

Introduction to Deep Learning

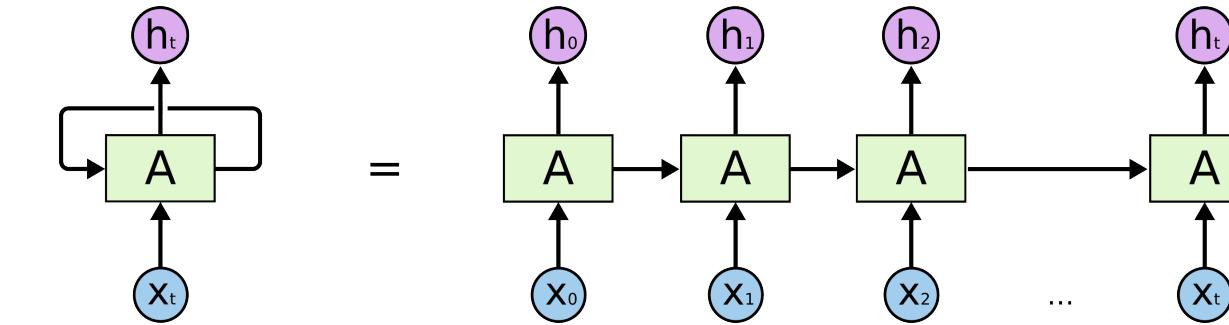
Lecture: Transformers and the Attention Mechanism

Francis Engelmann

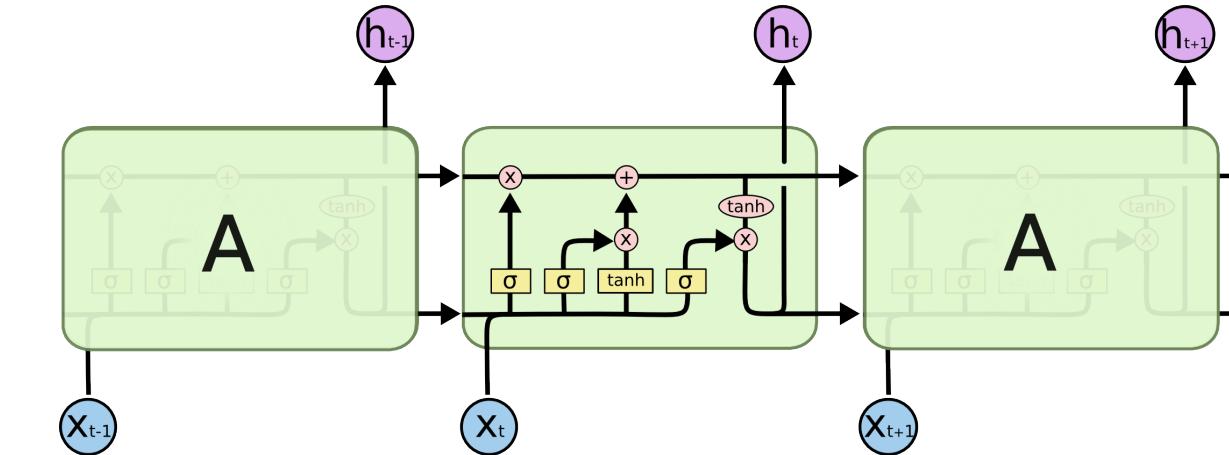
June 7, 2024

Last Week: Sequence Modeling (Recap)

- Recurrent Neural Networks (RNN)



- Long Short Term Memory (LSTM)



- Gated Recurrent Unit (GRU)

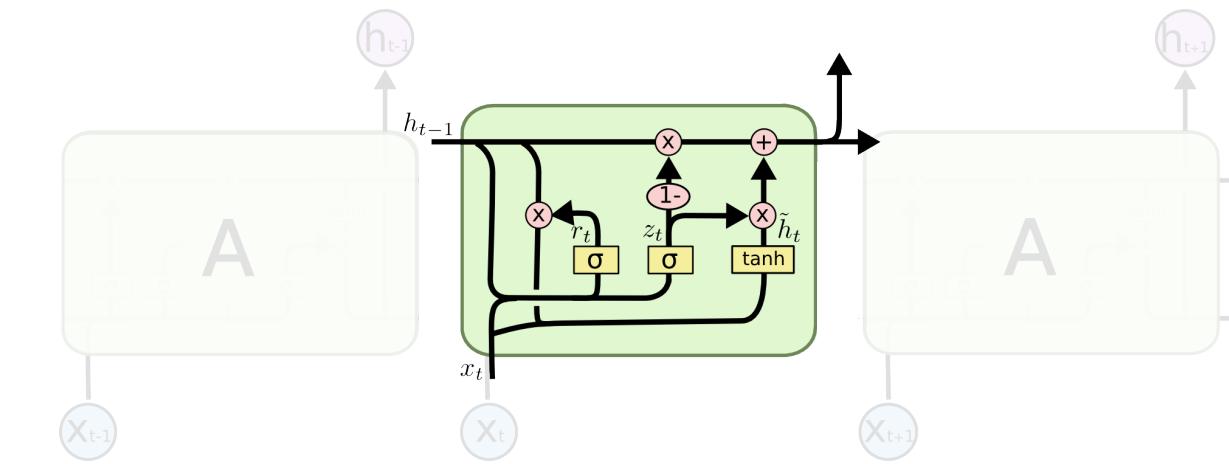
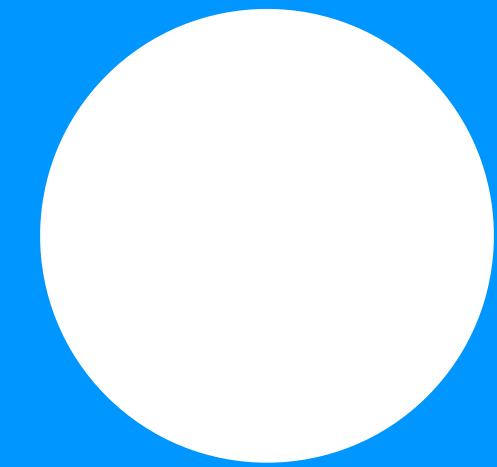
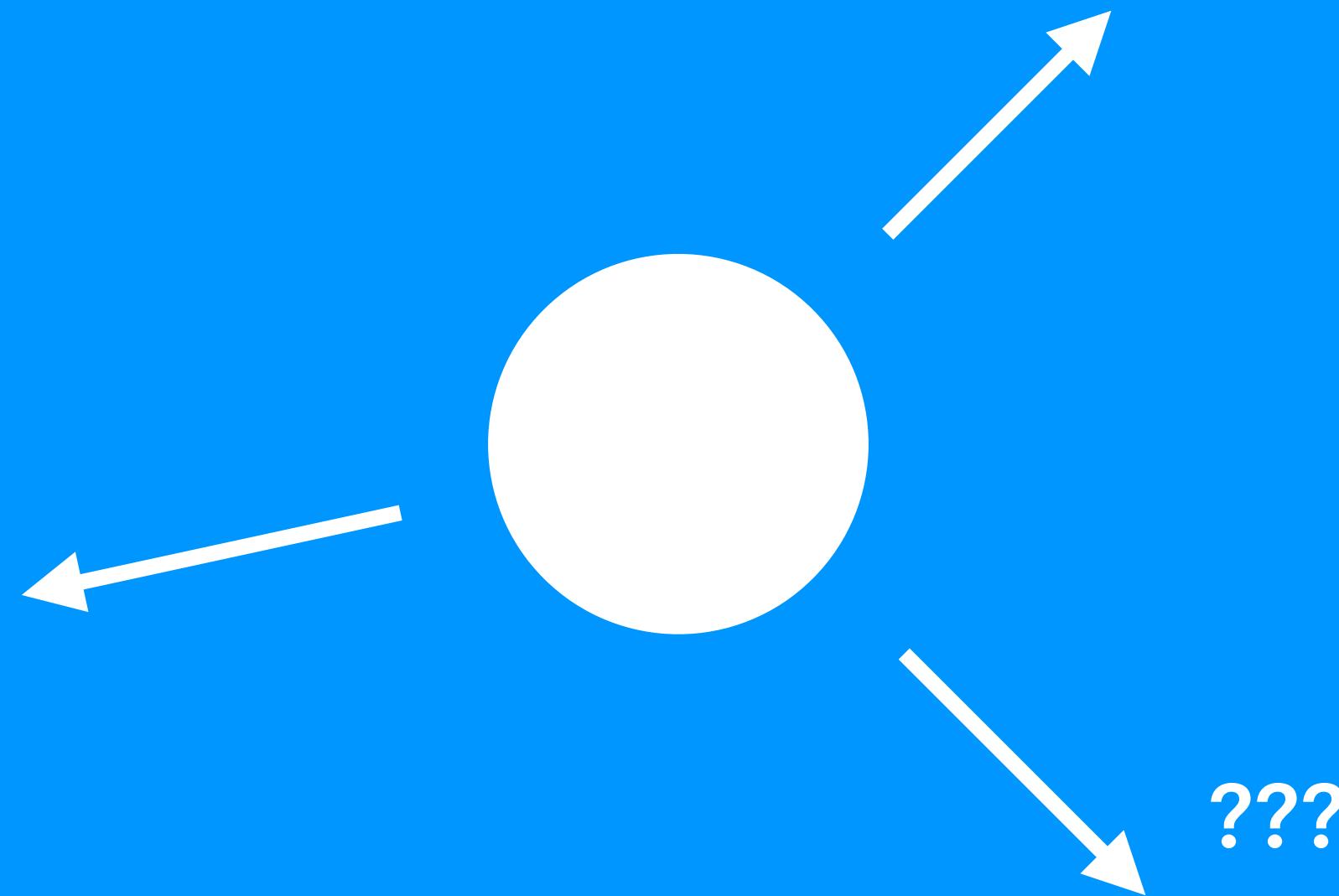


Image credit: Chris Olah

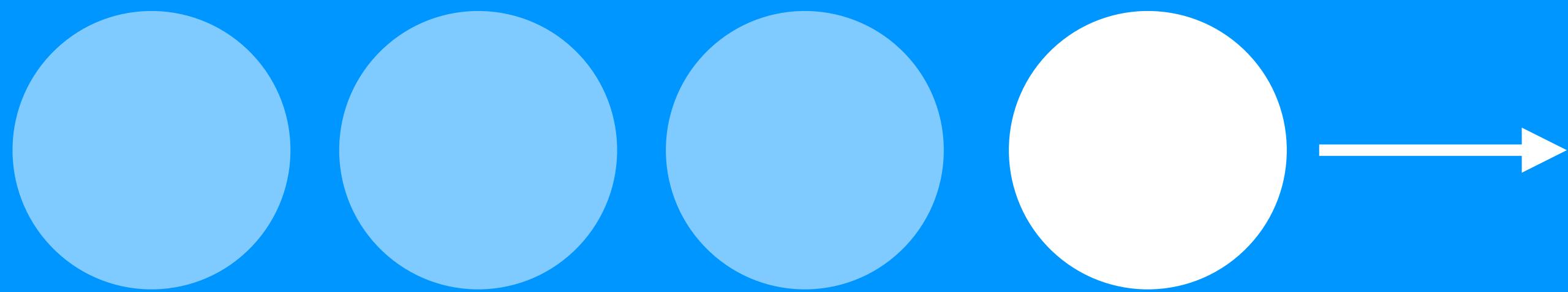
Given an image of a ball,
can you predict where it will go next?



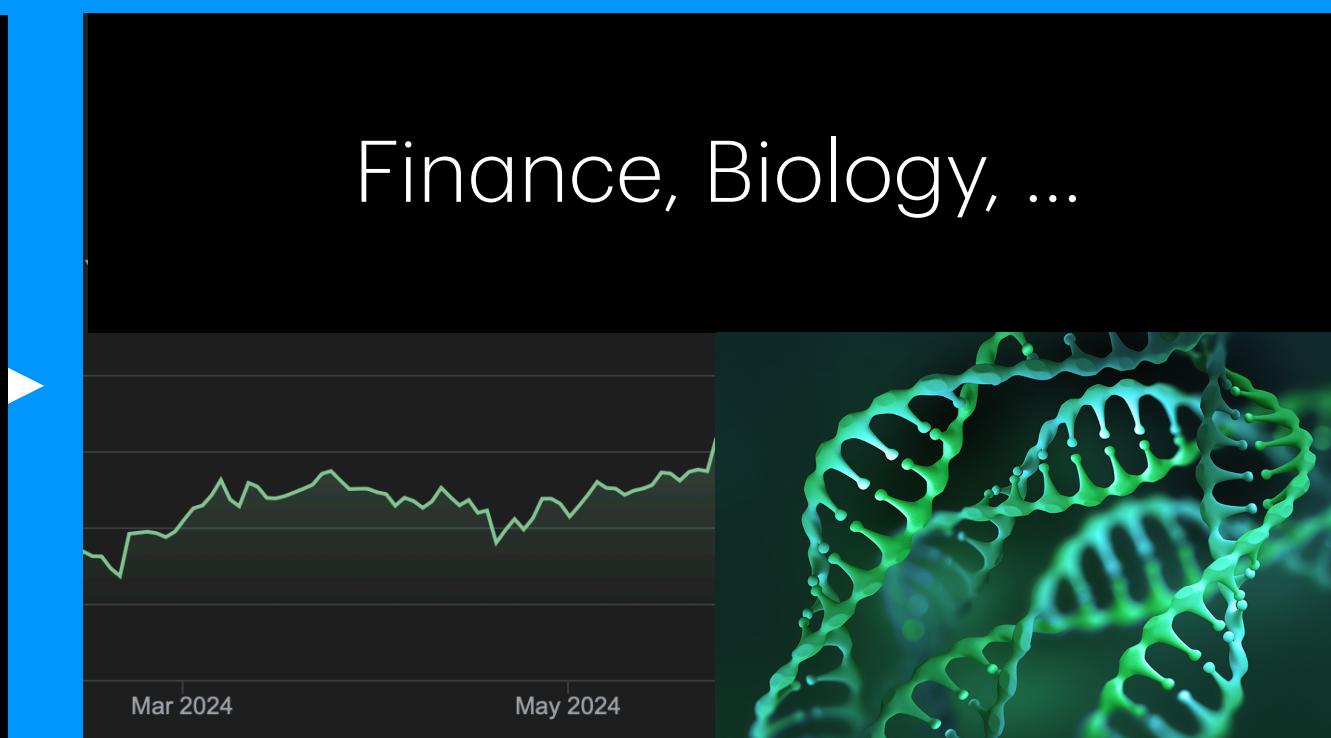
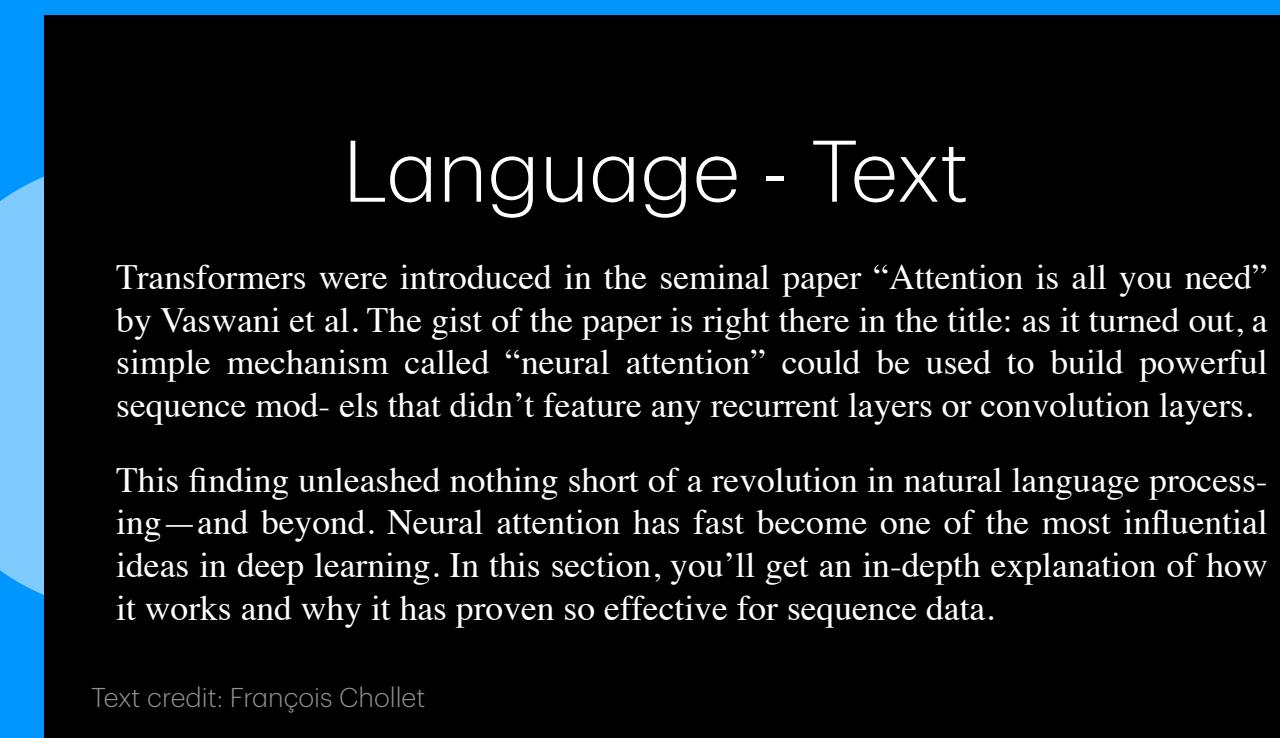
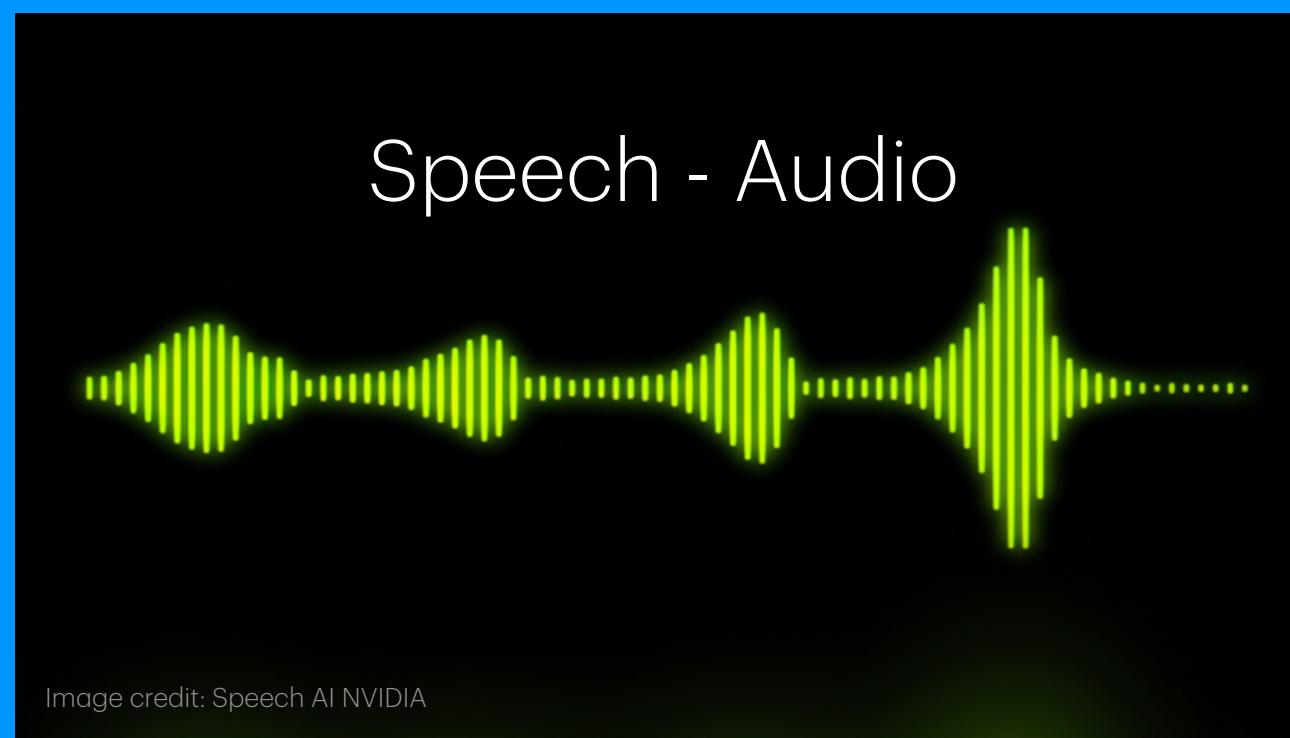
Given an image of a ball,
can you predict where it will go next?



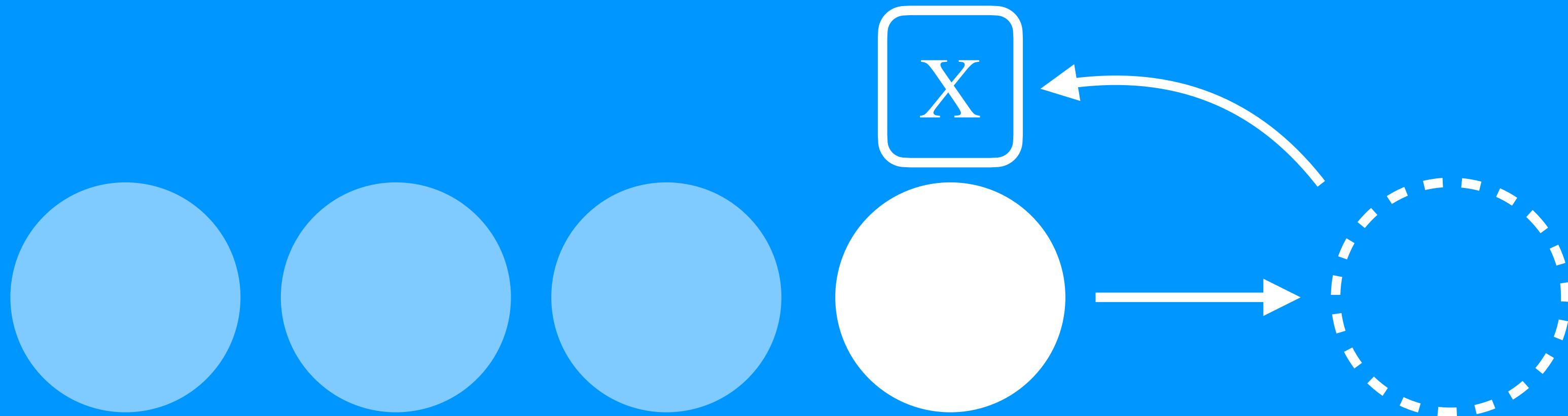
Given an image of a ball,
can you predict where it will go next?



Given an image of a ball, can you predict where it will go next?

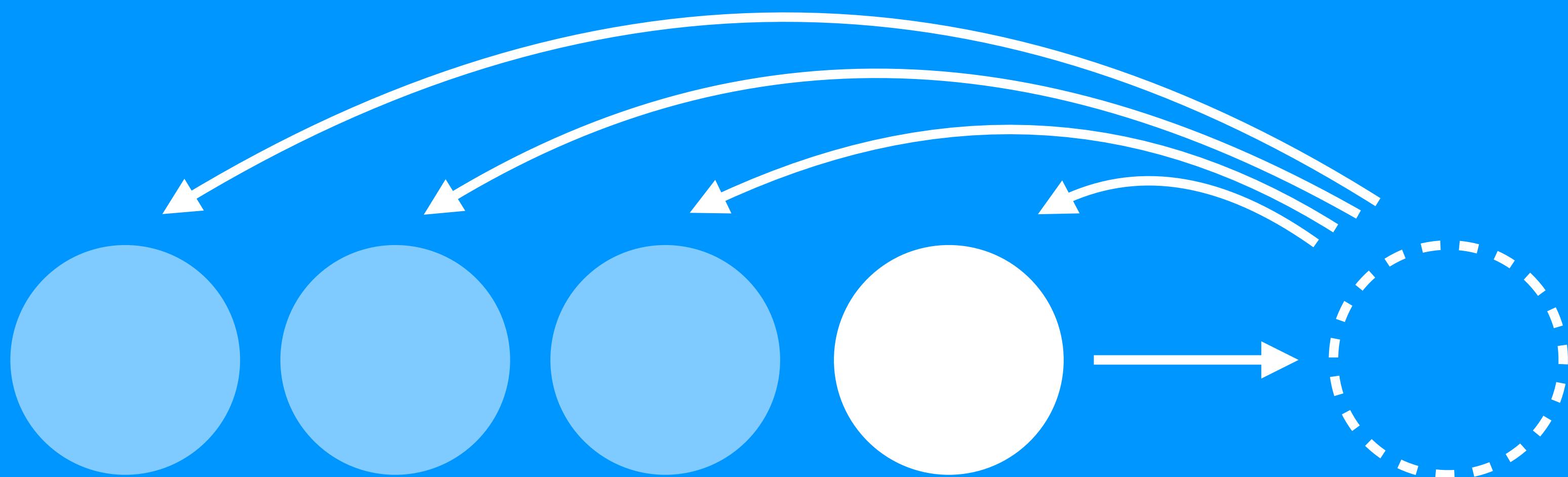


RNN



"Luxembourg is where I grew up, but now I live in Zurich. My mother tongue is ____."

Attention



Last Week: Sequence Modeling (Recap)

- Recurrent Neural Networks (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- **Today:** Attention and the Transformer model

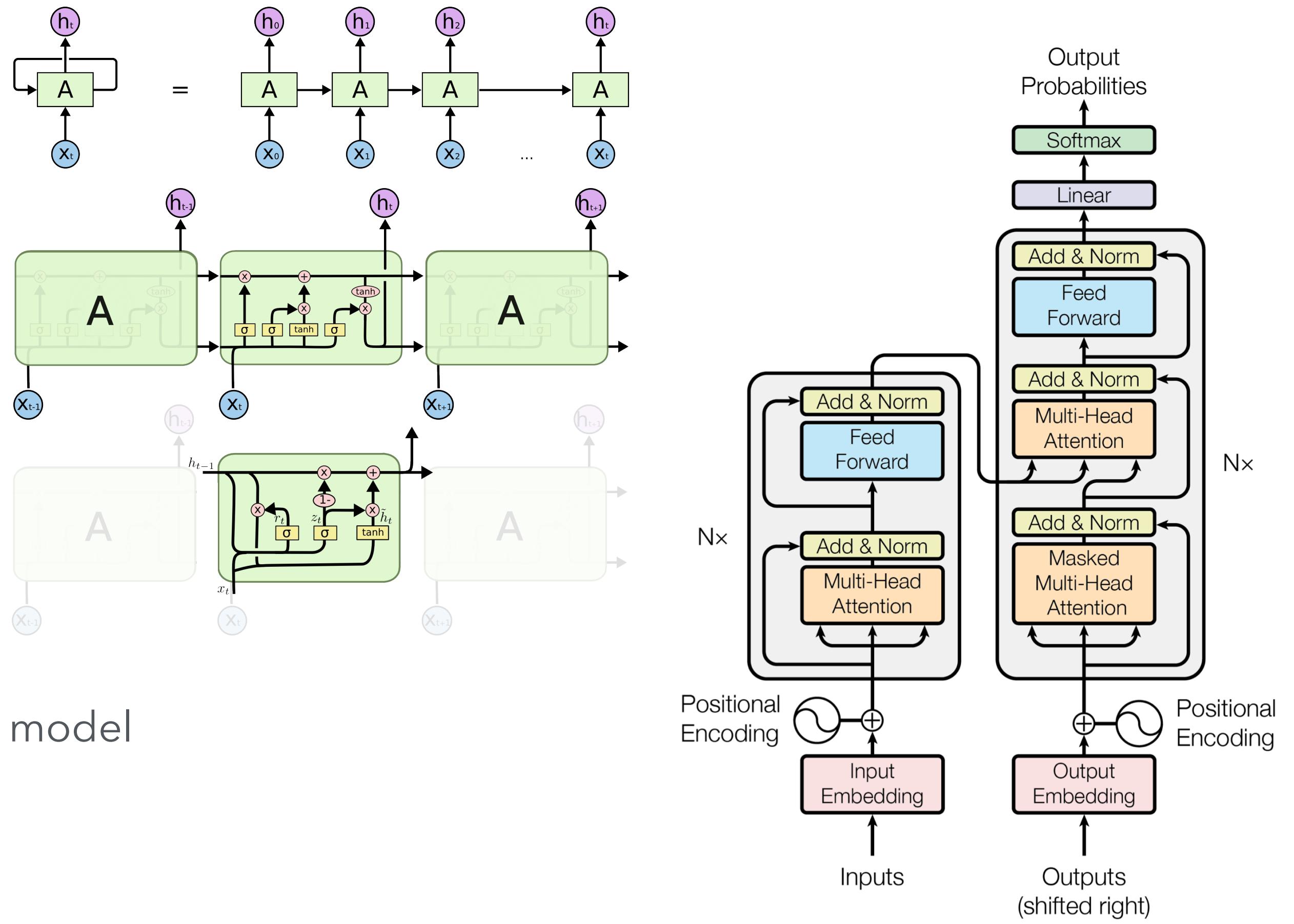
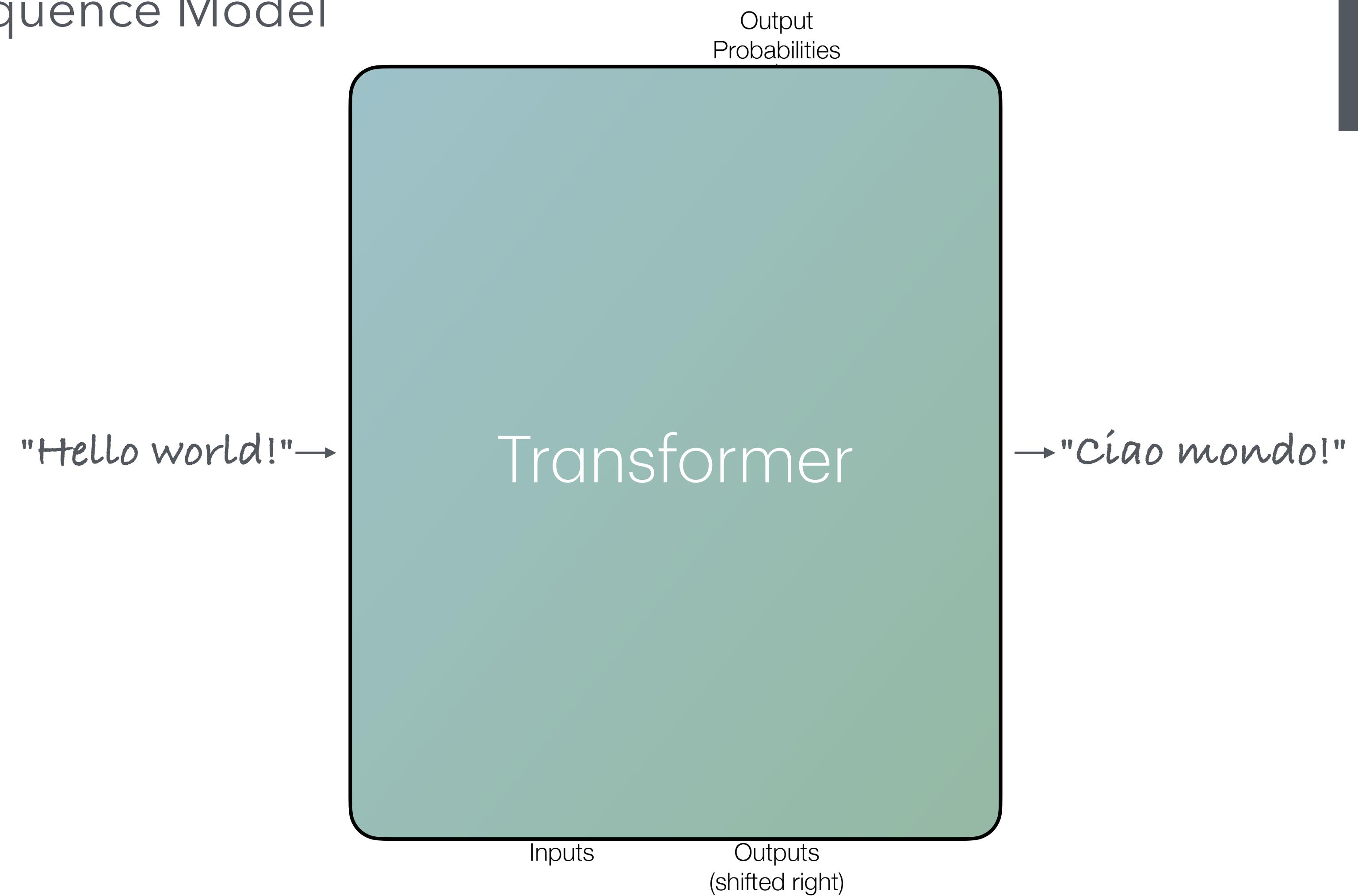


Image credit: Chris Olah, Vaswani et al.

Transformer: Overview

Sequence-to-Sequence Model

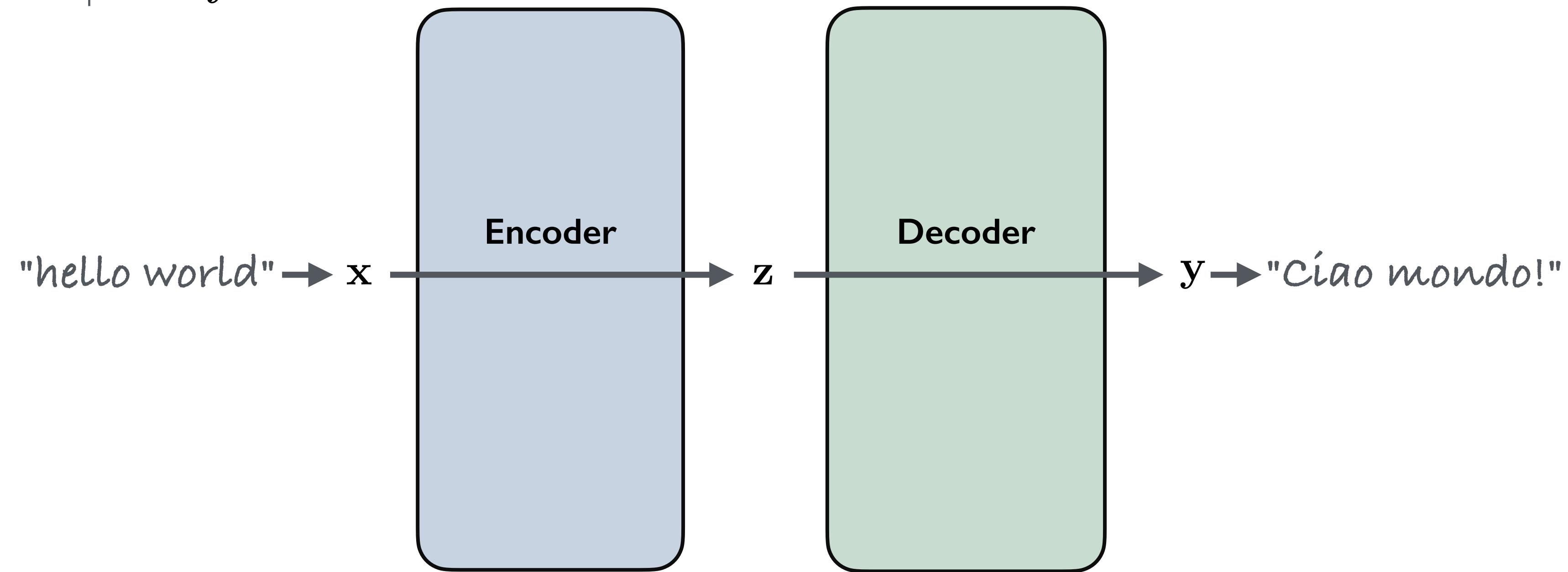
Code
example at
the end!



Transformer: Overview

Encoder-Decoder Architecture

Mapping from input sequence of tokens \mathbf{x}
to sequence of learned encodings \mathbf{z} ,
decoded into output sequence \mathbf{y} .



Transformer: Overview

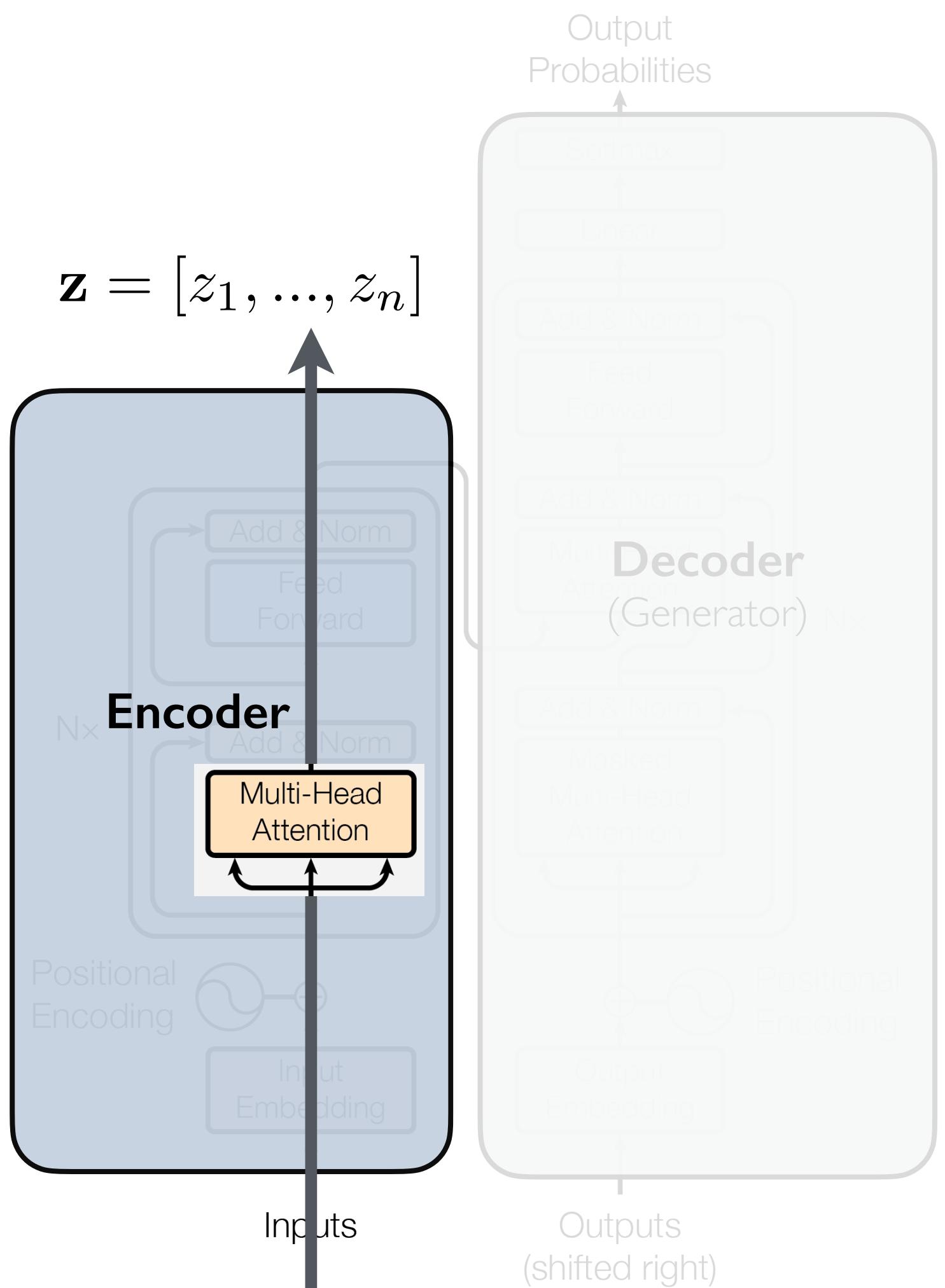
Transformer Encoder

Mapping from input sequence of tokens \mathbf{x} to sequence of learned encodings \mathbf{z} .

Main Goal: "high level, contextualized" representation of input tokens.

Meaning: each z_i can "look at" all x_i .

"hello world" $\rightarrow \mathbf{x} = [x_1, \dots, x_n]$



Transformer: Overview

Transformer Decoder

output
→ encoded input

Goal: $p(\mathbf{y}|\mathbf{z})$

E.g., predict probability distribution over all possible sentences \mathbf{y} in the target language, conditioned on the input sentence \mathbf{z} in the source language.

Approach: decompose using chain rule

$$p(\mathbf{y}|\mathbf{z}) = p(y_1|\mathbf{z}) \cdot p(y_2|y_1, \mathbf{z}) \cdot p(y_3|y_2, y_1, \mathbf{z}) \cdot \dots$$

"Hello world!" → $\mathbf{x} = [x_1, \dots, x_n]$

$$p(y_{t+1}|\mathbf{y}, \mathbf{z})$$

Output
Probabilities

$$\mathbf{z} = [z_1, \dots, z_n]$$



Outputs
(shifted right)

$$\mathbf{y} = [y_1, \dots, y_t]$$

$$\mathbf{y} = [y_1, \dots, y_m] \rightarrow \text{"ciao mondo!"}$$

Decoder: Auto-regressive prediction of next token y_{t+1} from the so far predicted tokens \mathbf{y} while cross-attending to the encodings \mathbf{z}

Each p is a full pass through the model (can be slow for large sequences)

Transformer: Overview

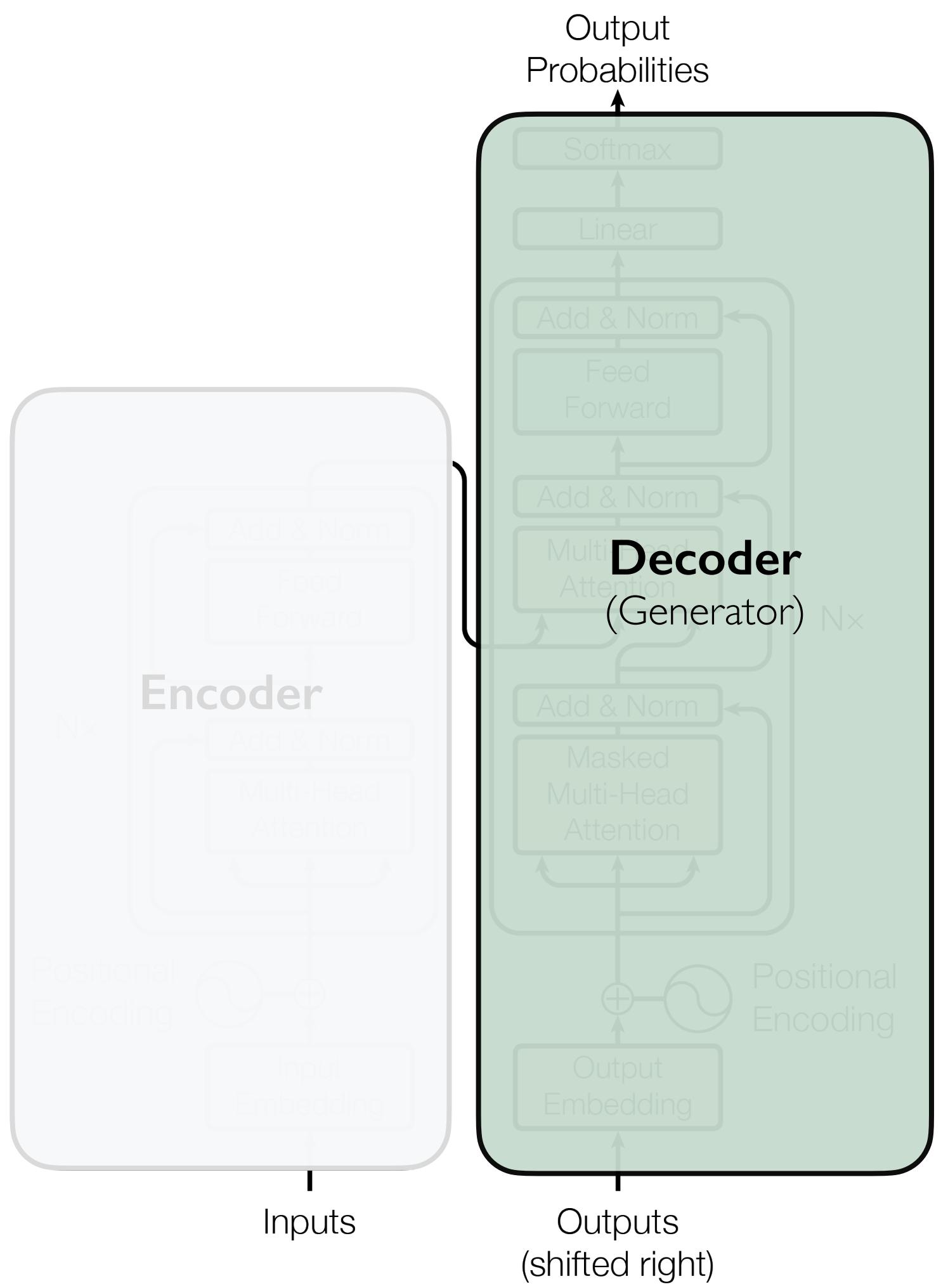
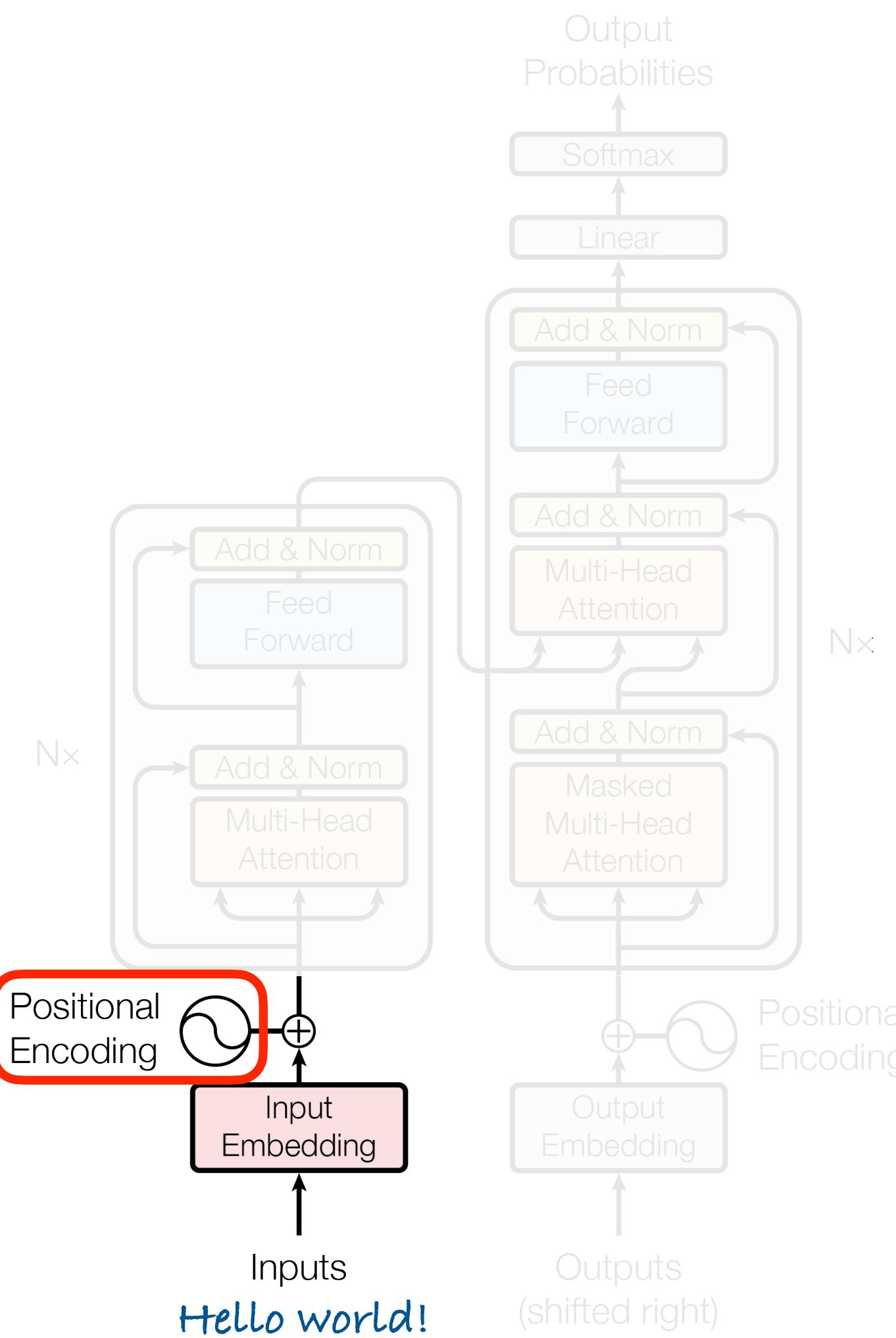


Image credit: Vaswani et al. "Attention is all you need"

Transformer: Encoder



Input Embedding

1) Split the input text into pieces, e.g., characters, words, "tokens":

Hello world! --> [Hel, lo_, wor, ld!]

Another (trivial) Vocabulary based on characters

| | | | | |
|-------|---|---|---|-----|
| char | a | b | c | ... |
| index | 0 | 1 | 2 | ... |

hello --> [7, 4, 11, 11, 14]

2) Tokens are *indices* into the model's "vocabulary":

[Hel, lo_, wor, ld!] --> [34, 23, 89, 42]

3) Each vocabulary entry corresponds to a learned d-dimensional vector:

[34, 23, 89, 42] --> $d \times d$ matrix = [[-0.789, -0.708, ...], [...], [...], [...]]

Positional Encoding

Problem: Attention is permutation invariant (like a dict), but language is not!

The food was good, not bad at all. \neq *The food was bad, not good at all.*

Solution: Need to encode position of each token, and add:

$$\text{Input } x = \begin{matrix} \text{Hel} & \text{lo} & \text{wor} & \text{ld} \end{matrix} + \begin{matrix} 0 & 1 & 2 & 3 \end{matrix}$$

Transformer: Encoder

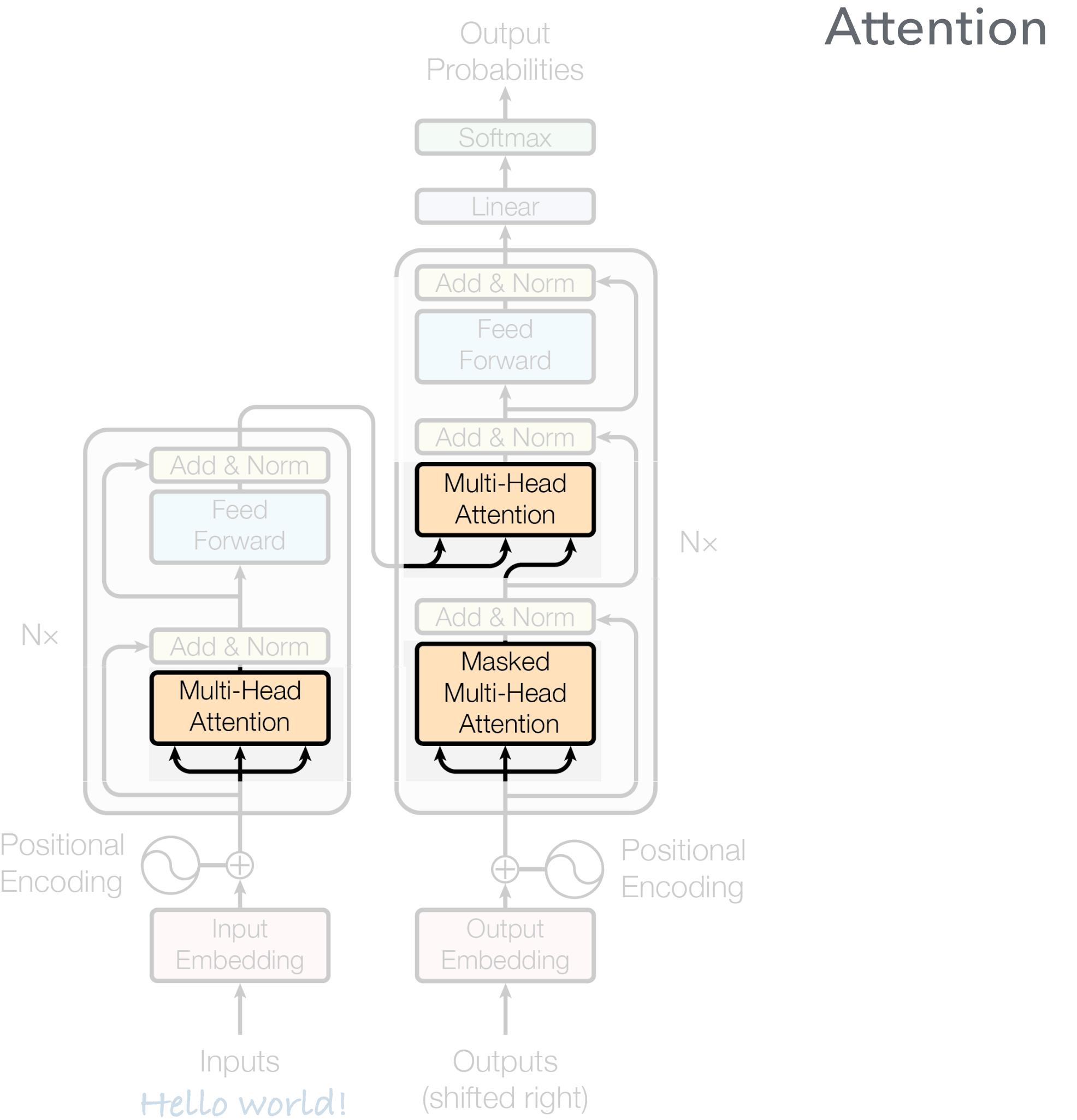


Image credit: Vaswani et al. "Attention is all you need"
Slide inspired by Lucas Beyer

Output

Multi-Head
Attention

Query Key Value

Transformer: Attention



Attention

Query



SEONG-JIN CHO – Piano Concerto in E minor, Op. 11 (final stage of the Chopin Competition 2015)

17 MIO. Aufrufe · vor 8 Jahren

Chopin Institute ✓

17TH INTERNATIONAL FRYDERYK CHOPIN PIANO COMPETITION 18 October 201...



Allegro maestoso | Romance. Larghetto | Rondo... 3 Stellen ▾

Key 1

Score 1 = 0.05



Attention mechanism: Overview

116.164 Aufrufe · vor 11 Monaten

Google Cloud Tech ✓

This video introduces you to the attention mechanism, a powerful technique that allows neural networks to focus on specific parts ...

Untertitel

Key 2

Score 2 = 0.91



Day 1 Highlights // ROCKWOOL Canada Sail Grand Prix | SailGP

13.849 Aufrufe · vor 19 Stunden

SAIL GP

Subscribe today <https://bit.ly/2IZR3n4> Sign up to the SailGP fan loyalty program, The Dock at sailgp.com/dock Follow SailGP On ...

Neu

Key 3

Score 3 = 0.04

Transformer: Attention

Attention as "soft" dictionary lookup

1) **Attention weights a** are **query-key** similarities:

$$\hat{a}_i = \mathbf{q} \cdot \mathbf{k}_i$$

Normalize via softmax: $a_i = e^{\hat{a}_i} / \sum_j e^{\hat{a}_j}$

2) **Output z** is the **attention**-weighted sum of **values v** :

$$\mathbf{z} = \sum_i a_i \mathbf{v}_i = \mathbf{a} \cdot \mathbf{v}$$

3) Usually, **k** and **v** are derived from the same **input x** :

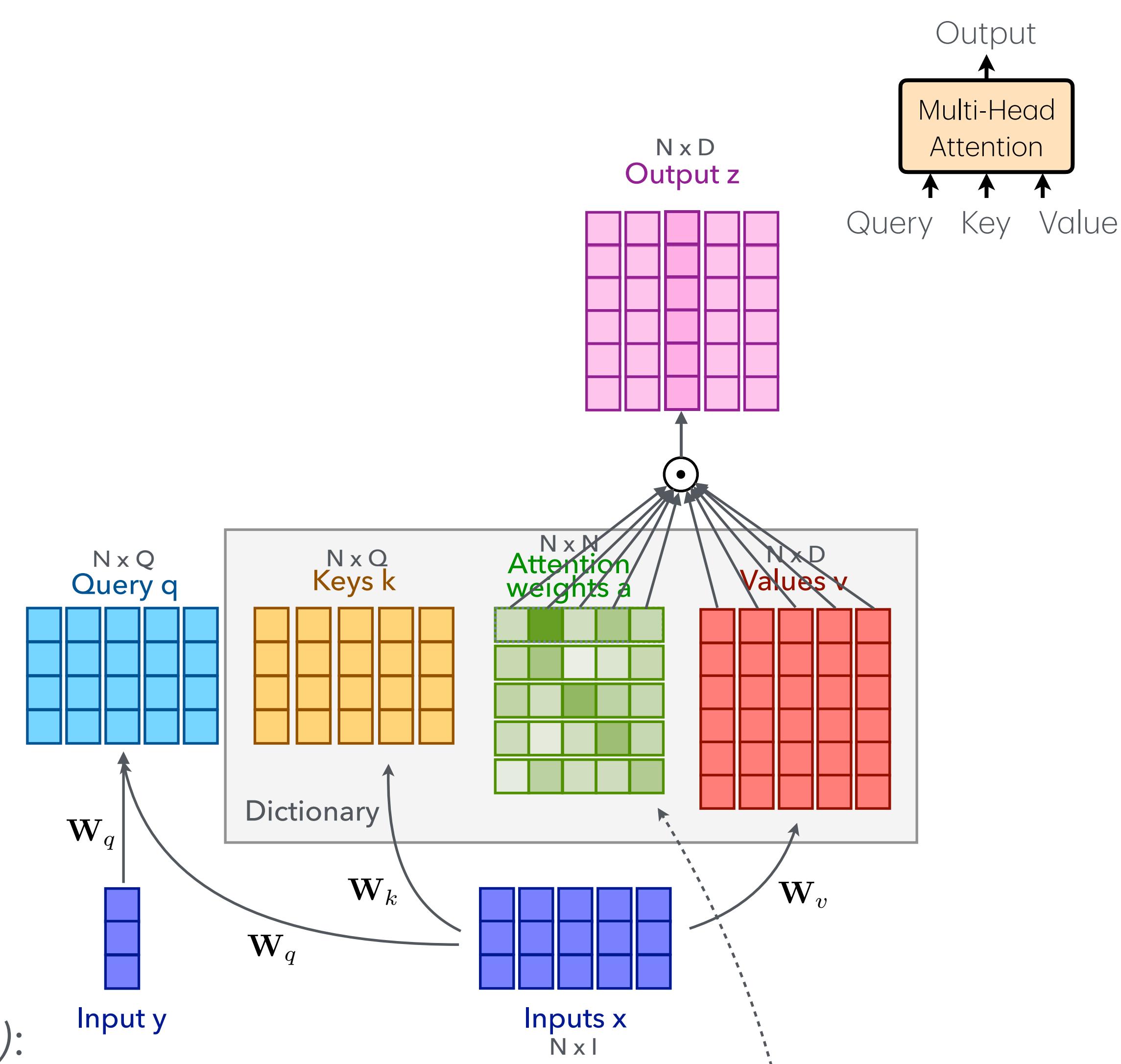
$$\mathbf{k} = \mathbf{W}_k \cdot \mathbf{x} \quad \mathbf{v} = \mathbf{W}_v \cdot \mathbf{x}$$

4) **Query q** comes from a separate **input y** (cross-attention):

$$\mathbf{q} = \mathbf{W}_q \cdot \mathbf{y}$$

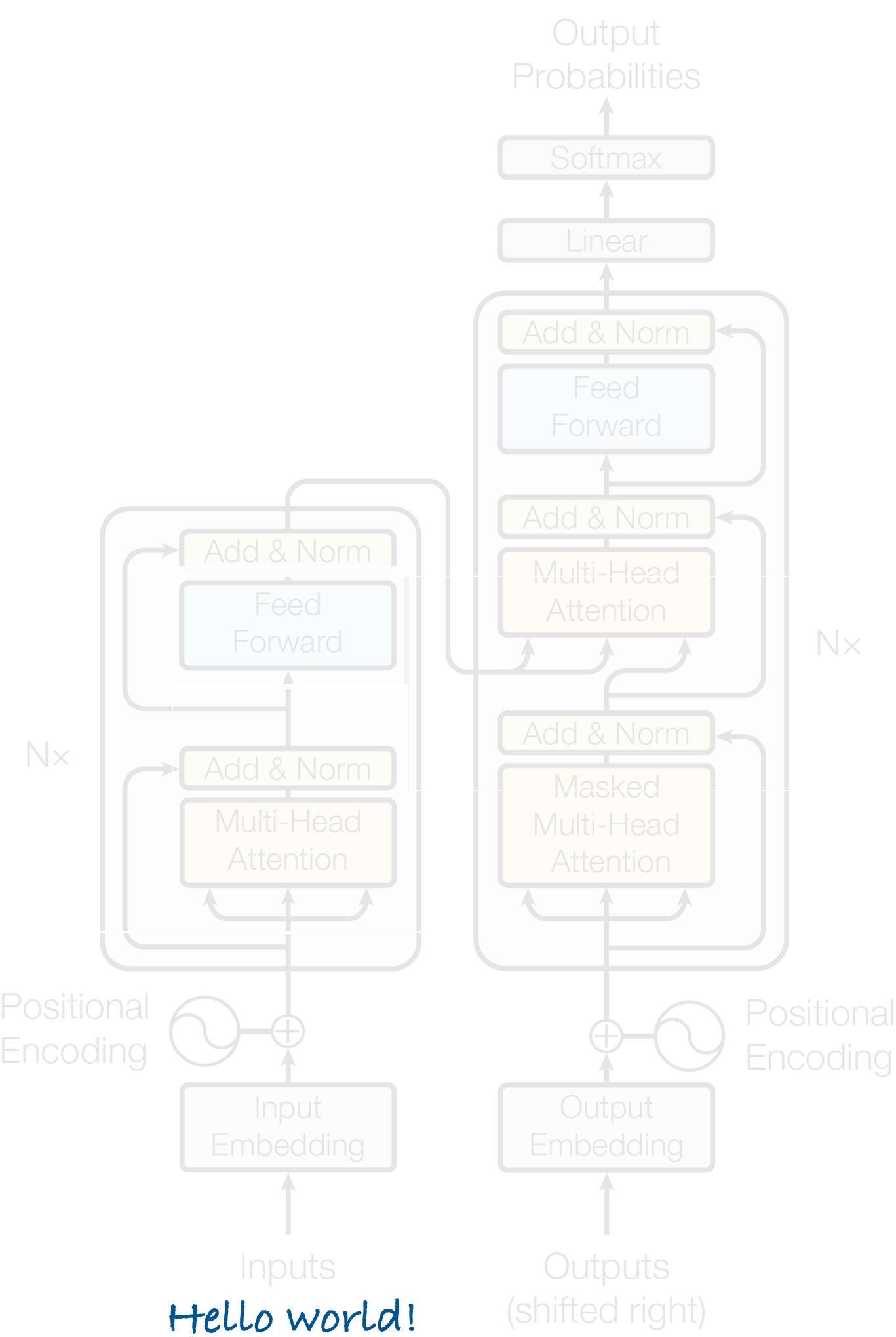
or from the same **input x** (self-attention):

$$\mathbf{q} = \mathbf{W}_q \cdot \mathbf{x}$$



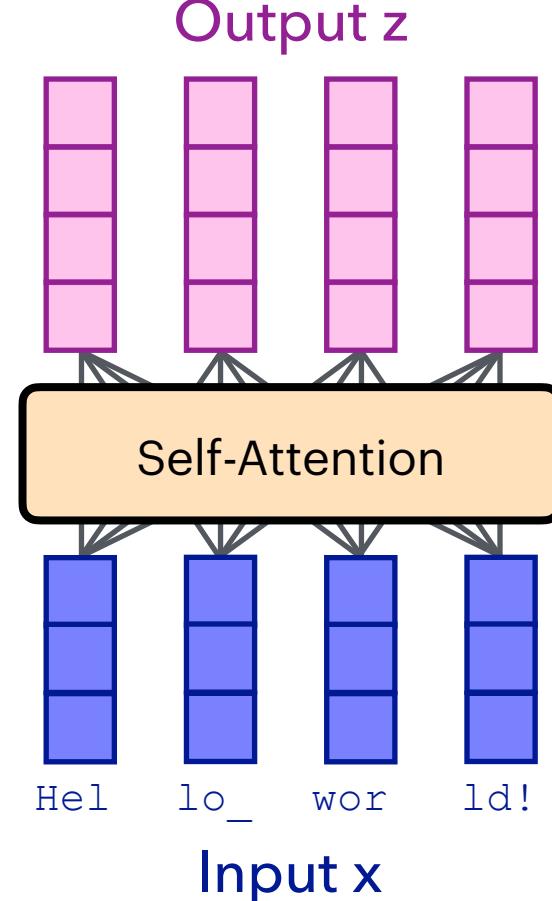
Note that the complexity is $\mathcal{O}(N^2)$
(N : #input tokens / context length)

Transformer: Encoder



Self Attention

The **input sequence** is used to create **queries**, **keys** and **values**. Each token can "look around" the whole input, and decide how to update the **output representation** based on what it sees.

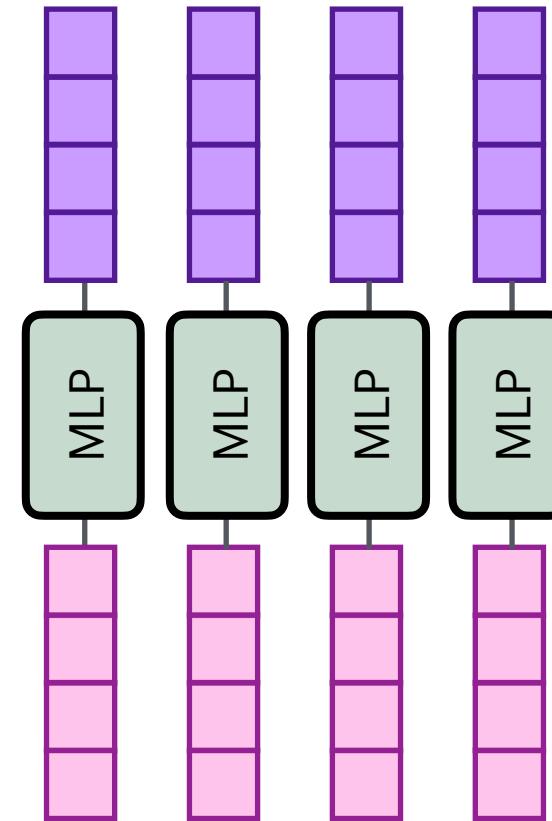


Feed Forward

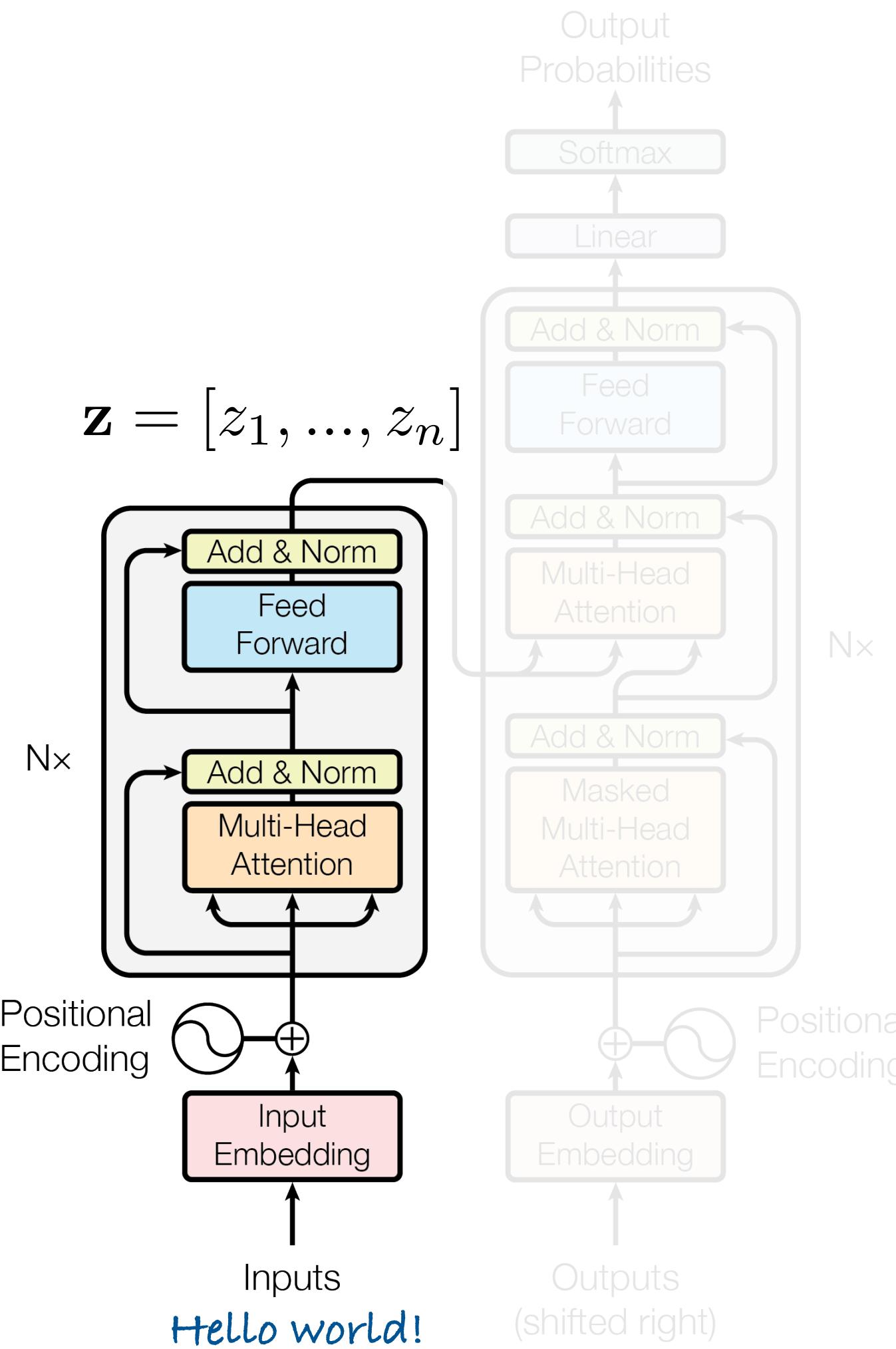
A simple MLP applied to each token individually:

$$z_i = W_2 \text{ReLU}(W_1 x + b_1) + b_2$$

Meaning: each token is pondering for itself about what it has observed previously.



Transformer: Encoder



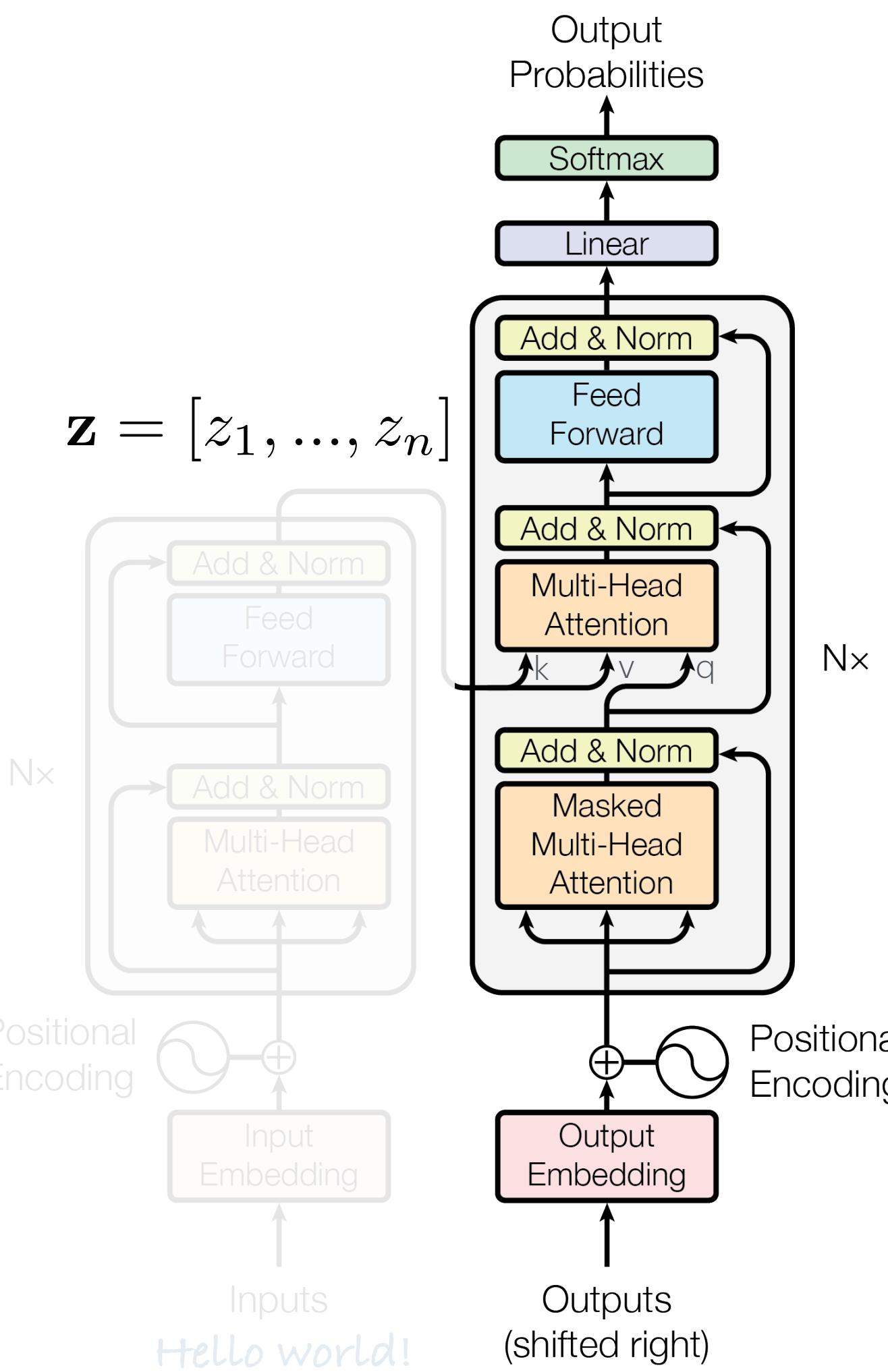
Encoder

Since the input and output dimensions are identical, we can stack N such encoder blocks.

The encoder output is a "processed" version of the input tokens, i.e., an intermediate representation.

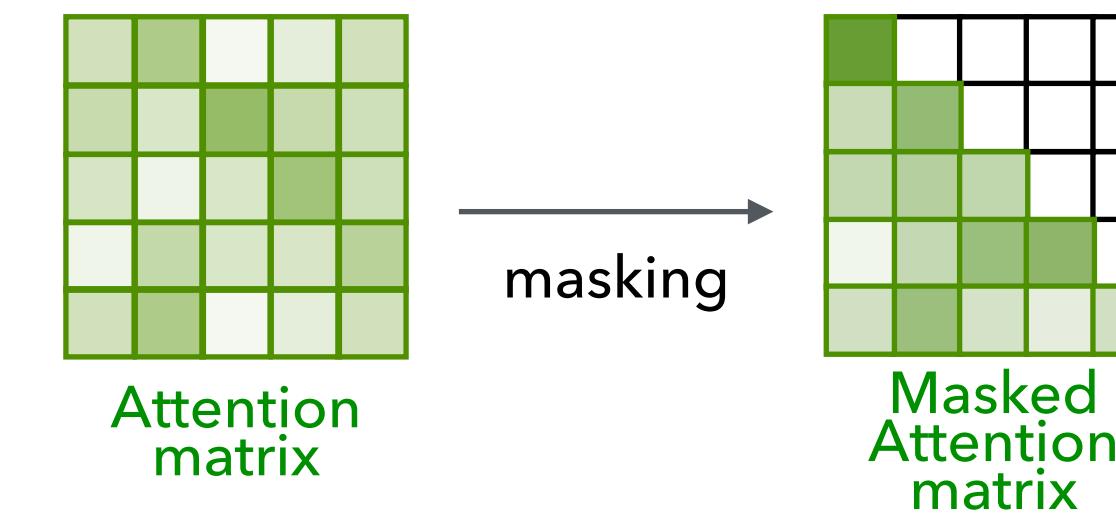
The actual transformer output, comes from the decoder.

Transformer: Decoder (next week)



Masked Self-Attention

Like regular self-attention (as in the encoder), but masked.
Restricts each token to see only the already generated ones.



Cross-Attention

Each decoded token can "look at" the encoder's output.
Queries come from decoder, keys and values from the encoder.

Transformer: Summary

1. Sequence modeling using RNN/LSTM
2. Motivation for the attention mechanism
3. Attention mechanism
4. Transformer (encoder)

Next week: Transformer (decoder)



References and Further Reading

- Slides:



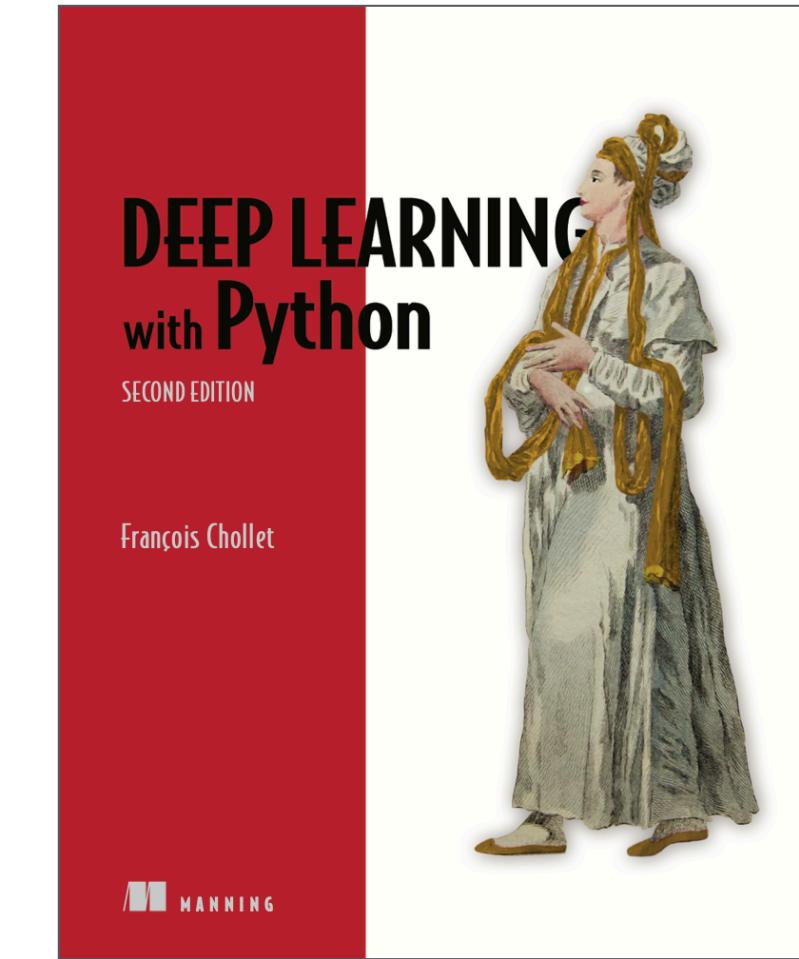
- Example Code:



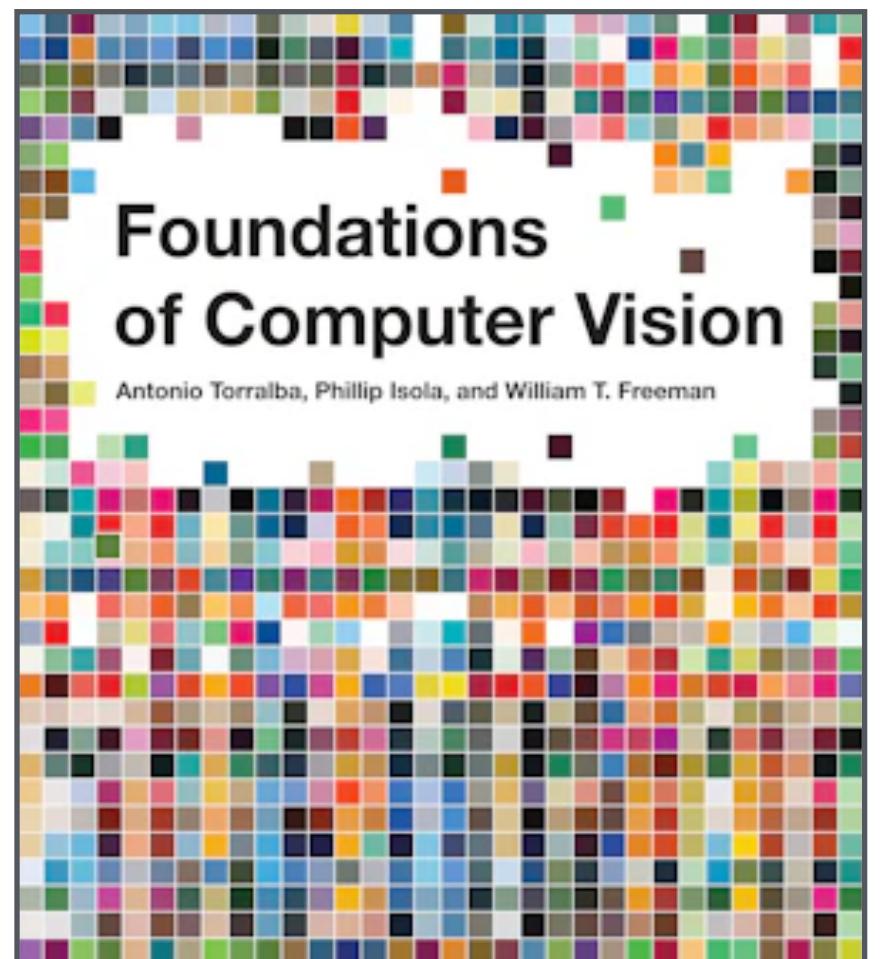
- Book Chapters on Transformers:

- Deep Learning with Python (François Chollet)
- Foundations of Computer Vision (Torralba, Isola, Freeman)

- Code example MinGPT by Andrej Karpathy <https://github.com/karpathy/minGPT?tab=readme-ov-file>
- RNN/LSTM by Chris Olah <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Lecture slides by Lucas Beyer <http://lucasbeyer.be/transformer>



Chapter 11.4, 11.5



Chapter 26, 51