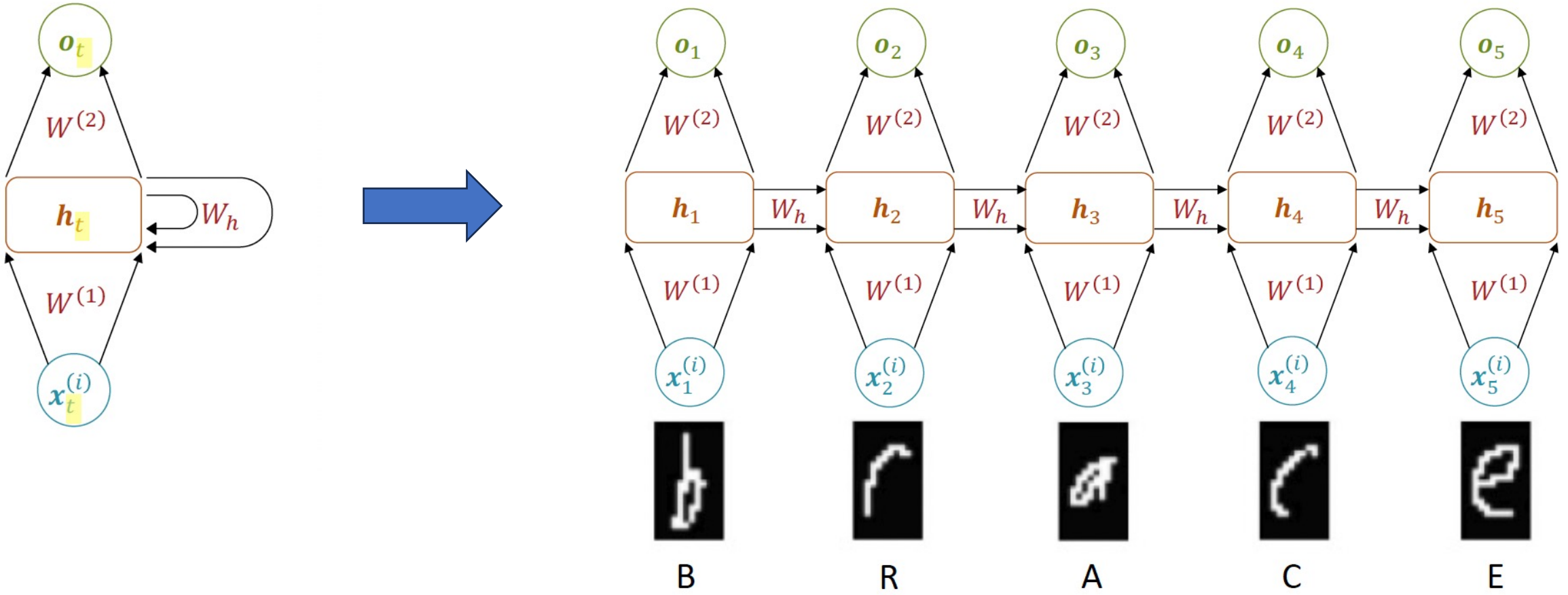
$$\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\right)\right\}_{n=1}^{N}$$

$$\boldsymbol{x}^{(n)} = \left[\boldsymbol{x}_1^{(n)}, \ldots, \boldsymbol{x}_{T_n}^{(n)}\right]$$

$$\boldsymbol{y}^{(n)} = \left[\boldsymbol{y}_1^{(n)}, \ldots, \boldsymbol{y}_{T_n}^{(n)}\right]$$

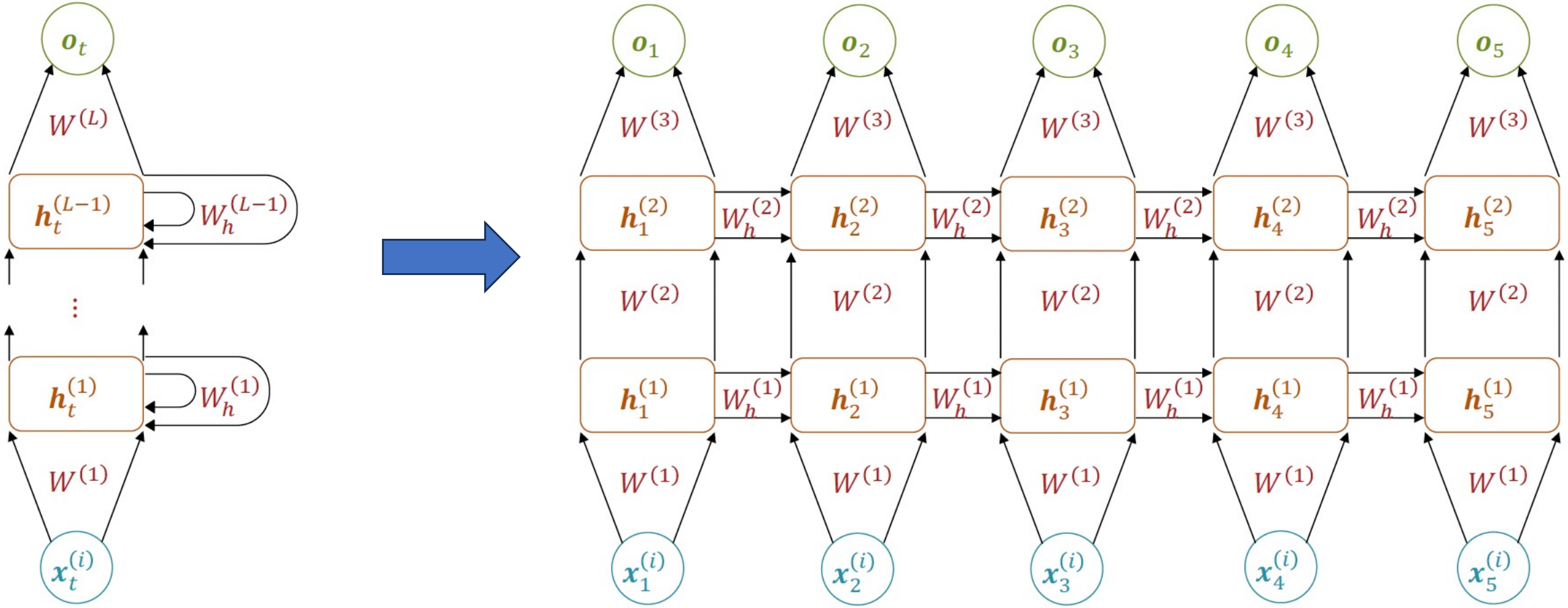
Unrolling RNN

$$\boldsymbol{h}_t = \left[1, \theta\left(W^{(1)}\boldsymbol{x}_t^{(i)} + W_h\boldsymbol{h}_{t-1}\right)\right]^T \text{ and } \boldsymbol{o}_t = \hat{y}_t^{(i)} = \theta\left(W^{(2)}\boldsymbol{h}_t\right)$$

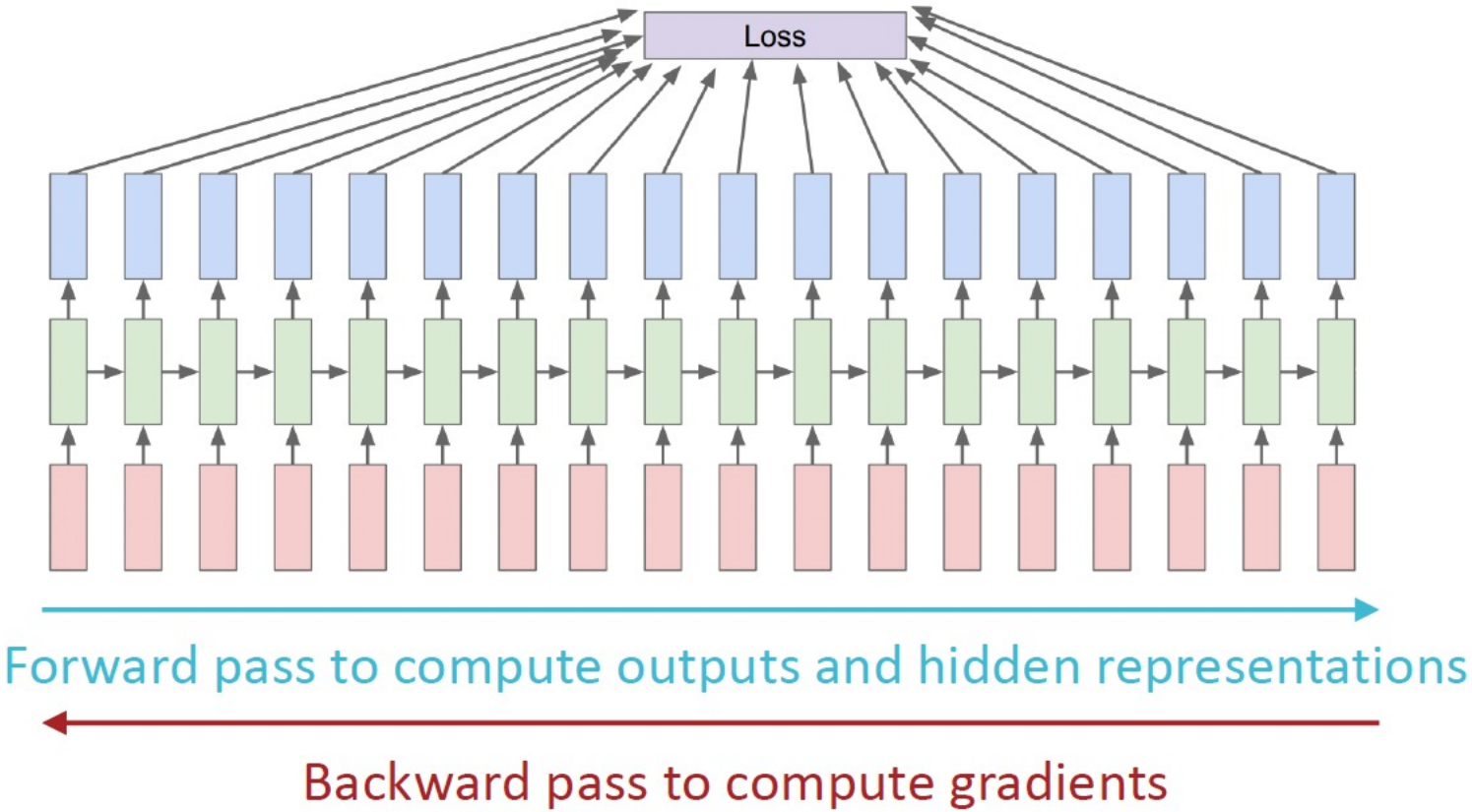


Deep RNN

$$\boldsymbol{h}_t^{(l)} = \left[1, \theta\left(W^{(l)}\boldsymbol{h}_t^{(l-1)} + W_h^{(l)}\boldsymbol{h}_{t-1}^{(l)}\right)\right]^T \text{ and } \boldsymbol{o}_t = \hat{y}_t^{(i)} = \theta\left(W^{(L)}\boldsymbol{h}_t^{(L-1)}\right)$$

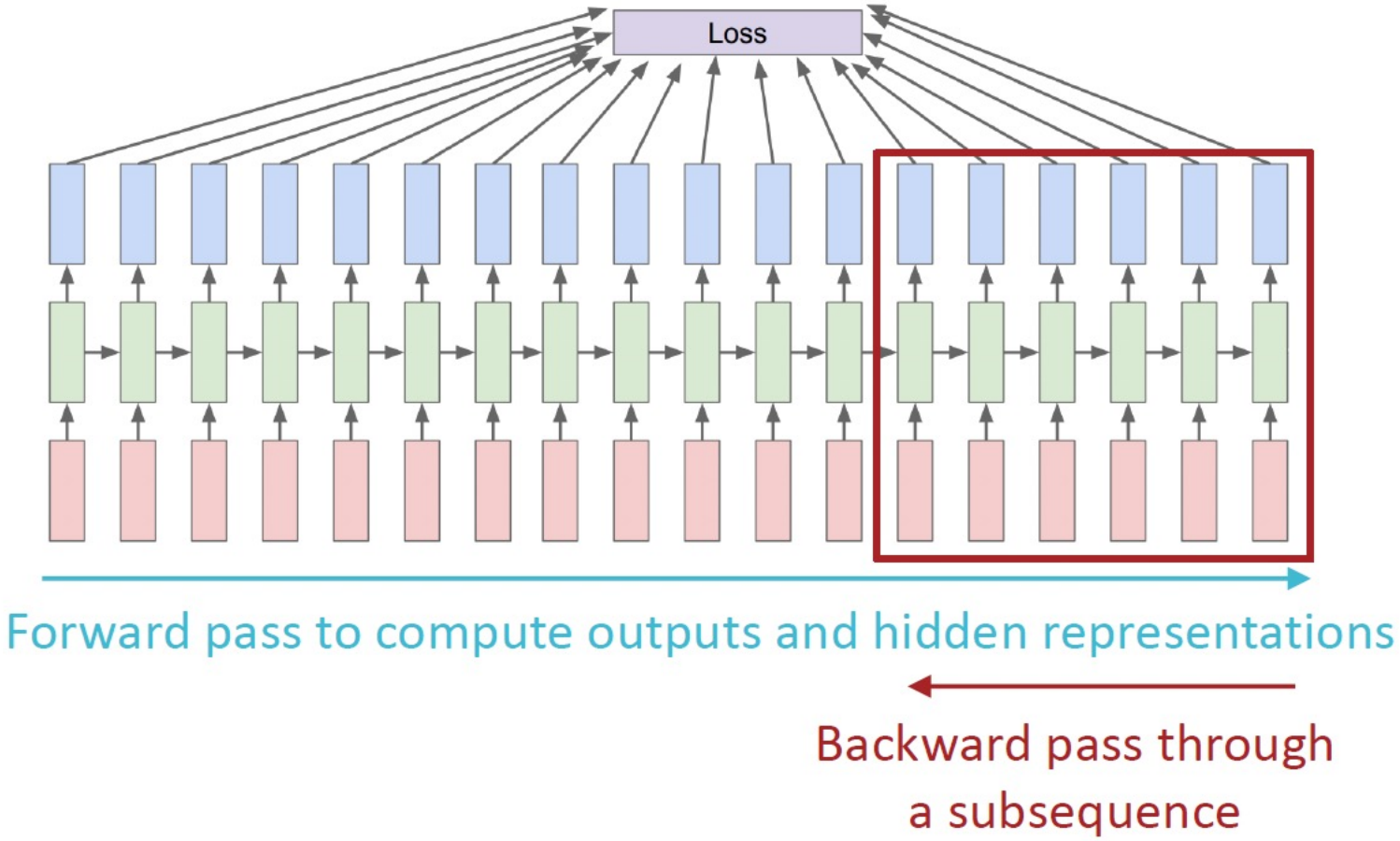


Training RNNs



- Issue: as the sequence length grows, the gradient is more likely to explode or vanish

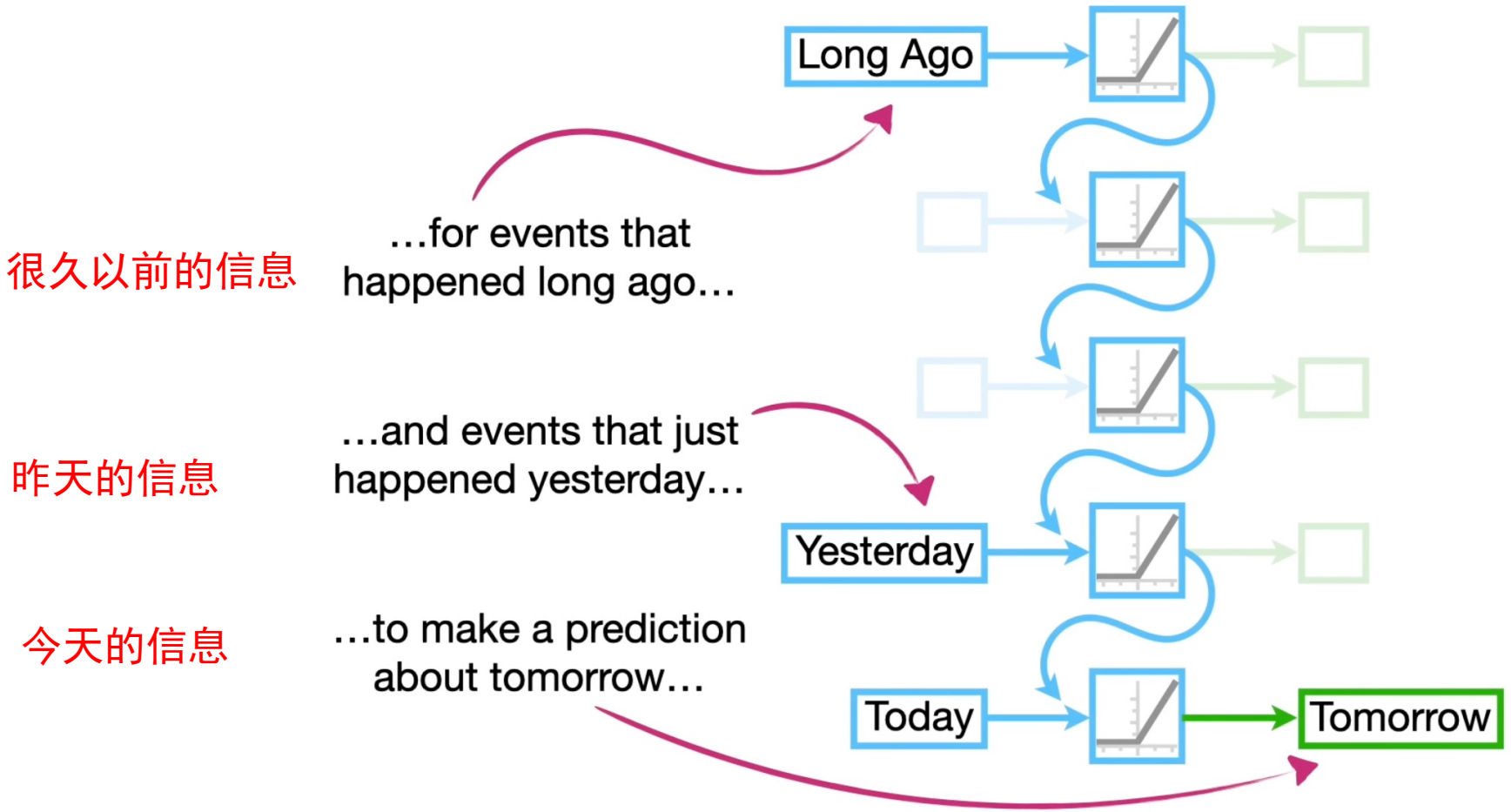
Training RNNs



- Idea: limit the number of time steps to backprop through

LSTM

- LSTM networks address the vanishing gradient problem by replacing hidden layers with memory cells



LSTM

- LSTM networks address the vanishing gradient problem by replacing hidden layers with memory cells
- Each cell still computes a hidden representation but also maintains a separate internal state, C_t

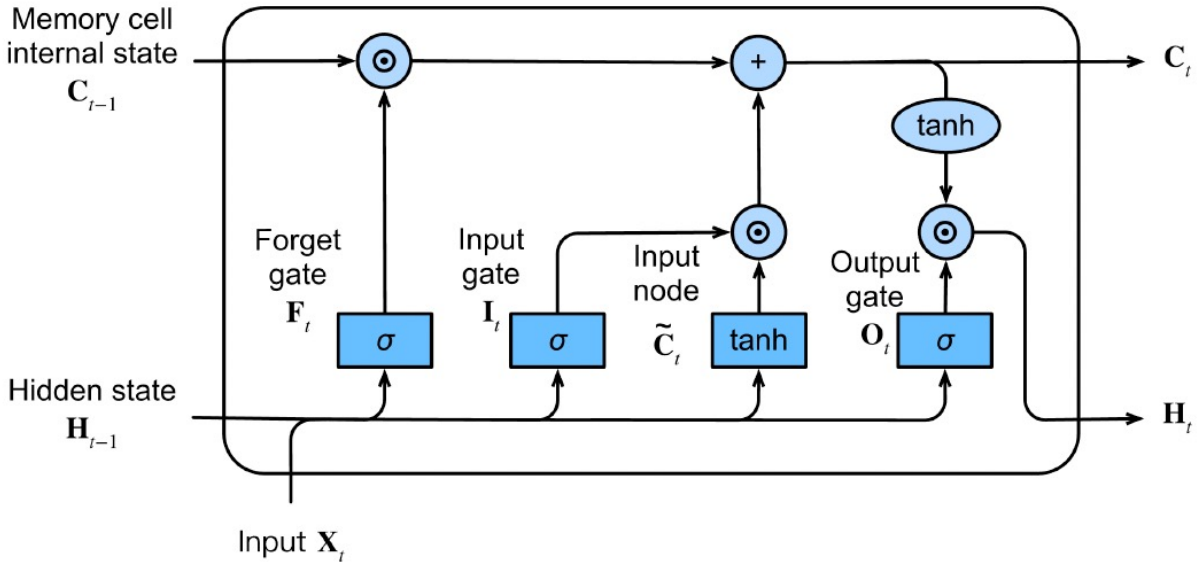
$$I_t = \sigma\left(W_{ix}\boldsymbol{x}_t^{(i)} + W_{ih}\boldsymbol{h}_{t-1}\right)$$

$$O_t = \sigma\left(W_{ox}\boldsymbol{x}_t^{(i)} + W_{oh}\boldsymbol{h}_{t-1}\right)$$

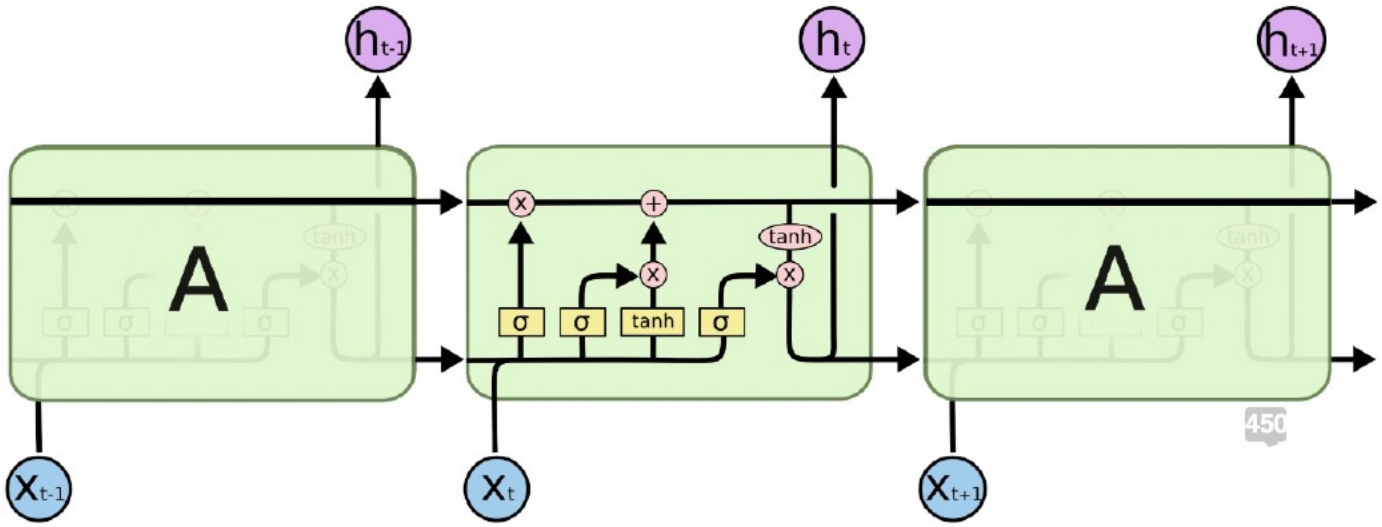
$$F_t = \sigma\left(W_{fx}\boldsymbol{x}_t^{(i)} + W_{fh}\boldsymbol{h}_{t-1}\right)$$

$$C_t = F_t \odot C_{t-1} + I_t \odot \theta\left(W^{(1)}\boldsymbol{x}_t^{(i)} + W_h\boldsymbol{h}_{t-1}\right)$$

$$\boldsymbol{h}_t = C_t \odot O_t$$



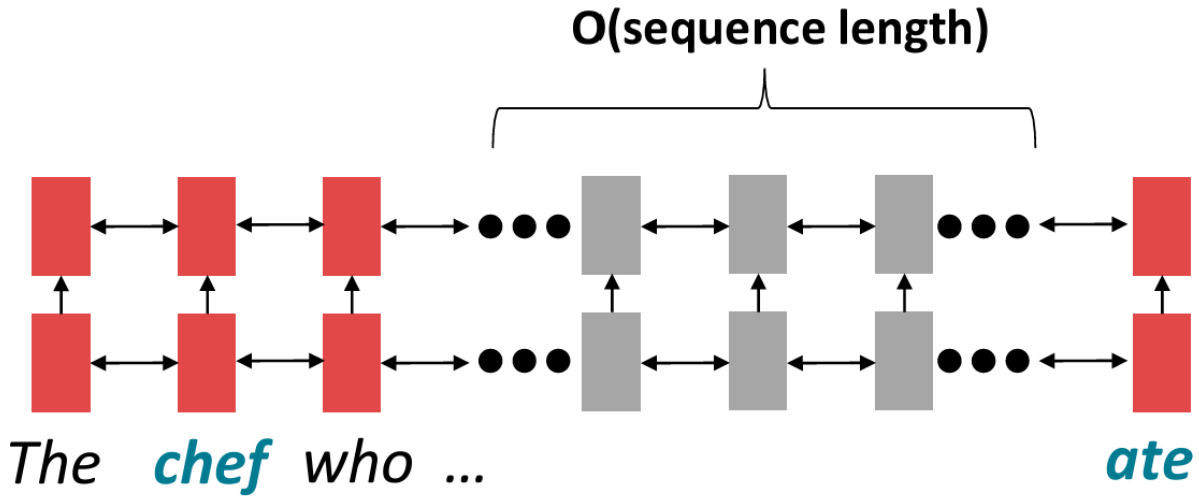
LSTM



- The internal state allows information to move through time without needing to affect the hidden representations!

RNN的长距离依赖问题

- RNN“从左到右”顺序处理输入
- RNN编码了线性局部性
  - 相邻的词语往往会相互影响它们的含义
- **Problem**:
  - RNNs需要 **O(序列长度) 的**步骤才能与距离较远的词进行交互
  - 前向和后向传播都有 **O(序列长度)** 的无法并行的操作



主语***chef*** 的信息需要传递 **O(序列长度)** 层才能与动词***ate***交互!

Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

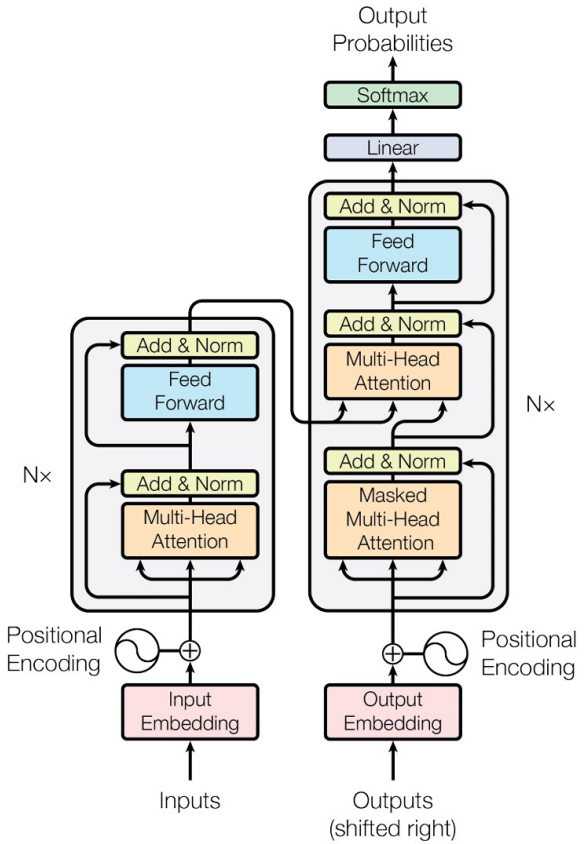
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

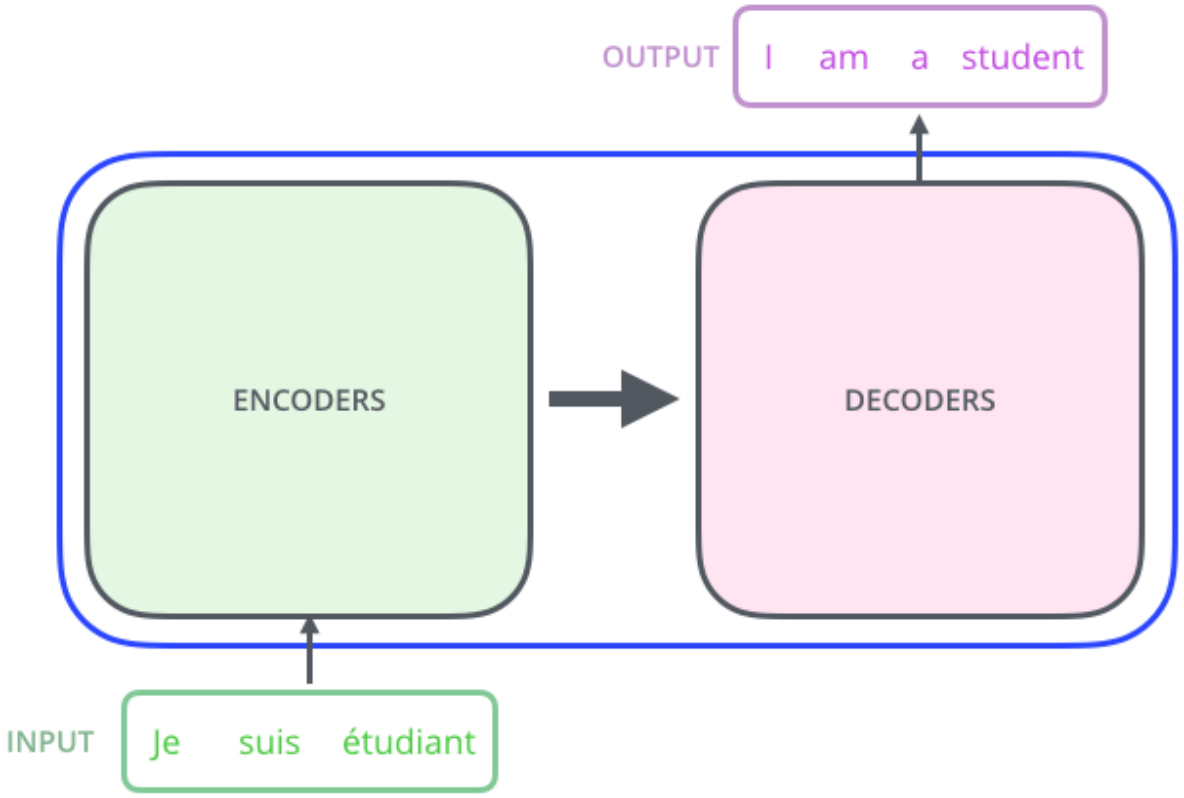
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.



- 2017年发布以来，引用量14万+
- 三个要素：Token，Attention和Position Encoding

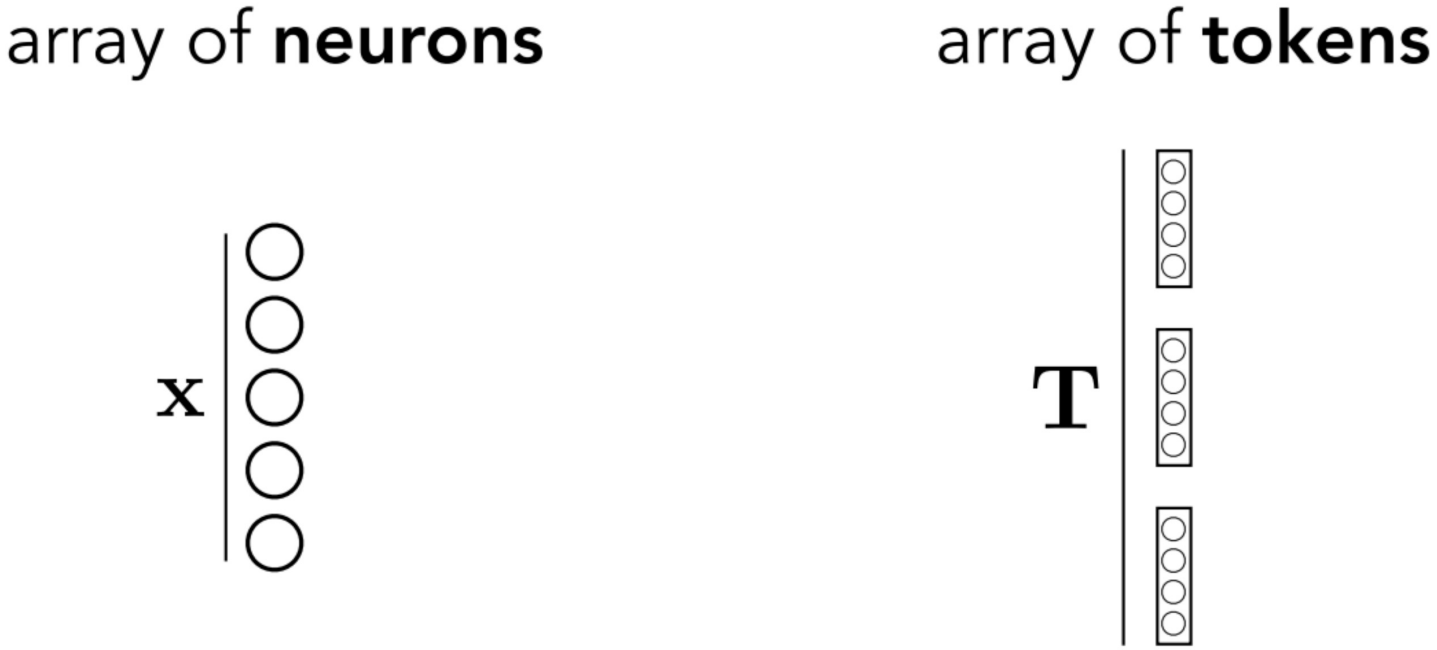
Transformer



- **Encoder** - 将输入序列映射为一个抽象的连续表示，该表示包含了输入中所有学习到的信息
- **Decoder** – 利用encoder输出的信息，同时将自身先前的输出作为输入进行处理，生成下一步的输出

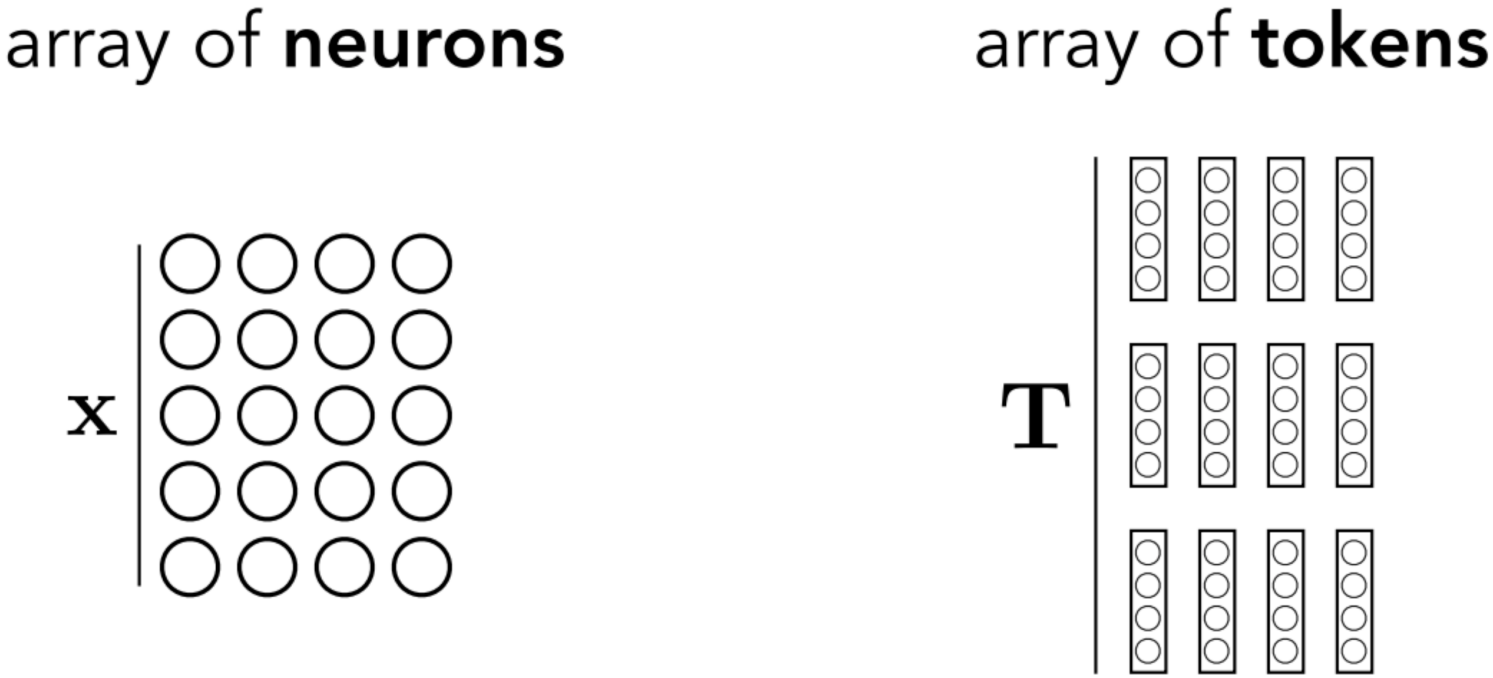
Tokens

- A vector of neurons
- An encapsulated bundle of information



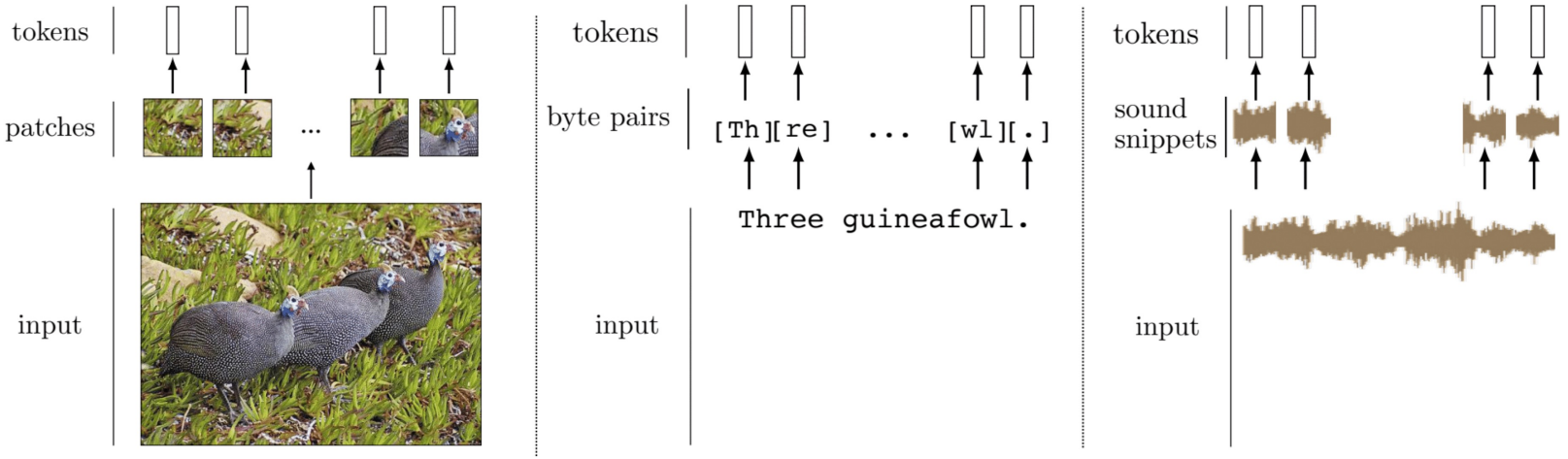
Tokens

- A vector of neurons
- An encapsulated bundle of information



Tokens

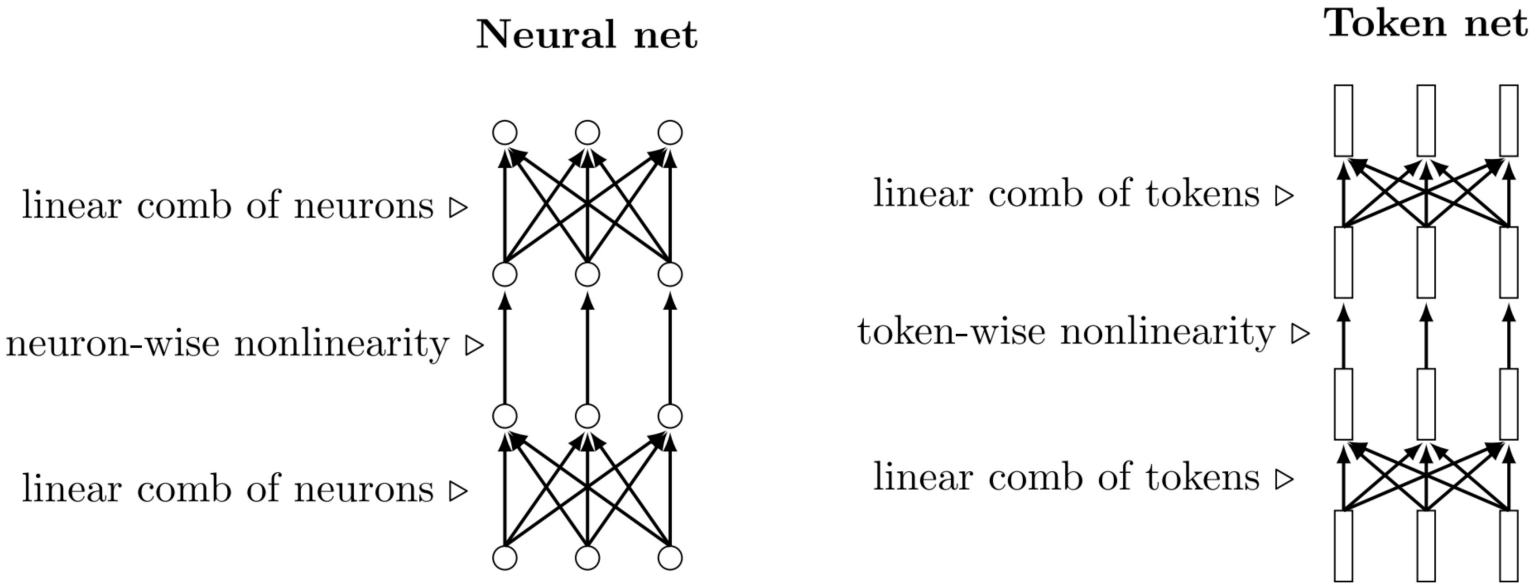
- 通过将输入切成小块，可以Tokenize万物



AN IMAGE IS WORTH 16X16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

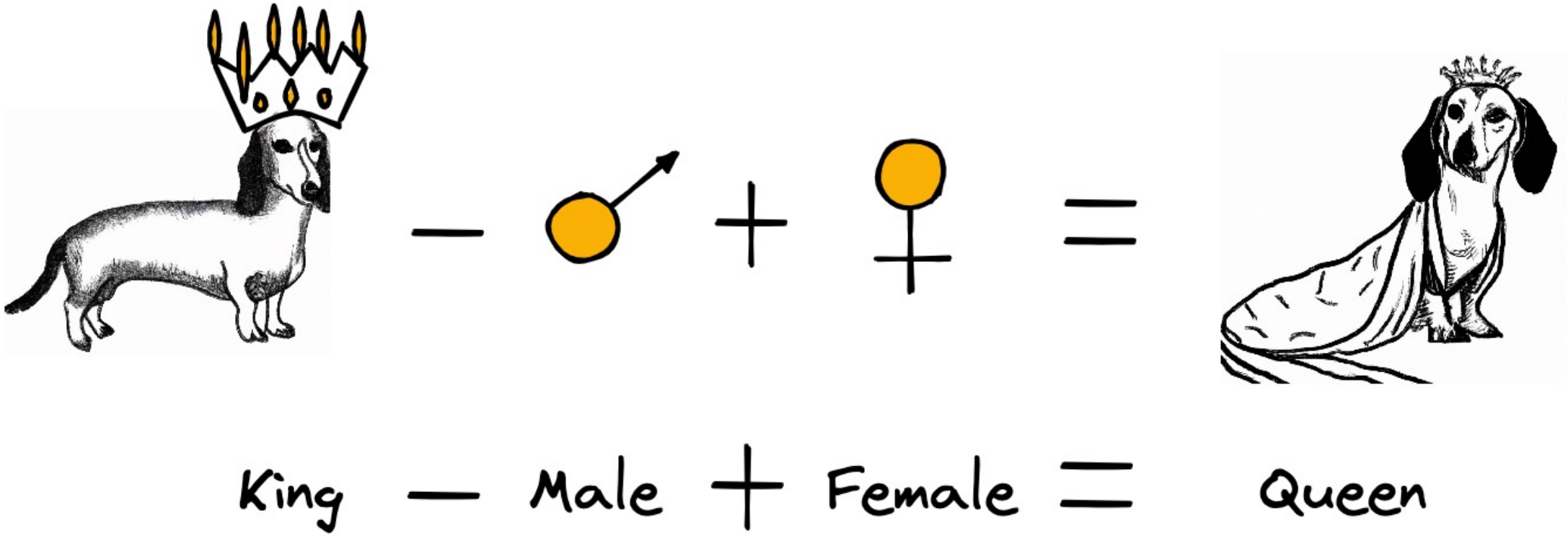
Tokens

- Token运算



Tokens

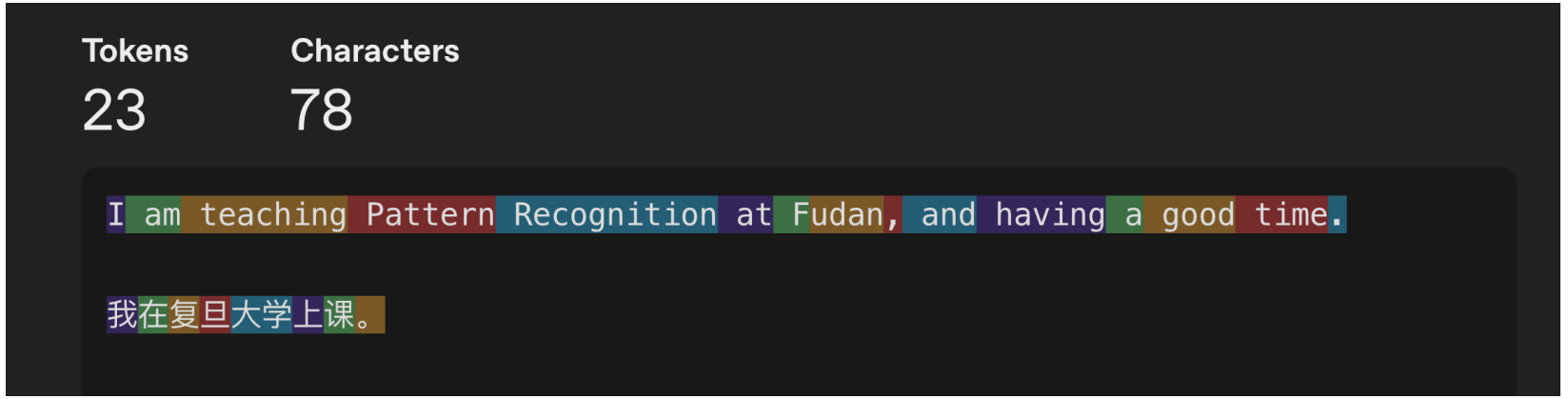
- Token的优势：能学习到词形的语义和相关性



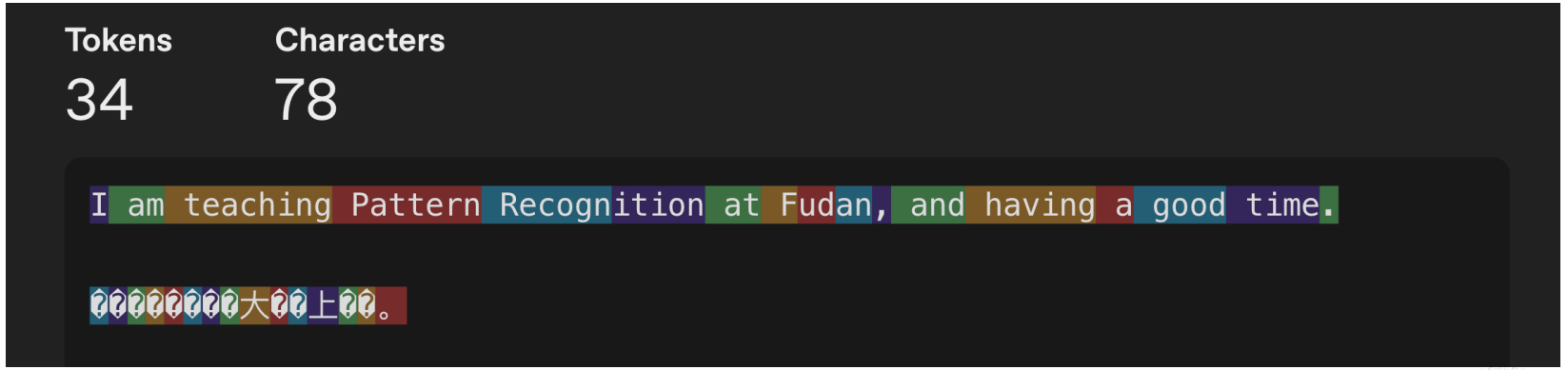
Tokens

- Tokenizer是一个关键，仍在更新

GPT-4o

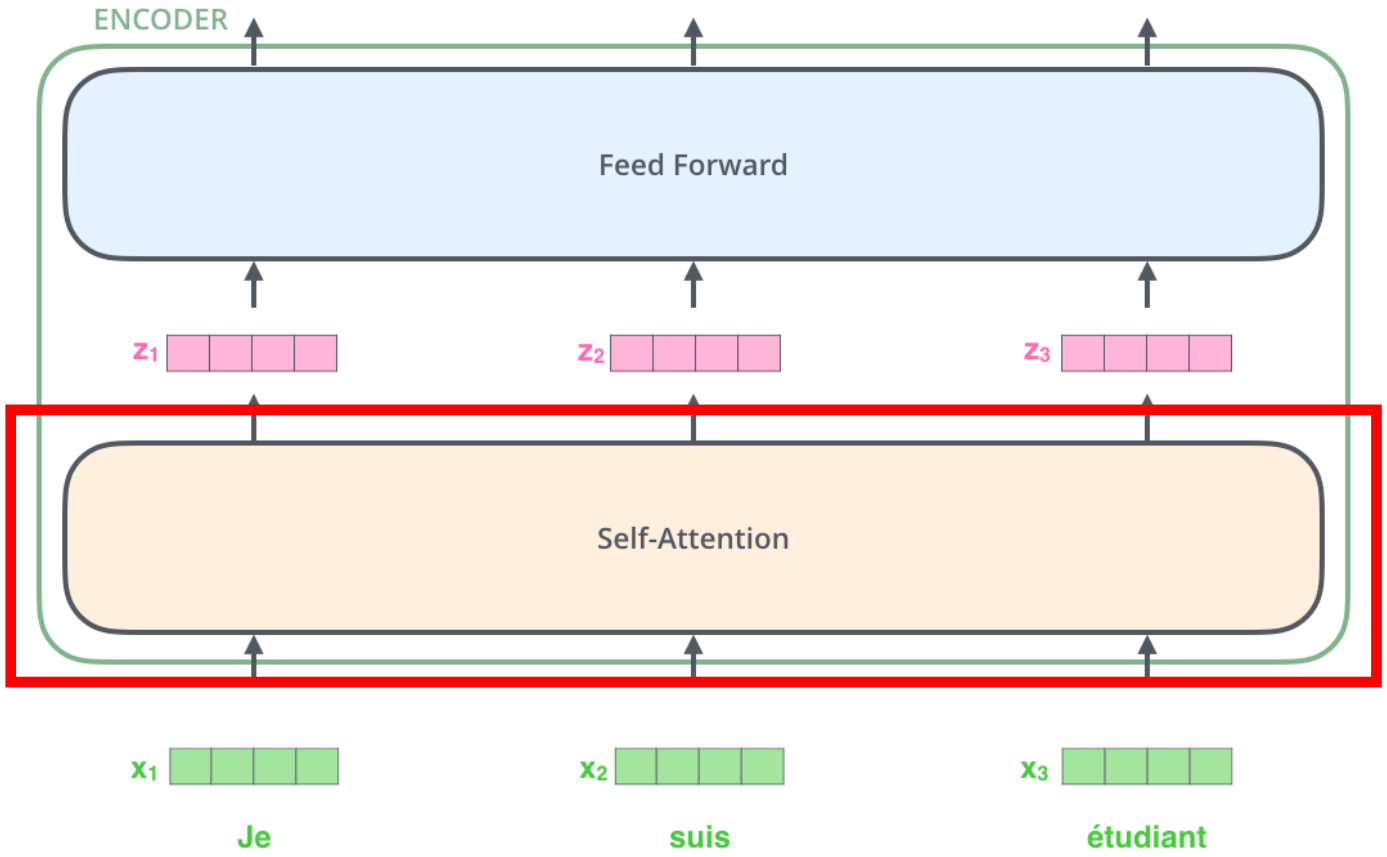


GPT-3



Self Attention

- **自注意力 Self-Attention**:
  自注意力机制使模型能够查看输入序列中的其他位置，以获取有助于更好地编码该词的线索

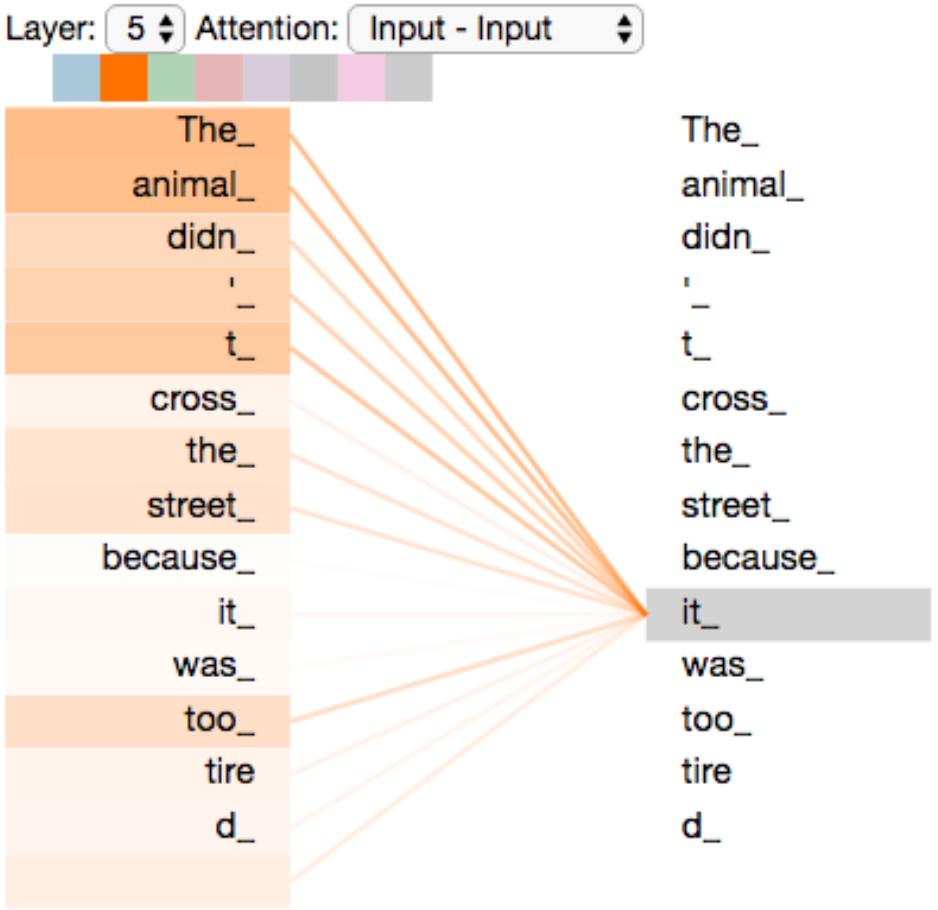


Self Attention

*“The animal didn't cross the street because **it** was too tired”*

- 这句话中的“**it**”指的是什么？
- 是指“**street**”还是“**animal**”？

- 当模型处理“it”这个词时，自注意力机制会让模型把“it”和“animal”联系起来



Self Attention

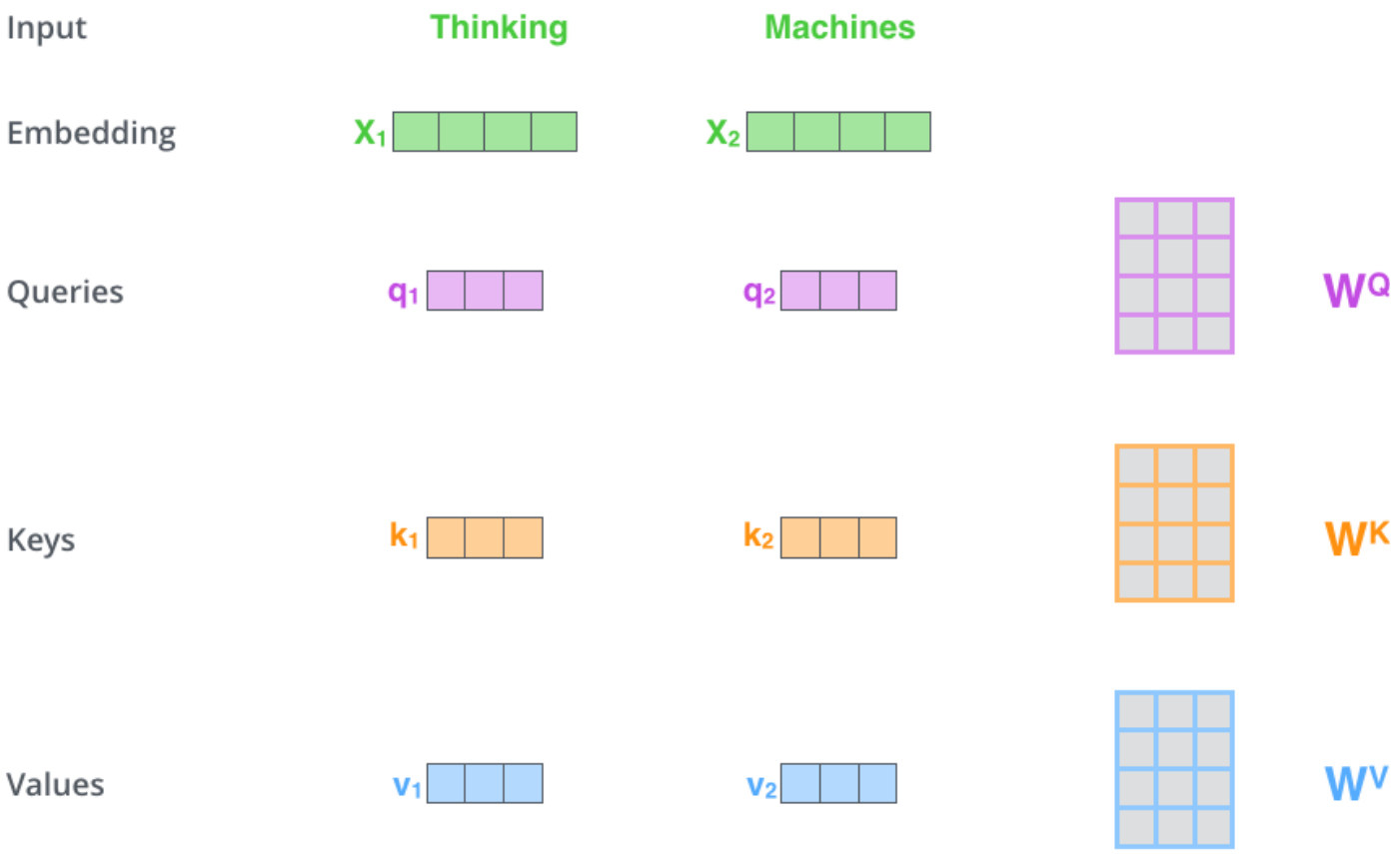
Step 1

- **创建三个向量** (Query, Key and Value)
- 这些向量由词嵌入分别与训练得到的三个权重矩阵 (W^Q, W^K, W^V) 相乘得到

$$q_i = x_i W^Q$$

$$k_i = x_i W^K$$

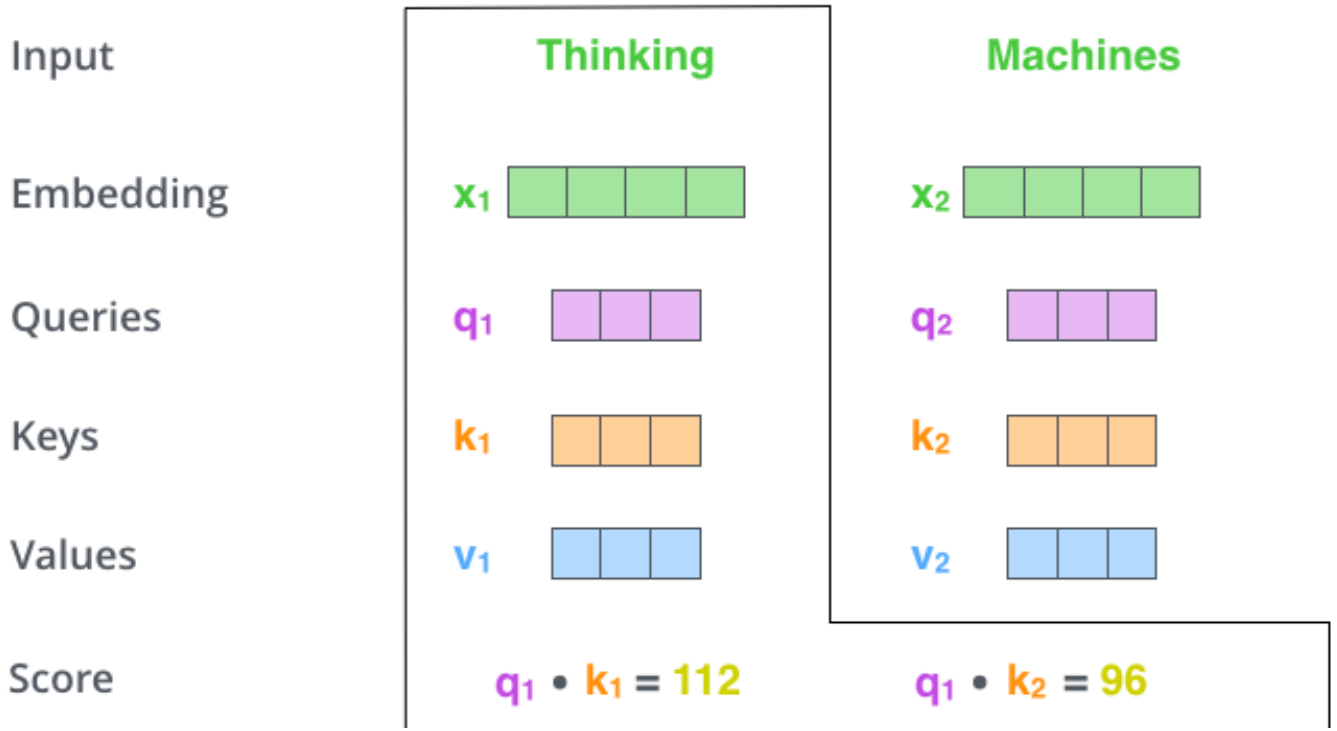
$$v_i = x_i W^V$$



Self Attention

Step 2

- **计算注意力得分**
- 将当前单词的query与句中对应的所有key进行点乘，得到注意力得分
- 这个分数决定了我们进行编码时，当前单词对输入句子各个部分的关注程度



Self Attention

Step 3

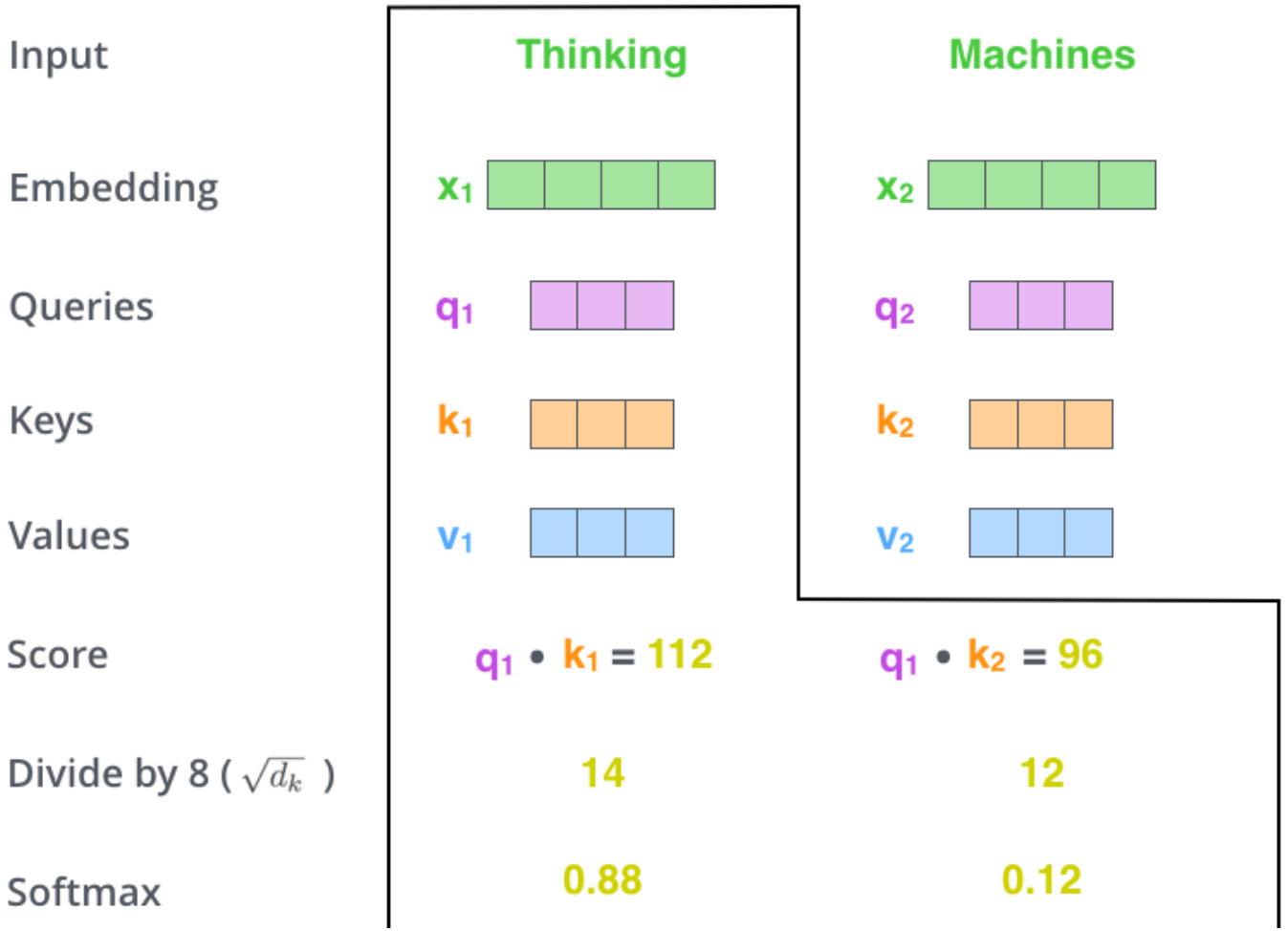
- 将分数**除以向量维度的平方根($\sqrt{d_k}$)**

Step 4

- **计算分数的Softmax**
- Softmax对分数进行归一化处理，得到各个value的权重（正值，和为1）

Quick Statistics Review:

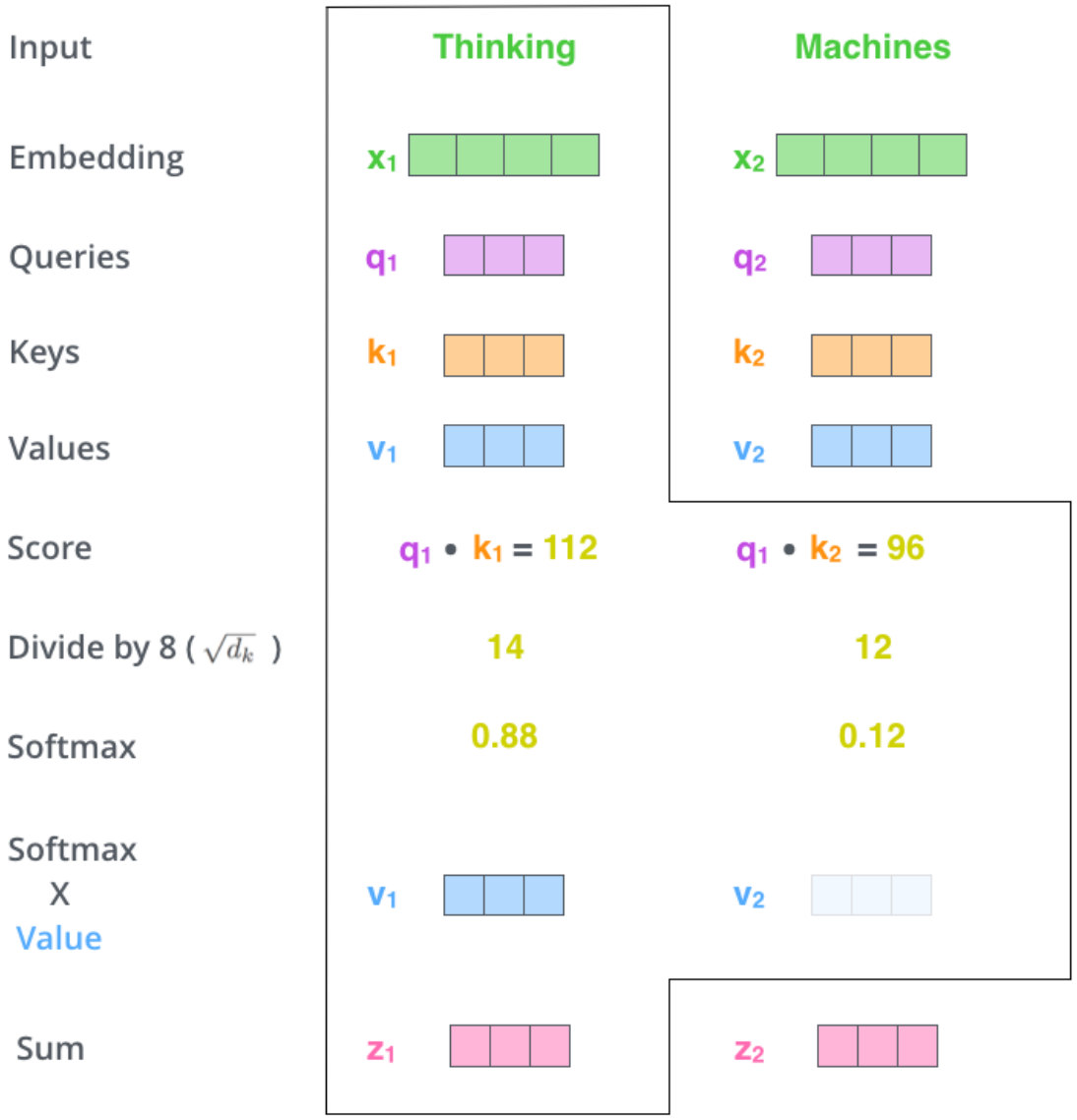
- Mean of sum = sum of means = $d_k * 0 = 0$
- Variance of sum = sum of variances = $d_k * 1 = d_k$
- To set the variance to 1, simply divide by $\sqrt{d_k}$!



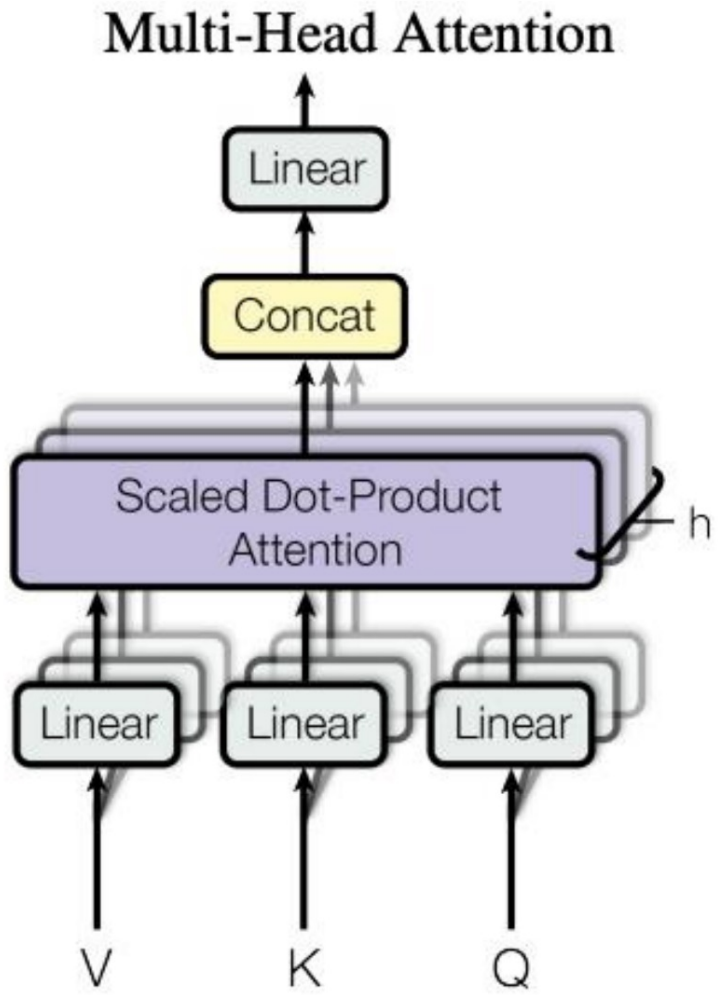
Self Attention

Step 5

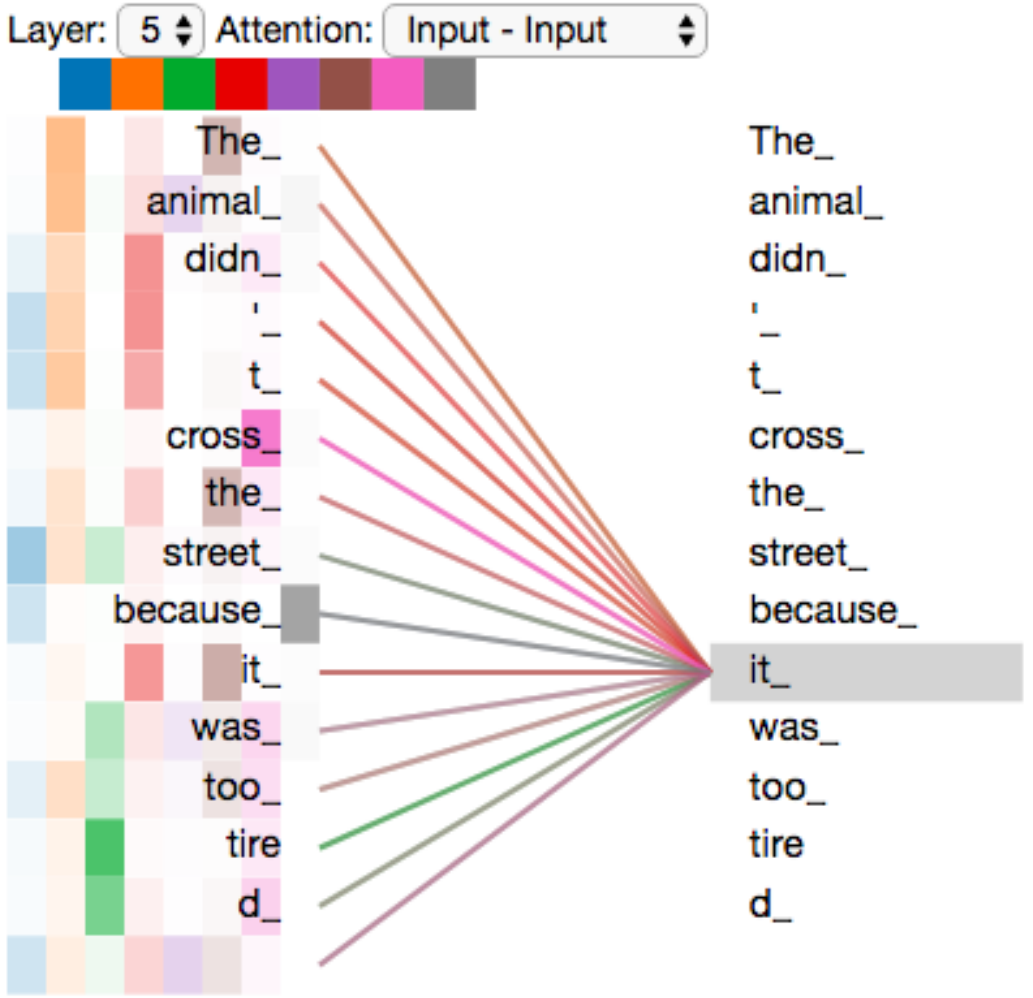
- 根据softmax计算的权重，对句中所有value **加权求和**
- 通过乘以较小的权重，忽略不相关的单词
- 将我们想要关注的单词编码到最终的结果中



Self Attention



[Vaswani et al. 2017]

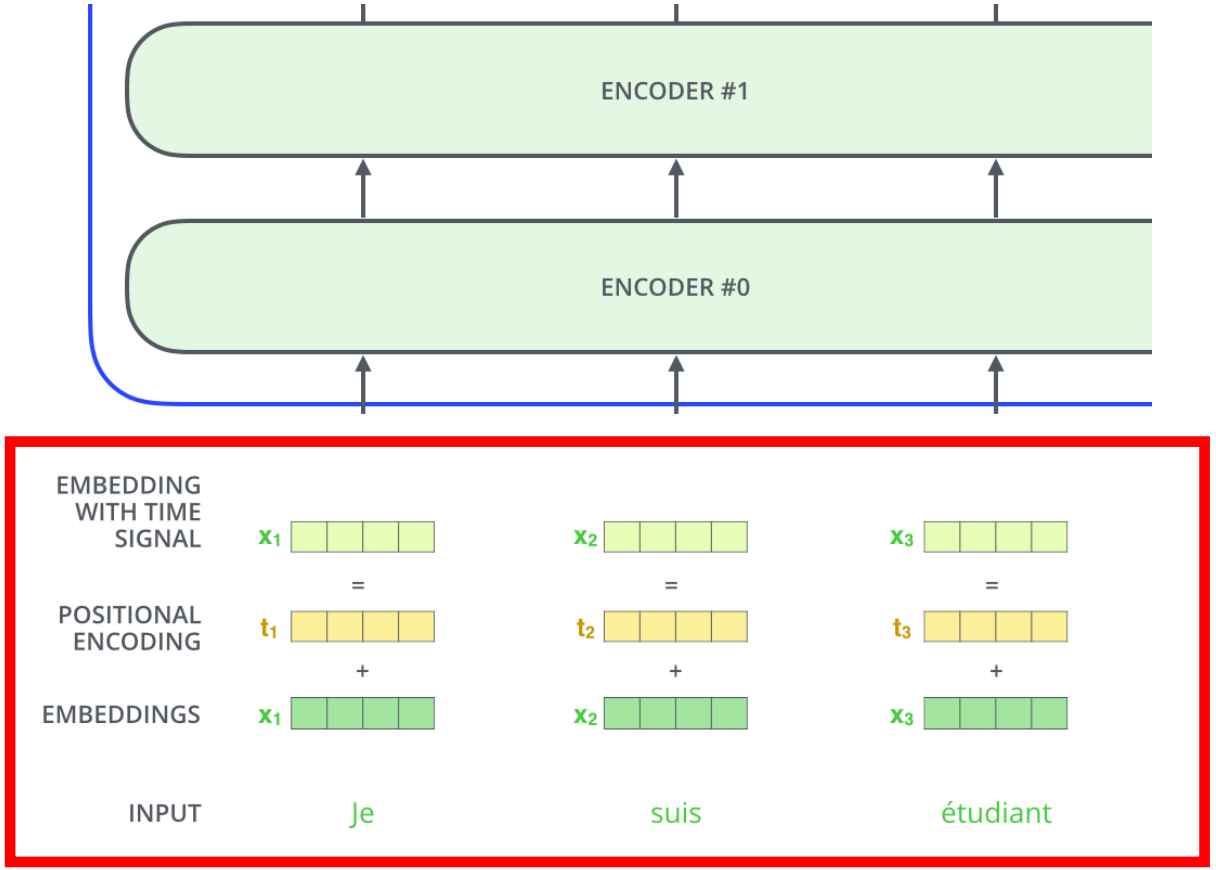


多头自注意力

Positional Encoding

- **Positional Encoding**:

  由于自注意力机制没有考虑序列的顺序信息，我们需要对句子的顺序进行编码

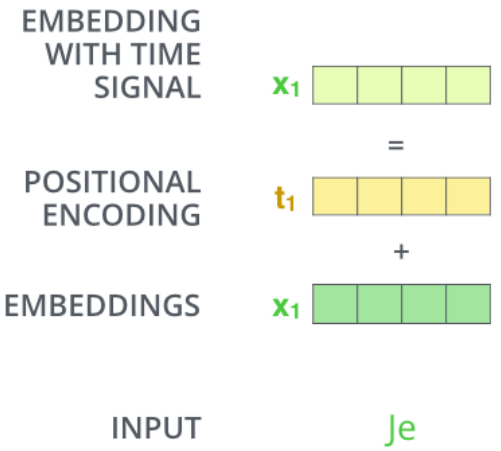


Positional Encoding

- 将第i个位置的词嵌入加上位置向量p_i
- 这种位置向量需要遵循一定的模型可以学习到的模式，帮助模型确定每个单词的位置或者序列中不同单词之间的距离

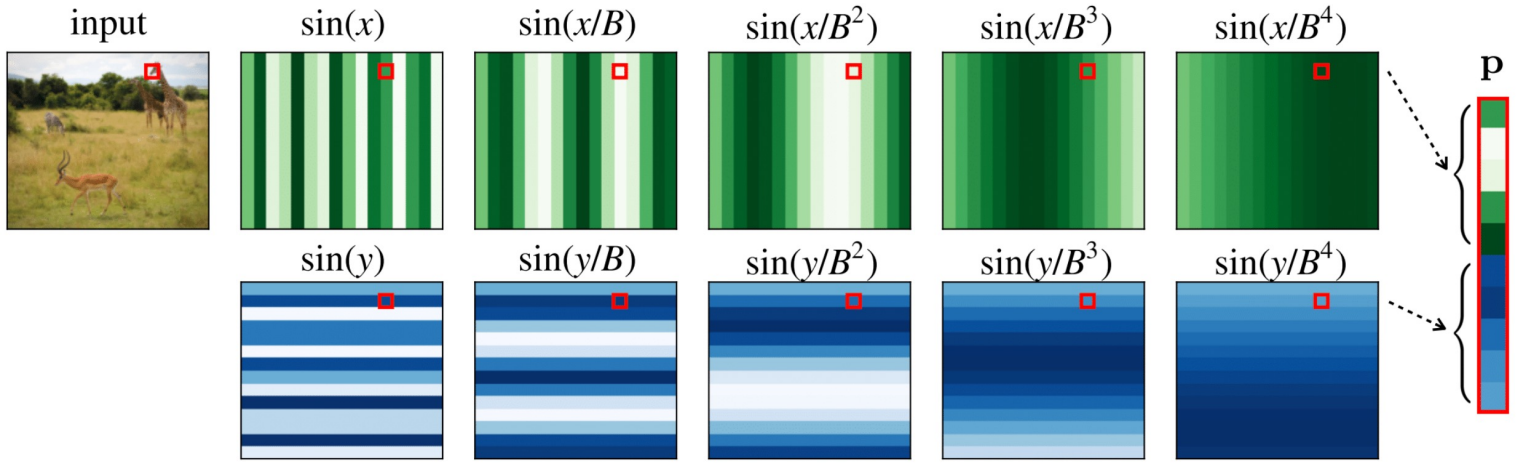
$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$

- **Key insight**：最重要的位置信息是单词之间的**相对关系**（如 “cat ”是 “eat ”之前的单词），而不是它们的**绝对位置**（如 “cat ”是句中的第二个单词）。



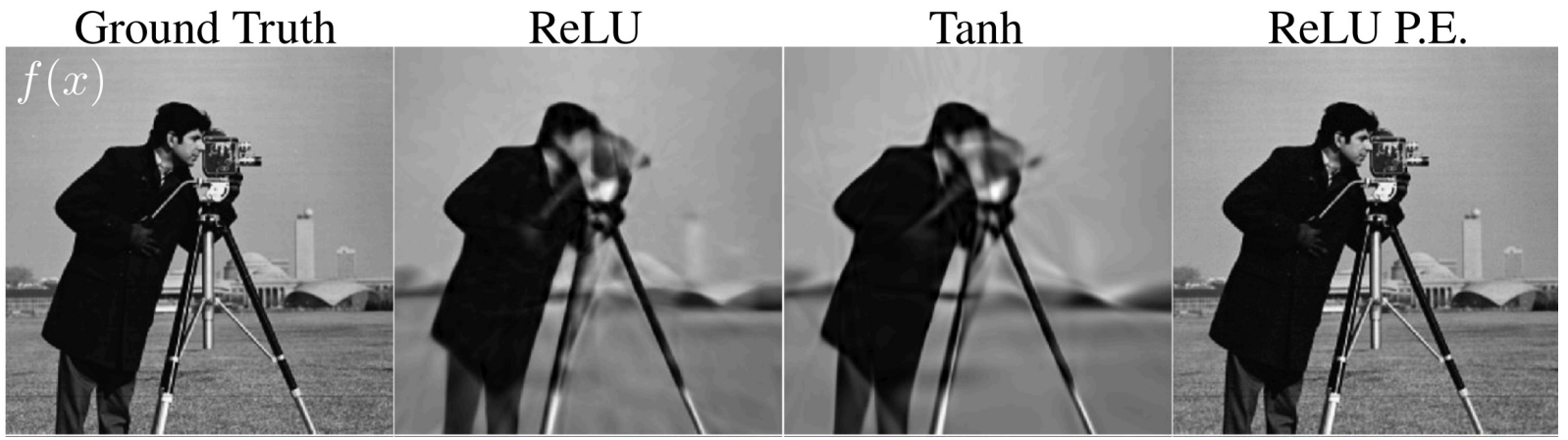
Positional Encoding

- 本质上是时域信号在傅立叶基上的投影

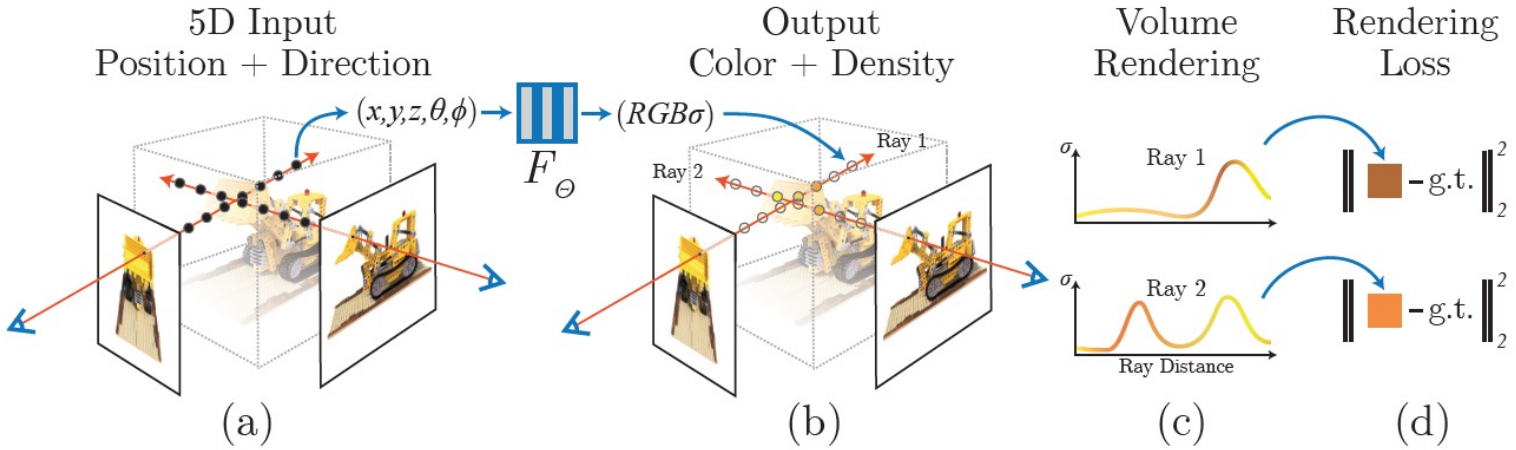


Positional Encoding

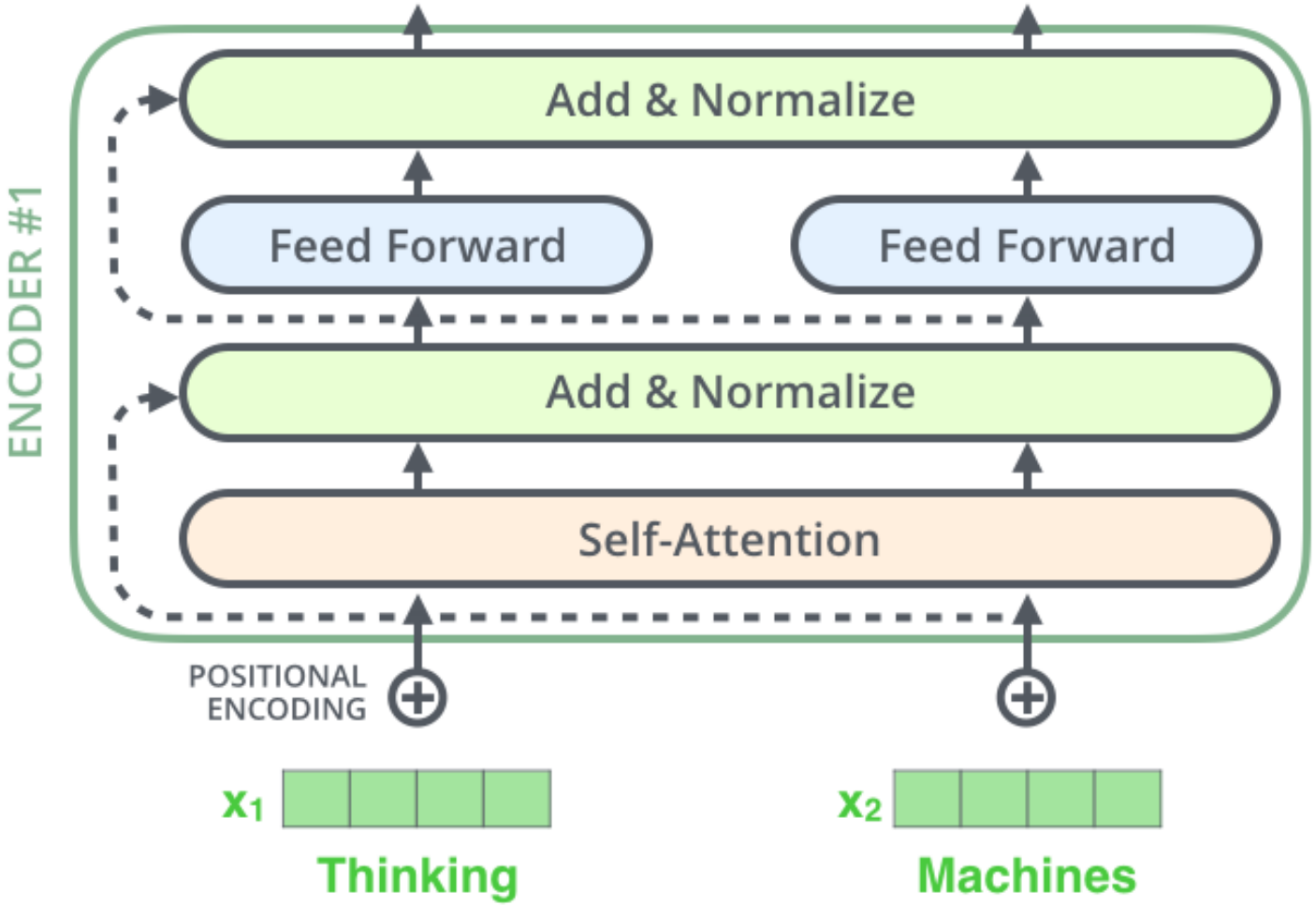
- 图像重建



- 三维重建

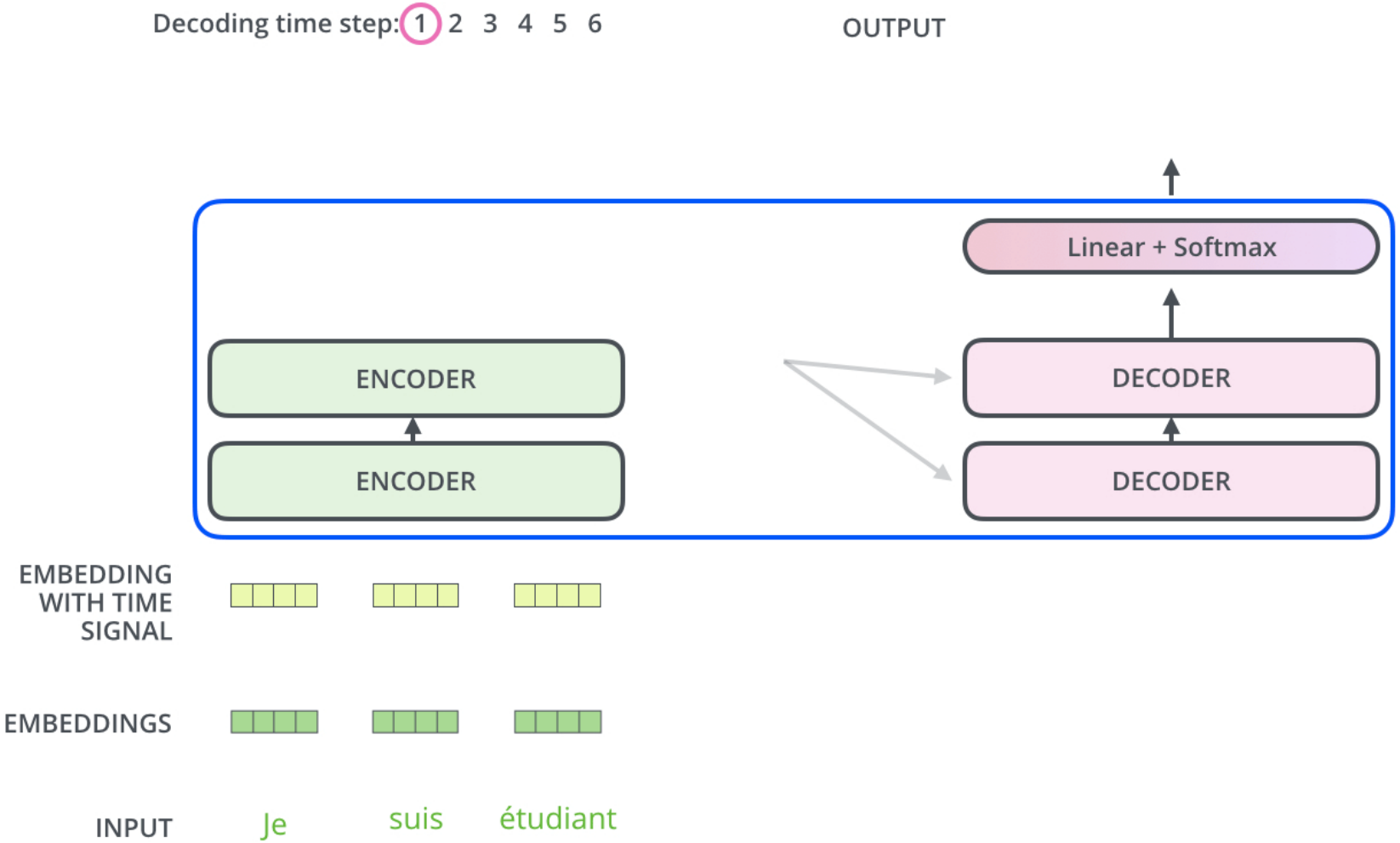


Encoder

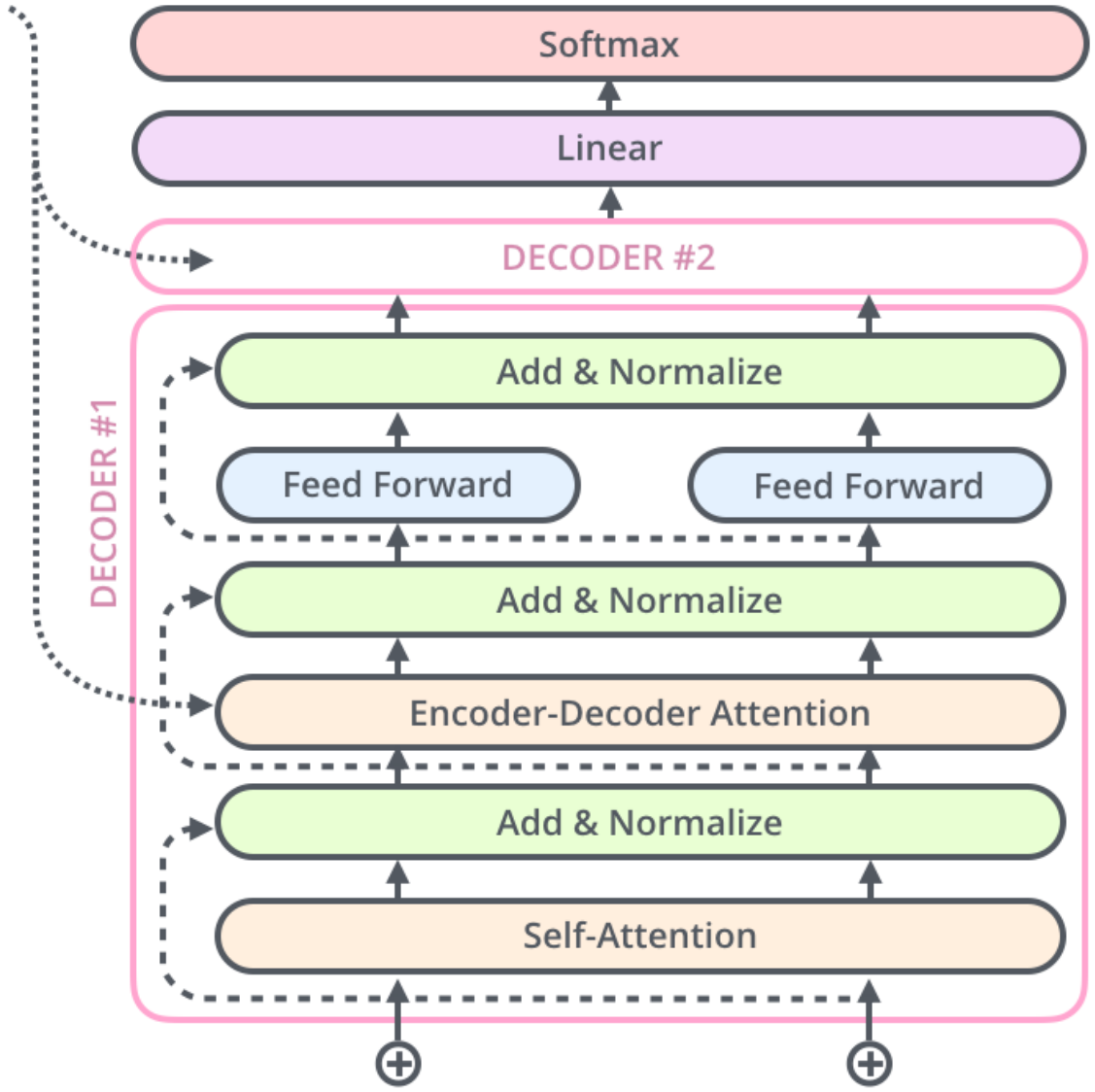


- **Feed Forward Neural Network**：捕捉输入数据中的复杂关系
- **Add & Normalize**：提高训练速度和稳定性
- **Self-Attention**：对词语之间的关系进行编码
- **Positional Encoding**：对词语的位置进行编码
- **Word Embedding**：将词语编码为数字

Encoder

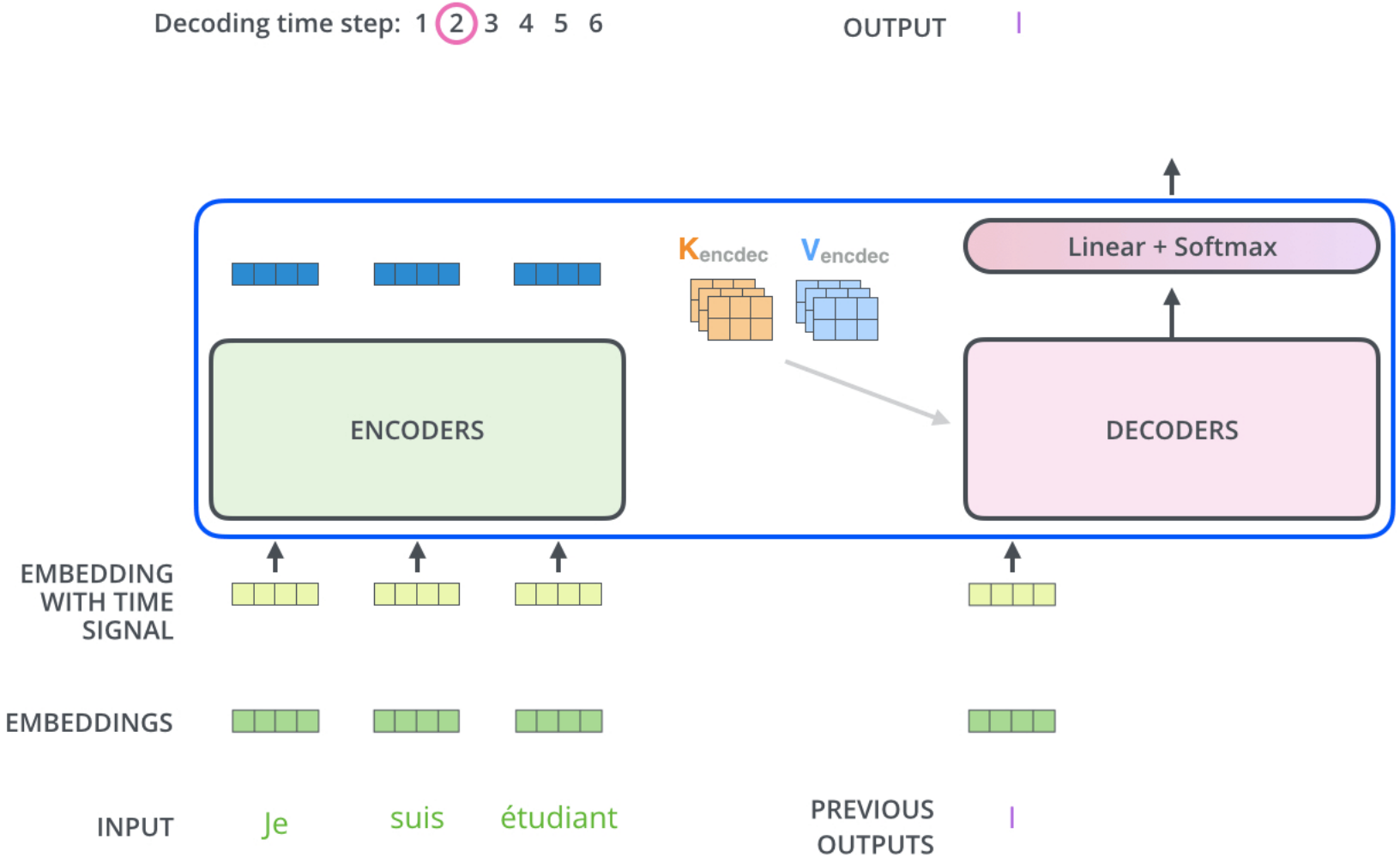

Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

Decoder



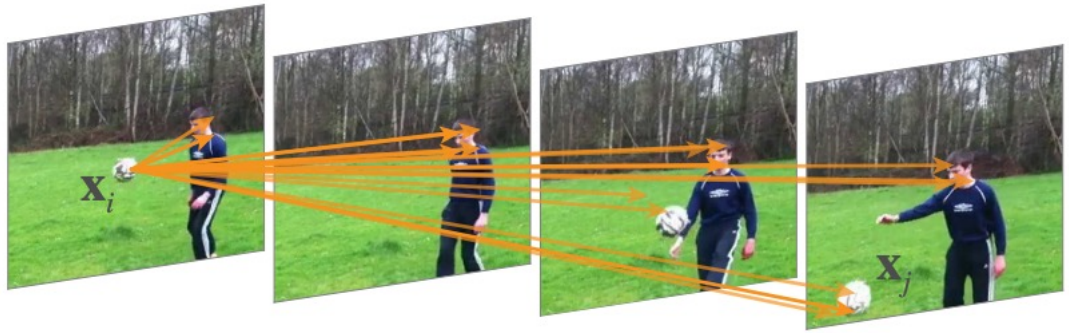
- **Softmax Layer**：将分数转化为概率分布
- **Linear Layer**：将decoder生成的向量映射到logits向量中，为输出词汇表中的每个词分配一个分数
- **Encoder-Decoder Attention**：追踪输入中的重要信息
- **Word Embedding, Positional Encoding, Self-Attention, …**

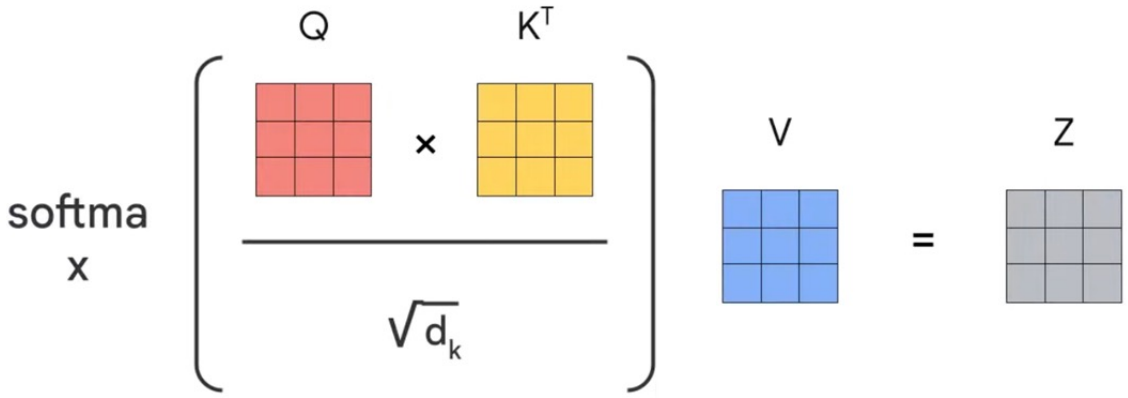
Decoder

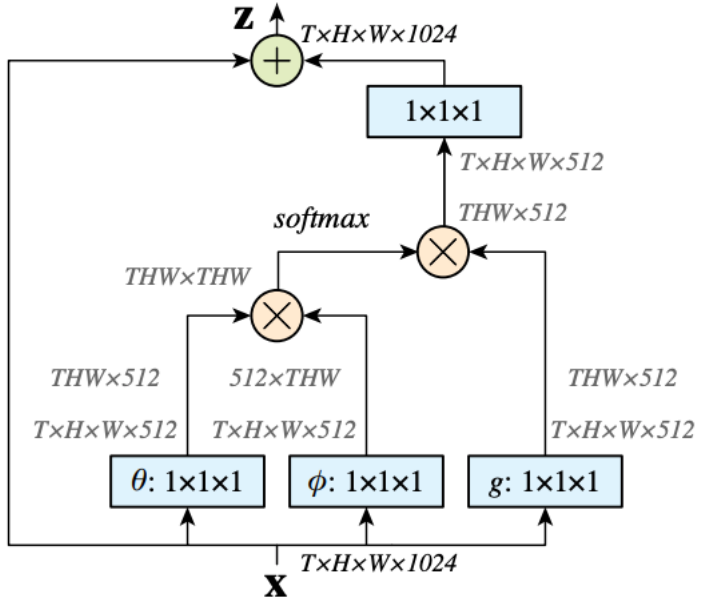

Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K encdec
V encdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Attention is all you need

- Non-local operation

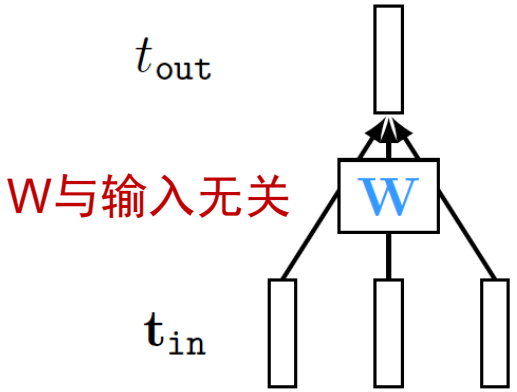






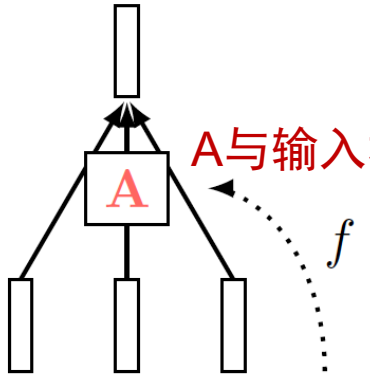
Attention is all you need

全连接层



W与输入无关

注意力层



A与输入相关

$$\mathbf{Q}_{\text{in}} = \mathbf{Z}_{\text{in}}\mathbf{W}_q \quad \triangleleft \text{ query matrix}$$
$$\mathbf{K}_{\text{in}} = \mathbf{Z}_{\text{in}}\mathbf{W}_k \quad \triangleleft \text{ key matrix}$$
$$\mathbf{V}_{\text{in}} = \mathbf{Z}_{\text{in}}\mathbf{W}_v \quad \triangleleft \text{ value matrix}$$
$$\mathbf{A} = f(\mathbf{t}_{\text{in}}) = \texttt{softmax}(\frac{\mathbf{Q}_{\text{in}}\mathbf{K}_{\text{in}}^T}{\sqrt{d}}) \quad \triangleleft \text{ attention matrix}$$
$$\mathbf{Z}_{\text{out}} = \mathbf{A}\mathbf{V}_{\text{in}}$$

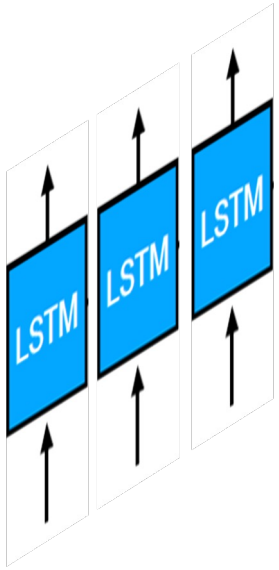
相当于在WX的基础上乘了一个attention权重A，其中A只与输入X自己有关（即句子X中各单词间的自注意力）

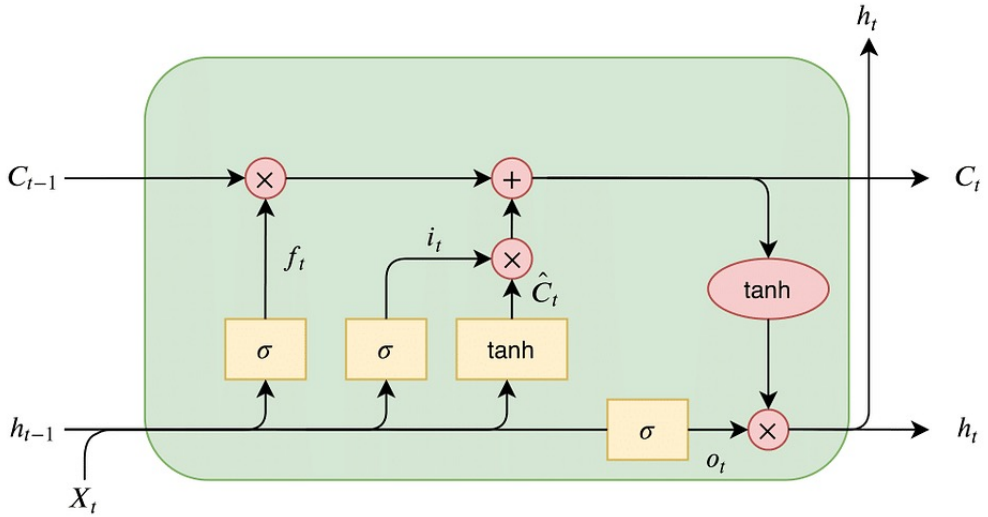
V相当于全连接层的WX

W is free *parameters*.

A is a function of some input *data*. The data tells us which tokens to attend to (assign high weight in weighted sum)

Transformer的理解



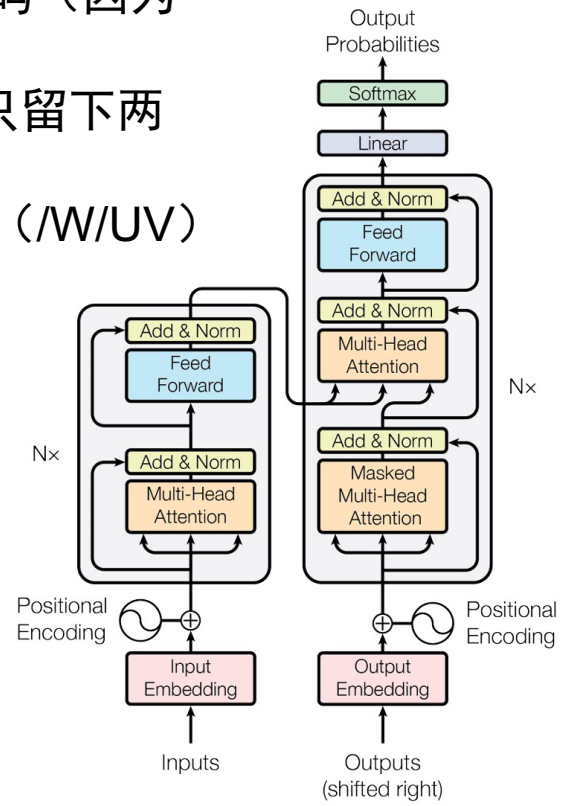


从MLP角度：

- Transformer很像MLP

从LSTM 到Transformer：

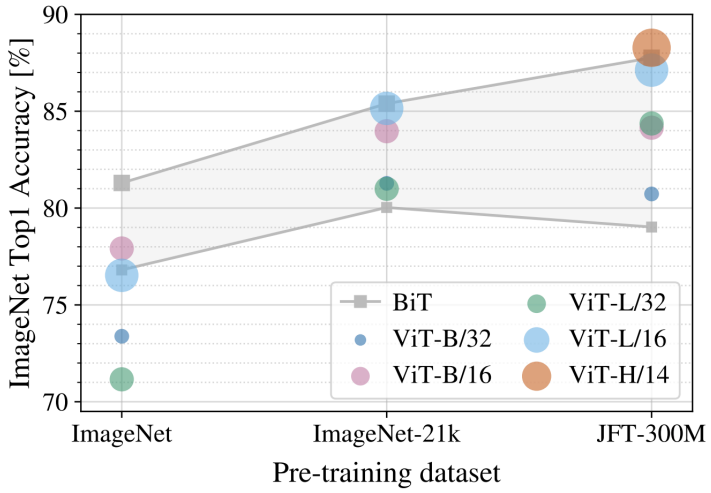
- 串行改并行，单头改多头
- 增加cosine/sine 位置编码（因为没有顺序输入了）
- 减少了内部门的数量，只留下两个门
- 都有三套参数，Wq/k/v （/W/UV）

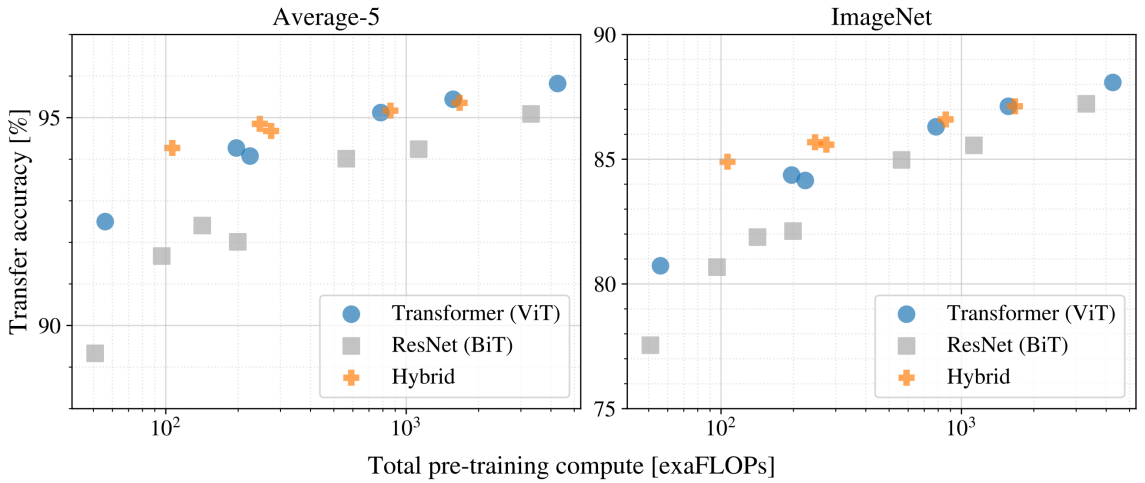


目录

预训练

- 因为（并行）自注意力机制和相同层叠结构，Transformer适合Scaling
- Transformer没有空间先验，需要大量预训练





预训练

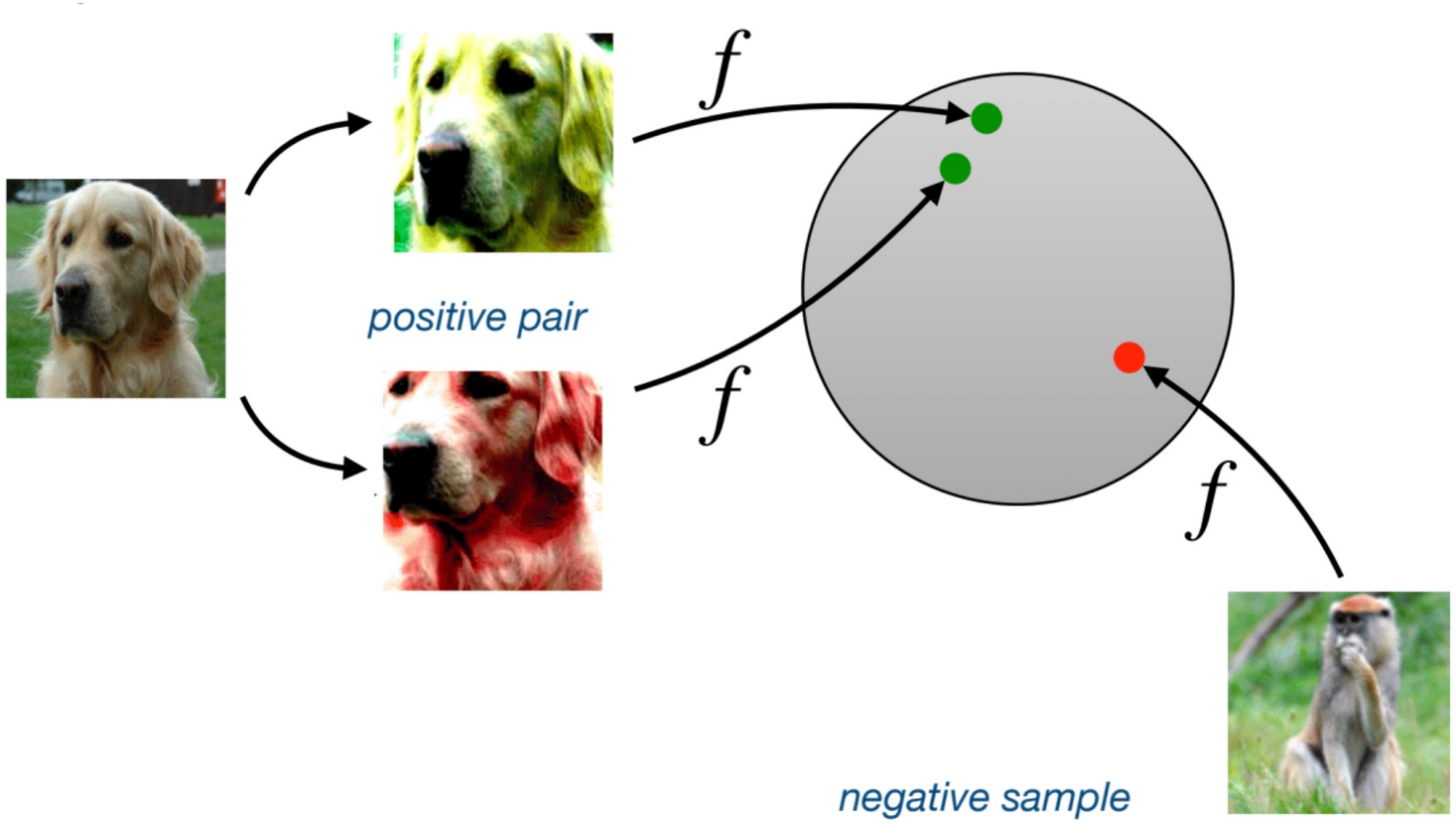
- 什么是好的表征?

- “Generally speaking, a good representation is one that makes a subsequent learning task easier.” — Deep Learning, Goodfellow et al. 2016

- 1. Concentration/Alignment: Data from the same class is close together
- 2. Separation: Classes are well separated
- 3. Robustness to irrelevant perturbations

预训练：自监督

- SimCLR：利用正负样本对

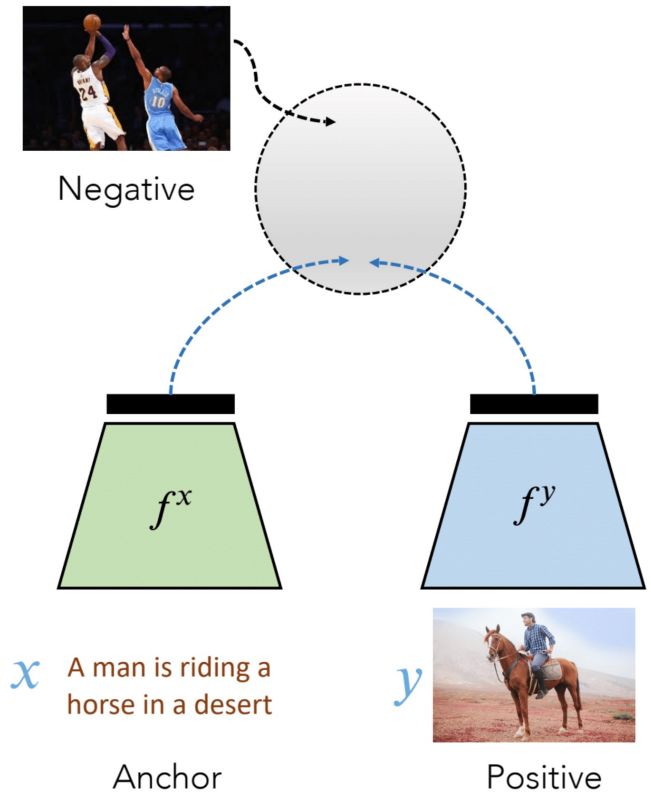


预训练：自监督

- 不用标签就可以识别图像中重要区域和物体关系
- DINO: self-supervised vision transformers

预训练：自监督

- CLIP：文字-图像对比



(x, y) are two "views" of the same scene

Language-Vision Representation Learning

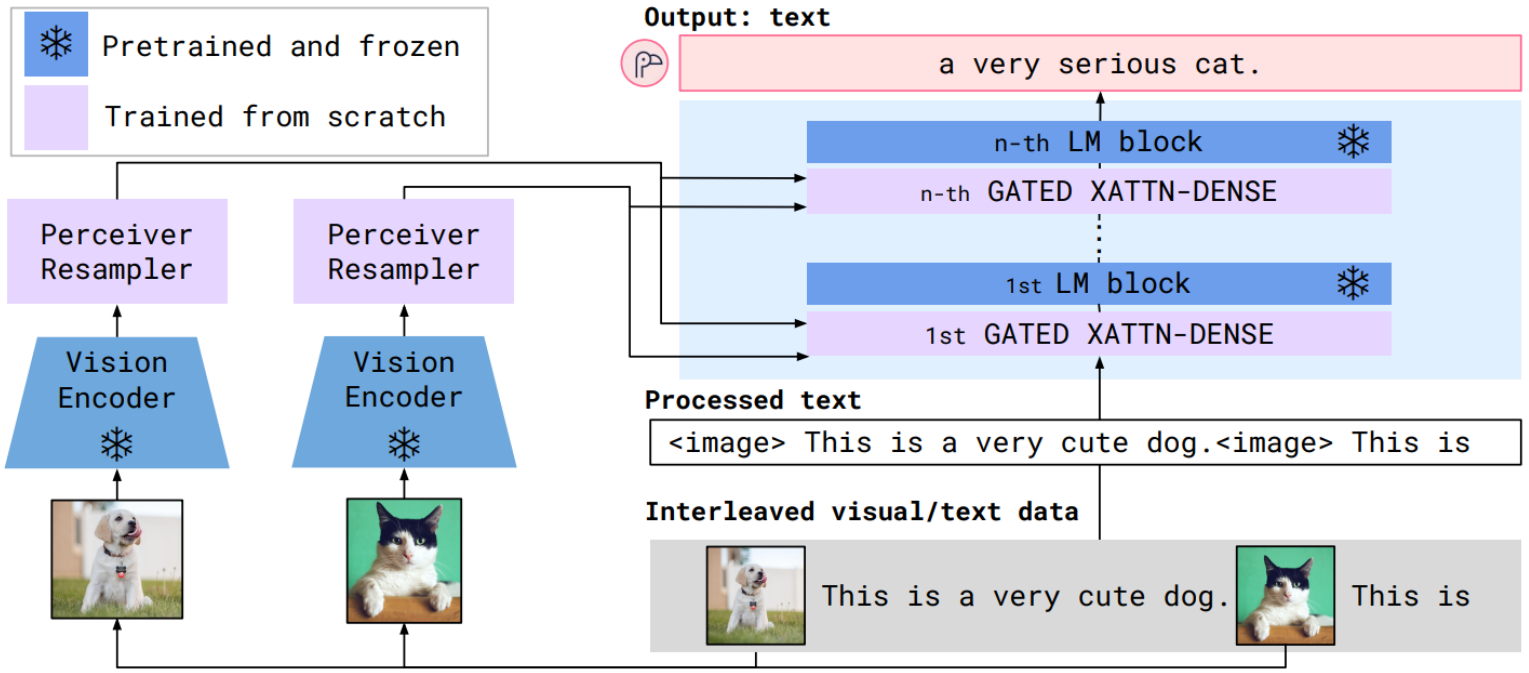
[Karpathy, Joulin, Fei-Fei 2014]

⋮

[CLIP, Radford, Kim et al. 2021]

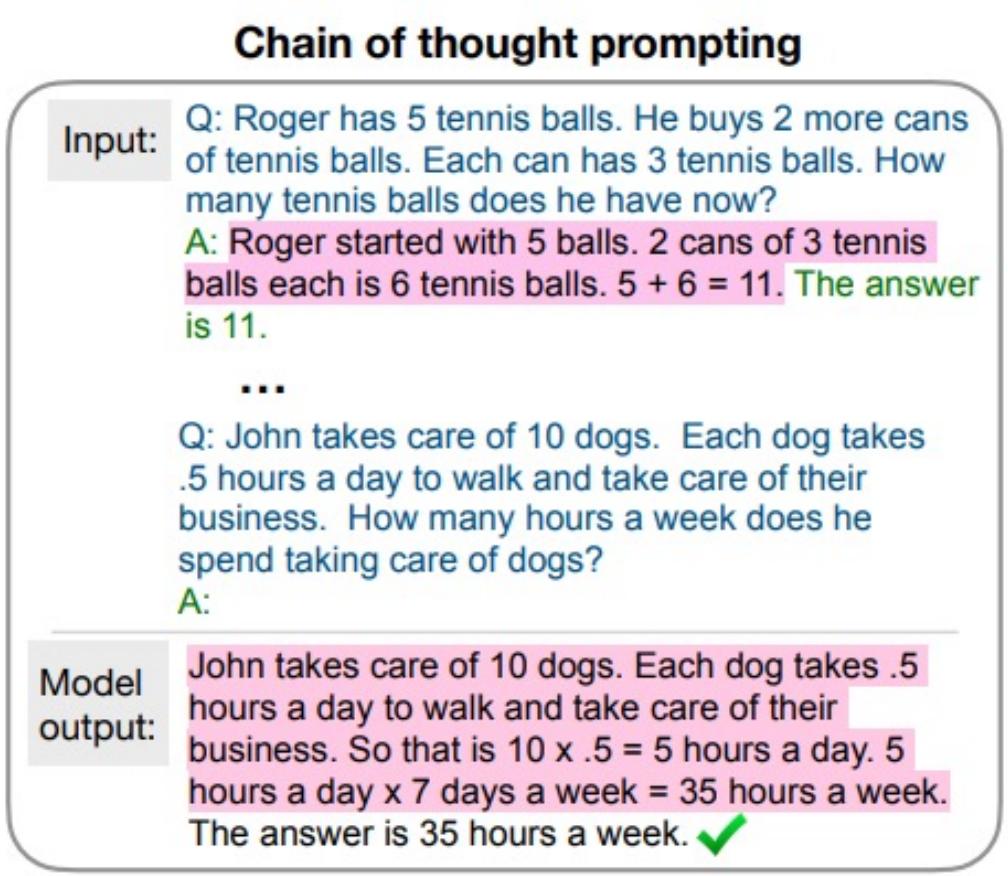
上下文学习

- Context Learning (aka few-shot learning)：不用更新参数处理新任务



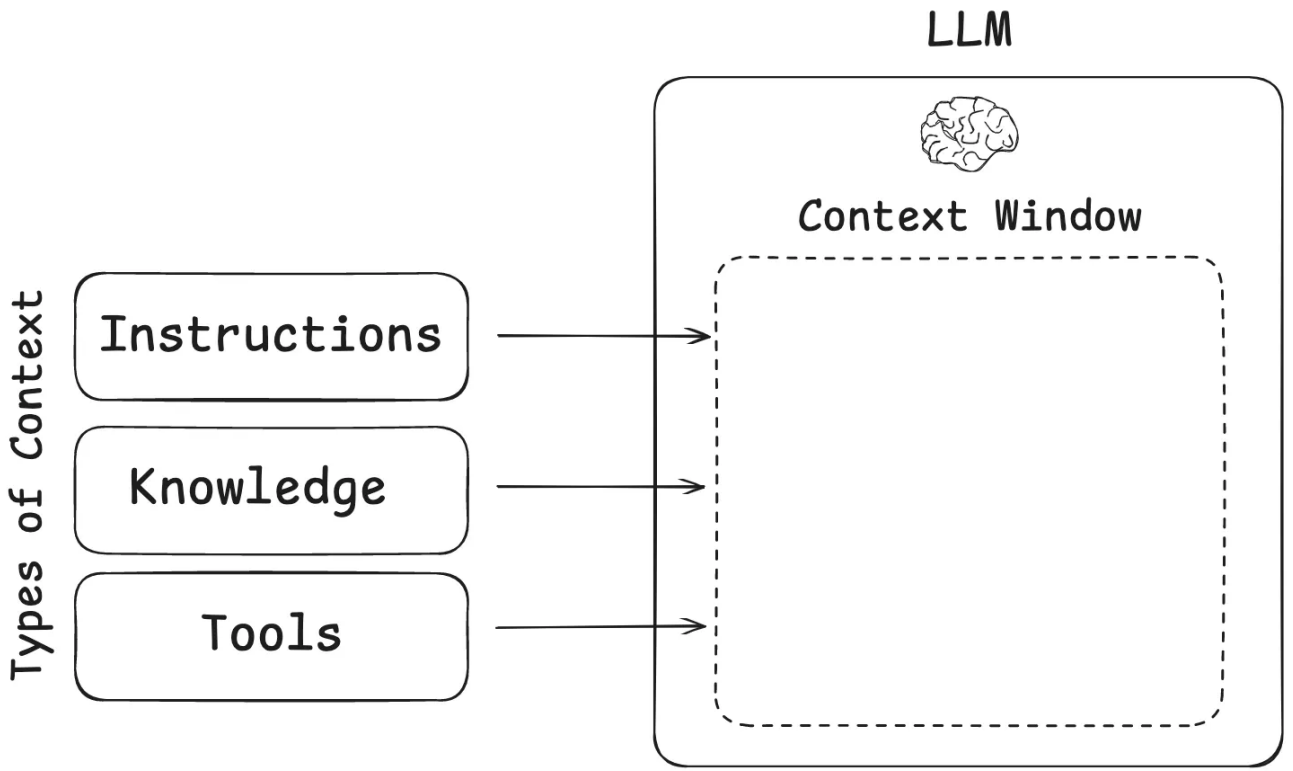
上下文学习

- Prompt Engineering：如何设计好prompt



上下文学习

- Context Engineering：如何设计好LLM需要的信息



小结

- RNN可建模时序数据，但1）训练梯度容消失并2）难以建模长距离关系
- LSTM可以缓解梯度消失问题
- Transformer并行建模长距离：Token，Attention和Positional Encoding
- Transformer需要预训练增加模型通用能力
- 上下文学习在不需要大量数据时应用到下游任务