Lecture 2:
LSTM is dead. Long Live Transformers!

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env

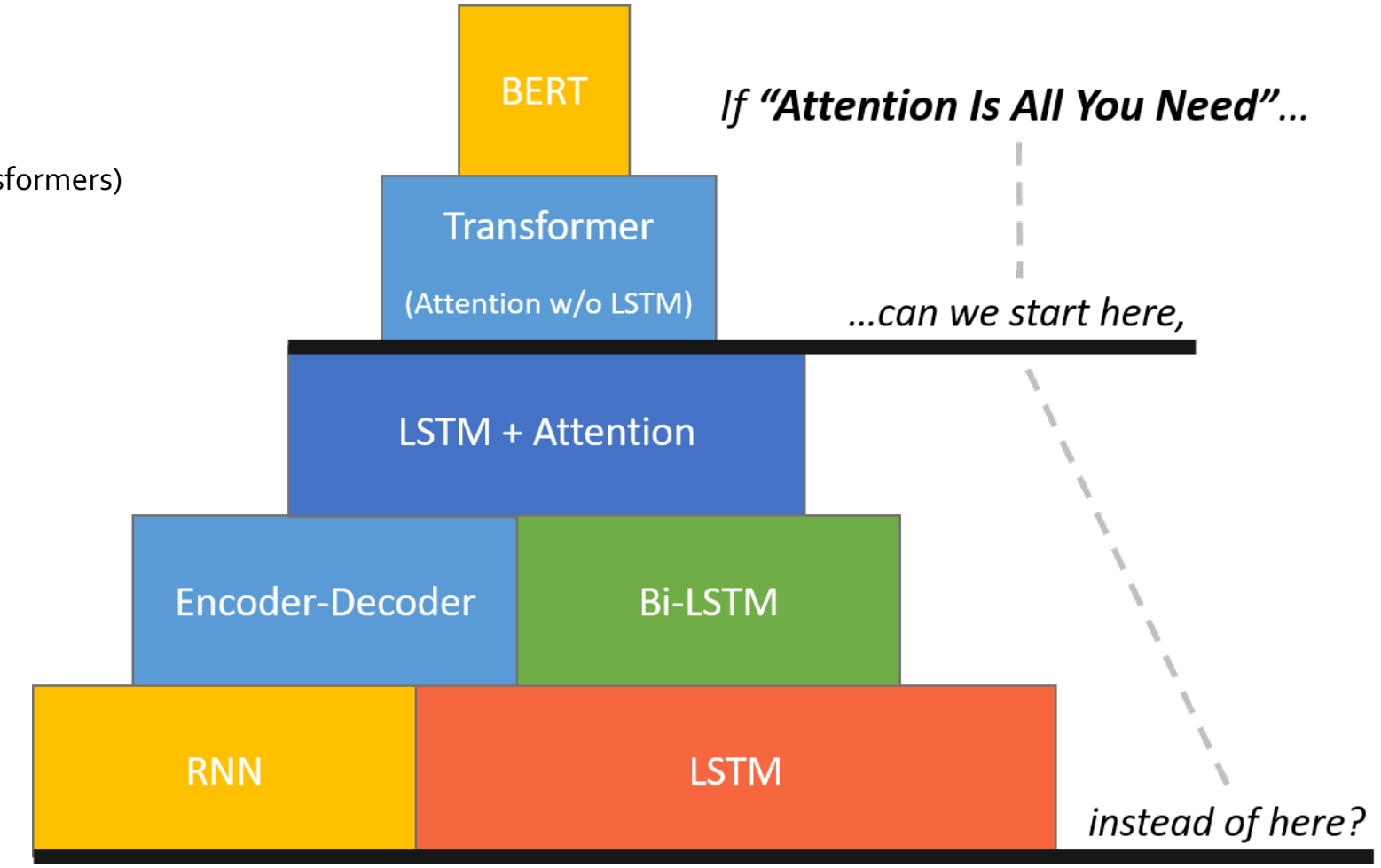
reda.bouadjenek@deakin.edu.au

May 28, 2020

The BERT Mountain!

By Chris McCormick

(Bidirectional Encoder Representations from Transformers)



Outline

- Background: NLP, Sequence Modeling.
- LSTM: Awesome but not good enough.
- Transformers: How & Why?

NLP background (1/3)

Classic techniques: Sparse, Sequence models

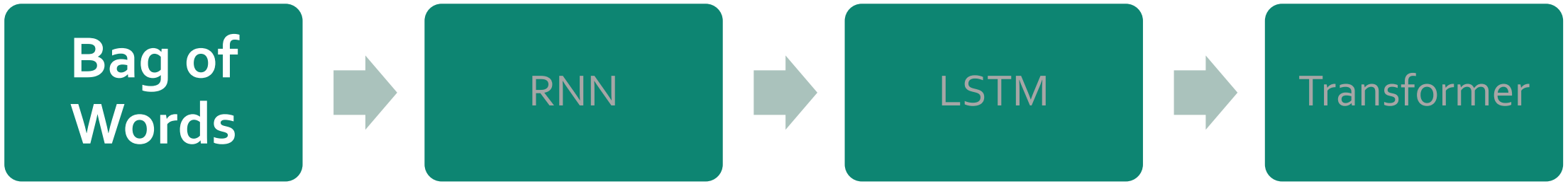
$f($  $) \approx y$

Document Is Spam?

Sequence modeling is a problem

$$f : \mathbb{R}^d \mapsto \mathbb{R}$$

Fixed size vector



Bag of Words

- A document is represented as vector of words.
  - One dimension per word.
  - Vector size is the vocabulary size, e.g., English may contain 100k words.
  - Different weighting schemas can be used, e.g., tf, log(tf), tf-idf, Boolean, etc.
  - Sparse vector, e.g., almost all values are zeros.

Order matters for NLP tasks!

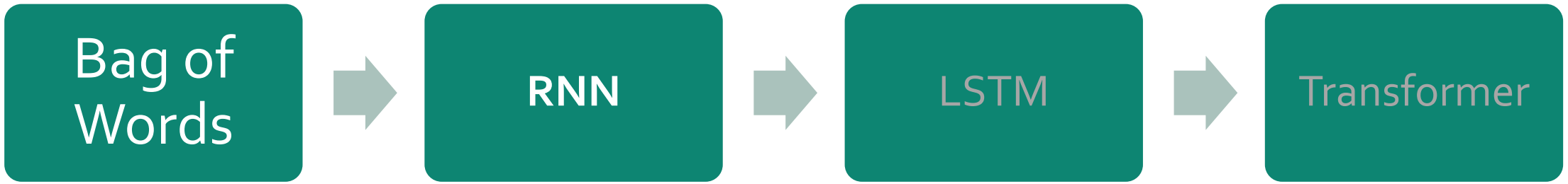
- Assumes independence between words:
  - The sentences "**John likes Mary**" has the same representation as "**Mary likes John**" – even though the semantic is different).
- May work well for Information Retrieval tasks, but not for NLP tasks!
  - Sentiment analysis:

    "Ah **no**, there are good movies on Netflix!" vs. "Ah, there are **no** good movies on Netflix!"

Order matters for NLP tasks!

"Ah **no**, there are good movies on Netflix!" vs. "Ah, there are **no** good movie on Netflix!"

- Use N-grams.
  - Dimensionality grows exponentially V^N.
  - 3-grams with English: $(10^5)^3 = 10^{15} = 1{,}000{,}000{,}000{,}000{,}000$ entries.
  - Too expensive!



Recurrent Neural Network

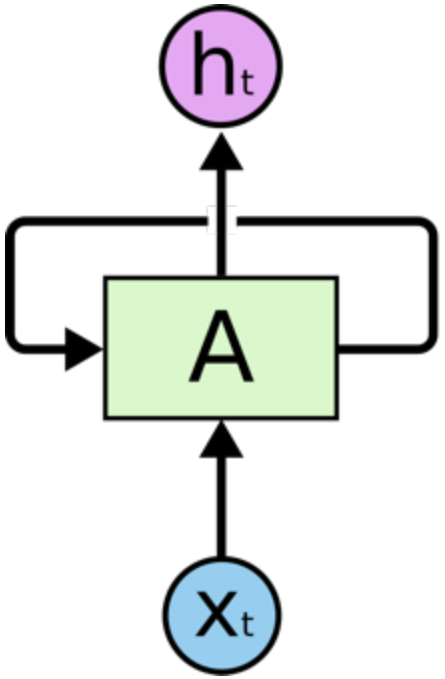
RNN: a new approach

- How to calculate: $f(x_1, x_2, x_3, \cdots, x_N)$?
- A for-loop:

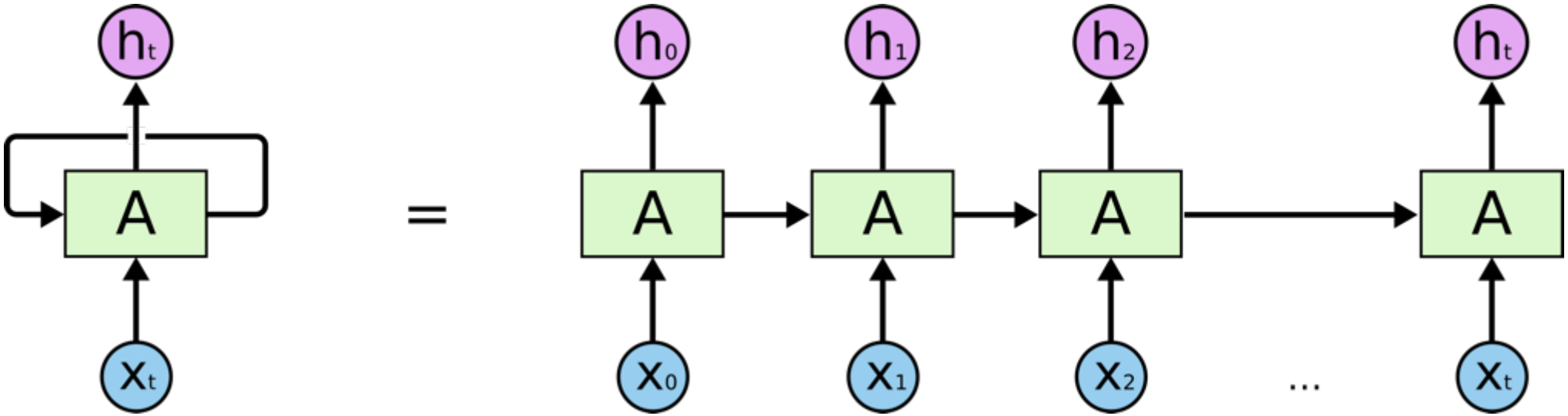
For $i = 0$ to N:

$$h_{i+1} = A(h_i, x_i)$$

Return $f(\vec{x}) = h_N$



Unrolled RNN



Vanishing & Exploding Gradients

- $h_{i+1} = A(h_i, x_i)$
- $A(h_i, x_i) = \mathbb{W}x_i + Zh_i$
- $h_3 = A(A(A(h_0, x_0), x_1), x_2)$
- $h_N = A(A(\cdots A(h_0, x_0), x_1), x_2 \cdots), x_{N-2}), x_{N-1})$
- $h_N = \mathbb{W}x_{N-1} + Z\mathbb{W}x_{N-2} + Z^2\mathbb{W}x_{N-3} + \cdots + Z^{N-1}\mathbb{W}x_0 + Z^N h_i$

Multiplying matrix N times!

Vanishing & Exploding Gradients

- Numbers < 1:

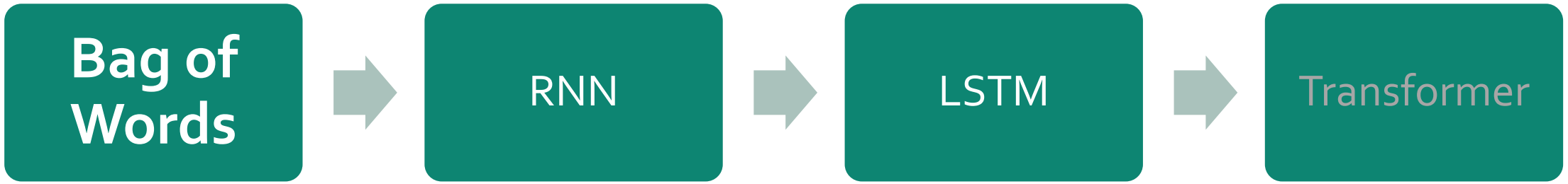
```
>>> 0.9 **100
2.6561398887587544e-05
>>> 0.9 **200
7.055079108655367e-10
```

- Numbers > 1:

```
>>> 1.1 ** 100
13780.61233982238
>>> 1.1 ** 200
189905276.4604649
```

LSTM to the Rescue! (2/3)

The Rise and Fall and Rise and Fall of LSTM



LONG SHORT-TERM MEMORY

Neural Computation 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

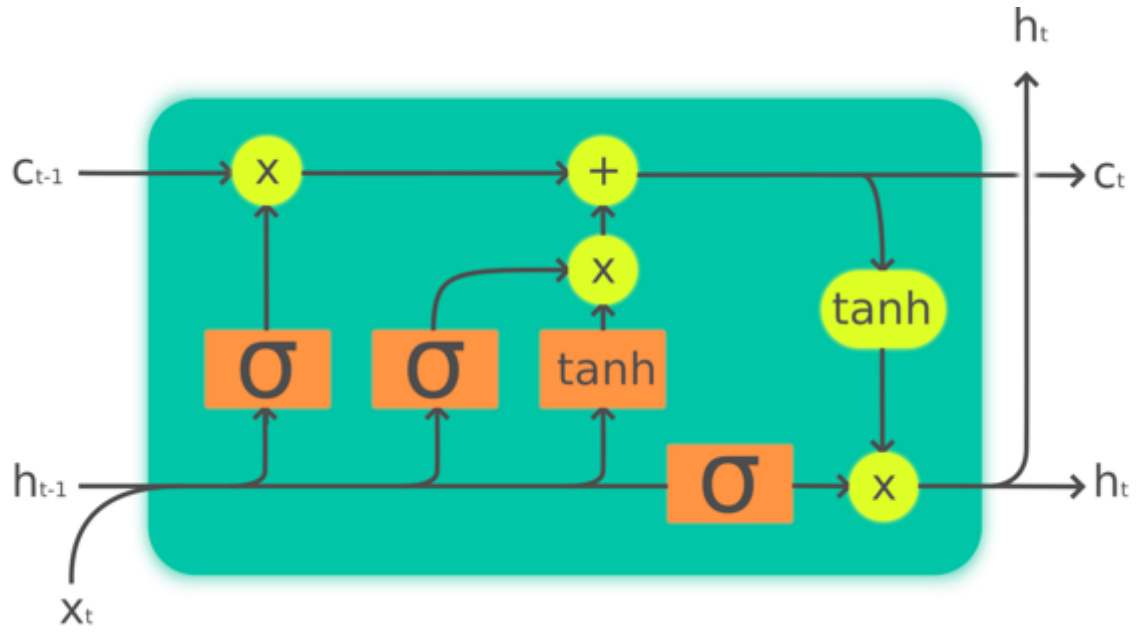
Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

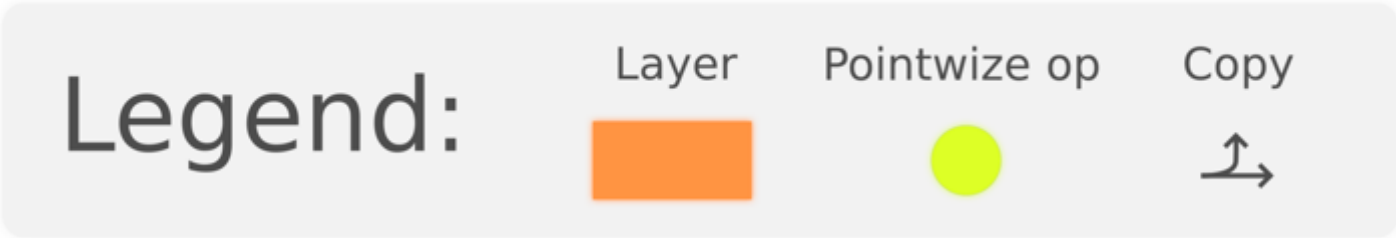
Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight

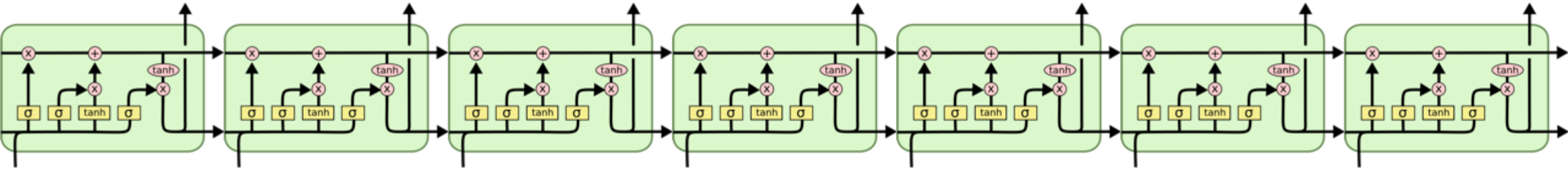
Long-short Term Memory

- Keeps a long-term memory.
- Similar to ResNet but jumping over layers.



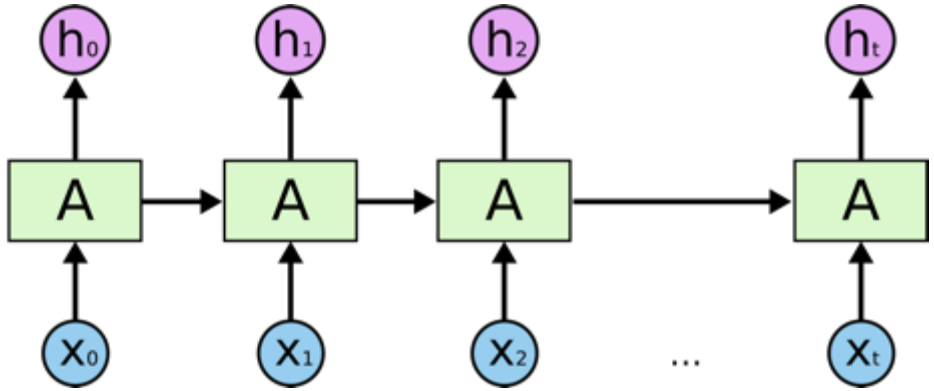


Long-short Term Memory



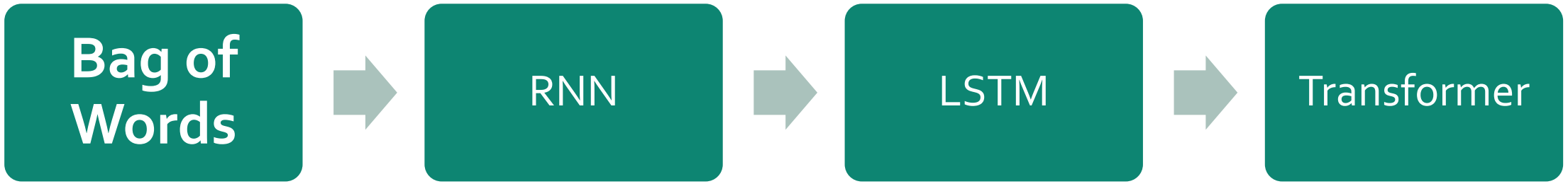
LSTM's limitations

- Difficult to train.
- Very long gradient paths.
  - LSTM on 100-word doc has gradients 100-layer network.
- Transfer learning never really worked.
- Needs specific labelled dataset for every task.



Transformers & Muppets (3/3)

The next big thing, for now



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

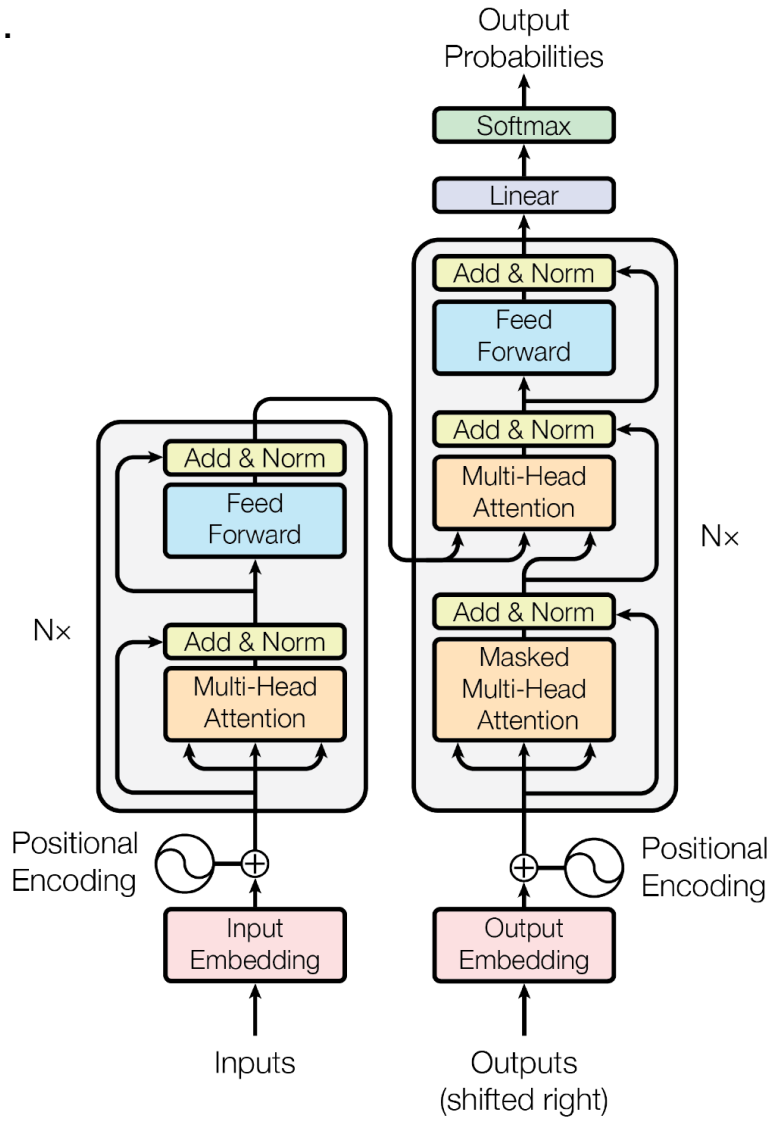
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or

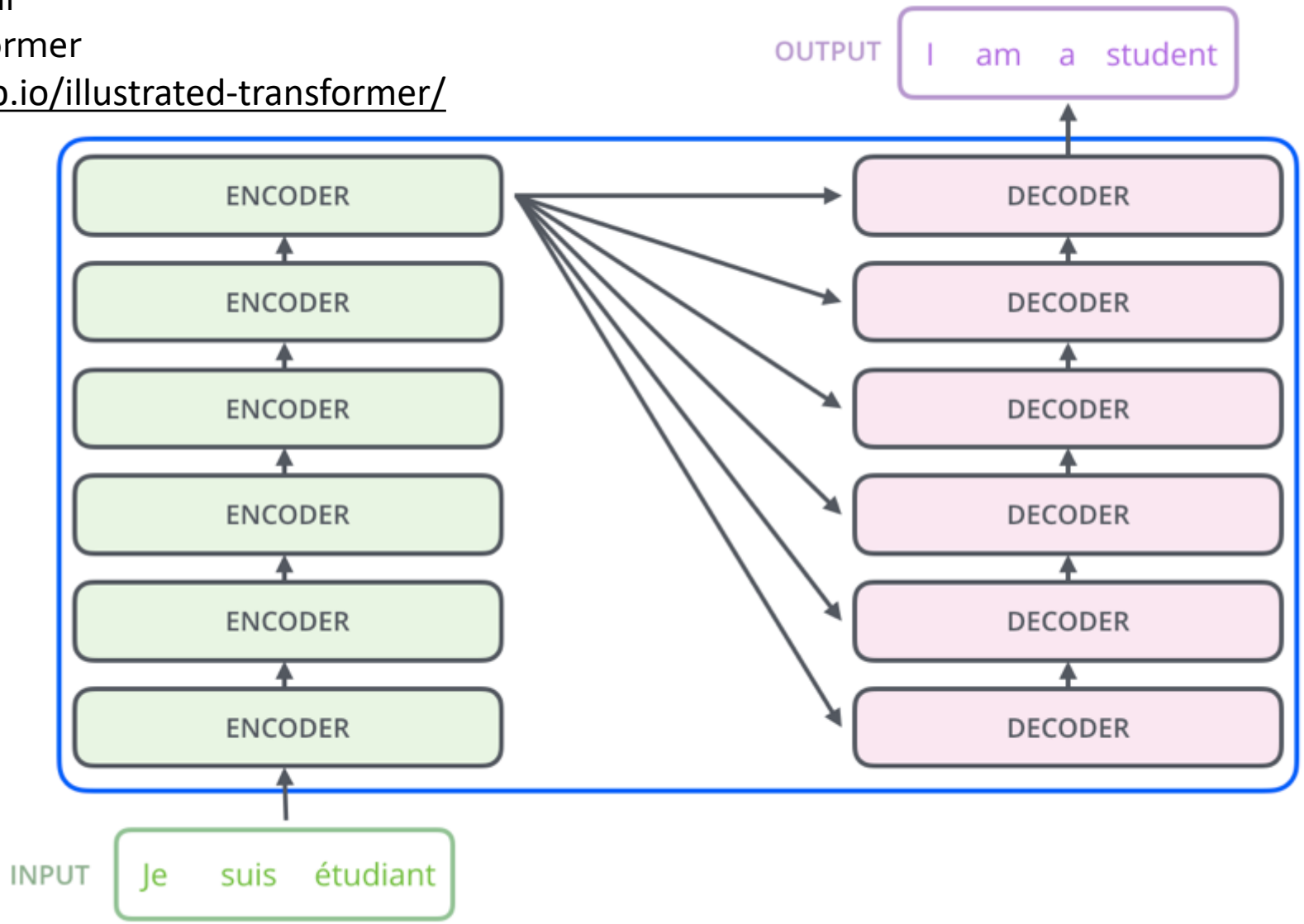
Encoder-Decoder

Ashish Vaswani et. Attention Is All You Need. NIPS 2017.



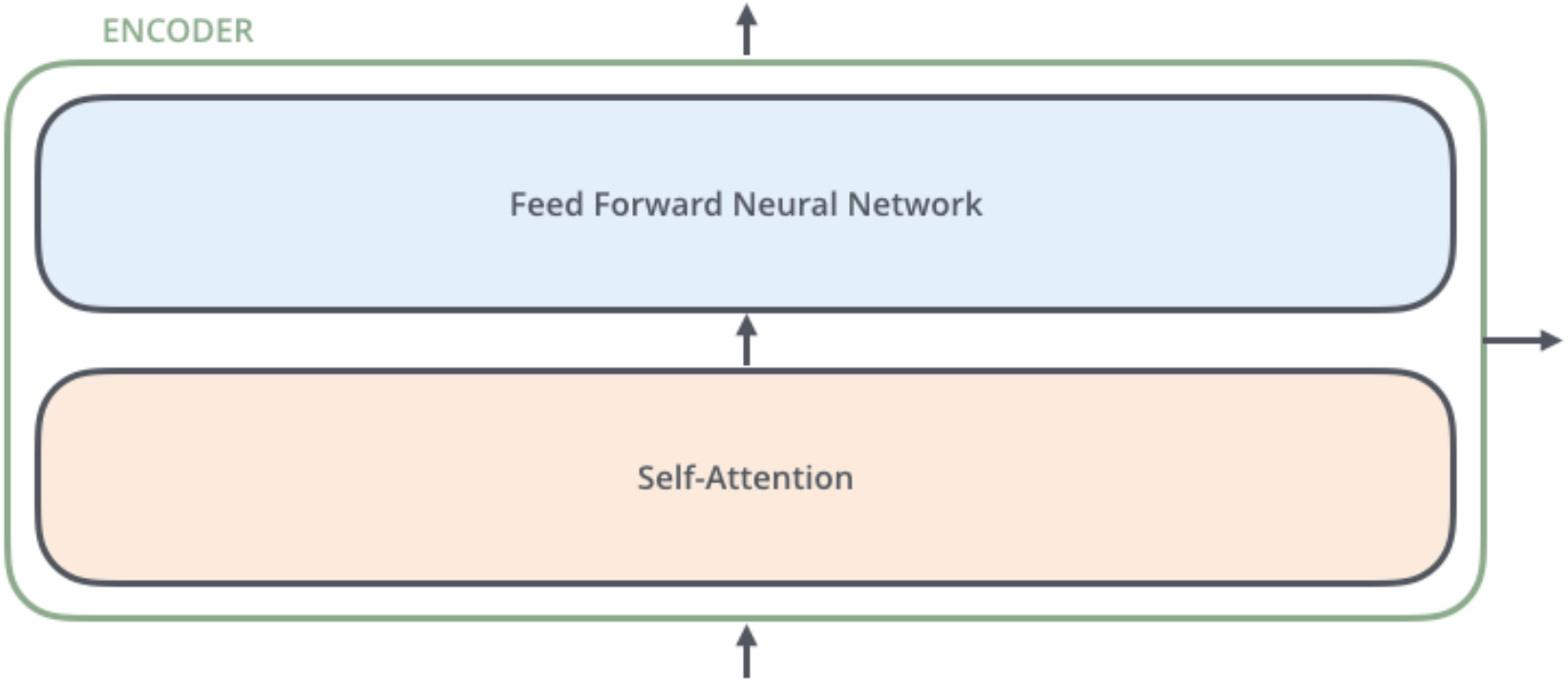
Encoder-Decoder

Figure by: Jay Alammar
The Illustrated Transformer
http://jalammar.github.io/illustrated-transformer/



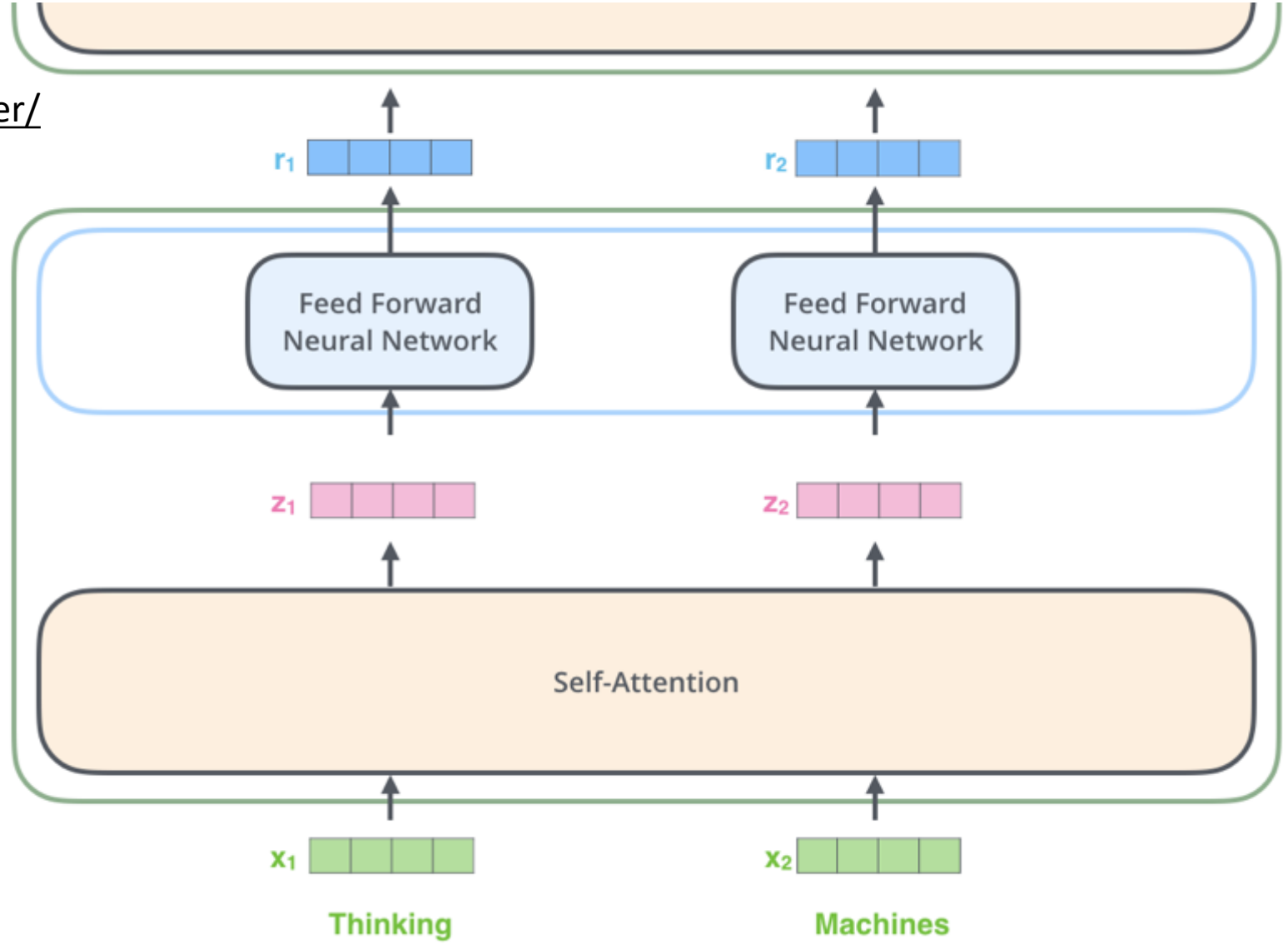
An Encoder Block: same structure, different parameters

Figure by: Jay Alammar
The Illustrated Transformer
http://jalammar.github.io/illustrated-transformer/



Encoder

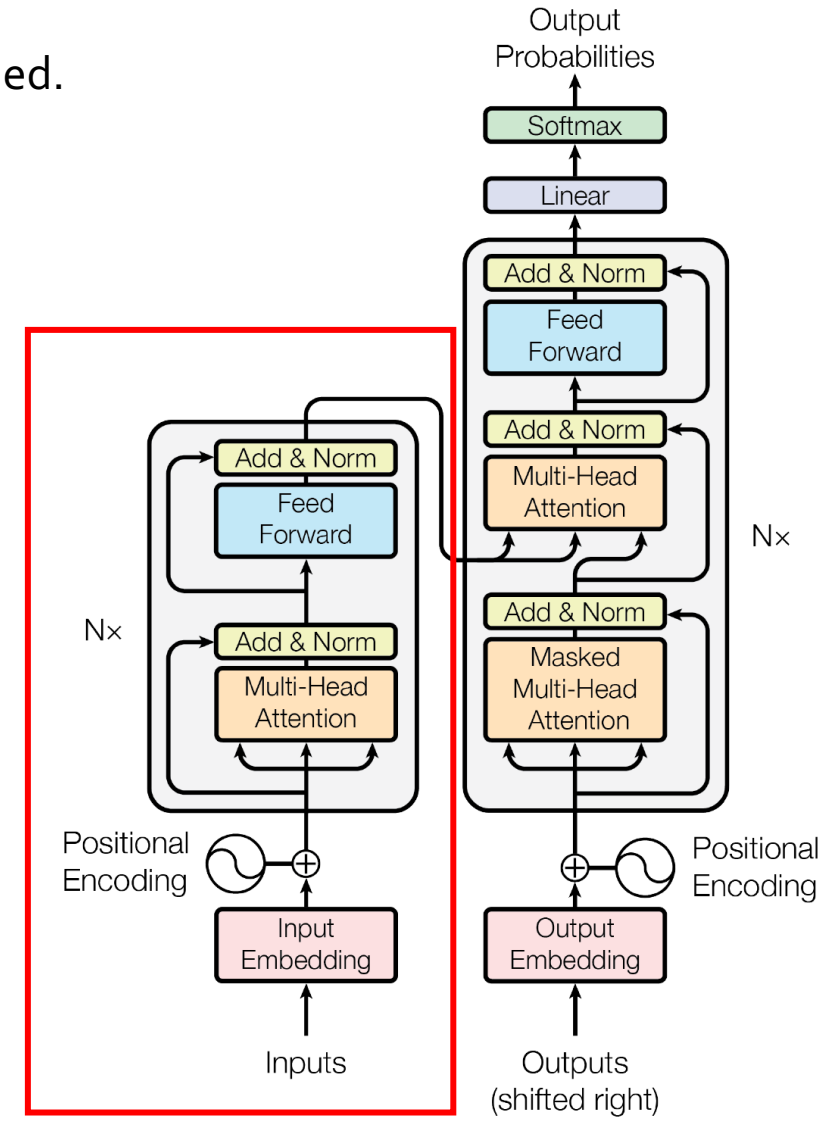
Figure by: Jay Alammar
The Illustrated Transformer
http://jalammar.github.io/illustrated-transformer/



Note: The FFNN is independent for each word.
Hence can be parallelized.

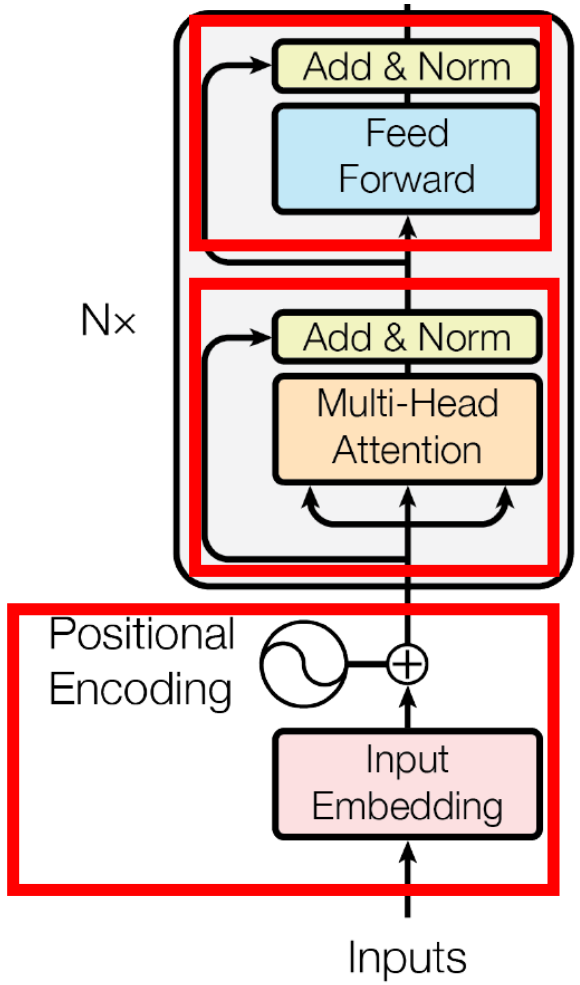
Encoder-Decoder

Ashish Vaswani et. Attention Is All You Need. NIPS 2017.



Encoder

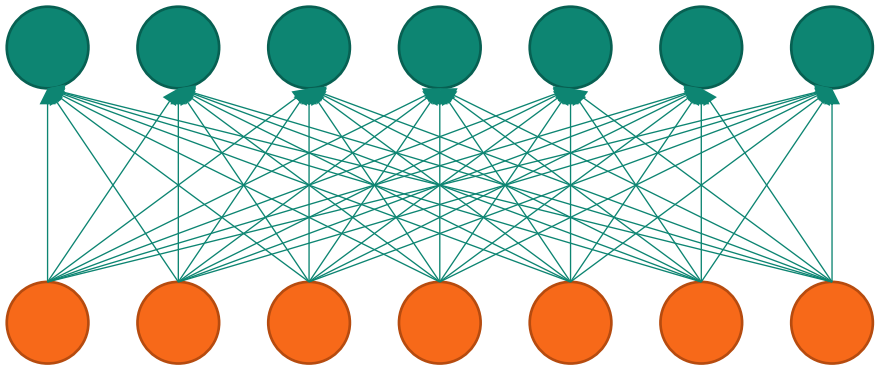
Ashish Vaswani et. Attention Is All You Need. NIPS 2017.



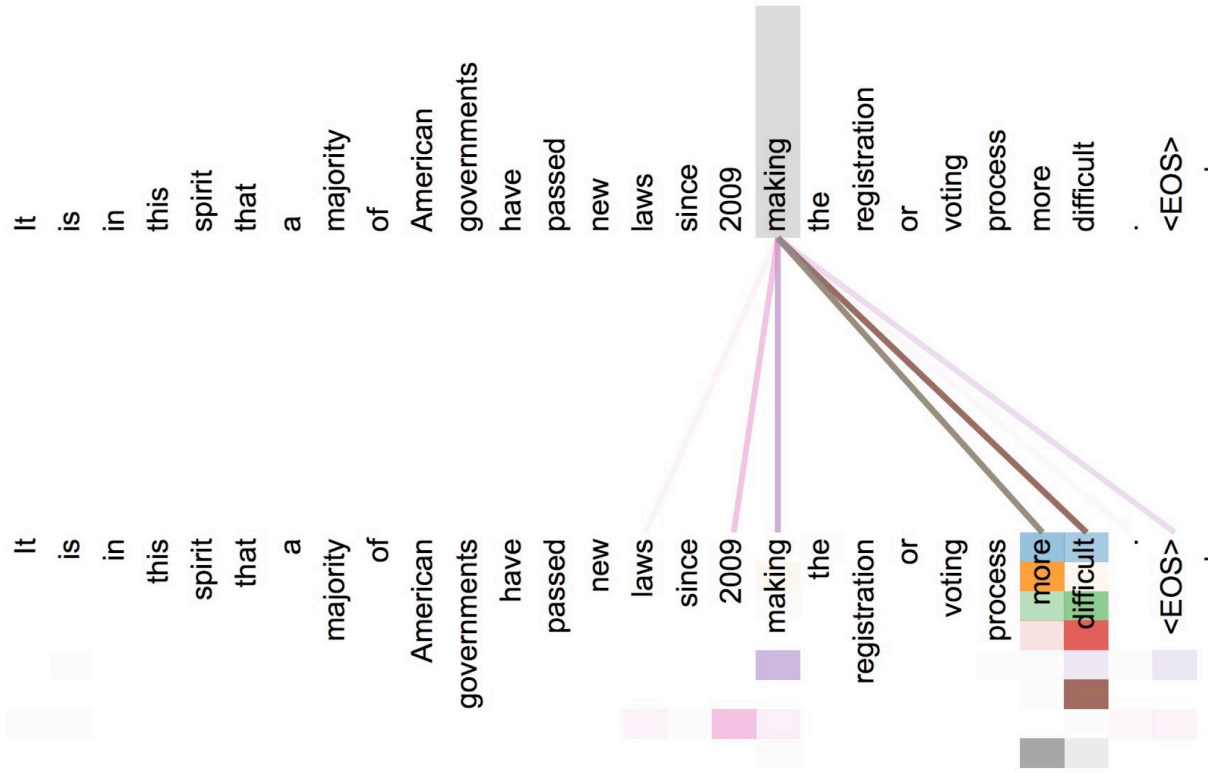
- Self-attention
- Multi-Head Self-Attention

Self-Attention Mechanism

- All-to-all comparison.
  - Each layer is $O(N^2)$ for sequence of length N - **self attention**.
- Every output is a weighted sum of every input.
  - The weighting is a function to learn.



Relevance scores from each input to output



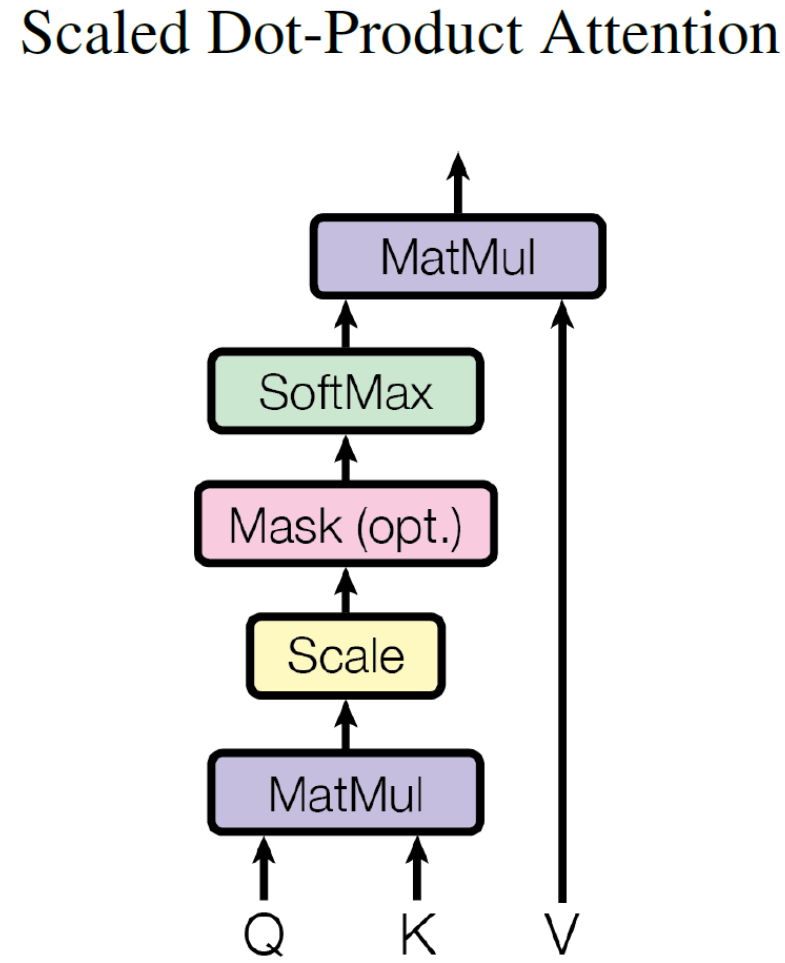
query[17] # “making”

key[24] # “difficult”

Relevance[17,24]=query[17] * key[24]
relevance of difficult to making

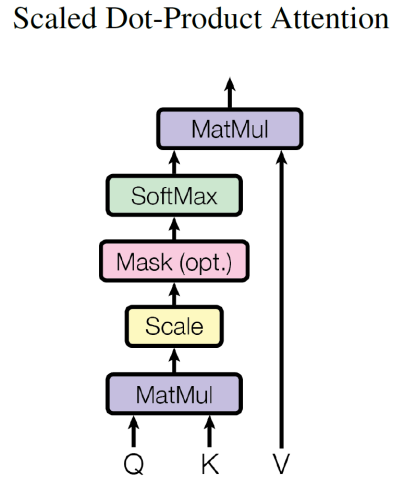
Attention

- Q: Query (output token)
- K: Key (input token)
- Relevance = Q*K
- V: Value (input token)
- Out = softmax(relevance)*V



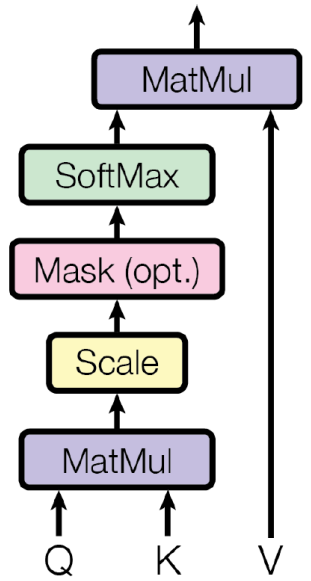
Attention in “pseudo-code”

```
def attention(self, X_in:List[Tensor]):
    # For every token transform previous layer’s out
    for i in rang(self.sequence_lenght):
        query[i] = self.Q * X_in[i]
        key[i]   = self.K * X_in[i]
        value[i] = self.V * X_in[i]
```



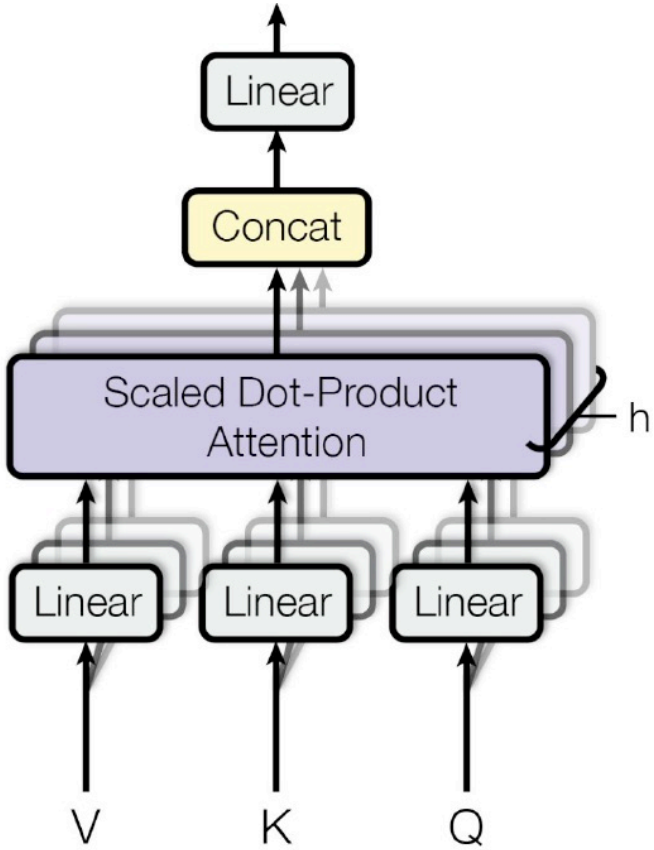
```
def attention(self, X_in:List[Tensor]):
        # For every token transform previous layer's out
        for i in rang(self.sequence_lenght):
                query[i] = self.Q * X_in[i]
                key[i]   = self.K * X_in[i]
                value[i] = self.V * X_in[i]

        # Compute output values, one at a time
        for i in rang(self.sequence_lenght):
                this_query = query[i]
                # how relevance is each input to this output?
                for j in rang(self.sequence_lenght):
                        relevance[j] = this_query * key[j]
                # normalize relevance score to sum to 1
                relevance = scaled_softmax(relevance)
                # compute a weighted sum of values
                out[i]   = 0 # out[i] is a vector
                 for j in rang(self.sequence_lenght):
                        out[i] += relevance[j] * value[j]
        return out
```

Scaled Dot-Product Attention

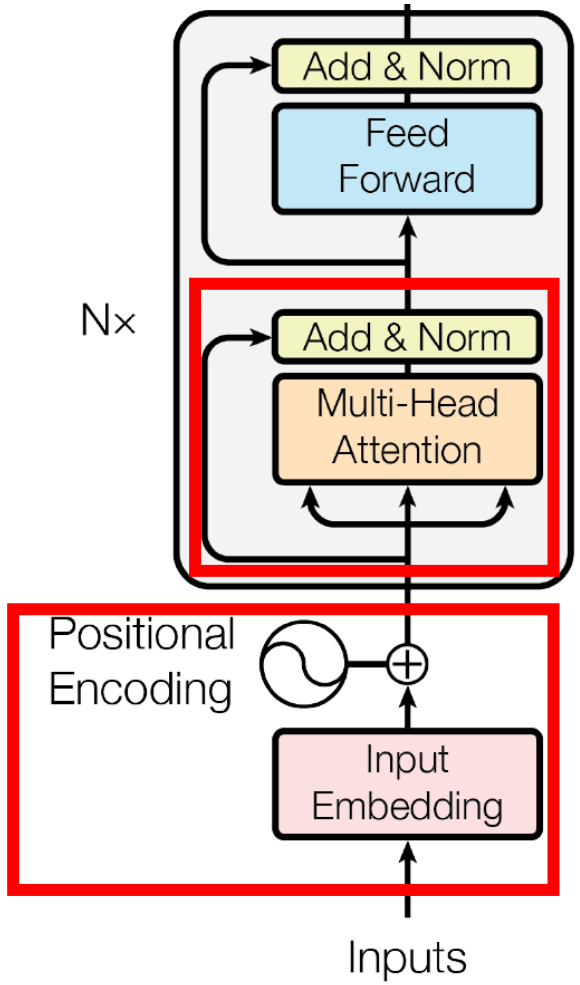


Multi-Headed Attention

- Clever, important innovation.
  - Not that hard.
- Just do that same thing 8 times with different Q,K,V matrices.
- Let the network learn 8 different semantic meanings of attention.
  - E.g., One grammar, one for vocabulary, one for conjugation, etc.
  - Very flexible mechanism for sequence processing.

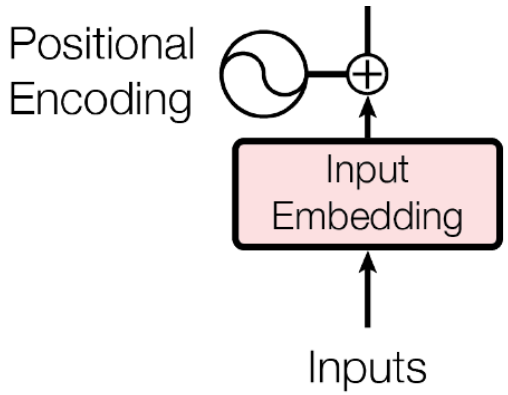


Encoder



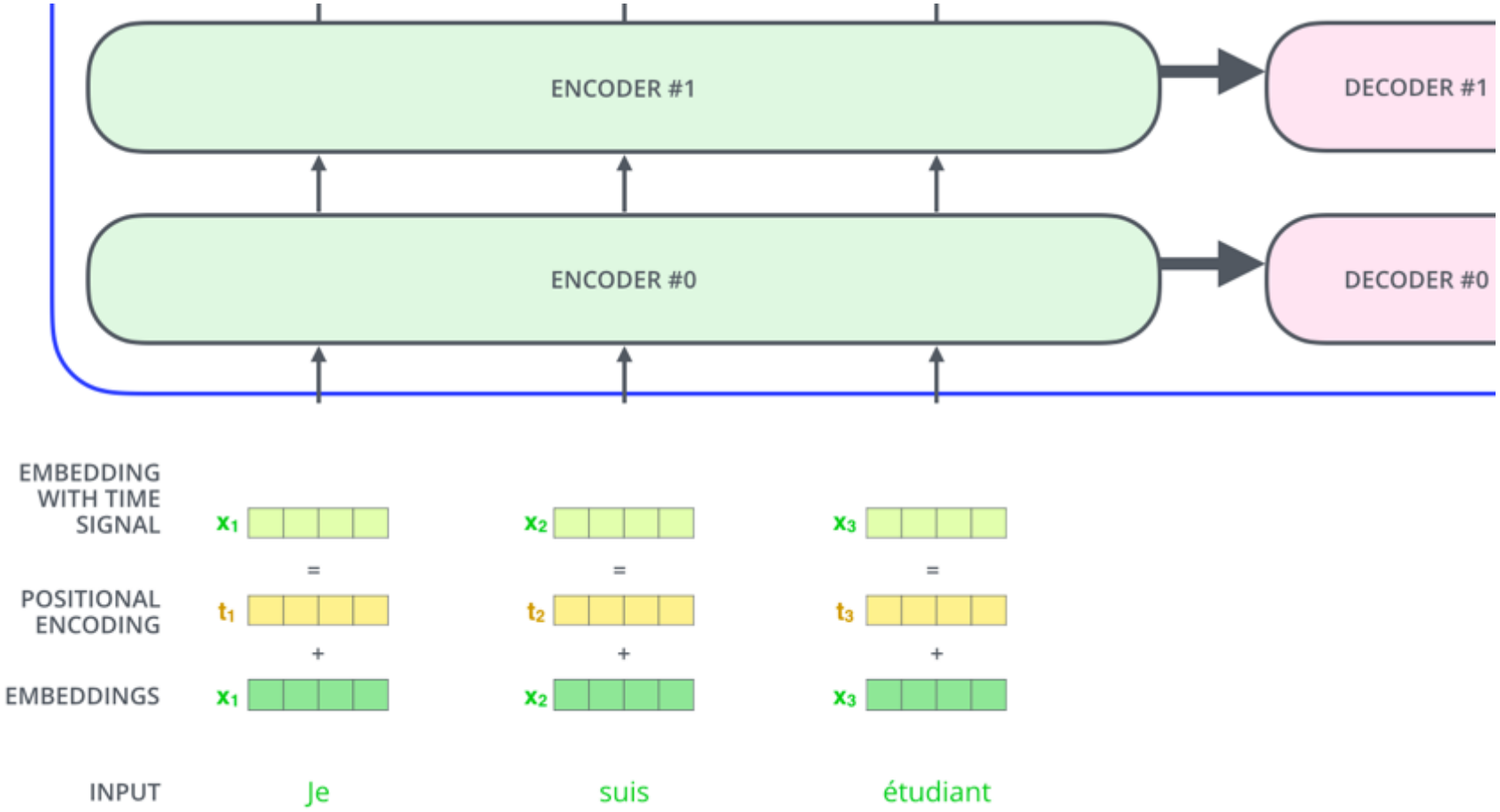
Position encoding

- Attention is “bag of words”.
- Input layer: add a word embedding and position embedding.
- Position can be either learned or fixed.
- Fixed allows extrapolating to longer sequences.



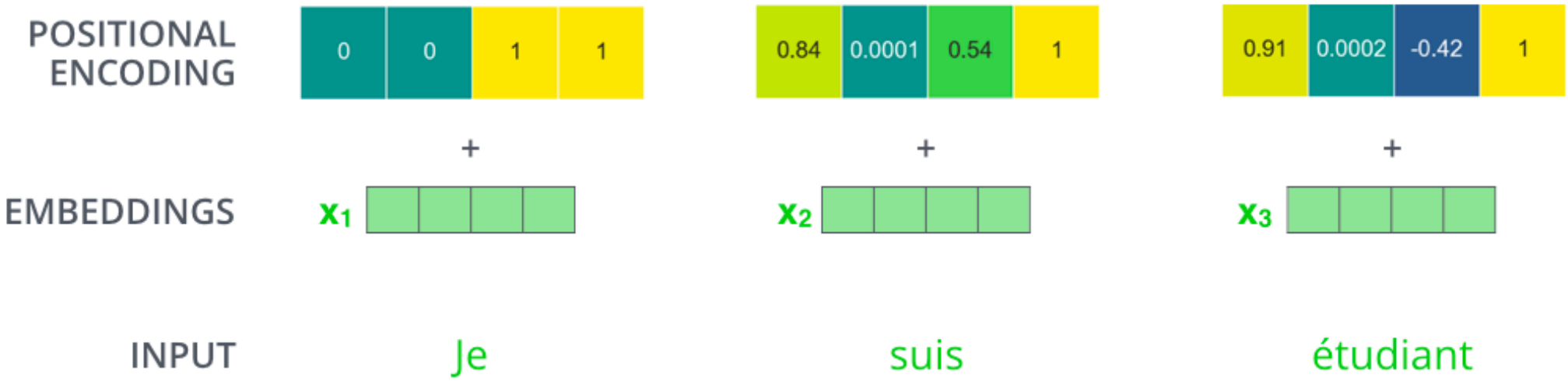
Position encoding

Figure by: Jay Alammar
The Illustrated Transformer
http://jalammar.github.io/illustrated-transformer/



Position encoding

- If we assumed the embedding has a dimensionality of 4, the actual positional encodings would look like this:



All together!

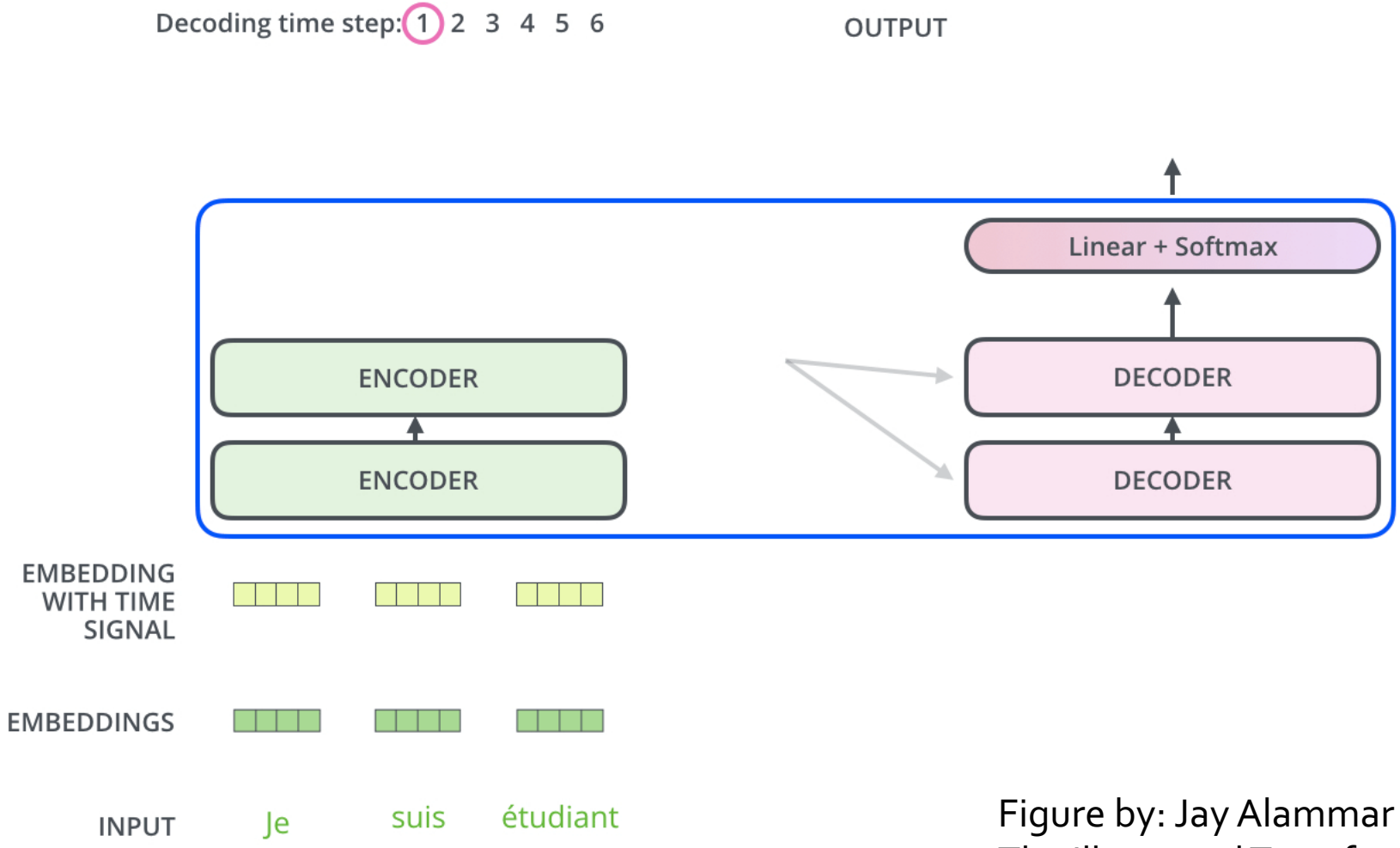
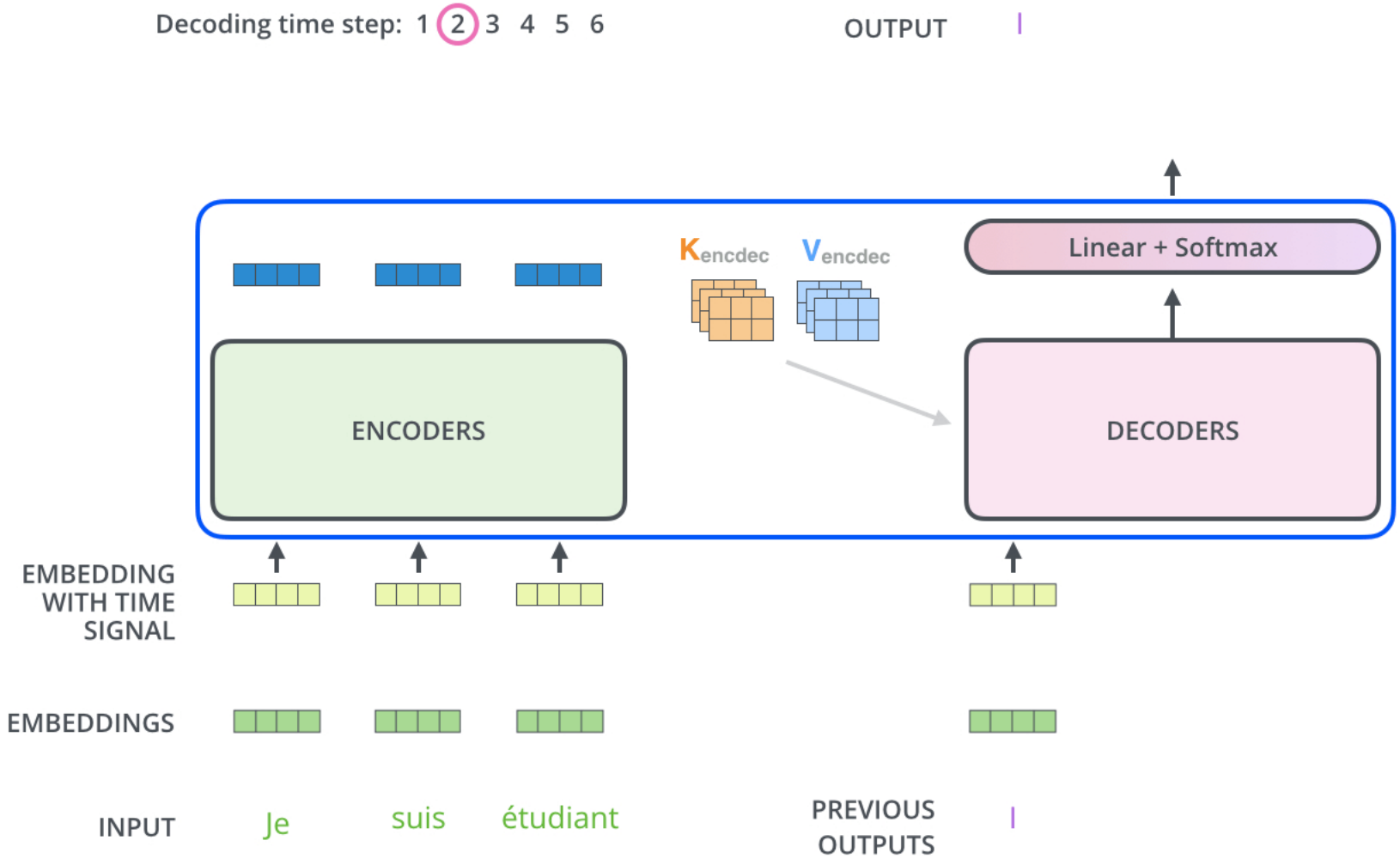


Figure by: Jay Alammar
The Illustrated Transformer
http://jalammar.github.io/illustrated-transformer/

All together!



Why Transformers are awesome?

- All-to-all comparisons can be done fully parallel.
  - GPUs change the game for compute!
  - N^2 but extra parallel operations can be “free”.
  - (RNN/LSTM must be computed in sequence per token).

Transformer results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Literature & Resources for Transformers

- Vaswani et al. Attention is all you need. 2017.
- Resources:
  - Leo Dirac, LSTM is dead. Long Live Transformers! https://www.youtube.com/watch?v=S27pHKBEp30&t=1073s
  - https://nlp.seas.harvard.edu/2018/04/03/attention.html (Excellent explanation of transformer model with codes.)
  - Jay Alammar, The illustrated transformer (from which I borrowed many pictures):
  - http://jalammar.github.io/illustrated-transformer/
  - Kate Logninova: Attention in NLP, summarizes all sorts of attentions. Can somebody present this and related literature? https://medium.com/@joealato/attention-in-nlp-734c6fa9d983
  - LSTM is dead. Long Live Transformers!

Questions?

Slides available on:
https://personal-sites.deakin.edu.au/~mohamedb/teaching.html