APPROACHES: DEEP LEARNING

Luísa Coheur

TÉCNICO LISBOA

inesc id lisboa

OVERVIEW

- Learning objectives
- Topics
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Sequence to Sequence Models
  - Autoencoders
  - Generative Adversarial Networks
  - Attention
  - Transformers
- Key takeaways
- Suggested readings

LEARNING OBJECTIVES

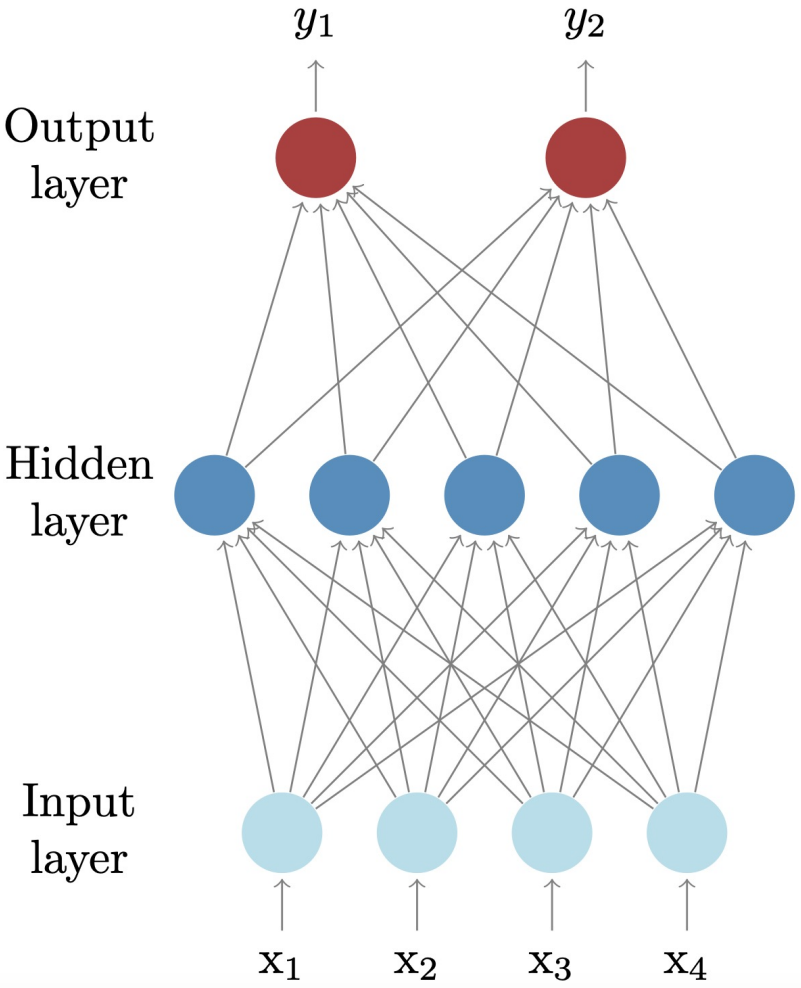
LEARNING OBJECTIVES

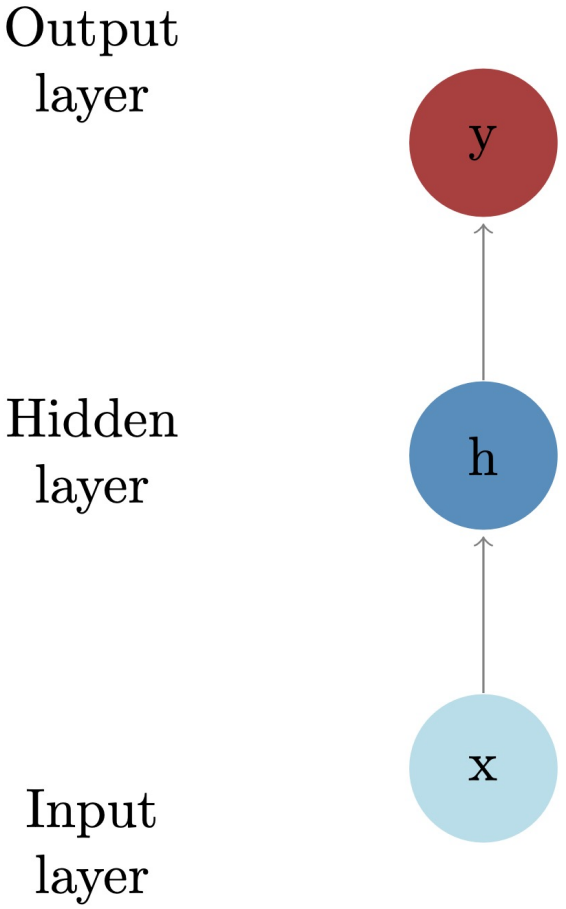
- After this class, students should be able to:
  - Learn to identify and describe different deep learning architectures and understand how these have progressed
  - Define several concepts related to Deep Learning

TOPICS

FEEDFORWARD NEURAL NETWORKS (FFNN)

- Remember:

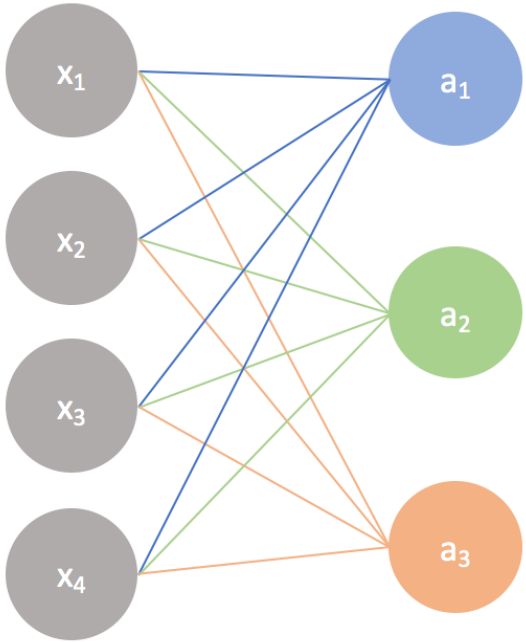




FEED-FORWARD NEURAL NETWORKS (FFNN)

- Remember:

Input layer Output layer



A simple neural network

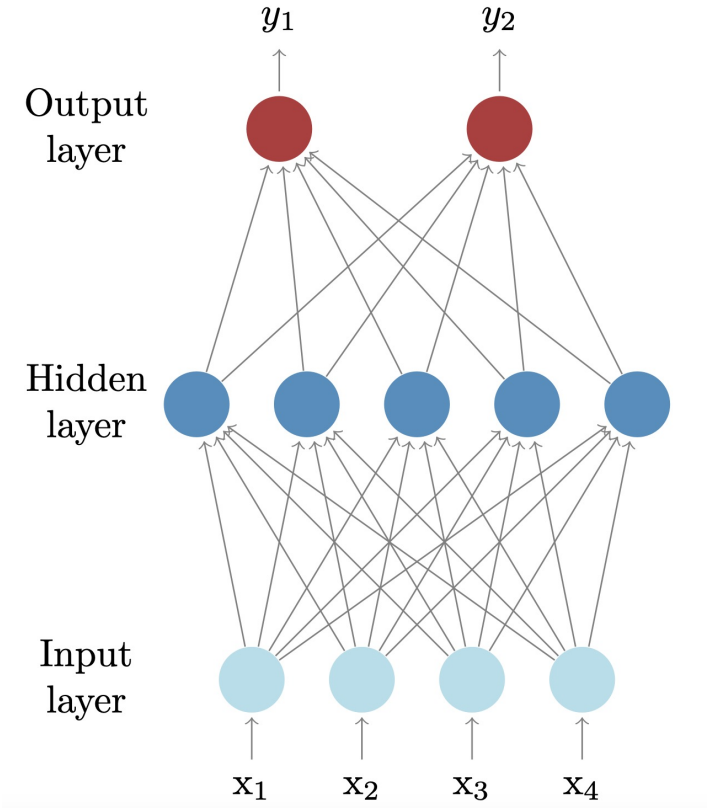
$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

FEED-FORWARD NEURAL NETWORKS

- Remember:
  - Initialization of the weights
  - Take input and process it (forward propagation)
  - Result is compared with the desired output => the error is obtained
  - Try to minimize the error, by updating weights in the network (backward propagation)
    - A learning rate is used

FEEDFORWARD NEURAL NETWORKS

- Problems with FFNN
  - inputs are independent from each other; the same for outputs
  - Unable to handle variable length inputs



FEEDFORWARD NEURAL NETWORKS

But… that is mandatory in NLP

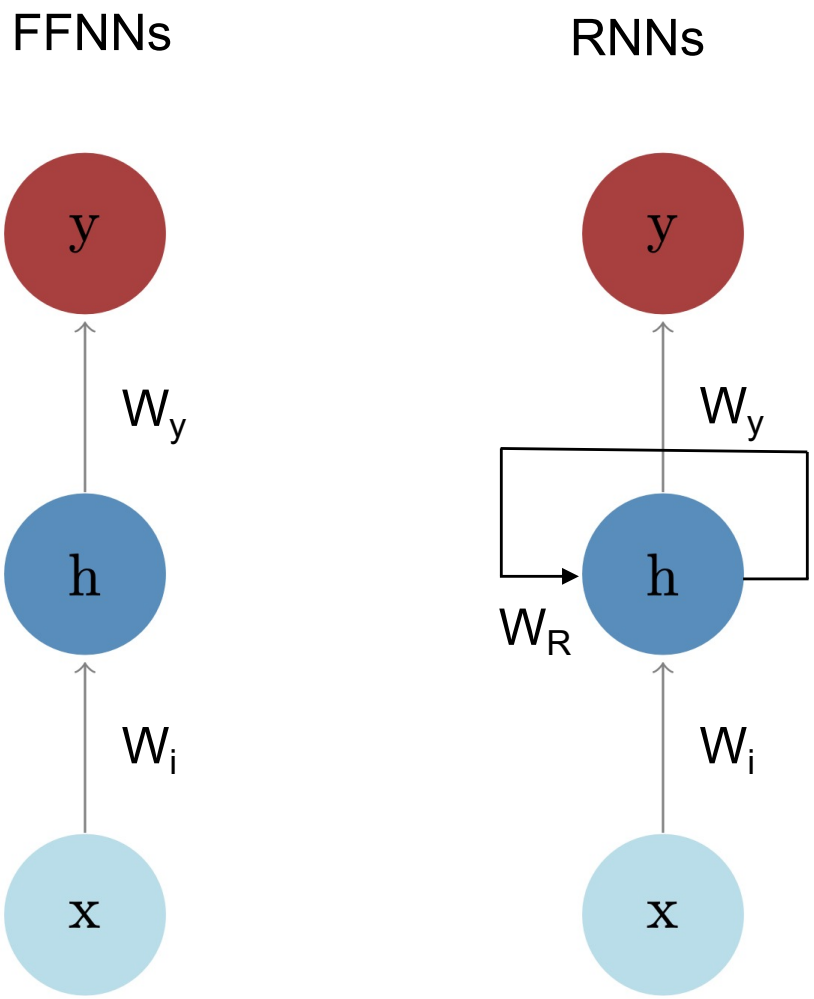
You are right, Ana!

If you want to predict the next word in a sentence (for instance) you better know which words came before it; also, sentences do not have a pre-defined number of words…

OVERVIEW

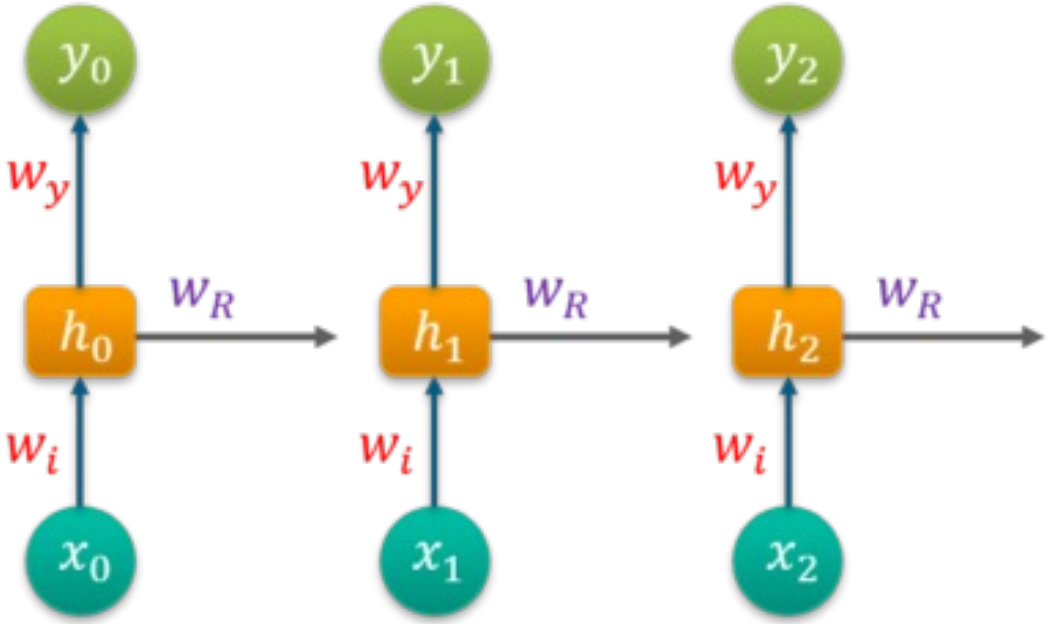
- Learning objectives
- Topics
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Sequence to Sequence Models
  - Autoencoders
  - Generative Adversarial Networks
  - Attention
  - Transformers
- Key takeaways
- Suggested readings

RECURRENT NEURAL NETWORKS (Elman 1990)



RECURRENT NEURAL NETWORKS

- At time step t
  - x_t is the input, h_t (the "memory") the hidden state and y_t the output
- The RNN shares the same parameters (W_i, W_y, W_R) across all steps (it uses the same weights for each step)



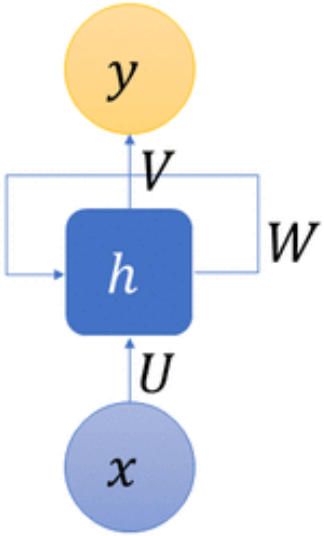
$$h^{(t)} = g_h\left(w_i x^{(t)} + w_R h^{(t-1)} + b_h\right)$$

$$y^{(t)} = g_y\left(w_y h^{(t)} + b_y\right)$$

Image from https://www.edureka.co

RECURRENT NEURAL NETWORKS

- Backpropagation needs to be adapted:
  - Backpropagation through time



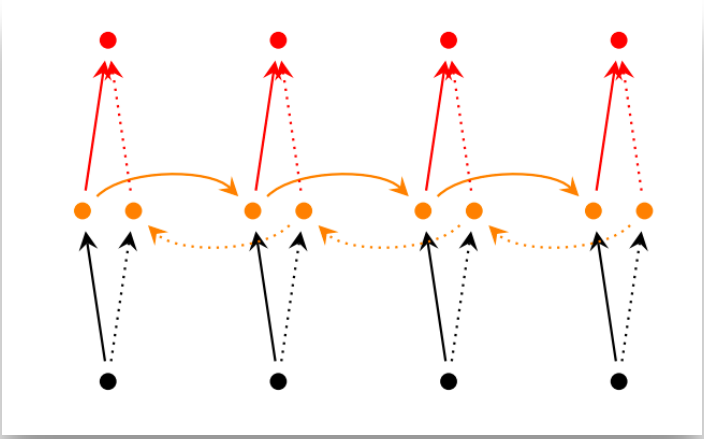
https://discuss.pytorch.org/t/implementing-backpropagation-through-time/69765

RECURRENT NEURAL NETWORKS (EXTENSIONS)

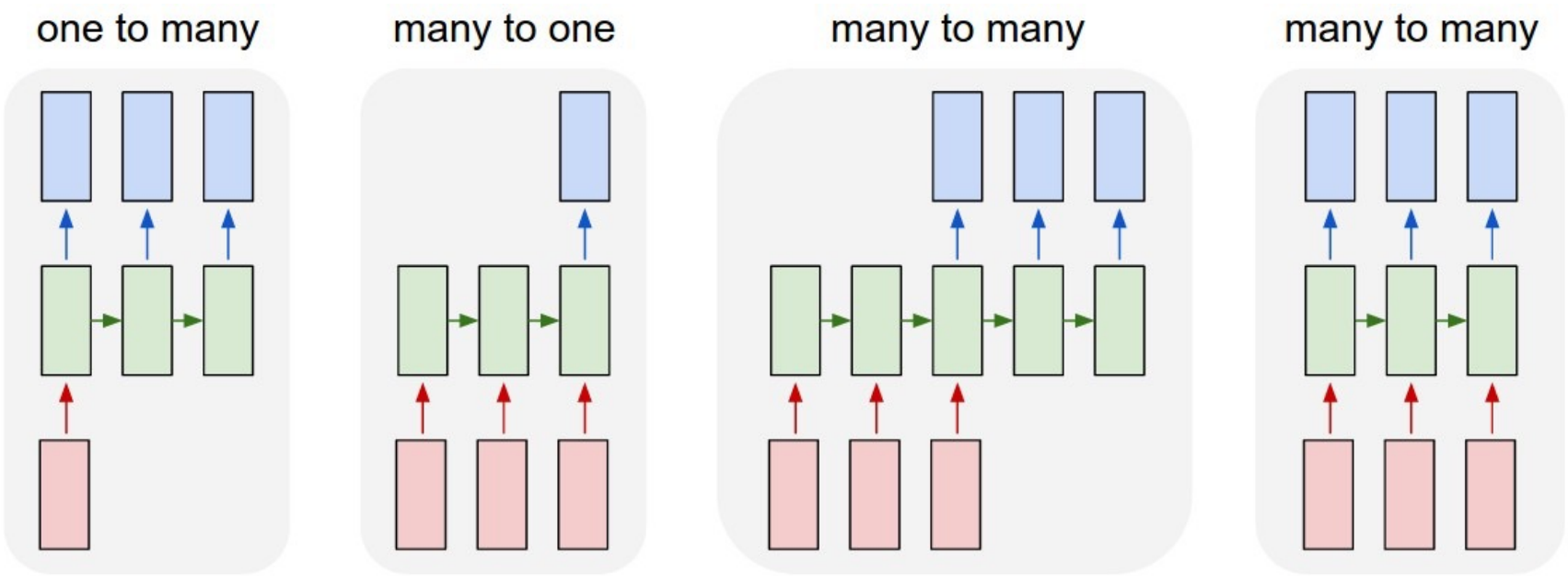
- In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps
- Extensions:
  - Long short-term memory (LSTM) (Sepp Hochreiter and Jürgen Schmidhuber, 1997)
  - Gated Recurrent Unit (GRU) (Cho, et al. in 2014)

RECURRENT NEURAL NETWORKS (EXTENSIONS)

- Bidirectional RNNs: output at time t may not only depend on the previous elements in the sequence, but also future elements
  - For example, to predict a missing word in a sequence you may want to look at both the left and the right context.



RECURRENT NEURAL NETWORKS (APPLICATIONS IN NLP)

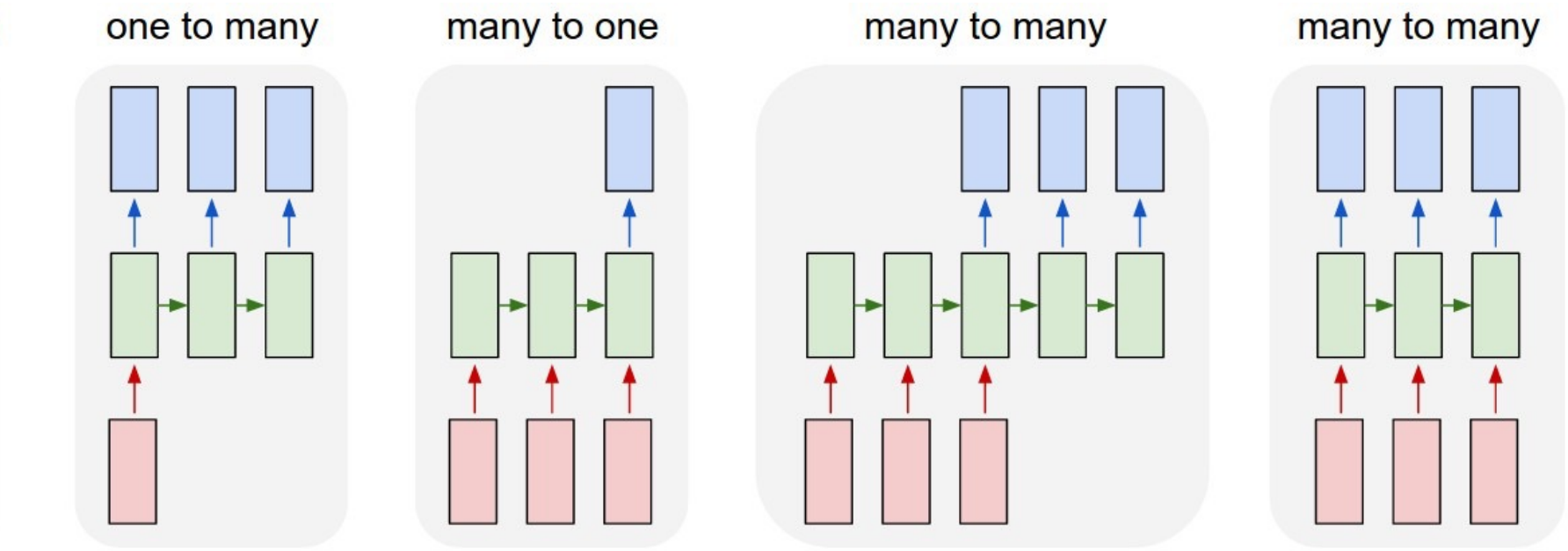


http://karpathy.github.io/2015/05/21/rnn-effectiveness/

ACTIVE LEARNING MOMENT: THINK-PAIR-SHARE

EXERCISE

Give examples of NLP-related applications that follow in these architectures:

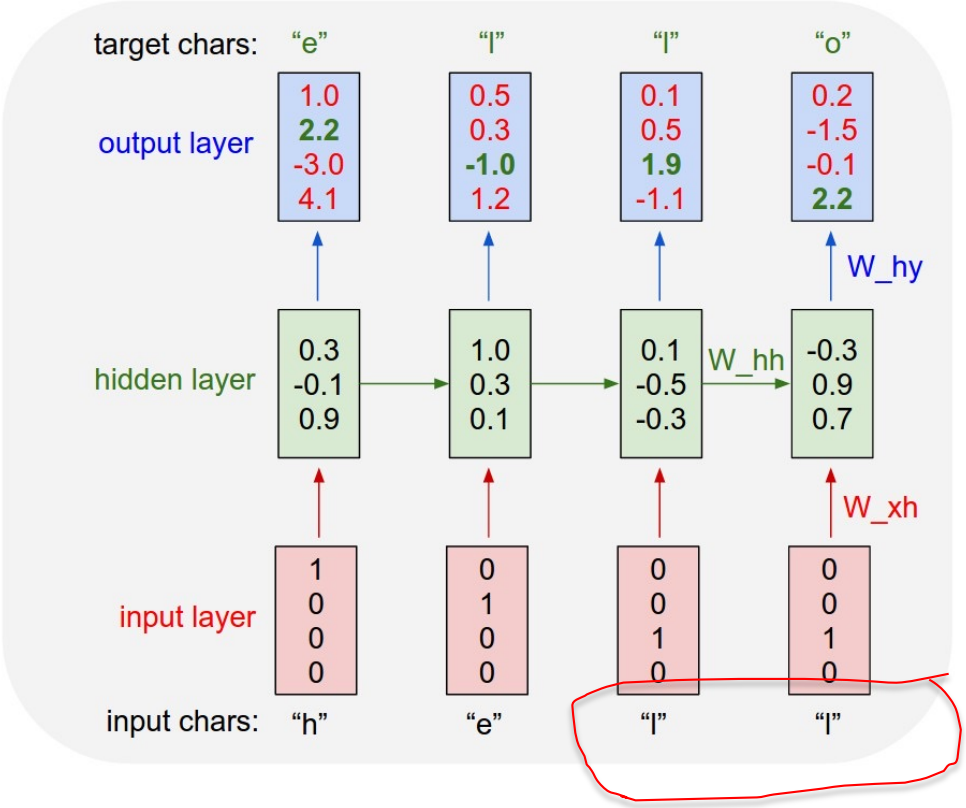


CHARACTER-BASED LANGUAGE MODELS

- Character level language models learn to generate the next character
  - model the probability distribution of the next character in the sequence, being given a sequence of previous characters

CHARACTER-BASED LANGUAGE MODELS

- The following examples are taken from "The unreasonable effectiveness of Recurrent Neural Networks". This was in 2015!!!



Test time: feed a character and feed next with the resulting character

First target: l; Second target: o
Context/memory is important!

CHARACTER-BASED LANGUAGE MODELS

- Shakespeare example: all the works of Shakespeare (4.4MB file). 3-layer RNN, 512 hidden nodes on each layer. After a few hours of training, obtained:

"Second Lord: They would be ruled after this chamber, and my fair nues begun out of the fact, to be conveyed, Whose noble souls I'll have the heart of the wars.
Clown: Come, sir, I will make did behold your worship.
VIOLA: I'll drink it."

CHARACTER-BASED LANGUAGE MODELS

- Wikipedia example: Hutter Prize 100MB dataset of raw Wikipedia. Trained a LSTM. After a few hours of training, obtained:

> *"Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. ."*

CHARACTER-BASED LANGUAGE MODELS

- Latex example: Raw Latex source file of algebraic stacks/geometry book (a 16MB file). Trained a multilayer LSTM:

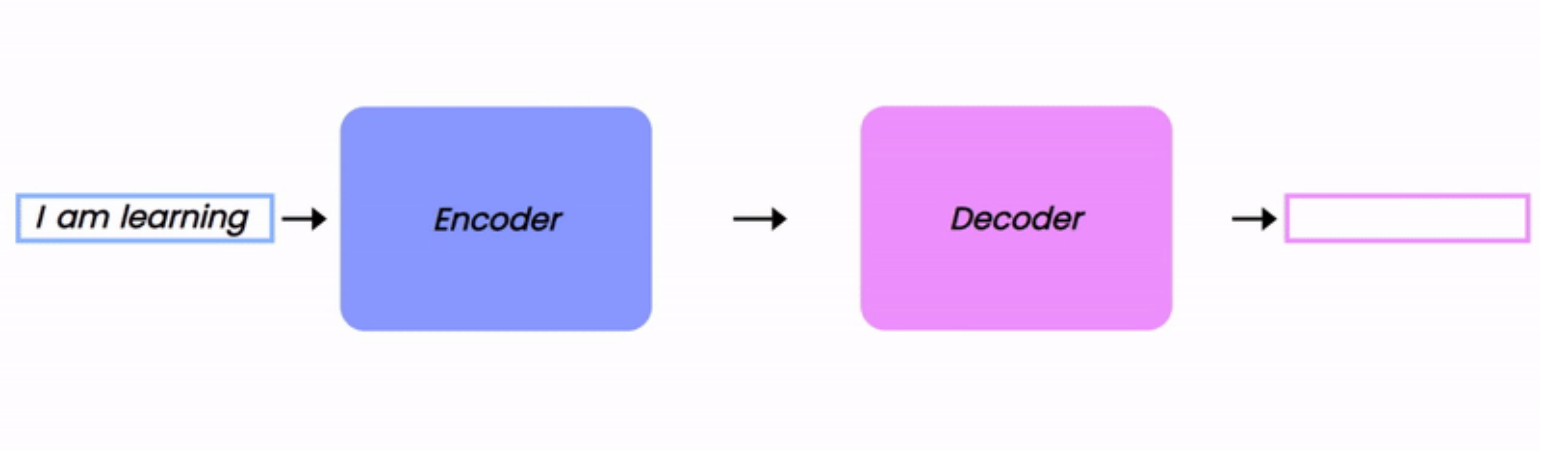
  - "\begin{proof} We may assume that $\mathcal{I}$ is an abelian sheaf on $\mathcal{C}$. \item Given a morphism $\Delta : \mathcal{F} \to \mathcal{I}$ is an injective and let $\mathfrak q$ be an abelian sheaf on X. Let $\mathcal{F}$ be a fibered complex. Let $\mathcal{F}$ be a category. \begin{enumerate} \item \hyperref[setain-construction-phantom]{Lemma} \label{lemma-characterize-quasi-finite}"

OVERVIEW

- Learning objectives
- Topics
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Sequence to Sequence Models
  - Autoencoders
  - Generative Adversarial Networks
  - Attention
  - Transformers
- Key takeaways
- Suggested readings

SEQUENCE TO SEQUENCE MODELS (S2S, seq2seq, Google, 2014)

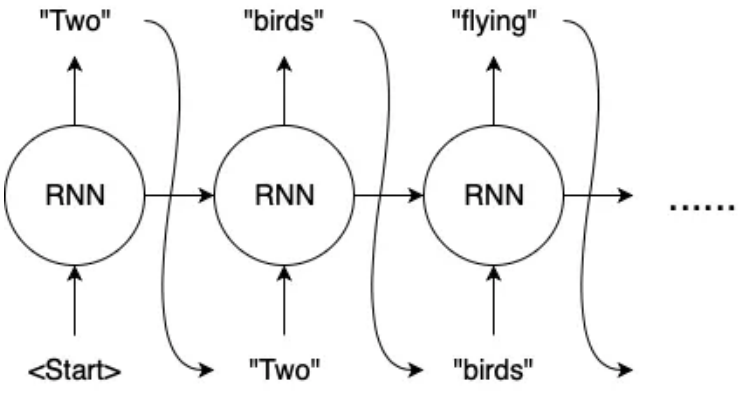
- Seq2Seq models have an encoder-decoder architecture:
  - the encoder processes the input and encodes the information into a context vector (of a fixed length)
    - Idea: context vector is expected to be a good summary of the entire input
  - The decoder is then initialized with this context vector and starts generating the transformed output



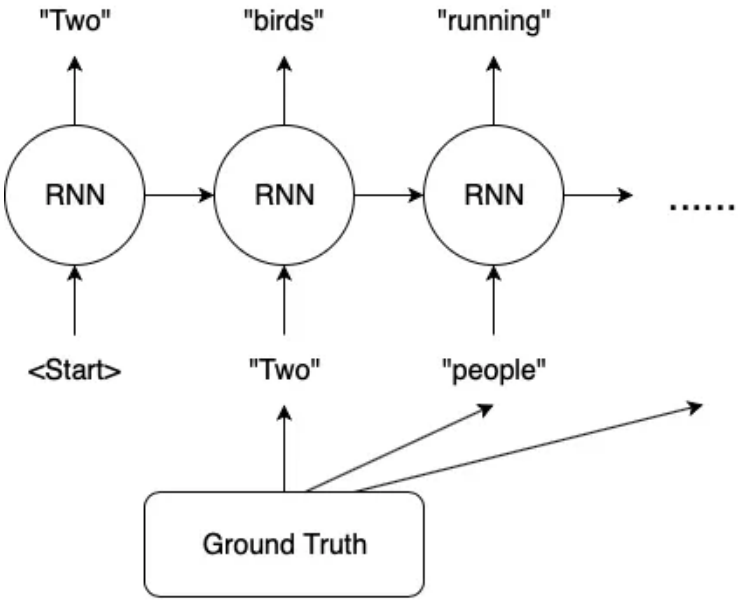
https://medium.com/geekculture/neural-machine-translation-using-sequence-to-sequence-model-164a5905bcd7

BY THE WAY… TEACHER FORCING (INTERESTING RELATED CONCEPTS)

- Teacher forcing: training technique used for sequence prediction tasks
  - the actual output from the training dataset at the current time step is used as the input at the next time step, rather than using the output generated by the model



Without Teacher Forcing



With Teacher Forcing

https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c

TEACHER FORCING (INTERESTING RELATED CONCEPTS)

- Benefits
  - The model learns faster because it is always guided by the correct sequences during training
  - Reduces error accumulation
- Drawbacks:
  - Over-reliance on teacher forcing might prevent the model from learning to recover from its own mistakes

TEACHER FORCING (INTERESTING RELATED CONCEPTS)

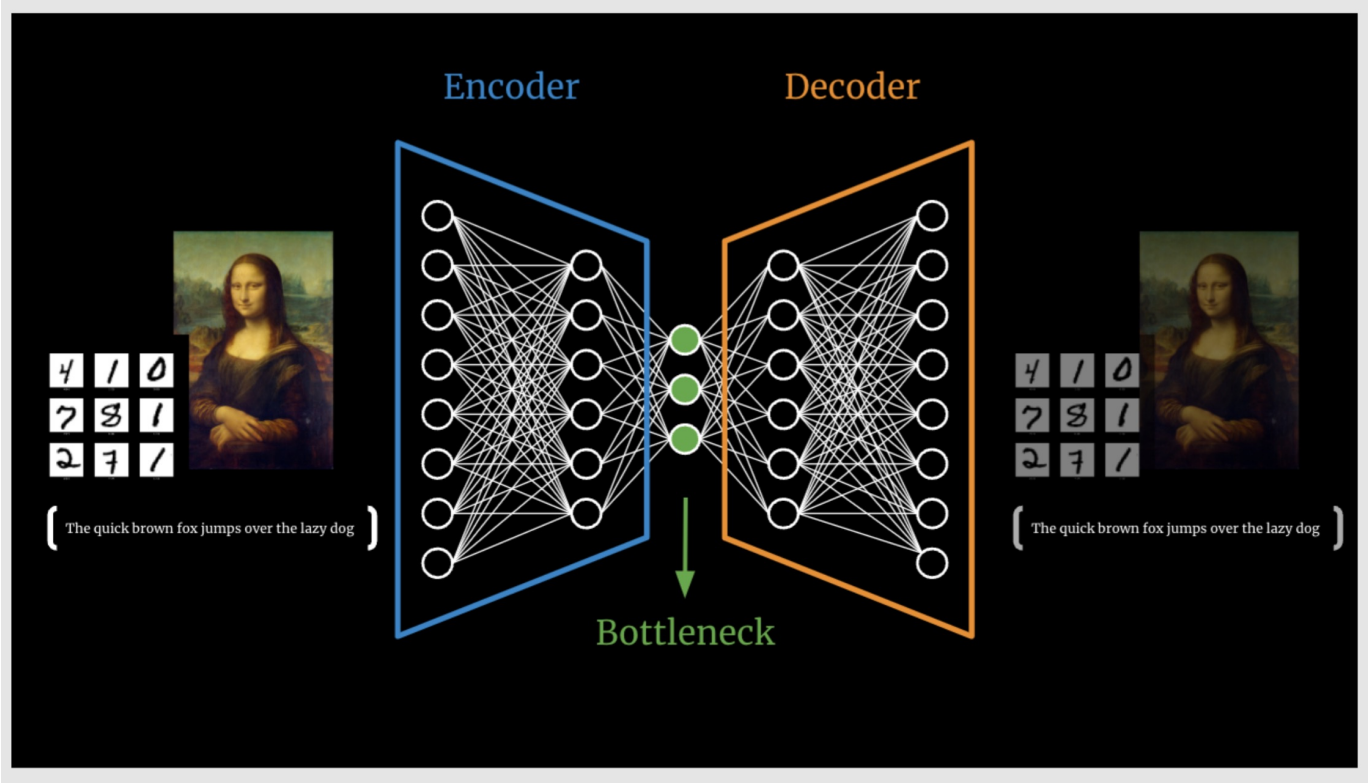
- Balancing Teacher Forcing:
  - Randomized Teacher Forcing: use teacher forcing randomly with a certain probability that can change over time
  - Scheduled Sampling: gradually decrease the use of teacher forcing as training progresses, allowing the model to learn from its own predictions

OVERVIEW

- Learning objectives
- Topics
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Sequence to Sequence Models
  - Autoencoders
  - Generative Adversarial Networks
  - Attention
  - Transformers
- Key takeaways
- Suggested readings

AUTOENCODERS

- Autoencoders learn a representation (encoding) for a set of data
  - keep the variations in the data required to reconstruct the input; ignore redundancies



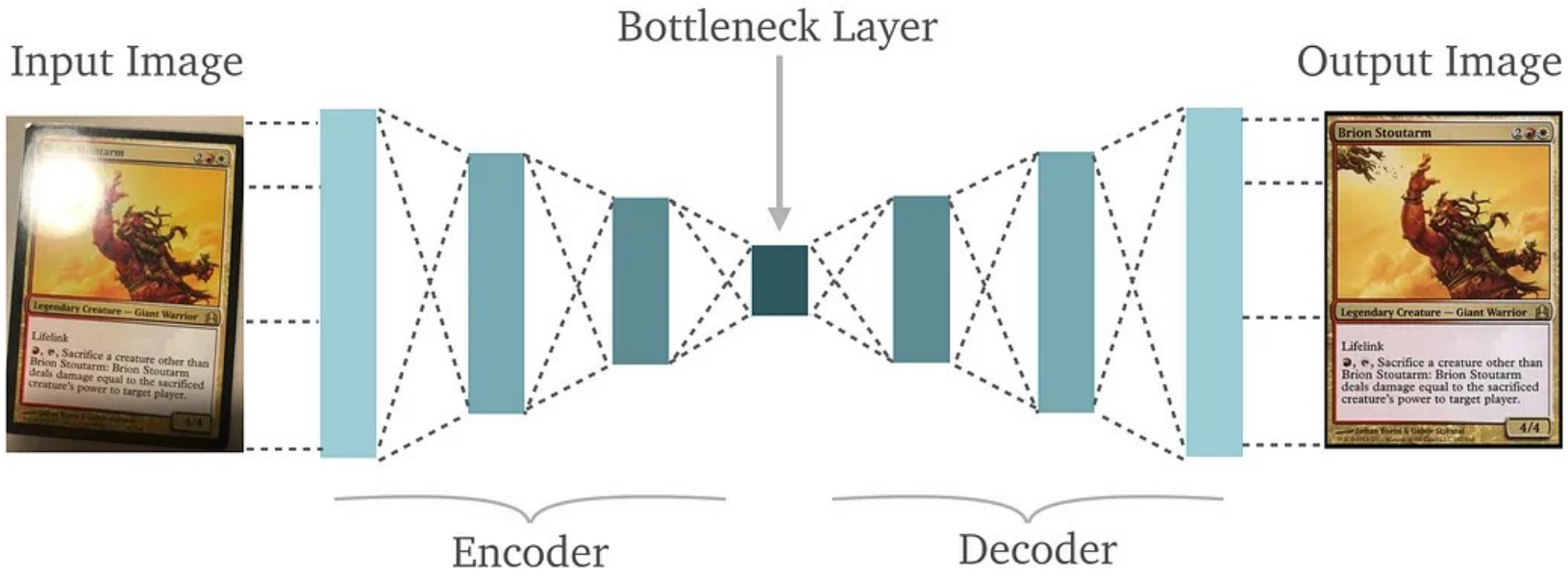
https://www.saberhq.com/blog/autoencoders

AUTOENCODERS

- Useful for representation learning and dimensionality reduction
  - the vector serving as a hidden representation compresses the data into a smaller number of dimensions
  - after training, only the encoder is used (the decoder is trashed)

AUTOENCODERS (INTERESTING APPLICATION)

- Denoising: during training, is presented with pairs of input data (noisy) and target data (clean).



https://medium.com/@sorenlind/a-deep-convolutional-denoising-autoencoder-for-image-classification-26c777d3b88e

OVERVIEW

- Learning objectives
- Topics
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Sequence to Sequence Models
  - Autoencoders
  - Generative Adversarial Networks
  - Attention
  - Transformers
- Key takeaways
- Suggested readings

GENERATIVE ADVERSARIAL NETWORKS (or GANs, Ian Goodfellow, 2014)

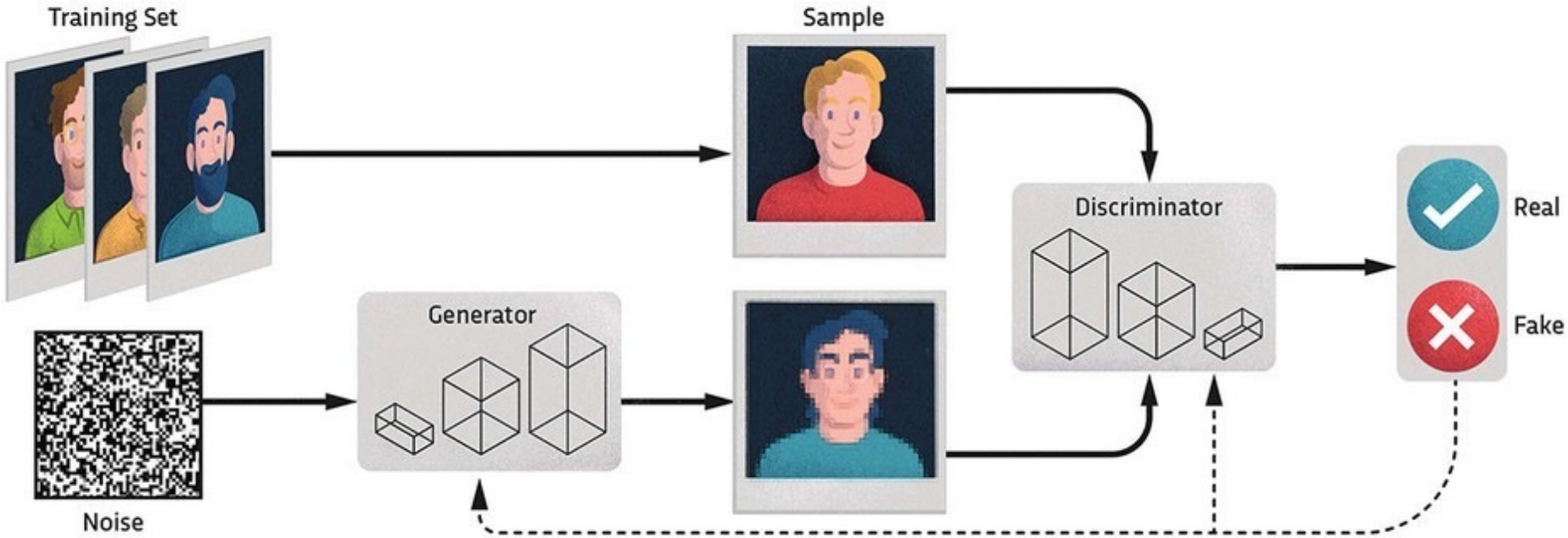
- A GAN is a deep neural net architecture that comprised two nets, pitting one against the other (thus "adversarial")

These people do not exist

GENERATIVE ADVERSARIAL NETWORKS

- One neural network (the generator) generates new data instances
  - The goal of the generator is to produce realistic-looking data samples that are indistinguishable from real data
- The other (the discriminator) takes input samples, either real or generated by the generator, and predicts whether each sample is real or fake
- During training, the generator and discriminator are trained iteratively
  - The discriminator is trained to maximize its classification accuracy, while the generator is trained to minimize the discriminator's accuracy by generating realistic samples

GENERATIVE ADVERSARIAL NETWORKS (APPLICATIONS)



https://www.linkedin.com/pulse/exploring-fascinating-realm-generative-adversarial-networks-kaurav/

ACTIVE LEARNING MOMENT: THINK-PAIR-SHARE

EXERCISE

Consider what you have learned about GENERATIVE ADVERSARIAL NETWORKS (GANs). What is the role of the "Discriminator" within a GAN? Illustrate with an example in NLP.

Possible Answer: the discriminator evaluates the instances generated by the generator for authenticity, that is, it decides whether each instance of data that it reviews belongs to the actual training dataset or not. For instance, in Question Generation, the discriminator will decide if a given question was taken from the training dataset or generated by the generator.

Generated by DALL-E

OVERVIEW

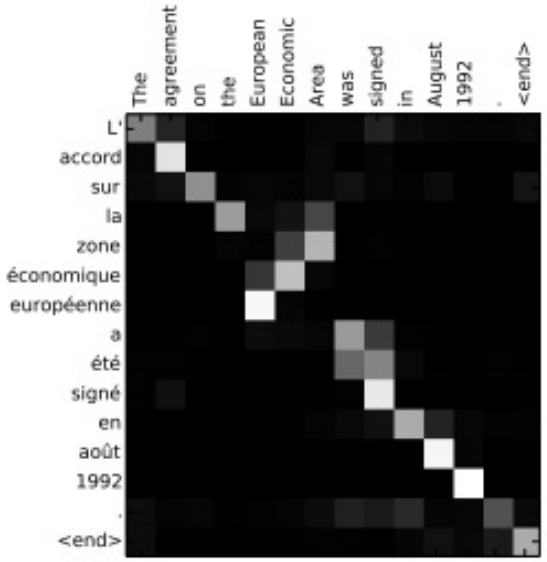
- Learning objectives
- Topics
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Sequence to Sequence Models
  - Autoencoders
  - Generative Adversarial Networks
  - Attention
  - Transformers
- Key takeaways
- Suggested readings

ATTENTION MECHANISMS

- Loosely based on the visual attention mechanism found in humans
  - Humans can focus on a certain region with “high resolution”, while perceiving the surroundings in “low resolution”, and then adjusting the focal point over time

ATTENTION MECHANISMS

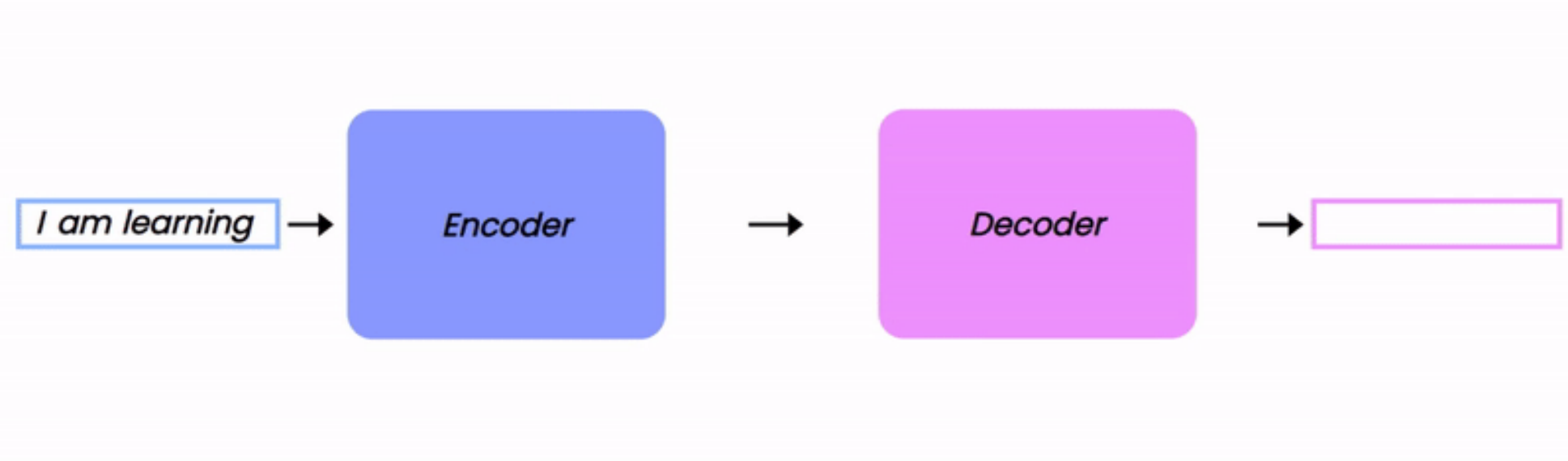
- Proposed as a method to both align and translate
  - Alignment: identifies which parts of the input sequence are relevant to each word in the output
  - Translation: use the relevant information to select the appropriate output



https://skymind.ai/wiki/attention-mechanism-memory-network

ATTENTION MECHANISMS

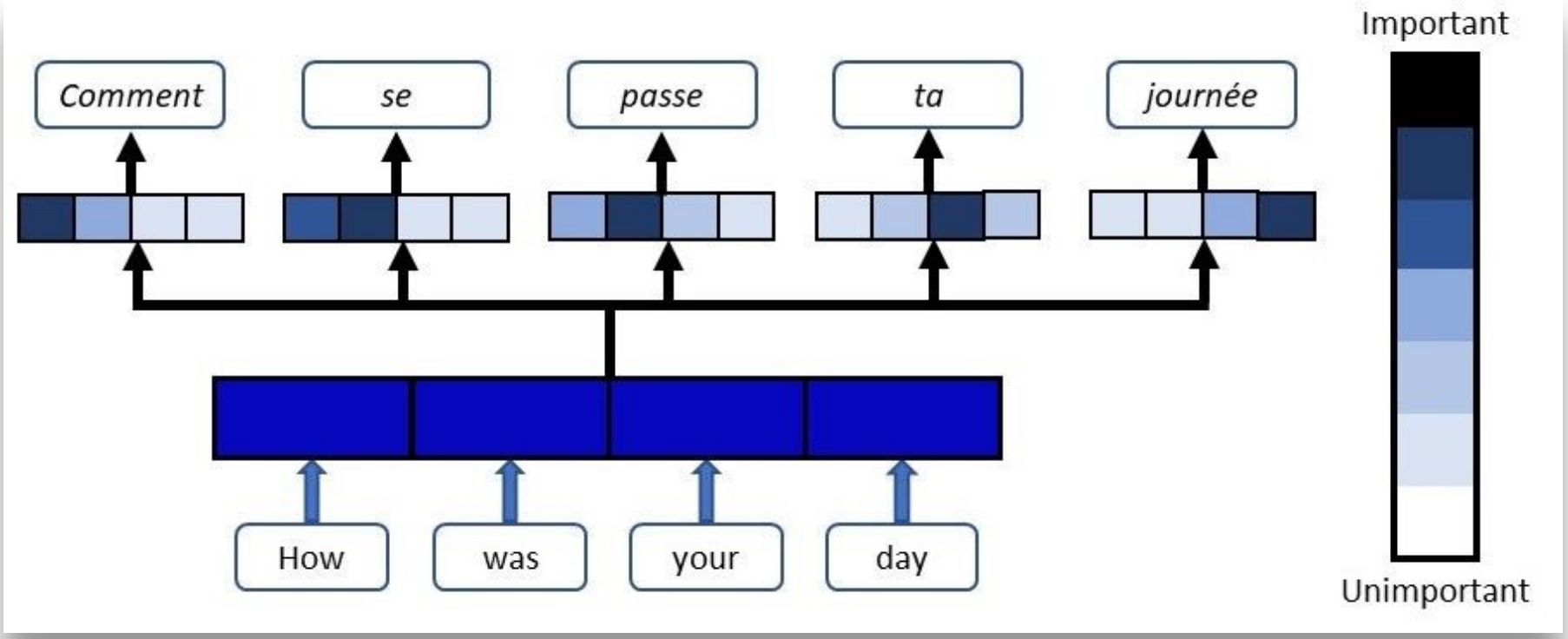
- Previously (seq2seq):



https://blog.floydhub.com/attention-mechanism/

ATTENTION MECHANISMS

- Now: instead of encoding the input sequence into a single fixed context vector, the attention model develops a context vector for each output time step

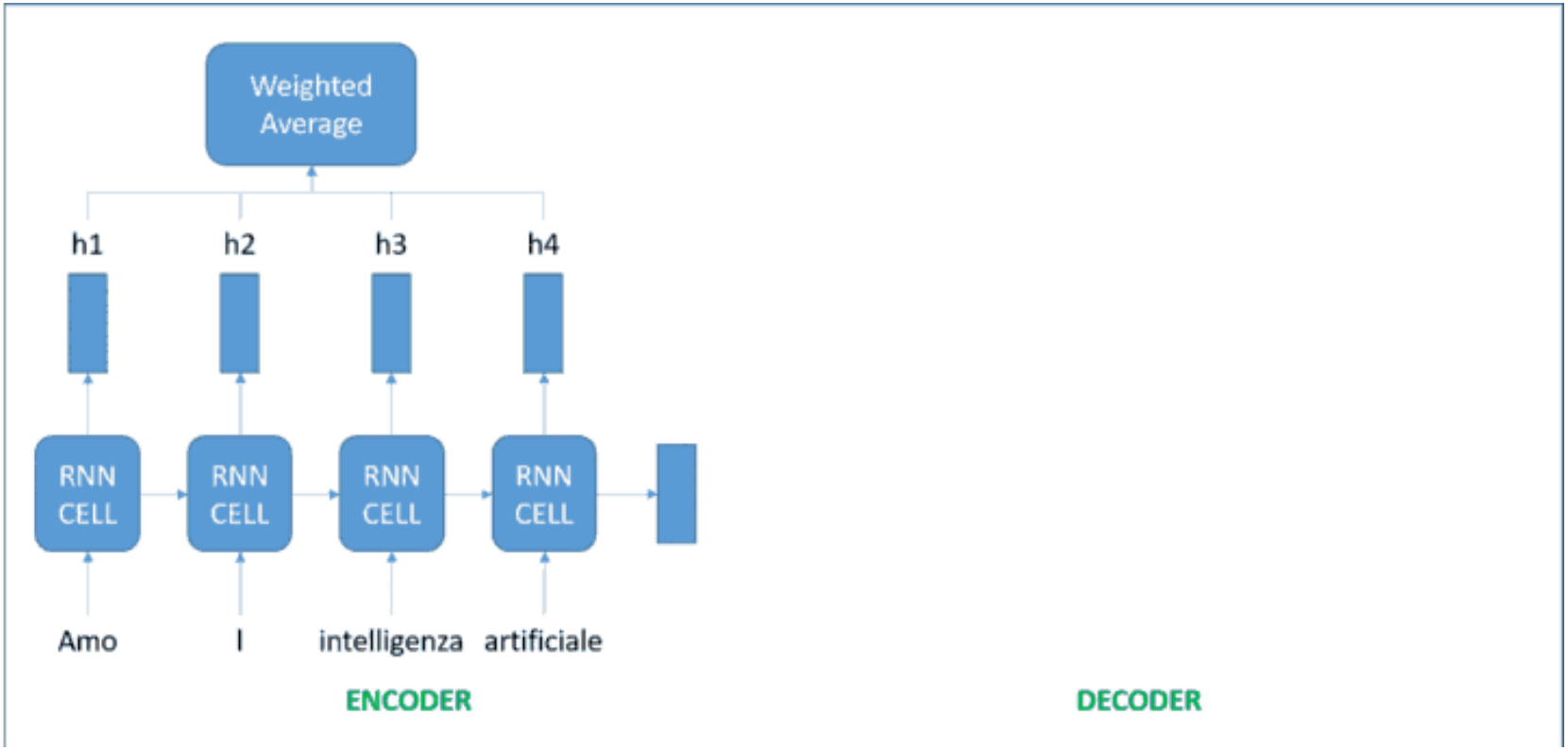


https://blog.floydhub.com/attention-mechanism/

ATTENTION MECHANISMS

- That is, we have an attention layer between encoder/decoder
  - the attention mechanism tells the decoder, which is the part of the input to which it should pay more attention (for each output time step).

https://medium.com/swlh/attention-please-1e16e7011a08



ATTENTION MECHANISMS

- There are several types of attention mechanisms
  - Soft Attention vs. Hard Attention: the first allows models to weigh the importance of different parts of the input data (what we have seen so far); hard attention selects specific parts of the input to focus on and ignores the rest
  - …

ATTENTION MECHANISMS

- There are several types of attention mechanisms
  - Self-attention (or Intra-attention): models how some parts of a sentence relate to other parts of the same sentence
    - We will see how self-attention is used in Transformers

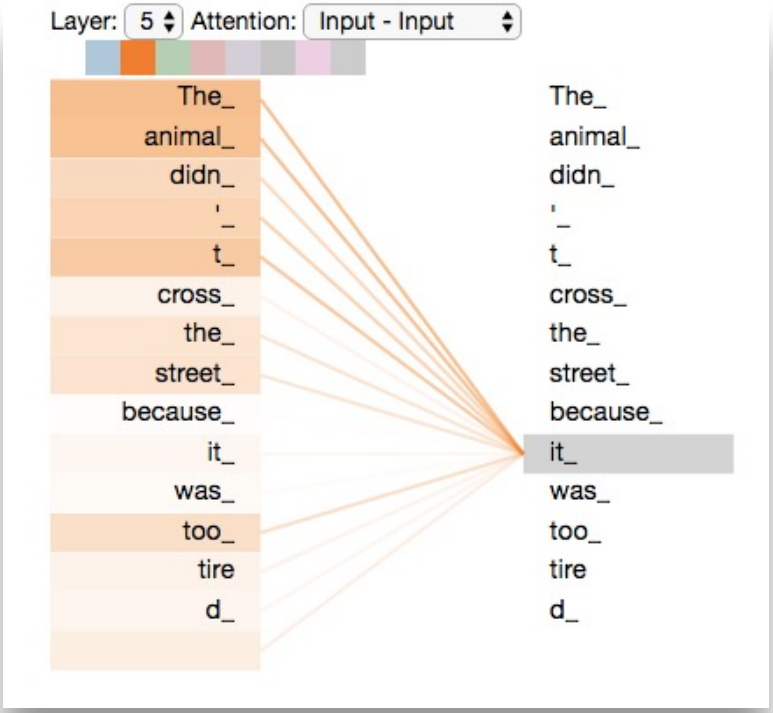


Image from http://jalammar.github.io/illustrated-transformer/

ATTENTION MECHANISMS

- There are several types of attention mechanisms
  - Multi-Head Attention: An extension of self-attention that runs several attention mechanisms in parallel. Allows the model to capture different types of relationships in the data, such as syntactic and semantic dependencies
    - We will see how multi-head-attention is used in Transformers

Image from http://jalammar.github.io/illustrated-transformer/

ATTENTION MECHANISMS

- Attention coefficients are also used to explain the decisions of the model

ATTENTION MECHANISMS

- Initially used in addition to other architectures, like RNNs
- However, it performs very well on its own, as combined with feed-forward layers, attention units can simply be stacked, to form encoders.

- Let us see how

OVERVIEW

- Learning objectives
- Topics
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Sequence to Sequence Models
  - Autoencoders
  - Generative Adversarial Networks
  - Attention
  - Transformers
- Key takeaways
- Suggested readings

TRANSFORMERS

Transformers are about this paper:

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.

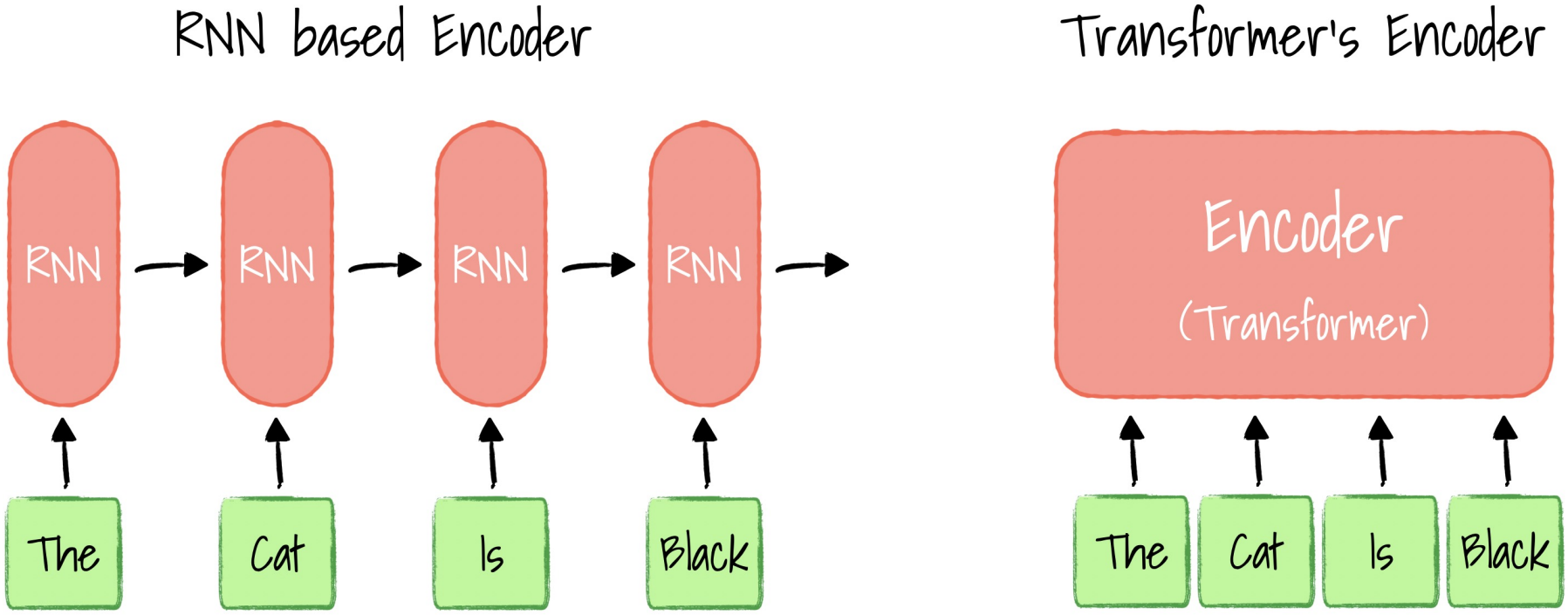
TRANSFORMERS

- What the…!!??

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

TRANSFORMERS

- Transformers: not recurrent; parallel



https://towardsdatascience.com/illustrated-guide-to-transformer-cf6969ffa067

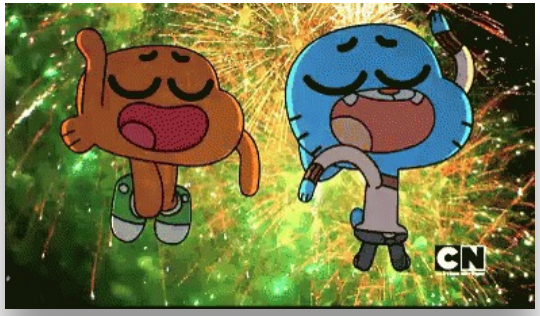
TRANSFORMERS

- Transformers use self-attention mechanisms to gather information about the relevant context of a given word, and then encode that context in the vector that represents the word

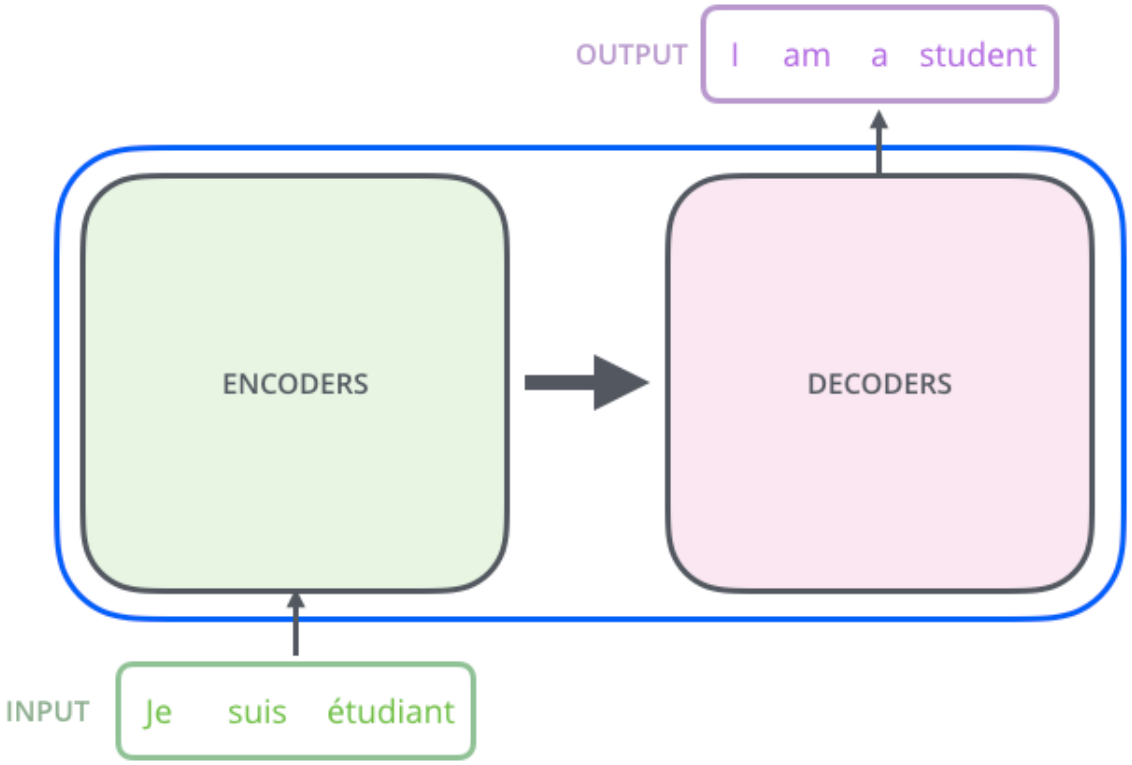
TRANSFORMERS

- Most of the following slides/figures are from:
  - The Illustrated Transformer (Jay Alammar):

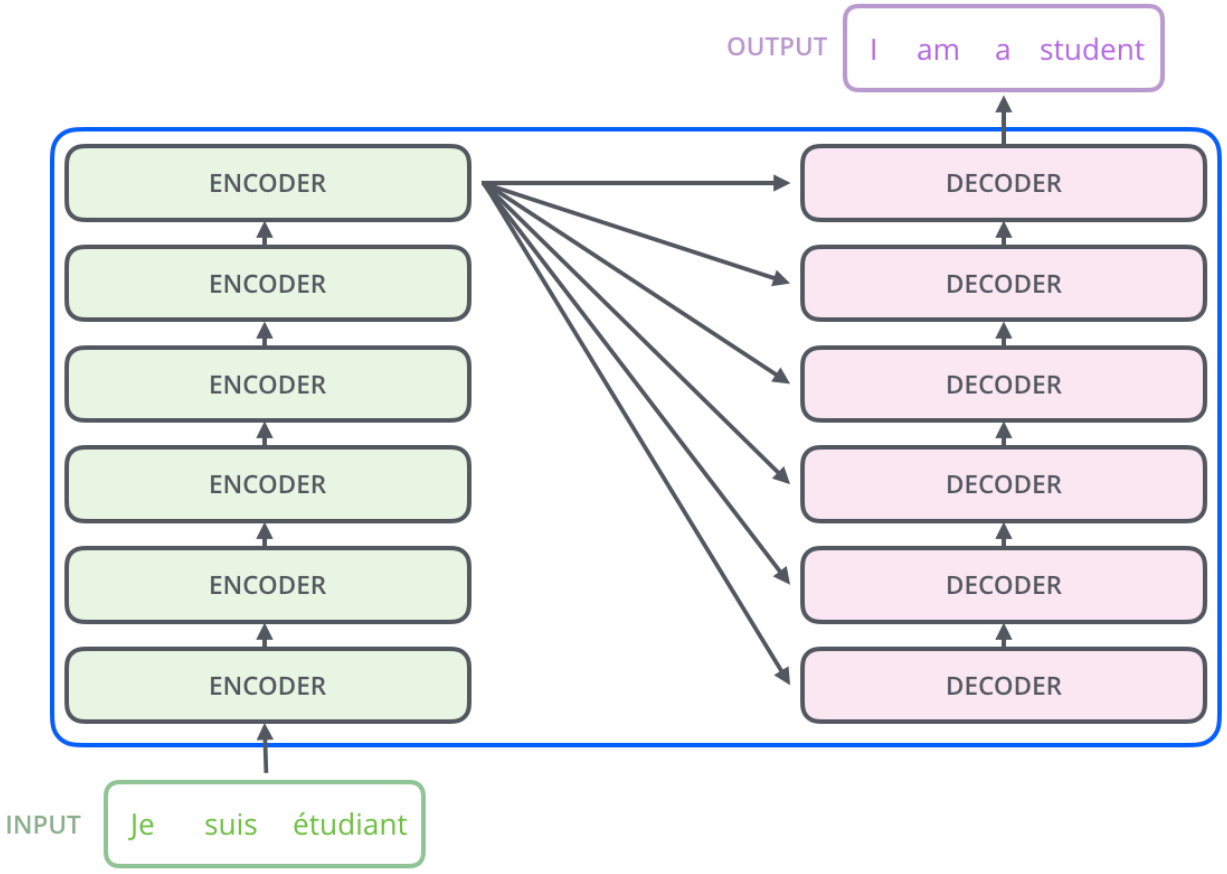
http://jalammar.github.io/illustrated-transformer/



TRANSFORMERS

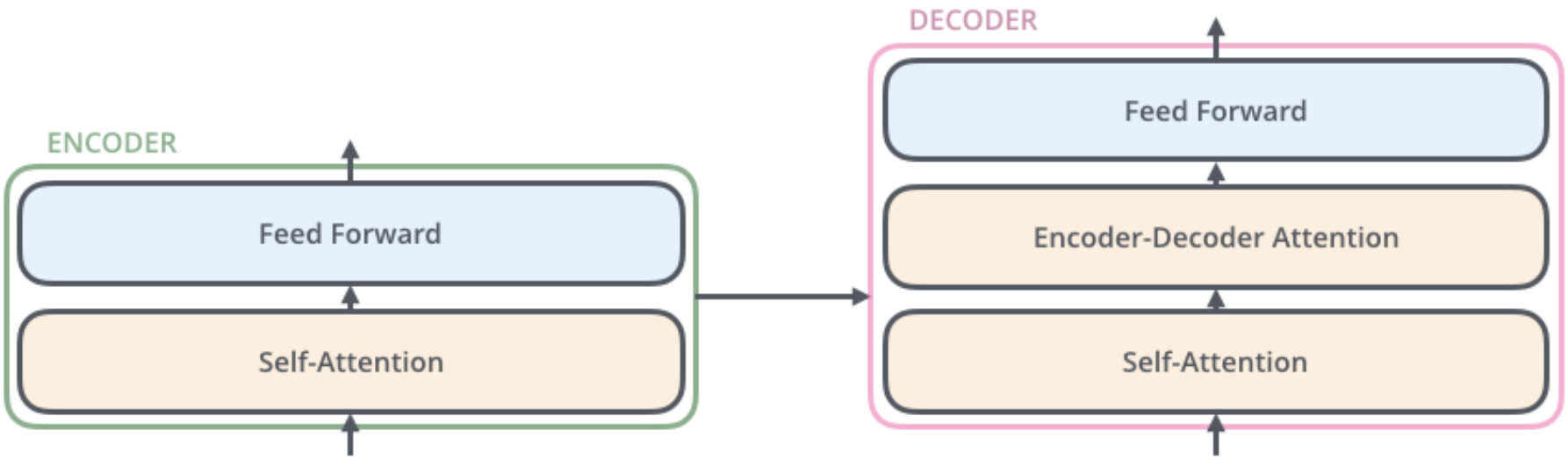


TRANSFORMERS



TRANSFORMERS

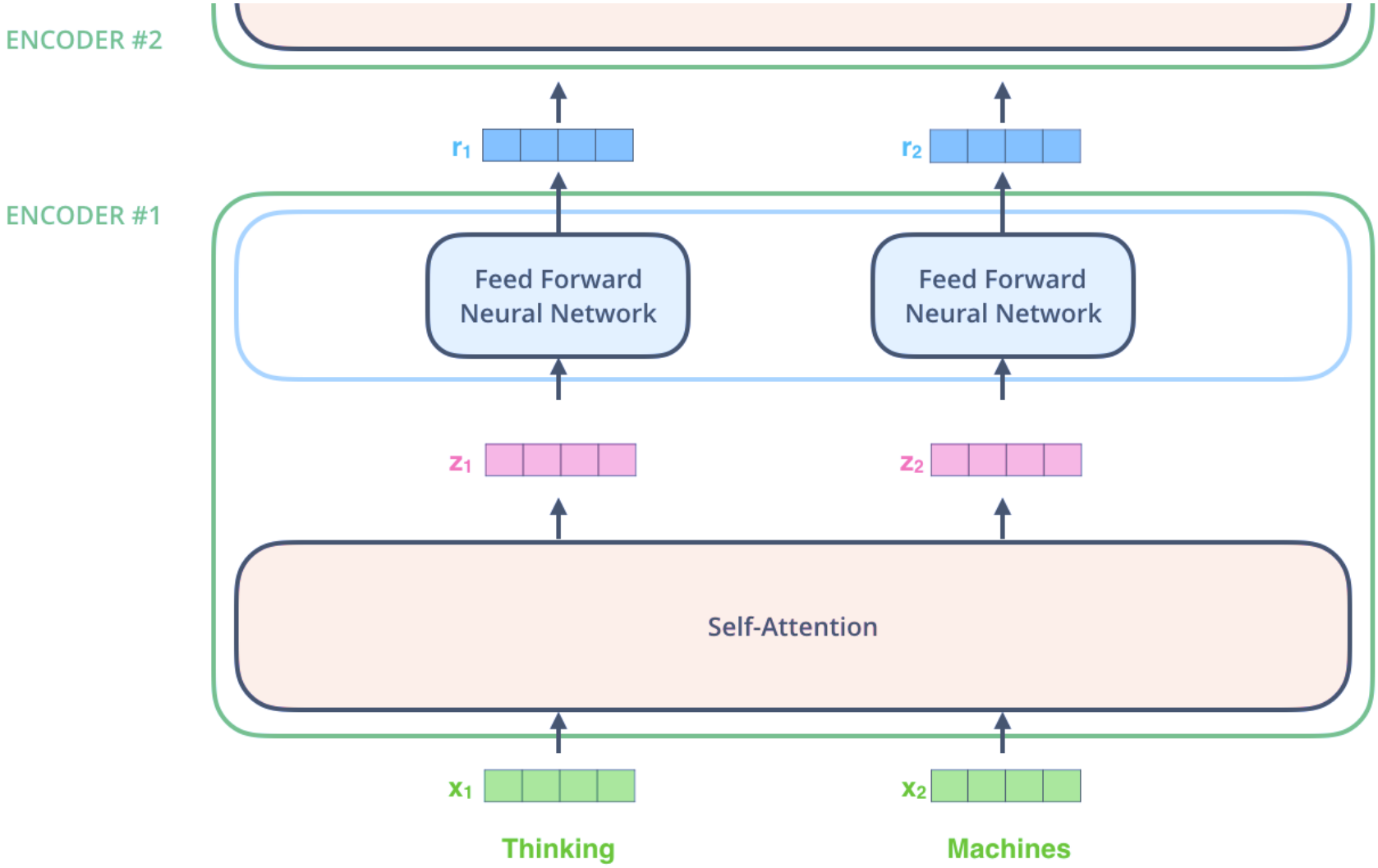
- General structure of each encoder/decoder:



TRANSFORMERS

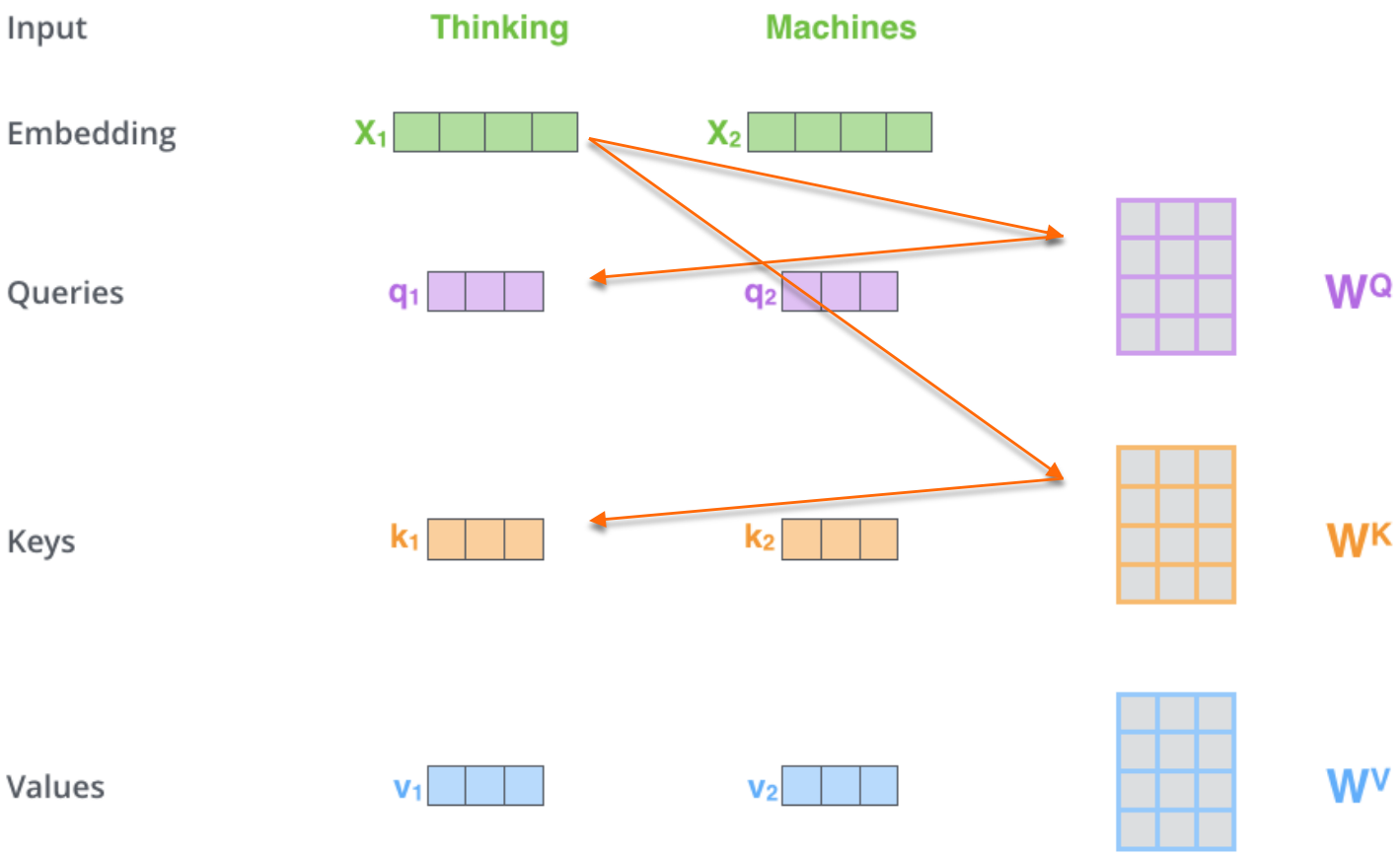
- Encoder

Parallel processing



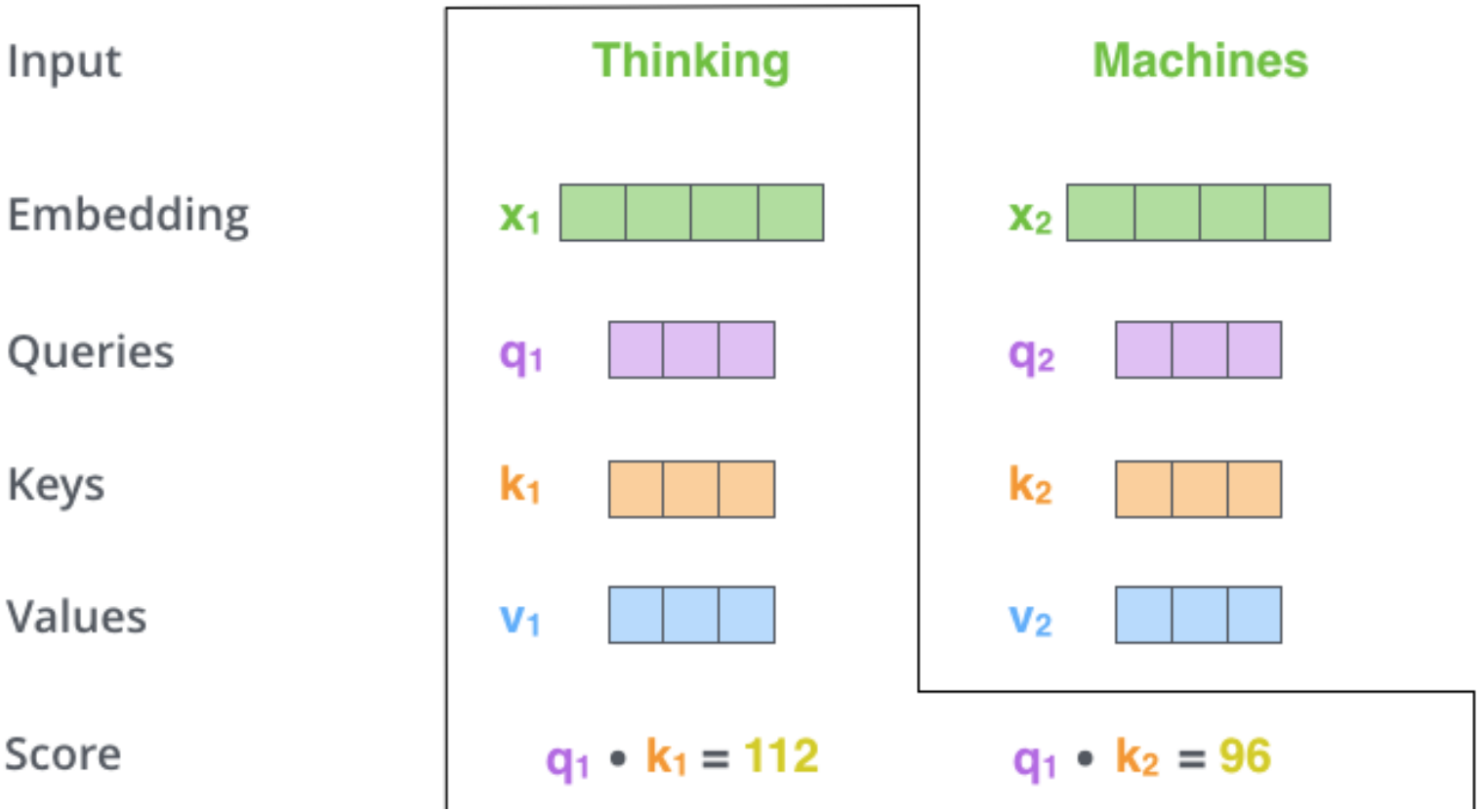
TRANSFORMERS
CALCULATING SELF-ATTENTION

- (1) Create vectors Query, Key and Value for each input vector by multiplying each input vector by three matrices (trained during the training process).



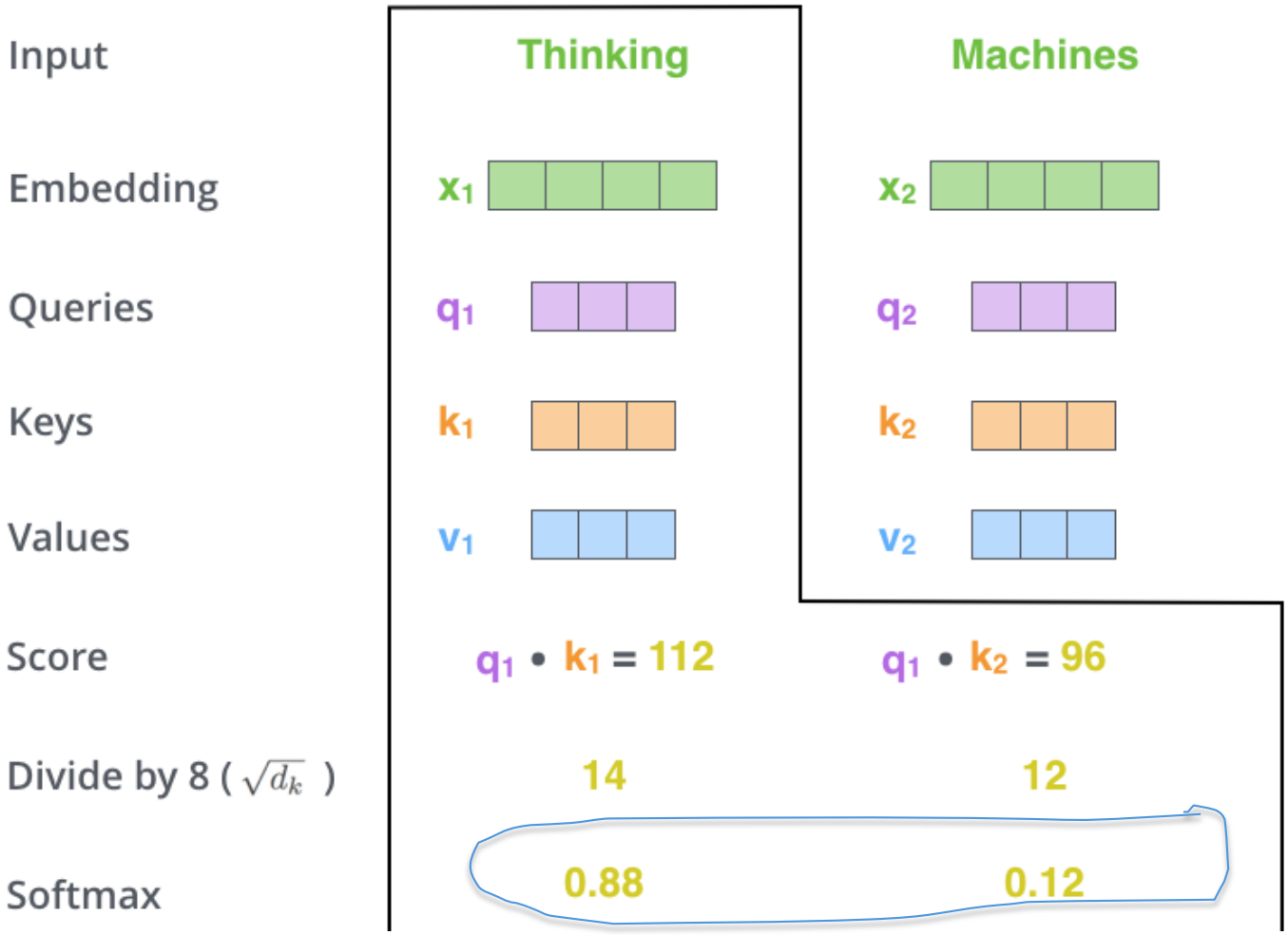
TRANSFORMERS
CALCULATING SELF-ATTENTION

- (2) Score each word of the input sentence against the others. The score determines how much focus to place on other parts of the input sentence as we encode a certain word



TRANSFORMERS
CALCULATING SELF-ATTENTION

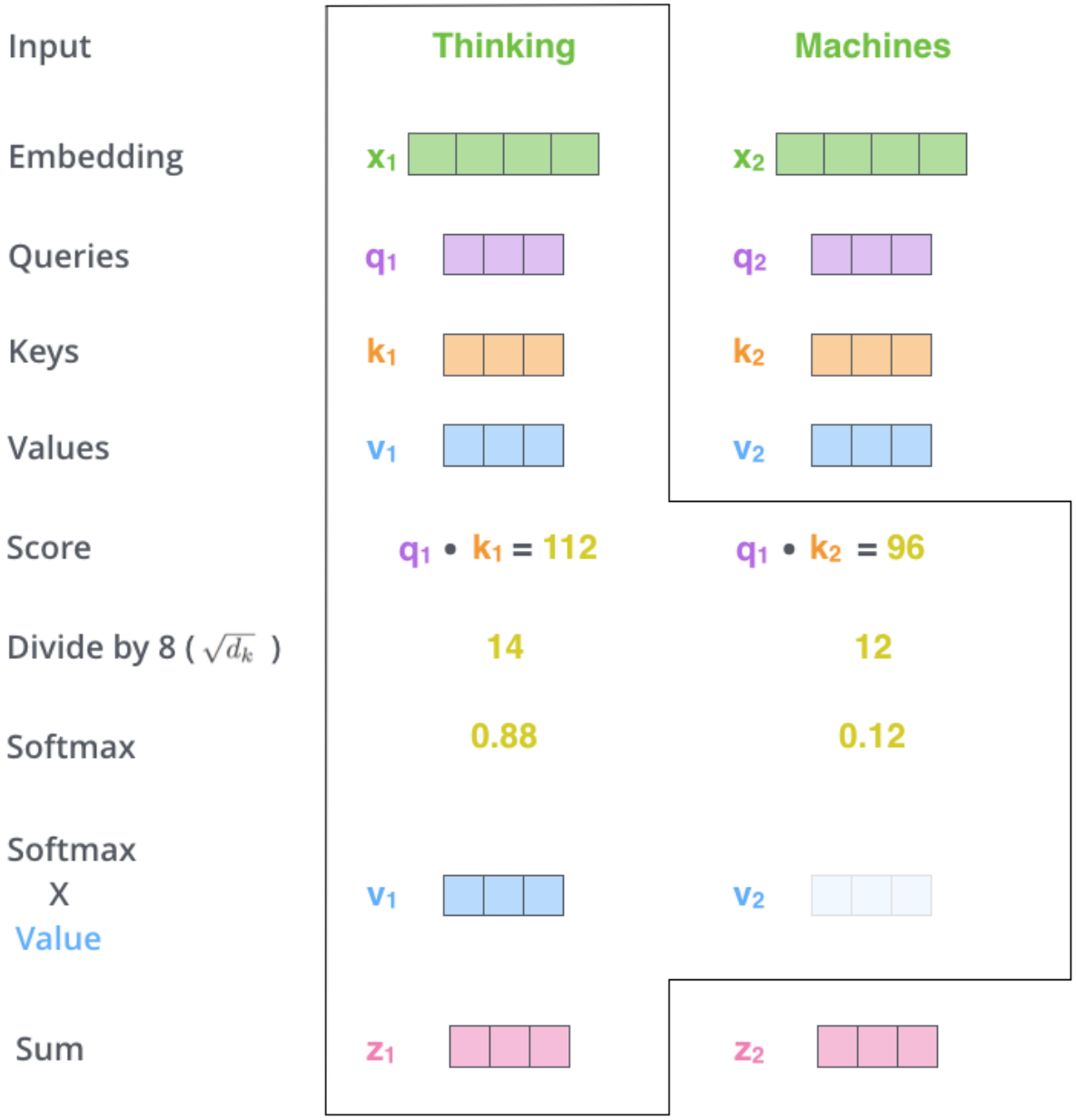
- (3) + (4): divide the scores by 8 (default), then pass the result through a Softmax operation



How much focus...

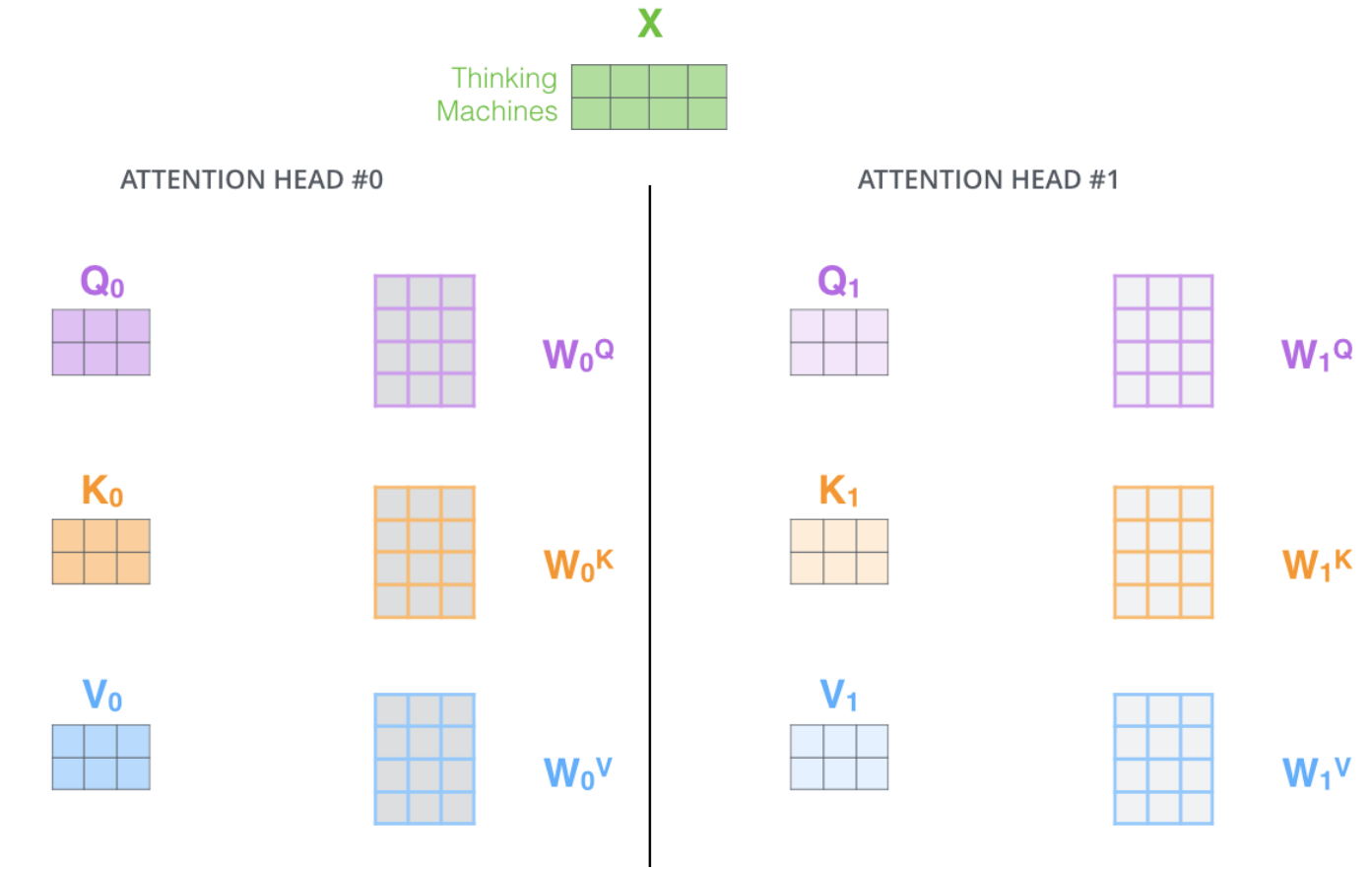
TRANSFORMERS
CALCULATING SELF-ATTENTION

- (5): multiply each value vector by the softmax score.
Idea: drown-out irrelevant words.
- (6): sum up the weighted value vectors.
  - This produces the output of the self-attention layer at this position



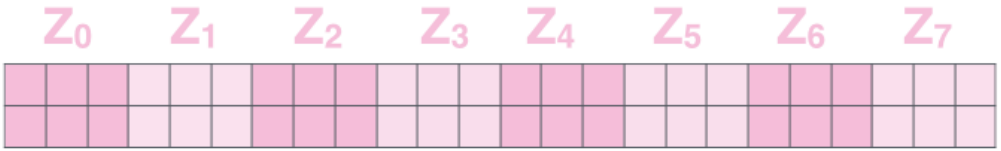
TRANSFORMERS
MULTI-HEAD ATTENTION

- Multi-head attention expands the model's ability to focus on different positions
  - multiple sets of Query/Key/Value weight matrices (the original Transformer uses 8).



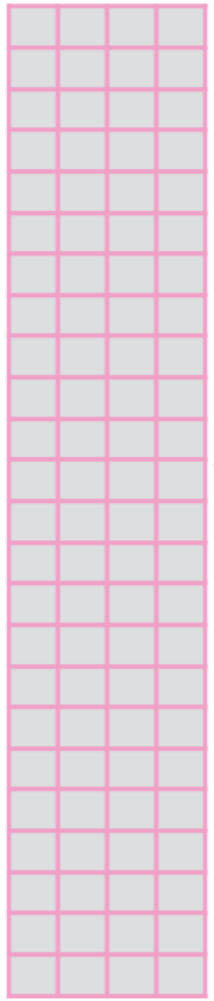
TRANSFORMERS
MULTI-HEAD ATTENTION

1) Concatenate all the attention heads



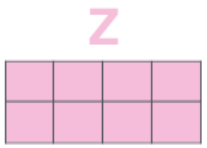
2) Multiply with a weight matrix W^O that was trained jointly with the model

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

=



TRANSFORMERS
ALL TOGETHER NOW – ENCODING

1) This is our input sentence*

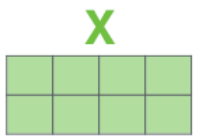
2) We embed each word*

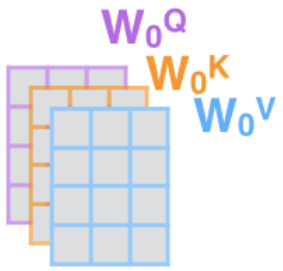
3) Split into 8 heads. We multiply X or R with weight matrices

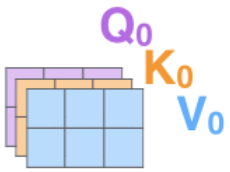
4) Calculate attention using the resulting Q/K/V matrices

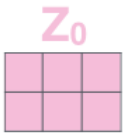
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

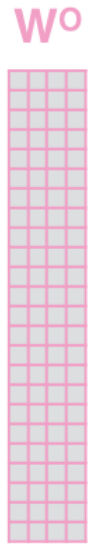
Thinking Machines

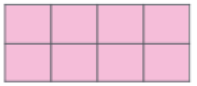




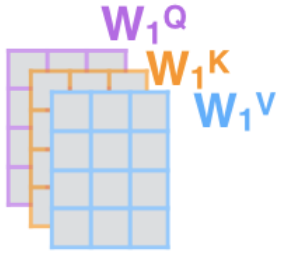


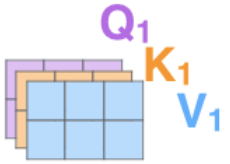


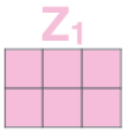


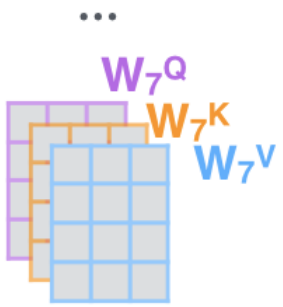


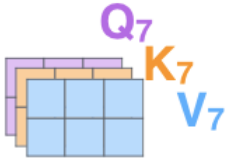
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

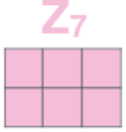




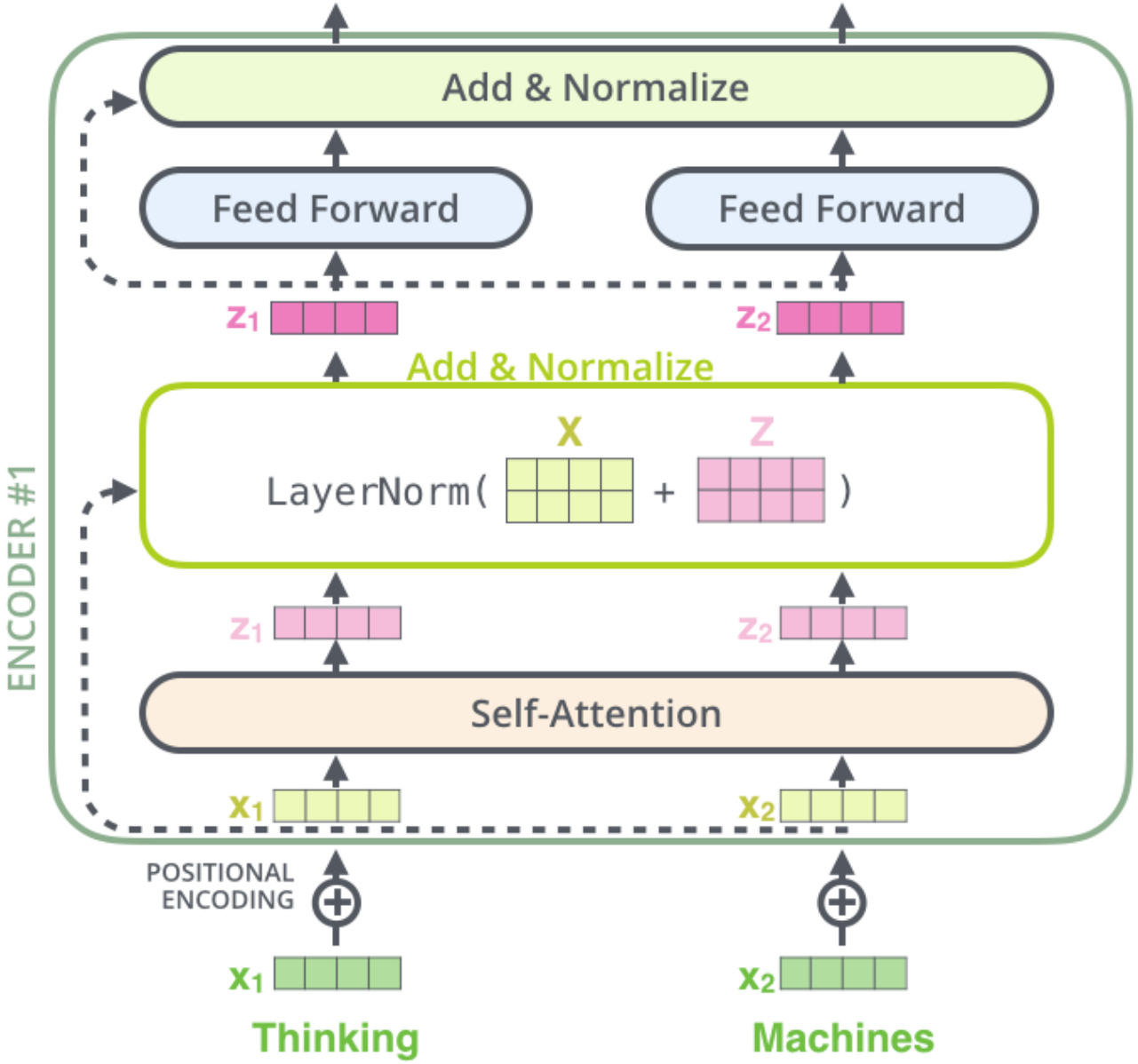






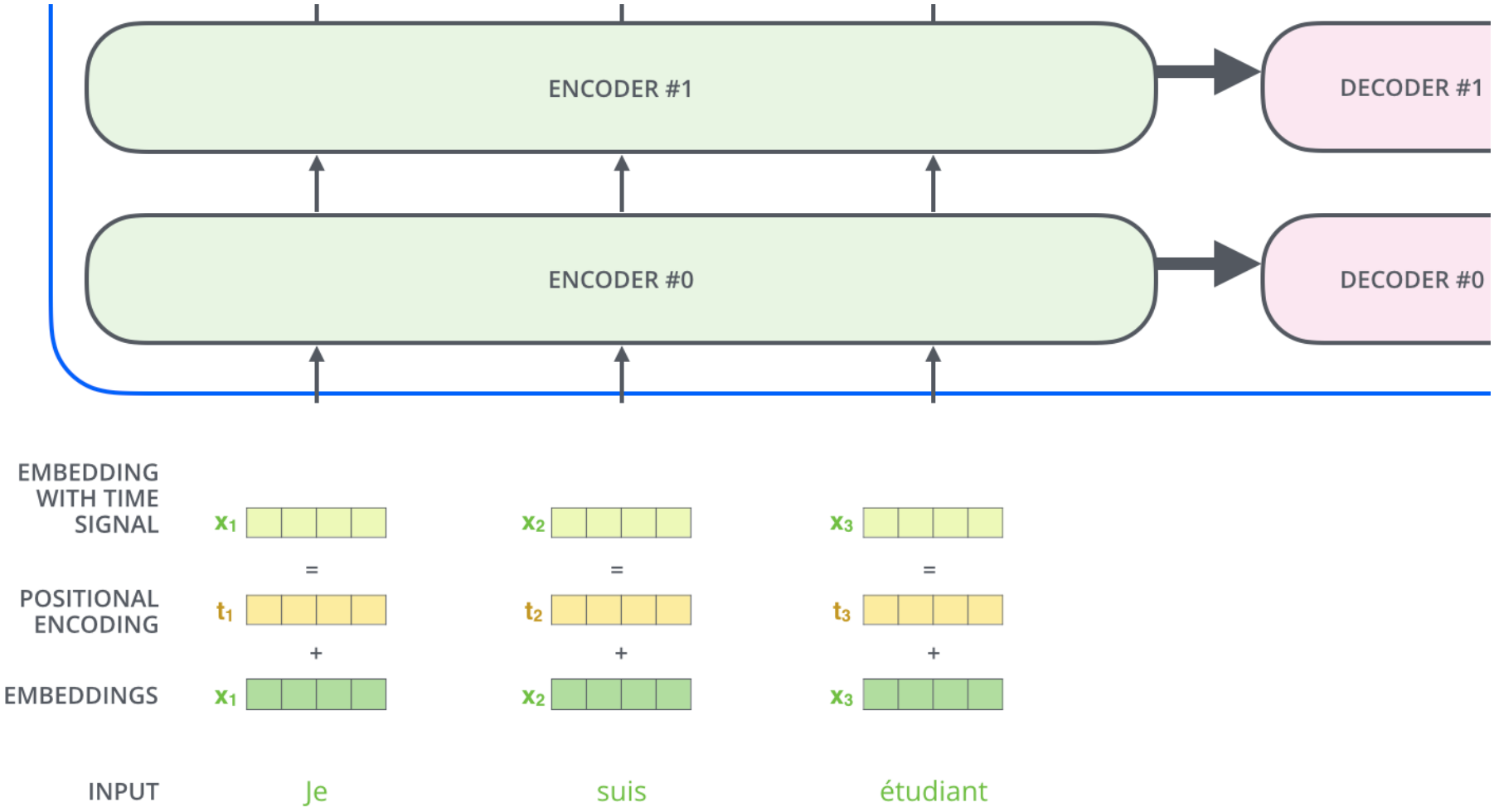


TRANSFORMERS
(AND WHEN WE THOUGHT IT WAS OVER)



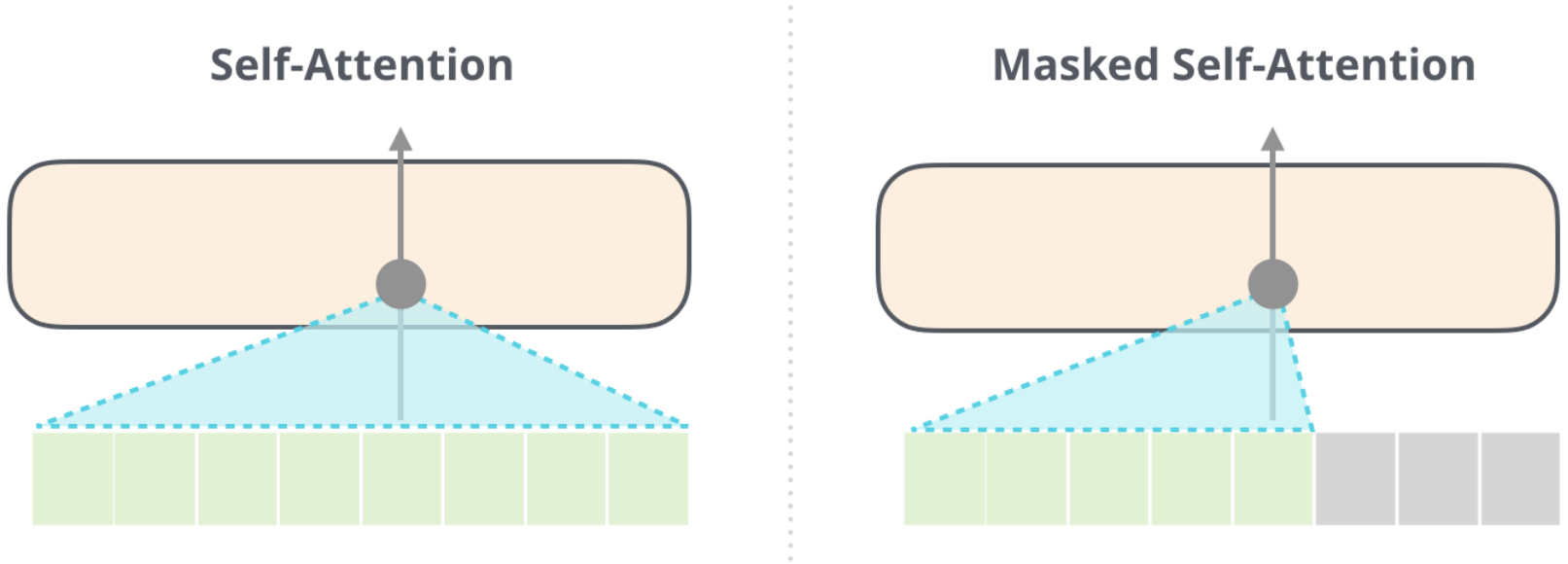
TRANSFORMERS

REPRESENTING THE ORDER OF THE SEQUENCE USING POSITIONAL ENCODING



TRANSFORMERS
SEVERAL STEPS AFTERWARDS

- … we still have the decoder…

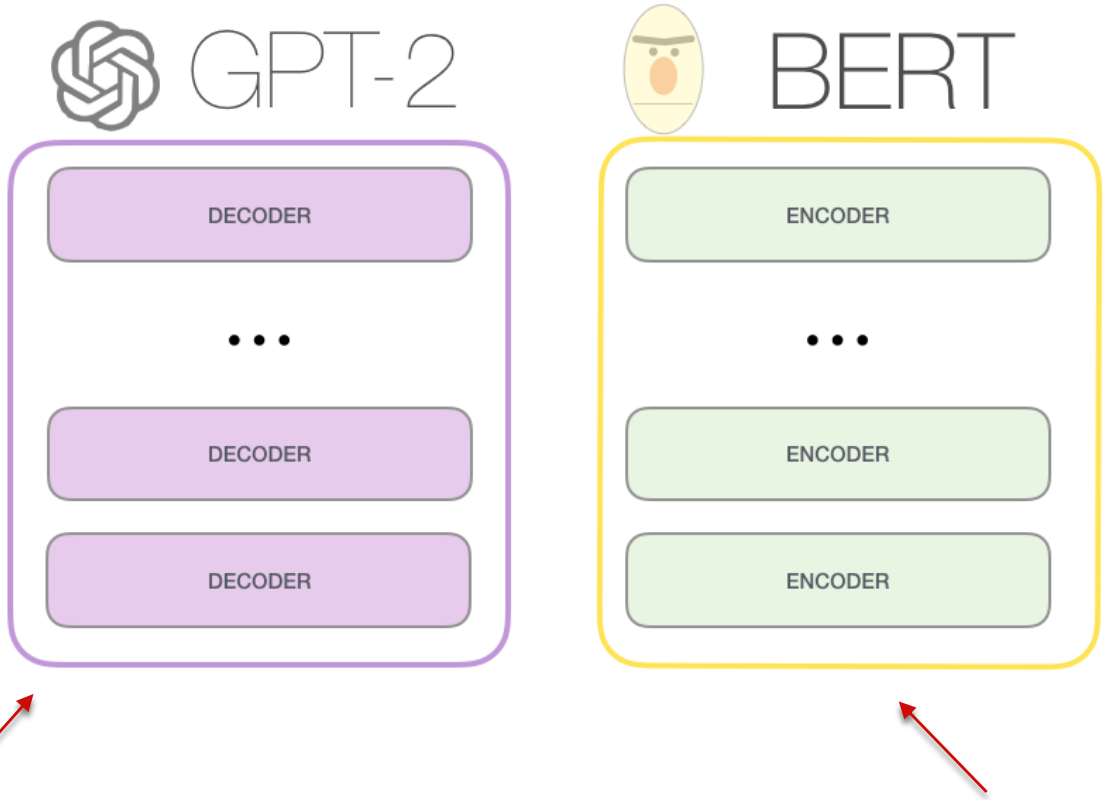


From: The Illustrated GPT-2 (Visualizing Transformer Language Models)

TRANSFORMERS

- But now we can understand…

TRANSFORMERS (APPLICATIONS)



Auto-regressive models

Auto-encoding models

ACTIVE LEARNING MOMENT

Let us try GPT-2 (?)

- https://play.aidungeon.io/main/home

- Suggestion:
  - Login, explore a World. I hope the internet will be ok.

KEY TAKEAWAYS

KEY TAKEAWAYS

- Concepts:
  - Feedforward Networks
  - Recurrent Neural Networks
    - RNNs, Bidirectional RNNs, LSTMs, GRUs
  - Sequence to Sequence Models (Seq2Seq)
  - Autoencoders
    - Denoising with autoencoders
  - Generative Adversarial Networks (GANs)
  - Attention Mechanisms:
    - Multi-Head Attention, Soft Attention, Hard Attention, Spatial Attention, Temporal Attention, Cross-Attention
  - Transformers
- Understand how we moved from FFNN to Transformers (and other architectures)

SUGGESTED READINGS

READINGS

- Jurafsky, Chapter 9, 10.1, 10.2, 10.3
- "The Illustrated Transformer" (Jay Alammar)