華東師範大學
EAST CHINA NORMAL UNIVERSITY

Attention Is All You Need

Brief Introduction to Attention Mechanism

Shang Gao

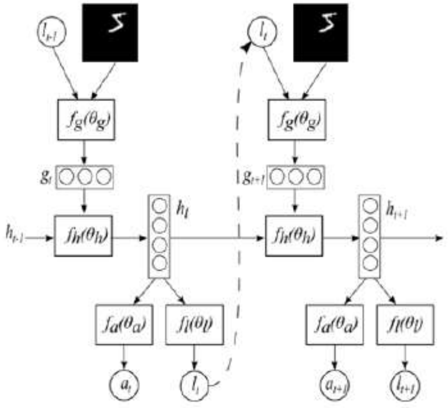
Multi-Agent Artificial Intelligence Laboratory
East China Normal University

Outline

- Introductions
- Attention Mechanisms
- Seq2Seq + Attention
- Transformer

Introductions

Recurrent Attention Model

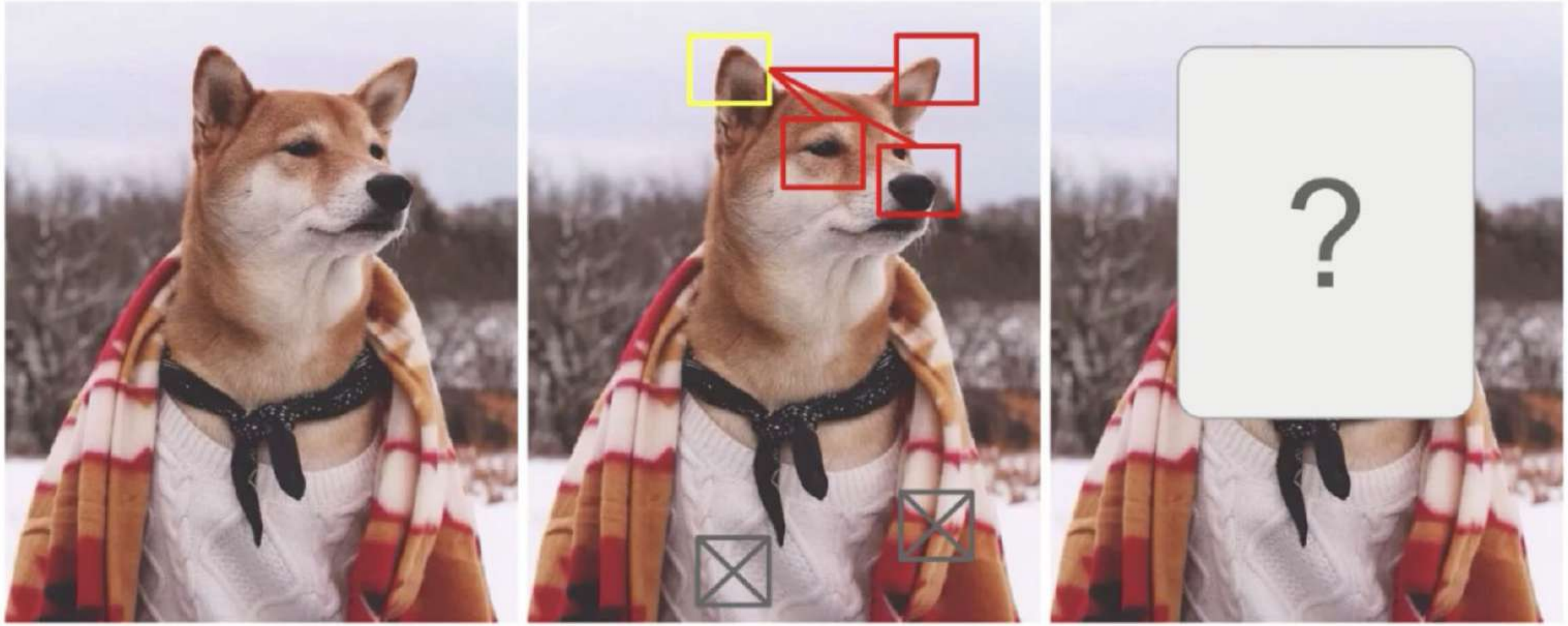


- The core network of the model $f_h(.;\theta_h)$:
  - Input: Glimpse representation g_t;
  - Input: Internal representation at previous time step h_{t-1};
  - Output: New internal state of the model h_t.
- The location network $f_l(.;\theta_l)$:
  - Input: Internal state h_t;
  - Output: location to attend to l_t.
- The action network $f_a(.;\theta_a)$:
  - Input: Internal state h_t;
  - Output: Action/classification a_t.

Mnih Volodymyr, Nicolas Heess, and Alex Graves. "Recurrent models of visual attention." *Advances in neural information processing systems*. 2014. [arxiv]

- Consider the attention problem as the **sequential decision process** of a goal-directed agent interacting with a visual environment;
- At each time step t:
  - The agent observes the environment only via a bandwidth-limited sensor;
  - The agent can actively control how to deploy its sensor resources and affect the true state of the environment by executing actions;
  - The agent receives a scalar reward.

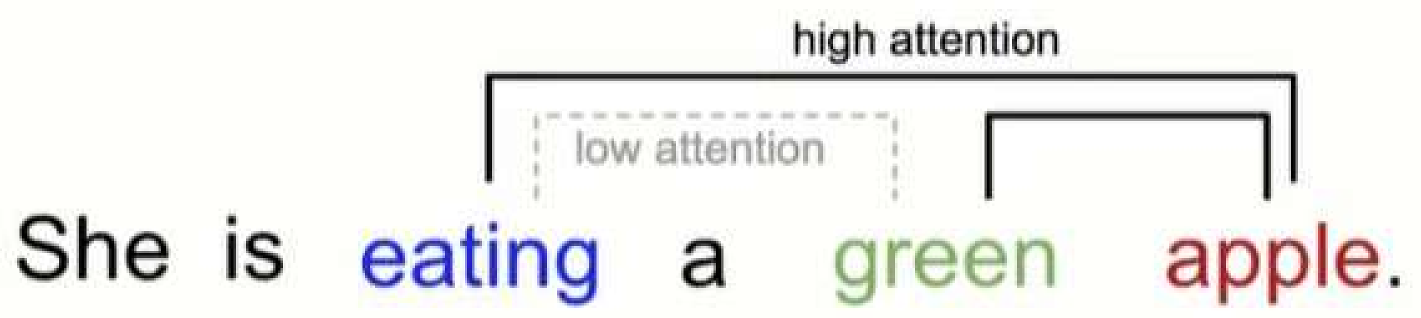
Attention in CV



Attention in CV (Cont.)

- CNN is computationally expensive for large images:
  - The amount of computation scales linearly with the number of image pixels.
- Human perception:
  - Not tend to process a whole scene but focus attention selectively on parts of the visual space

Attention in NLP



Attention in NLP (Cont.)

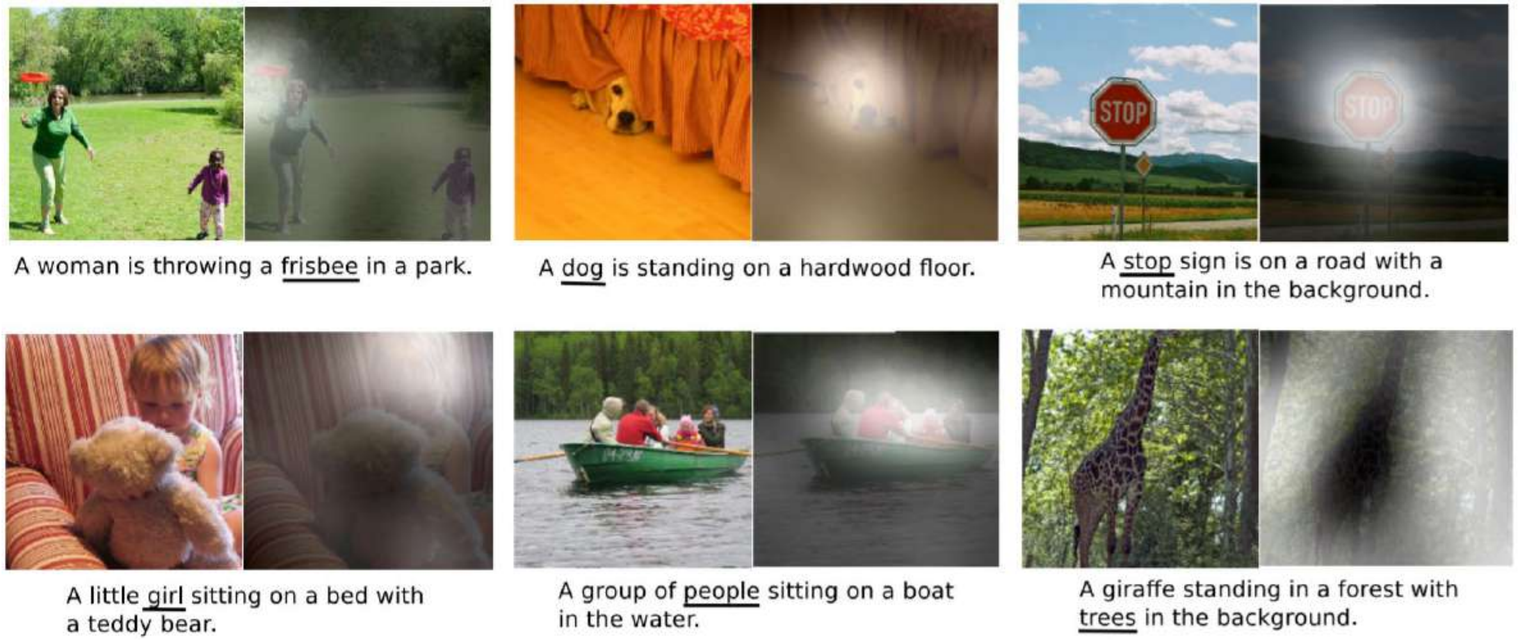
- Despite attention mechanisms was first applied in CV, using attention in NLP is naturally better than in CV
  - Translating Chinese into English in NLP: Recurrent structures (RNN/LSTM) are able to apply attention for Seq2Seq problems
  - Image Classification in CV: Image is not a sequence
- In NLP, BERT and GPT, which use attention-based Transformer, work surprisingly well.

Why Do You Need Attention

- Smaller:
  - Compared with CNN and RNN, the model complexity is smaller, and the parameters are fewer. So, it requires less computing power.
- Faster:
  - Attention solves the problem that RNN cannot be calculated in parallel, while each calculation of the Attention mechanism does not depend on the calculation result of the previous step. So, it can be processed in parallel with CNN.
- Better:
  - Before attention was introduced, long-distance information will be weakened, just like people with weak memory can't remember the past. However, attention takes the most important parts, even if the text is relatively long, you can get the point from it without losing important information.

Attention Mechanisms

What is Attention

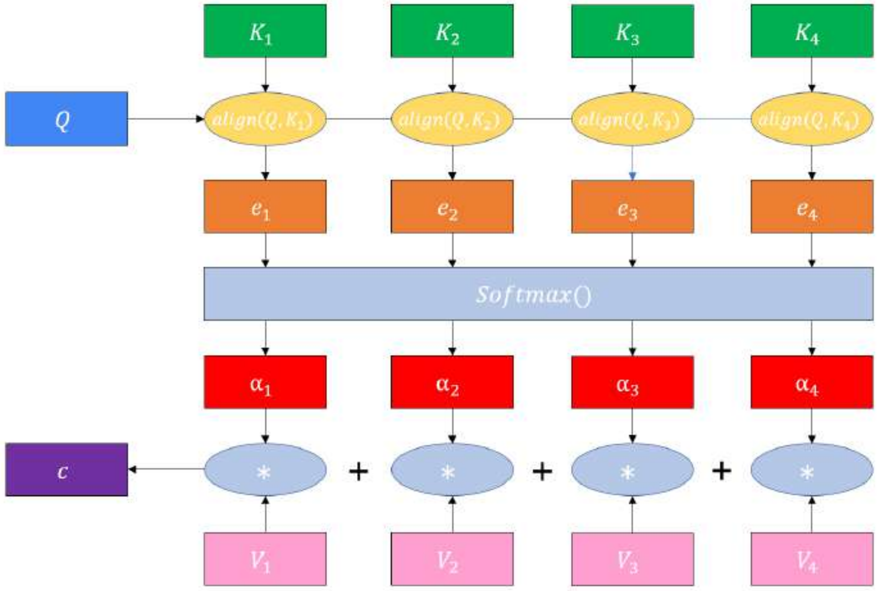


Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015. [arxiv]

What is Attention (Cont.)

- Attention mechanisms are essentially designed to mimic the way humans look at objects.
- The key is：
  - From focusing all to focusing core
- Or, let’s say:
  - Weighted summation

What is Attention (Cont.)



- Here are the 3 steps:
  - Get weight values from the similarity of the query and the key using alignment scores
  - Normalize weight values and get available weights
  - Get weighted sum of weight and value

Alignment scores

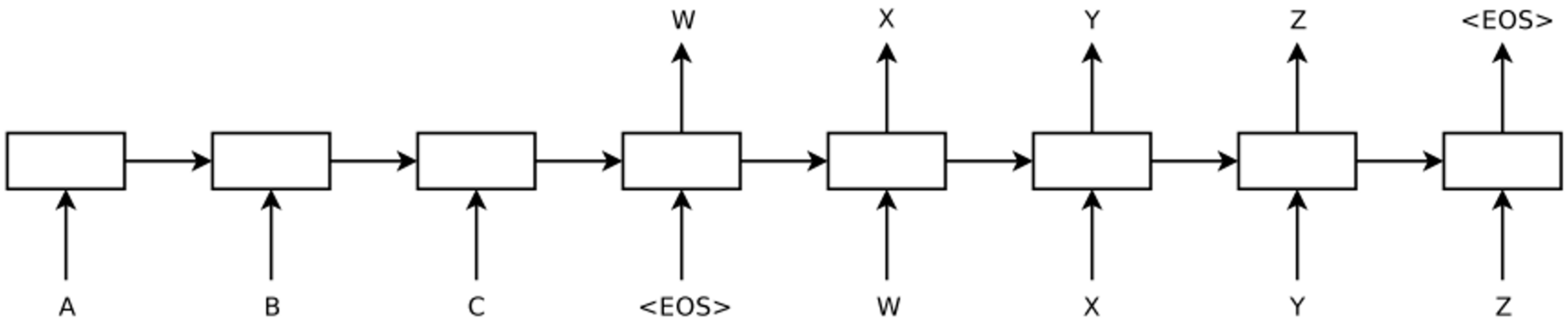
Name	Alignment Score Function
Dot-Product Attention [arxiv]	$align(s_t, h_i) = s_t^\top h_i$
Scaled Dot-Product Attention [arxiv]	$align(s_t, h_i) = \frac{s_t^\top h_i}{\sqrt{n}}$
Additive / Bahdanau Attention [arxiv]	$align(s_t, h_i) = v_a^\top \tanh(W_a[s_t; h_i])$ Note: W_a, V_a are trainable weight matrices.
General / Multiplicative / Luong Attention [arxiv]	$align(s_t, h_i) = s_t^\top W_a h_i$ Note: W_a is trainable weight matrix.
Concatenating Attention	$align(s_t, h_i) = W[s_t; h_i]$
Perceptron Attention	$align(s_t, h_i) = v_a^\top \tanh(W_a s_t + U_a h_i)$
Content-Based Attention [arxiv]	$align(s_t, h_i) = \text{cosine}[s_t, h_i] = \frac{s_t^\top h_i}{\lVert s_t \rVert \cdot \lVert h_i \rVert}$
Location-Based Attention [arxiv]	$\alpha_{t,i} = softmax(W_a s_t)$ Note: Simplified the $softmax()$ alignment to only depend on the target position.

Types of Attention

- Self-Attention [arxiv] [arxiv]
- Soft / Global Attention [arxiv]
- Hard / Local Attention [arxiv] [arxiv]

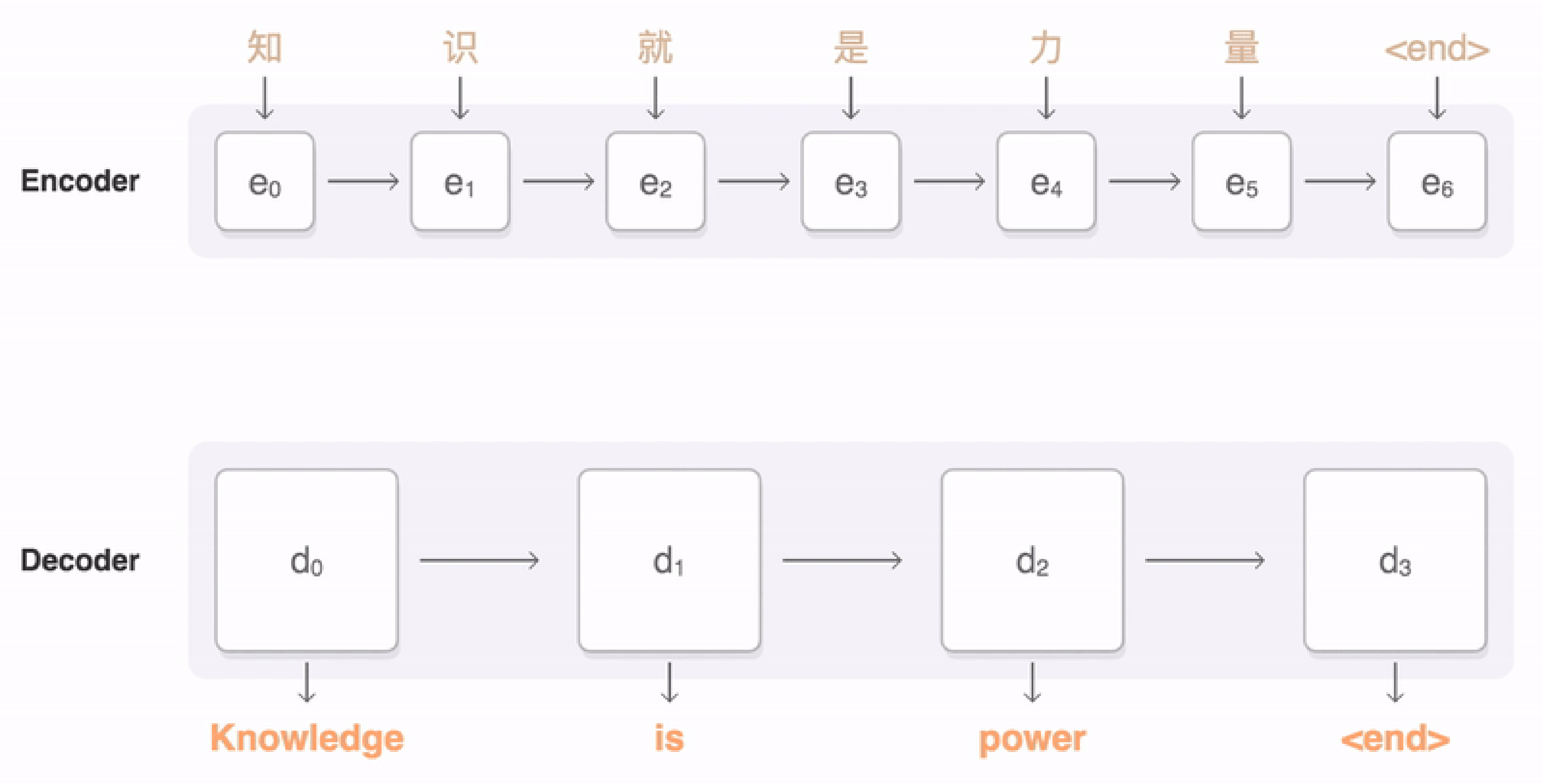
Seq2Seq + Attention

Preliminaries: Seq2Seq Model



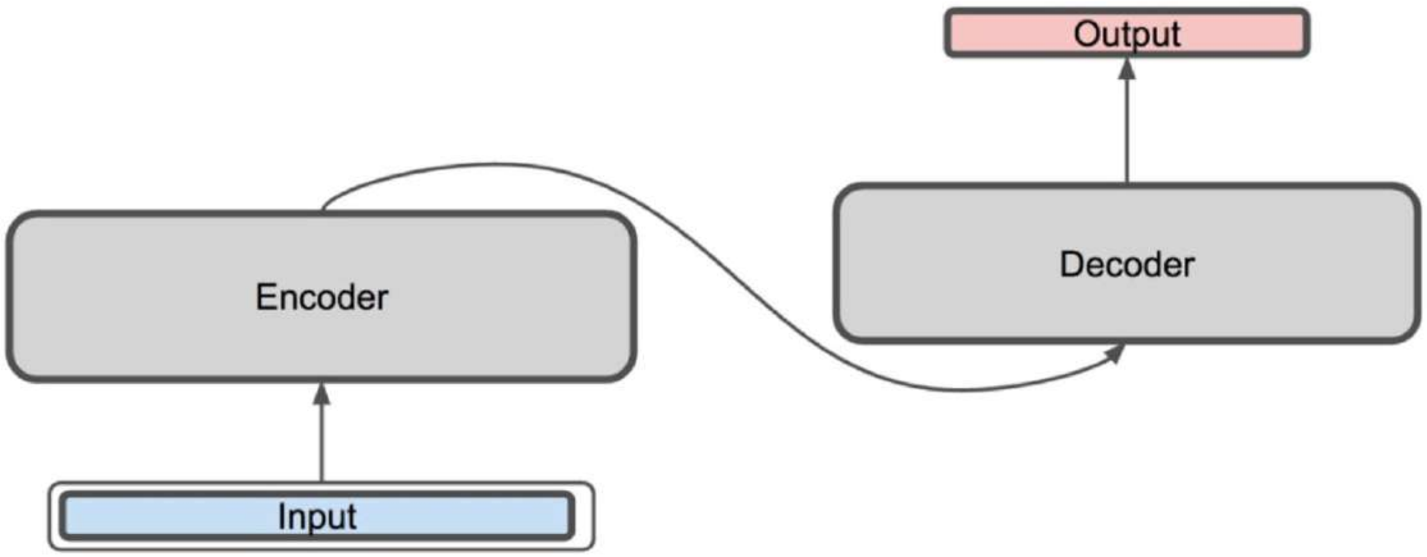
Sutskever, I., O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks." Advances in NIPS 2014. [arxiv]

Seq2Seq + Attention



Britz, Denny, et al. "Massive exploration of neural machine translation architectures." *arXiv preprint arXiv:1703.03906* (2017). [arxiv]

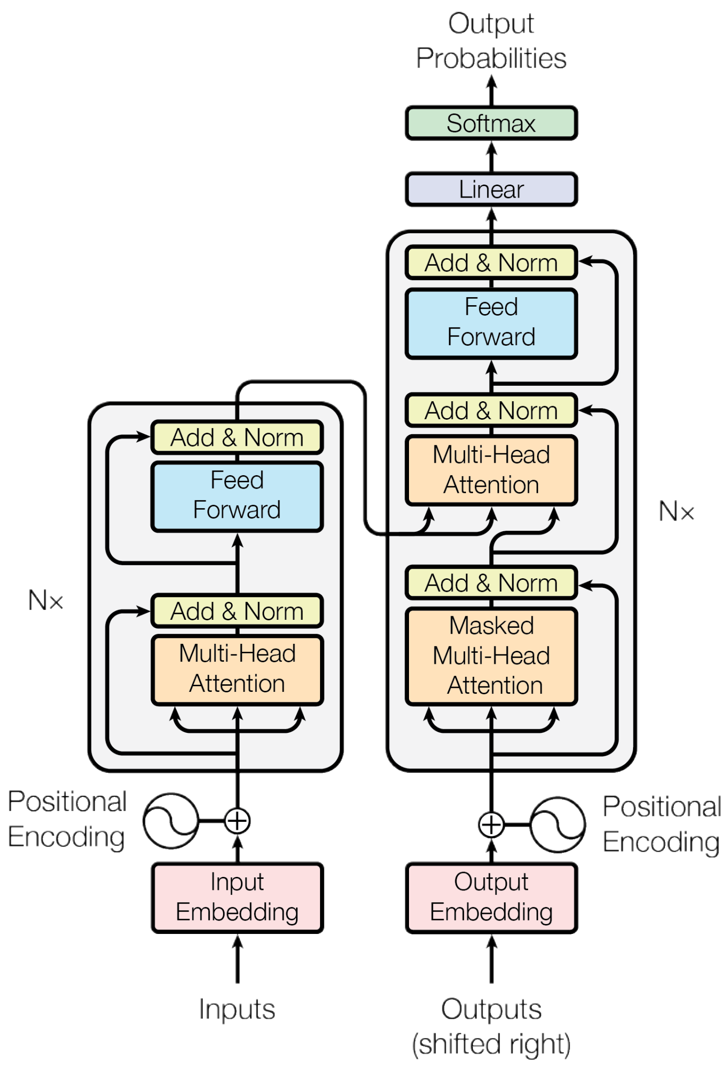
Seq2Seq + Attention (Cont.)



Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025*. (2015). [arxiv]

Transformer

Overview



Transformer

- **Encoder-decoder attention**

$$Y = \mathrm{MultiHead}(V, K, Q) = \mathrm{MultiHead}(X_e, X_e, X_d)$$

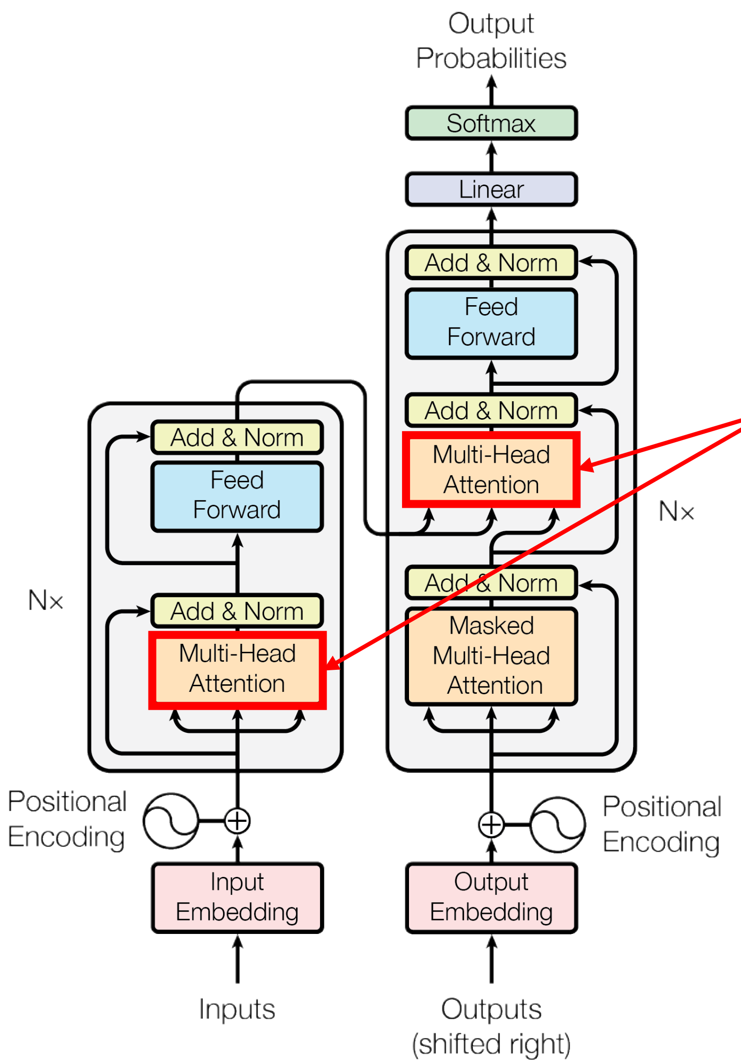
- **Self-attention layers in the encoder**

$$Y = \mathrm{MultiHead}(V, K, Q) = \mathrm{MultiHead}(X_e, X_e, X_e)$$

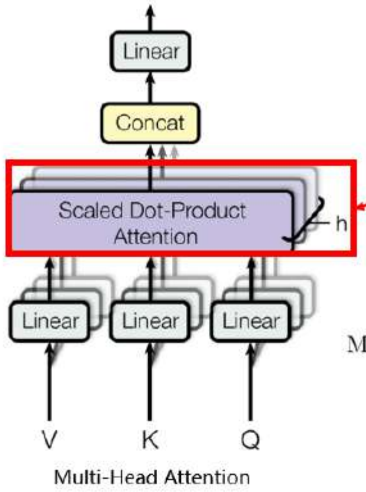
- **Self-attention layers in the decoder**

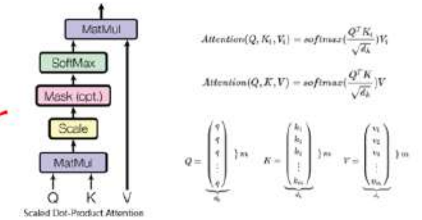
$$Y = \mathrm{MultiHead}(V, K, Q) = \mathrm{MultiHead}(X_d, X_d, X_d)$$

Model Architecture



Transformer

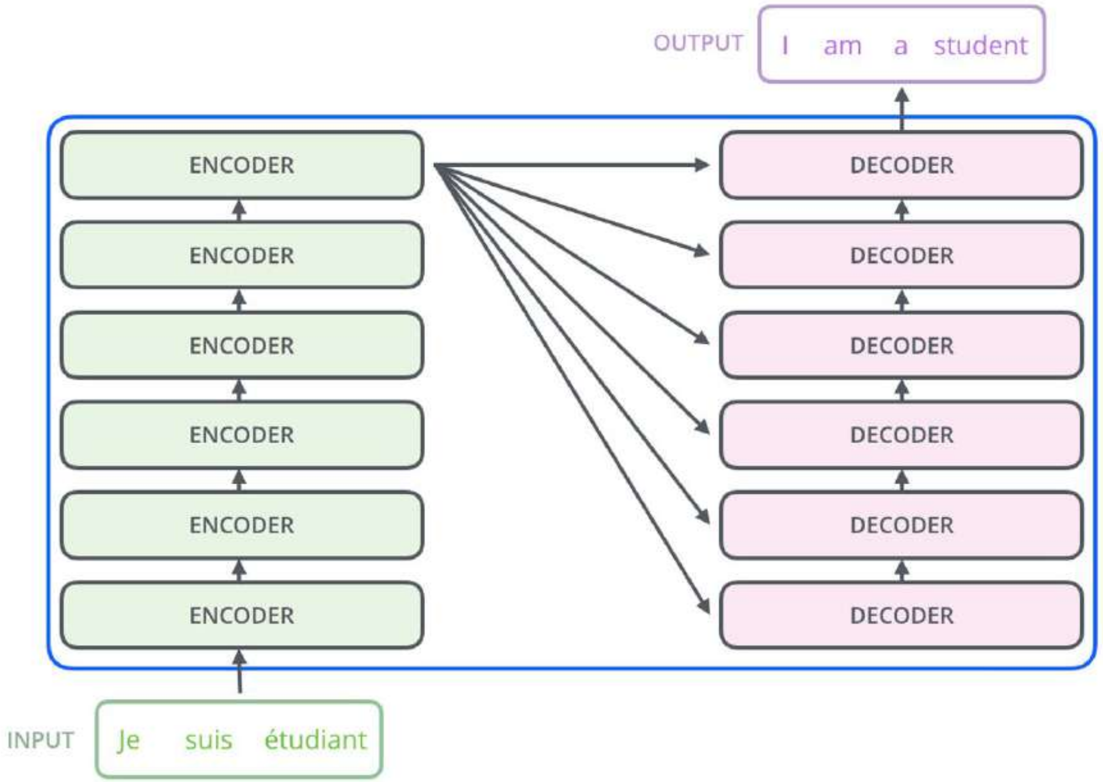




$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1,\ldots,\mathrm{head}_h)W^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$LayerNorm(x + Sublayer(x))$$

Model Architecture (Cont.)



Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]

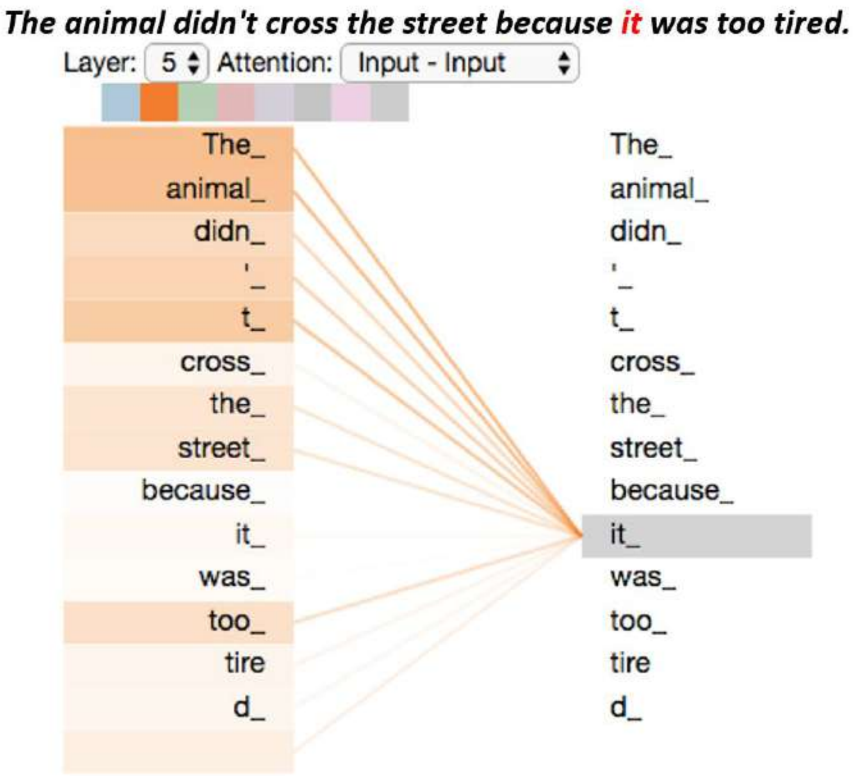
Self Attention

- Input: Sequence $(x_1, \dots, x_T)$
- Output: Sequence $(y_1, \dots, y_{T'})$
  - Note: T may differ from T'
- Criteria:
  - Total computational complexity per layer
  - The amount of computation that can be parallelized
  - Path length between long-range dependencies in the network

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

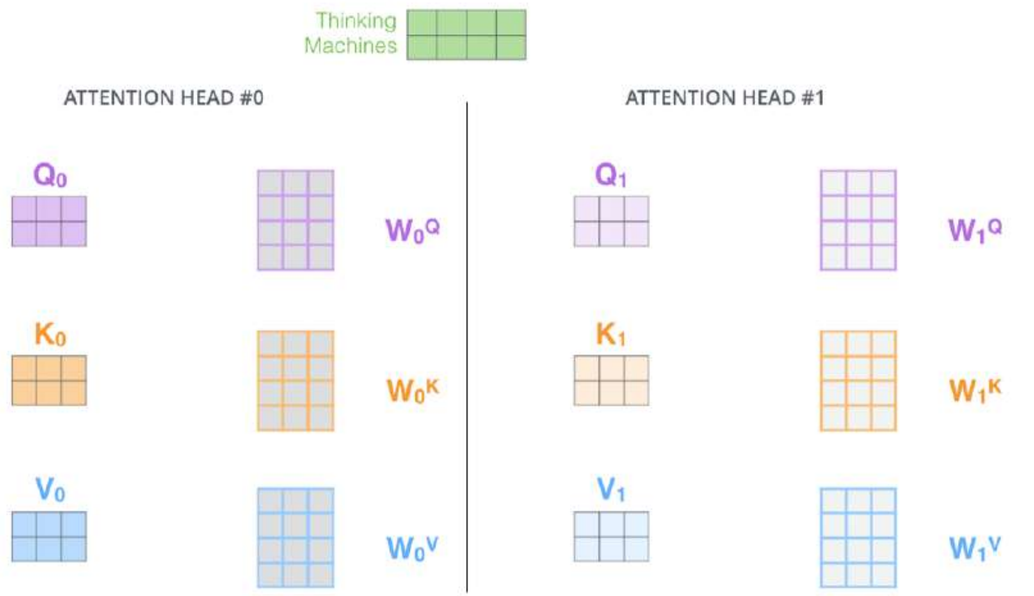
Self Attention (Cont.)



Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]

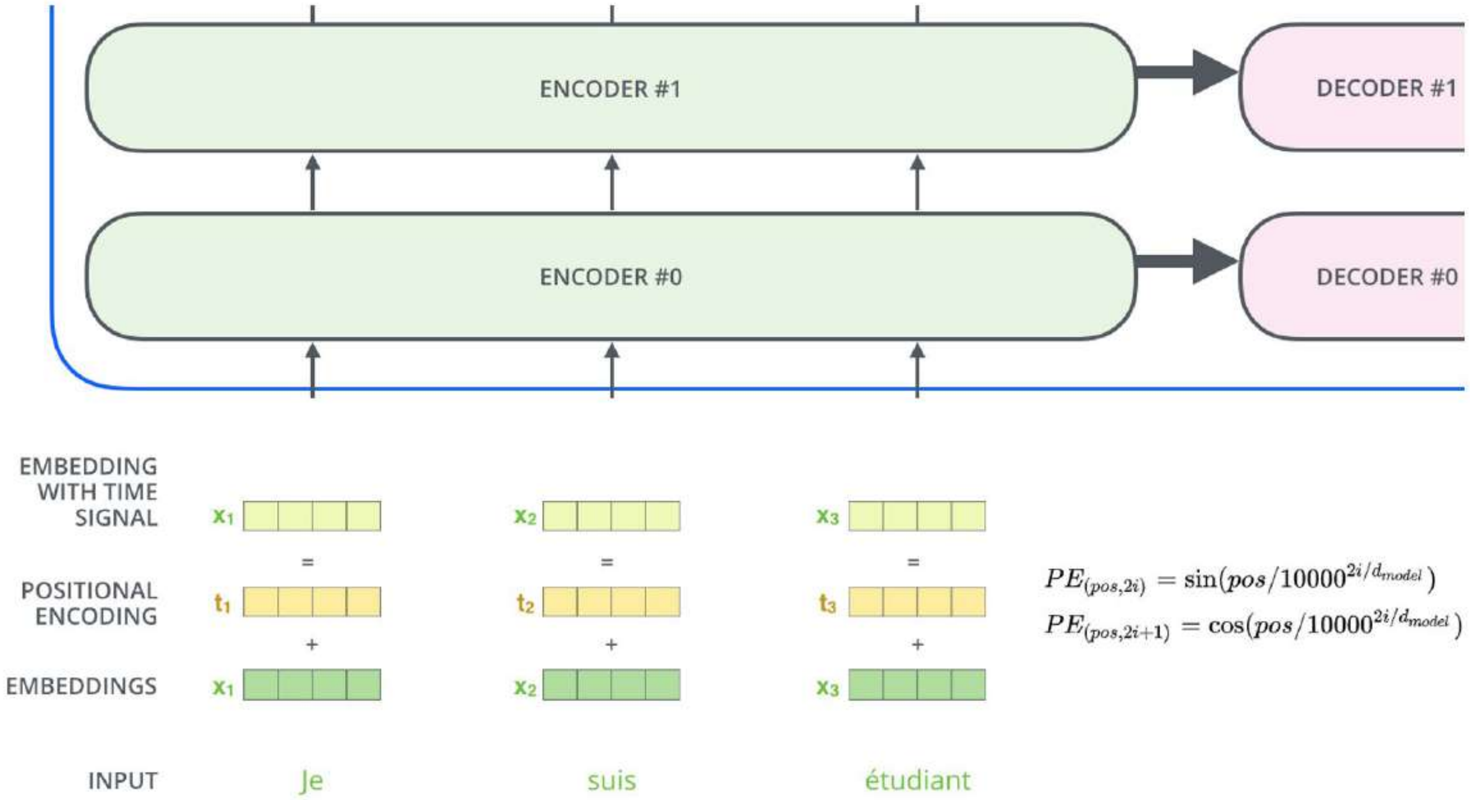
Multi-Head Attention

- Expands the model’ s ability to focus on different positions
- Gives the attention layer multiple “representation subspaces”



Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]

Positional Encoding



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

Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]

Training

- Dataset: the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs
- Hardware: 8 NVIDIA P100 GPUs
  - Base Model: 100,000 steps (12 hours)
  - Big Model: 300,000 steps (3.5 days)
- Optimization：
  - Optimizer: Adam
  - Warmup
- Regularization
  - Residual Dropout
  - Label Smoothing

References

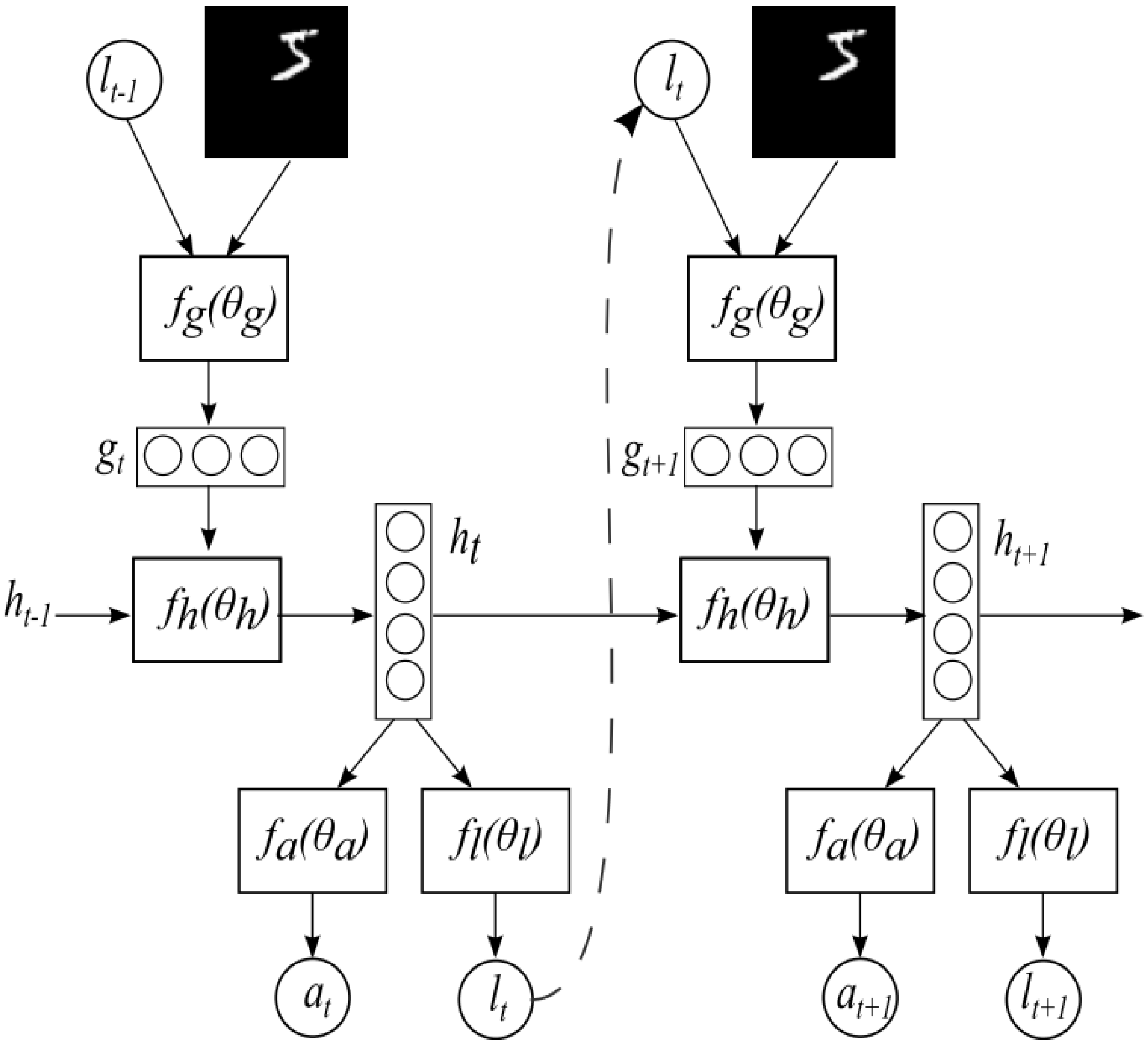
References

- Mnih Volodymyr, Nicolas Heess, and Alex Graves. "Recurrent models of visual attention." *Advances in NIPS*. 2014. [arxiv]
- Sutskever, I., O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks." *Advances in NIPS.* 2014. [arxiv]
- Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." *arXiv preprint arXiv:1410.5401*. (2014). [arxiv]
- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473*. (2014). [arxiv]
- Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025*. (2015). [arxiv]

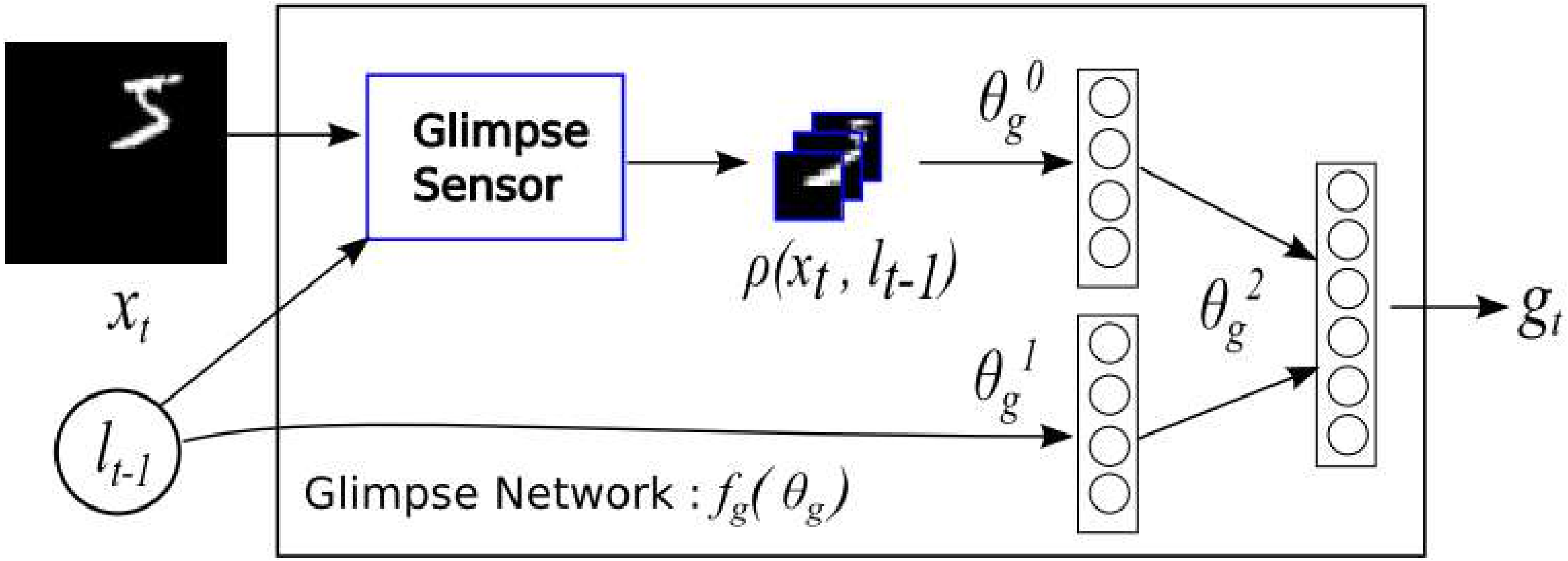
References

- Vaswani, Ashish, et al. "Attention is all you need." *Advances in NIPS*. 2017. [arxiv]
- Cheng, Jianpeng, Li Dong, and Mirella Lapata. “Long short-term memory-networks for machine reading.” *arXiv preprint arXiv:1601.06733*. (2016). [arxiv]
- Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015. [arxiv]
- Britz, Denny, et al. "Massive exploration of neural machine translation architectures." *arXiv preprint arXiv:1703.03906* (2017). [arxiv]

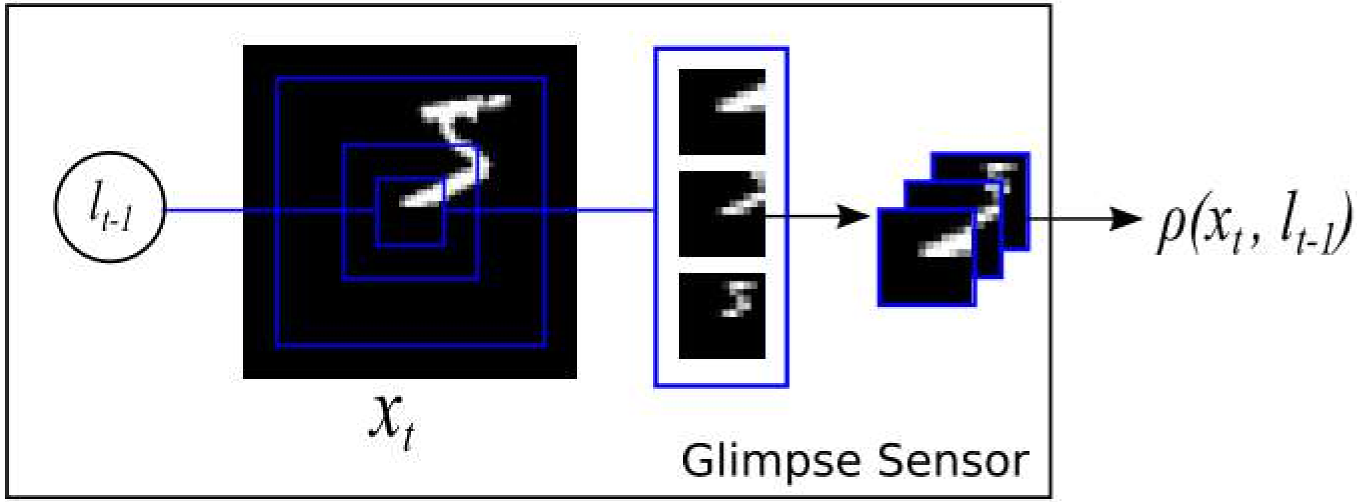
Thank You



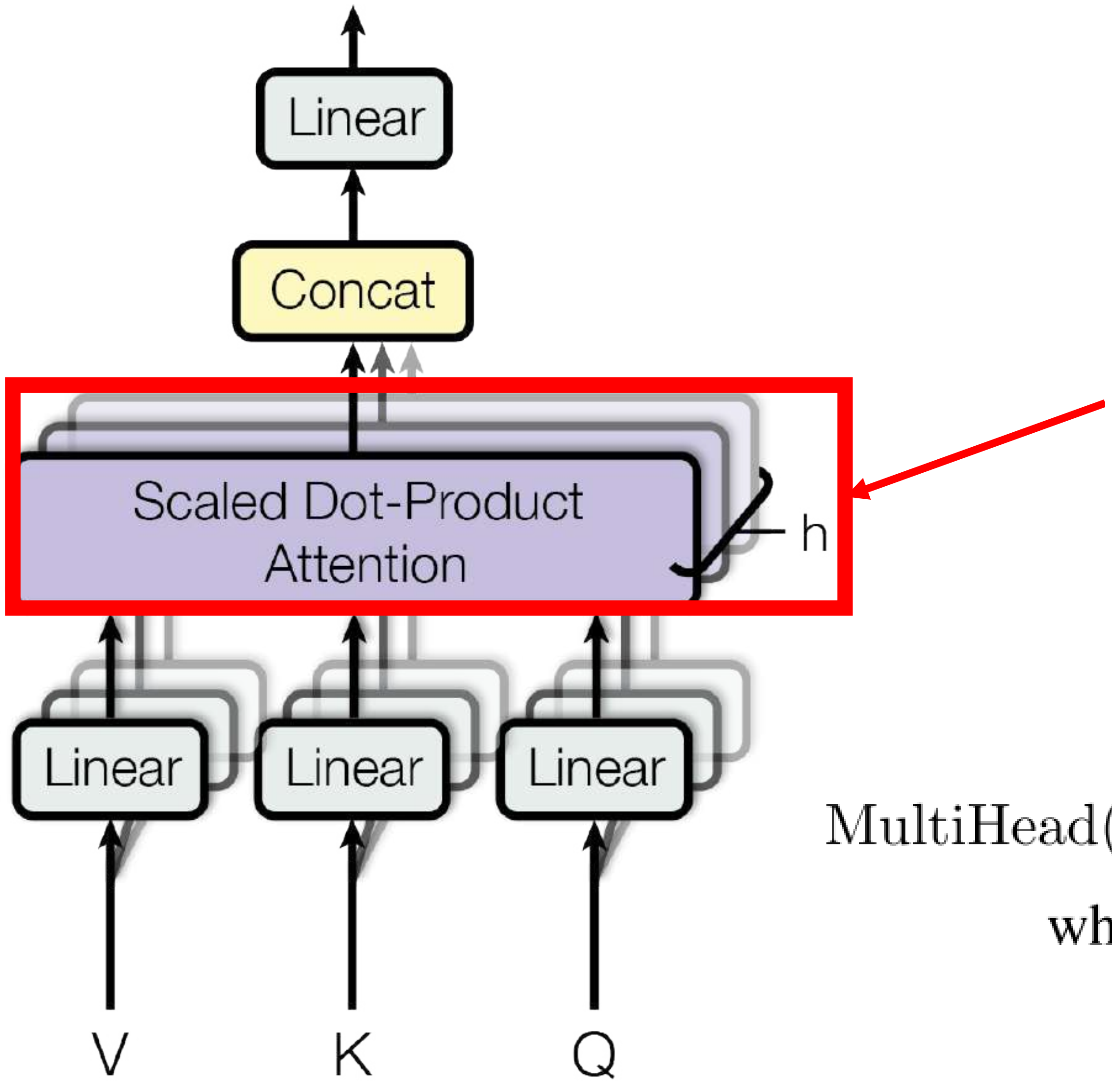
- The core network of the model $f_h(.;\ \theta_h)$:
  - Input: Glimpse representation g_t;
  - Input: Internal representation at previous time step h_{t-1};
  - Output: New internal state of the model h_t.
- The location network $f_l(.;\ \theta_l)$:
  - Input: Internal state h_t;
  - Output: location to attend to l_t.
- The action network $f_a(.;\ \theta_a)$:
  - Input: Internal state h_t;
  - Output: Action/classification a_t.



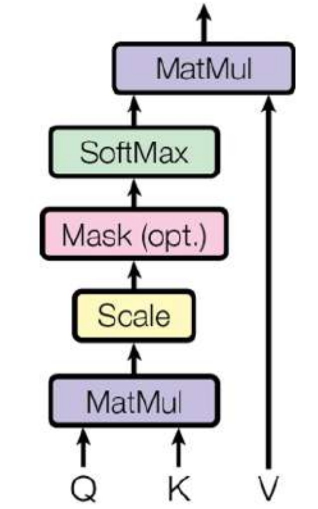
- The glimpse network $f_g(.;\{\theta_g^0,\theta_g^1,\theta_g^2\})$ defines a trainable bandwidth limited sensor for the attention network producing the glimpse representation g_t.
- Linear layers parameterized by $\theta_g^0,\theta_g^1,\theta_g^2$:
  - Activation Function : ReLu.



- Input:
  - Location $(l_t - 1)$;
  - Input image (x_t);
- Output:
  - Retina representation $\rho(x_t, l_{t-1})$;



Multi-Head Attention



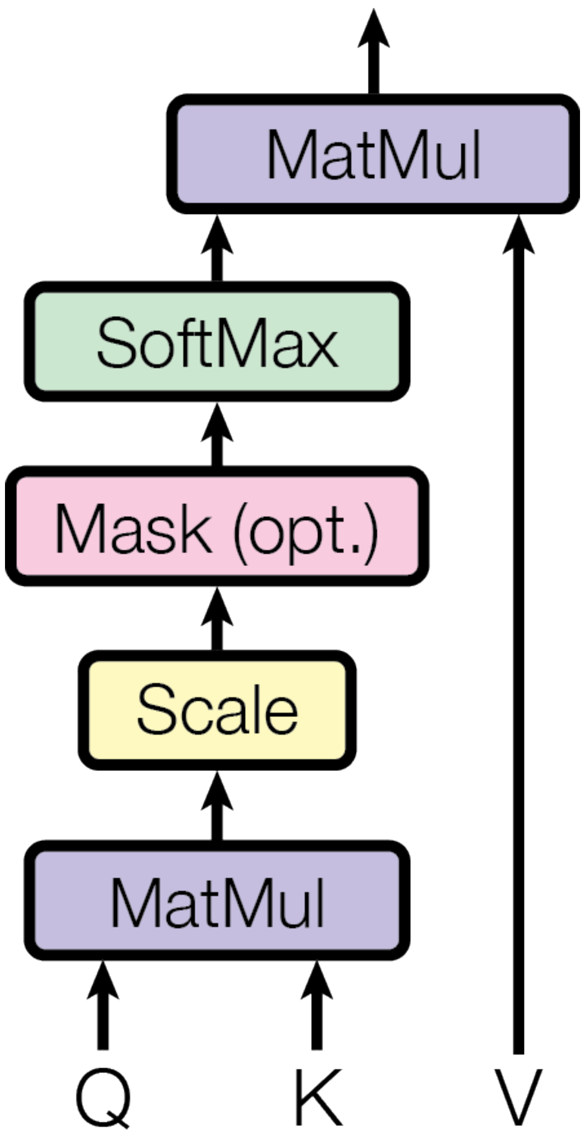
Scaled Dot-Product Attention

$$Attention(Q, K_i, V_i) = softmax(\frac{Q^T K_i}{\sqrt{d_k}})V_i$$

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

$$Q = \underbrace{\begin{pmatrix} q \\ q \\ q \\ \vdots \\ q \end{pmatrix}}_{d_k} \}m \quad K = \underbrace{\begin{pmatrix} k_1 \\ k_2 \\ k_3 \\ \vdots \\ k_m \end{pmatrix}}_{d_k} \}m \quad V = \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_m \end{pmatrix}}_{d_v} \}m$$

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

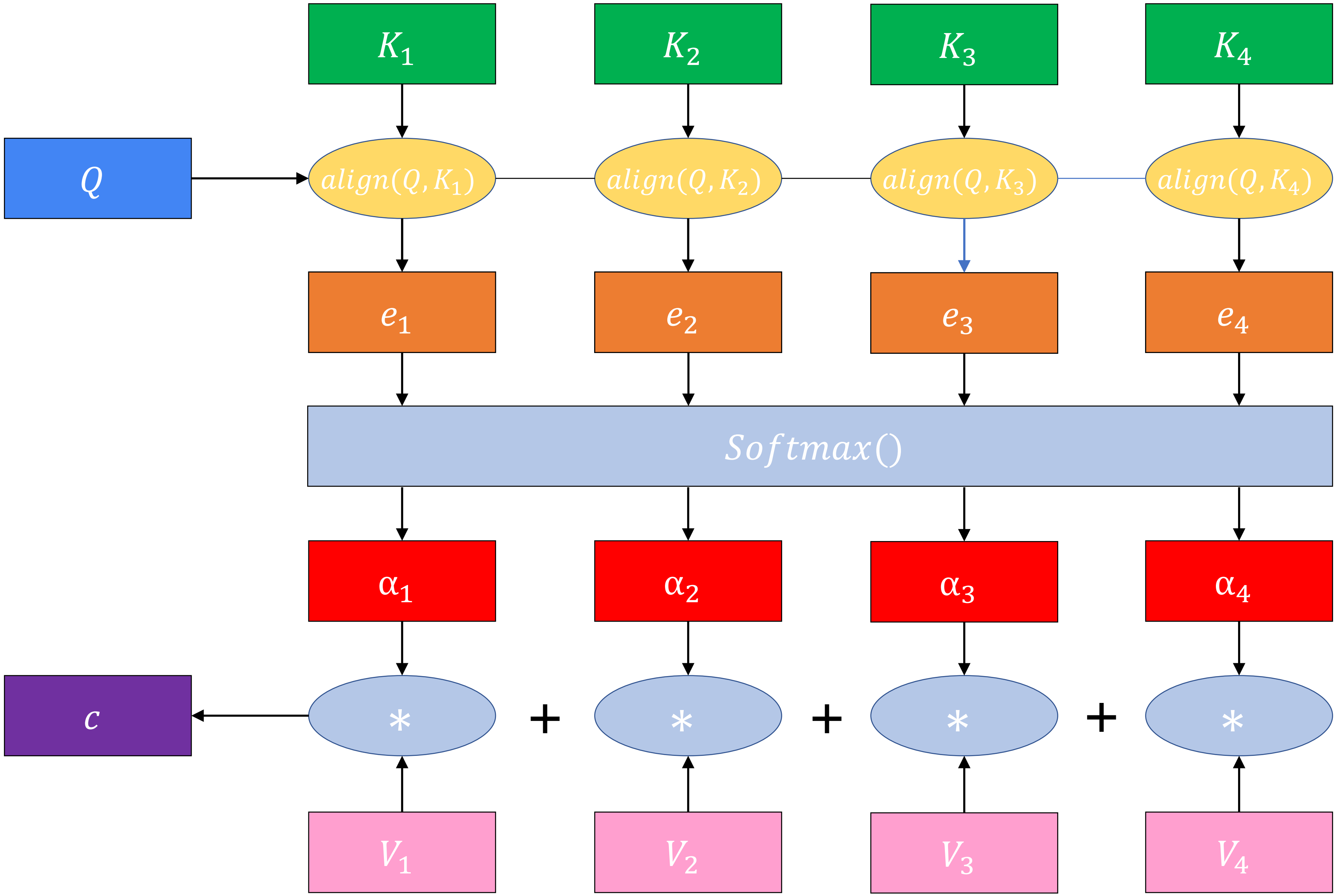


Scaled Dot-Product Attention

$$Attention(Q, K_i, V_i) = softmax(\frac{Q^T K_i}{\sqrt{d_k}})V_i$$

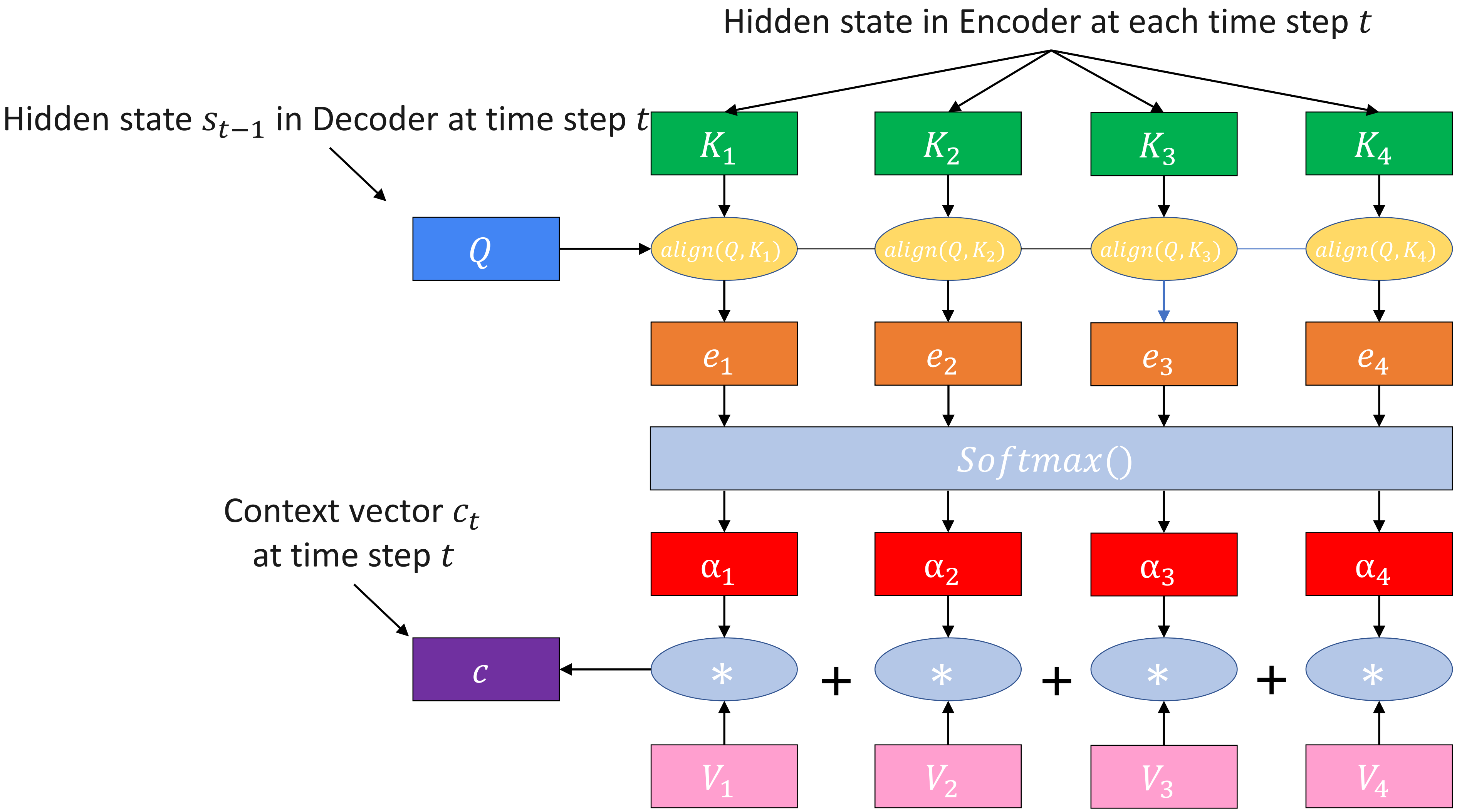
$$Attention(Q, K, V) = softmax(\frac{Q^T K}{\sqrt{d_k}})V$$

$$Q = \underbrace{\begin{pmatrix} q \\ q \\ q \\ \vdots \\ q \end{pmatrix}}_{d_k} \}m \qquad K = \underbrace{\begin{pmatrix} k_1 \\ k_2 \\ k_3 \\ \vdots \\ k_m \end{pmatrix}}_{d_k} \}m \qquad V = \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_m \end{pmatrix}}_{d_v} \}m$$

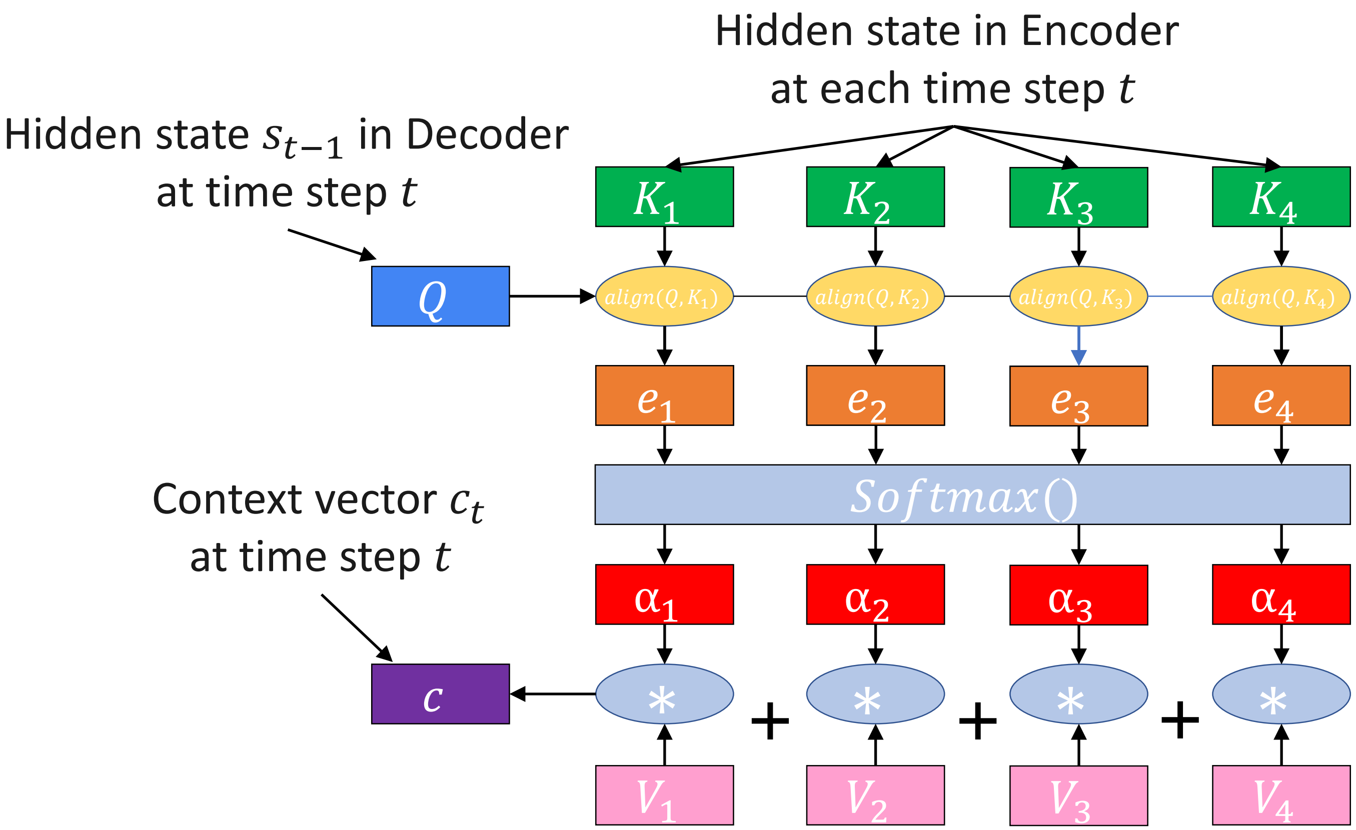


K_1
K_2
K_3
K_4
Q
$align(Q, K_1)$
$align(Q, K_2)$
$align(Q, K_3)$
$align(Q, K_4)$
e_1
e_2
e_3
e_4
$Softmax()$
α_1
α_2
α_3
α_4
c
$*$
+
$*$
+
$*$
+
$*$
V_1
V_2
V_3
V_4

Example: Machine Translation

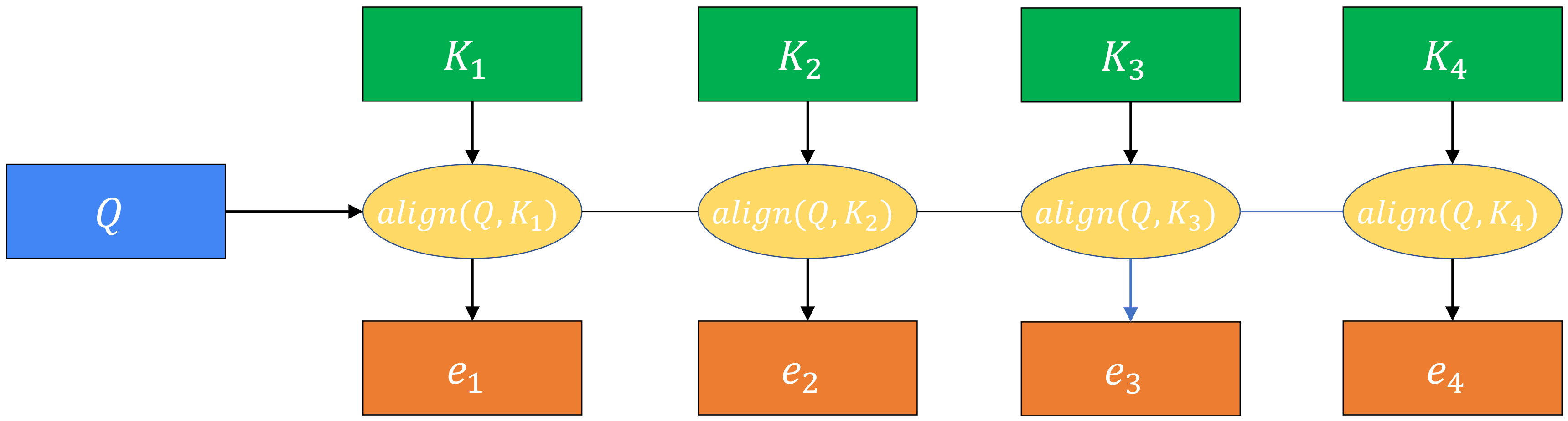


Example: Machine Translation (Cont.)

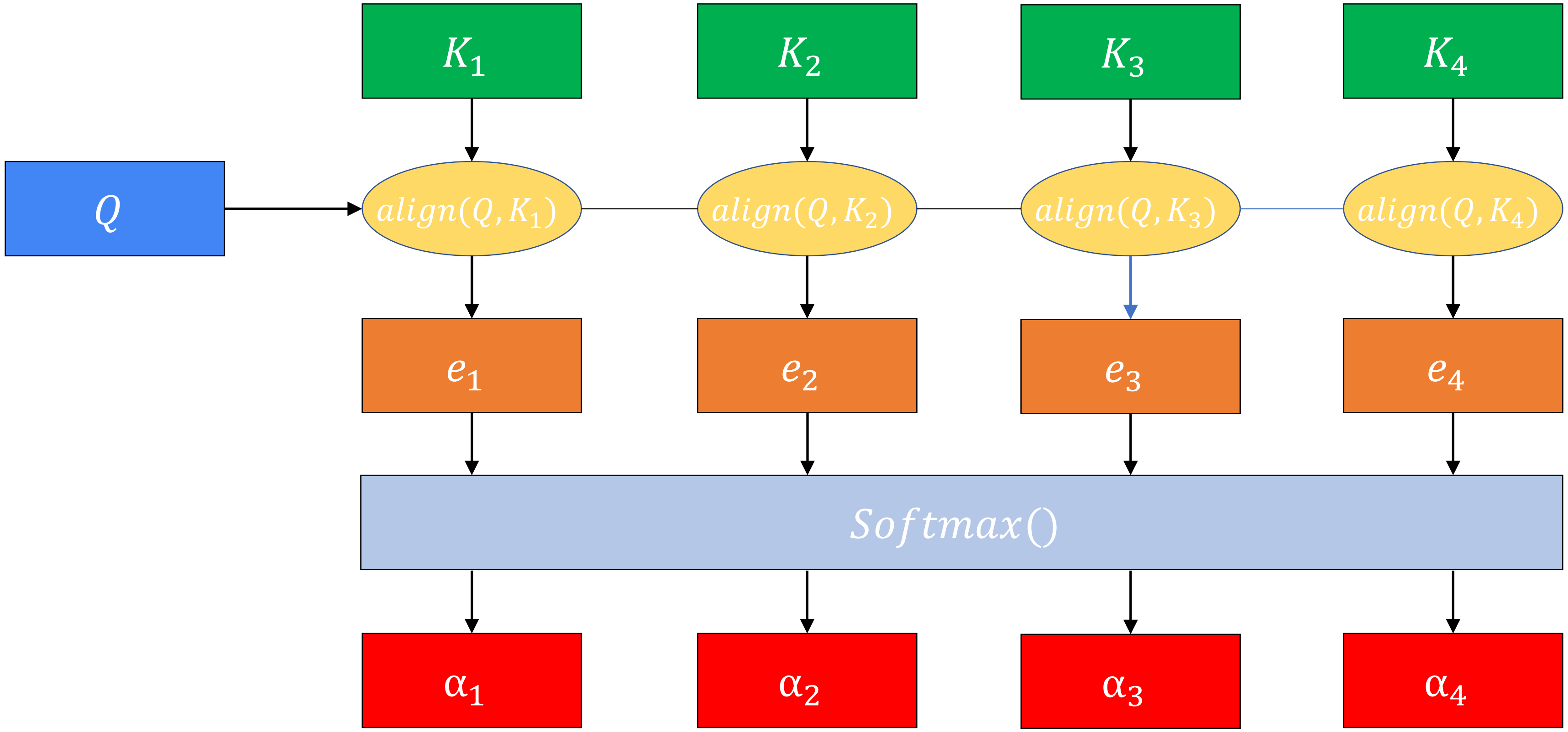


There are three input time steps: $x_1, x_2, \dots, x_n$

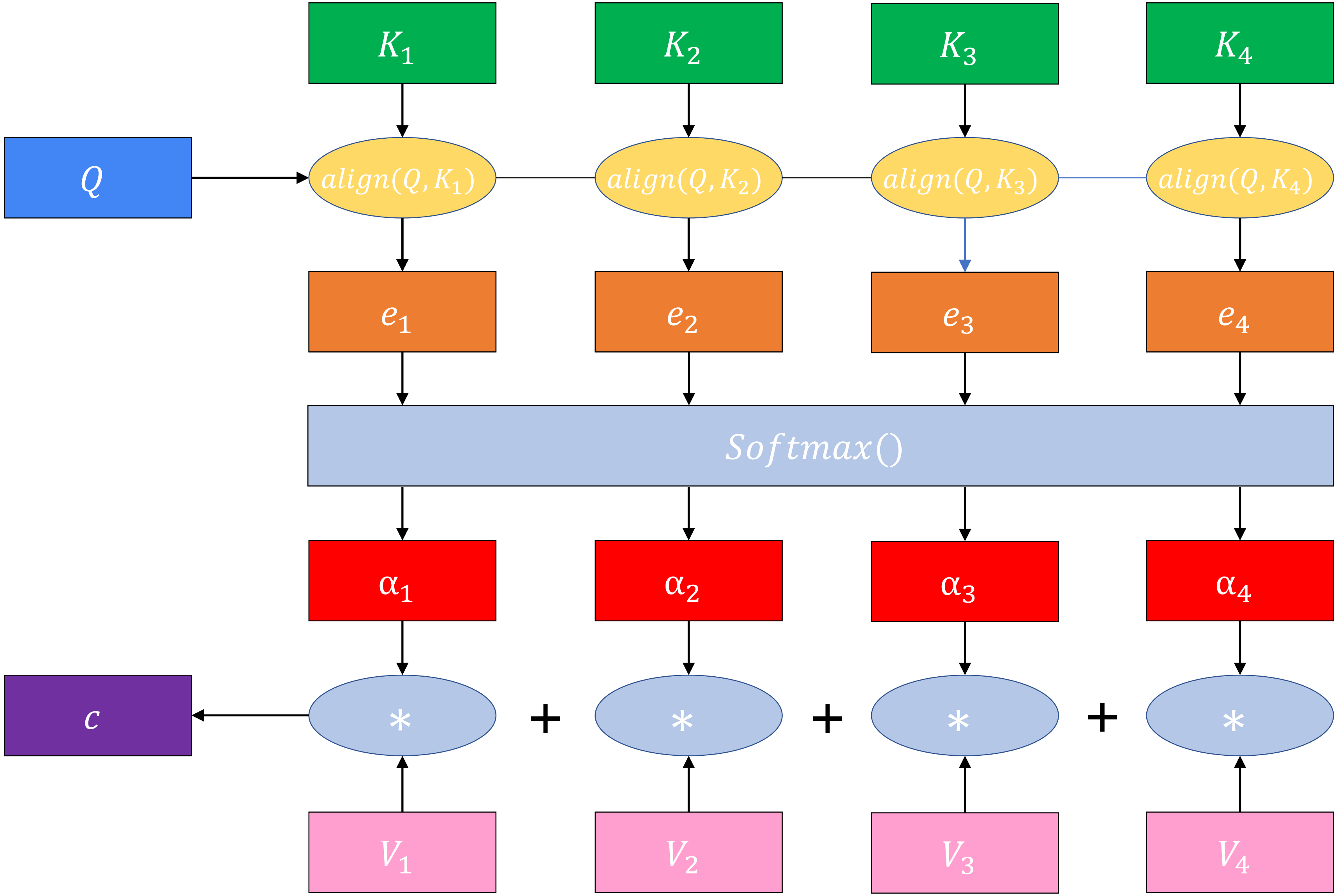
- Encoding
  - $h_1, h_2, \dots, h_n = Encoder(x_1, x_2, \dots, x_n)$
- Alignment
  - Scores how well each encoded input matches the current output of the decoder at time step t: $e_{t,i} = align(s_{t-1}, h_i), i = 1 \dots n$
- Weighting
  - $\alpha_{t,i} = \sigma_i(e_t) = \frac{\exp(e_{t,i})}{\sum_{j=1}^{n} \exp(e_{t,j})}$
- Context Vector
  - $c_t = \sum_{j=1}^{n} \alpha_{t,j} h_j$
- Decode
  - $s_t = \mathrm{Decoder}(c_t)$



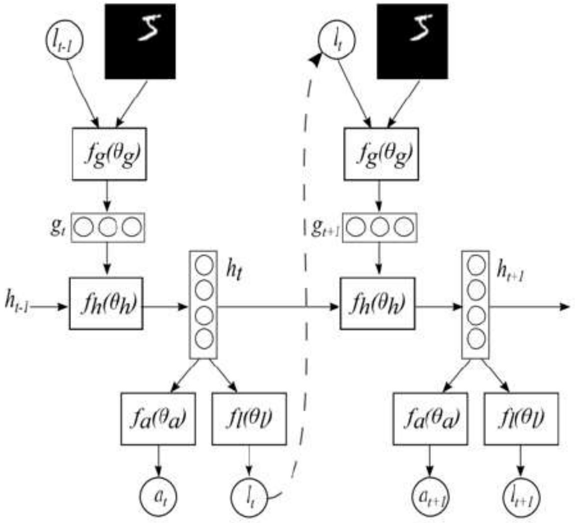
Step 1: Get weight values from the similarity of the query and the key using alignment scores



Step 2: Normalize weight values and get available weights



Step 3: Get weighted sum of weight and value

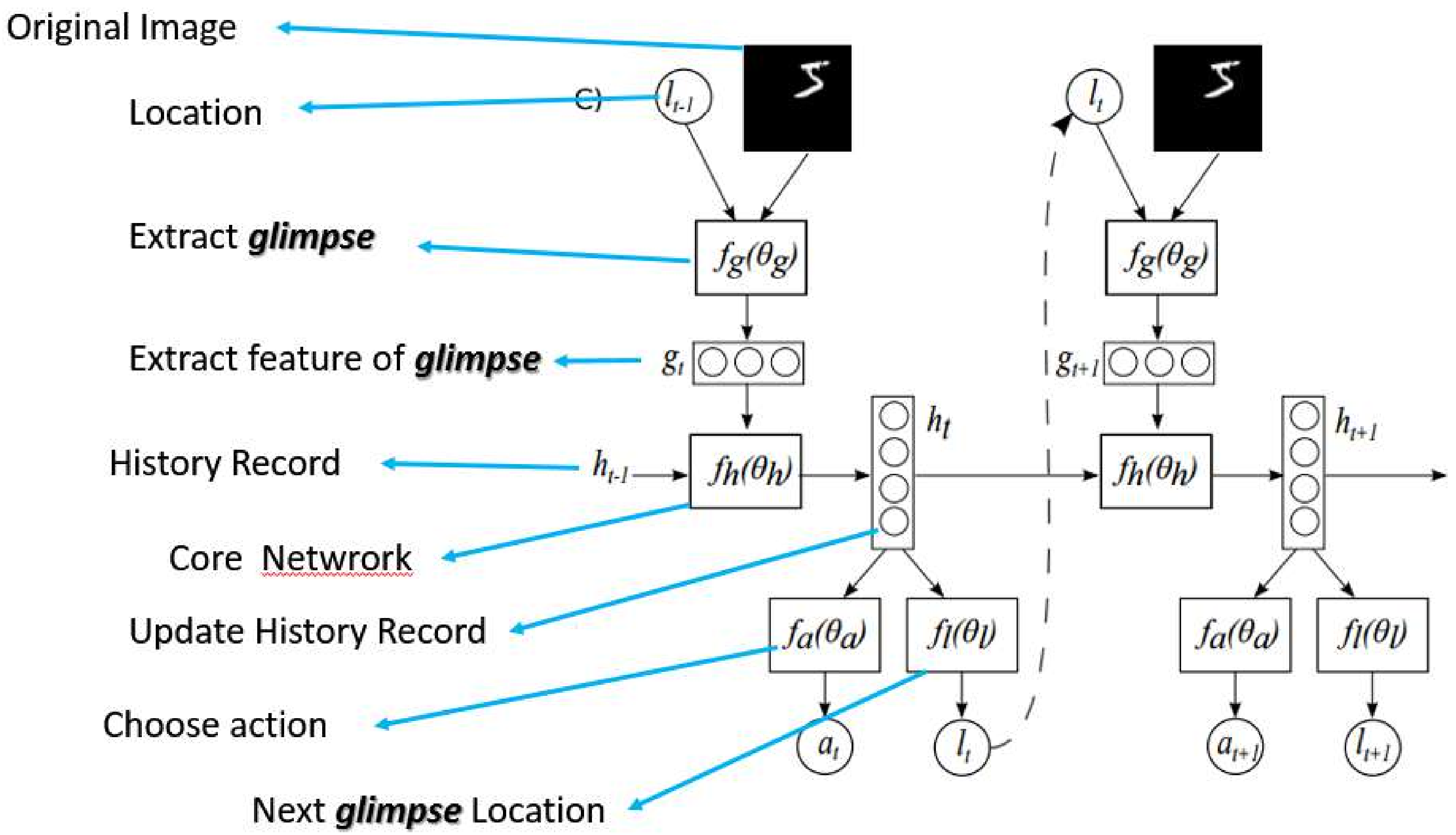


- The core network of the model $f_h(.;\ \theta_h)$:
  - Input: Glimpse representation g_t;
  - Input: Internal representation at previous time step h_{t-1};
  - Output: New internal state of the model h_t.
- The location network $f_l(.;\ \theta_l)$:
  - Input: Internal state h_t;
  - Output: location to attend to l_t.
- The action network $f_a(.;\ \theta_a)$:
  - Input: Internal state h_t;
  - Output: Action/classification a_t.

Mnih Volodymyr, Nicolas Heess, and Alex Graves. "Recurrent models of visual attention." *Advances in neural information processing systems*. 2014. [arxiv]

- Consider the attention problem as the **sequential decision process** of a goal-directed agent interacting with a visual environment;
- At each time step t:
  - The agent observes the environment only via a bandwidth-limited sensor;
  - The agent can actively control how to deploy its sensor resources and affect the true state of the environment by executing actions;
  - The agent receives a scalar reward.

Recurrent Attention Model (Cont.)



Mnih Volodymyr, Nicolas Heess, and Alex Graves. "Recurrent models of visual attention." *Advances in neural information processing systems*. 2014. [arxiv]

Recurrent Attention Model (Cont.)

Action:

- Location action l_t :
  - Decides how to deploy its sensor via the sensor control;
  - Chosen stochastically, $l_t \sim p(\cdot\,|f_l(h_t;\,\theta_l))$ at time t
- Environment action a_t:
  - Might affect the state of the environment
  - Chosen stochastically, $a_t \sim p(\cdot\,|f_a(h_t;\,\theta_a))$ at time t

Recurrent Attention Model (Cont.)

Reward:

- After executing an action the agent receives a new visual observation of the environment x_{t+1} and a reward signal r_{t+1};
- Reward Function: $R = \sum_{t=1}^{T} \gamma^{t-1} r_t$
  - For object recognition: $r_T = 1$ if the object is classified correctly after T steps and 0 otherwise.

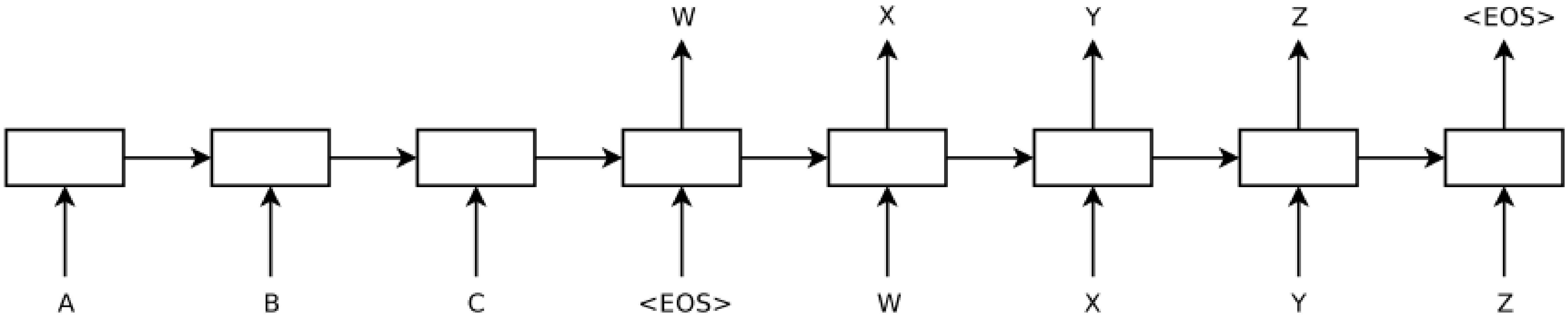
Recurrent Attention Model (Cont.)

Cost Function:

$$J(\theta) = E_{p(s_{\mathbb{1}},T;\theta)}\left[\sum_{t=1}^{T} r_t\right] = E_{p(s_{\mathbb{1}},T;\theta)}[R]$$

$$\nabla_\theta J = \sum_{t=1}^{T} E_{p(s_{\mathbb{1}:T};\theta)}[\nabla_\theta \log \pi\,(u_t|s_{1:t};\theta)R] \approx \frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{T} \nabla_\theta \log \pi\left(u_t^i \middle| s_{1:t}^i;\theta\right)R^i$$

where, $\theta = \{\theta_q, \theta_h, \theta_a\}$ and $s^{i'}$ s are interaction sequences obtained by running the current agent π_θ for $i = 1 \ldots M$ episodes.



Sutskever, I., O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks." Advances in NIPS 2014. [arxiv]

Overview

- Input: Sequence $(x_1, \dots, x_T)$
- Output: Sequence $(y_1, \dots, y_{T'})$
  - Note: T may differ from T'
- Goal: Estimate the conditional probability

$$p(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots, y_{t-1})$$

where, v is the fixed-dimensional representation of the input sequence from the encoder

- Training objective:

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p\,(T|S)$$

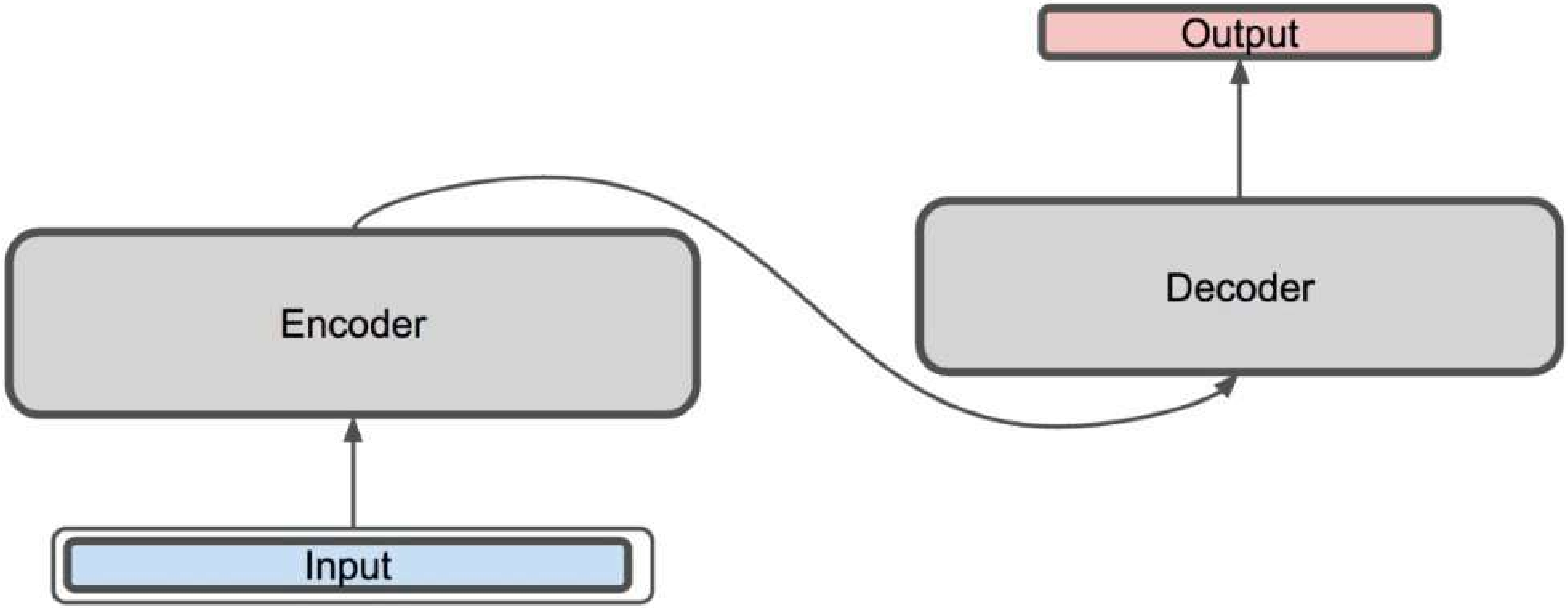
where, $\mathcal{S}$ is the training set.

Overview

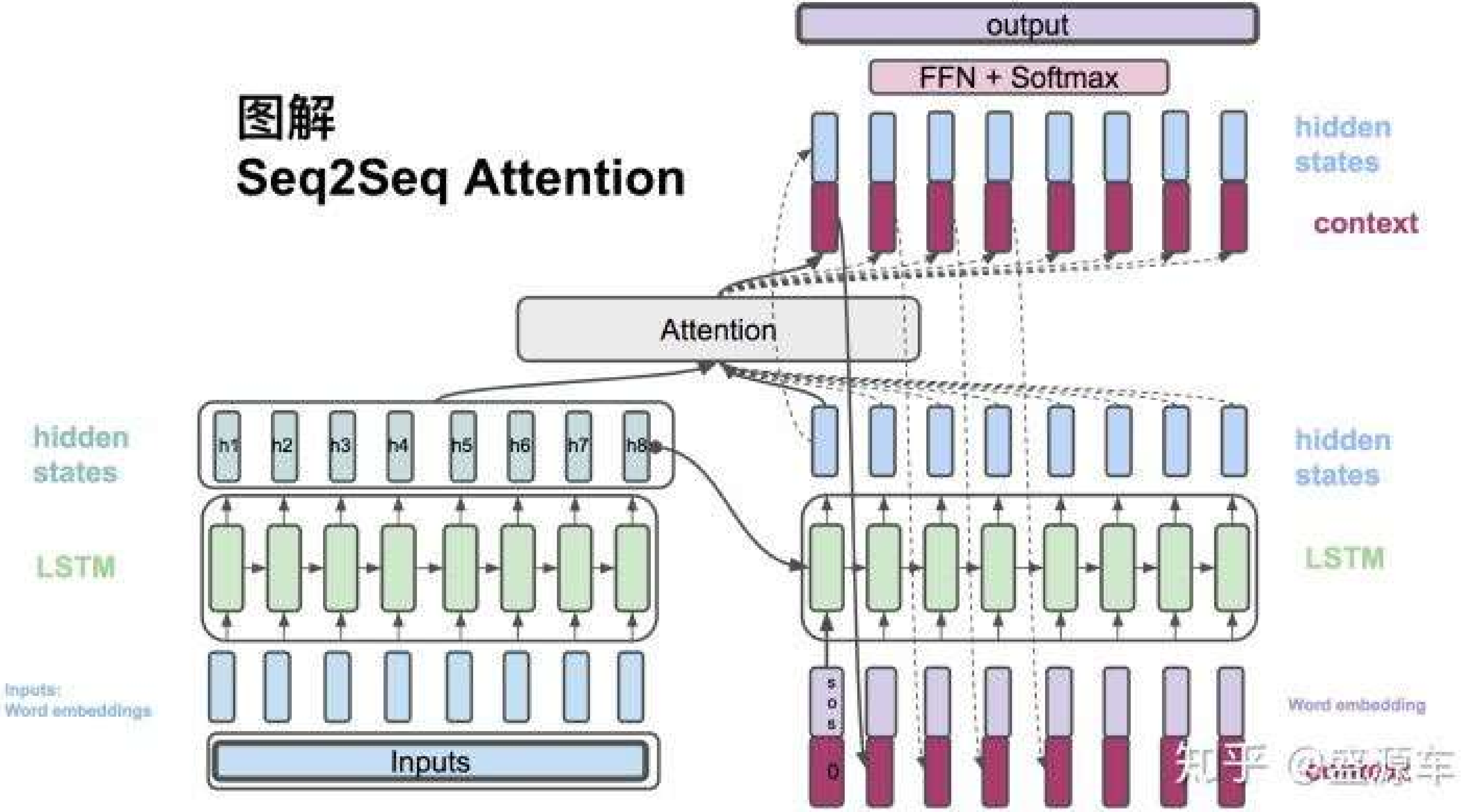
- Predict:

$$\hat{T} = \arg\max_{T} p\,(T|S)$$

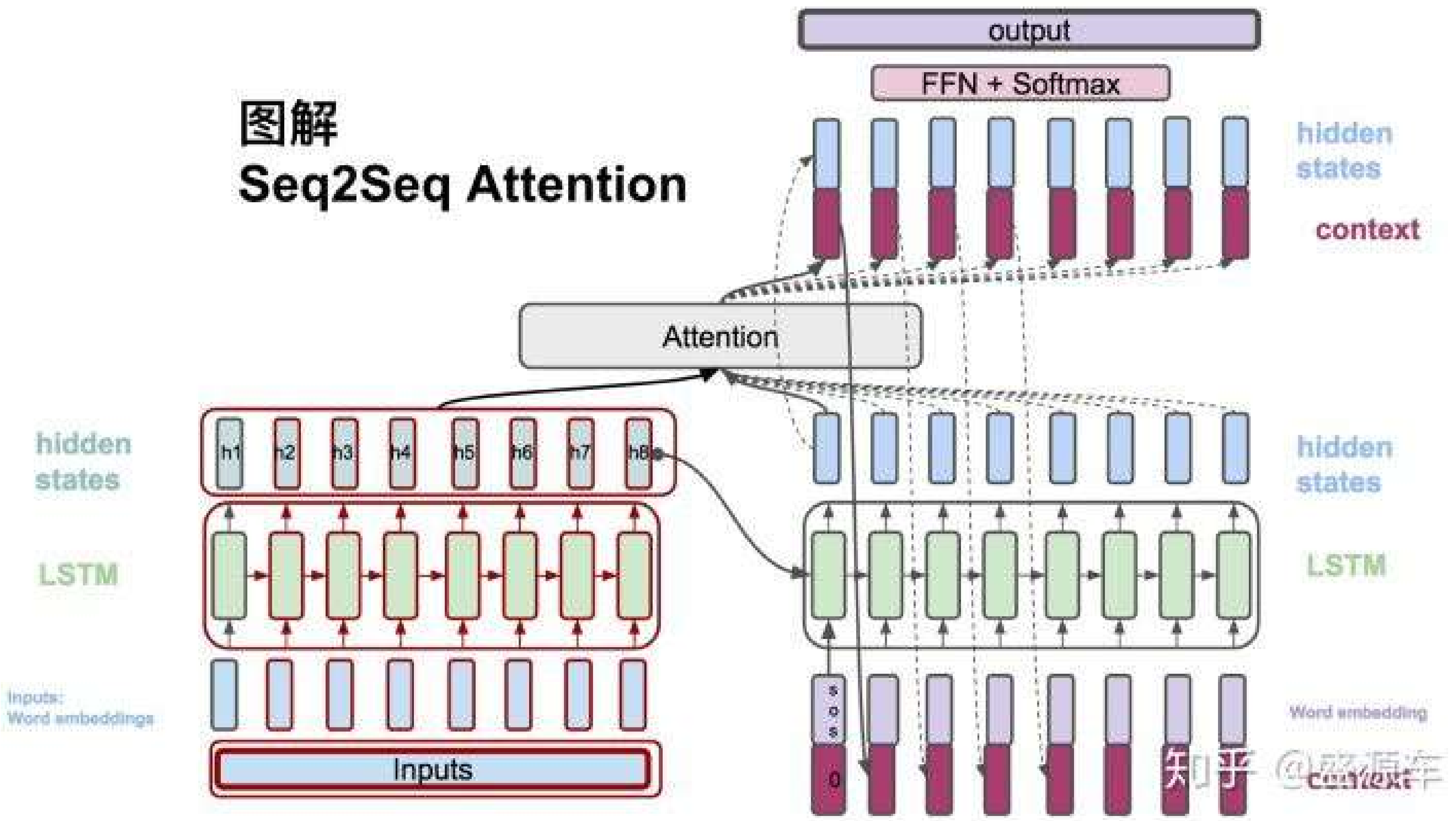
- Left-to-right Beam Search Decoder with Beam Width B:
  - At each timestep, extend each partial hypothesis in the beam with every possible word in the vocabulary.
  - Keep the top B partial hypothesis in the beam and discard others
  - As soon as the “<EOS>” symbol is appended, it is removed from the beam and is added to the set of complete hypotheses.



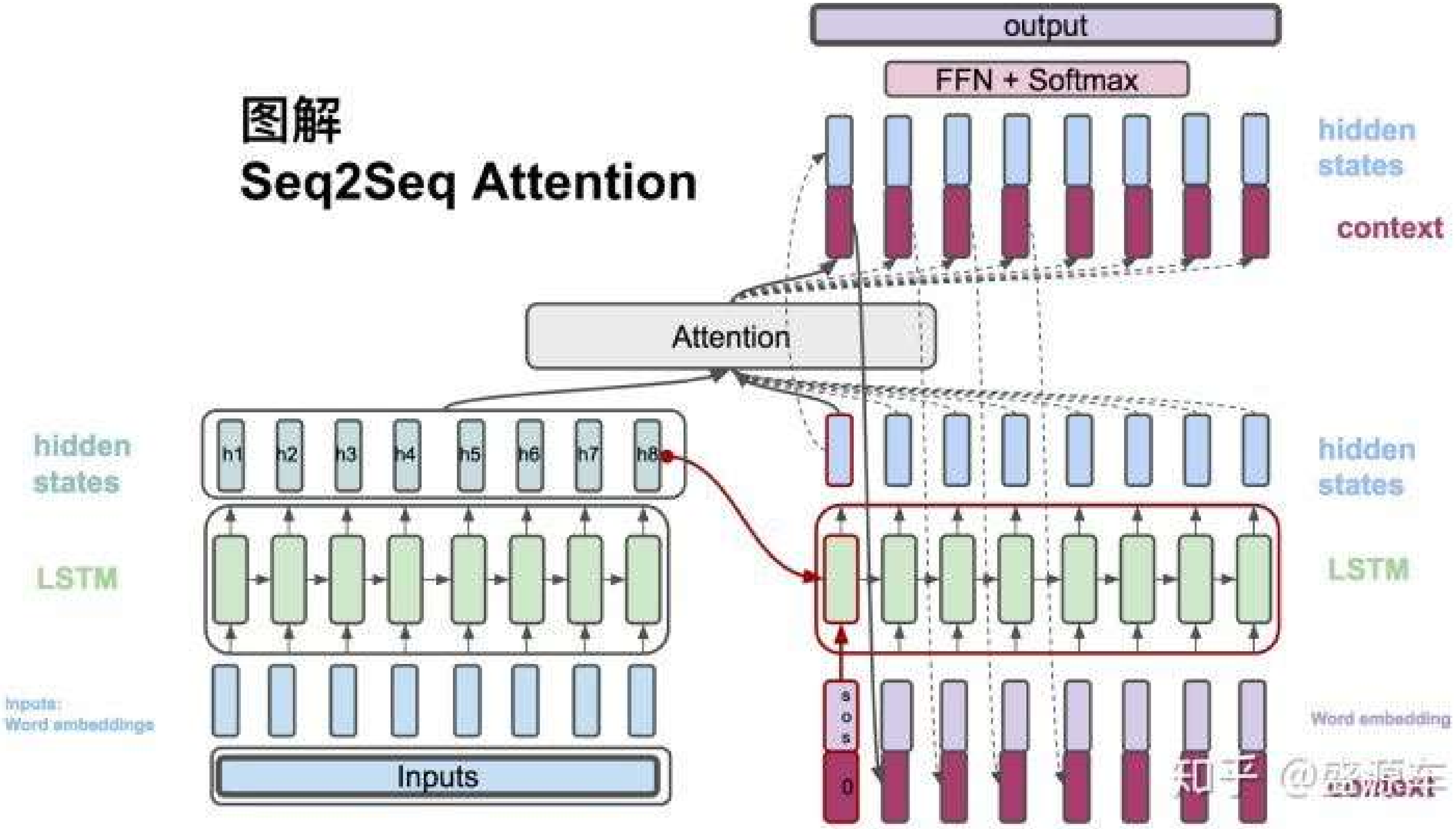
Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025.* (2015). [arxiv]



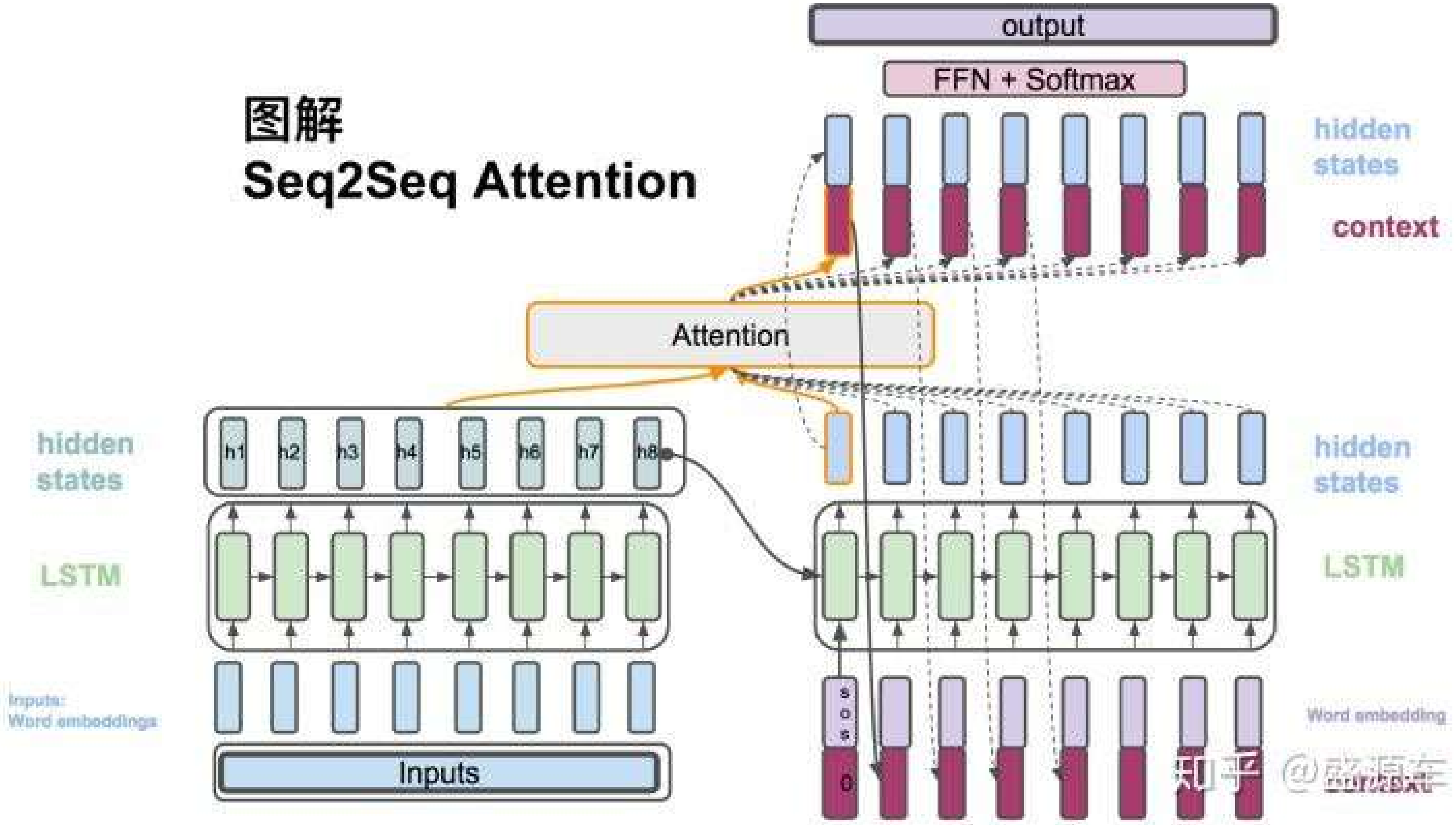
Yuanche.Sh “真正的完全图解Seq2Seq Attention模型” 知乎. 2019. [知乎专栏]



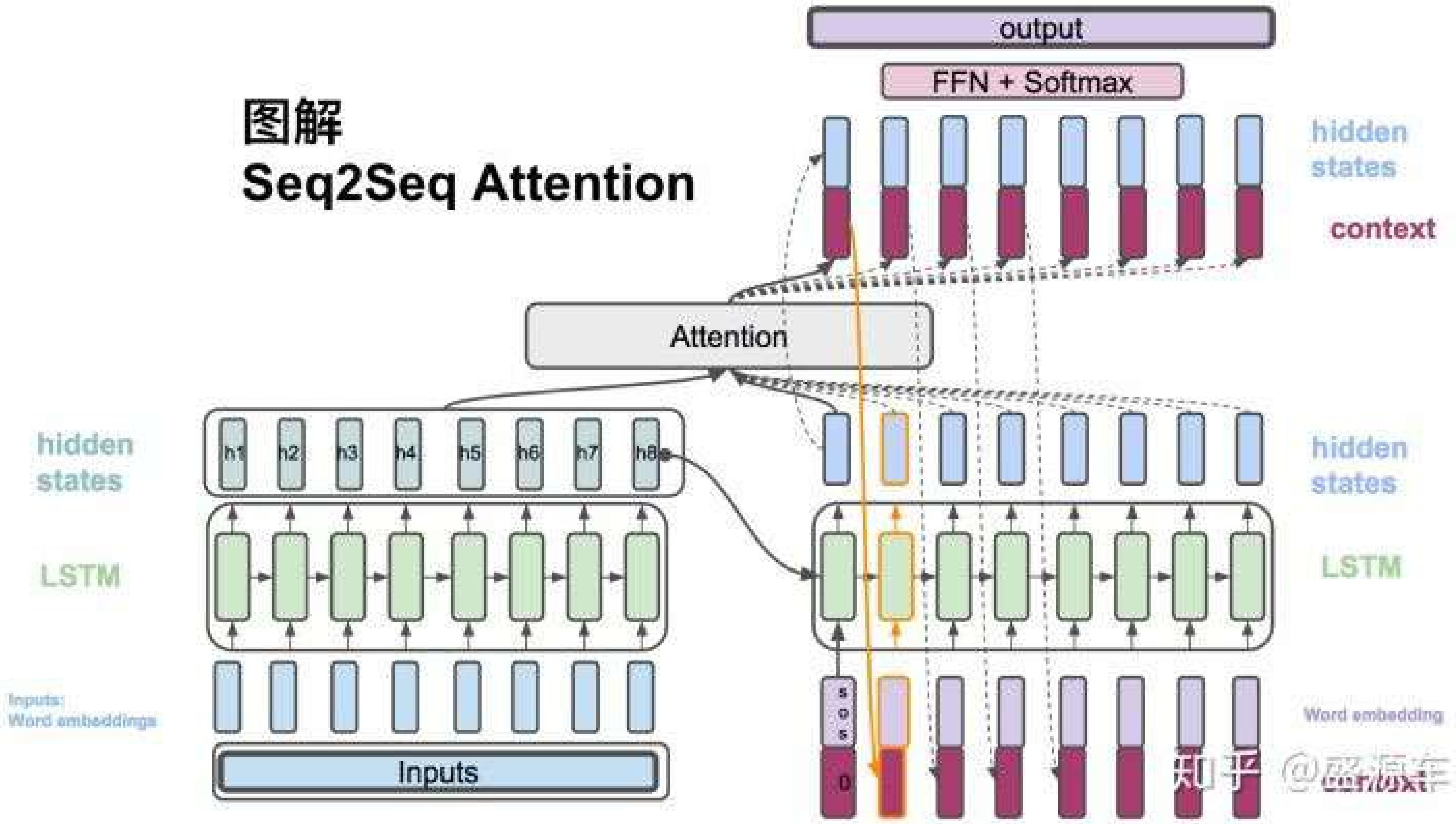
Yuanche.Sh “真正的完全图解Seq2Seq Attention模型”知乎. 2019. [知乎专栏]



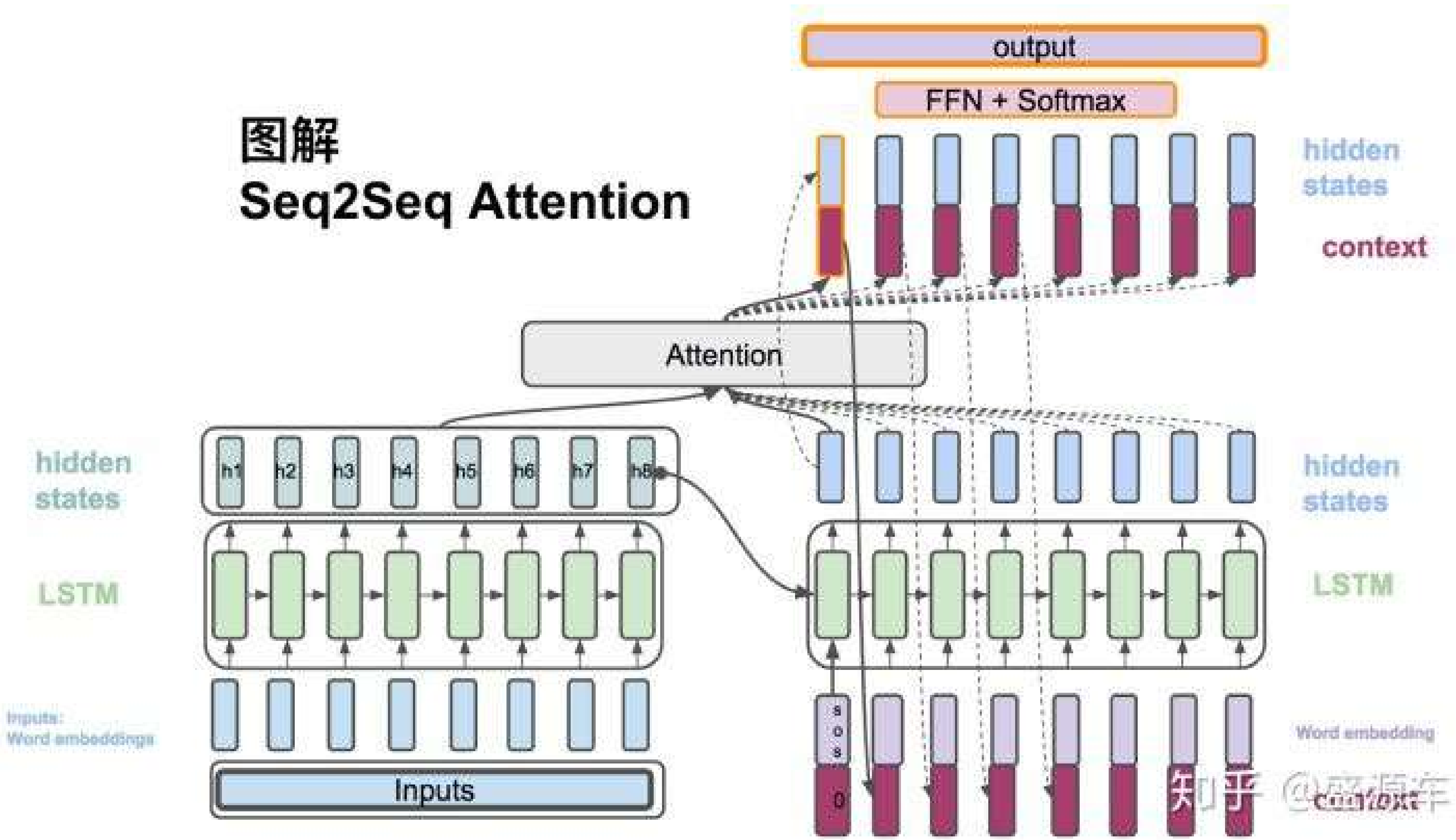
Yuanche.Sh “真正的完全图解Seq2Seq Attention模型” 知乎. 2019. [知乎专栏]



Yuanche.Sh “真正的完全图解Seq2Seq Attention模型” 知乎. 2019. [知乎专栏]

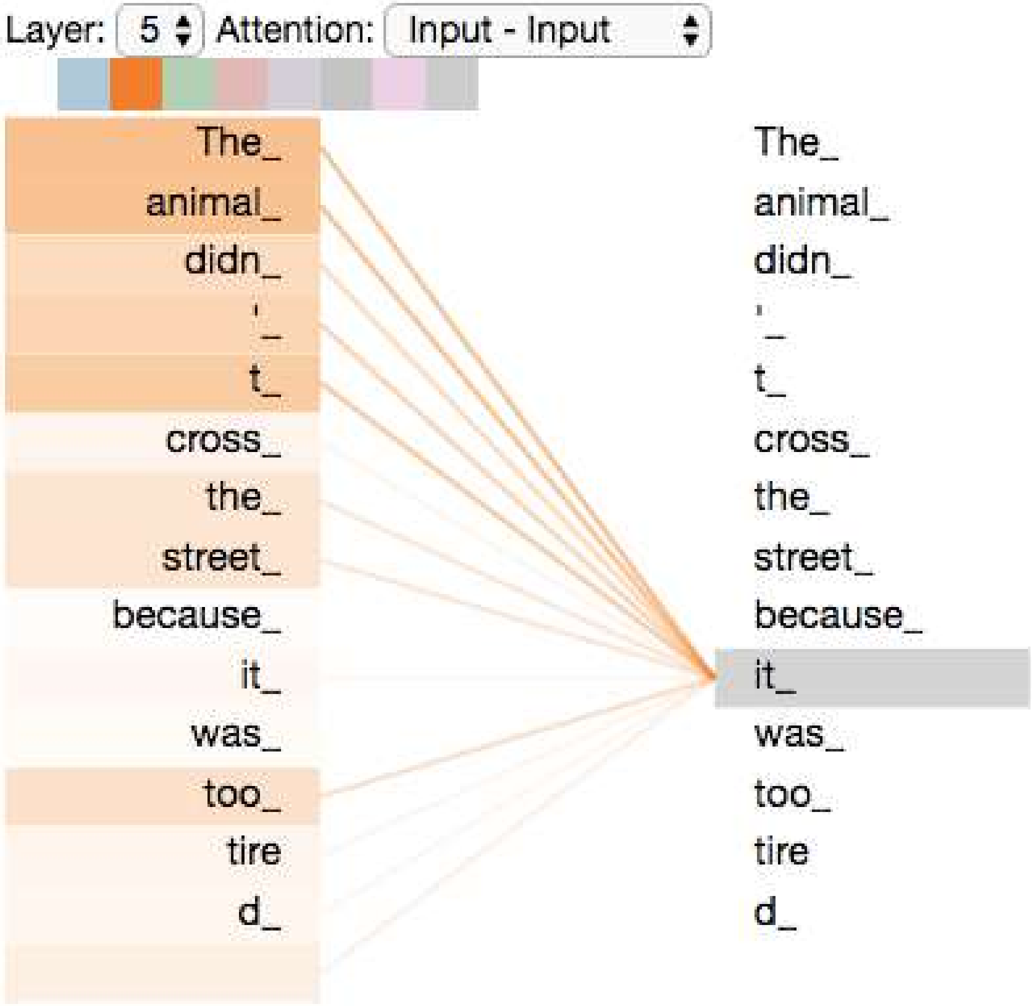


Yuanche.Sh “真正的完全图解Seq2Seq Attention模型”知乎. 2019. [知乎专栏]

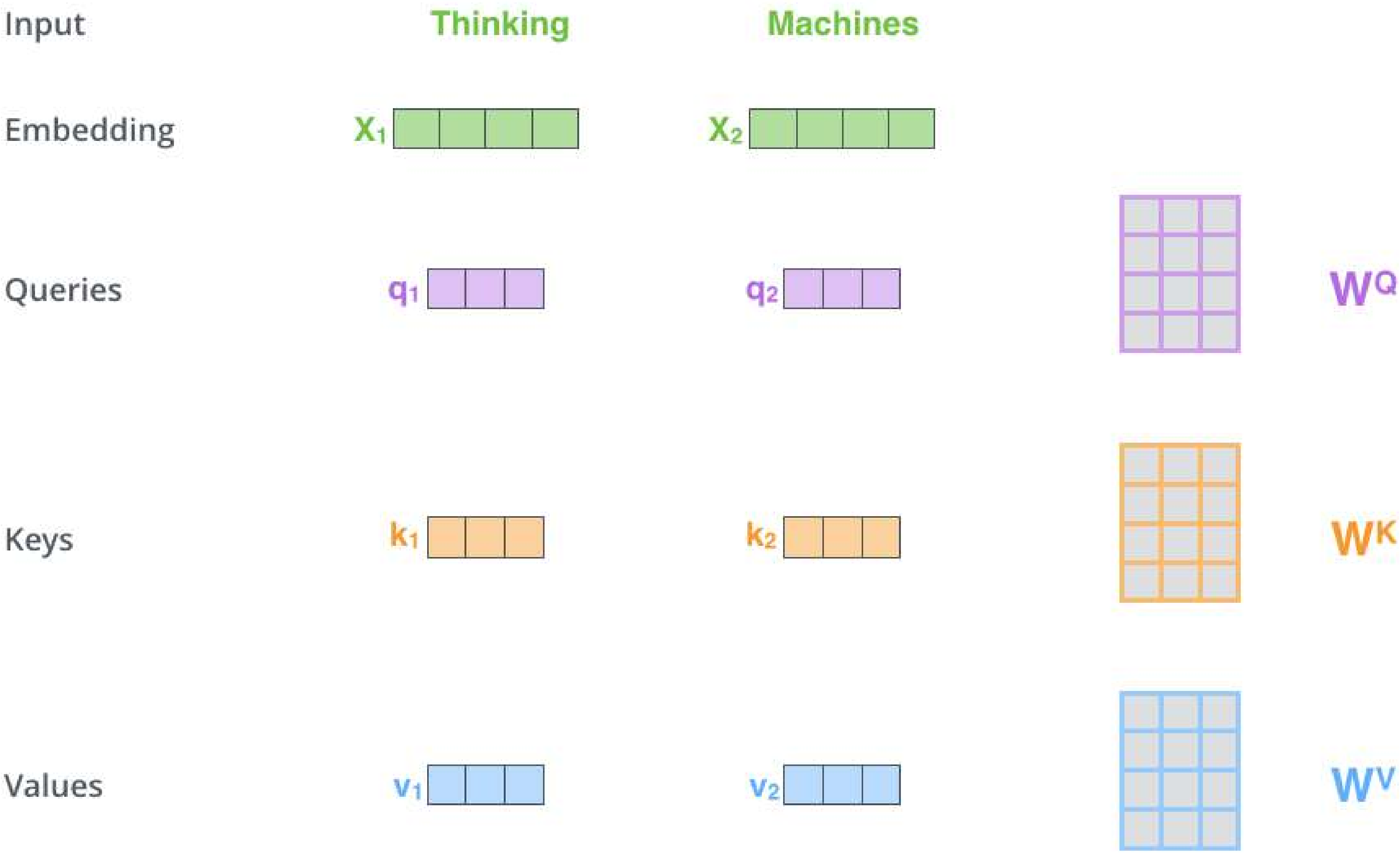


Yuanche.Sh “真正的完全图解Seq2Seq Attention模型”知乎. 2019. [知乎专栏]

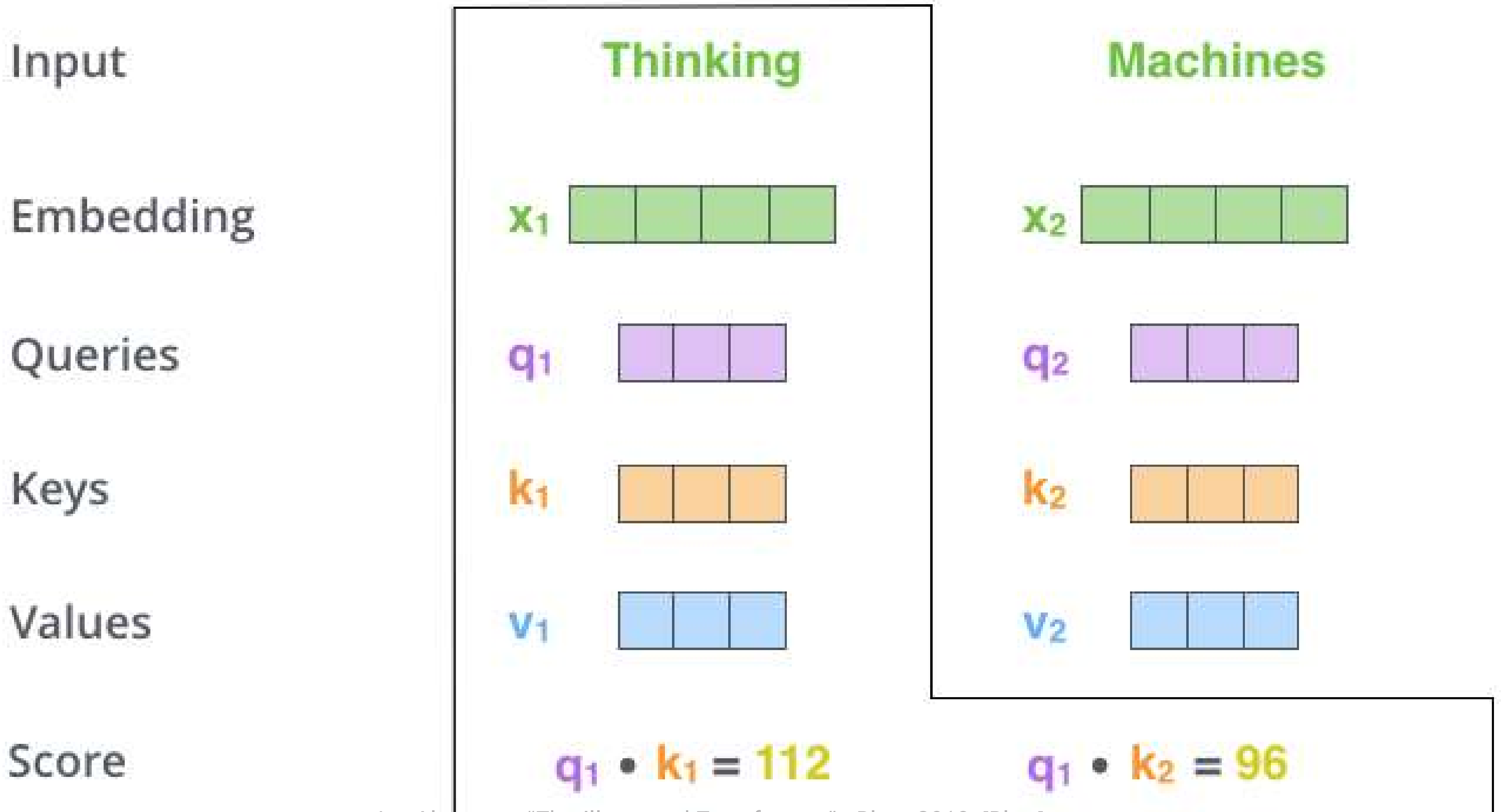
The animal didn't cross the street because it was too tired.



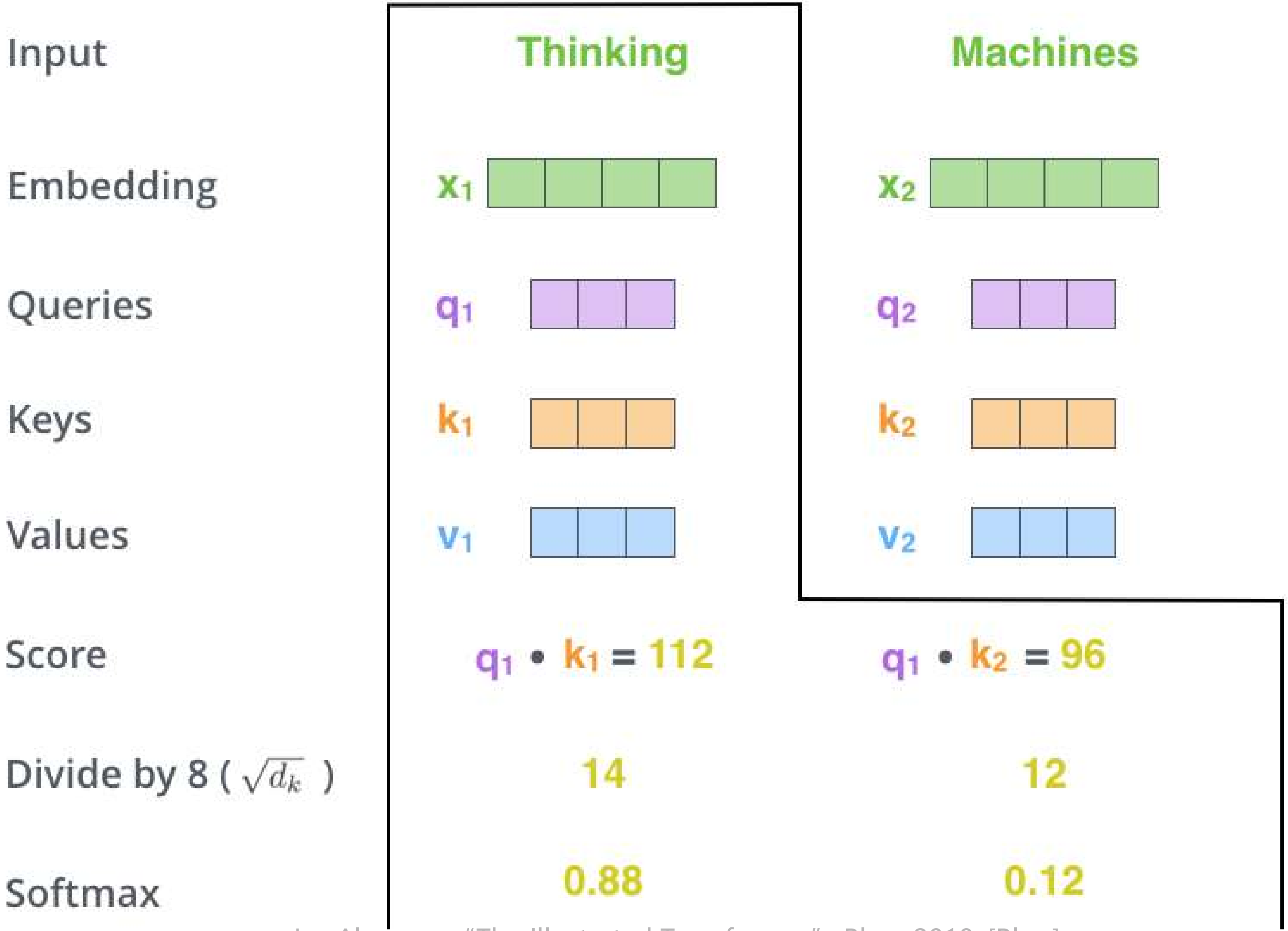
Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]



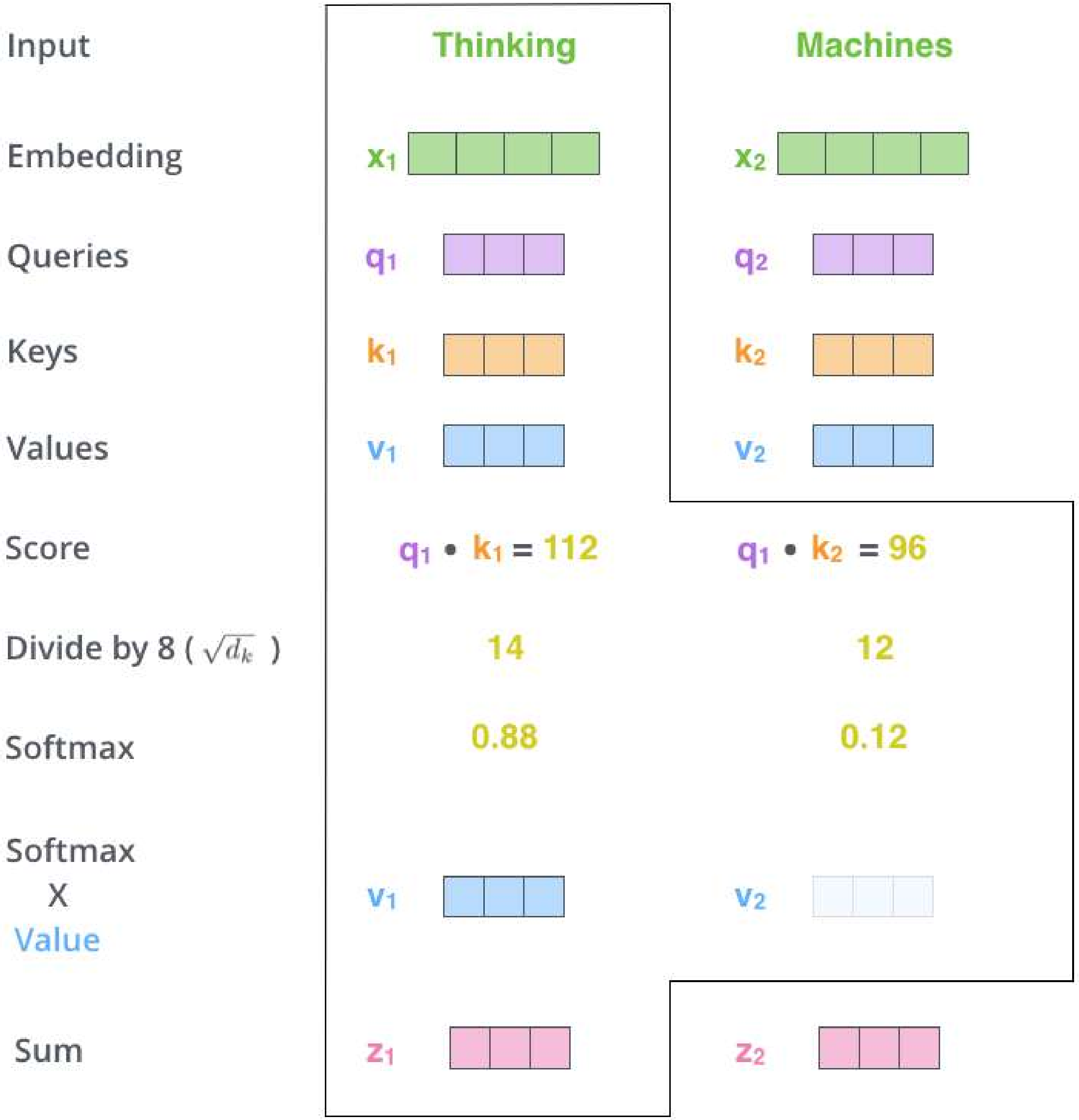
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



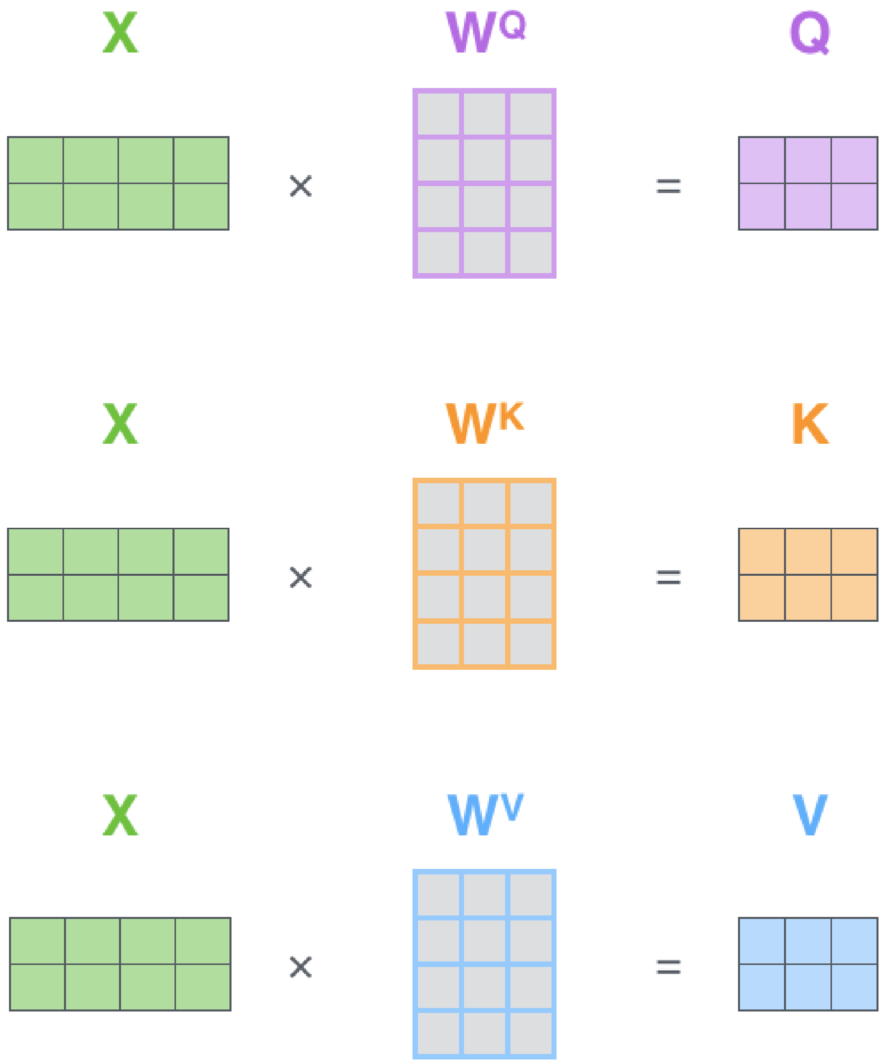
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



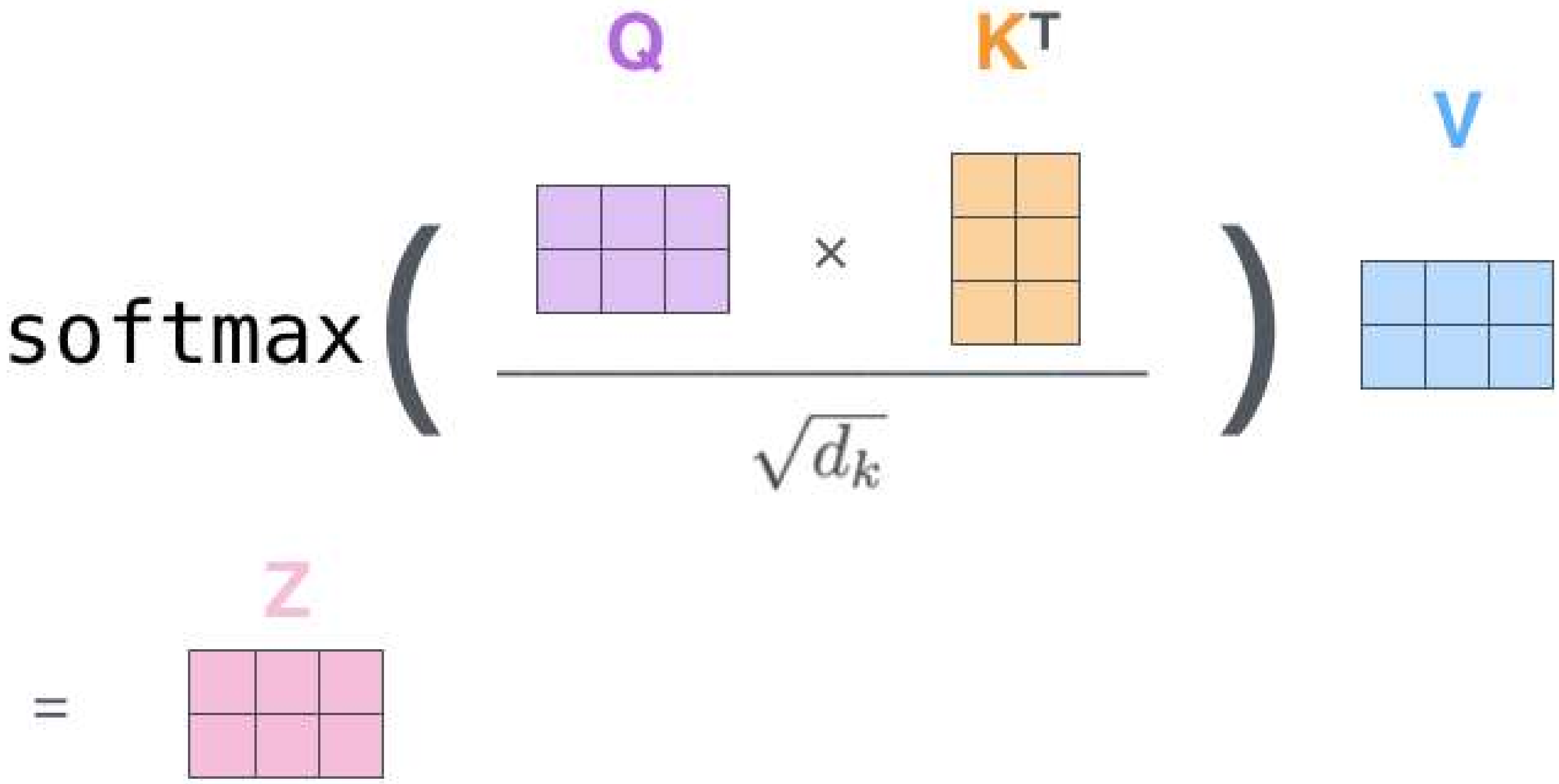
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]

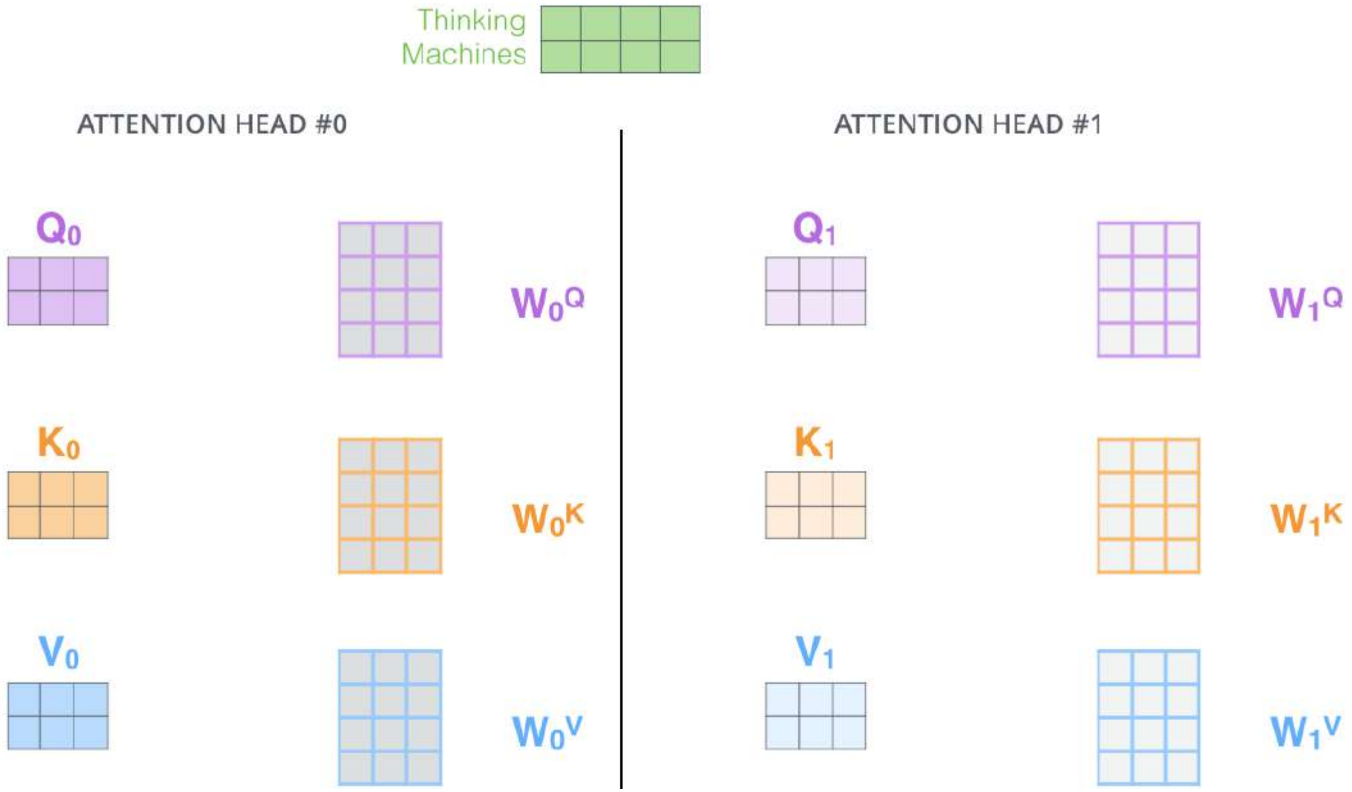


Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]

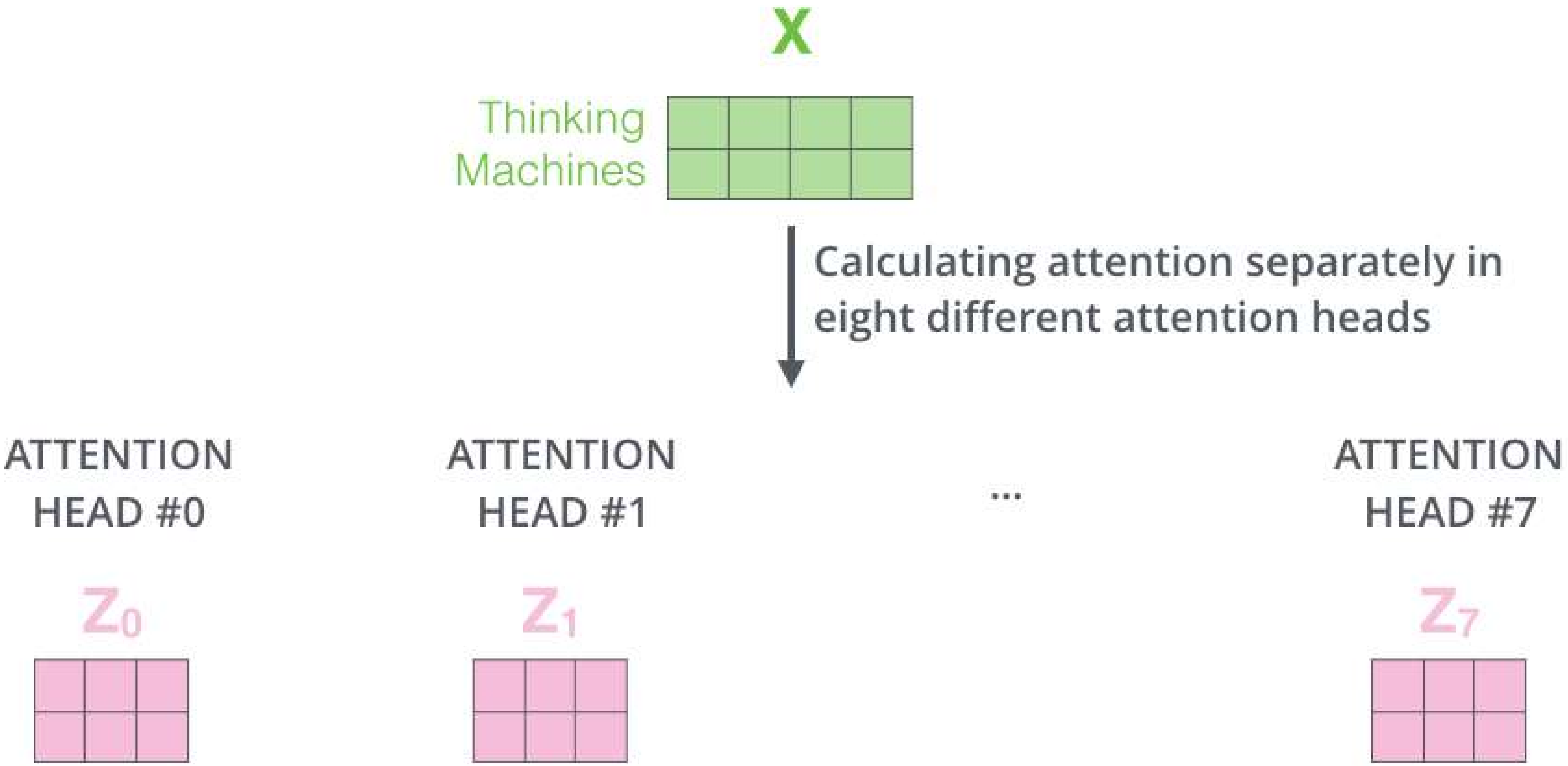


Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]

- Expands the model’ s ability to focus on different positions
- Gives the attention layer multiple “representation subspaces”

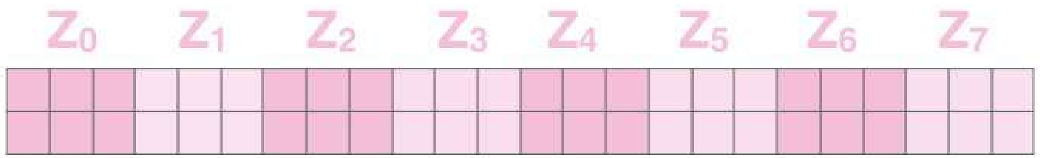


Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



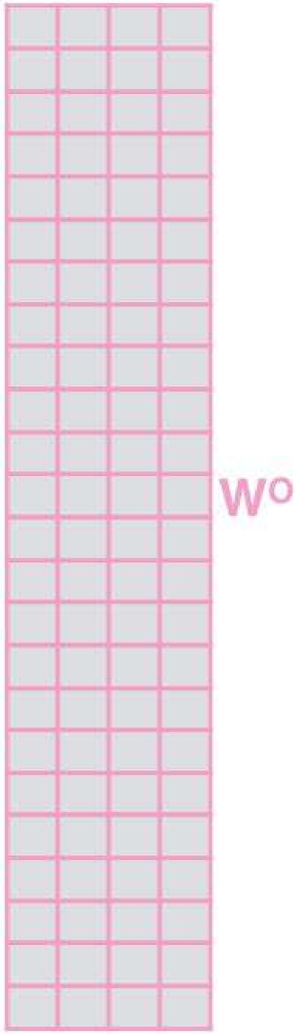
Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]

1) Concatenate all the attention heads



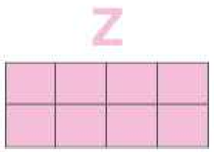
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

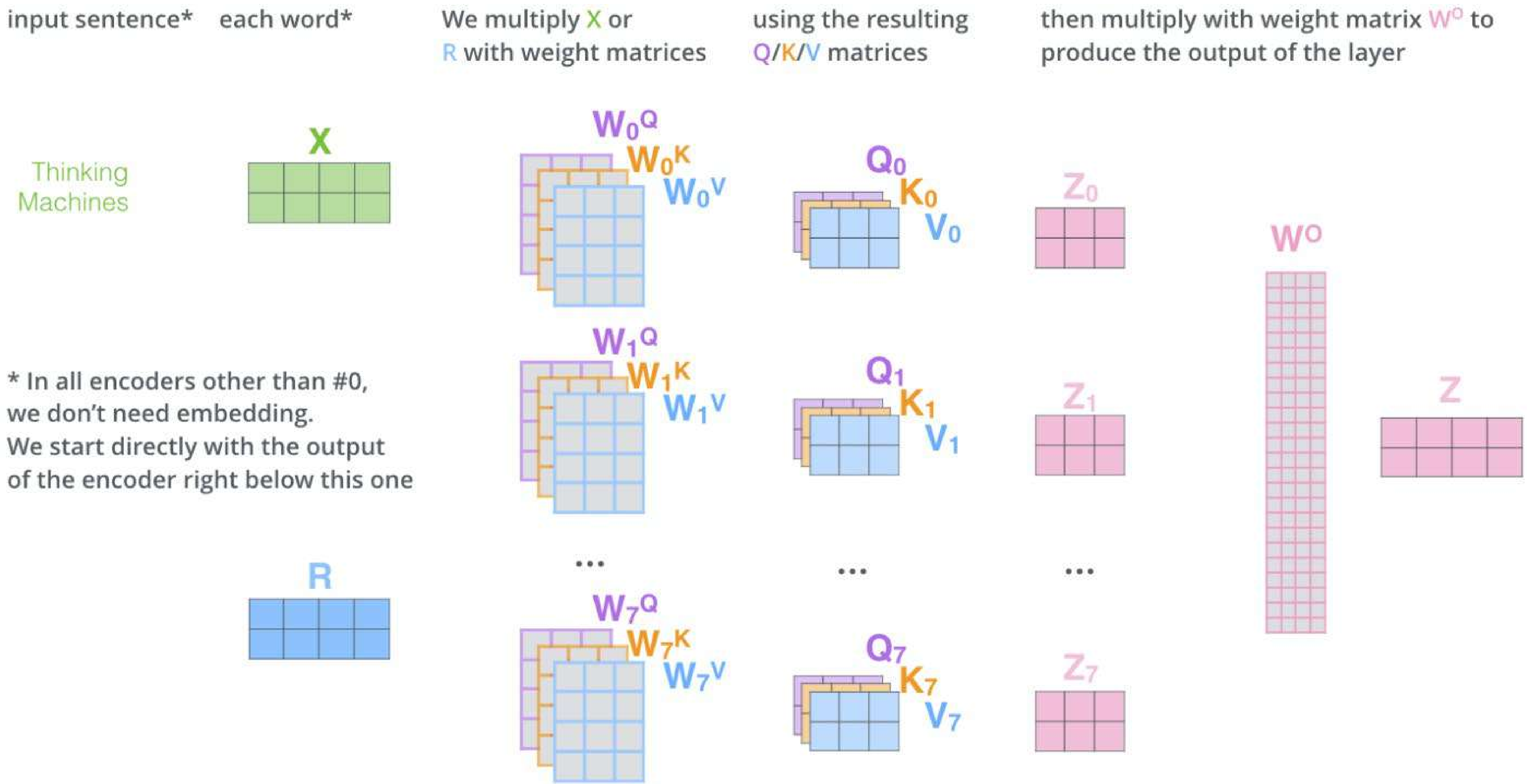


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

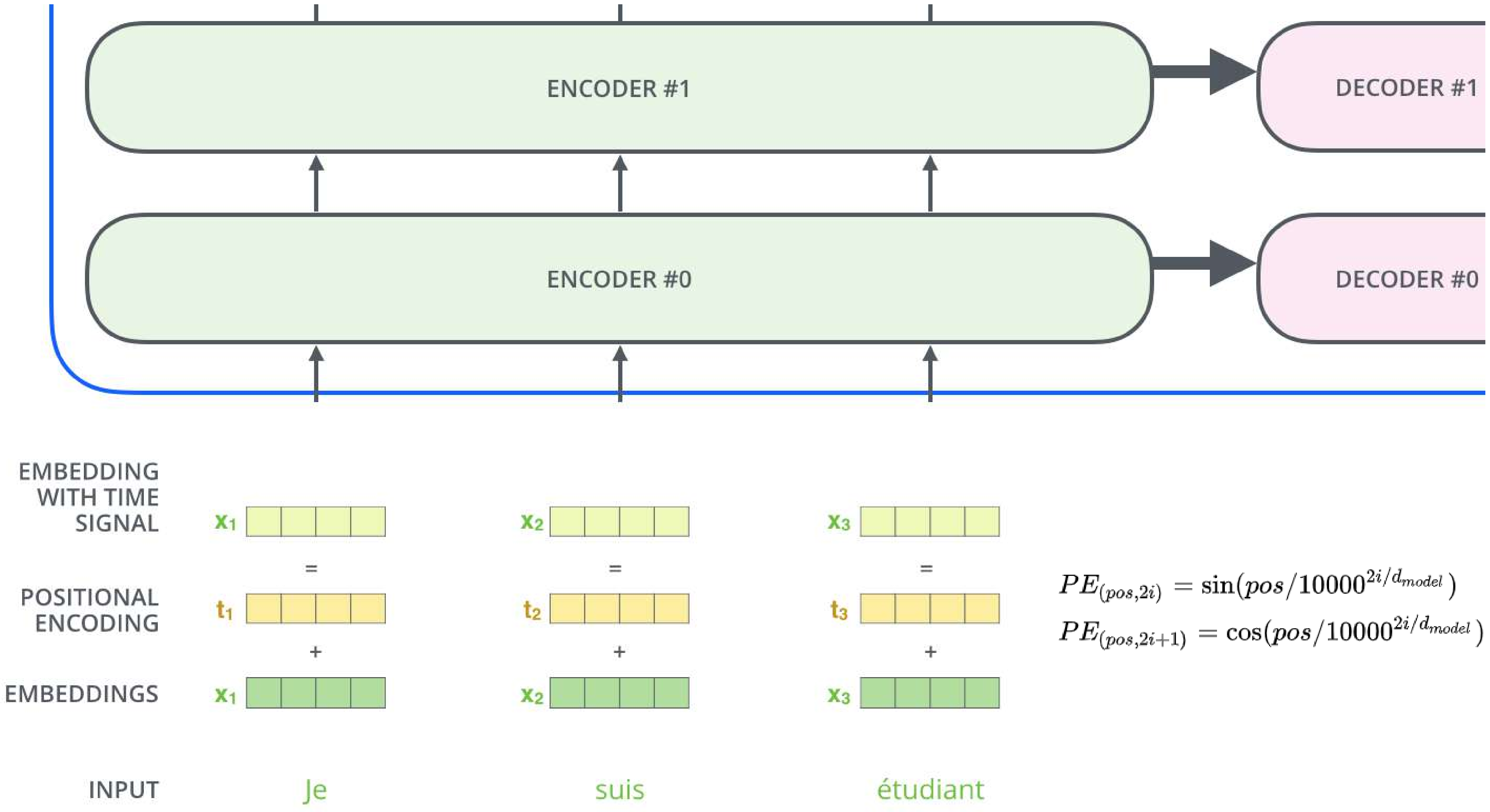
=



Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]

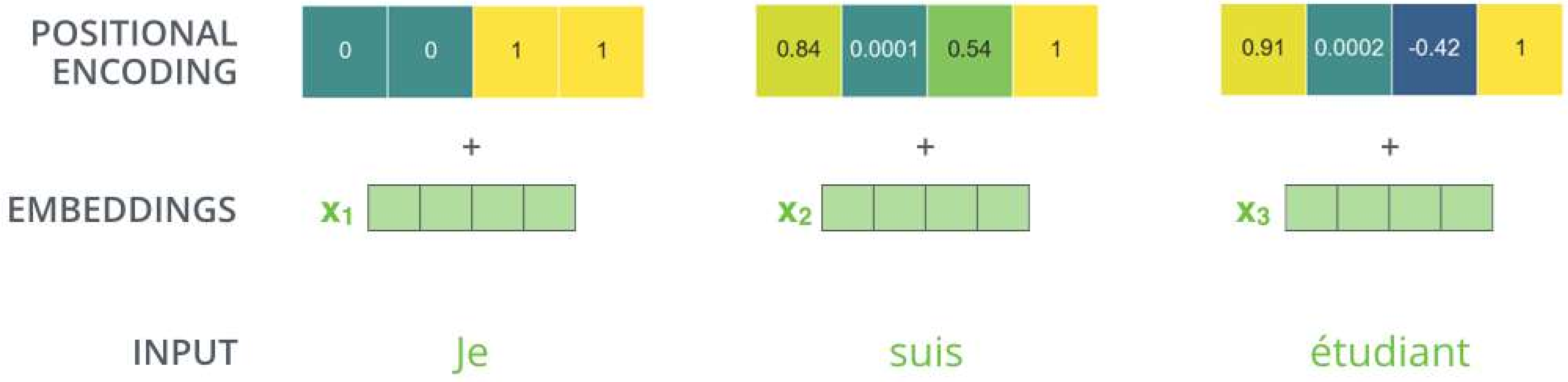


Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]

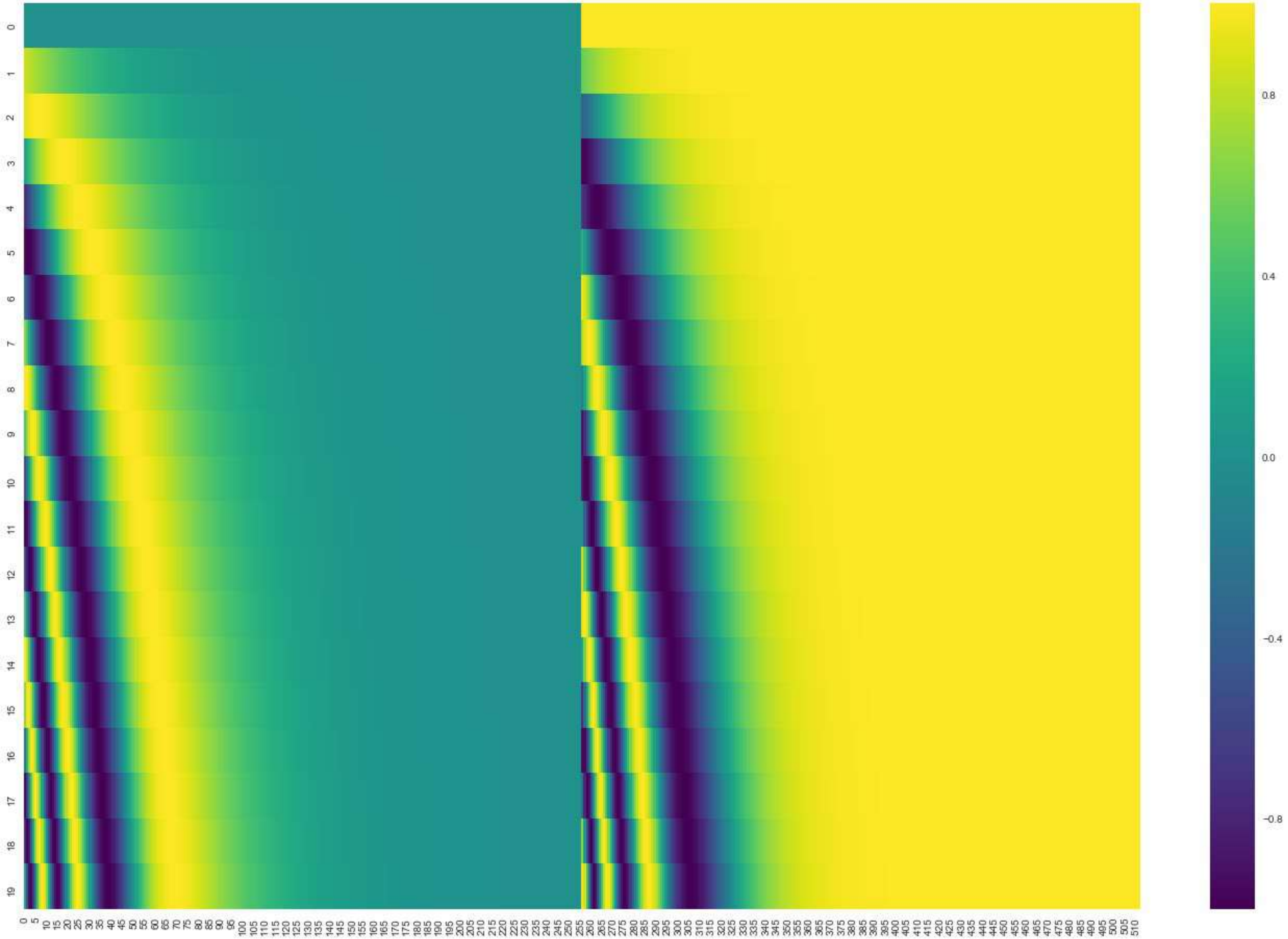


$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

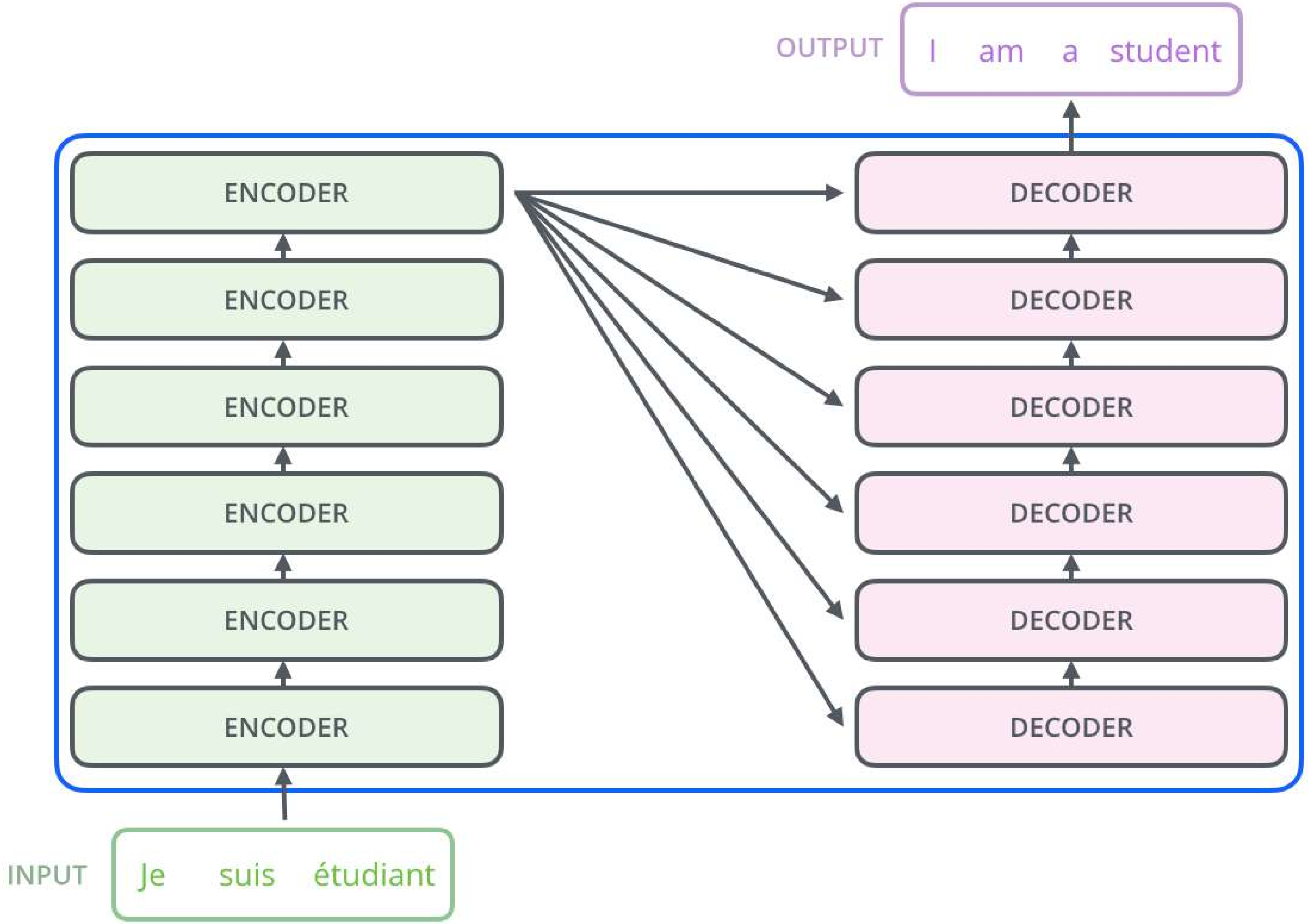
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



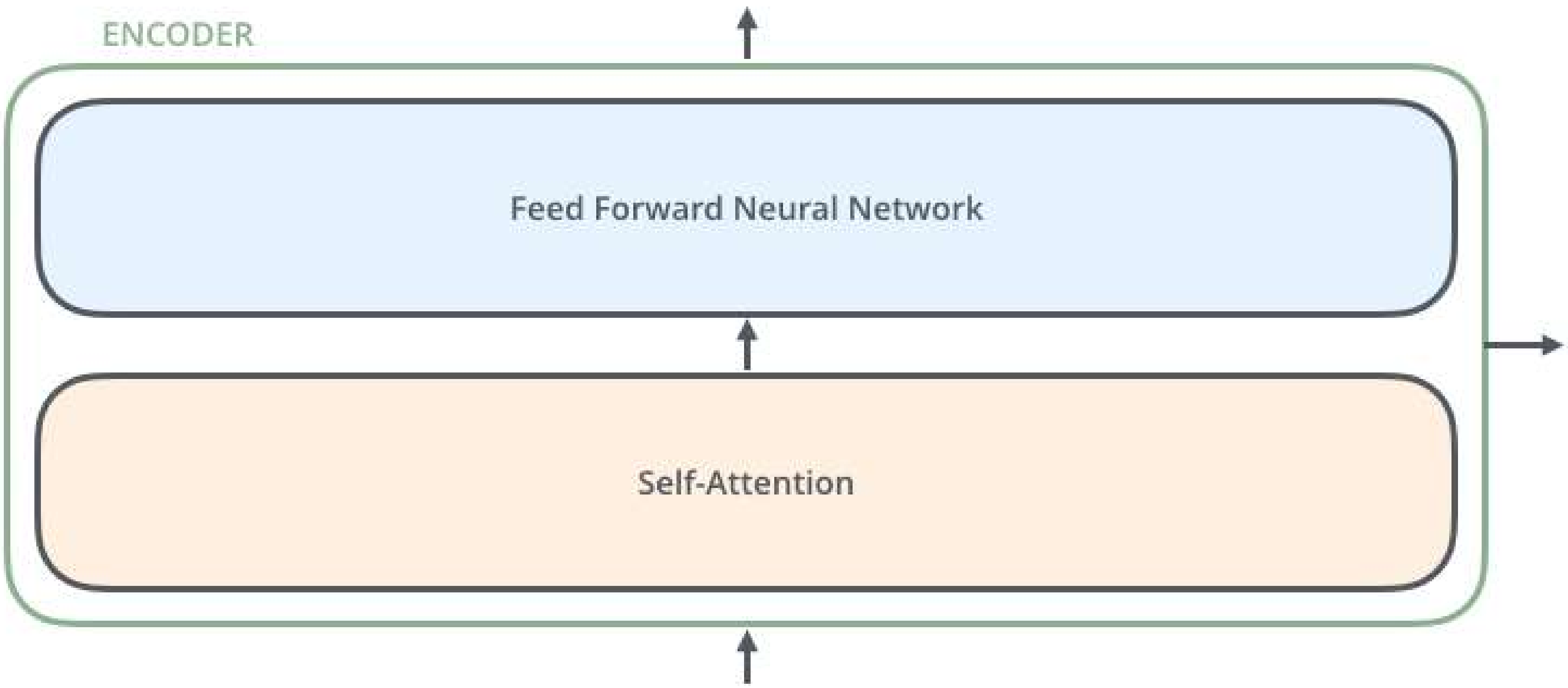
Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]



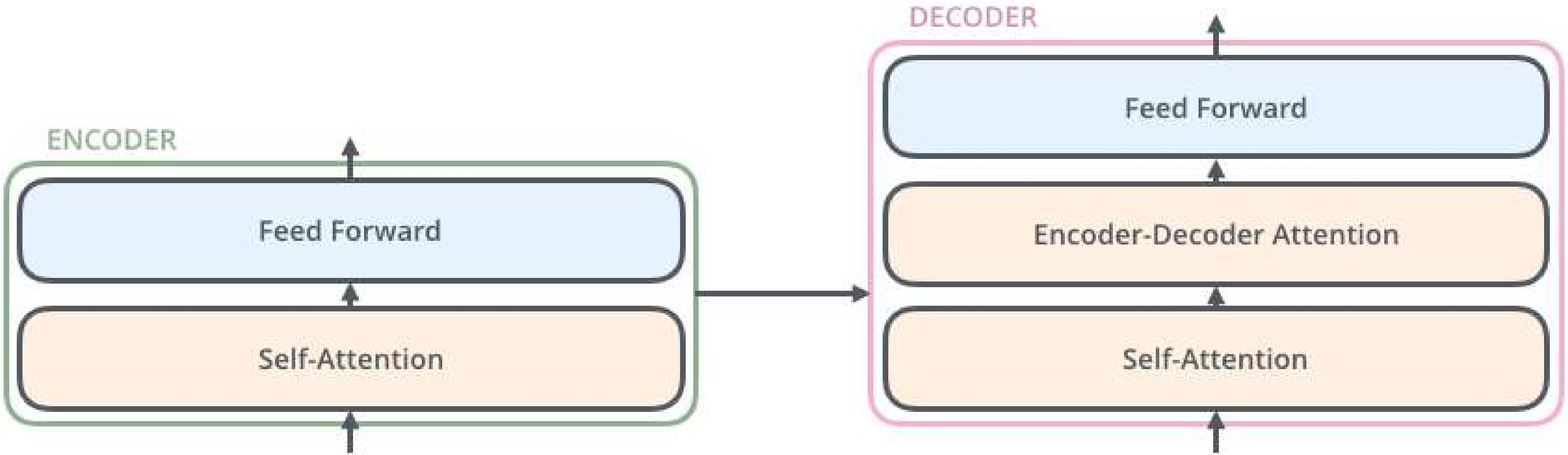
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



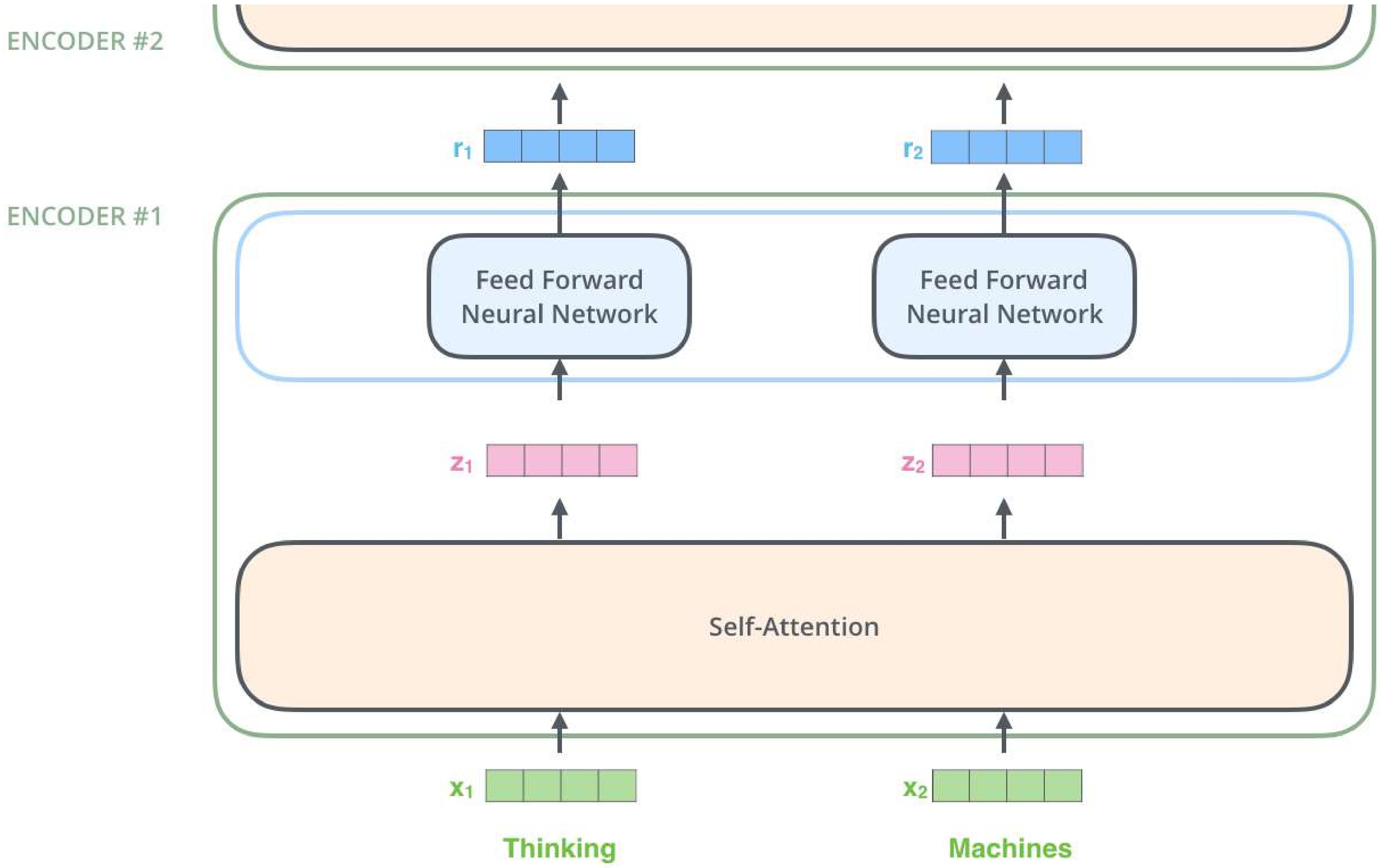
Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]



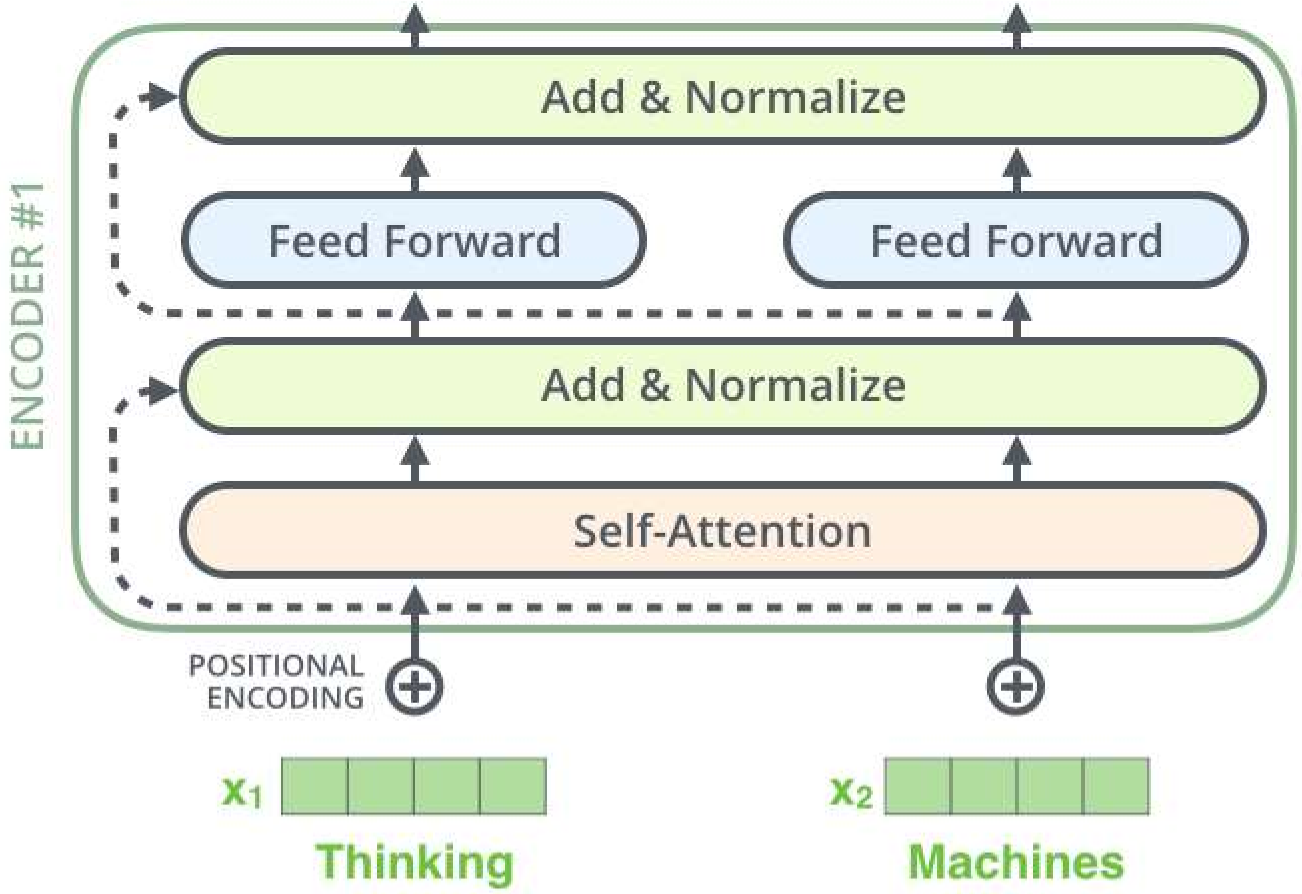
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



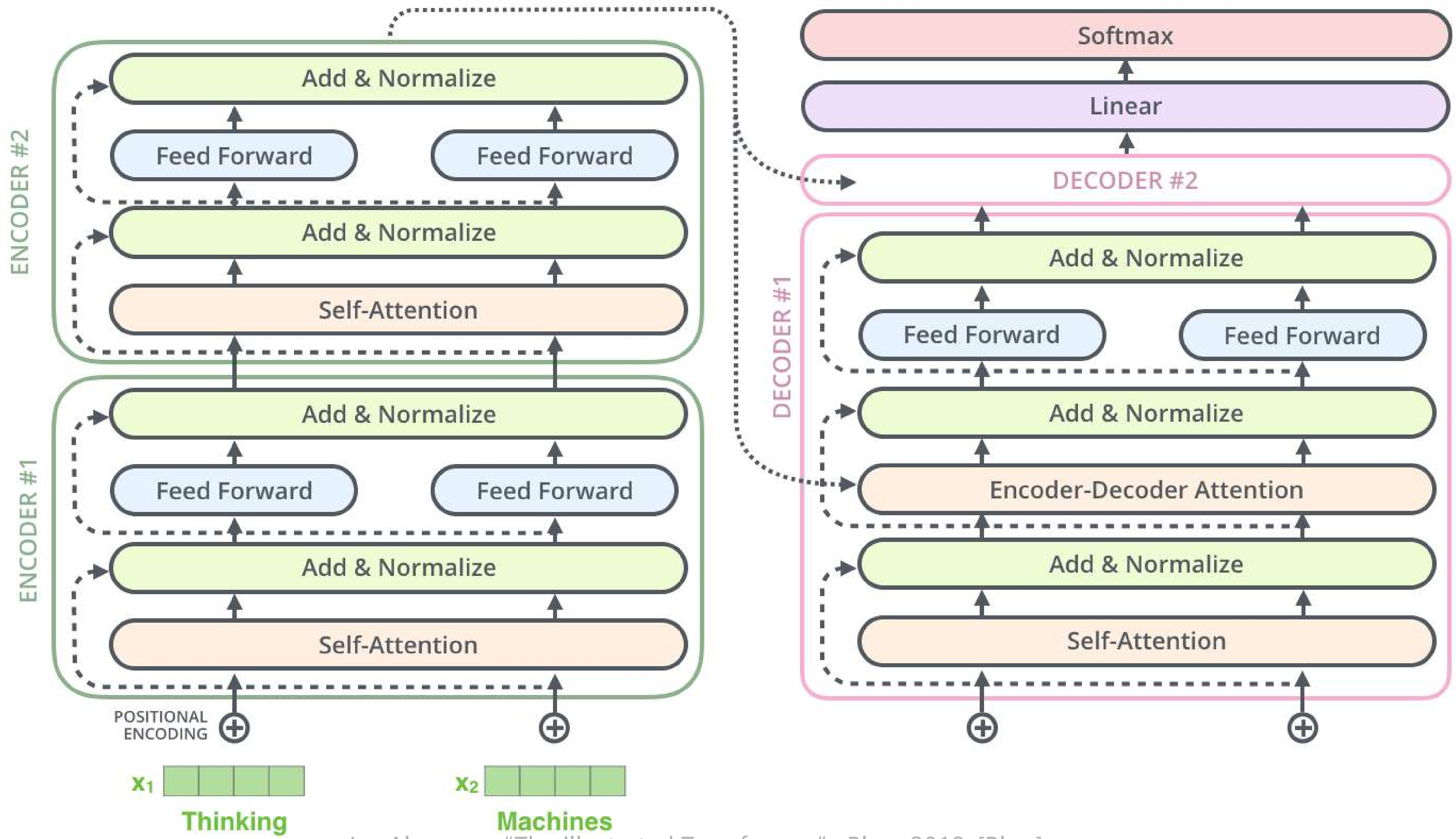
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



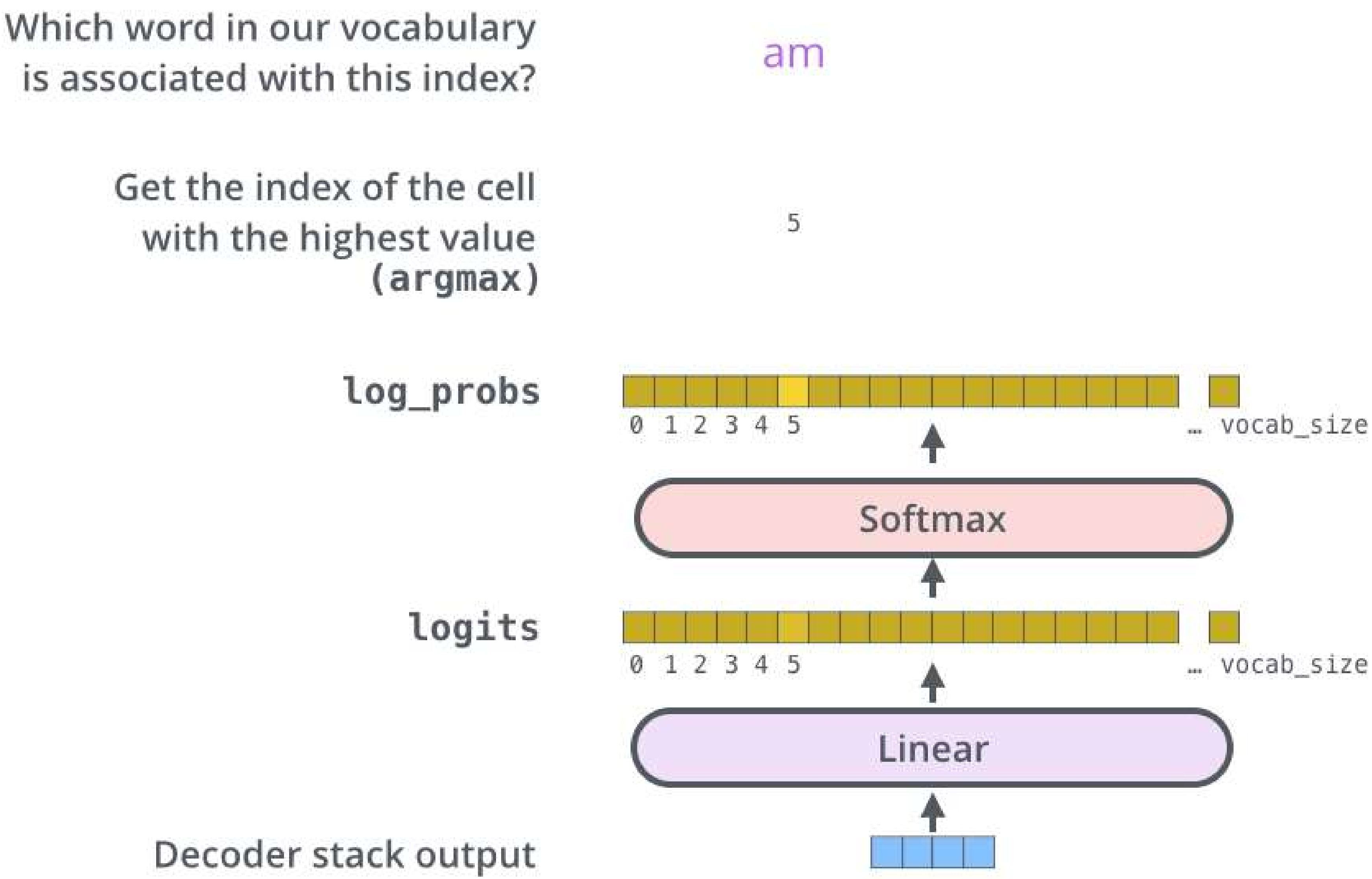
Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



Jay Alammar. “The Illustrated Transformer” Blog. 2018. [Blog]



Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]



Jay Alammar. "The Illustrated Transformer" Blog. 2018. [Blog]