CS 639: Foundation Models
Transformers & Attention I

Fred Sala

University of Wisconsin-Madison

Feb. 12, 2026

Announcements

- **Homework 1: out!**
  - Due 2 weeks from release
- **Resources**
  - https://jalammar.github.io/illustrated-transformer/ Very nice resource for following along
- Class roadmap:

Thursday Feb. 12	Transformers and Attention II
Tuesday Feb. 17	Architectures: Encoder-Only
Thursday Feb. 19	Architectures: Others
Tuesday Feb. 24	Attention Variants

LLMs, FMs and Arch

Outline

- **Review From Last Time Basic Attention**
  - Self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Full architecture**
  - Encoder layer, decoder layer, full original Transformer architecture (2017)

Outline

- **Review From Last Time Basic Attention**
  - Self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Full architecture**
  - Encoder layer, decoder layer, full original Transformer architecture (2017)

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/

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

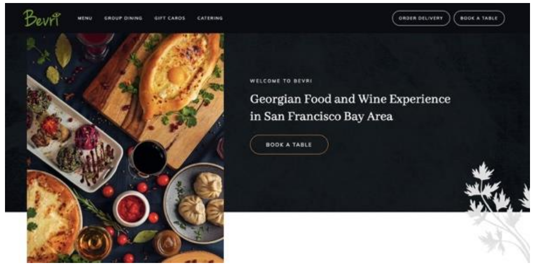
Objects:

Query

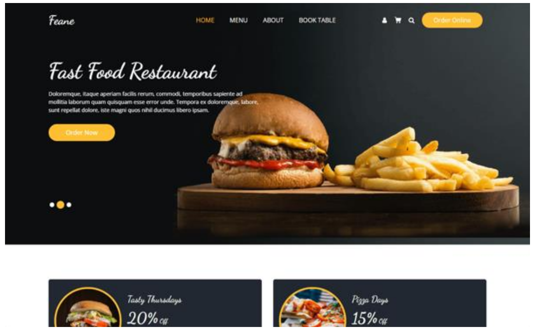
Key

Value

Score 0.3



Score 0.7



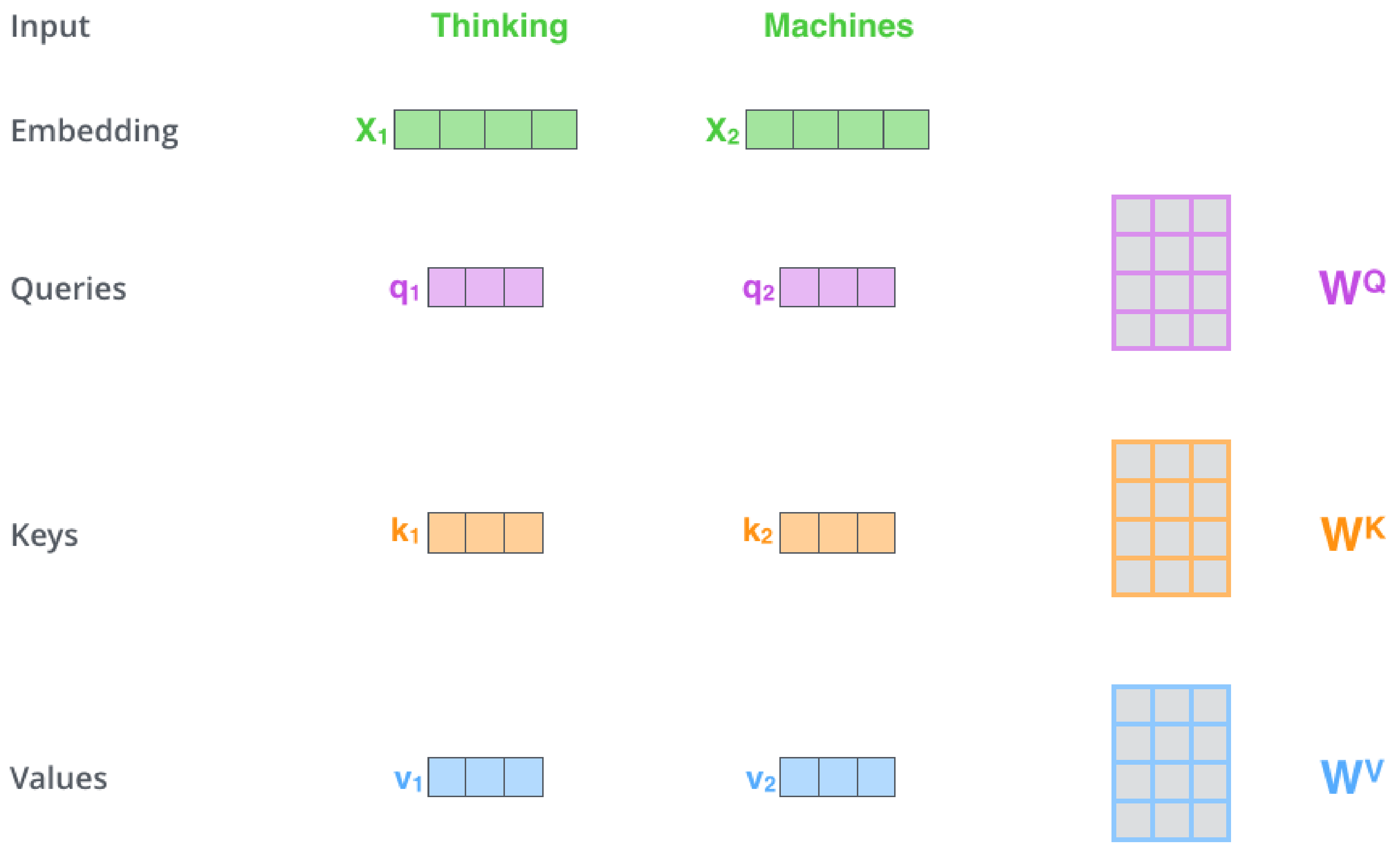
Self-Attention: Query, Key, Value Vectors

- *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

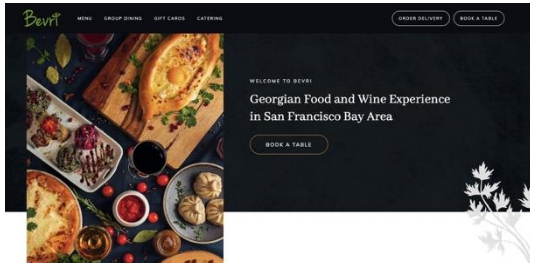
Have **query, key, value** vectors

- Next, get our **score**

q_1 k_2

Query **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**

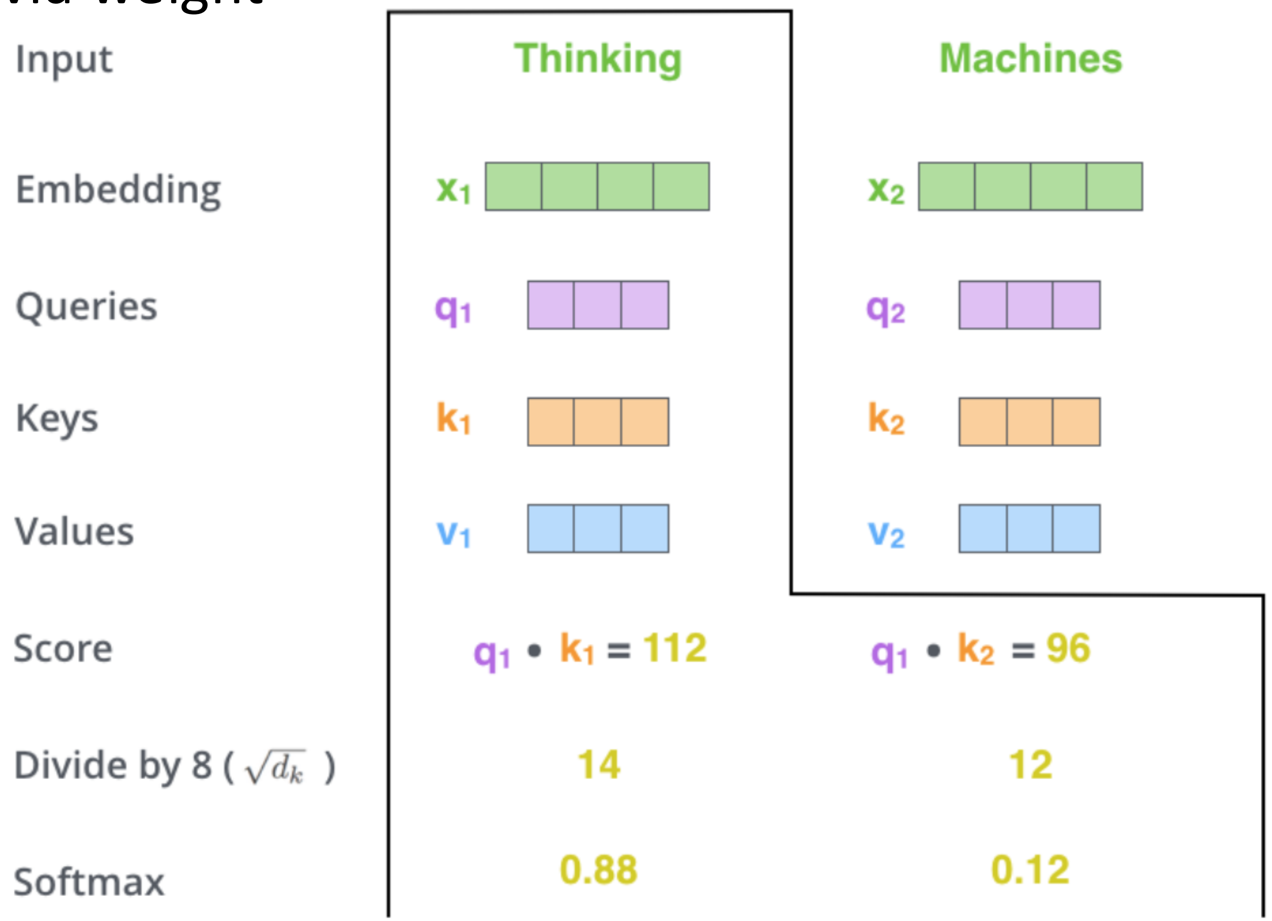
Self-Attention: Scoring and Scaling

- *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

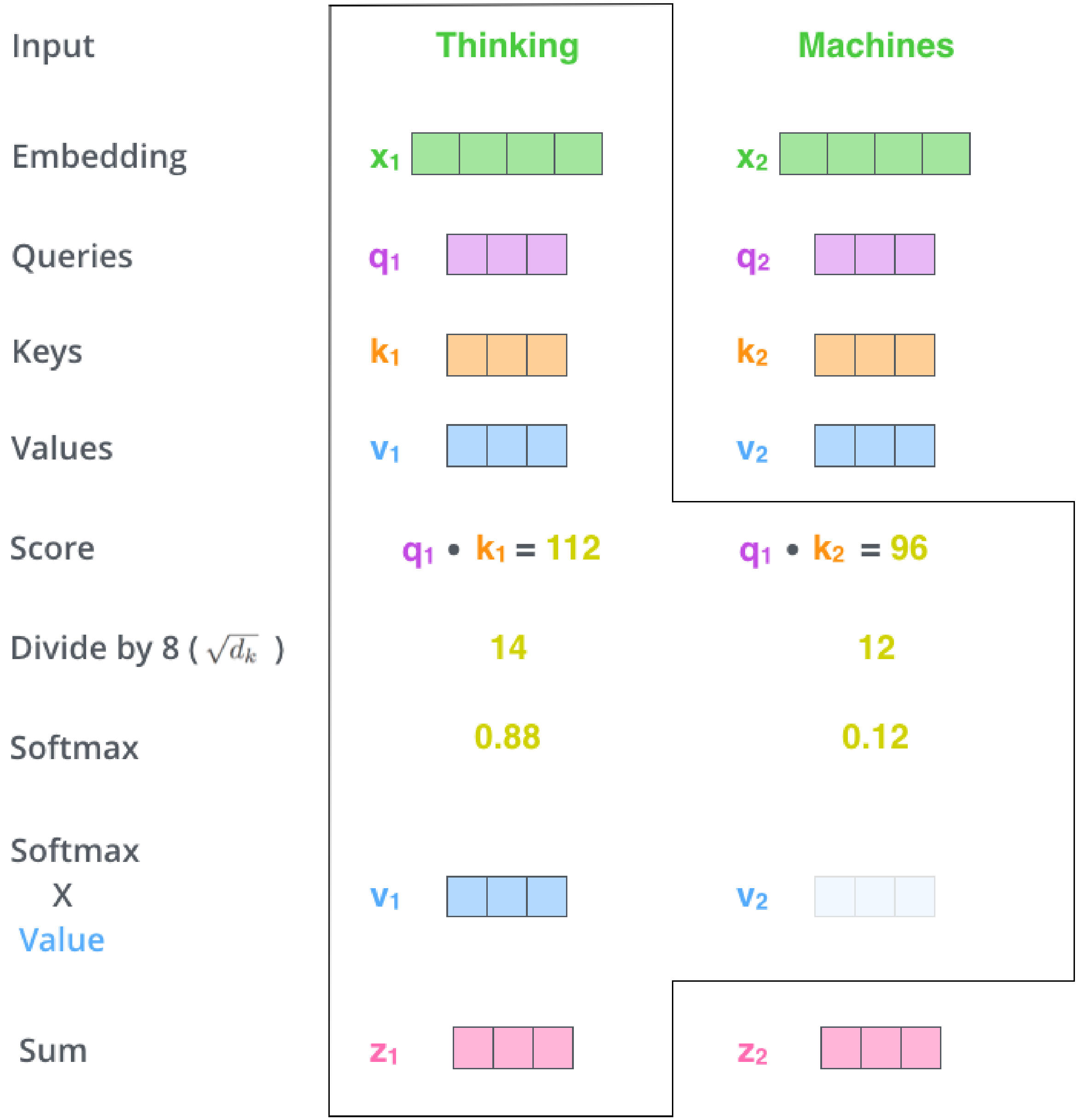


Self-Attention: Putting it Together

- 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

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

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

Note: softmax is applied to the **rows** of this matrix!

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

Break & Questions

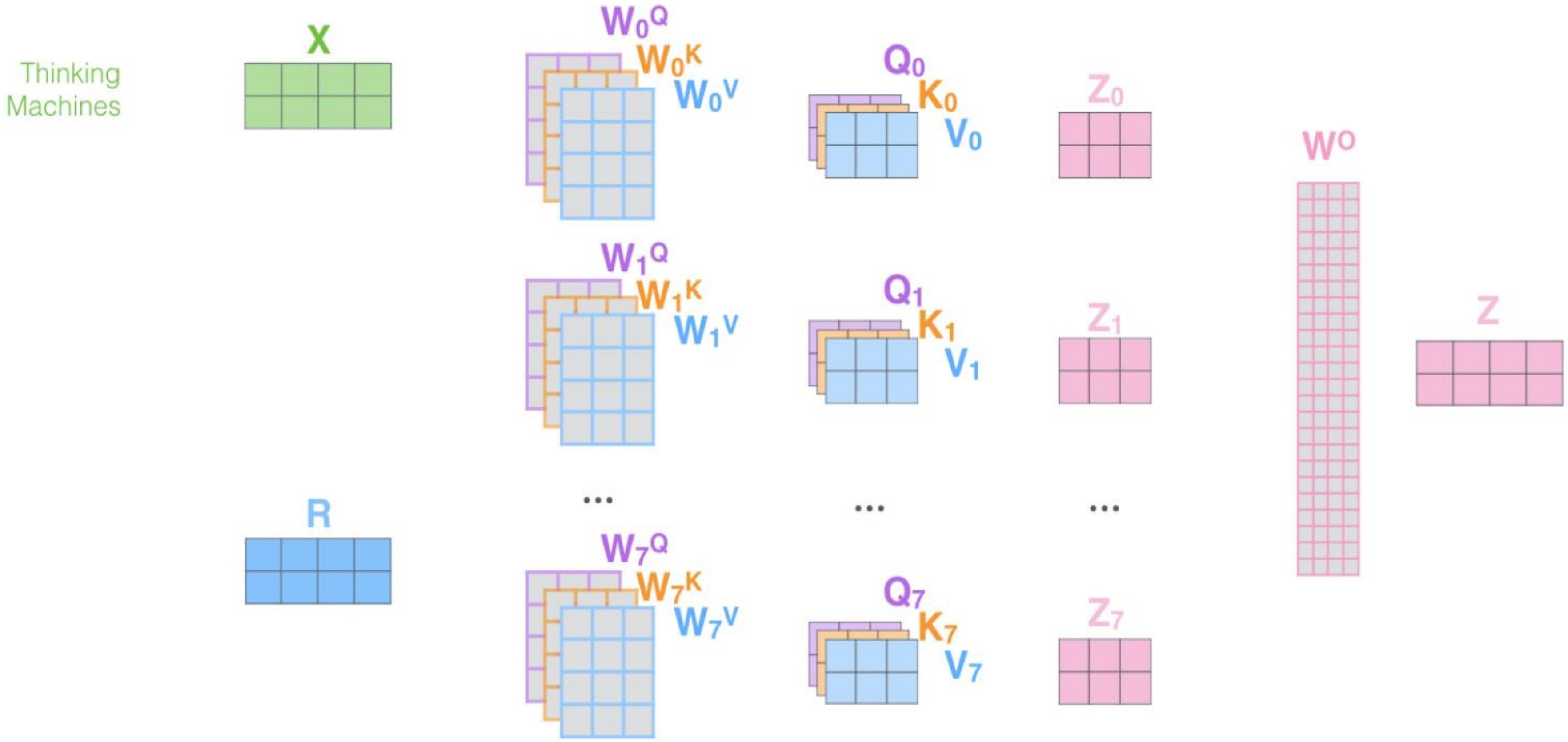
Outline

- **Review From Last Time Basic Attention**
  - Self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Full architecture**
  - Encoder layer, decoder layer, full original Transformer architecture (2017)

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

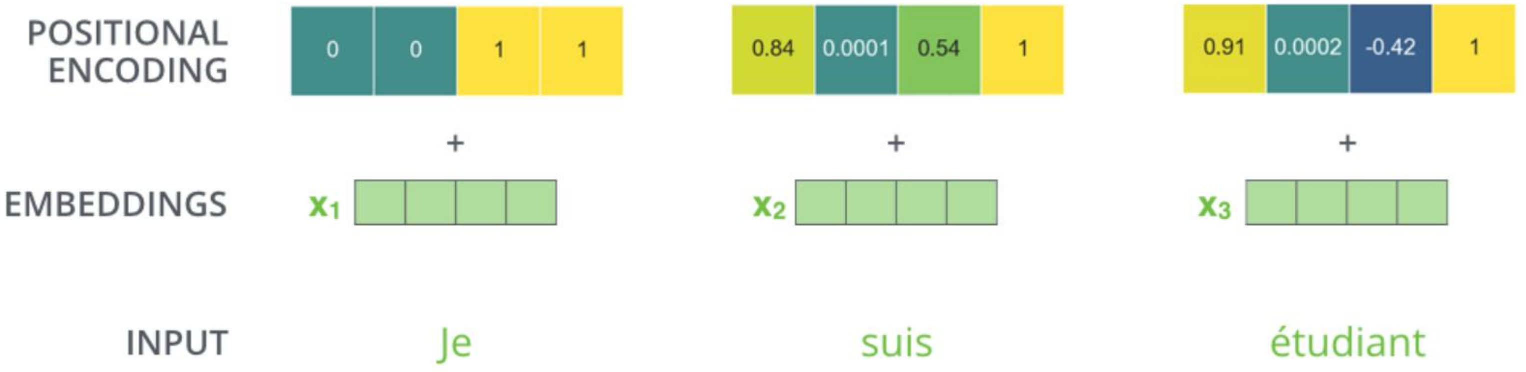
Almost have a full layer designed.

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

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

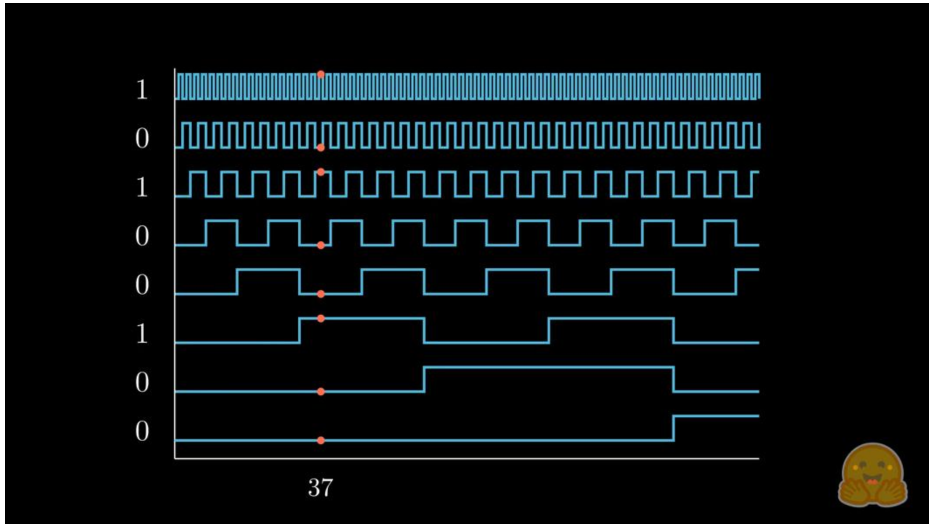
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Location index



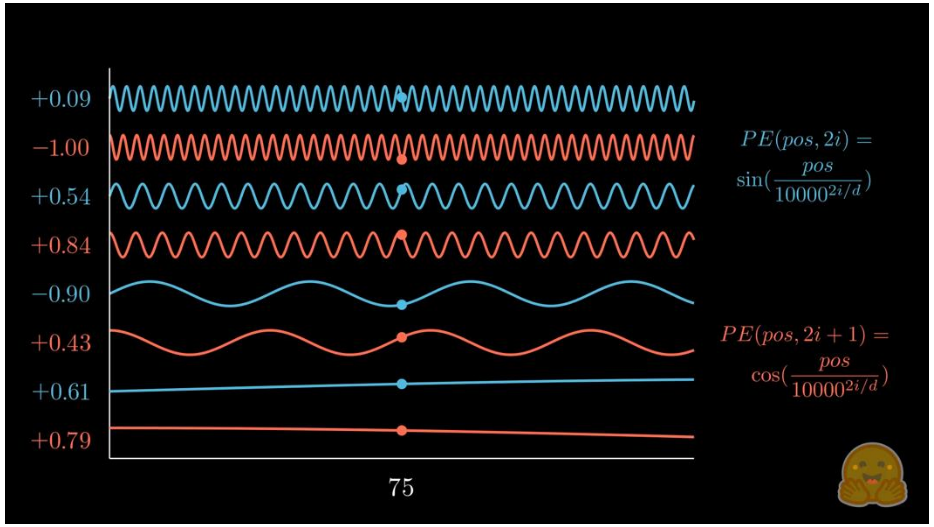
Self-Attention: Positional Encodings

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



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

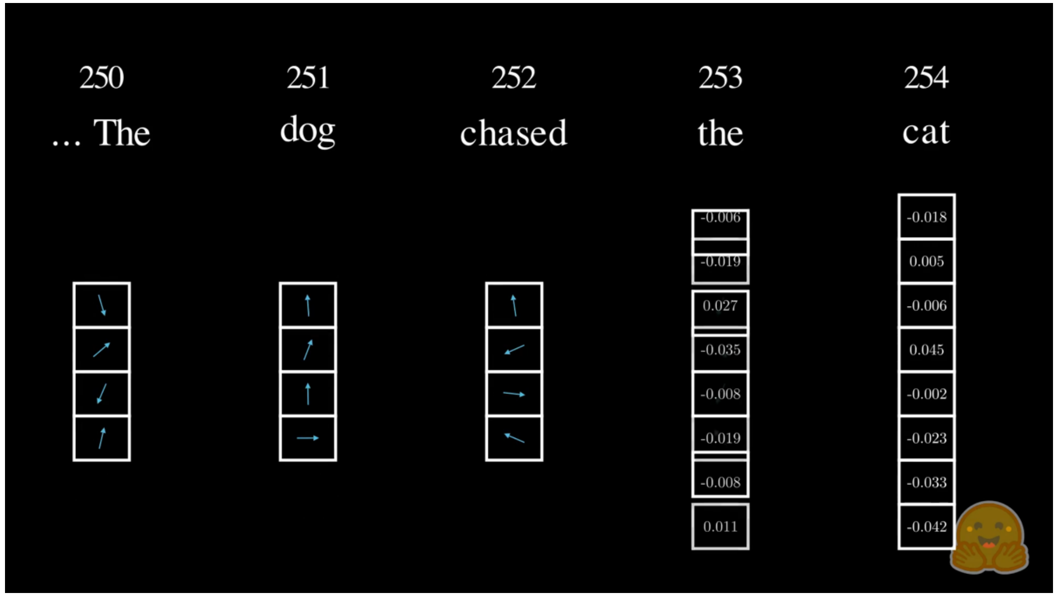
Self-Attention: Modern Positional Encodings

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

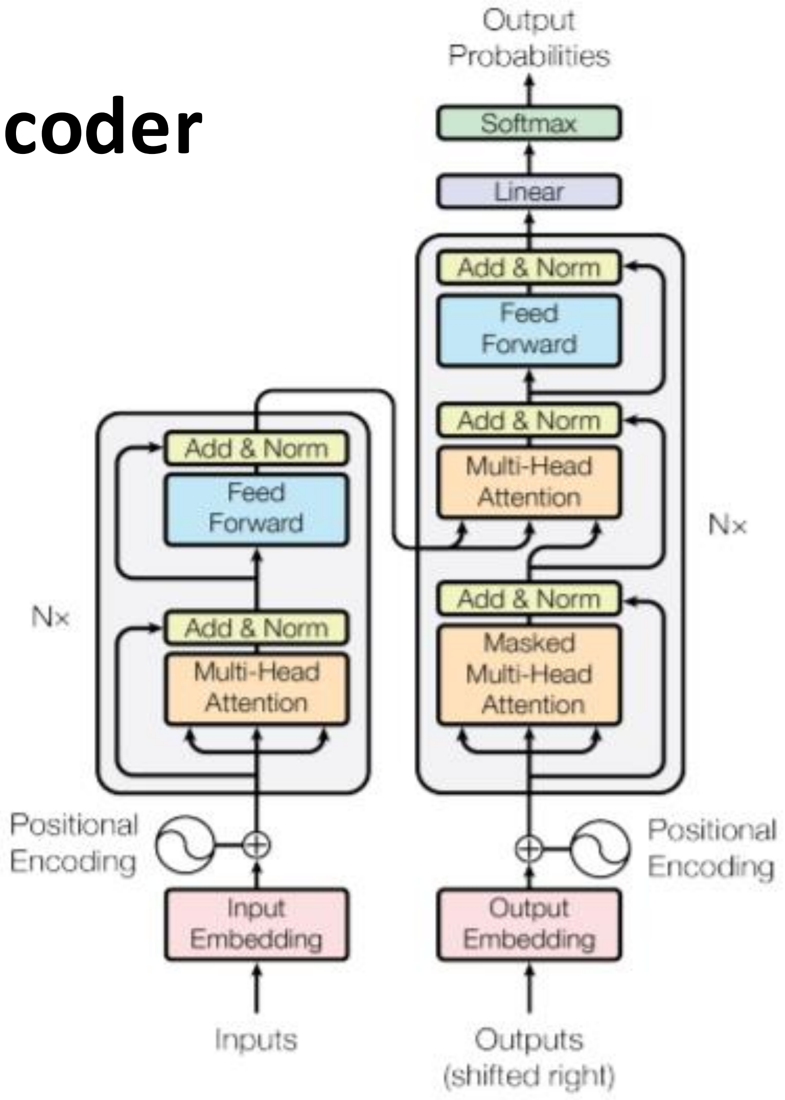
Outline

- **Review From Last Time Basic Attention**
  - Self-attention, basic attention layer, QKV setup and intuition
- **Additional Elements**
  - Multi-head attention, positional encodings
- **Full architecture**
  - Encoder layer, decoder layer, full original Transformer architecture (2017)

Transformers: Model Architecture

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

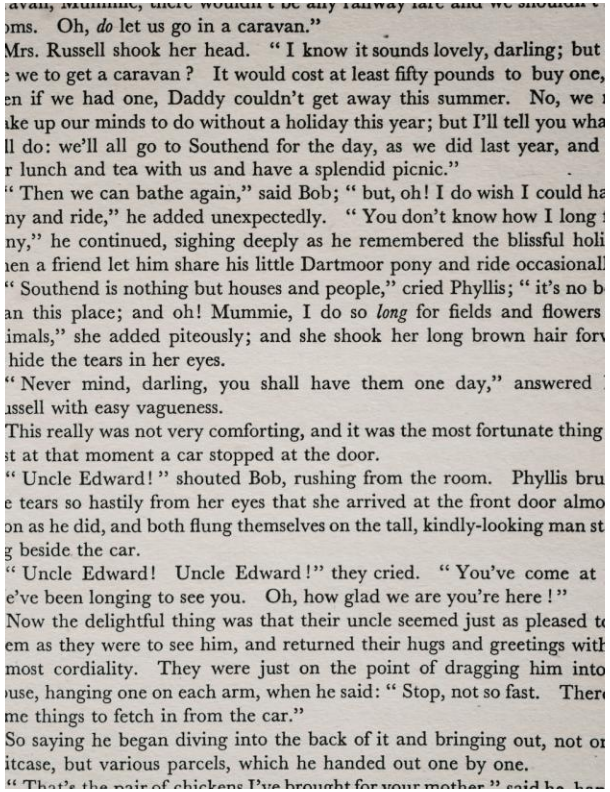
- Architecture
  - You may have seen this picture
  - Centered on self-attention blocks

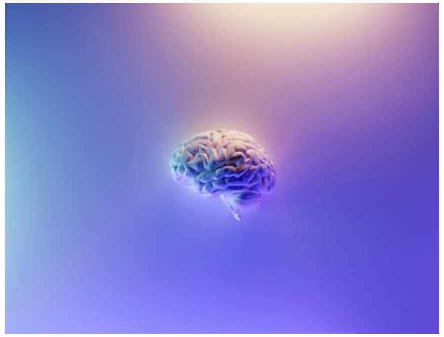


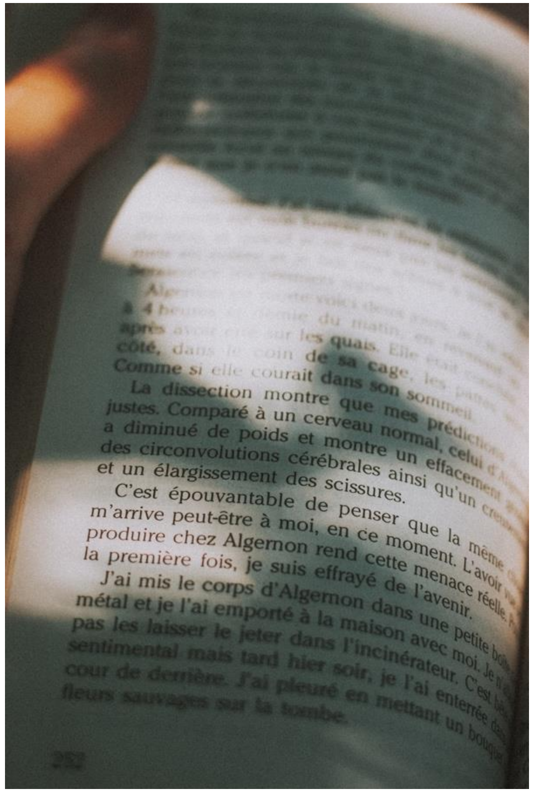
Vaswani et al. ‘17

Interlude: Encoder-Decoder Models

- Translation tasks: natural encoder-decoder architecture
- Intuition:

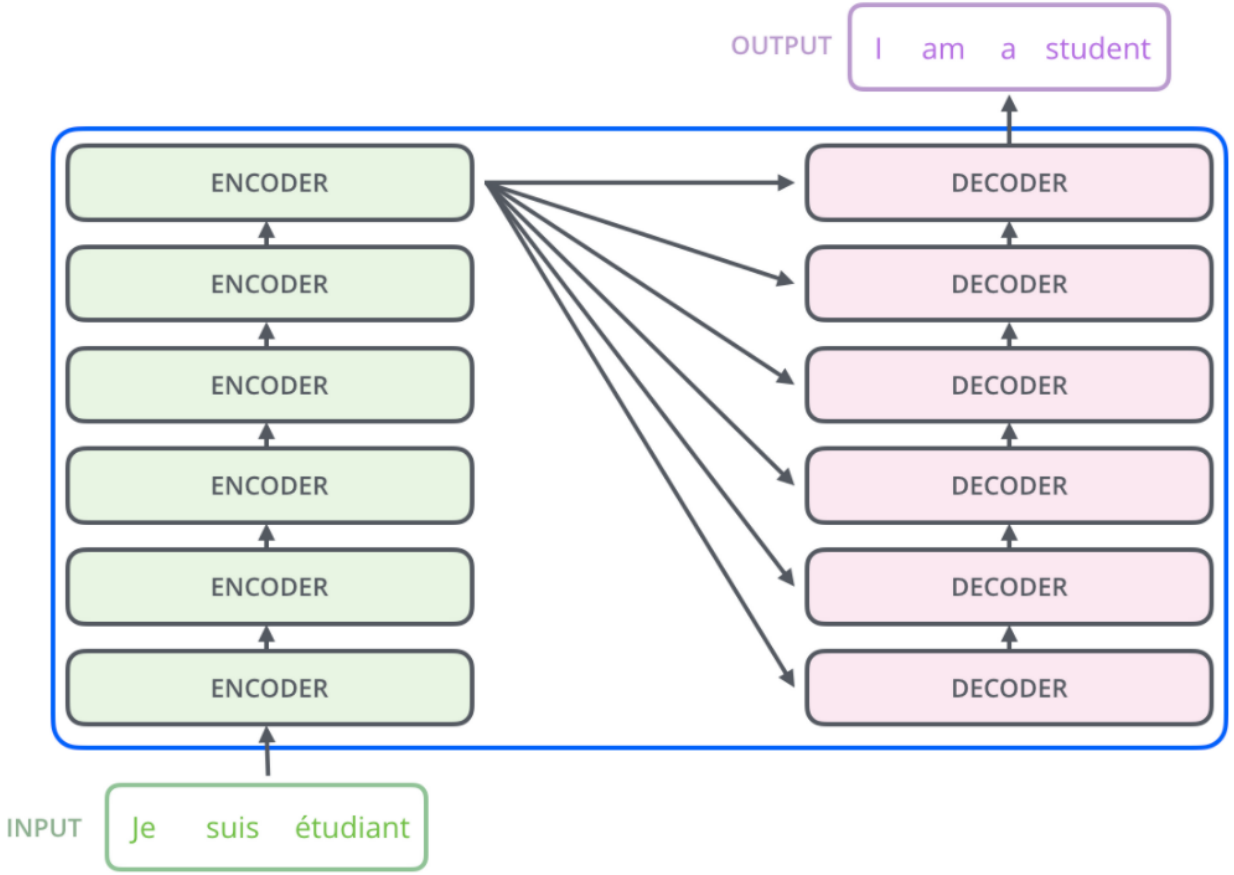

oms. Oh, *do* let us go in a caravan."
Mrs. Russell shook her head. "I know it sounds lovely, darling; but
we to get a caravan? It would cost at least fifty pounds to buy one,
en if we had one, Daddy couldn't get away this summer. No, we
ake up our minds to do without a holiday this year; but I'll tell you wha
ll do: we'll all go to Southend for the day, as we did last year, and
r lunch and tea with us and have a splendid picnic."
"Then we can bathe again," said Bob; "but, oh! I do wish I could ha
ny and ride," he added unexpectedly. "You don't know how I long
ny," he continued, sighing deeply as he remembered the blissful holi
en a friend let him share his little Dartmoor pony and ride occasionall
"Southend is nothing but houses and people," cried Phyllis; "it's no b
an this place; and oh! Mummie, I do so *long* for fields and flowers
imals," she added piteously; and she shook her long brown hair forv
hide the tears in her eyes.
"Never mind, darling, you shall have them one day," answered
ssell with easy vagueness.
This really was not very comforting, and it was the most fortunate thing
st at that moment a car stopped at the door.
"Uncle Edward!" shouted Bob, rushing from the room. Phyllis bru
e tears so hastily from her eyes that she arrived at the front door almo
on as he did, and both flung themselves on the tall, kindly-looking man st
g beside the car.
"Uncle Edward! Uncle Edward!" they cried. "You've come at
e've been longing to see you. Oh, how glad we are you're here!"
Now the delightful thing was that their uncle seemed just as pleased t
em as they were to see him, and returned their hugs and greetings with
most cordiality. They were just on the point of dragging him into
use, hanging one on each arm, when he said: "Stop, not so fast. Ther
me things to fetch in from the car."
So saying he began diving into the back of it and bringing out, not o
itcase, but various parcels, which he handed out one by one.





Transformers: Architecture

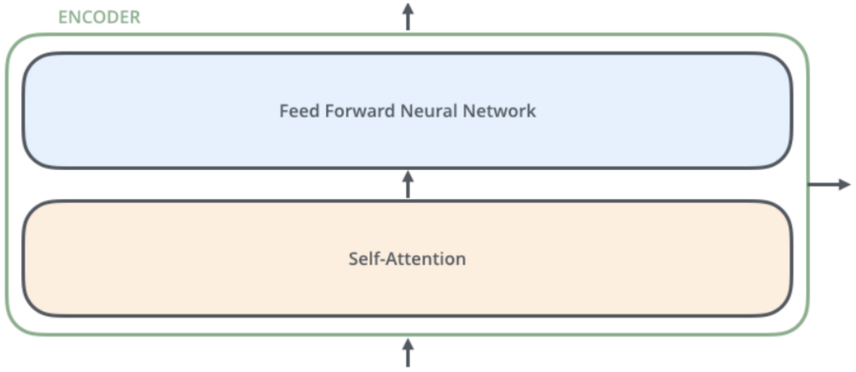
- **Sequence-sequence** model with **stacked** encoders/decoders:
  - For example, for French-English translation:

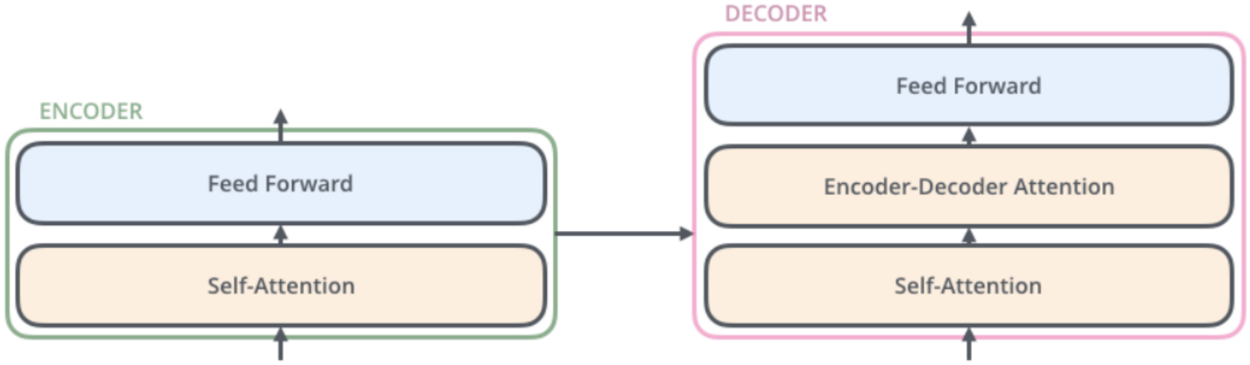


Transformers: Architecture

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

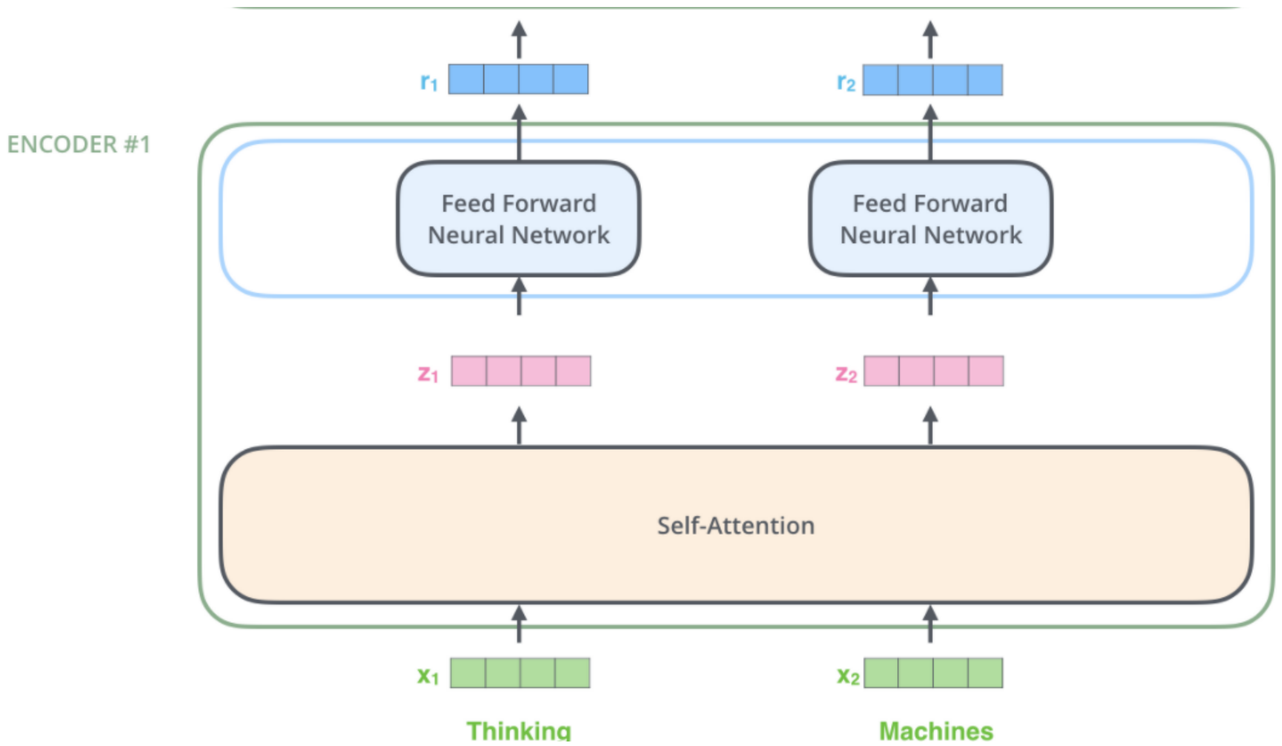
  - Focus encoder first: **pretty simple**! 2 components:
    - Self-attention block
    - Fully-connected layers (i.e., an MLP)





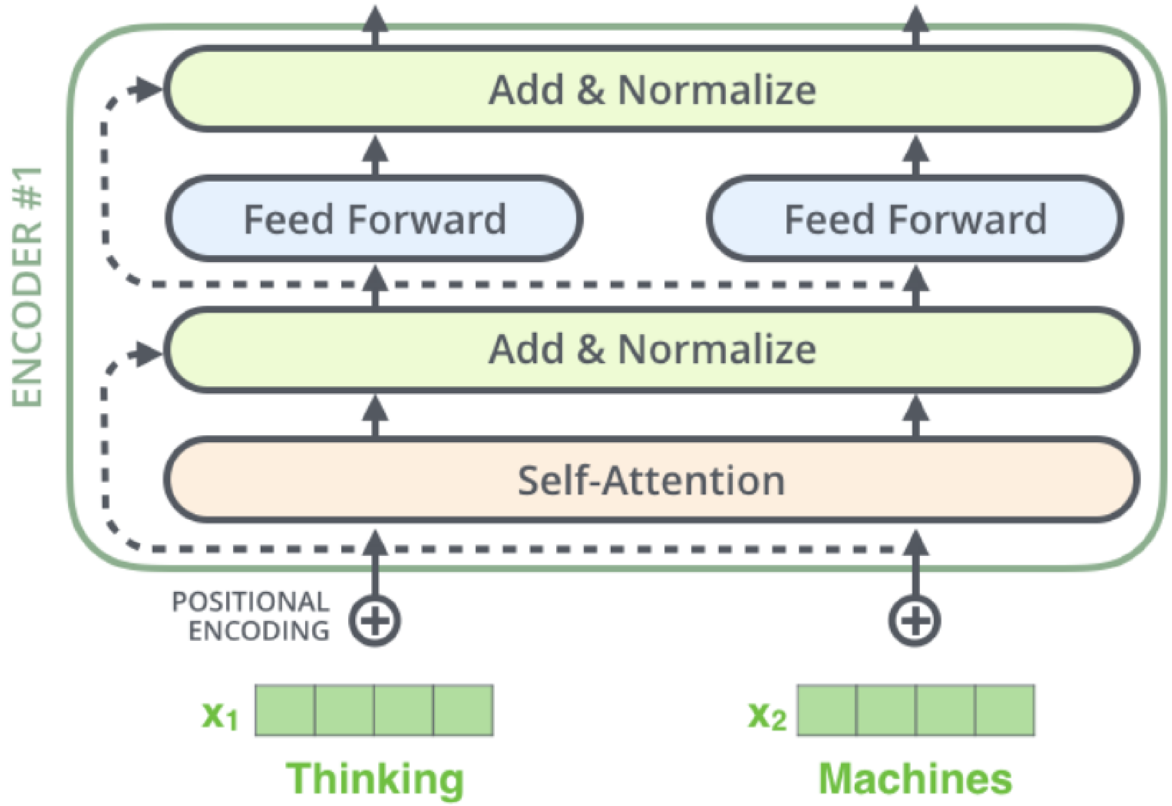
Transformers: Inside an Encoder

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



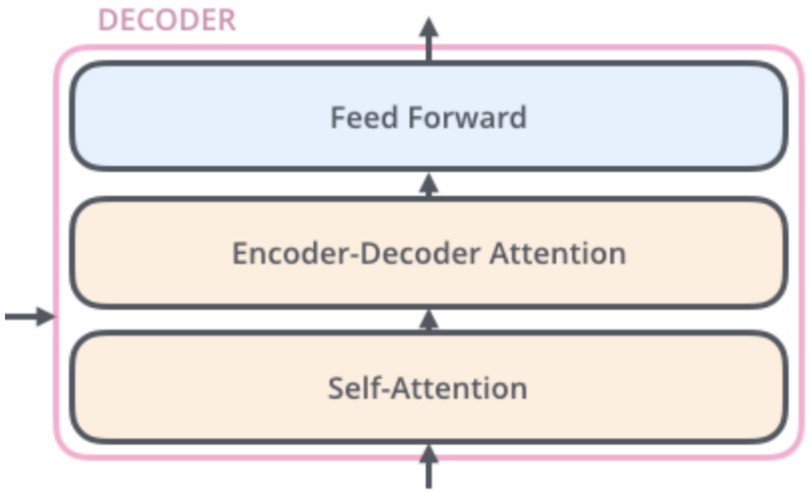
Transformers: More Tricks

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



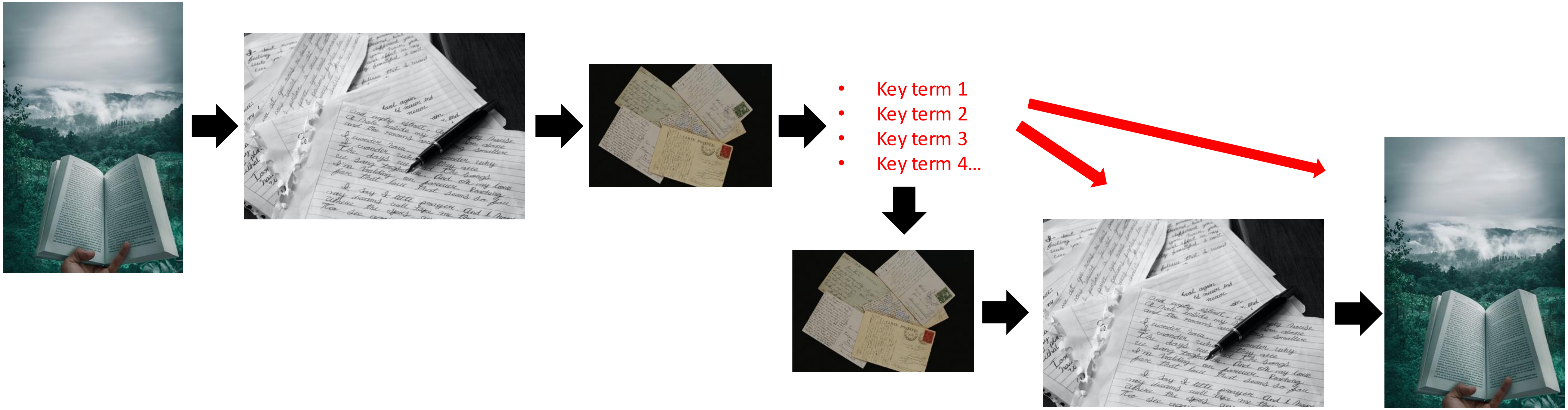
Transformers: Inside a Decoder

- 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**



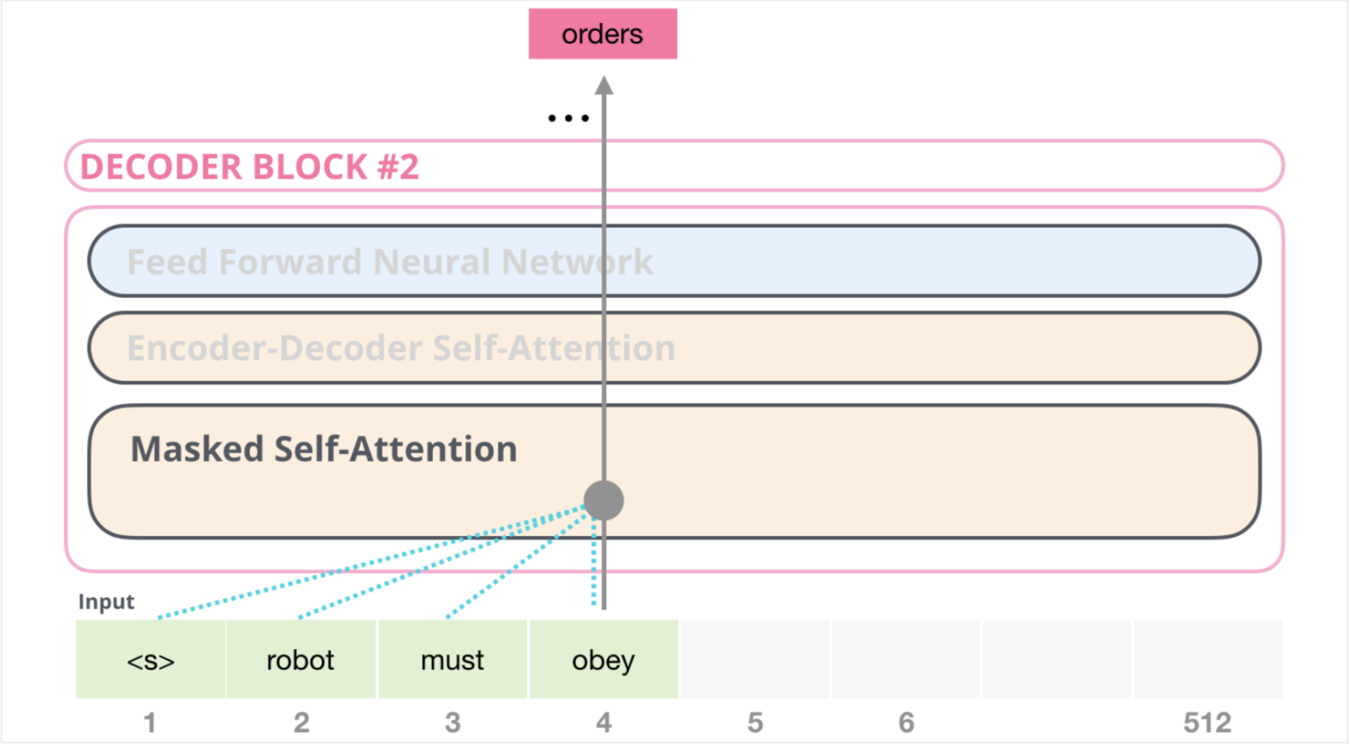
Transformers: Cross-Attention

- Why encoder-decoder attention ?
  - Recall: same as before, but K, V come from encoder
  - Actually more traditional, but… **intuition**:



Transformers: Decoder Masking

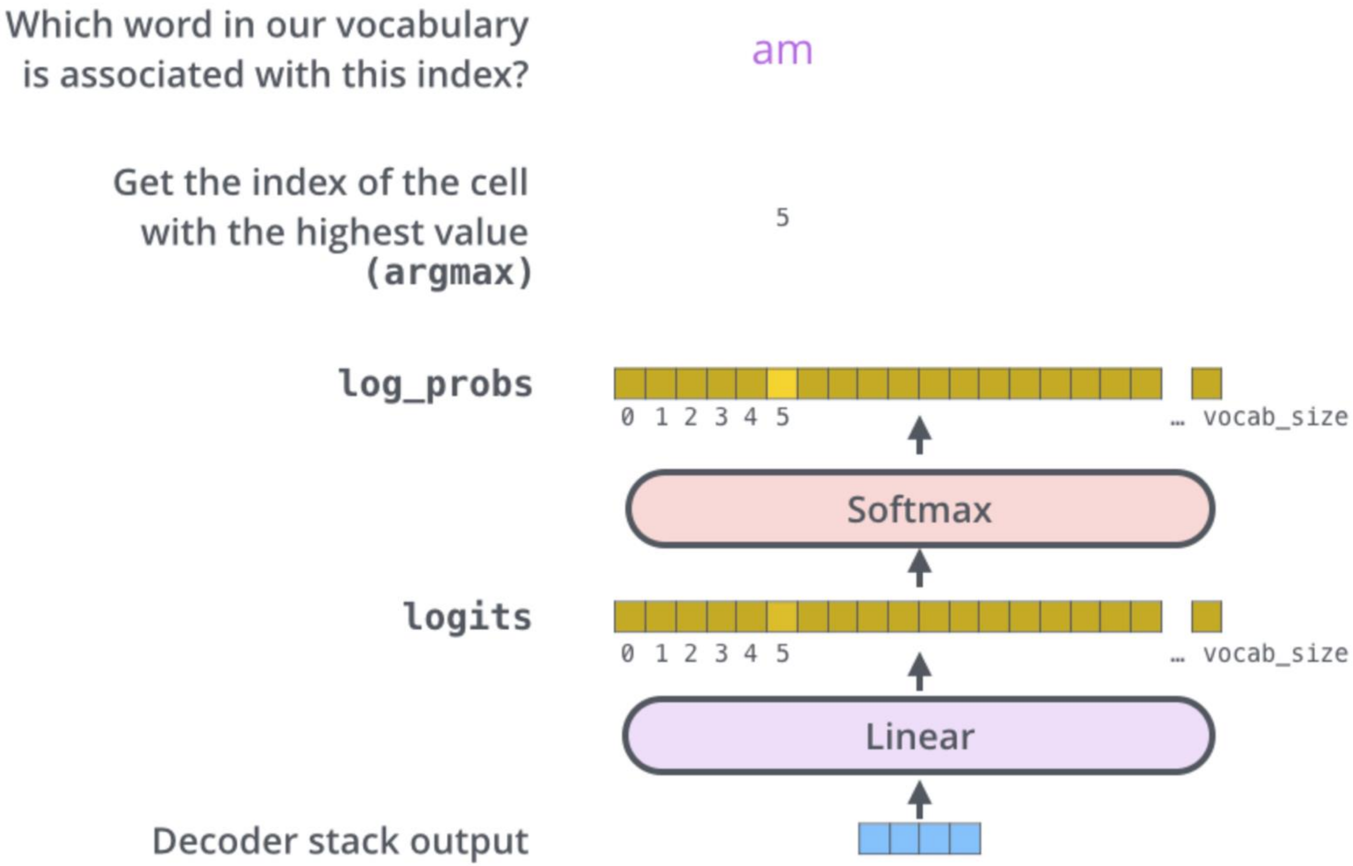
- One more interesting bit!
  - At the decoder level, self-attention changes a bit:
  - Masked instead: block *future* words from being attended to
    - Sorta obvious for inference (test-time), but **important in training**
    - **How to mask?** Add $-\infty$ before softmax



Transformers: Last Layers

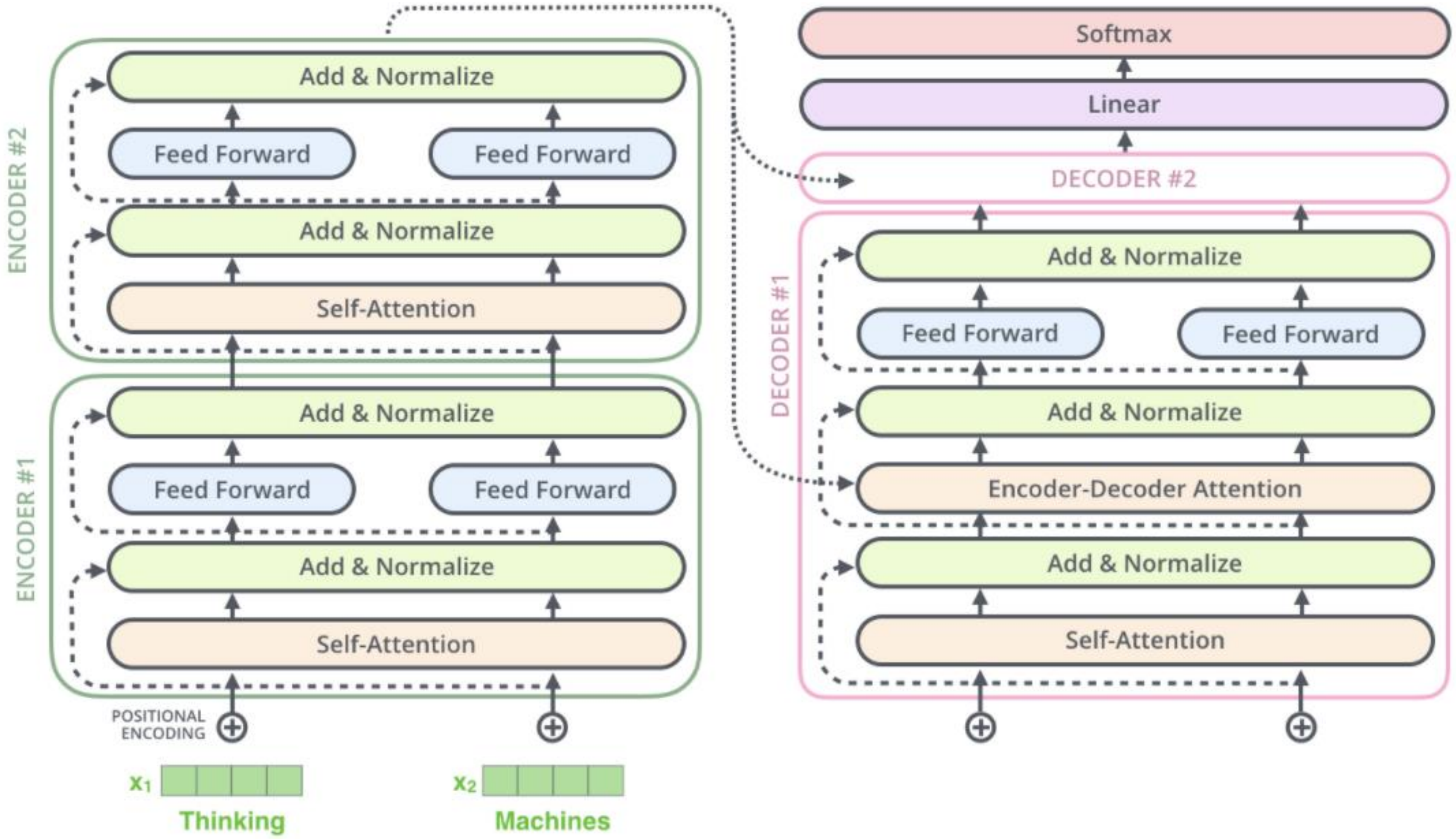
- Next let's look at the end. Similar to a CNN,
  - 1. Linear layer
  - 2. Softmax

Get probabilities of words



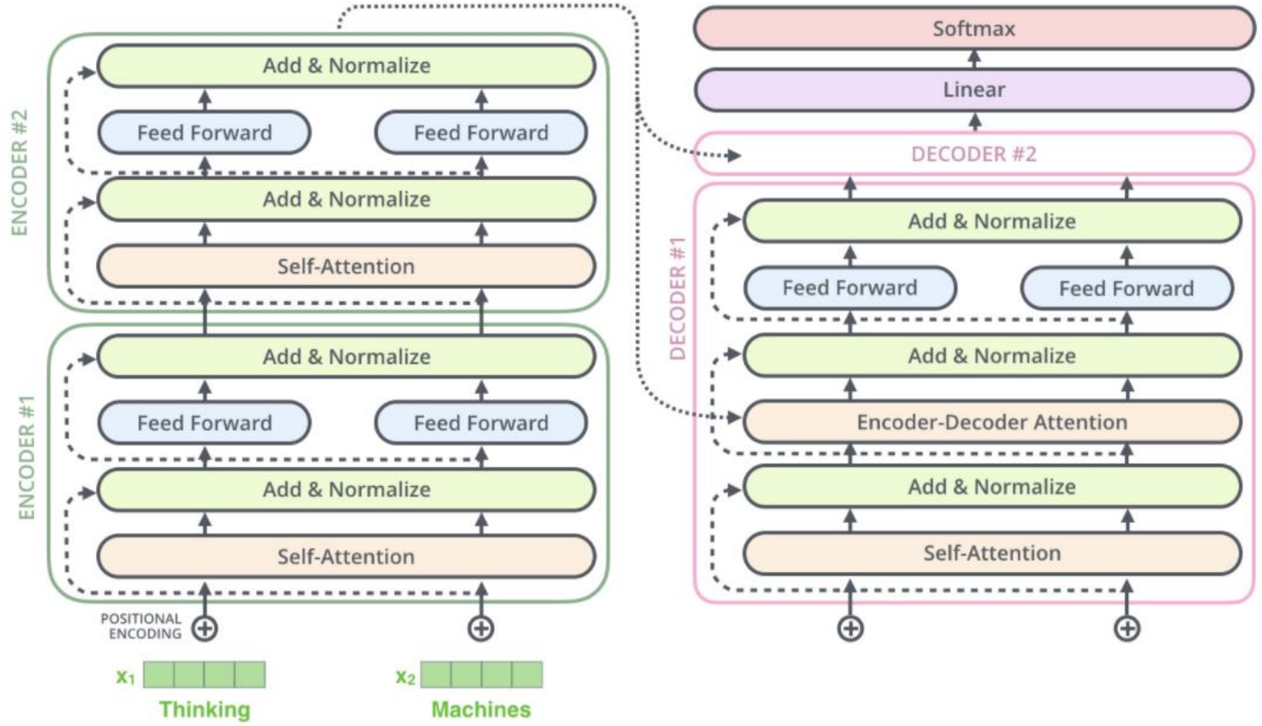
Transformers: Putting it All Together

- What does the full architecture look like?



Transformers: Training

- Data: standard datasets (WMT English-German)
  - Note: **supervised task**. Soon: switch to self-supervised
  - ~5 million pairs of sentences for this dataset
  - Training procedure not special: cross-entropy loss, Adam optimizer



Transformers: The Rest

- Next time: we'll talk about
  - How to **use** it (i.e., outputs)
  - How to **train** it in various other ways (not supervised!)
  - How to **rip** it apart and build other models with it.