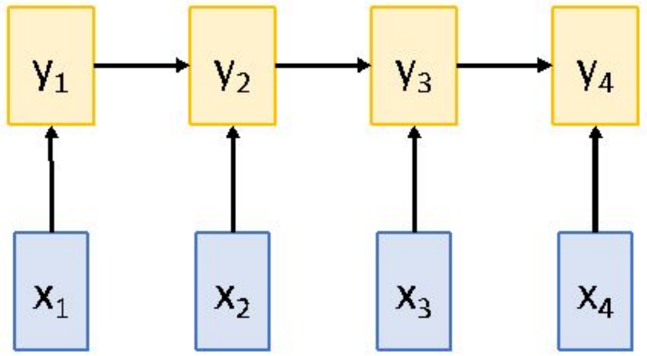
Self-Attention Layer

What happens in a layer ?

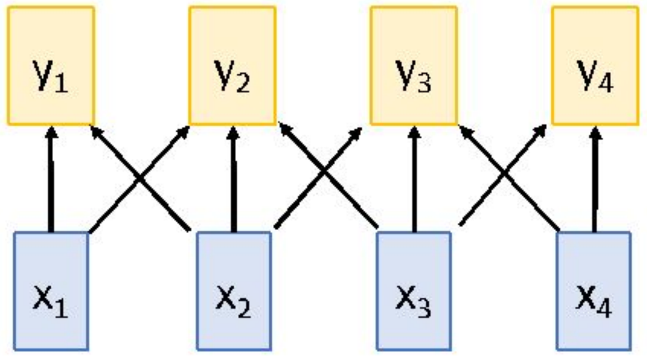
- Input: a set/sequence/grid of vectors of representations
- Output: a set/sequence/grid of vectors of representation
- **Solution:**
  - **RNN Layer**
  - **CNN Layer**
  - **Self Attention Layer/Transformer**
- **Deep Neural Architecture**
  - **Representation goes through a sequence of I/O transformations that enrich the semantics and make it more suitable for tasks**

Conceptual Comparison

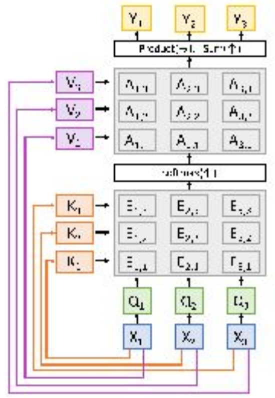
Recurrent Neural Network



1D Convolution



Self-Attention



Works on **Ordered Sequences**
(+) Good at long sequences: After one RNN layer, h_T "sees" the whole sequence
(-) Not parallelizable: need to compute hidden states sequentially

Works on **Multidimensional Grids**
(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence
(+) Highly parallel: Each output can be computed in parallel

Works on **Sets of Vectors**
(-) Good at long sequences: after one self-attention layer, each output "sees" all inputs!
(+) Highly parallel: Each output can be computed in parallel
(-) Very memory intensive

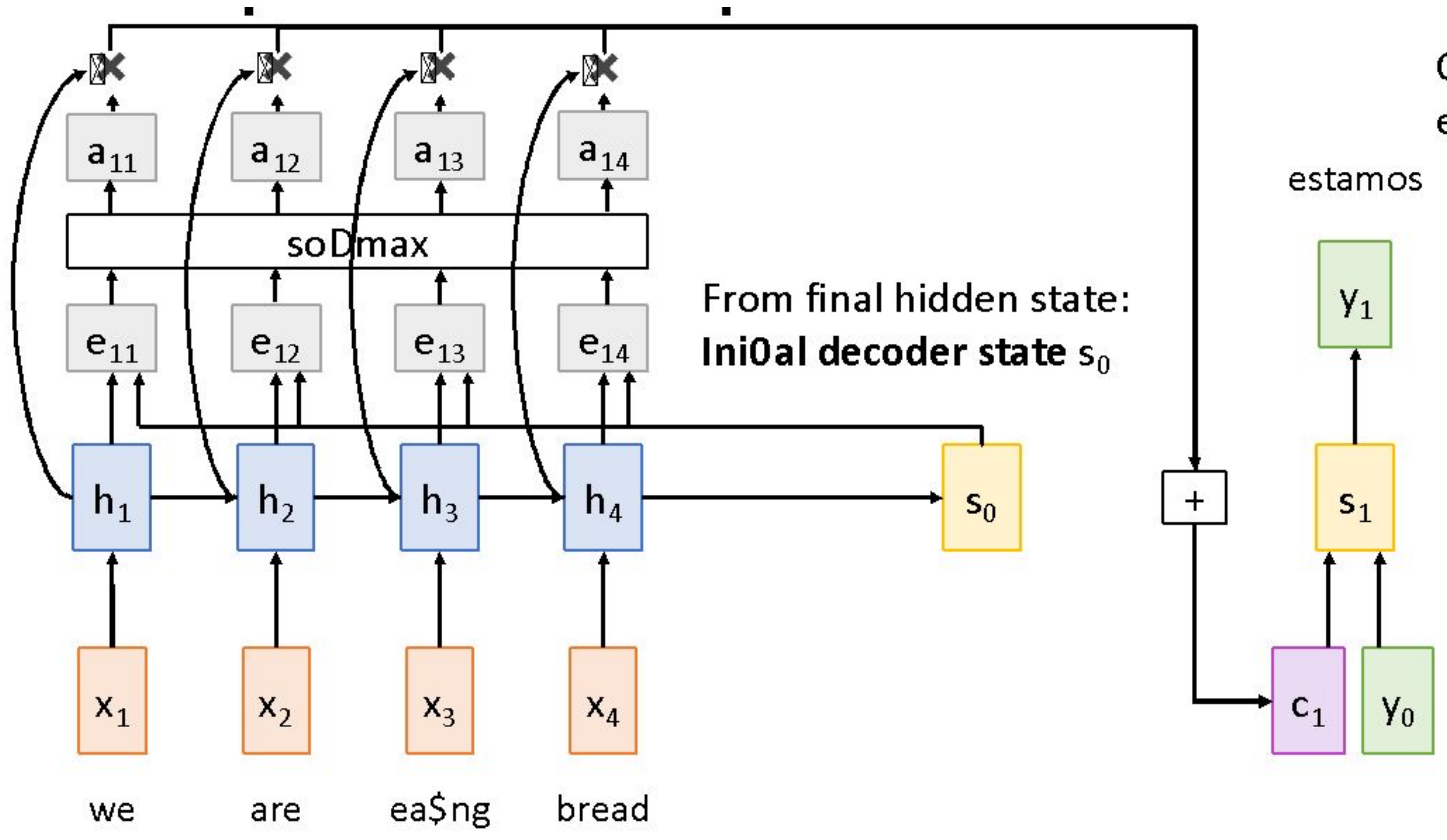
Convolution Layer Vs SA Layer

Blank Slide

Comments

Challenges with RNNs	Transformer Networks
• Long range dependencies	• Facilitate long range dependencies
• Gradient vanishing and explosion	• No gradient vanishing and explosion
• Large # of training steps	• Fewer training steps
• Recurrence prevents parallel computation	• No recurrence that facilitate parallel computation

Attention in RNNs



From final hidden state: **Ini0al decoder state** s_0

Compute (scalar) **alignment scores**
$e_{t,i} = f_{aE}(s_{t-1}, h_i)$ (f_{aE} is an MLP)

Normalize alignment scores to get **a< en0on weights**
$0 < a_{t,i} < 1$ $\sum_i a_{t,i} = 0$

Compute context vector as linear combina$on of hidden states
$c_t = \sum_i a_{t,i} h_i$

Use context vector in decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

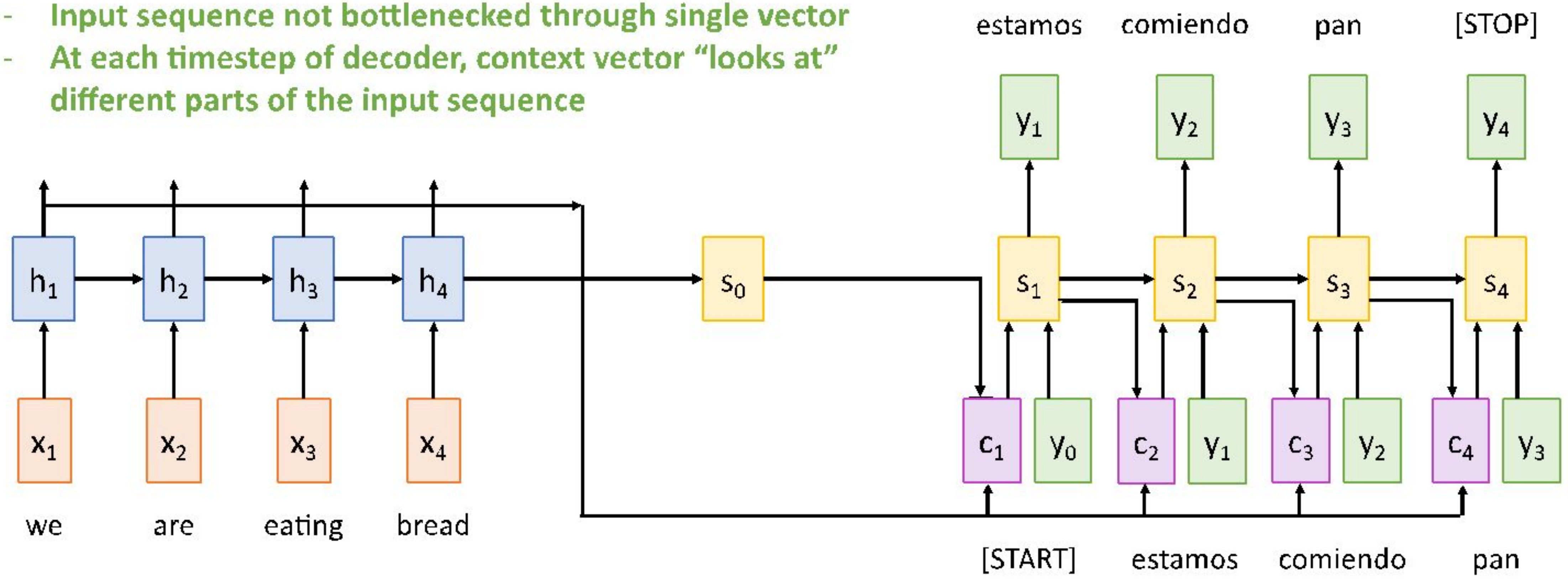
This is all differen0able! Do not supervise a< en0on weights – backprop through everything

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

Attention in RNNs (Seq2Seq)

Use a different context vector in each timestep of decoder

- Input sequence not bottlenecked through single vector
- At each timestep of decoder, context vector "looks at" different parts of the input sequence



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

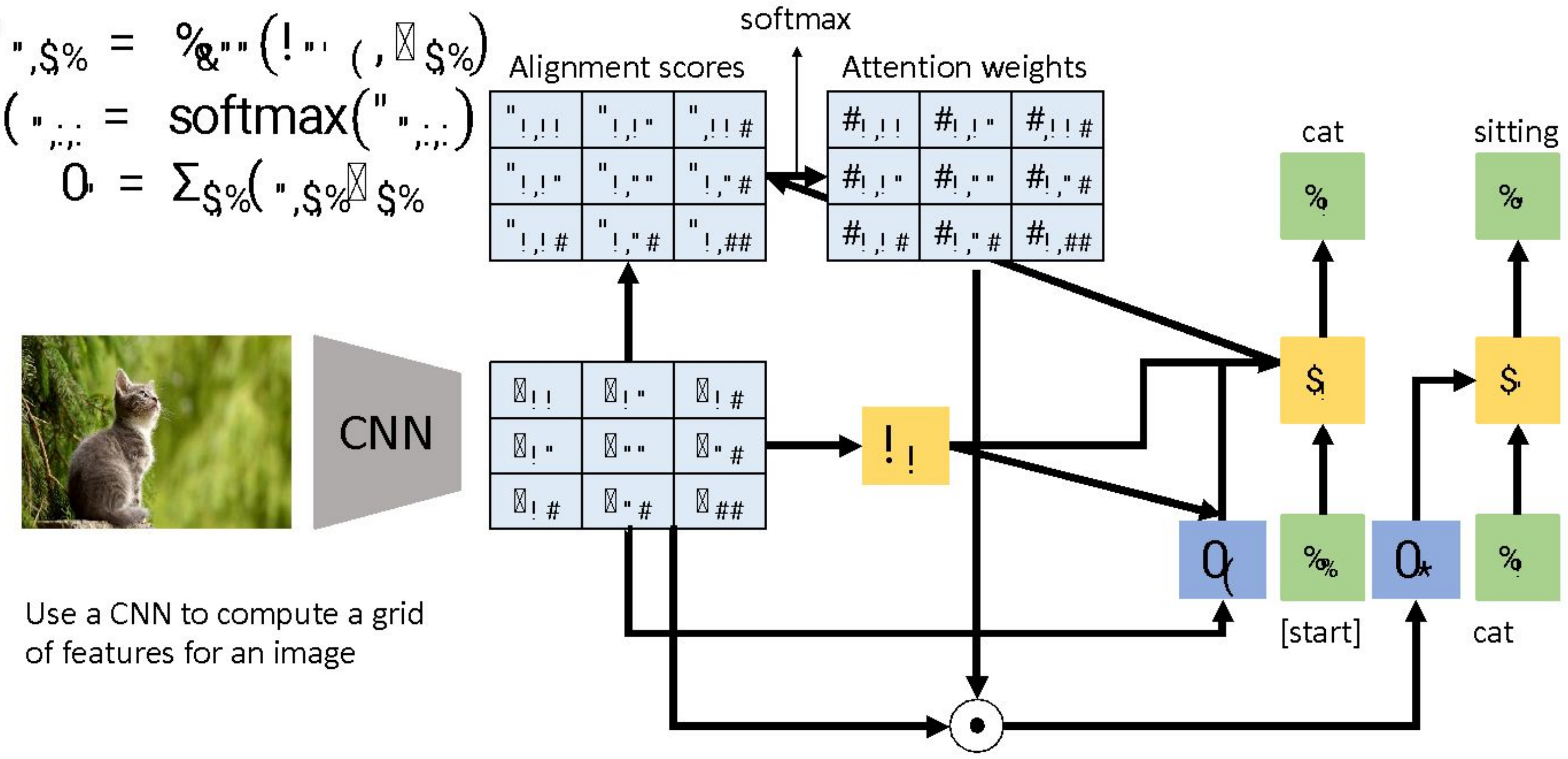
Blank Slide

Attention in CNNs (Image Captioning)

" ,$% = %&"" (! "' (, ⌧ $%)

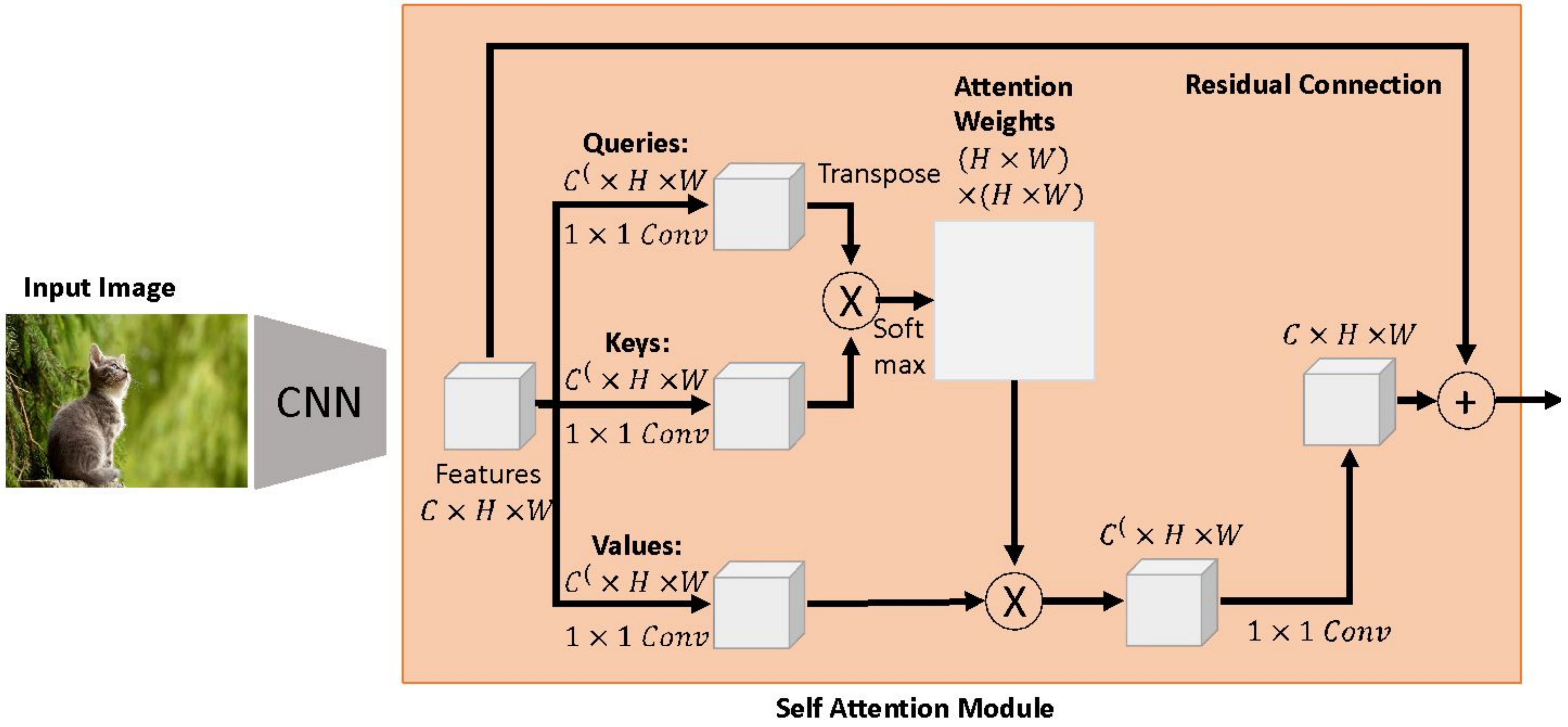
(",;,: = softmax(" ",;,:)

0' = Σ$%(",$%⌧ $%

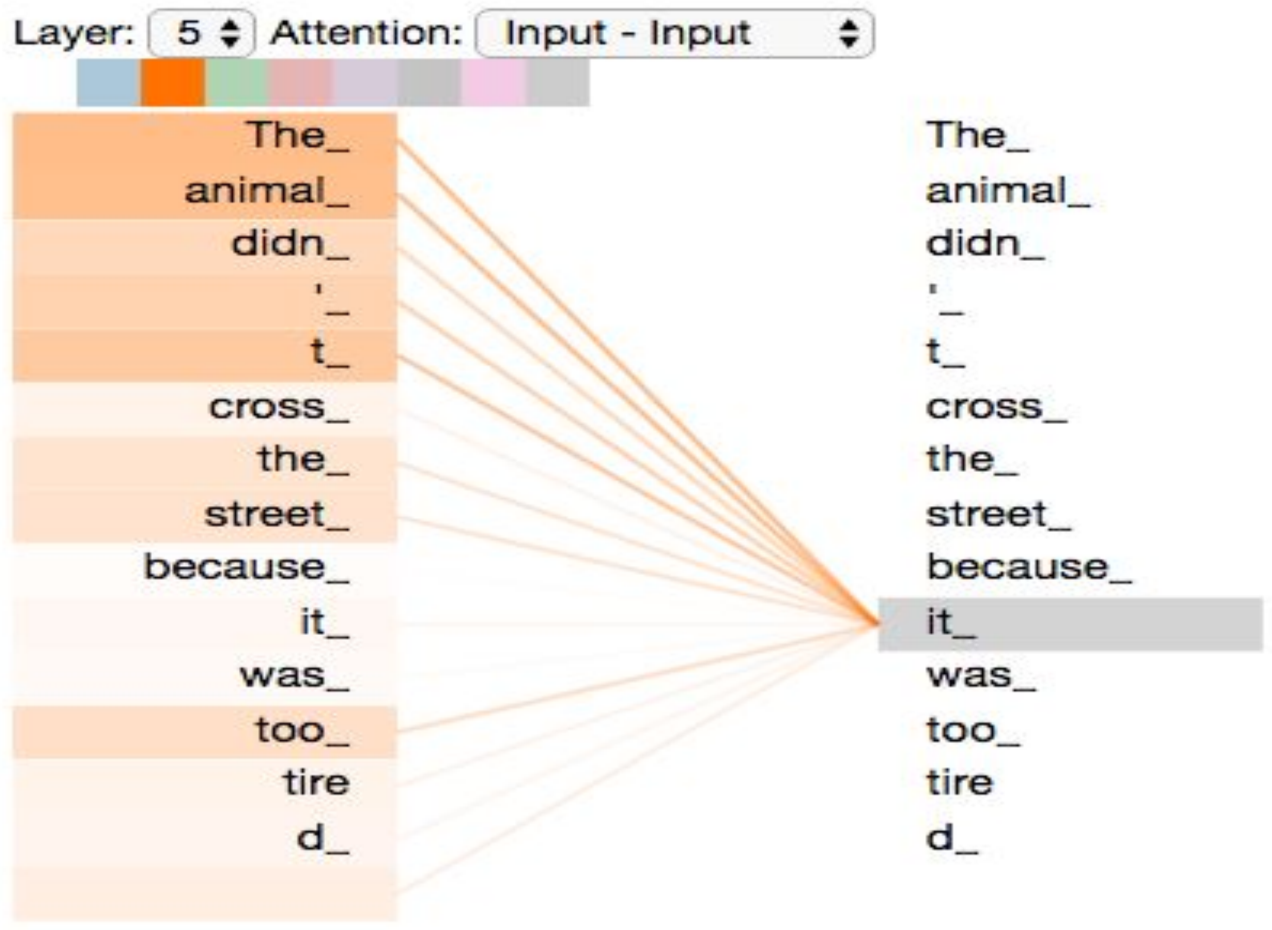


Use a CNN to compute a grid of features for an image

Attention in CNNs



Self Attention



1. ***The animal** did not cross the road because **it** was too tired.*
2. *The animal did not cross **the road** because **it** was too wide.*

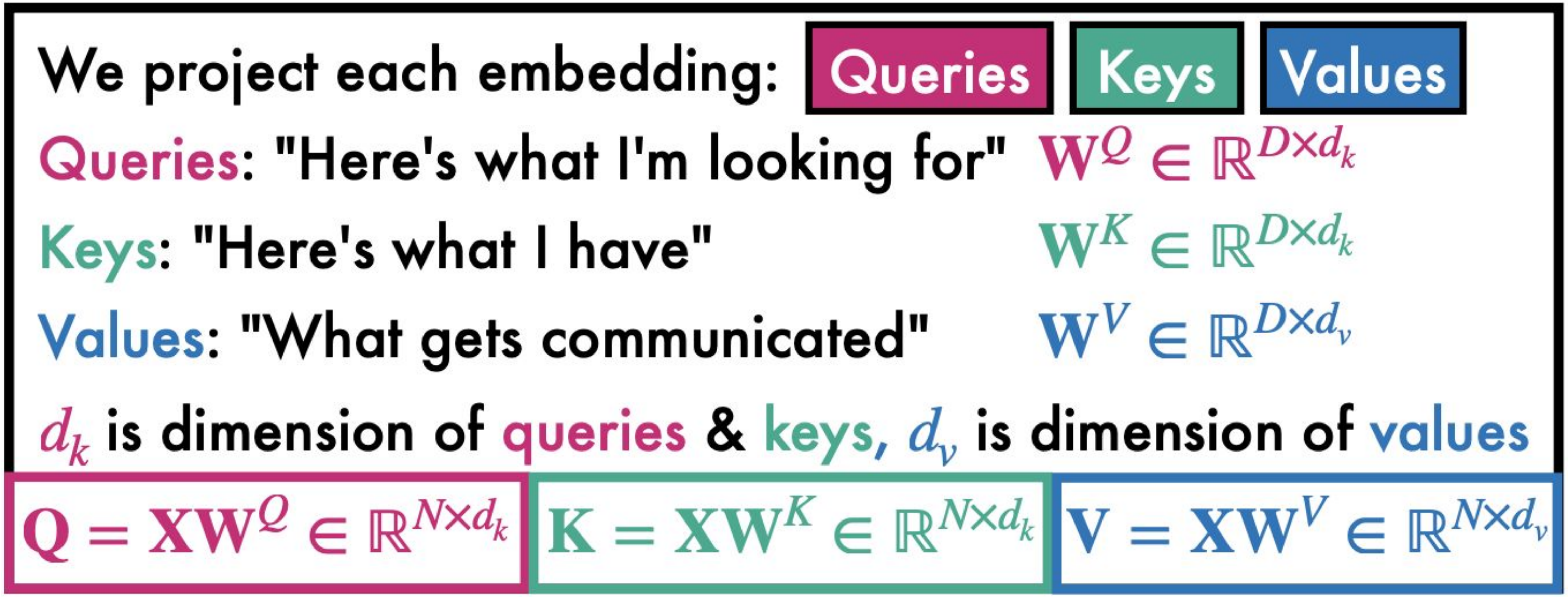
What do we want to happen in the SA layer?
How does it compare with what we do in FC, or CNN?

Blank Slide

Self Attention: Comments

- Self attention learns the relationship between elements in a sequence.
  - say between words in a sentence
- Self Attention Vs Convolution
  - Filters are dynamically calculated instead of static filters
  - SA is invariant to changes in the input points
  - SA can operate on irregular inputs
- SA allows to learn global and local features
  - Hierarchical feature learning by cascading

Query, Key and Value



We project each embedding: Queries Keys Values

Queries: "Here's what I'm looking for" $\mathbf{W}^{Q} \in \mathbb{R}^{D \times d_k}$

Keys: "Here's what I have" $\mathbf{W}^{K} \in \mathbb{R}^{D \times d_k}$

Values: "What gets communicated" $\mathbf{W}^{V} \in \mathbb{R}^{D \times d_v}$

d_k is dimension of queries & keys, d_v is dimension of values

$\mathbf{Q} = \mathbf{X}\mathbf{W}^{Q} \in \mathbb{R}^{N \times d_k}$ $\mathbf{K} = \mathbf{X}\mathbf{W}^{K} \in \mathbb{R}^{N \times d_k}$ $\mathbf{V} = \mathbf{X}\mathbf{W}^{V} \in \mathbb{R}^{N \times d_v}$

Query, Key and Value

- As *the current focus of attention* when being compared to all of the other preceding inputs. We'll refer to this role as a **query**.
- In its role as *a preceding input* being compared to the current focus of attention. We'll refer to this role as a **key**.
- And finally, as a **value** used to compute the output for the current focus of attention.

$$\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^\mathbf{Q};\ \ \mathbf{k}_i = \mathbf{x}_i\mathbf{W}^\mathbf{K};\ \ \mathbf{v}_i = \mathbf{x}_i\mathbf{W}^\mathbf{V}$$

Blank Slide

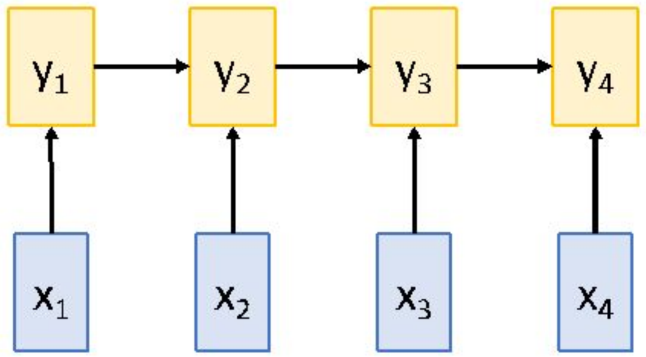
Computation

$$
\begin{aligned}
q_i = x_i W^Q ; k_i &= x_i W^K ; v_i = x_i W^V \\
\mathrm{score}(x_i, x_j) &= \frac{q_i \cdot k_j}{\sqrt{d_k}} \\
a_{ij} &= \mathrm{softmax}(\mathrm{score}(x_i, x_j)) \quad \forall j \leq i \\
a_i &= \sum_{j \leq i} a_{ij} v_j
\end{aligned}
$$

Blank Slide

Conceptual Comparison

Recurrent Neural Network

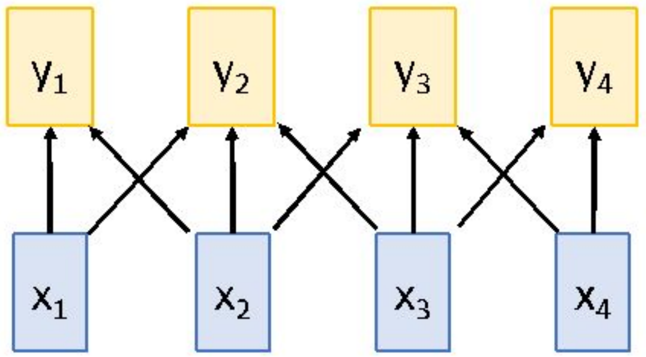


Works on **Ordered Sequences**

(+) Good at long sequences: After one RNN layer, h_T ”sees” the whole sequence

(-) Not parallelizable: need to compute hidden states sequentially

1D Convolution

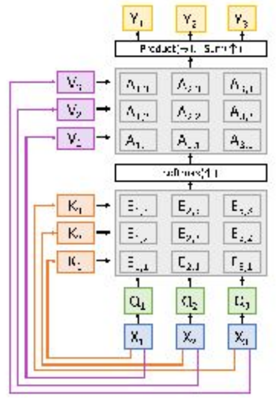


Works on **Multidimensional Grids**

(-) Bad at long sequences: Need to stack many conv layers for outputs to “see” the whole sequence

(+) Highly parallel: Each output can be computed in parallel

Self-Attention



Works on **Sets of Vectors**

(-) Good at long sequences: after one self-attention layer, each output “sees” all inputs!

(+) Highly parallel: Each output can be computed in parallel

(-) Very memory intensive

Convolution Layer Vs SA Layer

Blank Slide

Blank Slide

Blank Slide

Where we want to go?

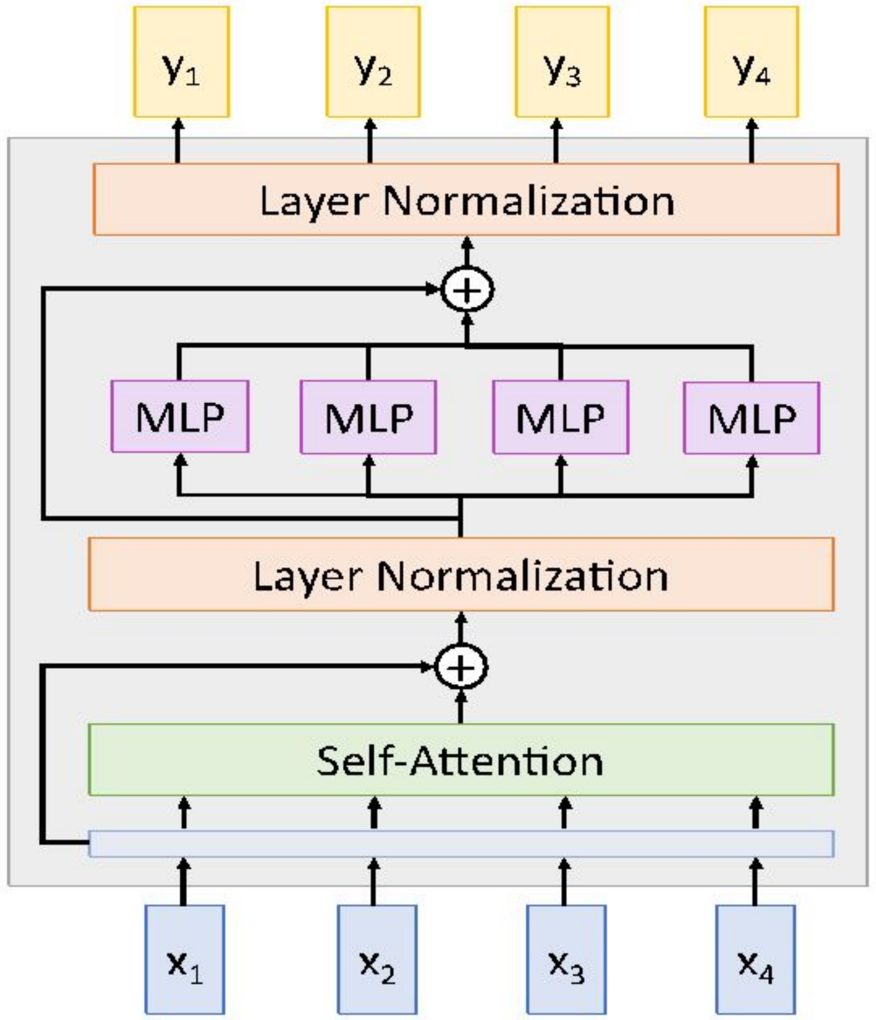
The Transformer

Transformer Block:
Input: Set of vectors x
Output: Set of vectors y

Self-attention is the only interaction between vectors!

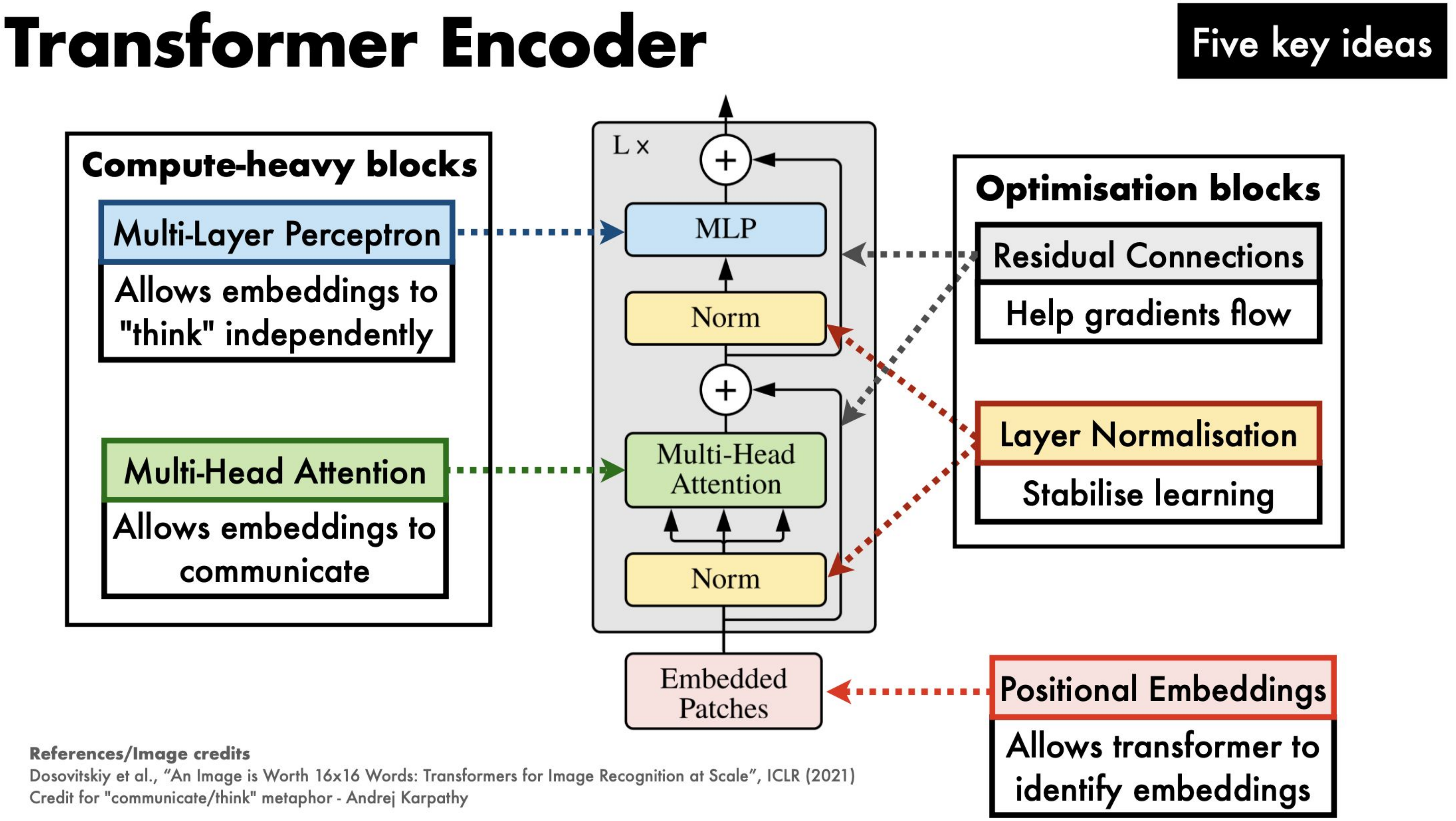
Layer norm and MLP work independently per vector

Highly scalable, highly parallelizable



Vaswani et al, "Attention is all you need", NeurIPS 2017

Transformers



References/Image credits
Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR (2021)
Credit for "communicate/think" metaphor - Andrej Karpathy

Blank Slide

Blank Slide

Thanks!!

Questions?