上海交通大学

计算机视觉

教师：赵旭

班级：AI4701

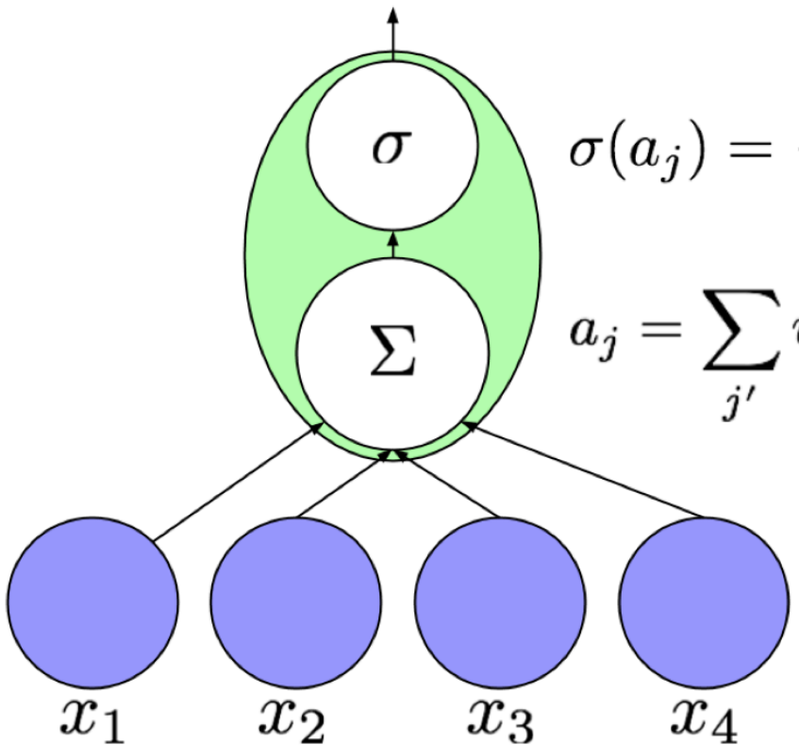
2024 春

19. ViT (Transformer)

主要内容

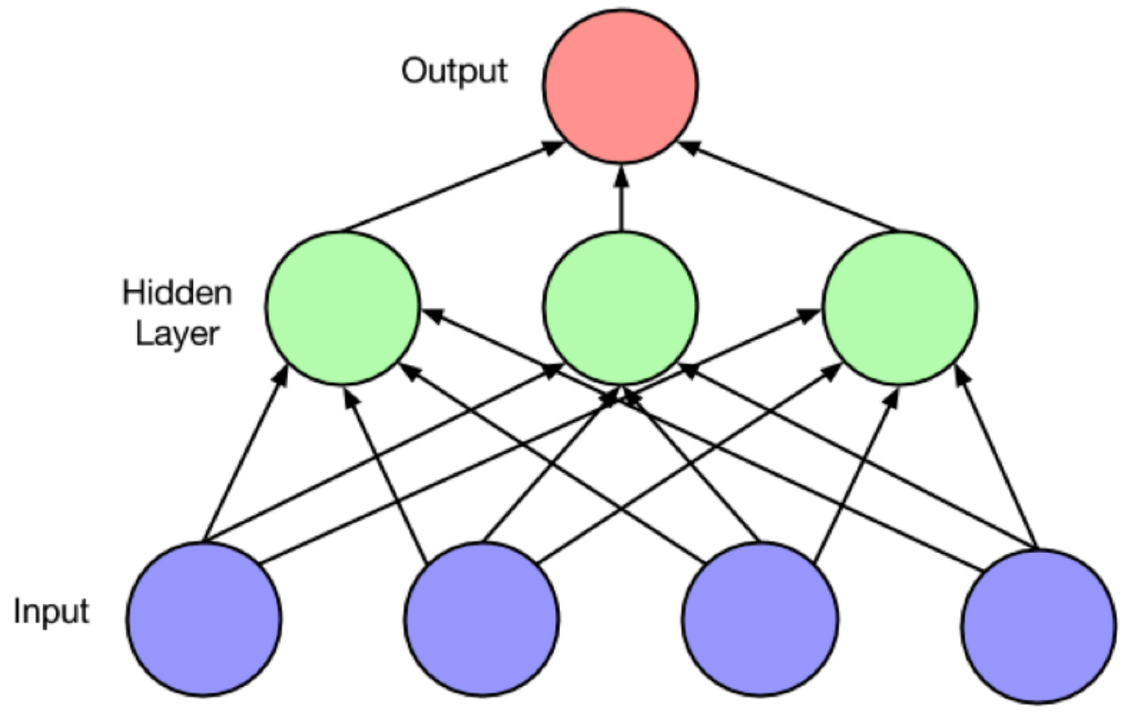
- 循环神经网络
- 注意力机制
- Transformer
- 视觉 Transformer

前馈神经网络 (FFNN)



$$\sigma(a_j) = \frac{1}{1 + e^{-a_j}}$$

$$a_j = \sum_{j'} w_{jj'} x_{j'}$$

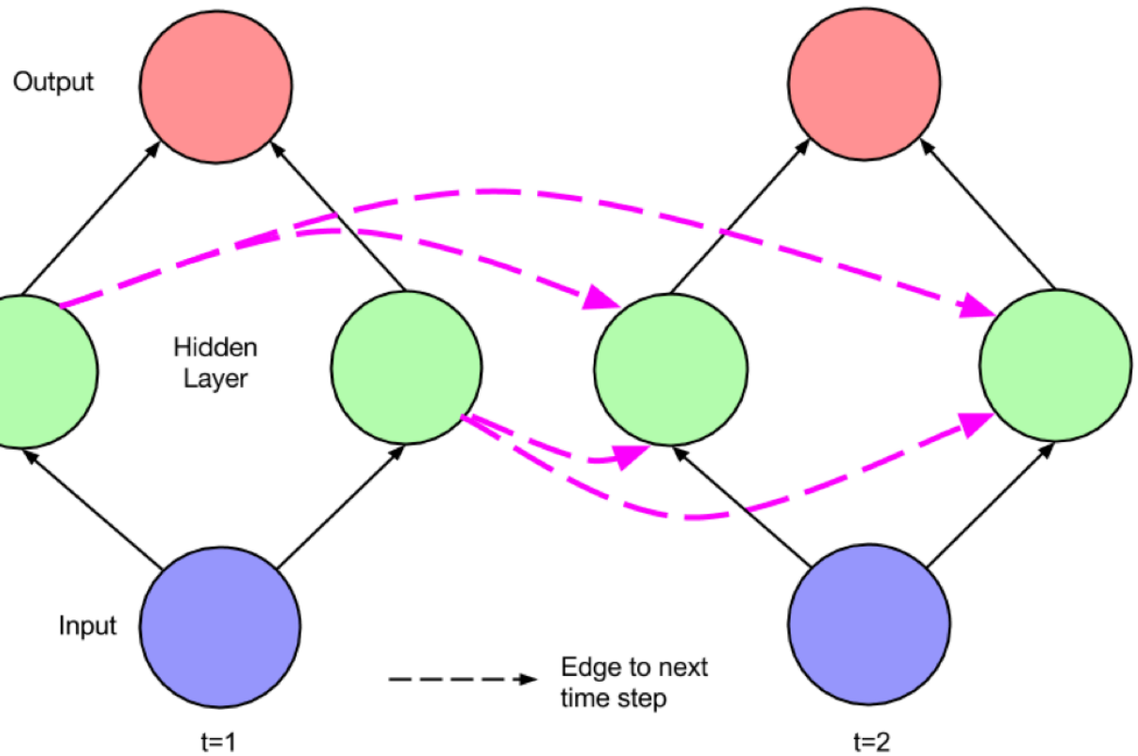


Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A critical review of recurrent neural networks for sequence learning." *arXiv preprint arXiv:1506.00019* (2015).

循环神经网络 (RNN)

- 表示输入序列的时间依赖性

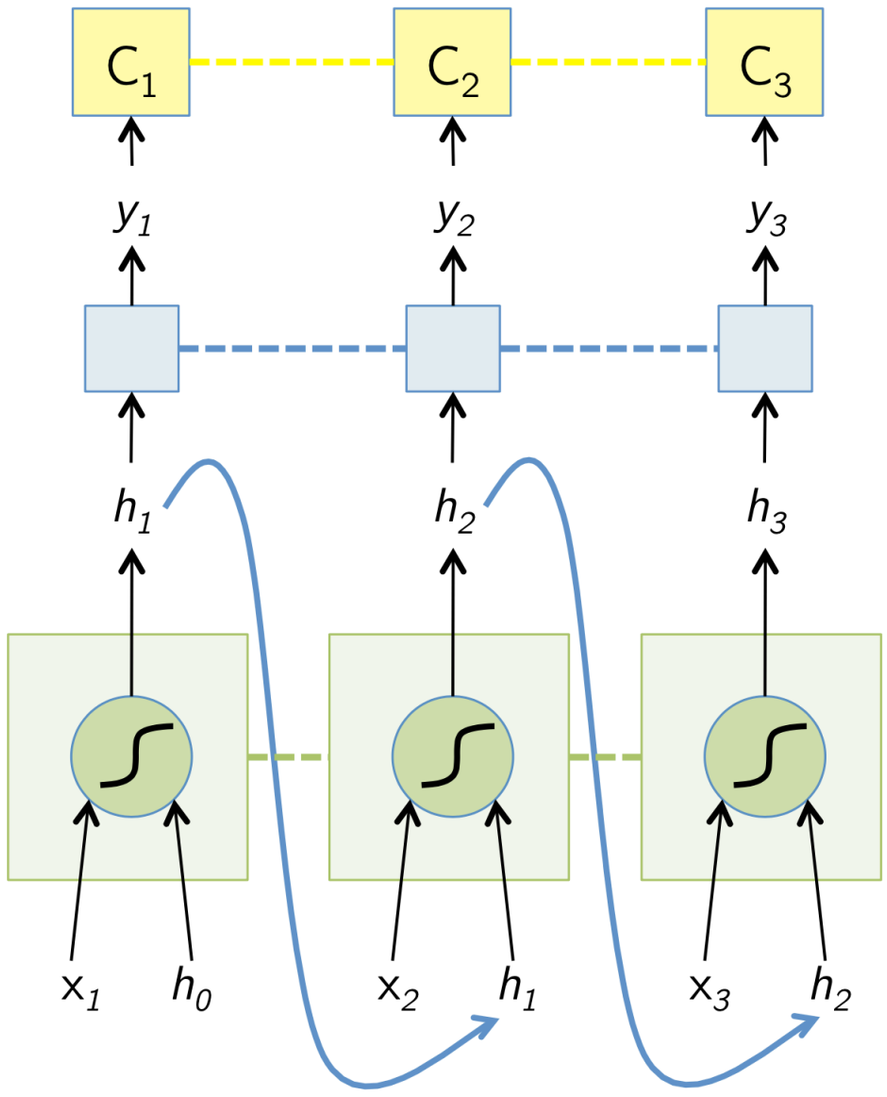
$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(W^{\text{yh}}\boldsymbol{h}^{(t)} + \boldsymbol{b}_y).$$



$$\boldsymbol{h}^{(t)} = \sigma(W^{\text{hx}}\boldsymbol{x}^{(t)} + W^{\text{hh}}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h)$$

Lipton, Zachary C., John Berkowitz, and Charles Elkan. "A critical review of recurrent neural networks for sequence learning." *arXiv preprint arXiv:1506.00019* (2015).

基础 RNN 结构



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights

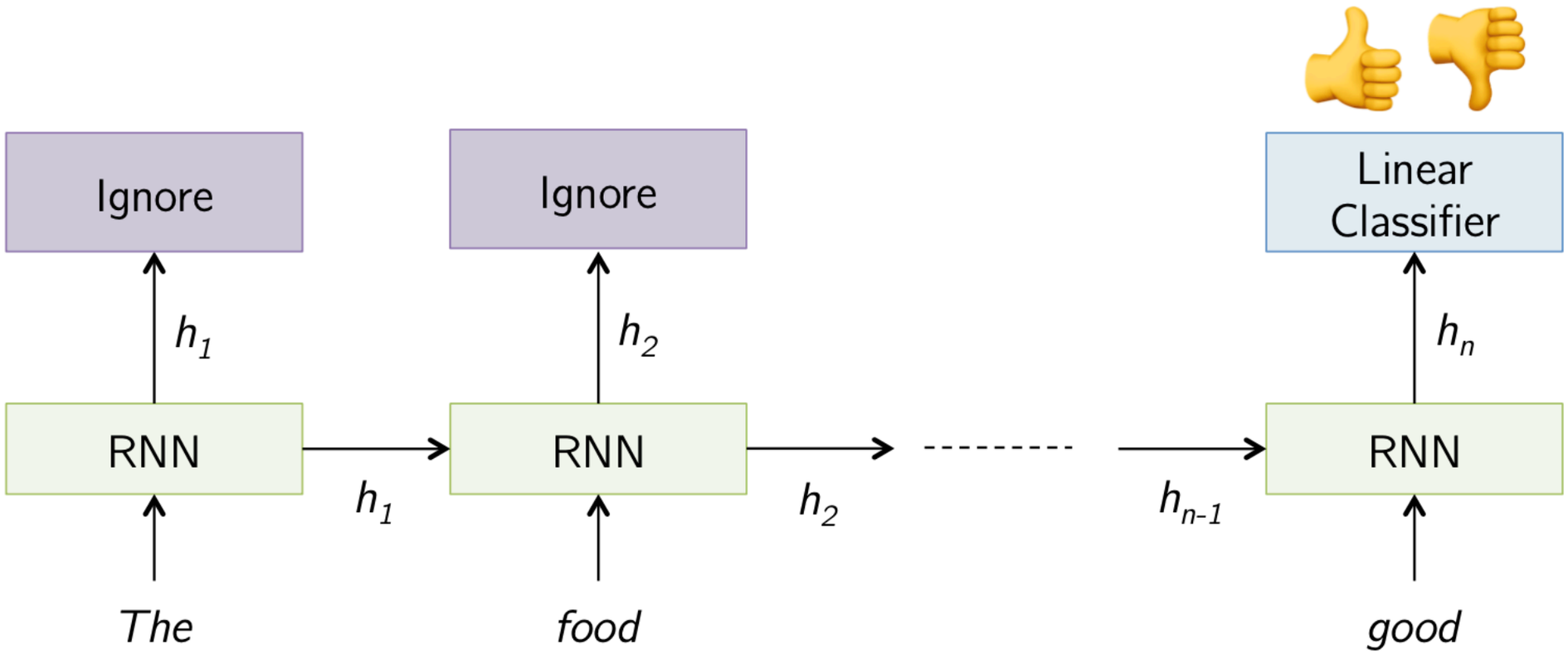
$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) = f(f(\mathbf{h}^{(t-2)}, \mathbf{x}^{(t-1)}; \theta), \mathbf{x}^{(t)}; \theta)$$

例：情感分类

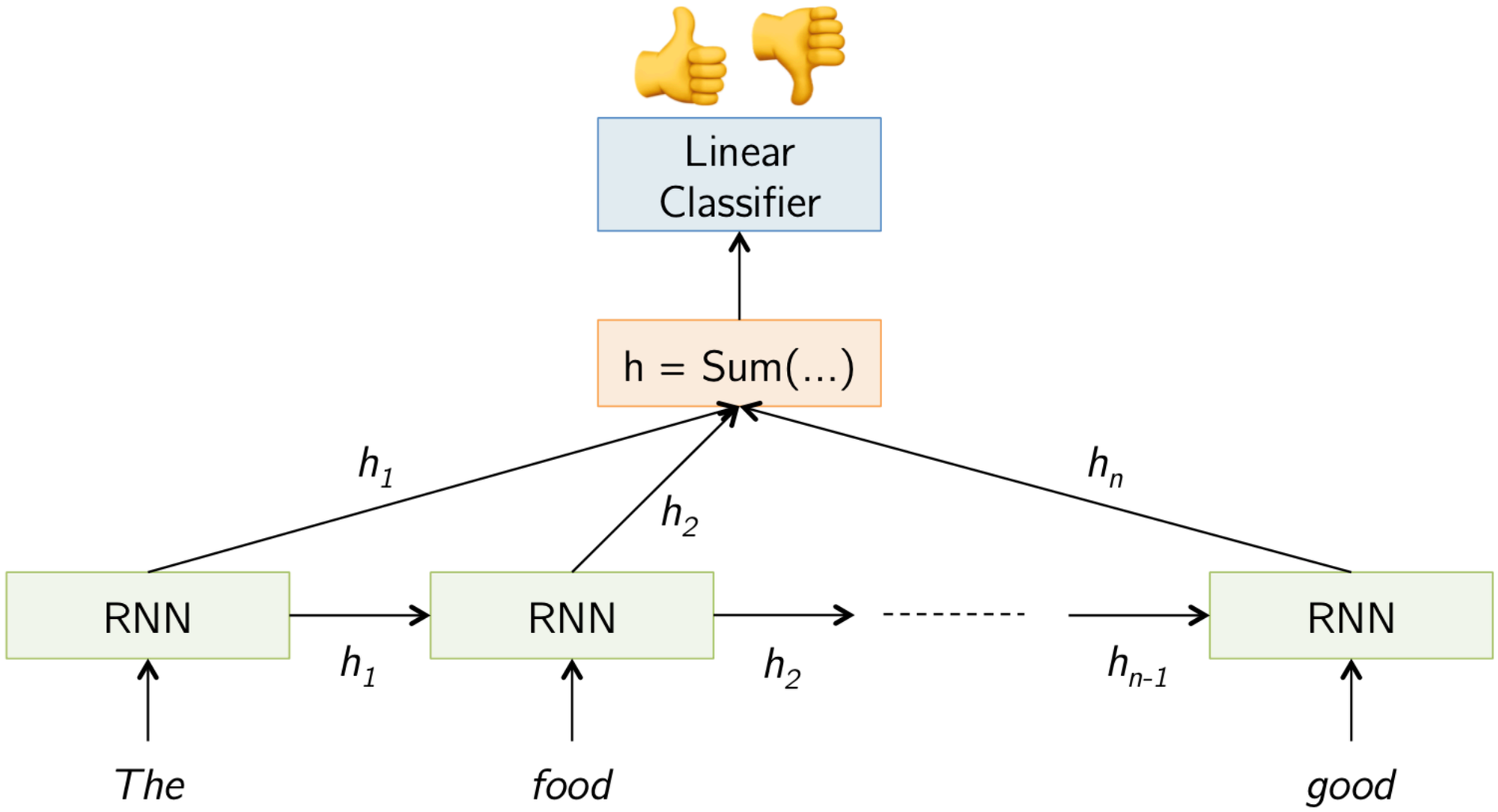
- 将Yelp上的餐厅评论或IMDB上的电影评论分类为正面或负面。
- **输入**: 多个单词，一个或多个句子
- **输出**:正面/负面分类
- “The food was really good”!

From Arun Mallya: Introduction to RNN

例：情感分类


Ignore
Ignore
Linear Classifier
h_1
h_2
h_n
RNN
RNN
RNN
h_1
h_2
h_{n-1}
The
food
good

例：情感分类



From Arun Mallya: Introduction to RNN

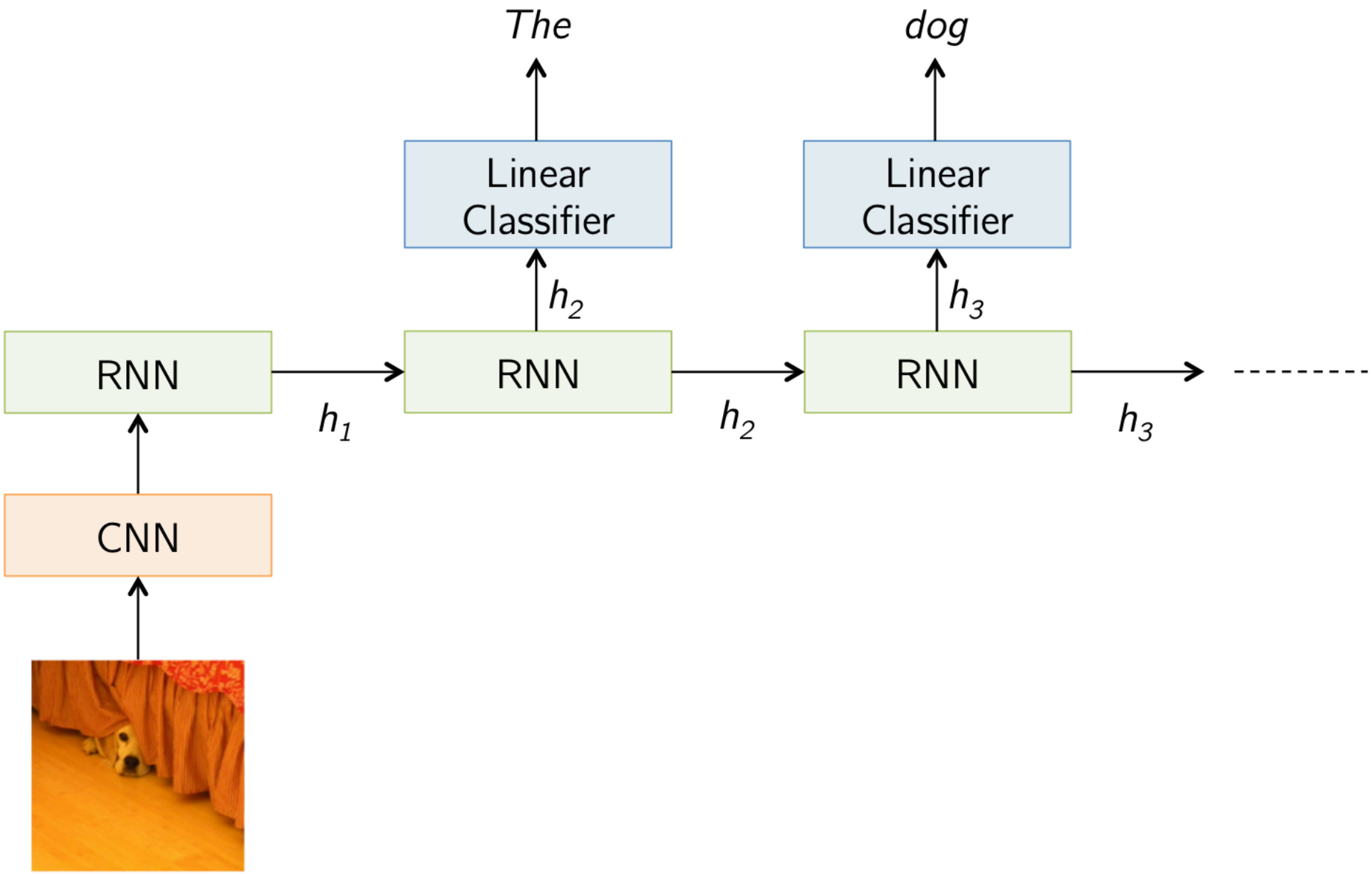
例: 图片描述

- 给定一张图片，生成描述其内容的句子。
- **输入**: 图像特征 (由卷积神经网络提取得到)
- **输出**: 多个单词 (不妨视为一句话)


: The dog is hiding

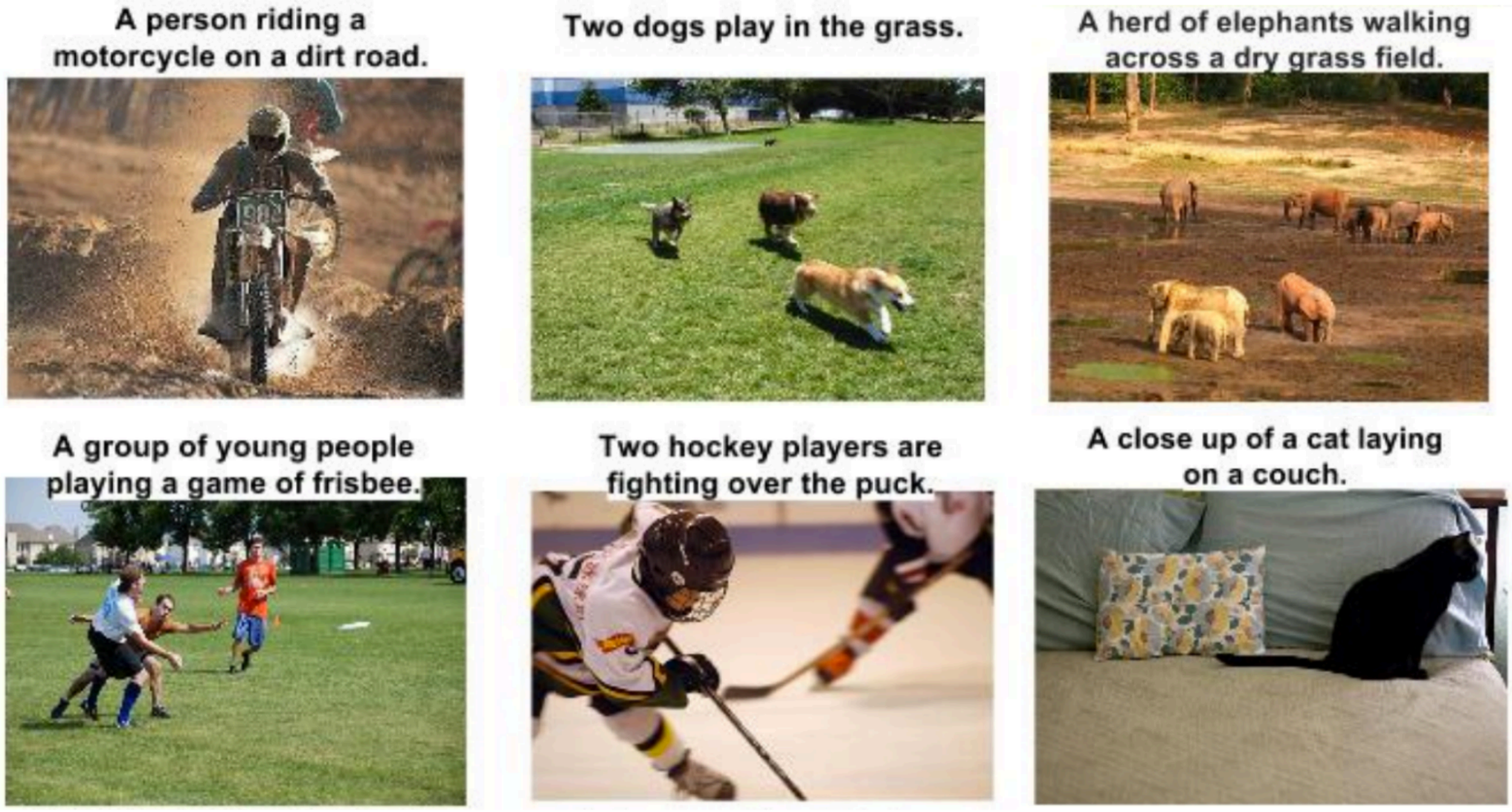
From Arun Mallya: Introduction to RNN

例: 图片描述



From Arun Mallya: Introduction to RNN

RNN 用于图像描述



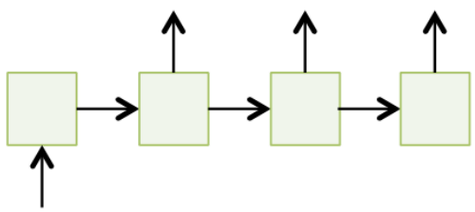
[Show and Tell: A Neural Image Caption Generator, CVPR 15]

RNN: 输入输出的多种形式

单对单

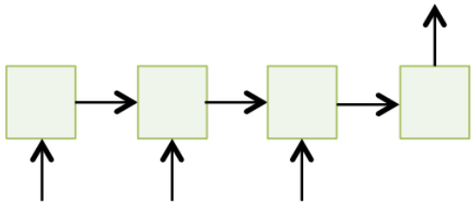
前向网络

单对多



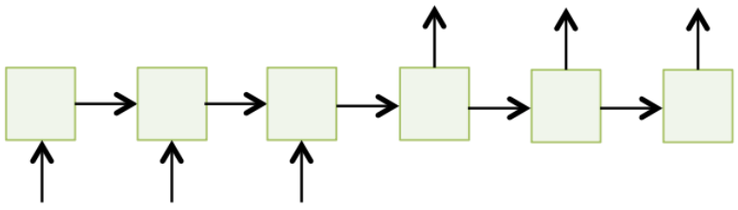
图像描述

多对单

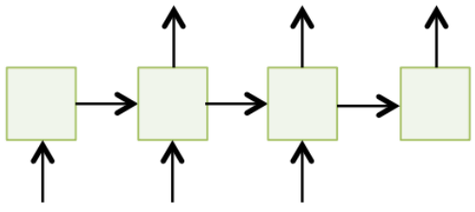


情感分类

多对多



翻译



图像描述

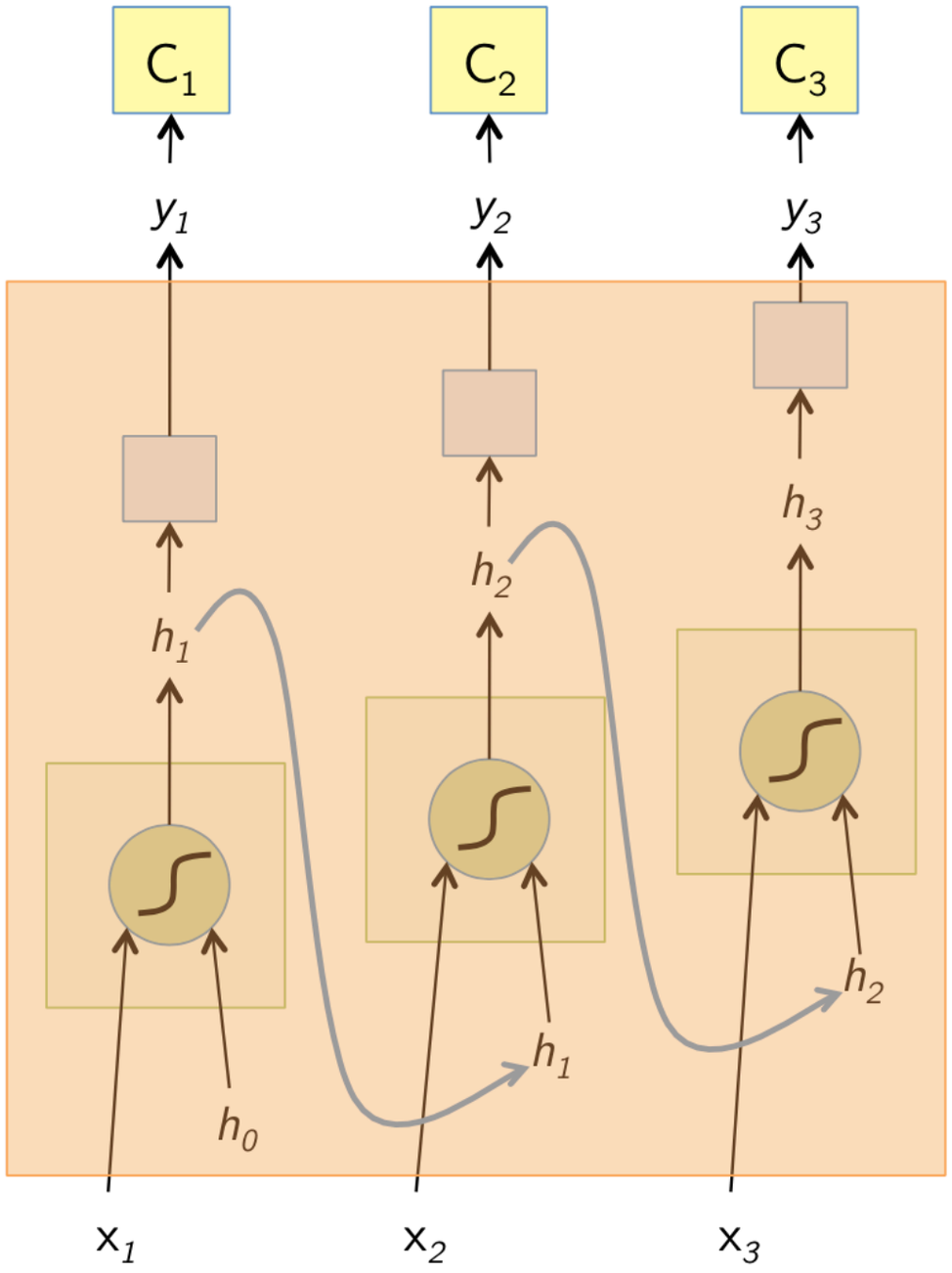
注意: 我们可能会故意为特定的问题构建专用的输入输出形式，以便更容易训练或获得更好的性能。例如，在每个时间步骤中，为图像描述任务提供前一个单词作为输入。（“单对多” 到 “多对多”模式）

From Arun Mallya: Introduction to RNN

RNN训练：时序反向传播算法（BPTT）

- 训练RNN使用的方法之一。
- 在前向传播过程中使用的展开网络被视为一个大型的前馈网络。
- 这个展开的网络接受整个时间序列作为输入。
- 然后在展开网络中的每个副本上计算权重更新，然后将这些更新求和（或平均），最后应用到RNN的权重上。

From Arun Mallya: Introduction to RNN



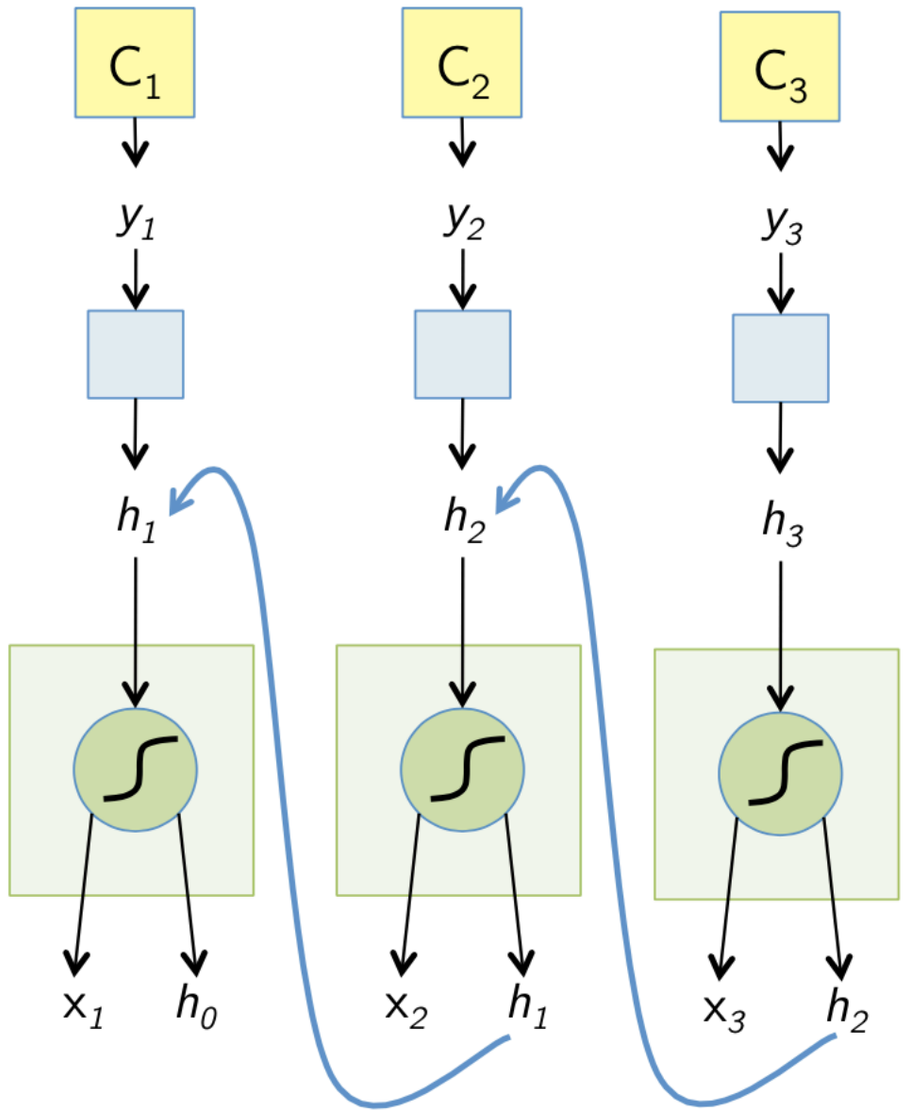
1.将展开的网络看作一个大的前向网络

2.该网络把整个序列作为输入

3.利用反向传播计算梯度

4.更新共享权重

BPTT：RNN反向传播算法



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

$$\frac{\partial C_t}{\partial h_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right)$$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)$$

From Arun Mallya: Introduction to RNN

RNN的注意力机制

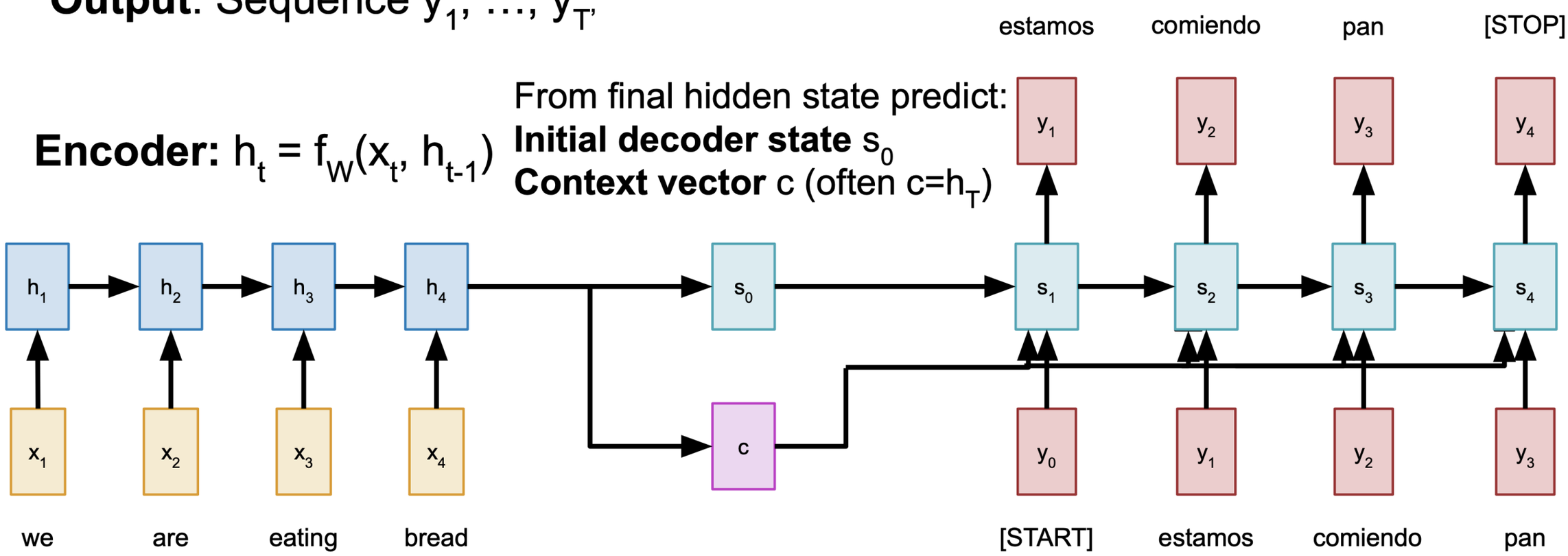
基于RNN实现的序列到序列转换（翻译任务）

Input: Sequence $x_1, \ldots x_T$
Output: Sequence $y_1, \ldots, y_{T'}$

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

From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c=h_T$)

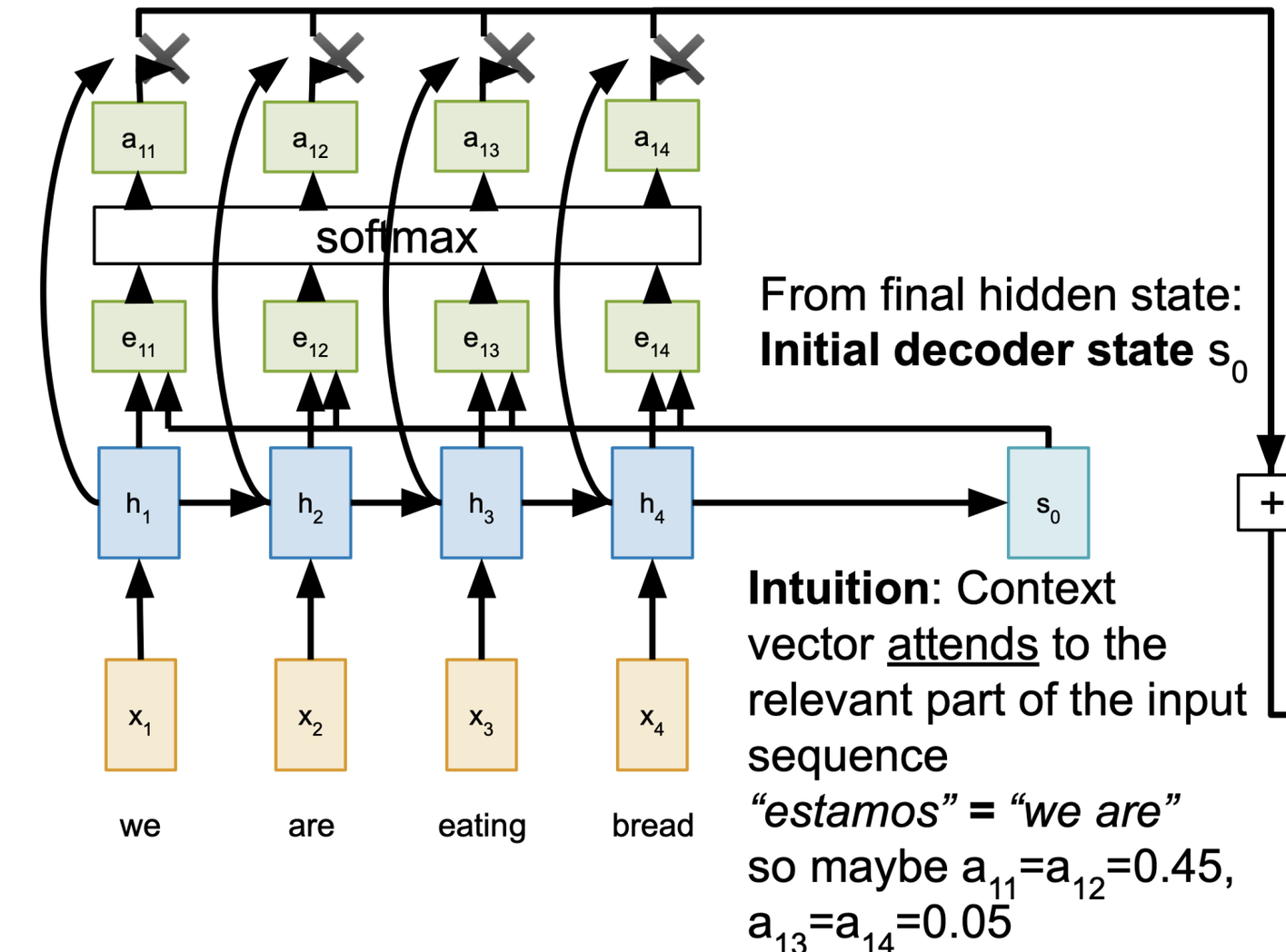
Decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c)$



From Fei-Fei Li et al: Attention and Transformers

RNN的注意力机制

基于RNN和注意力机制实现的序列到序列转换（翻译任务）



1. 计算带缩放的对齐分数

$e_{t,i} = f_{att}(s_{t-1}, h_i)$, f_{att} 是MLP

2. 归一化对齐分数以获得注意力权重

$0 < a_{t,i} < 1, \sum_i a_{t,i} = 1$

3. 计算上下文向量 (context vector) 与隐状态线性组合

$c_t = \sum_i a_{t,i} h_i$

4. 在解码器中使用上下文向量

$s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

全程可微，无需注意力真值监督！

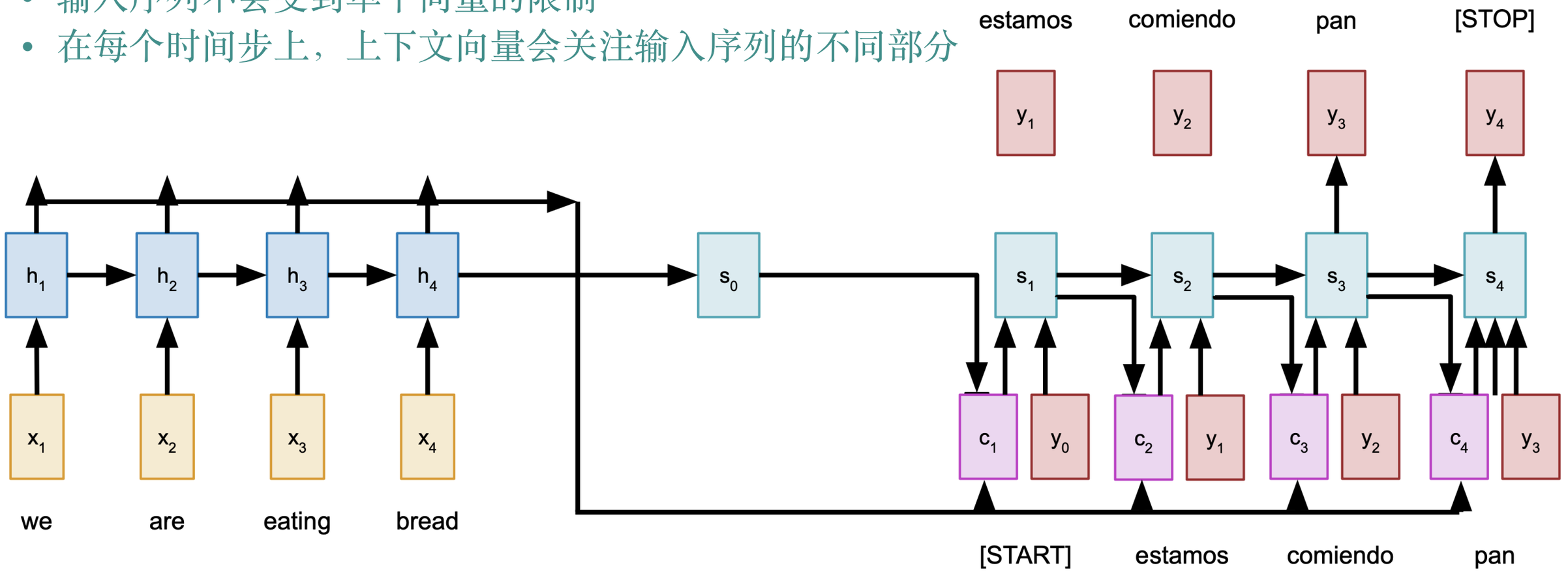
From Fei-Fei Li et al: Attention and Transformers

RNN的注意力机制

基于RNN和注意力机制实现的序列到序列转换（翻译任务）

在解码器的每个时间步上用不同的上下文向量 c_i

- 输入序列不会受到单个向量的限制
- 在每个时间步上，上下文向量会关注输入序列的不同部分



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

From Fei-Fei Li et al: Attention and Transformers

RNN的注意力机制

基于RNN和注意力机制实现的序列到序列转换（翻译任务）

例：英语-法语翻译任务

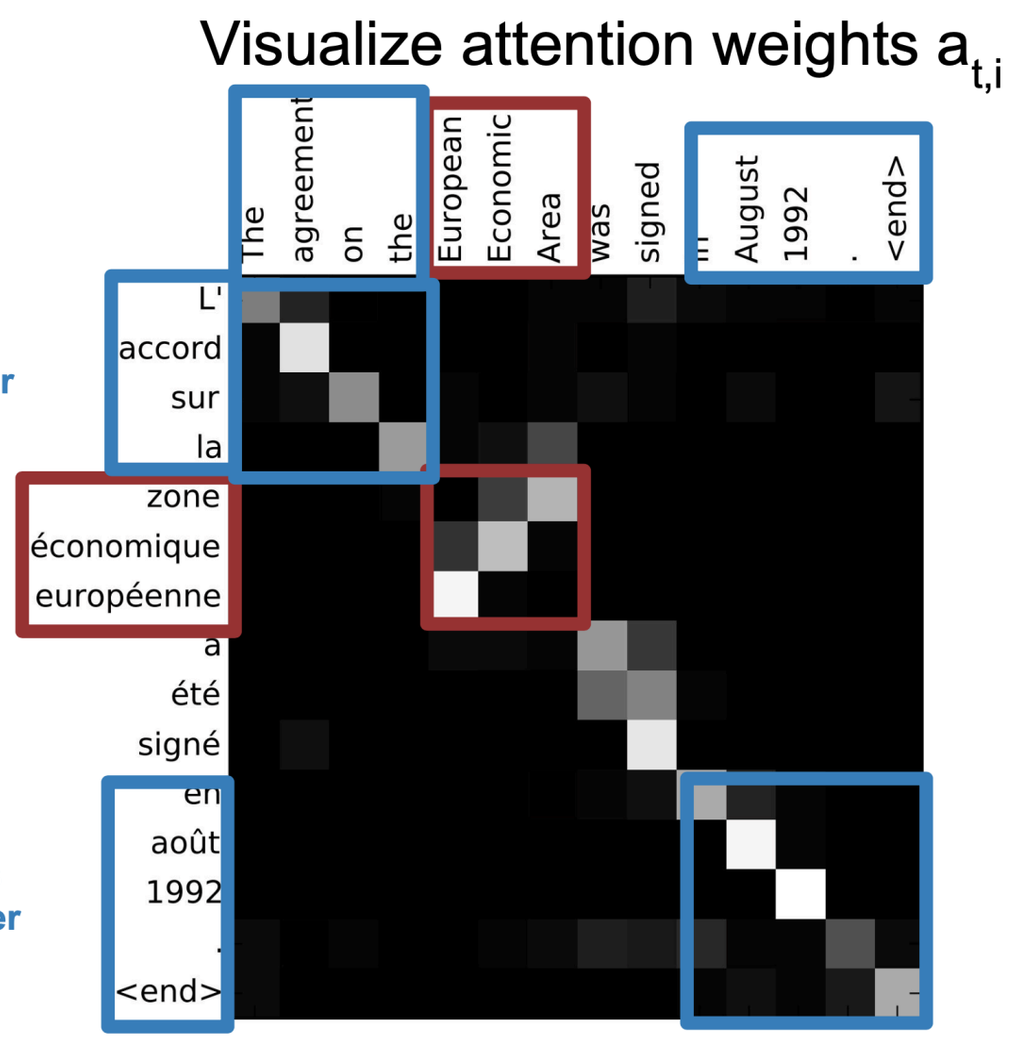
Input: “**The agreement on the European Economic Area** was signed **in August 1992**.”

Output: “**L’accord sur la zone économique européenne** a été signé **en août 1992**.”

Diagonal attention means words correspond in order

Attention figures out different word orders

Diagonal attention means words correspond in order



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

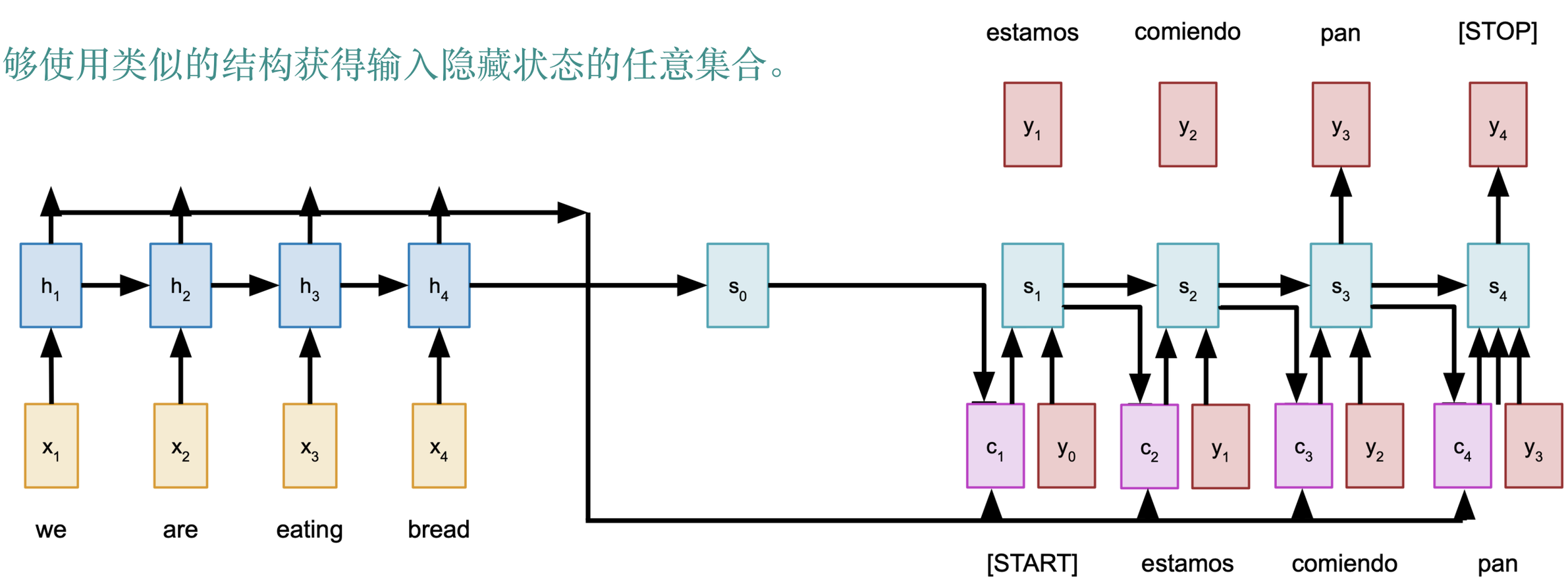
From Fei-Fei Li et al: Attention and Transformers

RNN的注意力机制

基于RNN和注意力机制实现的序列到序列转换（翻译任务）

解码器会忽略h_i是来自一个有序的序列，它直接将其视为无序集合。

能够使用类似的结构获得输入隐藏状态的任意集合。



Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

From Fei-Fei Li et al: Attention and Transformers

例：图像描述

利用空间特征的图像描述

Input: Image **I**
Output: Sequence **y** = $y_1, y_2, ..., y_T$

Encoder: $h_0 = f_{\mathbf{W}}(\mathbf{z})$
where **z** is spatial CNN features
$f_{\mathbf{W}}(.)$ is an MLP

Decoder: $y_t = g_{\mathbf{V}}(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

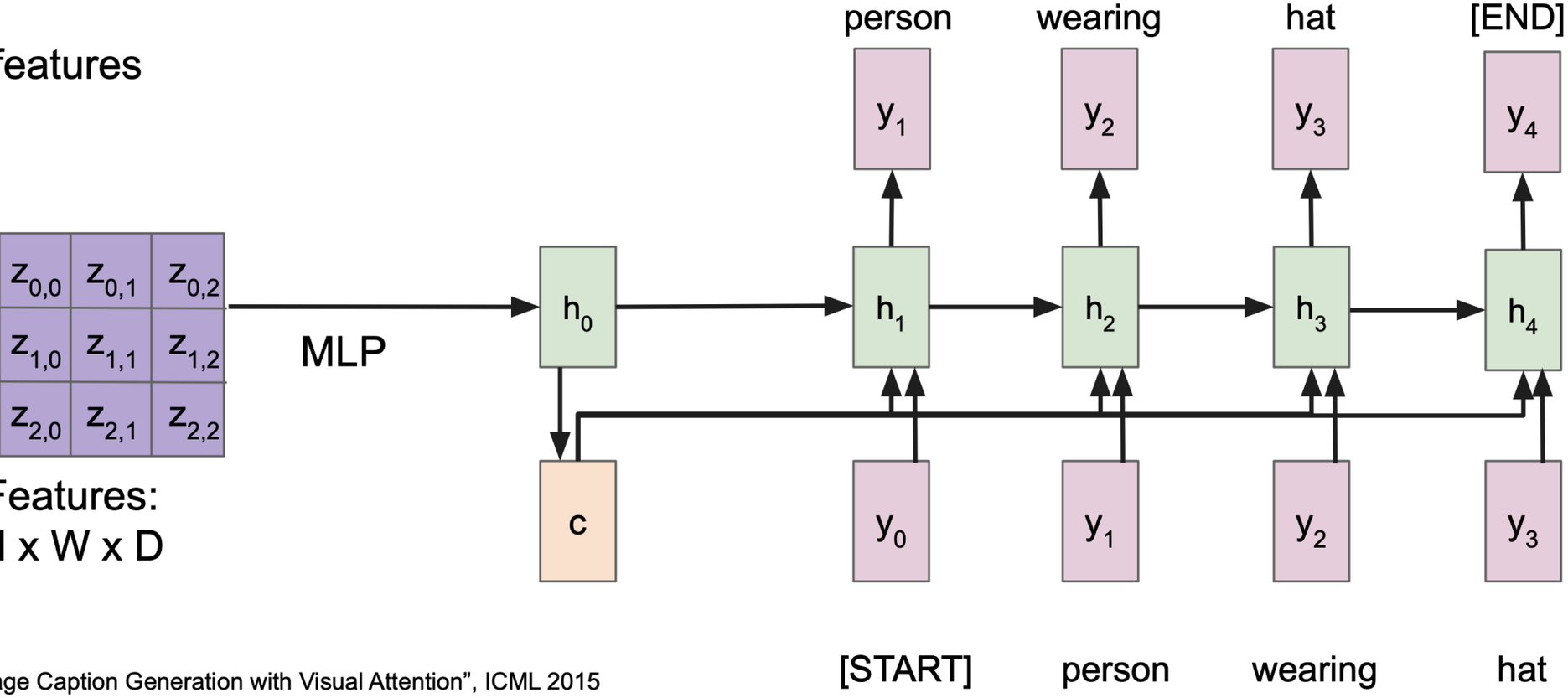
CNN

$z_{0,0}$	$z_{0,1}$	$z_{0,2}$
$z_{1,0}$	$z_{1,1}$	$z_{1,2}$
$z_{2,0}$	$z_{2,1}$	$z_{2,2}$

Features:
H x W x D

MLP

用CNN提取
空间特征



Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

From Fei-Fei Li et al: Attention and Transformers

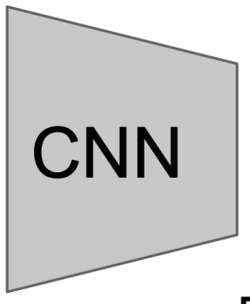
例：图像描述

利用空间特征的图像描述

Problem: Input is "bottlenecked" through c

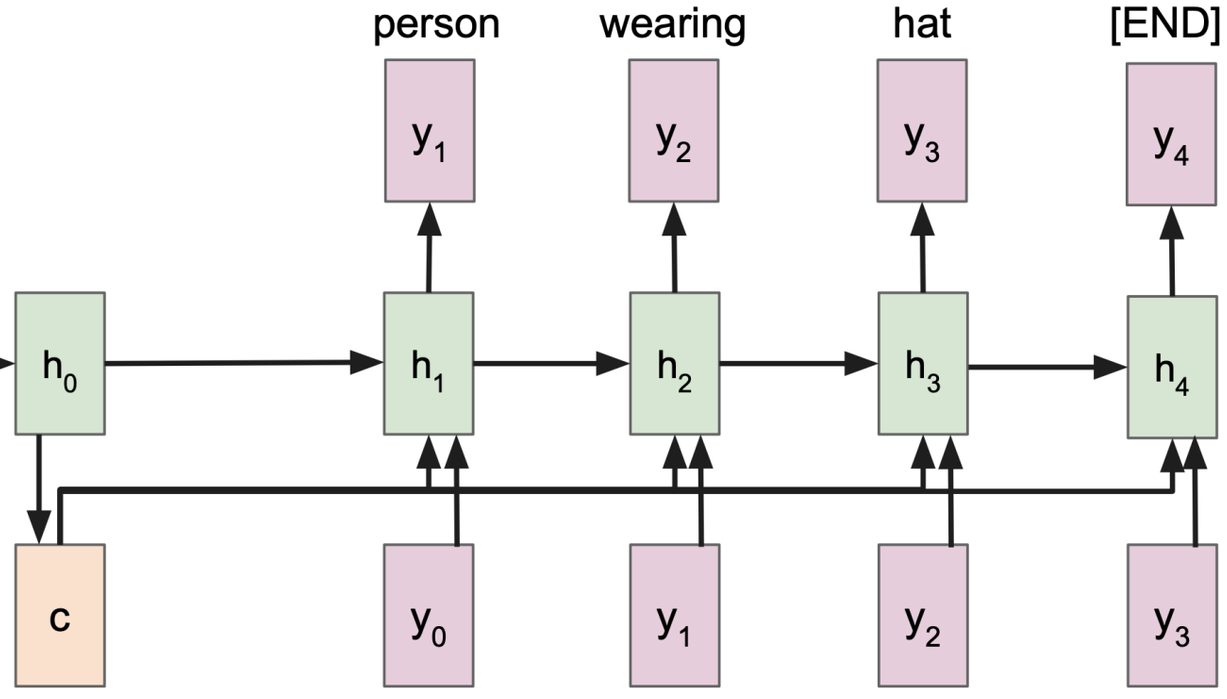
- Model needs to encode everything it wants to say within c

This is a problem if we want to generate really long descriptions? 100s of words long



Features:
H x W x D

MLP



用CNN提取空间特征

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

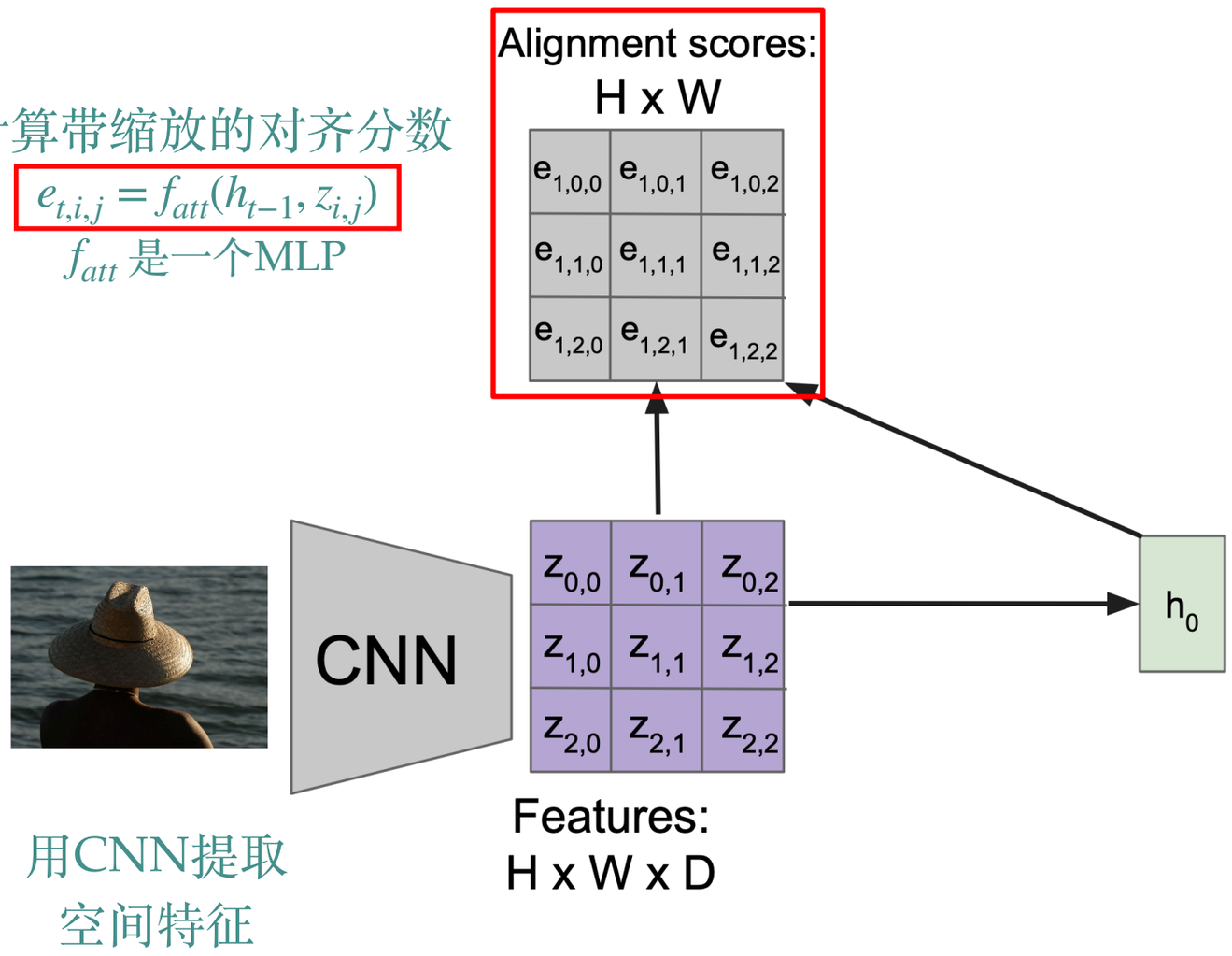
例：图像描述

基于RNN和注意力机制的图像描述

计算带缩放的对齐分数

$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$

f_{att} 是一个MLP



用CNN提取空间特征

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

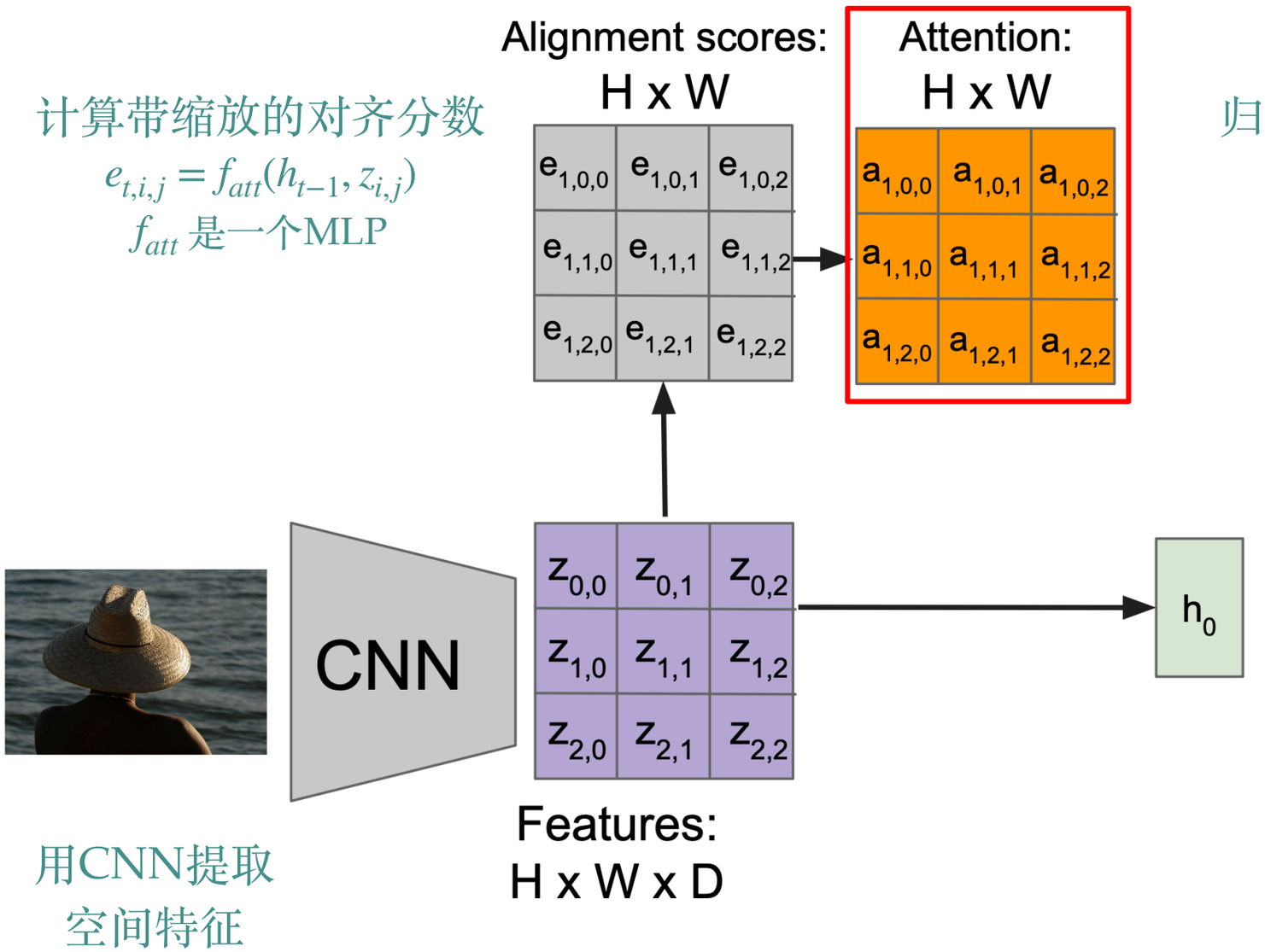
例：图像描述

基于RNN和注意力机制的图像描述

计算带缩放的对齐分数

$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$

f_{att} 是一个MLP



归一化以获得注意力权重

$a_{t,:,:} = softmax(e_{t,:,:})$

$0 < a_{t,i,j} < 1$

注意力的值总和为1

用CNN提取空间特征

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

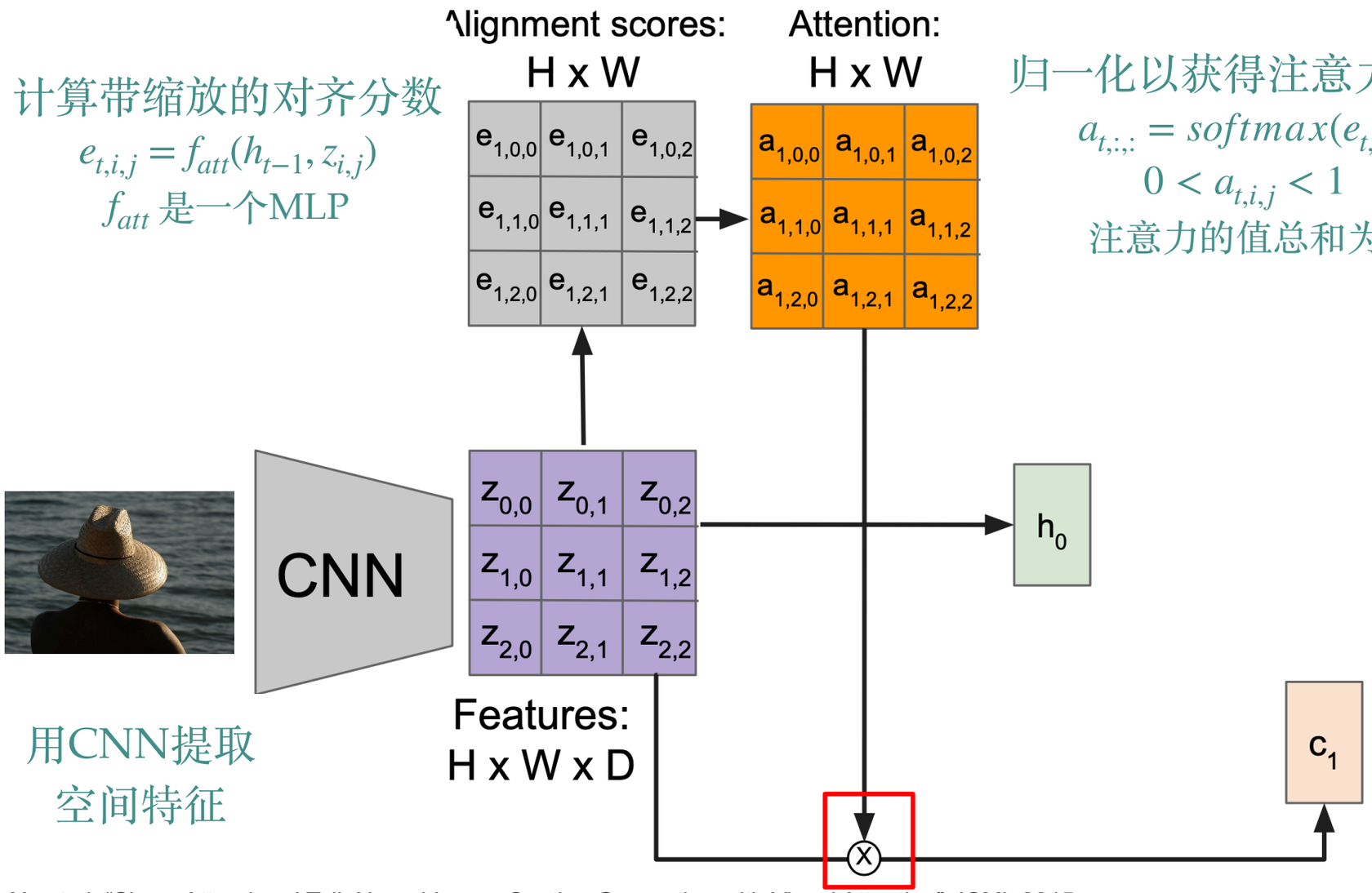
案例：图像描述

基于RNN和注意力机制的图像描述

计算带缩放的对齐分数

$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$

f_{att} 是一个MLP



用CNN提取空间特征

归一化以获得注意力权重

$a_{t,:,:} = softmax(e_{t,:,:})$

$0 < a_{t,i,j} < 1$

注意力的值总和为1

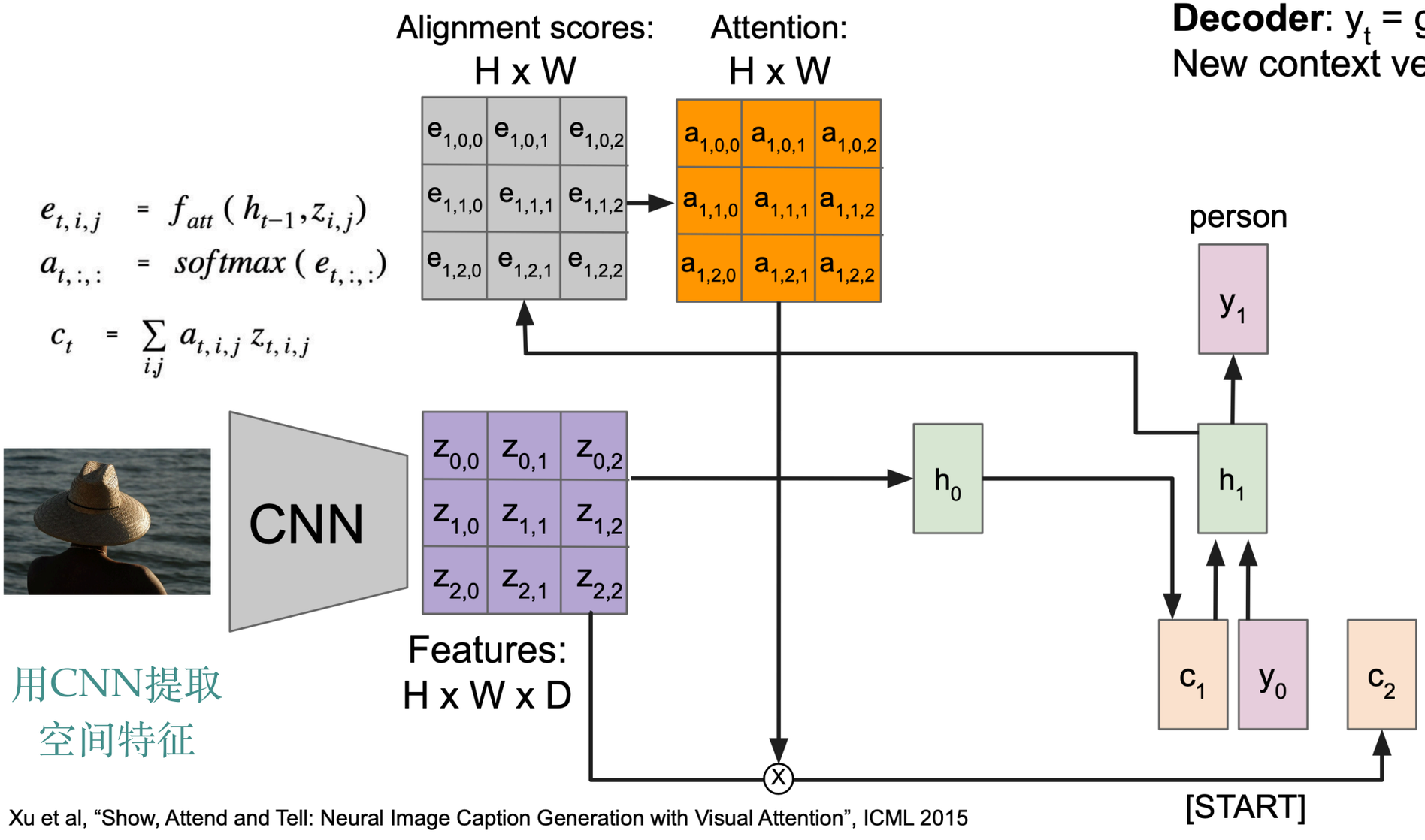
Compute context vector:

$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

例：图像描述

基于RNN和注意力机制的图像描述

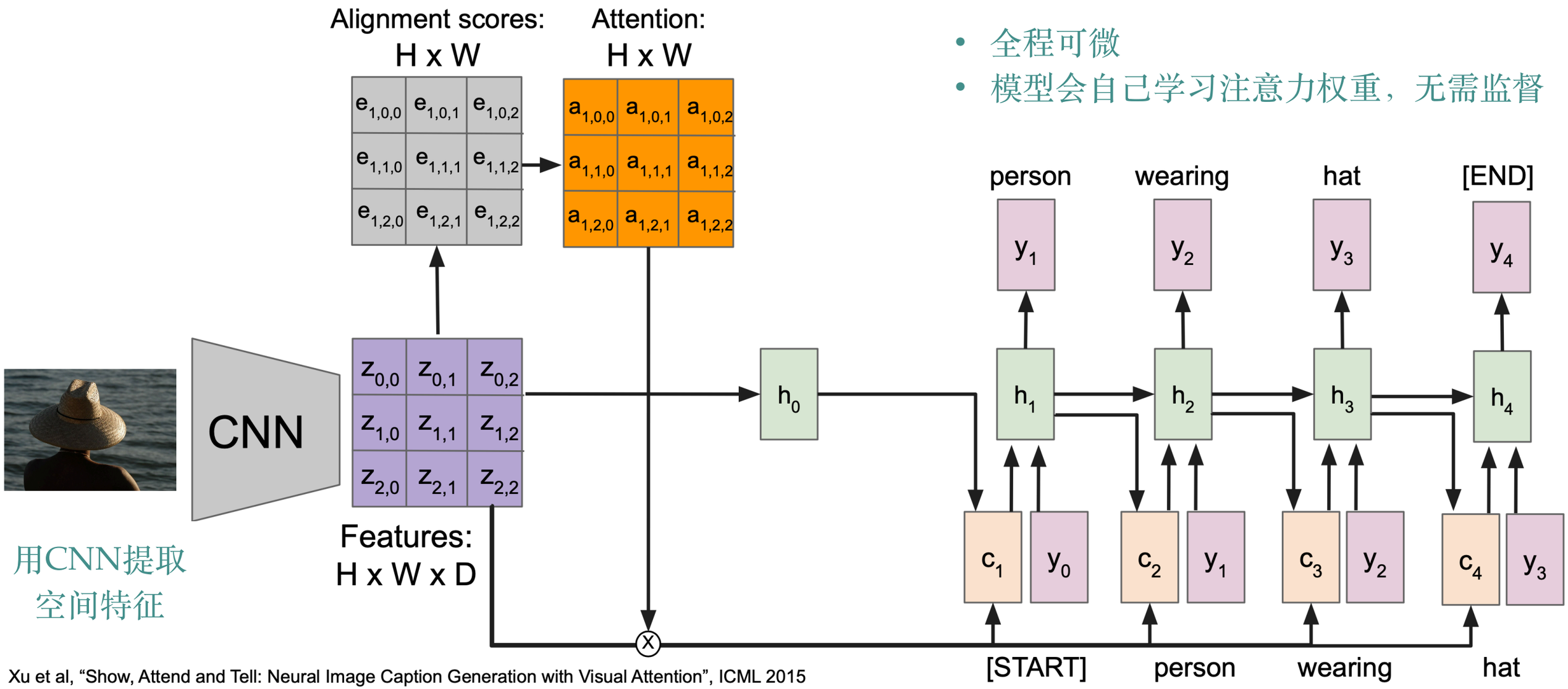


Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

From Fei-Fei Li et al: Attention and Transformers

例：图像描述

基于RNN和注意力机制的图像描述



- 全程可微
- 模型会自己学习注意力权重，无需监督

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

From Fei-Fei Li et al: Attention and Transformers

基于注意力机制的图像描述

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

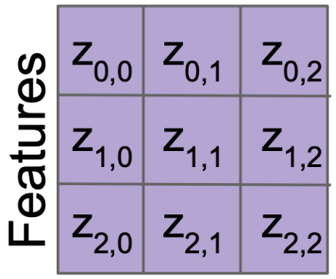
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu et al, “Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

图像描述任务中的注意力机制



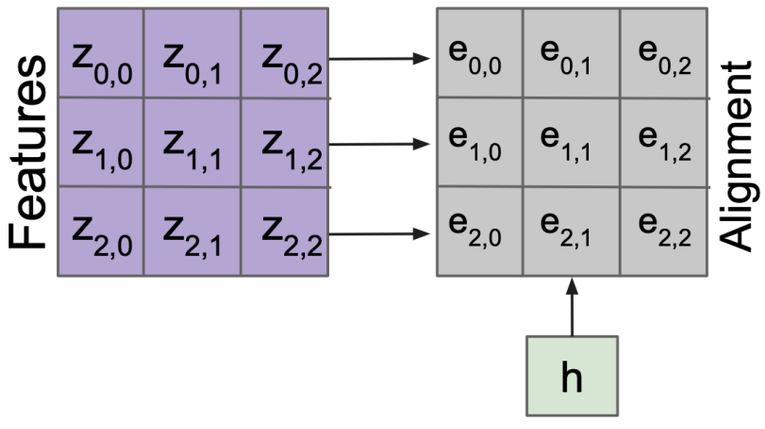
h

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

From Fei-Fei Li et al: Attention and Transformers

图像描述任务中的注意力机制

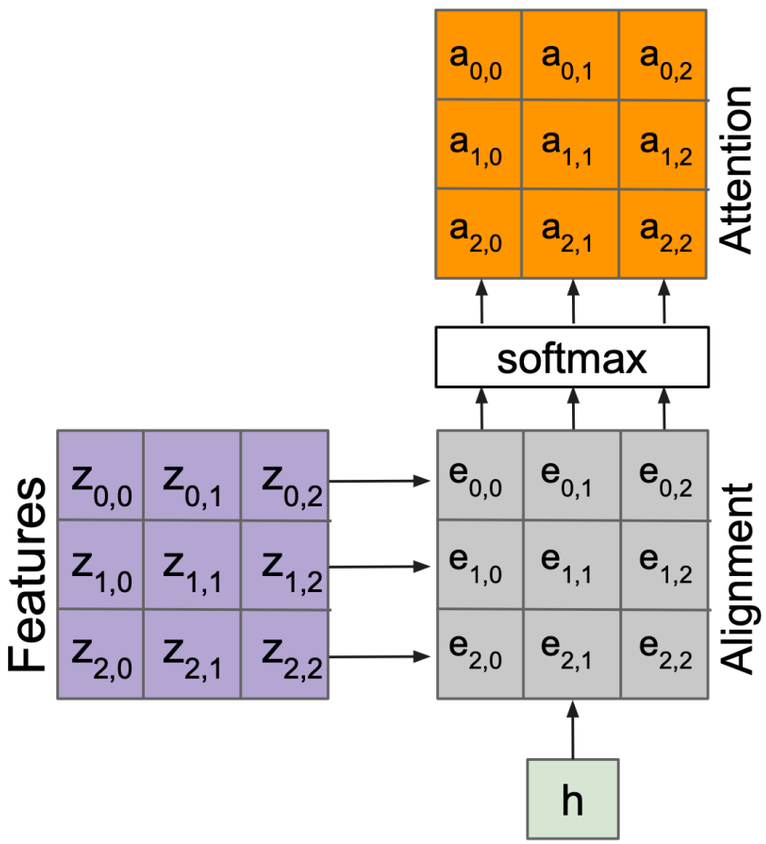
Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$



Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

From Fei-Fei Li et al: Attention and Transformers

图像描述任务中的注意力机制

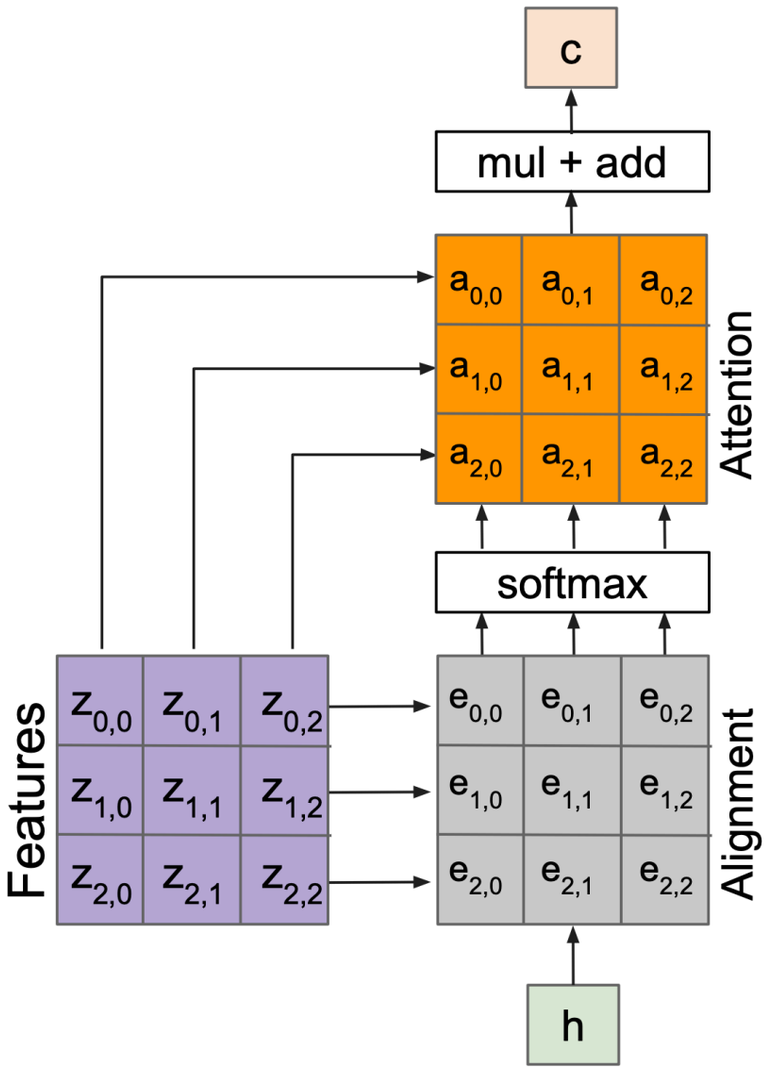


Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

From Fei-Fei Li et al: Attention and Transformers

图像描述任务中的注意力机制



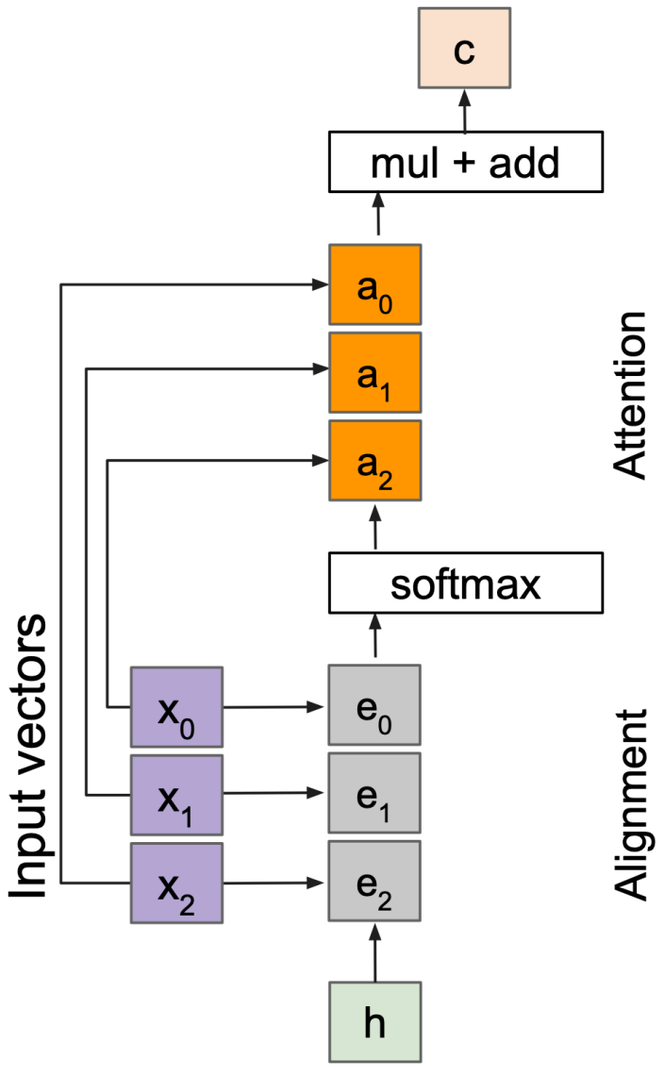
Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
Attention: **a** = softmax(**e**)
Output: $\mathbf{c} = \sum_{i,j} a_{i,j} z_{i,j}$

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

From Fei-Fei Li et al: Attention and Transformers

基础注意力层



Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_i = f_{att}(h, x_i)$
Attention: **a** = softmax(**e**)
Output: $\mathbf{c} = \sum_i a_i x_i$

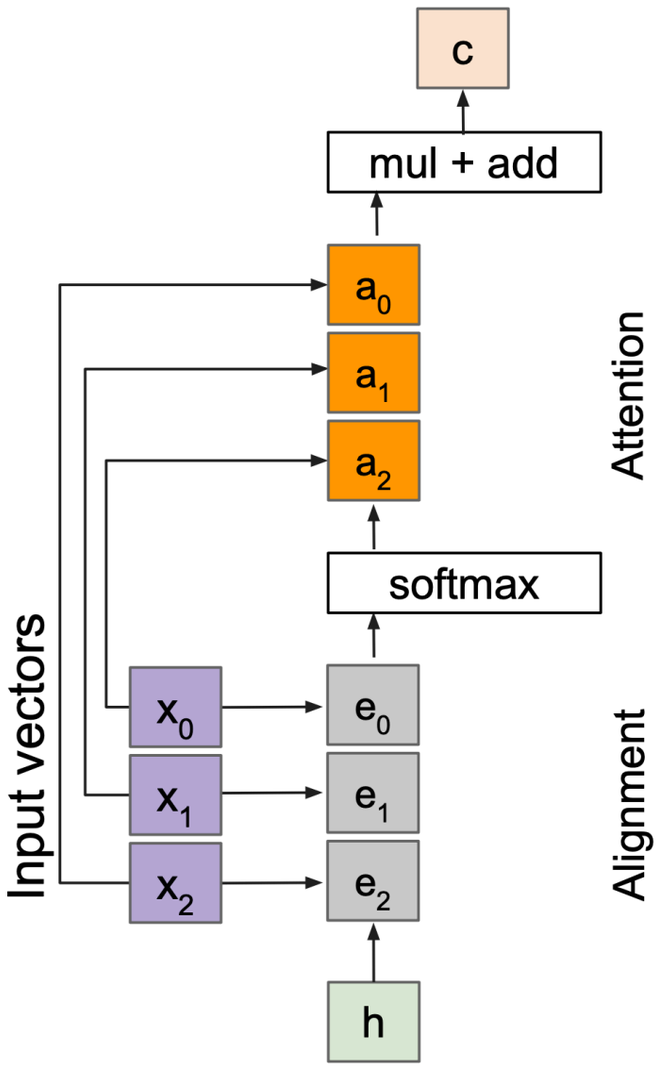
Inputs:
Input vectors: **x** (shape: N x D)
Query: **h** (shape: D)

注意力操作具备排列不变性

- 即与特征的顺序无关
- 直接将H x W(=N)维特征拉平成N维向量处理

From Fei-Fei Li et al: Attention and Transformers

基础注意力层



Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_i = h \cdot x_i$
Attention: **a** = softmax(**e**)
Output: $\mathbf{c} = \sum_i a_i x_i$

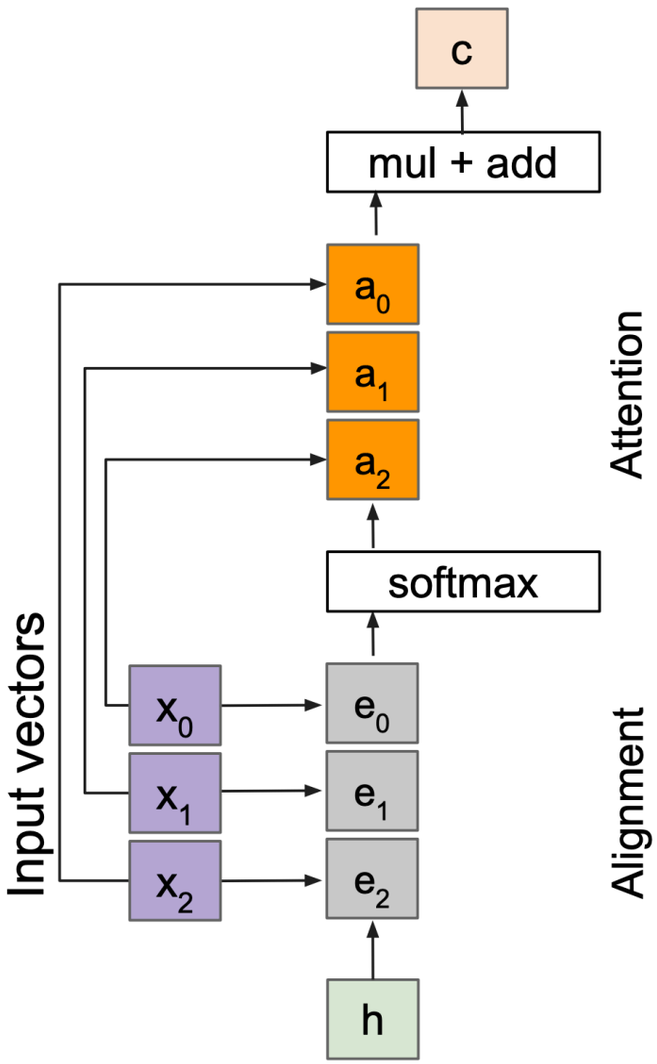
Change $f_{att}(.)$ to a simple dot product

- only works well with key & value transformation trick (will mention in a few slides)

Inputs:
Input vectors: **x** (shape: N x D)
Query: **h** (shape: D)

From Fei-Fei Li et al: Attention and Transformers

基础注意力层



Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_i = h \cdot x_i / \sqrt{D}$
Attention: **a** = softmax(**e**)
Output: $\mathbf{c} = \sum_i a_i x_i$

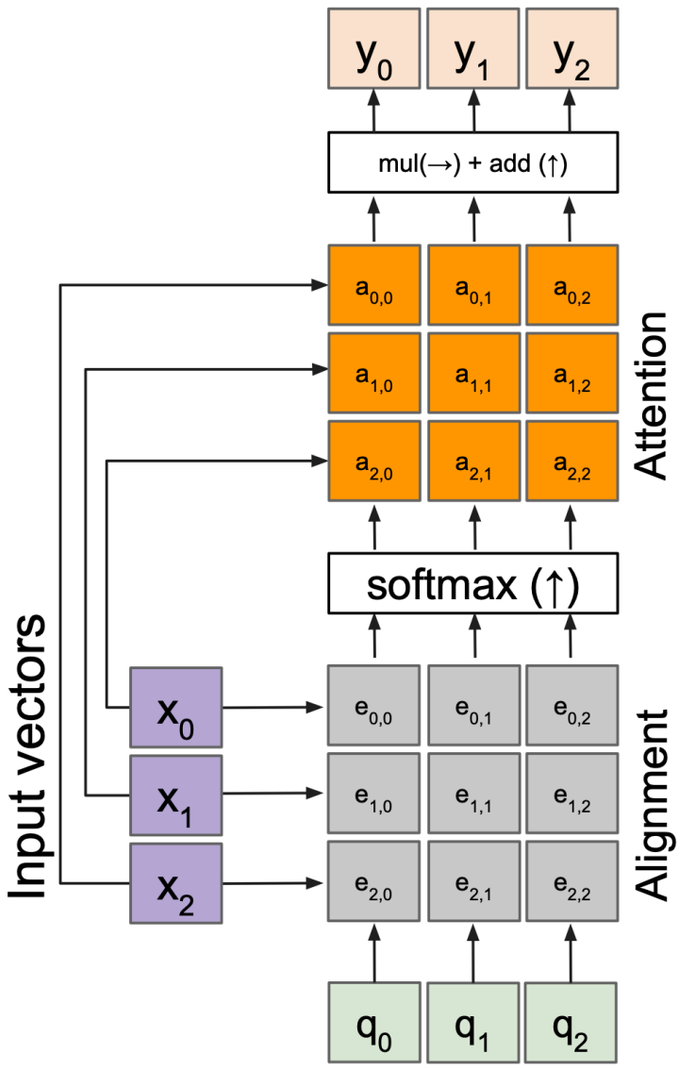
Inputs:
Input vectors: **x** (shape: N x D)
Query: **h** (shape: D)

Change $f_{att}(.)$ to a scaled simple dot product

- Larger dimensions means more terms in the dot product sum.
- So, the variance of the logits is higher. Large magnitude vectors will produce much higher logits.
- So, the post-softmax distribution has lower-entropy, assuming logits are IID.
- Ultimately, these large magnitude vectors will cause softmax to peak and assign very little weight to all others
- Divide by $\sqrt{D}$ to reduce effect of large magnitude vectors

From Fei-Fei Li et al: Attention and Transformers

基础注意力层



Outputs:
context vectors: **y** (shape: D)

Multiple query vectors
- each query creates a new output context vector

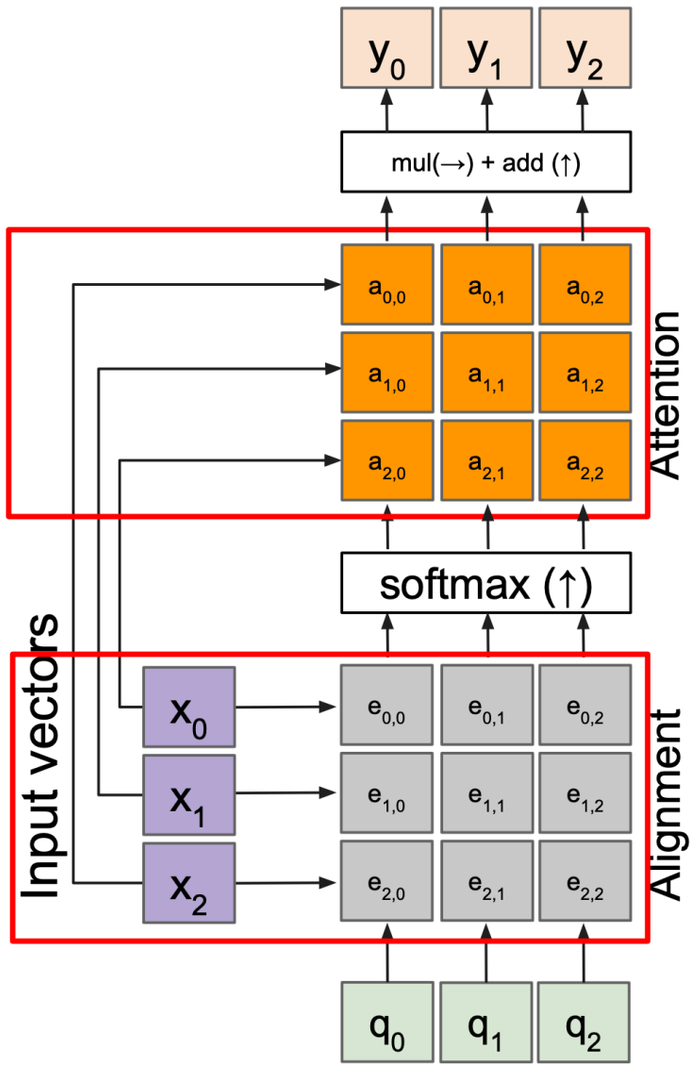
Operations:
Alignment: $e_{i,j} = q_j \cdot x_i / \sqrt{D}$
Attention: **a** = softmax(**e**)
Output: $y_j = \sum_i a_{i,j} x_i$

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D)

Multiple query vectors

From Fei-Fei Li et al: Attention and Transformers

基础注意力层



Outputs:
context vectors: **y** (shape: D)

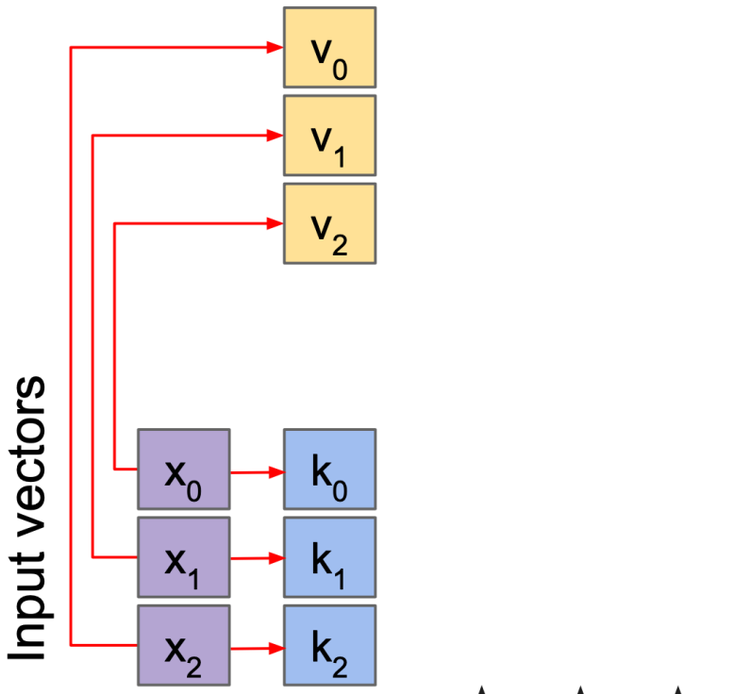
Operations:
Alignment: $e_{i,j} = q_j \cdot x_i / \sqrt{D}$
Attention: **a** = softmax(**e**)
Output: $y_j = \sum_i a_{i,j} x_i$

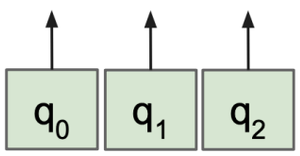
Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D)

Notice that the input vectors are used for both the alignment as well as the attention calculations.

- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

基础注意力层





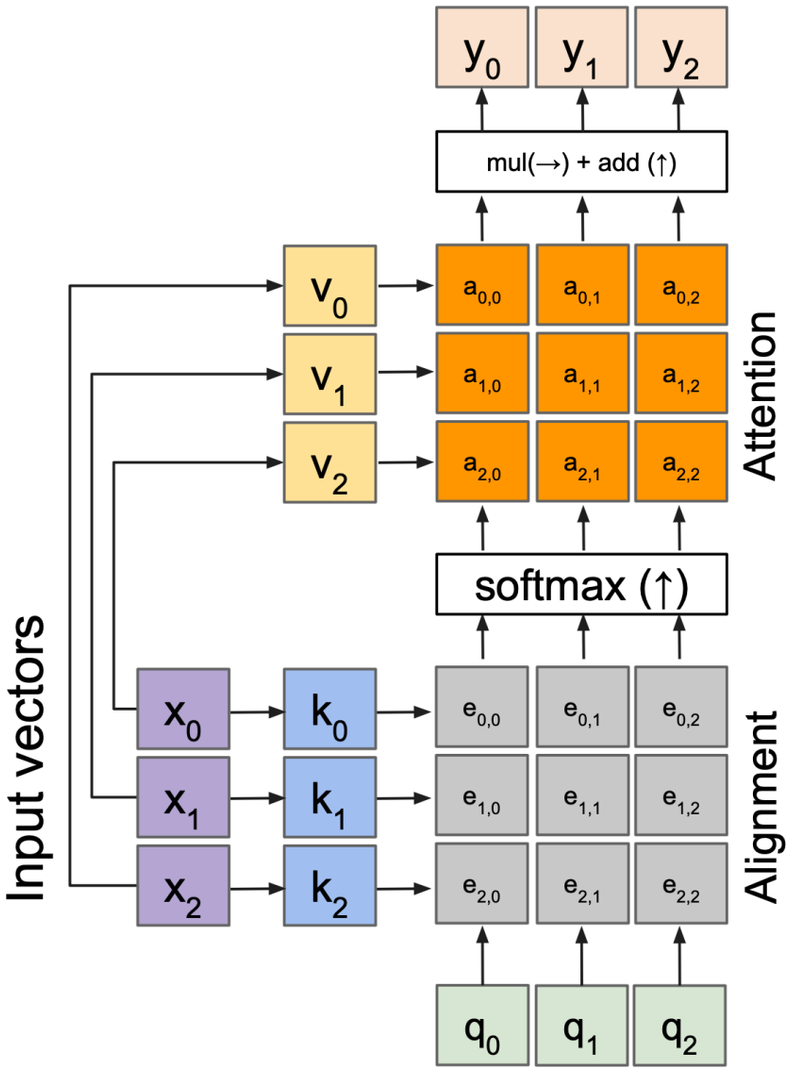
Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

Notice that the input vectors are used for both the alignment as well as the attention calculations.

- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

基础注意力层



Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $a = \text{softmax}(e)$
Output: $y_j = \sum_i a_{i,j} v_i$

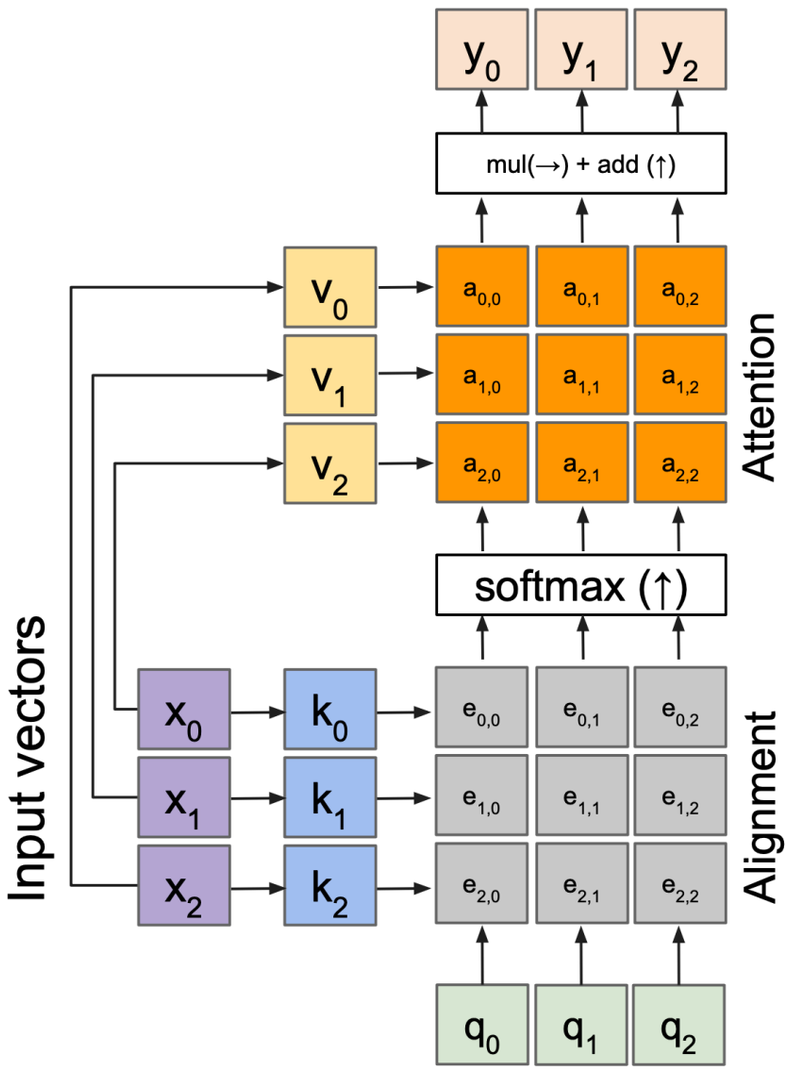
Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

The input and output dimensions can now change depending on the key and value FC layers

Notice that the input vectors are used for both the alignment as well as the attention calculations.

- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

基础注意力层



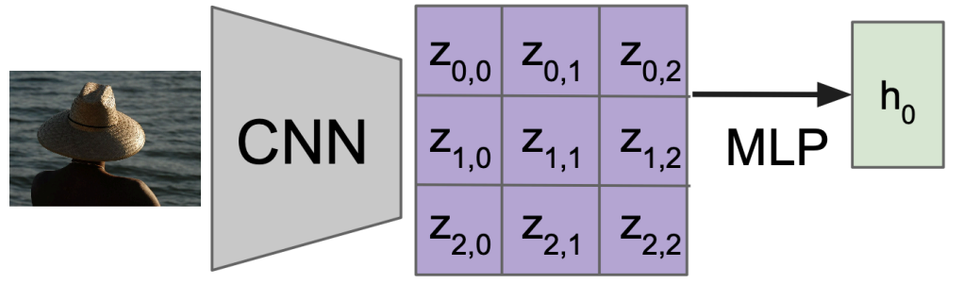
Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W}_k$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W}_v$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)
Queries: **q** (shape: M x D_k)

Recall that the query vector was a function of the input vectors

Encoder: $h_0 = f_{\mathbf{W}}(\mathbf{z})$
where **z** is spatial CNN features
$f_{\mathbf{W}}(.)$ is an MLP



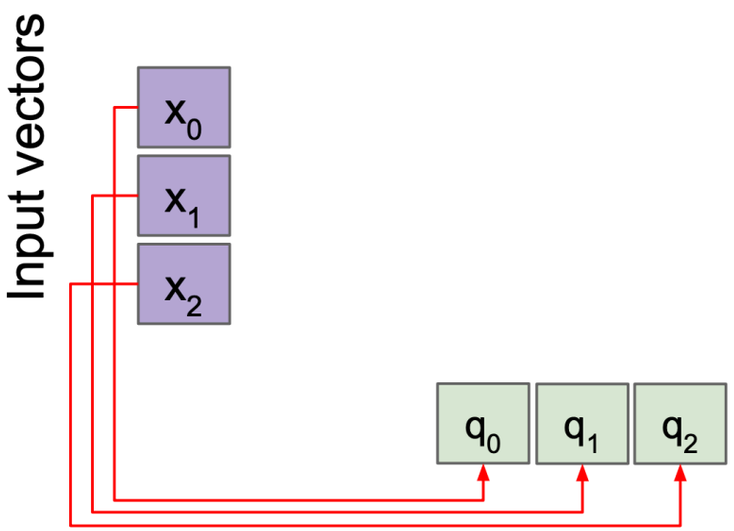
From Fei-Fei Li et al: Attention and Transformers

自注意力层

We can calculate the query vectors from the input vectors, therefore, defining a "self-attention" layer.

Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$
Query vectors: $q = xW_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $a = \text{softmax}(e)$
Output: $y_j = \sum_i a_{i,j} v_i$

Instead, query vectors are calculated using a FC layer.

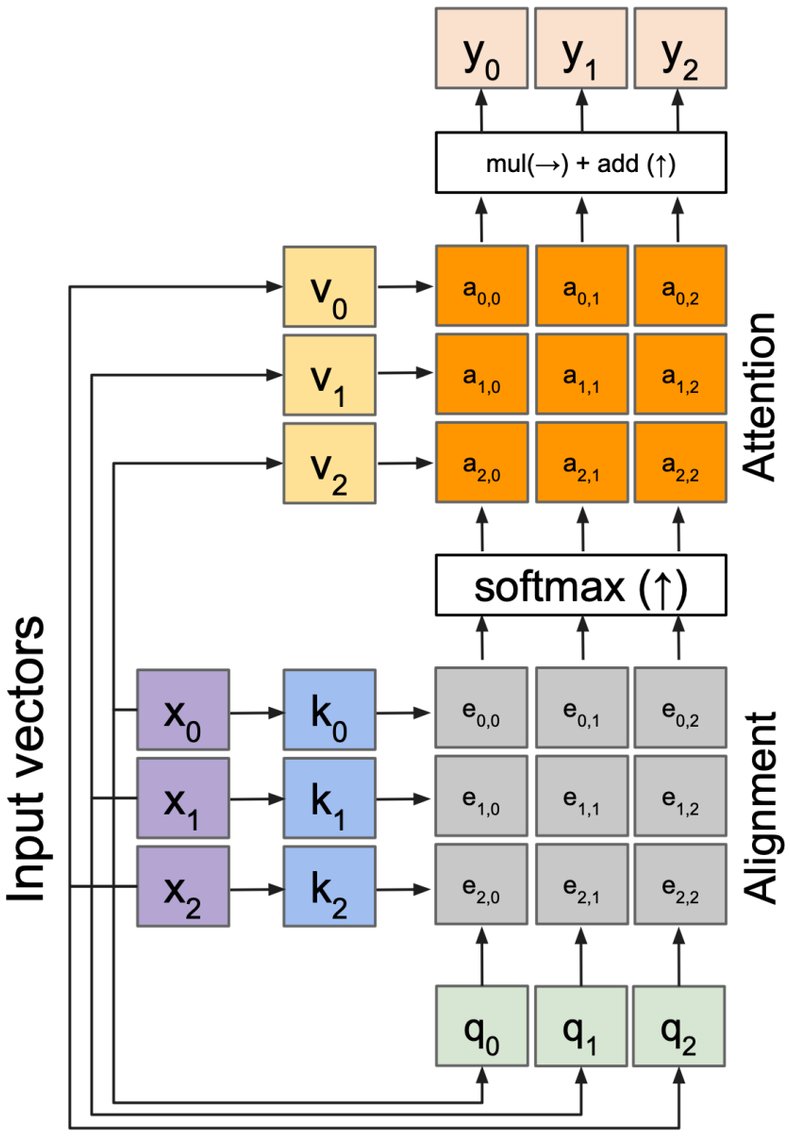


No input query vectors anymore

Inputs:
Input vectors: **x** (shape: N x D)
~~Queries: **q** (shape: M x D_k)~~

From Fei-Fei Li et al: Attention and Transformers

自注意力层



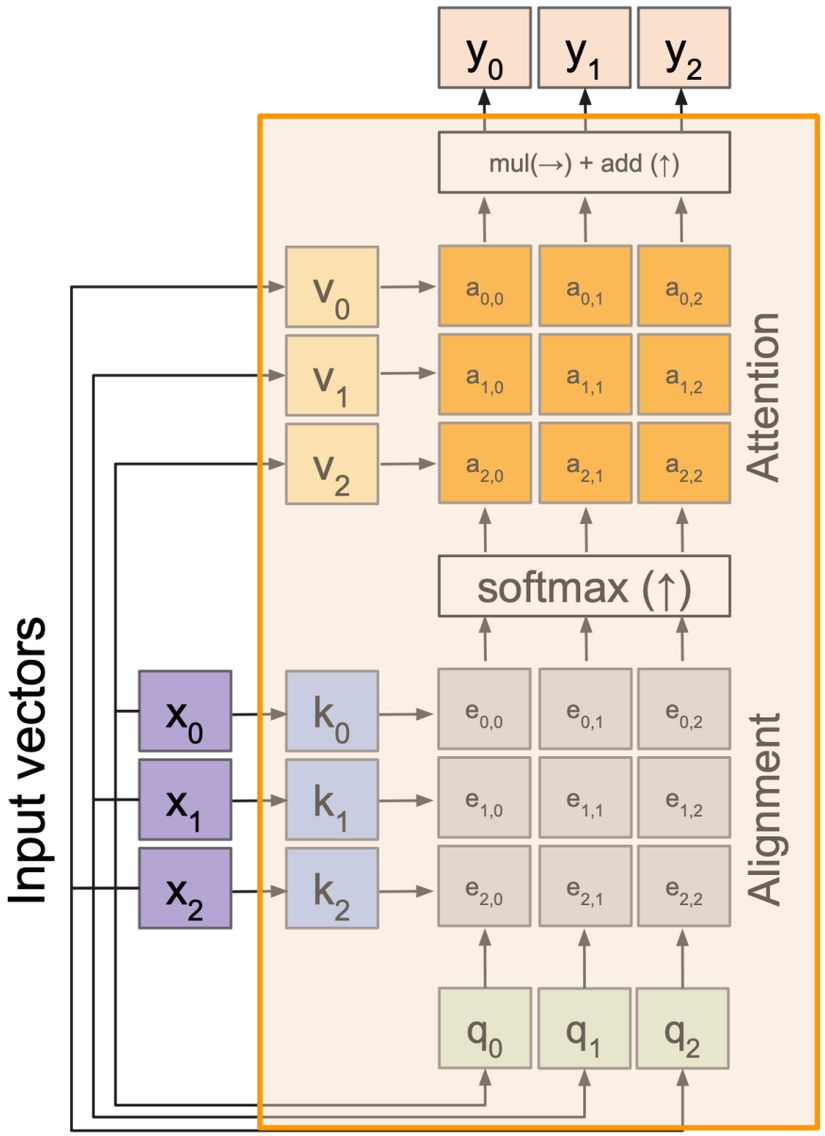
Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)

From Fei-Fei Li et al: Attention and Transformers

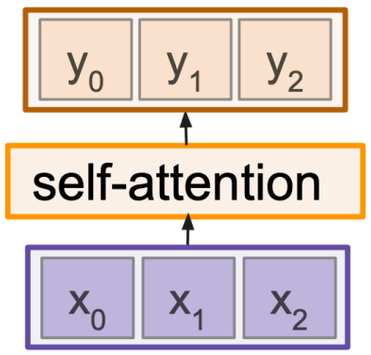
自注意力层：在多个输入上并行注意力计算



Outputs:
context vectors: **y** (shape: D_v)

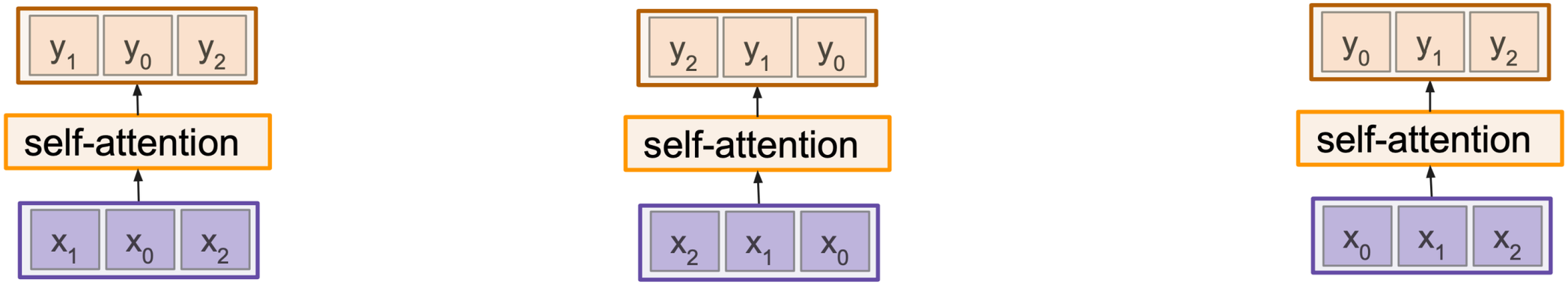
Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)



From Fei-Fei Li et al: Attention and Transformers

自注意力层：在多个输入上并行注意力计算

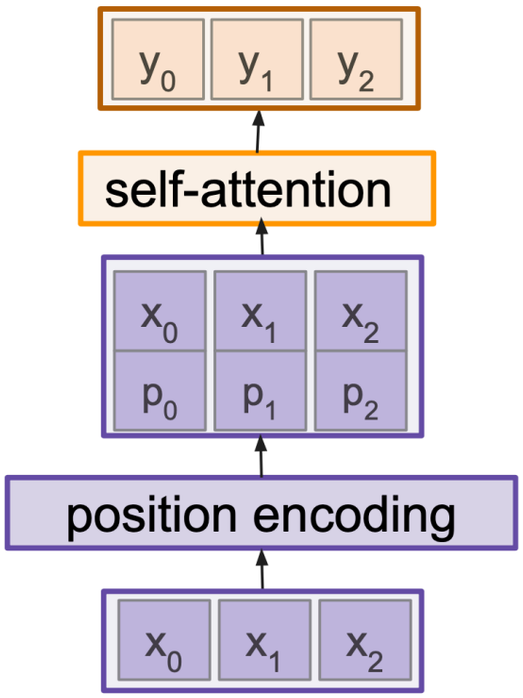


Permutation equivariant

Self-attention layer doesn’t care about the orders of the inputs!

Problem: How can we encode ordered sequences like language or spatially ordered image features?

位置编码



Concatenate/add special positional encoding p_j to each input vector x_j

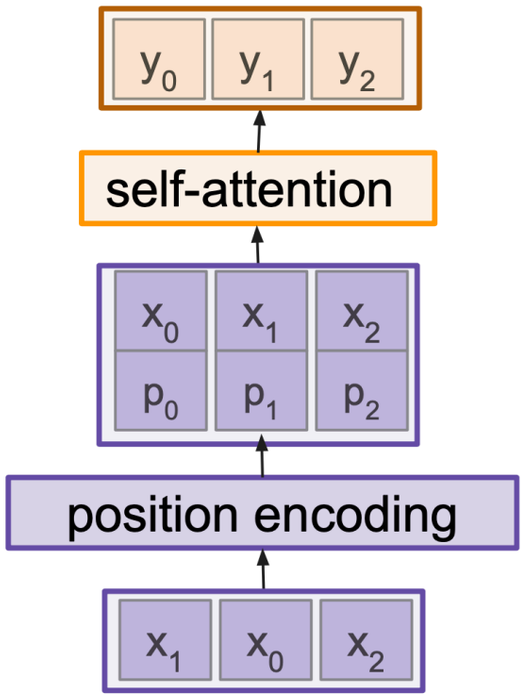
We use a function *pos*: N $\rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

So, p_j = *pos*(j)

Desiderata of *pos*(.) :

1. It should output a **unique** encoding for each time-step (word's position in a sentence)
2. **Distance** between any two time-steps should be consistent across sentences with different lengths.
3. Our model should generalize to **longer** sentences without any efforts. Its values should be bounded.
4. It must be **deterministic**.

位置编码



Concatenate special positional encoding p_j to each input vector x_j

We use a function *pos*: N $\rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

So, p_j = *pos*(j)

Options for *pos*(.)

1. Learn a lookup table:
   - Learn parameters to use for *pos*(t) for t ε [0, T)
   - Lookup table contains T x d parameters.

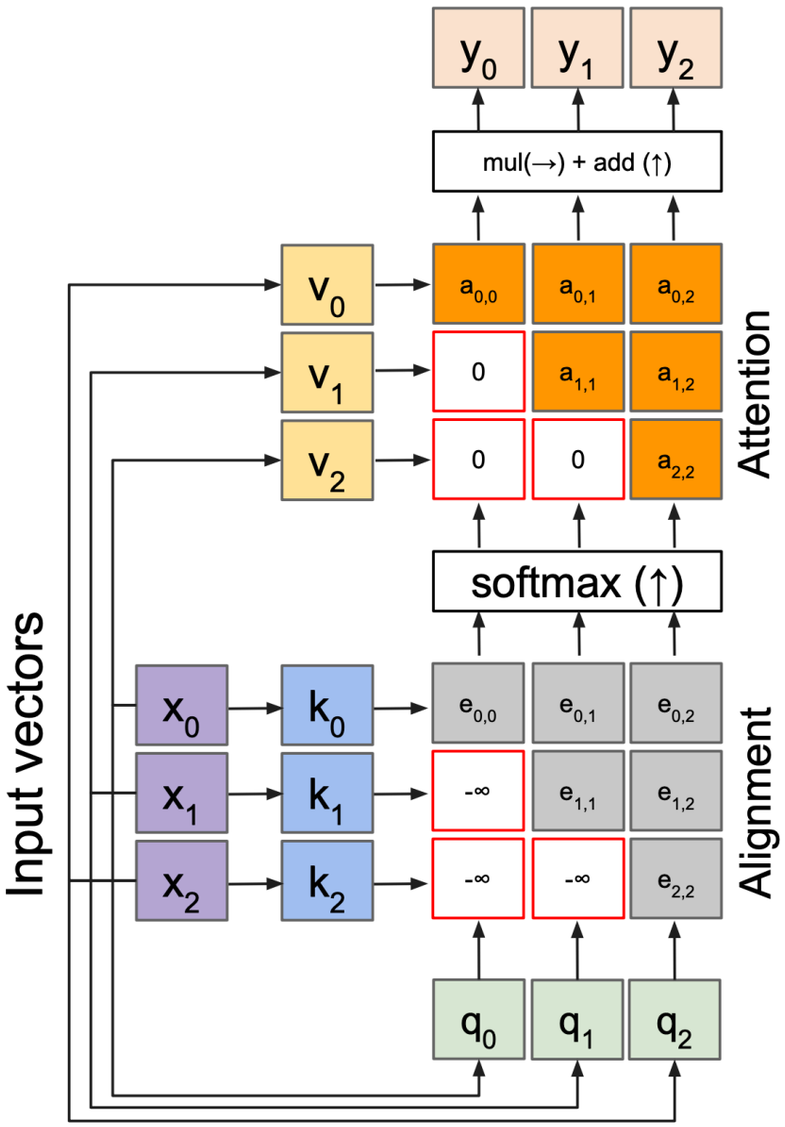
2. Design a fixed function with the desiderata

$$\mathbf{p(t)} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_d \quad \text{where } \omega_k = \frac{1}{10000^{2k/d}}$$

Vaswani et al, “Attention is all you need”, NeurIPS 2017

From Fei-Fei Li et al: Attention and Transformers

带掩膜的自注意力层



Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

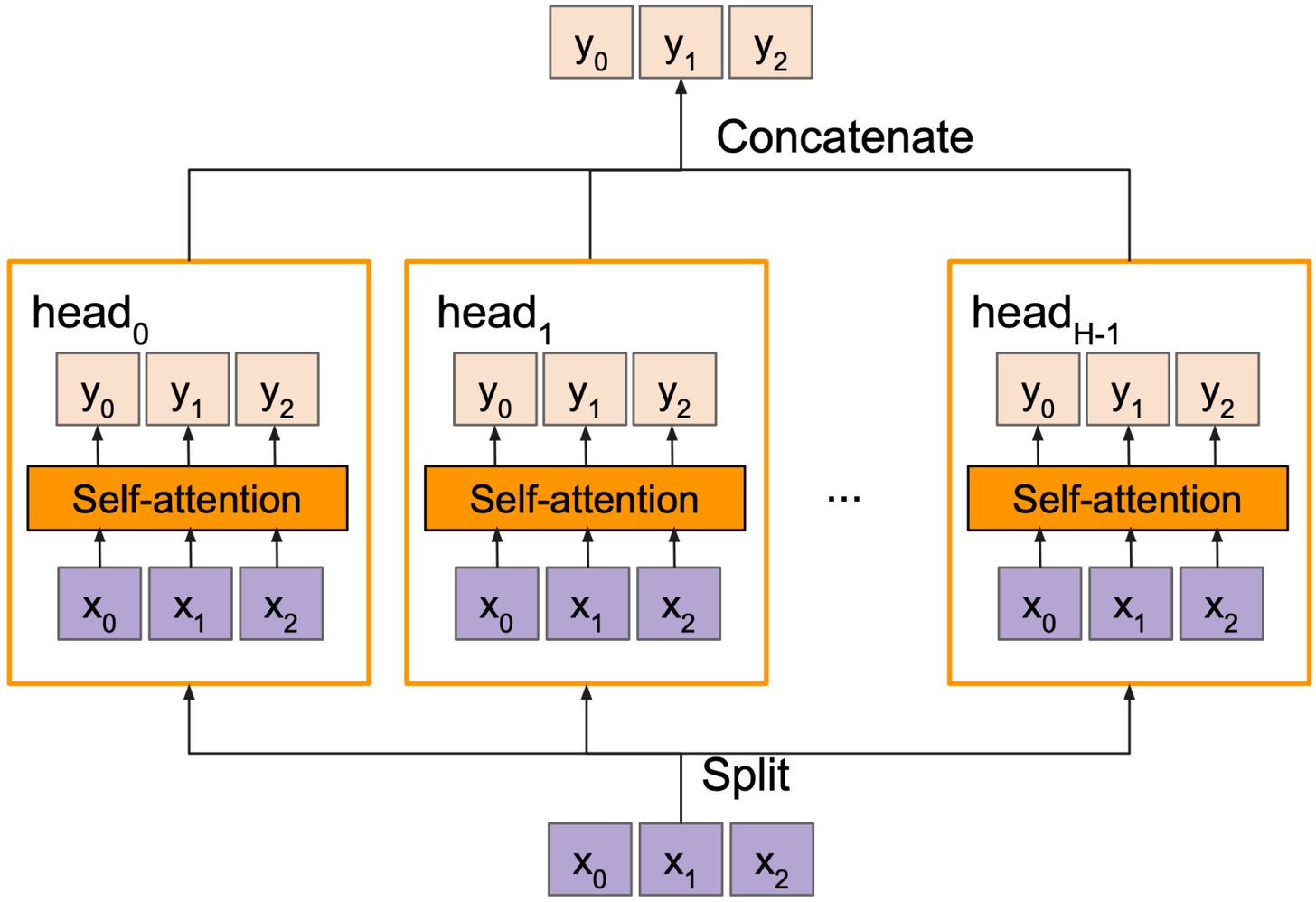
Inputs:
Input vectors: **x** (shape: N x D)

- Prevent vectors from looking at future vectors.
- Manually set alignment scores to -infinity

From Fei-Fei Li et al: Attention and Transformers

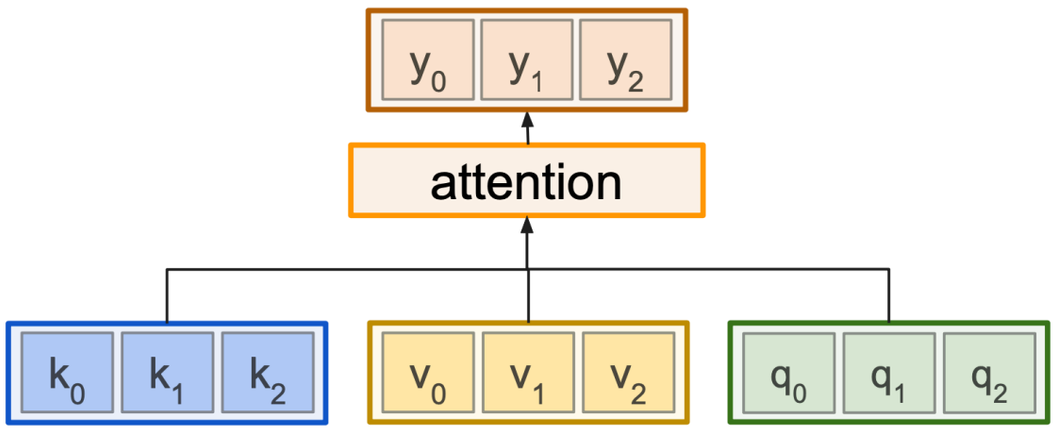
多头自注意力层

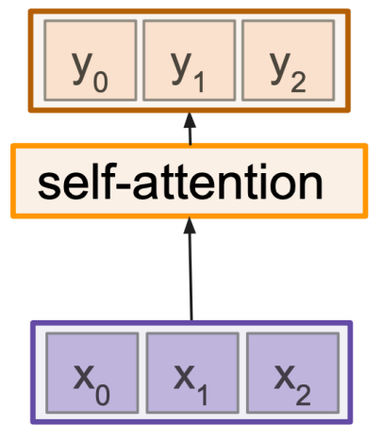
- Multiple self-attention heads in parallel



From Fei-Fei Li et al: Attention and Transformers

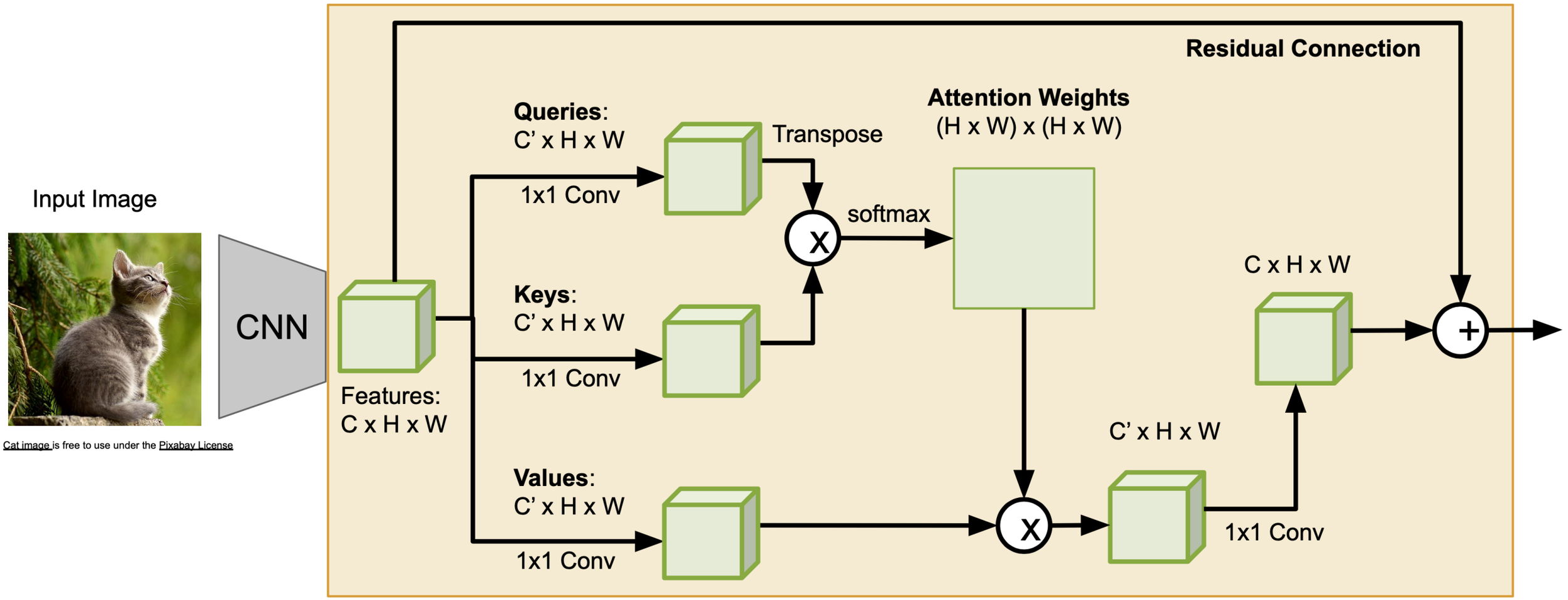
基础注意力与自注意力的对比





From Fei-Fei Li et al: Attention and Transformers

例：带自注意力机制的卷积神经网络



Cat image is free to use under the Pixabay License

Self-Attention Module

Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

Slide credit: Justin Johnson

RNN 与 Transformer 的对比

RNNs
(+) LSTMs work reasonably well for long sequences.
(-) Expects an ordered sequences of inputs
(-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.

Transformer:
(+) Good at long sequences. Each attention calculation looks at all inputs.
(+) Can operate over unordered sets or ordered sequences with positional encodings.
(+) Parallel computation: All alignment and attention scores for all inputs can be done in parallel.
(-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head. (but GPUs are getting bigger and better)

From Fei-Fei Li et al: Attention and Transformers

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

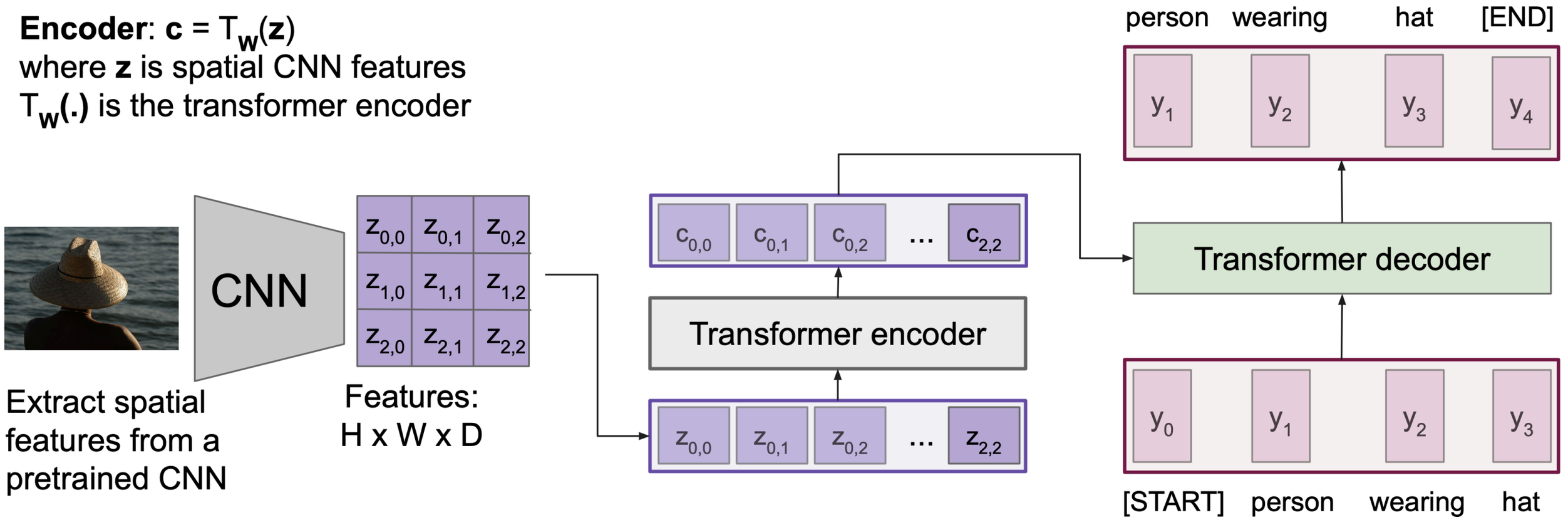
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

基于Transformer的图像描述

Input: Image **I**
Output: Sequence **y** = $y_1, y_2, ..., y_T$

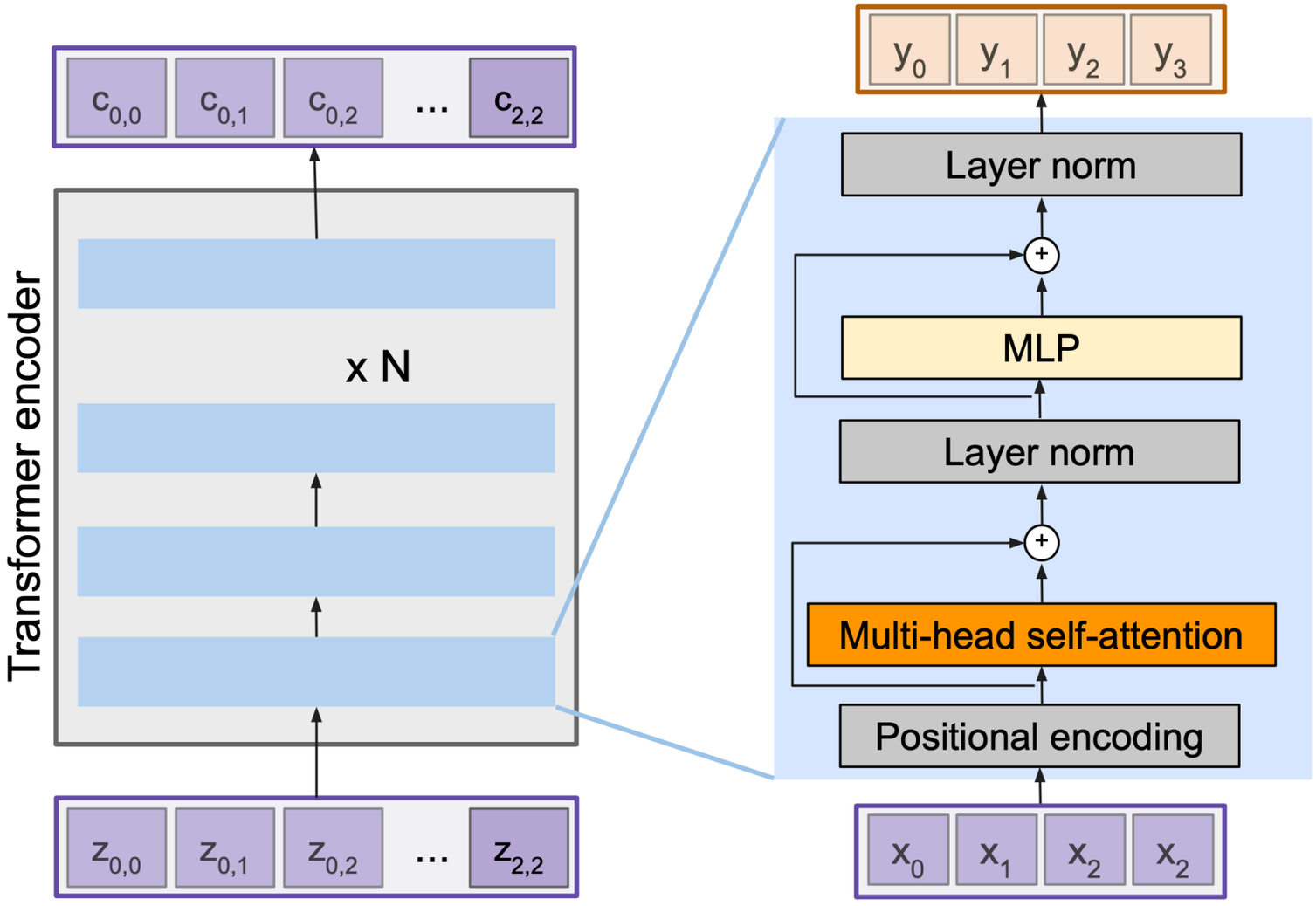
Encoder: **c** = $T_W(\mathbf{z})$
where **z** is spatial CNN features
$T_W(.)$ is the transformer encoder

Decoder: $y_t = T_D(\mathbf{y}_{0:t-1}, \mathbf{c})$
where $T_D(.)$ is the transformer decoder



From Fei-Fei Li et al: Attention and Transformers

Transformer编码器模块



Transformer Encoder Block:

Inputs: Set of vectors **x**
Outputs: Set of vectors **y**

Self-attention is the only interaction between vectors.

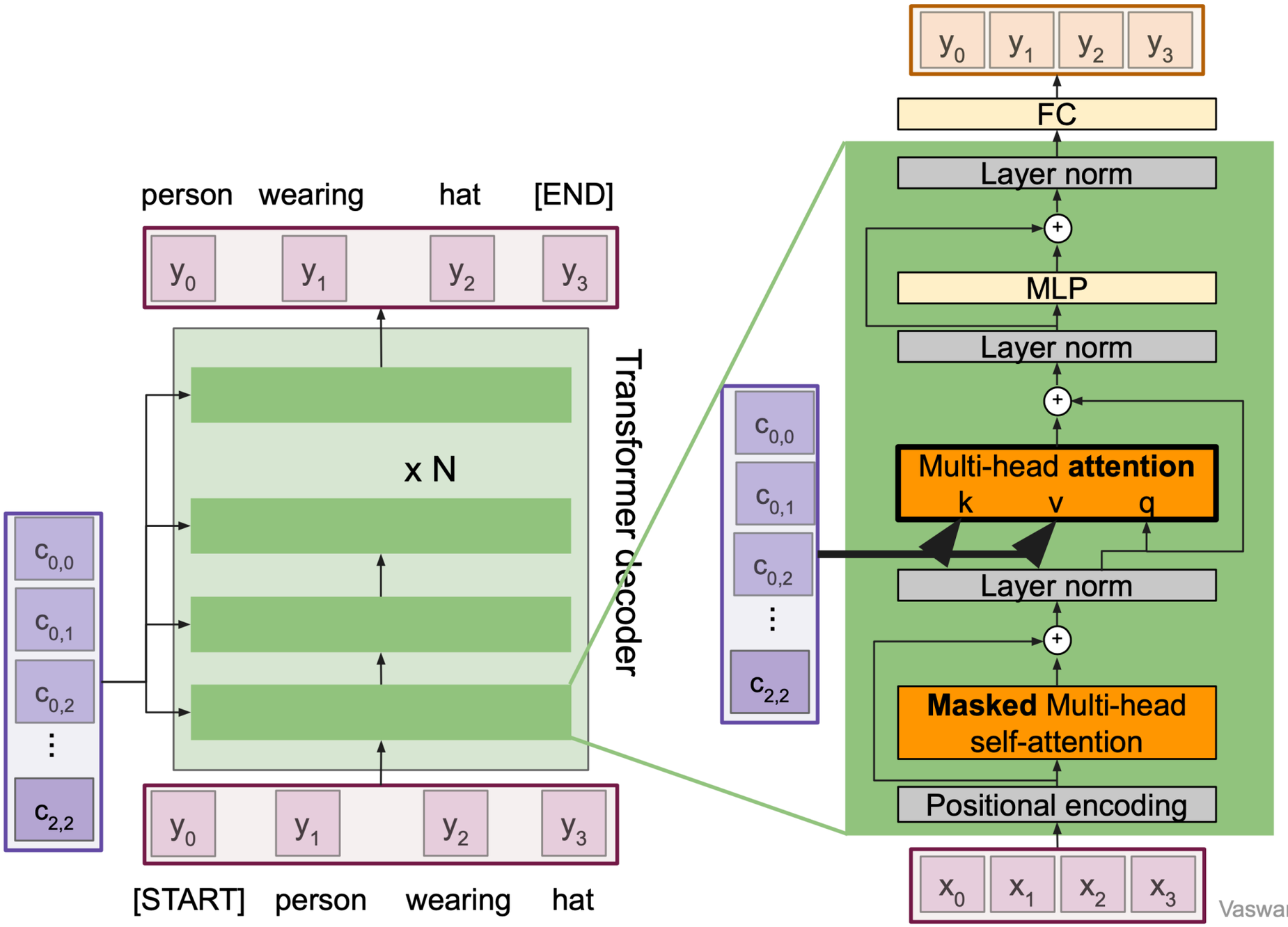
Layer norm and MLP operate independently per vector.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, “Attention is all you need”, NeurIPS 2017

From Fei-Fei Li et al: Attention and Transformers

Transformer解码器模块

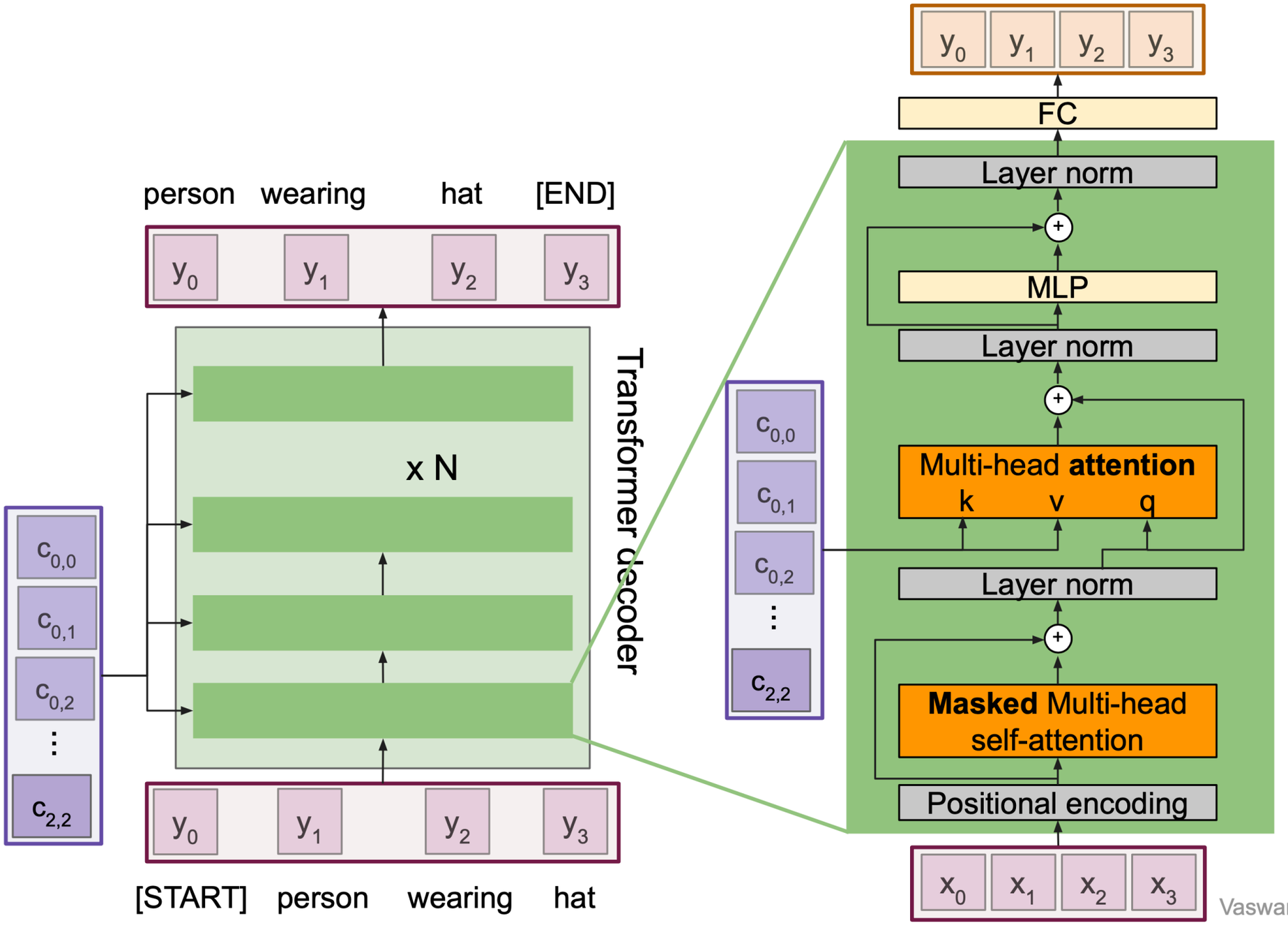


Multi-head attention block attends over the transformer encoder outputs.

For image captions, this is how we inject image features into the decoder.

Vaswani et al, “Attention is all you need”, NeurIPS 2017

Transformer解码器模块



Transformer Decoder Block:

Inputs: Set of vectors **x** and Set of context vectors **c.**
Outputs: Set of vectors **y.**

Masked Self-attention only interacts with past inputs.

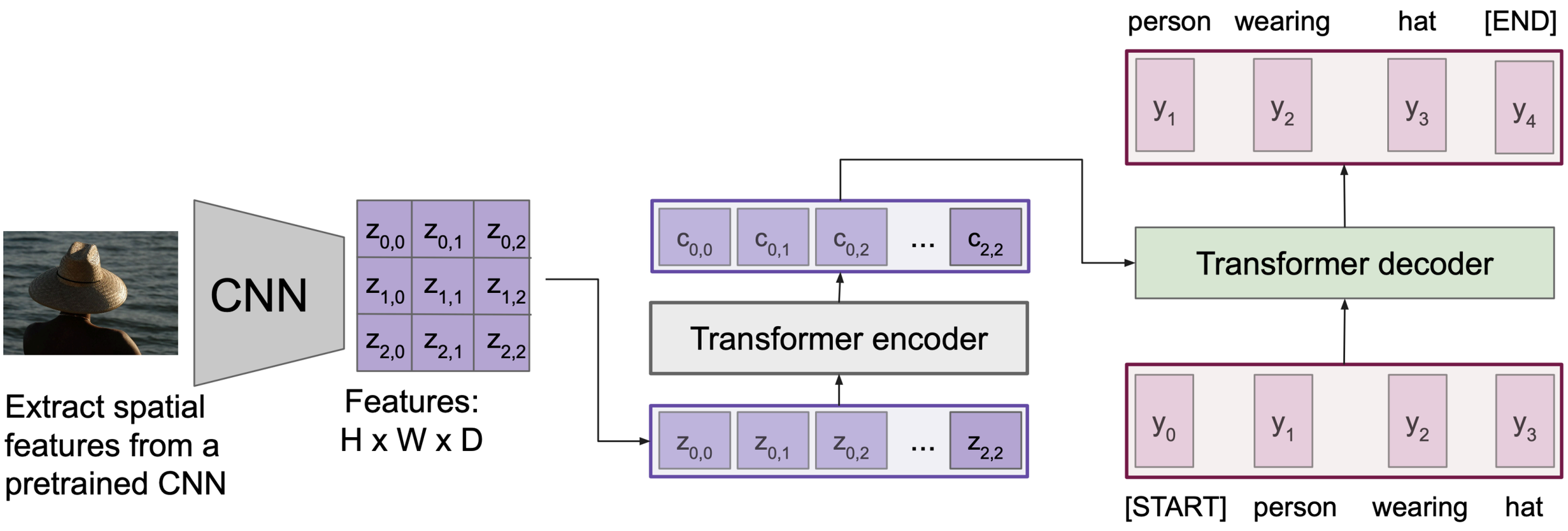
Multi-head attention block is NOT self-attention. It attends over encoder outputs.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, “Attention is all you need”, NeurIPS 2017

基于Transformer的图像描述

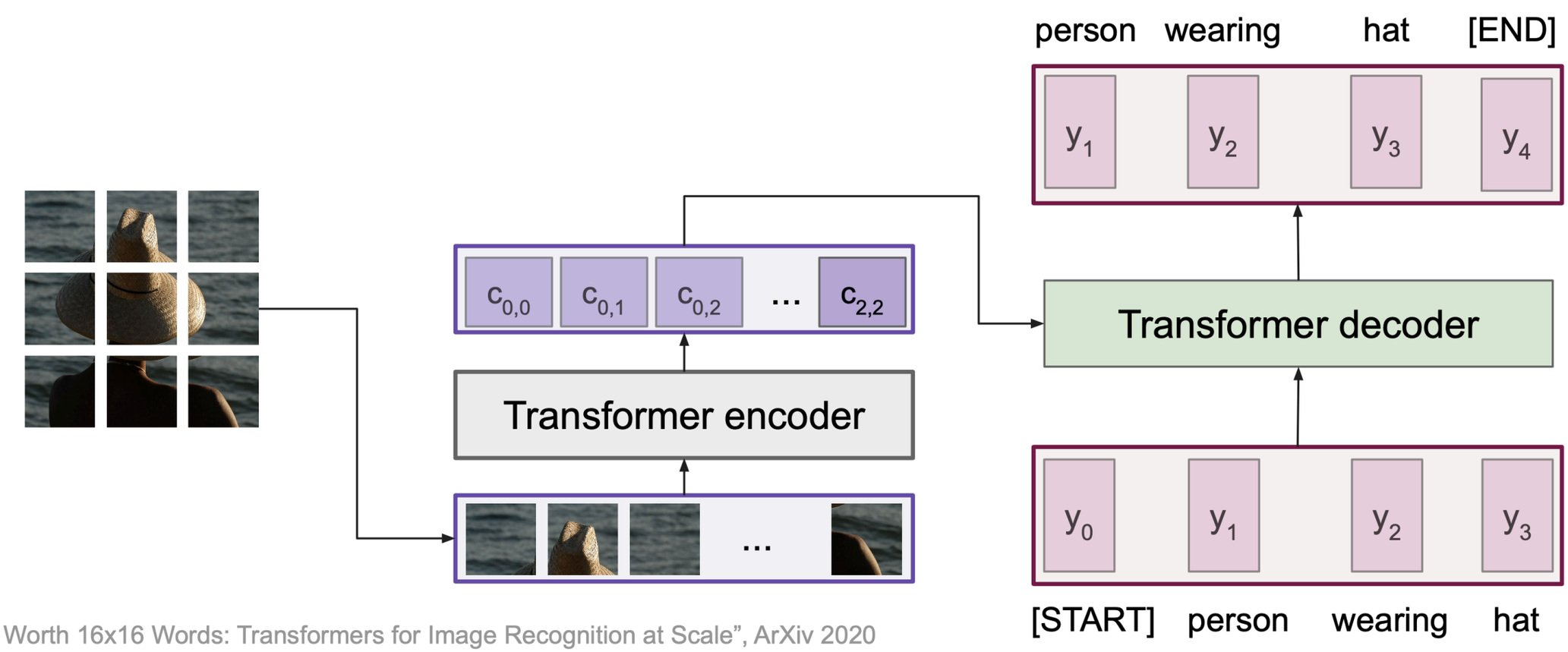
- **No recurrence at all**



From Fei-Fei Li et al: Attention and Transformers

只使用Transformer的图像描述

- **Transformers from pixels to language**



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ArXiv 2020
Colab link to an implementation of vision transformers

From Fei-Fei Li et al: Attention and Transformers

Vision Transformers vs. ResNets

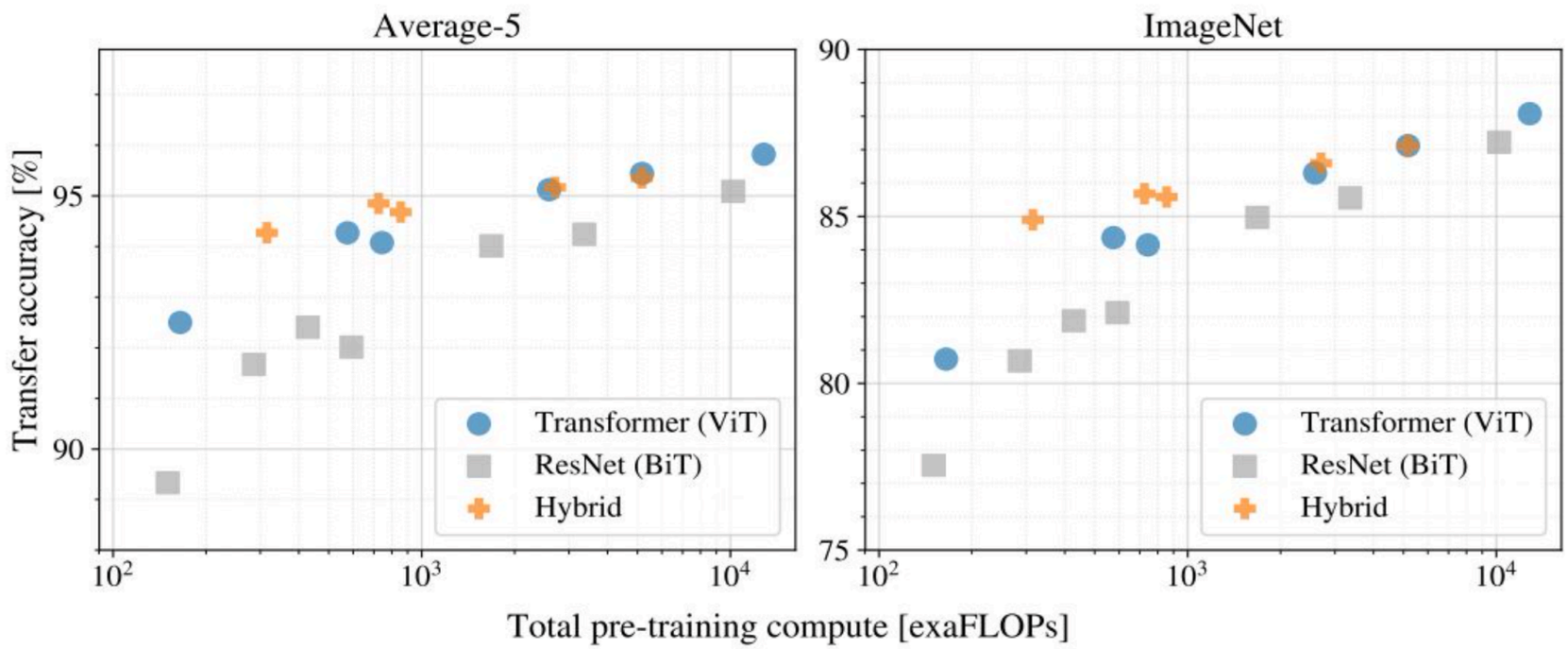
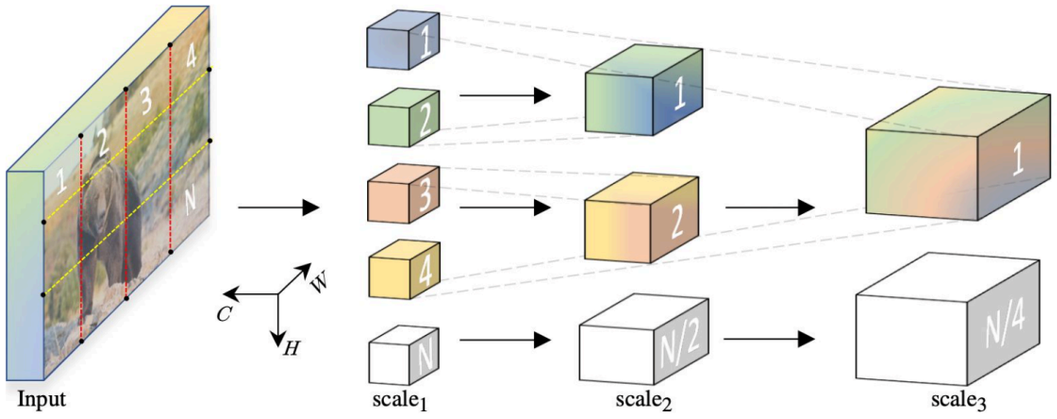


Figure 5: Performance versus cost for different architectures: Vision Transformers, ResNets, and hybrids. Vision Transformers generally outperform ResNets with the same computational budget. Hybrids improve upon pure Transformers for smaller model sizes, but the gap vanishes for larger models.

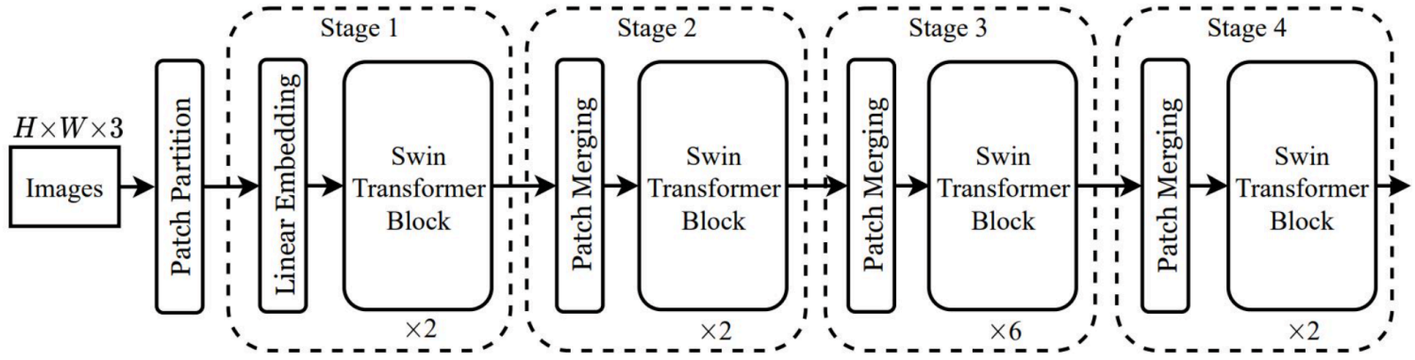
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ArXiv 2020
Colab link to an implementation of vision transformers

From Fei-Fei Li et al: Attention and Transformers

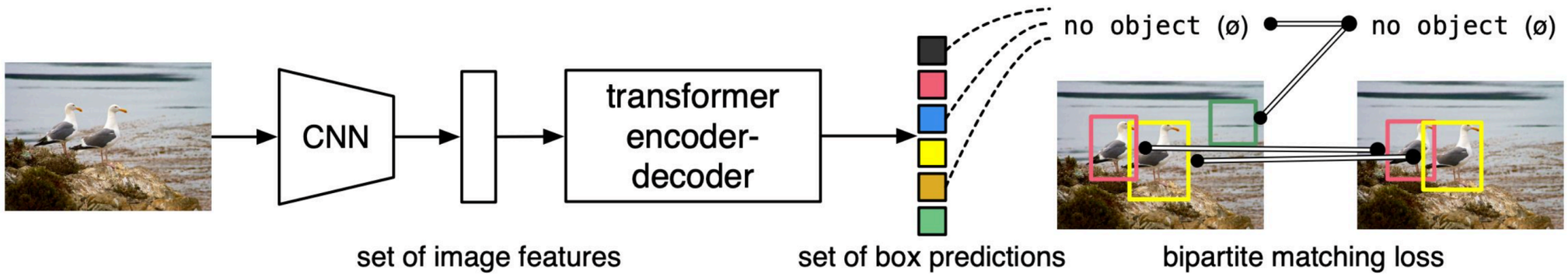
Vision Transformers — 视觉 Transformer



Fan et al, “Multiscale Vision Transformers”, ICCV 2021



Liu et al, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, CVPR 2021



Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

总结:

- 在 RNN中增加注意力机制能使其在每个时间步上关注输入的不同部分;
- 注意力层是当前构建深度神经网络的基础模块;
- Transformer 的主要模块包括自注意力层和归一化层
- 它具备极强的可扩展性和可并行性;
- 使用 Transformer 可以更快训练出更大的模型，在自然语言处理和计算机视觉任务上都有很好的表现;
- 它很快取代了RNN，LSTM等时序模型，甚至CNN也有可能被完全取代（？）;

From Fei-Fei Li et al: Attention and Transformers