CS 639: Foundation Models
Transformers & Attention I

Fred Sala

University of Wisconsin-Madison

Feb. 10, 2026

Announcements

- **Homework 1: delayed to this evening**
  - Bonus OH: Tomorrow (Weds) at 2:30-4:00 PM
- **Resources**
  - https://jalammar.github.io/illustrated-transformer/ Very nice resource for following along
- Class roadmap:

Tuesday Feb. 10	Transformers and Attention I
Thursday Feb. 12	Transformers and Attention II
Tuesday Feb. 17	Architectures: Encoder-Only
Thursday Feb. 19	Architectures: Others

Start FMs and Arch

Outline

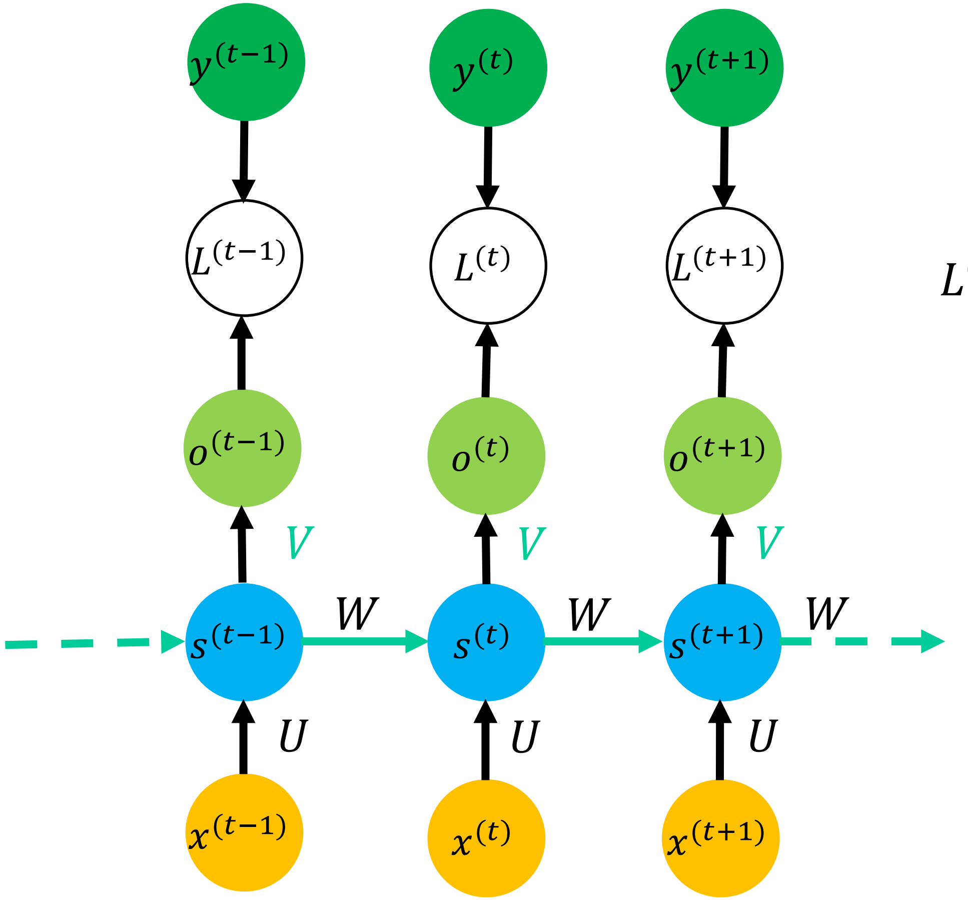
- **Basic Attention**
  - Notions of attention, self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Transformers**
  - Architecture, encoder and decoder setups

Outline

- **Basic Attention**
  - Notions of attention, self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Transformers**
  - Architecture, encoder and decoder setups

History of Attention

- Recall our RNN: all the information had to go in the state *s*
  - A **fixed-length context vector**



$$
\begin{aligned}
a^{(t)} &= b + Ws^{(t-1)} + Ux^{(t)} \\
s^{(t)} &= \tanh\left(a^{(t)}\right) \\
o^{(t)} &= c + Vs^{(t)} \\
\hat{y}^{(t)} &= \mathrm{softmax}\left(o^{(t)}\right) \\
L^{(t)} &= \mathrm{CrossEntropy}(y^{(t)}, \hat{y}^{(t)})
\end{aligned}
$$

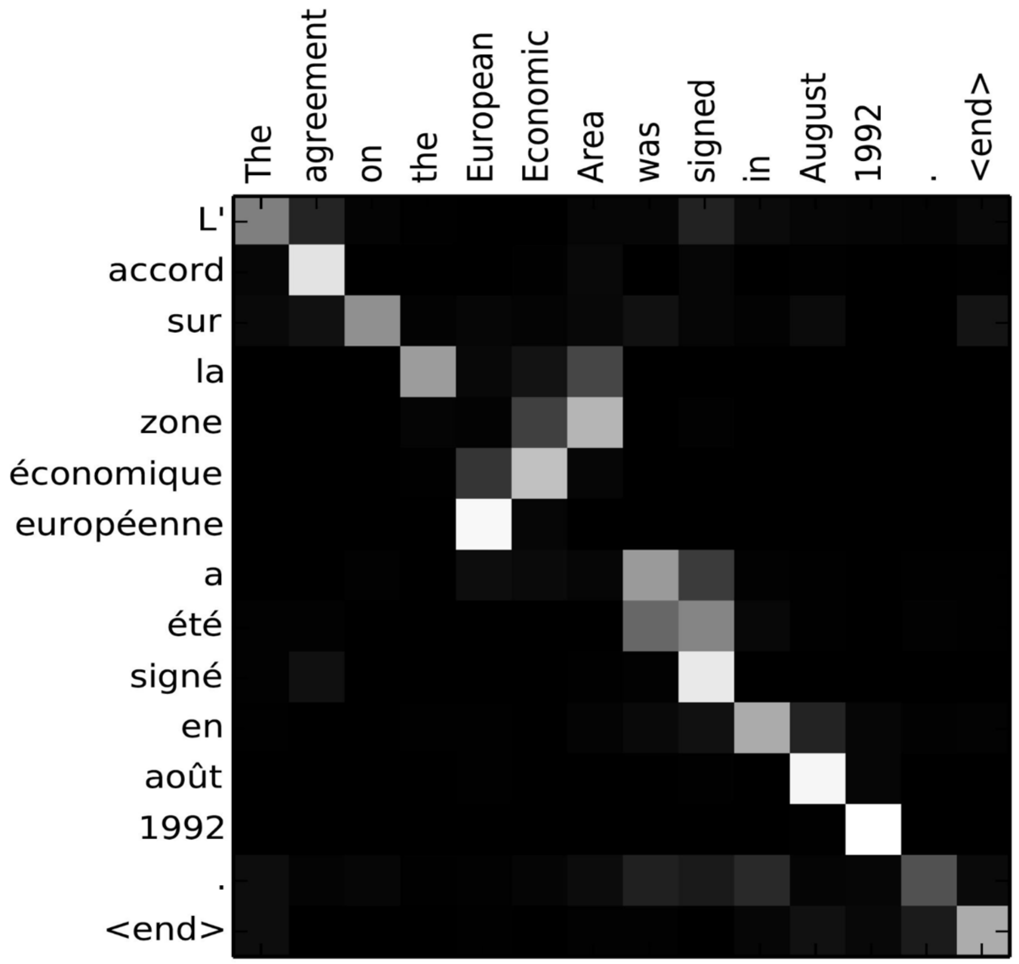
History of Attention

Basic motivation: what if the **fixed length context vector is not enough**

- Why?
  - Words depend on each other
  - Dependencies are complex

- Need: mechanism to help model **focus** on the right “part”

Lots of approaches from 2014 on
- Bahdanau et al, 2014

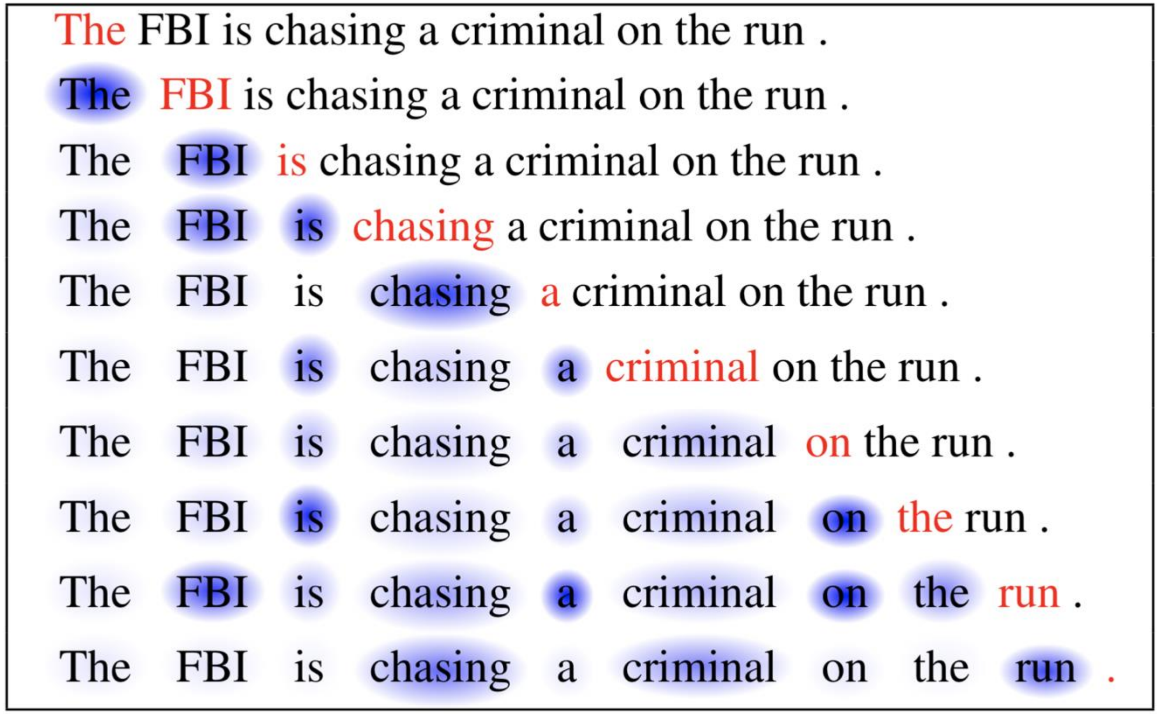


Bahdanau et al, 2014

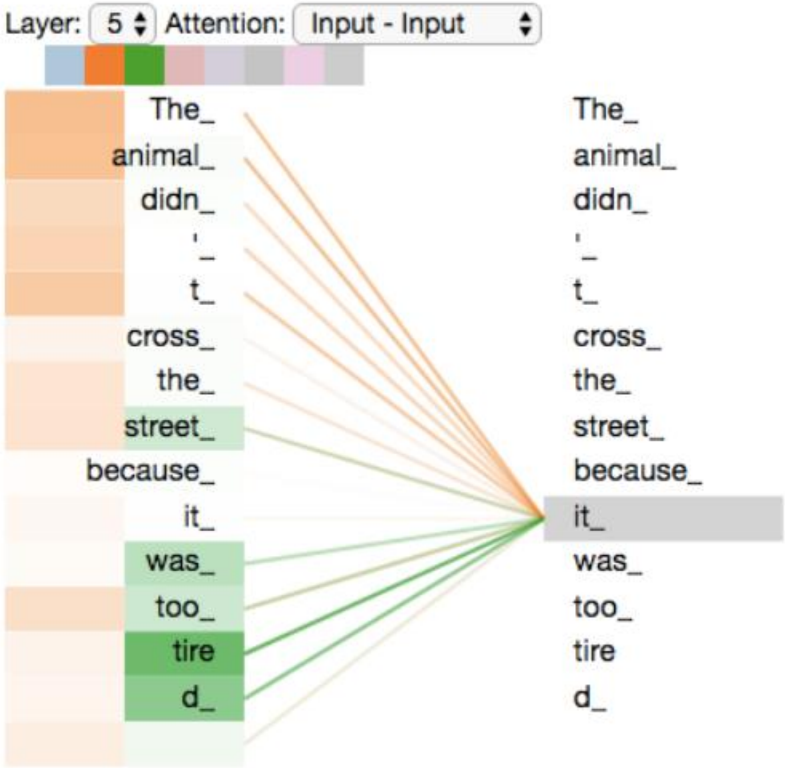
Self-Attention: Motivation

Popularized from 2017 on…

- From bottom-up. Let's design a basic layer.
  - Intuition: dependencies **within** same sentence



Cheng et al, 2016



Jay Alammar

Building Layers

Recall how we built a convolutional layer

- We defined an operation (**convolution**)
- We decided how it should operate over the inputs (from previous layer) to produce the layer's outputs

- We'll do the same to build an attention-based layer

- **Input**: vectors for words

Note: All visualizations are due to Jay Alammar

Excellent resource: https://jalammar.github.io/illustrated-transformer/

Word Embeddings

Terminology: related to representations

- Embeddings: fixed-dimensional vector for each *word* in our *vocabulary*
  - Really, for each “token”. First we’ll run a tokenizer to split up raw text into tokens, then we’ll grab the embedding for each token
  - Word2Vec: word co-occurrences ⟷ embedding distances

Self-Attention: Goals and Inputs

From bottom-up. Let's design a basic layer.

- Two criteria
  - *Transform* incoming word vectors,
  - Enable *interactions* between words

- Input: vectors for words

Self-Attention: Retrieval Intuition

- How should we design the interactions?
  - Analogy: **search**

“Which restaurants near me are open at 9:00 pm?”

Query

Key

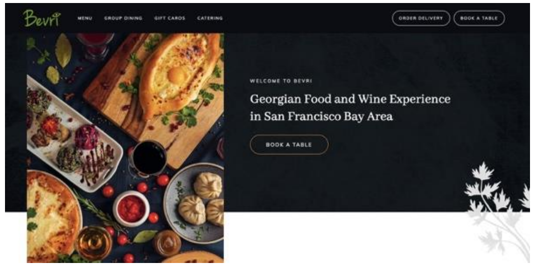
Value

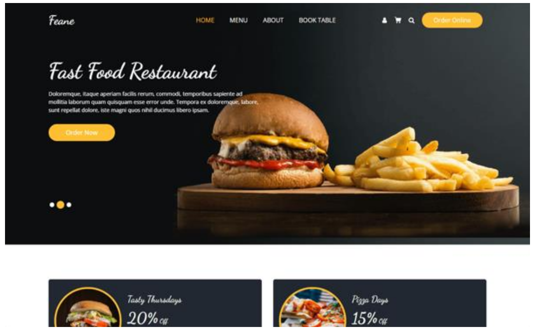
Objects:

Query
Key
Value

Score 0.3

Score 0.7

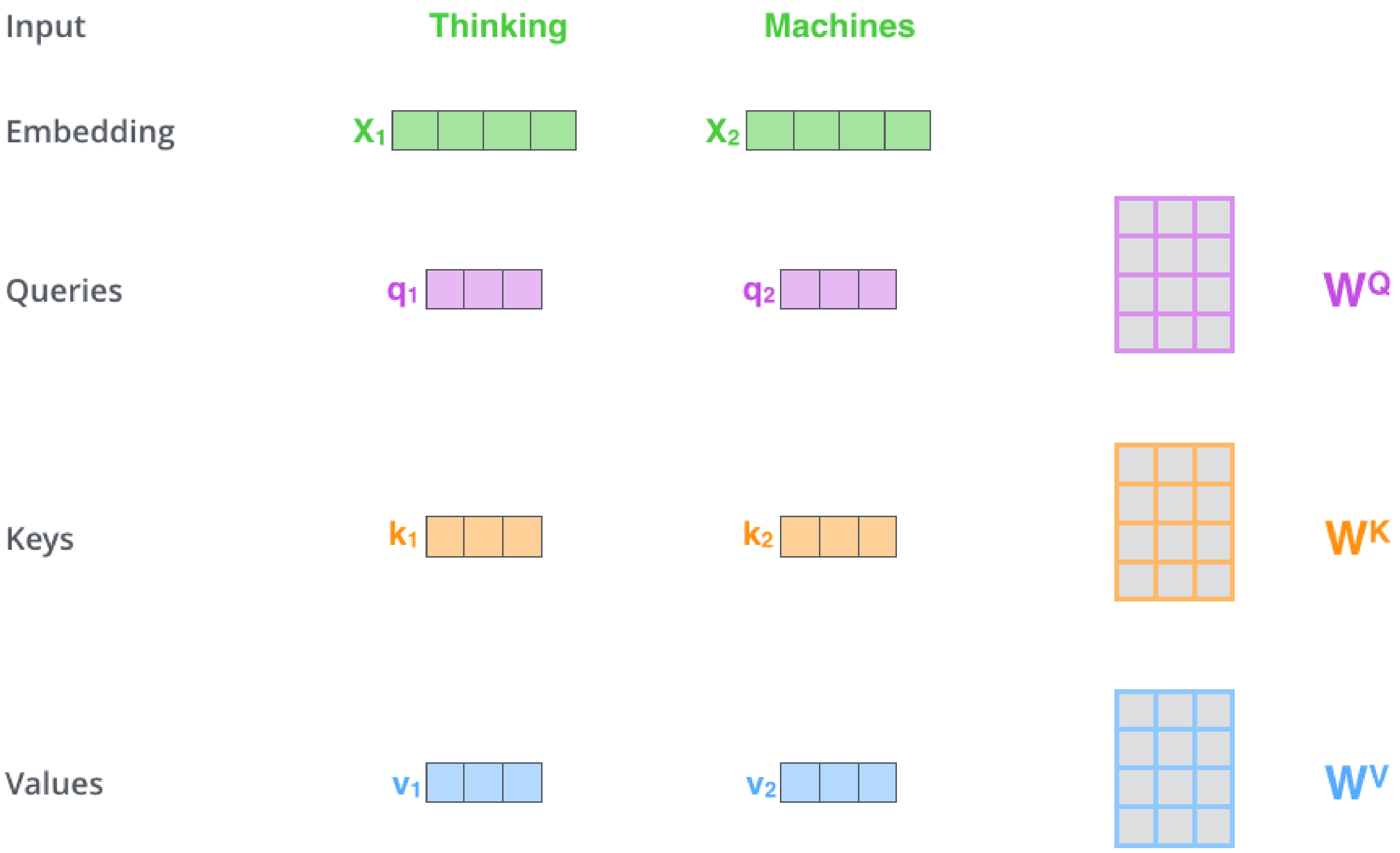




Self-Attention: Query, Key, Value Vectors

- *Transform* incoming word vectors,
- Enable *interactions* between words

- Get our **query, key, value** vectors via weight matrices: linear transformations!



Objects:

Query
Key
Value

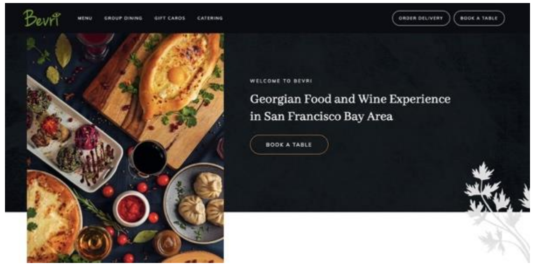
Self-Attention: Score Functions

Have **query, key, value** vectors

- Next, get our **score**

q_1 $\underbrace{\boxed{}\boxed{}\boxed{}}_{\text{Query}}$ k_2 $\underbrace{\boxed{}\boxed{}\boxed{}}_{\text{Key}}$

Score 0.3



- Lots of things we could do --- **simpler** is usually better!
- Dot product $q_1 \bullet k_2 = 96$

- Then we'll do **softmax**

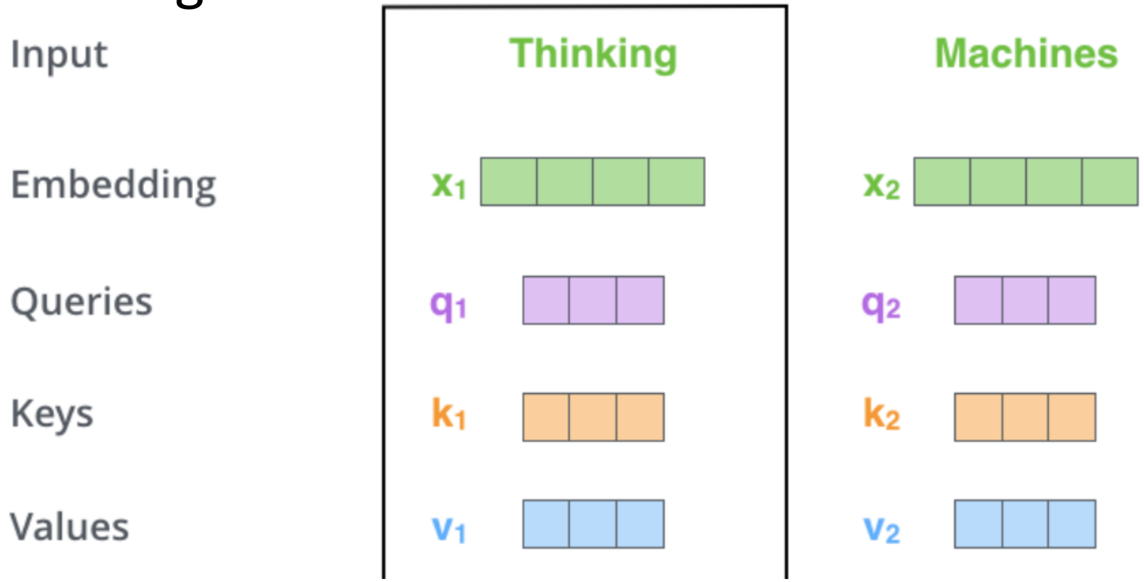
Self-Attention: Scoring and Scaling

- *Transform* incoming word vectors,
- Enable *interactions* between words

- Get our **query, key, value** vectors via weight matrices: linear transformations!
- Compute scores

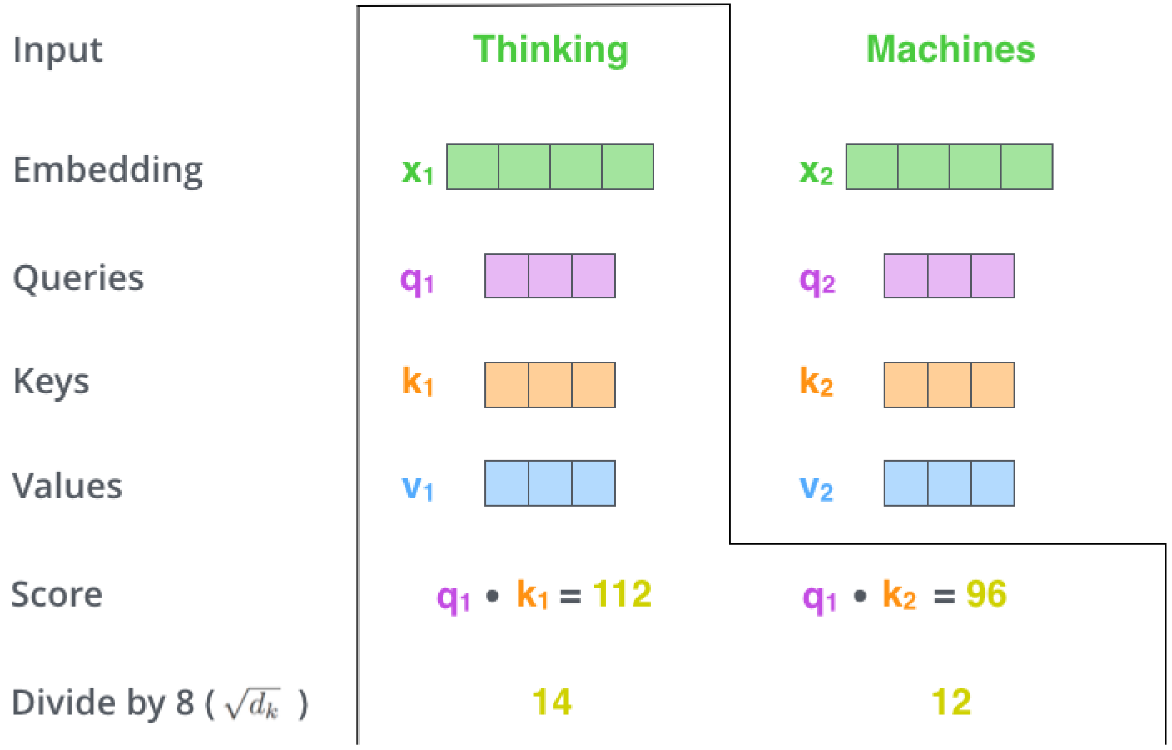
Objects:

Query
Key
Value



Self-Attention: Putting it Together

- Have **query, key, value** vectors via weight matrices: linear transformations!
- Have softmax score outputs (**focus**)
- Add up the values!



Objects:

Query
Key
Value

Self-Attention: Matrix Formulas

- Have **query, key, value** vectors via weight matrices: linear transformations!
- Have softmax score outputs (**focus**)
- Add up the values!

Objects:

Query
Key
Value

$$Q = XW_Q, K = XW_K, V = XW_V$$

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(X\frac{W_Q {W_K}^T}{\sqrt{d_k}}X^T\right)V$$

Break & Questions

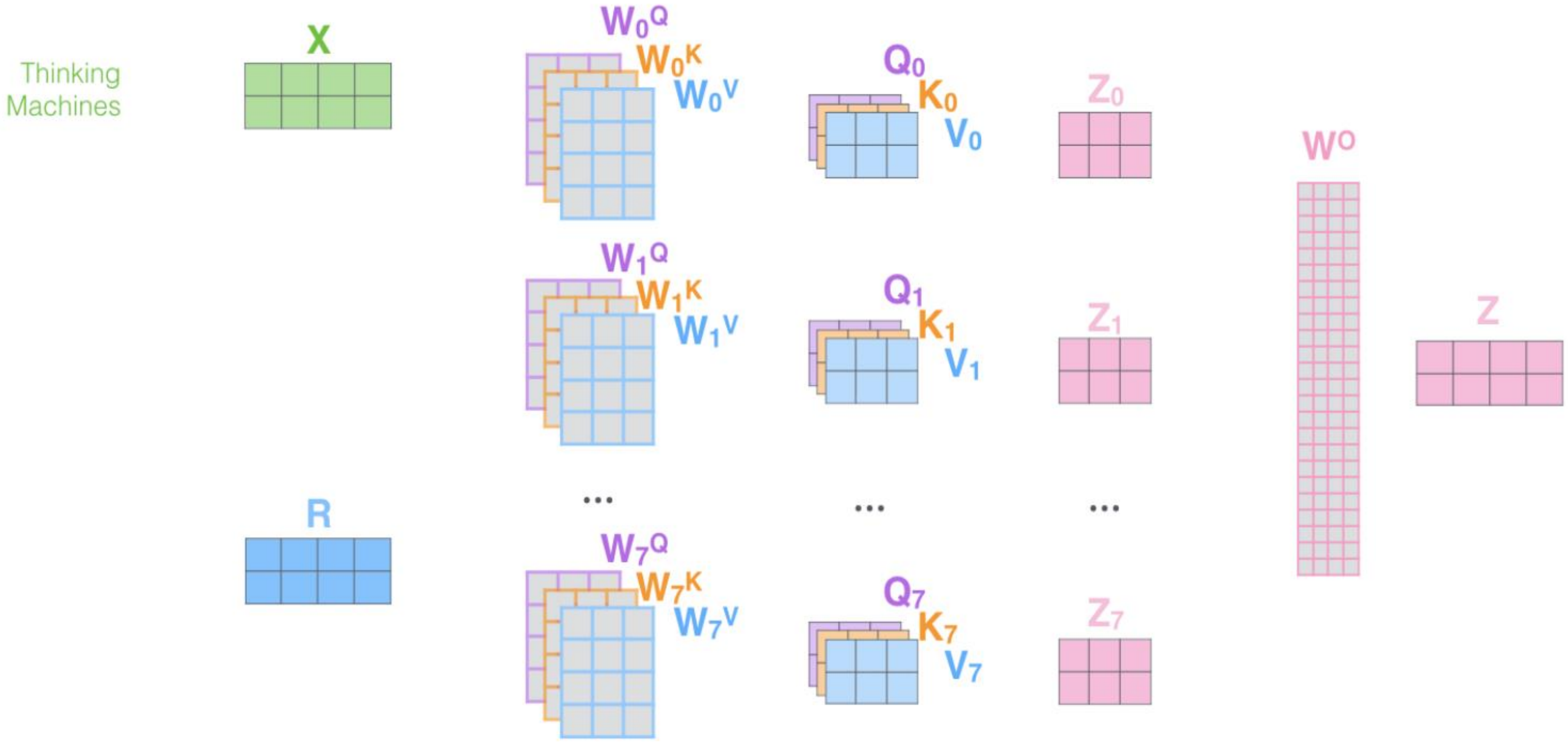
Outline

- **Basic Attention**
  - Notions of attention, self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Transformers**
  - Architecture, encoder and decoder setups

Self-Attention: Multi-head

This is great but will we capture everything in one?

- Do we use just 1 kernel in CNNs? **No!**
- Do it many times in parallel: **multi-headed attention.** Concatenate outputs



Self-Attention: Positional Encodings

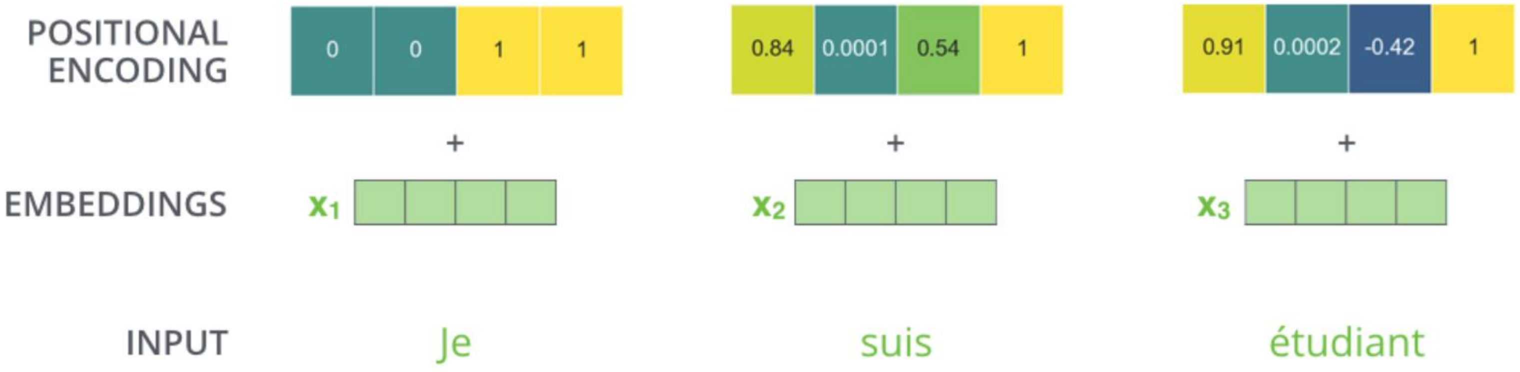
Almost have a full layer designed.

- One annoying issue: so far, order of words **(position) doesn't matter**!
- Solution: add positional encodings

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

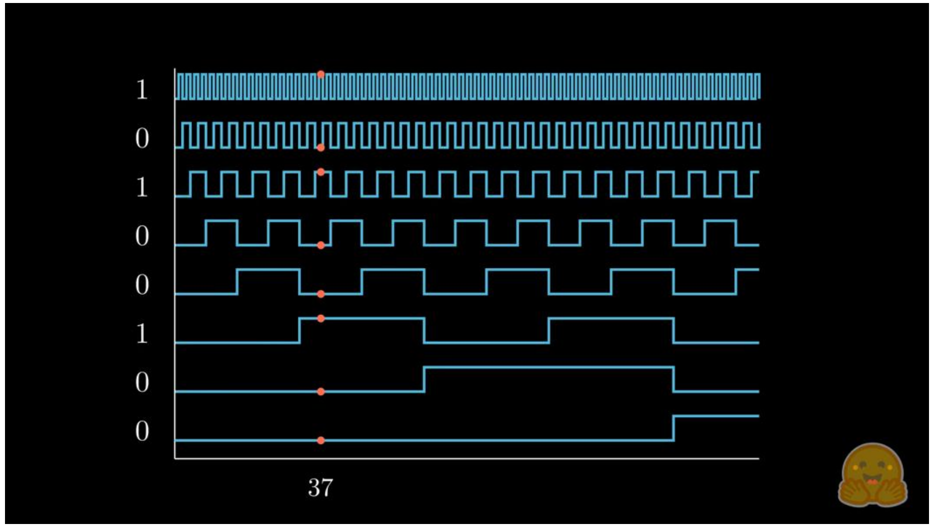
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Location index



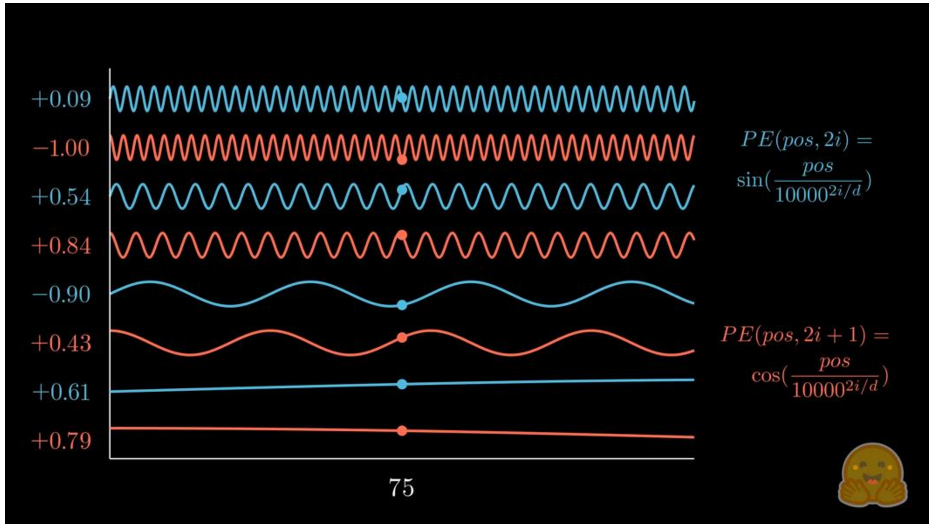
Self-Attention: Positional Encodings

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Why these **mysterious formulas**? Want properties:

- Consistent encoding
- Smooth
- Linearity across positions
  - Alternating sin and cos: can multiply by rotation matrix to obtain shifts



https://huggingface.co/blog/designing-positional-encoding

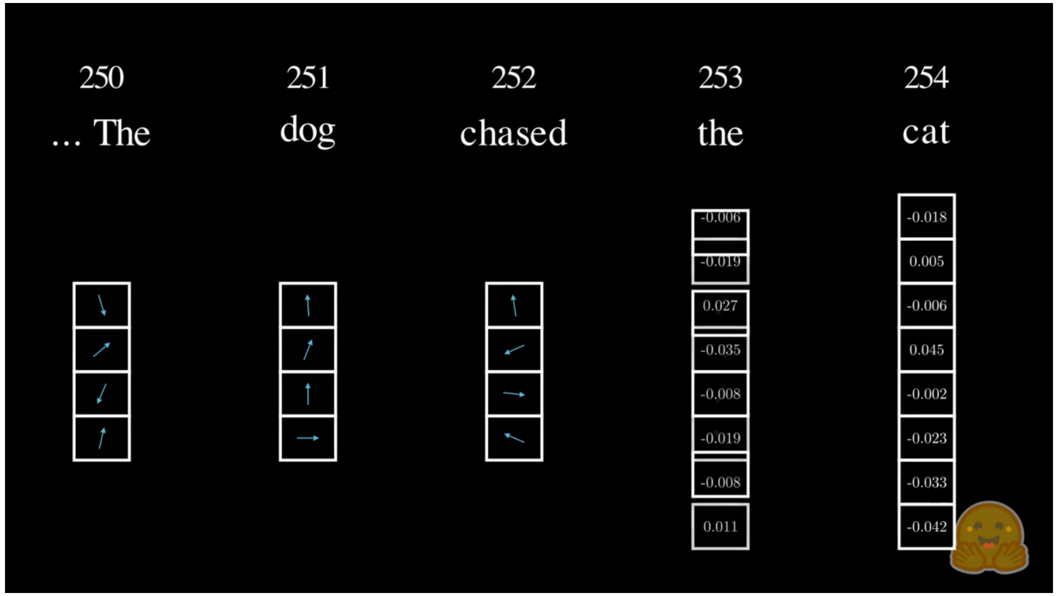
Self-Attention: Modern Positional Encodings

These ***sinusoidal*** embeddings were defined in the original Transformers paper,

- Added once (as we saw) prior to the first layer

Many new variants of positional encodings that behave slightly differently

- Example: ***multiplicative*** instead of ***additive***
- Popular: **Rotary Positional Encoding (RoPE)**
- Note: perform in every attention layer



https://huggingface.co/blog/designing-positional-encoding

Break & Questions

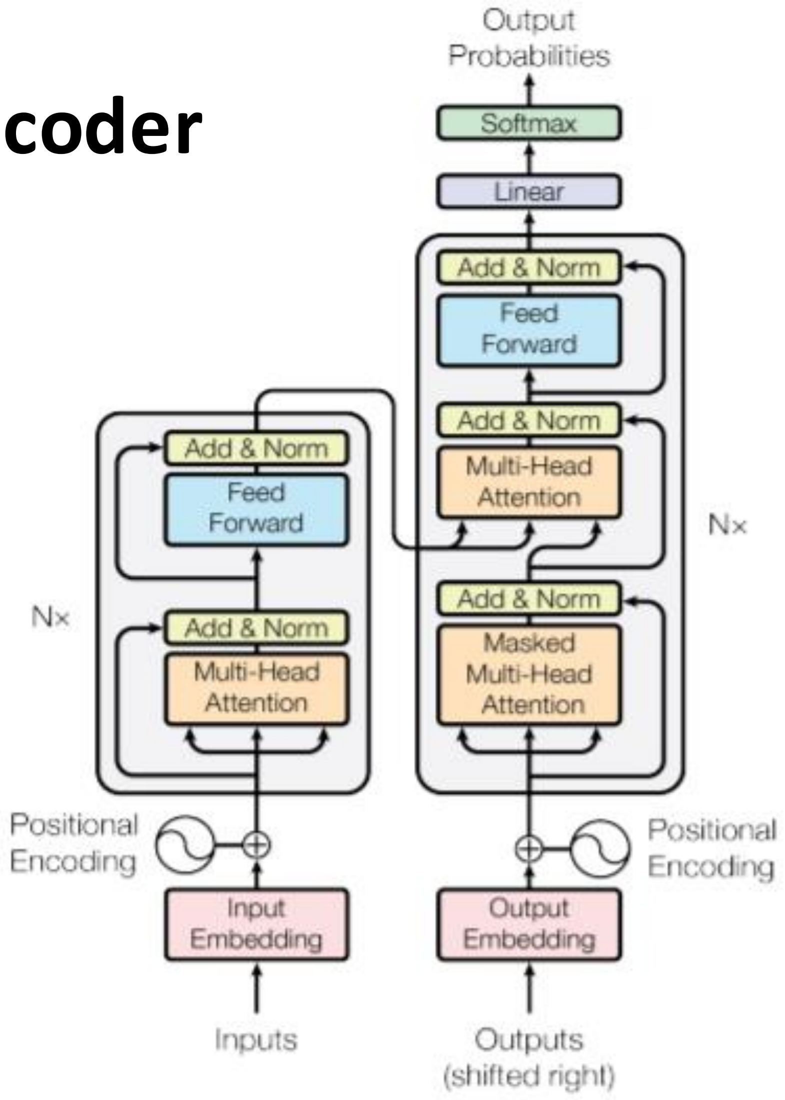
Outline

- **Basic Attention**
  - Notions of attention, self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Transformers**
  - Architecture, encoder and decoder setups

Transformers: Model Architecture

- Initial goal for an architecture: **encoder-decoder**
  - Get **rid of recurrence**
  - Replace with **self-attention**

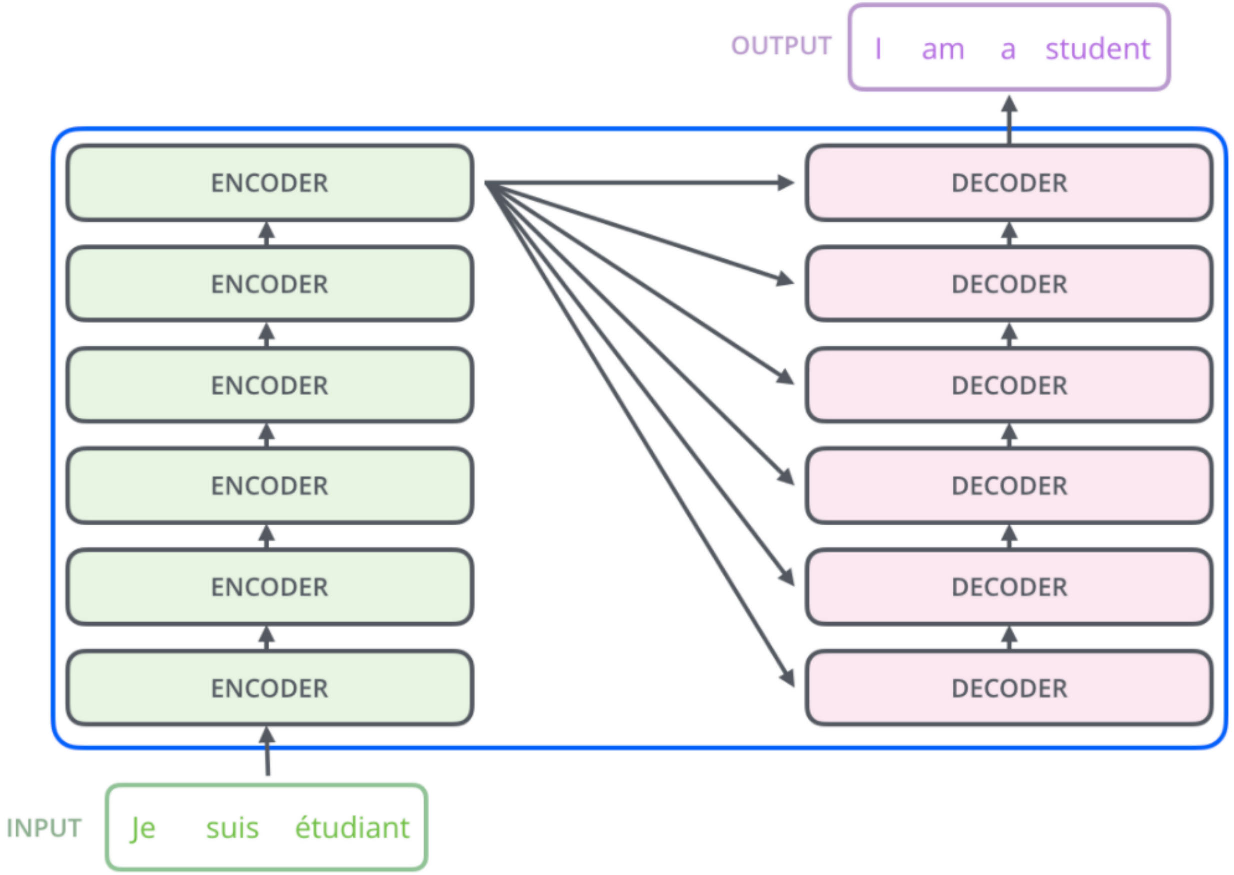
- Architecture
  - You may have seen this picture
  - Centered on self-attention blocks



Vaswani et al. '17

Transformers: Architecture

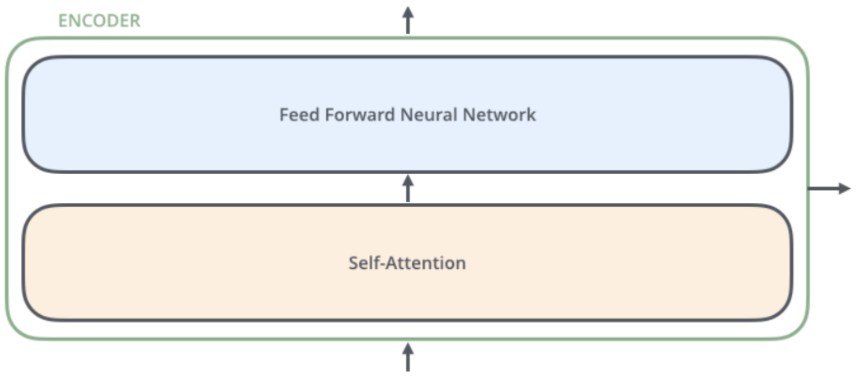
- **Sequence-sequence** model with **stacked** encoders/decoders:
  - For example, for French-English translation:

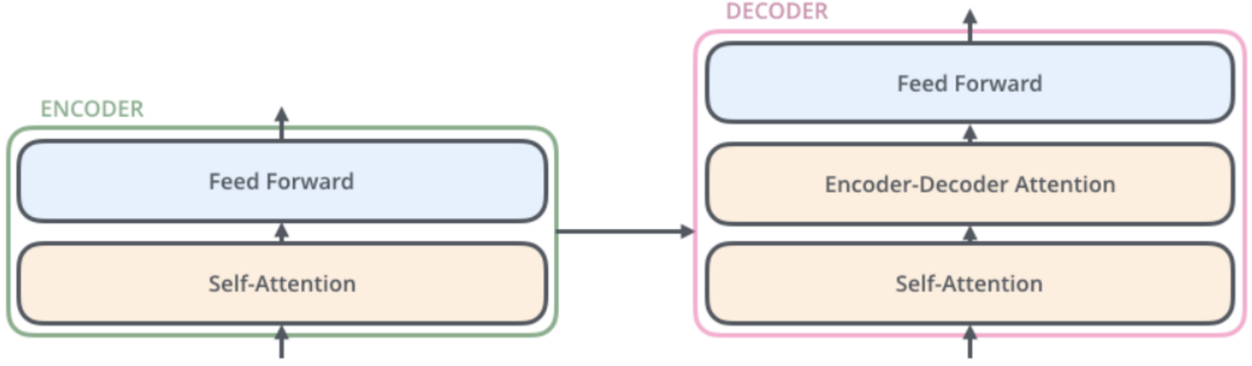


Transformers: Architecture

- Sequence-sequence model with **stacked** encoders/decoders:
  - What's inside each encoder/decoder unit?

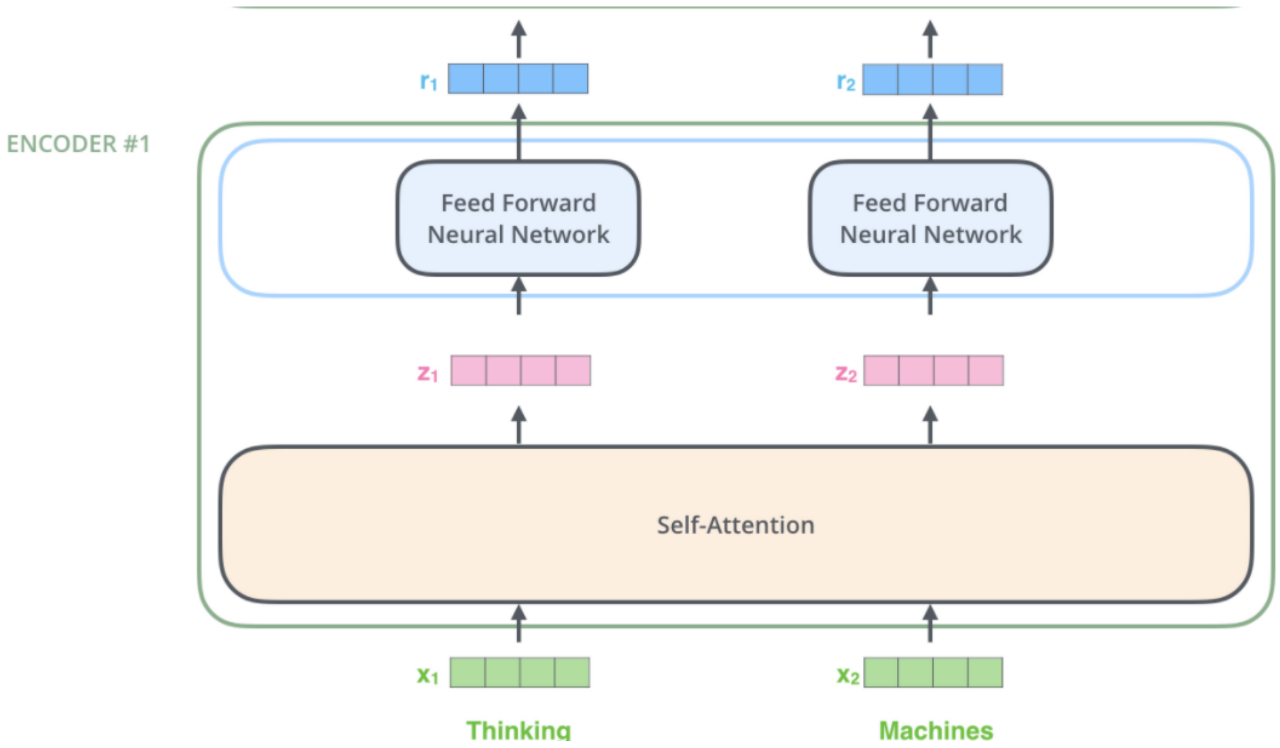
  - Focus encoder first: **pretty simple**! 2 components:
    - Self-attention block
    - Fully-connected layers (i.e., an MLP)





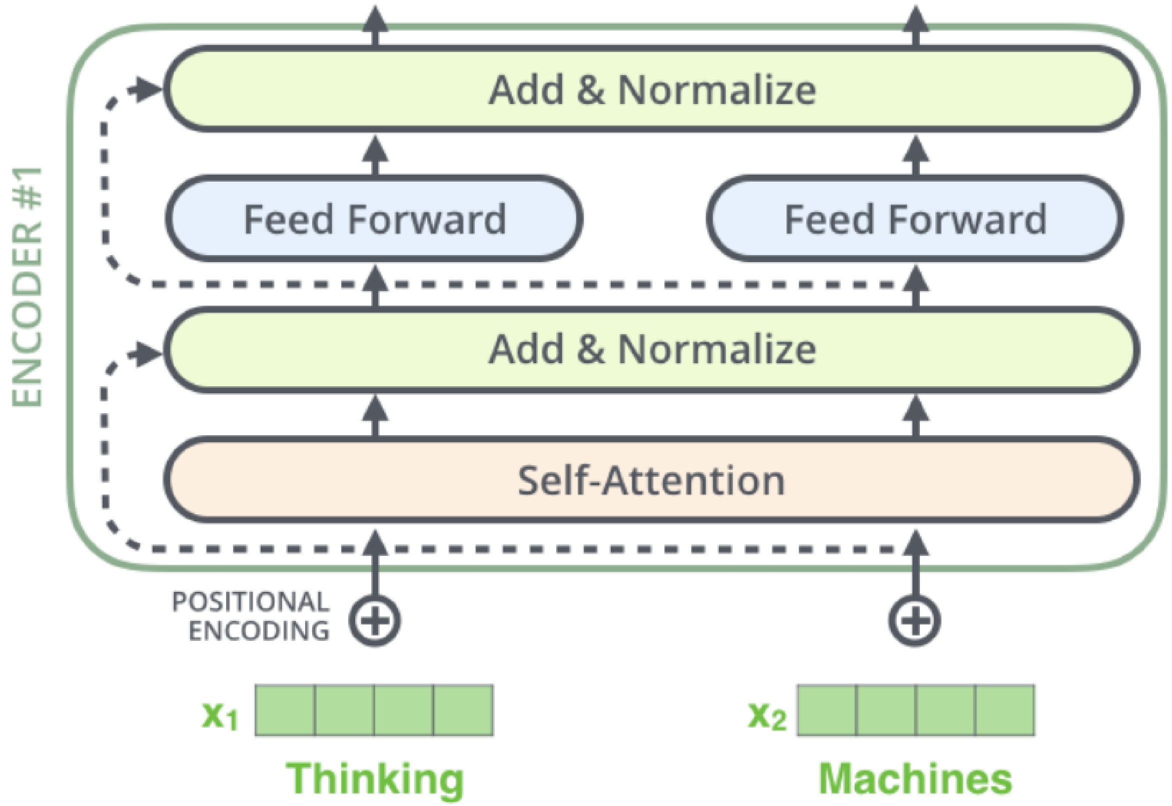
Transformers: Inside an Encoder

- Let's take a look at the encoder. Two components:
  - 1. **Self-attention** layer (covered this)
  - 2. "Independent" **feedforward nets**



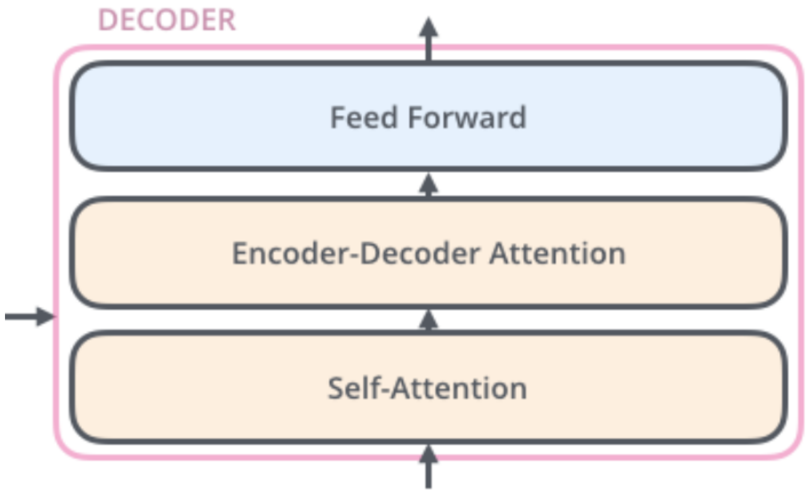
Transformers: More Tricks

- Recall a big innovation for ResNets: residual connections
  - And also layer normalizations
  - Apply to our encoder layers



Transformers: Inside a Decoder

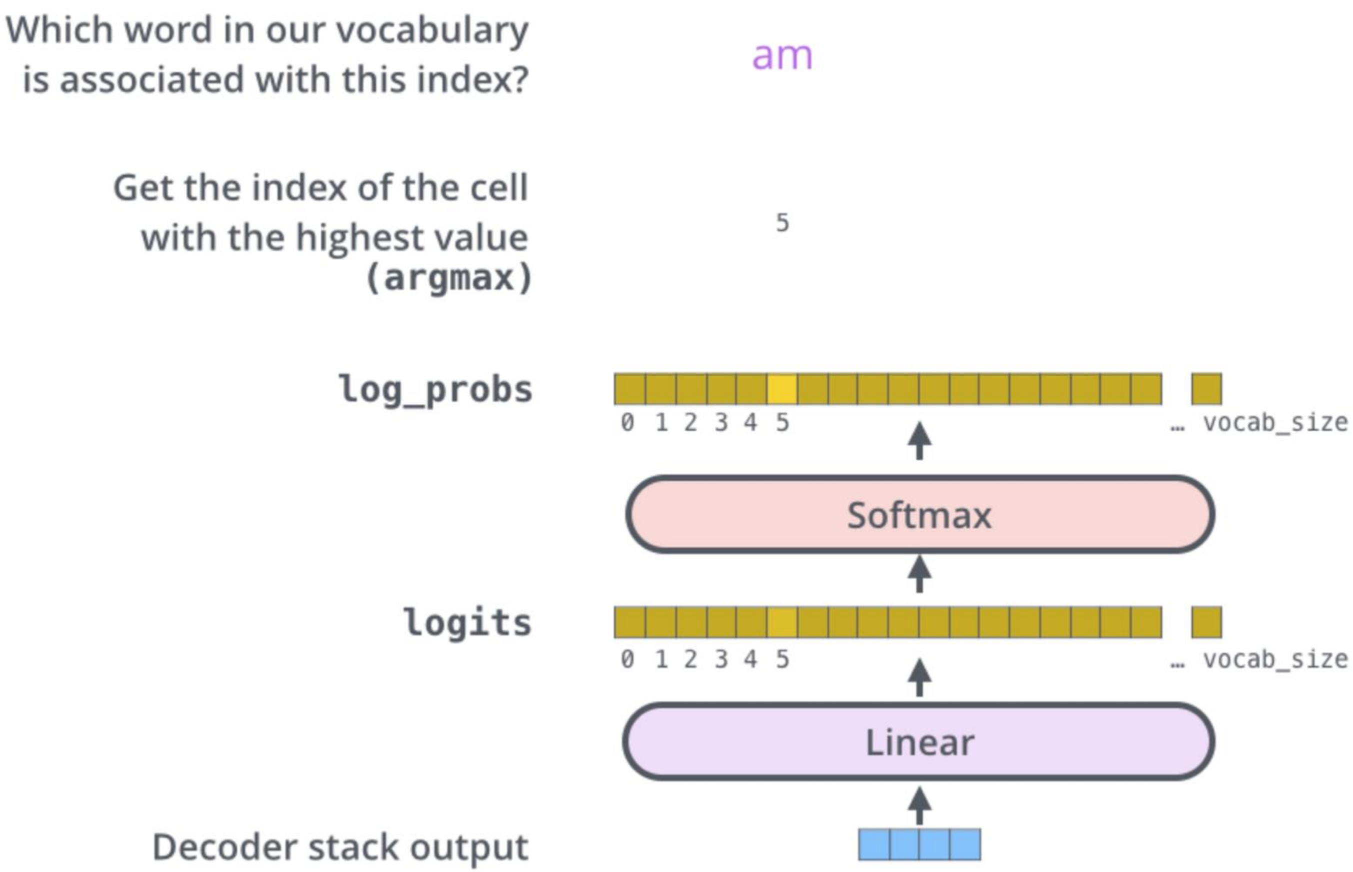
- Let’s take a look at the decoder. Three components:
  - 1. **Self-attention** layer (covered this)
  - 2. Encoder-decoder attention (same, but K, V come from encoder)
  - 3. “Independent” **feedforward nets**



Transformers: Last Layers

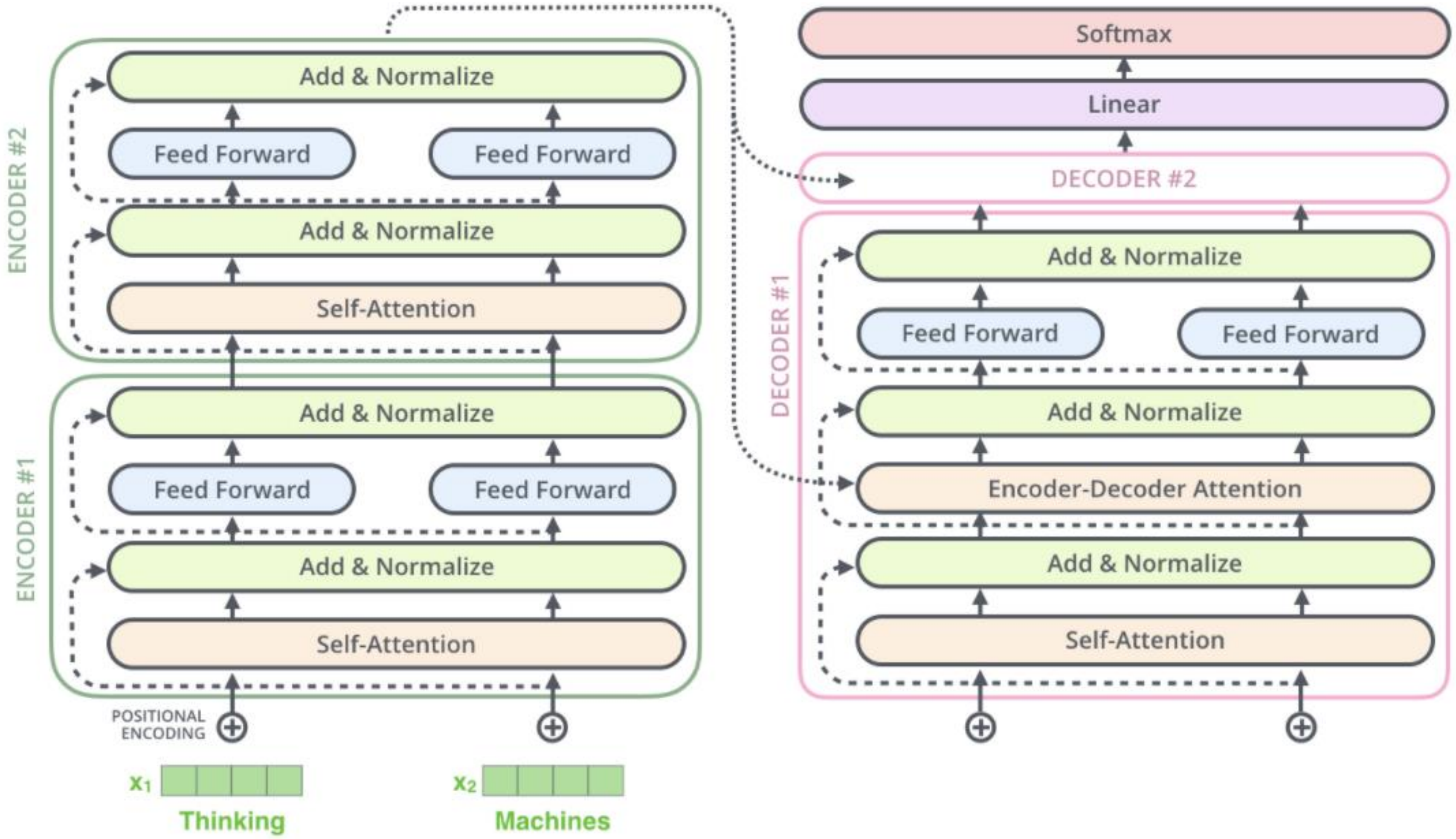
- Next let's look at the end. Similar to a CNN,
  - 1. Linear layer
  - 2. Softmax

Get probabilities of words



Transformers: Putting it All Together

- What does the full architecture look like?



Transformers: The Rest

- Next time: we'll talk about
  - How to **use** it (i.e., outputs)
  - How to **train** it
  - How to **rip** it apart and build other models with it.