Attentions Is All You Need

Transformers

Eng. Mohammad Fouad Shikh Ali

Attentions Is All You Need

Part 1

Tokenization

"Where can I find a pizzeria?"

Word-Level Tokenization

[where, can, i, find, a, pizzeria]

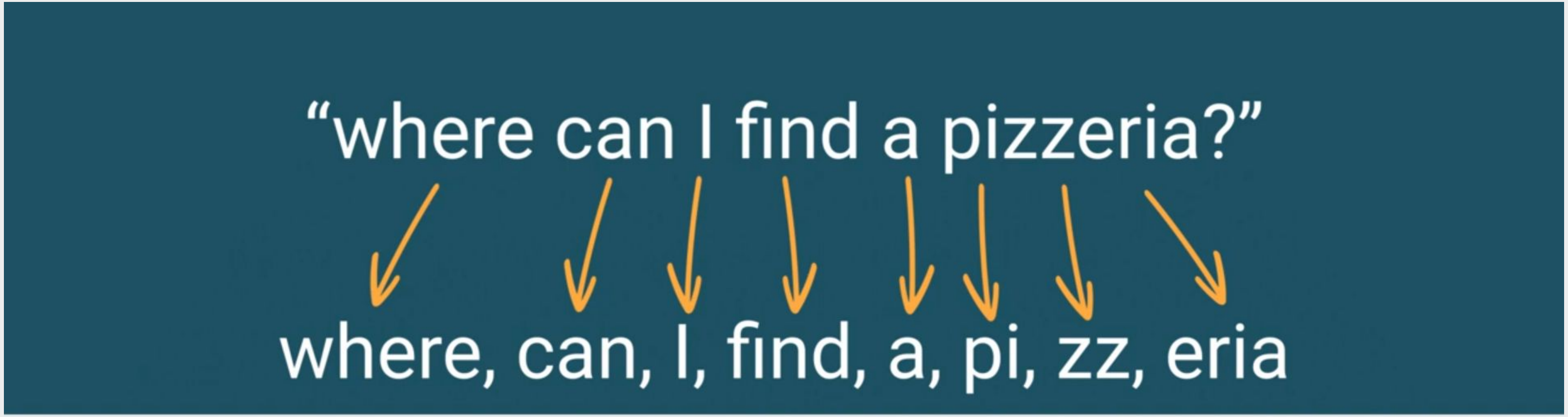
- + Intuitive.
- − Doesn't handle OOV words including new words, slang, play on words, misspellings, etc.
- − Huge vocabulary for large corpora; especially for languages with rich morphology (e.g. Hungarian).
- − Handling punctuation is challenging (e.g. "don't" vs "N.Y.C").

Char-Level Tokenization

[a, c, d, e, f, h, i, n, p, r, w, z]

- + Small memory footprint.
- + Handles OOV words.
- − Needs to go over all characters and learn a particular sequence for a given word.
- − Loss of performance.

Subword Tokenization



Subword tokenization has a better chance of handling OOV words while reducing vocabulary size and maintaining performance.

Byte Pair Encoding (BPE)

“she sells seashells by the seashore”

Corpus

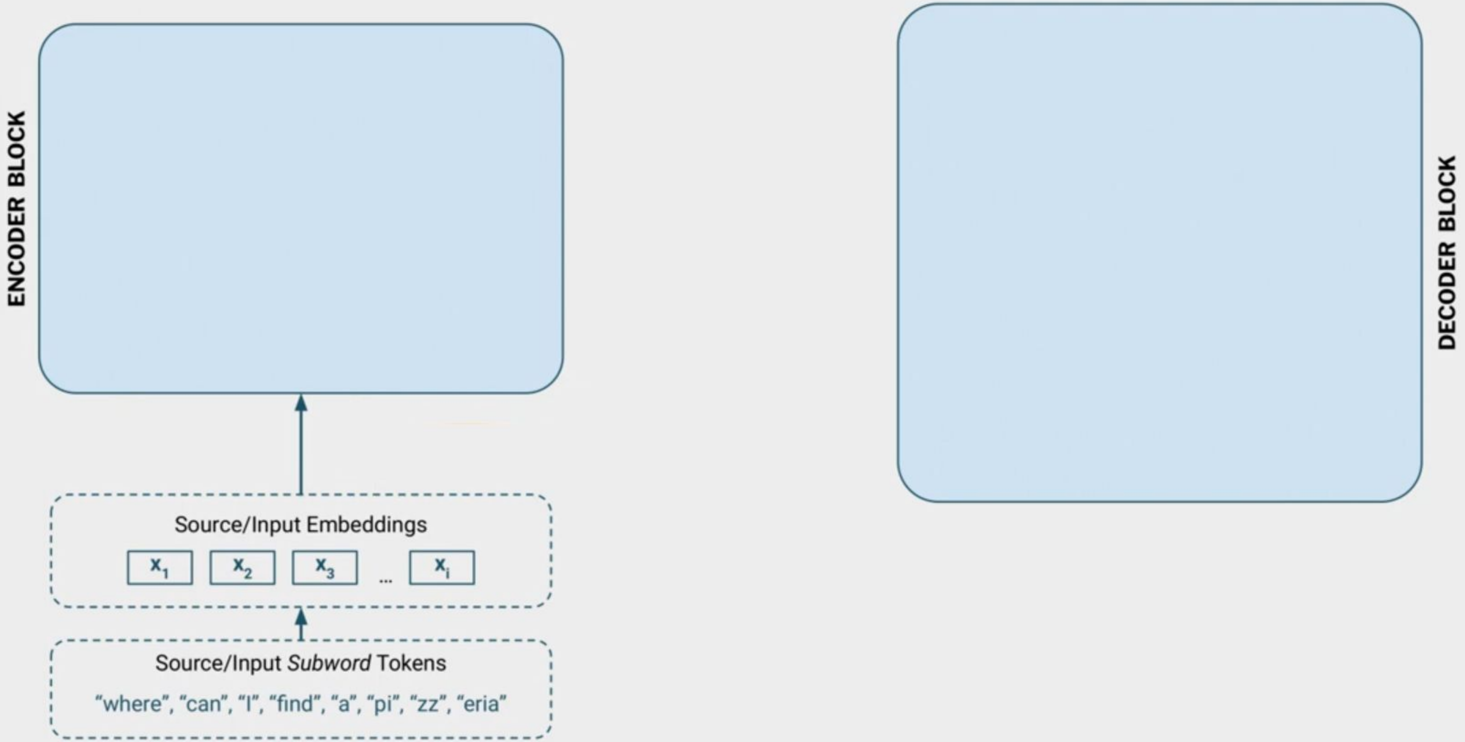
she_
sells_
seashells_
by_
the_
seashore_

Vocabulary

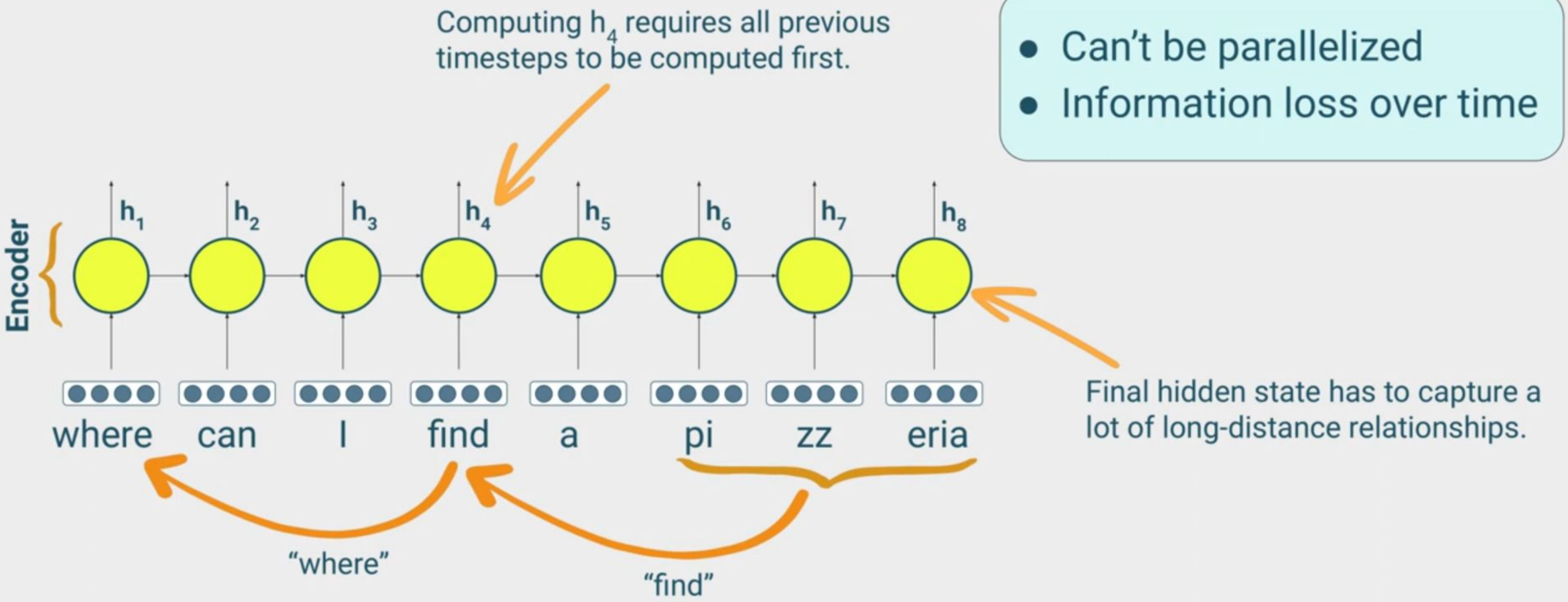
```
_, a, b, e, h, l, o, r, s, t, y
_, a, b, e, h, l, o, r, s, t, y, sh
_, a, b, e, h, l, o, r, s, t, y, sh, he
_, a, b, e, h, l, o, r, s, t, y, sh, he, e_
_, a, b, e, h, l, o, r, s, t, y, sh, he, e_, se
_, a, b, e, h, l, o, r, s, t, y, sh, he, e_, se, she
.
.
.
```

continue until N merges are performed.

Transformers

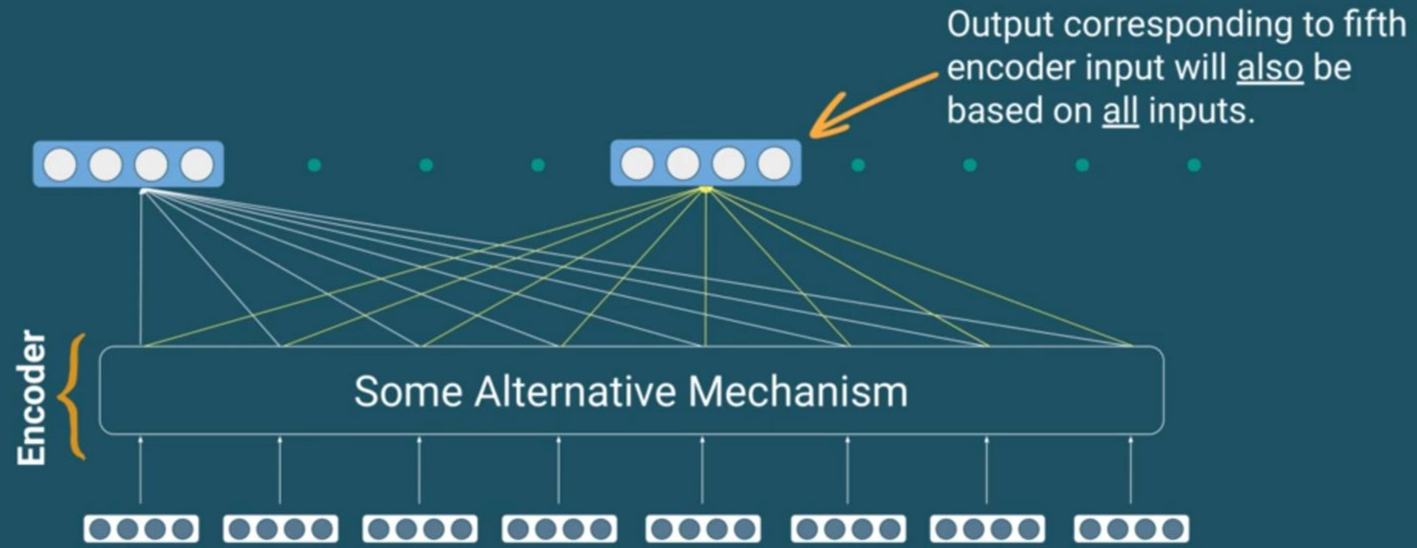


Why Transformers



- Can't be parallelized
- Information loss over time

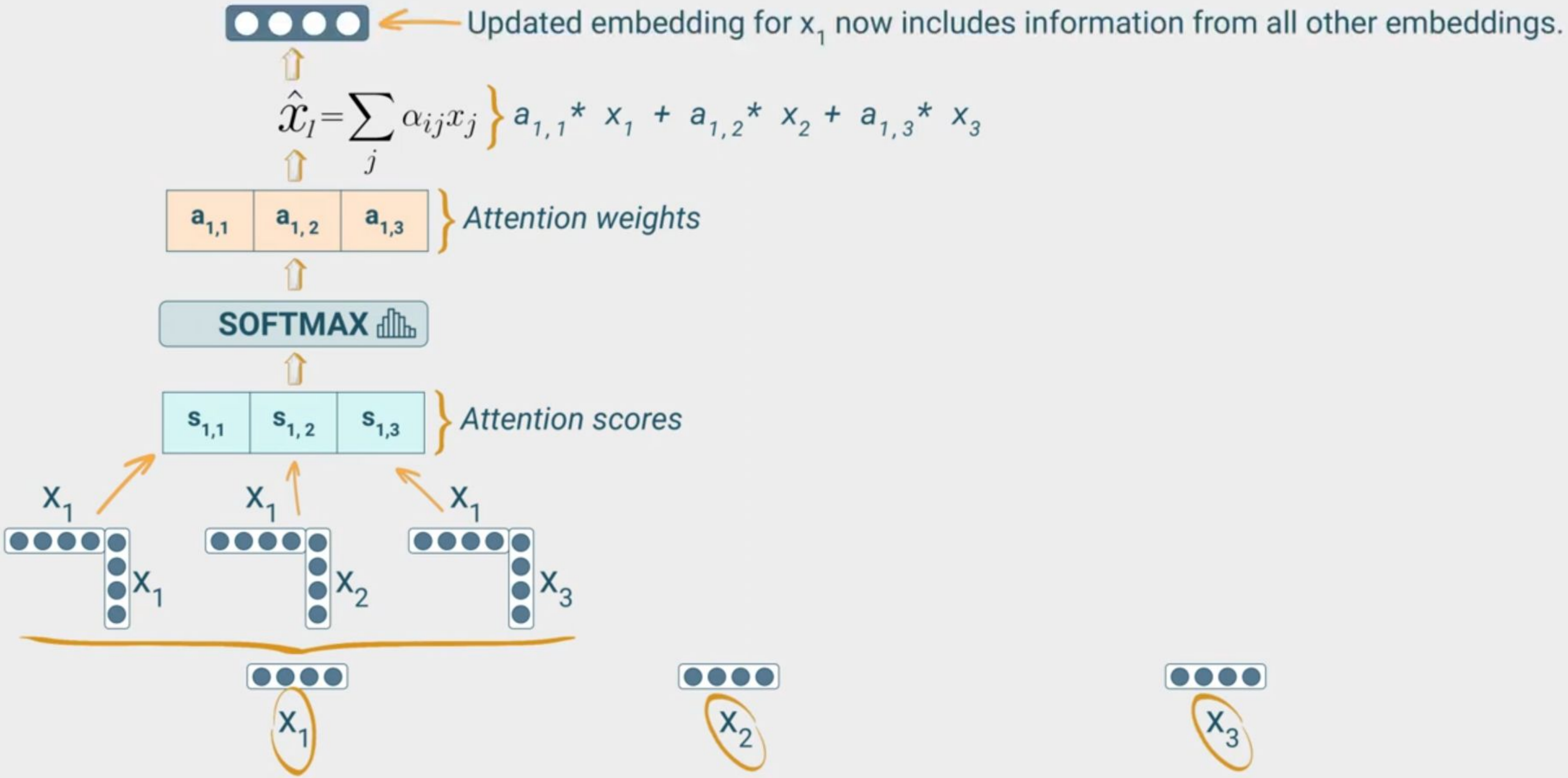
…throw away recurrence, and base EACH encoder OUTPUT on ALL the encoder INPUTS.



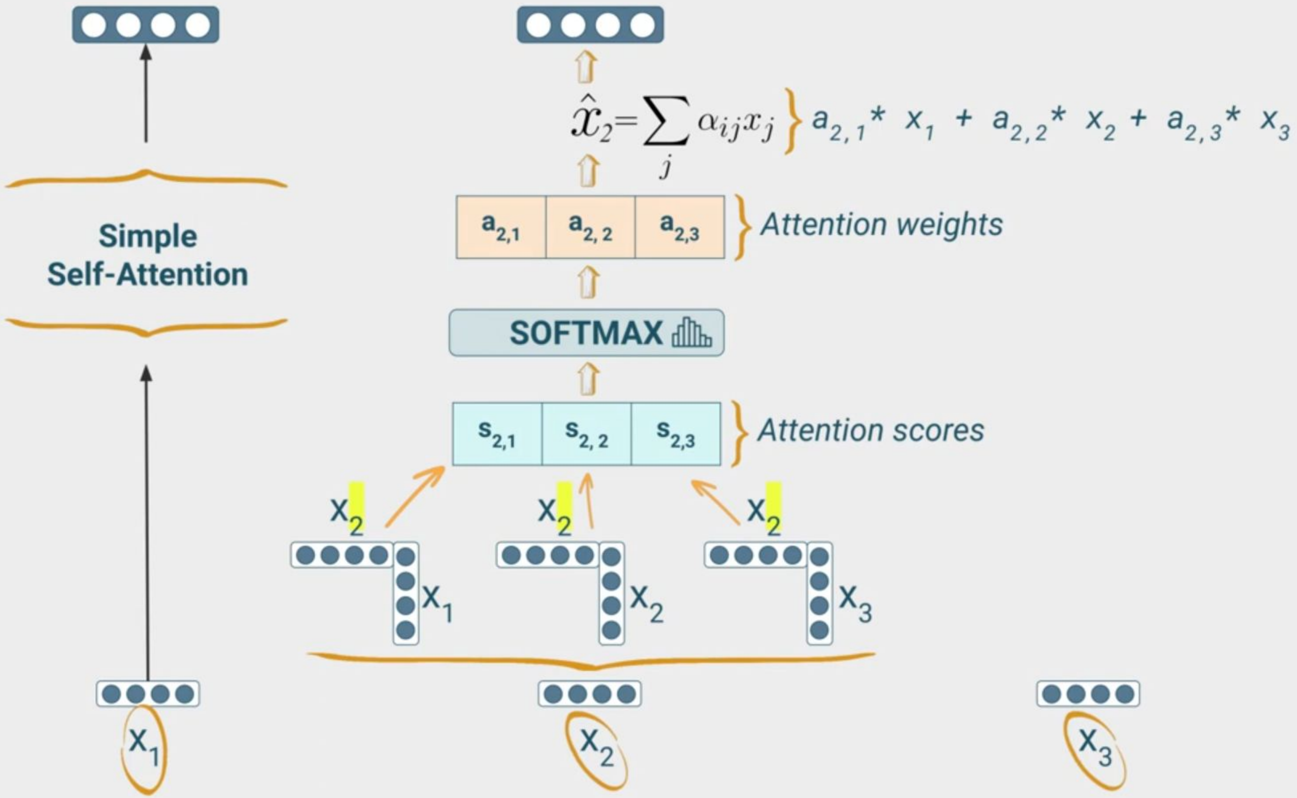
Do this by having the encoder perform attention on itself.

"Self-Attention"

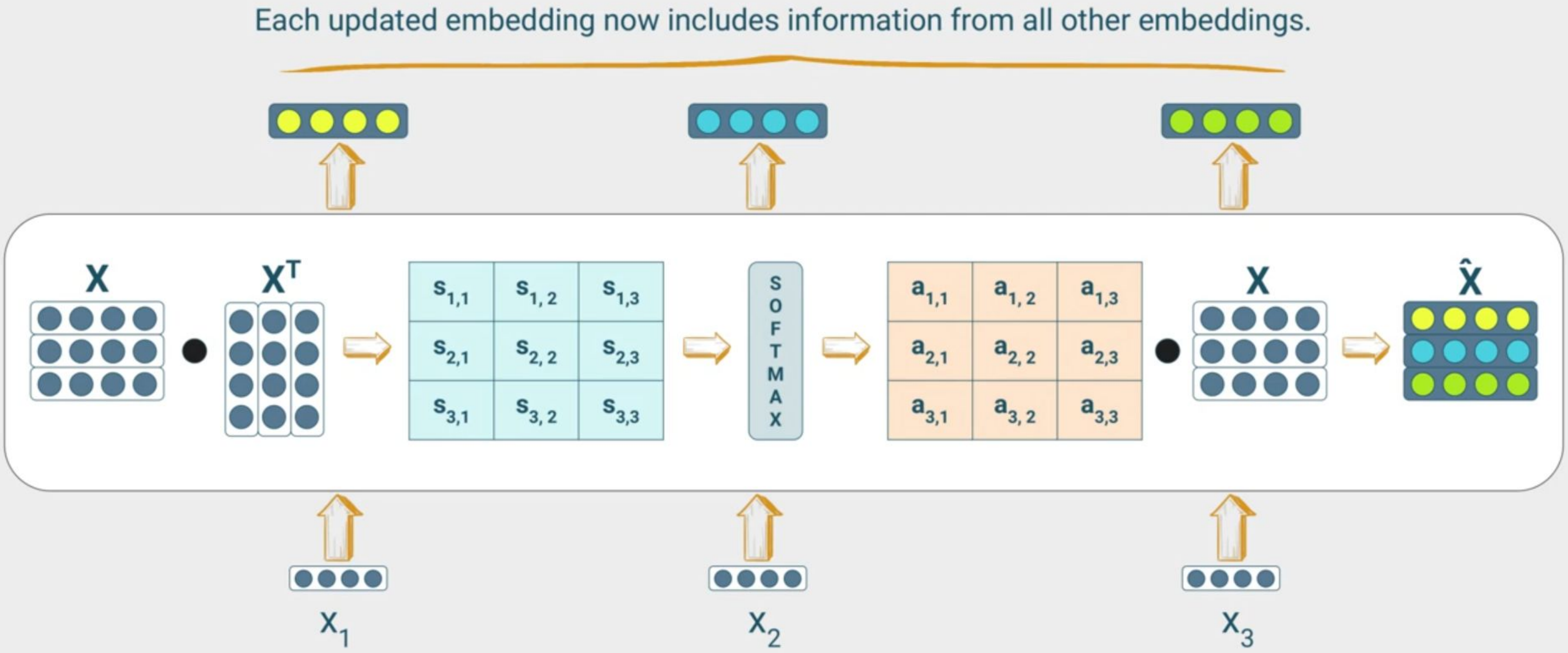
Self-Attention



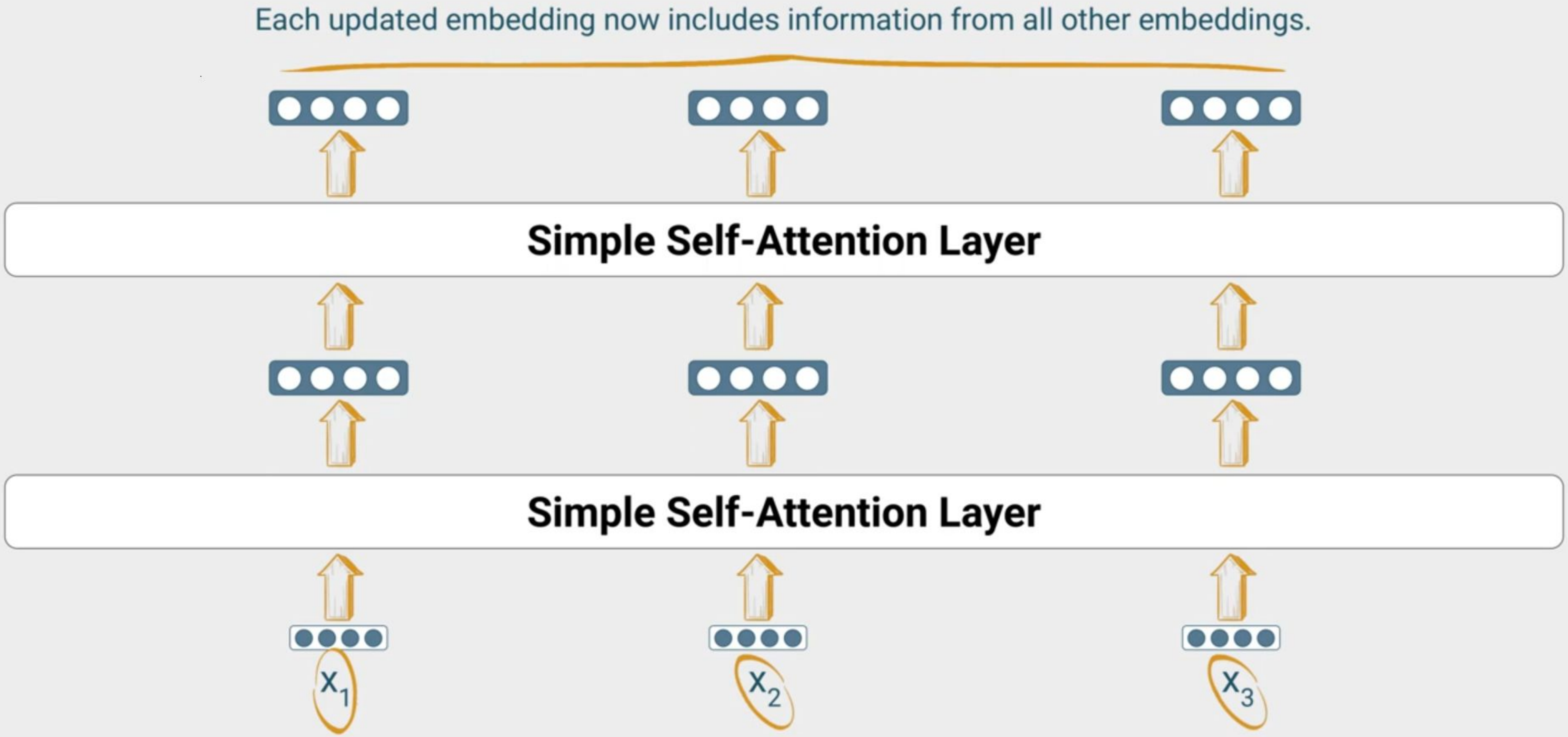
Self-Attention



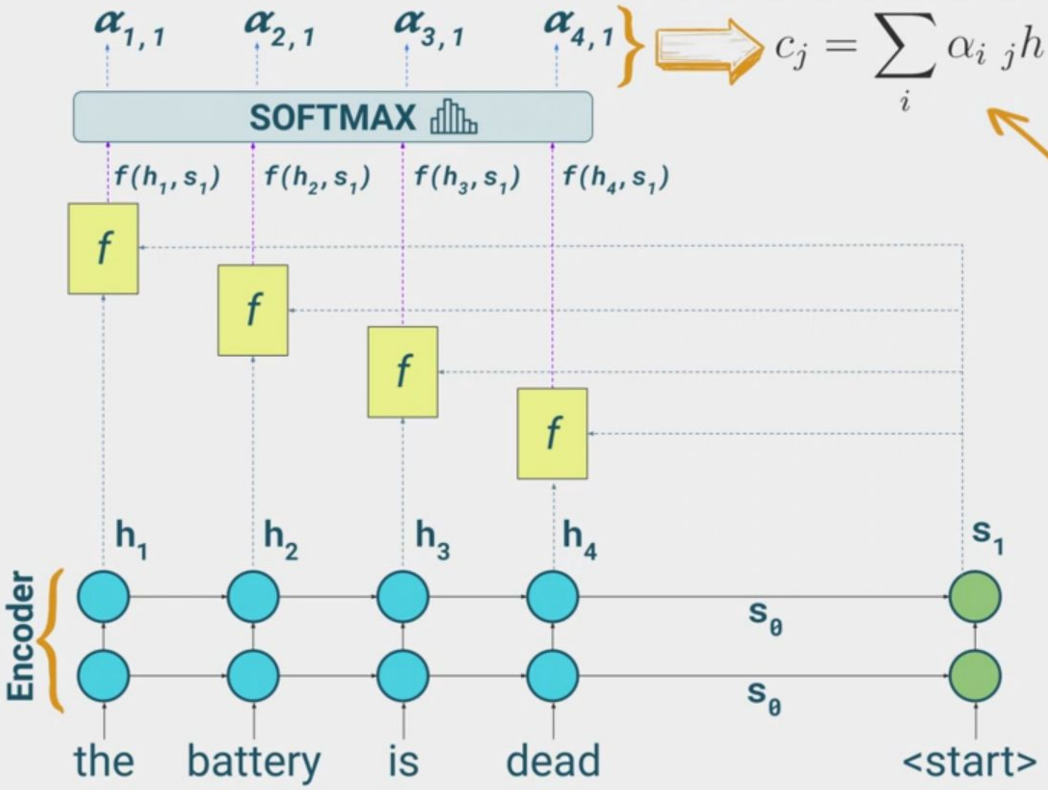
Self-Attention



Self-Attention



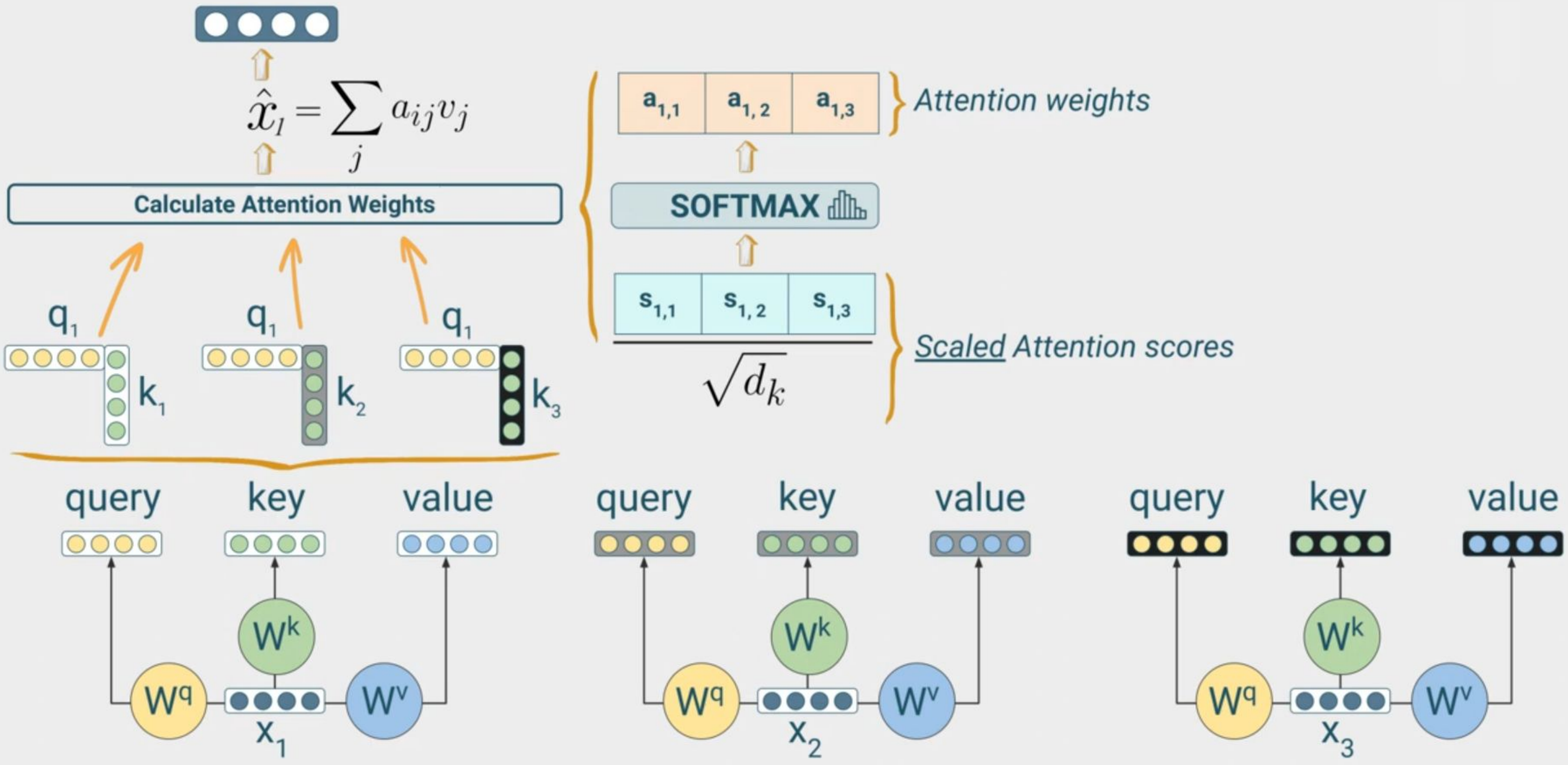
RNNs (Encoder-Decoder)



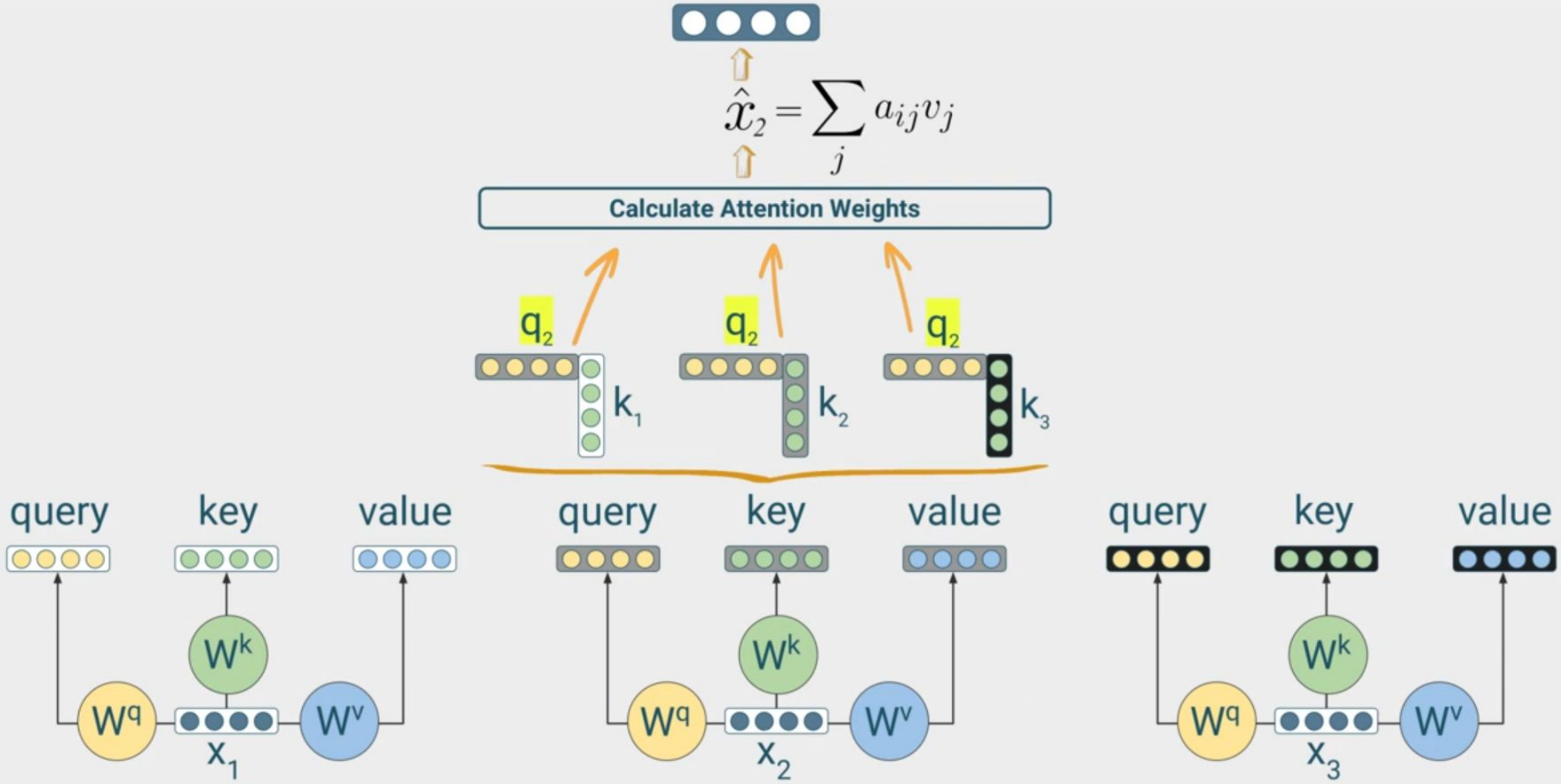
$$c_j = \sum_i \alpha_{i\ j} h_{\ i}$$

Rather than returning a single value, a blend of ALL values is returned (*attention* is more like mixing paint).

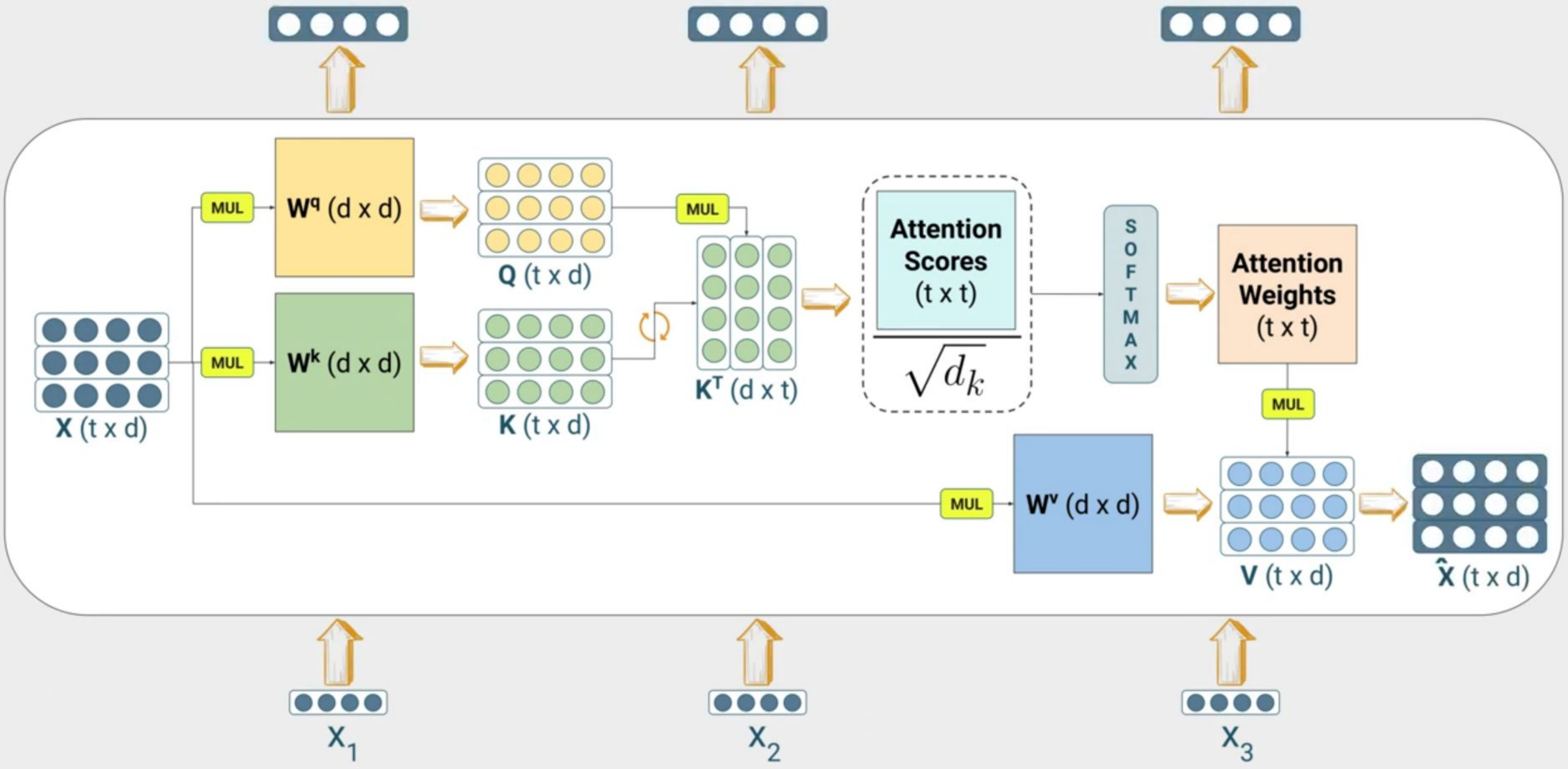
Scale Dot Product



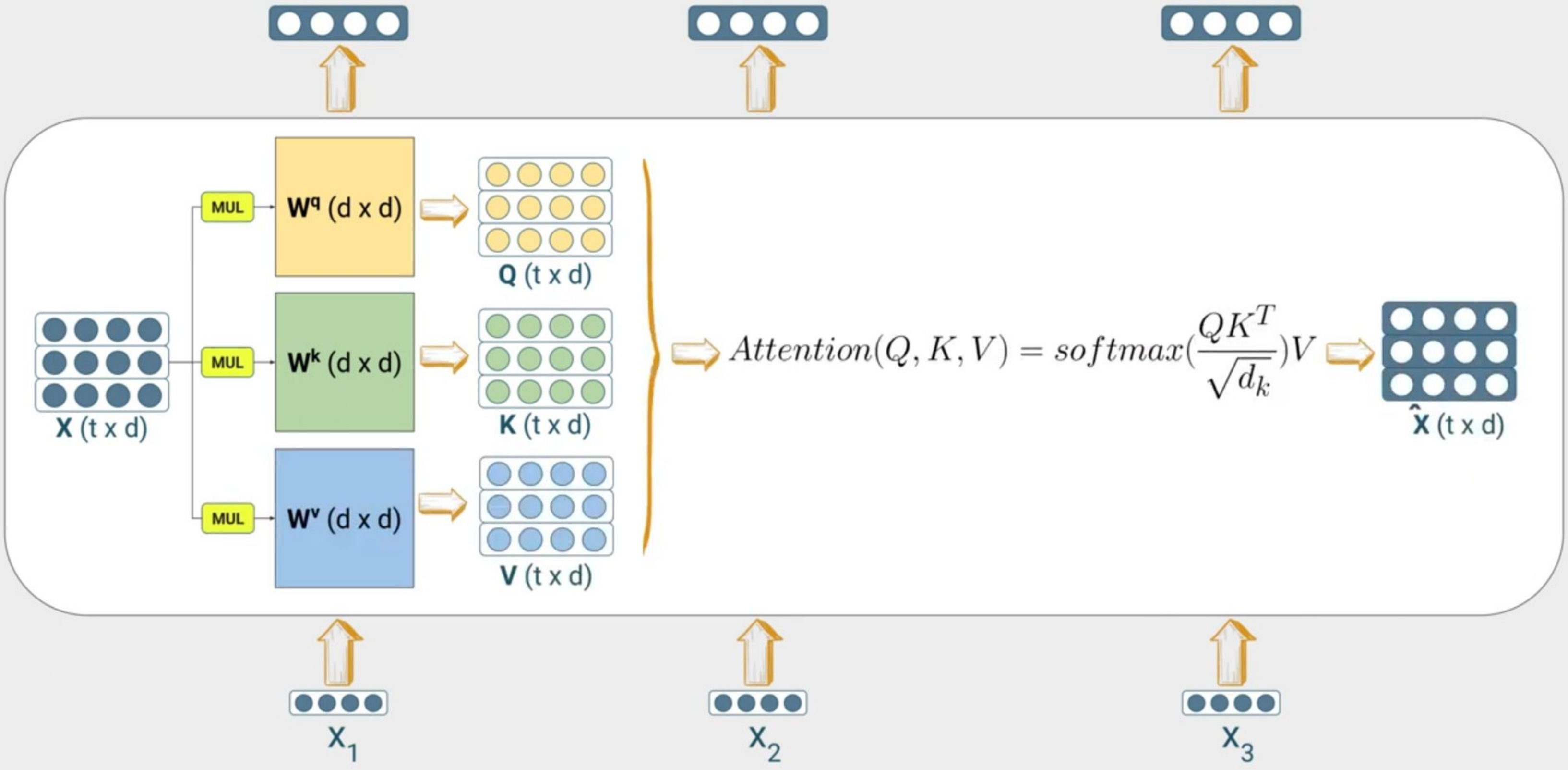
Scale Dot Product



Scale Dot Product Self-Attention



Attention Head

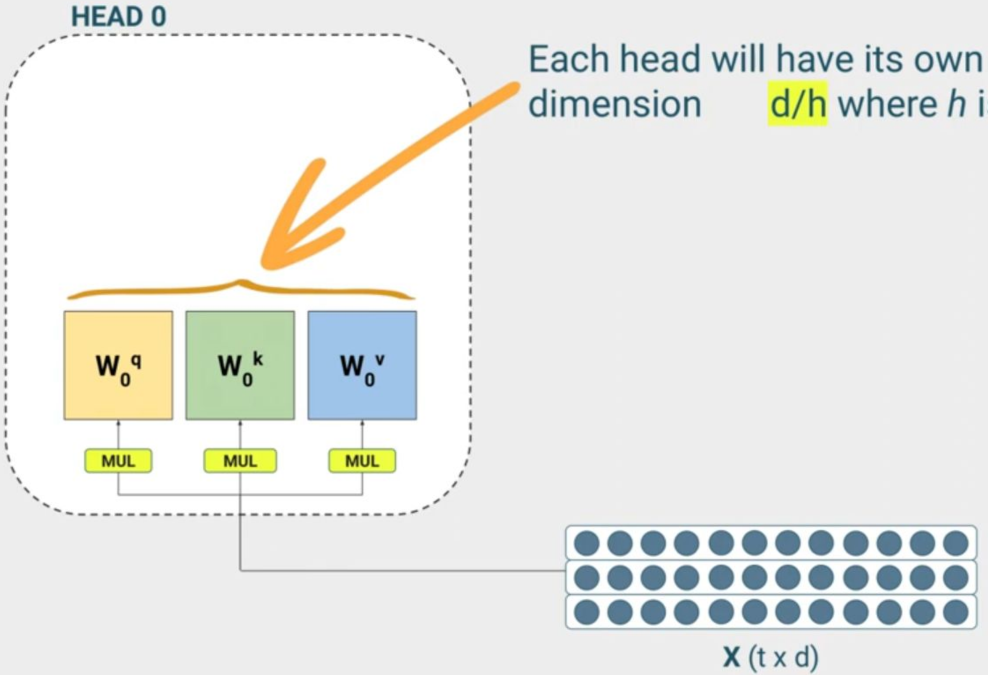


“Sarah went to a restaurant to meet her friend that night.”

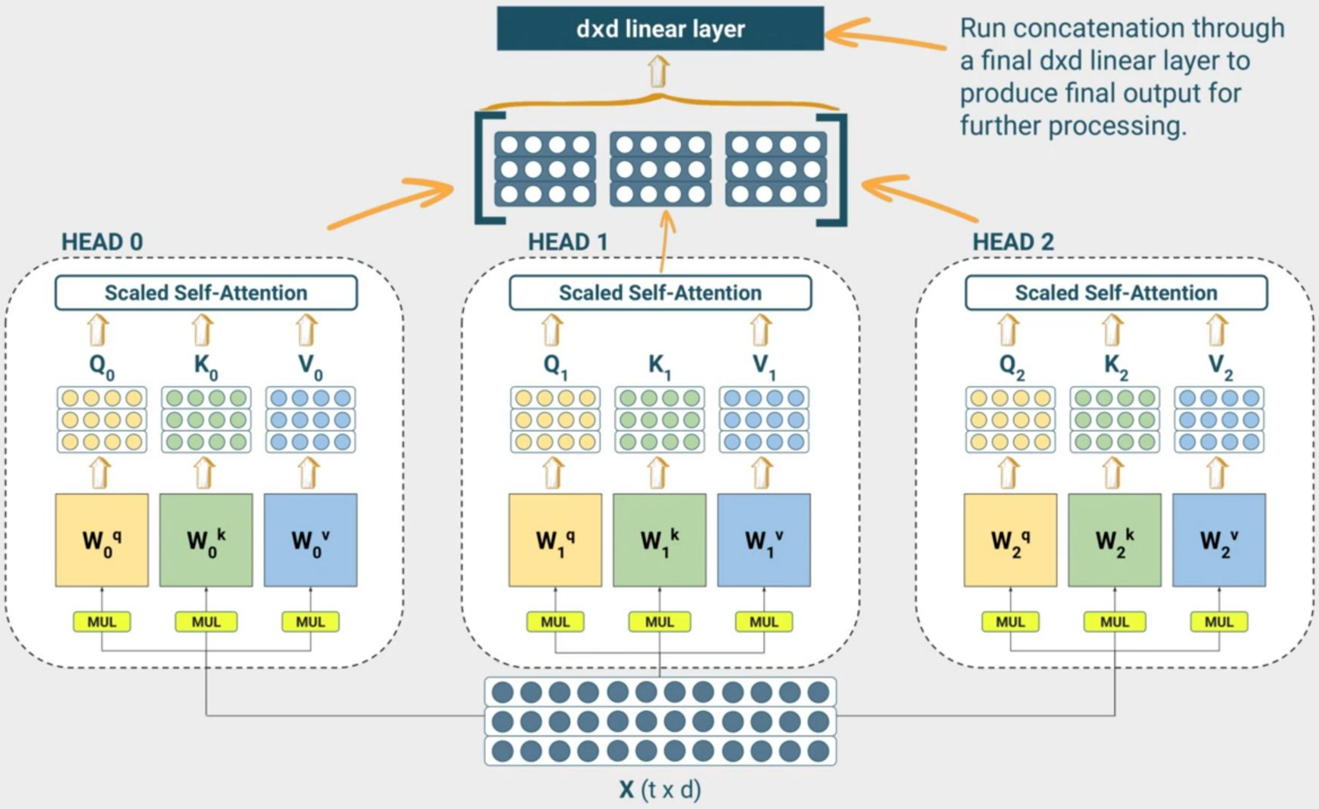
- What?
- Where?
- Who?
- When?

Multi-Head Self-Attention

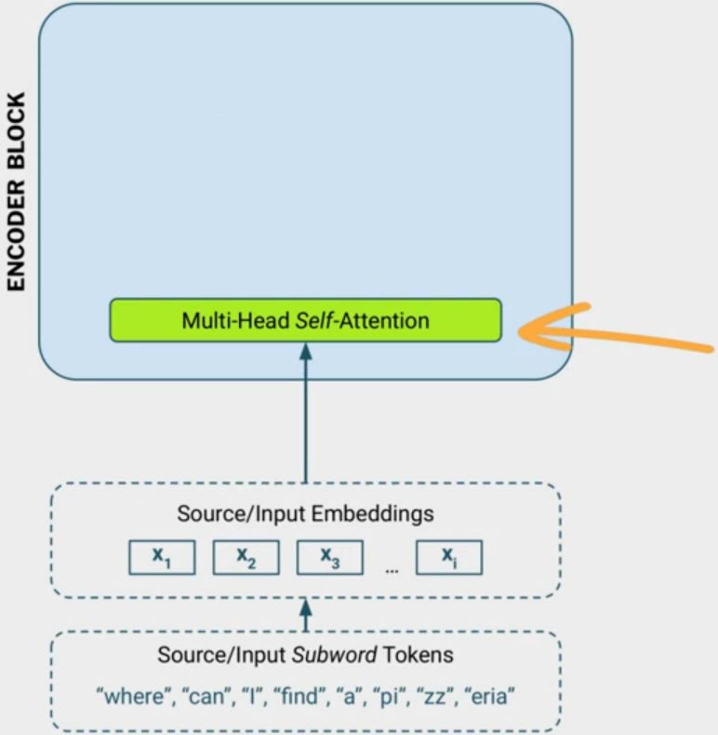
Each head will have its own set of Q, K, and V weights of dimension d/h where *h* is the number of heads.



Multi-Head Self-Attention



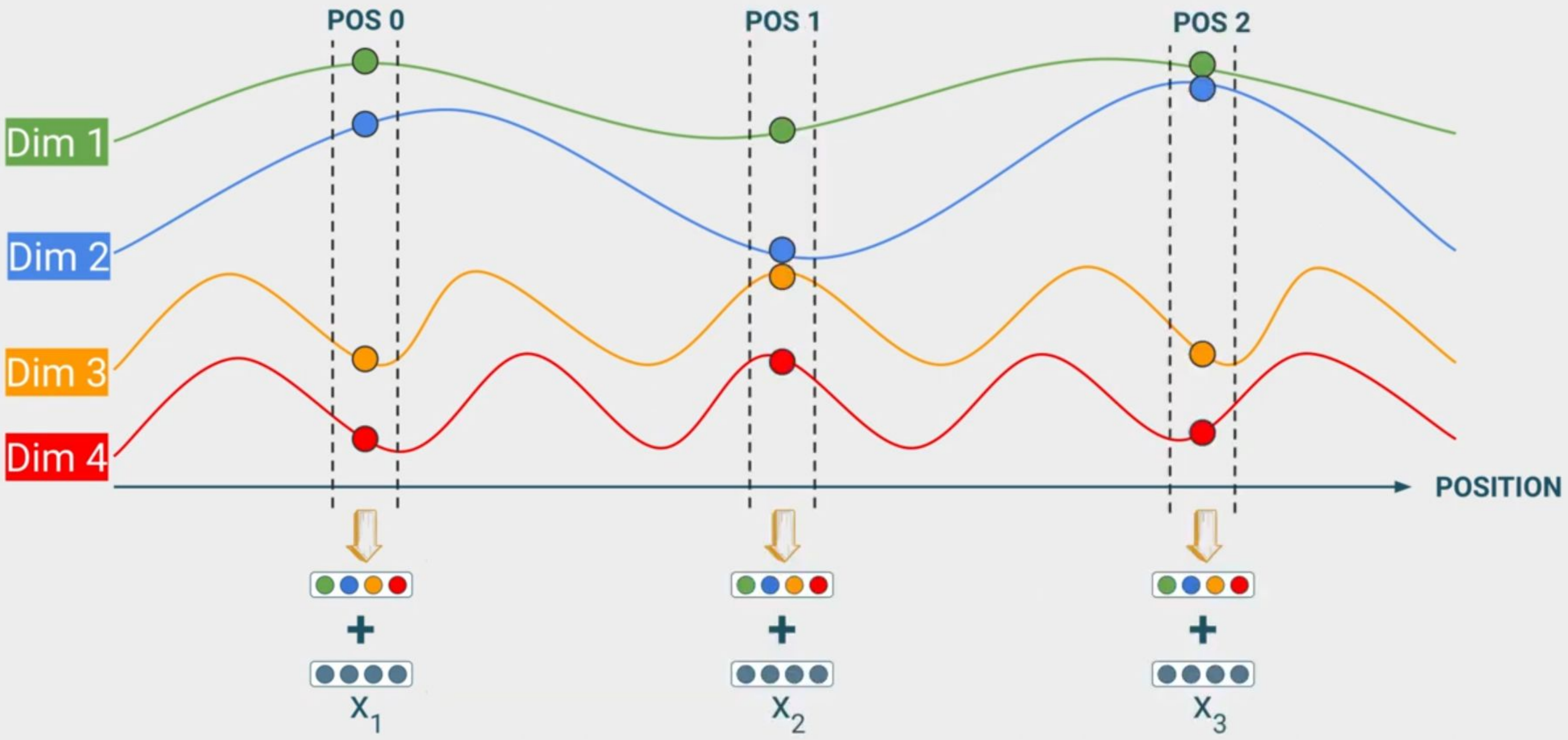
Paper



Original Transformer Dimensions:

- Embedding dimension of 512. This is known as the *model dimension* ($\mathbf{d}_{\texttt{model}}$).
- 8 attention heads, so d/h = 64.

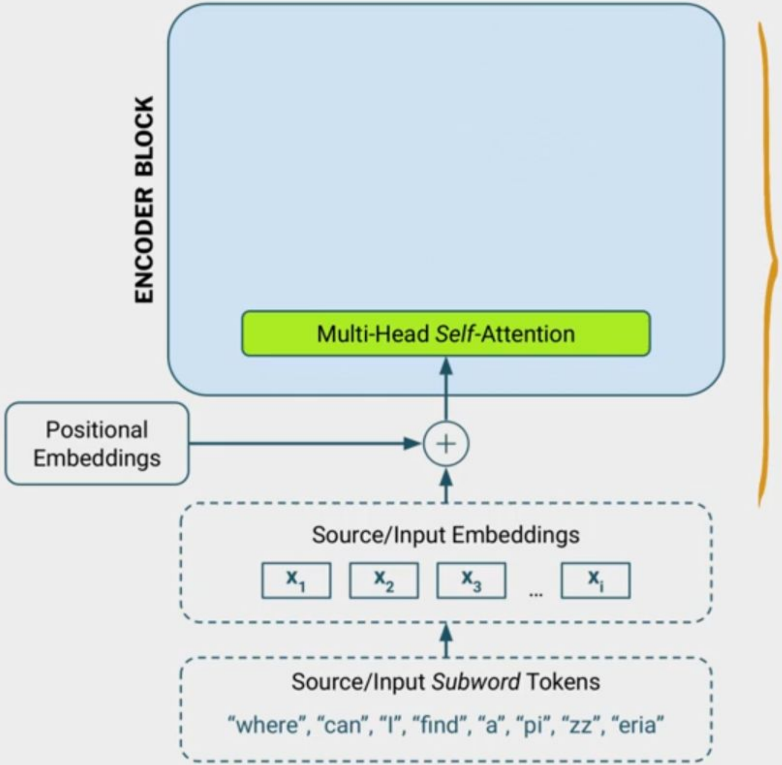
Add Positional Information



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

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

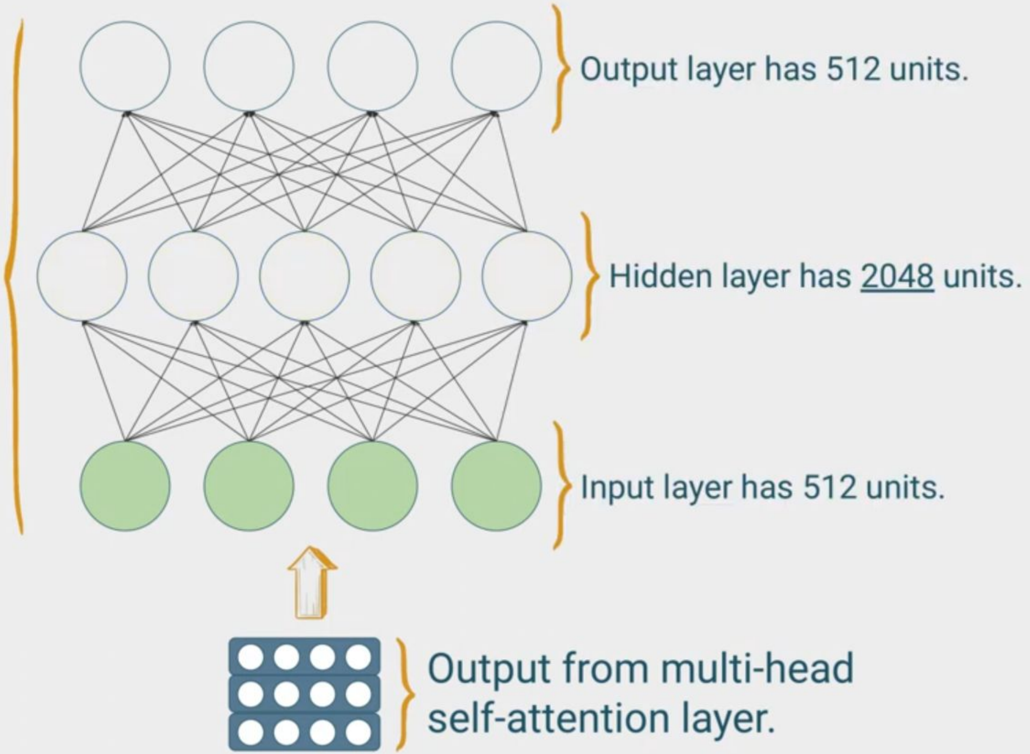
Transformers



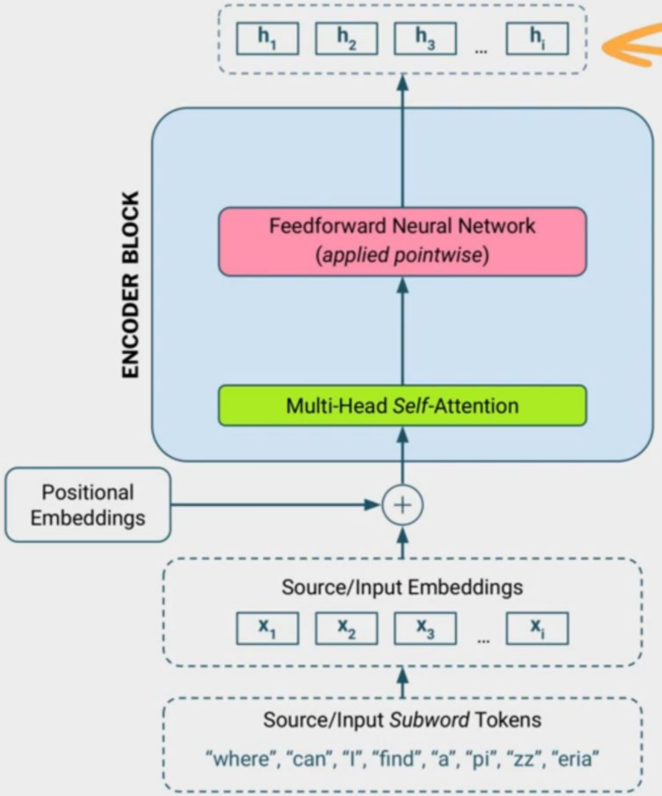
We added a number of learnable weights but there's still no non-linearity...

Adding Non-Linearity

Original transformer used a two-layer network with a ReLU activation in the hidden layer.

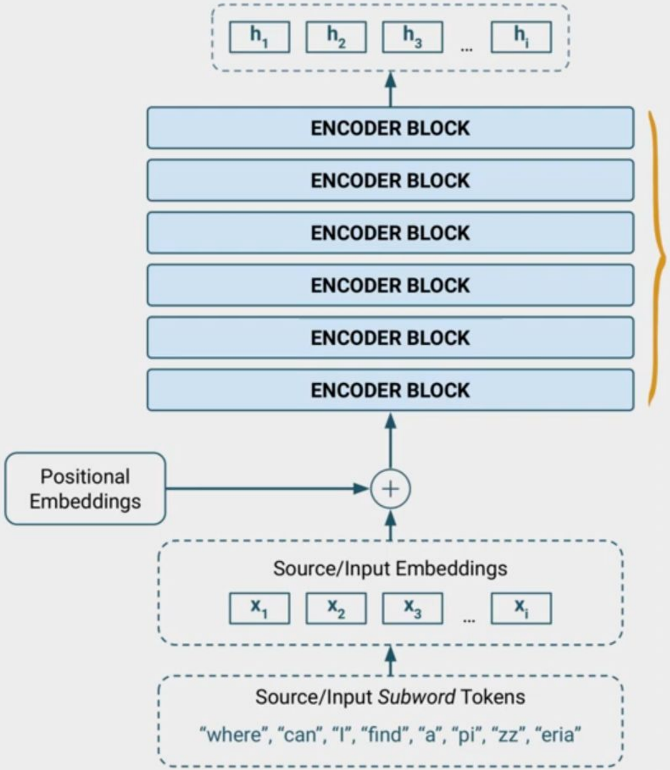


Transformers



Encoder output embeddings have the same dimensions as input embeddings (512).

Transformers

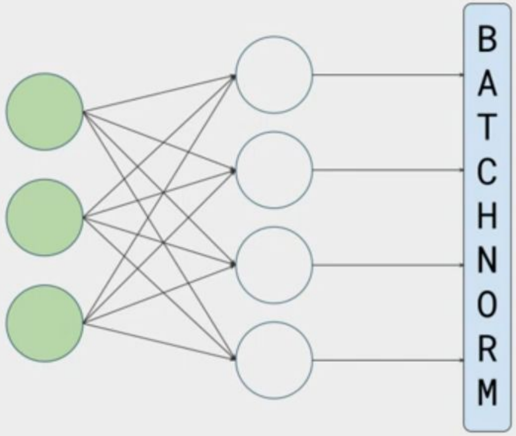


Two potential issues:

1. Shifting inputs from earlier encoder blocks add noise.

2. Depth leads to earlier information (e.g. positional embeddings) being "forgotten" over blocks, and vanishing gradients.

Batch Normalization



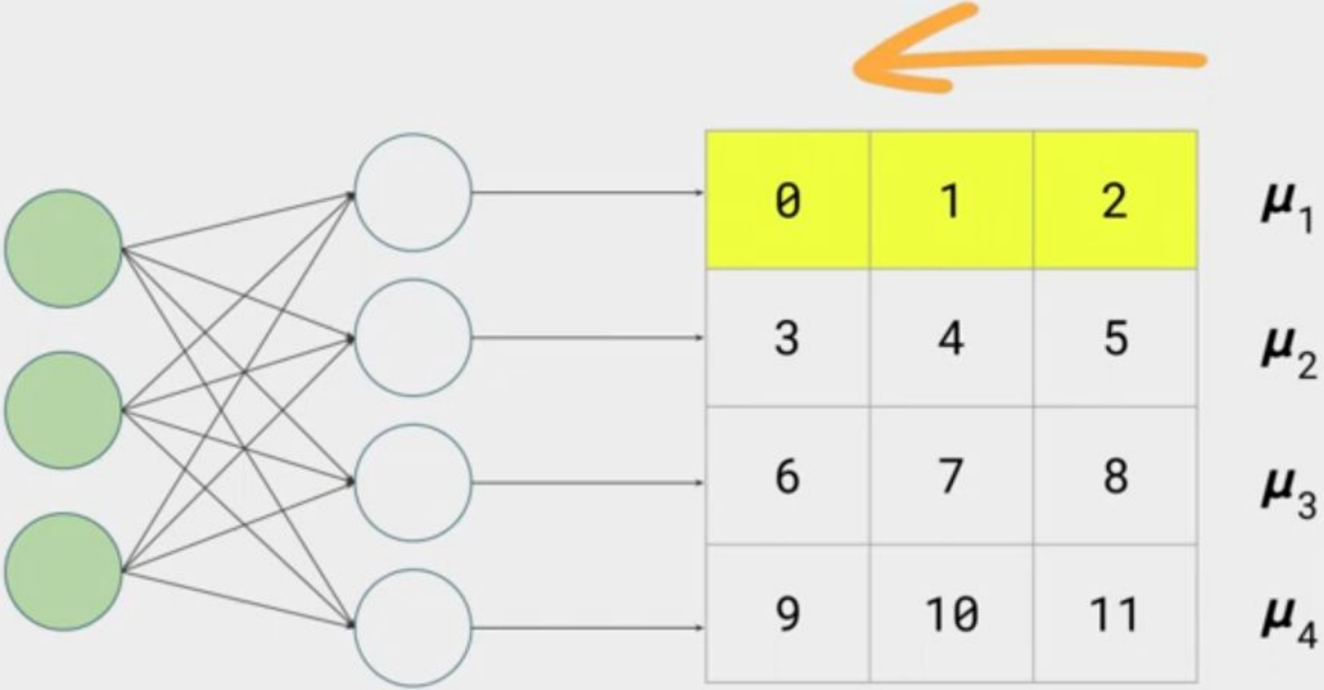
$$\boldsymbol{\mu}_b = \frac{1}{m_b}\sum_{i=1}^{m_b}\mathbf{h}_i$$ Vector of mini-batch output means.

$$\boldsymbol{\sigma}_b^2 = \frac{1}{m}\sum_{i=1}^{m_b}(\mathbf{h}_i - \boldsymbol{\mu}_b)^2$$ Vector of mini-batch output standard deviations.

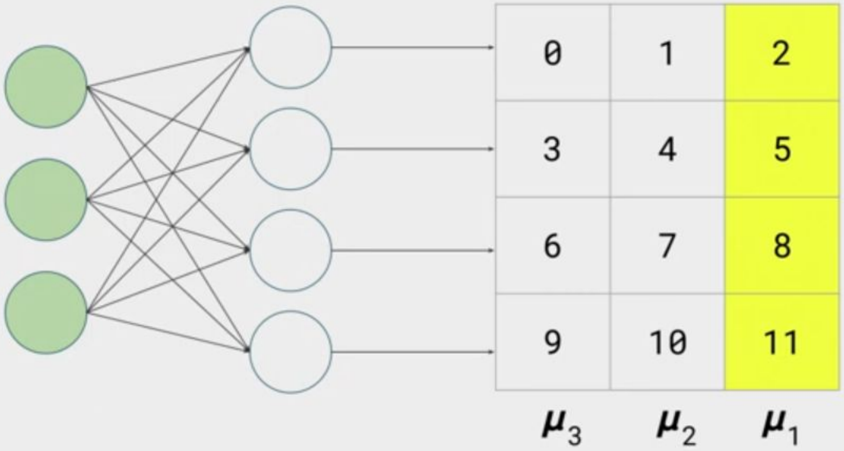
$$\hat{\mathbf{h}}_i = \frac{\mathbf{h}_i - \boldsymbol{\mu}_b}{\sqrt{\boldsymbol{\sigma}_b^2 + \epsilon}}$$ Vector of standardized outputs.

$$\mathbf{z}_i = \boldsymbol{\gamma} \odot \hat{\mathbf{h}}_i + \boldsymbol{\beta}$$ Vector of *scaled* and *shifted* outputs.

Batch Normalization



Layer Normalization



$$\mu = \frac{1}{d_h} \sum_{i=1}^{d_h} x_i$$

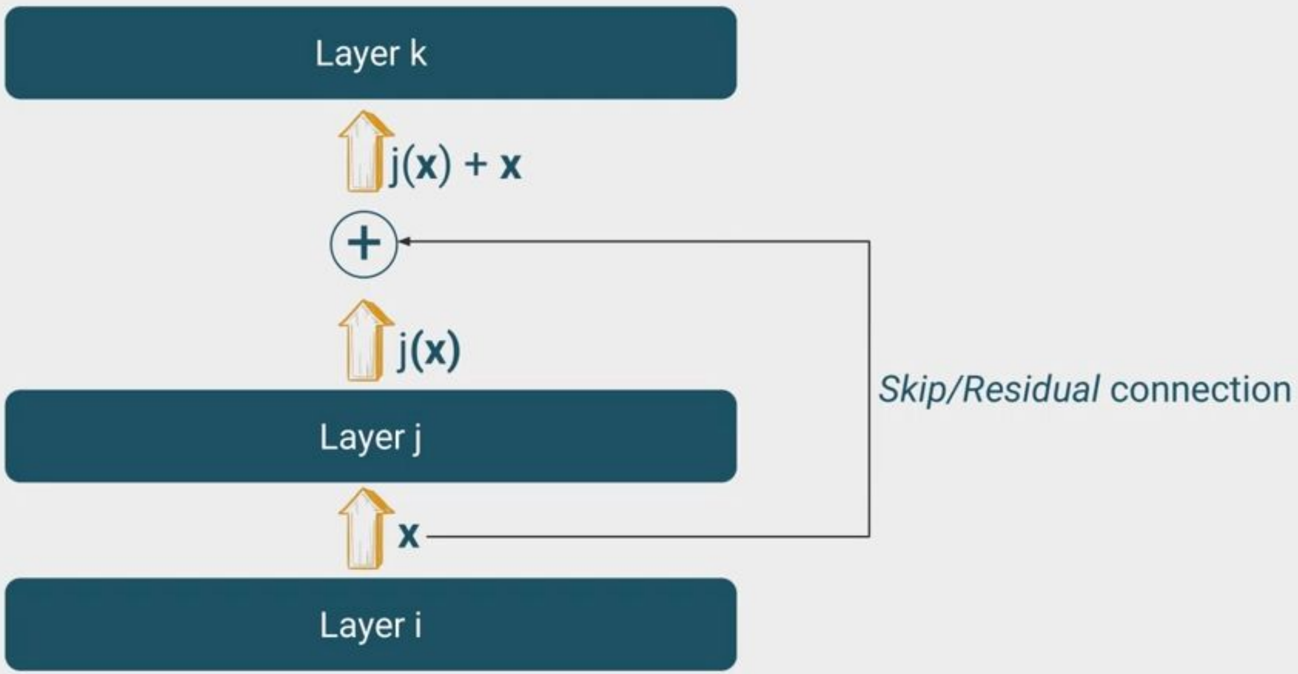
$$\sigma = \sqrt{\frac{1}{d_h} \sum_{i=1}^{d_h} (x_i - \mu)^2}$$

$$\hat{\mathbf{x}} = \frac{\mathrm{x} - \mu}{\sigma}$$

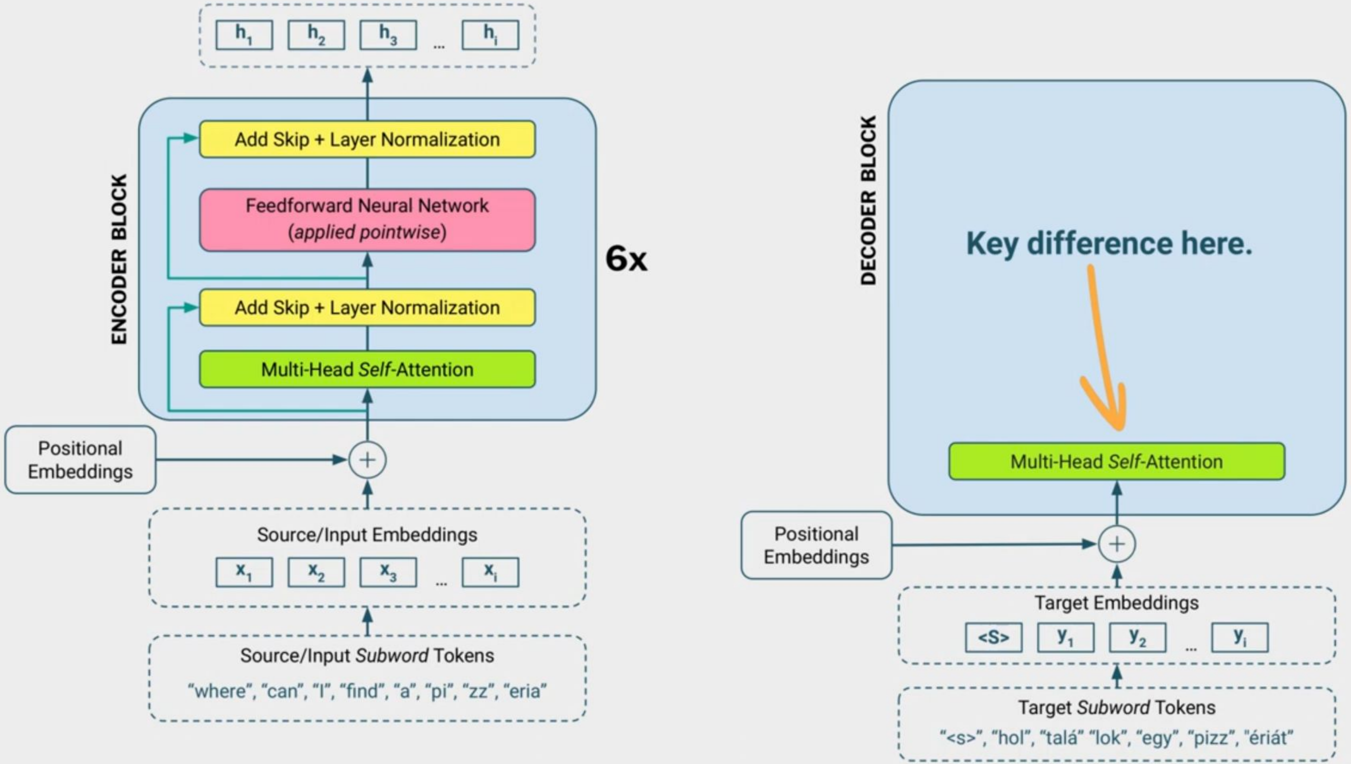
Learnable parameters to scale and shift as needed.

$$\mathbf{z} = \gamma \hat{\mathbf{x}} + \beta$$

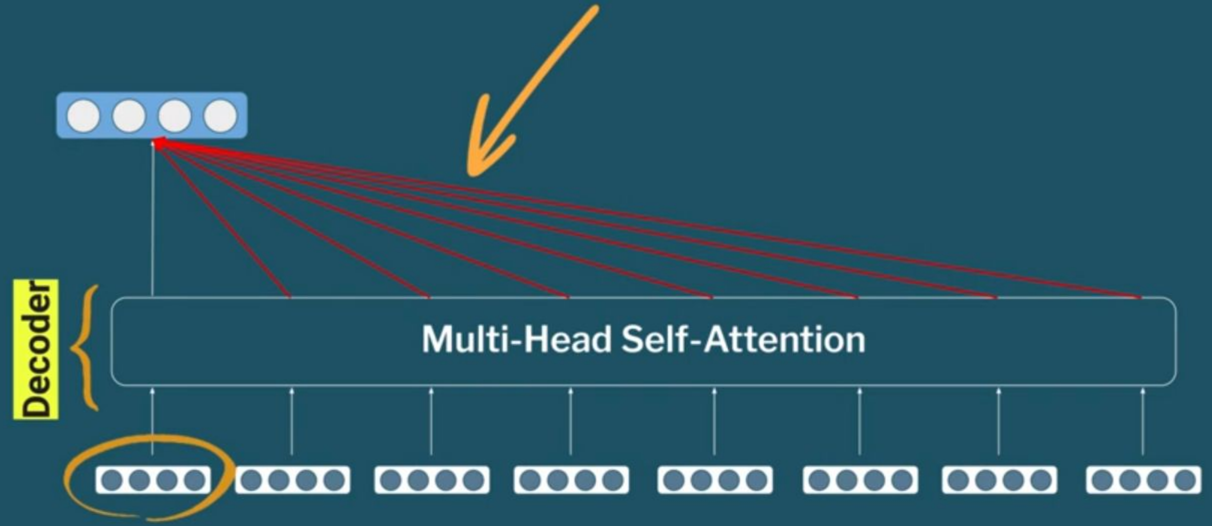
Skip\Residual Connection

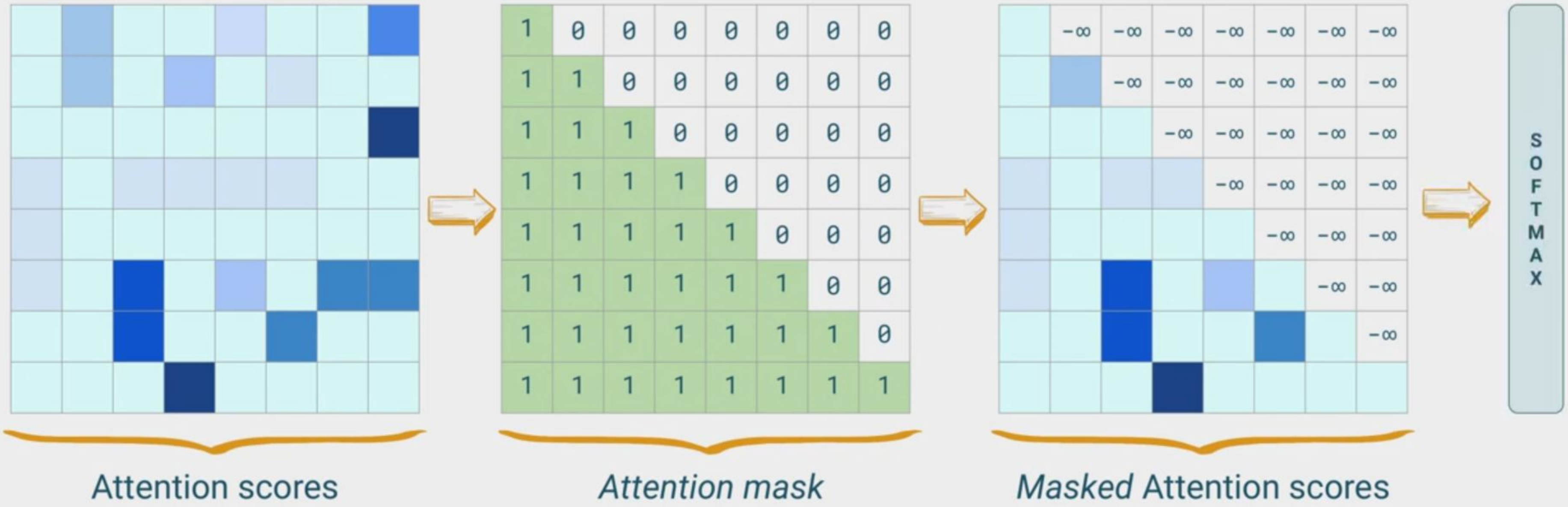


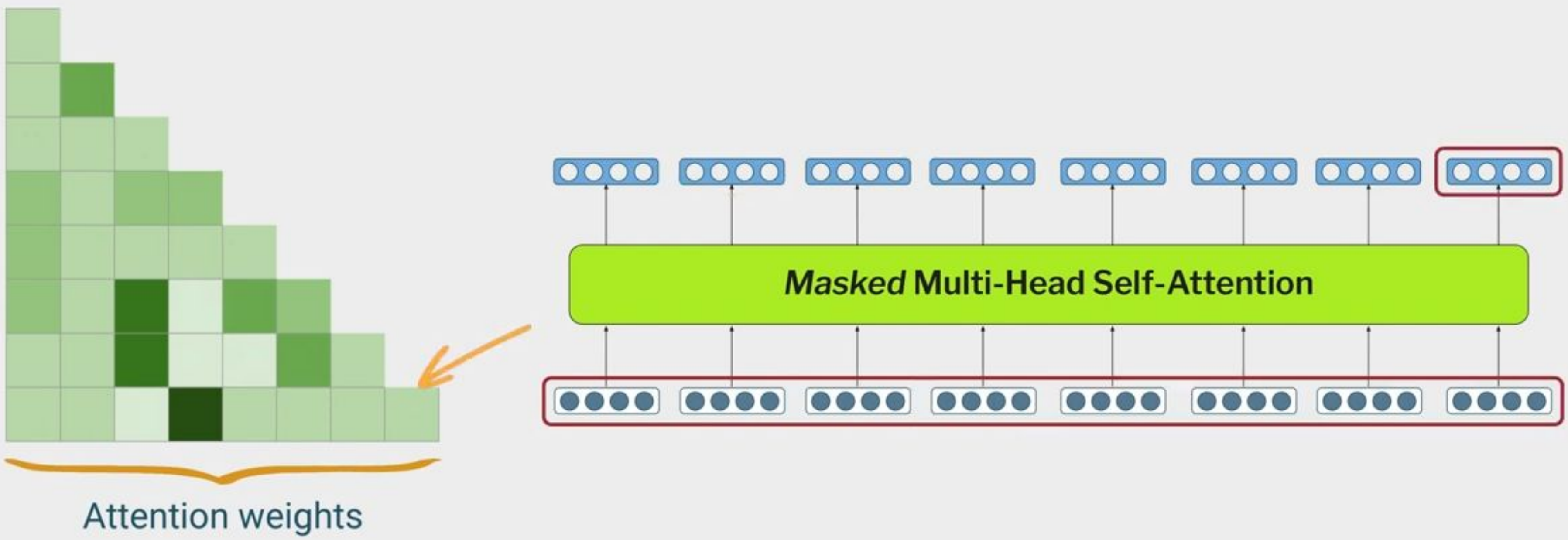
Transformers

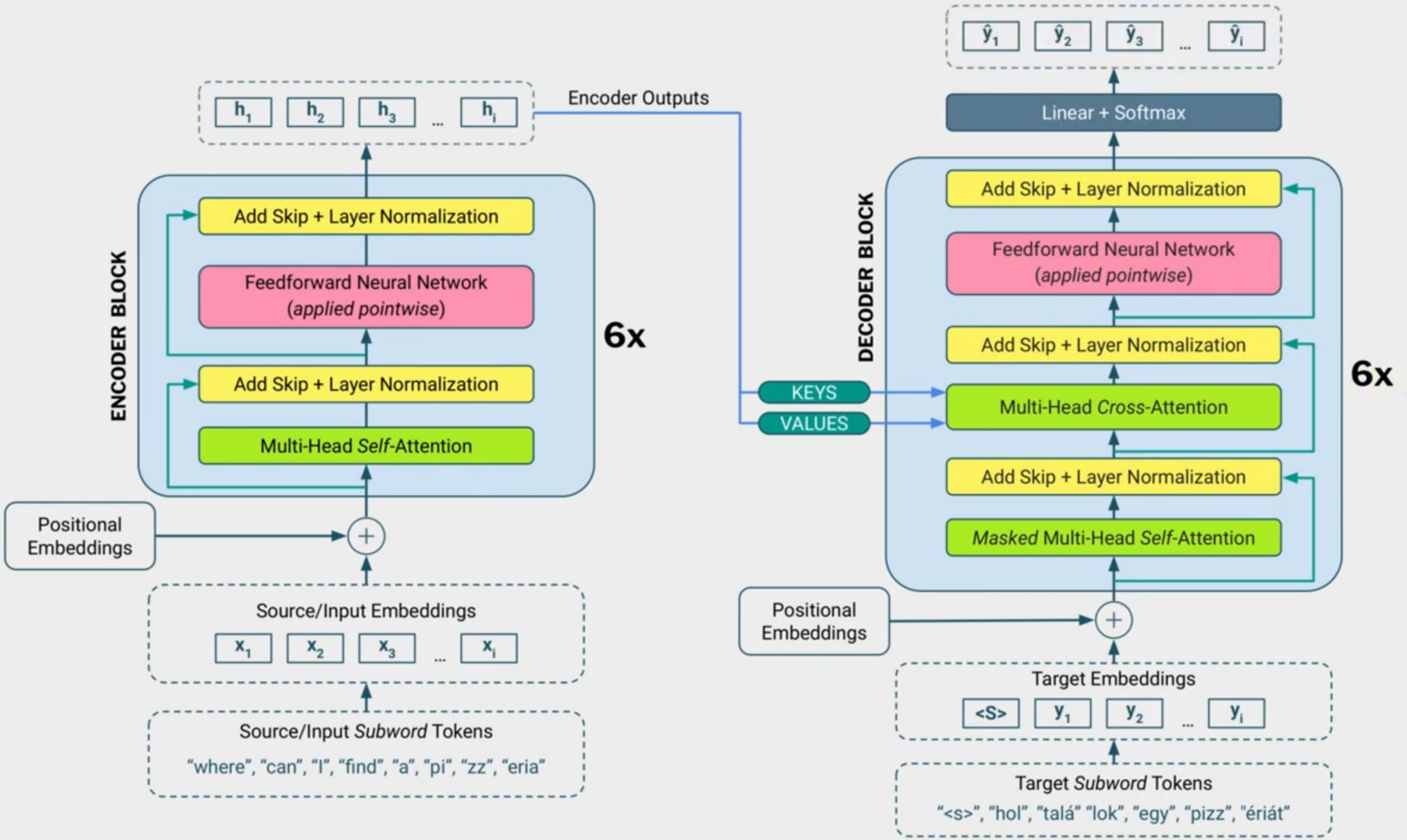


We need to block decoder from accessing any future parts of the sequence.




1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0
1 1 1 1 0 0 0 0
1 1 1 1 1 0 0 0
1 1 1 1 1 1 0 0
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1
-∞ -∞ -∞ -∞ -∞ -∞ -∞
-∞ -∞ -∞ -∞ -∞ -∞
-∞ -∞ -∞ -∞ -∞
-∞ -∞ -∞ -∞
-∞ -∞ -∞
-∞ -∞
-∞
SOFTMAX
Attention scores
Attention mask
Masked Attention scores


Masked Multi-Head Self-Attention
Attention weights



$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ ... $\hat{y}_i$
Linear + Softmax
h_1 h_2 h_3 ... h_i
Encoder Outputs
ENCODER BLOCK
Add Skip + Layer Normalization
Feedforward Neural Network (*applied pointwise*)
Add Skip + Layer Normalization
Multi-Head *Self*-Attention
6x
Positional Embeddings
Source/Input Embeddings
x_1 x_2 x_3 ... x_i
Source/Input *Subword* Tokens
"where", "can", "I", "find", "a", "pi", "zz", "eria"
DECODER BLOCK
Add Skip + Layer Normalization
Feedforward Neural Network (*applied pointwise*)
Add Skip + Layer Normalization
KEYS
VALUES
Multi-Head *Cross*-Attention
Add Skip + Layer Normalization
Masked Multi-Head *Self*-Attention
6x
Positional Embeddings
Target Embeddings
<S> y_1 y_2 ... y_i
Target *Subword* Tokens
"<s>", "hol", "talá" "lok", "egy", "pizz", "ériát"