PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Attention & Transformers

R. Q. Feitosa

LVC COMPUTER VISION LAB

Overview

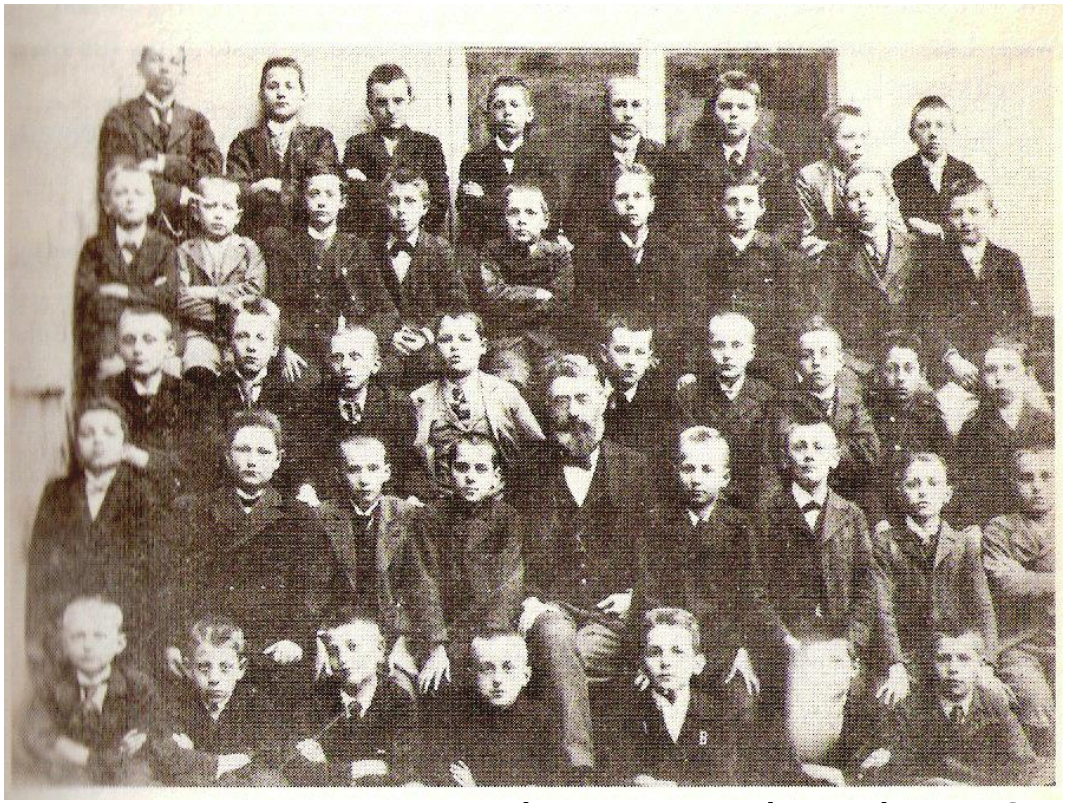
1. Motivation
2. Seq2Seq
3. Attention
4. Transformer
5. Applications in Computer Vision
6. Final Comments

Attention

*"In psychology, **attention** is the cognitive process of selectively concentrating on one or a few things while ignoring others."*

*"A neural network is considered to be an effort to mimic human brain actions in a simplified manner. **Attention** Mechanism is also* an attempt to implement the same action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks."

How eyes move over the picture



How many people are in the photo?

Who is the teacher?

Do you recognize some student?

Overview

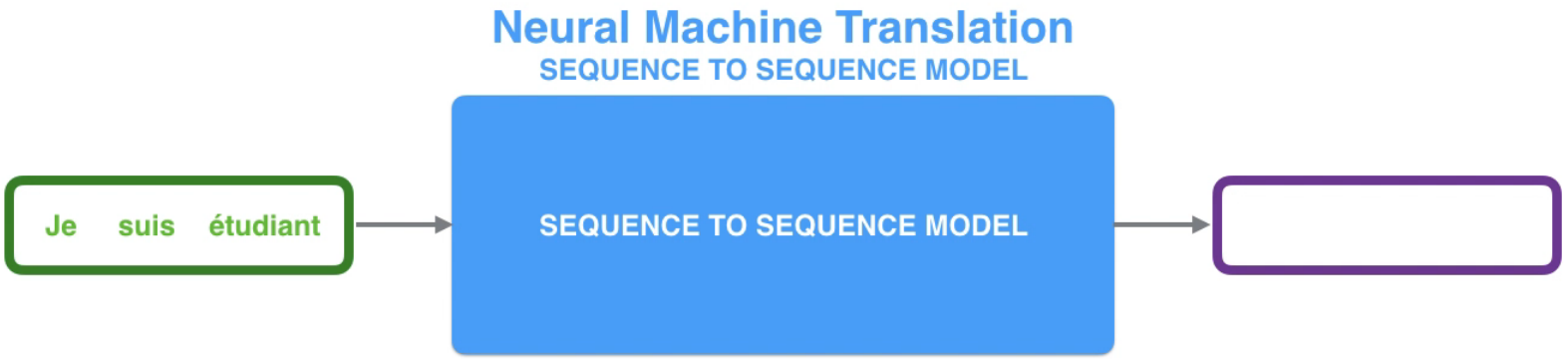
1. Motivation
2. Seq2Seq
3. Attention
4. Transformer
5. Applications in Computer Vision
6. Final Comments

Seq2Seq

A sequence-to-sequence model is a model that takes a sequence of items (words, letters, features of an images…etc) and outputs another sequence of items.

Seq2Seq: machine translation

In neural machine translation (NMT), a sequence is a series of words, processed one after another.

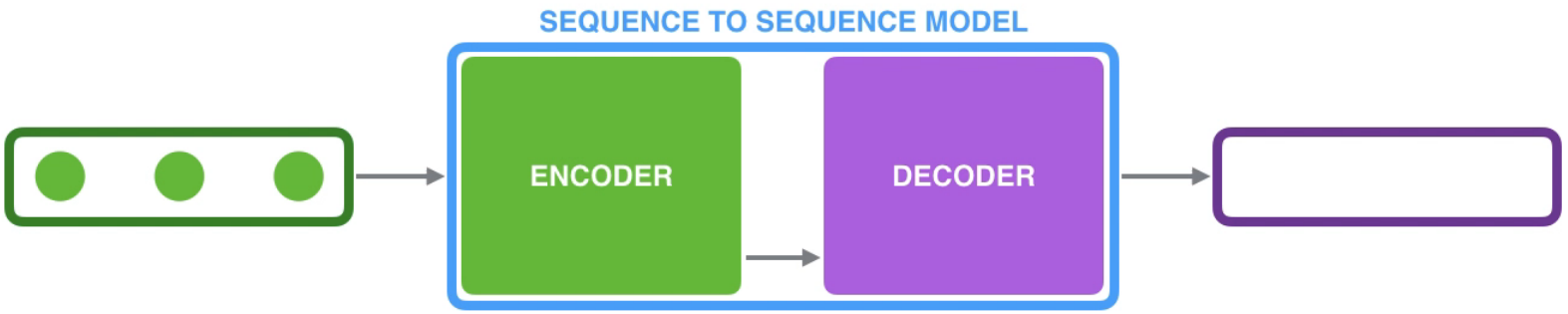


Encoder-Decoder

The model comprises an encoder and a decoder.

The encoder processes all items of the input sequence and computes a context vector, which represents the whole sequence.

The decoder takes the context and produces the output sequence.

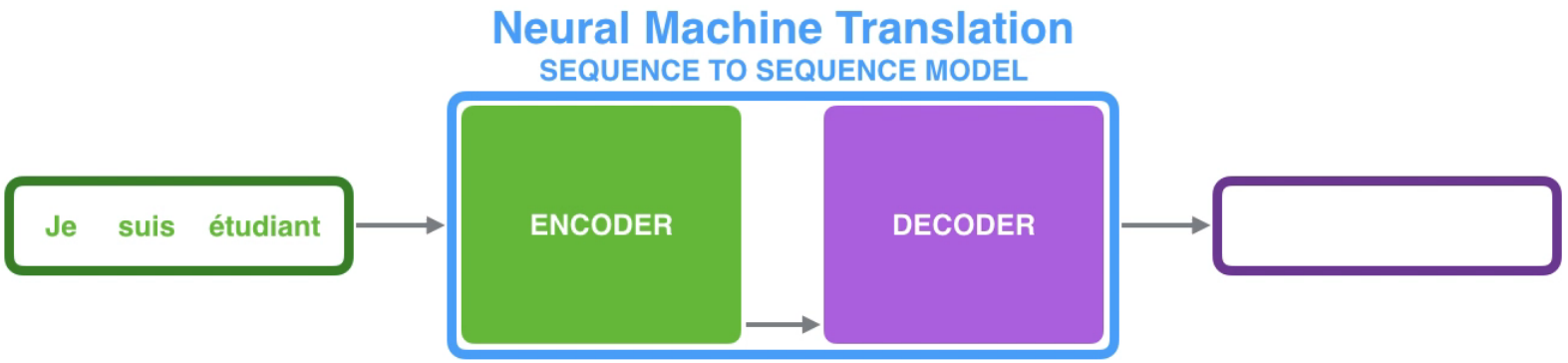


Encoder-Decoder: NMT(1)

The model comprises an encoder and a decoder.

The encoder processes all words of the input sentence and computes a context vector, which represents the whole sentence.

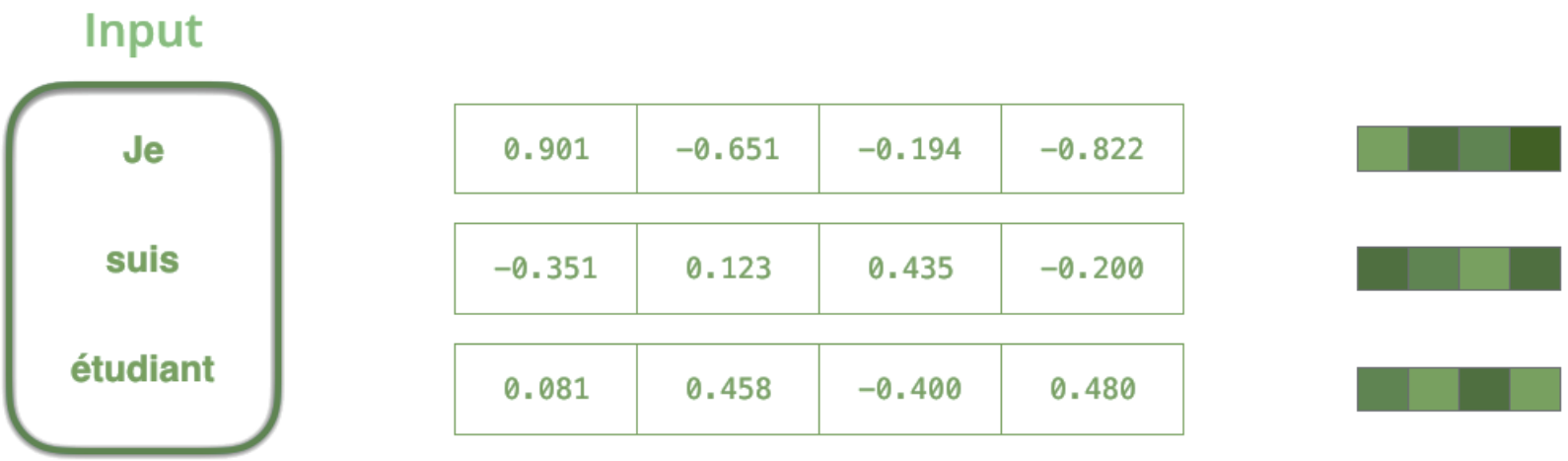
The decoder takes the context and produces the words of the translated sentence.



Word embeddings

We have to turn words into vectors for the RNNs to process them.

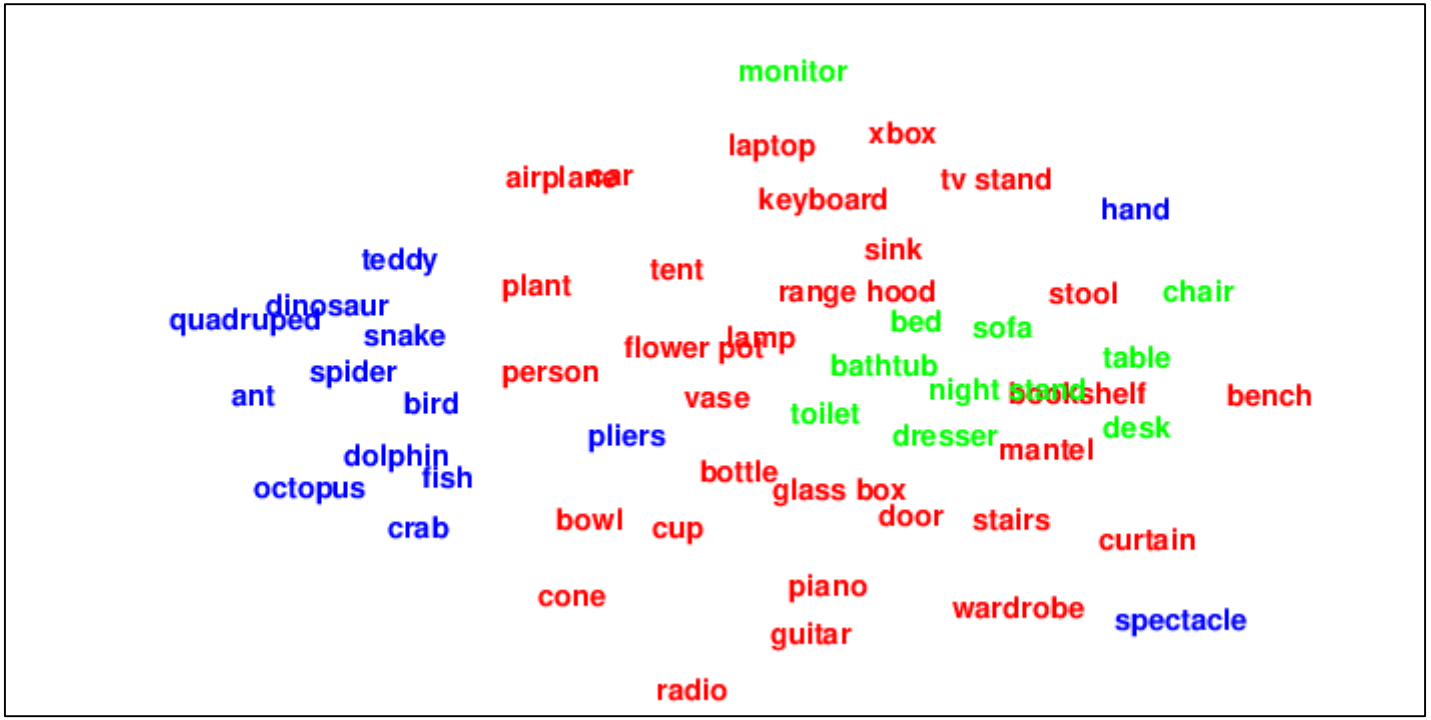
Algorithms called “word embedding” do it.



Word embeddings

Words with similar meanings are “close” in the embedding space.

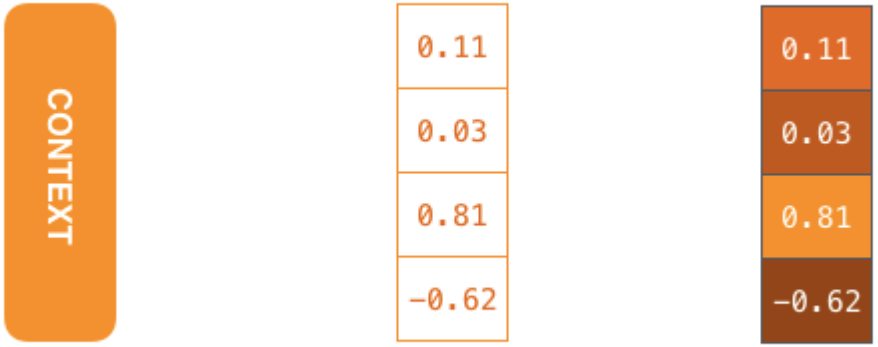
It admits thinks like “king – man + woman = queen”



Encoder-Decoder as RNNs

Let's assume that the encoder and the decoder are RNNs.

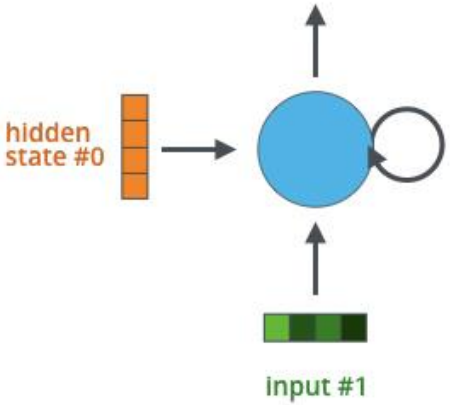
The size of context vector is basically the number of hidden units of the encoder RNN.



Recalling how RNNs work

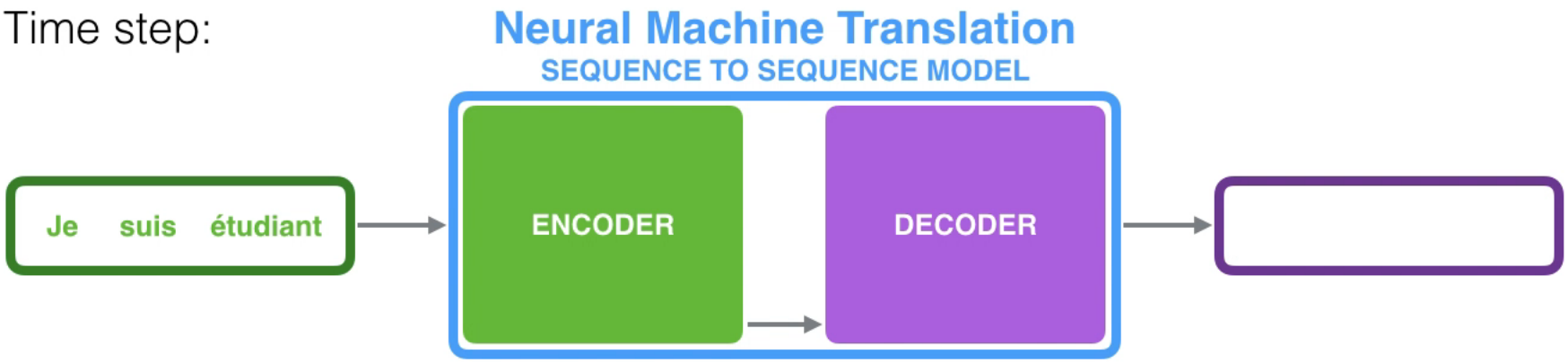
Recurrent Neural Network

Time step #1:
An RNN takes two input vectors:



Encoder-Decoder: NMT (2)

At each step the encoder of the decoder processes its inputs and delivers an output.



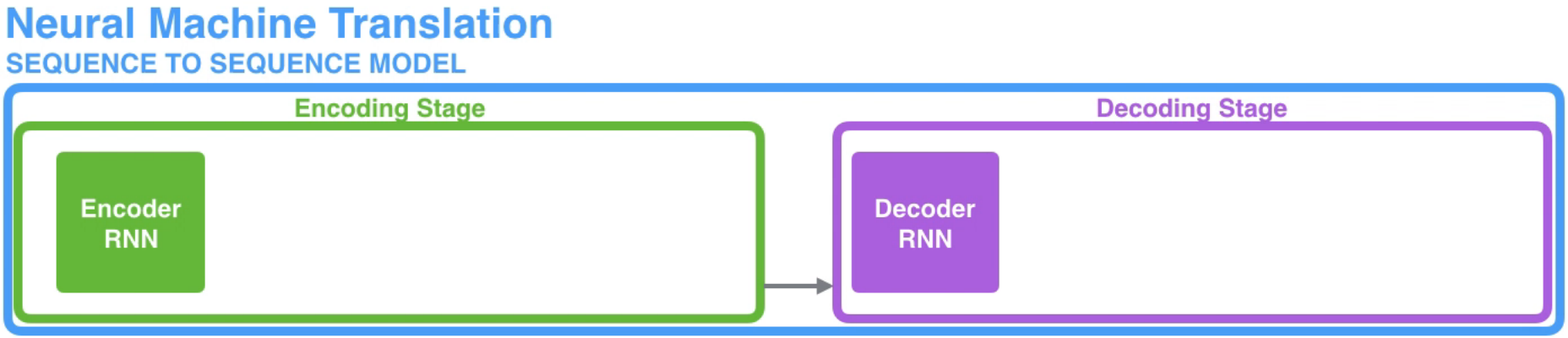
The last hidden state of the encoder is the context.

From then on the decoder outputs a word at each step.

(decoder hidden states are not shown, but exist)

Encoder-Decoder: NMT (2)

At each step the encoder or the decoder processes its inputs and delivers an output.



The last hidden state of the encoder is the context.

From then on the decoder outputs a word at each step.

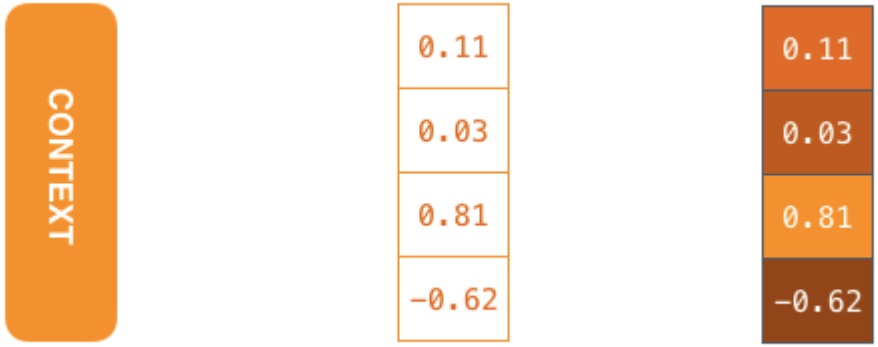
(decoder hidden states are not shown, but exist)

Overview

1. Motivation
2. Seq2Seq
3. Attention
4. Transformer
5. Applications in Computer Vision
6. Final Comments

Problem with Seq2Seq

The fixed size of context limits the length of sentences that can be handled.

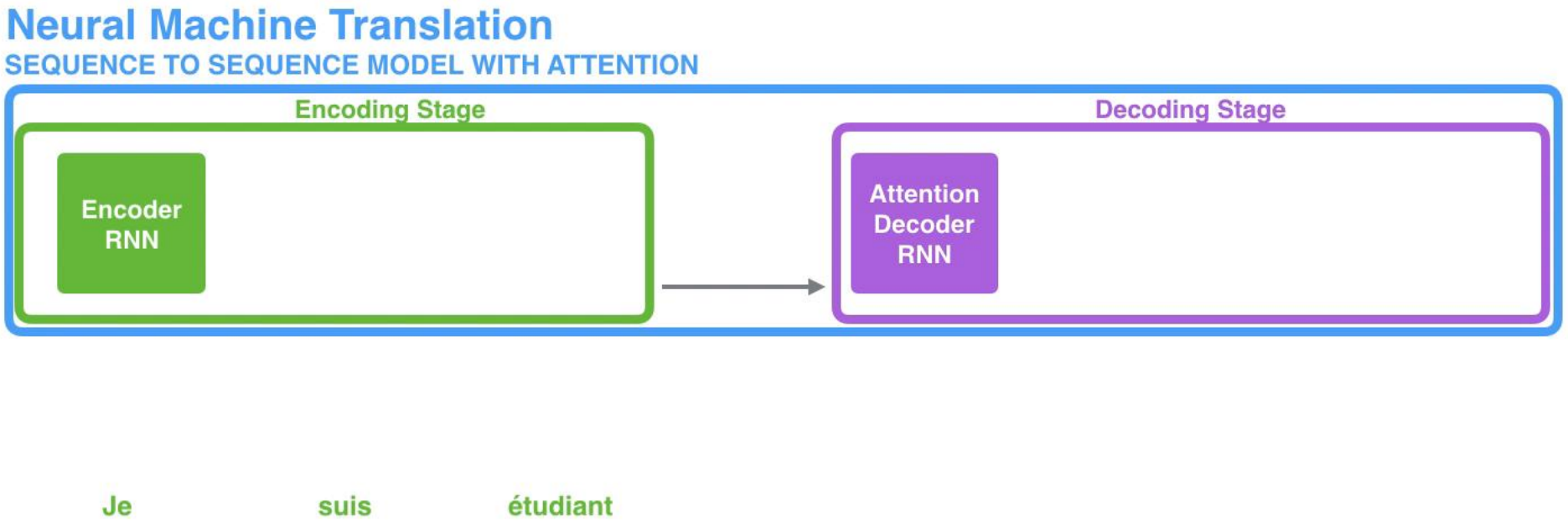


Bahdanau et al., 2014 and Luong et al., 2015 introduced and refined a technique called “Attention”, to overcome this problem.

Attention allows the model to focus on the relevant parts of the input sequence as needed.

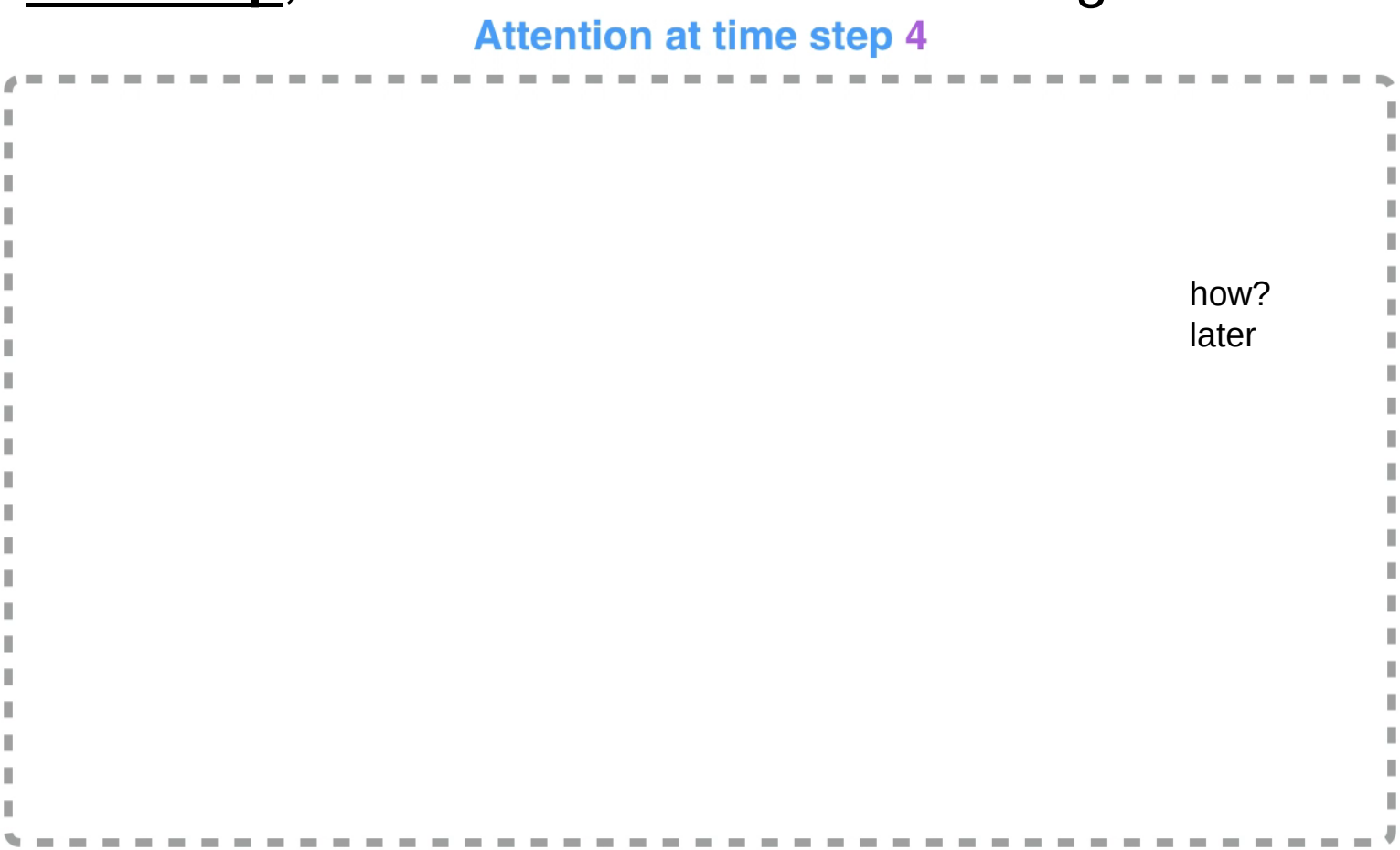
Differences between Attention and Seq2Seq

A. Instead of passing just the last hidden state of the encoding stage, the encoder passes *all* the hidden states to the decoder:



Differences between Attention and Seq2Seq

B. before producing its output, in order to focus on the parts of the input that are relevant **to this decoding time step**, the decoder does the following:

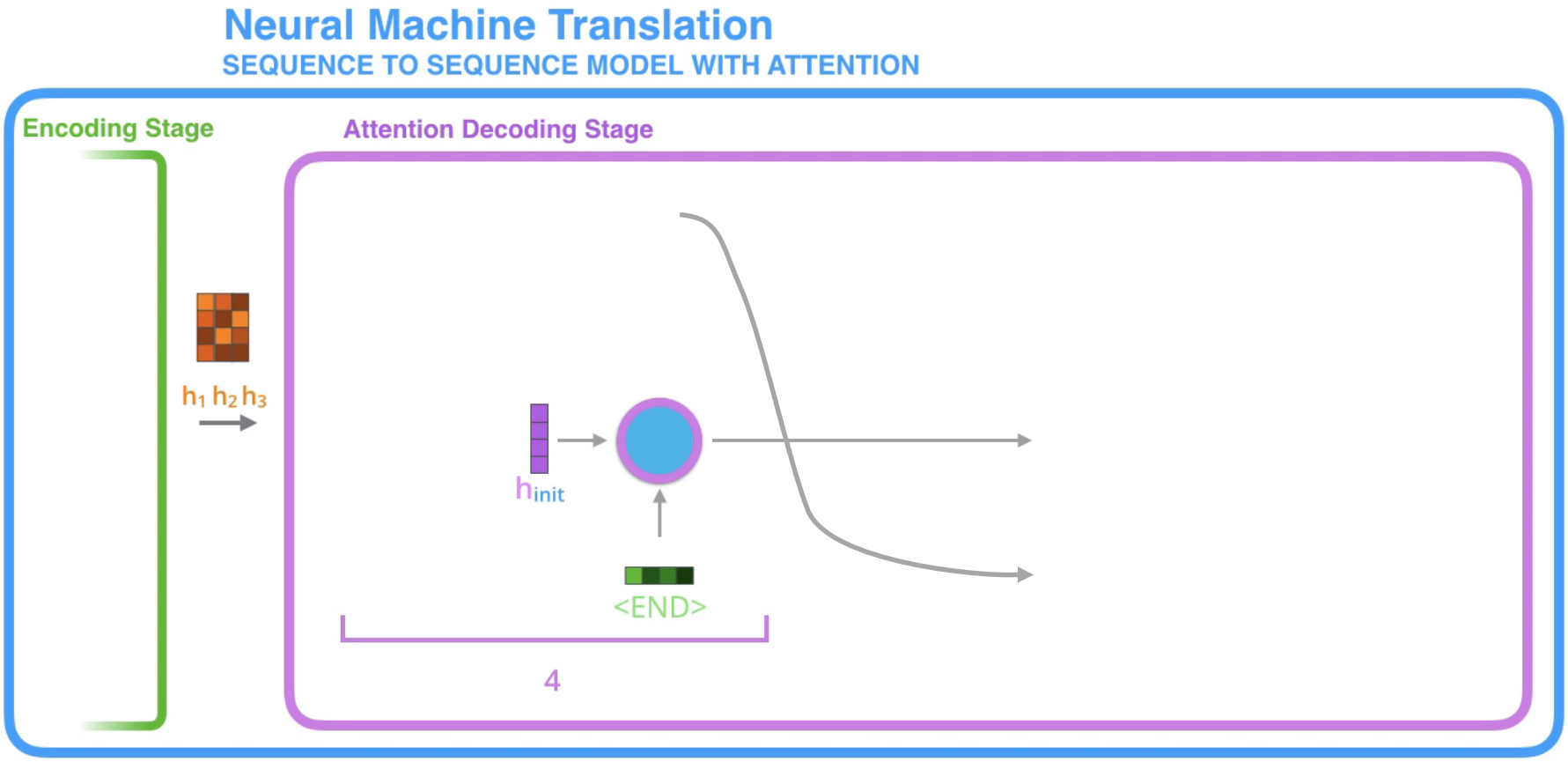


Decoder steps

1. The attention decoder RNN takes in the embedding of the <END> token, and an initial decoder hidden state.
2. The RNN processes its inputs, producing an output and a new hidden state vector (h4). The output is discarded.
3. **Attention Step**: We use the encoder hidden states and the h4 vector to calculate a context vector (C4) for this time step.
4. We concatenate h4 and C4 into one vector.
5. We pass this vector through a feedforward neural network (one trained jointly with the model).
6. The output of the feedforward neural networks indicates the output word of this time step.
7. Repeat from step 2 for the next time steps.

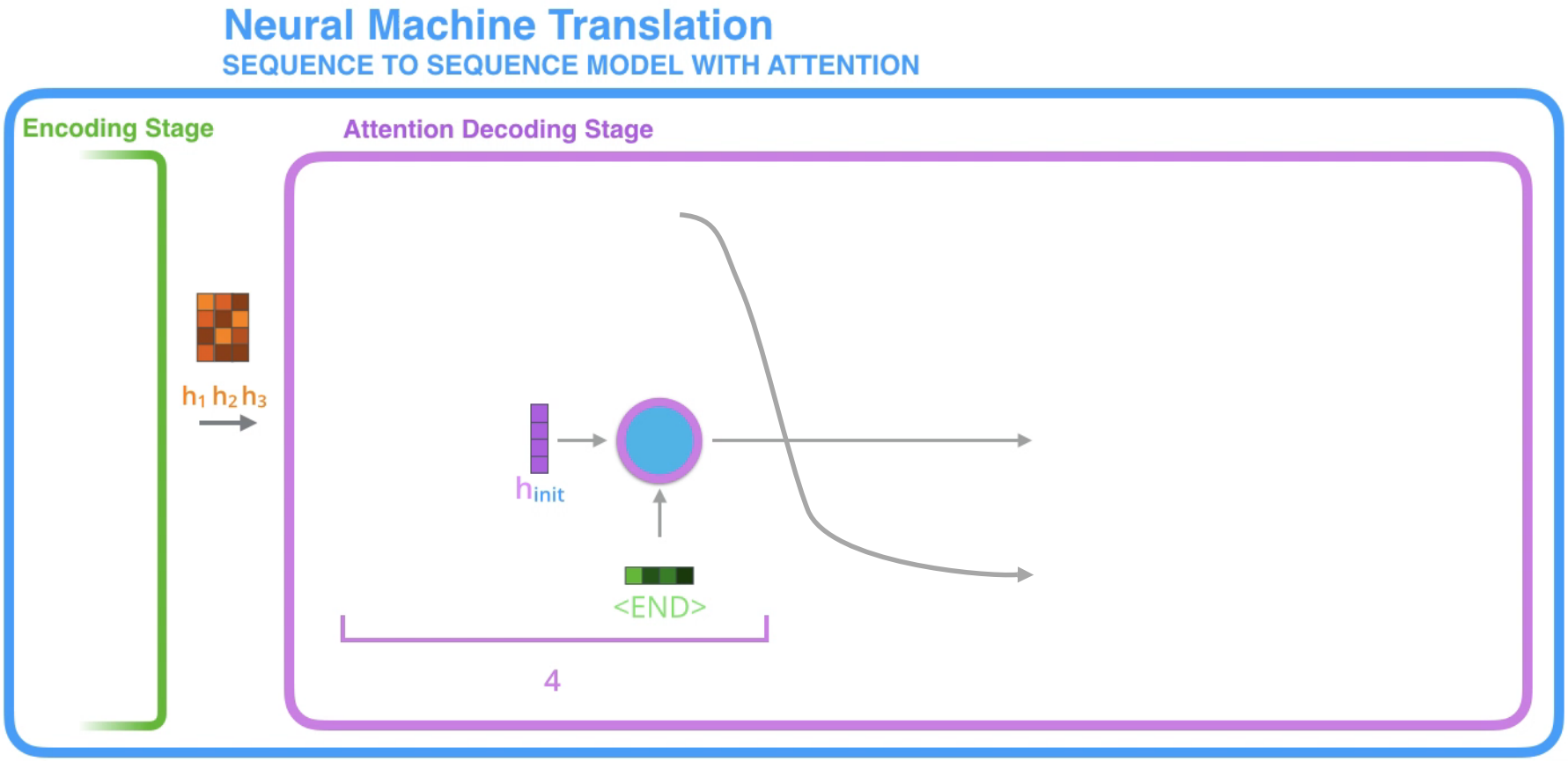
Decoder steps

7. Repeat from step 2 for the next time steps.



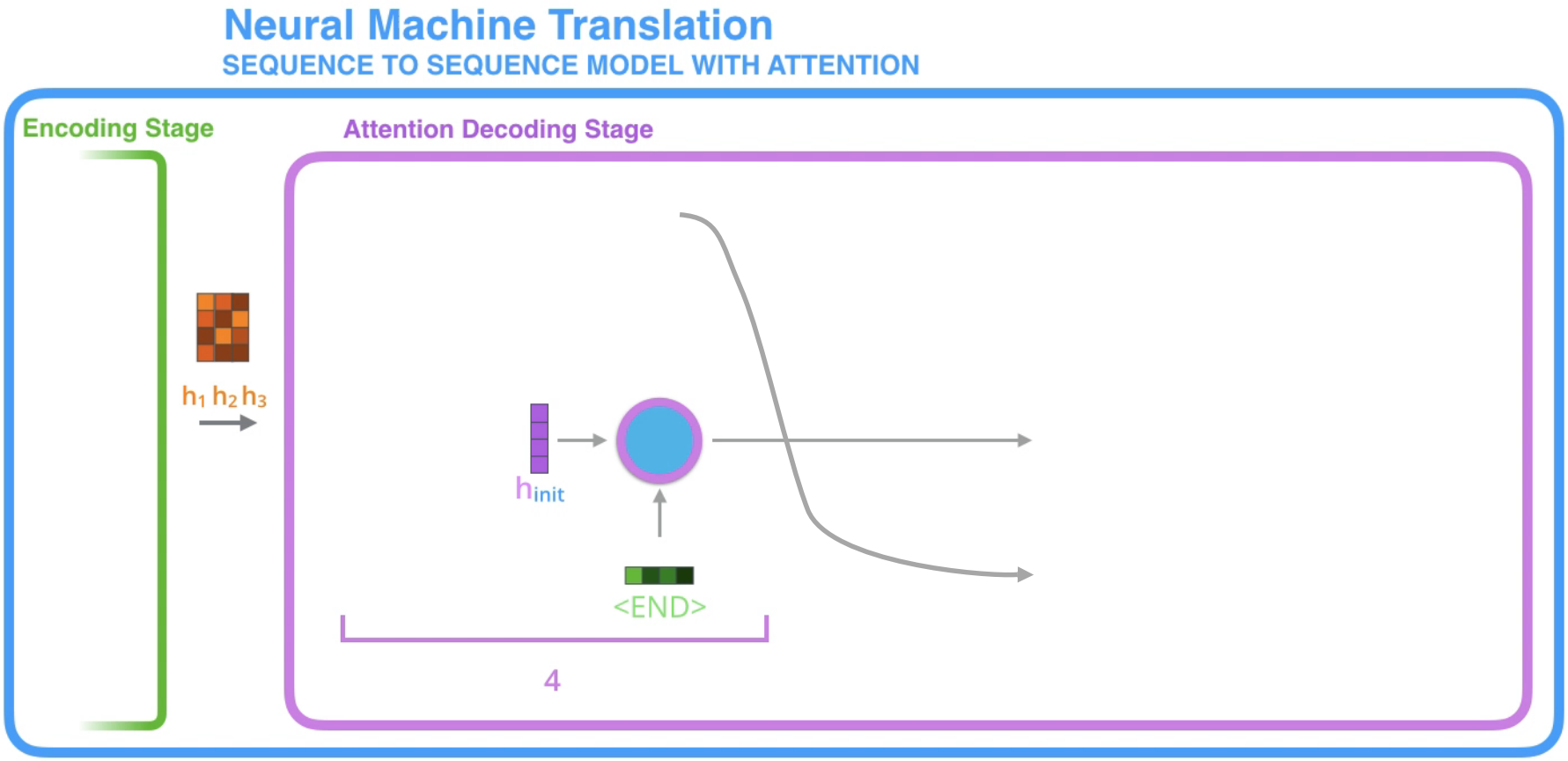
Decoder steps

1. The attention decoder RNN takes in the embedding of the <END> token, and an initial decoder hidden state.



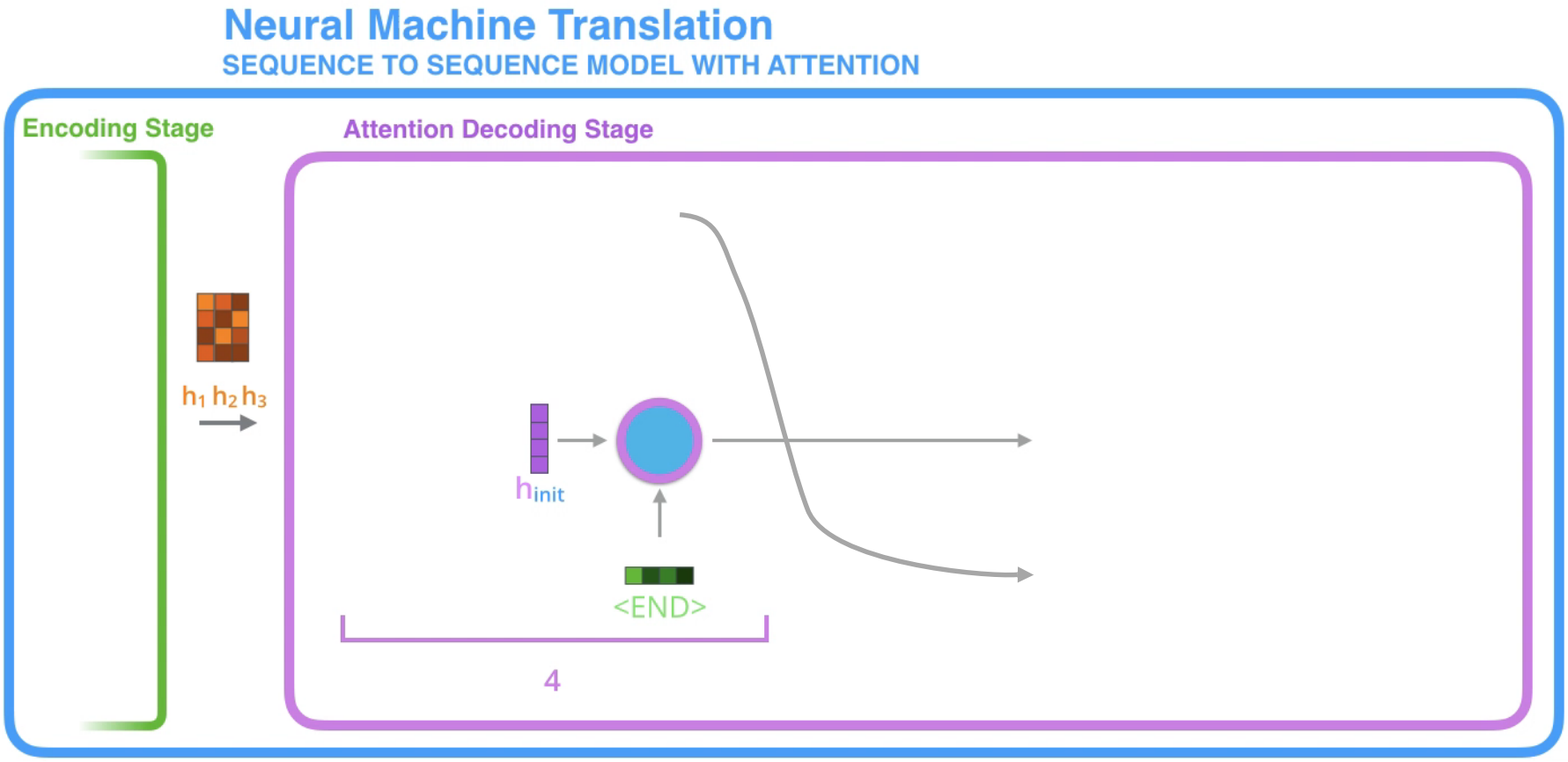
Decoder steps

2. The RNN processes its inputs, producing an output and a new hidden state vector (h4). The output is discarded.



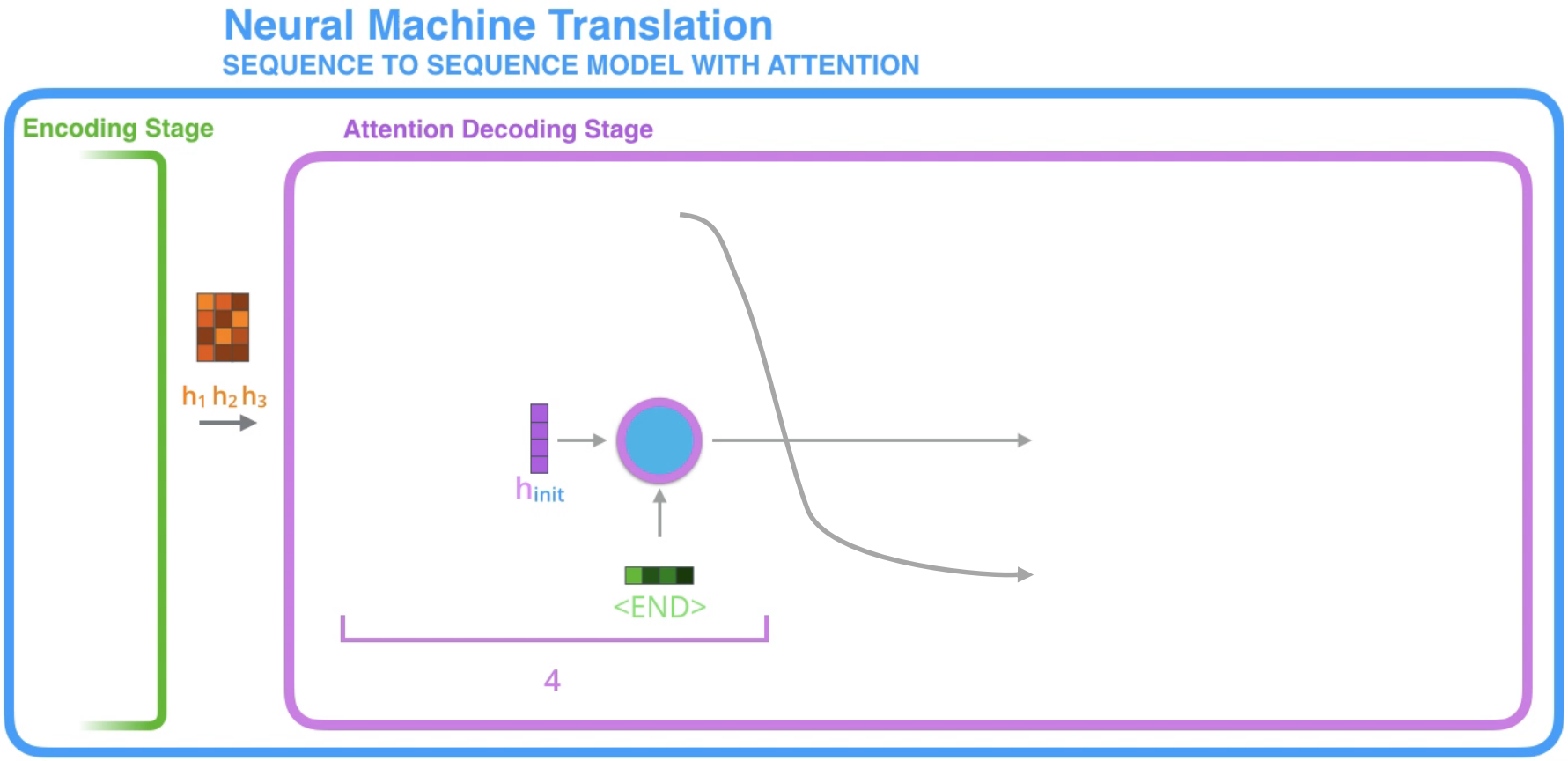
Decoder steps

3. Attention Step: We use the encoder hidden states and the h4 vector to calculate a context vector (C4) for this time step.



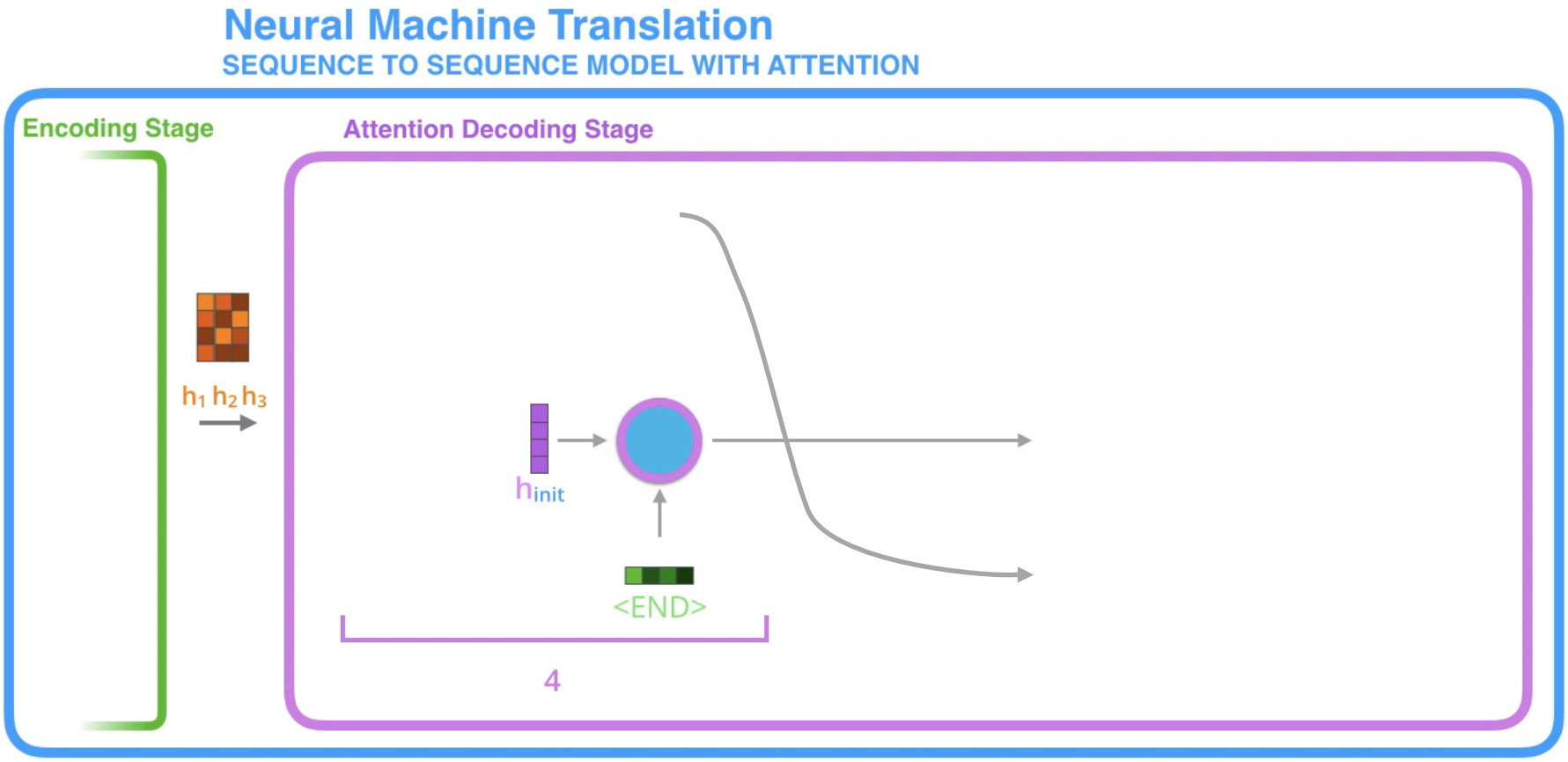
Decoder steps

4. We concatenate h4 and C4 into one vector.



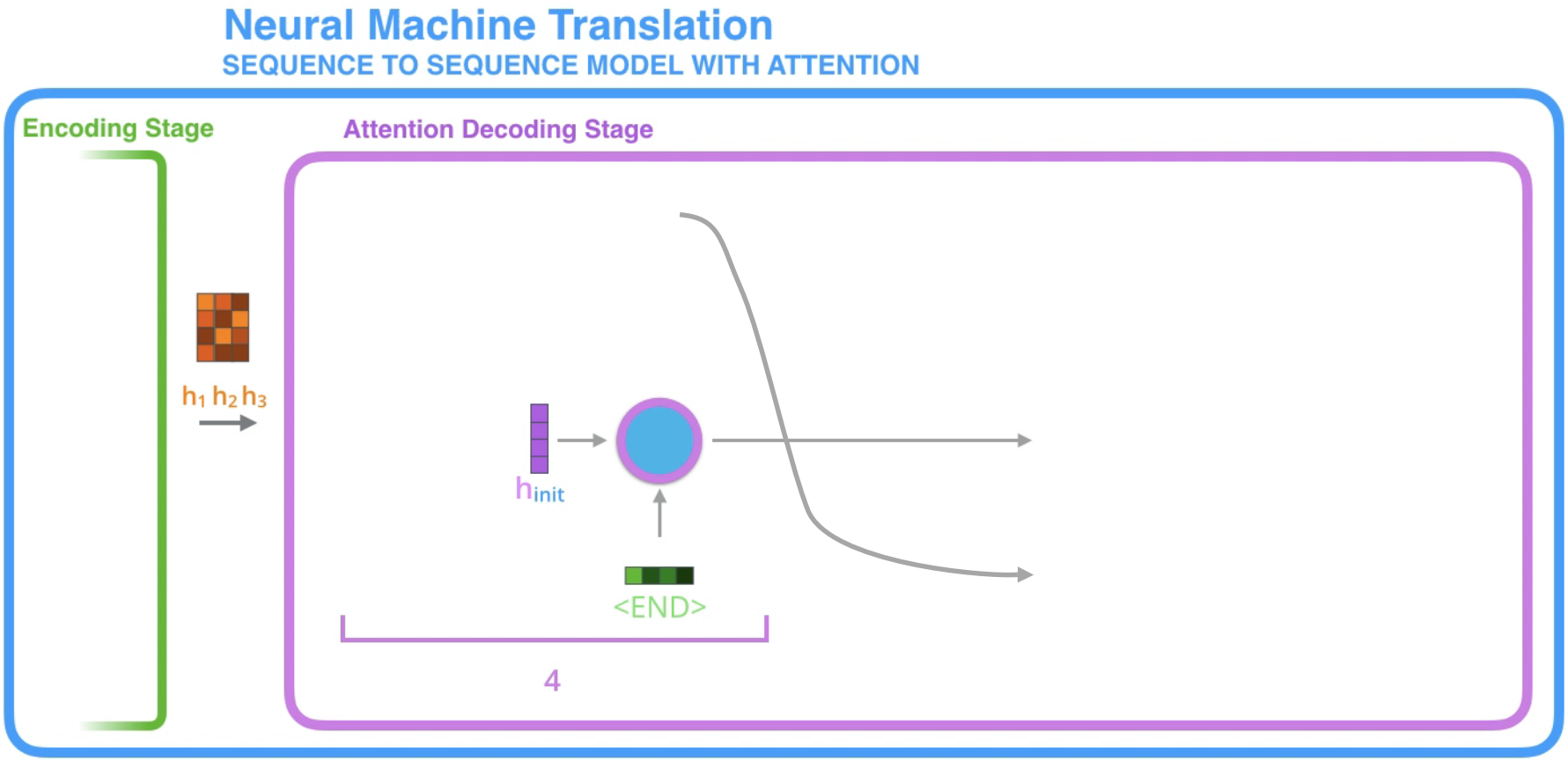
Decoder steps

5. We pass this vector through a feedforward neural network (one trained jointly with the model).



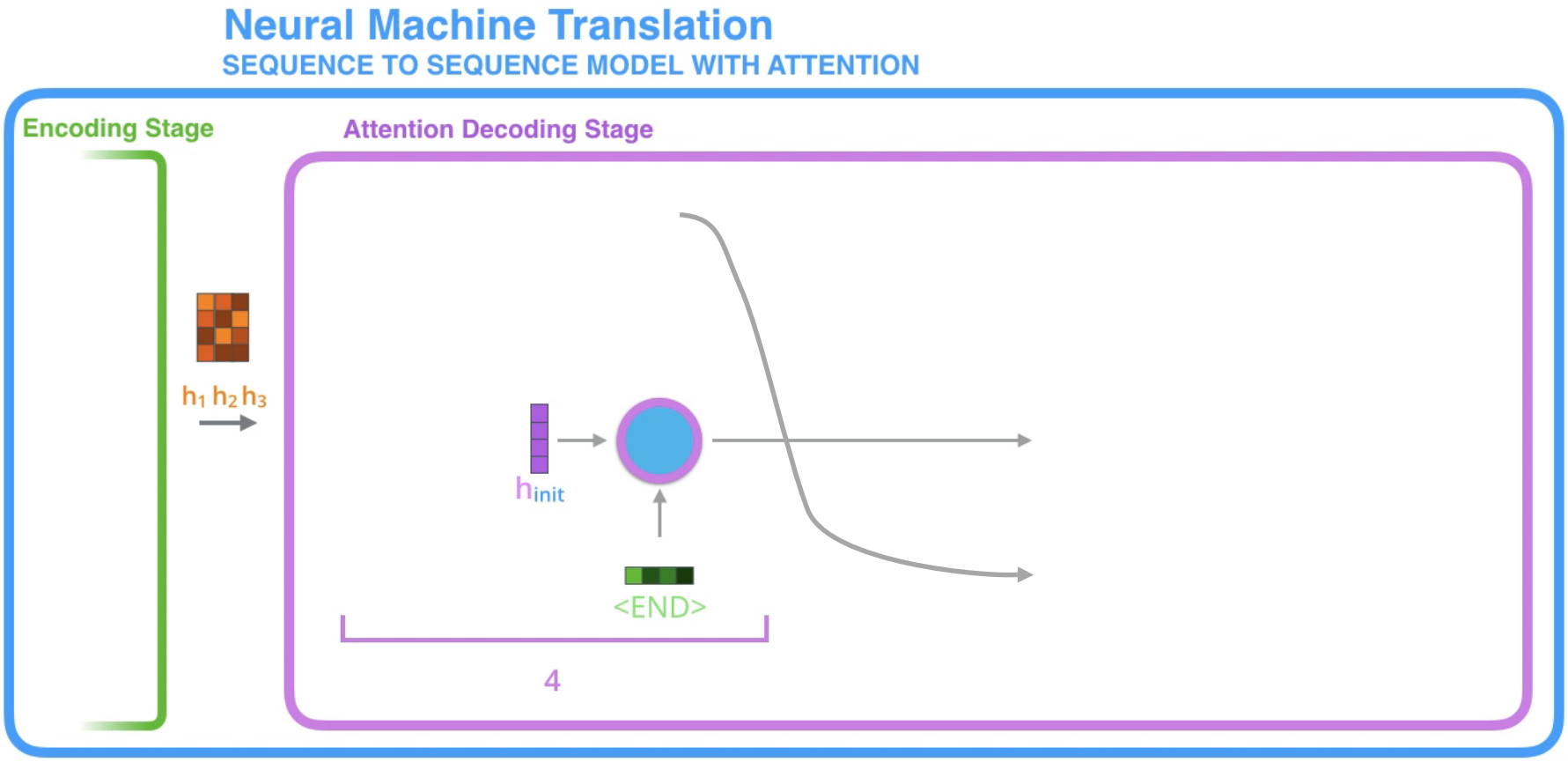
Decoder steps

6. The output of the feedforward neural networks indicates the output word of this time step.



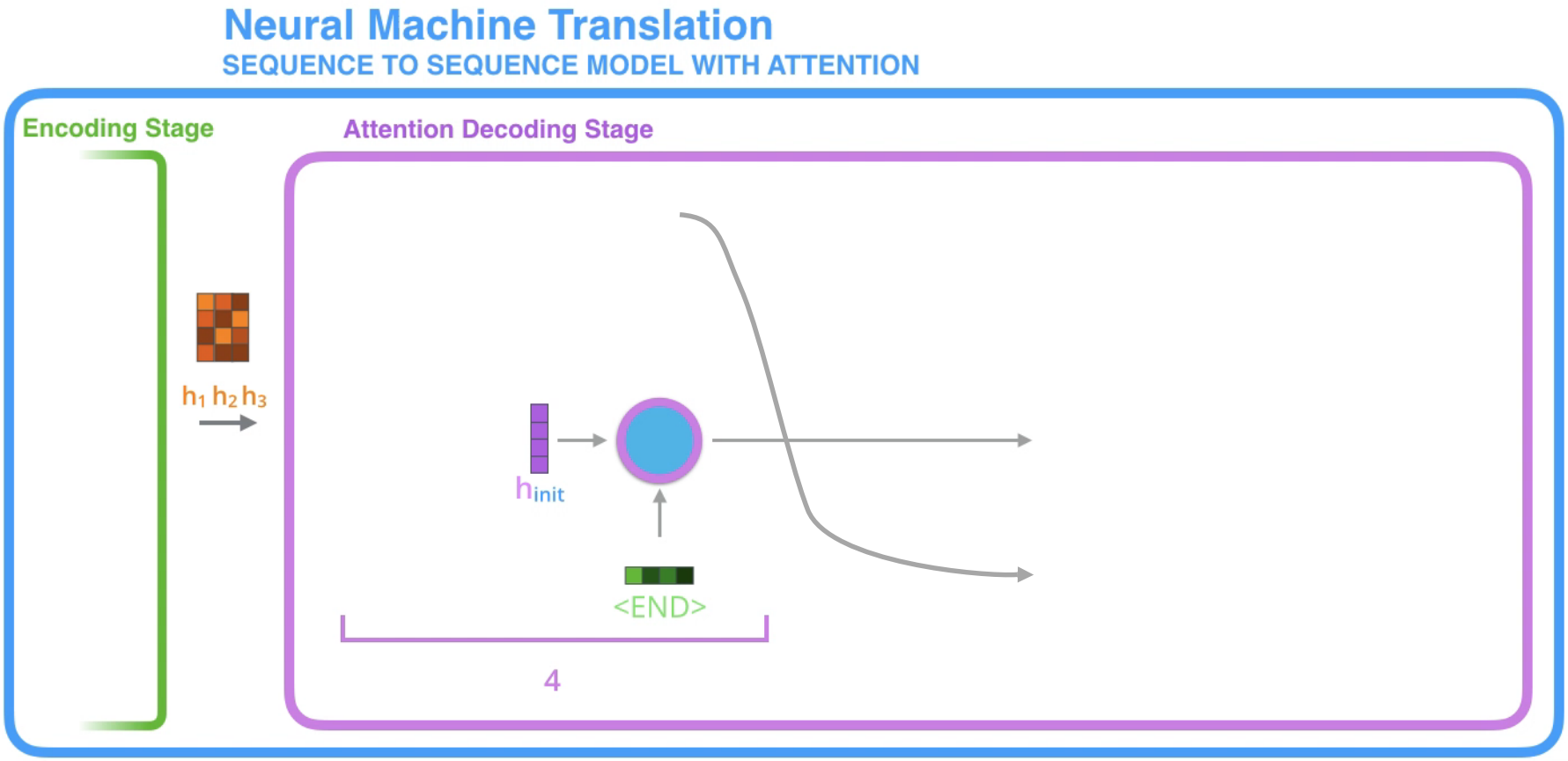
Decoder steps

7. Repeat from step 2 for the next time steps.



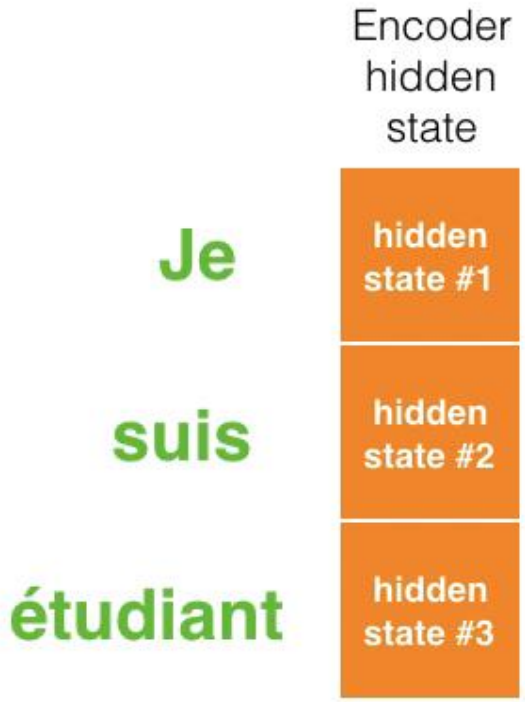
Decoder steps

1. The output of the feedforward neural networks indicates the output word of this time step.
2. Repeat for the next time steps



Decoder steps – another view

Note that the model isn't just mindless aligning the first word at the output with the first word from the input, and so on.



Attention example

You can see on the right how the model paid attention correctly when outputing "European Economic Area".

In French, the order of these words is reversed ("européenne économique zone") as compared to English.

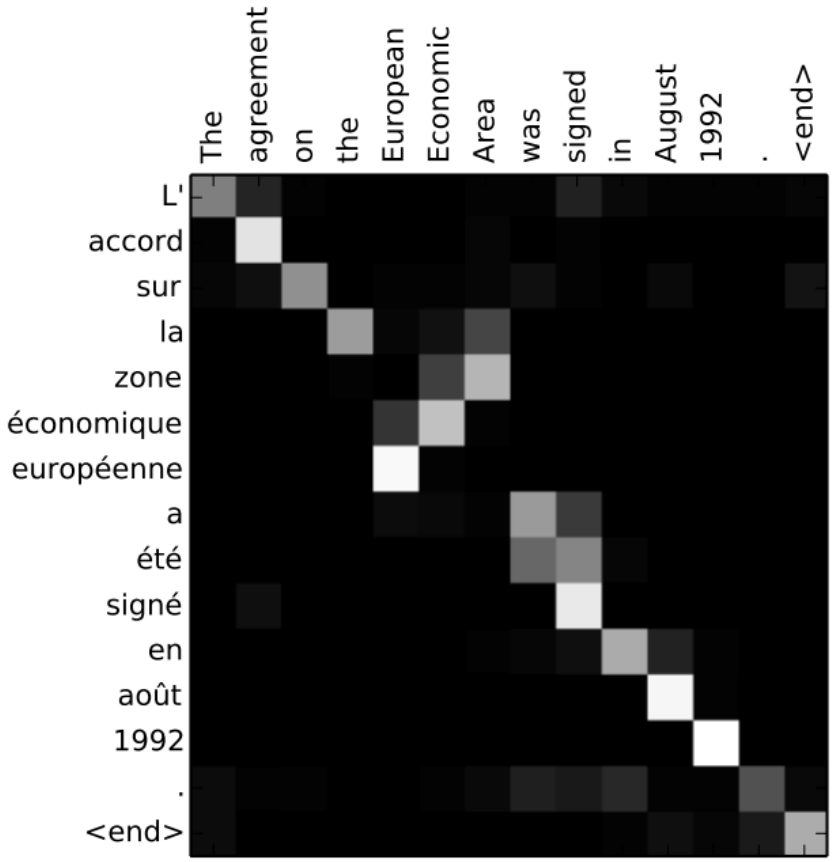


Figure source: Bahdanau et al. 2015

Overview

1. Motivation
2. Seq2Seq
3. Attention
4. Transformers
5. Applications in Computer Vision
6. Final Comments

Problems with RNNs

1. RNNs are slow to train (even with truncated BPTT).
2. Long sequences lead to vanishing/exploding gradients.
3. LSTM and GRU mitigates the problem above, partially.
4. LSTM/GRU are even slower to train (they are more complex).
5. RNNs need the result of previous states to start the computation of the current state ⇨ this sequential processing makes no proper use parallelism of today GPUs.

Attention Is All You Need

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.

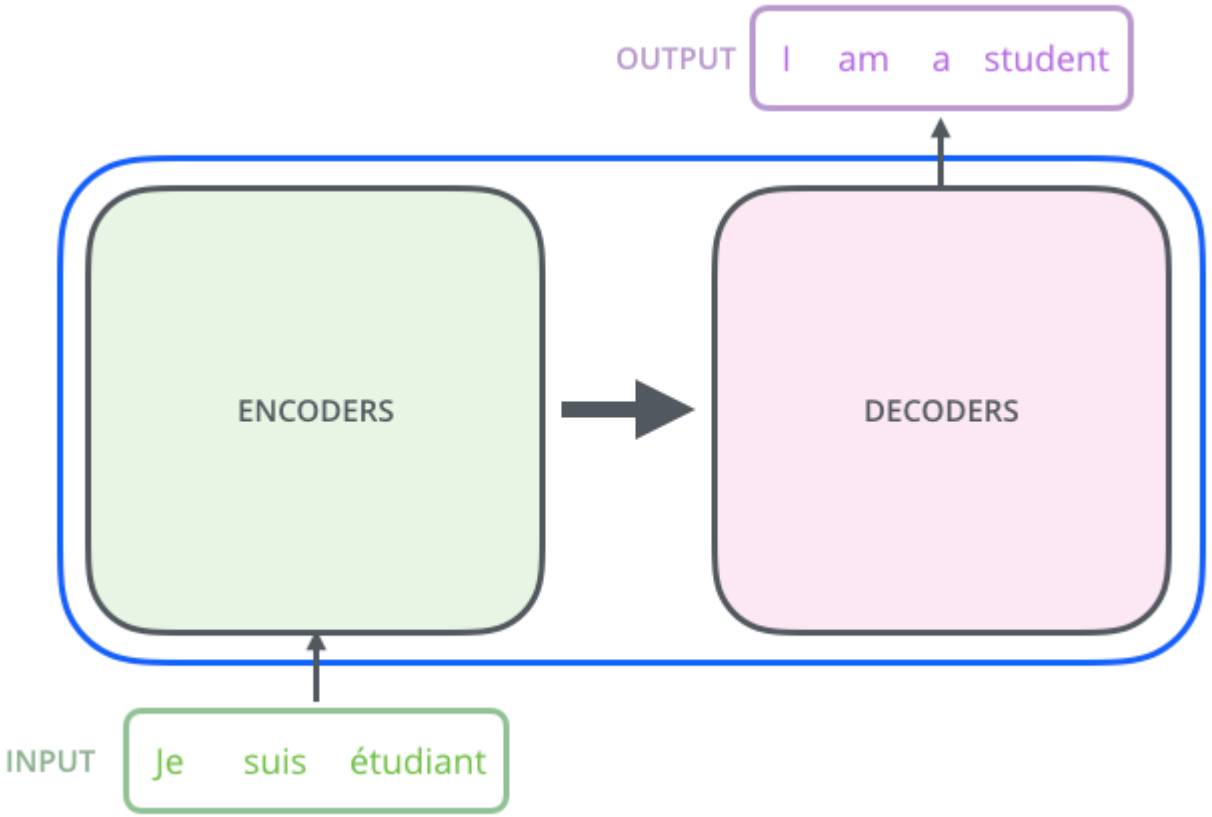
Baswani et al., 2017, Attention Is All You Need

Coarse view

It takes **a whole sentence** in one language and outputs is translation in another.

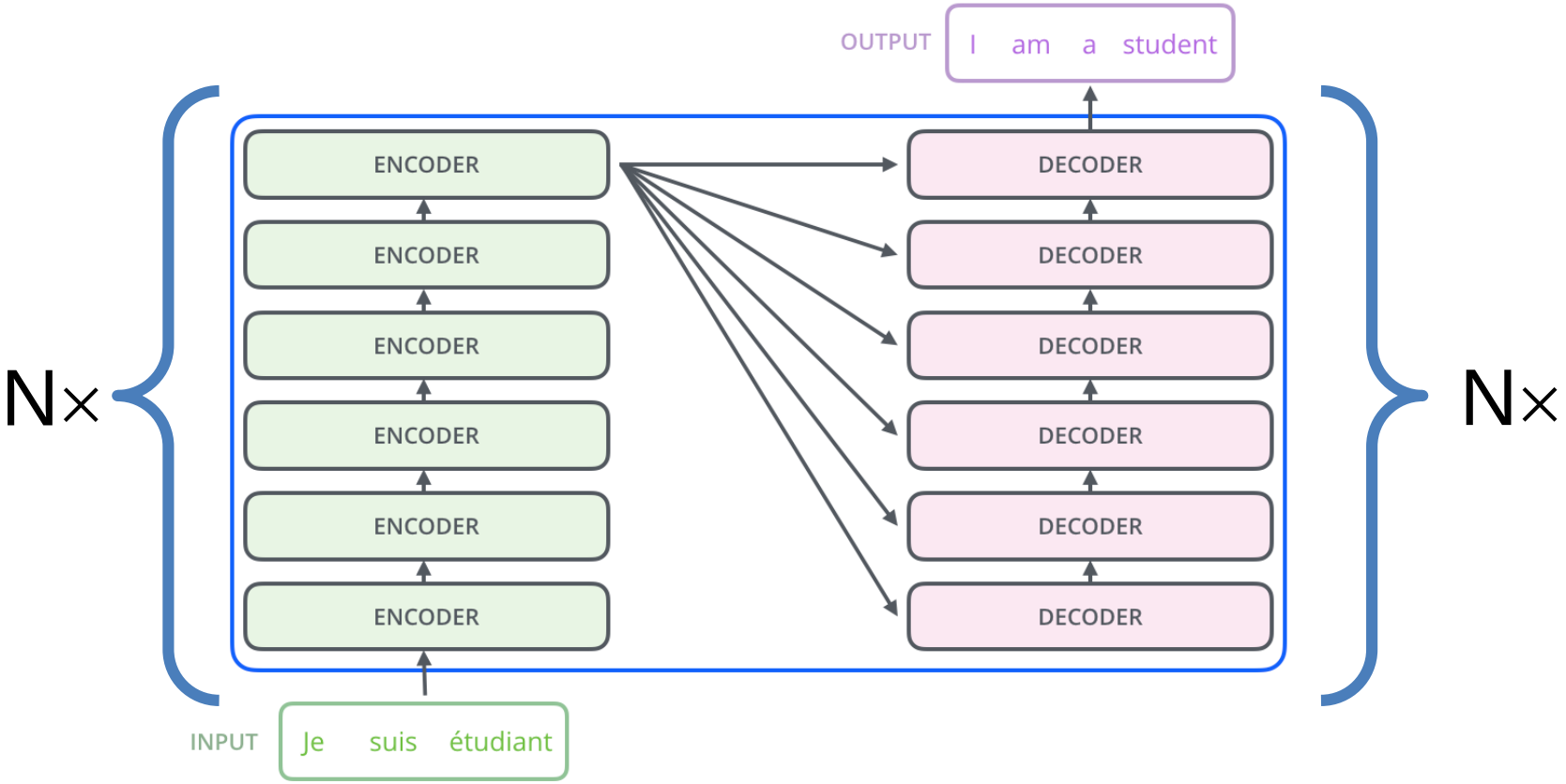
Coarse view

Again, an encoder-decoder scheme.



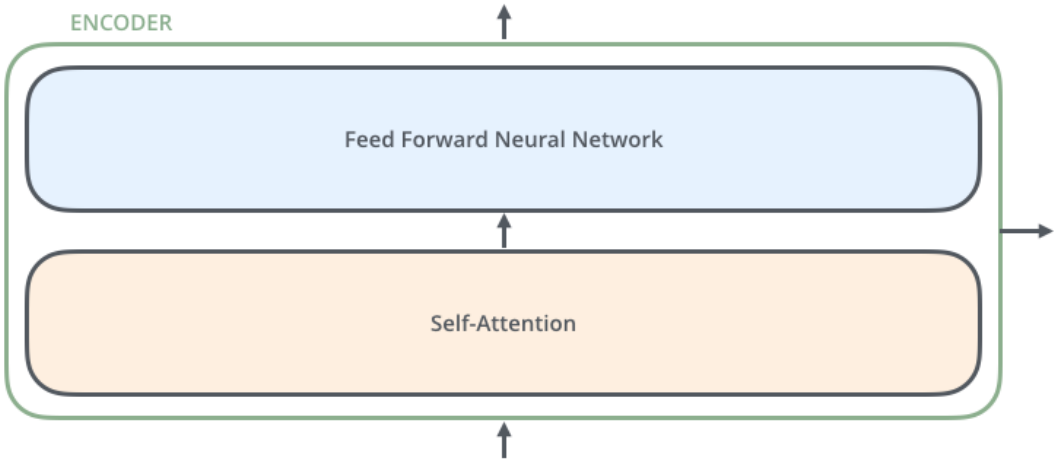
Looking inside the encoder/decoder

Actually, a stack of encoders and a stack of decoders.



Encoders' components

The encoders are all identical in structure (yet they do not share weights). Each one is broken down into two sub-layers:



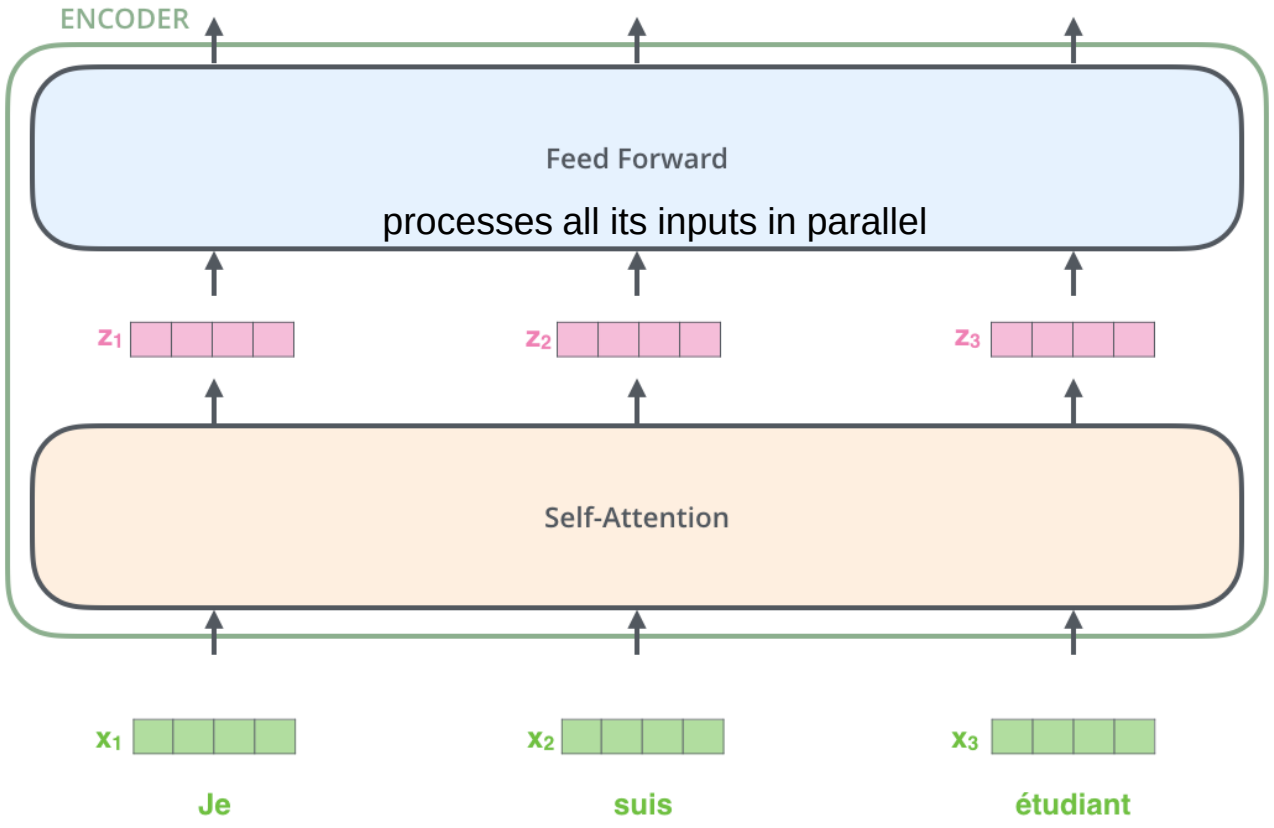
a) A self-attention layer helps the encoder look at other words in the input sentence as it encodes a specific word.

b) The outputs of the self-attention layer are fed to a feed-forward neural network.

c) The exact same feed-forward network is independently applied to each position.

Bringing in the tensors

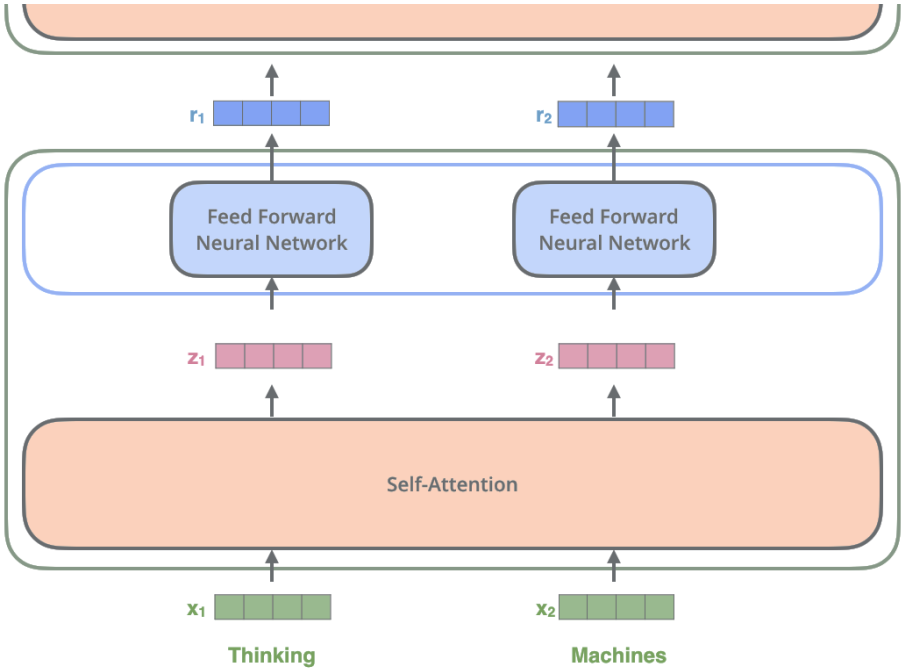
After embedding the words of the input sentence flows through both layers of the encoder.



The flow through the encoder

.

4. then upwards to the next encoder.
3. then into a feed-forward neural network
2. which flow into a ‘self-attention’ layer,
1. an encoder receives a list of vectors



Self-Attention

Motivation: Let's translate the following sentence:

"The animal didn't cross the street because it was too tired"

What does "it" in this sentence refer to?

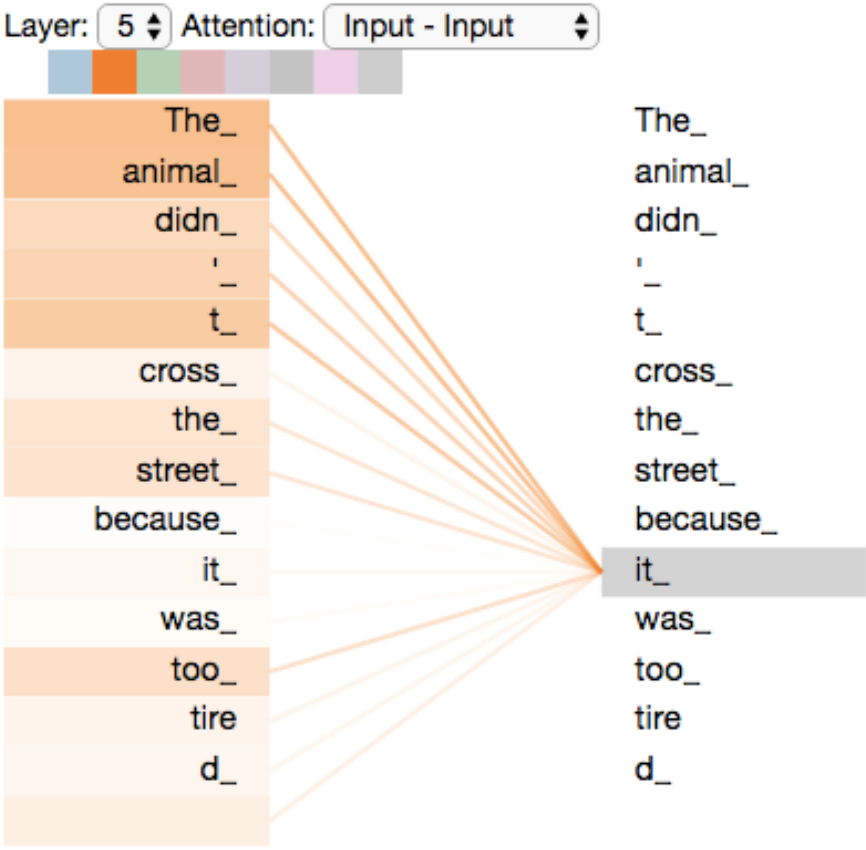
And in this sentence?

"The animal didn't cross the street because it was too wide"

Self-Attention

Self-attention is a method that exploits the “understanding” of other relevant words in the same sentence to determine the semantic of the word currently being processed.

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



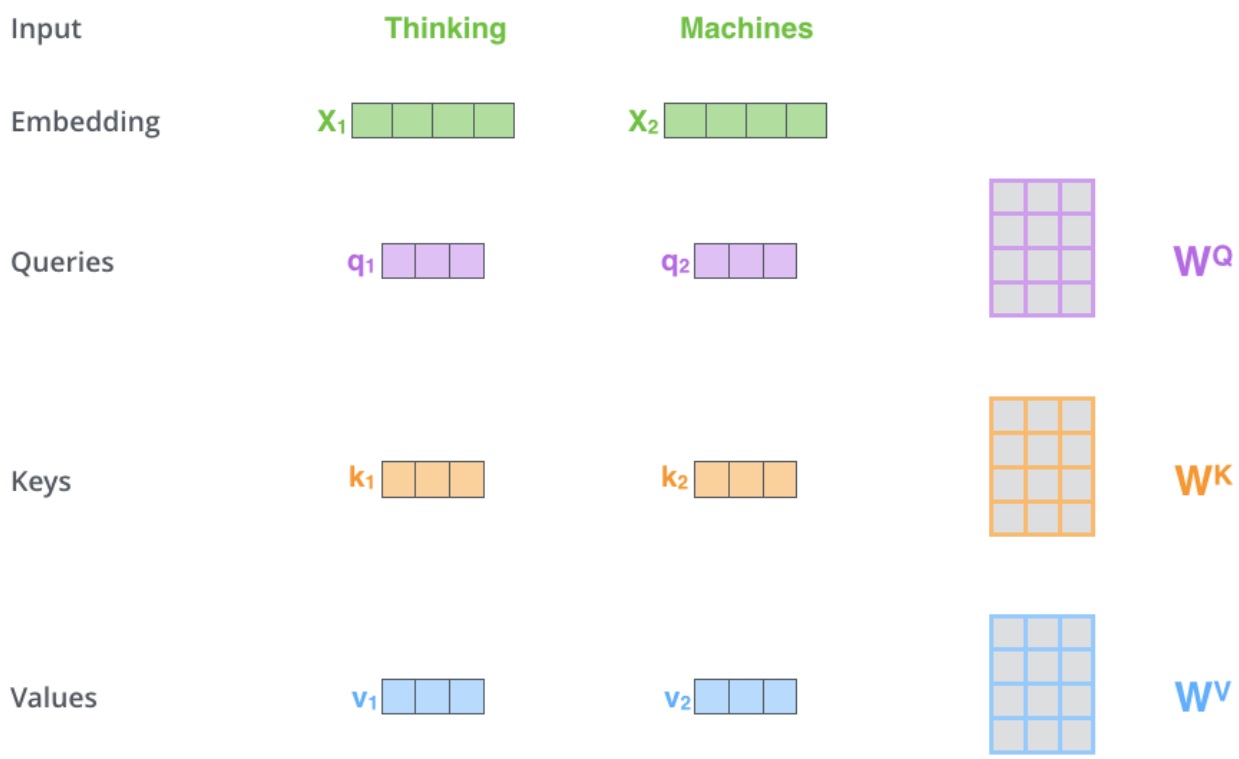
What are *key/value/query ?*

The key/value/query concepts come from retrieval systems.

For example, when you type a query to search for some video on Youtube, the search engine will map your **query** against a set of **keys** (video title, description, etc.) associated with candidate videos in the database, then present you the best matched videos (**values**)

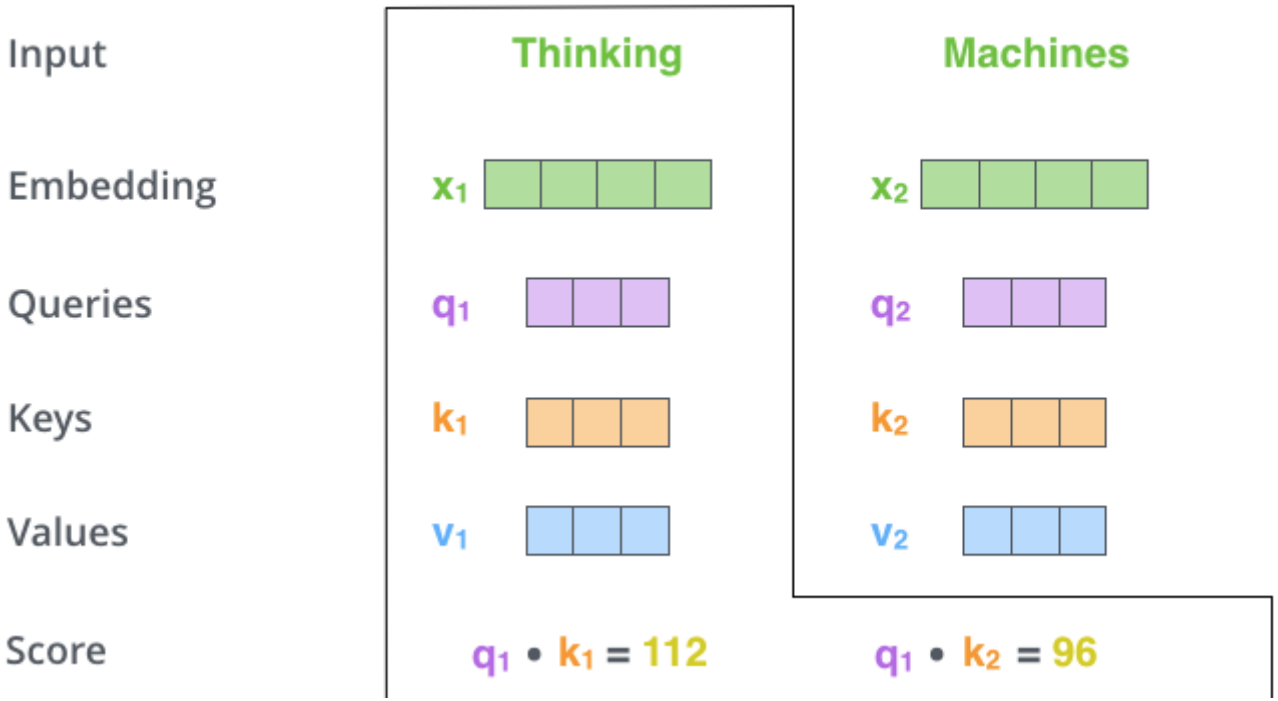
Calculating Self-Attention in Detail

1. create a Query, a Key, and a Value vector by multiplying the embedding by matrices learned during the training process.



Calculating Self-Attention in Detail

2. The **second step** is to calculate a score. It tells how much focus to place on other parts of the input sequence as we encode a word at a certain position.

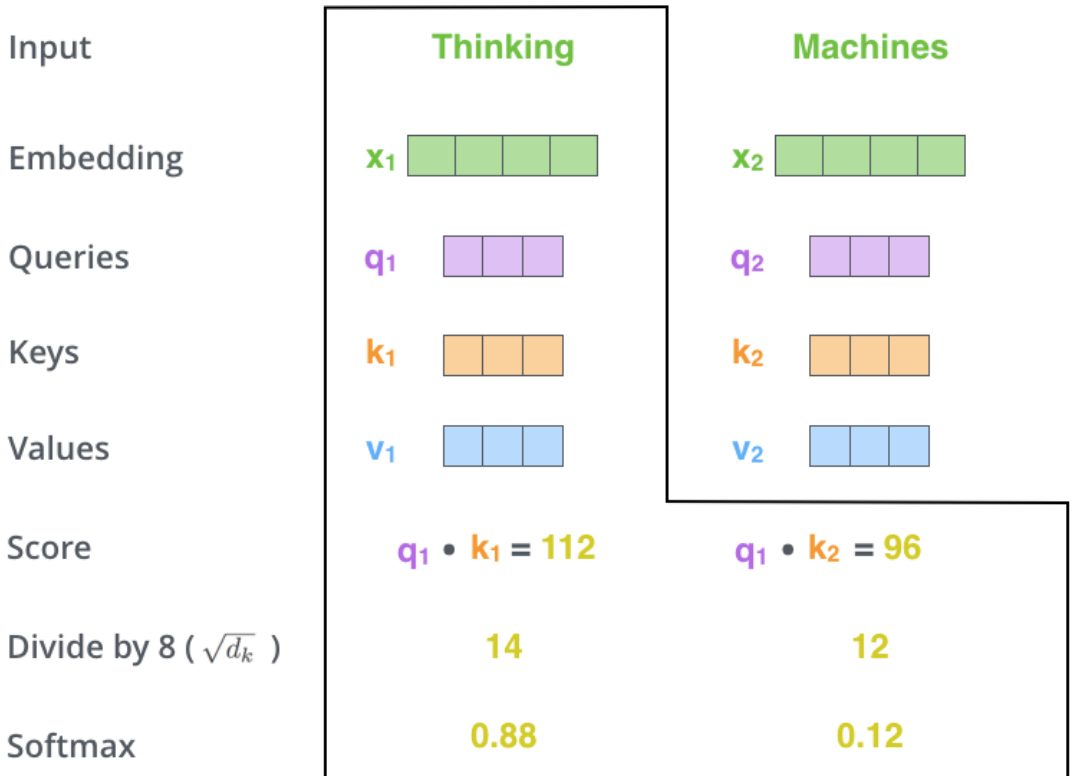


The score is the dot product of the query with the key vector.

Calculating Self-Attention in Detail

3. divide by the square root of the dimension of the key vectors (helps training)

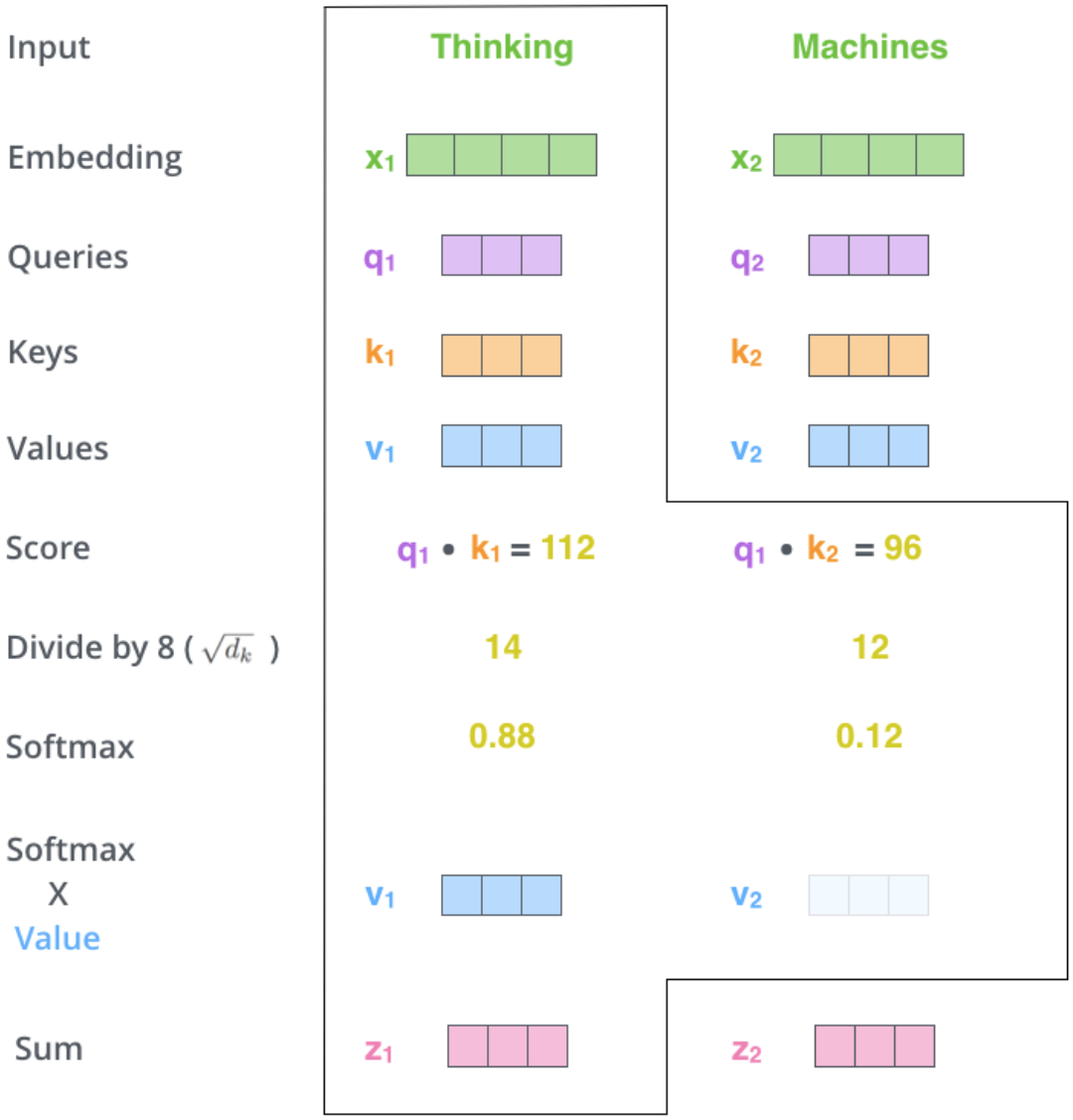
4. the results pass to a *softmax*, to put them in [0.1], and to peak the highest score.



Calculating Self-Attention in Detail

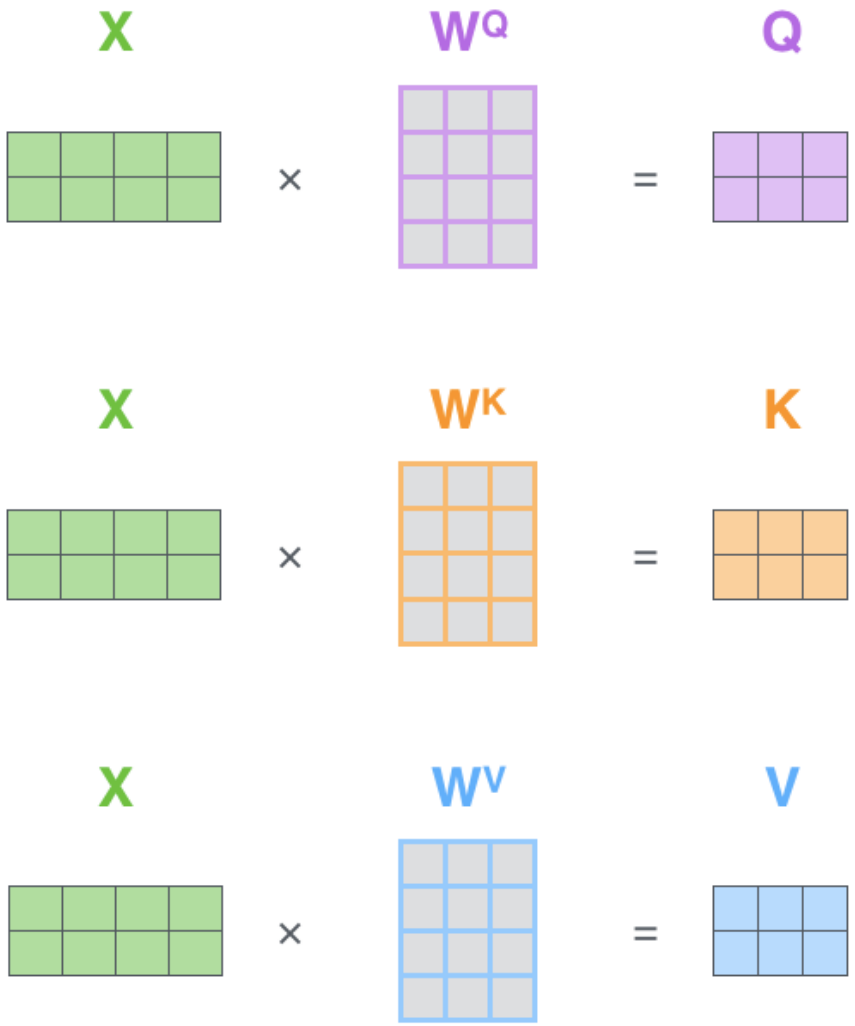
5. multiply each value vector by the *softmax* score The intuition here is to keep intact the values of the word(s) we want to focus on, and drown-out irrelevant words (by multiplying them by tiny numbers).

6. sum up the weighted value vectors. This produces the output of the self-attention layer at this position (for the first word).



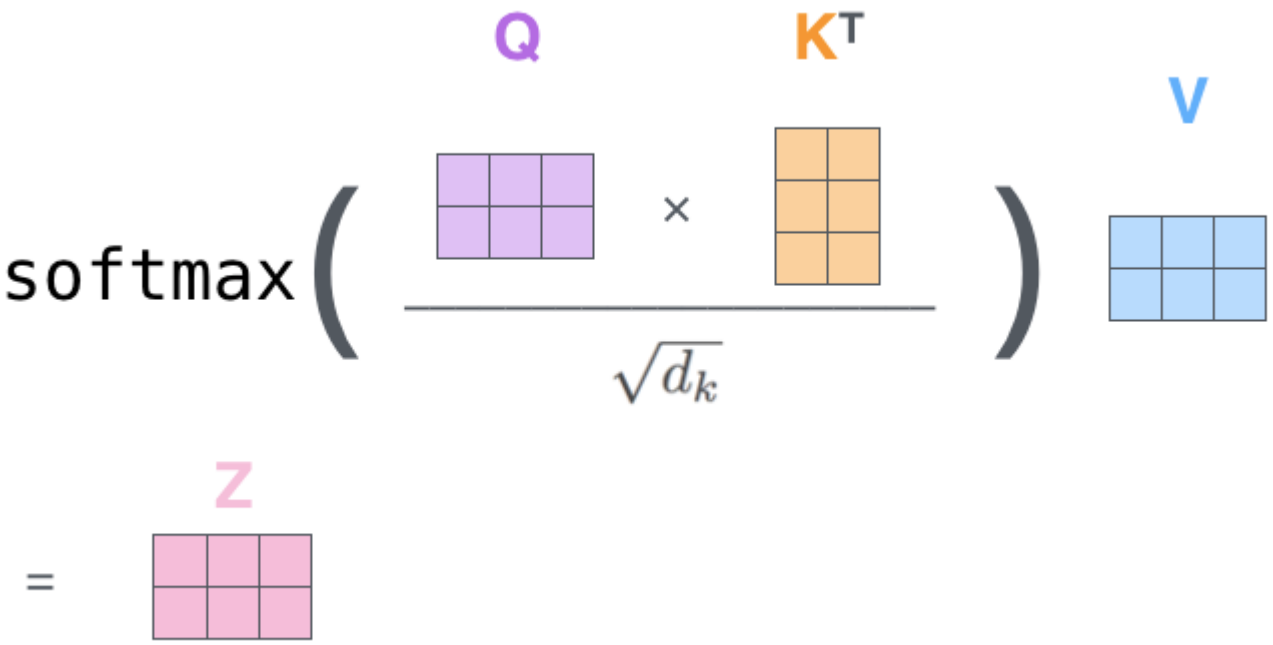
Self-Attention in Matrix form

The **first step** is done by packing all embeddings into a matrix (**X**) and multiply by the trained weight matrices (W^Q, W^K, W^Q)



Self-Attention in Matrix form

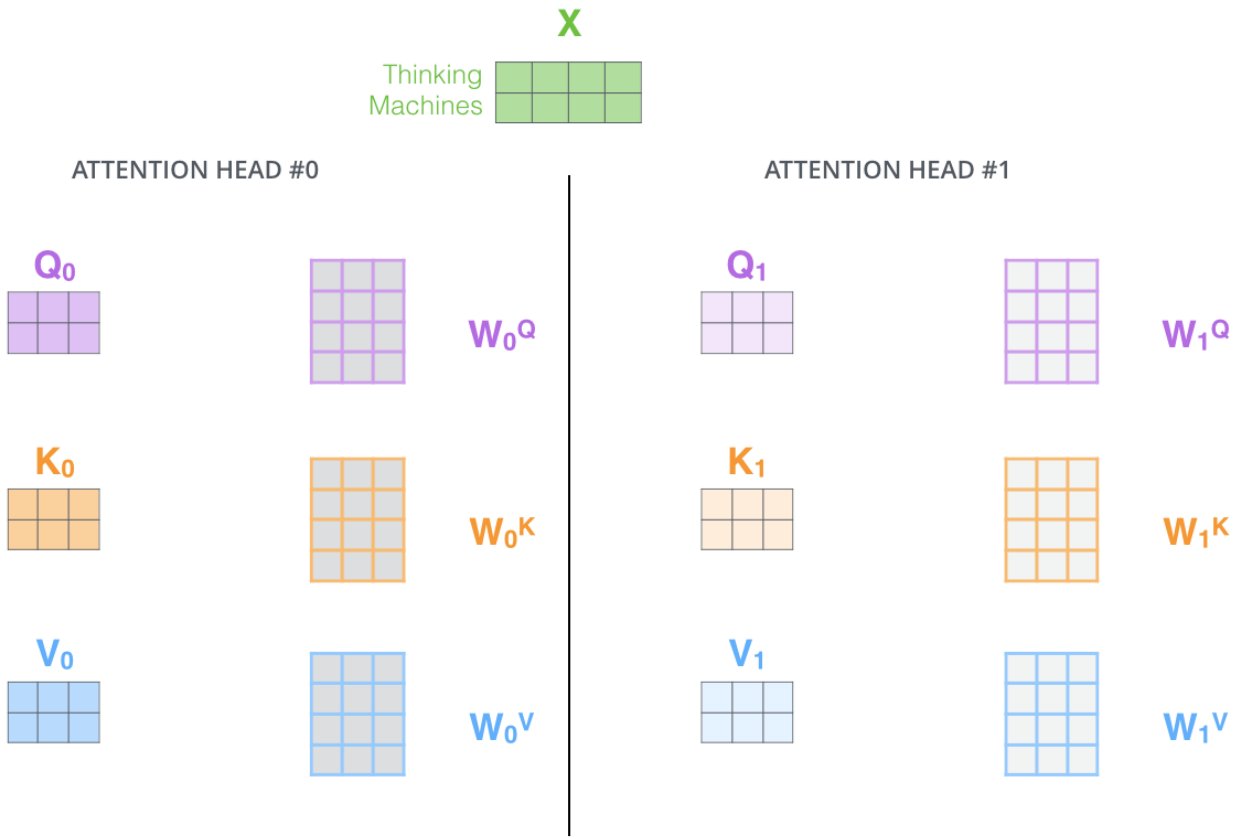
The **final steps** are condensed in a simple formula:



So the self-attention involves matrix operations, and can be efficiently implemented in a GPU.

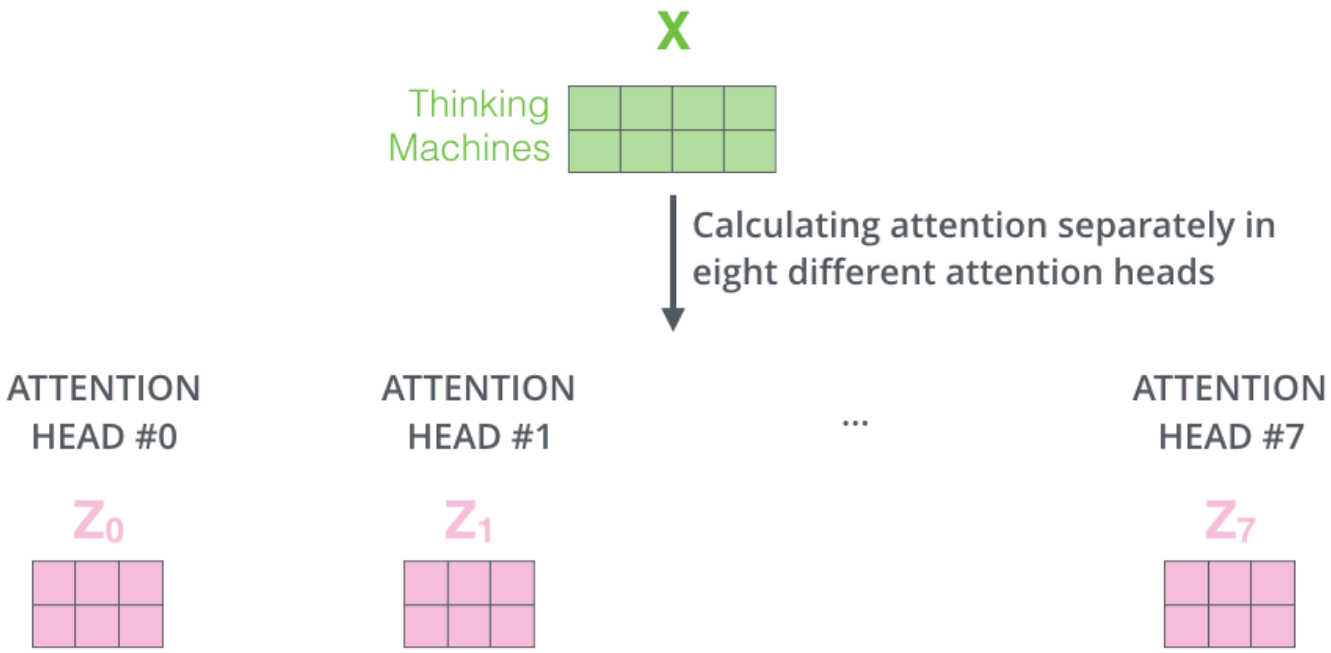
Multiple Heads

To avoid biased values for W^Q, W^K and W^Q we run the training procedure multiple times (heads) with different initializations and obtain multiple estimates for them.



Multiple Heads

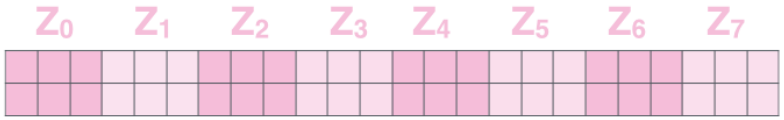
If you do the self-attention calculation using 8 estimates you will get 8 different Z matrices



Multiple Heads

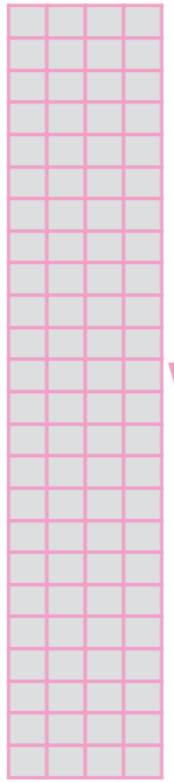
The feed-forward layer expects a single matrix (a vector for each word).

1) Concatenate all the attention heads

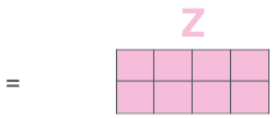


2) Multiply with a weight matrix W^O that was trained jointly with the model

X



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



To condense the results, we concat the matrices then multiply them by an additional weights matrix W^O.

Multiple Heads Summary

1) This is our input sentence*

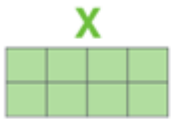
2) We embed each word*

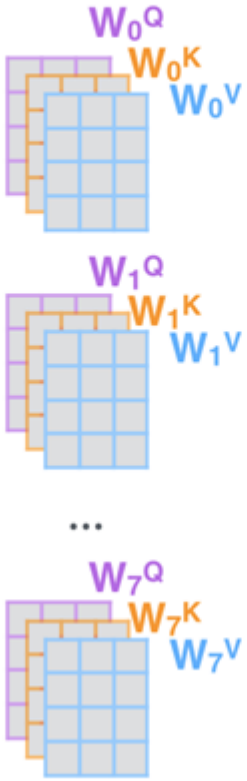
3) Split into 8 heads. We multiply X or R with weight matrices

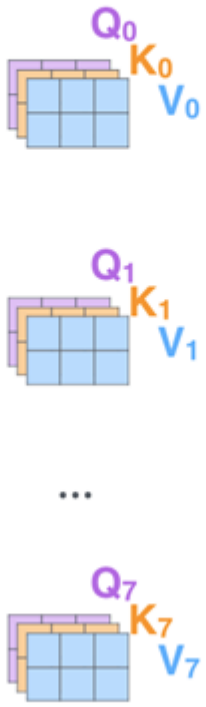
4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

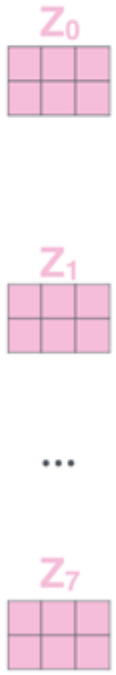
Thinking Machines

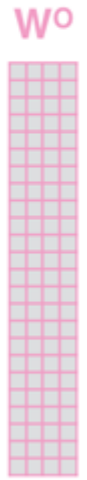


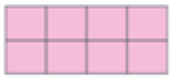




$$softmax\left(QK^T/\sqrt{d_k}\right)V$$







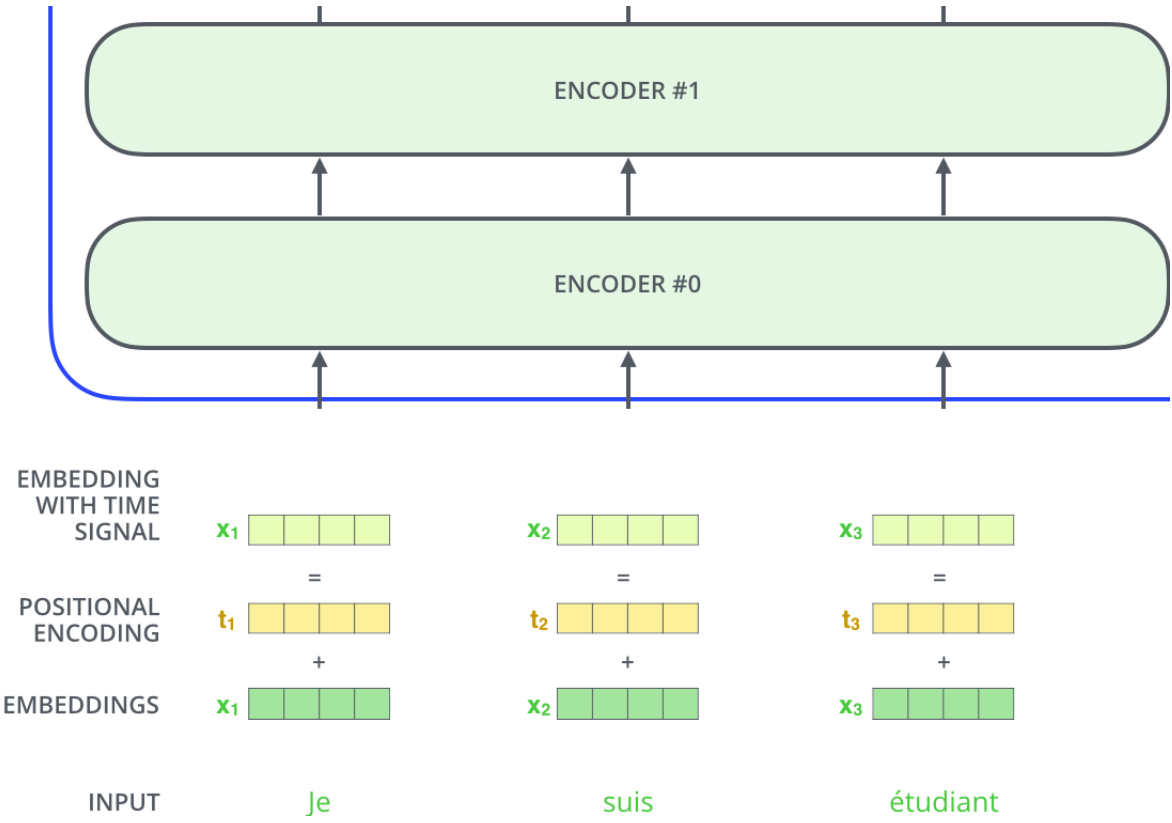
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Positioning encoding

The order of the words in the sentence matters.

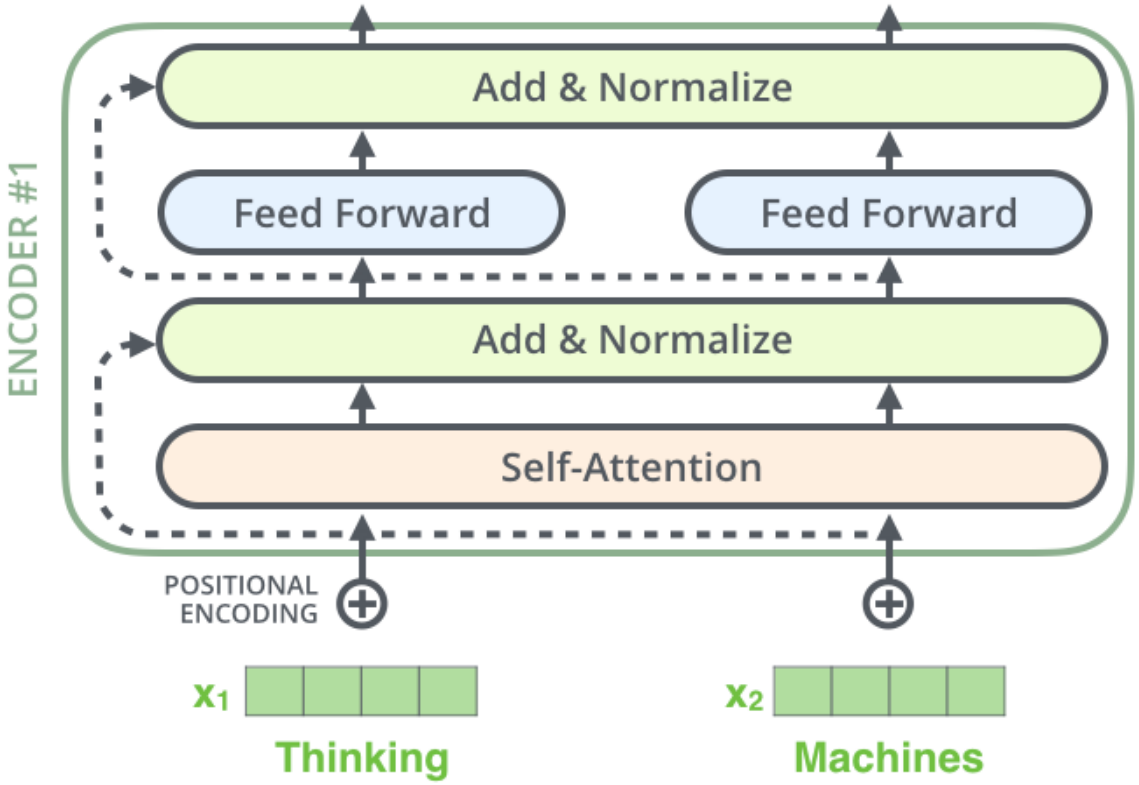
Close words tend to be more relevant to each other than distant ones.

To address this, a positioning encoding vector is added to each input embedding.



The Residuals

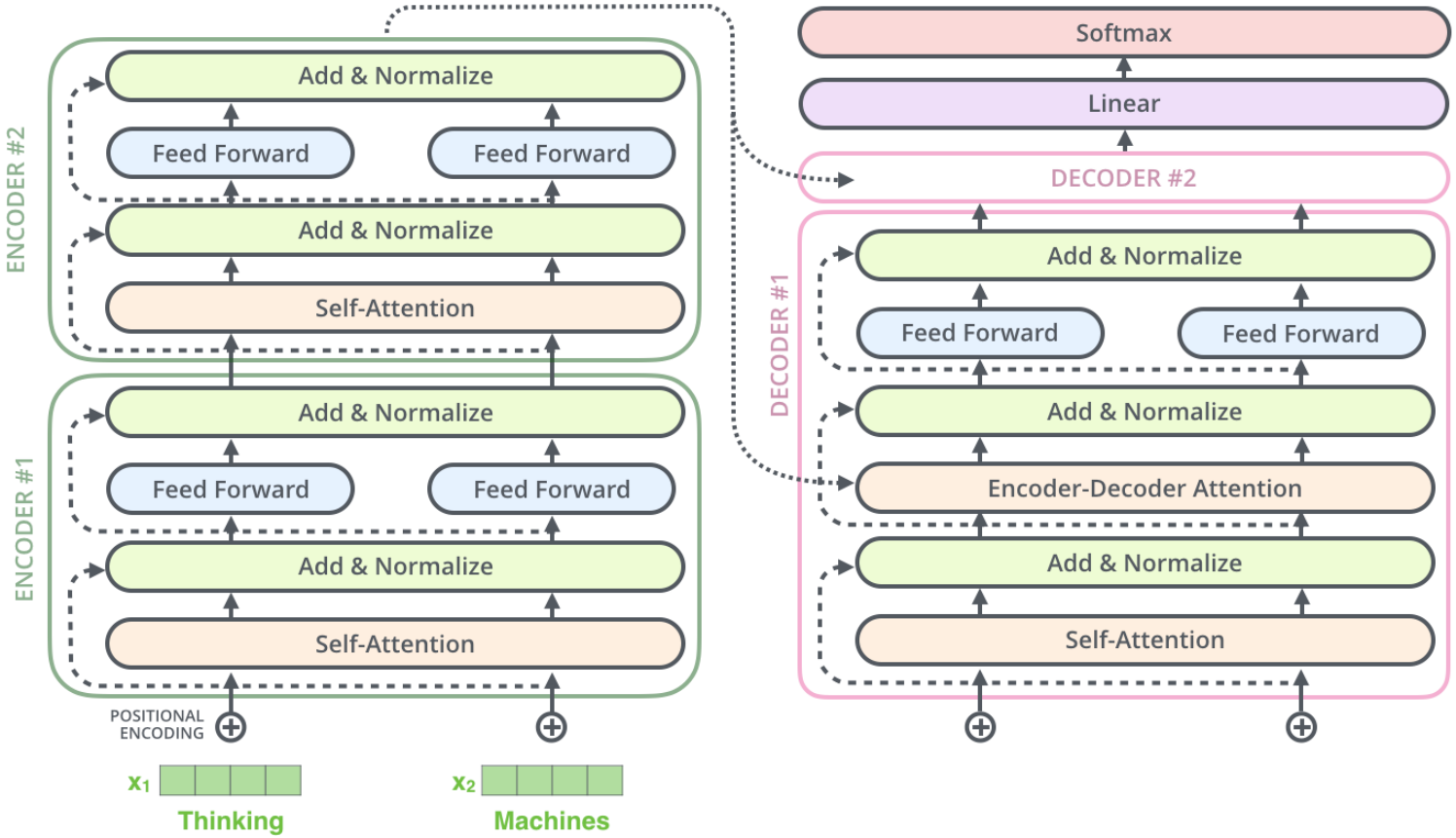
each sub-layer (self-attention, ffnn) in each encoder has a residual connection around it, and is followed by a layer-normalization step



The Residuals

Also for the decoder sub-layers.

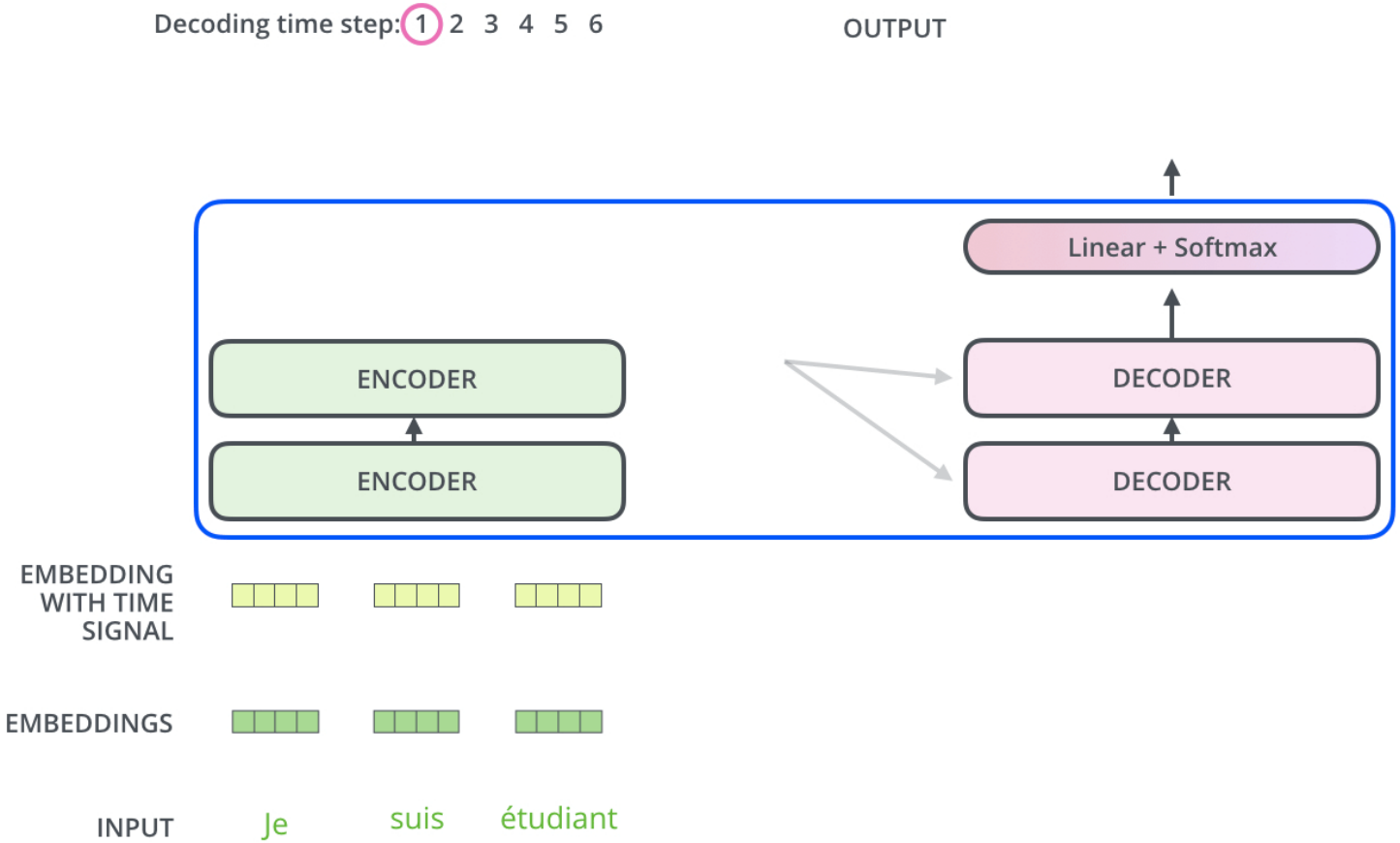
Example for 2 stacked encoders/decoders:



The Decoder

The output of the top encoder is transformed into a set of attention vectors K and V.

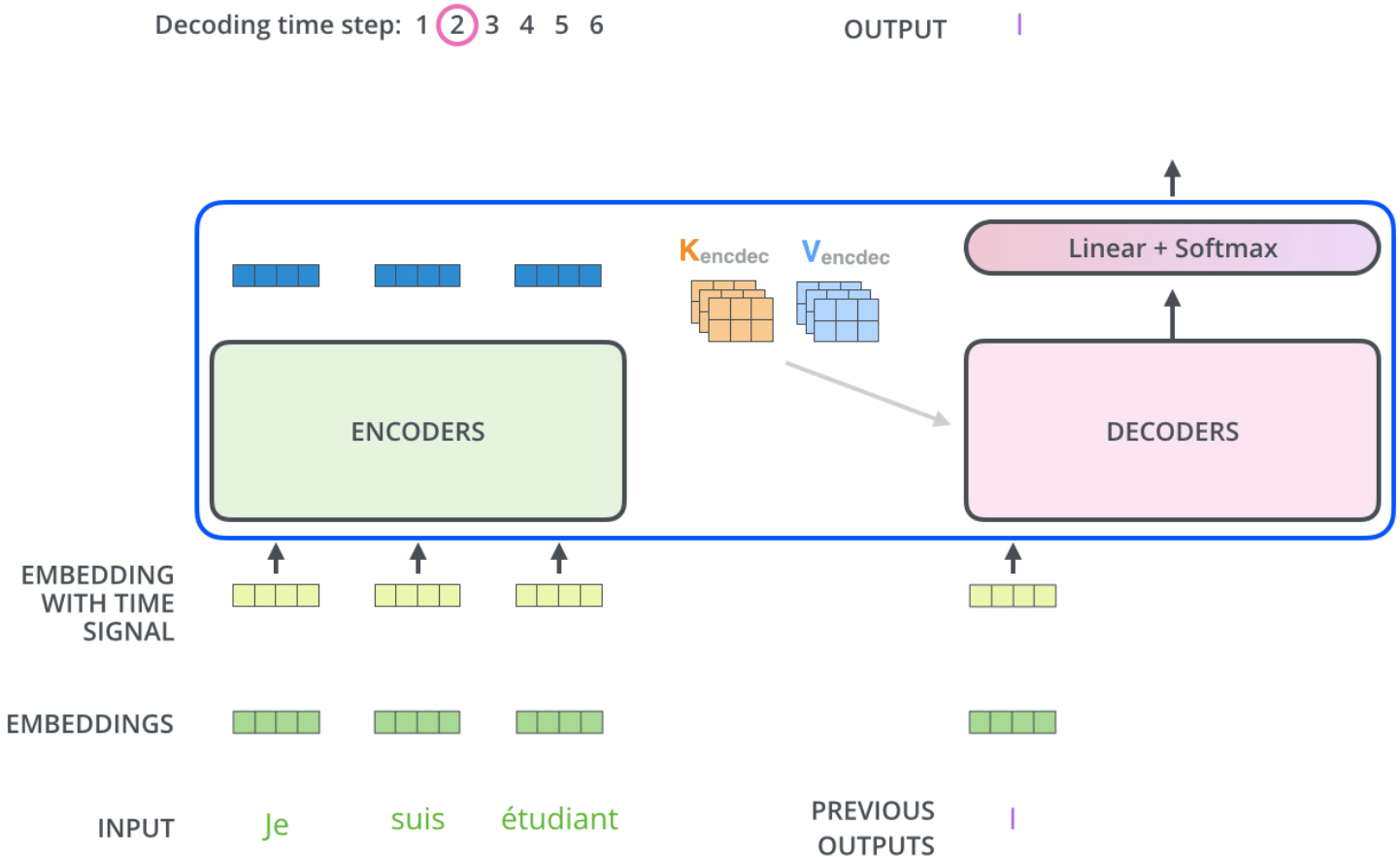
These are to be used by each decoder in its “encoder-decoder attention” layer which helps the decoder focus on appropriate places in the input sequence:



The Decoder

It goes on until a special symbol indicates it is done.

The output of each step is fed to the bottom decoder in the next time step, after embedding and positional encoding.



Overview

1. Motivation
2. Seq2Seq
3. Attention
4. Transformers
5. Applications in Computer Vision
6. Final Comments

Attention in Computer Vision

Image Generation

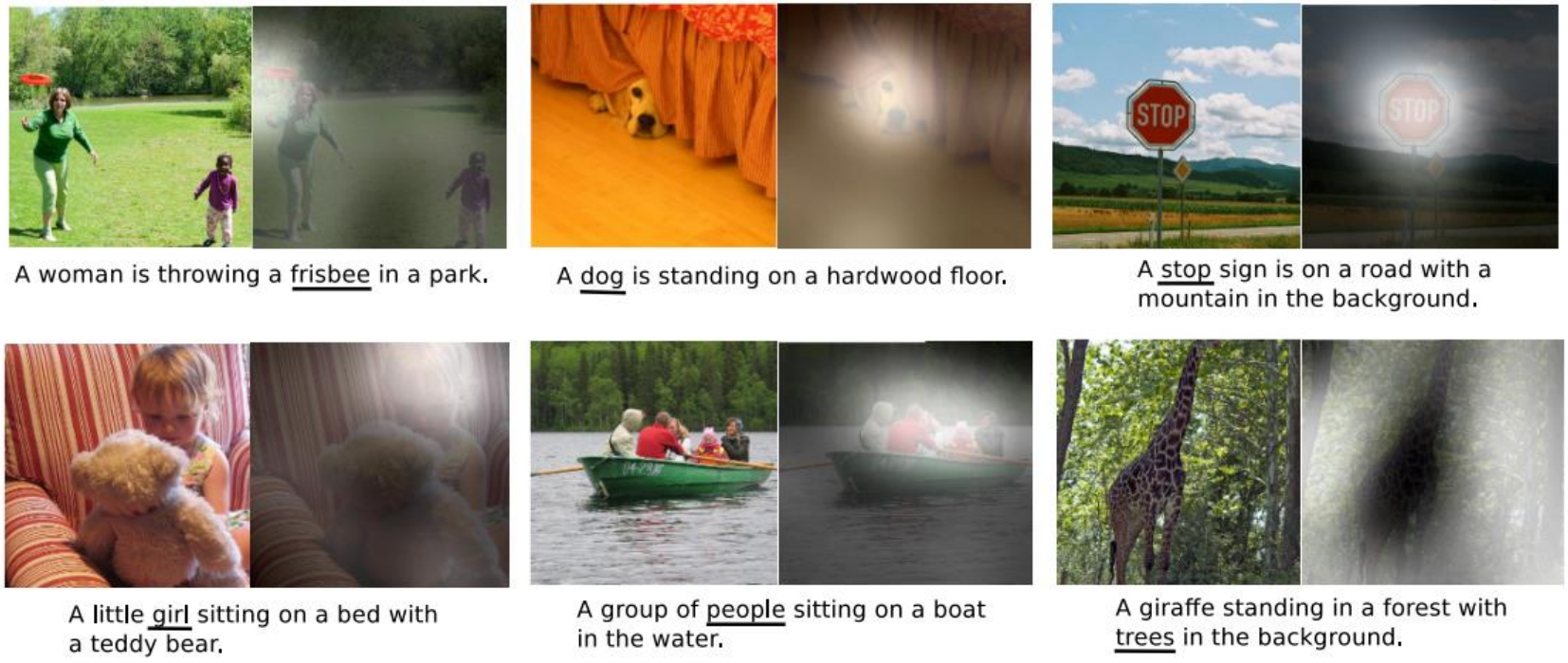
Figure 9. **Generated SVHN images.** The rightmost column shows the training images closest (in L^2 distance) to the generated images beside them. Note that the two columns are visually similar, but the numbers are generally different.

Source: Gregor, et al., 2015 DRAW: A Recurrent Neural Network For Image Generation

Attention in Computer Vision

Image Captioning

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



Source: [Xu, et al., 2016 Show, Attend and Tell: Neural Image Caption Generation with Visual Attention]

Attention in Computer Vision

Attention to Scale for Semantic Segmentation

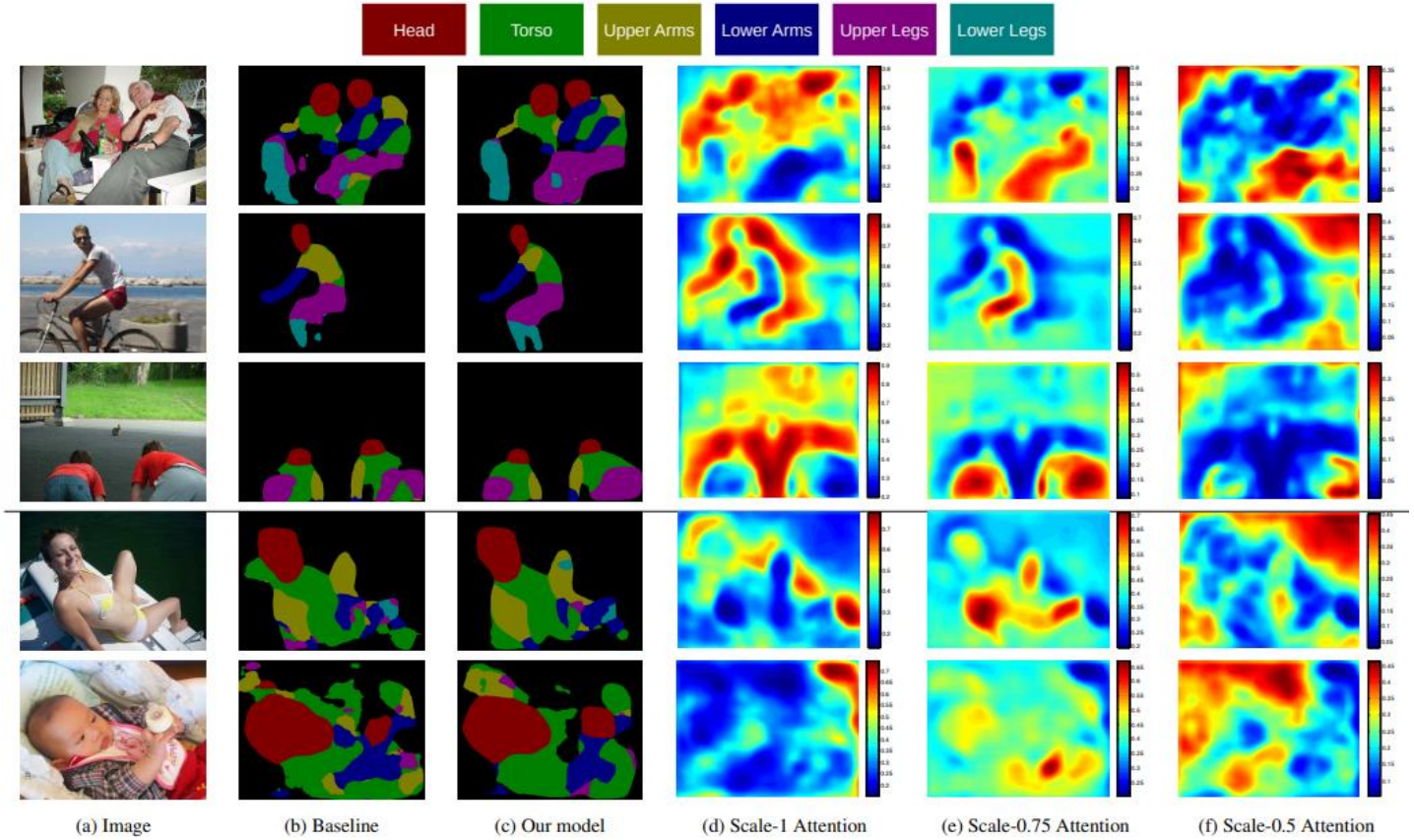


Figure 5. Results on PASCAL-Person-Part *validation* set. DeepLab-LargeFOV with one scale input is used as the baseline. Our model employs three scale inputs, attention model and extra supervision. Scale-1 attention captures small-scale parts, scale-0.75 attention catches middle-scale torsos and legs, while scale-0.5 attention focuses on large-scale legs and background. Bottom two rows show failure examples.

Chen et al., 2016 [Attention to Scale: Scale-aware Semantic Image Segmentation]

Attention in Computer Vision

Self Attention Generative Adversarial Network (SAGAN).

It explores long range dependencies.

Figure 4. 128×128 examples randomly generated by the baseline model and our models "SN on G/D" and "SN on G/D+TTUR".

Source: Zhang, et al., 2019 https://arxiv.org/pdf/1805.08318.pdf

Attention in Computer Vision

Crop mapping from multitemporal RS images

(a) Kappa metric 23-class dataset

κ	RF	LSTM-RNN	Transformer	MS-ResNet	TempCNN
pre	0.76	$0.78^{\pm 0.01}$	$0.79^{\pm 0.03}$	$0.76^{\pm 0.03}$	$0.80^{\pm 0.00}$
raw	0.53	$0.71^{\pm 0.01}$	$0.71^{\pm 0.03}$	$0.69^{\pm 0.03}$	$0.67^{\pm 0.02}$

(b) Kappa metric 12-class dataset

κ	RF	LSTM-RNN	Transformer	MS-ResNet	TempCNN
pre	0.86	$0.87^{\pm 0.01}$	$0.87^{\pm 0.04}$	$0.85^{\pm 0.01}$	$0.88^{\pm 0.03}$
raw	0.62	$0.83^{\pm 0.01}$	$0.82^{\pm 0.02}$	$0.78^{\pm 0.02}$	$0.71^{\pm 0.07}$

(c) Overall accuracy metric 23-class dataset

acc.	RF	LSTM-RNN	Transformer	MS-ResNet	TempCNN
pre	0.83	$0.85^{\pm 0.01}$	$0.85^{\pm 0.02}$	$0.83^{\pm 0.02}$	$0.86^{\pm 0.00}$
raw	0.71	$0.81^{\pm 0.01}$	$0.80^{\pm 0.02}$	$0.79^{\pm 0.03}$	$0.79^{\pm 0.00}$

(d) Overall accuracy metric 12-class dataset

acc.	RF	LSTM-RNN	Transformer	MS-ResNet	TempCNN
pre	0.91	$0.92^{\pm 0.01}$	$0.92^{\pm 0.03}$	$0.91^{\pm 0.01}$	$0.92^{\pm 0.02}$
raw	0.80	$0.90^{\pm 0.00}$	$0.89^{\pm 0.01}$	$0.87^{\pm 0.01}$	$0.83^{\pm 0.04}$

(e) Class-mean f_1 score 23-class dataset

f1	RF	LSTM-RNN	Transformer	MS-ResNet	TempCNN
pre	0.38	$0.47^{\pm 0.02}$	$0.50^{\pm 0.06}$	$0.49^{\pm 0.03}$	$0.50^{\pm 0.02}$
raw	0.18	$0.43^{\pm 0.01}$	$0.45^{\pm 0.06}$	$0.44^{\pm 0.01}$	$0.36^{\pm 0.01}$

(f) Class-mean f_1 score 12-class dataset

f1	RF	LSTM-RNN	Transformer	MS-ResNet	TempCNN
pre	0.55	$0.60^{\pm 0.02}$	$0.66^{\pm 0.07}$	$0.64^{\pm 0.02}$	$0.60^{\pm 0.06}$
raw	0.34	$0.63^{\pm 0.01}$	$0.64^{\pm 0.07}$	$0.55^{\pm 0.04}$	$0.41^{\pm 0.07}$

Source: Russwurm & Körner, 2019, Self-Attention for Raw Optical Satellite Time Series Classification

Overview

1. Motivation
2. Seq2Seq
3. Attention
4. Transformers
5. Applications in Computer Vision
6. Final Comments

Pros & Cons

Pros:

- Faster training than RNN/LSTM/GRU with parallelization.
- Attention ignores order: useful for tasks where distant parts of a sequence are as likely to be import as the close ones.

Cons:

- Very Large models that require a *lot* of memory and compute to train
- Relatively young class of models, so we know less about them & how it works

Risks at jumping into it

Common errors:

- it demands a lot of data; if you don't have enough you'd better use another class of models
- you may run out of memory
- libraries are pretty young, you're are likely to be the first person to run in to a specific bug

References

- The seminal paper by Baswani et al., 2017, Attention Is All You Need

- Jay Alammar, 2018, Tutorial "Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)"

- Jay Alammar, 2018, Tutorial "The Illustrated Trasnformer"

- Leo Dirack, 2019, "LSTM is dead. Long Live Transformers", Nov. 2019

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Attention & Transformers

Thanks for your attention

R. Q. Feitosa