# Attention and Transformers-I

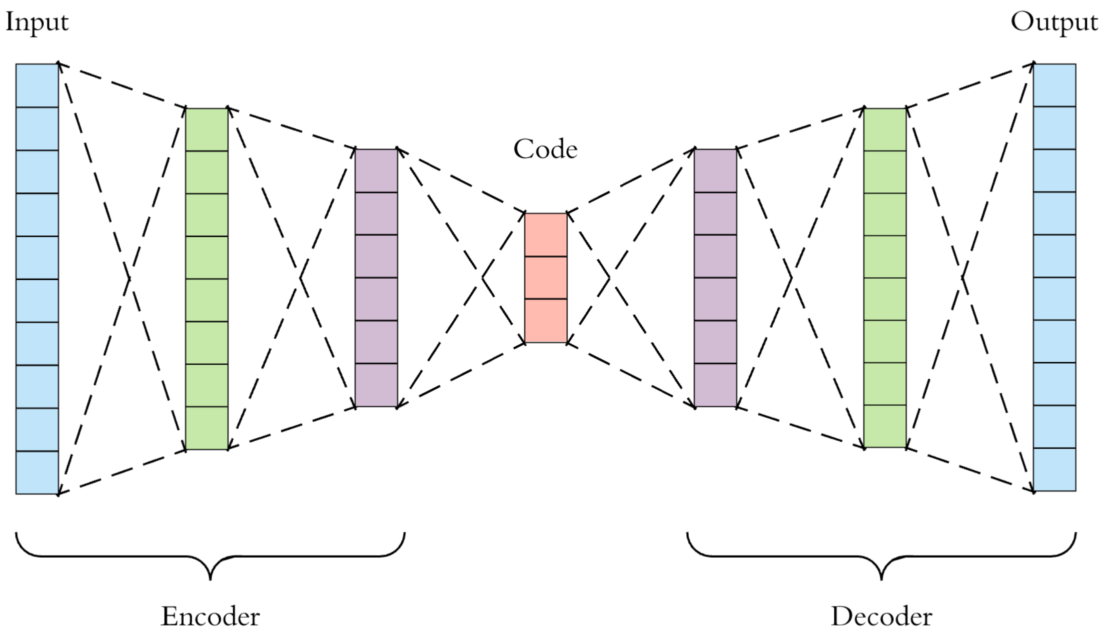
# Two Example Tasks

1. Machine Translation:
   - Input: “He loved to eat”
   - Output: “Er leibte zu essen” (German)
2. Image Captioning
   - Input: Image
   - Output: “Cat Sat Outside”

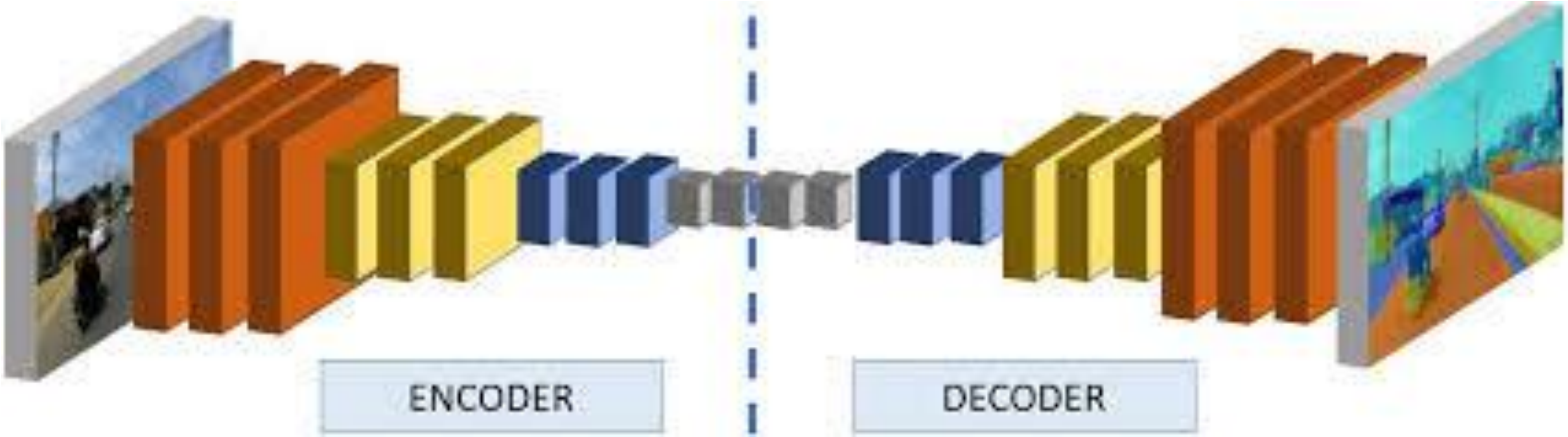
- Two Possible Steps
  - Encode the input information
  - Decode into the target modality/language

# Encoder-Decoder: Auto encoders

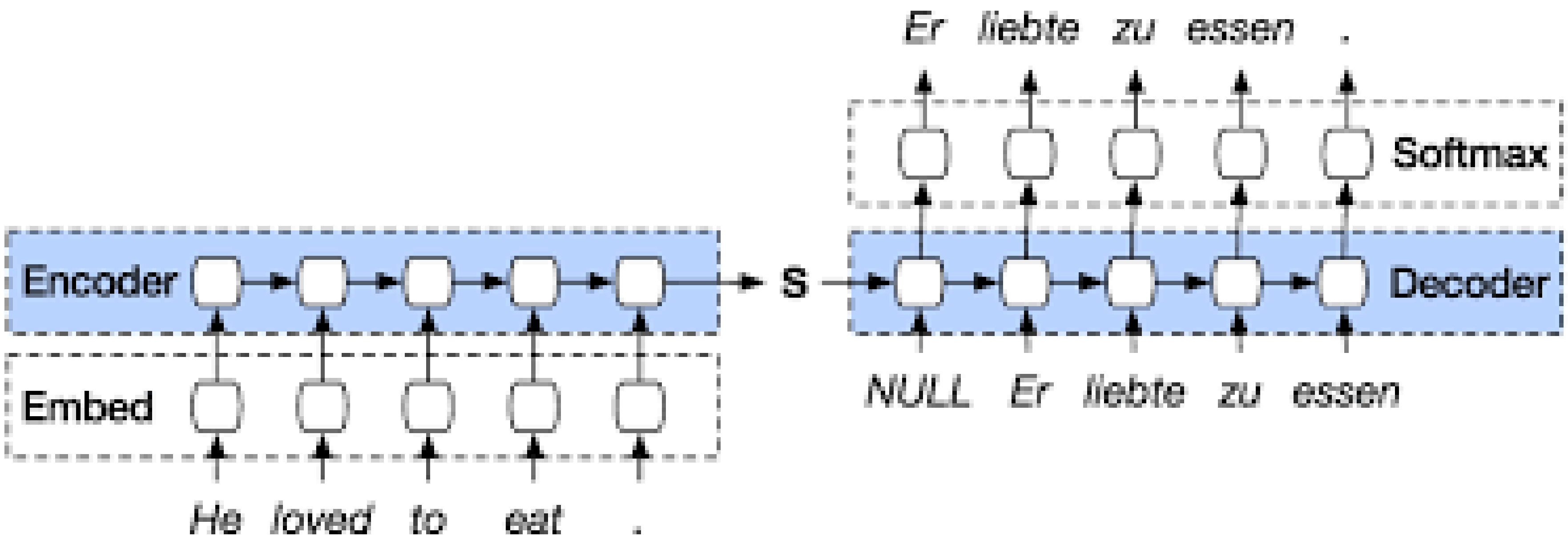
- Encoder Decoder architecture
  - Use for compression
  - Or throw away the decoder after training use the compressed code for supervised learning tasks (particularly useful in problems with small training set)



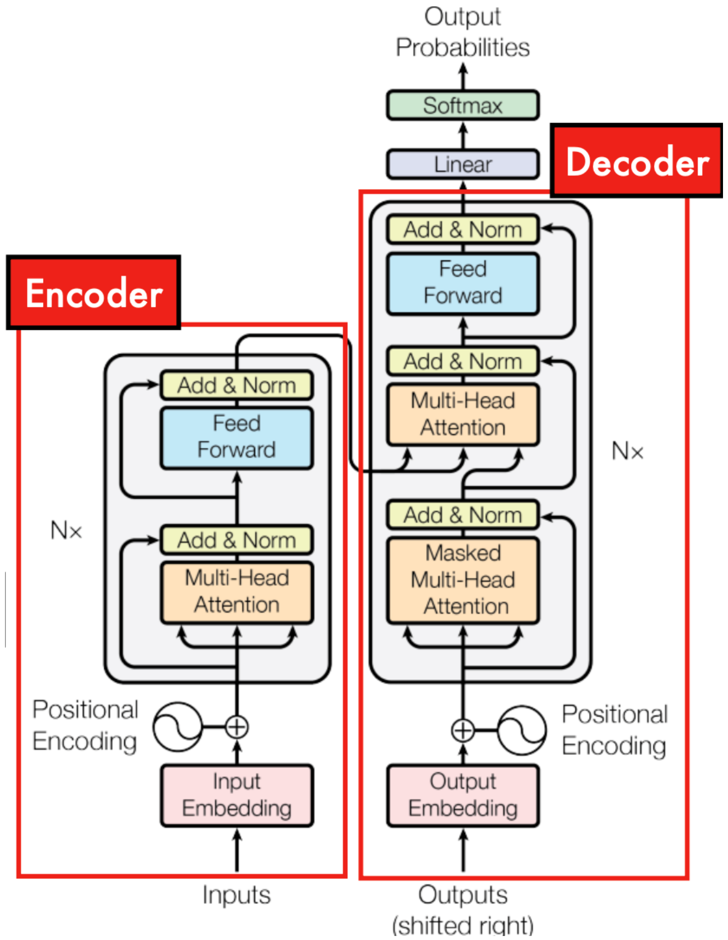
# Encoder-Decoder



# Encoder-Decoder Models: Seq2Seq



# Transformer: Encoder and Decoder

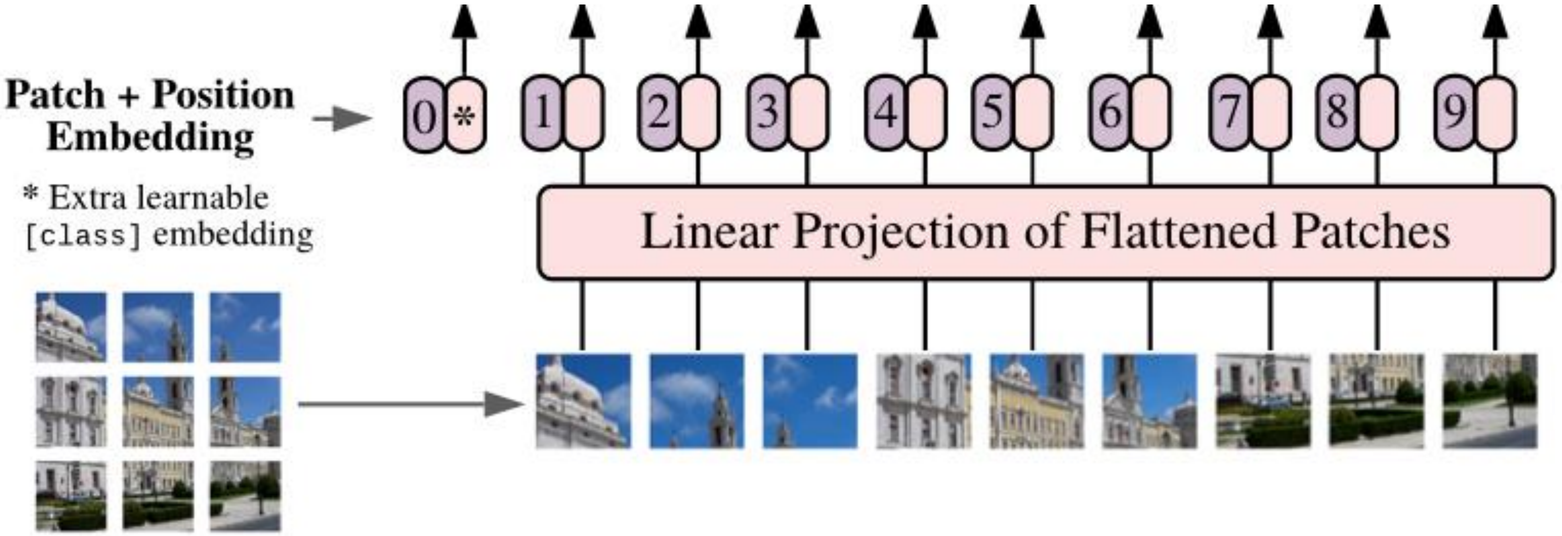


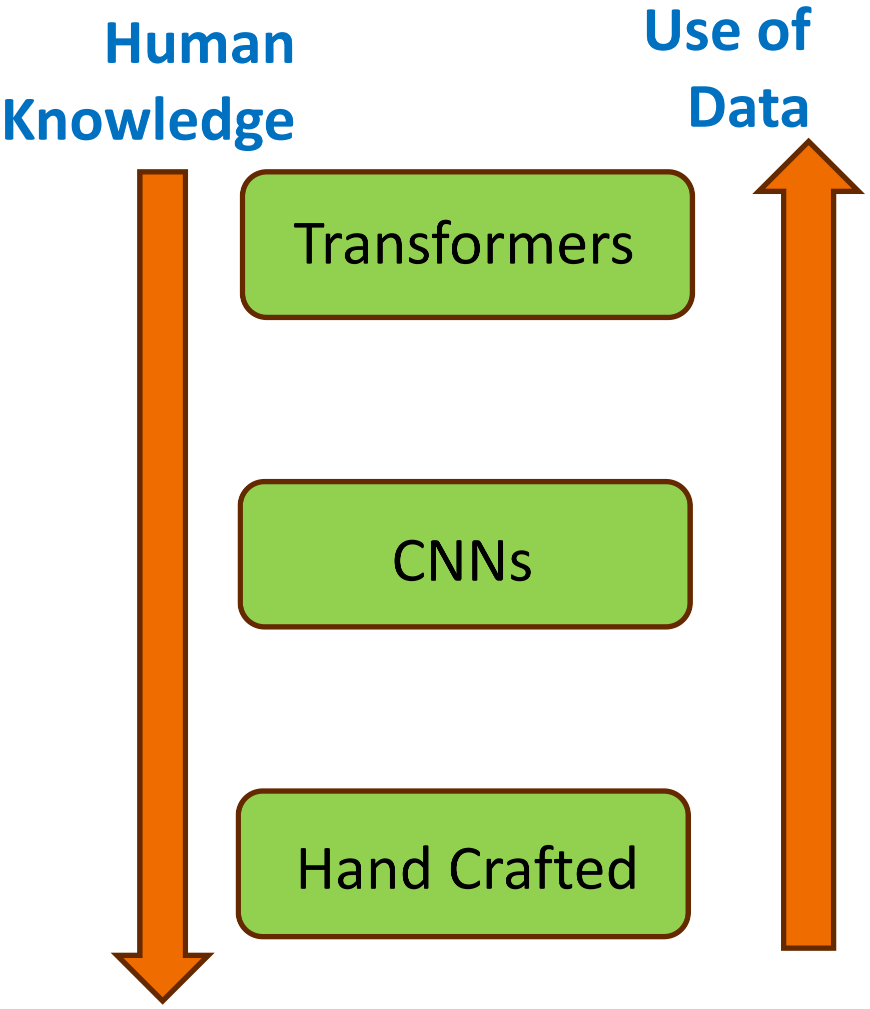

- **Encoder:**
  - Learn useful representation from the input
  - Eg. BERT (encoder only)
- **Decoder:**
  - Decodes the learned representation along with any other inputs and predict the output
  - Eg. GPT-3 (decoder only)
- **Encoder-Decoder:**
  - Eg. Sequence to Sequence Tasks, like Machine Translation

**Attention Is All You Need, 2017**

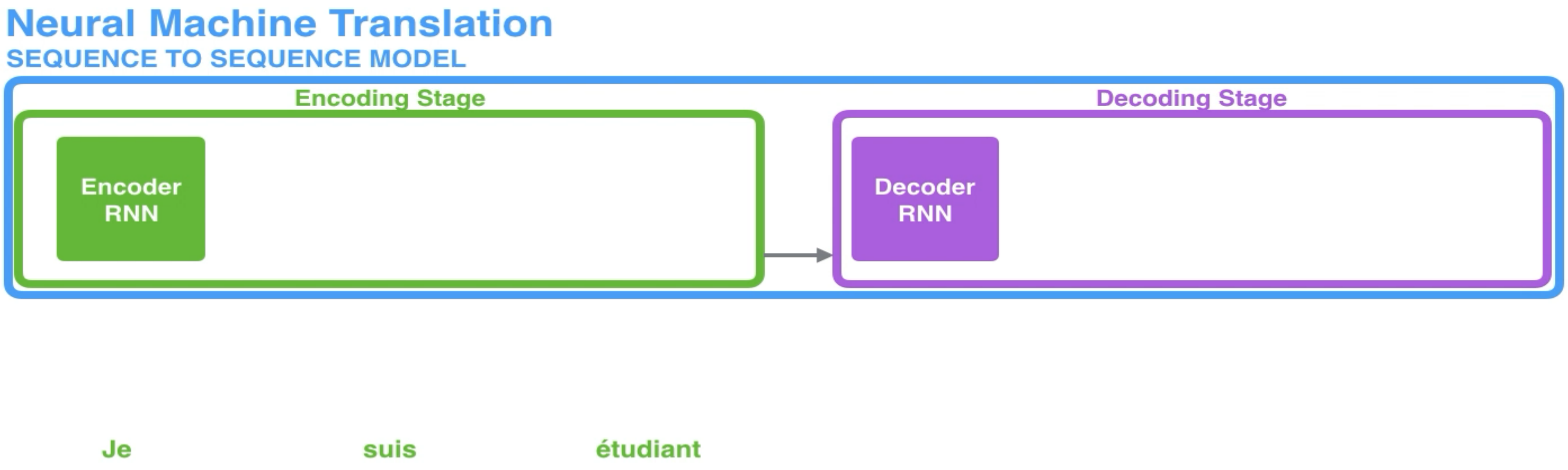
# Modelling Inter Dependency of Elements/Patches

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.*



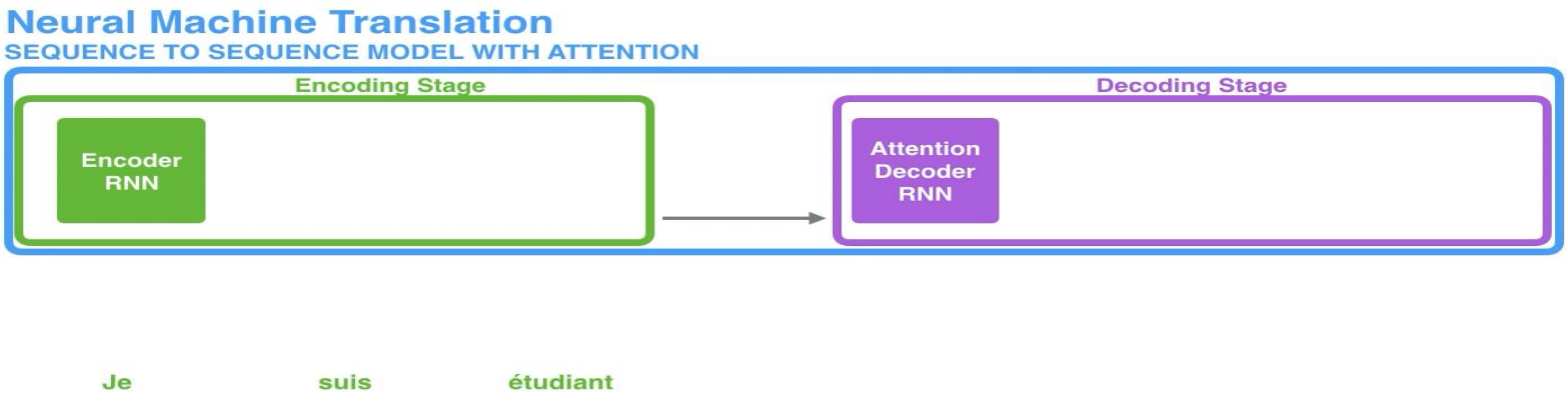



# Seq2Seq



Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL
Encoding Stage
Encoder RNN
Decoding Stage
Decoder RNN
Je
suis
étudiant

# Seq2Seq with attention




The encoder passes a lot more data/information/hidden states to the decoder.

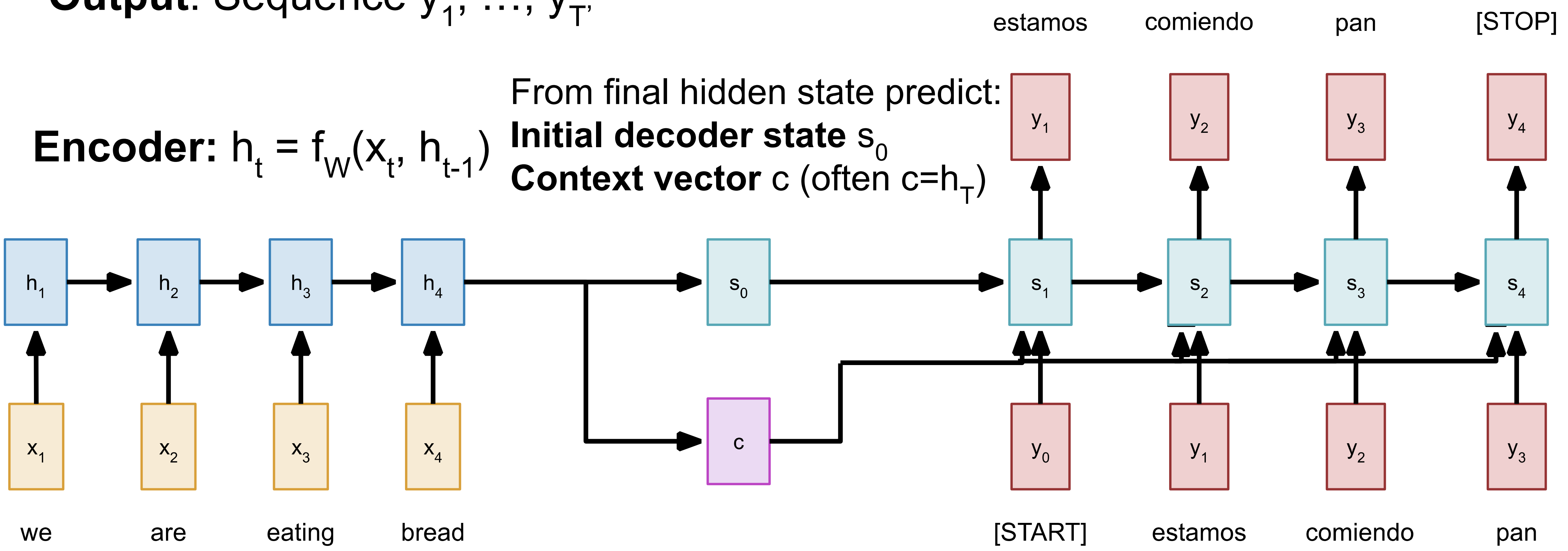
# Strategy : Seq2Seq with Context

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

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

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

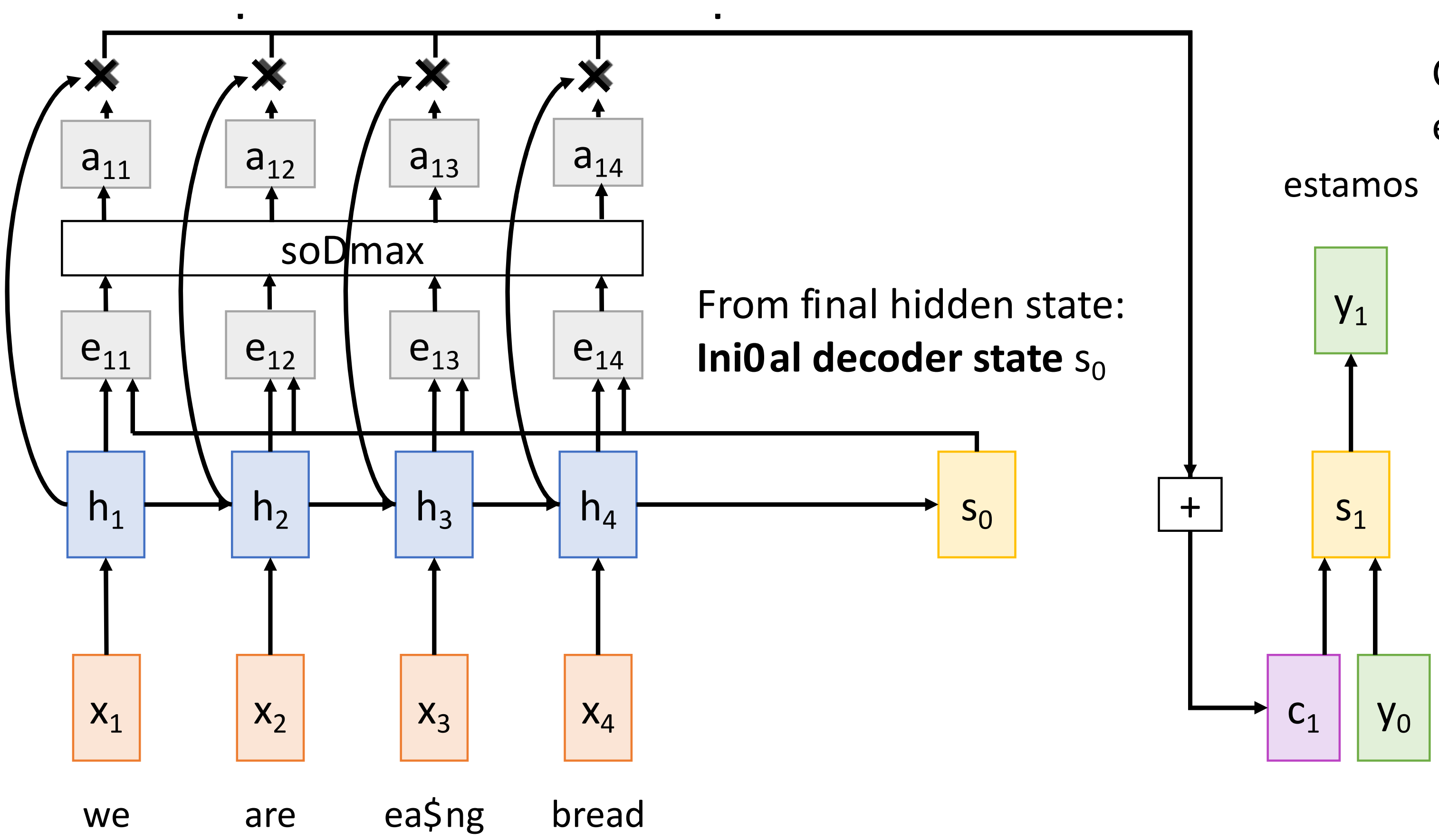
From final hidden state predict:
**Initial decoder state** $s_0$
**Context vector** c (often $c=h_T$)



# Problem

- Input sequence is summarized (bottlenecked) through a fixed size fixed vector.
  - Not good for long sequences/inputs
- **Solution:**
  - Provide more detailed context to the decoder.
  - Use new context vector at each time step.

# Improved Strategy: Computation with “Attention”




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

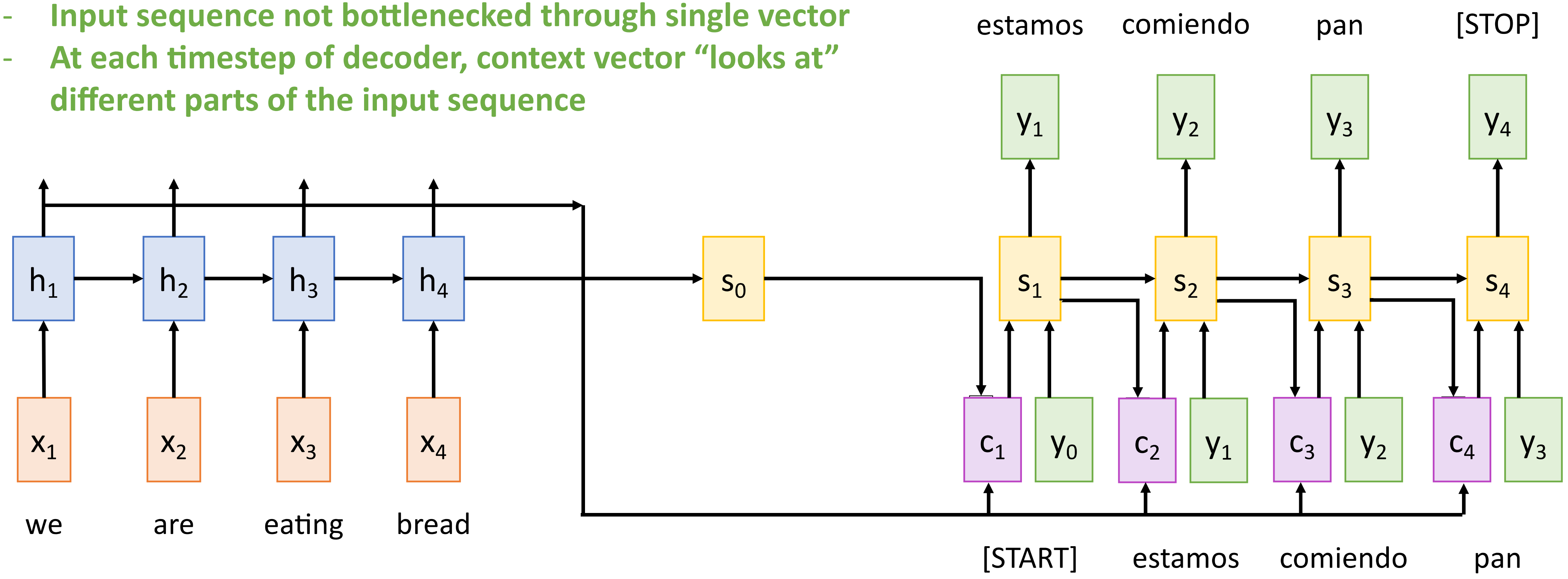
**This is all differen0able! Do not supervise a< en0on weights – backprop through everything**

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

# Attention in Translation (Seq2Seq)

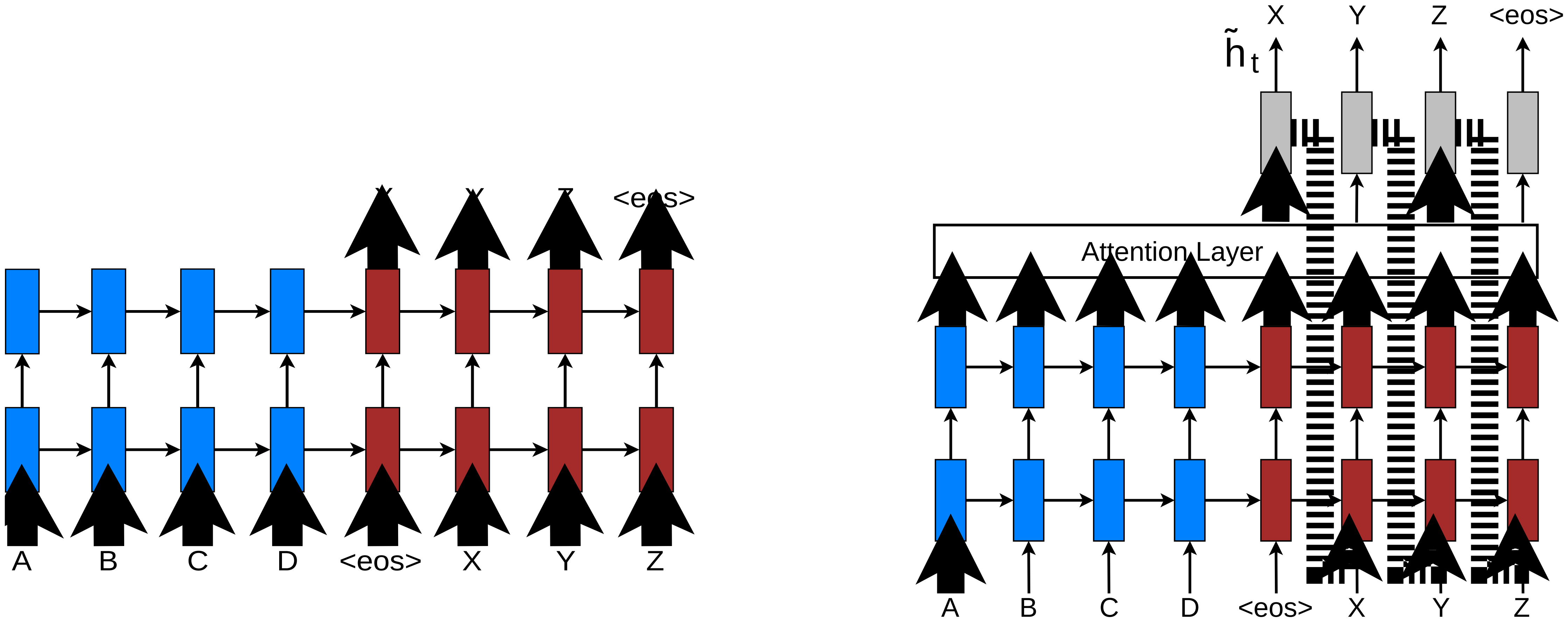
**Use a different context vector in each timestep of decoder**

- **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

# Attention Layer




Luong, Pham and Manning, 2015

# Visualization

**Example**: English to French translation

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

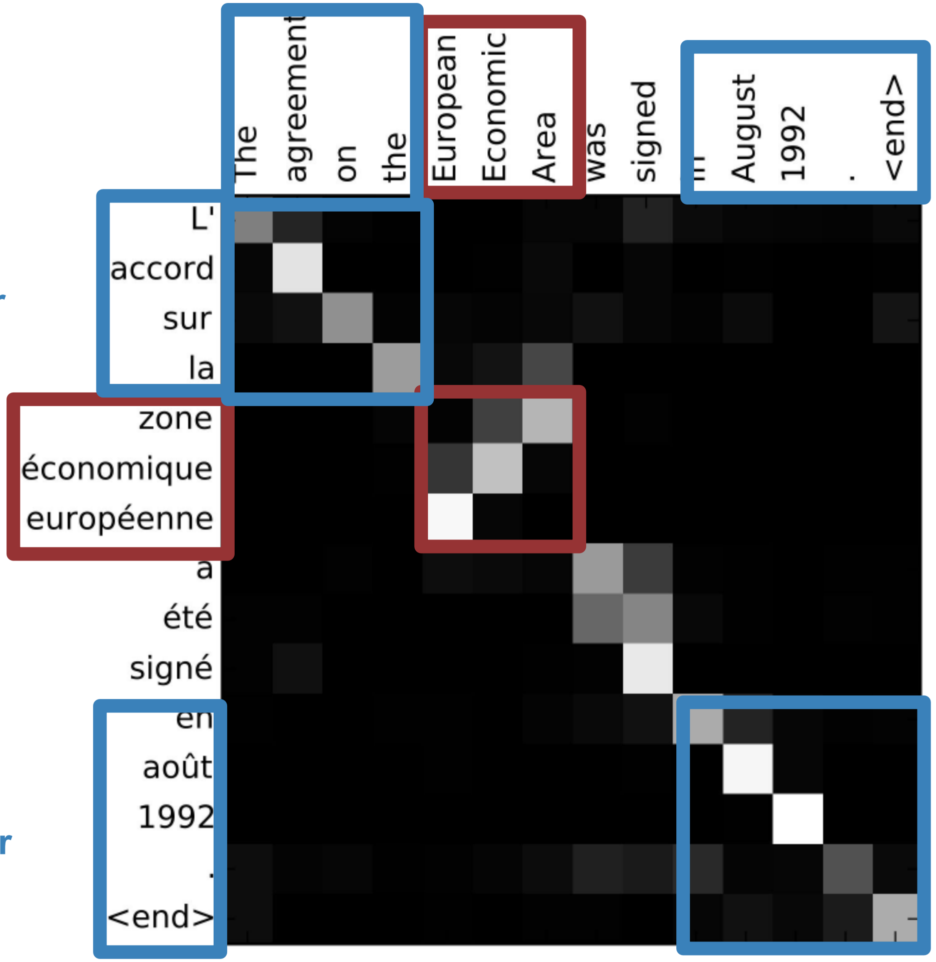
**Output**: "L'accord sur la zone économique européenne a été signé en août 1992."

Visualize attention weights $a_{t,i}$

Diagonal attention means words correspond in order

Attention figures out different word orders

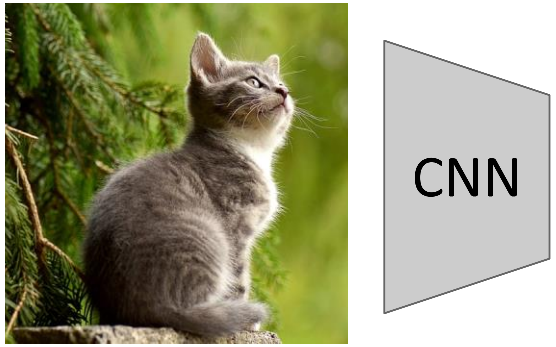
Diagonal attention means words correspond in order



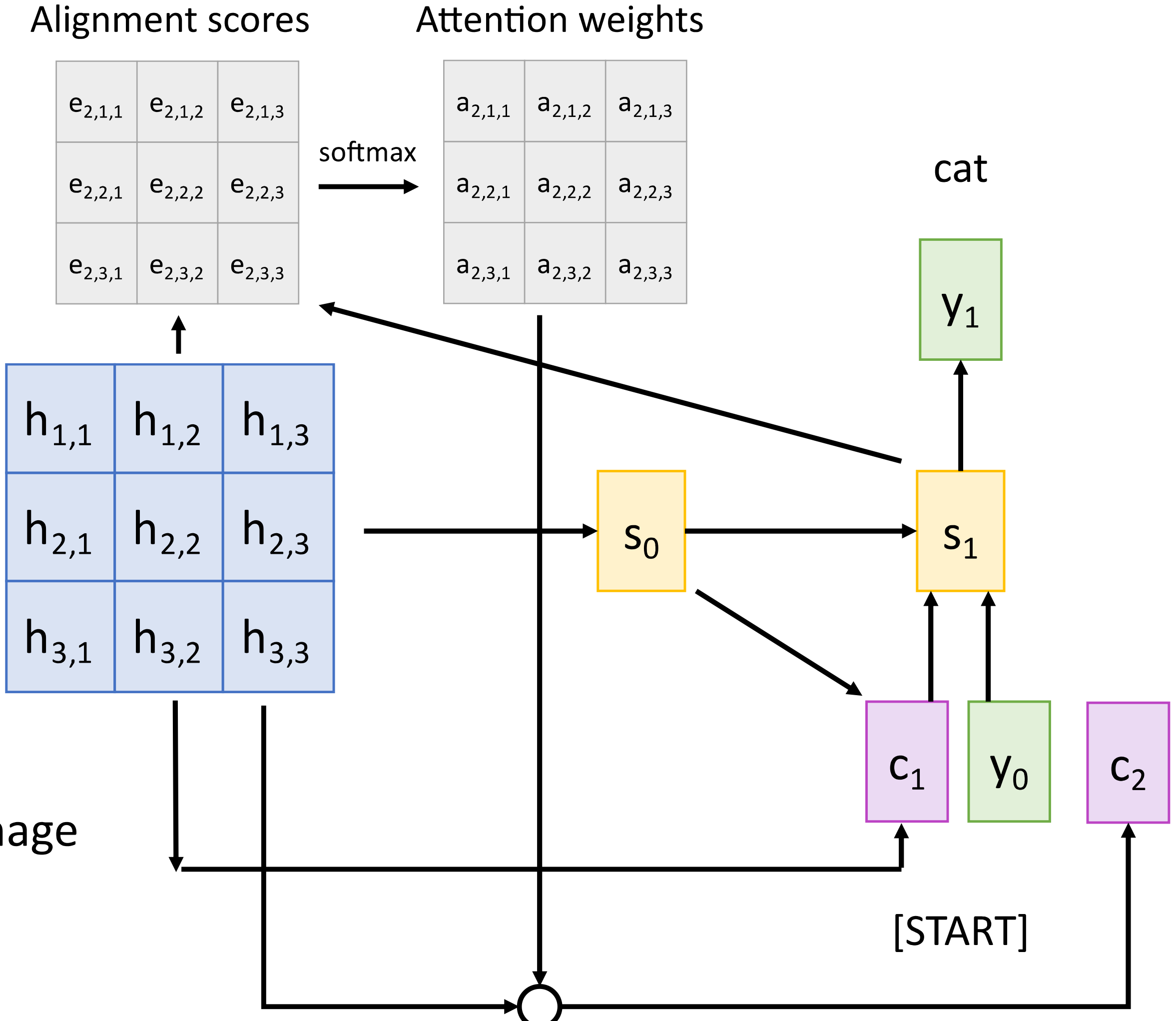

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

# Image captioning with RNN and Attention

$$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$$
$$a_{t,:,:} = \mathrm{softmax}(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$$




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

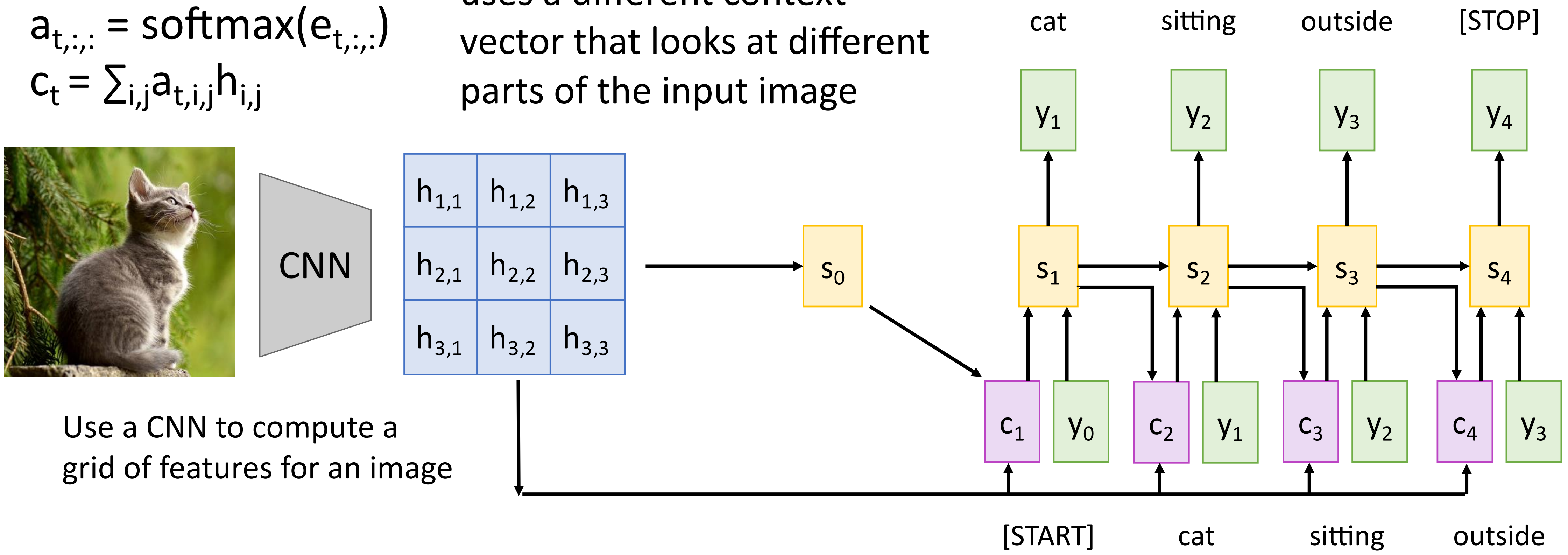



Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Image captioning with RNN and Attention

$$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$$
$$a_{t,:,:} = \mathrm{softmax}(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$$

Each timestep of decoder uses a different context vector that looks at different parts of the input image




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

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

# Blank Slide

# Blank Slide

# Query, Key and Value

# 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:

- Mimics the retrieval of a value v for a query q based on k in the database

q

| K1 | V1 |
|---|---|
| K2 | V2 |
| K3 | V3 |
| K4 | V4 |
| K5 | V5 |

A-i = Similarity (query, key-i)

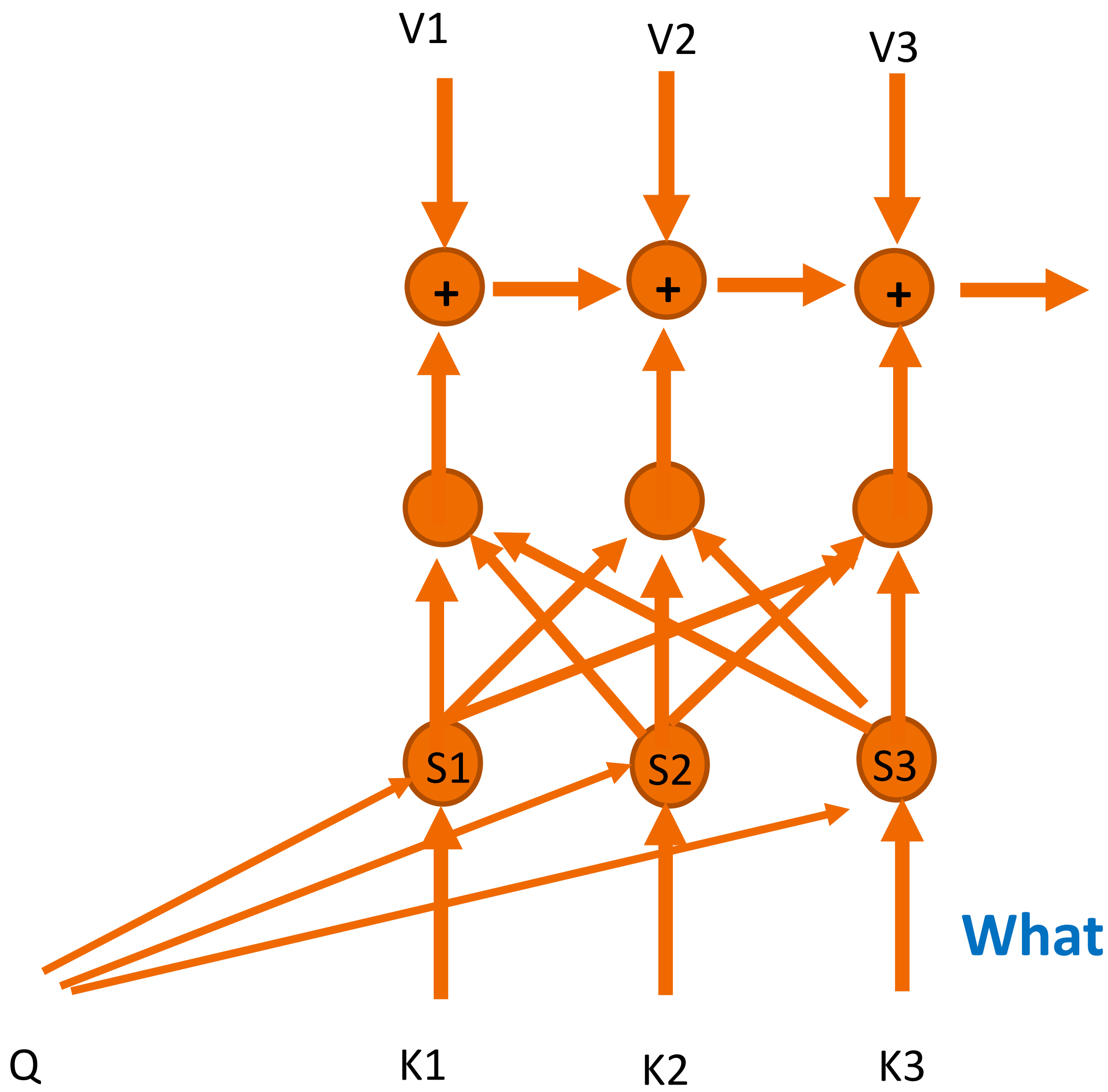
V = weighted sum of A-i and V-i

$$att(q, k, v) = \sum_i s(q, k_i) v_i$$

We compare ”Query” with “Key” and “Value” is returned.

**What is “Query”, “Key” and “Value”?**

# Attention




**What is "Query", "Key" and "Value"?**

# 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

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

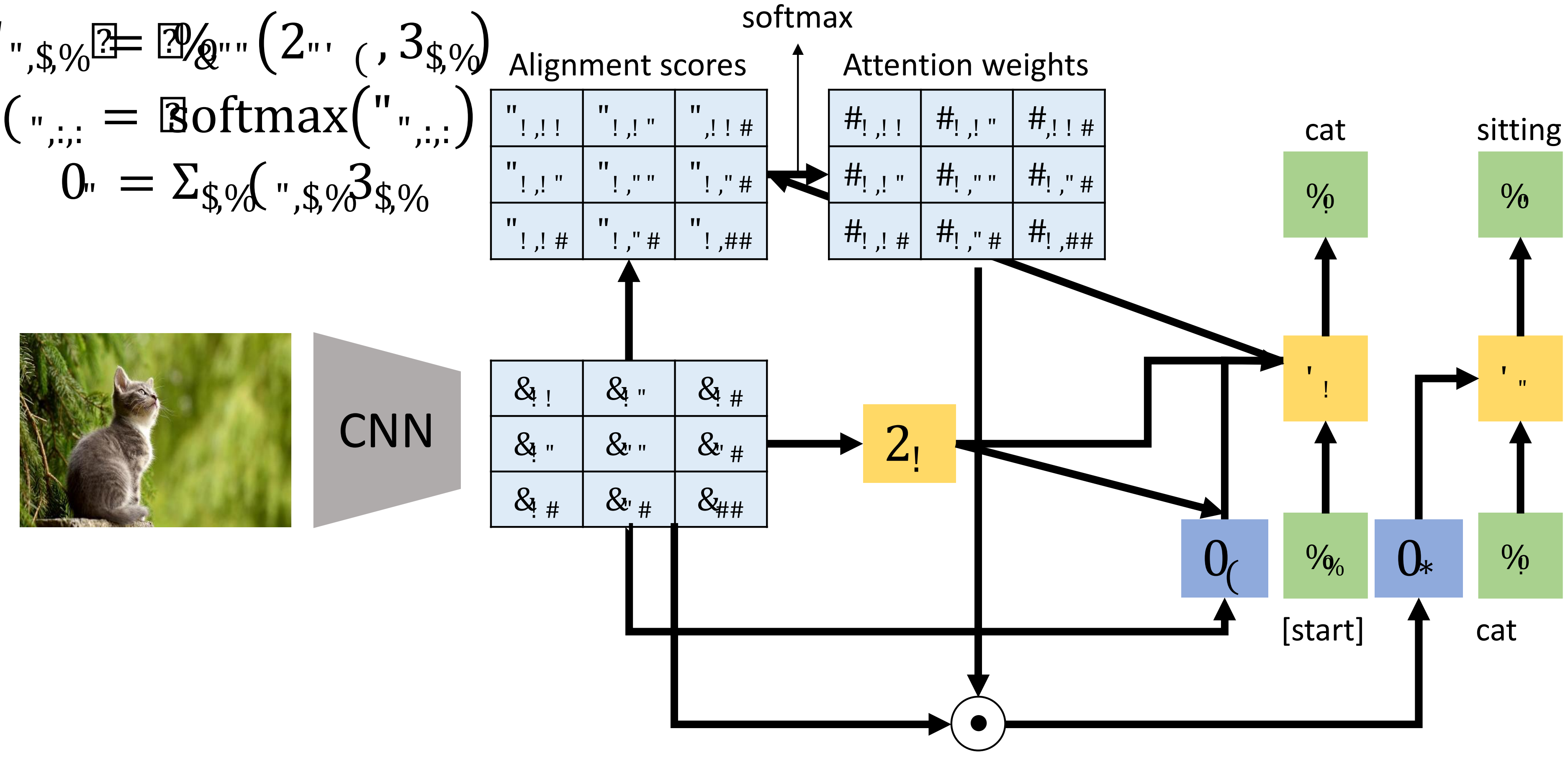
# Towards Scalable and More Generic

## Notation change: ! ¡ to " , and # to $

" ",$,% = %&"" (2"' (,3$,%)

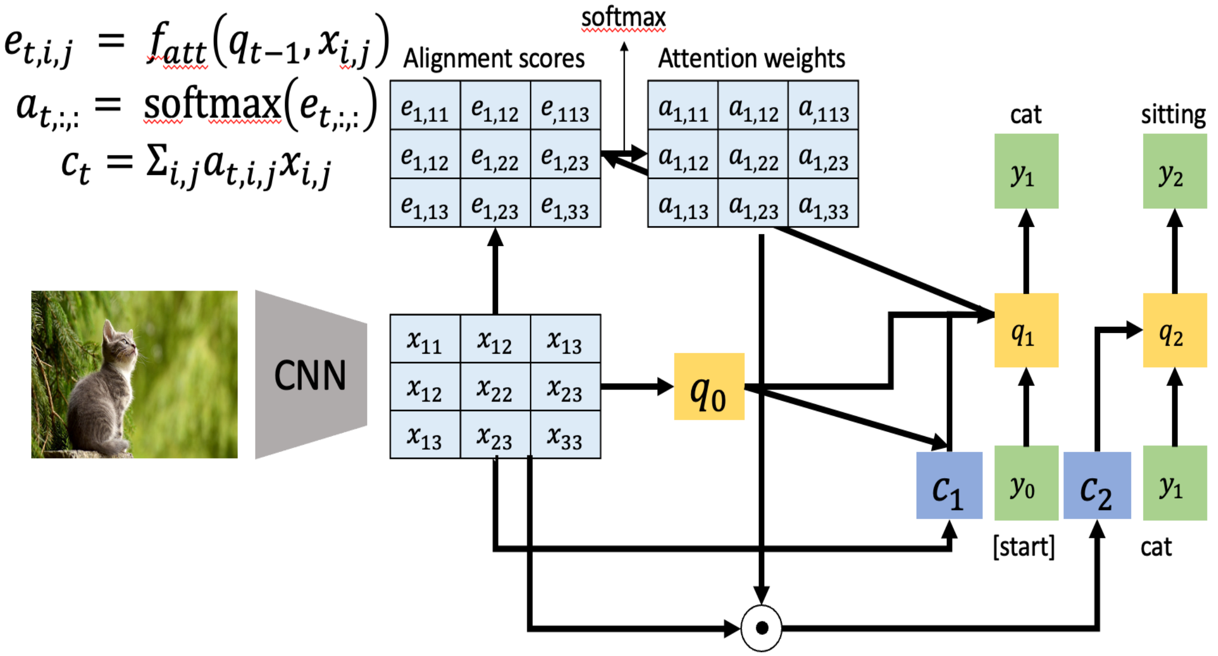
( ",:,: = softmax(" ",:,:)

0" = Σ$,% ( ",$,% 3$,%



# Towards Scalable and More Generic

$$e_{t,i,j} = f_{att}(q_{t-1}, x_{i,j})$$
$$a_{t,:,:} = \text{softmax}(e_{t,:,:})$$
$$c_t = \Sigma_{i,j} a_{t,i,j} x_{i,j}$$




**Inputs:**

**Query vector:** Q (Shape: $N_Q \times D_Q$)

**Input vectors:** $X$ (Shape: $N_X \times D_Q$)

**Key matrix:** $W_K (Shape: D_X \times D_Q)$

**Value matrix:** $W_V (Shape: D_X \times D_V)$

**Computation:**

**Key vectors:** $K = XW_) \ (Shape: N_" \times D_!)$

**Value Vectors:** $V = XW_* \ (Shape: N_" \times D_*)$

**Similarities:** $E = \frac{!\,)\,!}{\sqrt{\$\,"}} \ (Shape: N_! \times N_"), E_{\%} = (Q_{\%} \cdot K)/\sqrt{D_(}$

**Attention weights:** $A = \text{softmax}(E, \text{dim} = 1)$ (Shape: $N_! \times N_"$)

**Output vectors:** $Y = AV \ (Shape: N_! \times D_") Y_{\%} = \Sigma A_{\%} V$

Changes:

- Use scaled dot product for similarity
- Multiple query vectors
- Separate key and value

# Blank Slide

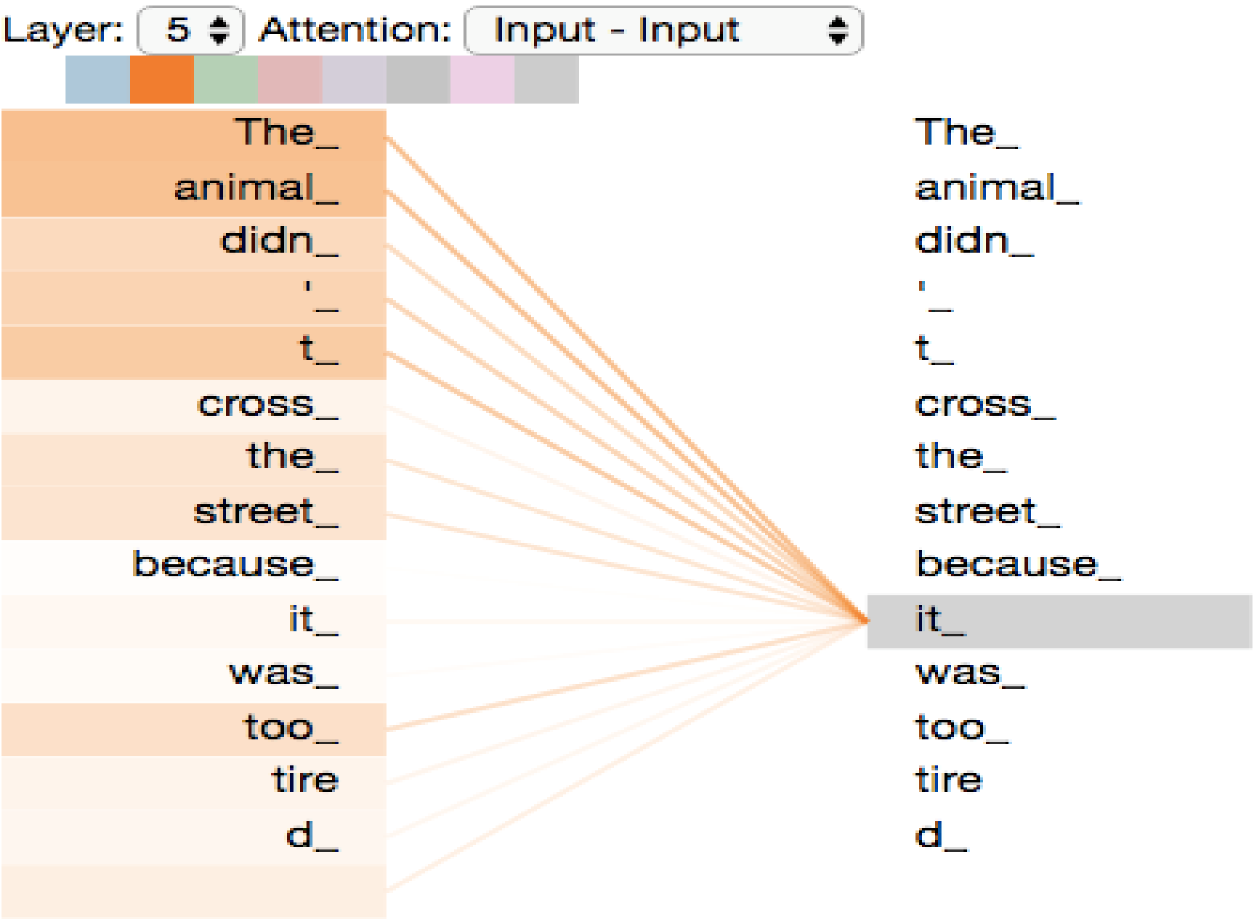
# Blank Slide

# Self Attention

# Self Attention

- Self attention learns the relationship between elements in a sequence.
  - say between words in a sentence

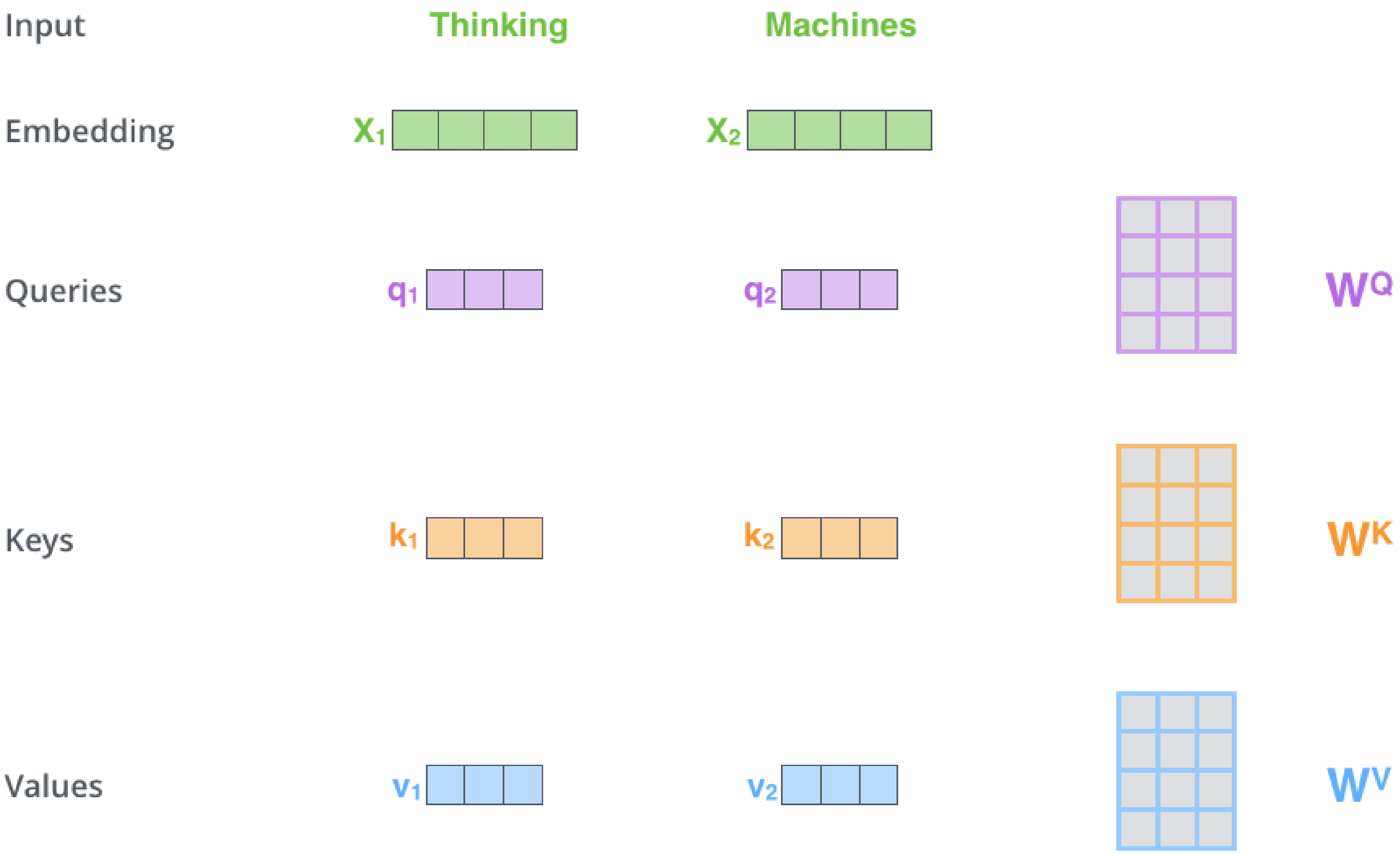
# Self Attention




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.*

# Blank Slide

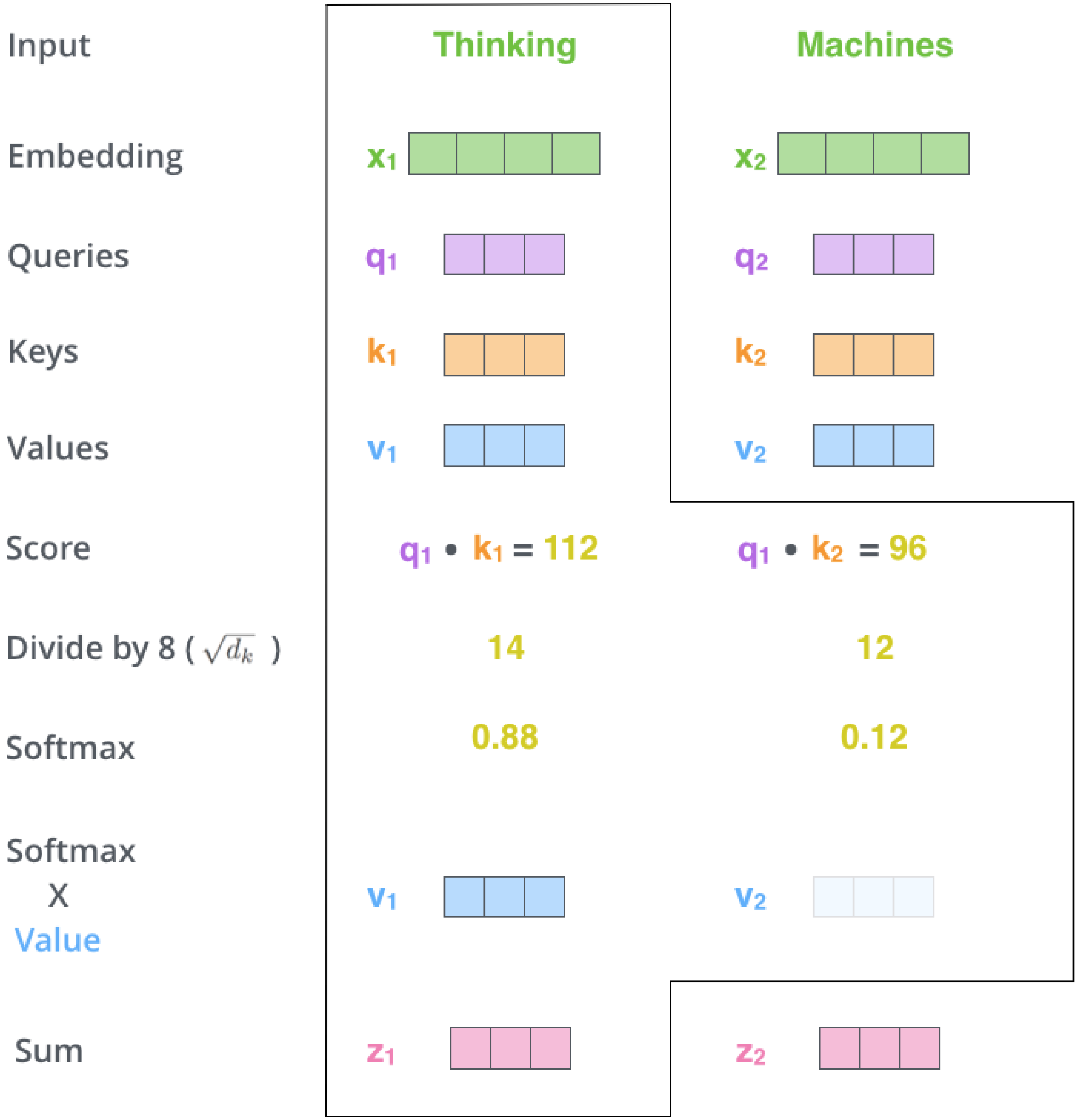
# Self Attention: Step 1



# Self Attention: Step 2

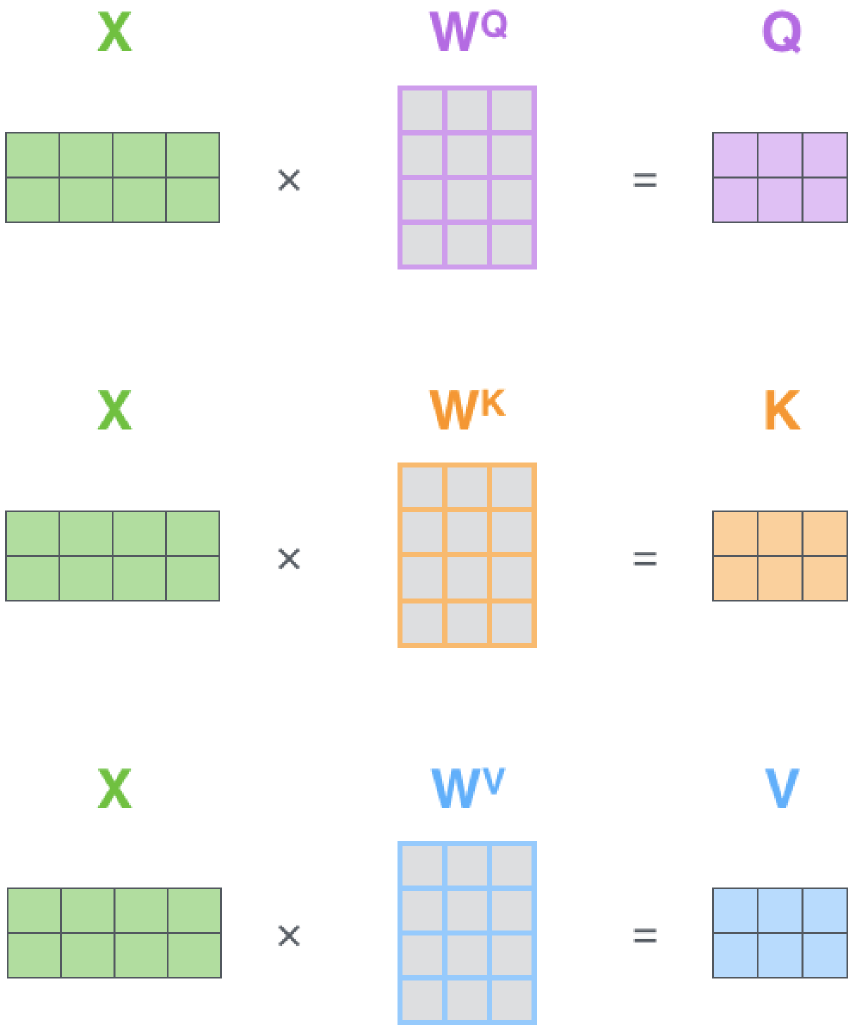
| | Thinking | Machines |
|---|---|---|
| Input | Thinking | Machines |
| Embedding | $x_1$ | $x_2$ |
| Queries | $q_1$ | $q_2$ |
| Keys | $k_1$ | $k_2$ |
| Values | $v_1$ | $v_2$ |
| Score | $q_1 \bullet k_1 = 112$ | $q_1 \bullet k_2 = 96$ |
| Divide by 8 ( $\sqrt{d_k}$ ) | 14 | 12 |
| Softmax | 0.88 | 0.12 |

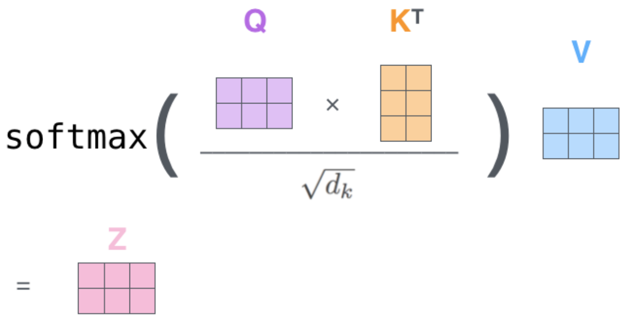
# Self Attention: Step 3



# Self Attention: Matrix View

# Matrix View



# Self Attention

One query per input vector

**Inputs:**

**Input vectors:** $X$ (Shape: $N_X \times D_Q$)

**Key matrix:** $W_K (Shape: D_X \times D_Q)$

**Value matrix:** $W_V (Shape:\ D_X \times D_V)$

**Query matrix:** $\mathbf{W}_\mathrm{Q}$ (Shape: $\mathrm{D_Q} \times D_Q$)

**Computation:**

**Query Vectors** $Q = XW_!$

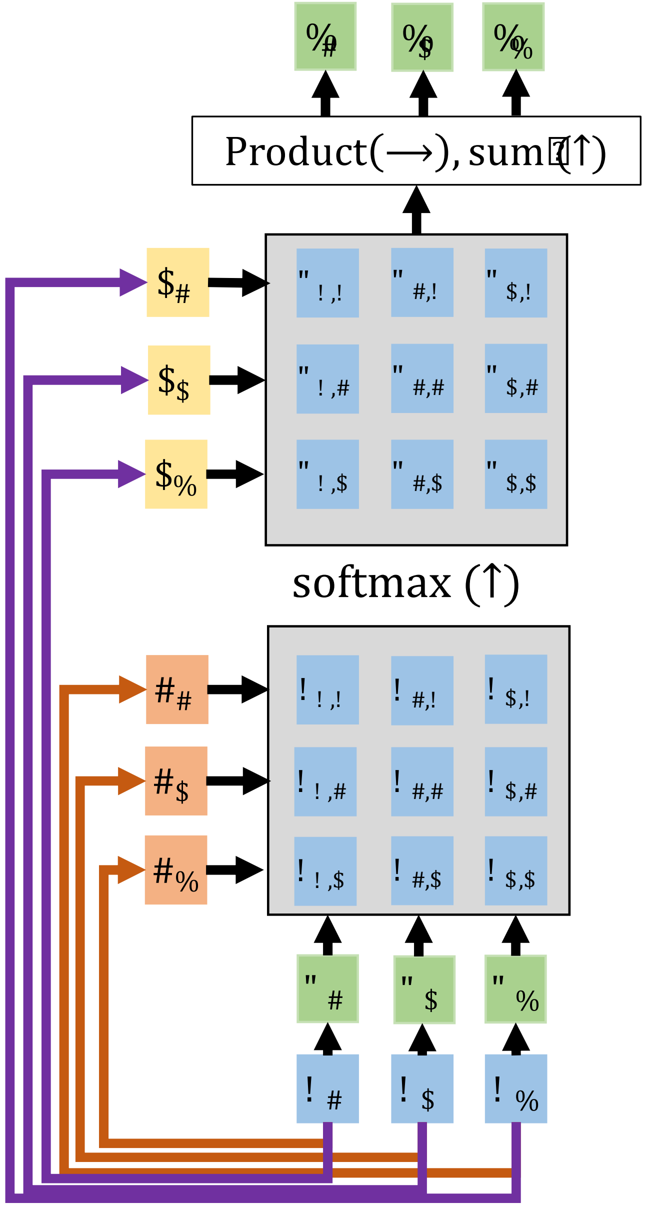
**Key vectors:** $K = XW_)$ $(Shape:\ N_" \times D_!)$

**Value Vectors:** $V = XW_*$ $(Shape:\ N_" \times D_*)$

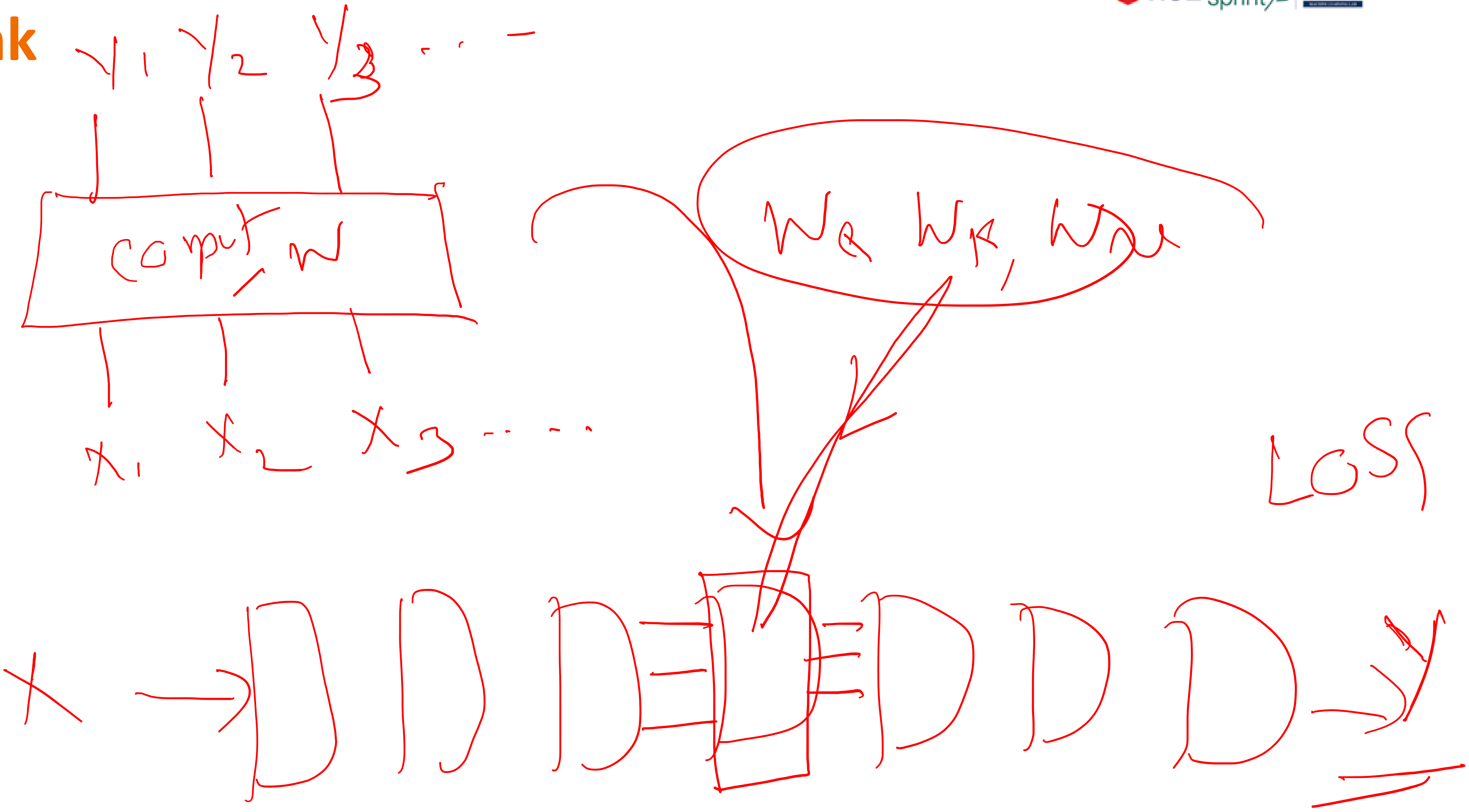
**Similarities:** $E = \frac{!\ "\ !}{\sqrt{\$\ "}}$ $(Shape:\ N_" \times N_")E_{\%} = (Q_{\%} \cdot K_{'})/\sqrt{D_(}$

**Attention weights:** $A = softmax(E, \mathrm{dim} = 1)$ $(Shape:\ N_" \times N_")$

**Output vectors:** $Y = AV$ $(Shape:\ N_" \times D_*)Y_{\%} = \Sigma_{'} A_{\%} V_{'}$

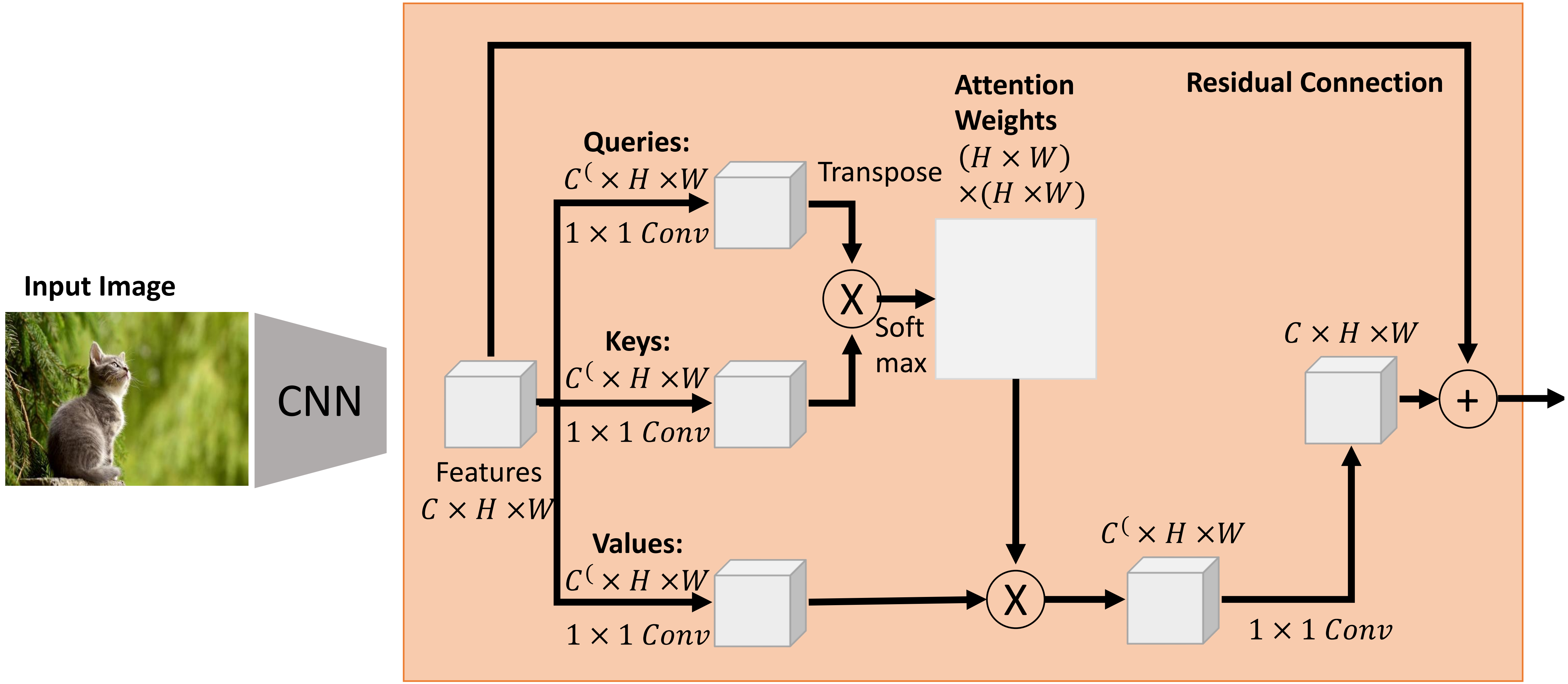


# Blank



# Self Attention Layer

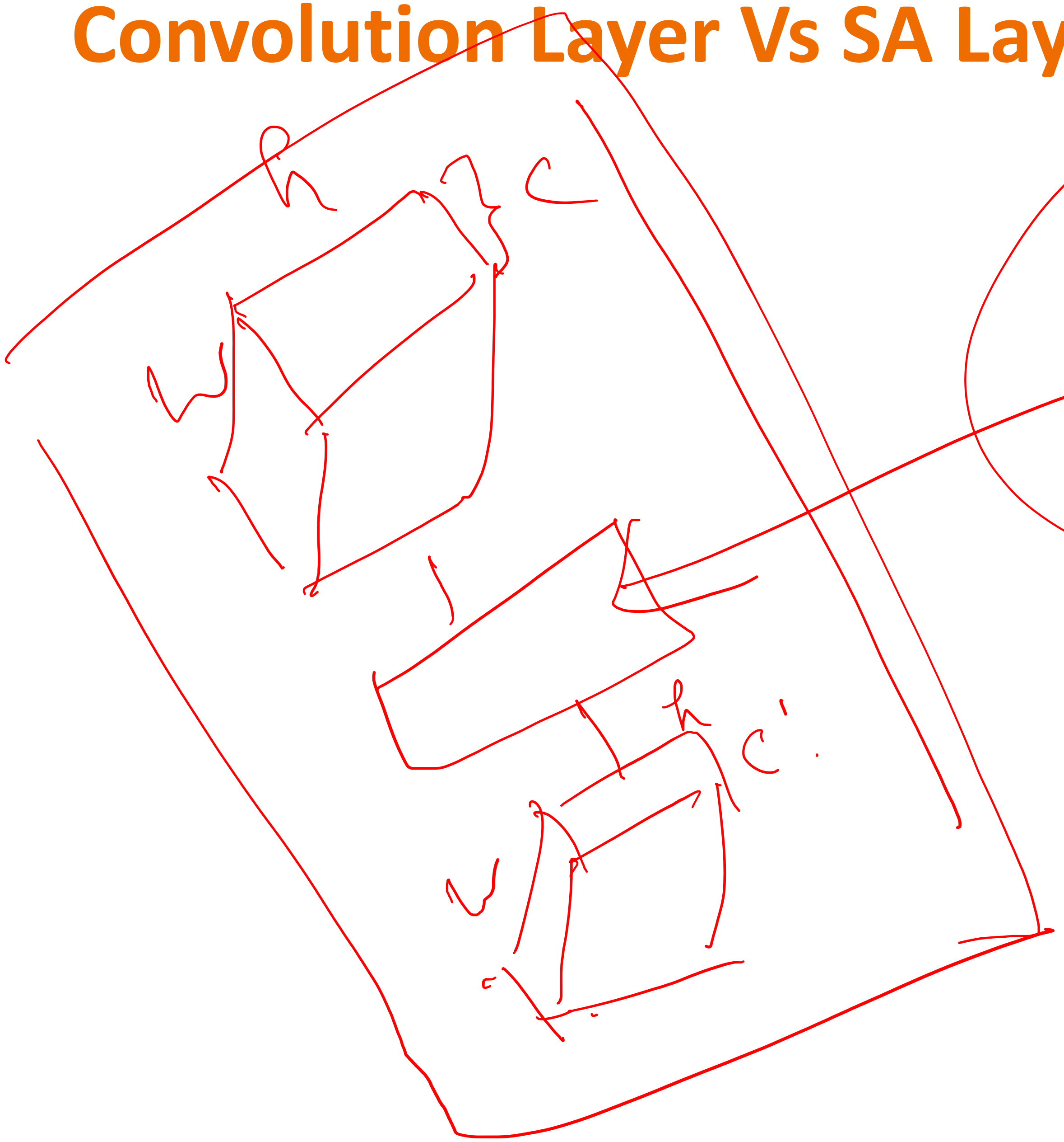
# CNN with Attention




Self Attention Module

# Blank Slide

# Convolution Layer Vs SA Layer





# 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

# Where we want to go?

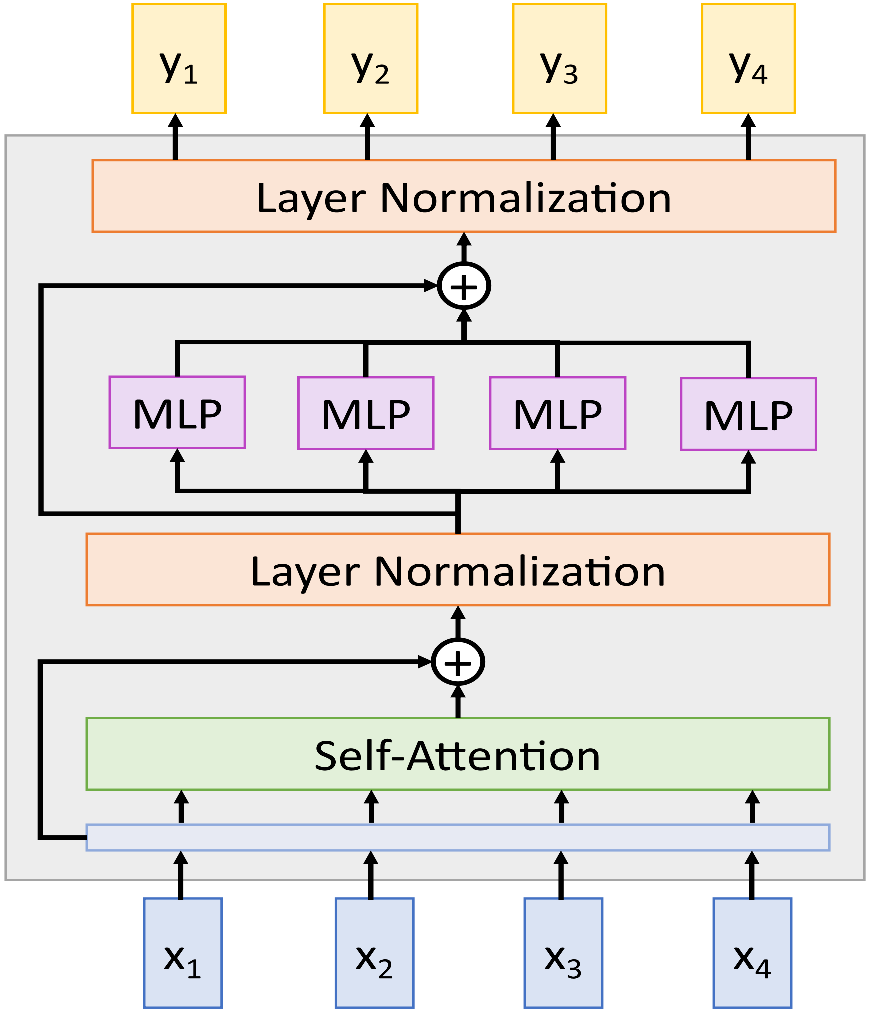
## The Transformer

**<u>Transformer Block</u>:**
**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

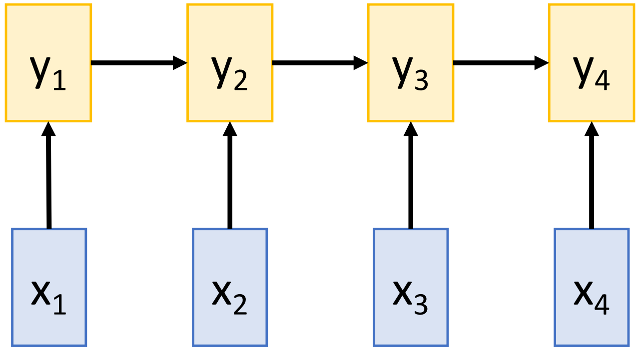
Highly scalable, highly parallelizable




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

# Conceptual Comparison

## Recurrent Neural Network

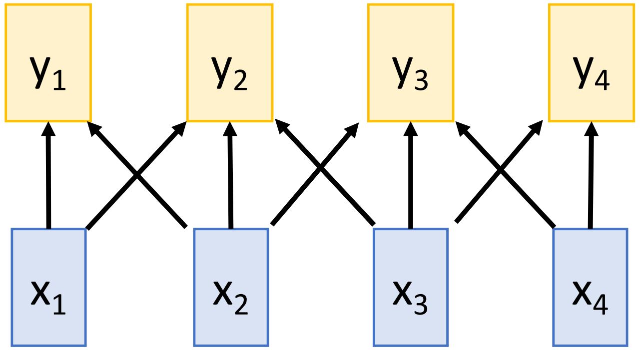



Works on **Ordered Sequences**

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

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

## 1D Convolution

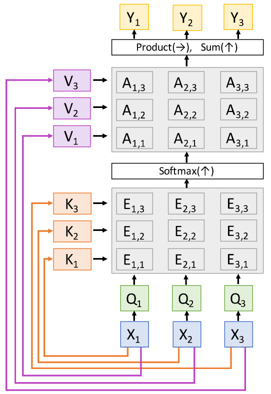



Works on **Multidimensional Grids**

(-) 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



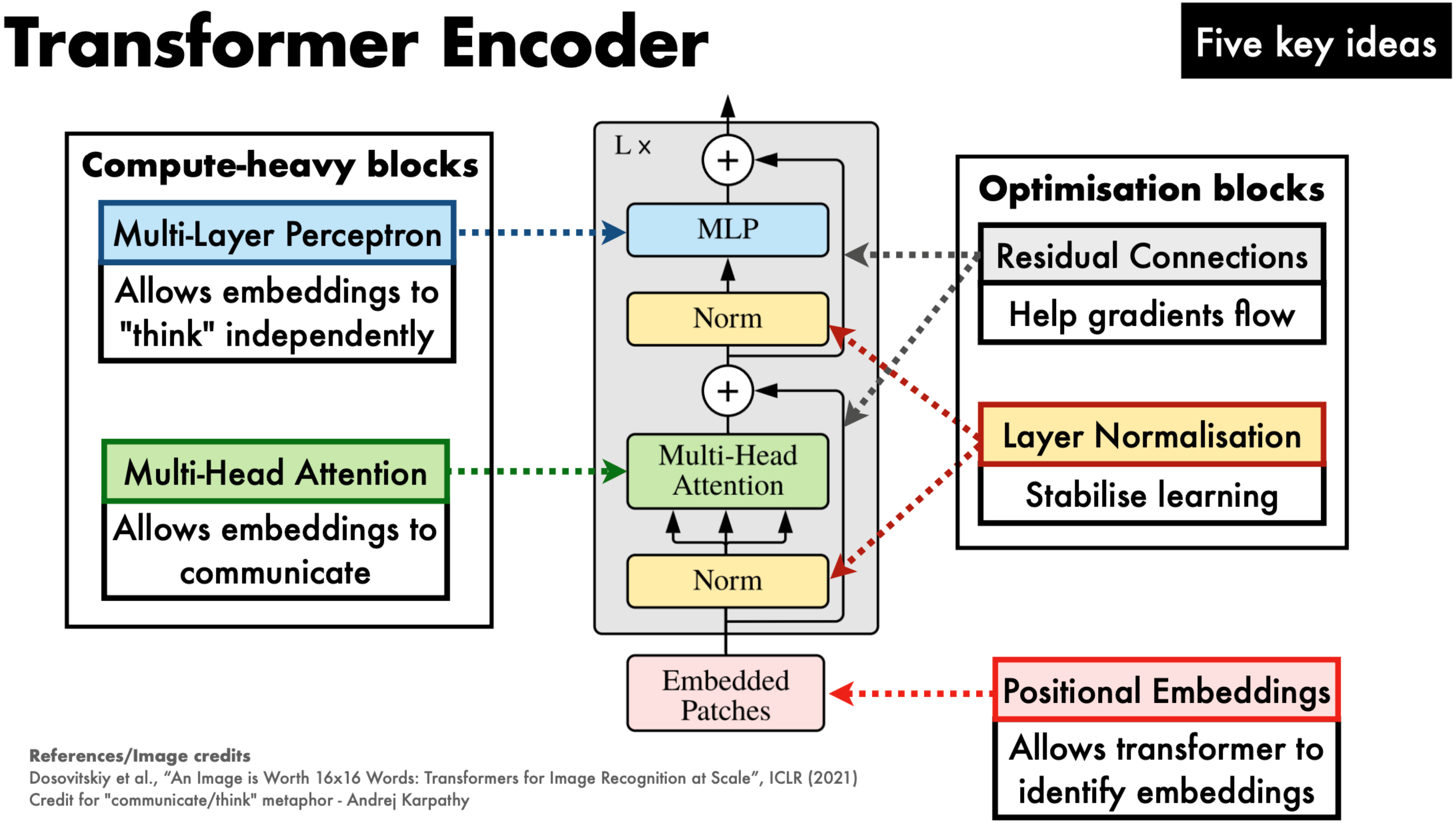

Works on **Sets of Vectors**

(-) 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

# 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

# Thanks!!

**Questions?**