THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 11: RNNs (Part 3), Applications

Course **Logistics**

- **Assignment 3** due date is Monday -> Wednesday
- **Assignment 4** is released Monday

- **Assignment 1** & **2** solutions are out

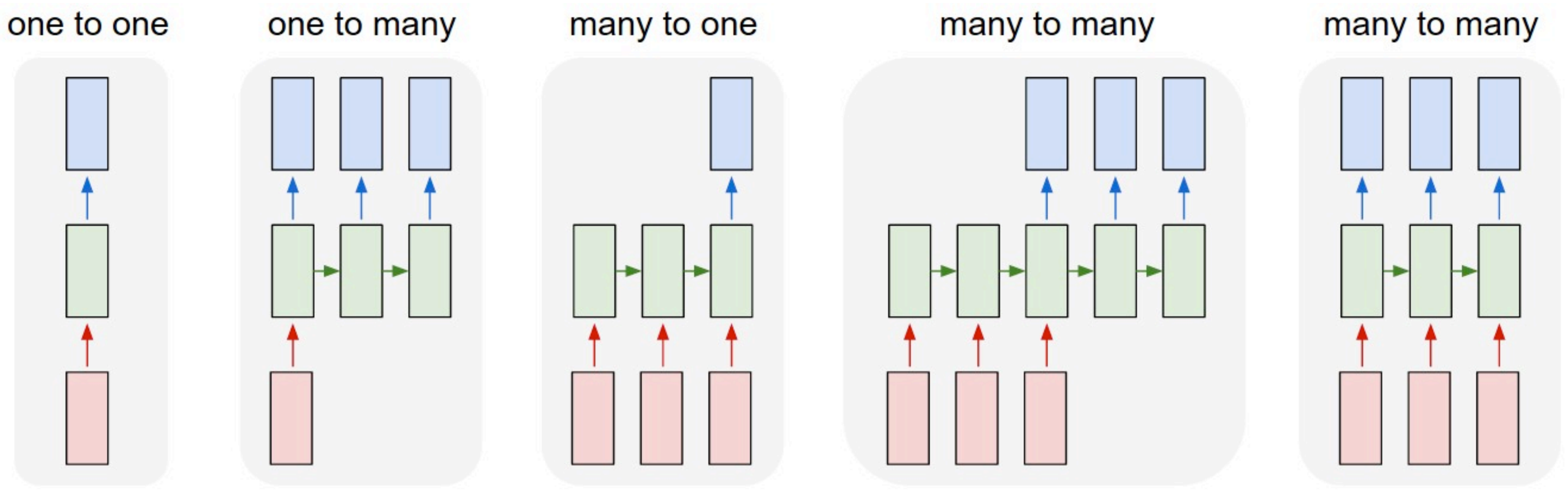
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



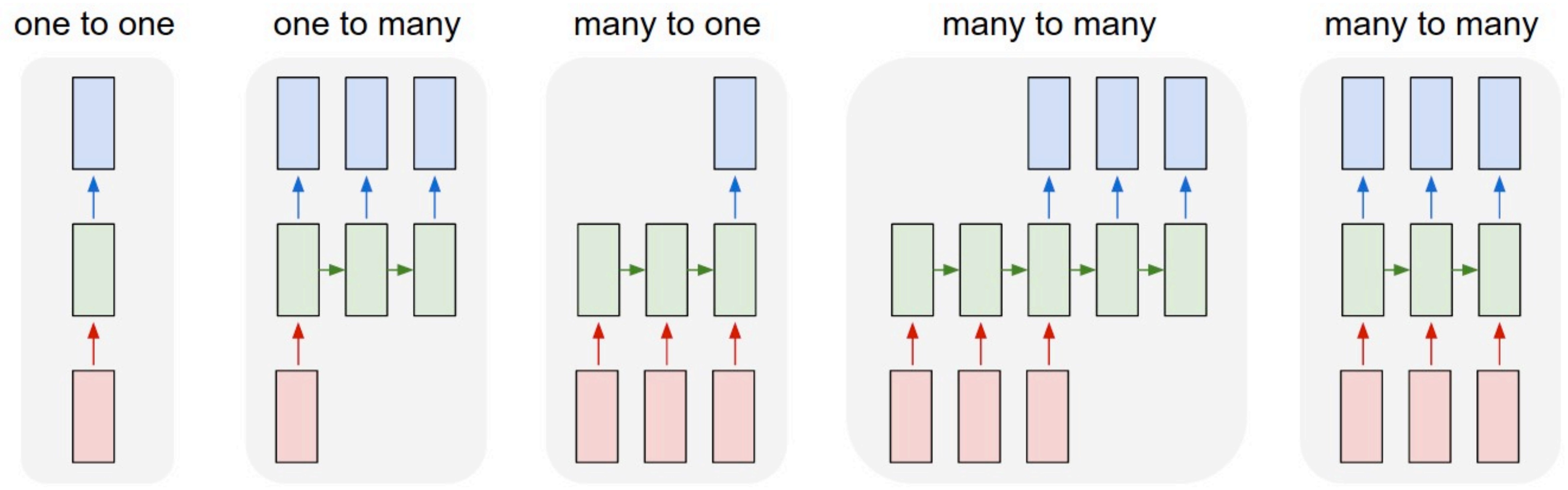
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



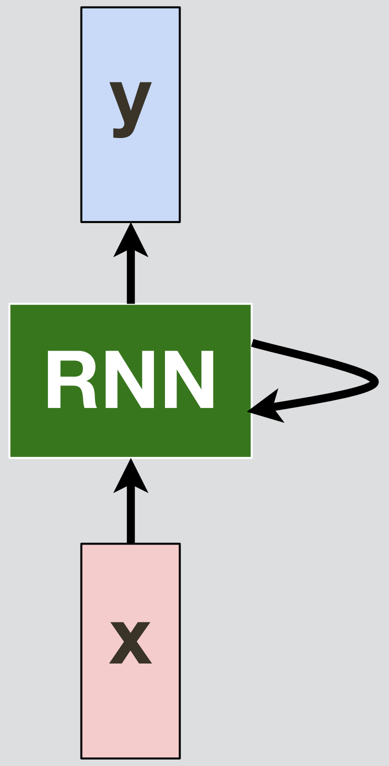
Vanilla **RNN**

$$y_t = W_{hy} h_t + b_y$$

$$h_t = f_W(h_{t-1}, x_t)$$

$\downarrow$

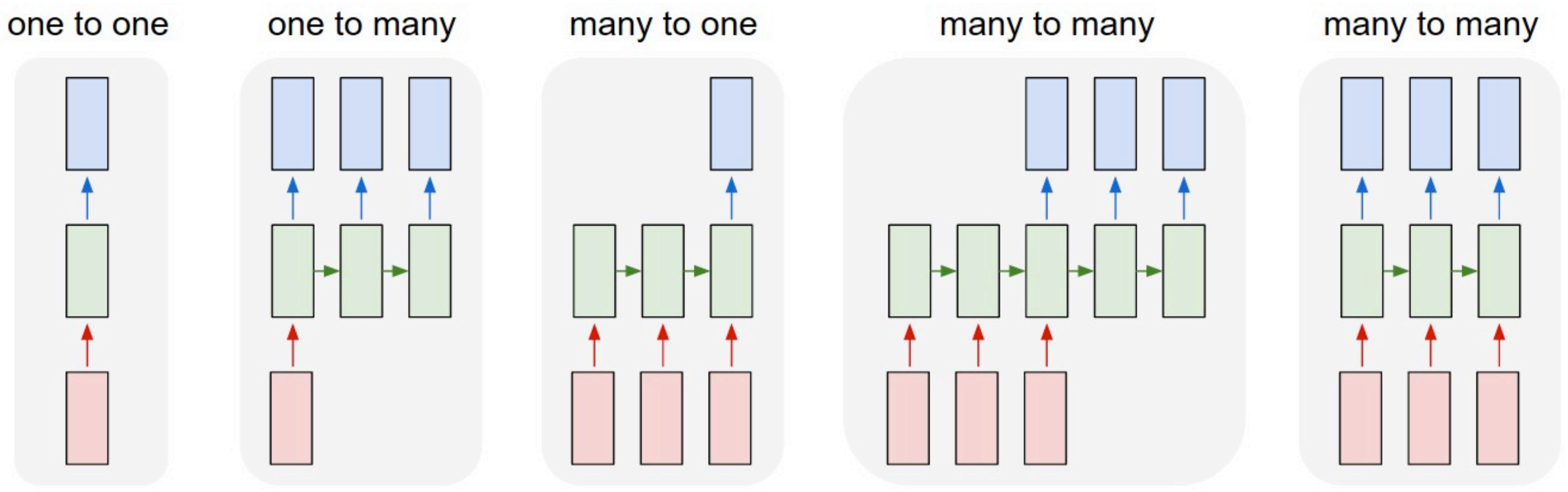
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$



RNNs: Review

Key Enablers:

- Parameter sharing in computational graphs
- "Unrolling" in computational graphs
- Allows modeling **arbitrary length sequences**!



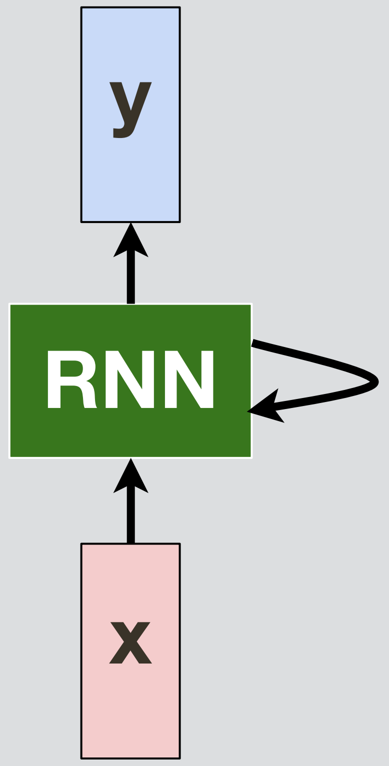
Vanilla **RNN**

$$y_t = W_{hy} h_t + b_y$$

$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$

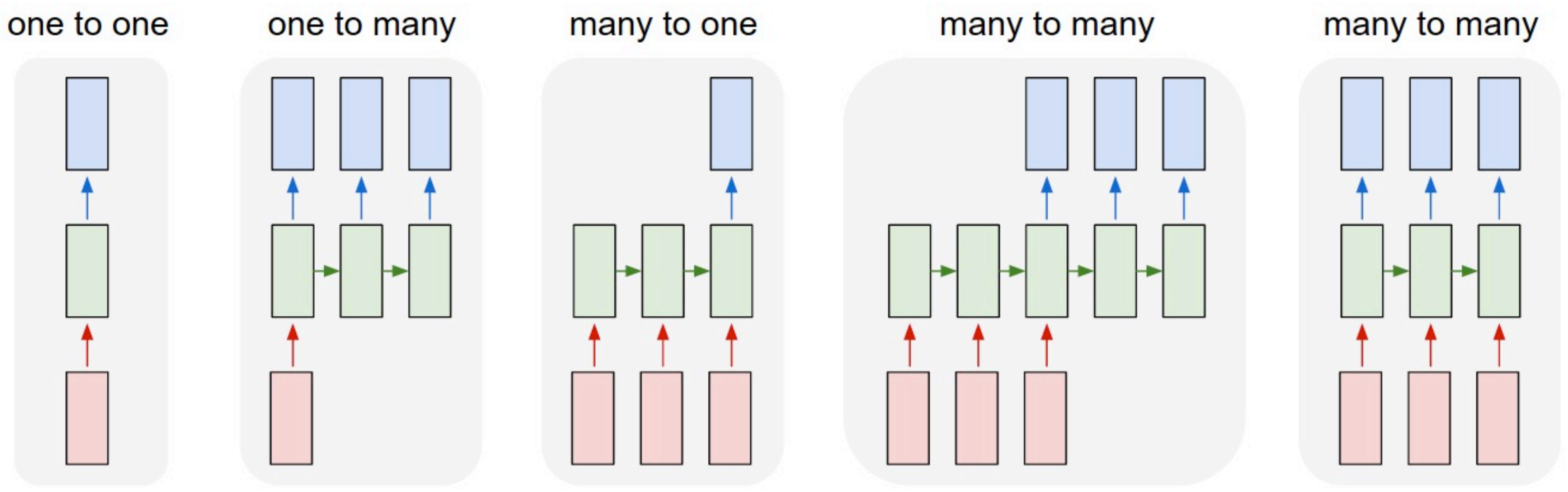


Vanishing
or
Exploding
Gradients

RNNs: Review

Key Enablers:

- Parameter sharing in computational graphs
- “Unrolling” in computational graphs
- Allows modeling **arbitrary length sequences**!

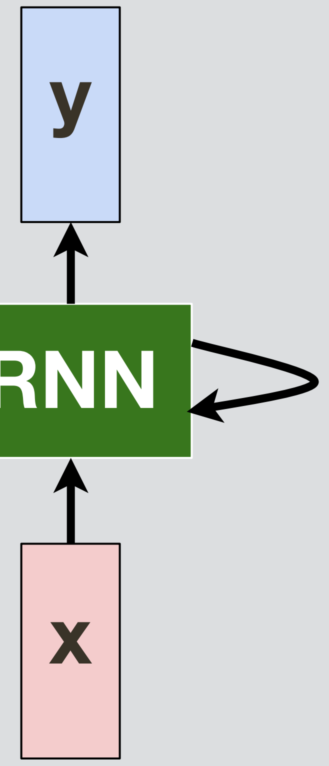


Vanilla **RNN**

$$y_t = W_{hy} h_t + b_y$$

$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$



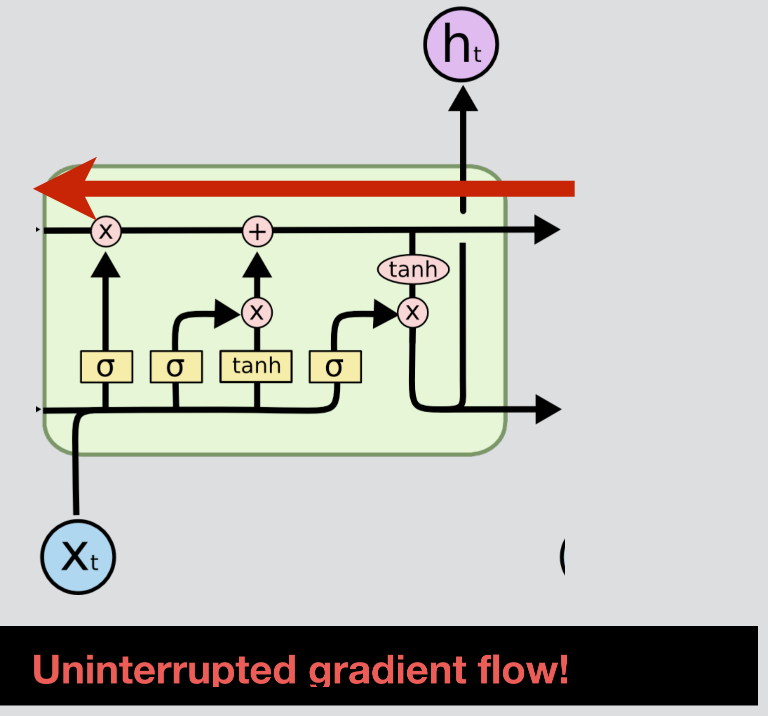
Vanishing or **Exploding** Gradients

Long-Short Term Memory (**LSTM**)

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$



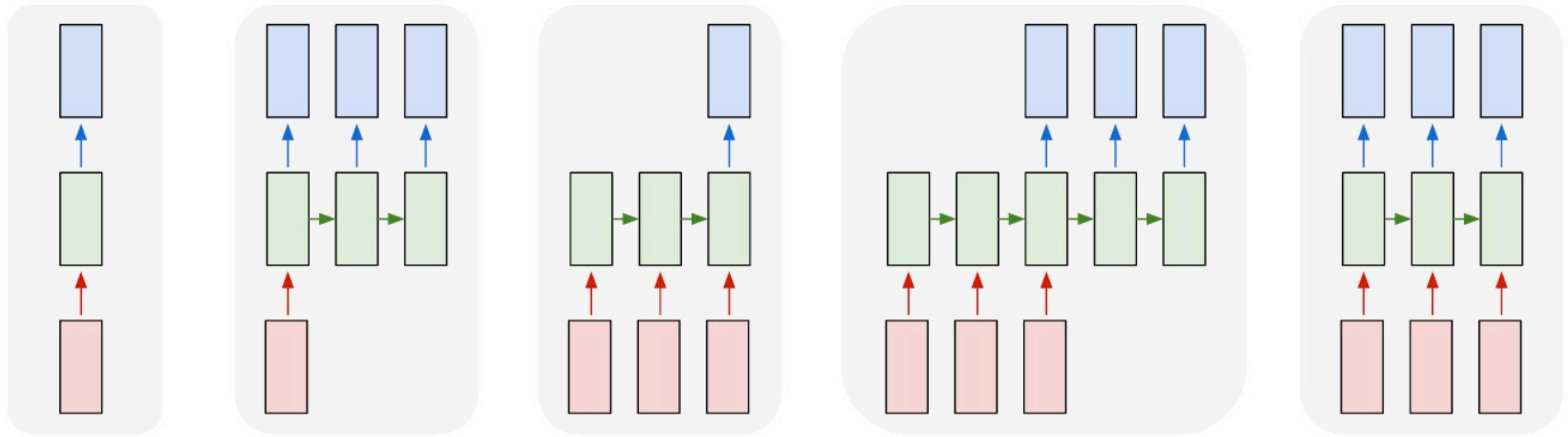
RNNs: Review

Key Enablers:

— Parameter sharing in computational graphs

— “Unrolling” in computational graphs

— Allows modeling **arbitrary length sequences**!



Loss functions: often cross-entropy (for classification); could be max-margin (like in SVM) or Squared Loss (regression)

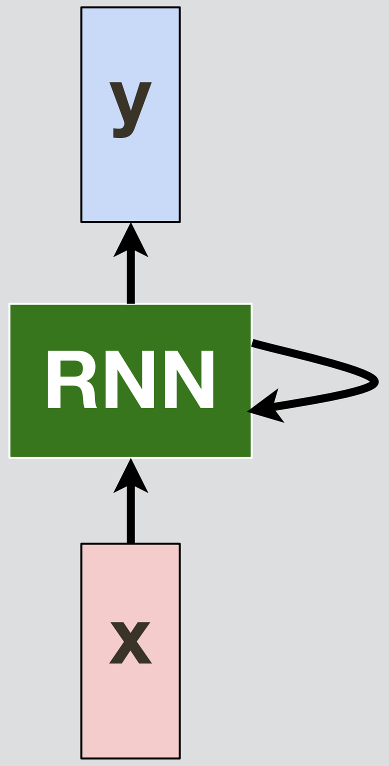
Vanilla **RNN**

$$y_t = W_{hy} h_t + b_y$$

$$h_t = f_W(h_{t-1}, x_t)$$

↓

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$



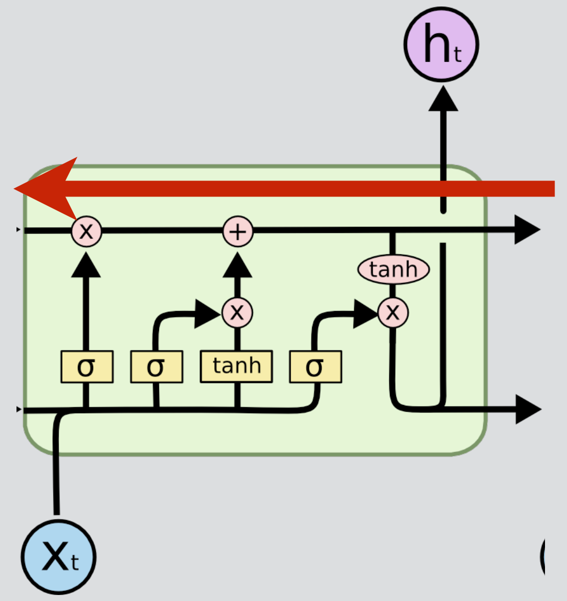
Vanishing or **Exploding** Gradients

Long-Short Term Memory (**LSTM**)

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

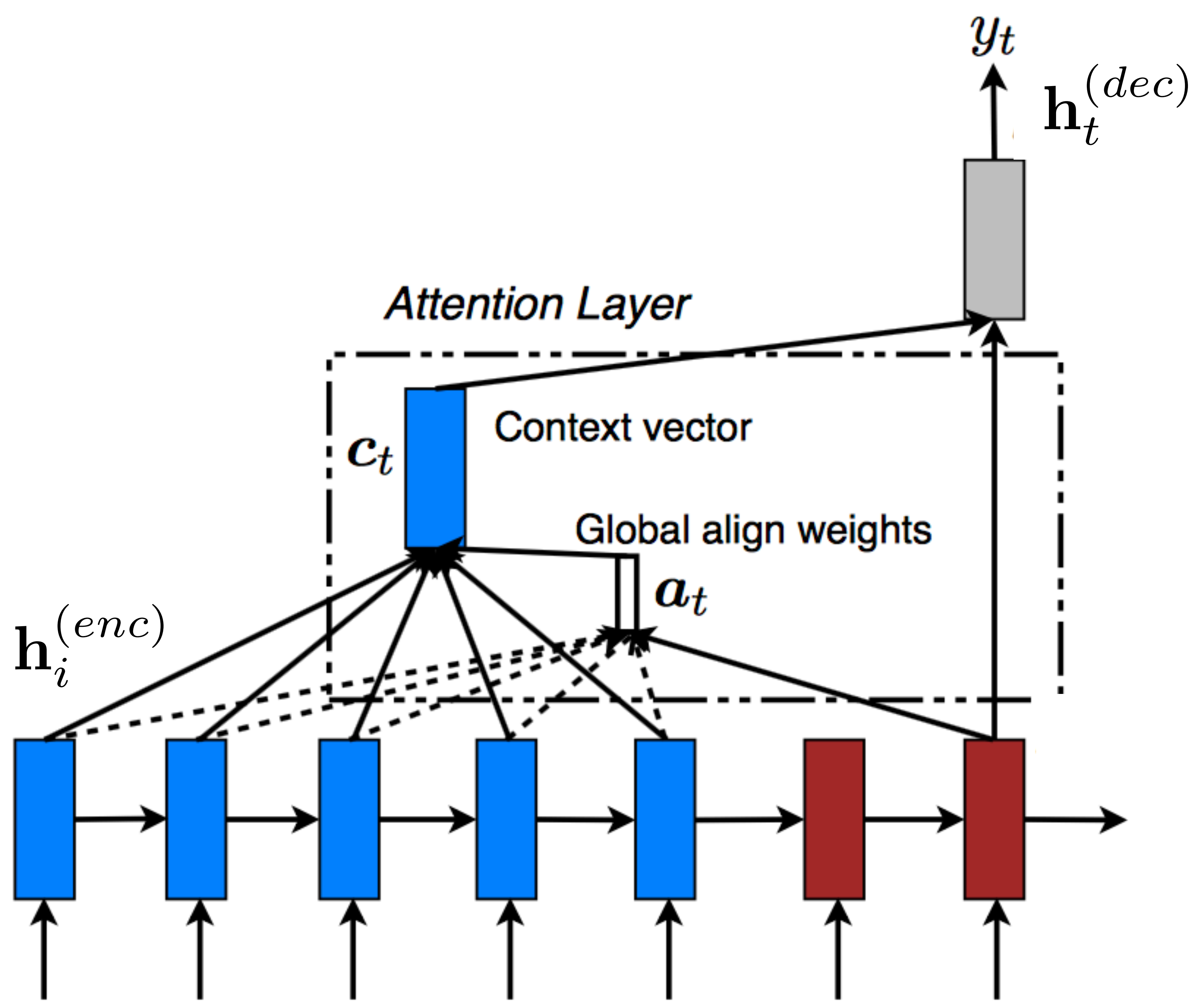
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$



Uninterrupted gradient flow!

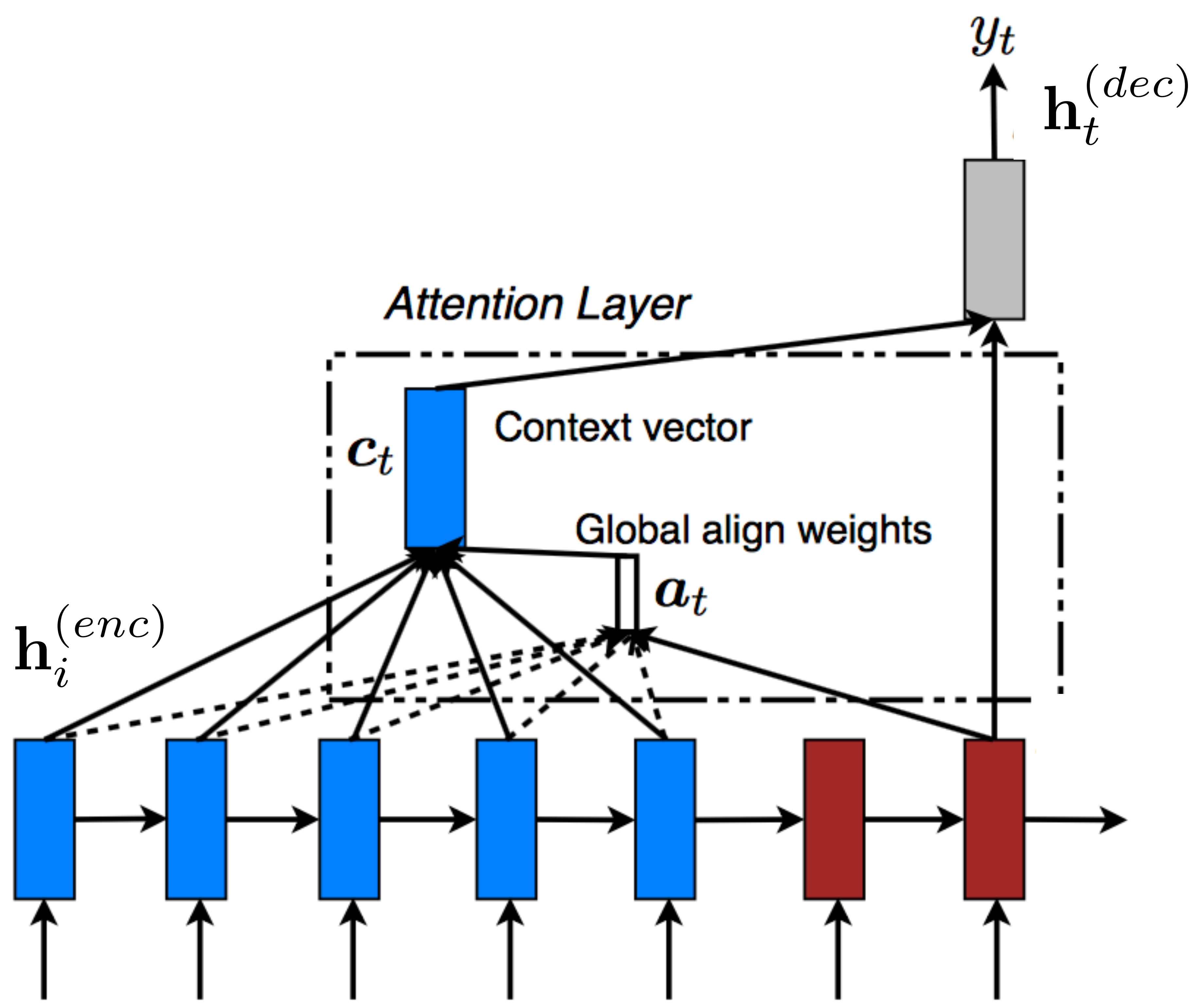
Soft **Attention** in details



Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

Relevance of encoding at token i for decoding token t



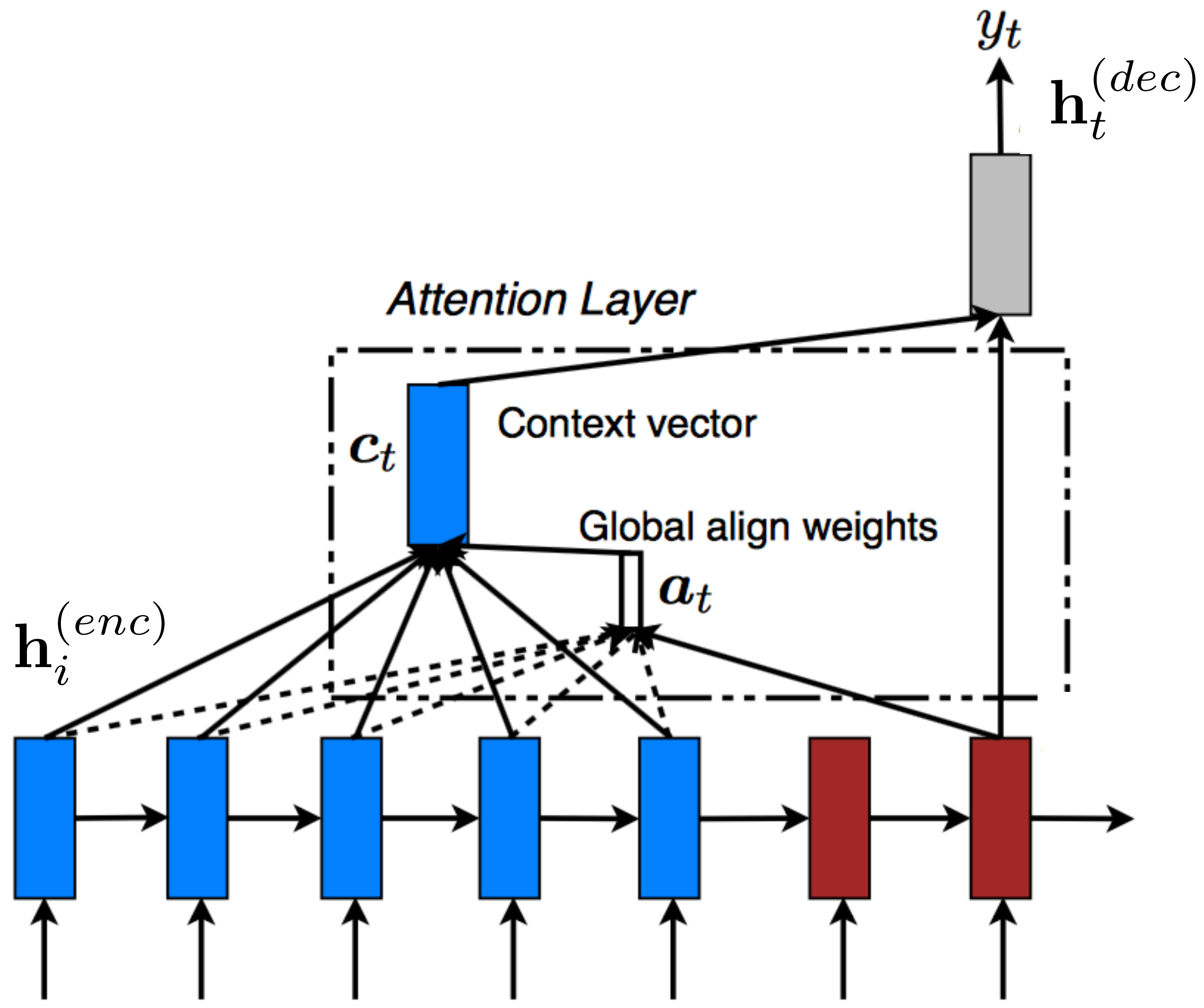
Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

Relevance of encoding at token i for decoding token t

$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

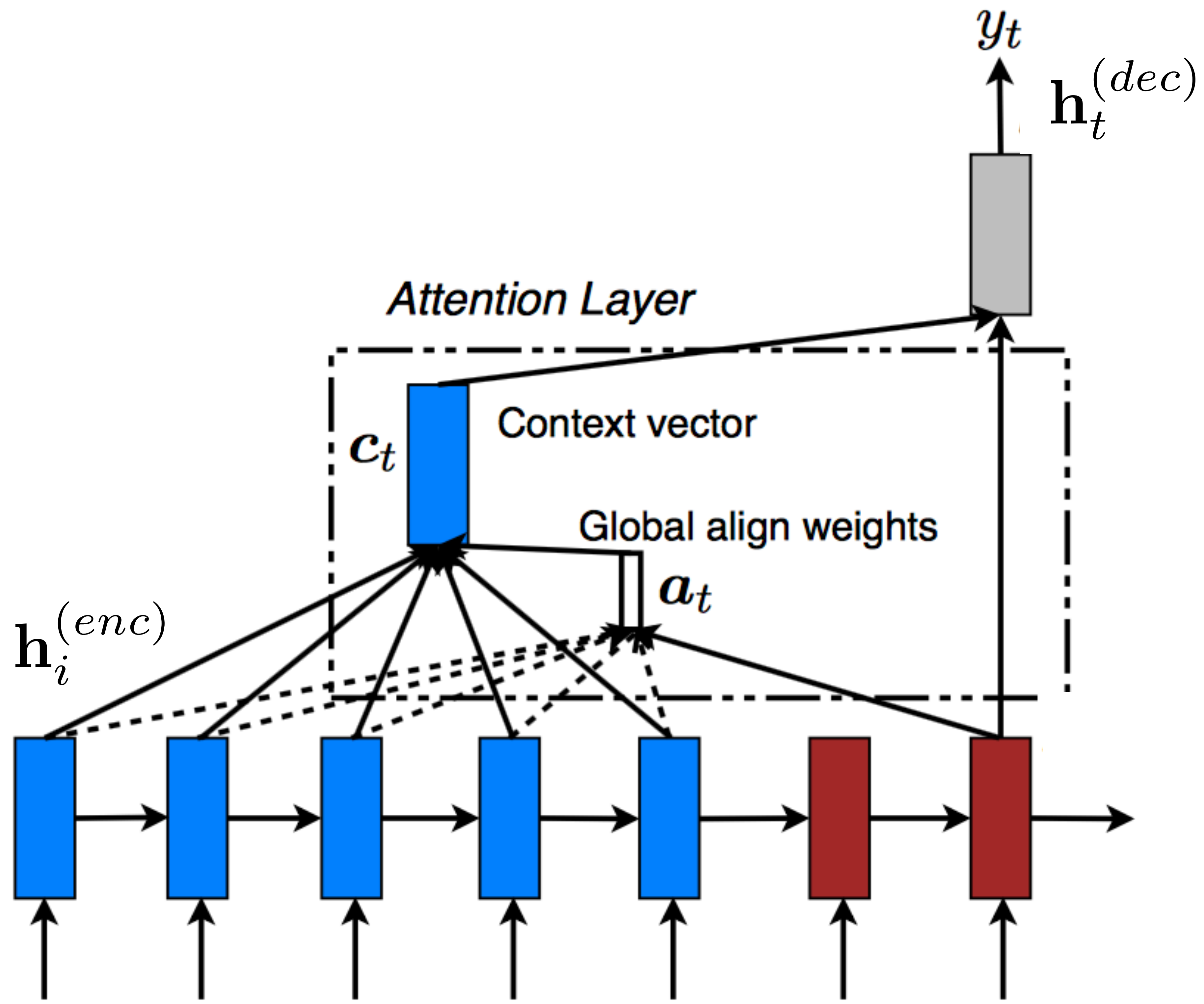
Normalize the weights to sum to 1



Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

Relevance of encoding at token i for decoding token t

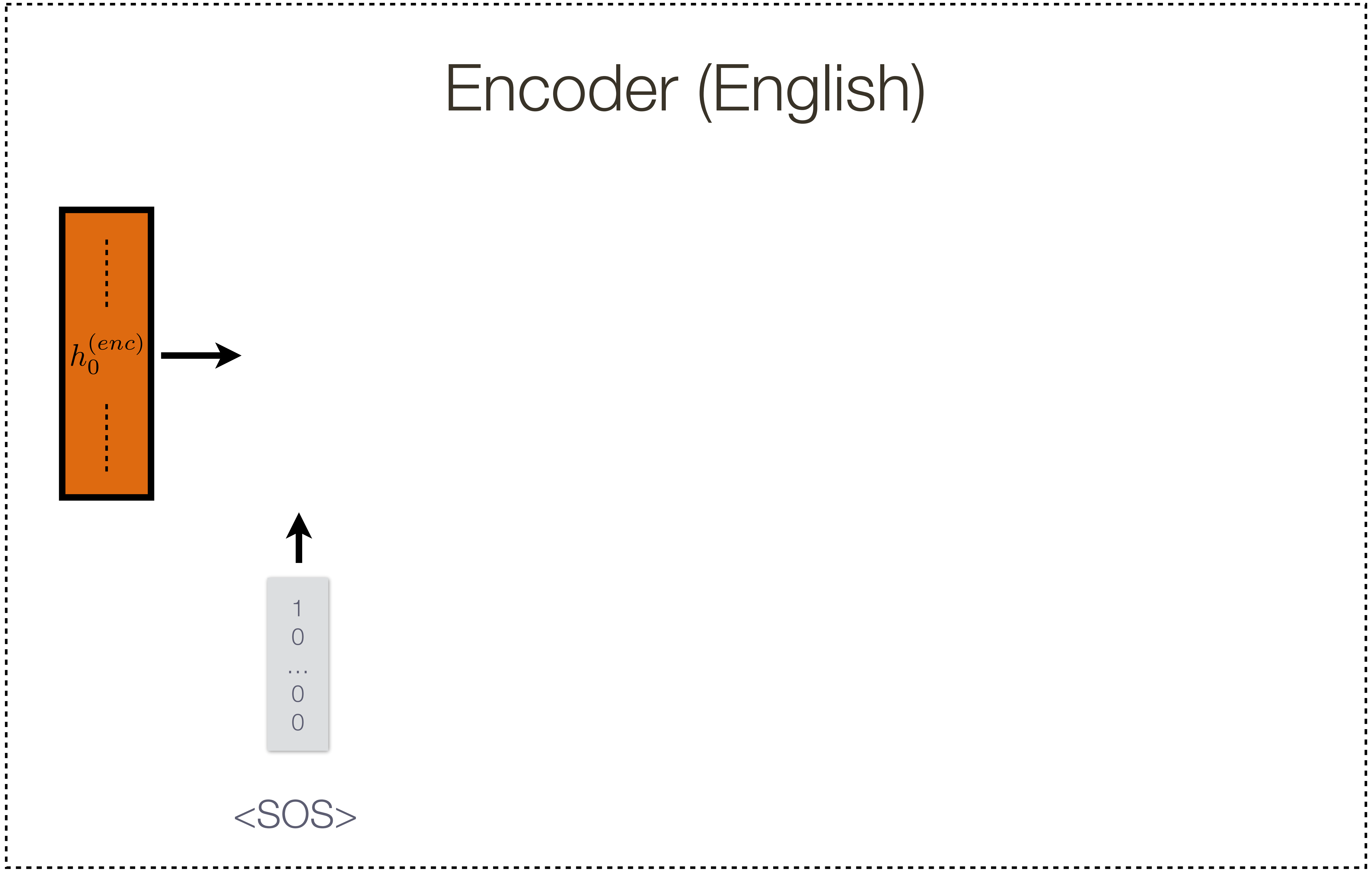


$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

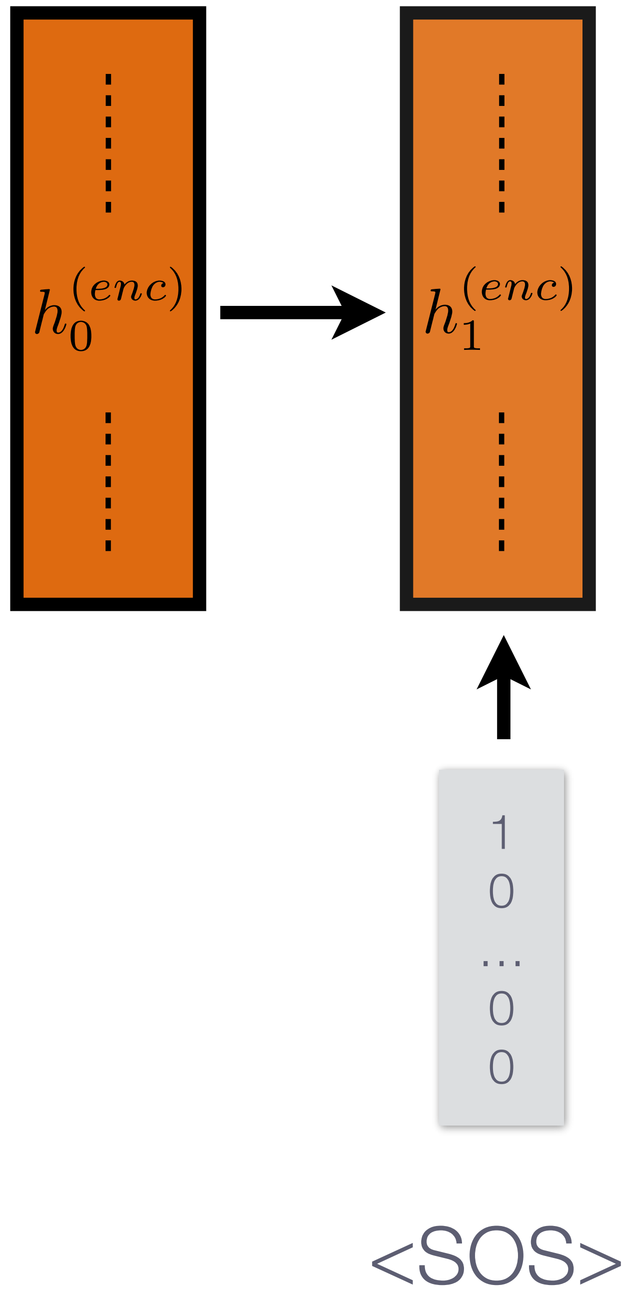
Normalize the weights to sum to 1

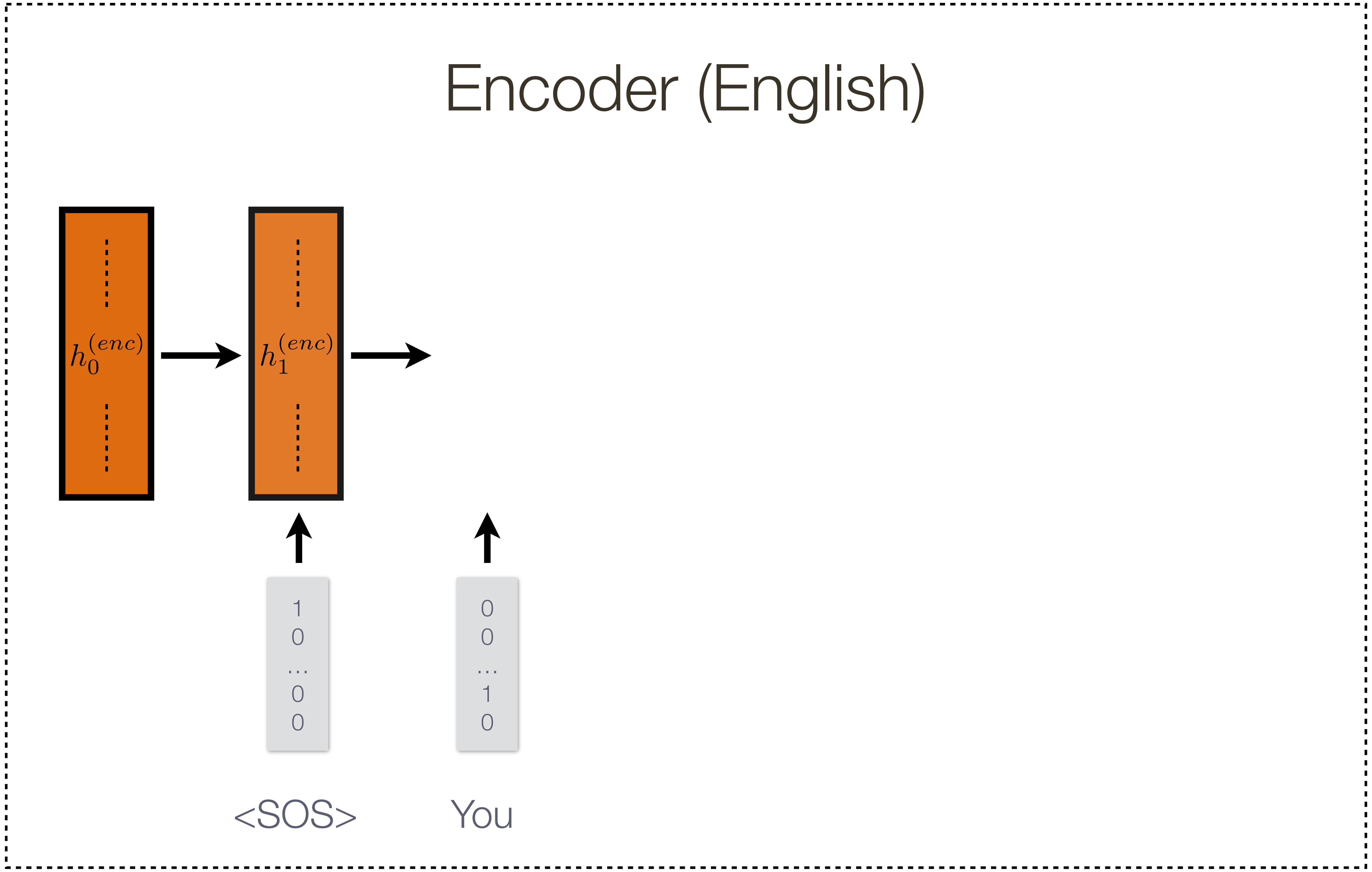
$$\mathbf{c}_t = \sum_i \alpha_{i,t} \mathbf{h}_i^{(enc)}$$

Form a context vector that would simply be added to the standard decoder input

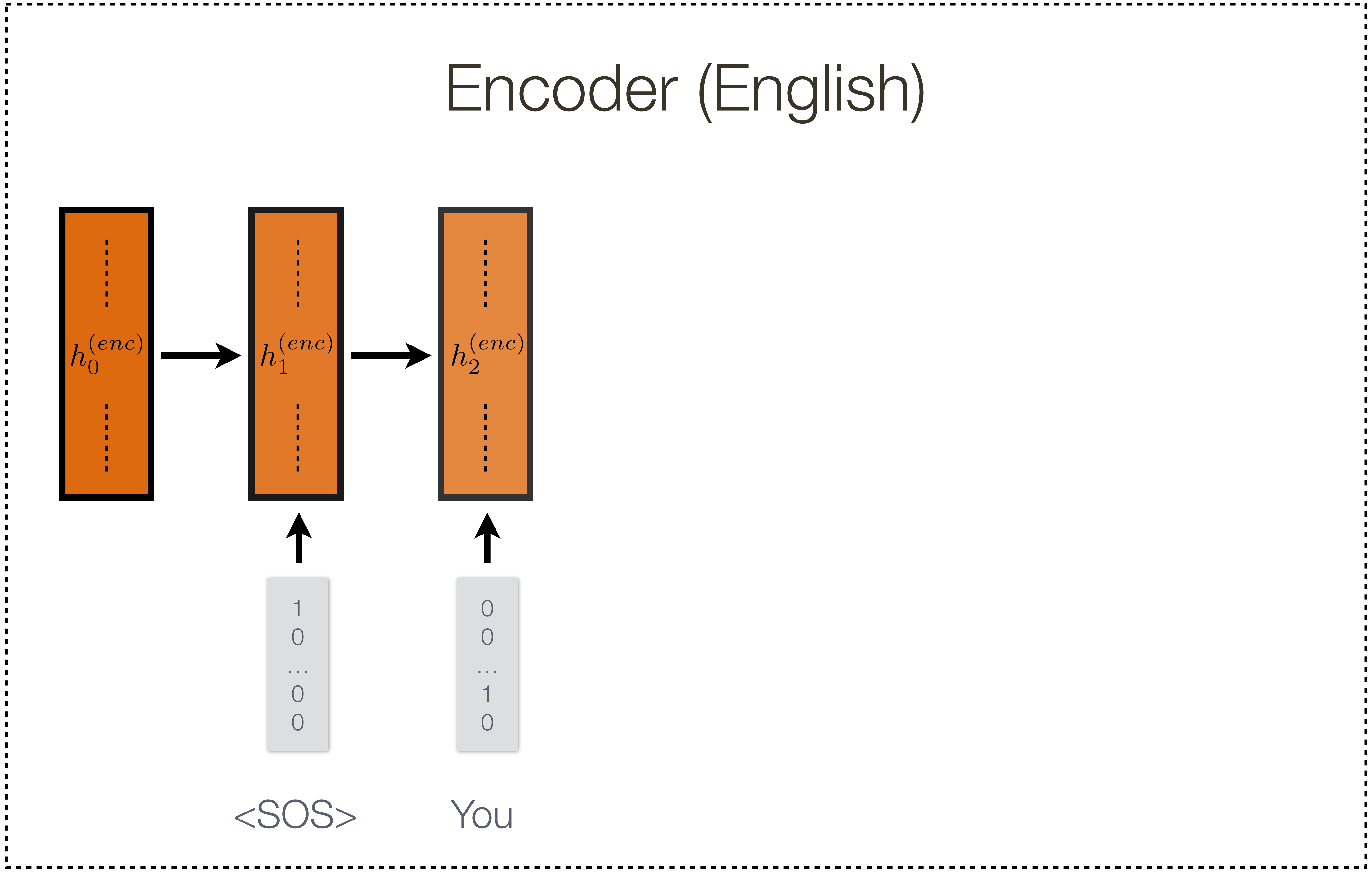

Encoder (English)
$h_0^{(enc)}$
1
0
...
0
0
<SOS>

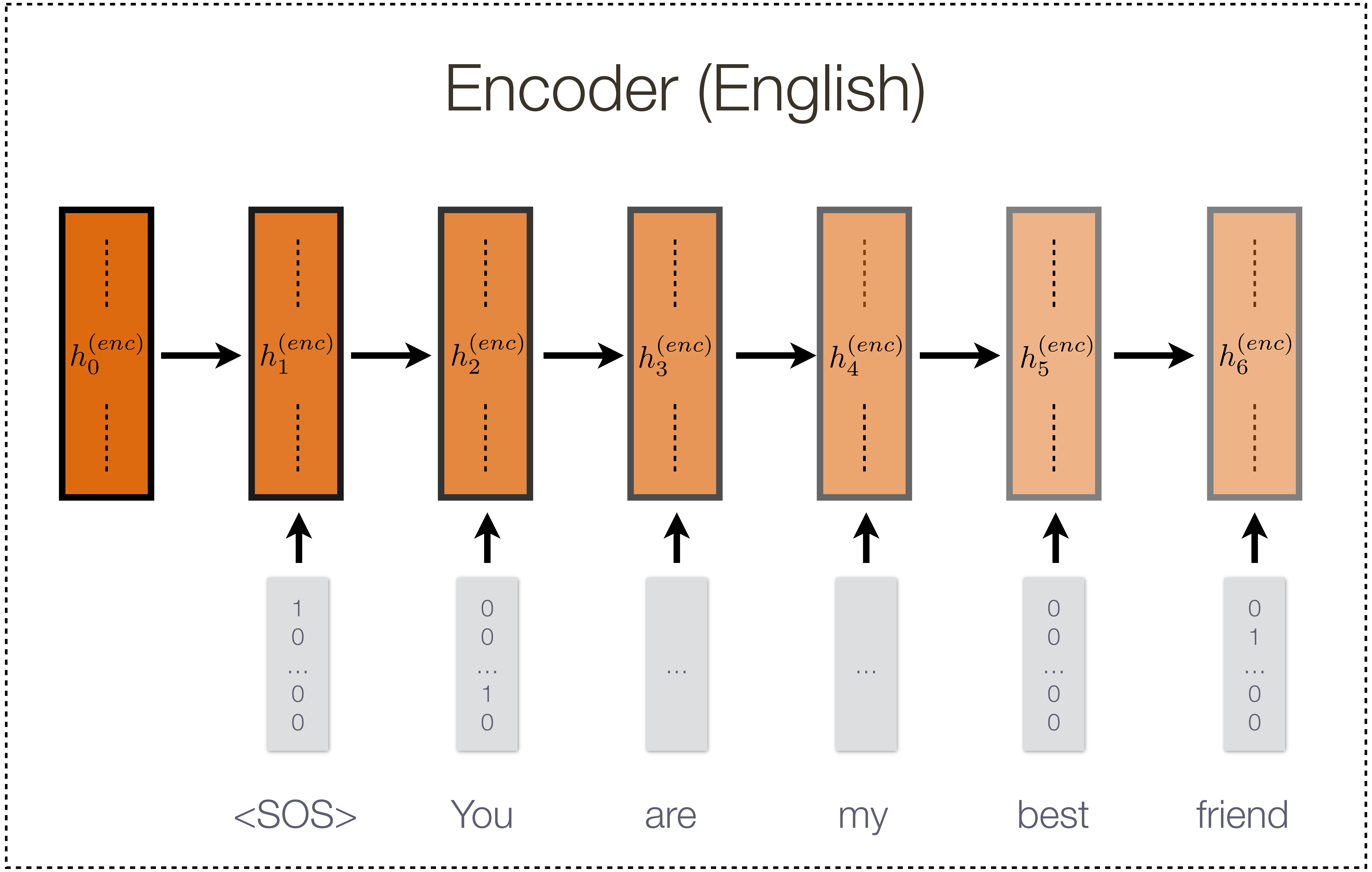
Encoder (English)

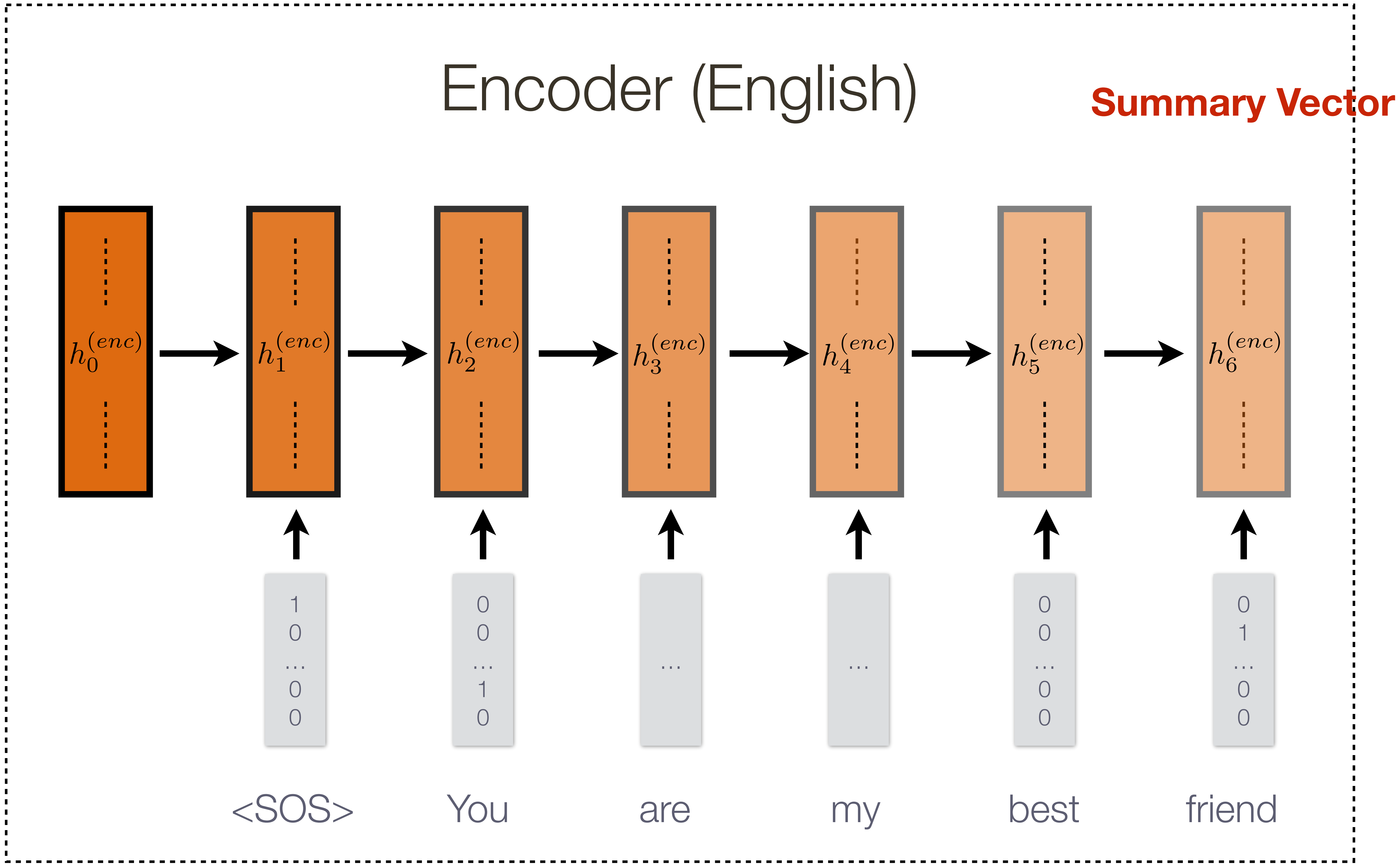




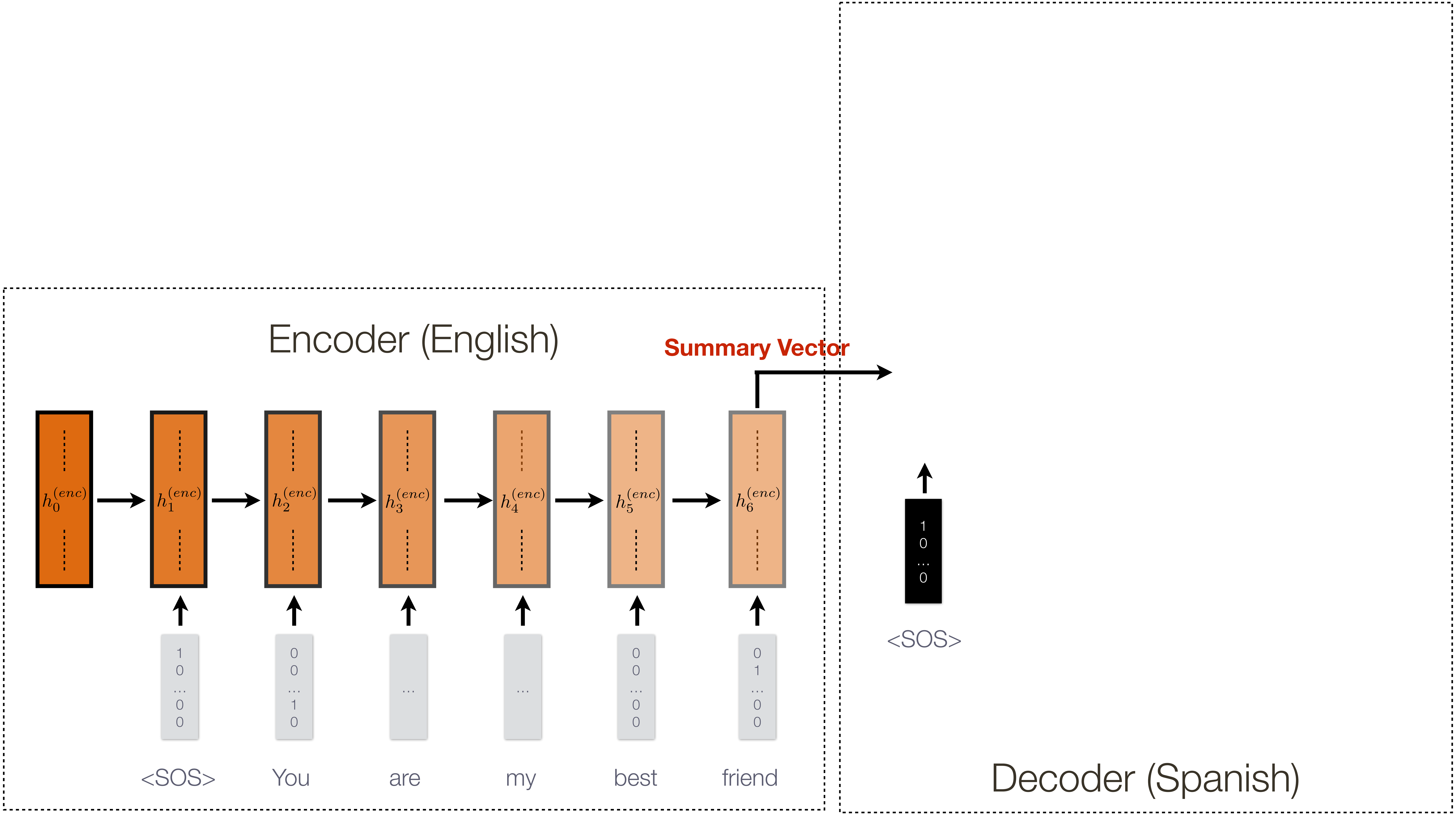
Encoder (English)
$h_0^{(enc)}$
$h_1^{(enc)}$
1
0
...
0
0
0
0
...
1
0
<SOS>
You


Encoder (English)
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
1
0
...
0
0
0
0
...
1
0
<SOS>
You

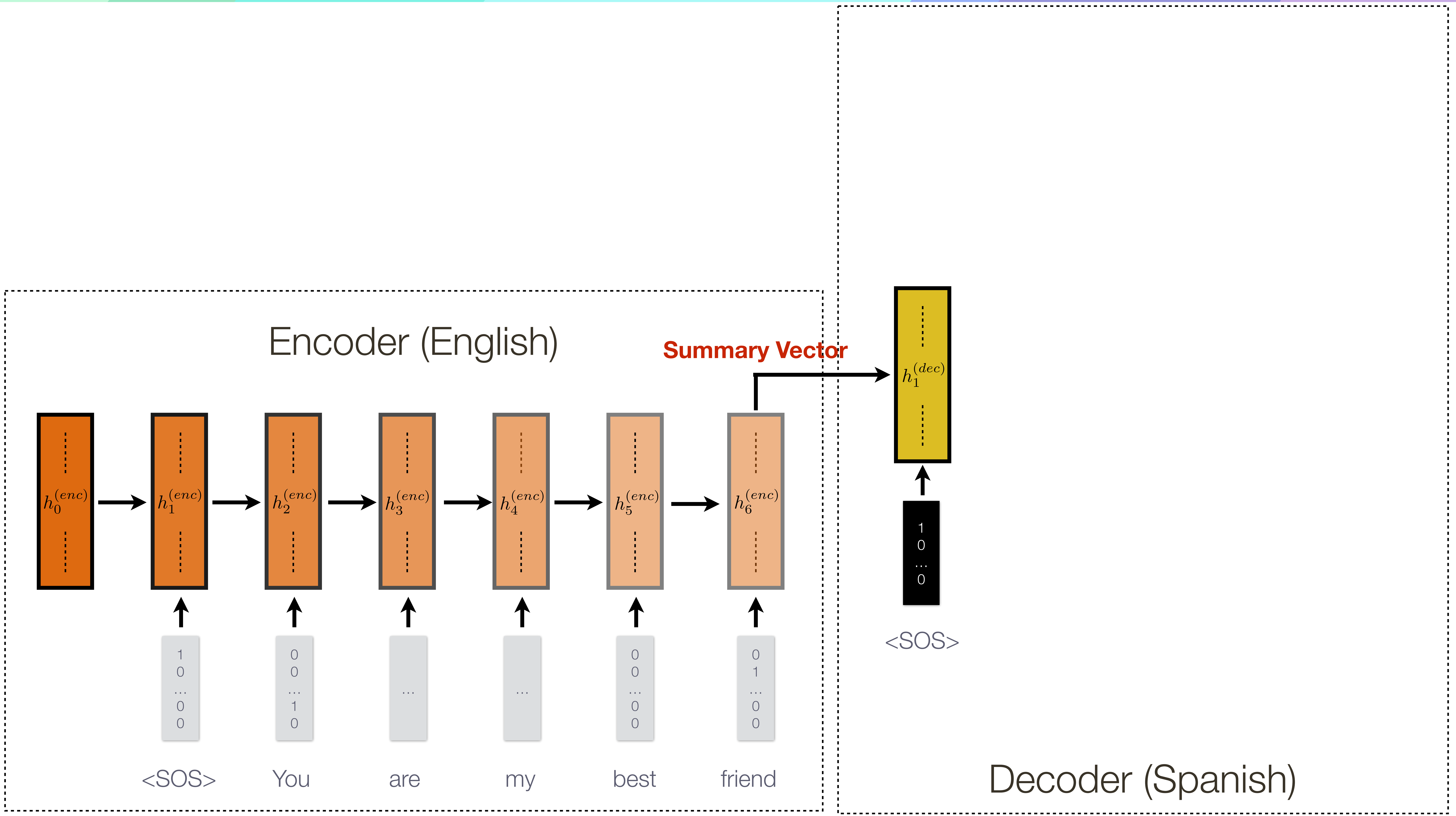

Encoder (English)
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend

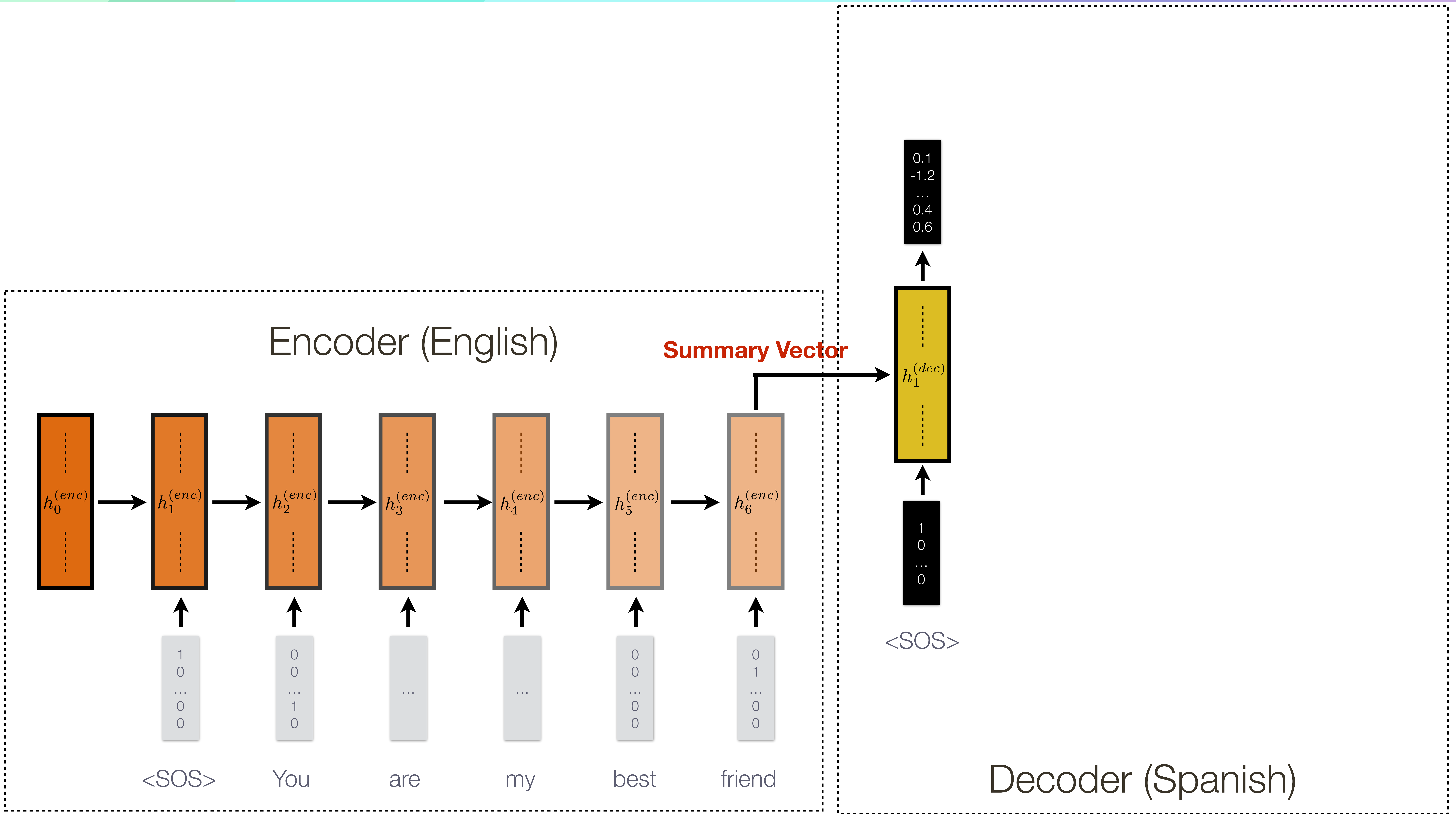


Encoder (English)
Summary Vector
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend

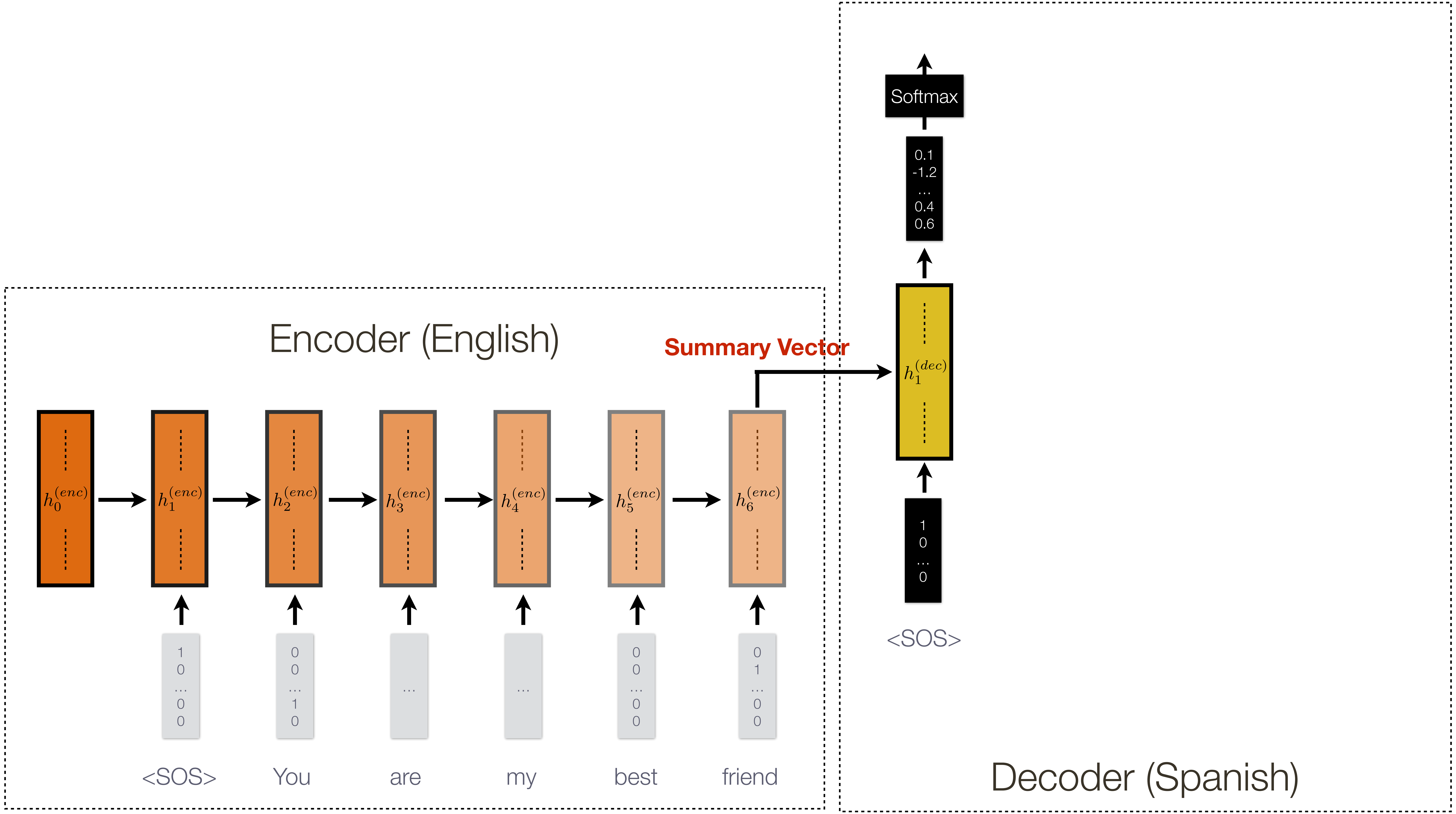


Encoder (English)
Summary Vector
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend
1 0 … 0
<SOS>
Decoder (Spanish)

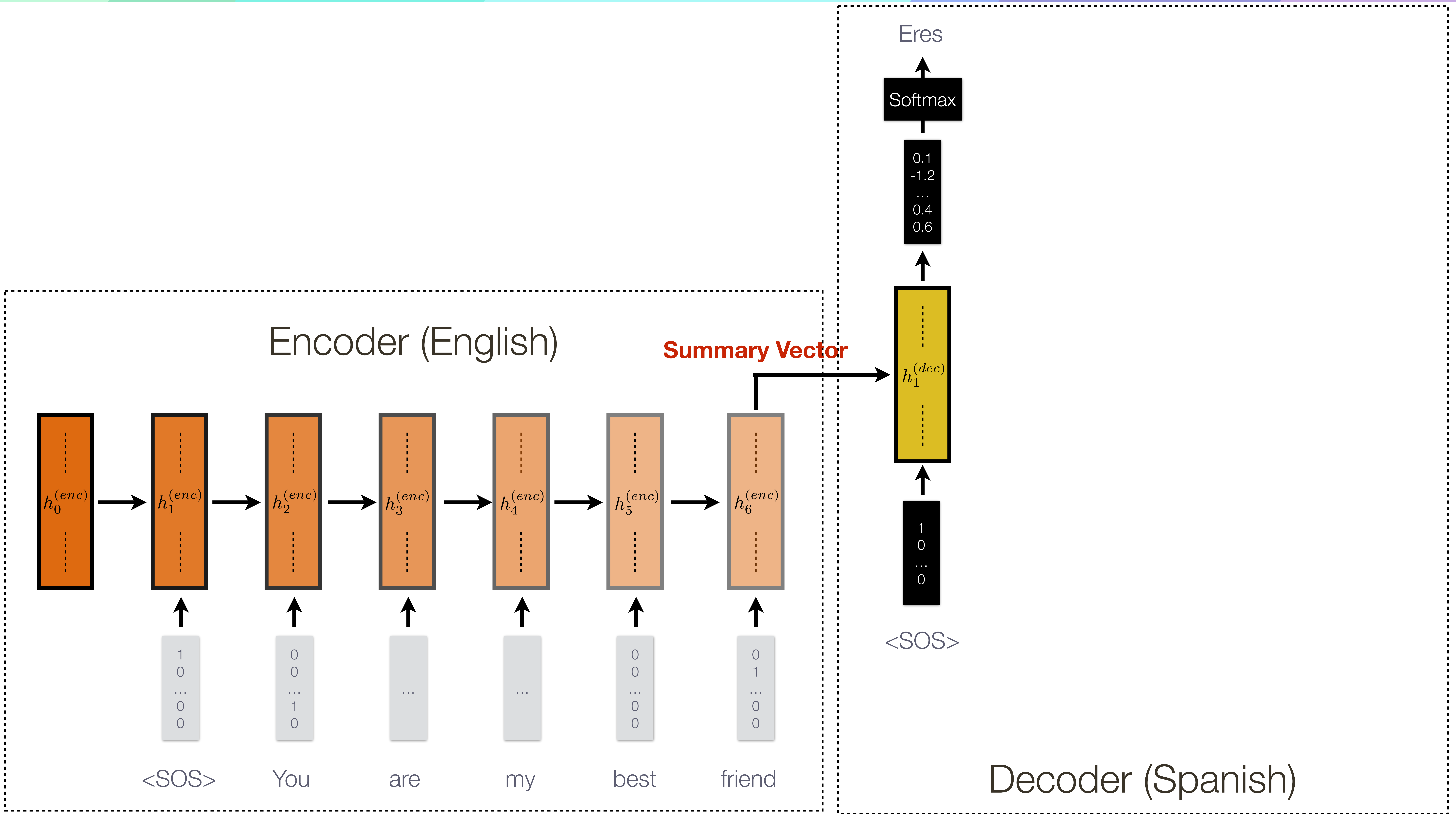

Encoder (English)
Summary Vector
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
$h_1^{(dec)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
1 0 … 0
<SOS>
You
are
my
best
friend
<SOS>
Decoder (Spanish)

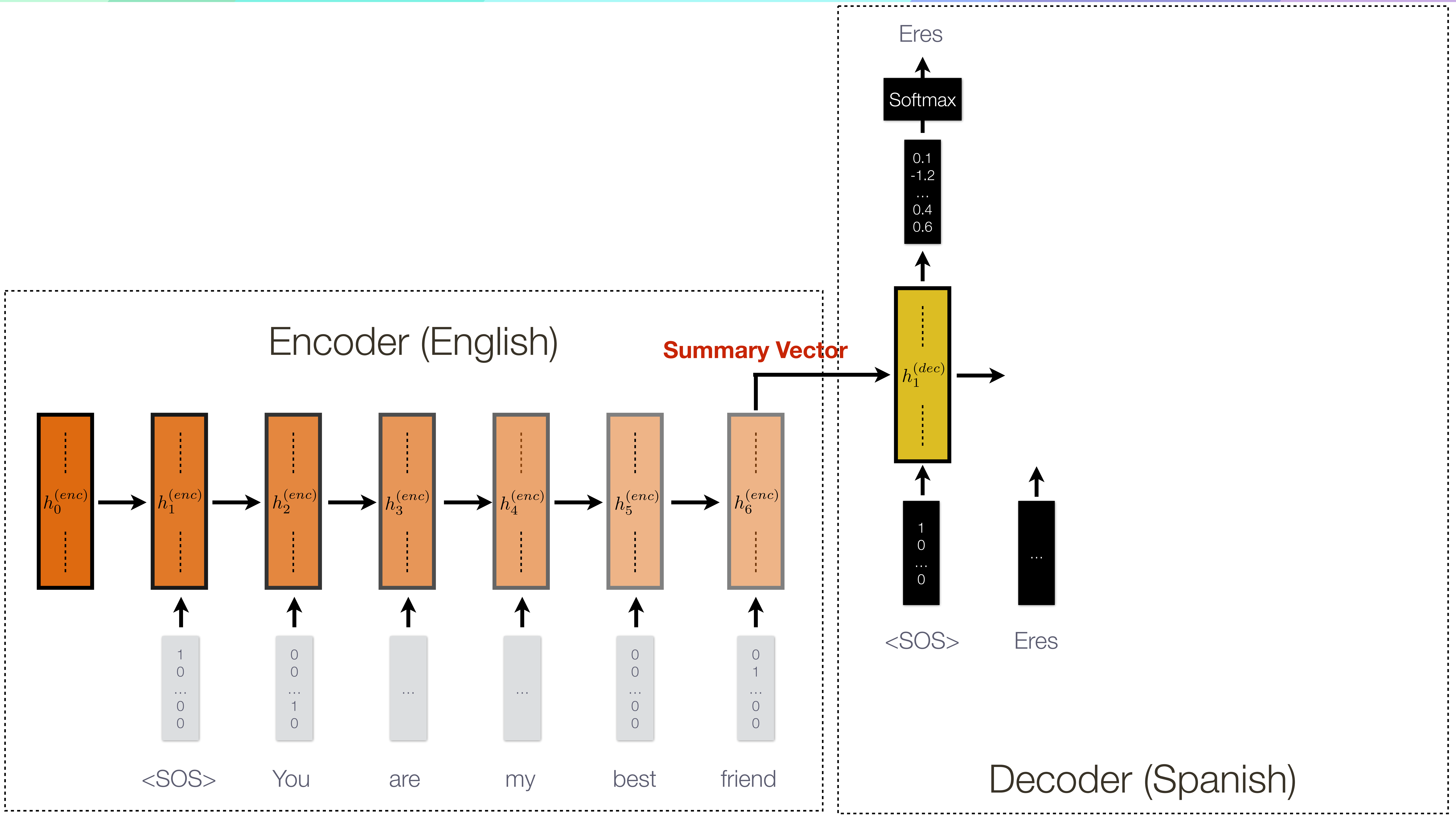


Encoder (English)
Summary Vector
Decoder (Spanish)
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
$h_1^{(dec)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
1 0 … 0
0.1 -1.2 … 0.4 0.6
<SOS>
You
are
my
best
friend
<SOS>

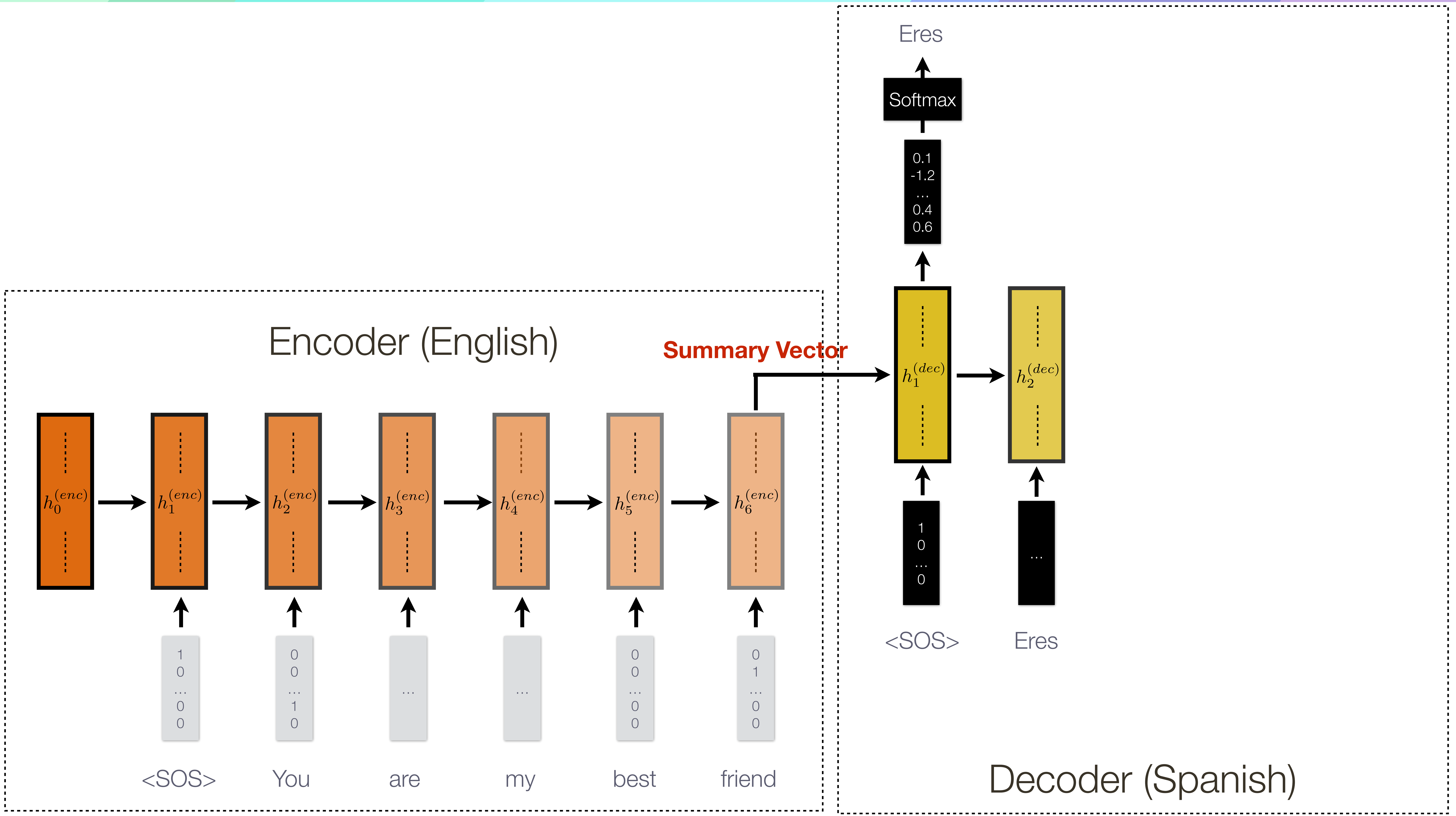


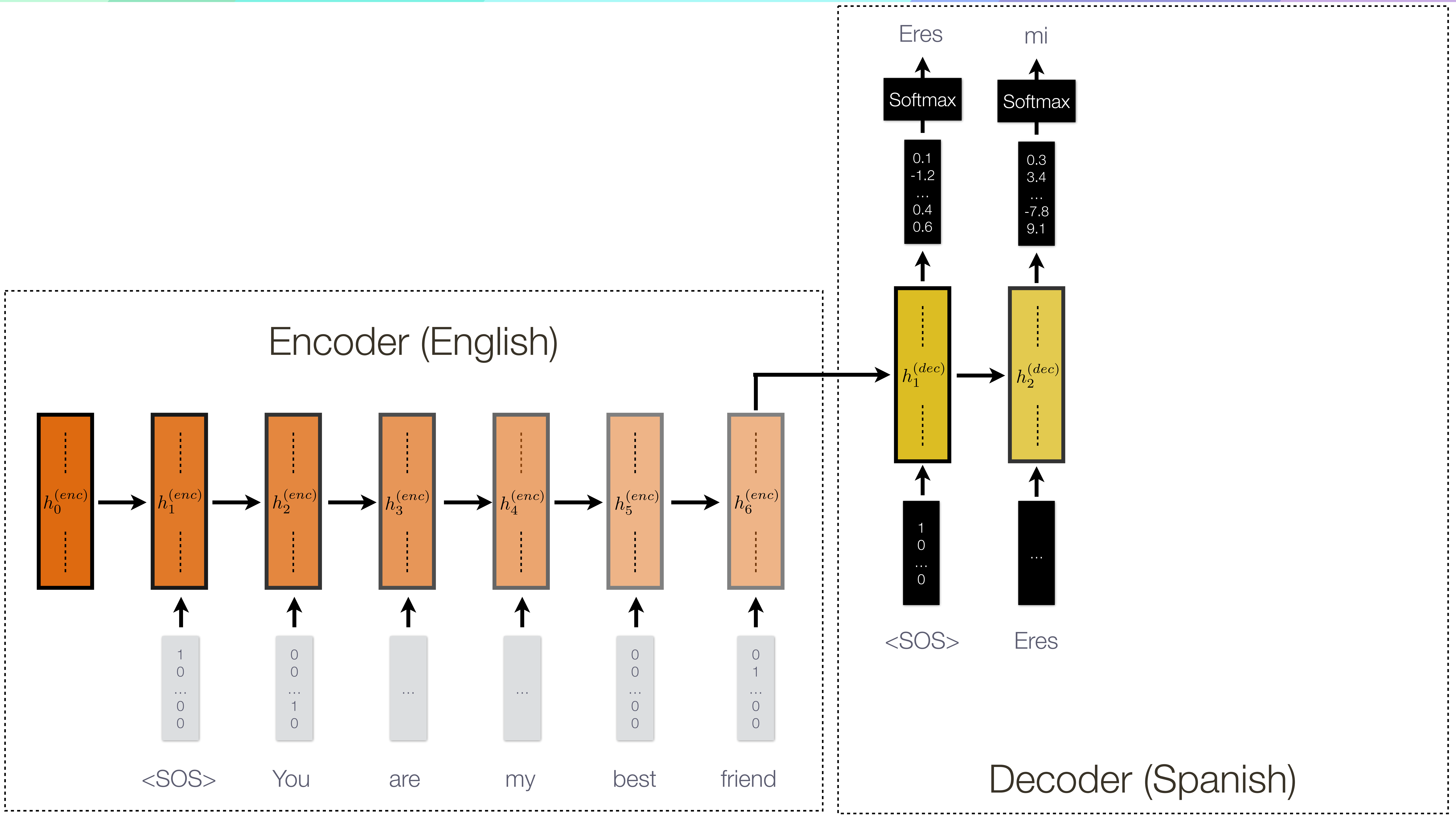
Encoder (English)
Summary Vector
Decoder (Spanish)
Softmax
0.1
-1.2
…
0.4
0.6
$h_1^{(dec)}$
1
0
…
0
<SOS>
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend


Eres
Softmax
0.1
-1.2
…
0.4
0.6
Encoder (English)
Summary Vector
$h_1^{(dec)}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0
<SOS>
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend
Decoder (Spanish)

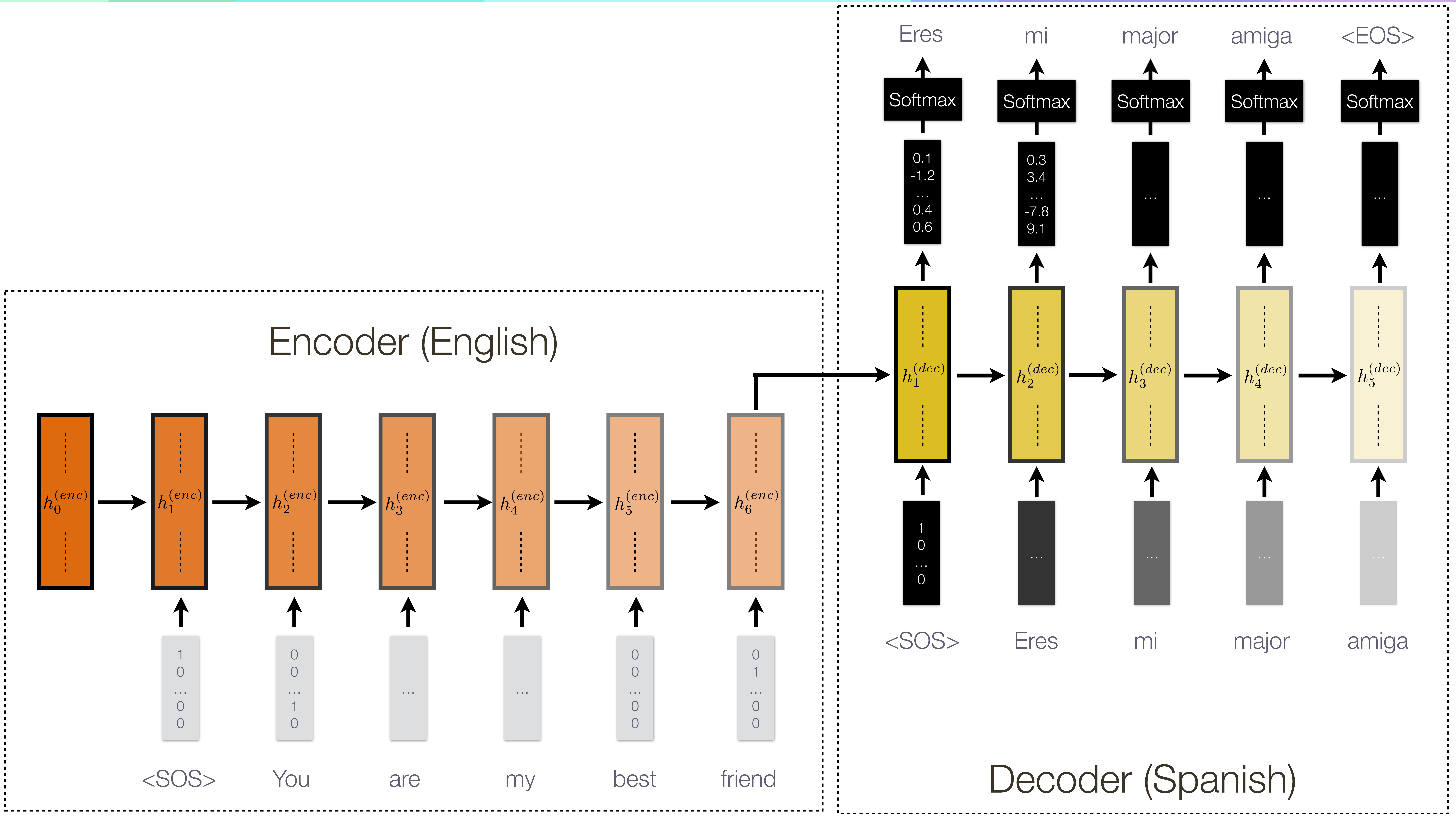


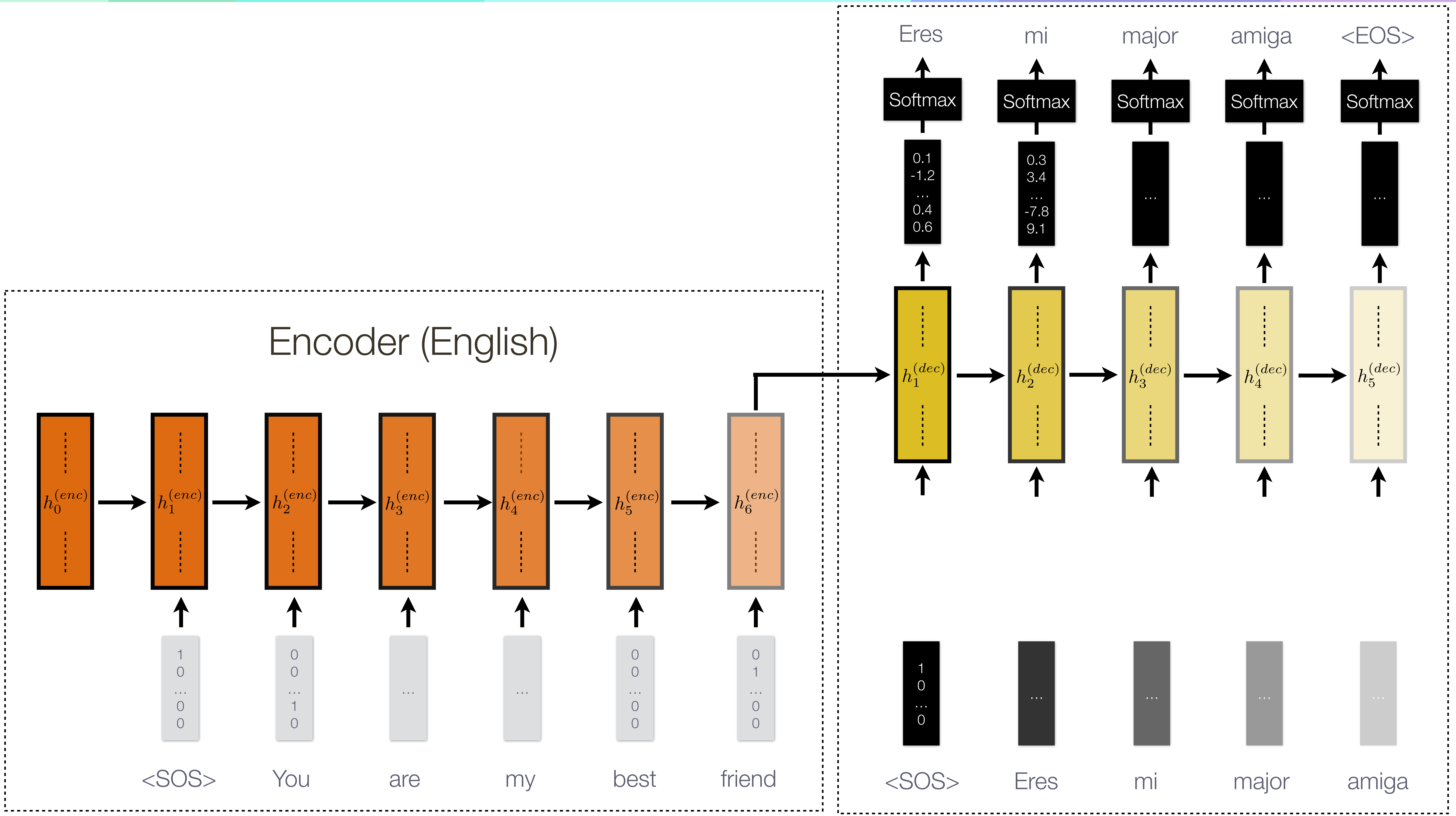
Eres
Softmax
0.1
-1.2
…
0.4
0.6
Encoder (English)
Summary Vector
$h_1^{(dec)}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
<SOS>
You
are
my
best
friend
<SOS>
Eres
Decoder (Spanish)


Eres
Softmax
0.1
-1.2
…
0.4
0.6
Encoder (English)
Summary Vector
$h_1^{(dec)}$
$h_2^{(dec)}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
<SOS>
Eres
<SOS>
You
are
my
best
friend
Decoder (Spanish)

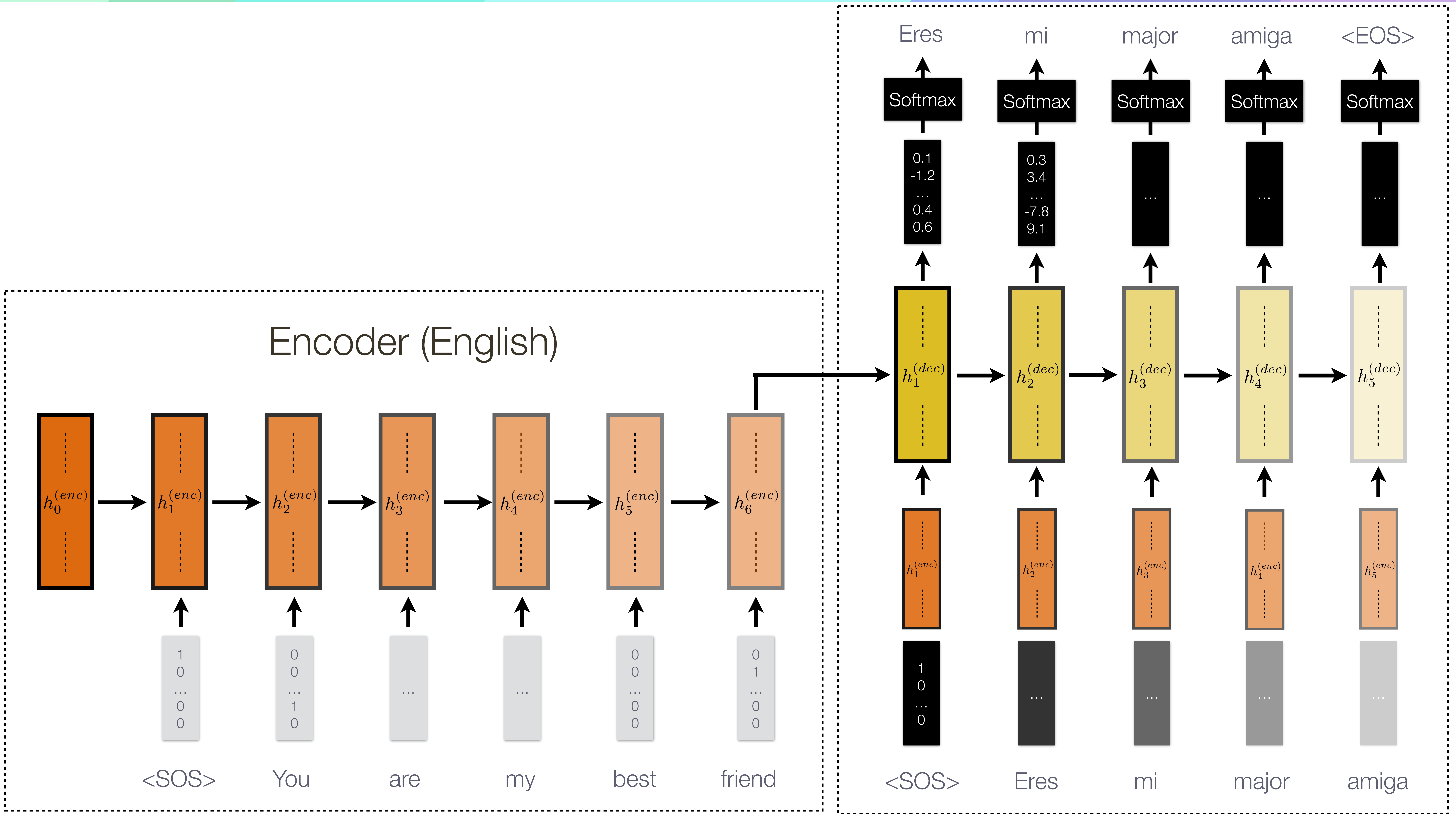


Encoder (English)
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend
Decoder (Spanish)
Eres
mi
Softmax
Softmax
0.1 -1.2 … 0.4 0.6
0.3 3.4 … -7.8 9.1
$h_1^{(dec)}$
$h_2^{(dec)}$
1 0 … 0
…
<SOS>
Eres

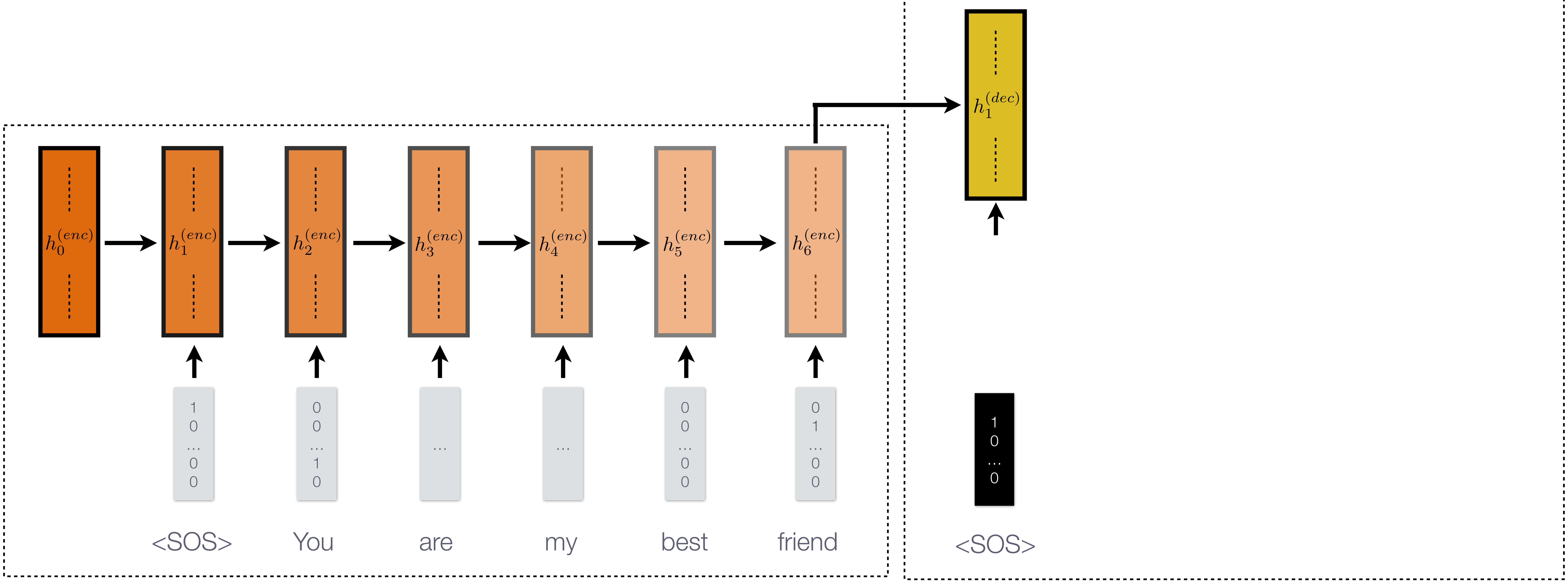

Encoder (English)
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend
Decoder (Spanish)
Eres
mi
major
amiga
<EOS>
Softmax
Softmax
Softmax
Softmax
Softmax
0.1 -1.2 … 0.4 0.6
0.3 3.4 … -7.8 9.1
…
…
…
$h_1^{(dec)}$
$h_2^{(dec)}$
$h_3^{(dec)}$
$h_4^{(dec)}$
$h_5^{(dec)}$
1 0 … 0
…
…
…
…
<SOS>
Eres
mi
major
amiga



Encoder (English)
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend
Eres
mi
major
amiga
<EOS>
Softmax
Softmax
Softmax
Softmax
Softmax
0.1 -1.2 … 0.4 0.6
0.3 3.4 … -7.8 9.1
…
…
…
$h_1^{(dec)}$
$h_2^{(dec)}$
$h_3^{(dec)}$
$h_4^{(dec)}$
$h_5^{(dec)}$
1 0 … 0
…
…
…
…
<SOS>
Eres
mi
major
amiga

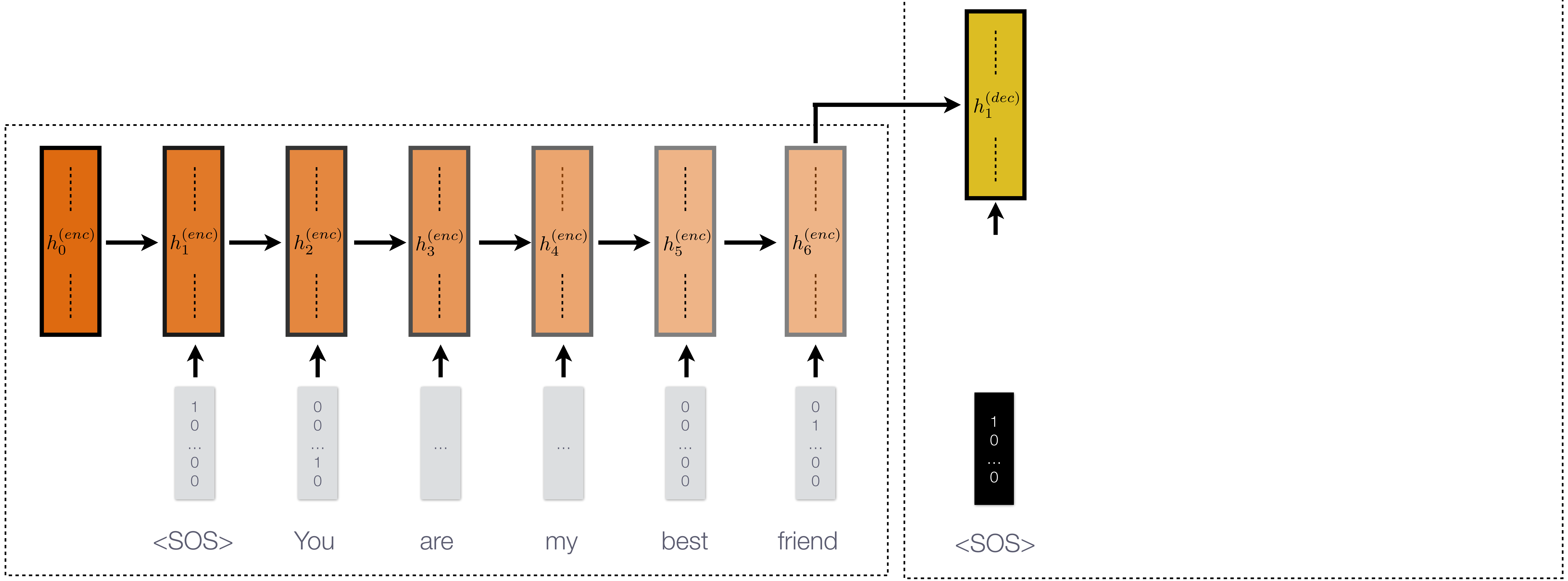

Encoder (English)
<SOS>
You
are
my
best
friend
Eres
mi
major
amiga
<EOS>
Softmax
0.1
-1.2
...
0.4
0.6
0.3
3.4
...
-7.8
9.1
<SOS>
Eres
mi
major
amiga

= Score(,   )

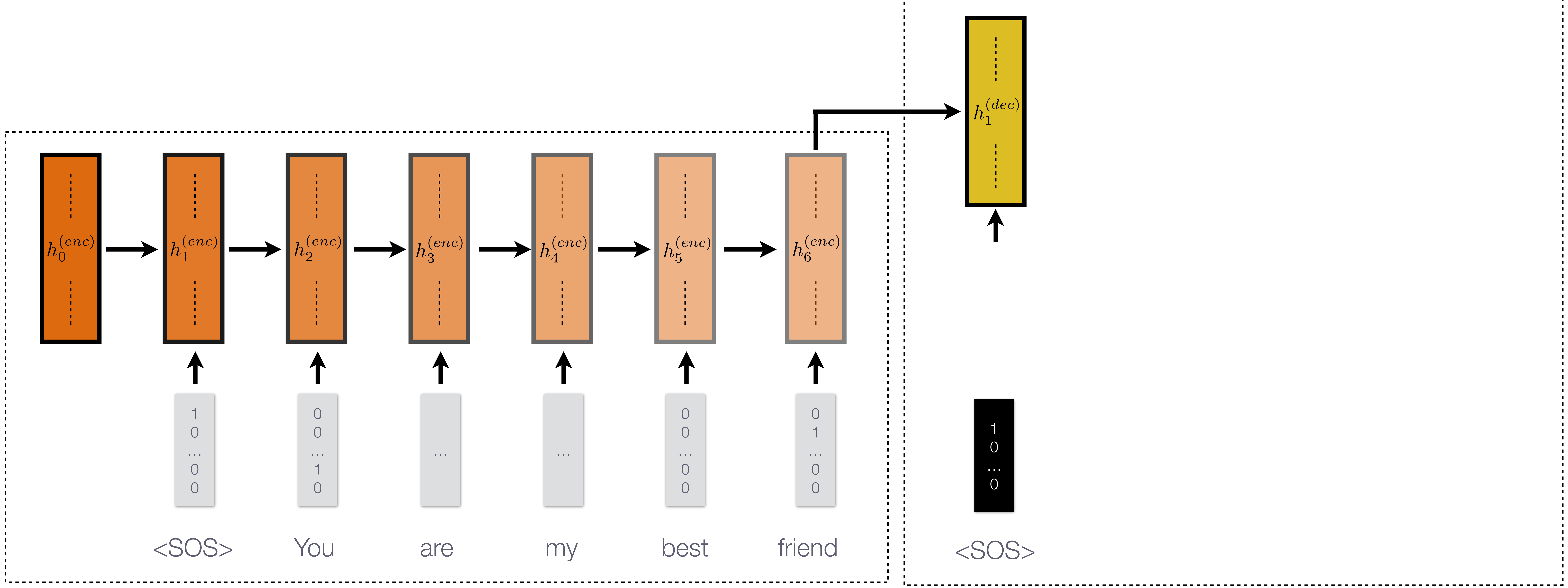


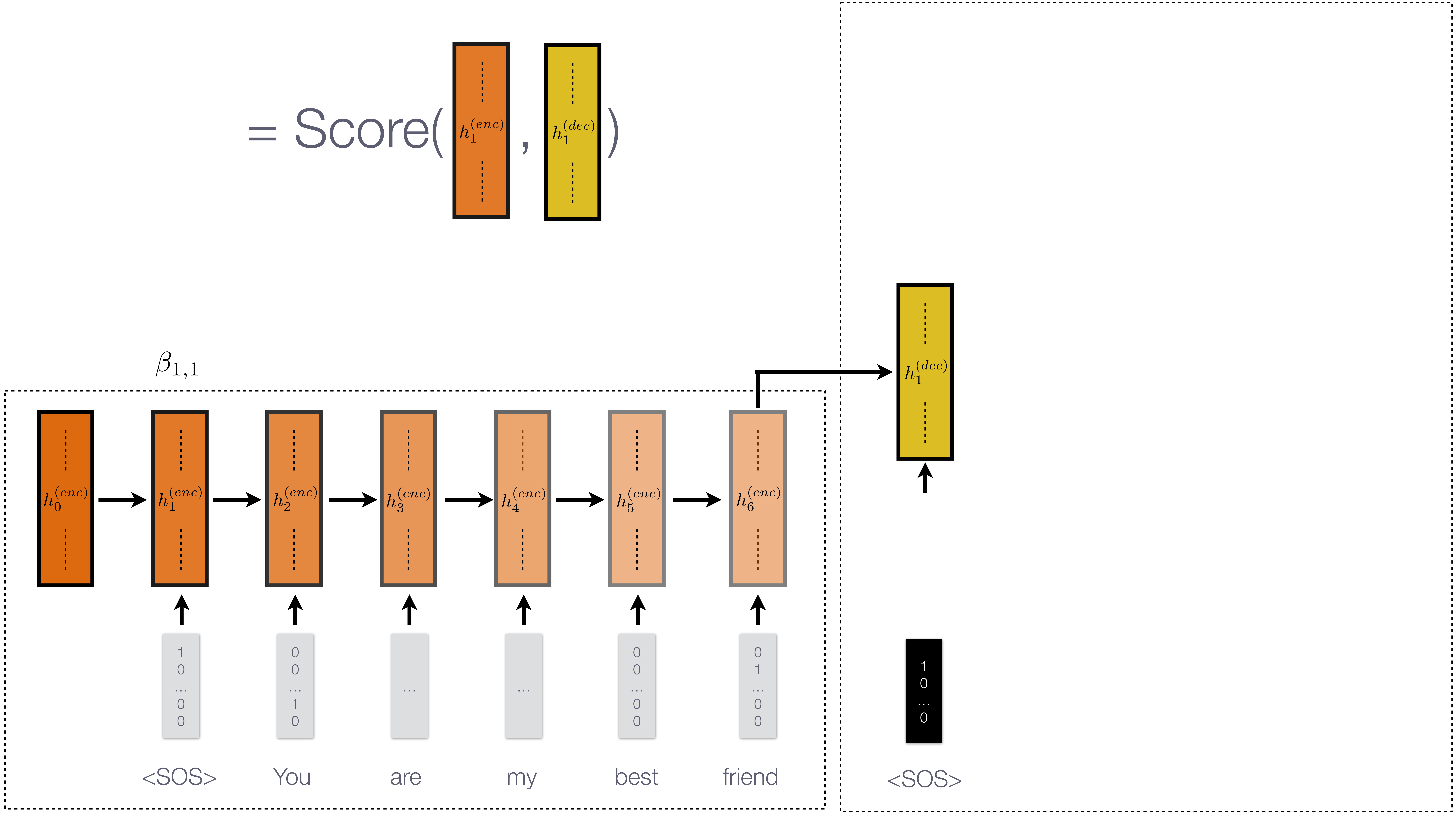
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
$h_1^{(dec)}$
<SOS>
You
are
my
best
friend
<SOS>

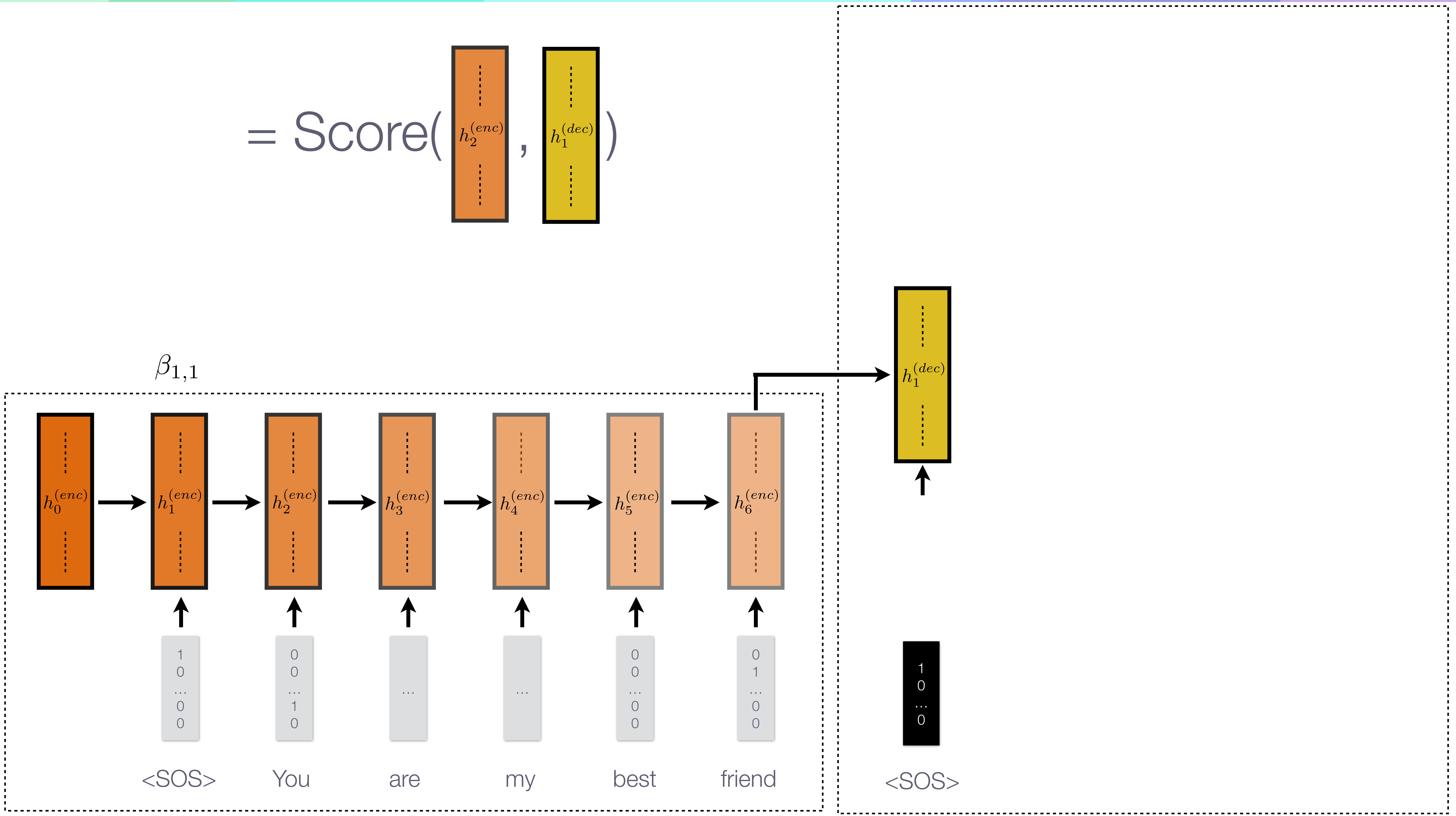
$$= \text{Score}\left(h_1^{(enc)}, h_1^{(dec)}\right)$$



$$\beta_{1,1} = \text{Score}\left(h_1^{(enc)}, h_1^{(dec)}\right)$$

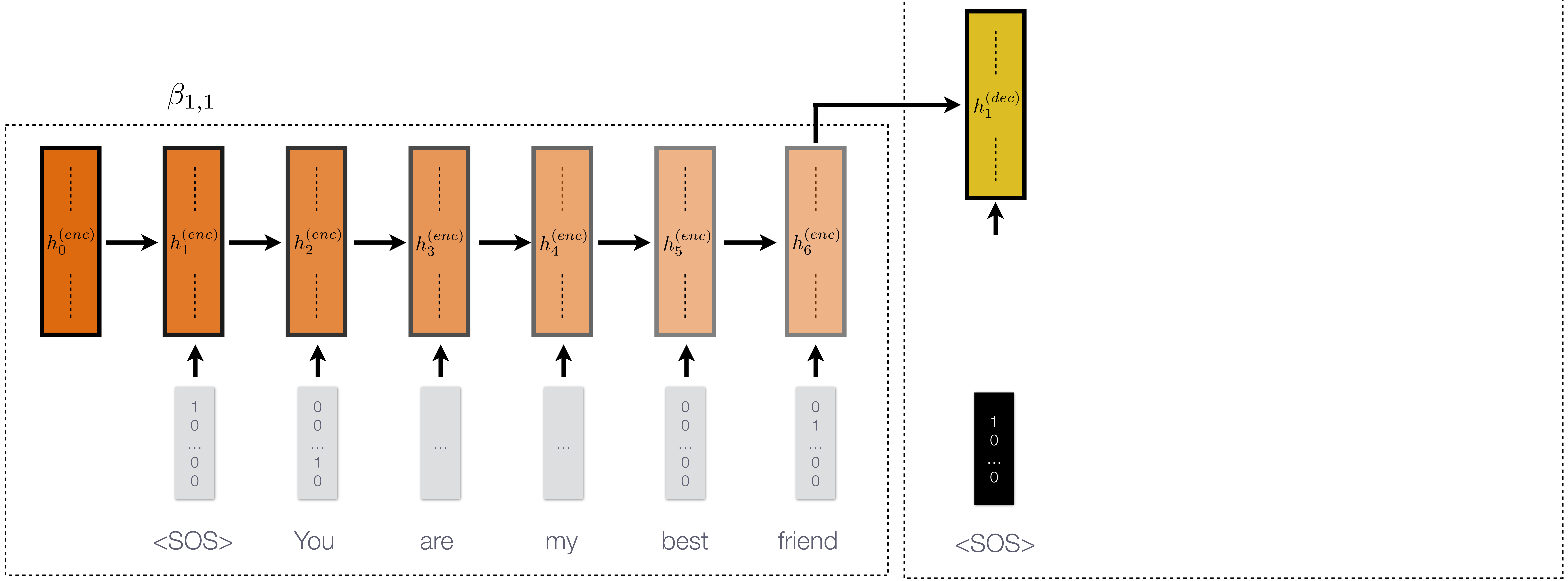


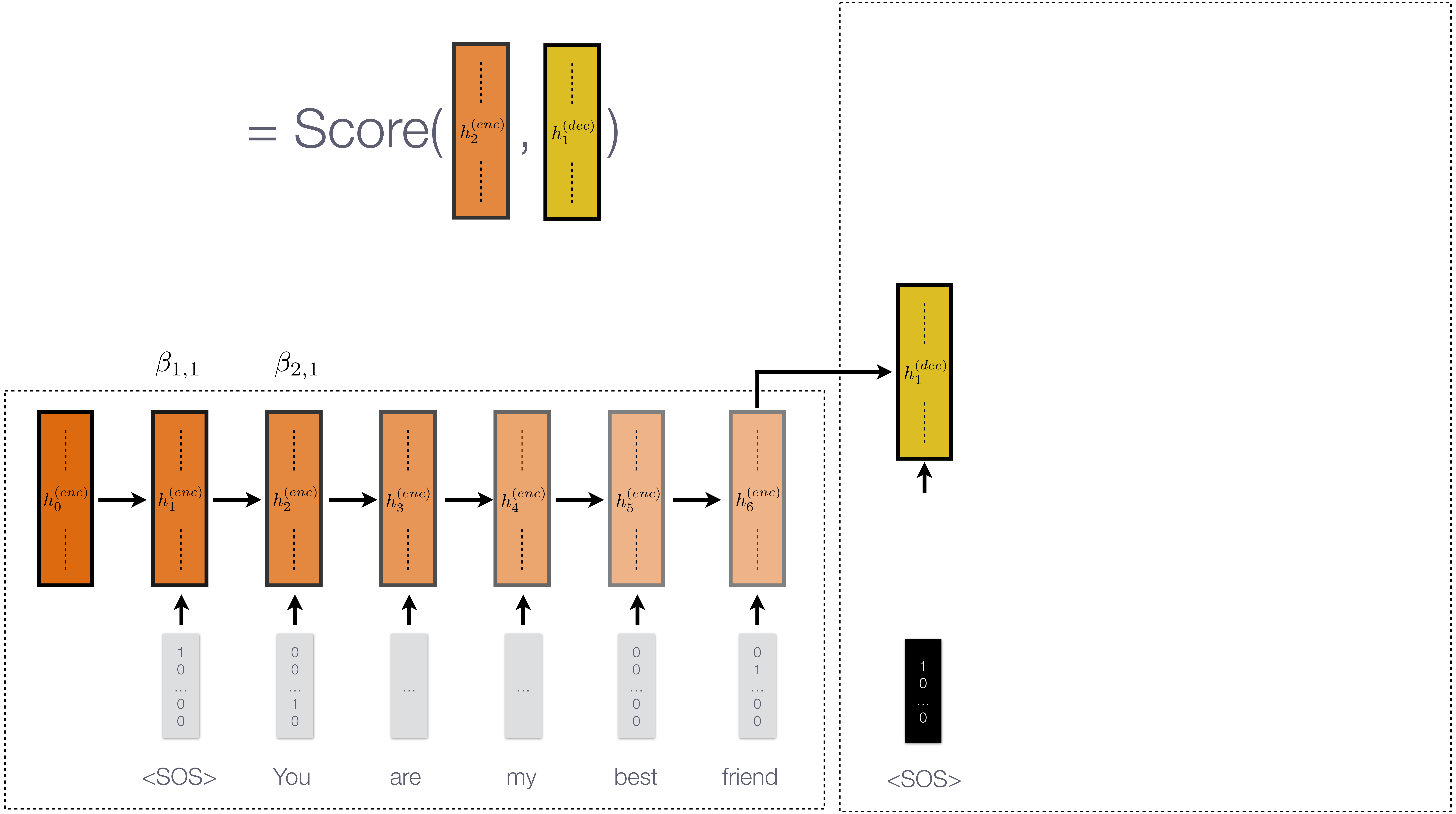

= Score($h_1^{(enc)}$, $h_1^{(dec)}$)
$\beta_{1,1}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
$h_1^{(dec)}$
<SOS>
You
are
my
best
friend
<SOS>



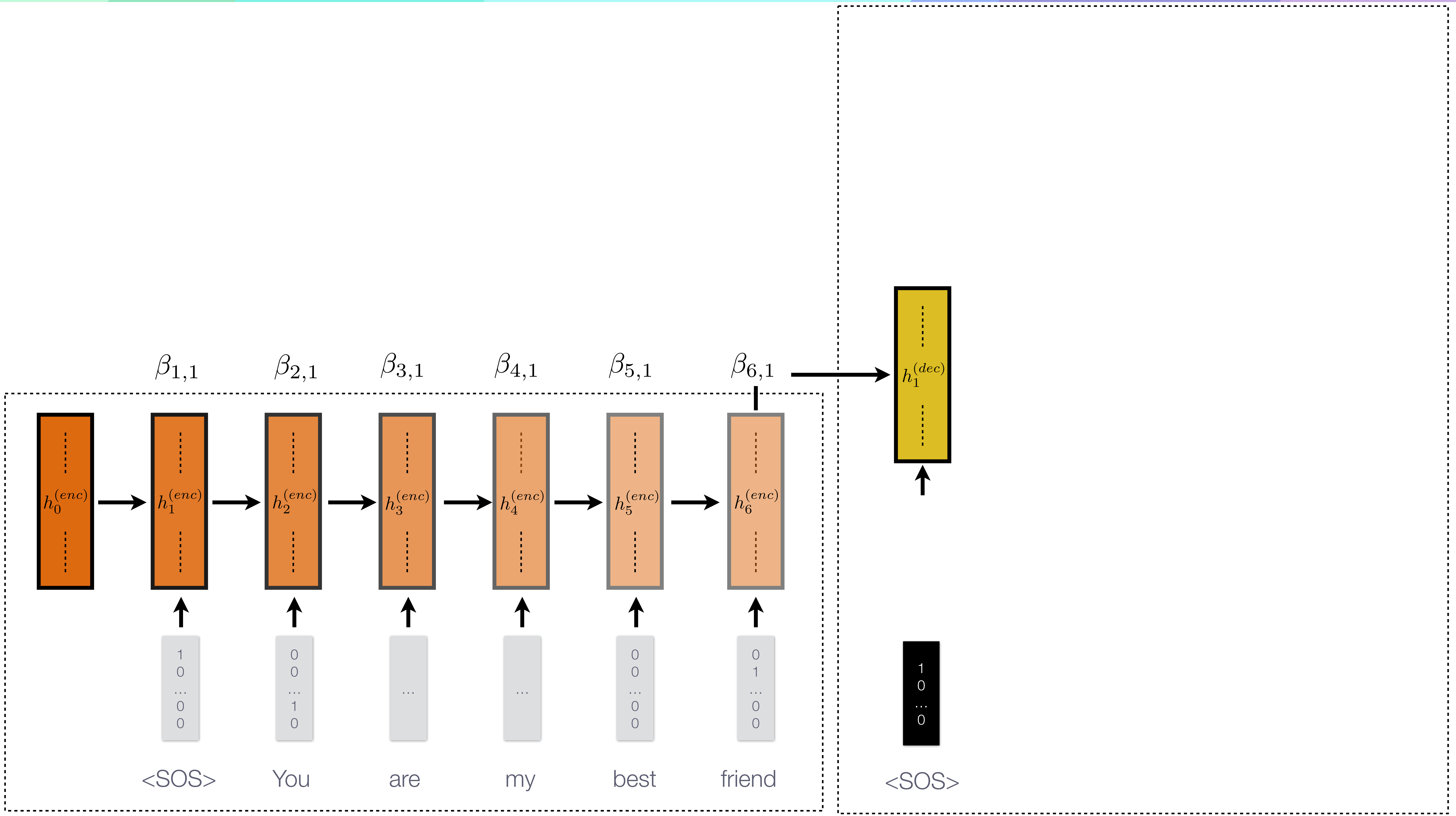
= Score($h_2^{(enc)}$, $h_1^{(dec)}$)
$\beta_{1,1}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
$h_1^{(dec)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
1 0 … 0
<SOS>
You
are
my
best
friend
<SOS>

$$\beta_{2,1} = \text{Score}(h_2^{(enc)}, h_1^{(dec)})$$

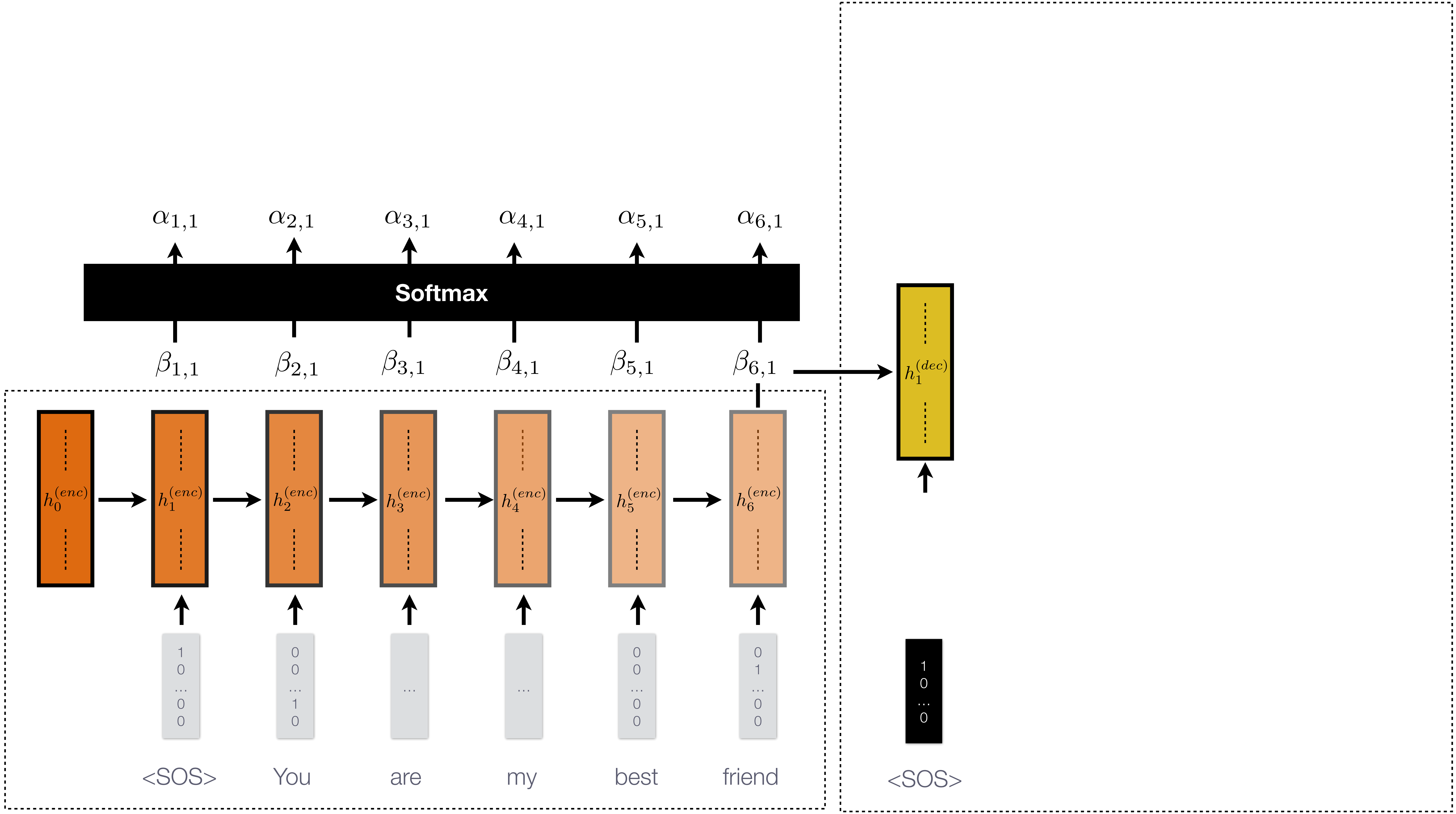




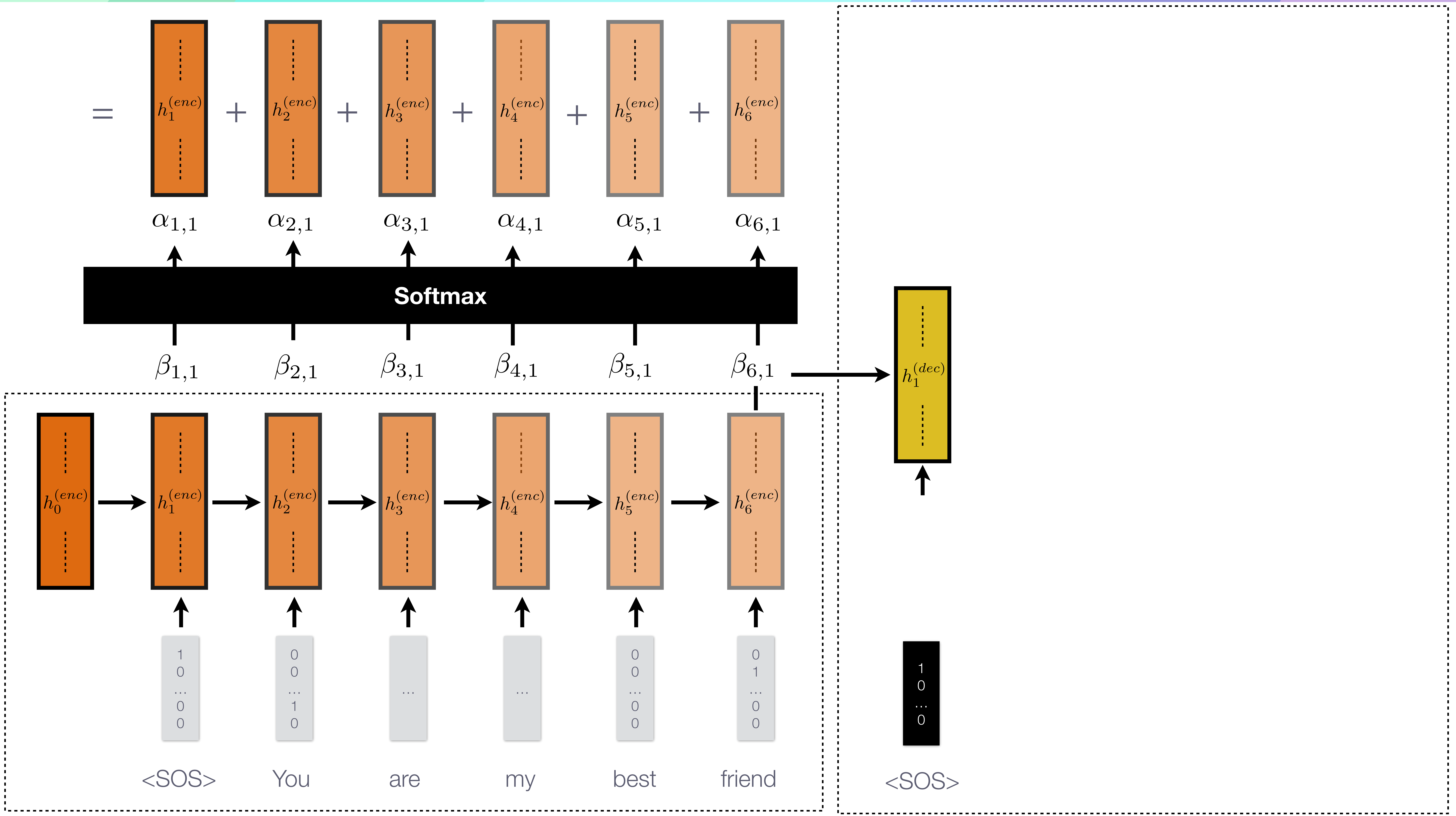
= Score($h_2^{(enc)}$, $h_1^{(dec)}$)
$\beta_{1,1}$
$\beta_{2,1}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
$h_1^{(dec)}$
<SOS>
You
are
my
best
friend
<SOS>



$\beta_{1,1}$ $\beta_{2,1}$ $\beta_{3,1}$ $\beta_{4,1}$ $\beta_{5,1}$ $\beta_{6,1}$
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
$h_1^{(dec)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
1 0 … 0
<SOS>
You
are
my
best
friend
<SOS>



$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$ $\alpha_{5,1}$ $\alpha_{6,1}$
Softmax
$\beta_{1,1}$ $\beta_{2,1}$ $\beta_{3,1}$ $\beta_{4,1}$ $\beta_{5,1}$ $\beta_{6,1}$
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
$h_1^{(dec)}$
<SOS>
You
are
my
best
friend
<SOS>



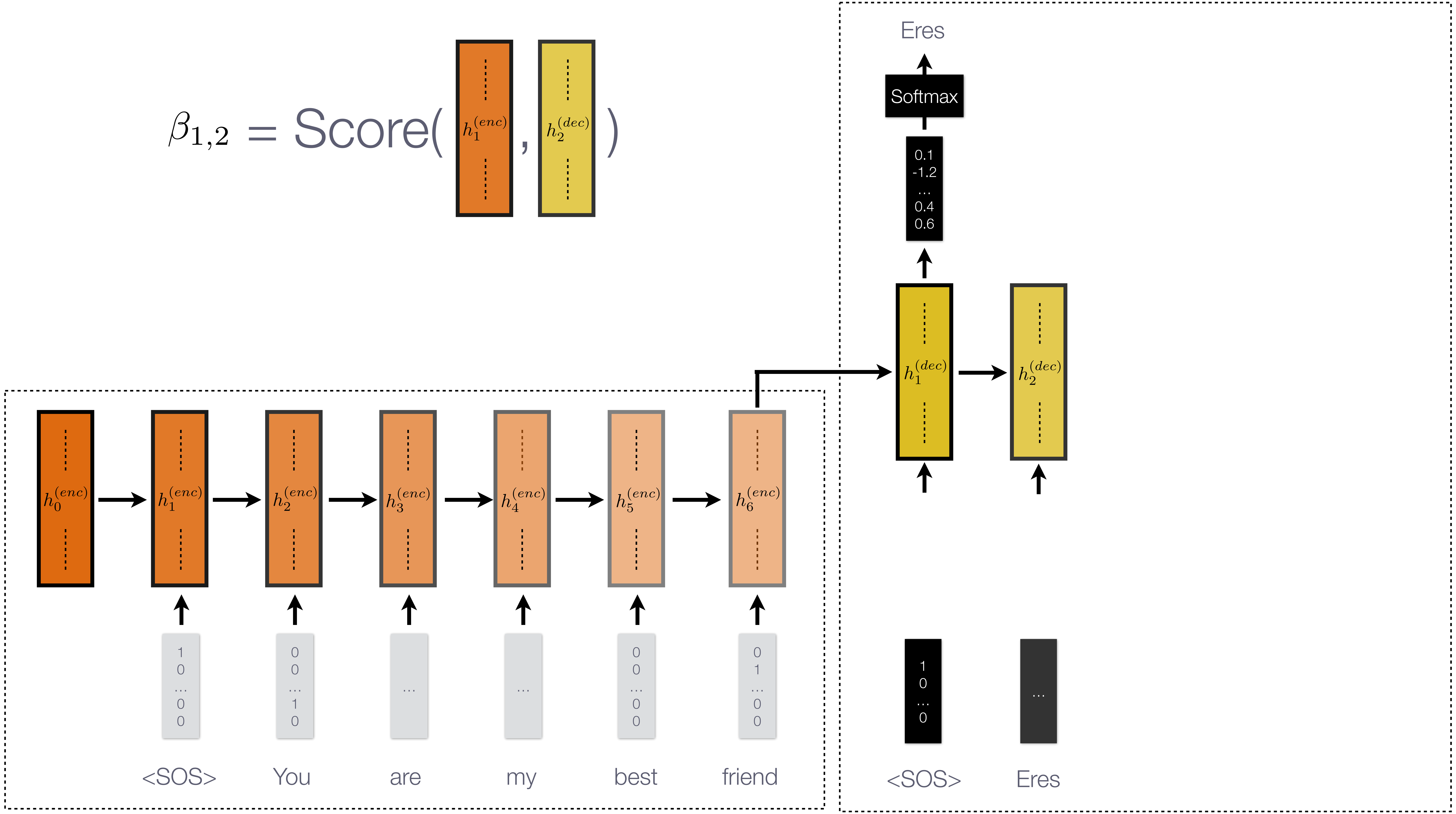
=
$h_1^{(enc)}$ + $h_2^{(enc)}$ + $h_3^{(enc)}$ + $h_4^{(enc)}$ + $h_5^{(enc)}$ + $h_6^{(enc)}$
$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$ $\alpha_{5,1}$ $\alpha_{6,1}$
Softmax
$\beta_{1,1}$ $\beta_{2,1}$ $\beta_{3,1}$ $\beta_{4,1}$ $\beta_{5,1}$ $\beta_{6,1}$
$h_1^{(dec)}$
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
1 0 … 0
<SOS>
You
are
my
best
friend
<SOS>

Context Vector
$c_1 = h_1^{(enc)} + h_2^{(enc)} + h_3^{(enc)} + h_4^{(enc)} + h_5^{(enc)} + h_6^{(enc)}$
$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$ $\alpha_{5,1}$ $\alpha_{6,1}$
Softmax
$\beta_{1,1}$ $\beta_{2,1}$ $\beta_{3,1}$ $\beta_{4,1}$ $\beta_{5,1}$ $\beta_{6,1}$
$h_1^{(dec)}$
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
1 0 … 0
<SOS> You are my best friend
<SOS>

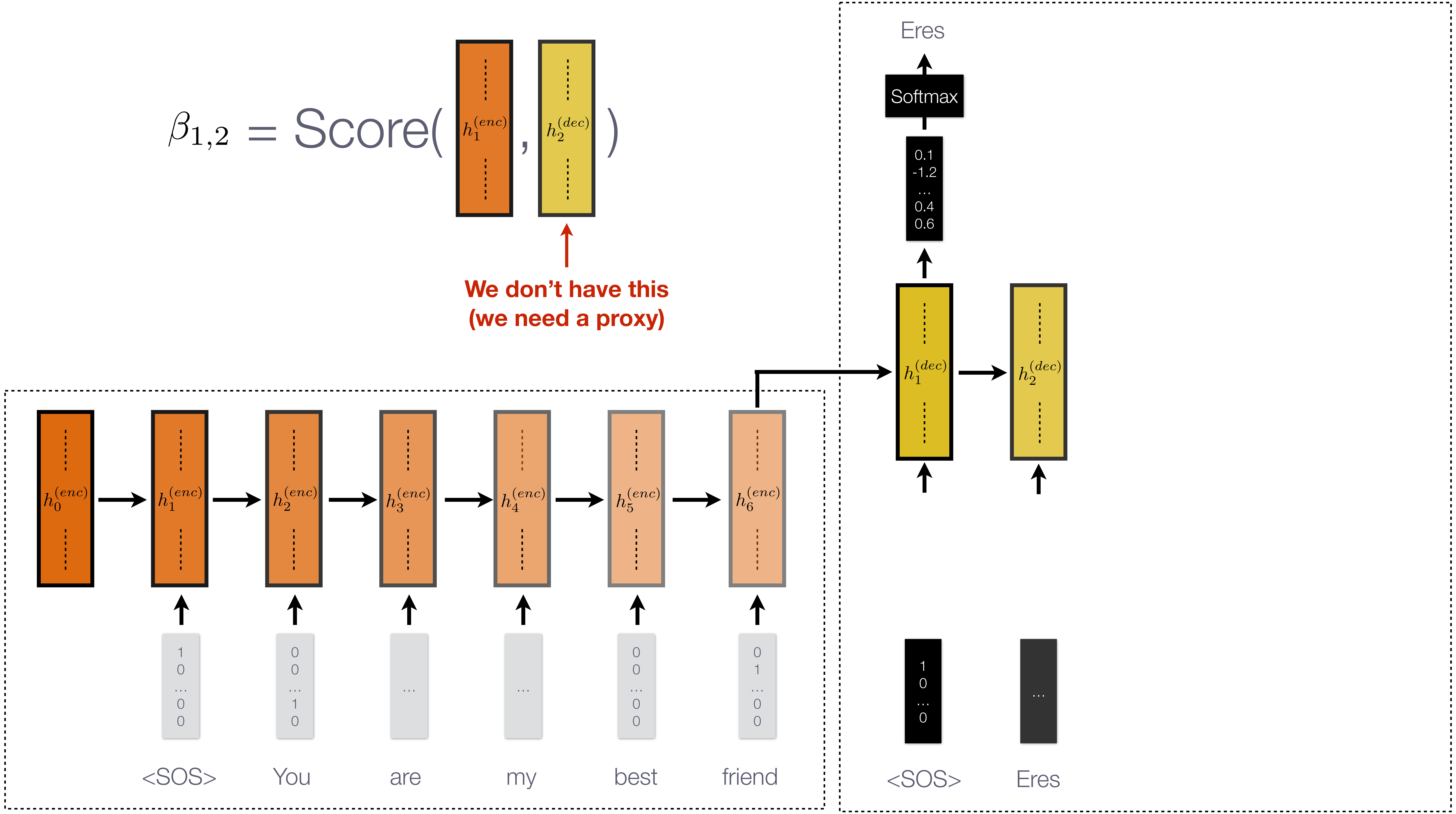
Context Vector
$c_1 = h_1^{(enc)} + h_2^{(enc)} + h_3^{(enc)} + h_4^{(enc)} + h_5^{(enc)} + h_6^{(enc)}$
$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$ $\alpha_{5,1}$ $\alpha_{6,1}$
Softmax
$\beta_{1,1}$ $\beta_{2,1}$ $\beta_{3,1}$ $\beta_{4,1}$ $\beta_{5,1}$ $\beta_{6,1}$
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
$h_1^{(dec)}$
<SOS> You are my best friend
Context Vector
c_1
<SOS>

Context Vector
$c_1 = h_1^{(enc)} + h_2^{(enc)} + h_3^{(enc)} + h_4^{(enc)} + h_5^{(enc)} + h_6^{(enc)}$
$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$ $\alpha_{5,1}$ $\alpha_{6,1}$
Softmax
$\beta_{1,1}$ $\beta_{2,1}$ $\beta_{3,1}$ $\beta_{4,1}$ $\beta_{5,1}$ $\beta_{6,1}$
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend
Eres
Softmax
0.1 -1.2 … 0.4 0.6
$h_1^{(dec)}$
Context Vector
c_1
1 0 … 0
<SOS>

Context Vector
$c_1 = h_1^{(enc)} + h_2^{(enc)} + h_3^{(enc)} + h_4^{(enc)} + h_5^{(enc)} + h_6^{(enc)}$
$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$ $\alpha_{5,1}$ $\alpha_{6,1}$
Softmax
$\beta_{1,1}$ $\beta_{2,1}$ $\beta_{3,1}$ $\beta_{4,1}$ $\beta_{5,1}$ $\beta_{6,1}$
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
<SOS>
You
are
my
best
friend
Eres
Softmax
0.1
-1.2
0.4
0.6
$h_1^{(dec)}$ $h_2^{(dec)}$
Context Vector
c_1
<SOS>
Eres

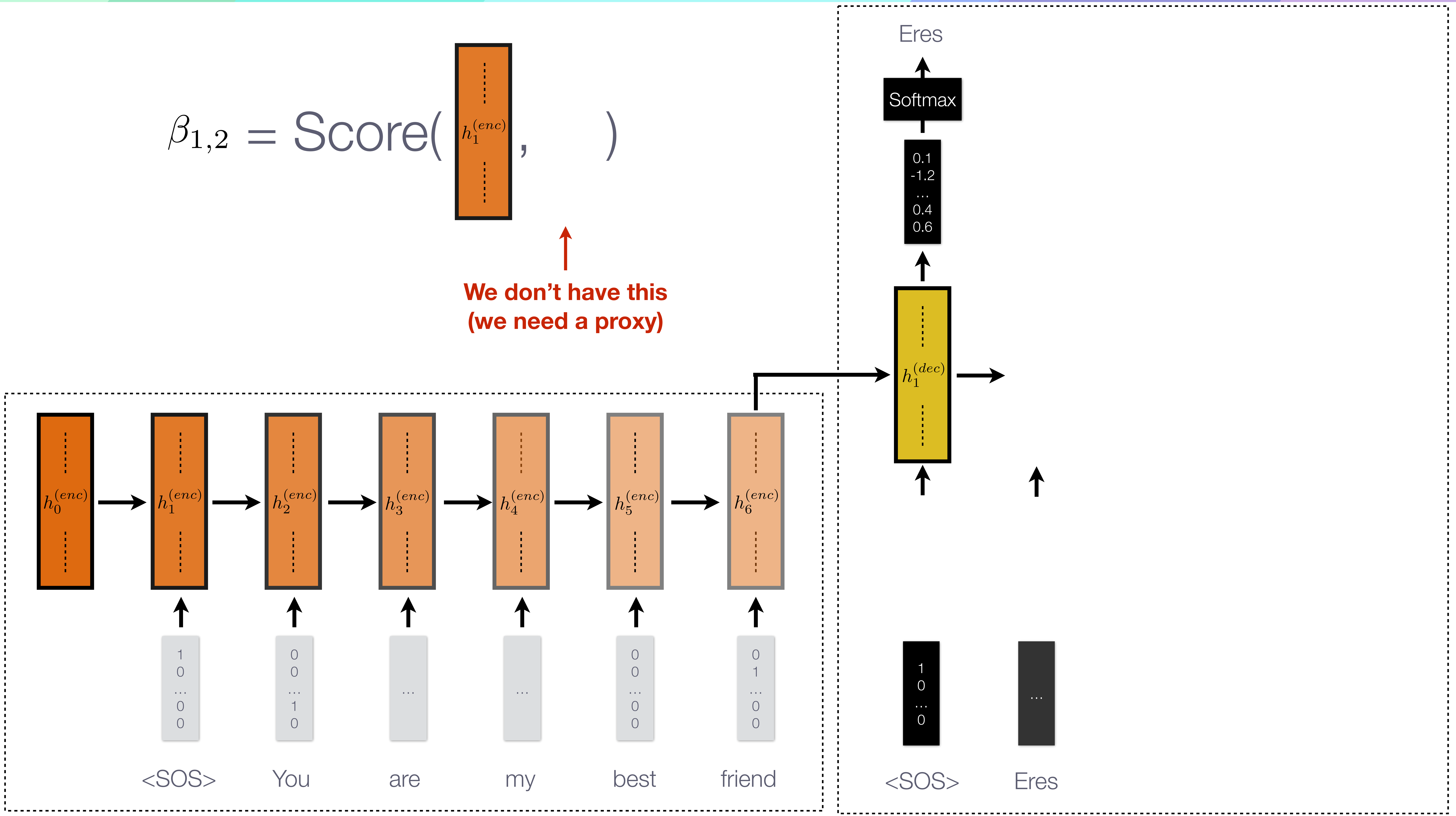


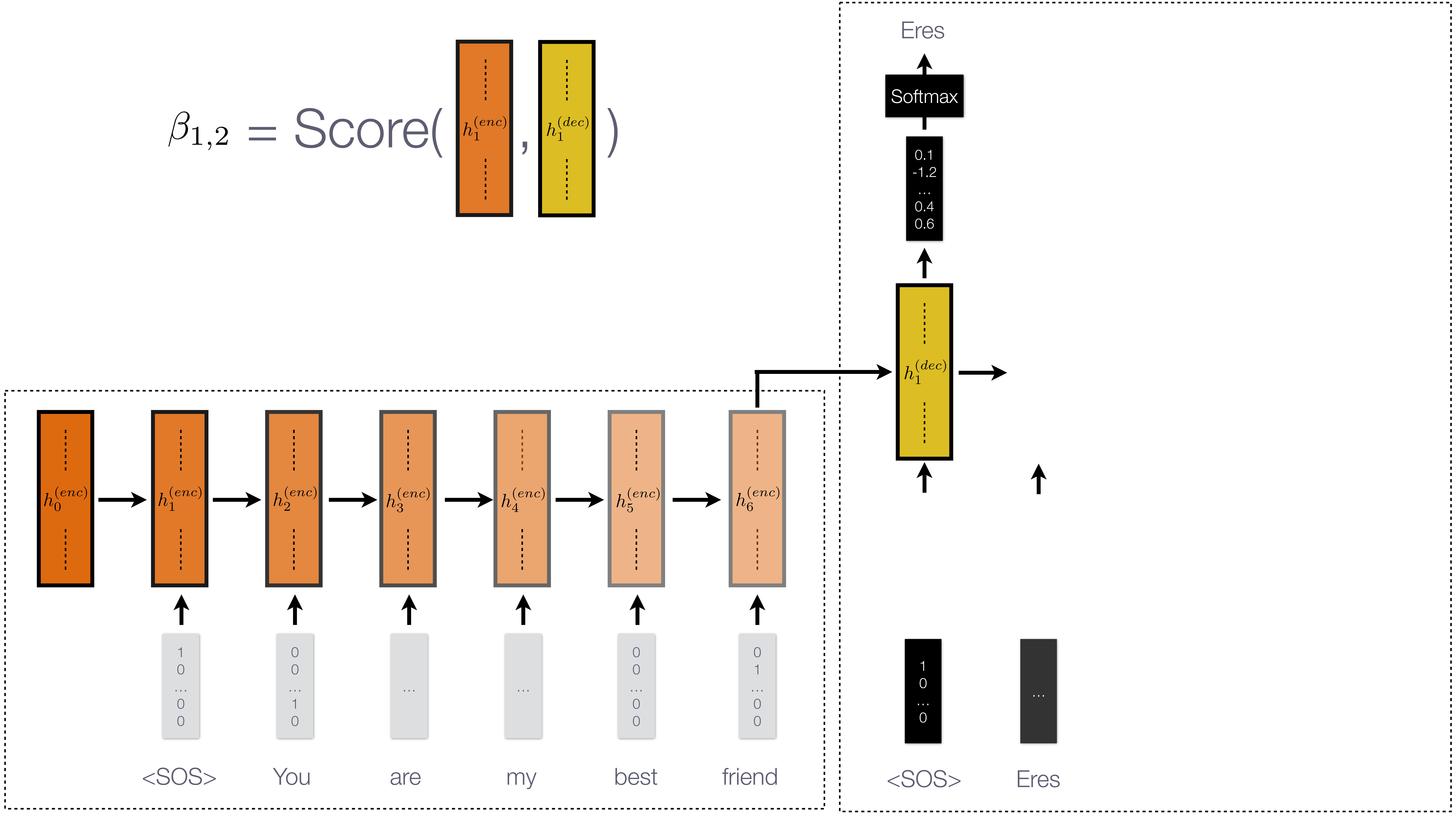
$\beta_{1,2} = \text{Score}(h_1^{(enc)}, h_2^{(dec)})$
Eres
Softmax
0.1
-1.2
...
0.4
0.6
$h_1^{(dec)}$
$h_2^{(dec)}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
<SOS>
You
are
my
best
friend
<SOS>
Eres


$\beta_{1,2}$ = Score($h_1^{(enc)}$, $h_2^{(dec)}$)
We don't have this
(we need a proxy)
Eres
Softmax
0.1
-1.2
…
0.4
0.6
$h_1^{(dec)}$
$h_2^{(dec)}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
<SOS>
You
are
my
best
friend
<SOS>
Eres

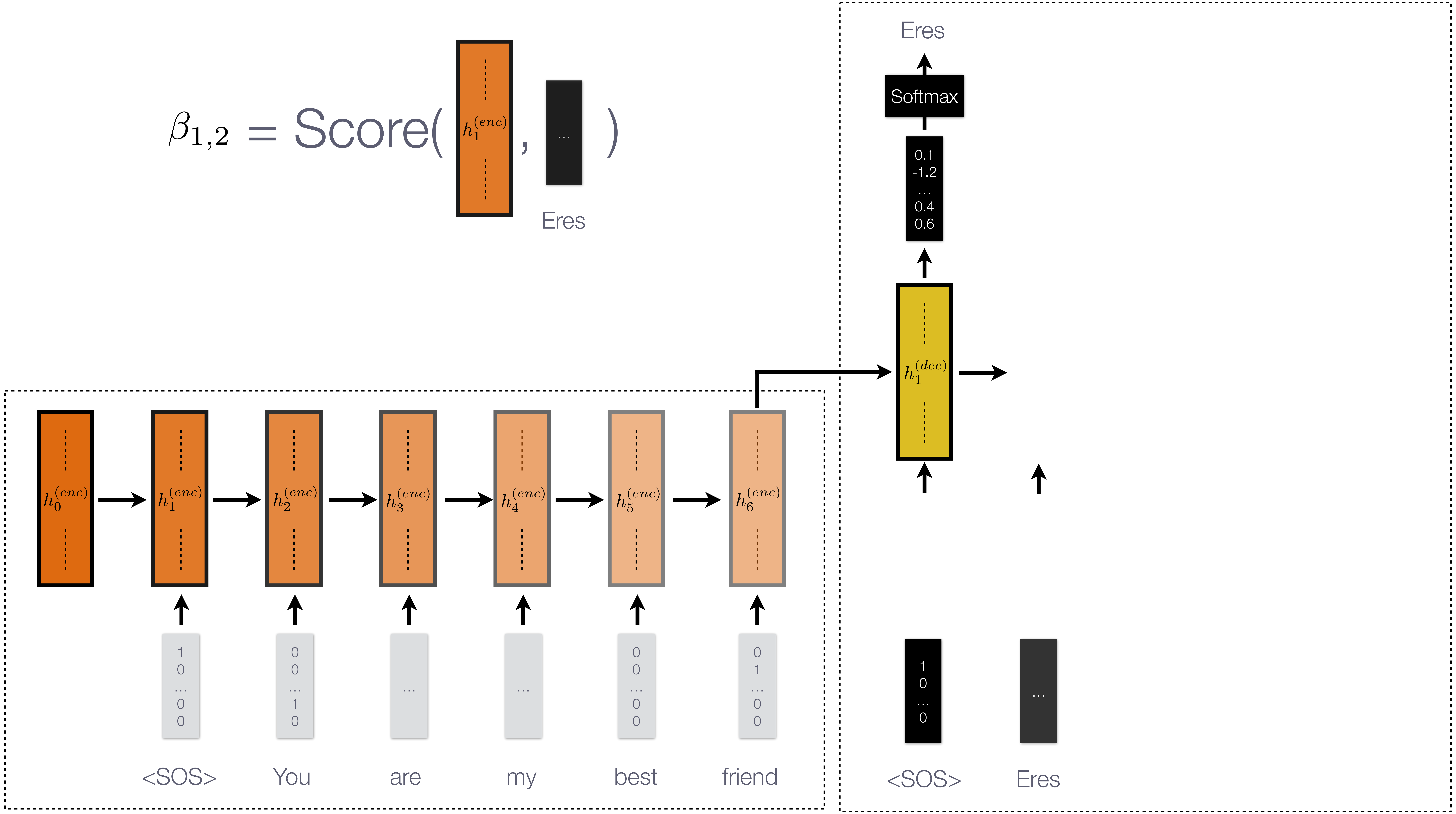
$$\beta_{1,2} = \text{Score}\left(h_1^{(enc)} , \quad \right)$$

We don't have this
(we need a proxy)

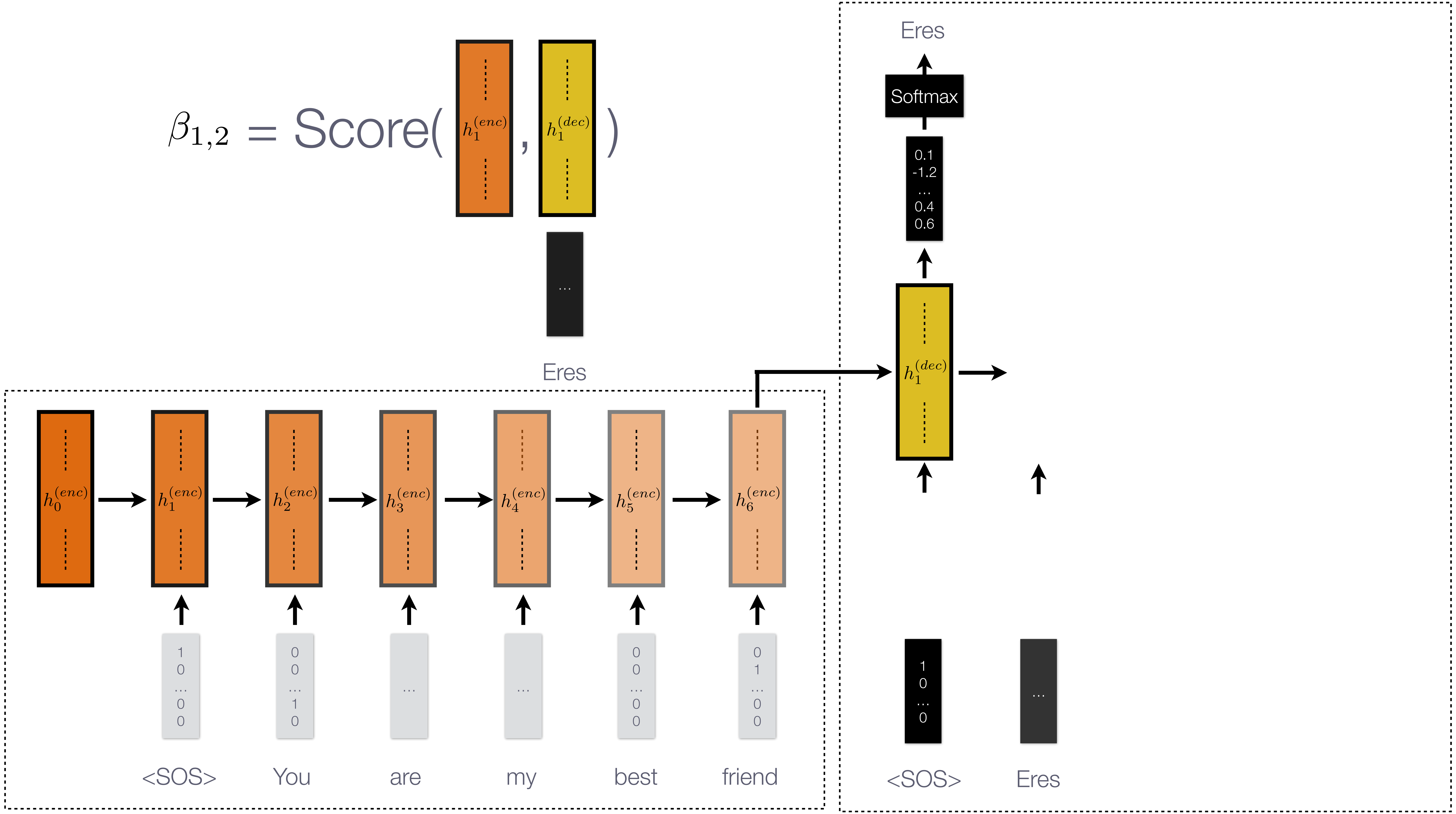




$\beta_{1,2} = \text{Score}(h_1^{(enc)}, h_1^{(dec)})$
Eres
Softmax
0.1
-1.2
…
0.4
0.6
$h_1^{(dec)}$
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
<SOS>
You
are
my
best
friend
<SOS>
Eres

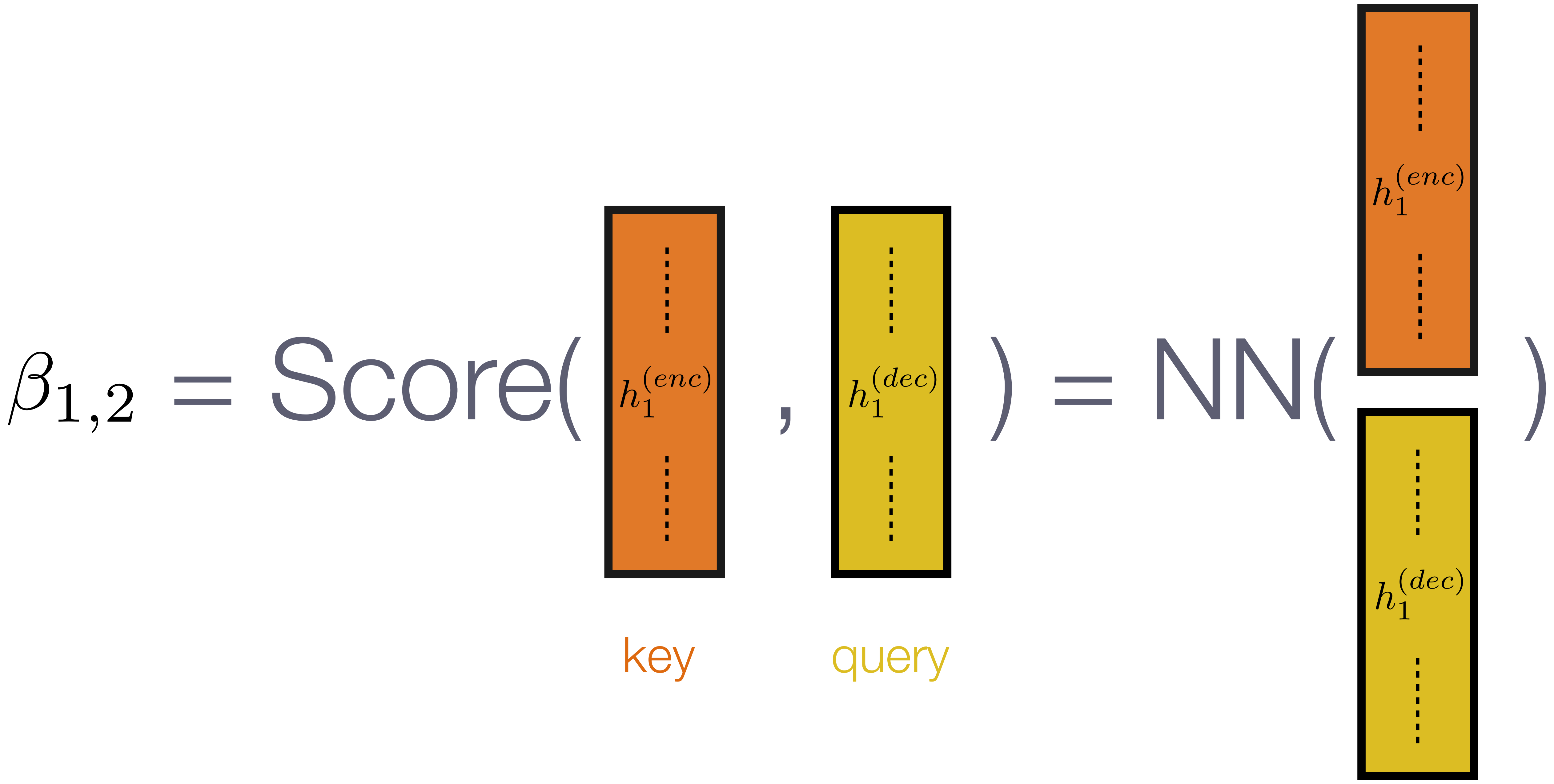


$\beta_{1,2} = \text{Score}(h_1^{(enc)}, \ldots)$
Eres
$h_0^{(enc)}$ $h_1^{(enc)}$ $h_2^{(enc)}$ $h_3^{(enc)}$ $h_4^{(enc)}$ $h_5^{(enc)}$ $h_6^{(enc)}$
<SOS>
You
are
my
best
friend
$h_1^{(dec)}$
0.1
-1.2
…
0.4
0.6
Softmax
Eres
<SOS>
Eres



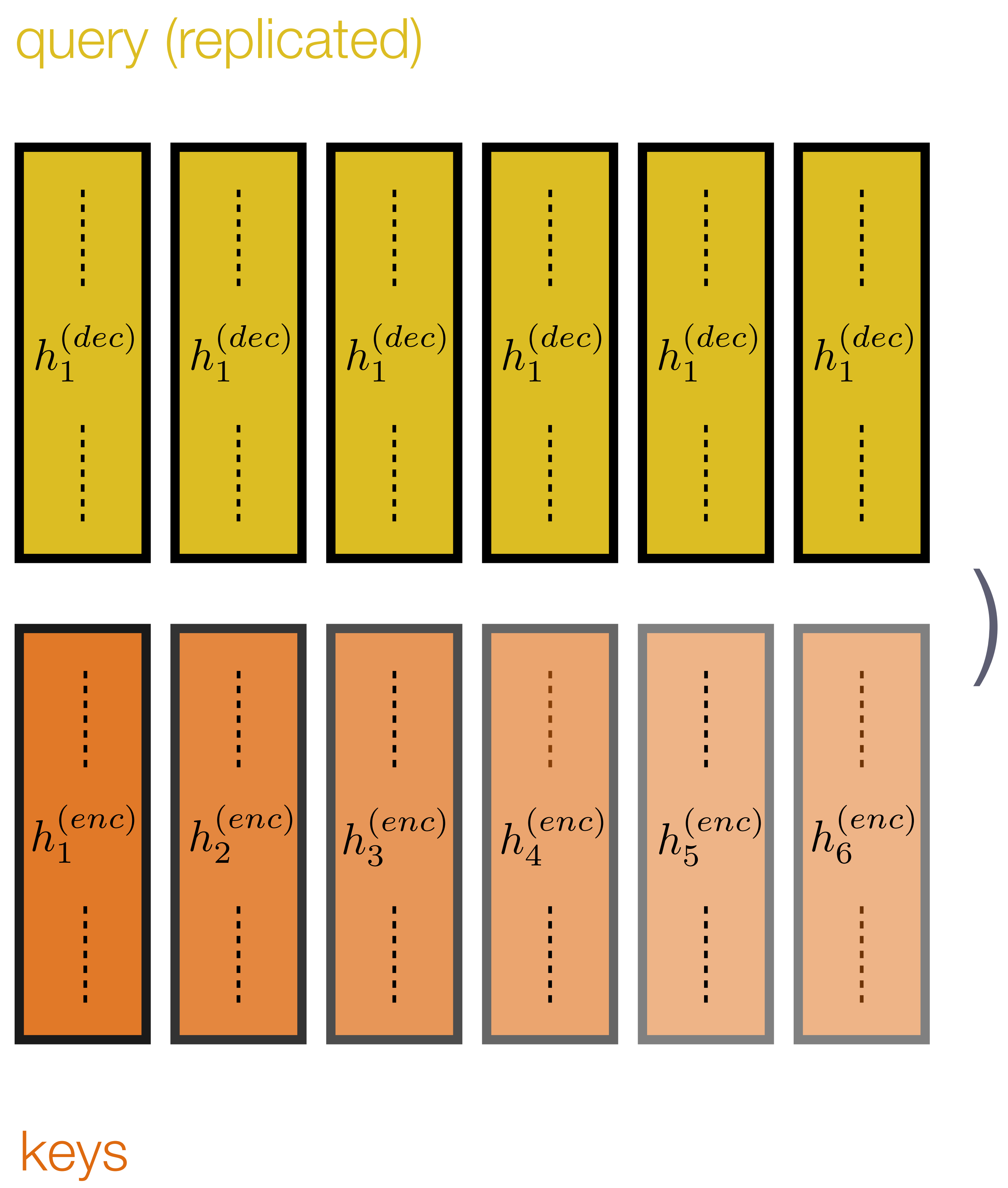
$\beta_{1,2} = \text{Score}(h_1^{(enc)}, h_1^{(dec)})$
Eres
Softmax
0.1
-1.2
…
0.4
0.6
$h_1^{(dec)}$
Eres
$h_0^{(enc)}$
$h_1^{(enc)}$
$h_2^{(enc)}$
$h_3^{(enc)}$
$h_4^{(enc)}$
$h_5^{(enc)}$
$h_6^{(enc)}$
1 0 … 0 0
0 0 … 1 0
…
…
0 0 … 0 0
0 1 … 0 0
<SOS>
You
are
my
best
friend
1 0 … 0
…
<SOS>
Eres

Additive Attention



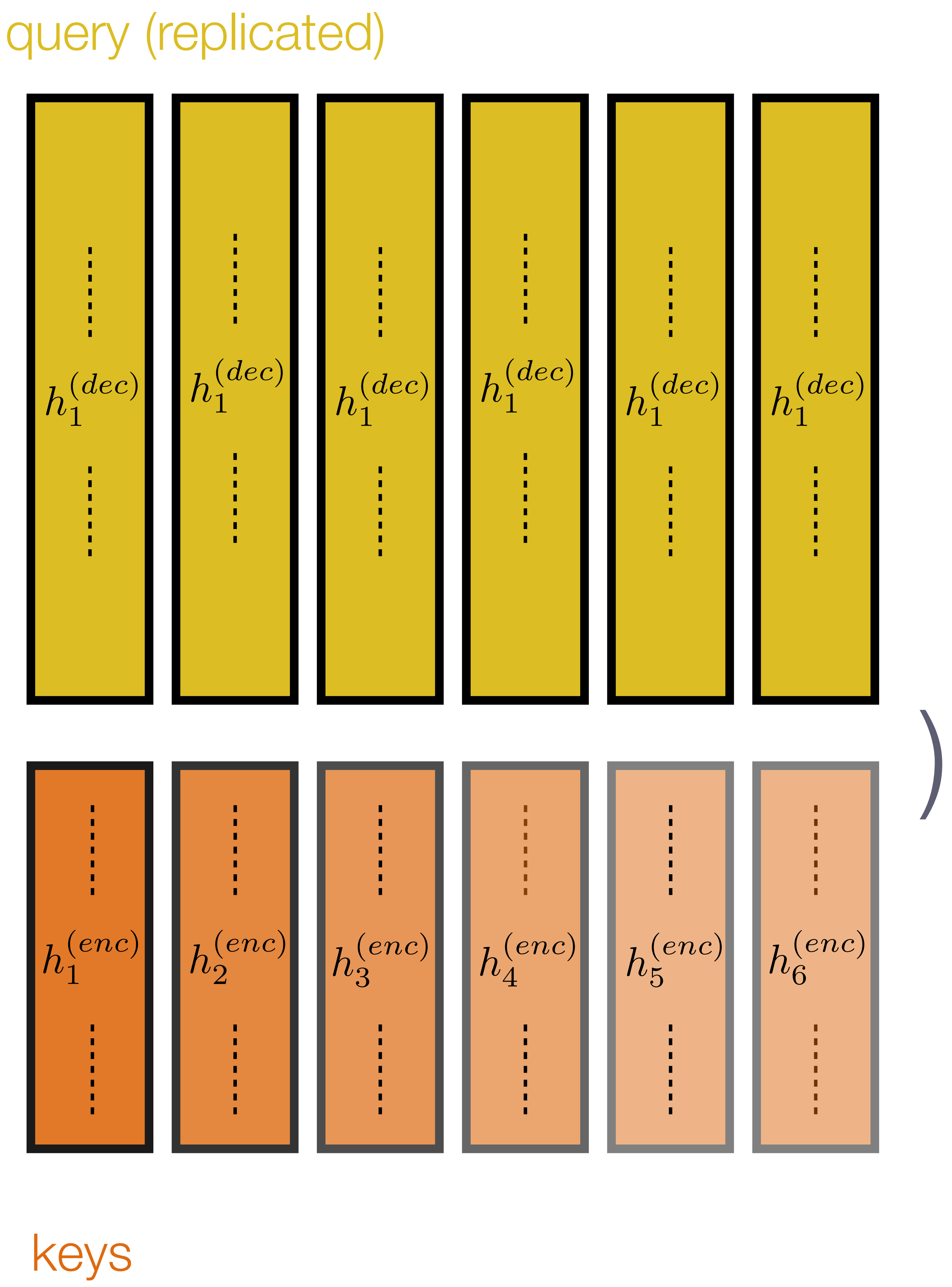
Additive Attention

$$\beta_{1,2}, \beta_{2,2}, \beta_{3,2}, \beta_{4,2}, \beta_{5,2}, \beta_{6,2} = \text{NN}(\quad)$$

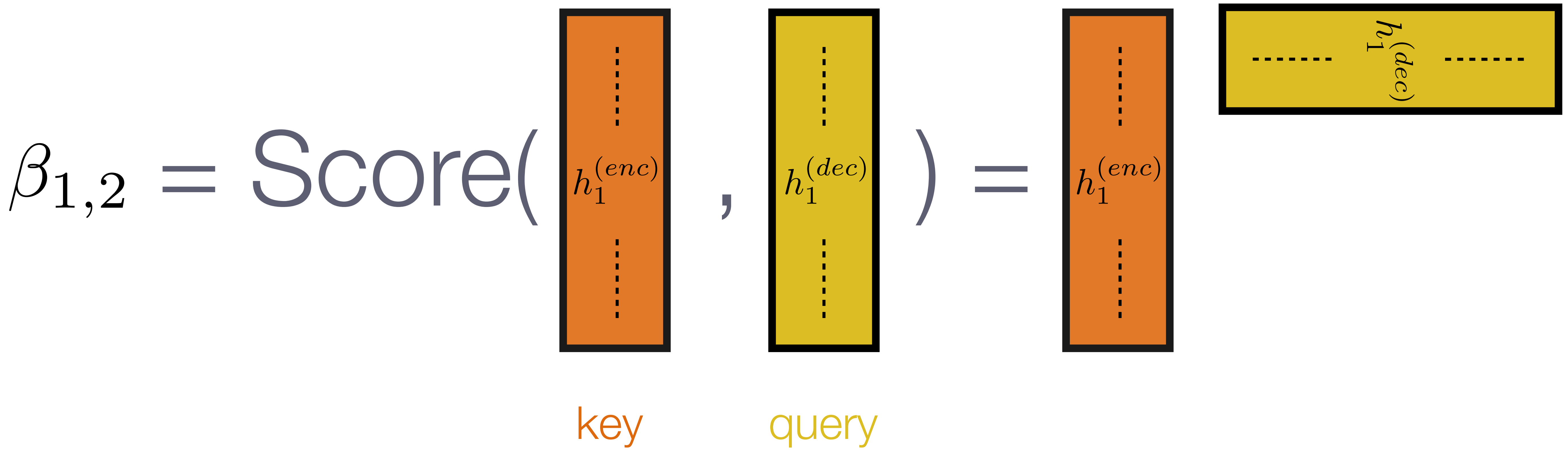


Additive Attention

$$\beta_{1,2}, \beta_{2,2}, \beta_{3,2}, \beta_{4,2}, \beta_{5,2}, \beta_{6,2} = \text{NN}(\quad)$$

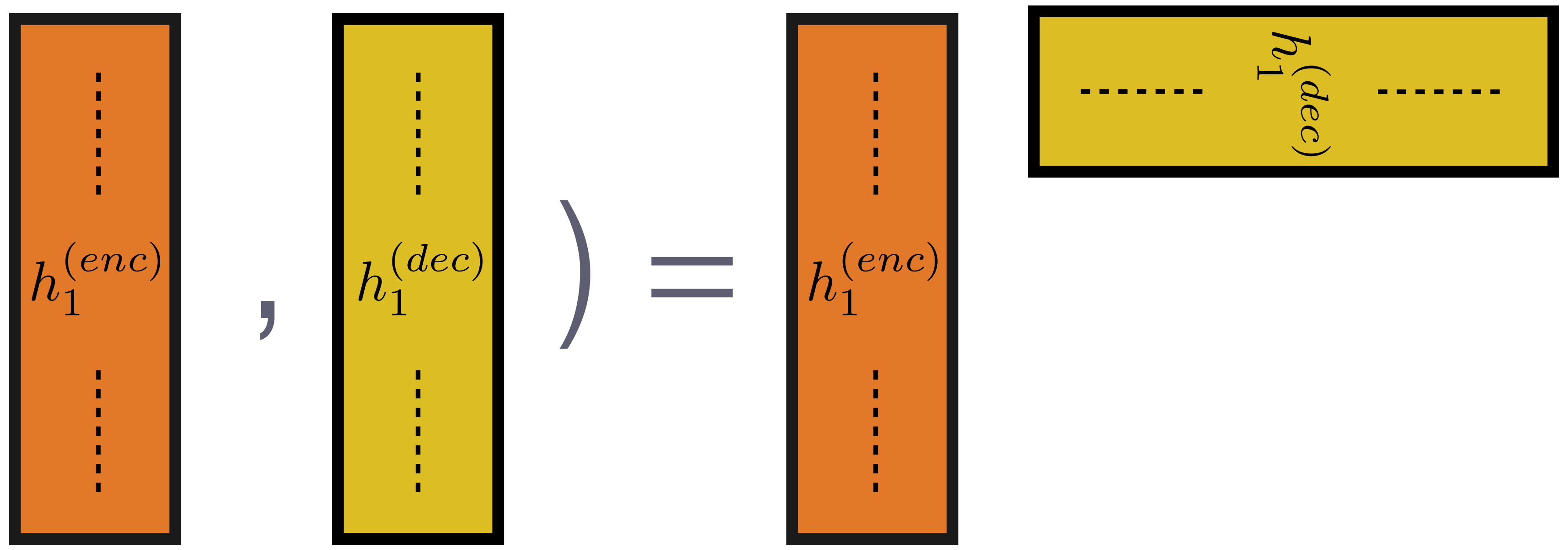


Dot-product Attention



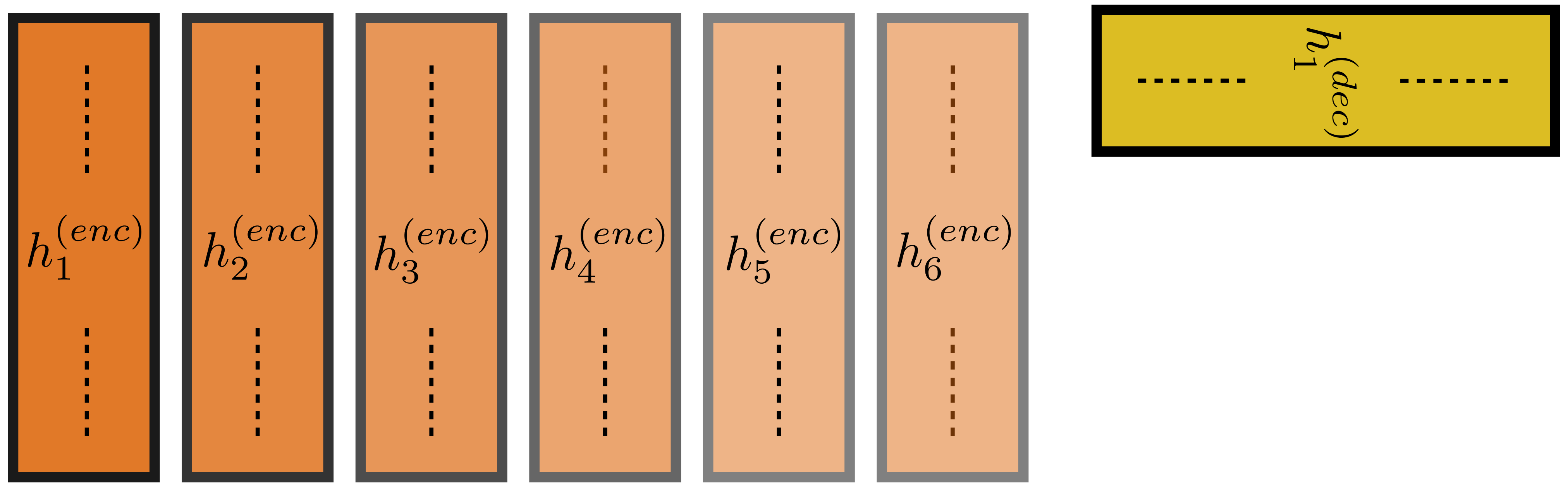
Dot-product Attention

$$\beta_{1,2} = \text{Score}(\;,\;) =$$



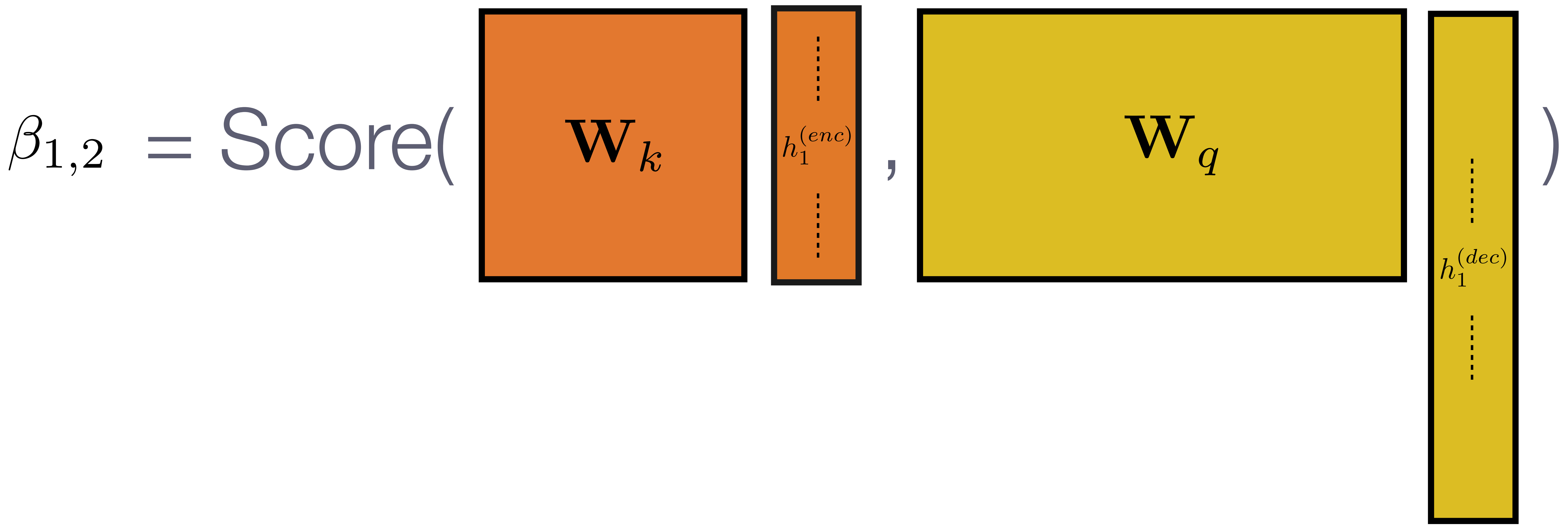
key   query

$$\beta_{1,2}, \beta_{2,2}, \beta_{3,2}, \beta_{4,2}, \beta_{5,2}, \beta_{6,2} \quad =$$



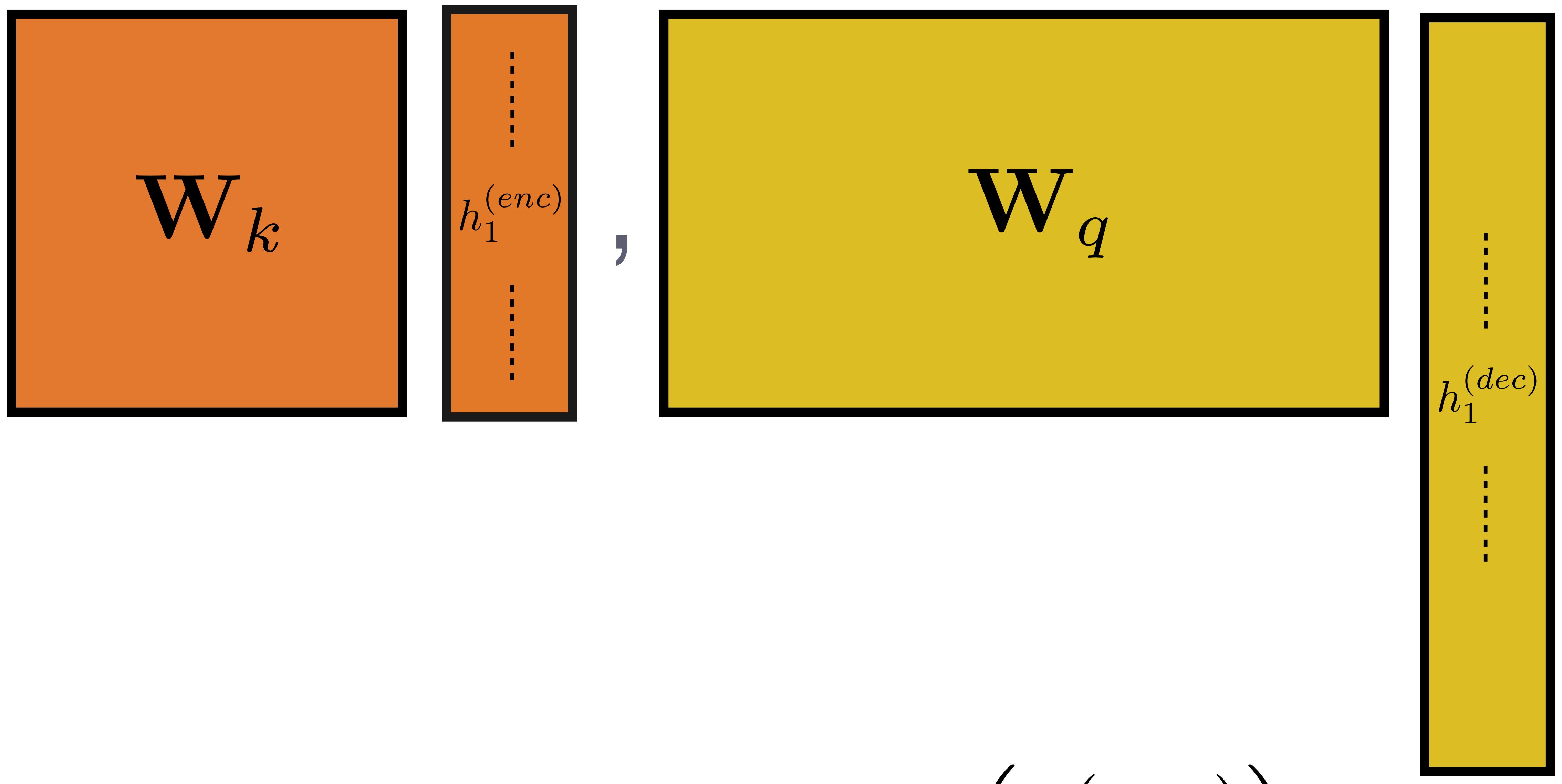
keys

General Dot-product Attention



Scaled General Dot-product Attention

$\beta_{1,2}$ = Score(



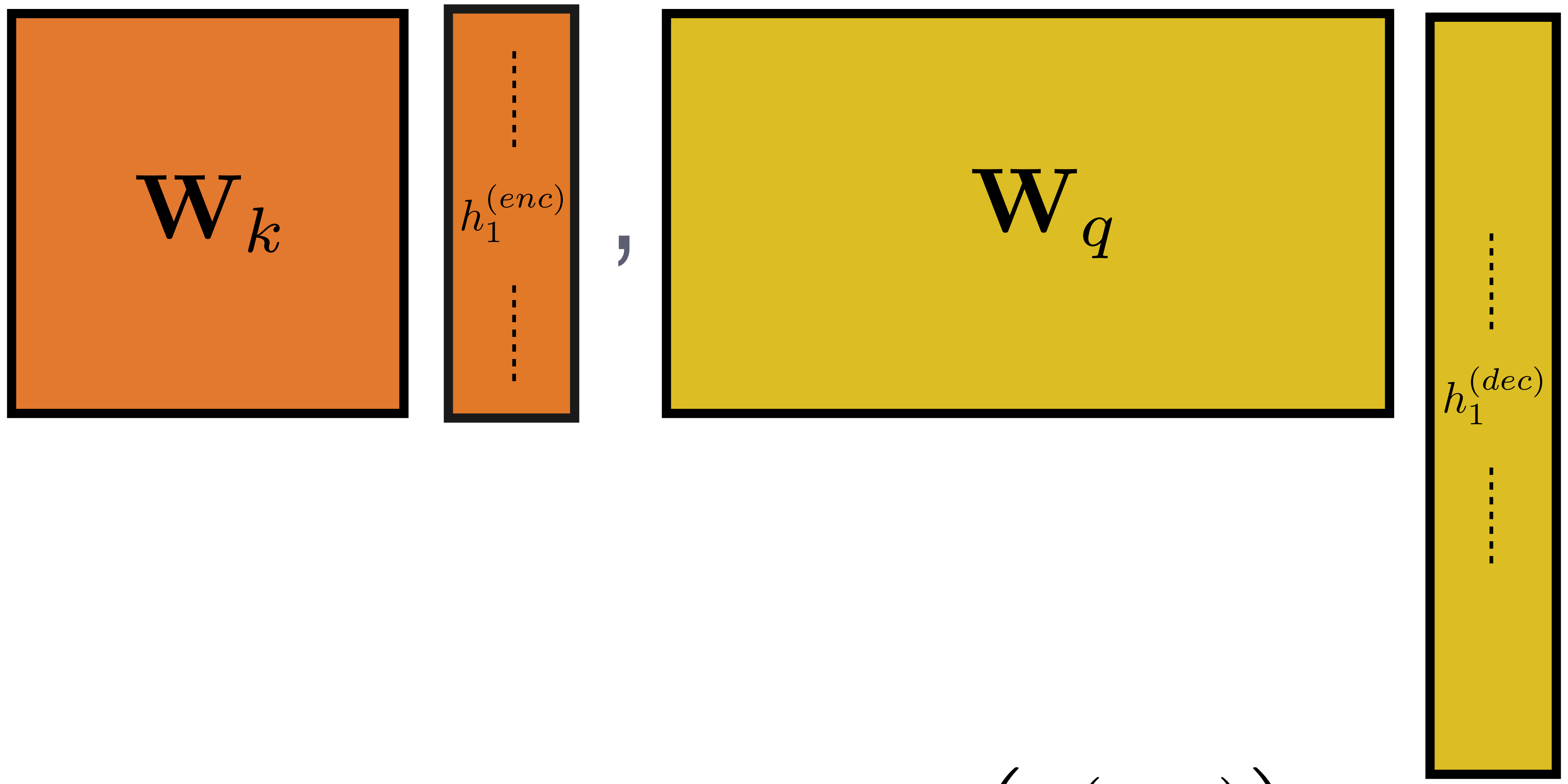
)

$$n = \text{Length}\left(h_i^{(enc)}\right)$$

$$\hat{\beta}_{1,2} = \frac{\beta_{1,2}}{\sqrt{n}}$$

Scaled General Dot-product Attention

$\beta_{1,2}$ = Score(



)

$$\hat{\beta}_{1,2} = \frac{\beta_{1,2}}{\sqrt{n}}$$

$$n = \text{Length}\left(h_i^{(enc)}\right)$$

$$n = \text{Length}\left(h_i^{(dec)}\right)$$

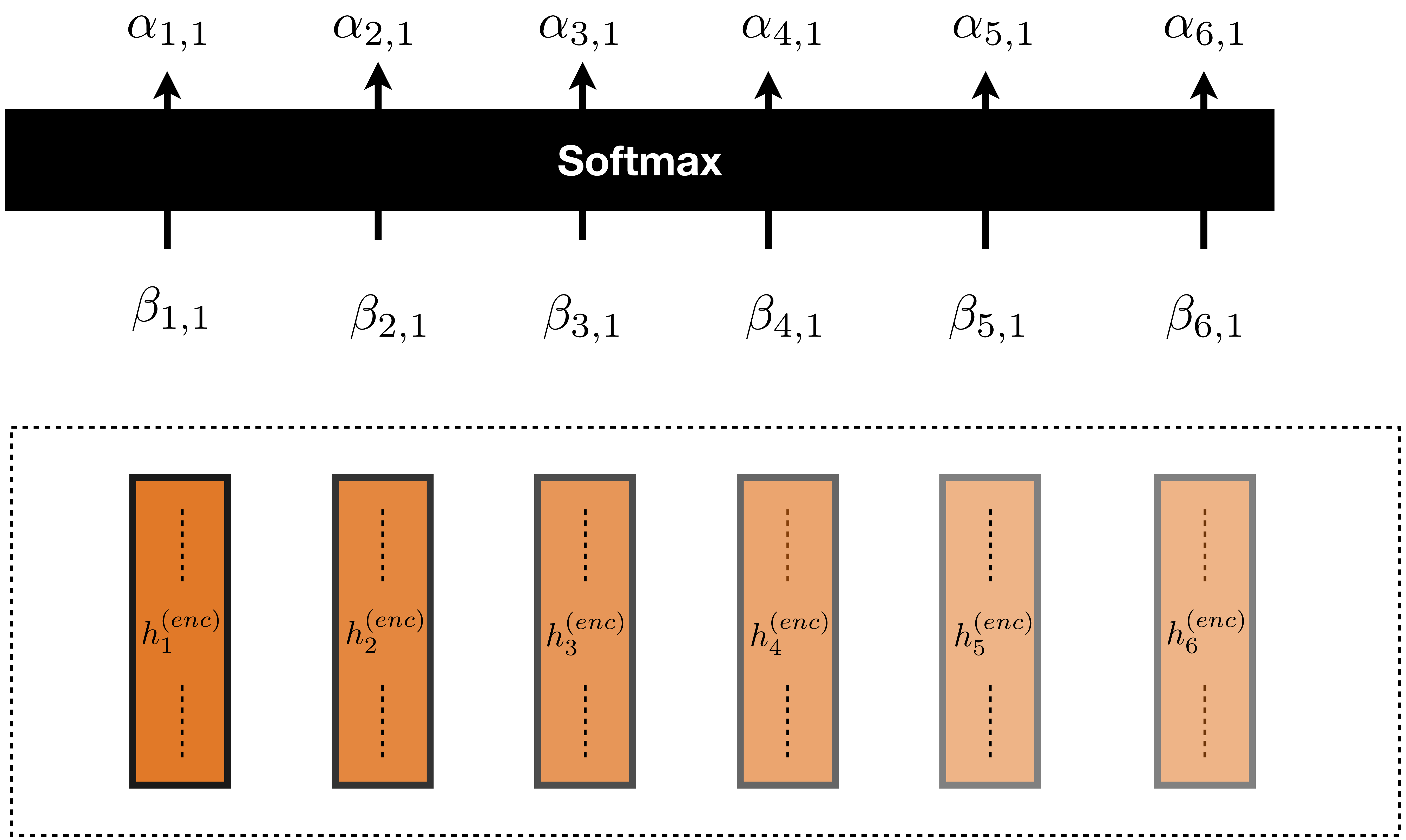
Soft **Attention** in details

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

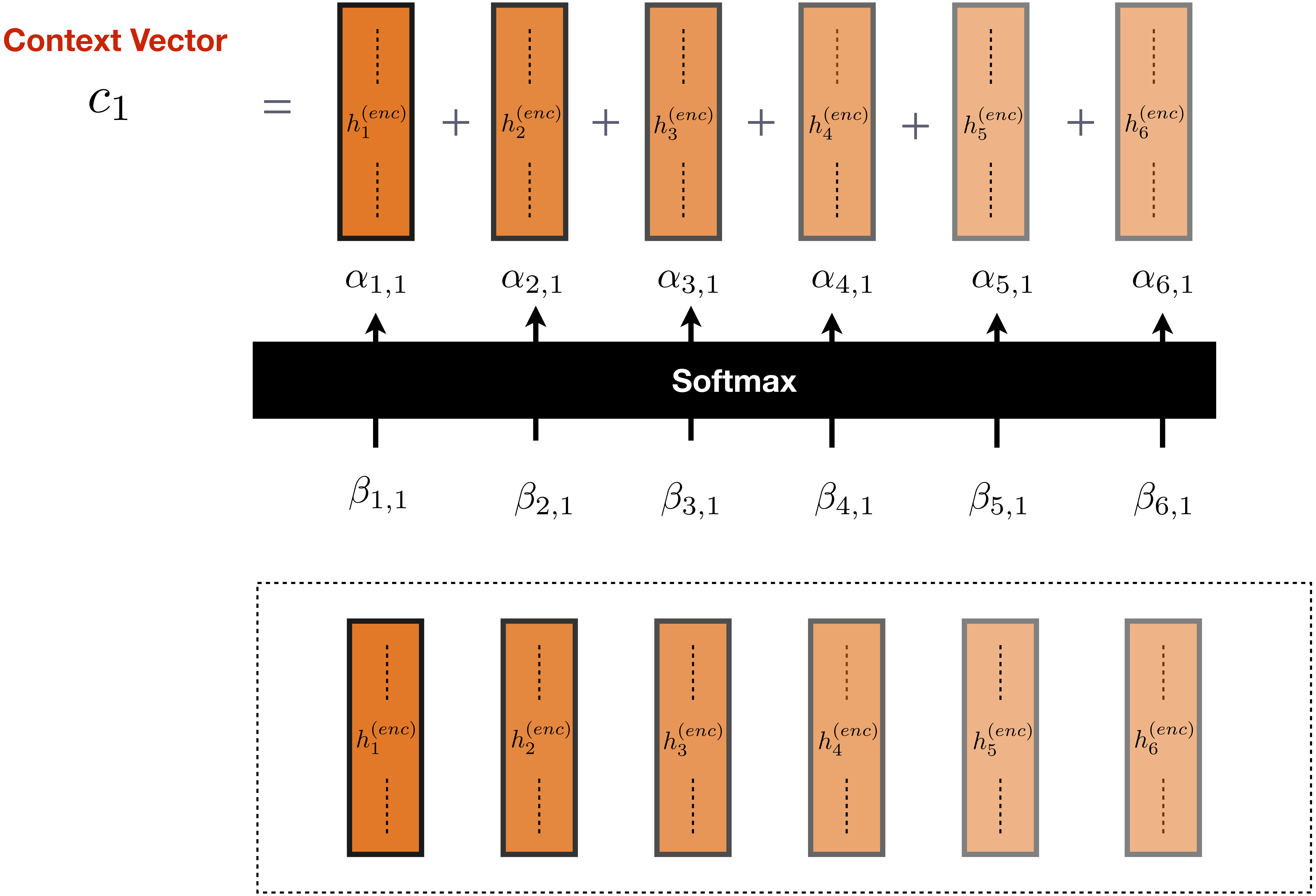
Forming a **Context** Vector

Context Vector

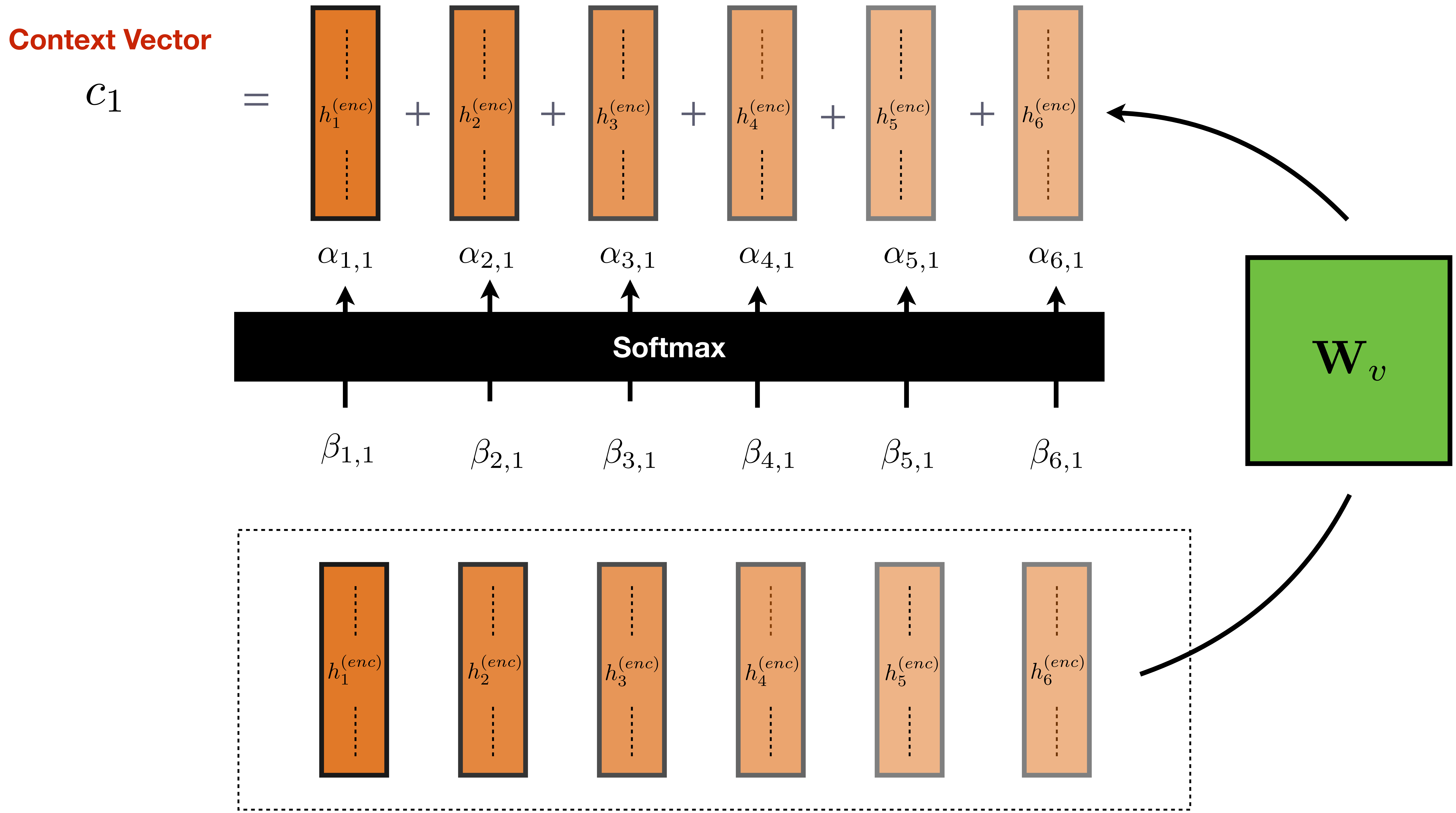
$$c_1 \qquad =$$



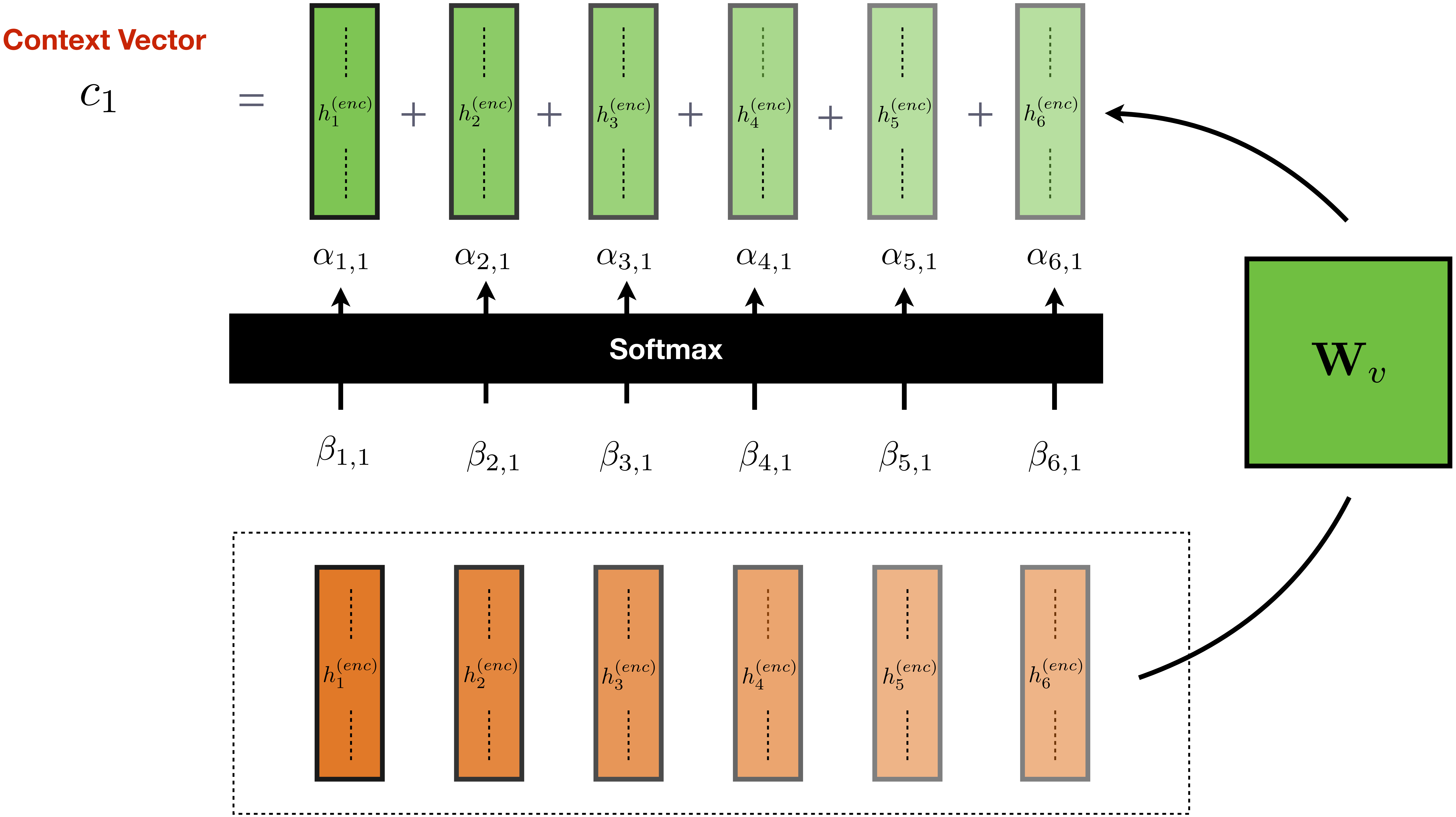
Forming a **Context** Vector



Forming a **Context** Vector



Forming a **General Context** Vector



Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

Relevance of encoding at token i for decoding token t

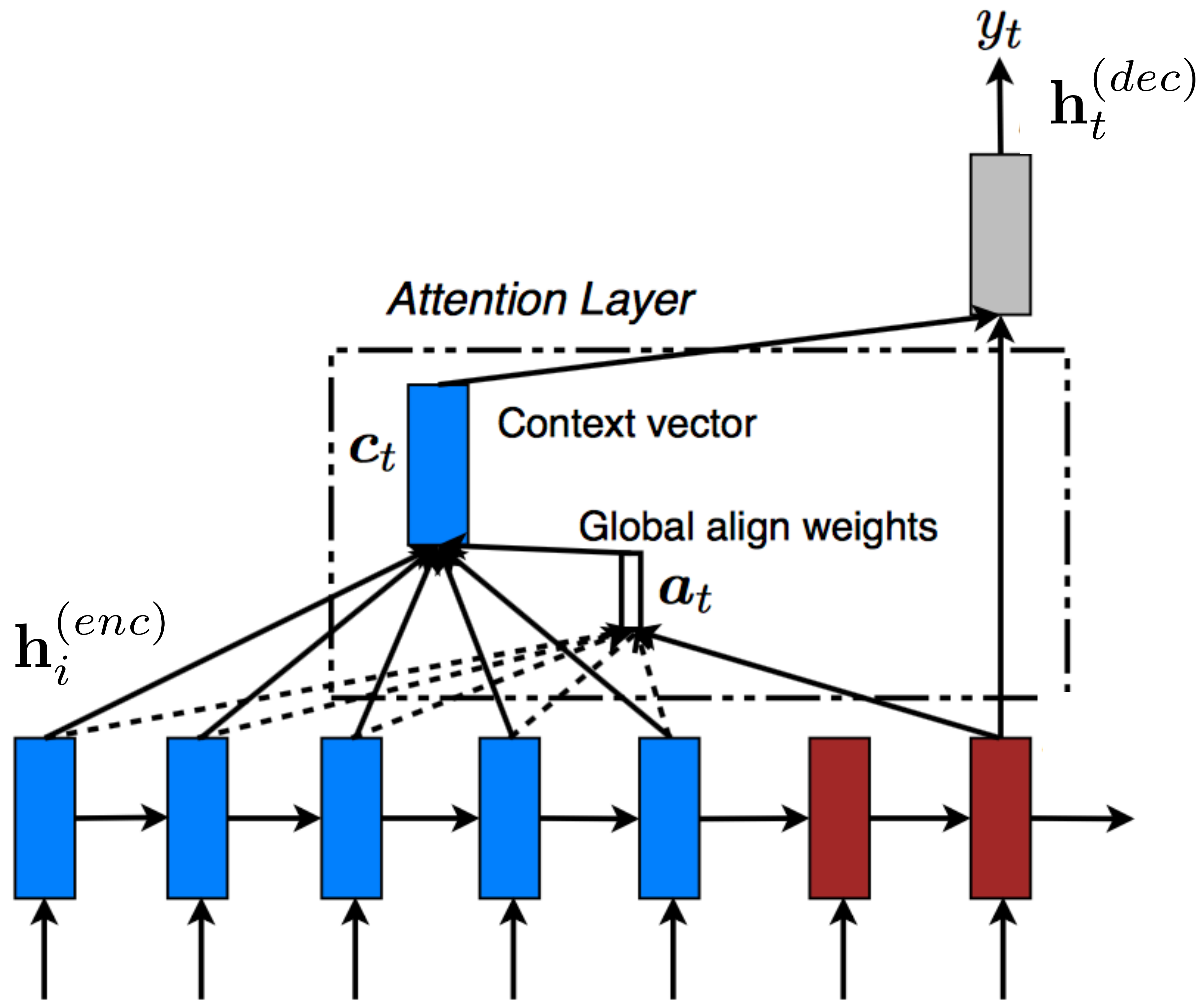
Query: $\mathbf{Q}_t$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{x}_t^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_{t-1}^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, [\mathbf{x}_t^{(dec)}, \mathbf{h}_{t-1}^{(dec)}])$$

Key: $\mathbf{K}_i$



$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

Normalize the weights to sum to 1

$$\mathbf{c}_t = \sum_i \alpha_{i,t} \mathbf{h}_i^{(enc)}$$

Value: $\mathbf{V}_i$

Form a context vector that would simply be added to the standard decoder input

Generalized Soft **Attention** in details

$$\beta_{i,t} = score(\mathbf{h}_i^{(enc)}, \mathbf{h}_t^{(dec)})$$

Relevance of encoding at token i for decoding token t

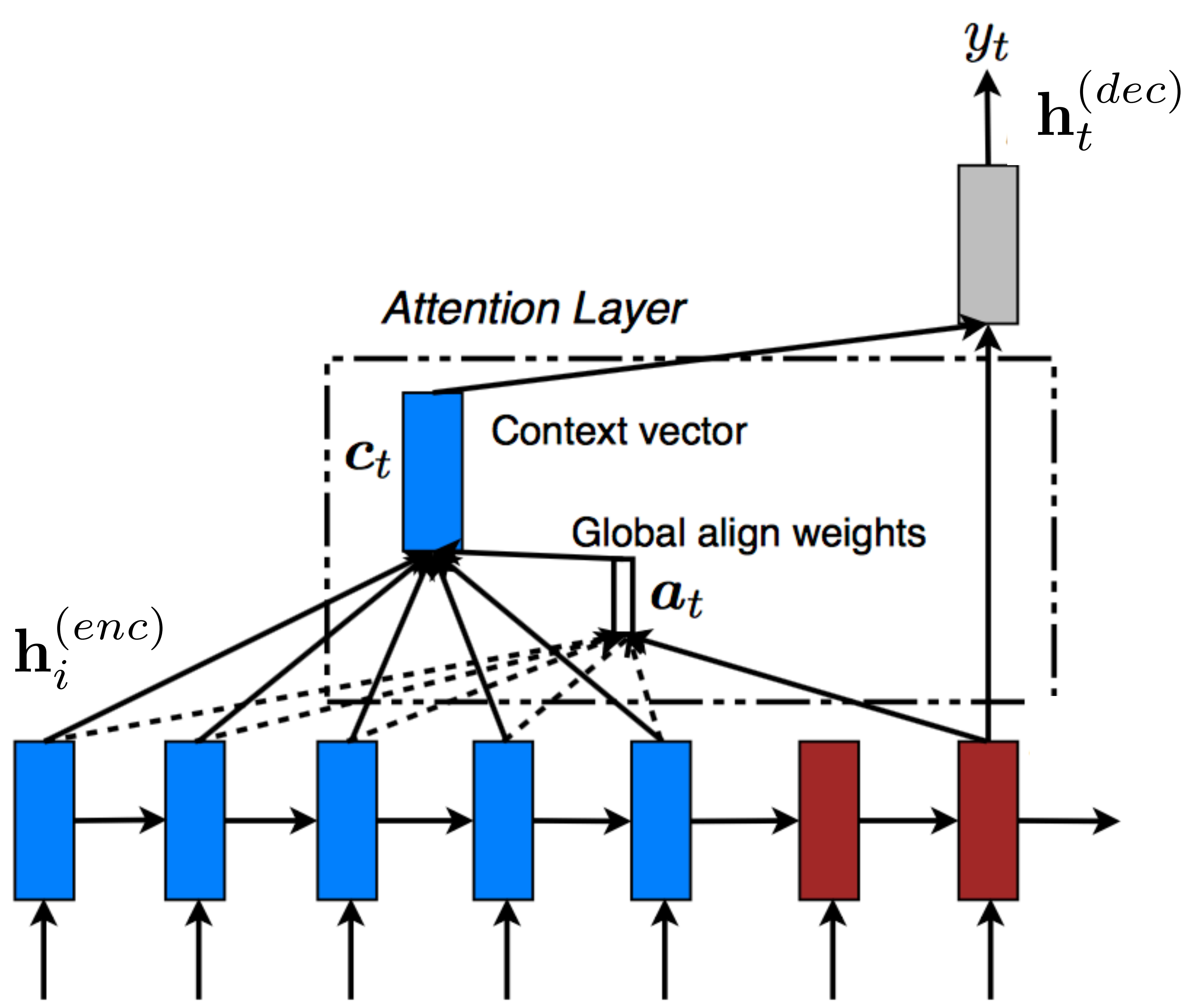
Query: $\mathbf{Q}_t$

$$\beta_{i,t} = score(\mathbf{W}_k \mathbf{h}_i^{(enc)}, \mathbf{W}_q \mathbf{x}_t^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{W}_k \mathbf{h}_i^{(enc)}, \mathbf{W}_q \mathbf{h}_{t-1}^{(dec)})$$

$$\beta_{i,t} = score(\mathbf{W}_k \mathbf{h}_i^{(enc)}, \mathbf{W}_q [\mathbf{x}_t^{(dec)}, \mathbf{h}_{t-1}^{(dec)}])$$

Key: $\mathbf{K}_i$



$$\alpha_{i,t} = \text{Softmax}(\beta_{i,t})$$

Normalize the weights to sum to 1

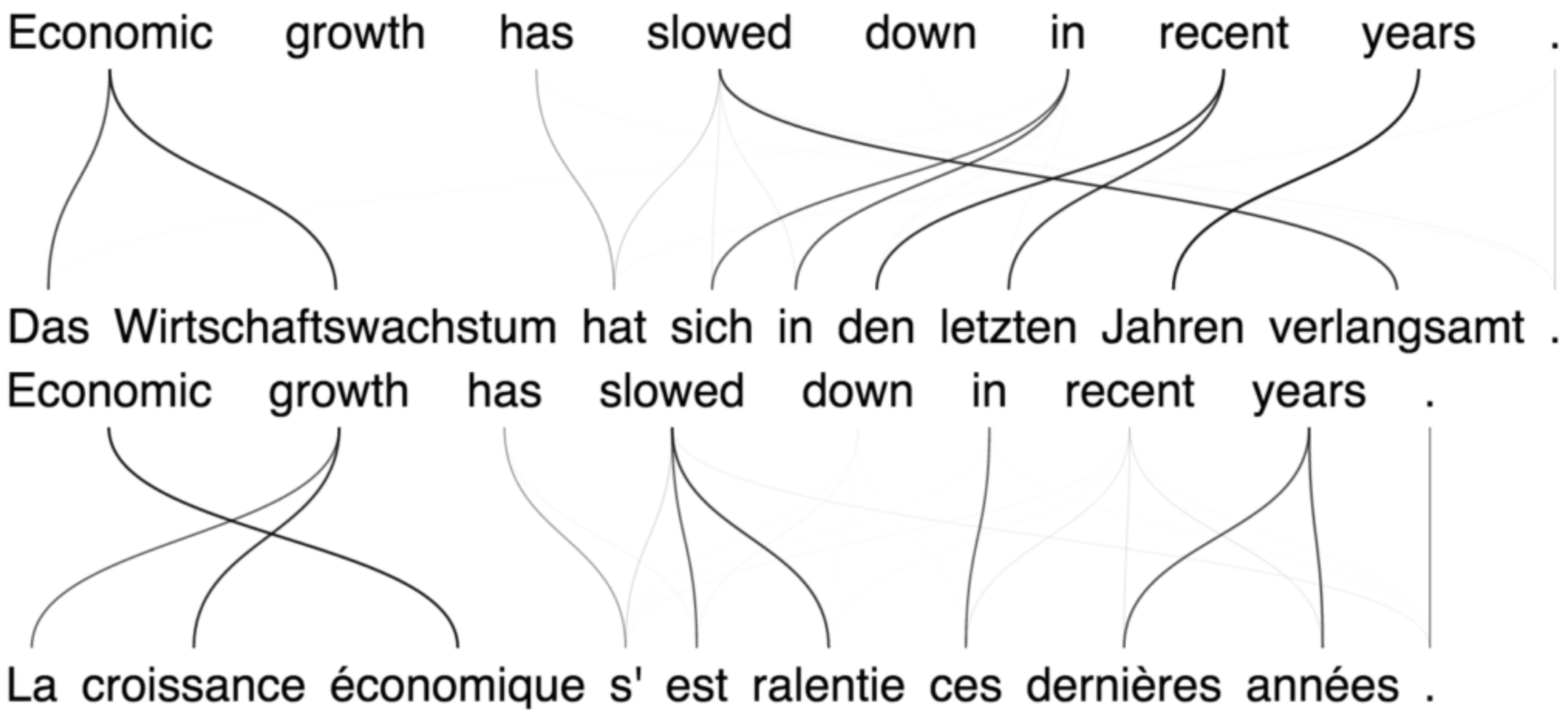
$$\mathbf{c}_t = \sum_i \alpha_{i,t} \mathbf{W}_v \mathbf{h}_i^{(enc)}$$

Value: $\mathbf{V}_i$

Form a context vector that would simply be added to the standard decoder input

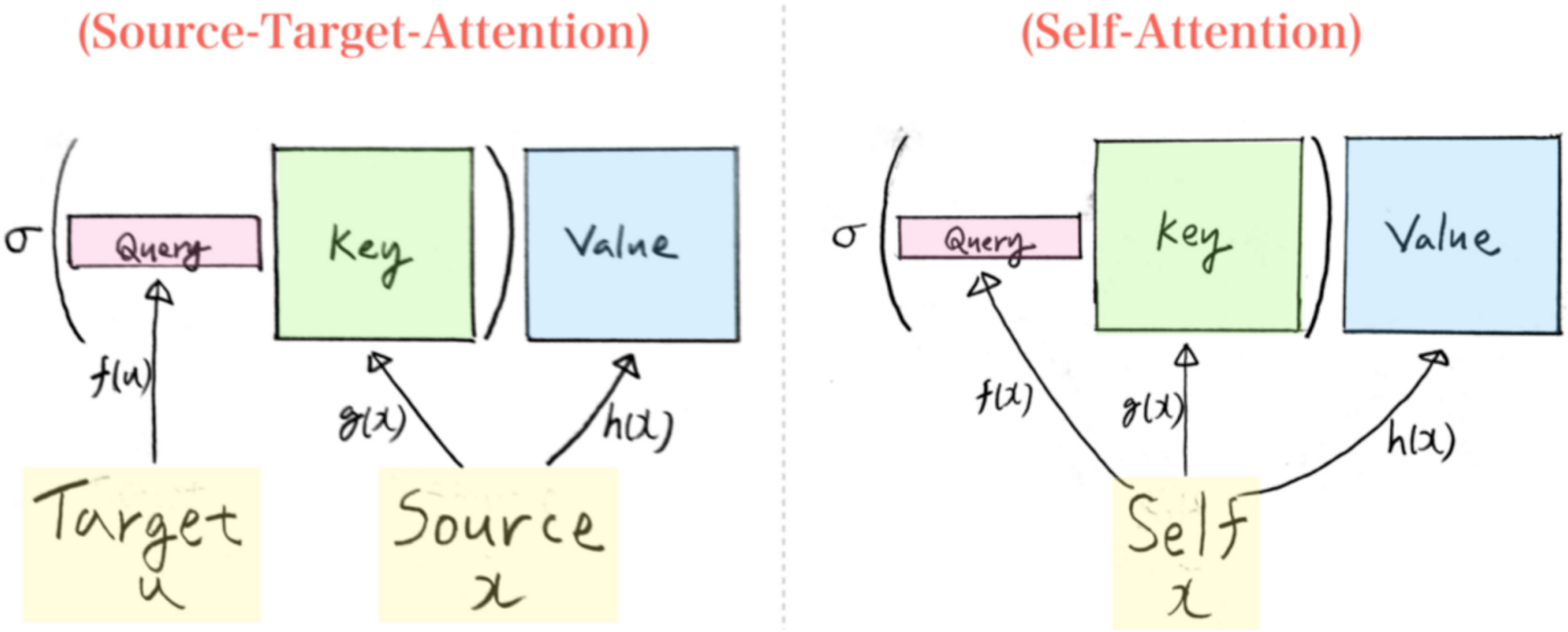
Attention Mechanisms and RNNs

[Cho et al., 2015]



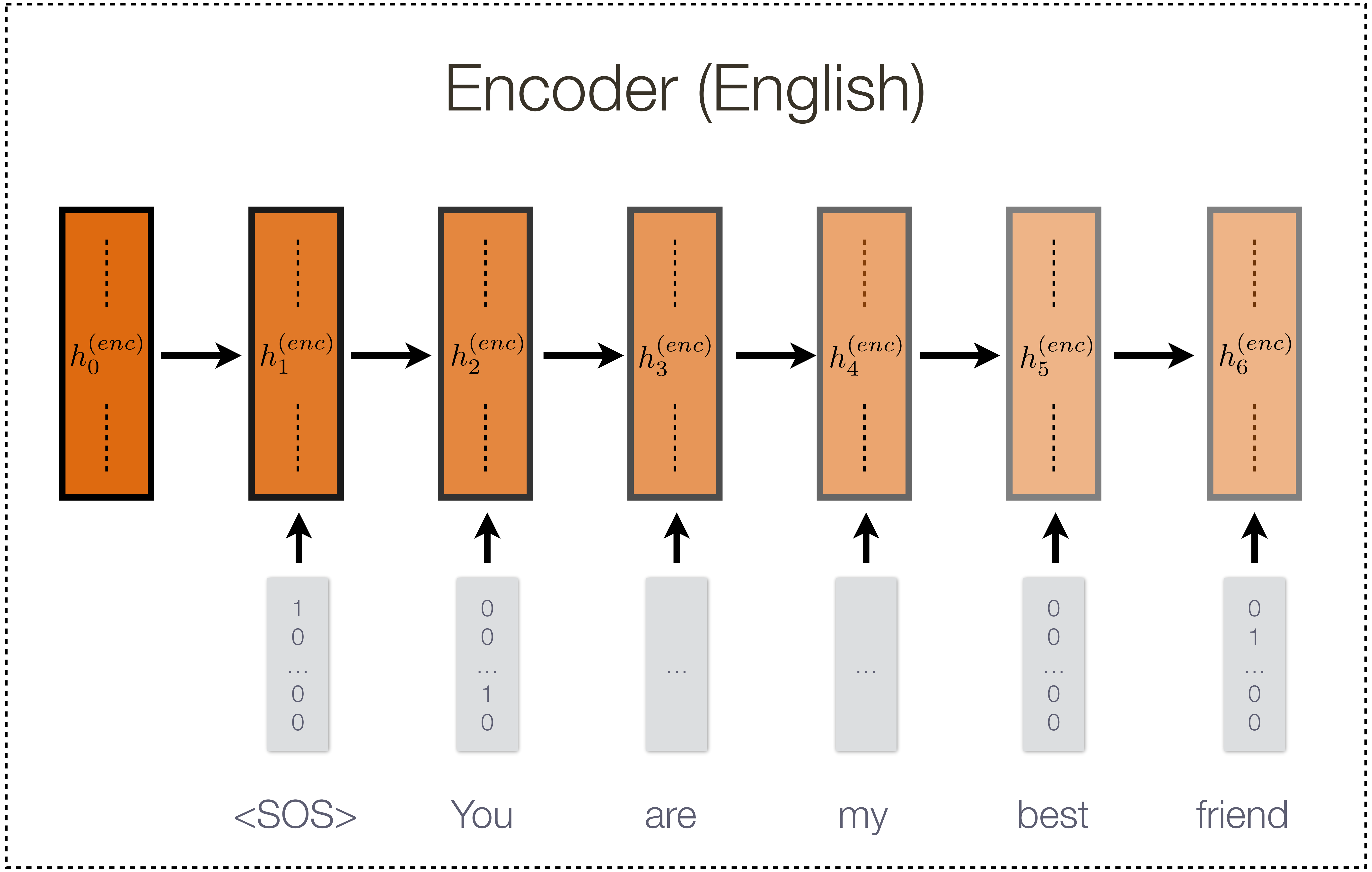
https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/

Self Attention

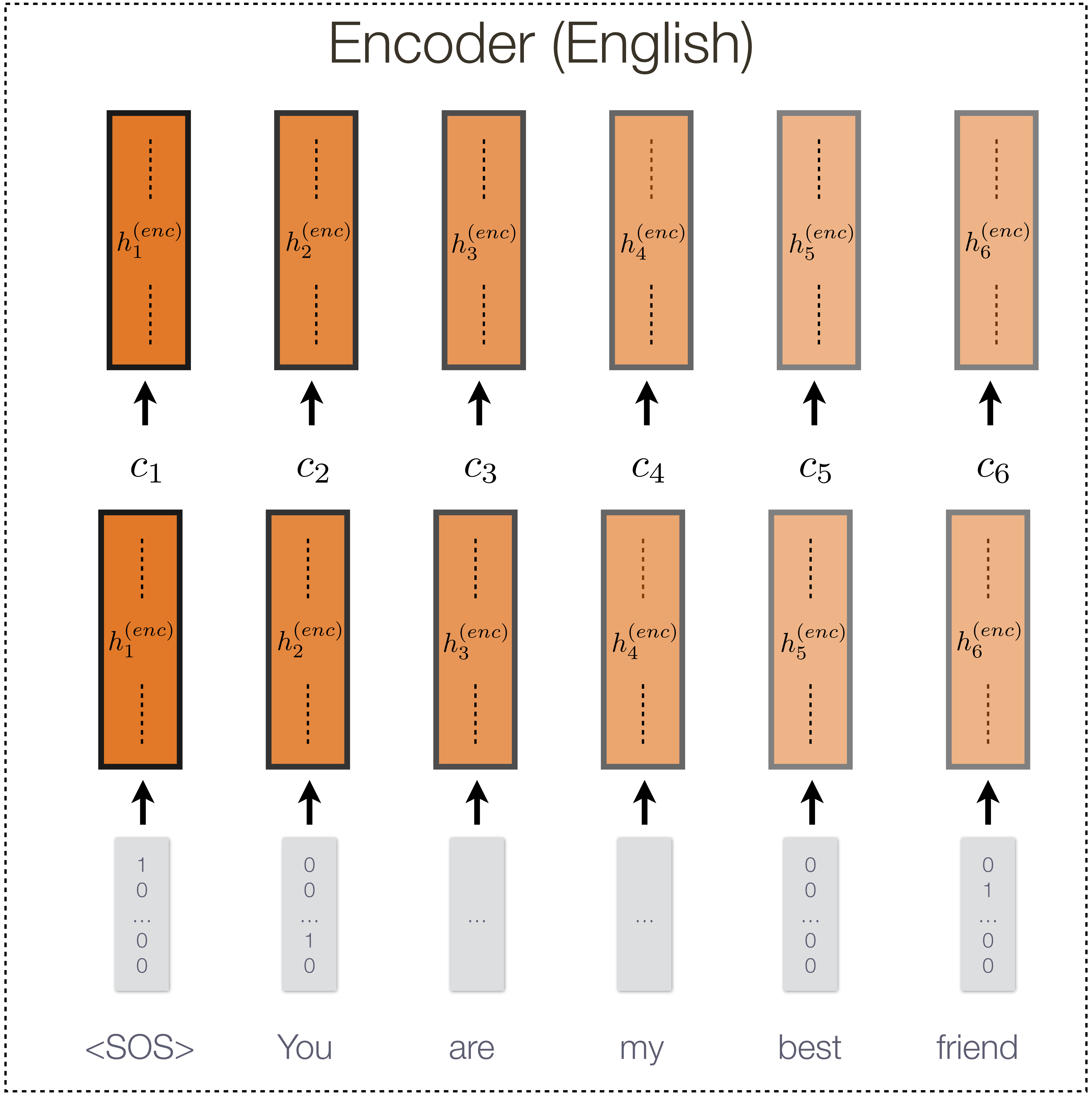


https://tzuruey.medium.com/attention-is-all-you-need-98d26aeb3517

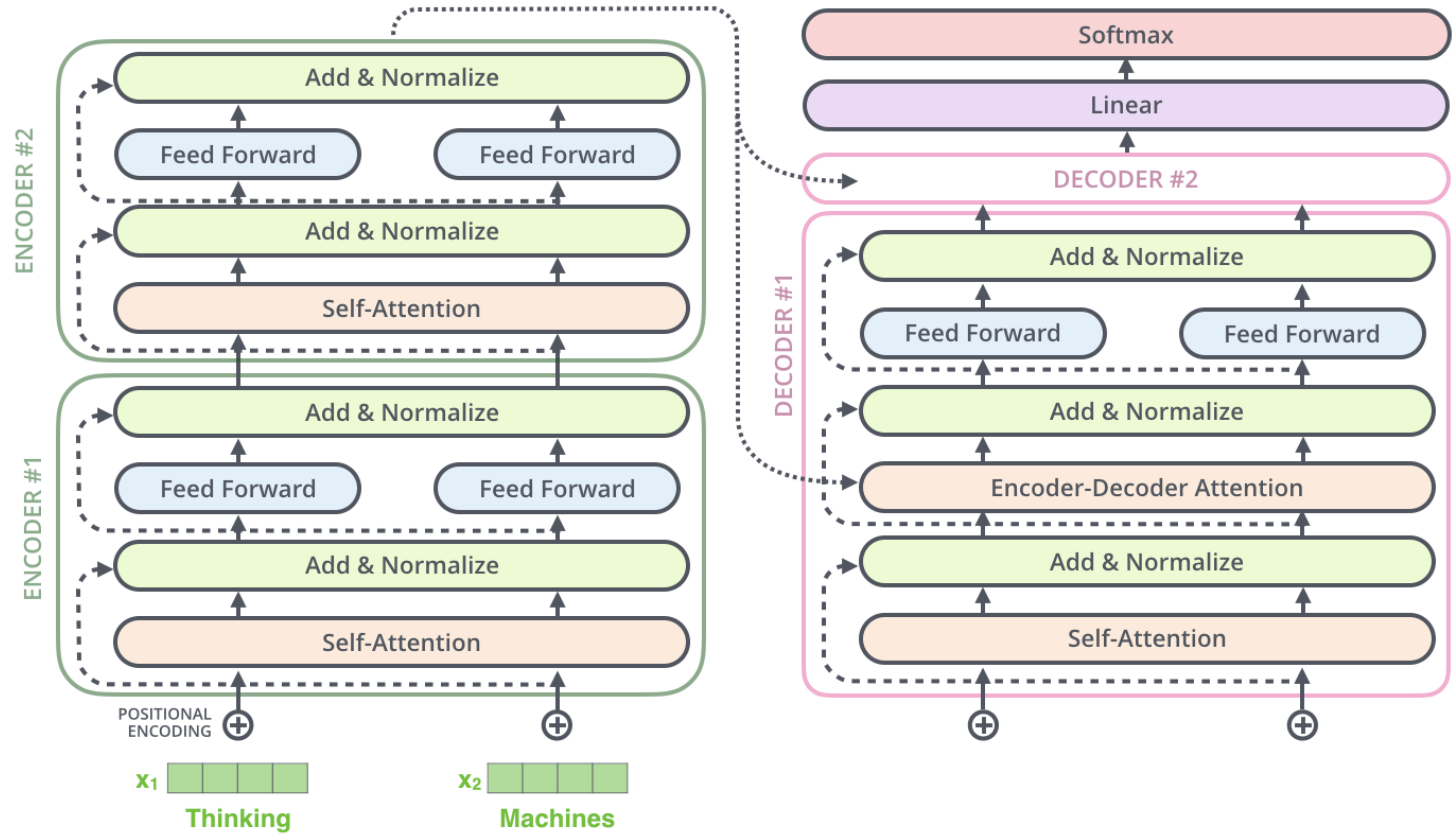
Self Attention



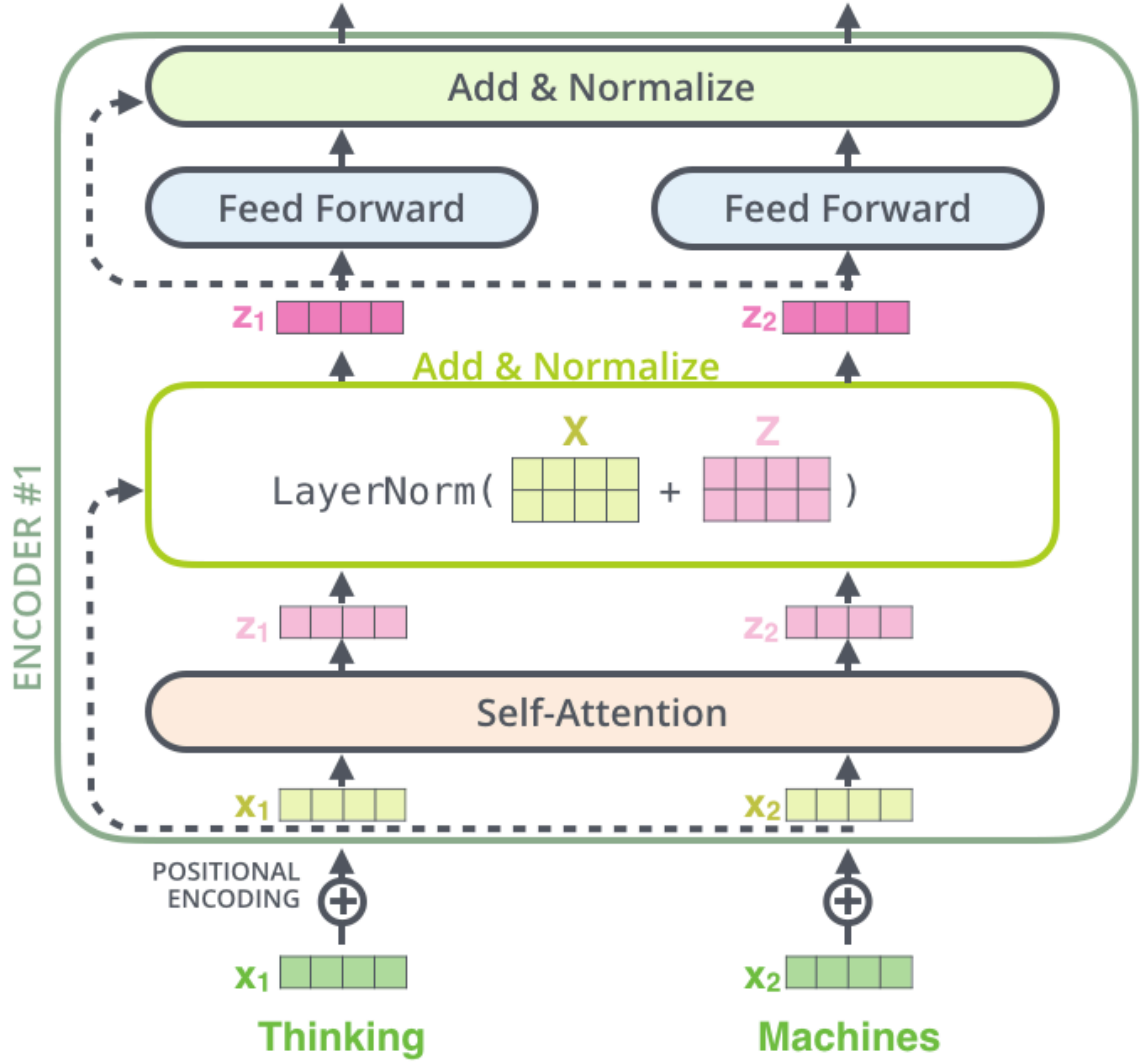
Self Attention



Transformers: Attention is all you need

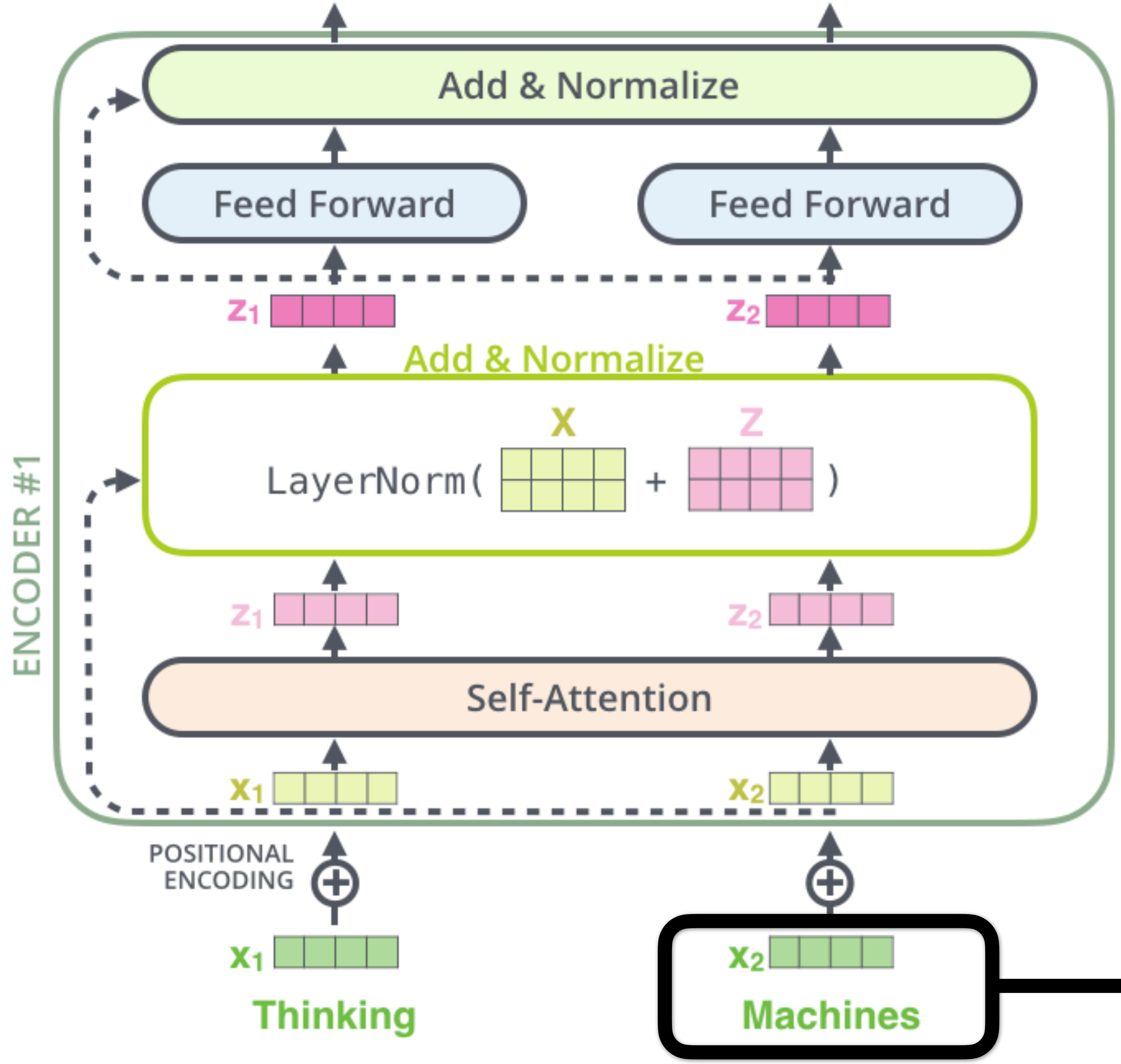


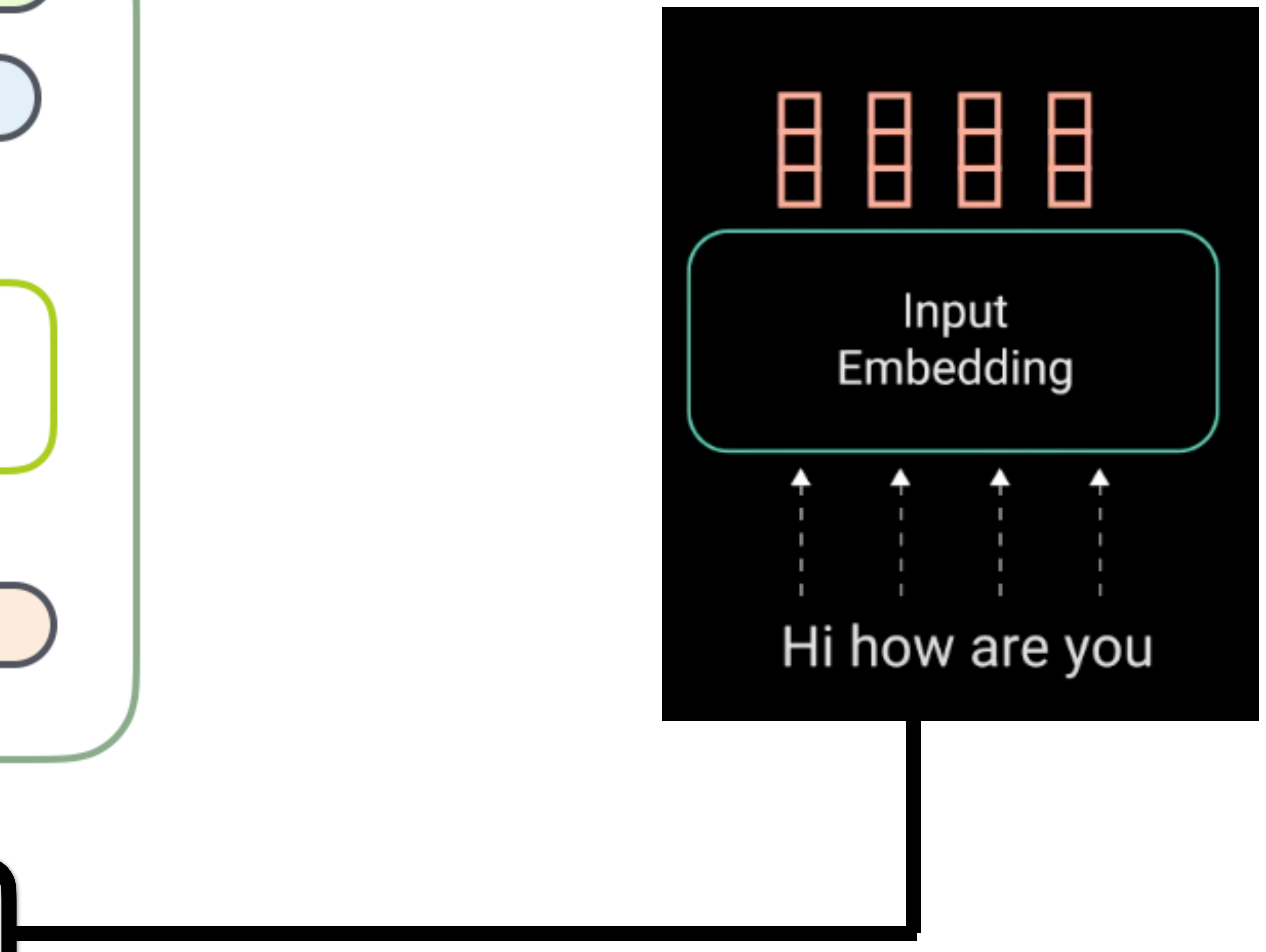
Transformers: Attention is all you need (Encoder)



Note: for assignment you are <u>not</u> implementing transformer encoder

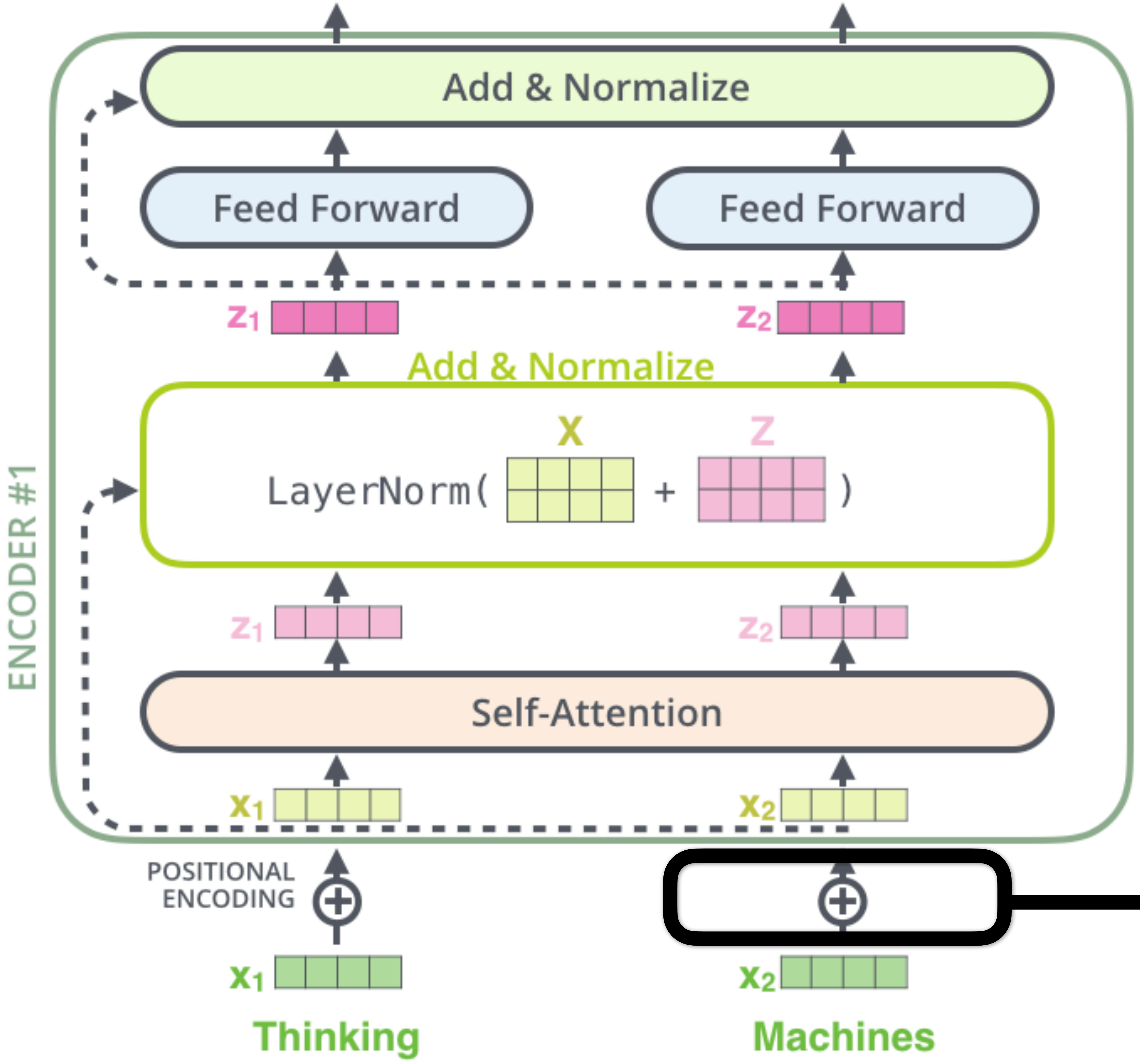
Transformers: Attention is all you need (Encoder)

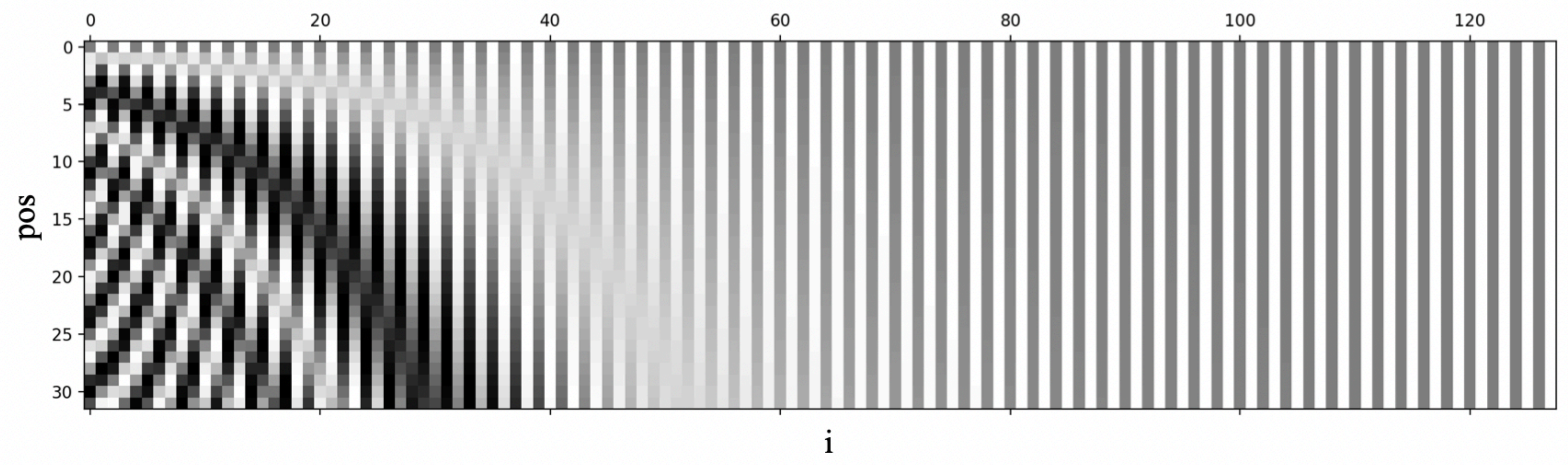




Note: for assignment you are not implementing transformer encoder

Transformers: Attention is all you need (Encoder)



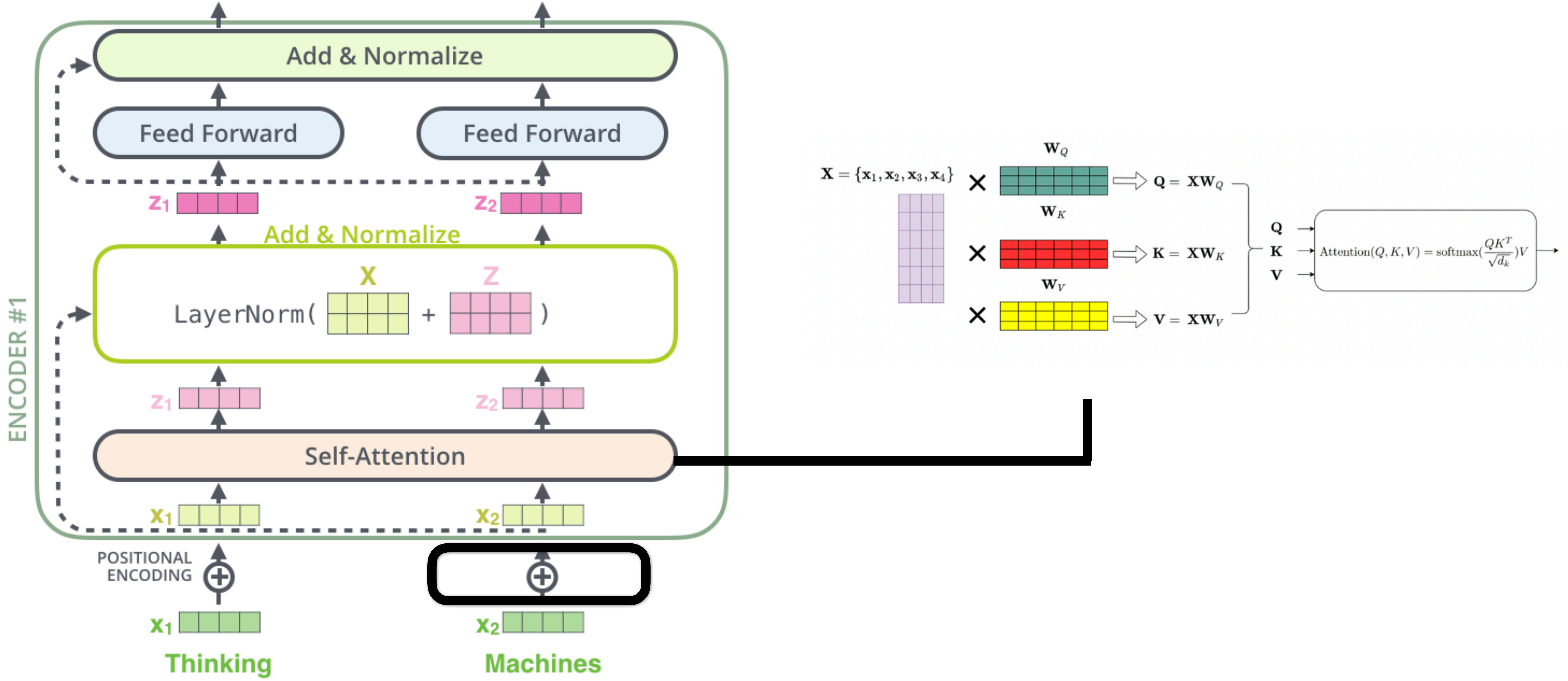


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

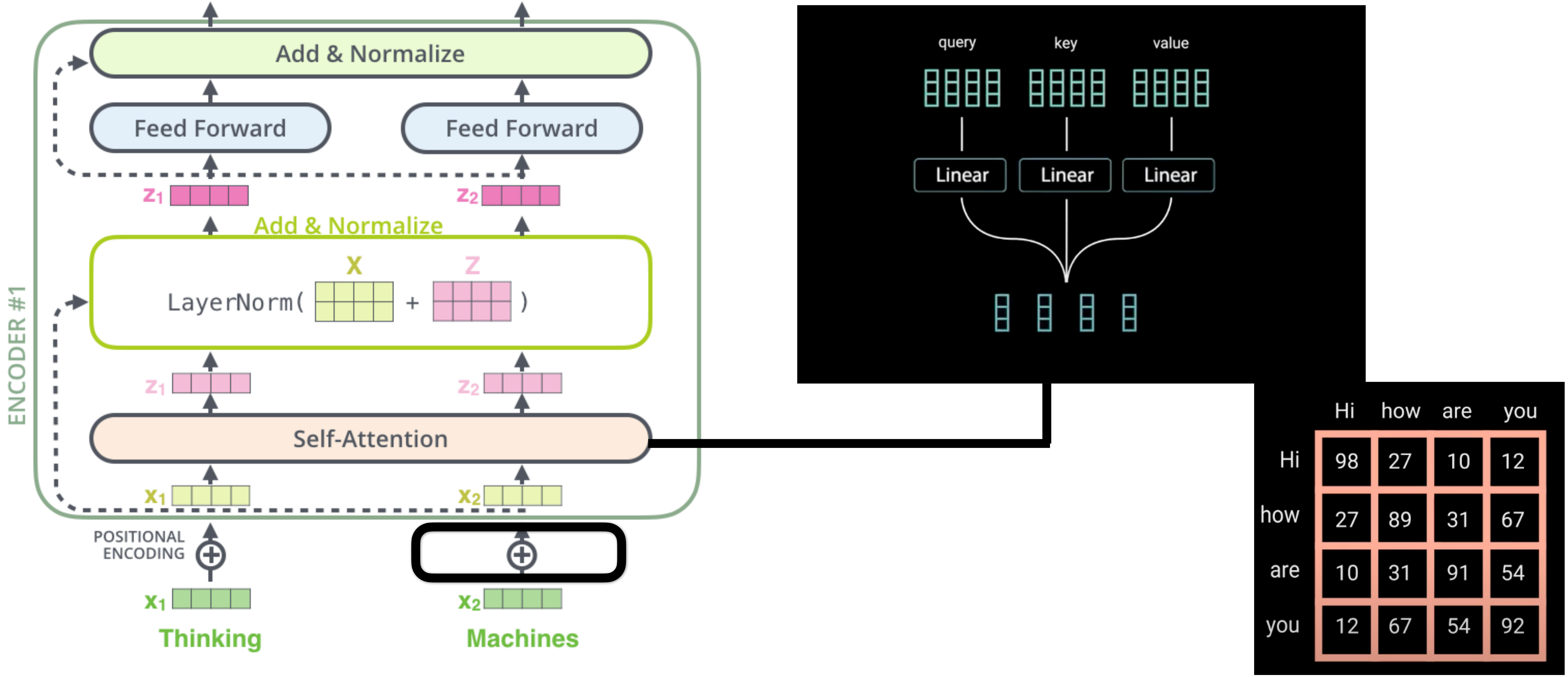
Note: for assignment you are <u>not</u> implementing transformer encoder

Transformers: Attention is all you need (Encoder)



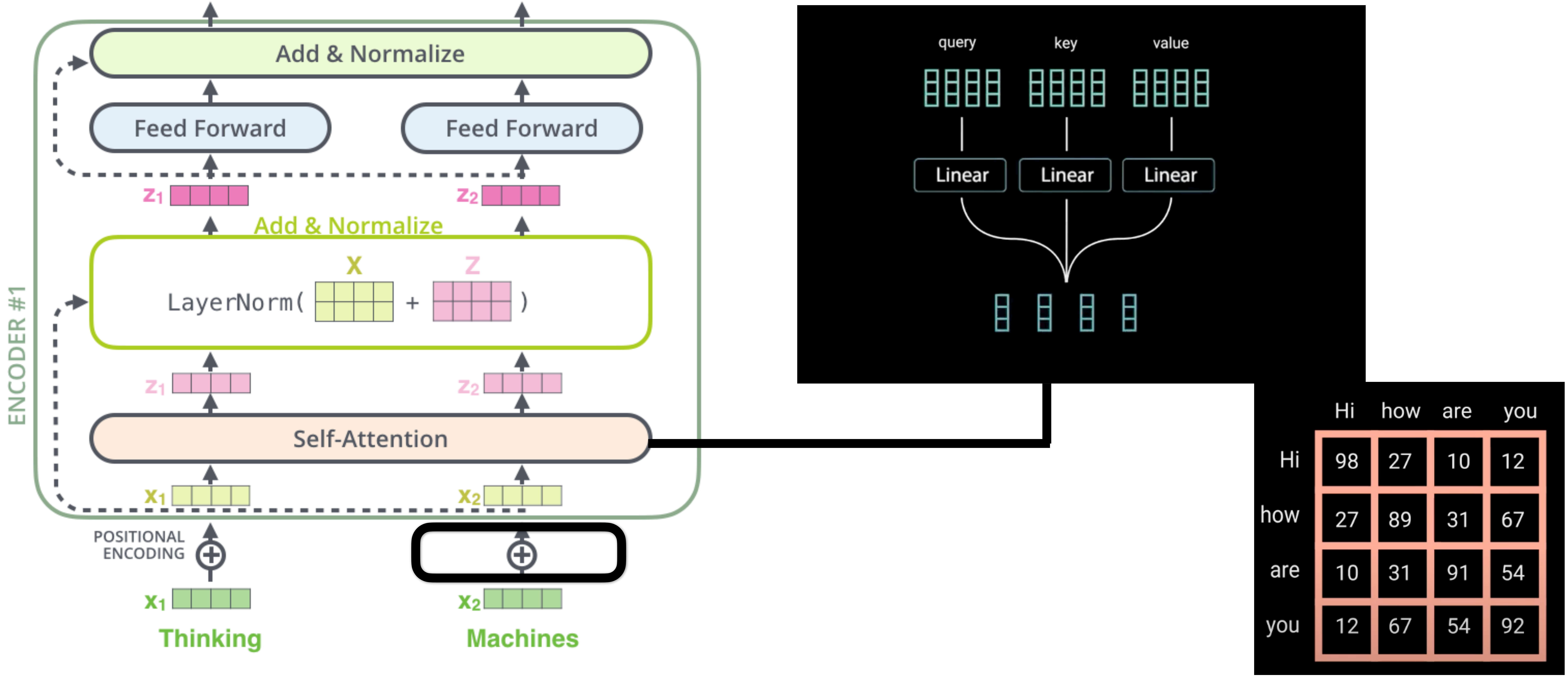
Note: for assignment you are <u>not</u> implementing transformer encoder

Transformers: Attention is all you need (Encoder)



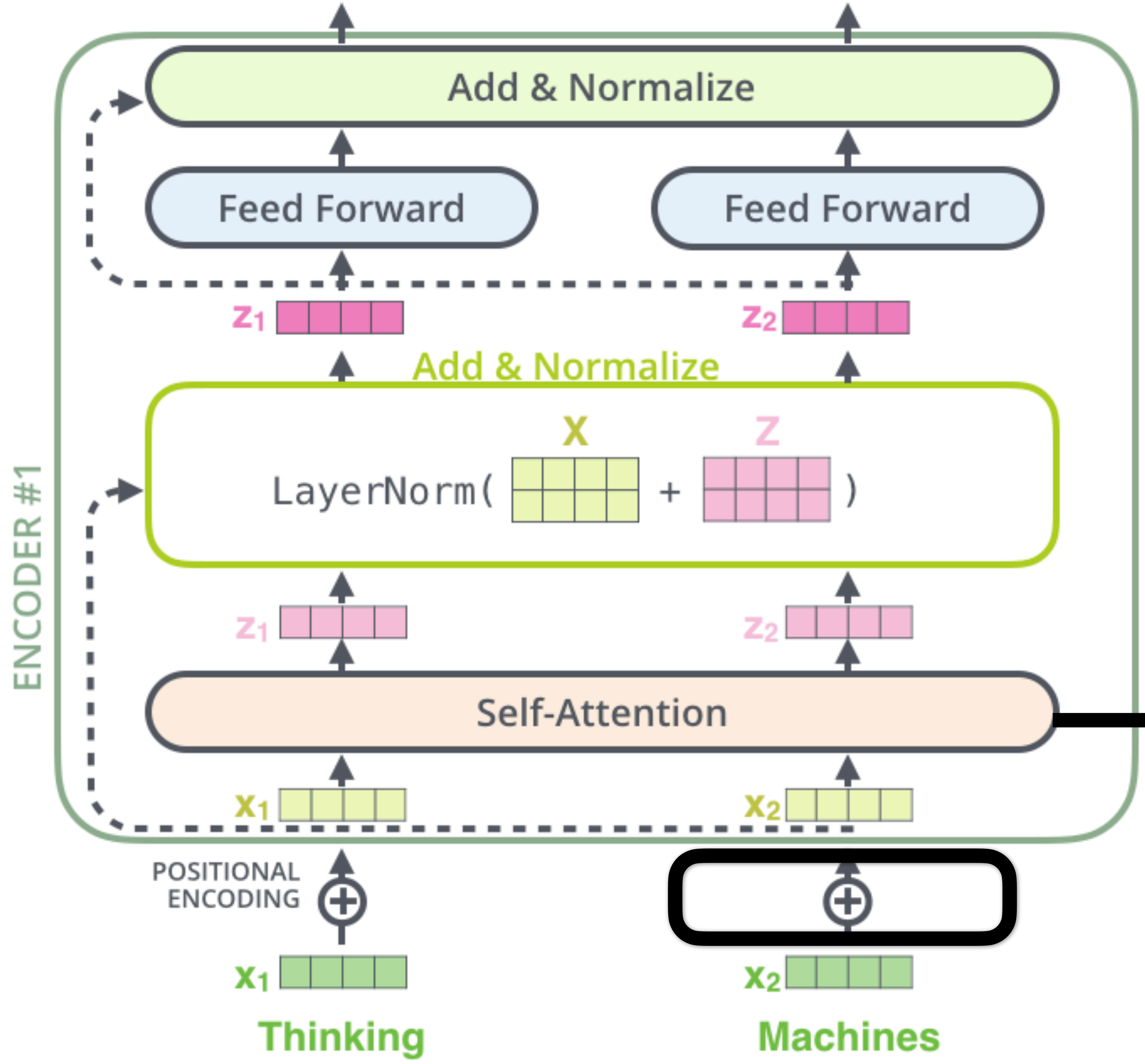
Note: for assignment you are <u>not</u> implementing transformer encoder

Transformers: Attention is all you need (Encoder)



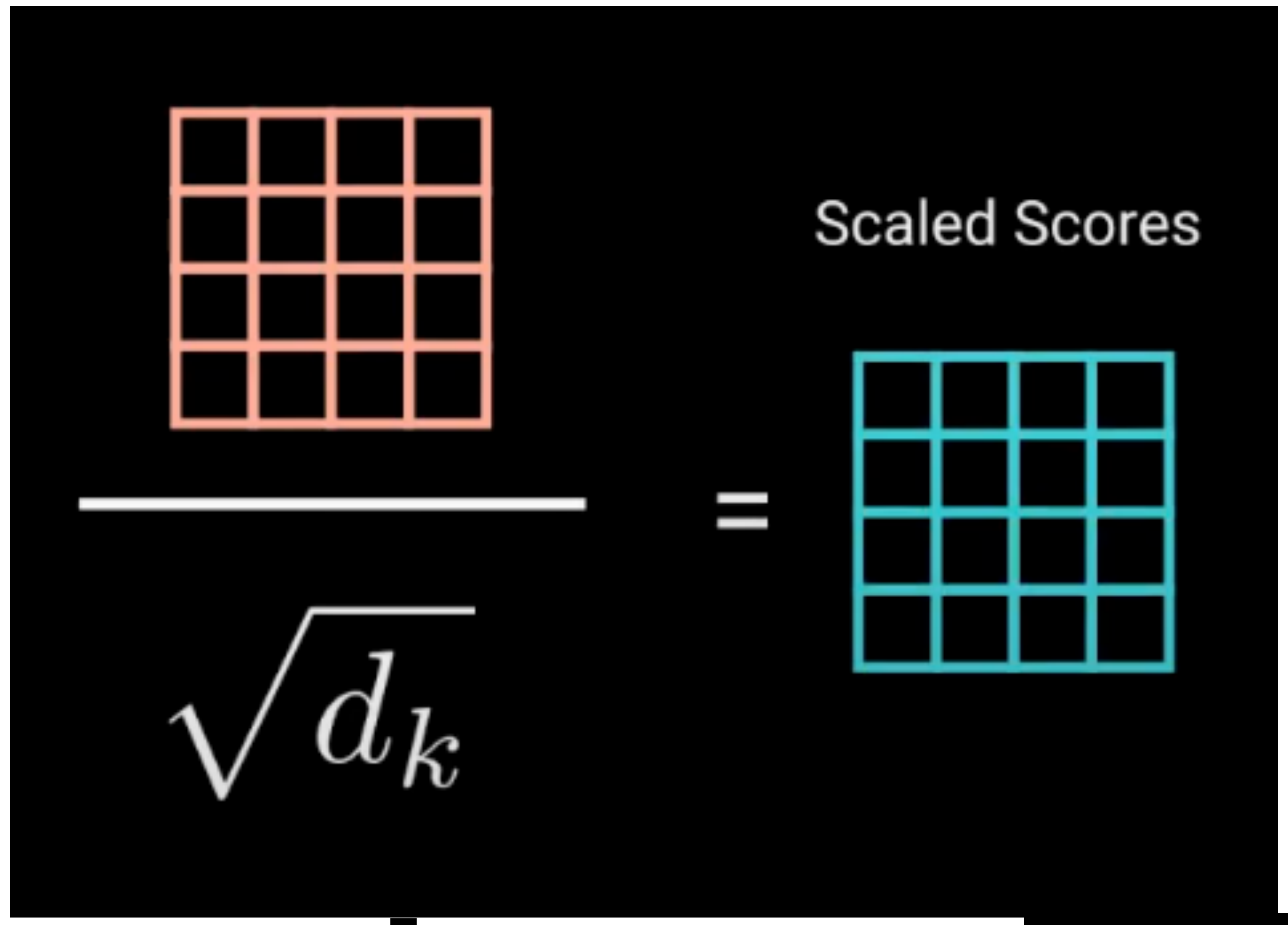
Note: for assignment you are not implementing transformer encoder

Transformers: Attention is all you need (Encoder)



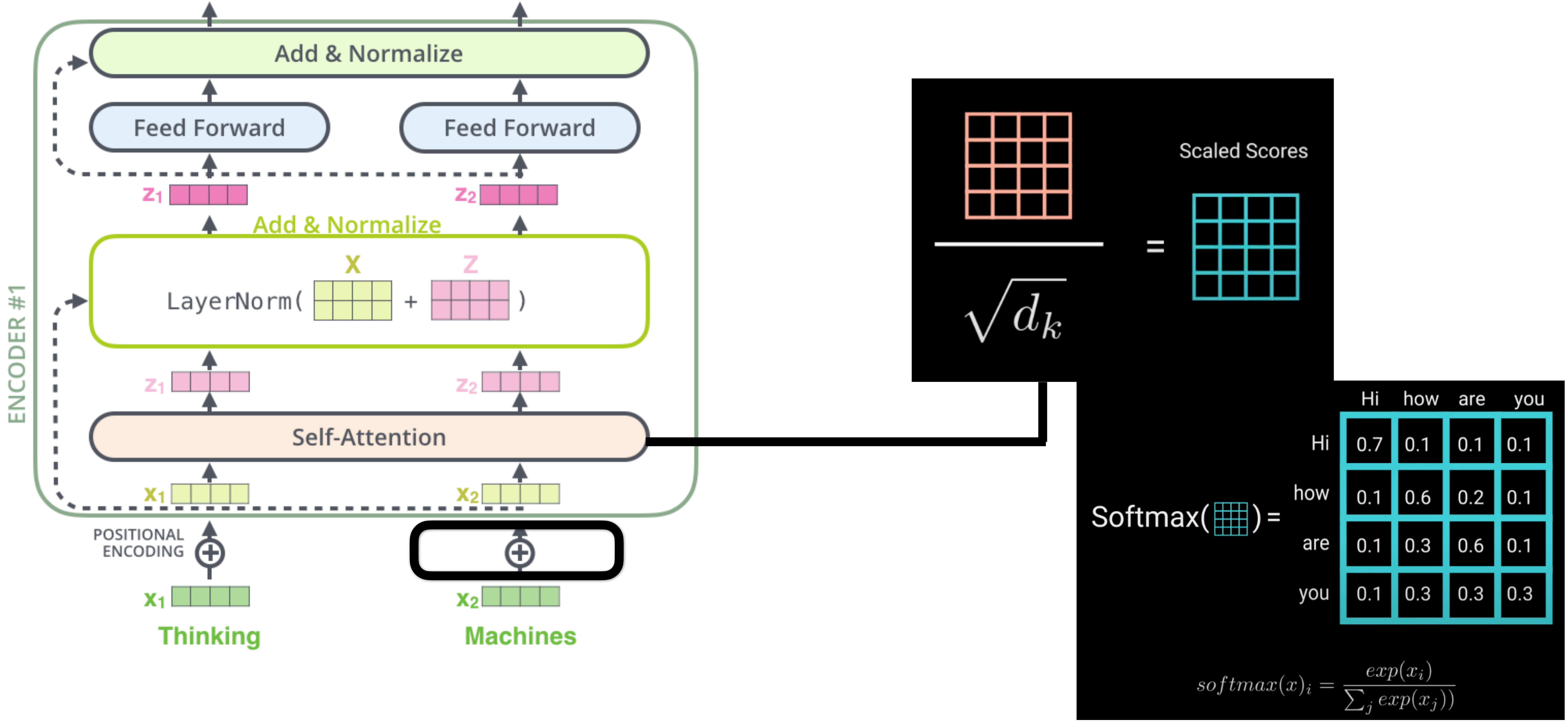
Normalize by sqrt of dimensionality

(leads to more stable gradients)

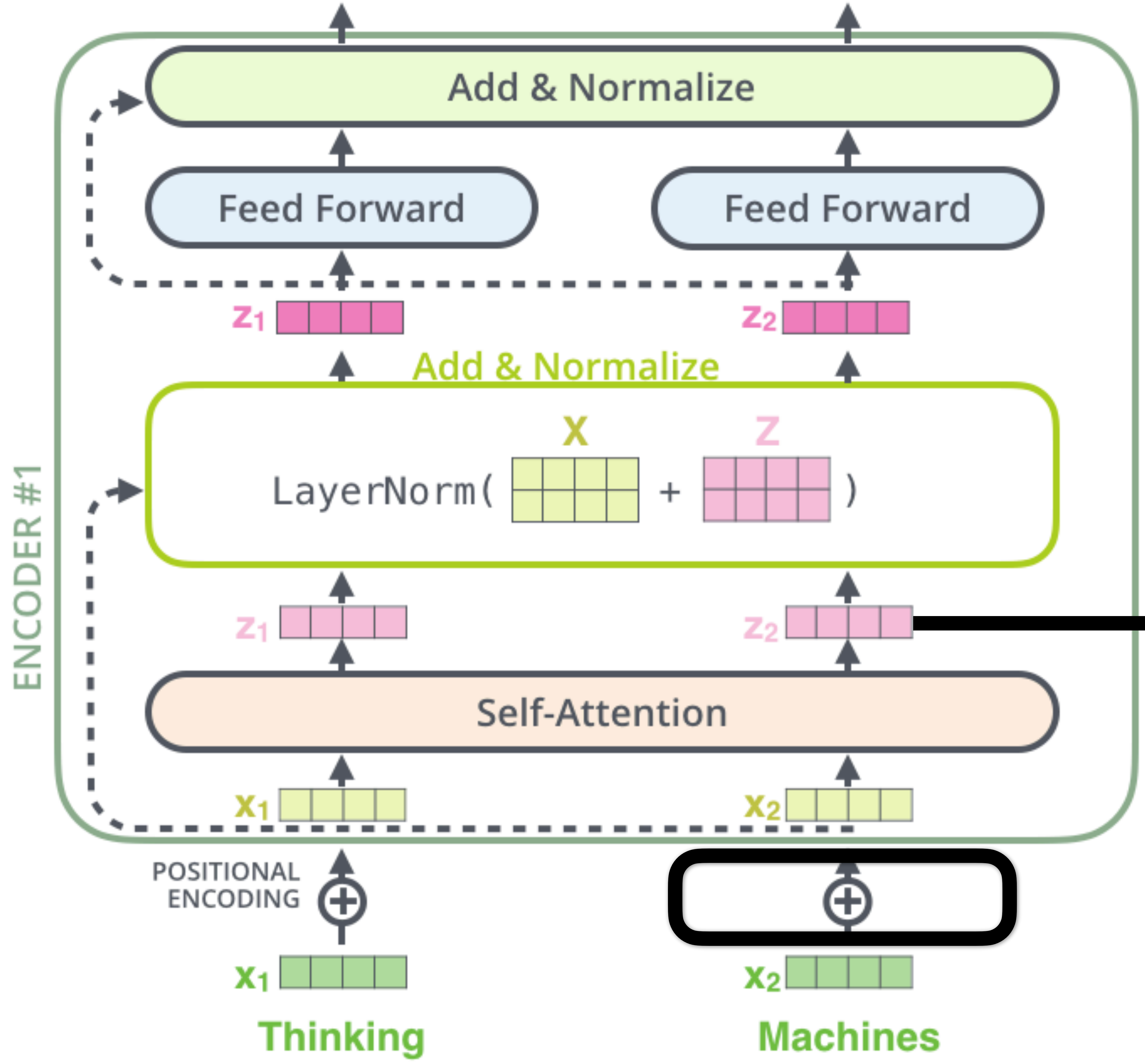


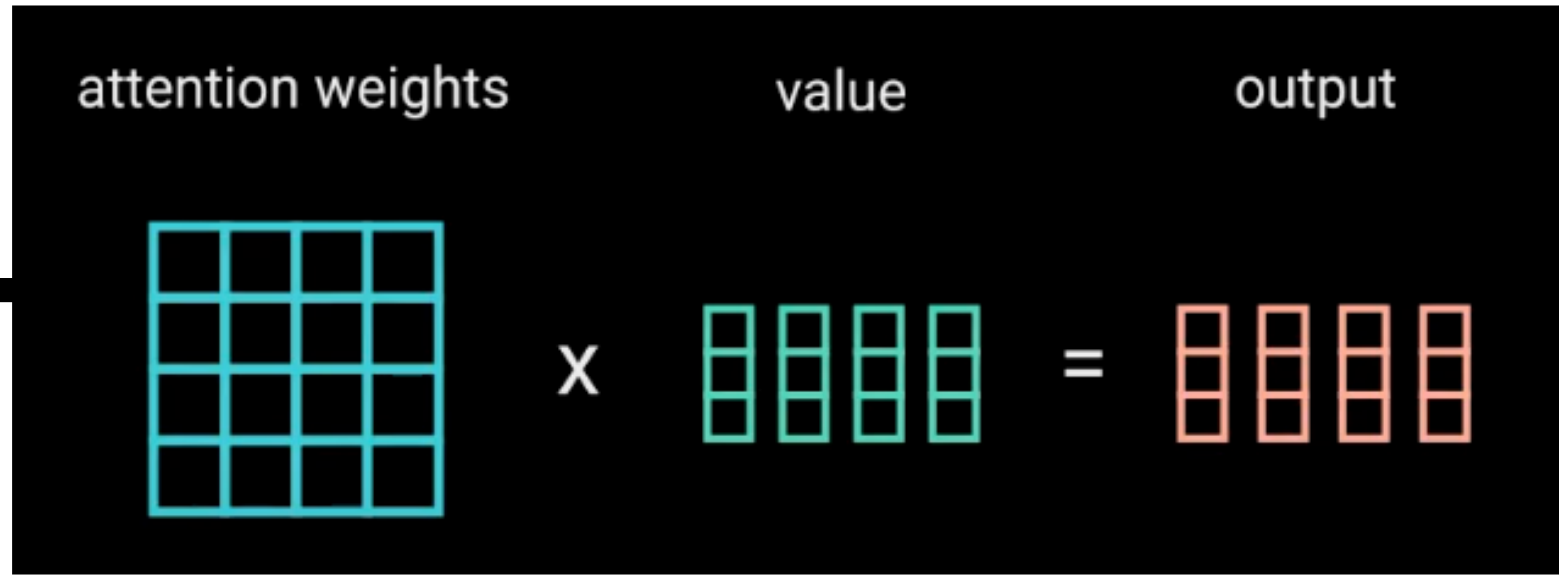
	Hi	how	are	you
Hi	98	27	10	12
how	27	89	31	67
are	10	31	91	54
you	12	67	54	92

Transformers: Attention is all you need (Encoder)

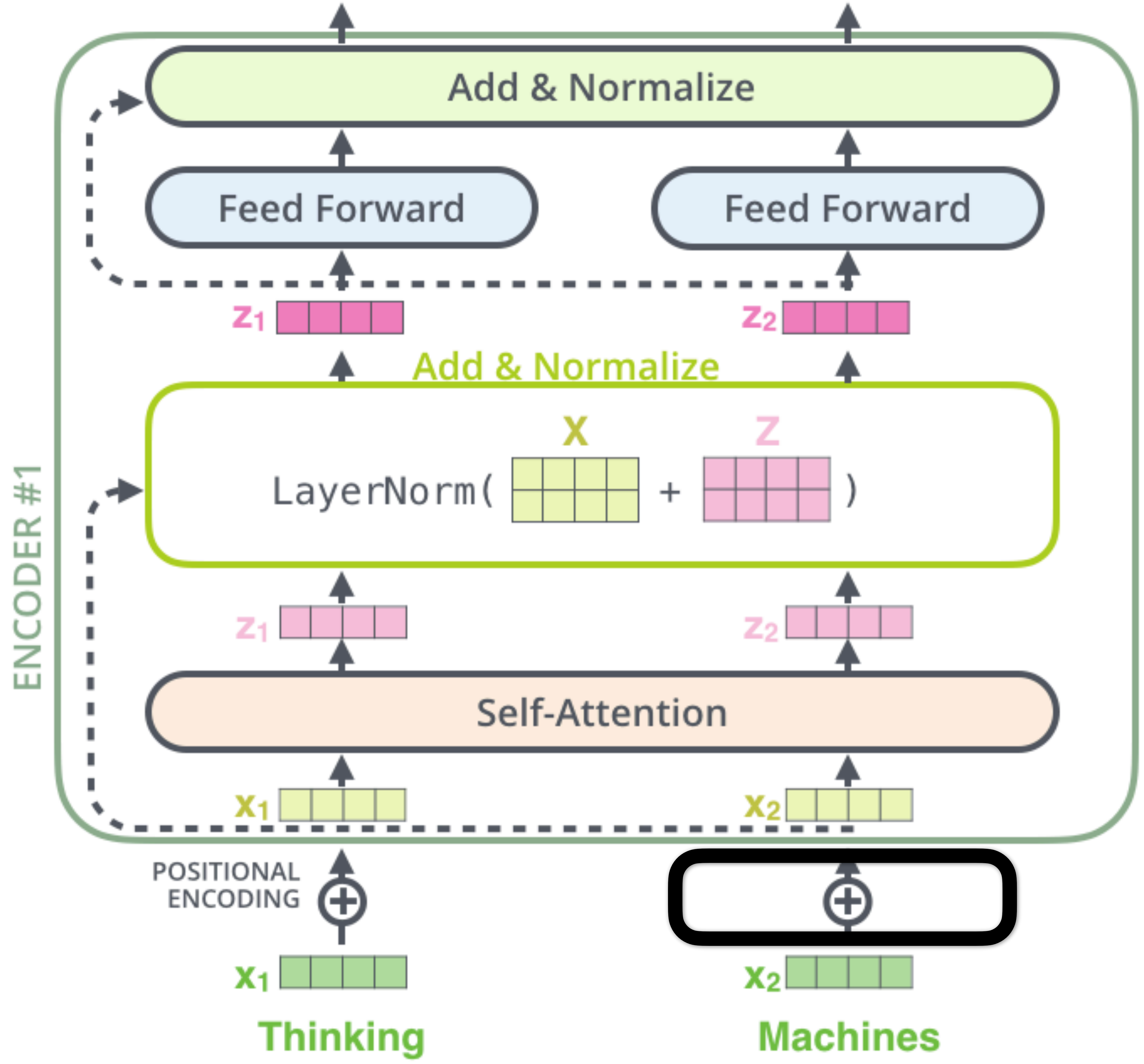


Transformers: Attention is all you need (Encoder)

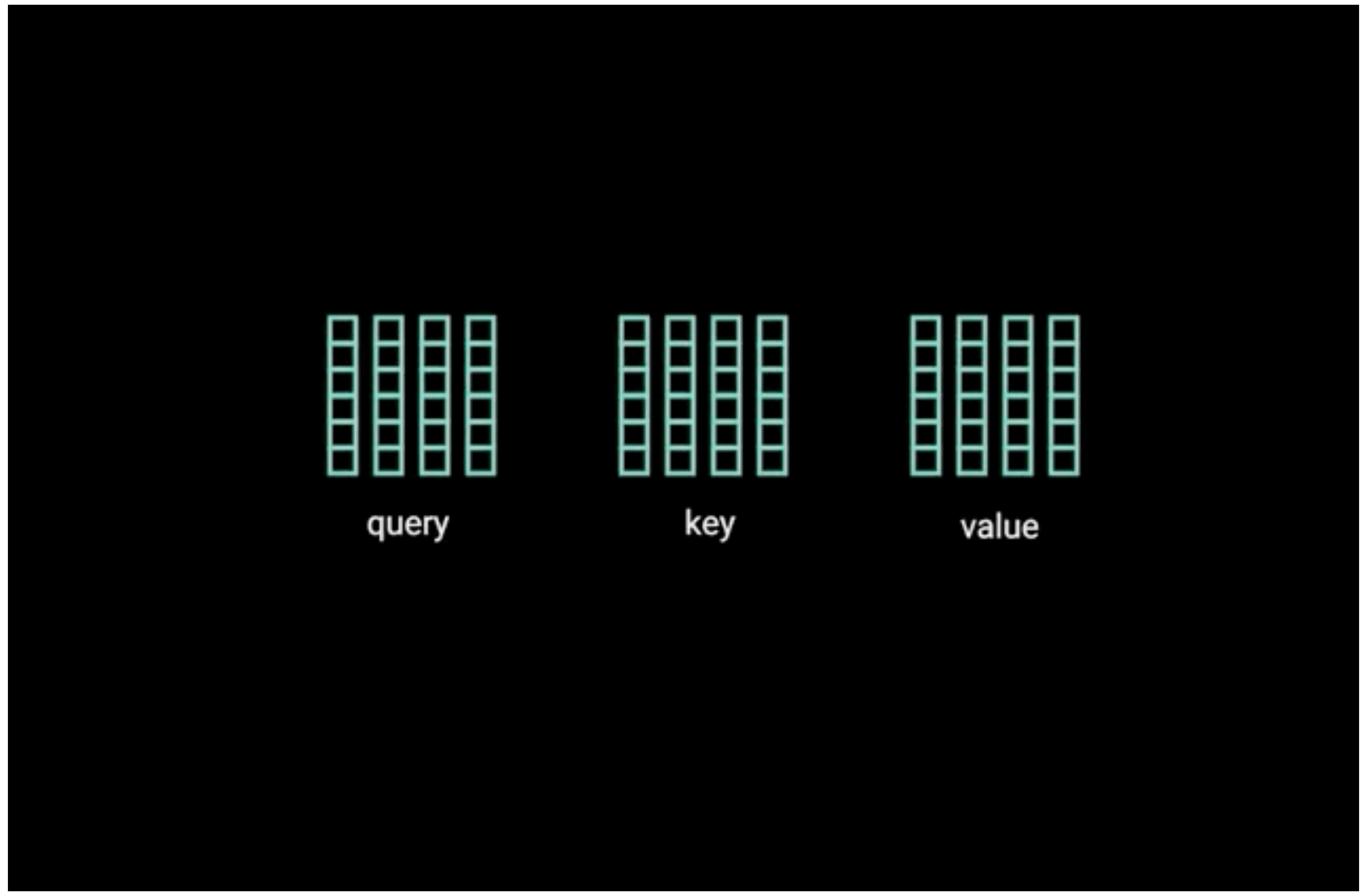




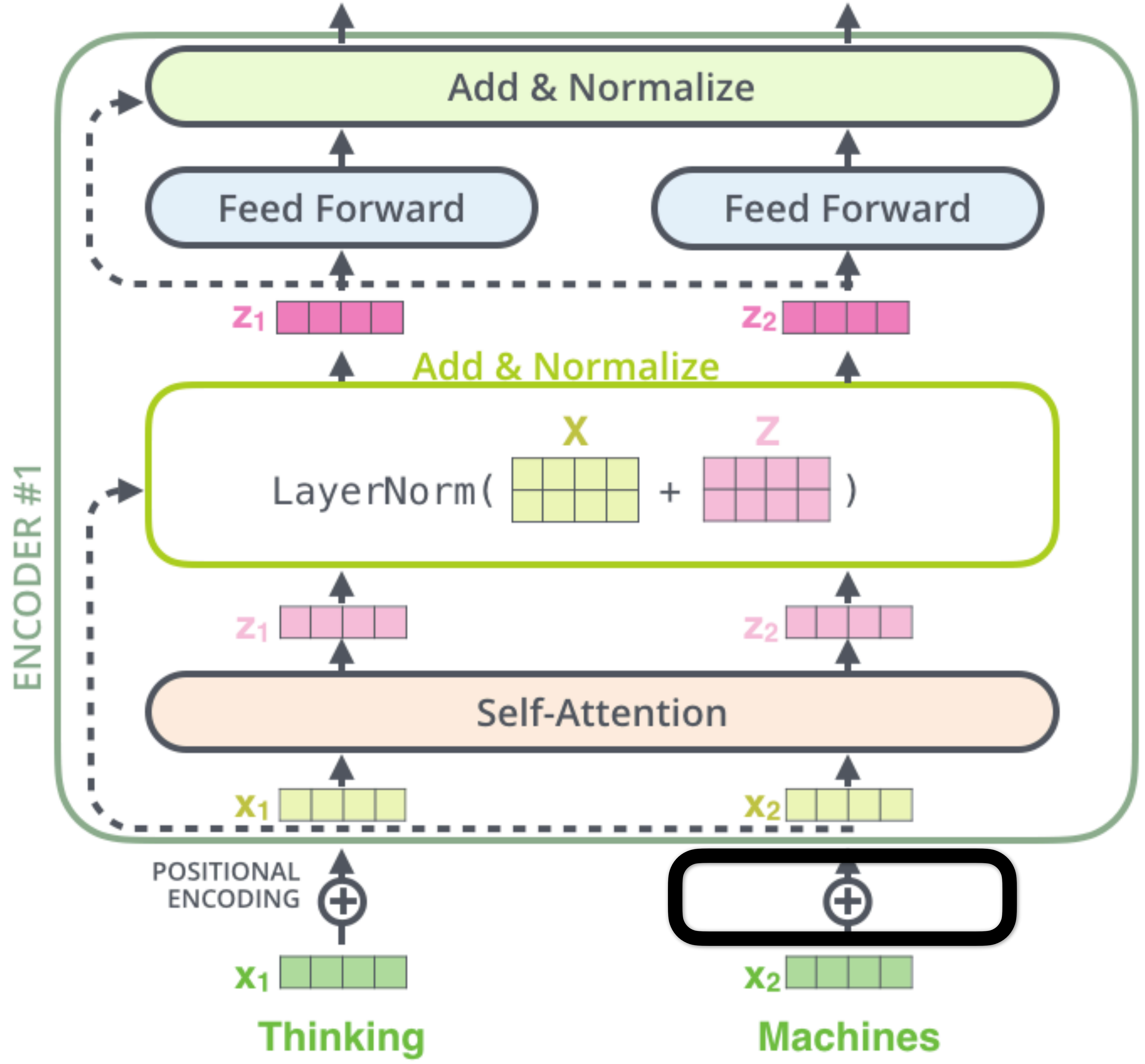
Transformers: Attention is all you need (Encoder)



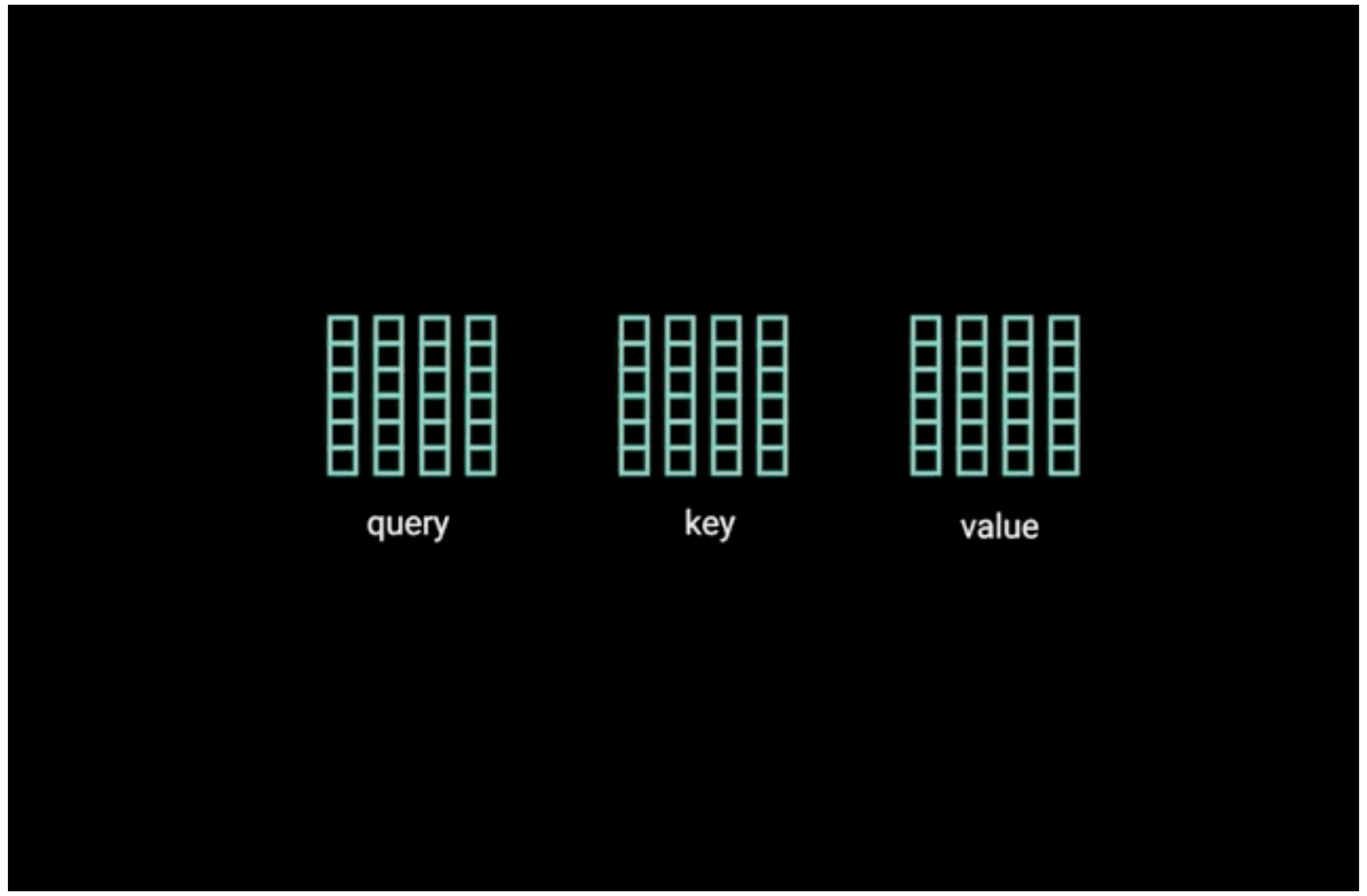
In practice, we use multiple self-attention heads



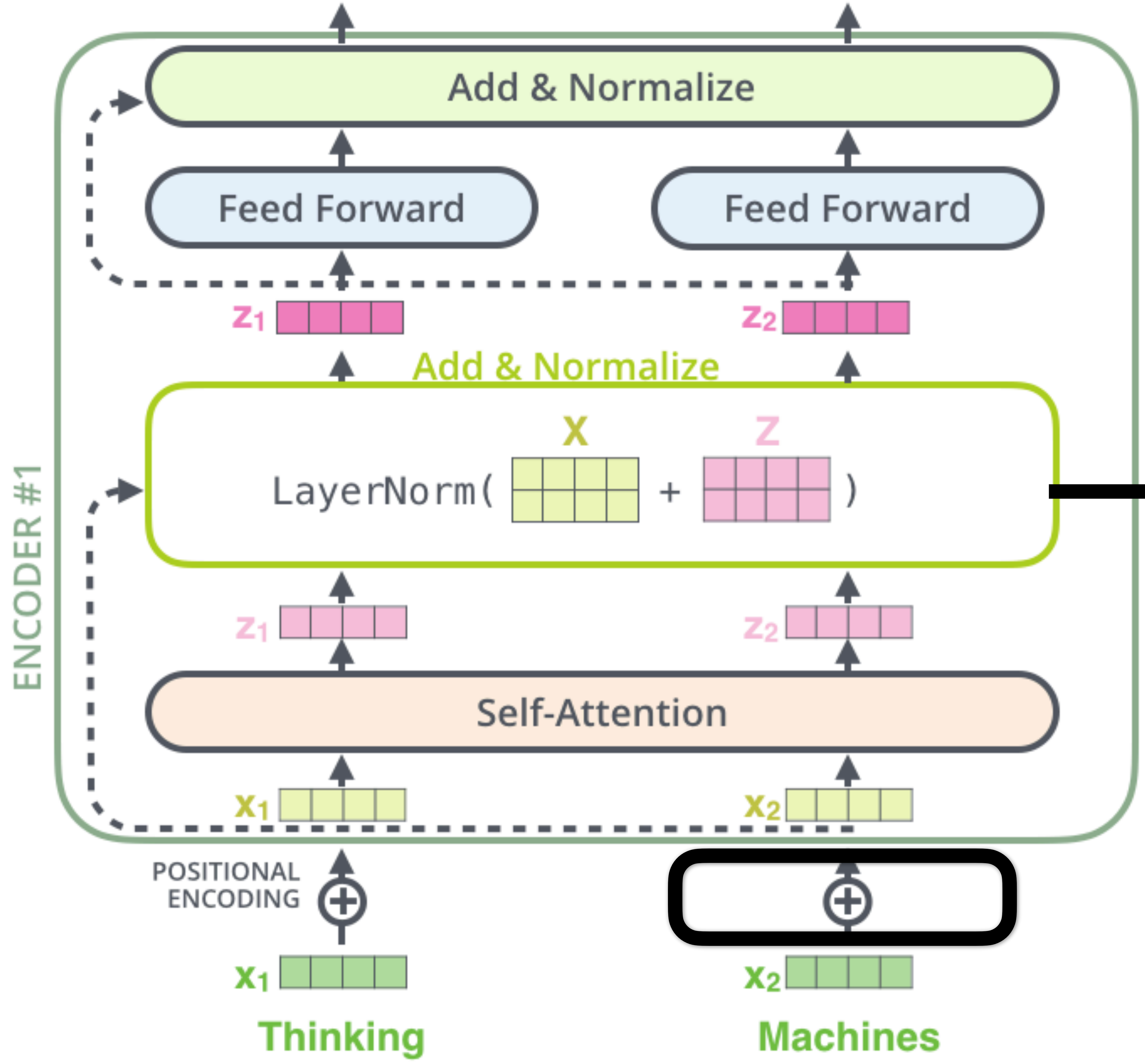
Transformers: Attention is all you need (Encoder)



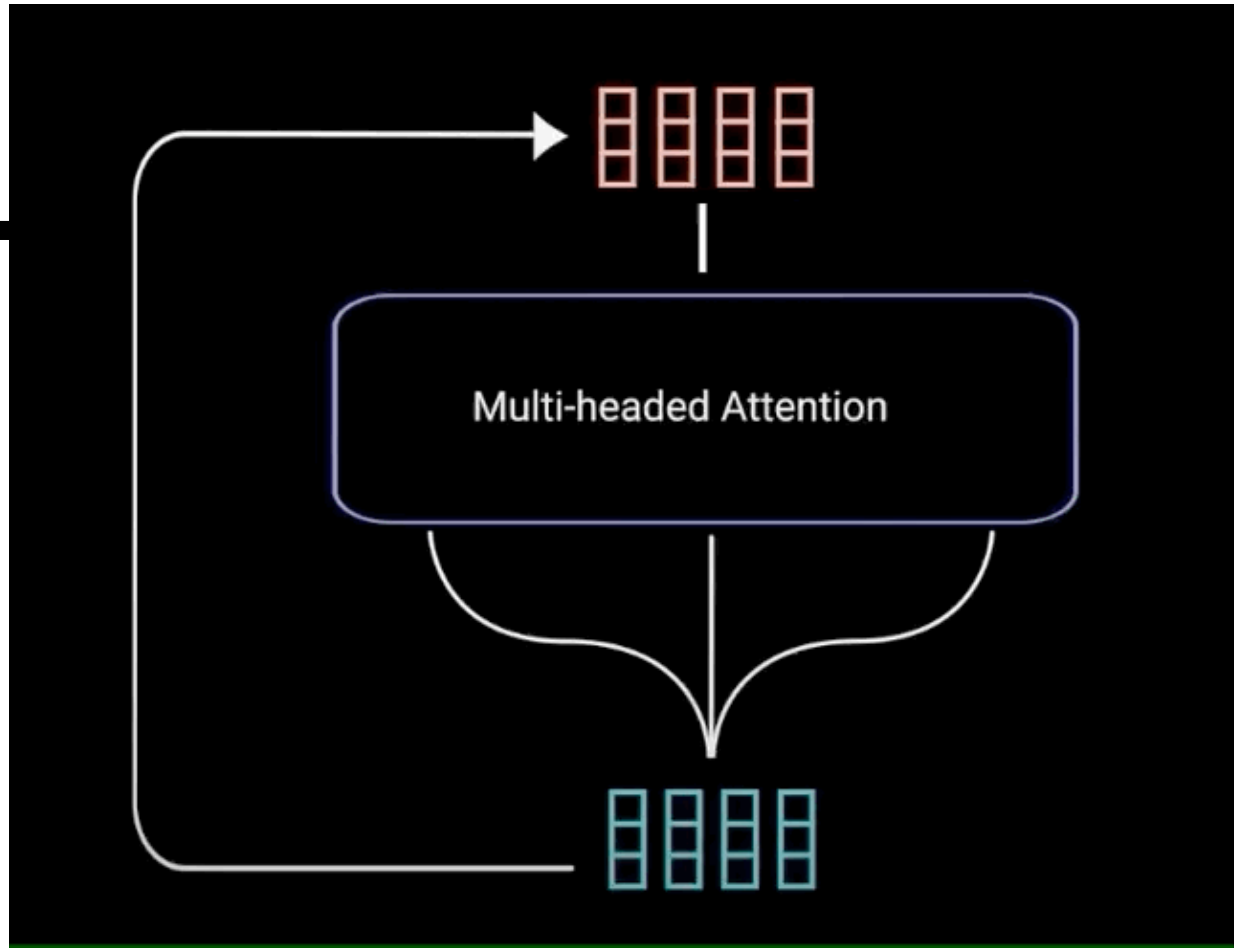
In practice, we use multiple self-attention heads



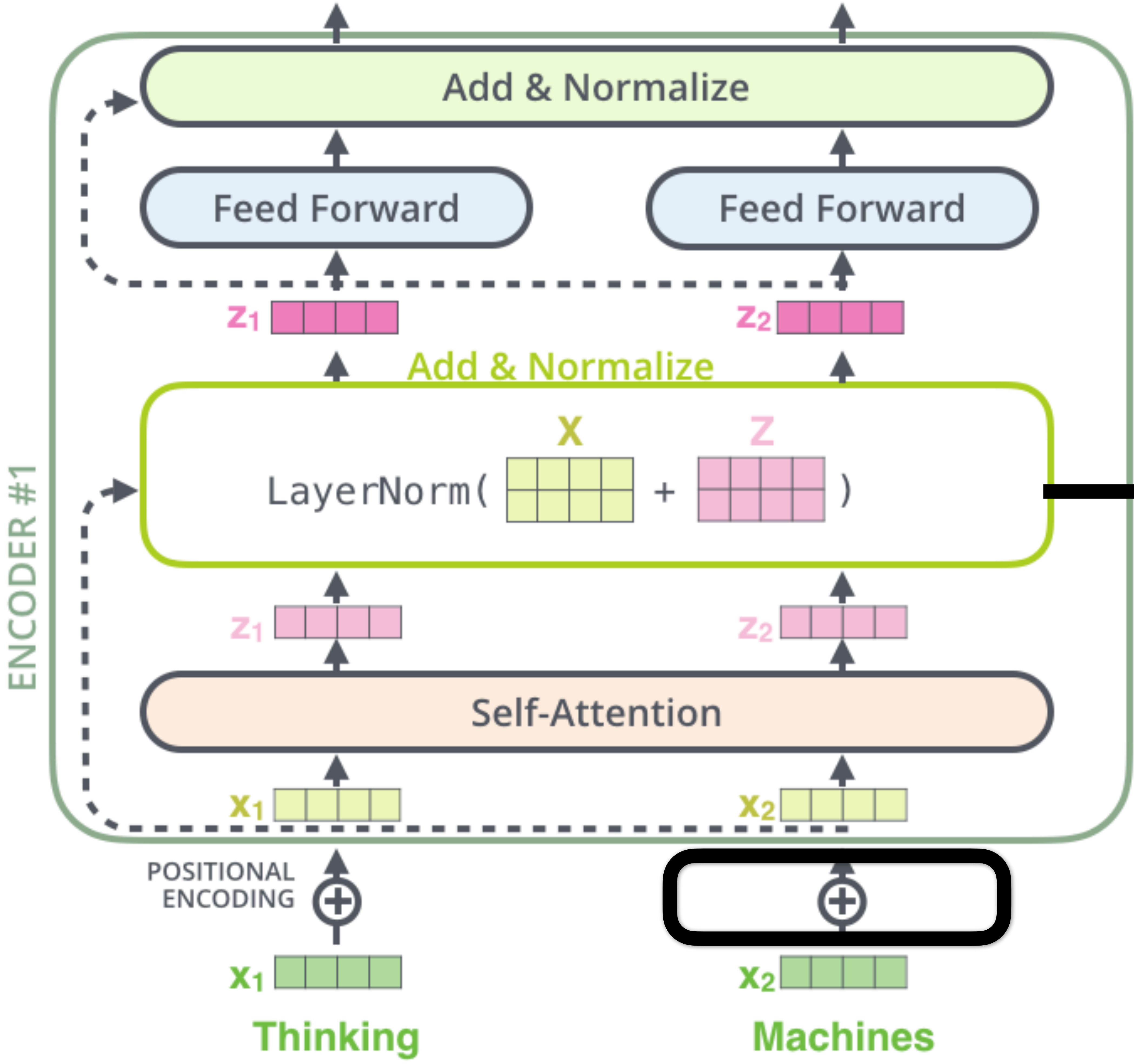
Transformers: Attention is all you need (Encoder)



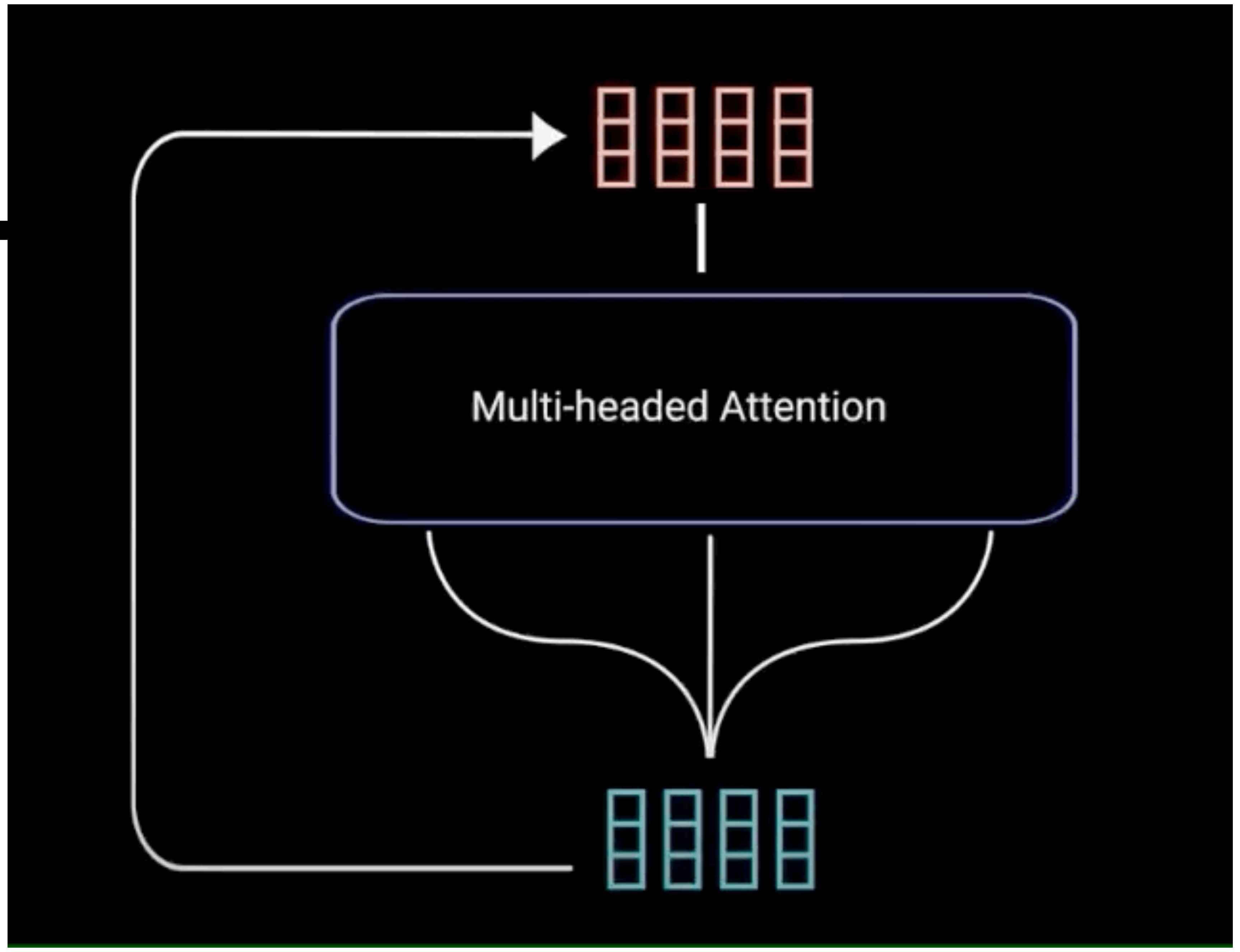
Residual connection with LayerNorm



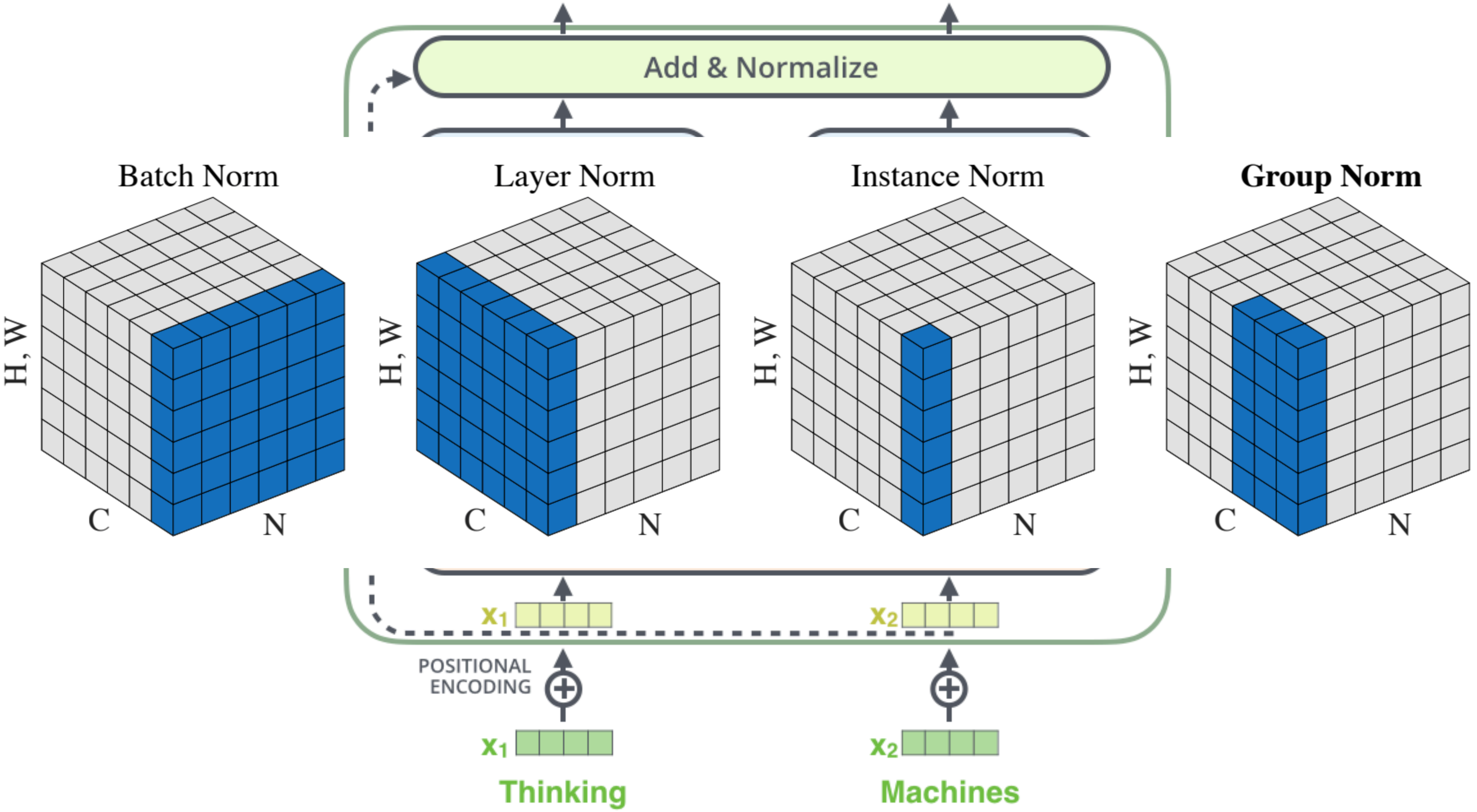
Transformers: Attention is all you need (Encoder)



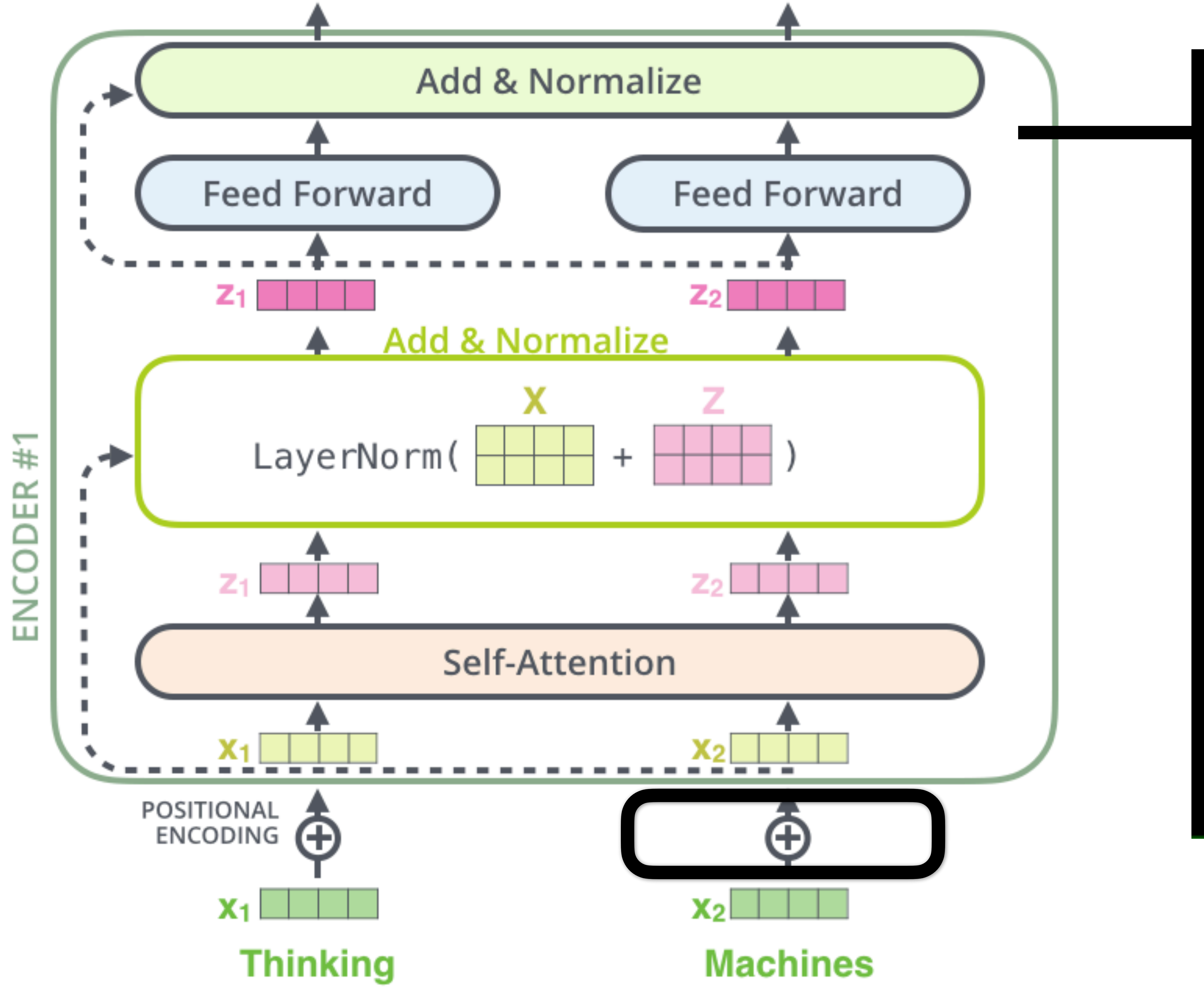
Residual connection with LayerNorm

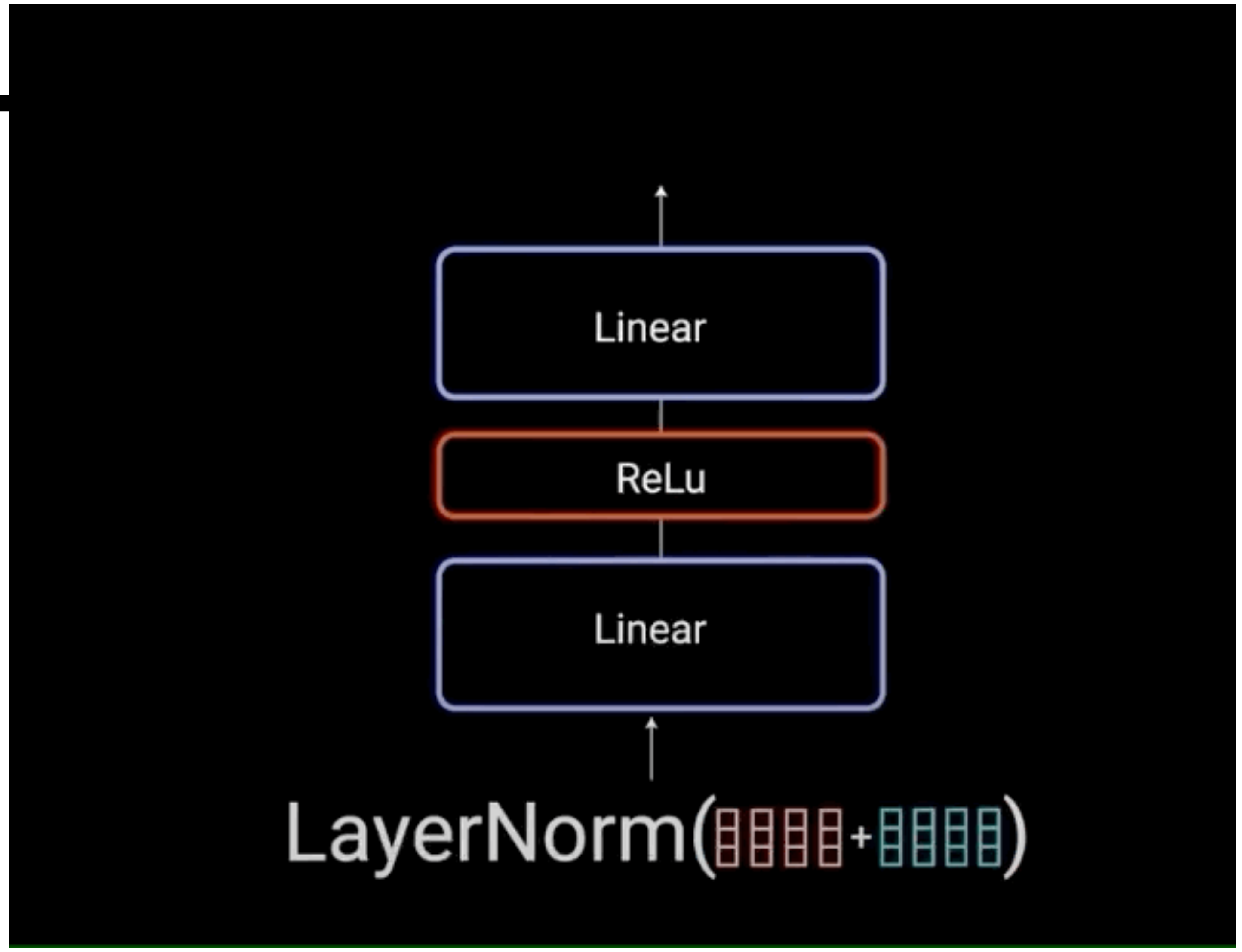


Transformers: Attention is all you need (Encoder)

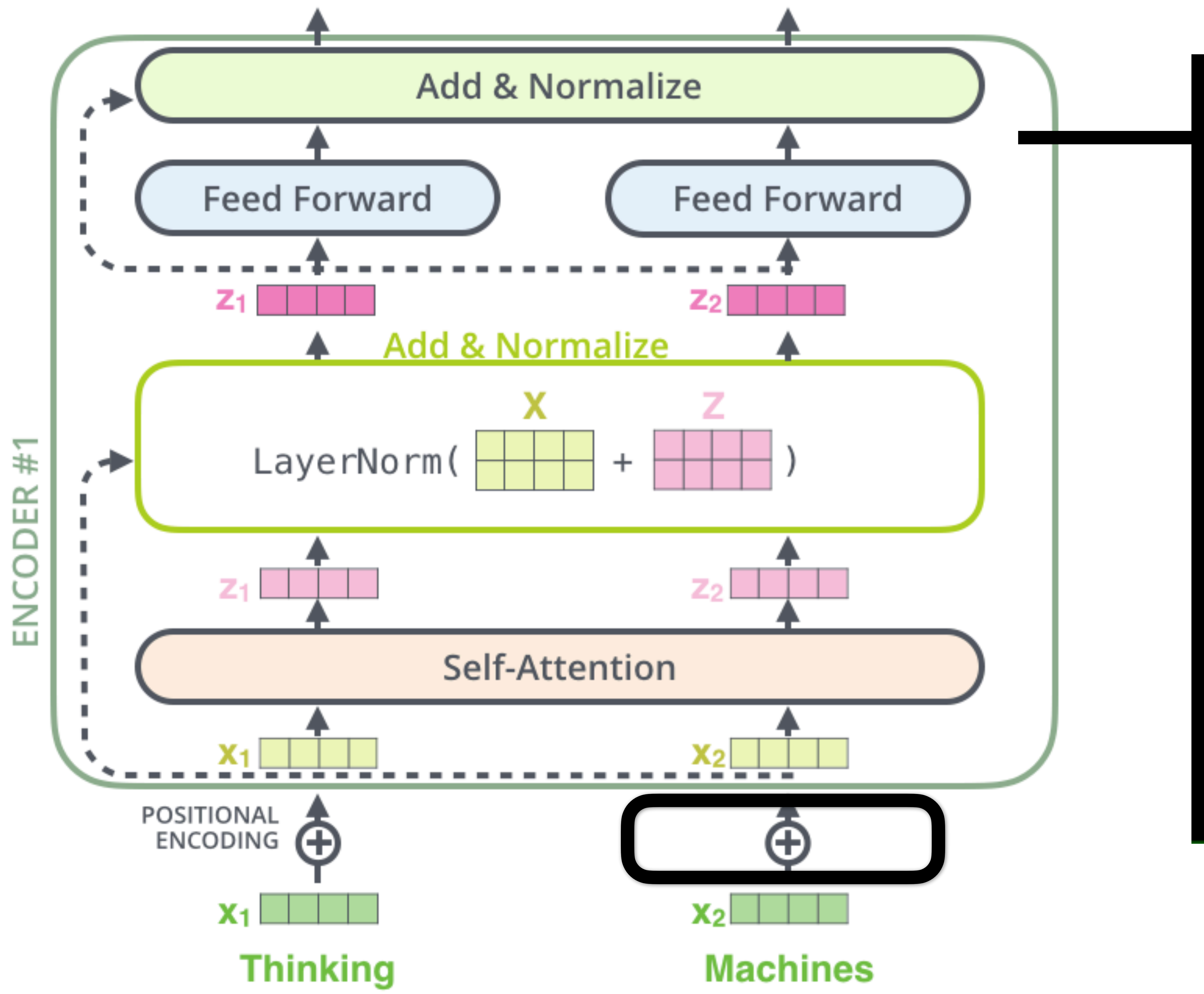


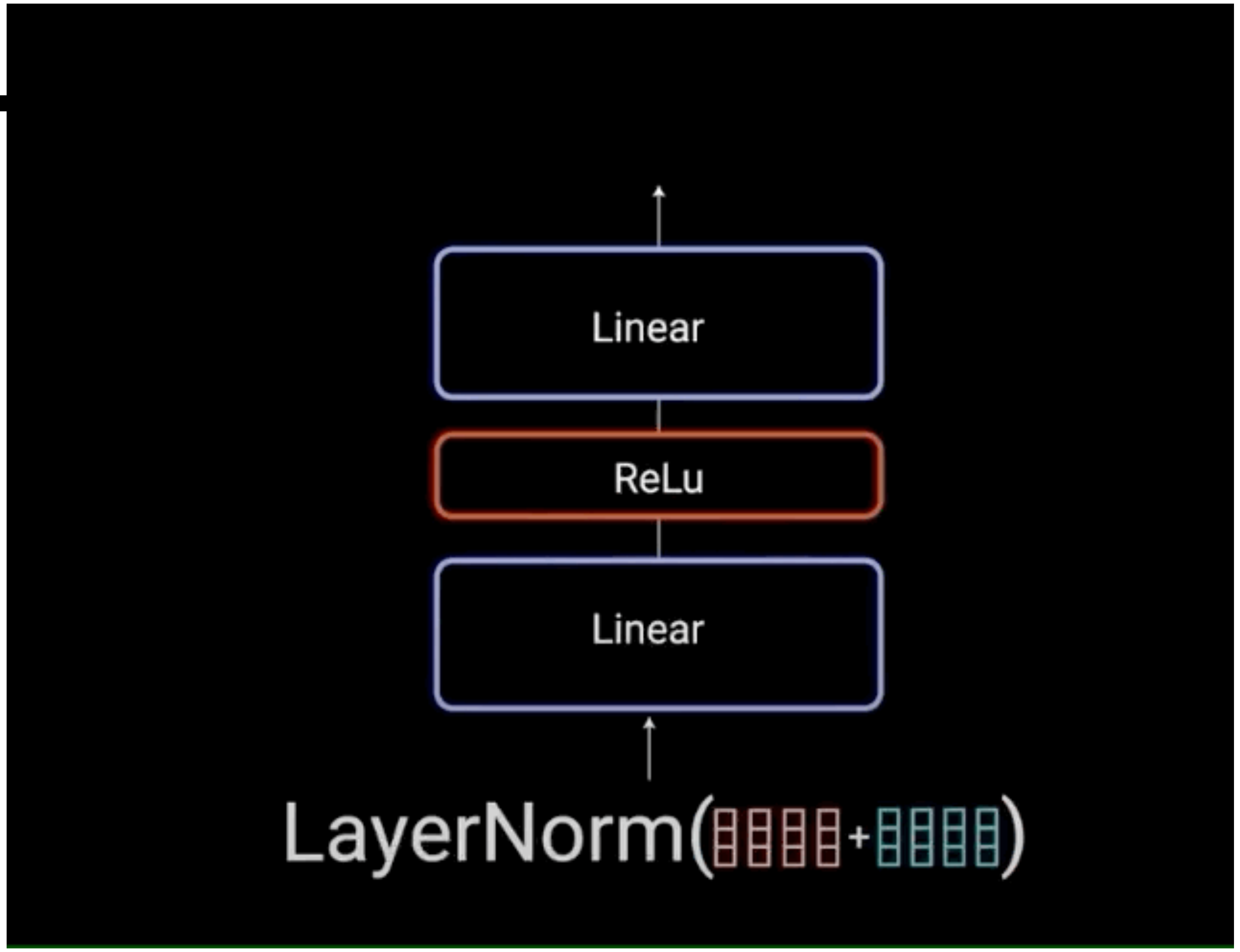
Transformers: Attention is all you need (Encoder)



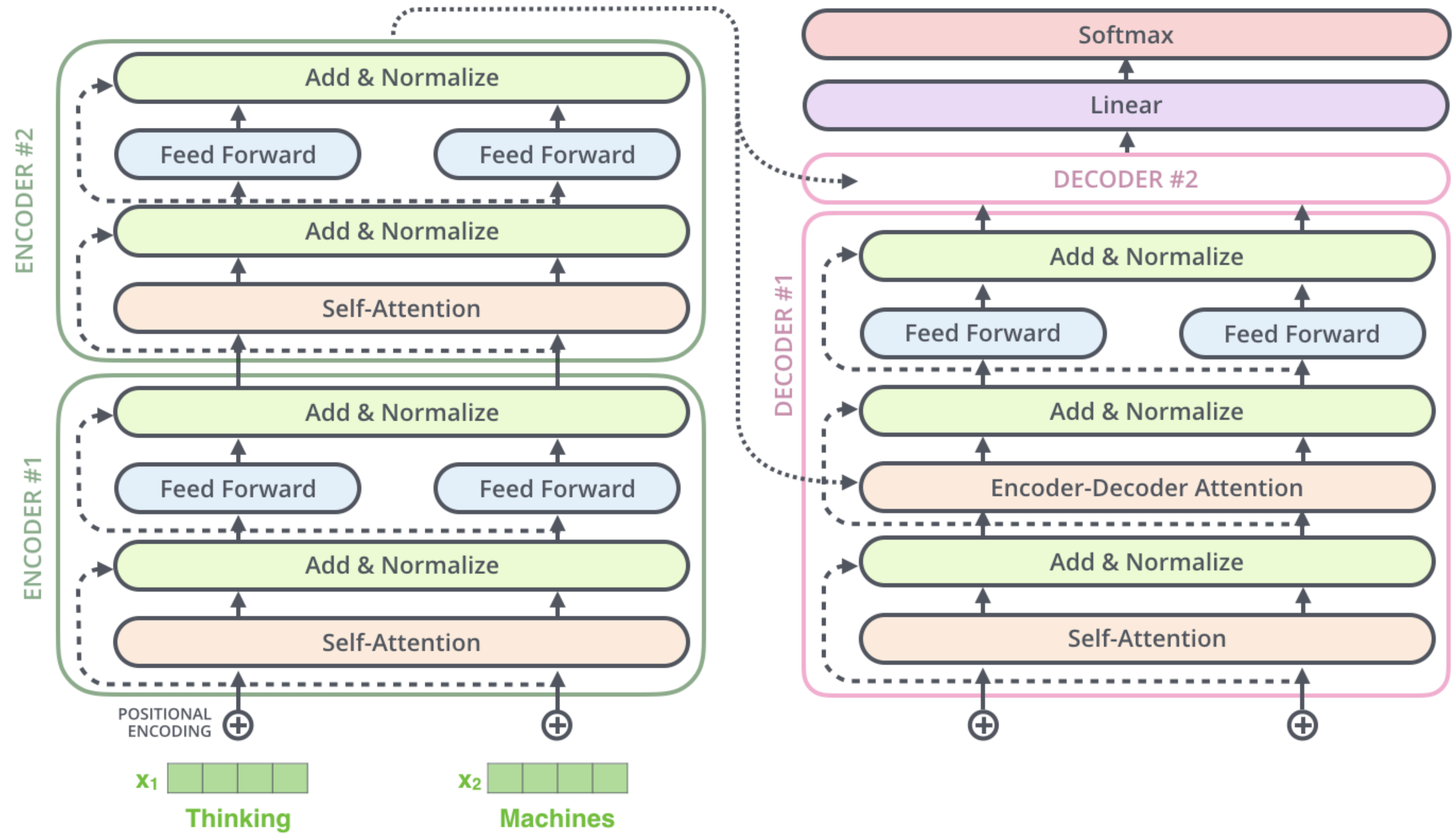


Transformers: Attention is all you need (Encoder)

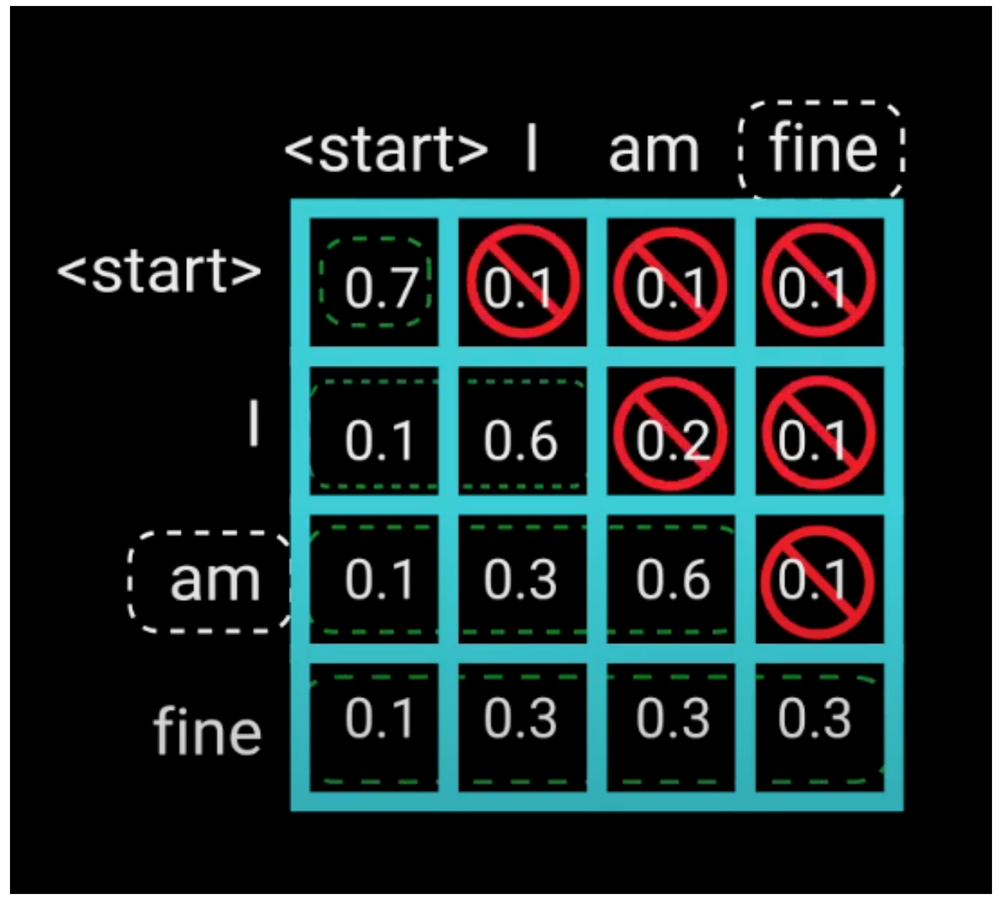


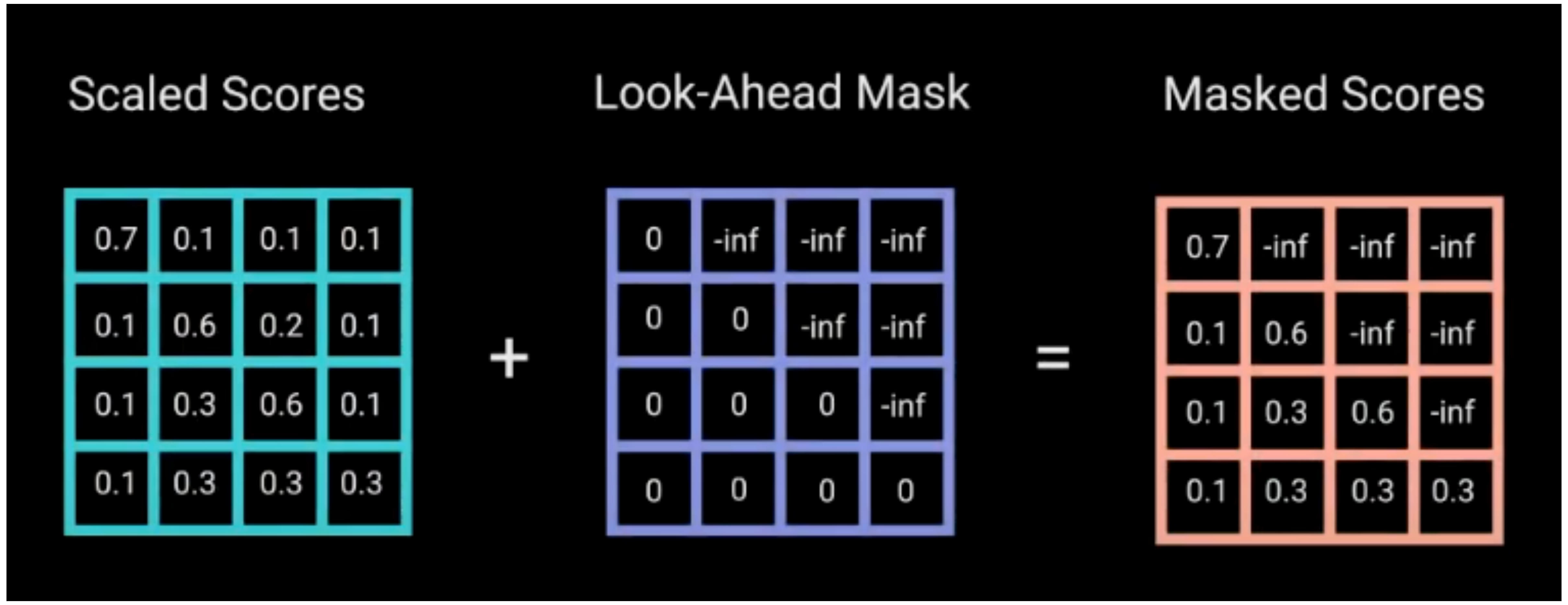


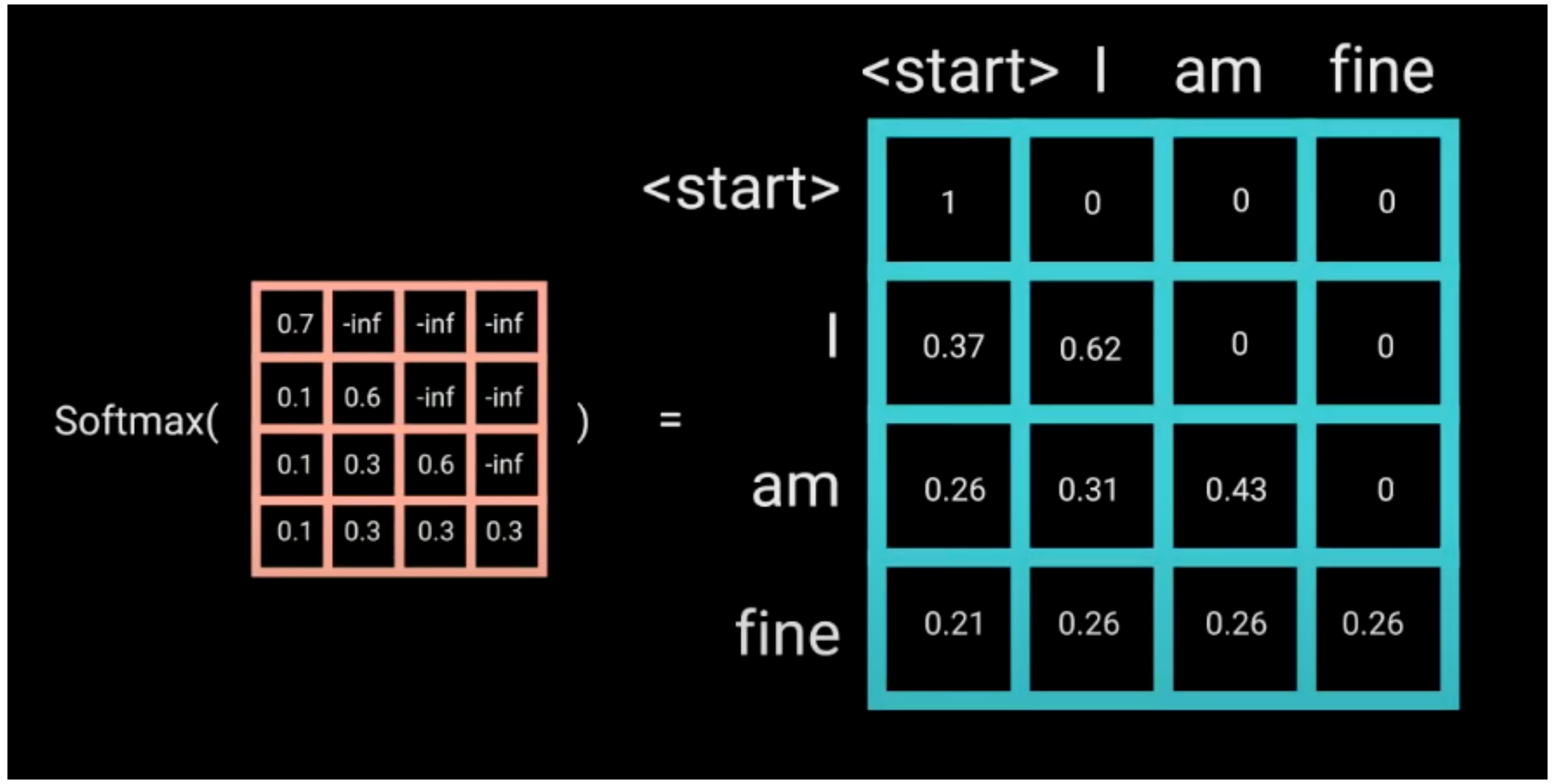
Transformers: Attention is all you need



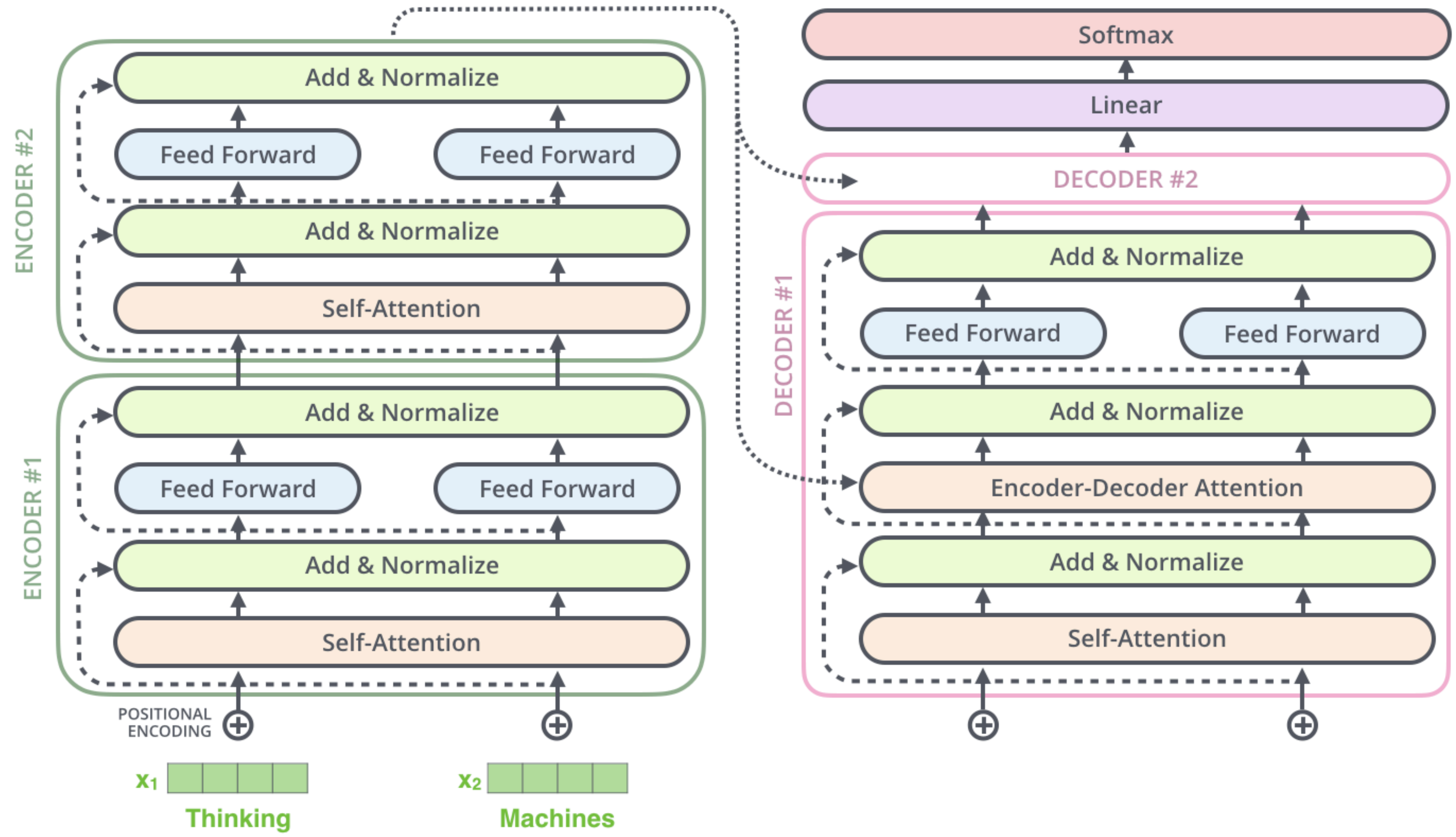
Transformers: Attention is all you need



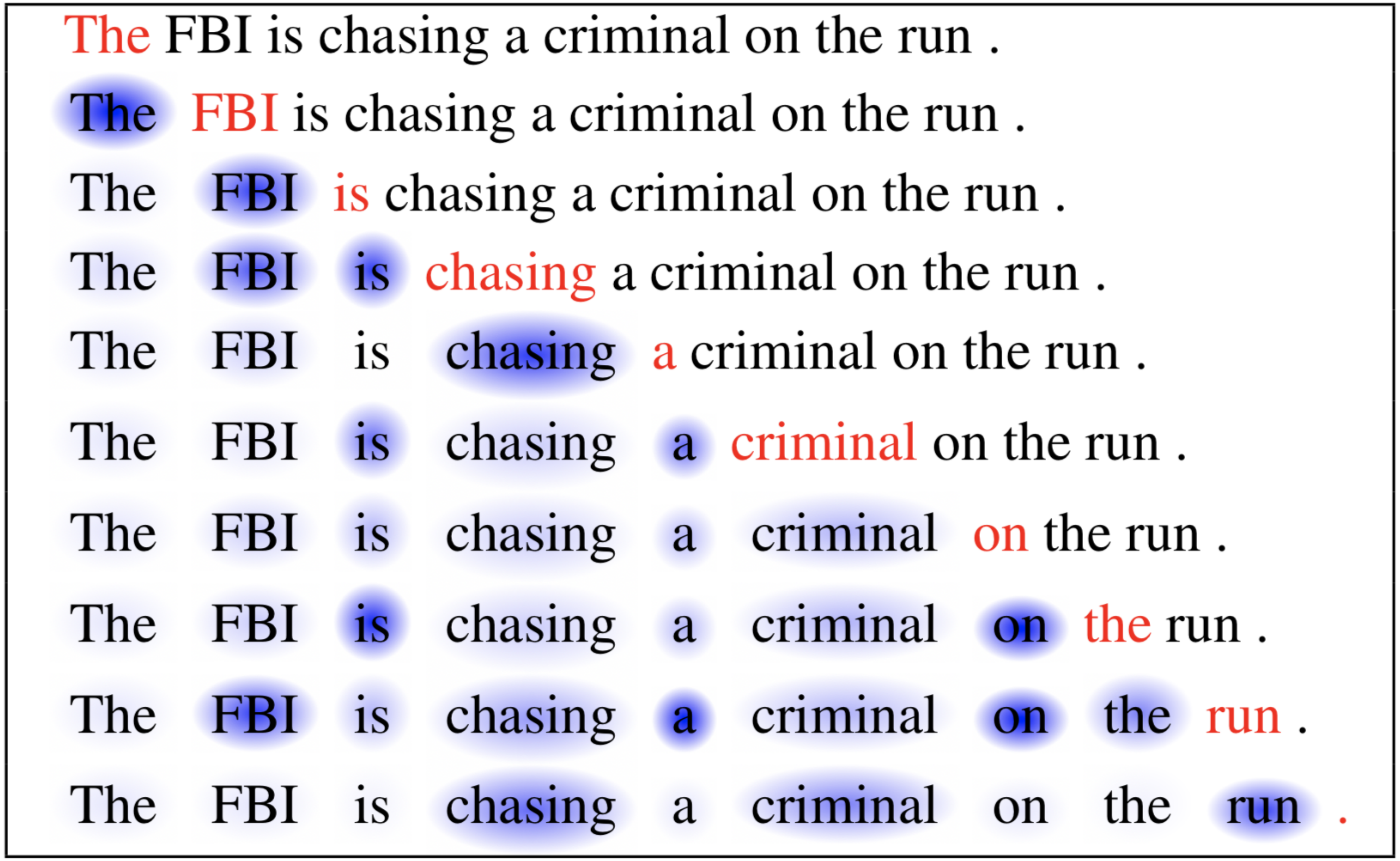




Transformers: Attention is all you need



Self Attention



Benefits of Transformers

1. Tokens are processes in **parallel** in both encoder and decoder, which is much faster than RNN or LSTM

2. Can (in principle) **model infinite history**, unlike RNN or LSTM that typically only carries context for relatively small number of steps

3. **No gradient flow issues**, due to residual architecture design of Transformer layers — similar to LSTM in some sense.

Benefits of Transformers

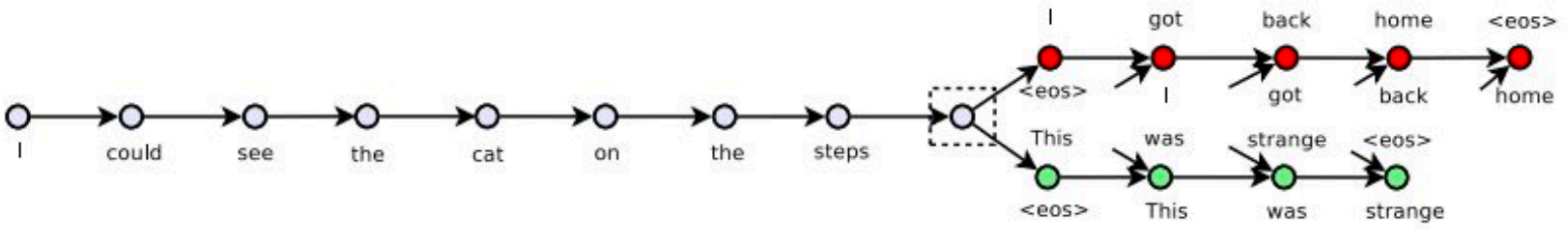
Note: In principle Transformer can model RNN-line or LSTM-like recursion by using causal mask and computing relevance based on “positional” information stored in a token representation and context based on “content” information stored in a token

(in other words, it is more or less strict generalization)

Let us look at some actual practical uses of RNNs

Applications: Skip-thought Vectors

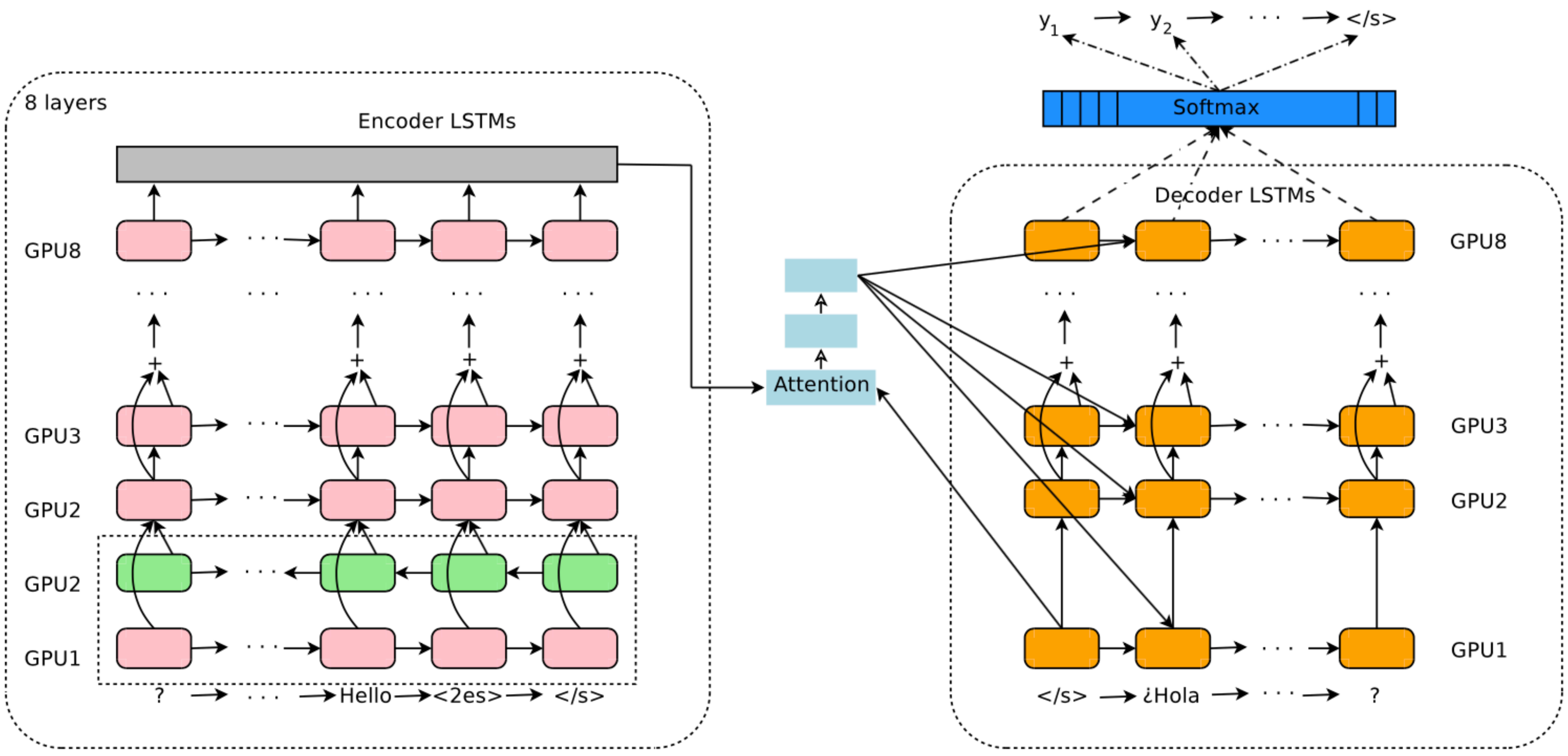
word2vec but for sentences, where each sentence is processed by an LSTM



[Kiros et al., 2015]

Applications: Google Language Translation

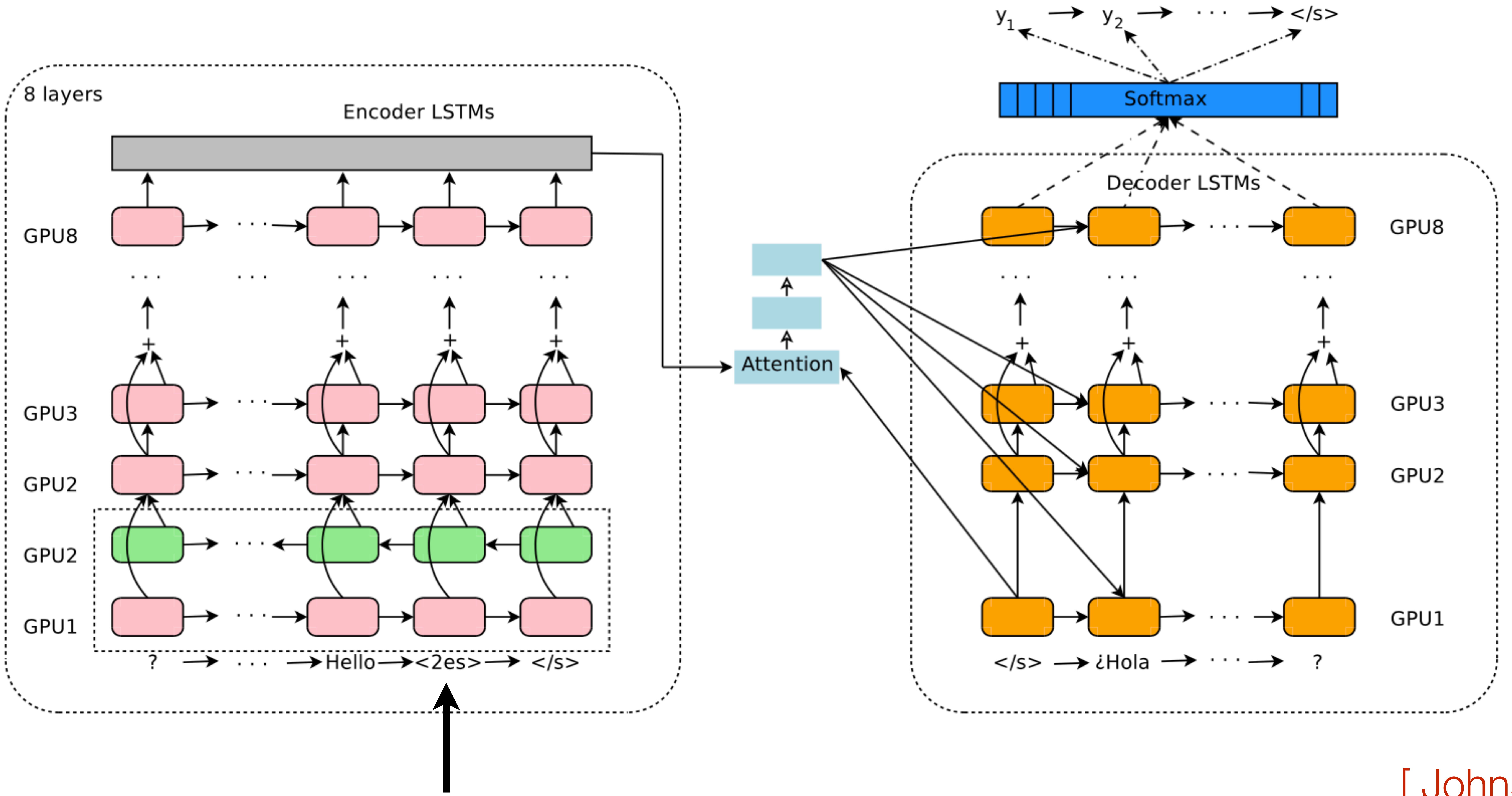
One model to translate from **any language** to any other language



[Johnson et al., 2017]

Applications: Google Language Translation

One model to translate from **any language** to any other language

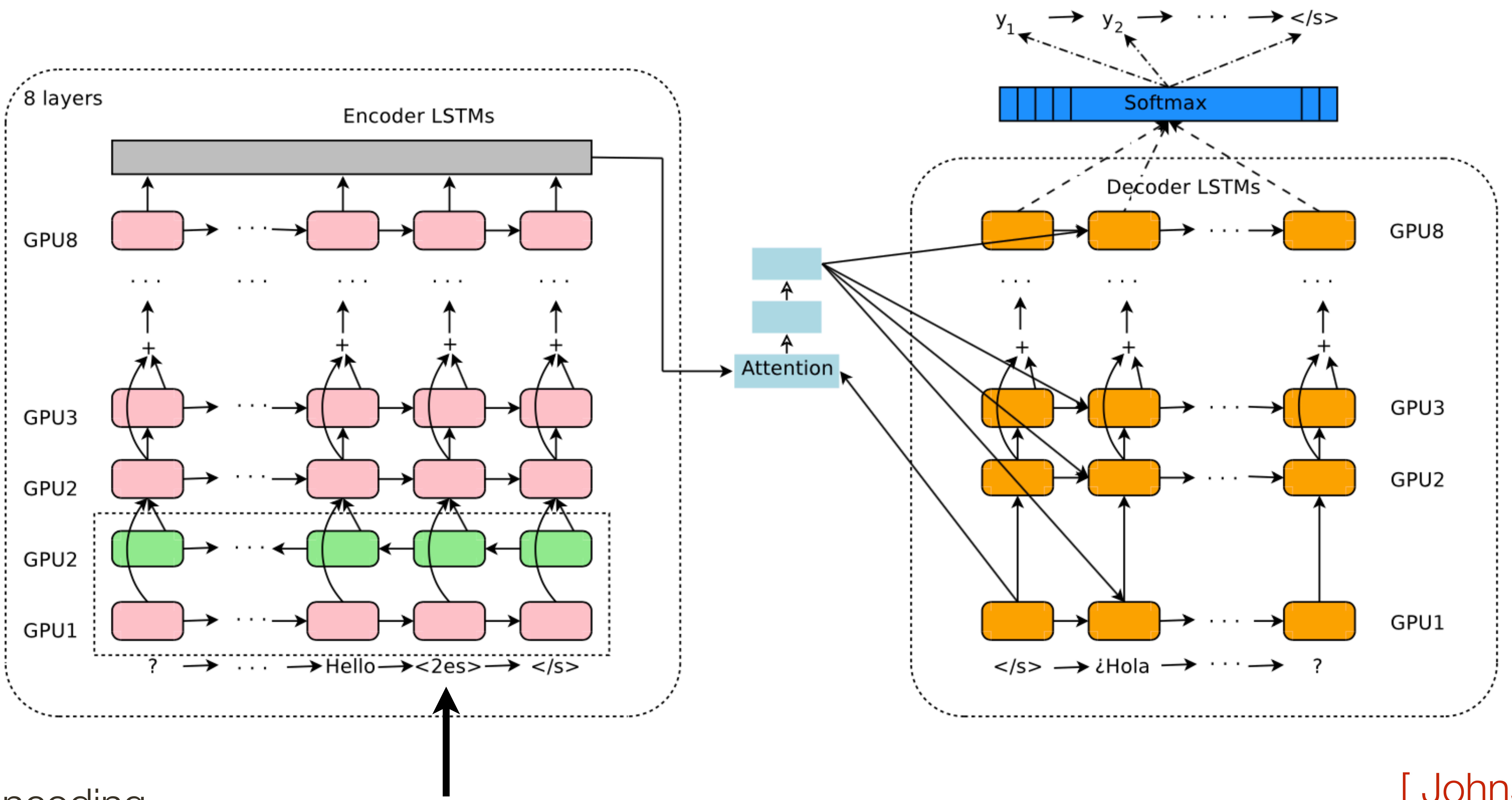


Token designating
target language

[Johnson et al., 2017]

Applications: Google Language Translation

One model to translate from **any language** to any other language



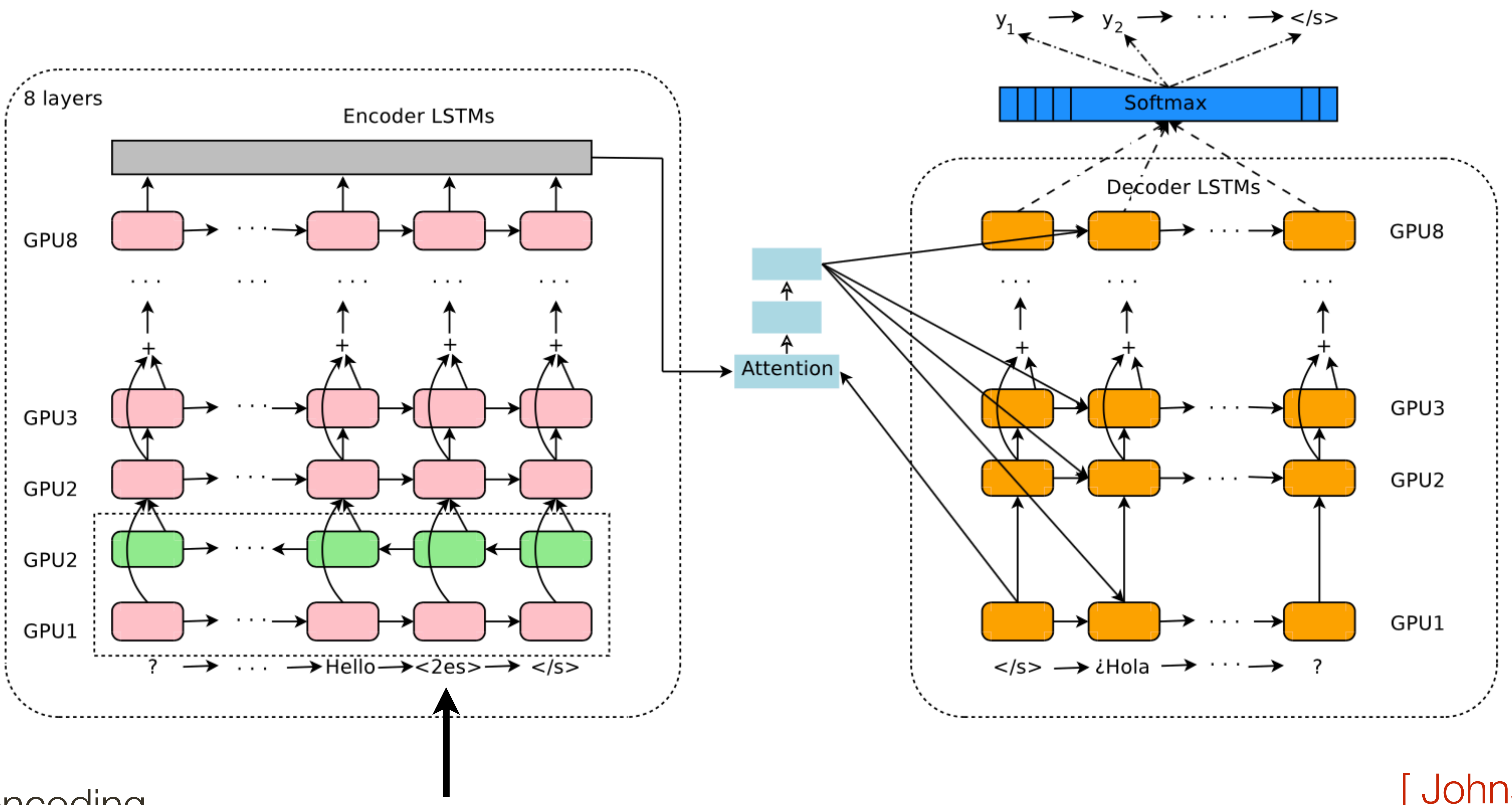
Flipped order encoding

Token designating **target** language

[Johnson et al., 2017]

Applications: Google Language Translation

One model to translate from **any language** to any other language



Flipped order encoding

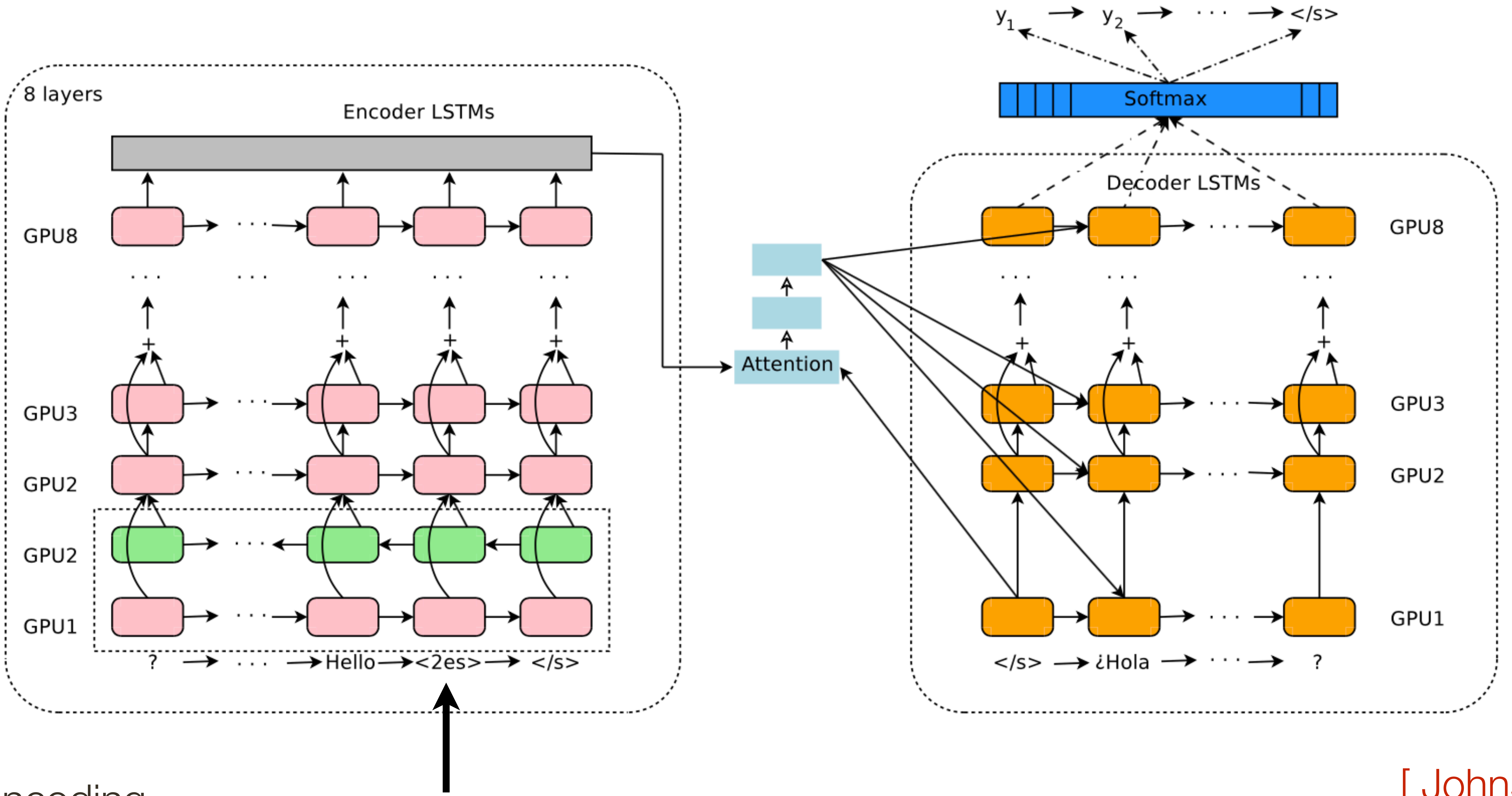
Why?

Token designating **target** language

[Johnson et al., 2017]

Applications: Google Language Translation

One model to translate from **any language** to any other language



Flipped order encoding

Token designating **target** language

[Johnson et al., 2017]

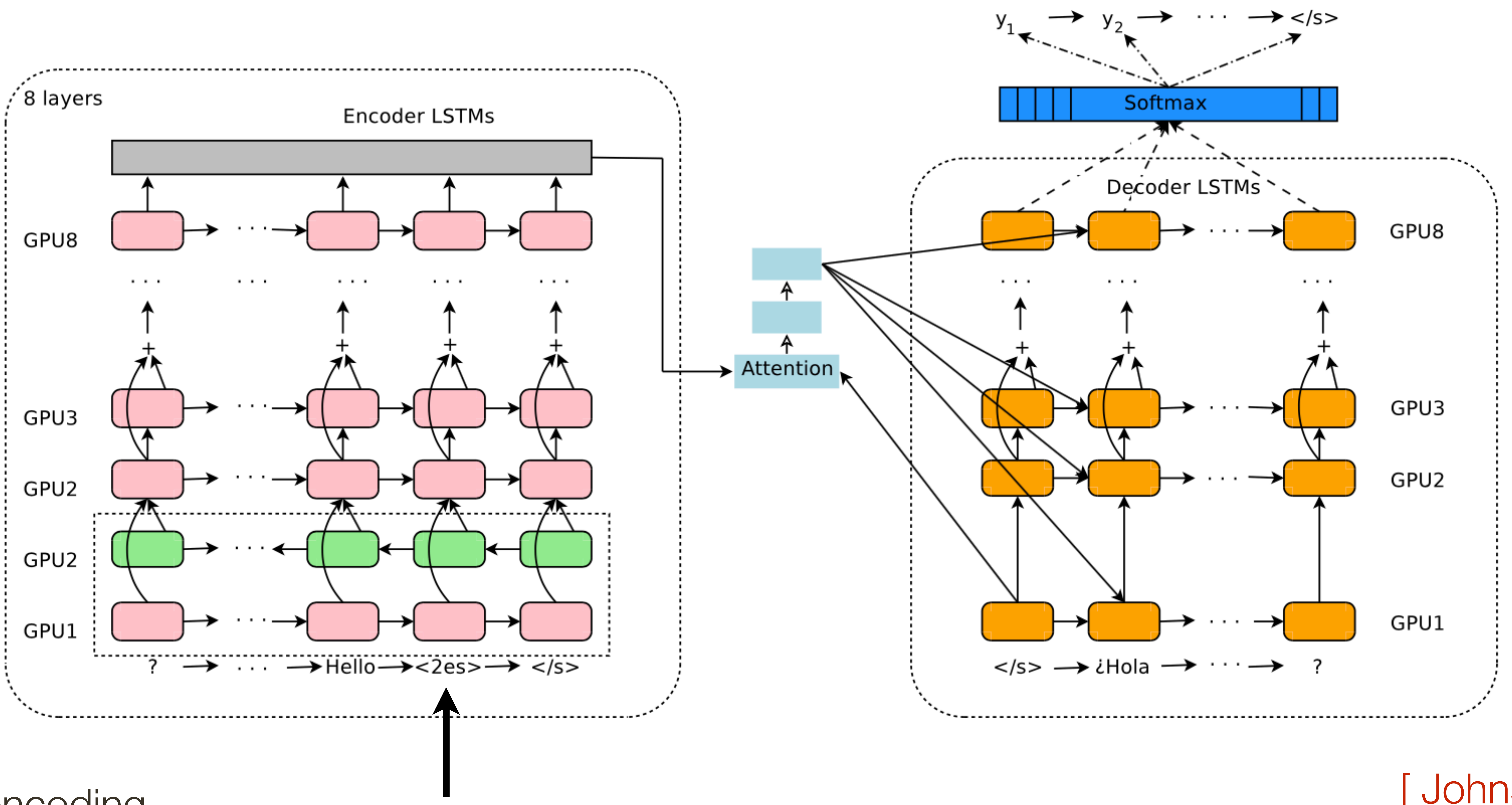
8! layer LSTM decoder and encoder

Applications: Google Language Translation

One model to translate from **any language** to any other language

Residual at other layers (ResNet style)

Bi-directional at lower layers



Flipped order encoding

Token designating **target** language

[Johnson et al., 2017]

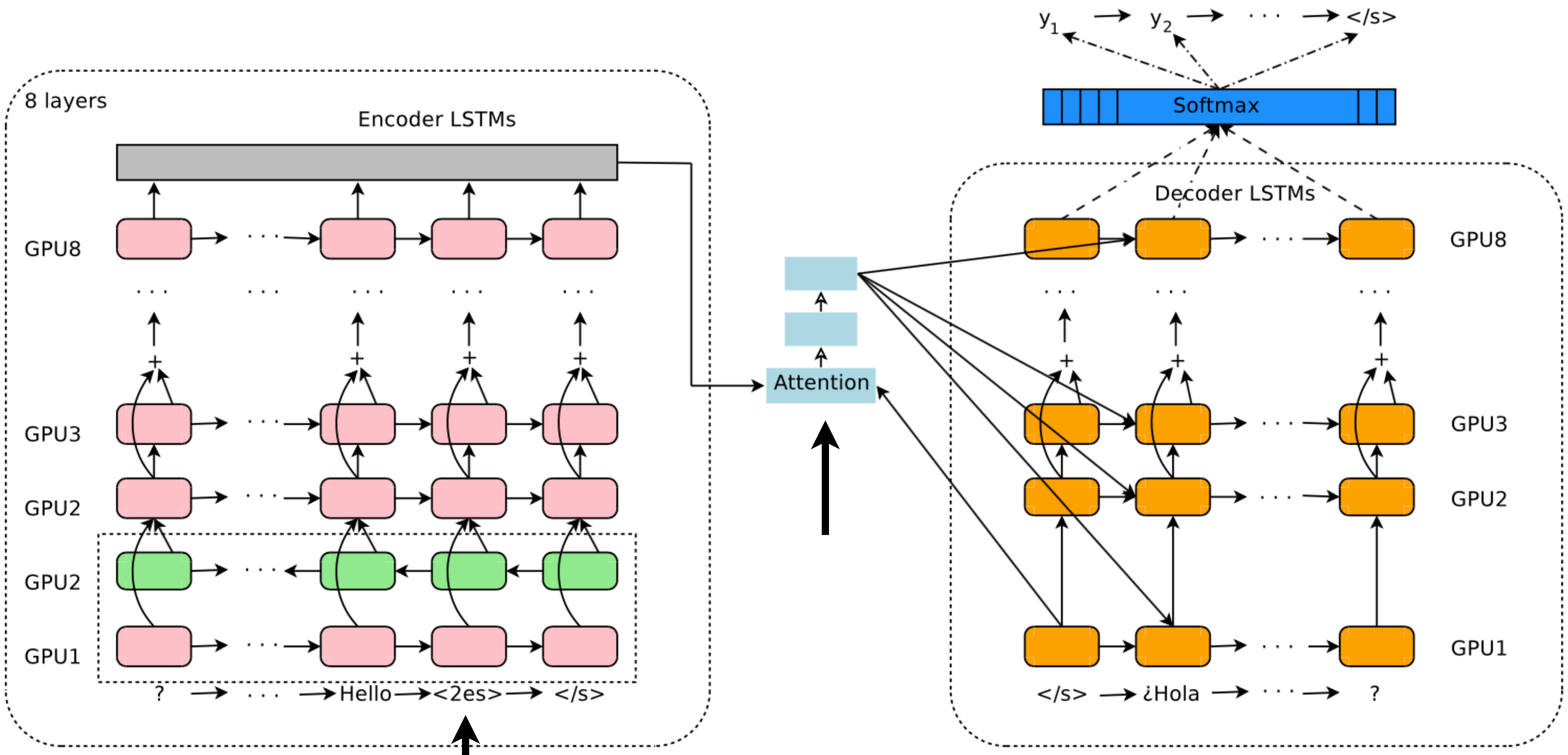
8! layer LSTM decoder and encoder

Applications: Google Language Translation

One model to translate from **any language** to any other language

Residual at other layers (ResNet style)

Bi-directional at lower layers



Flipped order encoding

Token designating **target** language

[Johnson et al., 2017]

8! layer LSTM decoder and encoder