CSCE 638 Natural Language Processing Foundation and Techniques

Lecture 7: Transformers

Kuan-Hao Huang

Spring 2025

(Some slides adapted from Chris Manning, Karthik Narasimhan, Danqi Chen, and Vivian Chen)

Project Sign-Up

- https://docs.google.com/spreadsheets/d/15Rj4AovtHtlZxILbX1ydrw7lEylamXuV7Dtg7cBD2EU/edit?usp=sharing
- 3~4 members per team
  - Form teams on your own
  - No solo teams (We have too many students!)

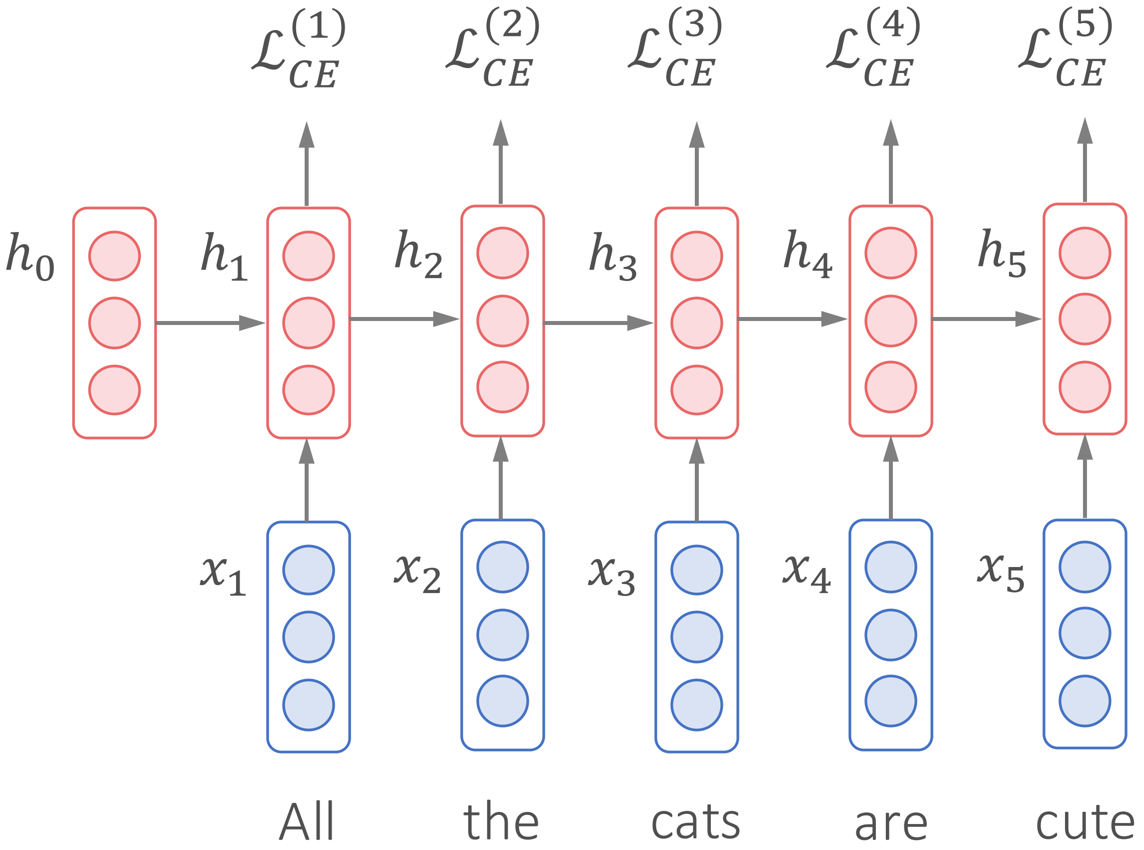
3~4 members per team									
	Project Topic	Member 1 (Name)	Member 1 (E-mail)	Member 2 (Name)	Member 2 (E-mail)	Member 3 (Name)	Member 3 (E-mail)	Member 4 (Name)	Member 4 (E-mail)
Team 1									
Team 2									
Team 3									
Team 4									
Team 5									
Team 6									
Team 7									
Team 8									
Team 9									
Team 10									
Team 11									
Team 12									

Lecture Plan

- Transformers
  - Attention
  - Self-Attention
  - Transformer Encoder
  - Positional Encoding

Recap: RNN as Encoder

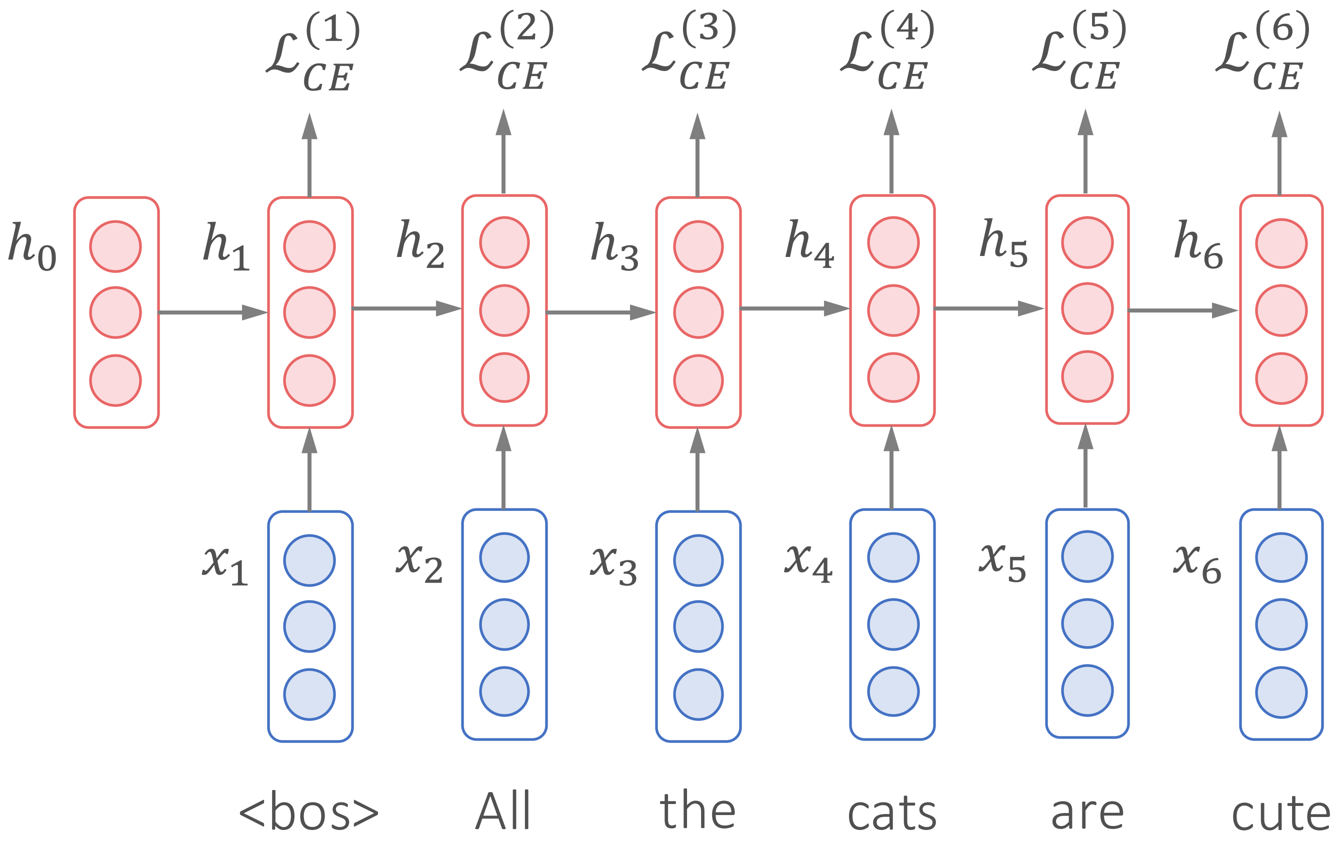
- Sequential labeling: A sequence of dependent classification



$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}^{(i)}$$

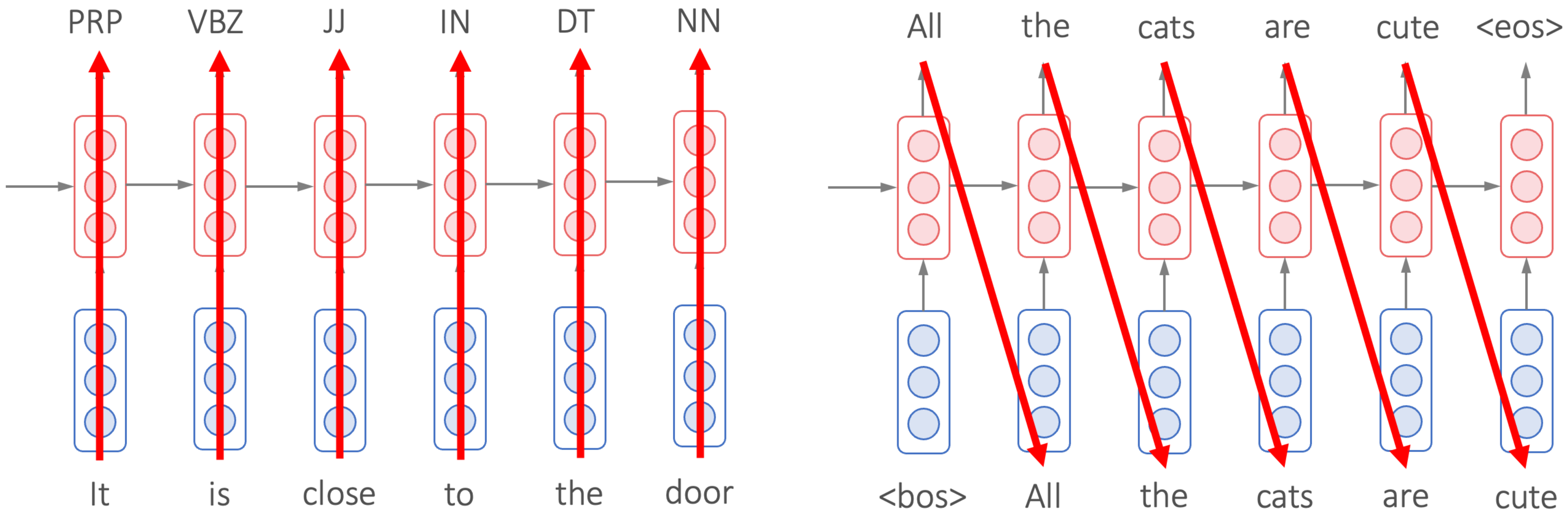
Recap: RNN as Decoder

- RNN Language Modeling
  - Generation is a sequence of word classification



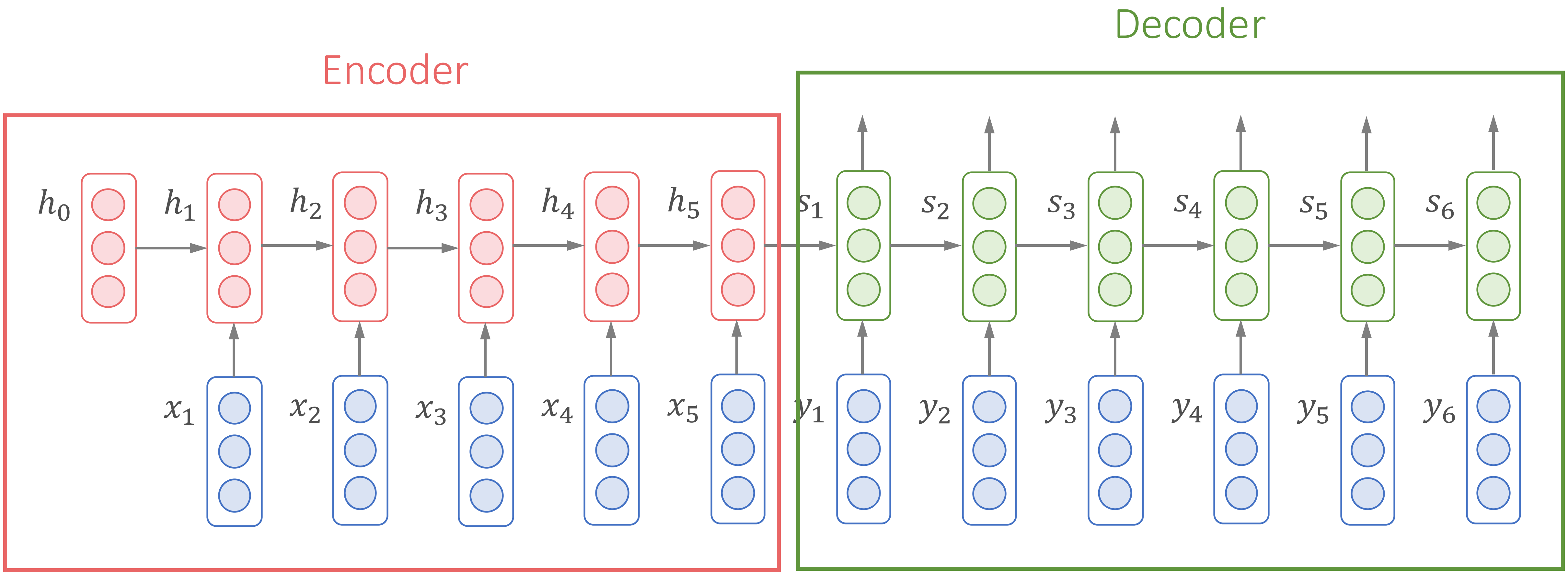
$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}^{(i)}$$

Recap: Encoder vs. Decoder



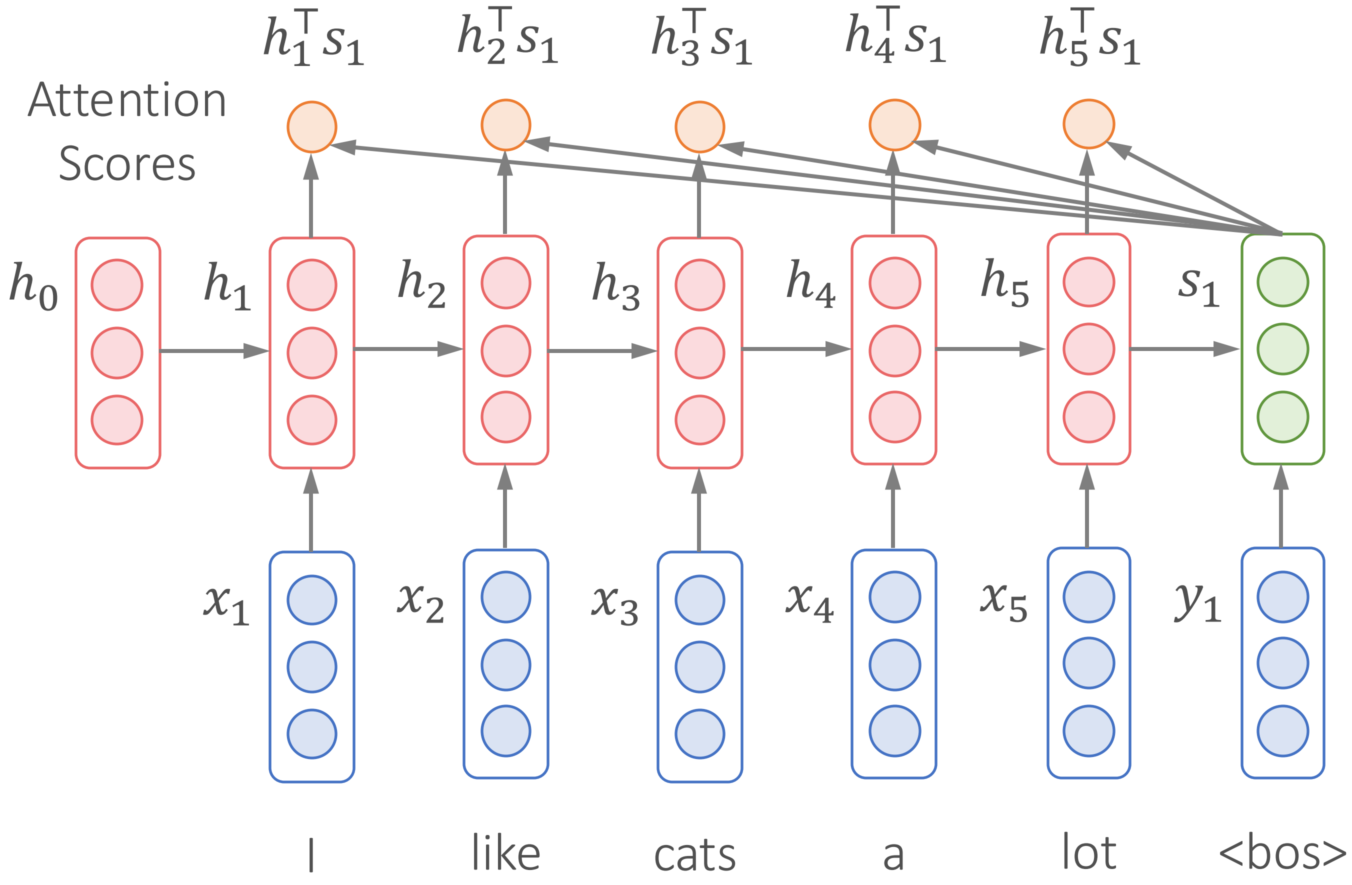
Recap: Sequence-to-Sequence Models (Seq2Seq)

- When we need understanding and generation at the same time

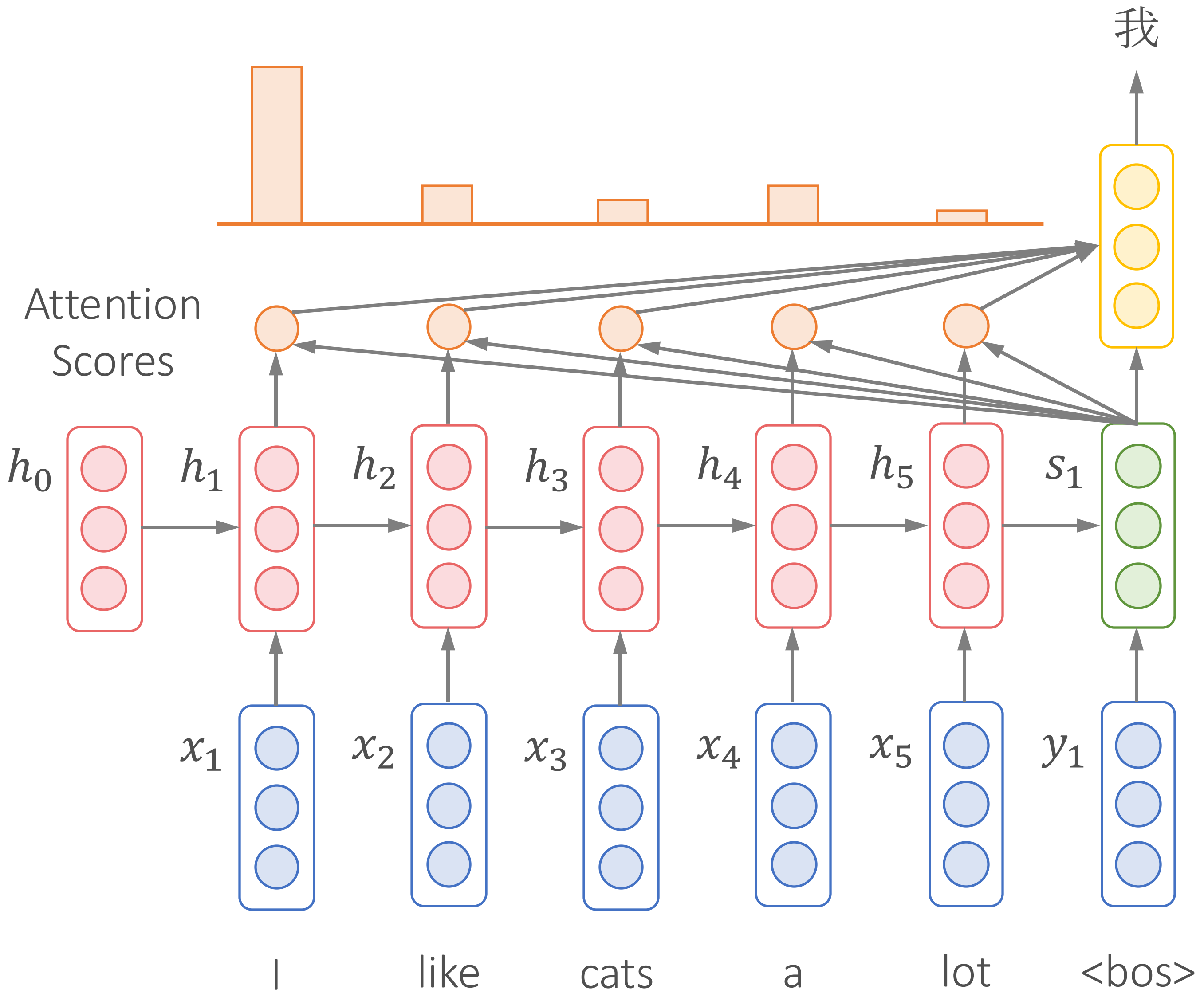


RNN with Attention

Attention Scores $\alpha_i = h_i^\top s_1$



RNN with Attention



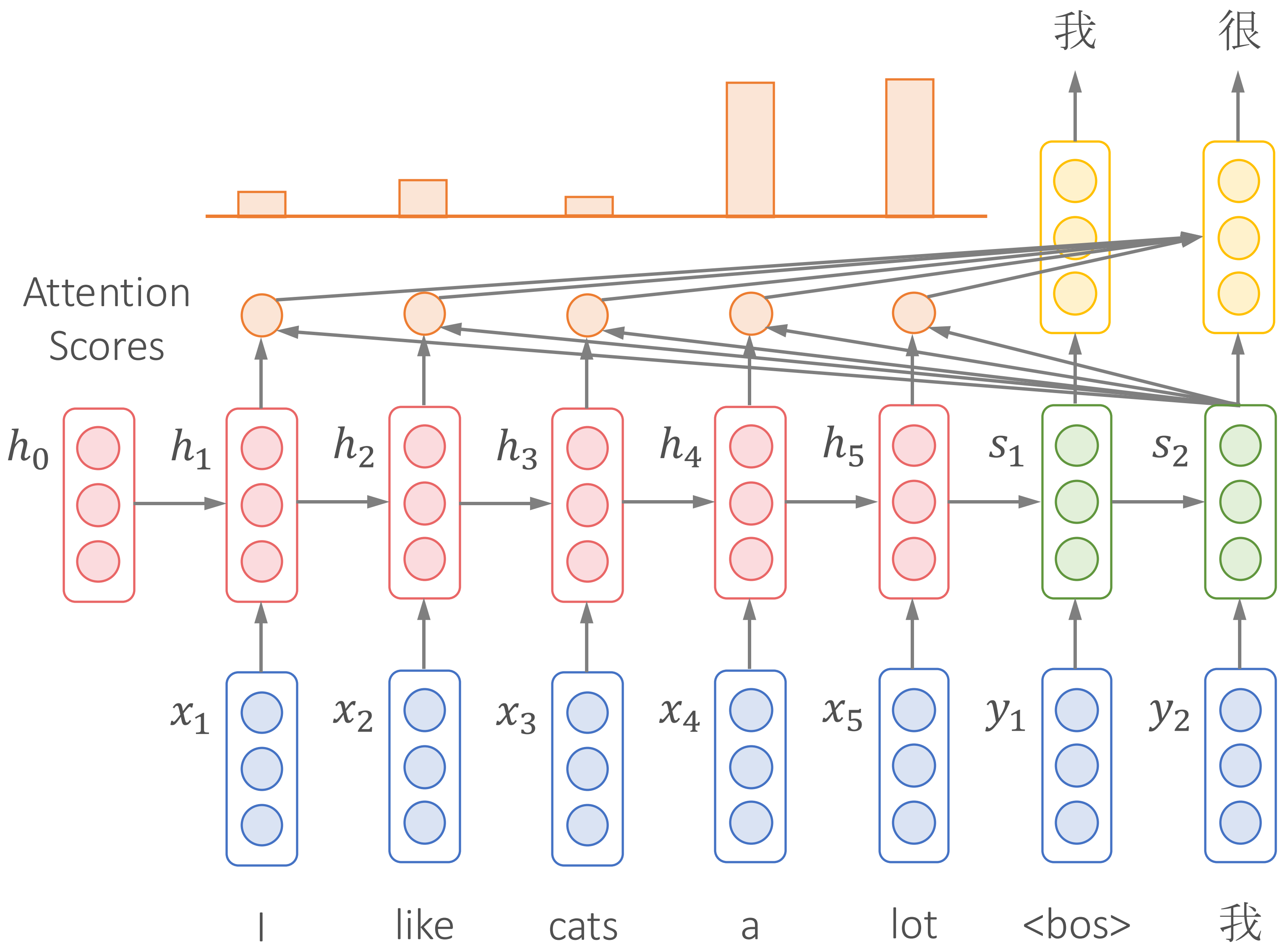
Attention Scores $\alpha_i = h_i^\top s_1$

Normalized Attention Scores $\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$

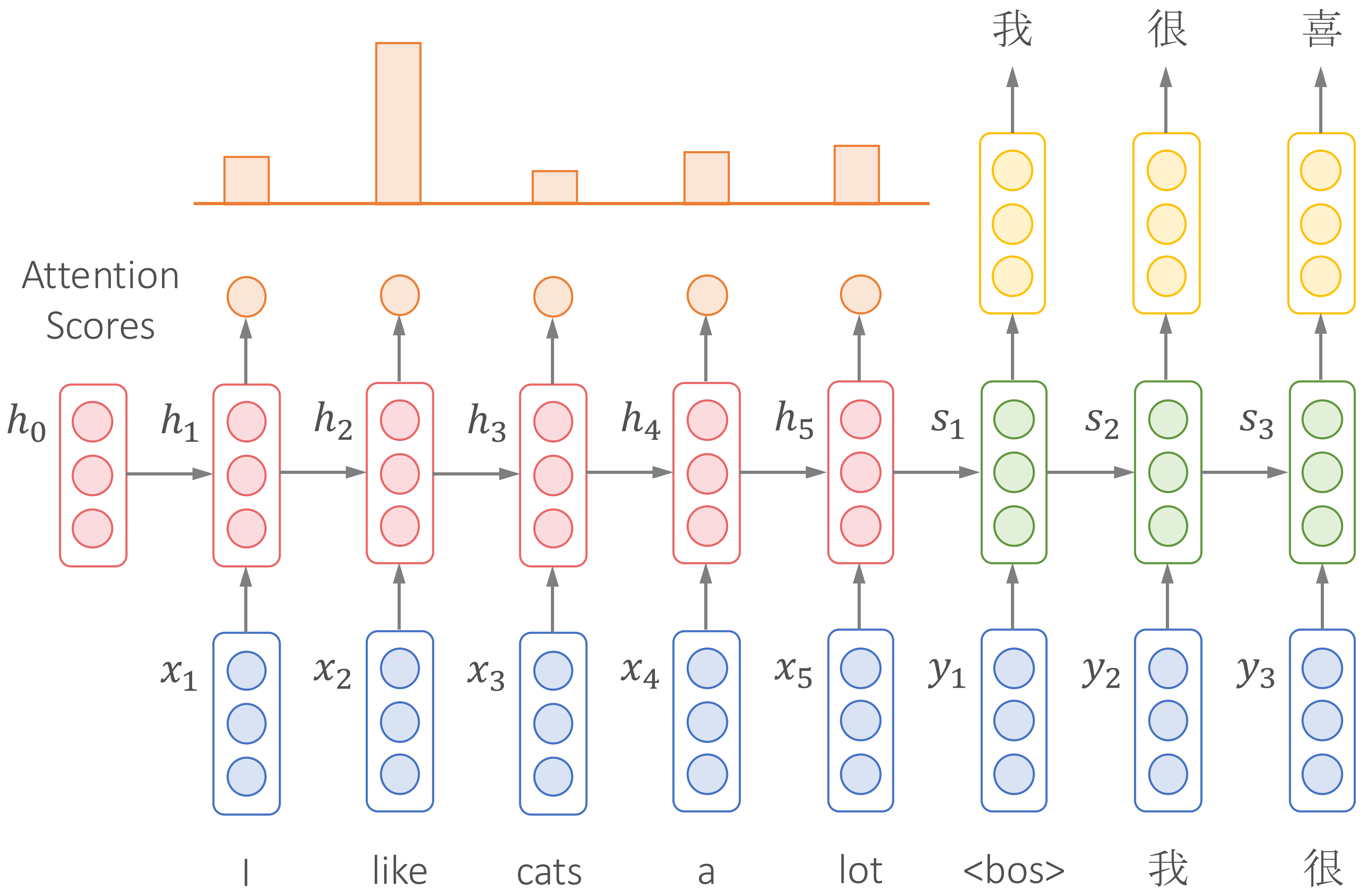
Weighted Sum $a = \sum_i \hat{\alpha}_i h_i$

Attention Output $\tanh(\mathbf{W}[a; s_1])$

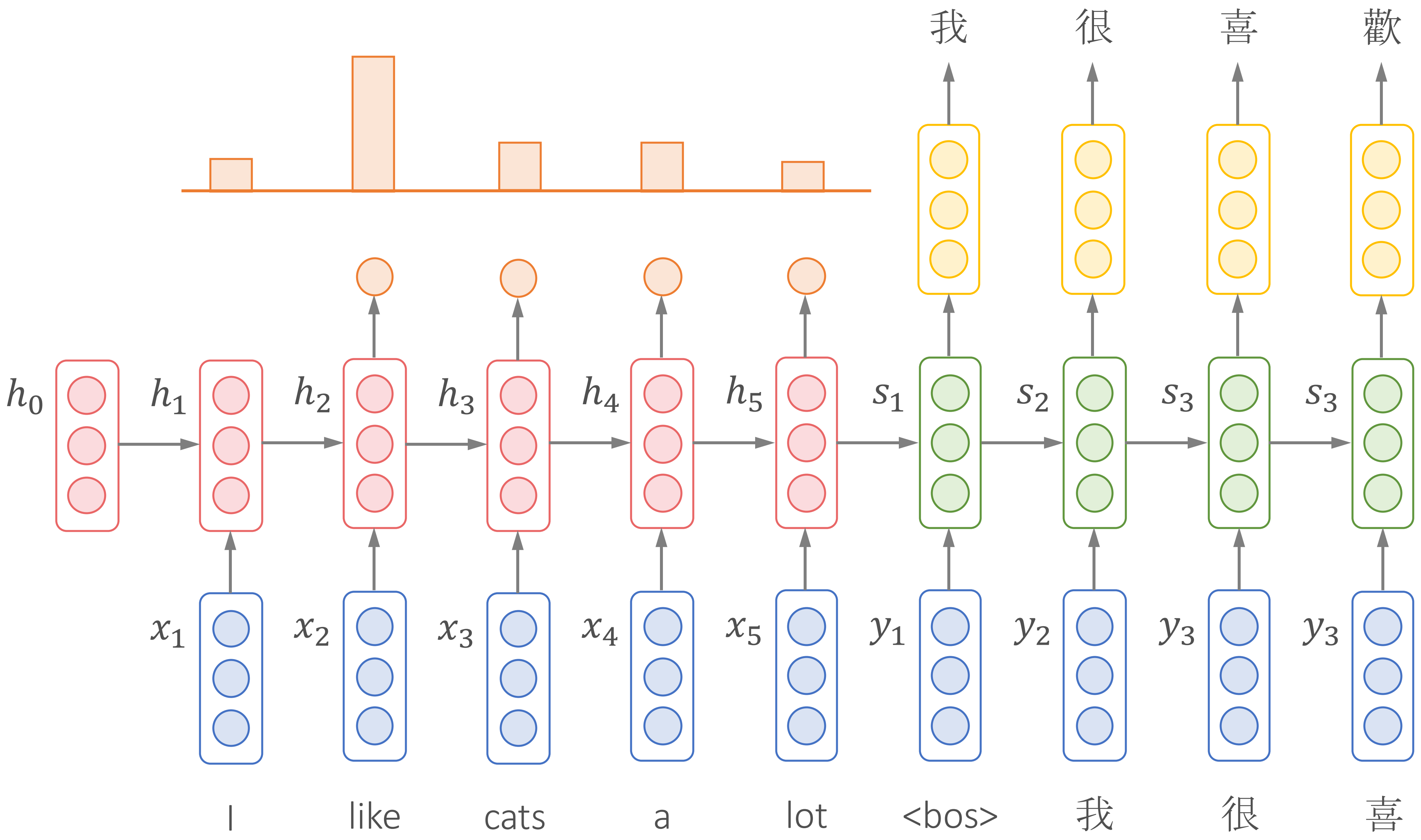
RNN with Attention



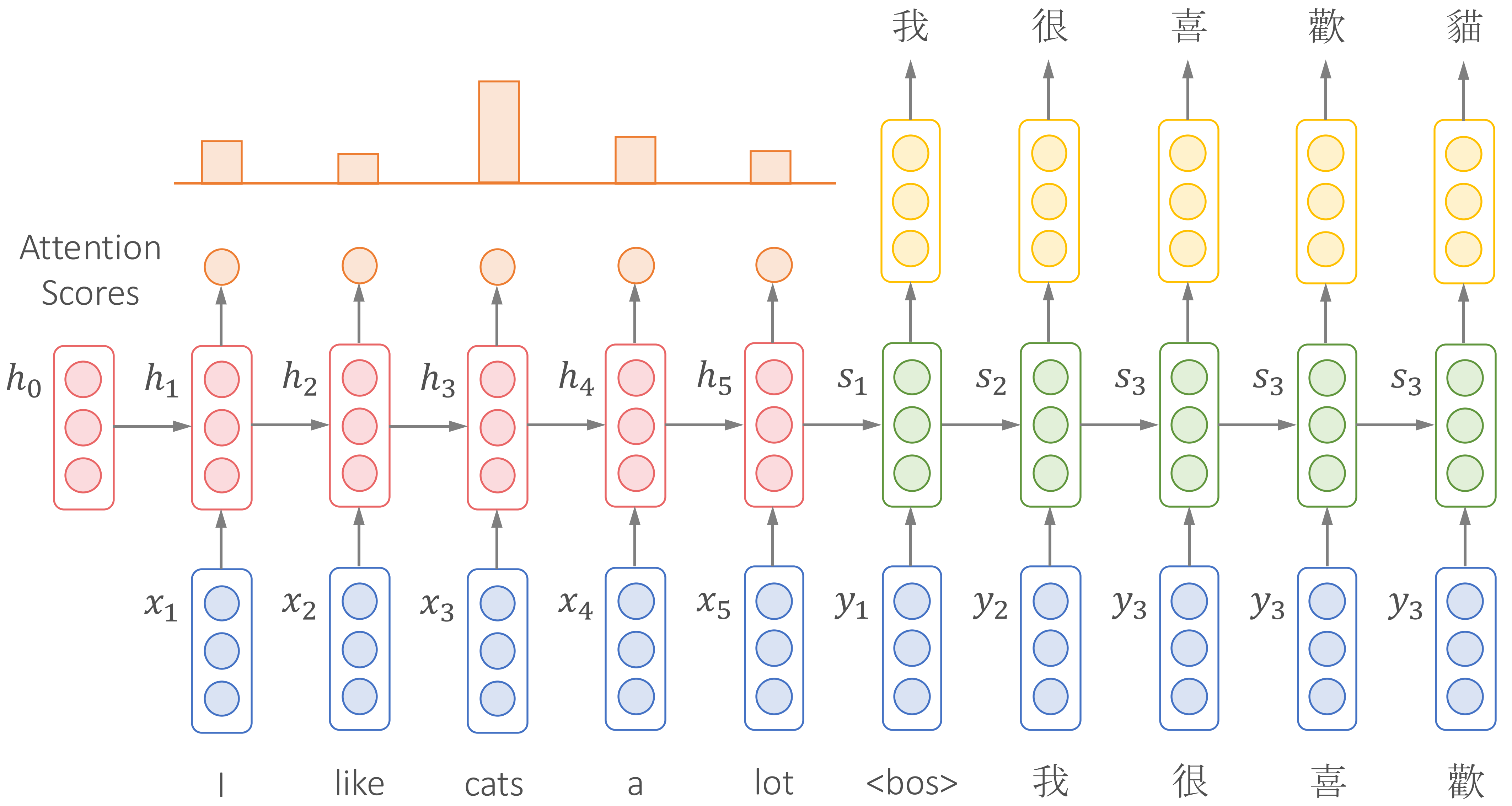
RNN with Attention



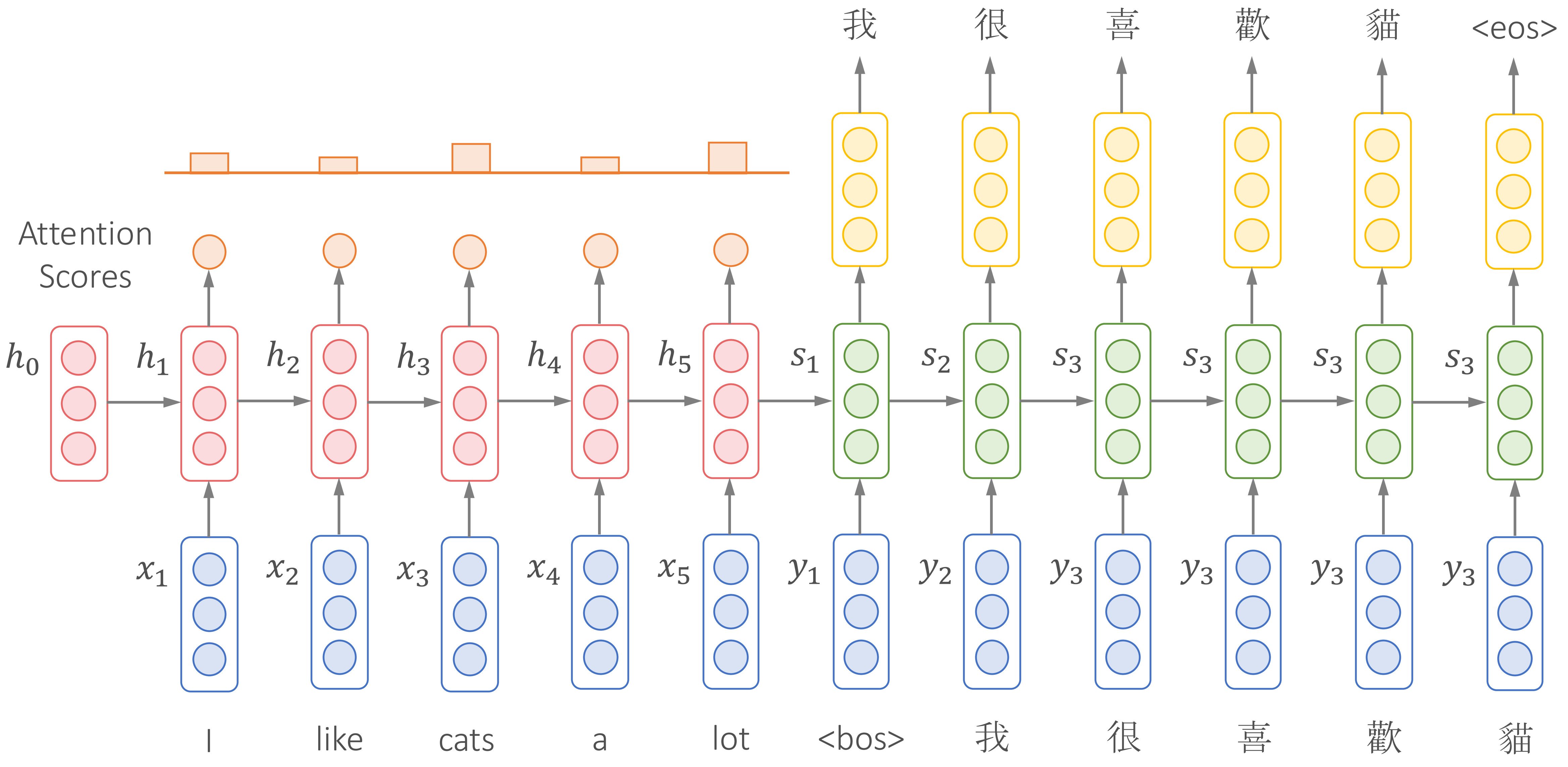
RNN with Attention



RNN with Attention



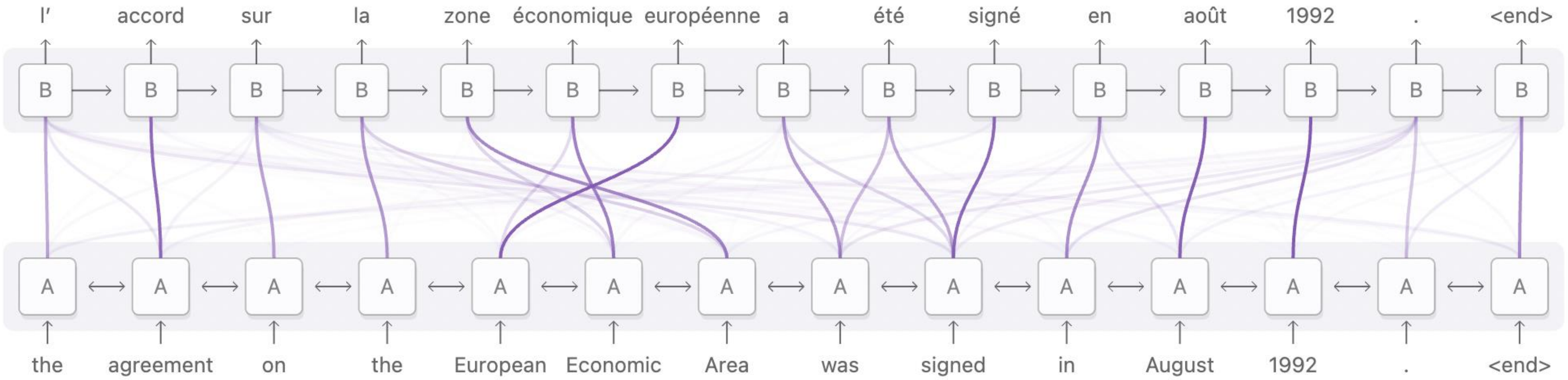
RNN with Attention



Different Types of Attention

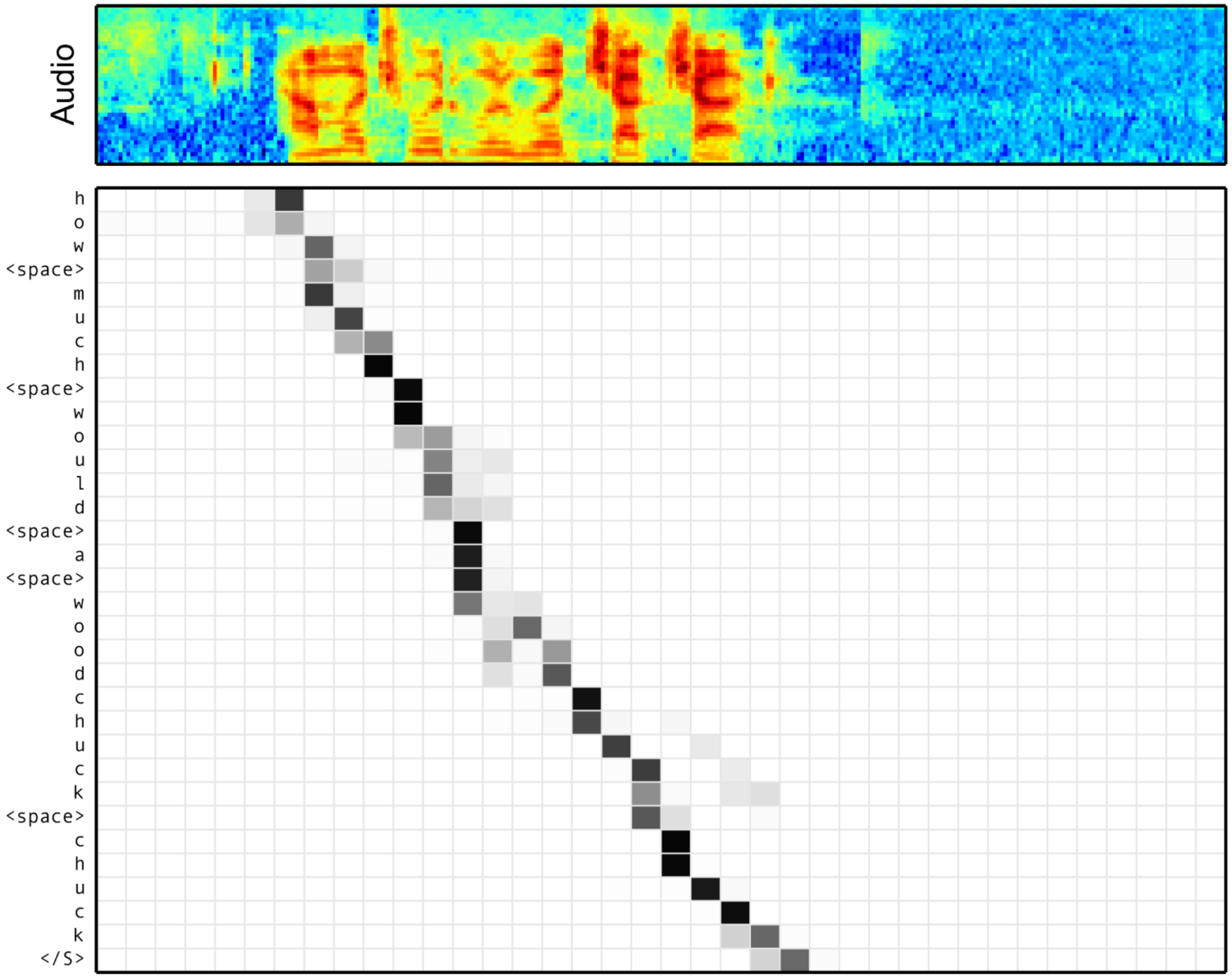
Dot-Product Attention	$h_i^\top s_j$
Multiplicative Attention	$h_i^\top W s_j$
Additive Attention	$v^\top \tanh\left(W_1 h_i + W_2 s_j\right)$

Machine Translation with Attention



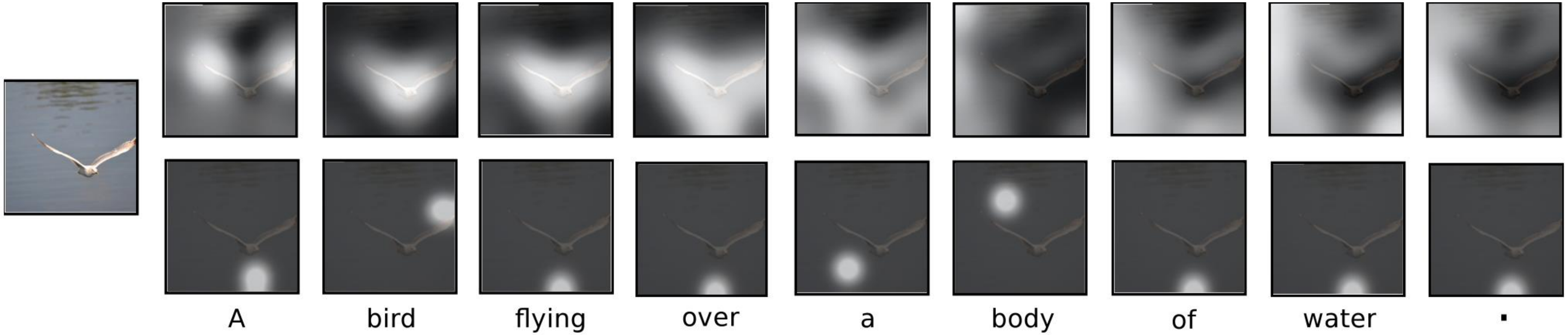
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Speech Recognition with Attention



Listen, Attend and Spell, 2015

Image Captioning with Attention



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

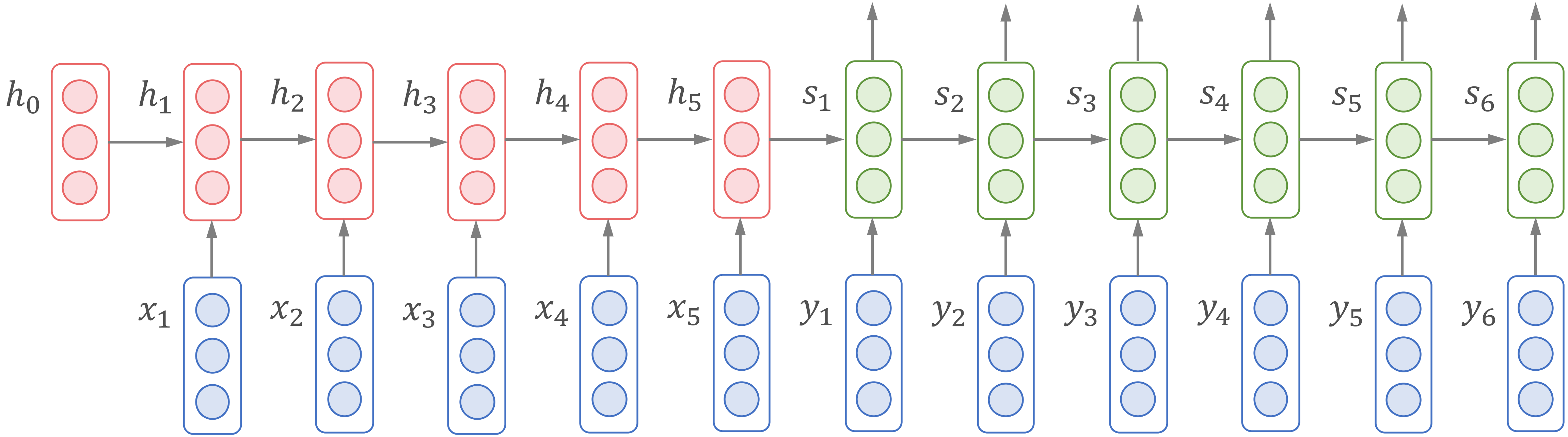
A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, 2015

Issues with RNN

- Longer sequences can lead to vanishing gradients → It is hard to capture long-distance information
- Lack parallelizability



Transformers: Attention Is All You Need!

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

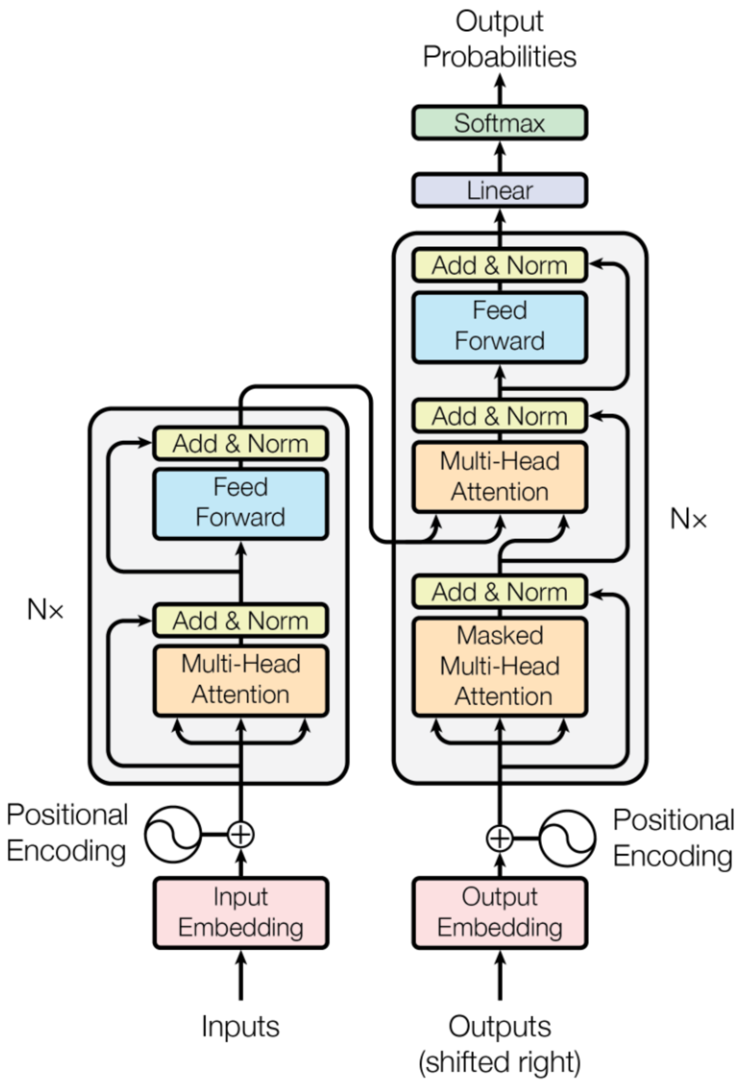
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

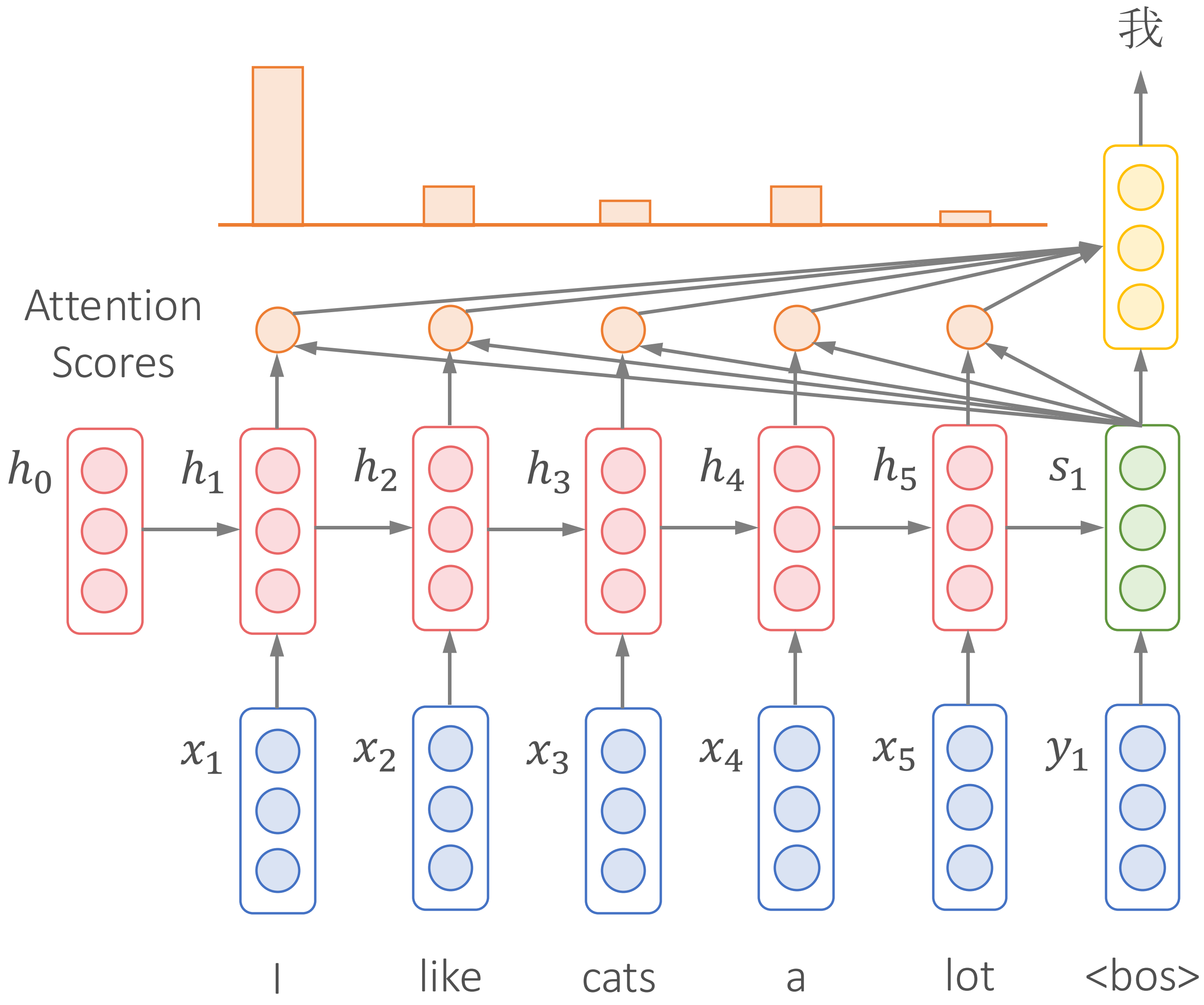
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com



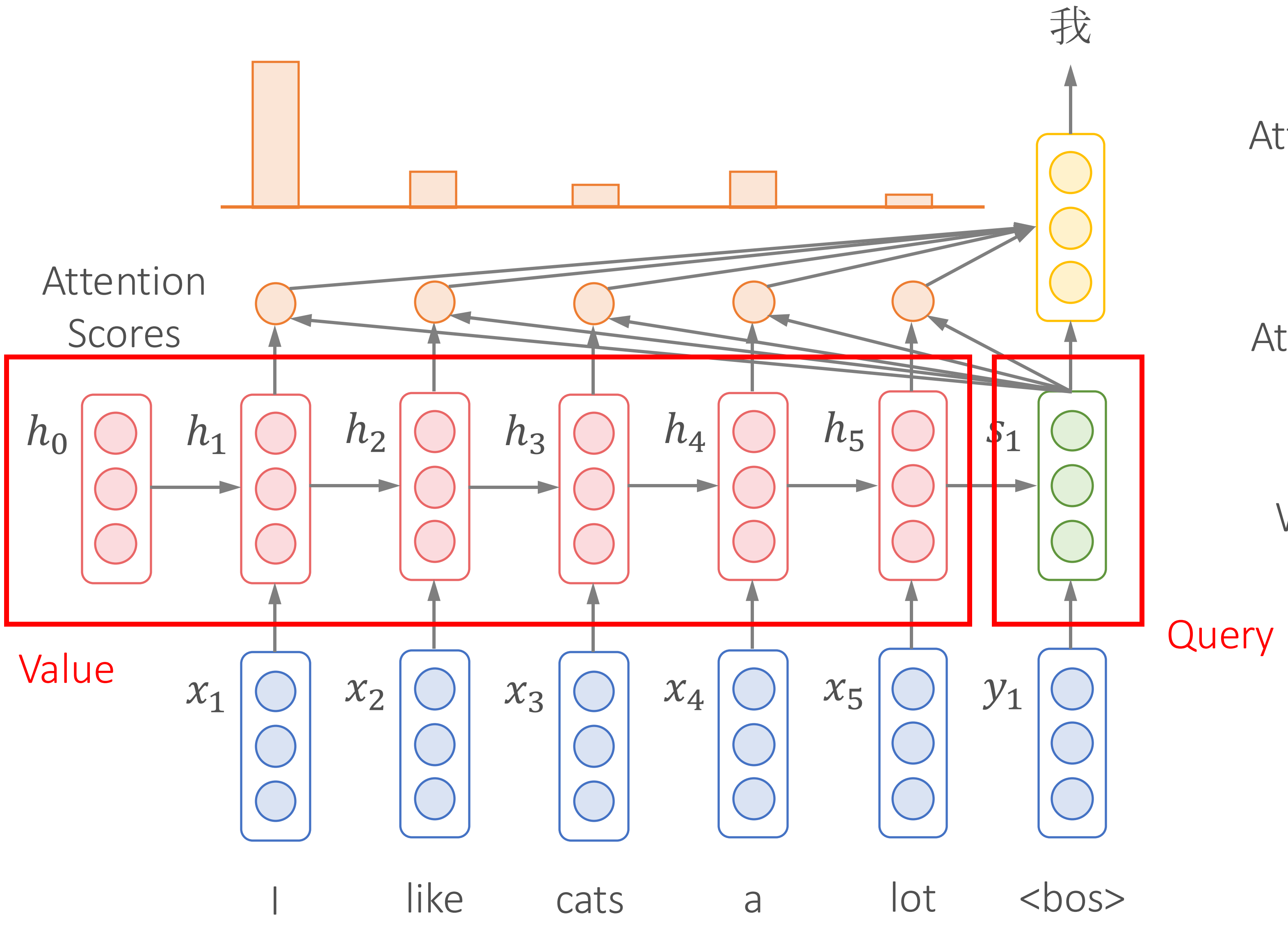
Attention Is All You Need, 2017

Look Back at RNN with Attention



Attention Scores	$\alpha_i = h_i^\top s_1$
Normalized Attention Scores	$\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$
Weighted Sum	$a = \sum_i \hat{\alpha}_i h_i$
Attention Output	$\tanh(\mathbf{W}[a; s_1])$

Look Back at RNN with Attention



Inner Product of Query and Values

Attention Scores $\boxed{\alpha_i = h_i^{\mathsf{T}} s_1}$

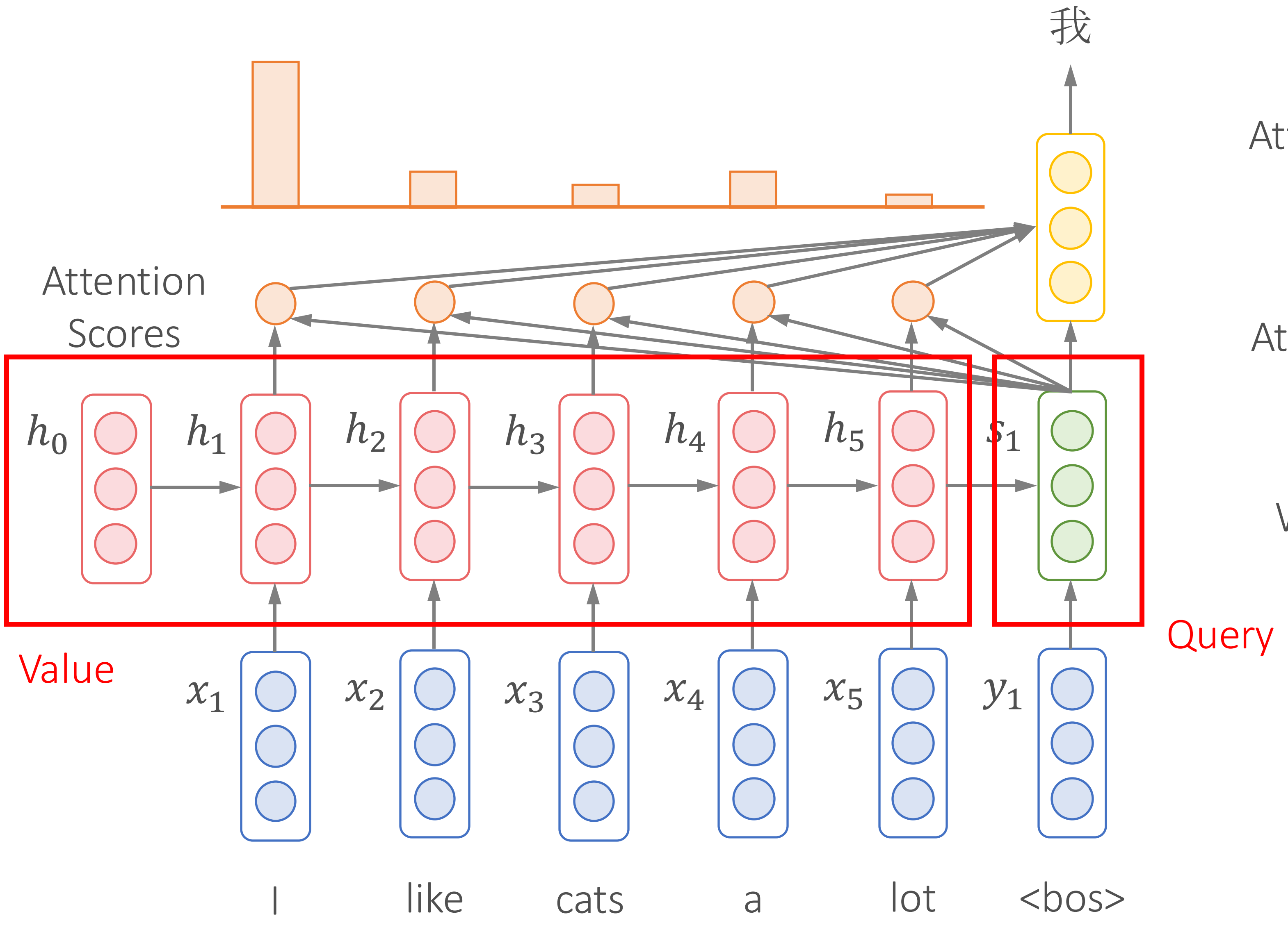
Normalized Attention Scores $\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$

Weighted Sum $\boxed{a = \sum_i \hat{\alpha}_i h_i}$

Weighted Sum of Values

Attention Output $\tanh(\mathbf{W}[a; s_1])$

Look Back at RNN with Attention – General Version



Inner Product of Query and Keys

Attention Scores $\boxed{\alpha_i = h_i'^{\top} s_1}$

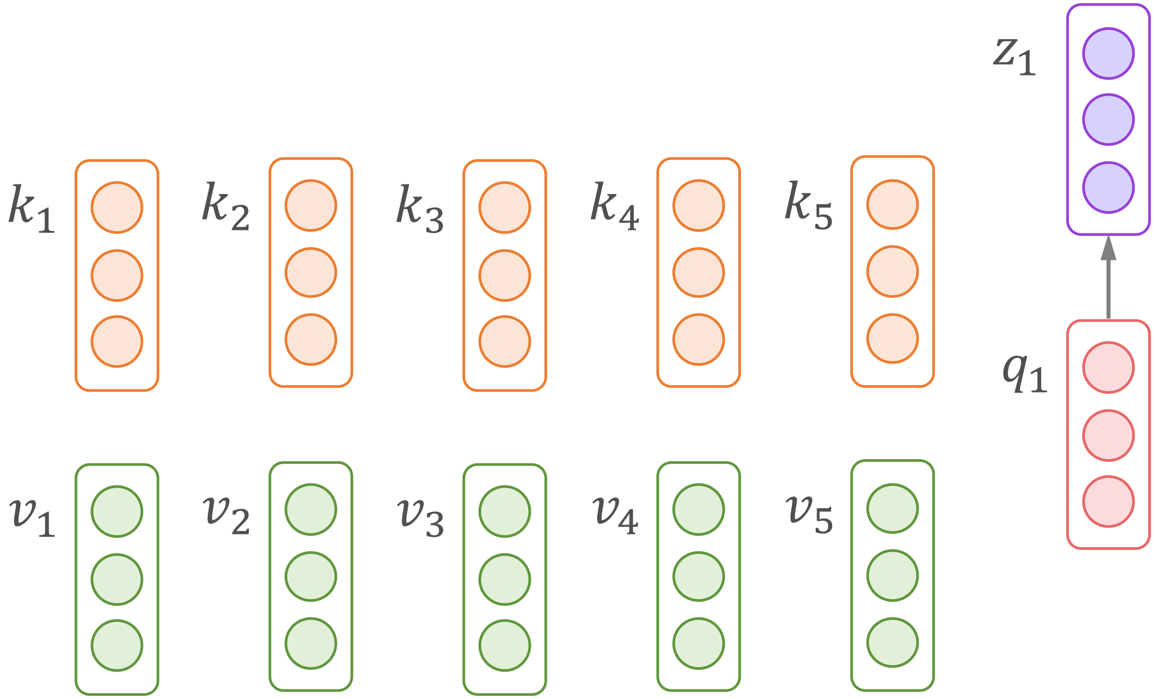
Normalized Attention Scores $\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$

Weighted Sum $\boxed{a = \sum_i \hat{\alpha}_i h_i}$

Weighted Sum of Values

Attention Output $\tanh(\mathbf{W}[a; s_1])$

Attention – General Version



Attention Scores $\alpha_i = k_i^\top q_1$

Normalized Attention Scores $\hat{\alpha}_i = \text{softmax}(\alpha_i)$

Weighted Sum $z_1 = \sum_i \hat{\alpha}_i v_i$

From Attention to Self-Attention

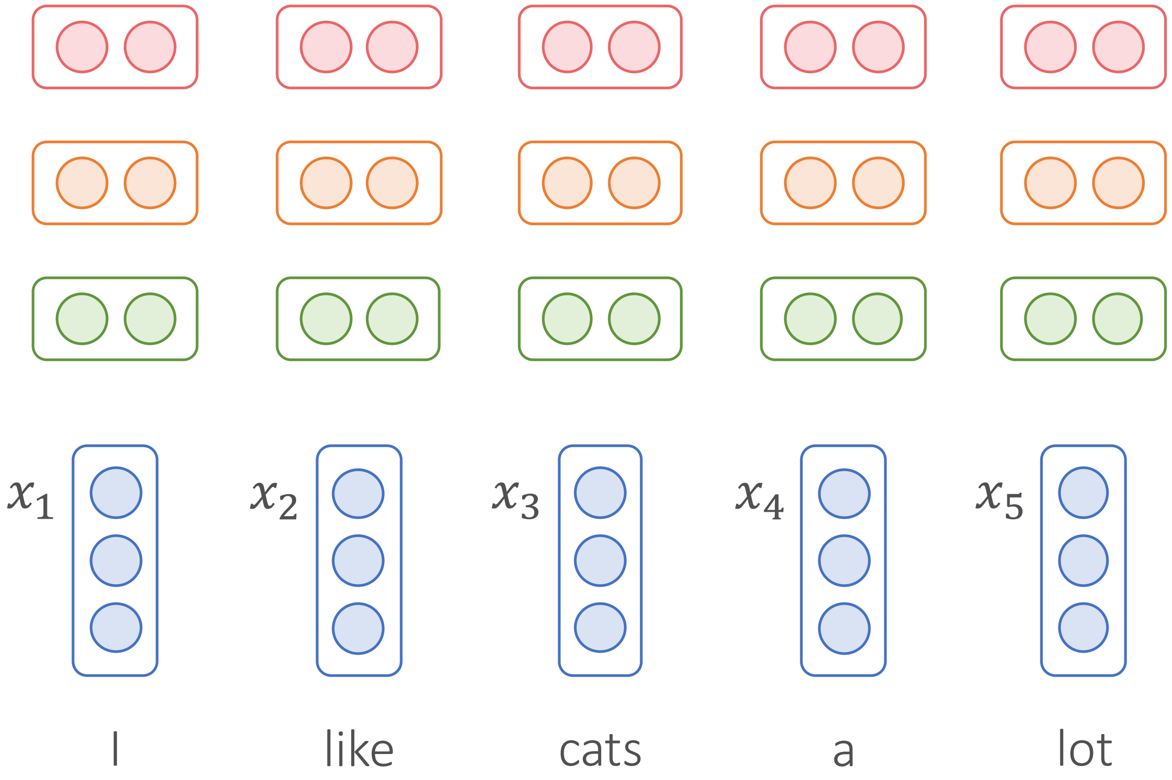
- Self-attention = attention from the sequence to itself
  - The queries, keys and values come from the same source
- Any word can be a query
- Any word can be a key
- Any word can be a value

Self-Attention

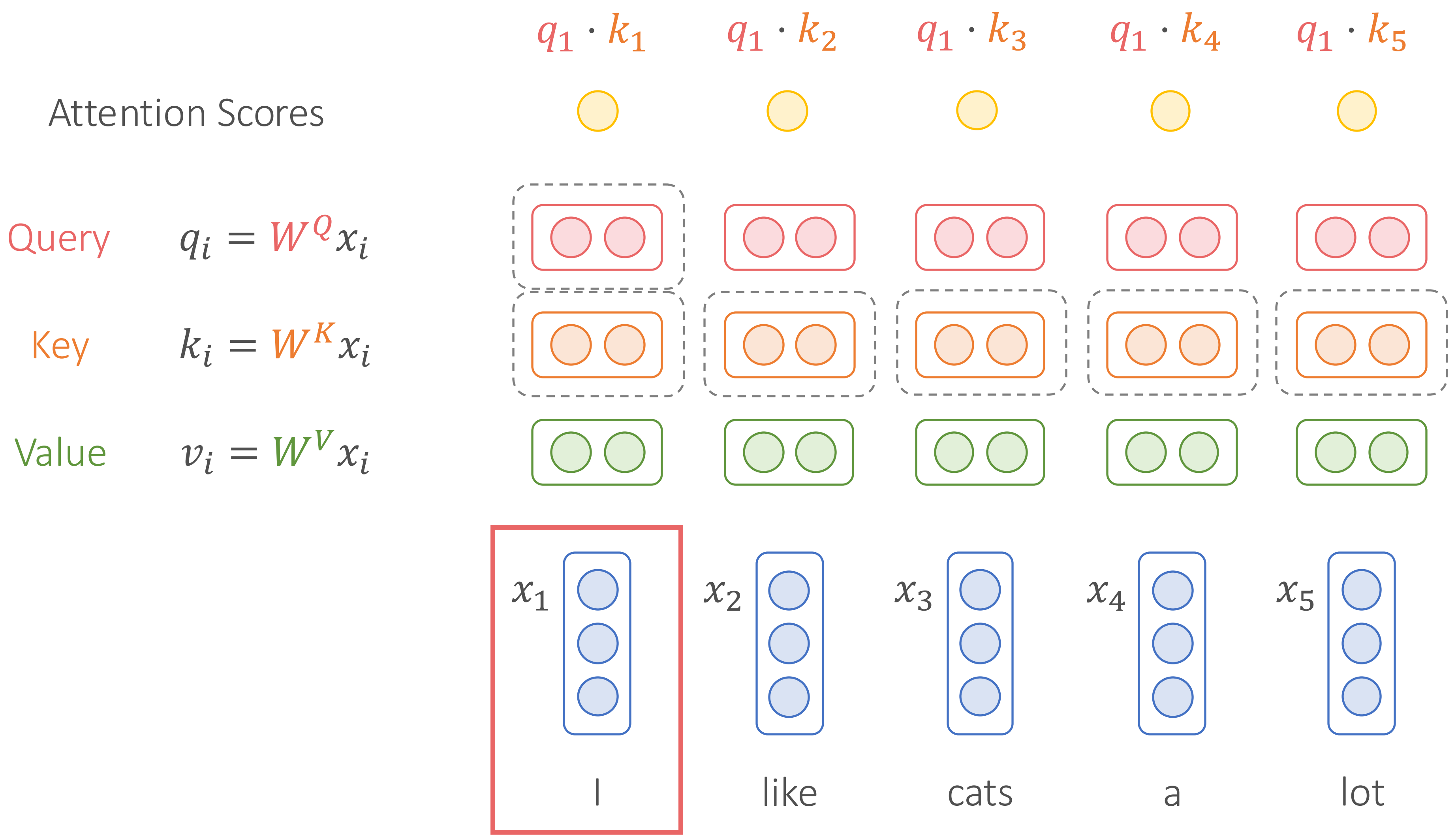
Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention



Self-Attention

$$\alpha_{1,i} = \text{softmax}\left(\frac{q_1 \cdot k_i}{\sqrt{d}}\right)$$

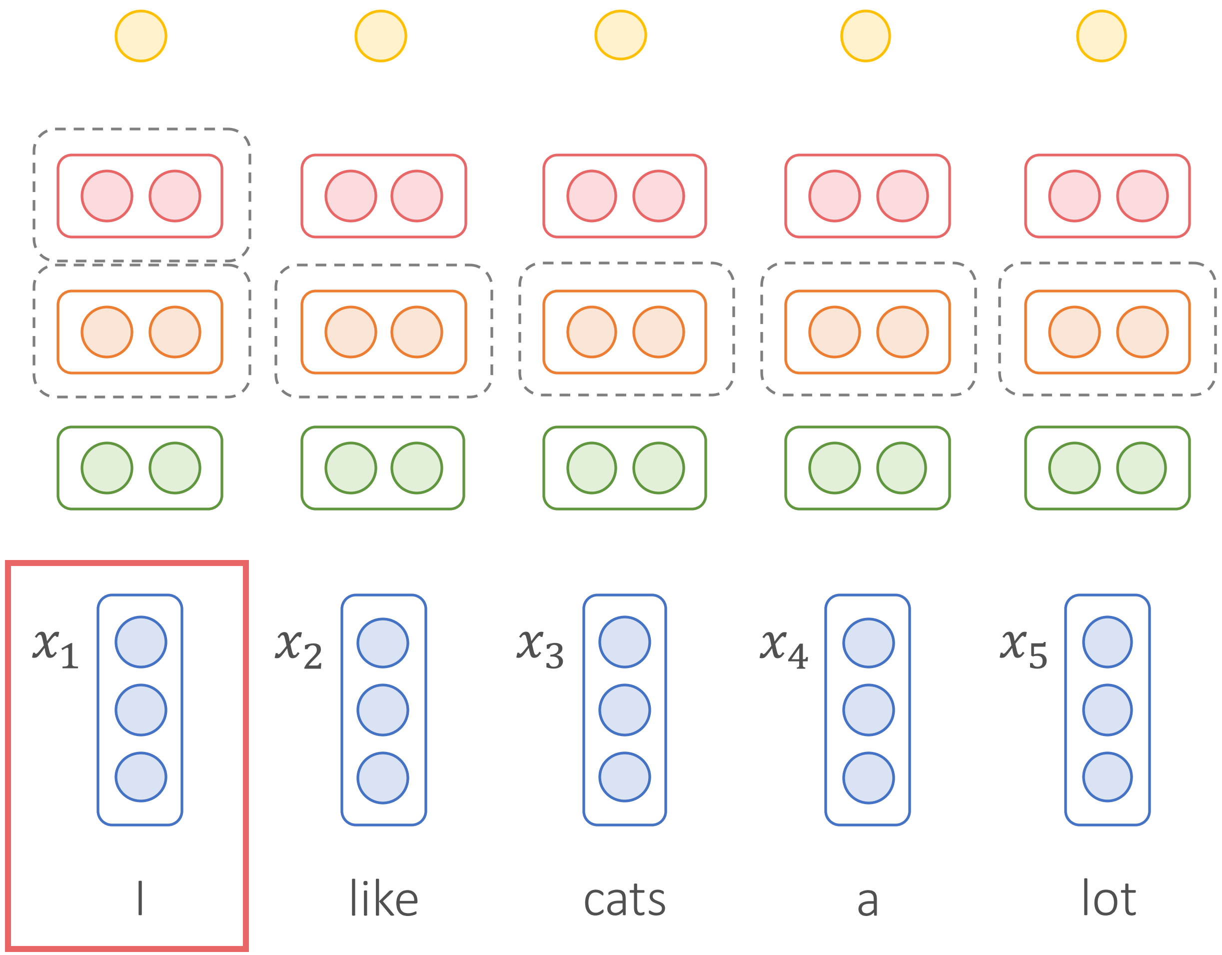
Vector dimension

Normalized
Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention

Weighted Sum

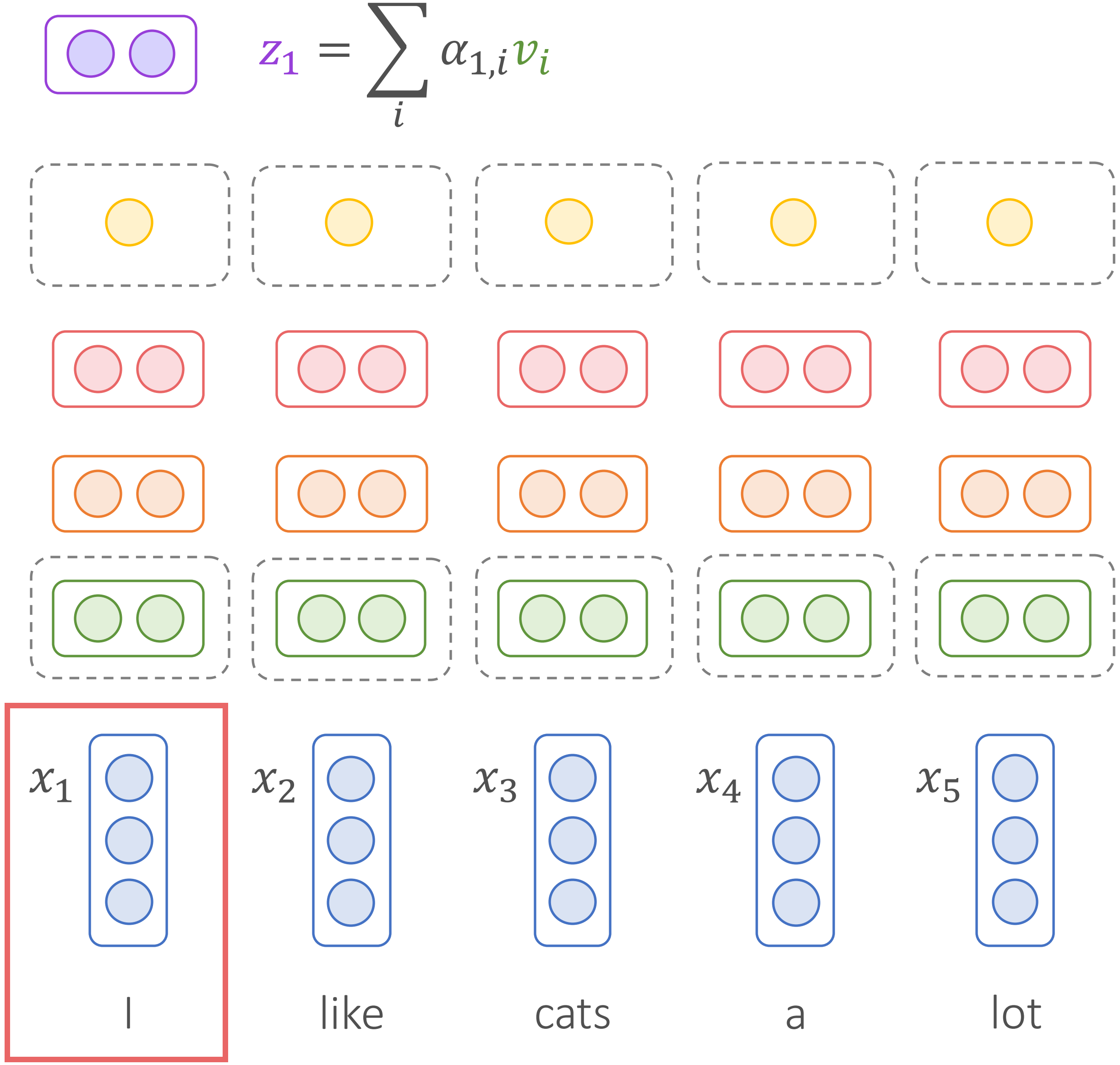
$$z_1 = \sum_i \alpha_{1,i} v_i$$

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



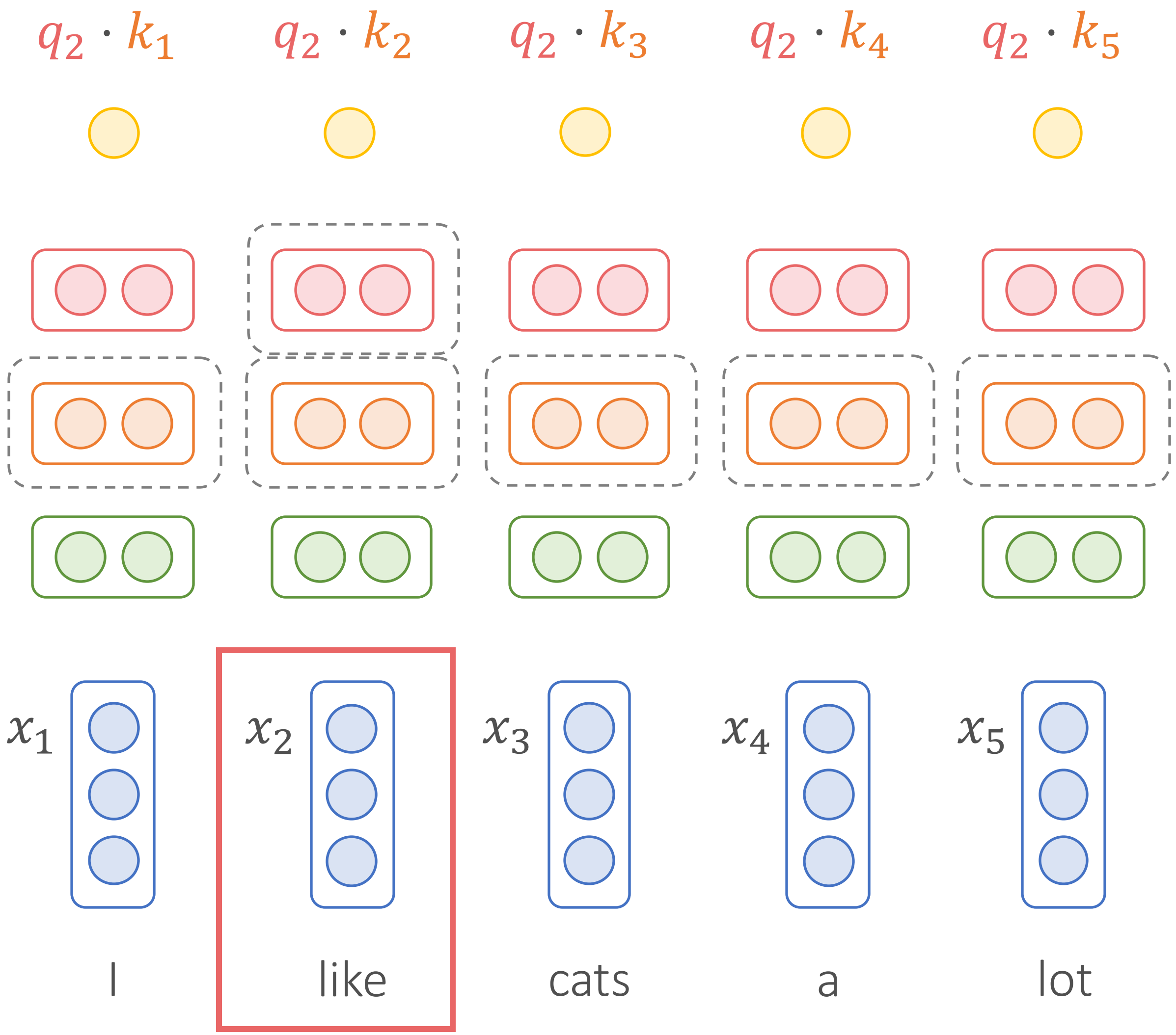
Self-Attention

Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention

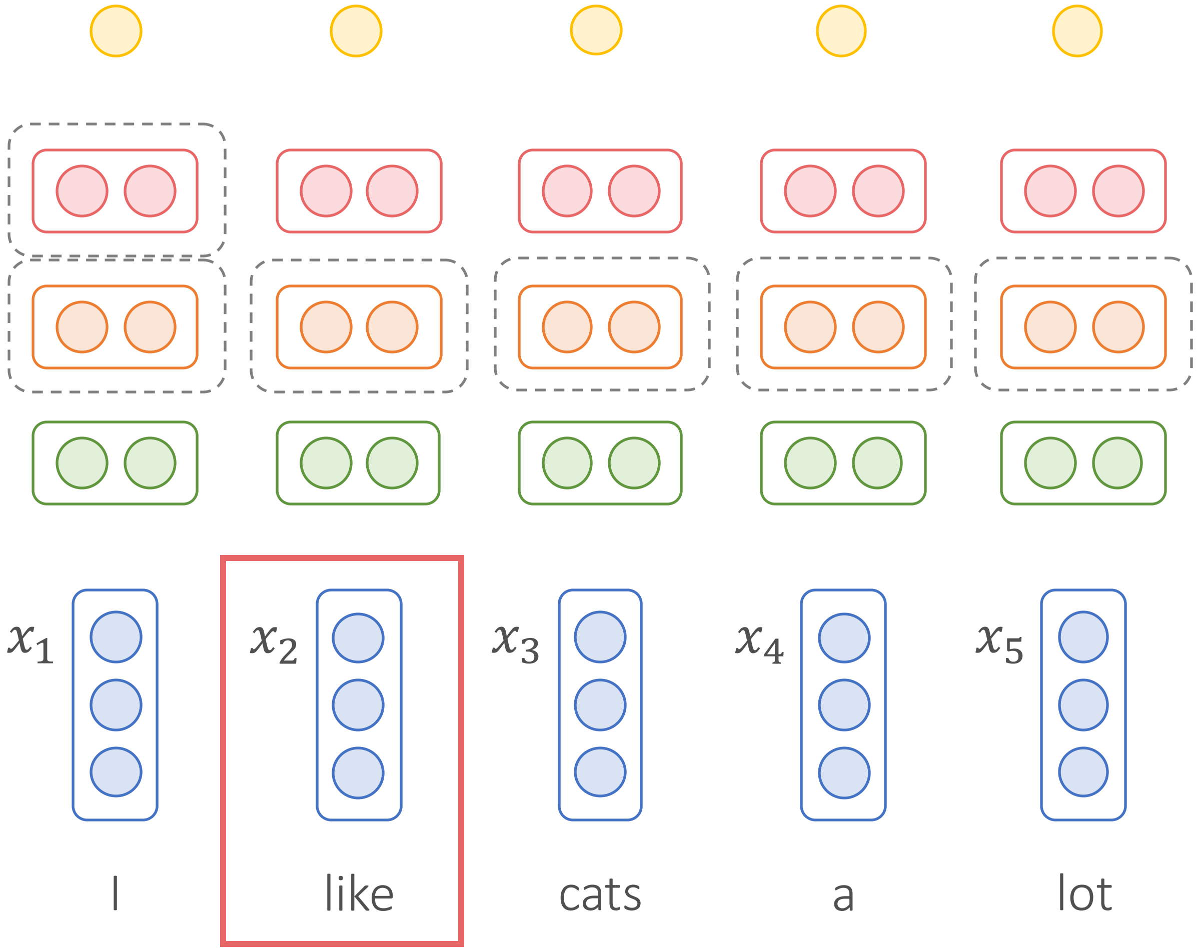
$$\alpha_{2,i} = \text{softmax}\left(\frac{q_2 \cdot k_i}{\sqrt{d}}\right)$$ Vector dimension

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention

Weighted Sum

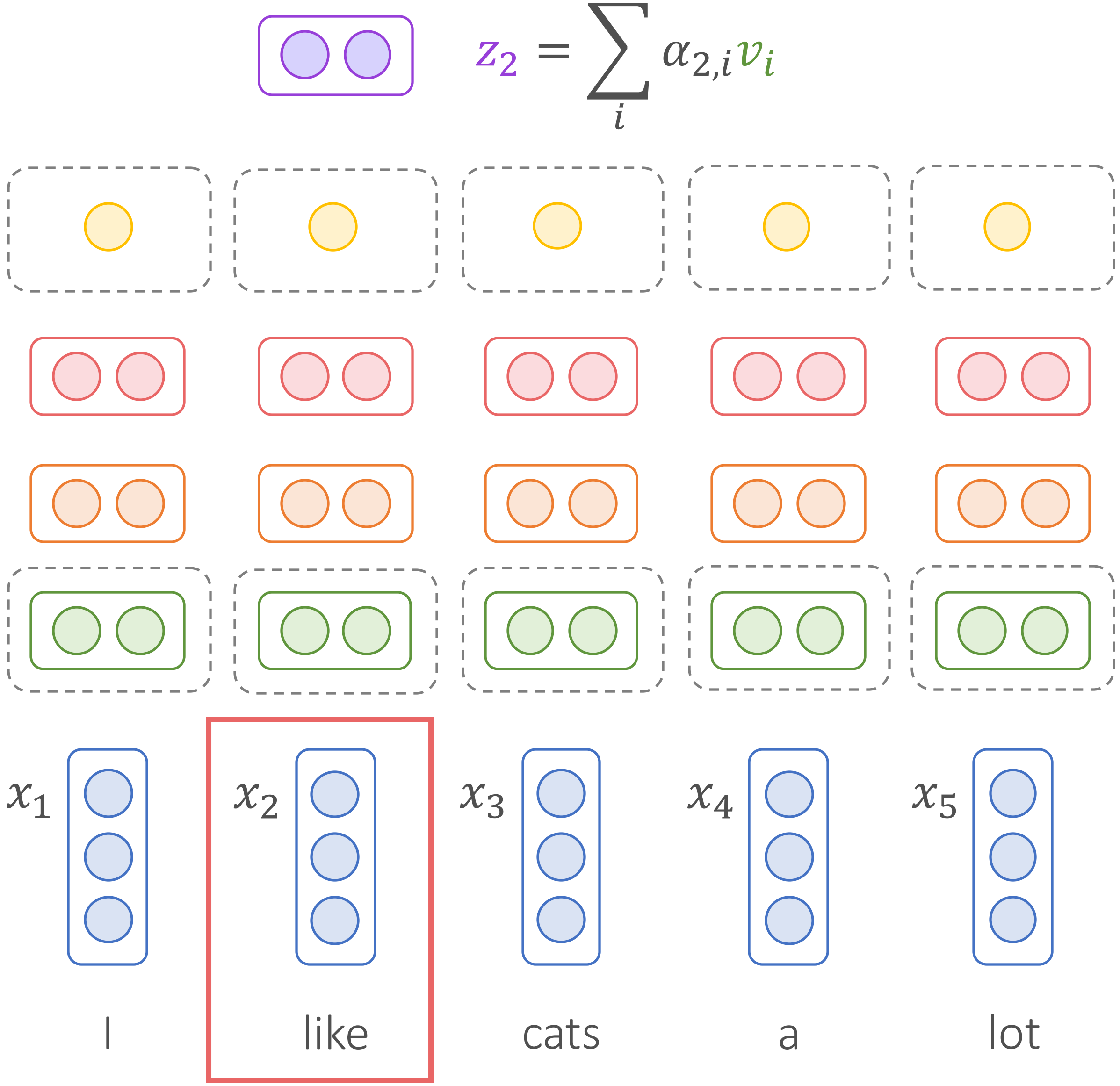
$$z_2 = \sum_i \alpha_{2,i} v_i$$

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



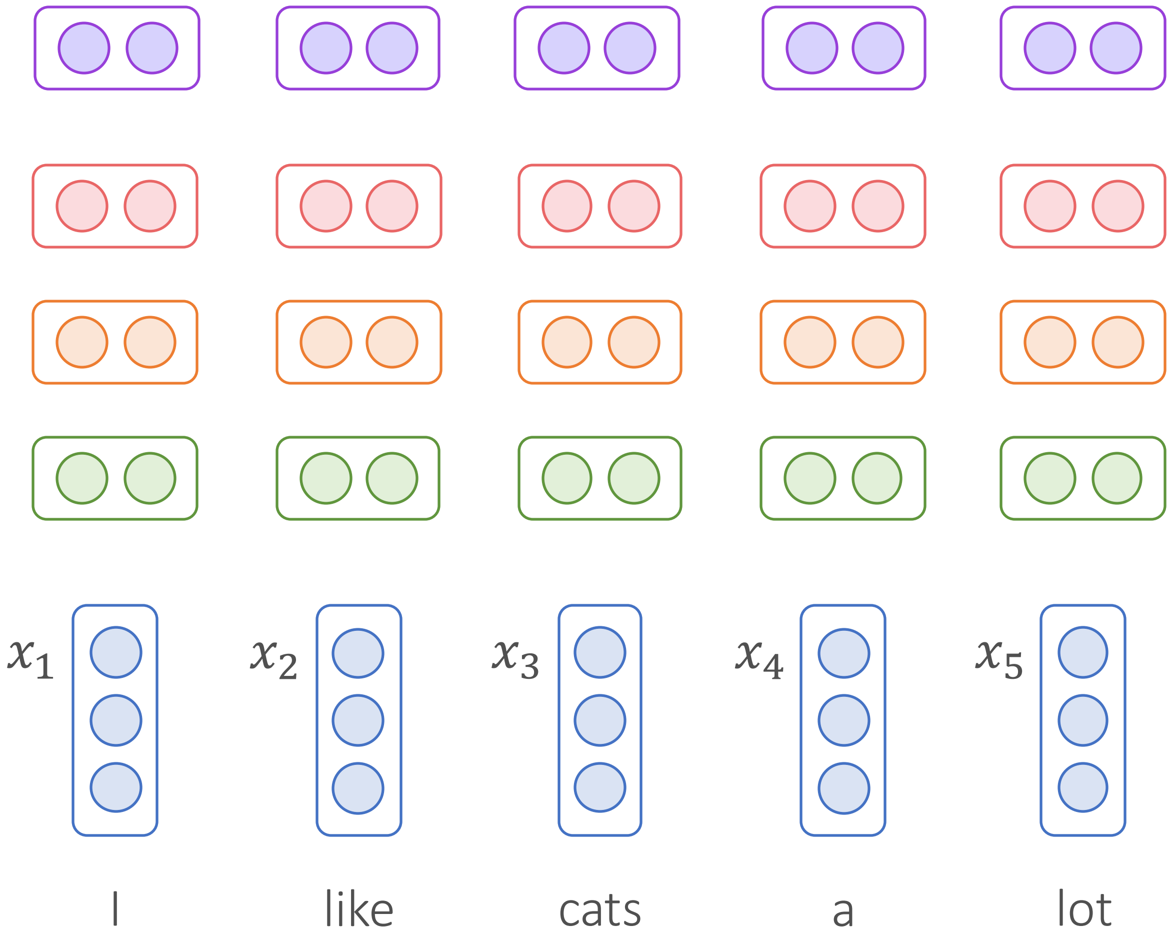
Self-Attention

Self-Attention Output

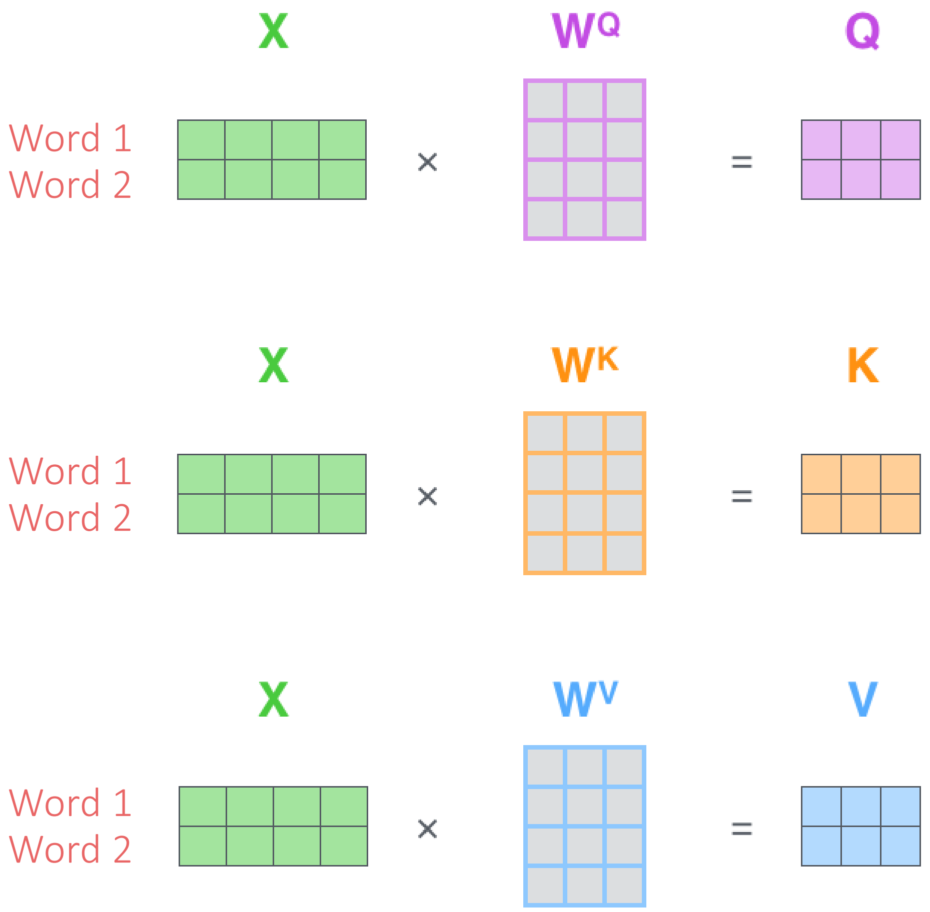
Query $q_i = W^Q x_i$

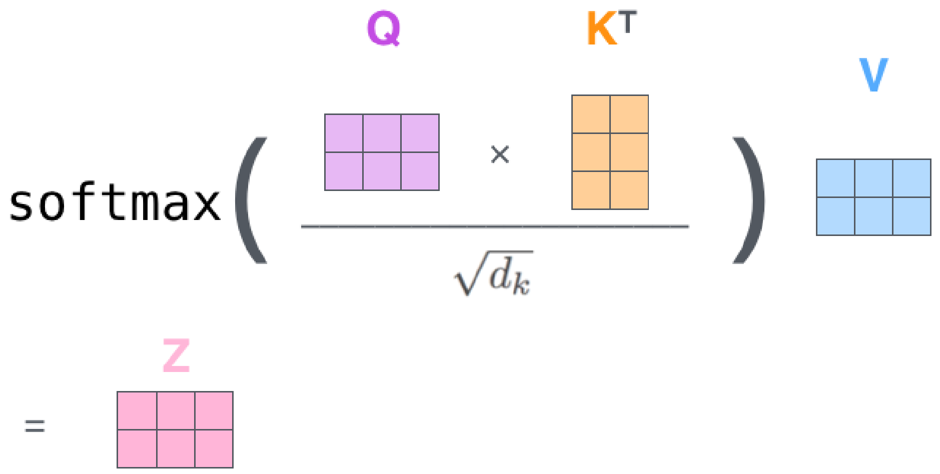
Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention – Matrix Form

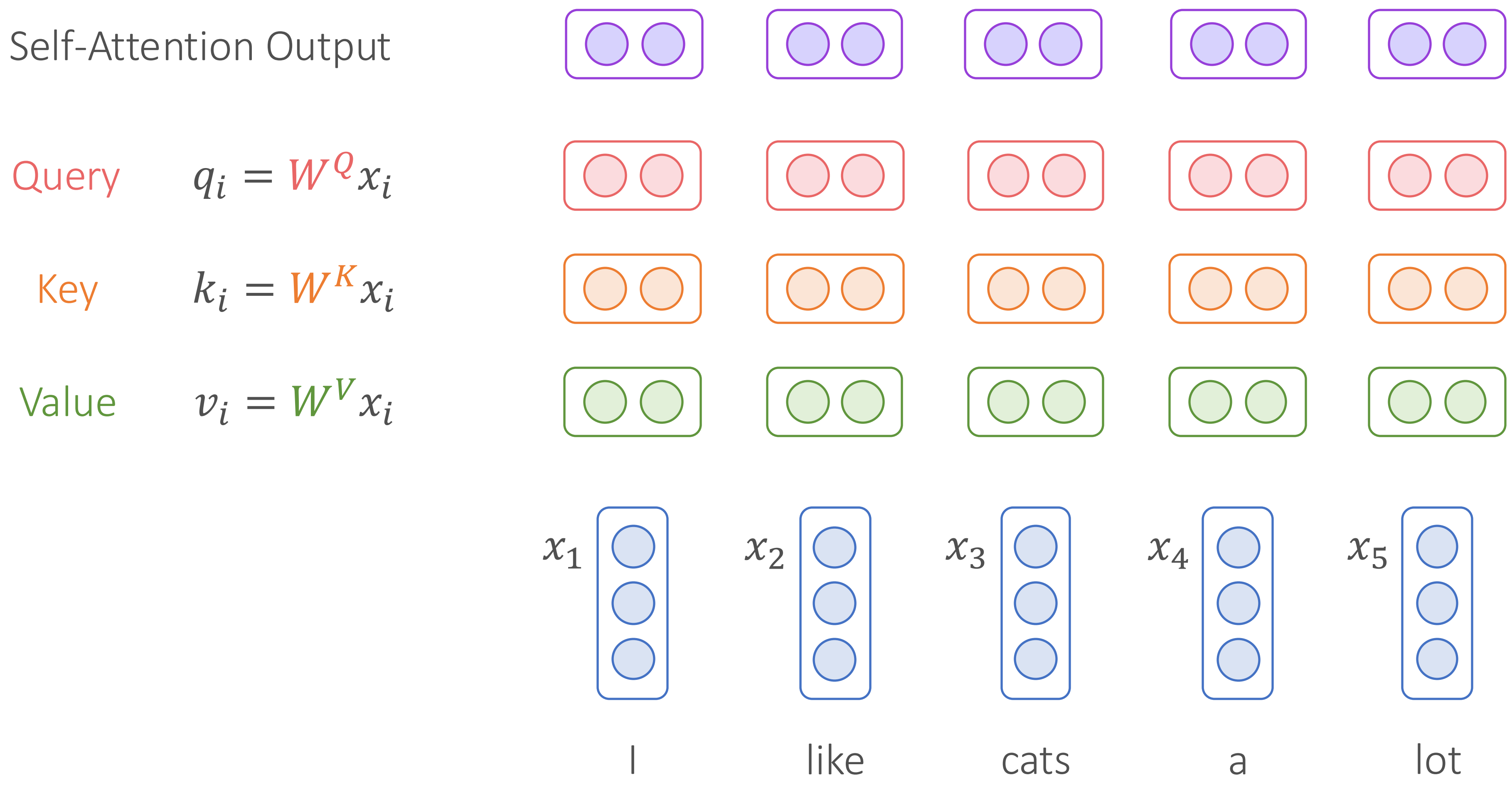




$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

http://jalammar.github.io/illustrated-transformer/

Single-Head Attention



Multi-Head Attention

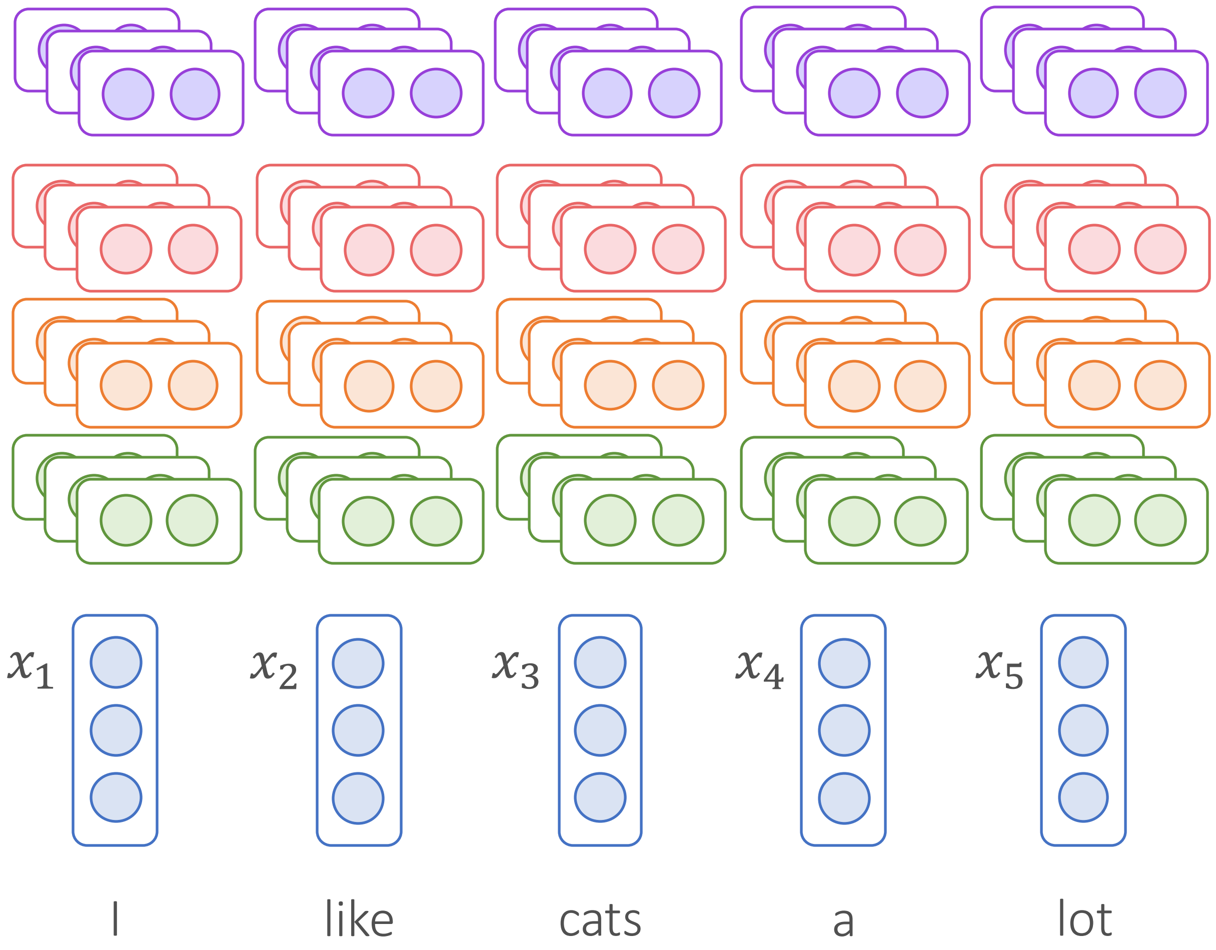
Each attention head focuses on different parts of understanding!

Multi-Attention Output

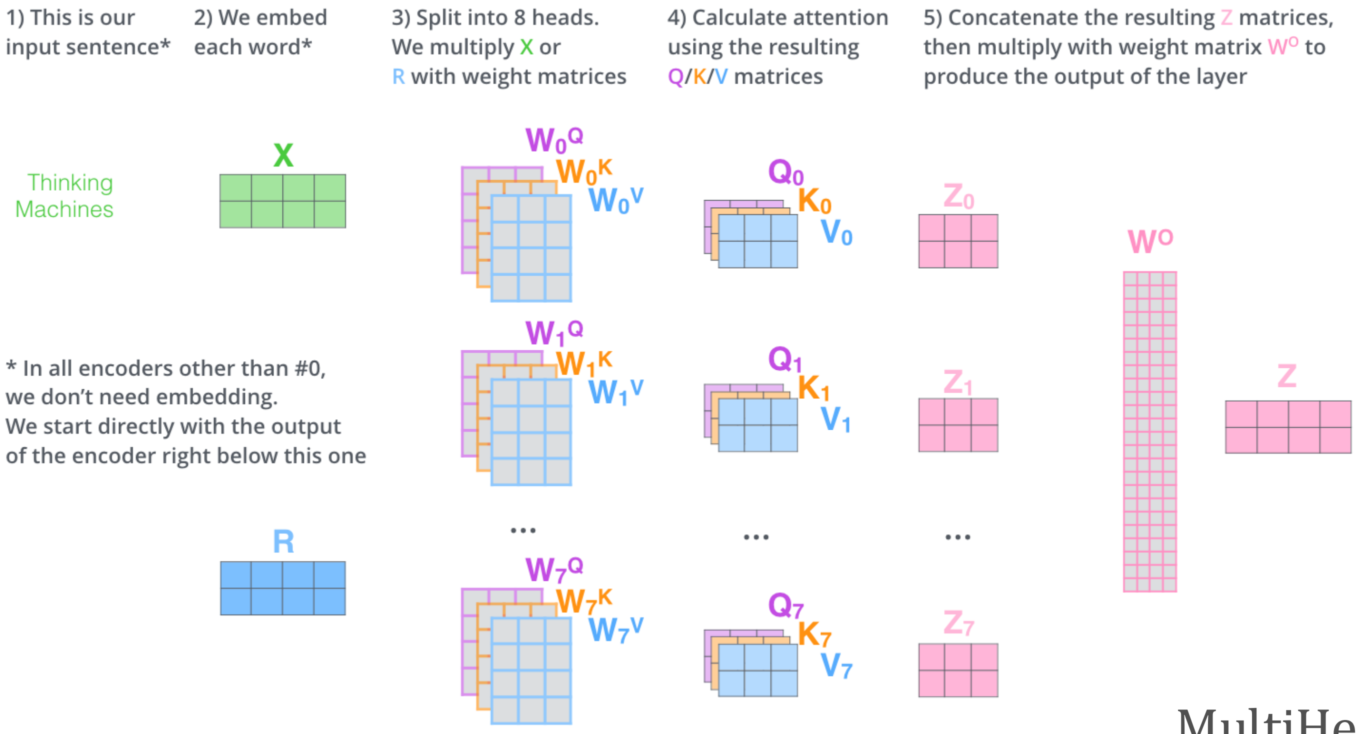
Query $q_i = W_j^Q x_i$

Key $k_i = W_j^K x_i$

Value $v_i = W_j^V x_i$



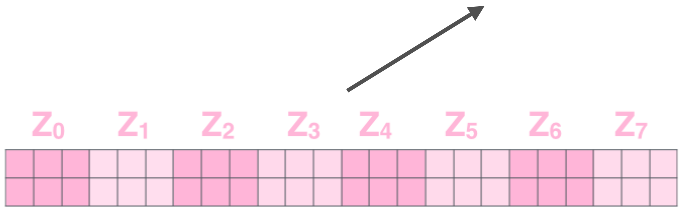
Multi-Head Attention – Matrix Form



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

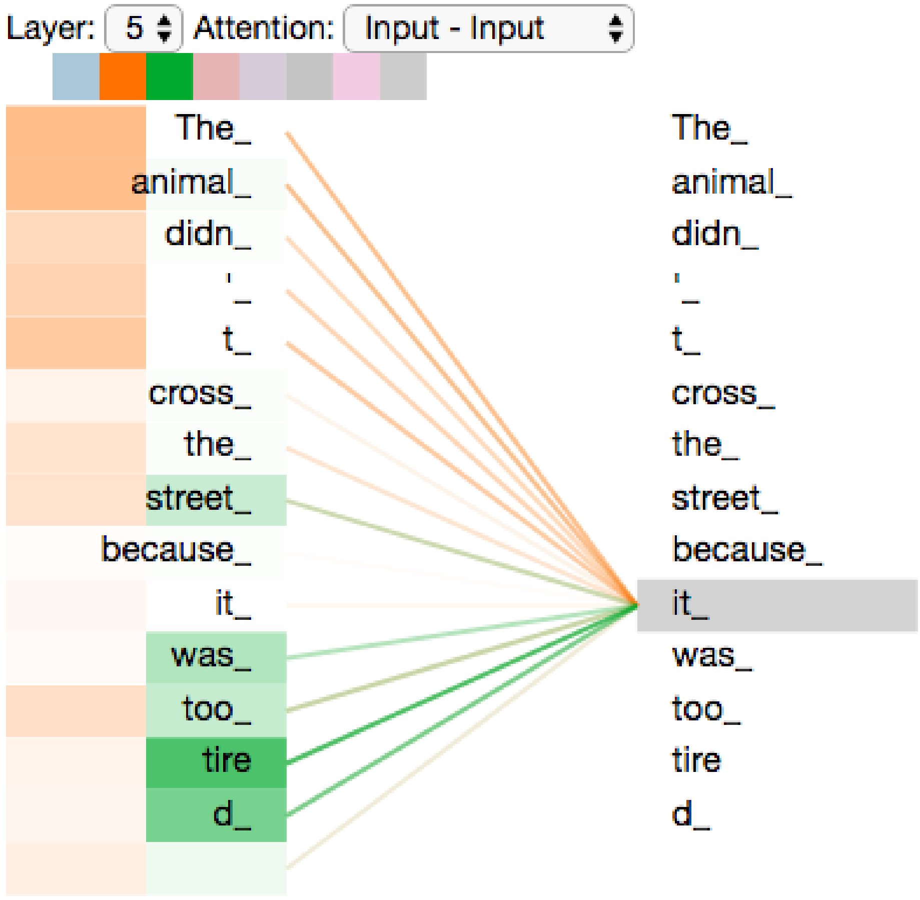
$$\mathrm{head}_i = \mathrm{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

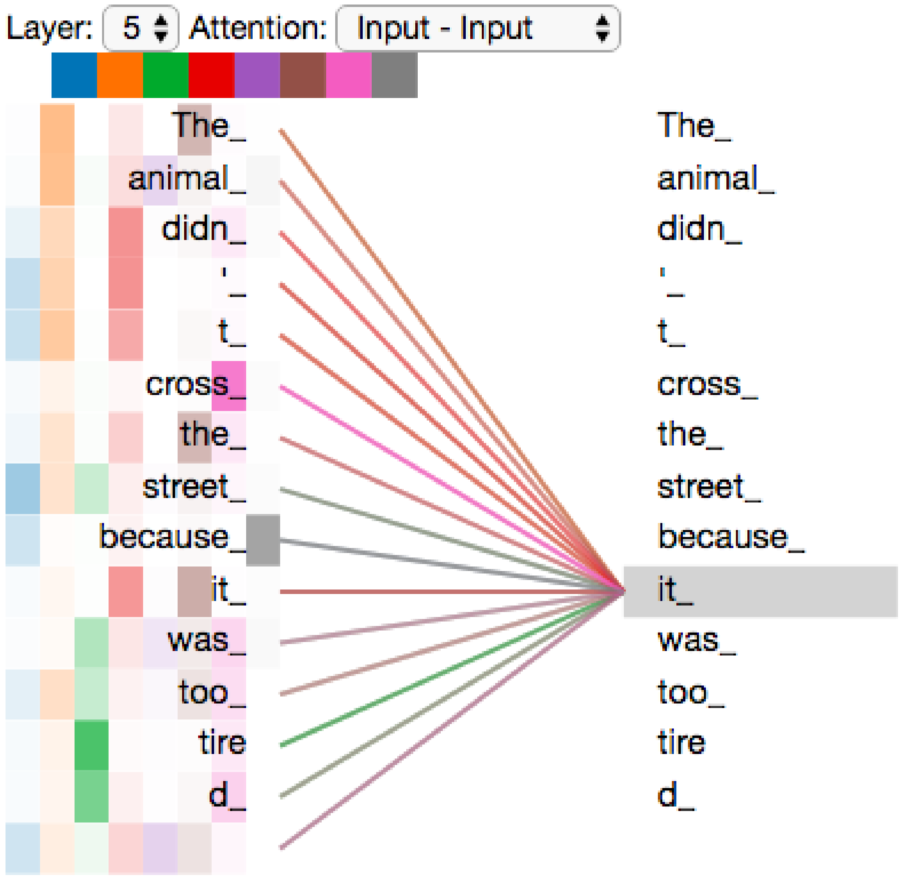
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, \dots, \mathrm{head}_h)W^O$$



http://jalammar.github.io/illustrated-transformer/

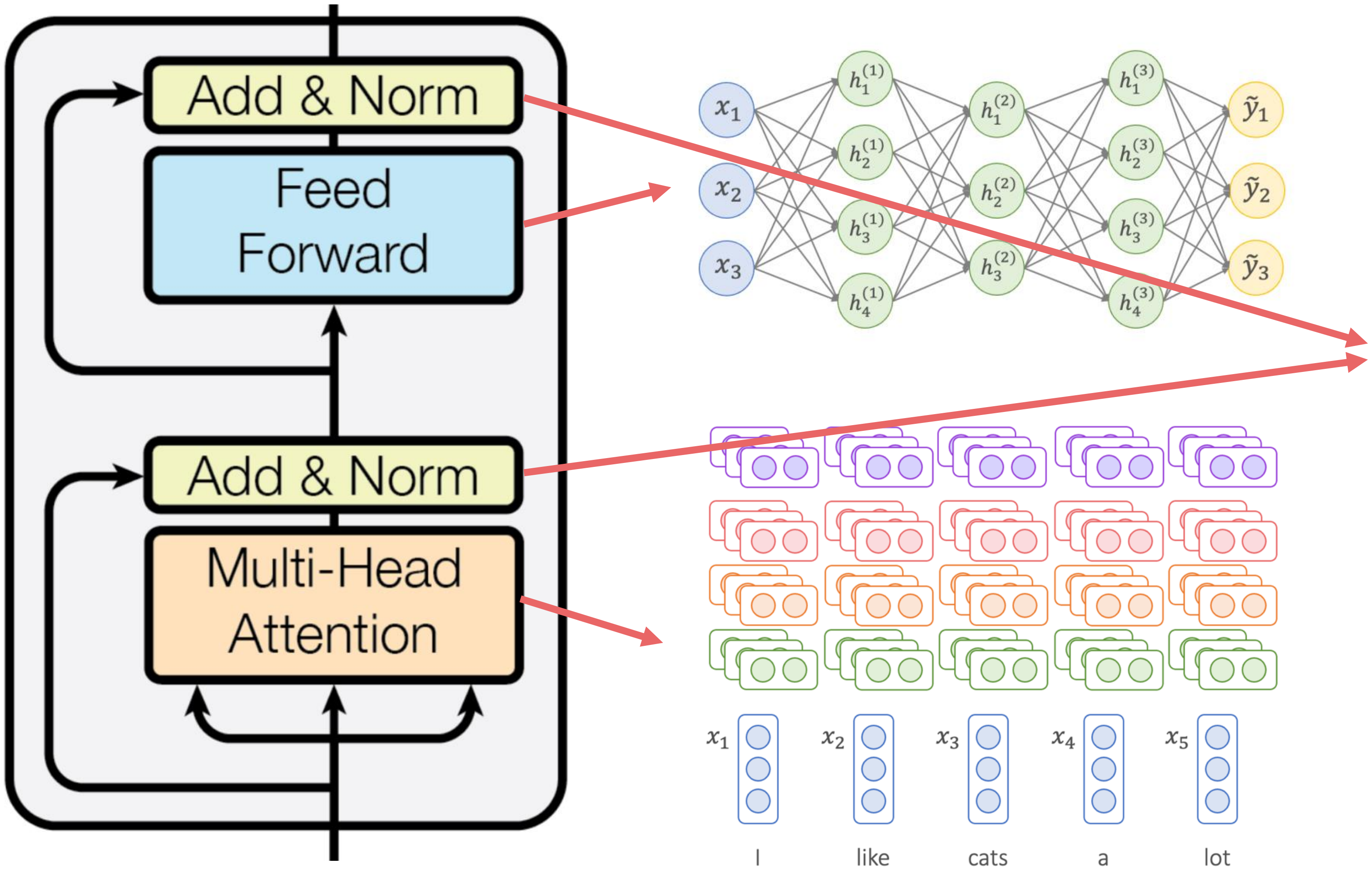
What Does Multi-Head Attention Learn?





http://jalammar.github.io/illustrated-transformer/

Transformer Layer



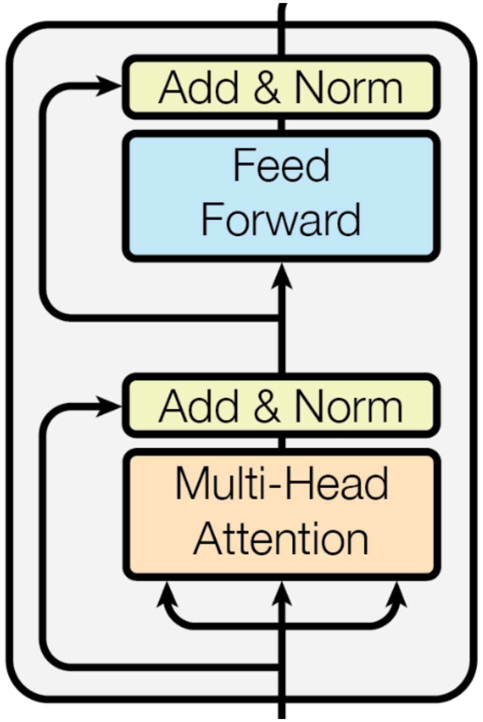
$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

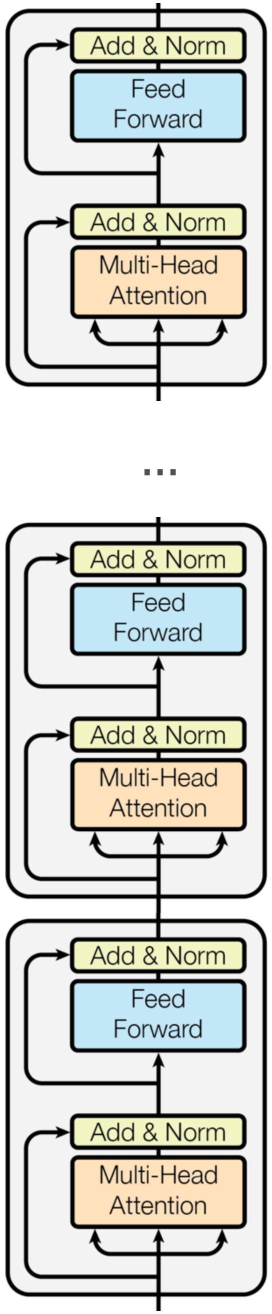
$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

Residual connection (He et al., 2016)
Layer normalization (Ba et al., 2016)

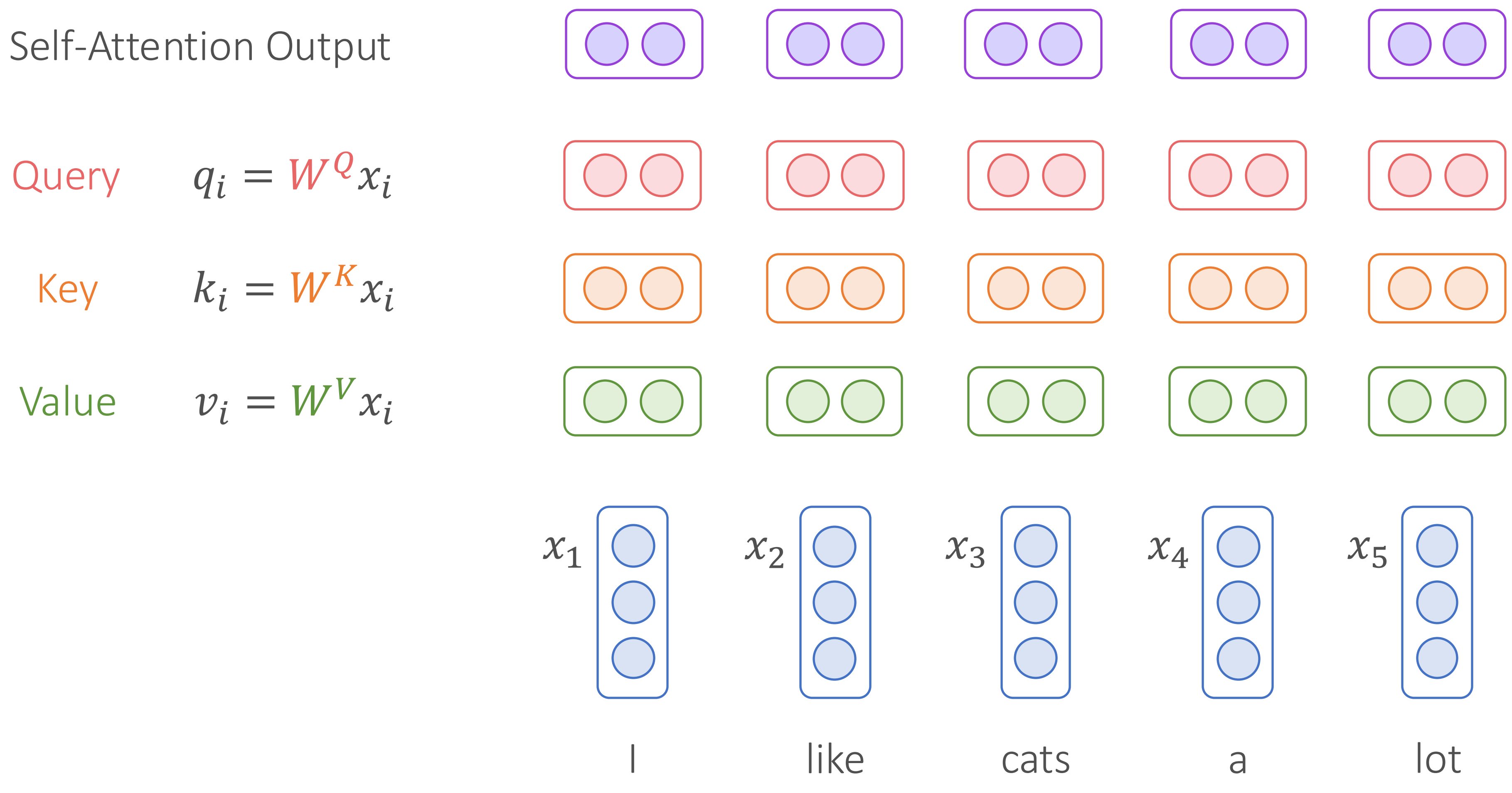
Transformer Encoder

Nx

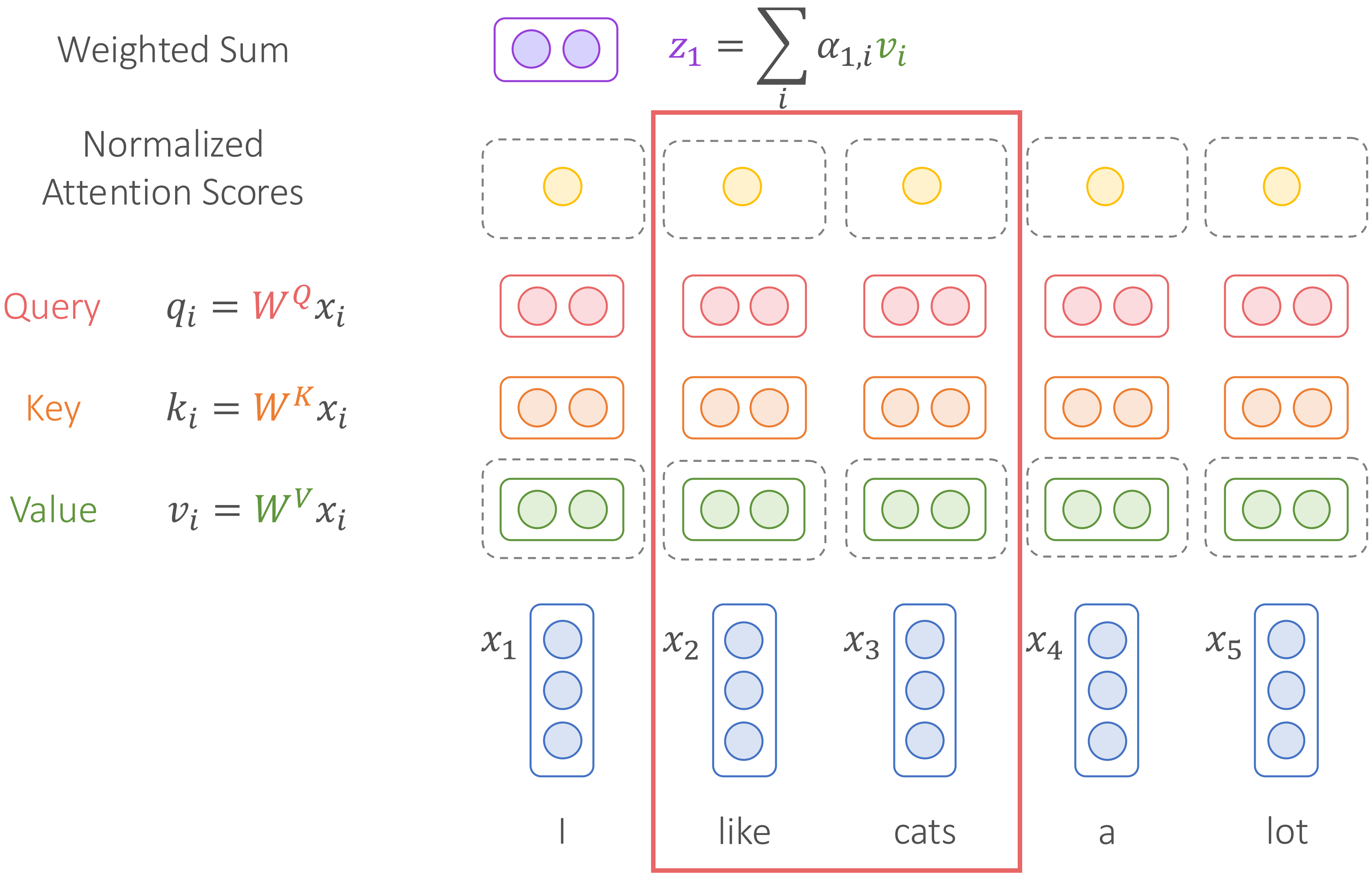




How About Word Order?



How About Word Order?



How About Word Order?

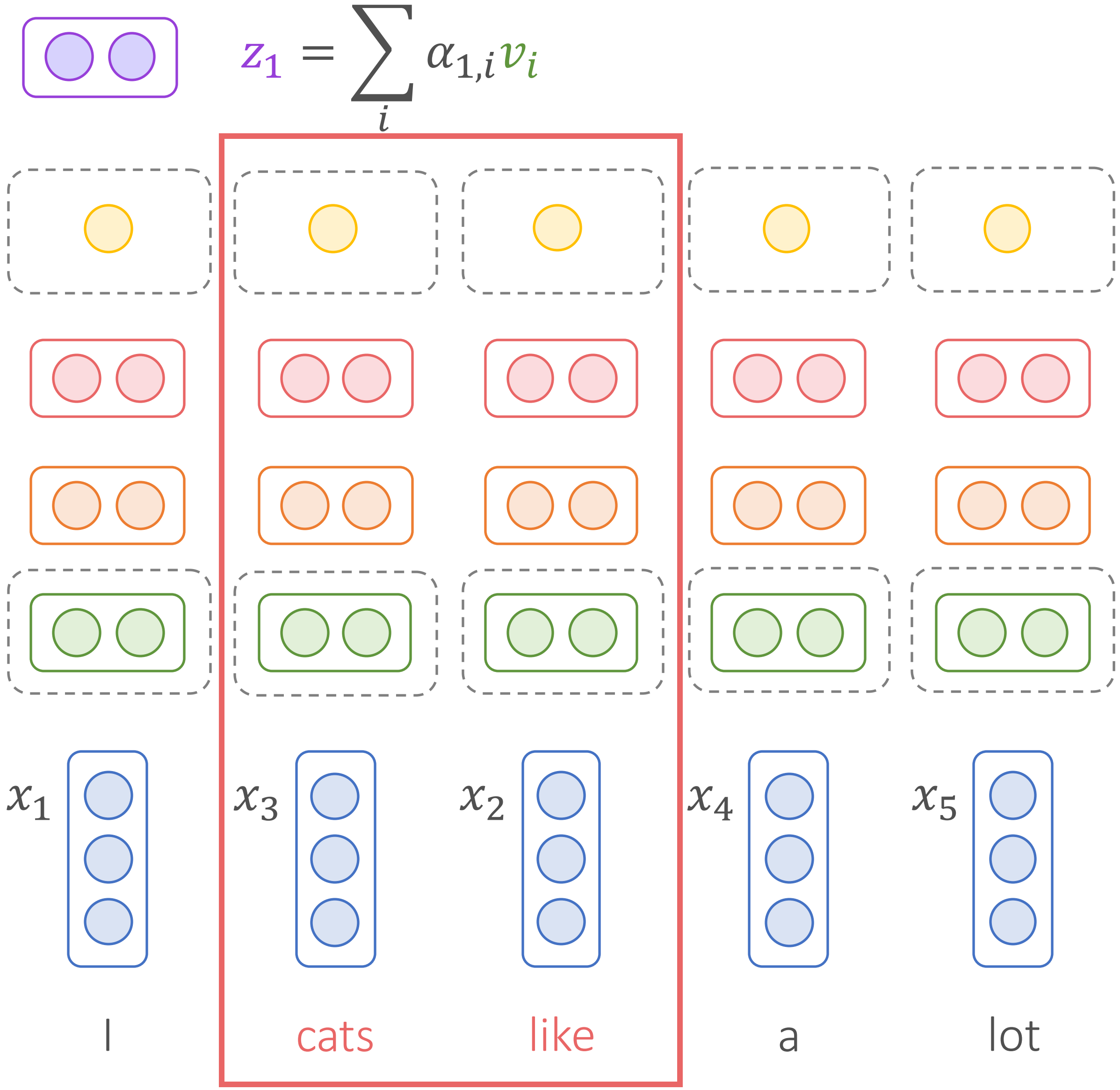
Weighted Sum

Normalized Attention Scores

Query $q_i = W^Q x_i$

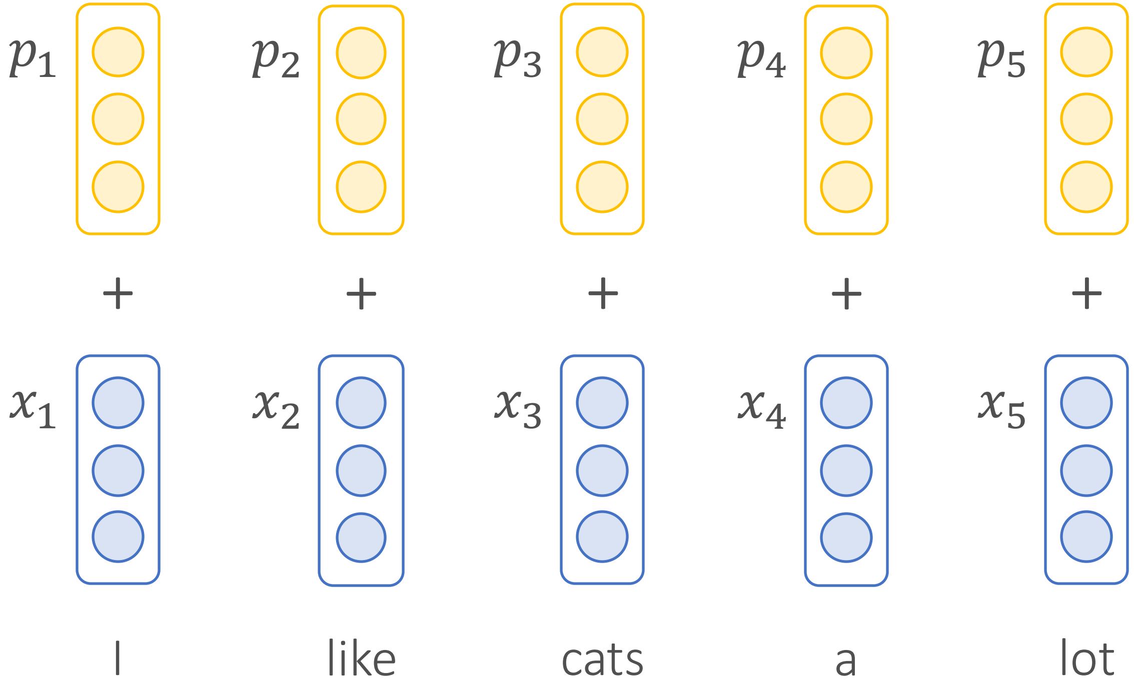
Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



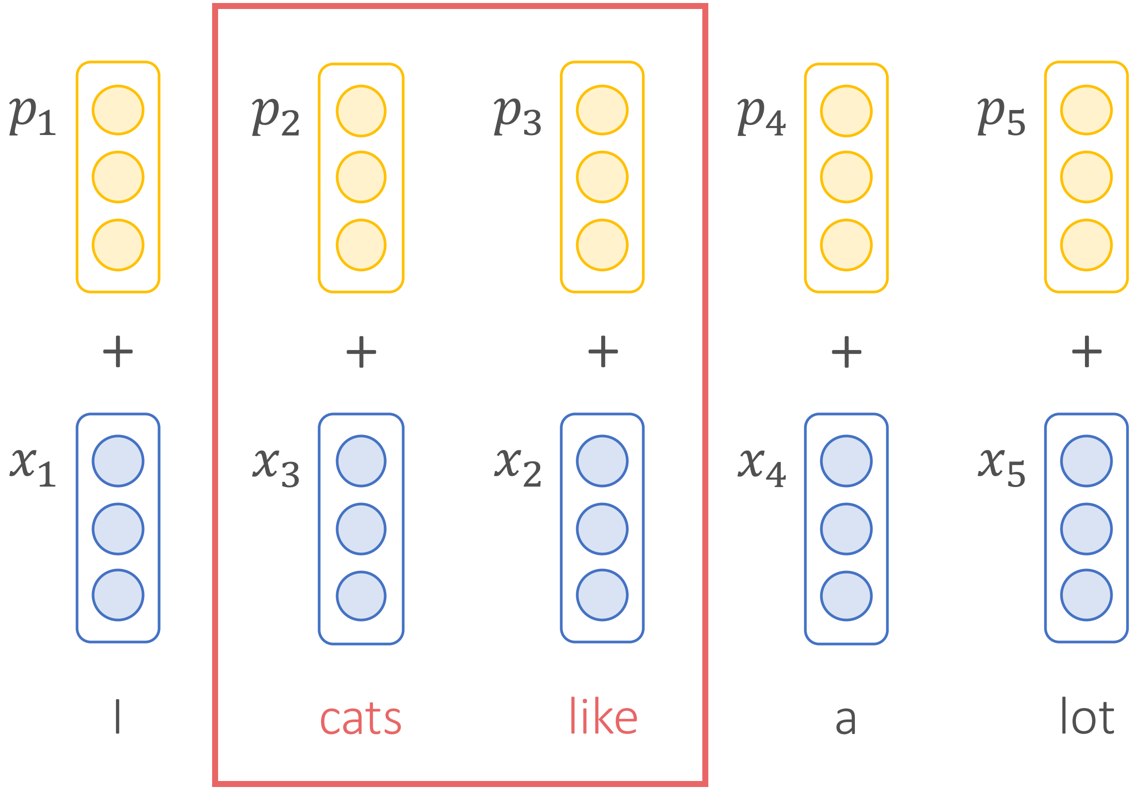
Solution: Positional Encoding

$$x_i \leftarrow x_i + PE_i$$



Solution: Positional Encoding

$$x_i \leftarrow x_i + PE_i$$



Solution: Positional Encoding

- Unique encoding for each position
- Closer positions should have more similar encodings
- Distance between neighboring positions should be the same

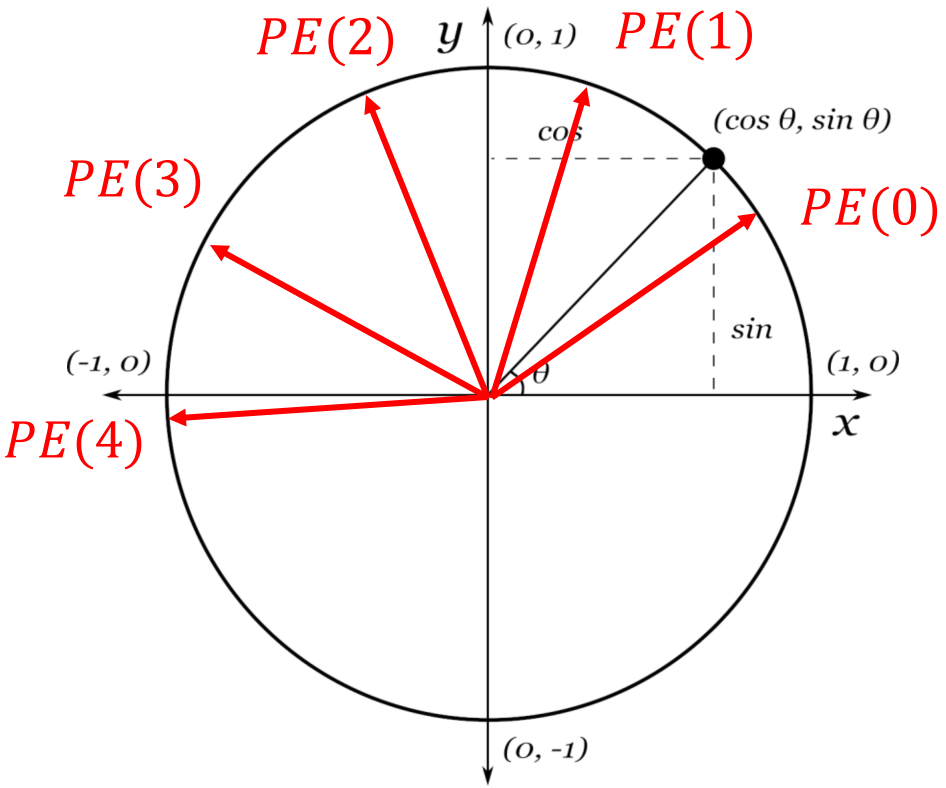
Sinusoidal Positional Encoding

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

Why this?

Sinusoidal Positional Encoding: Intuition

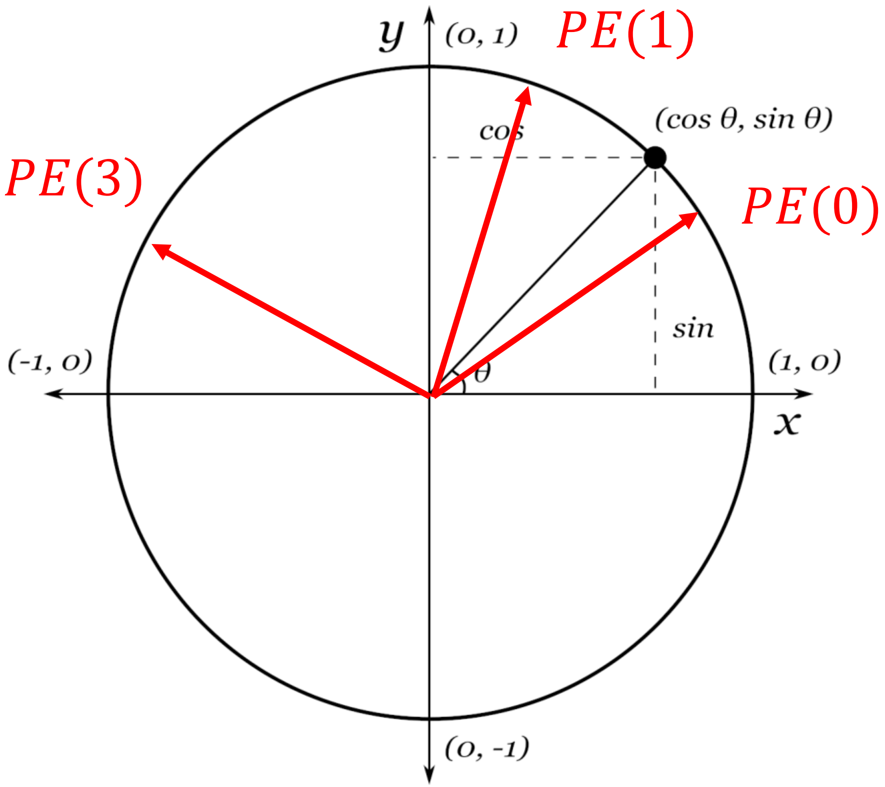
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



https://www.inchcalculator.com/unit-circle-calculator/

Sinusoidal Positional Encoding: Intuition

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



$Cosine\big(PE(0), PE(1)\big) > Cosine(PE(0), PE(3))$

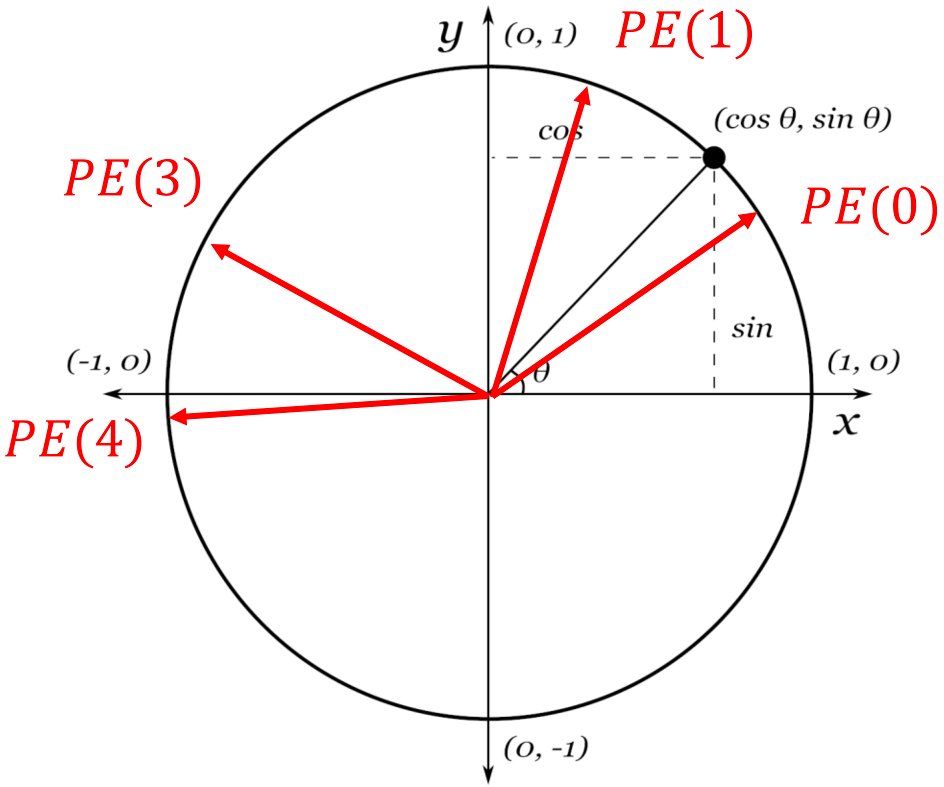
Closer positions should have more similar encodings

https://www.inchcalculator.com/unit-circle-calculator/

Sinusoidal Positional Encoding: Intuition

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

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



$$Cosine\big(PE(0), PE(1)\big) = Cosine(PE(3), PE(4))$$

Distance between neighboring positions should be the same

https://www.inchcalculator.com/unit-circle-calculator/

Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let's consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

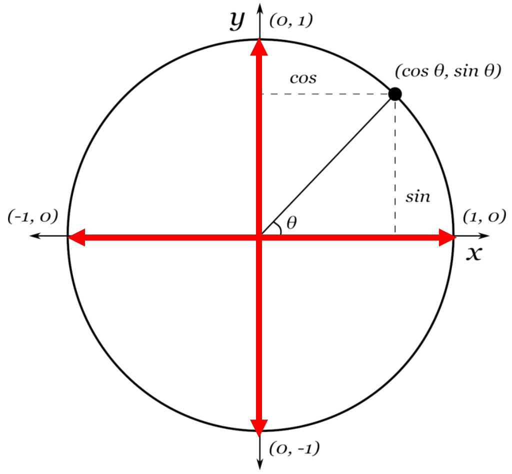
0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let's consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1



https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let's consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

High frequency rotation

Low frequency rotation

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let's consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

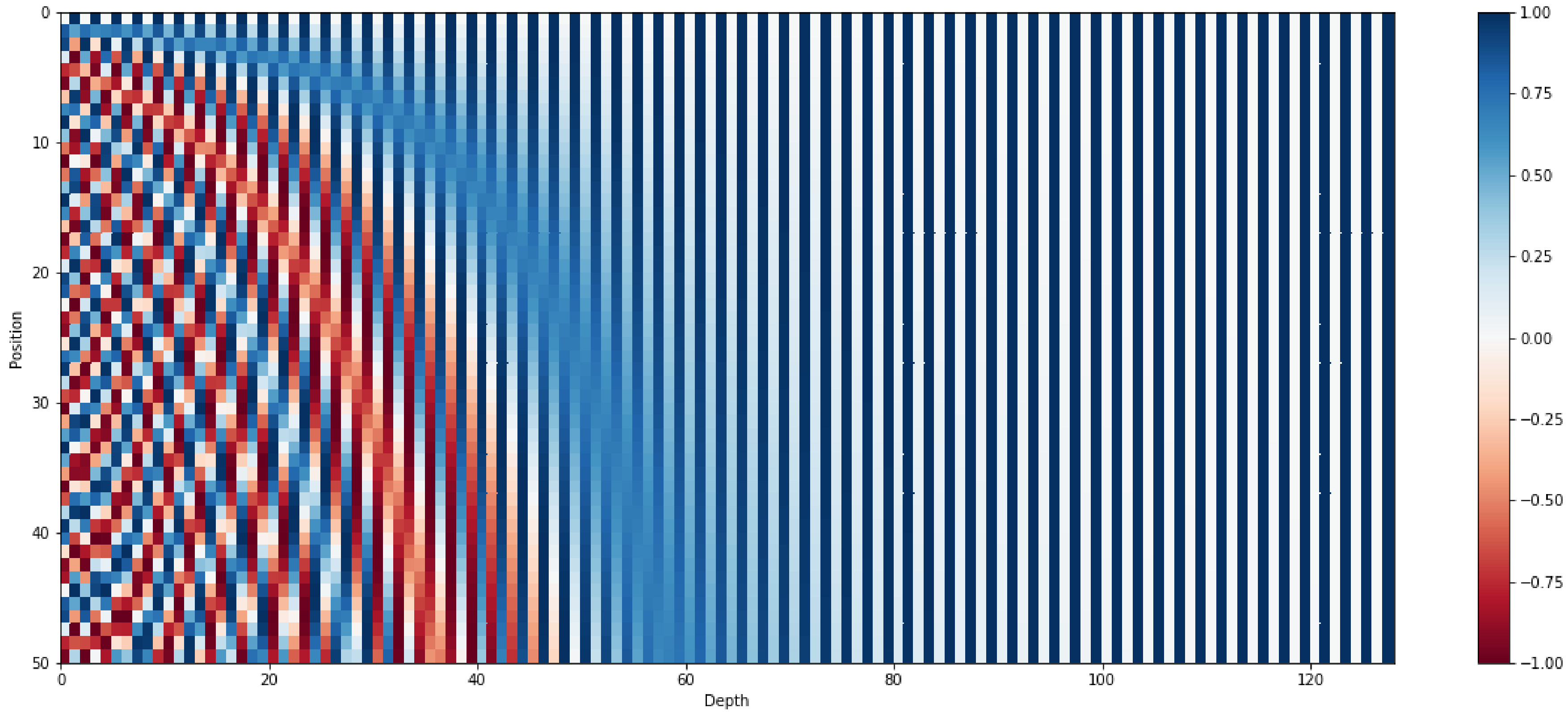
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Soft version of alternating bits

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

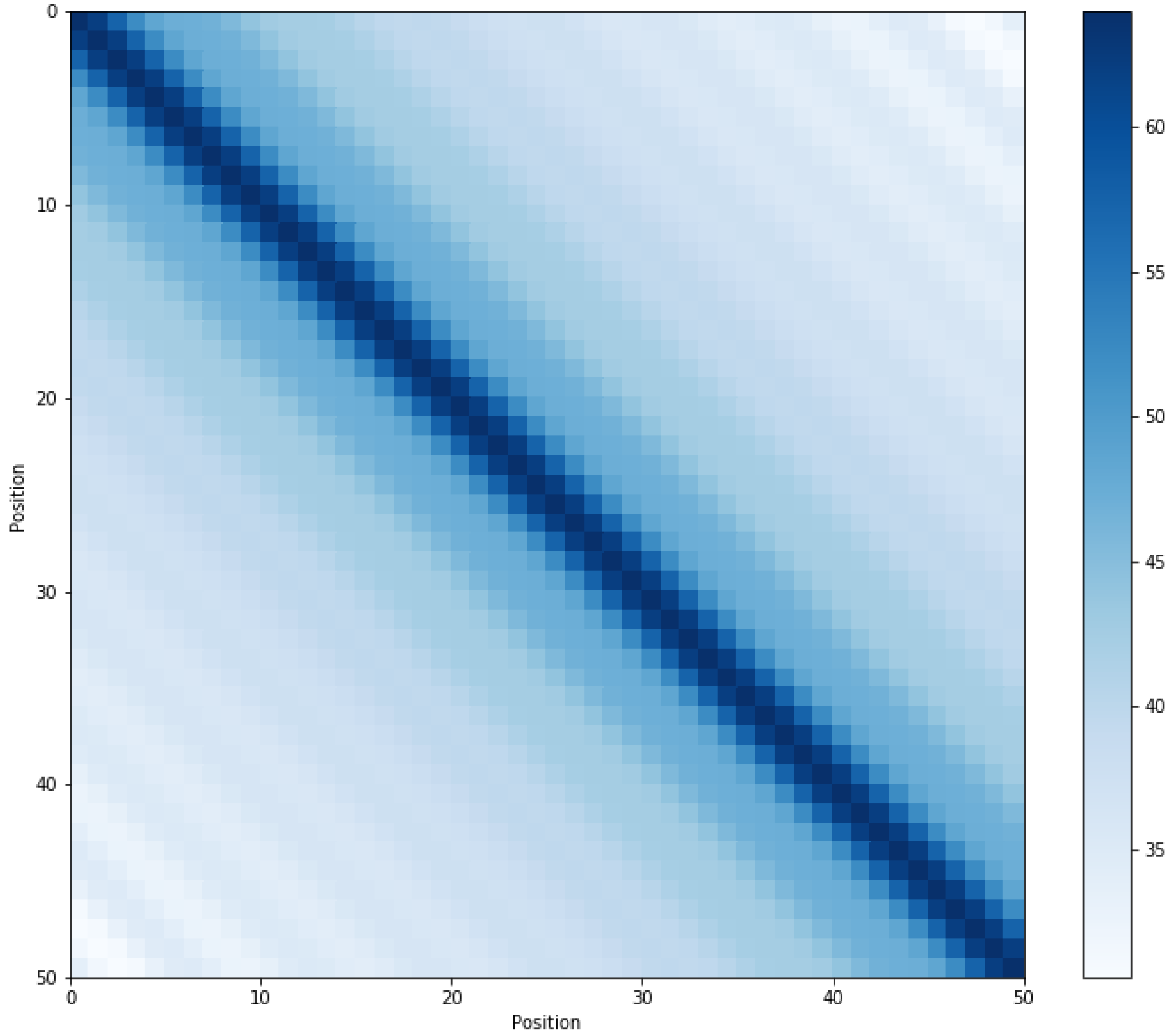
Sinusoidal Positional Encoding

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



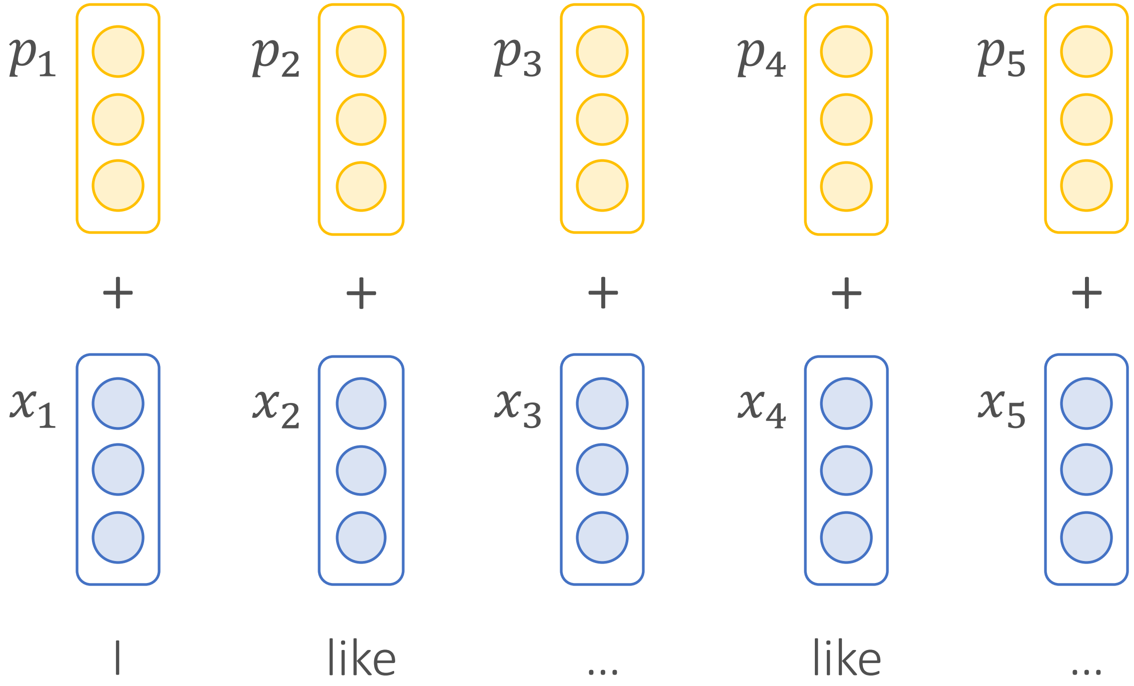
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Sinusoidal Positional Encoding



https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Positional Encoding



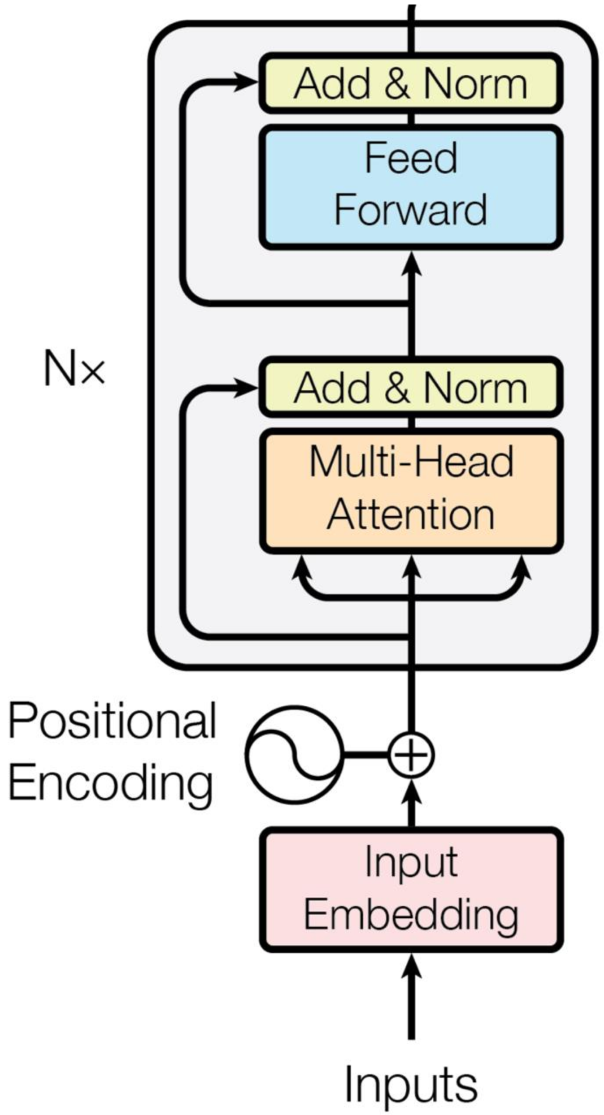
$$\left(E(\mathrm{I}) + PE(1)\right)\left(E(\mathrm{like}) + PE(2)\right) = E(\mathrm{I})E(\mathrm{like}) + \boxed{E(\mathrm{I})PE(2)} + PE(1)E(\mathrm{like}) + \boxed{PE(1)PE(2)}$$

$$\left(E(\mathrm{I}) + PE(1)\right)\left(E(\mathrm{like}) + PE(4)\right) = E(\mathrm{I})E(\mathrm{like}) + \boxed{E(\mathrm{I})PE(4)} + PE(1)E(\mathrm{like}) + \boxed{PE(1)PE(4)}$$

In expectation, they are the same

Position difference

Transformer Encoder with Positional Encoding

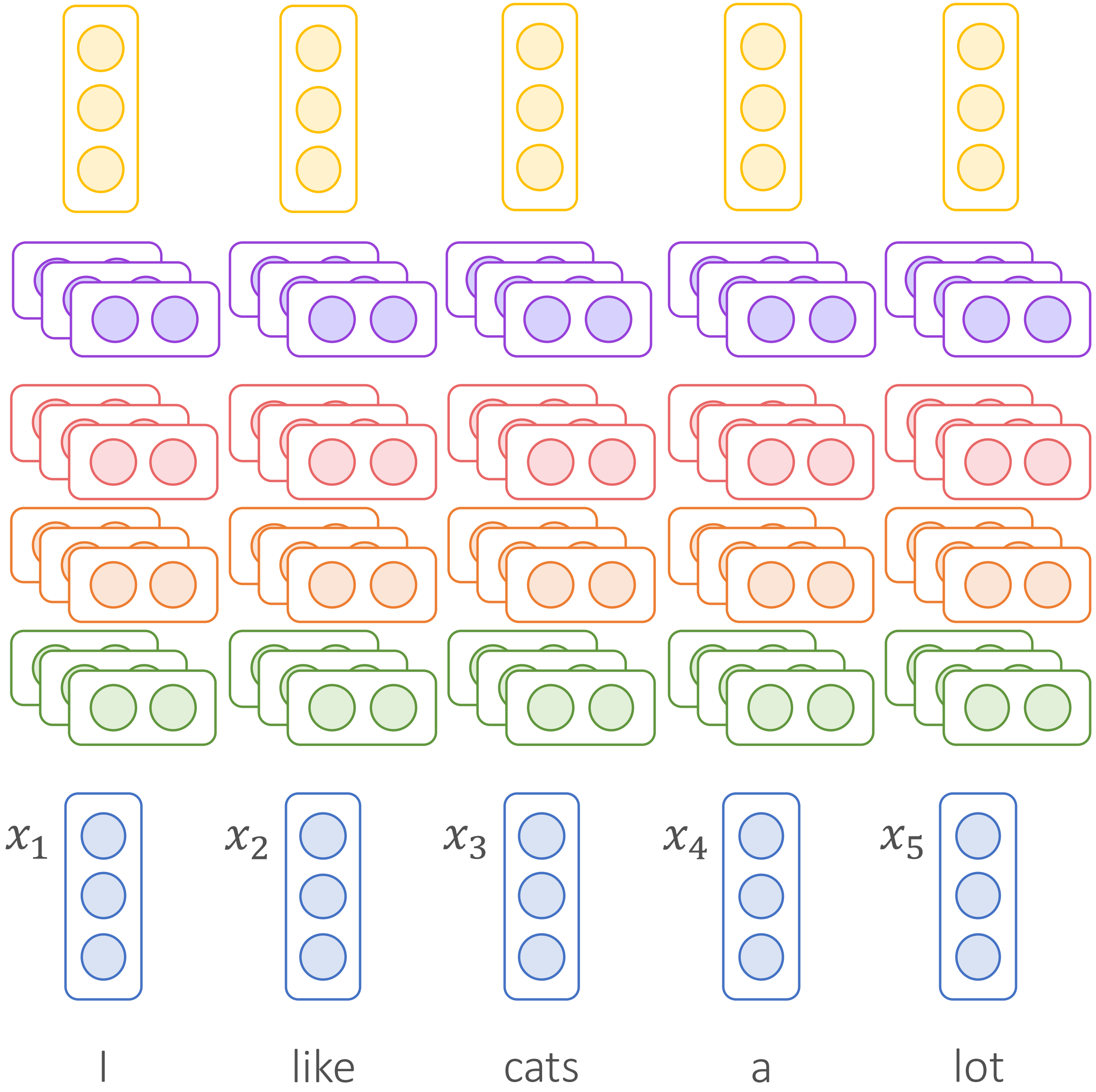


Transformer as Token-Level Encoder

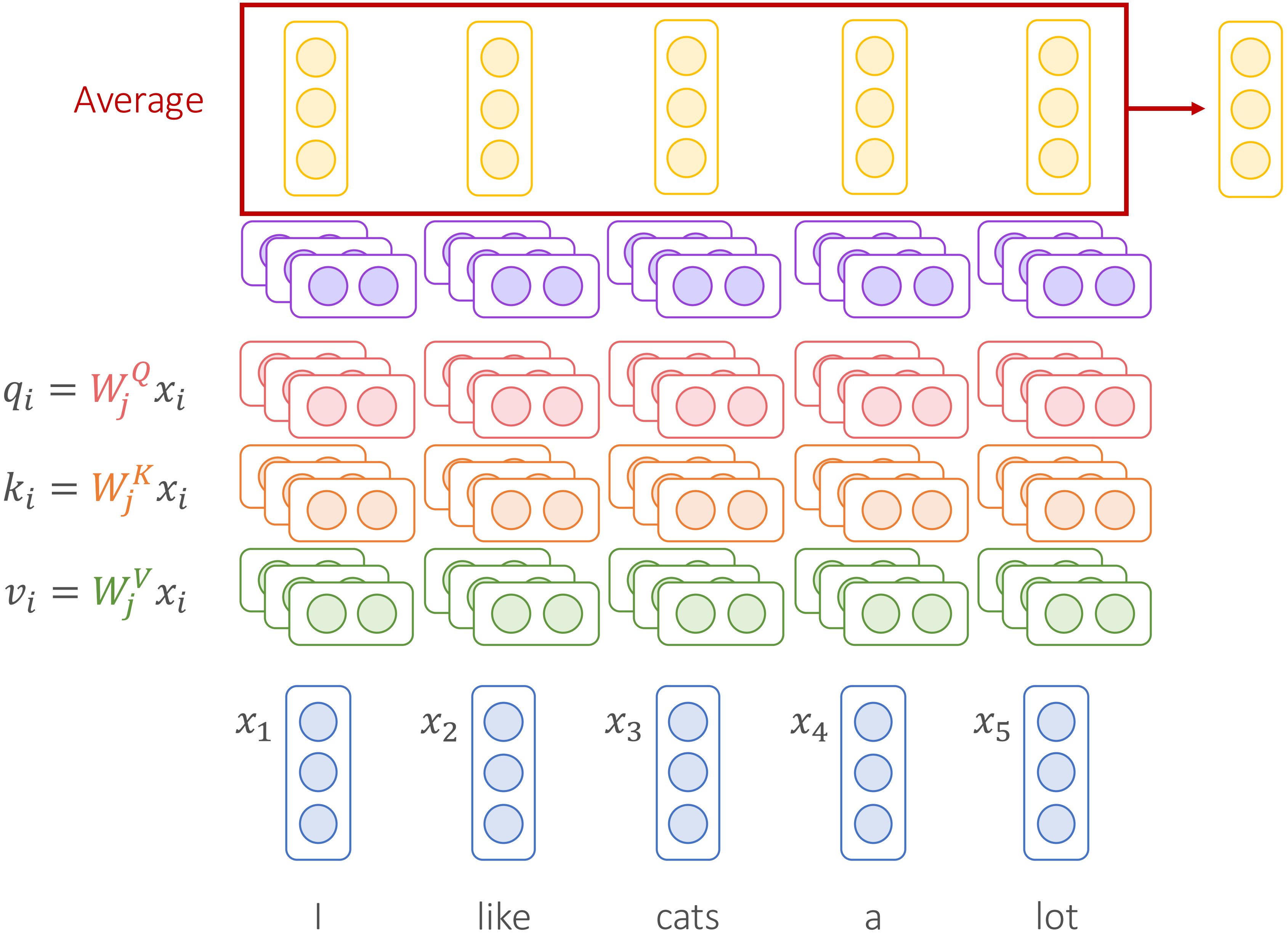
$$q_i = W_j^Q x_i$$

$$k_i = W_j^K x_i$$

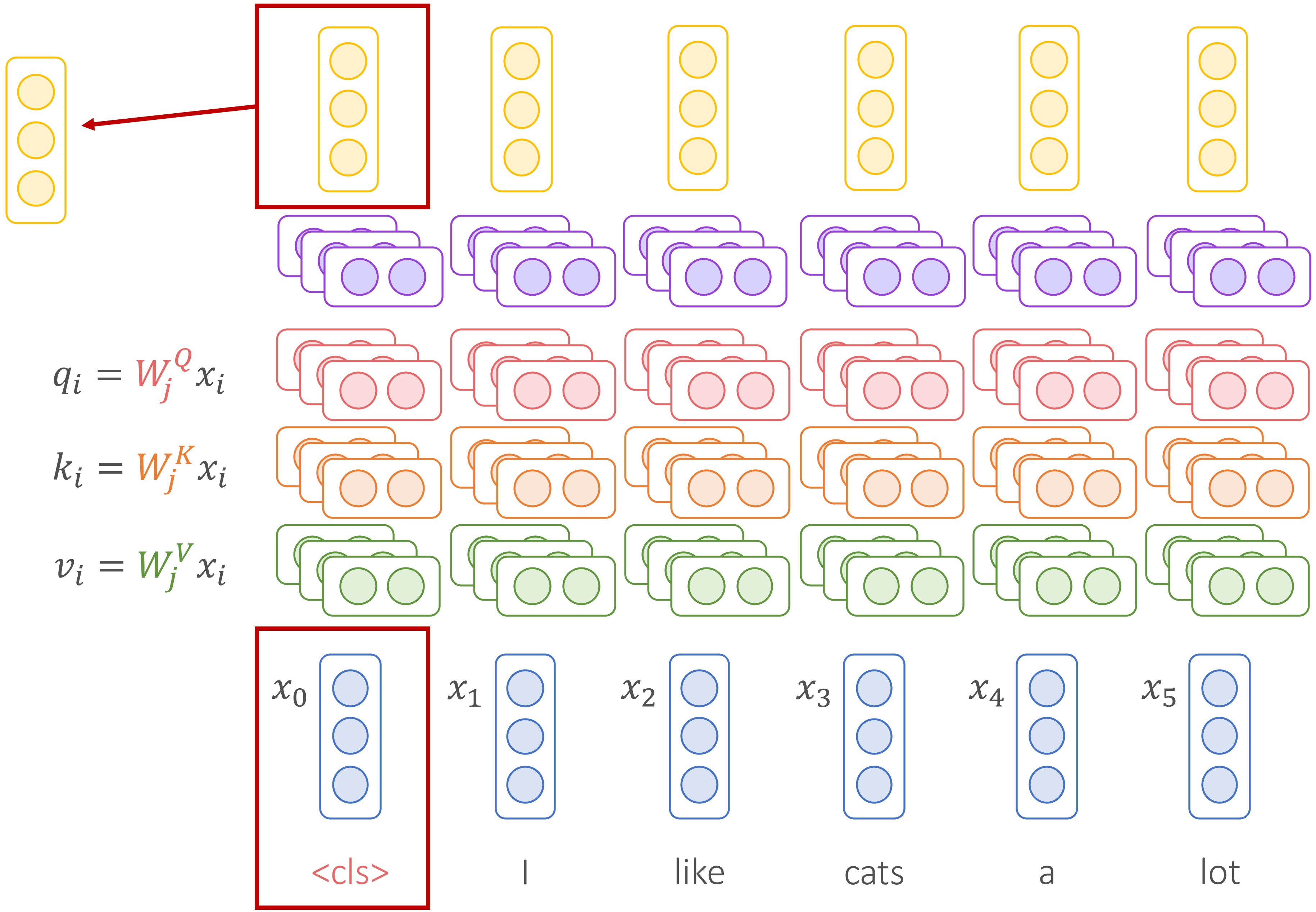
$$v_i = W_j^V x_i$$



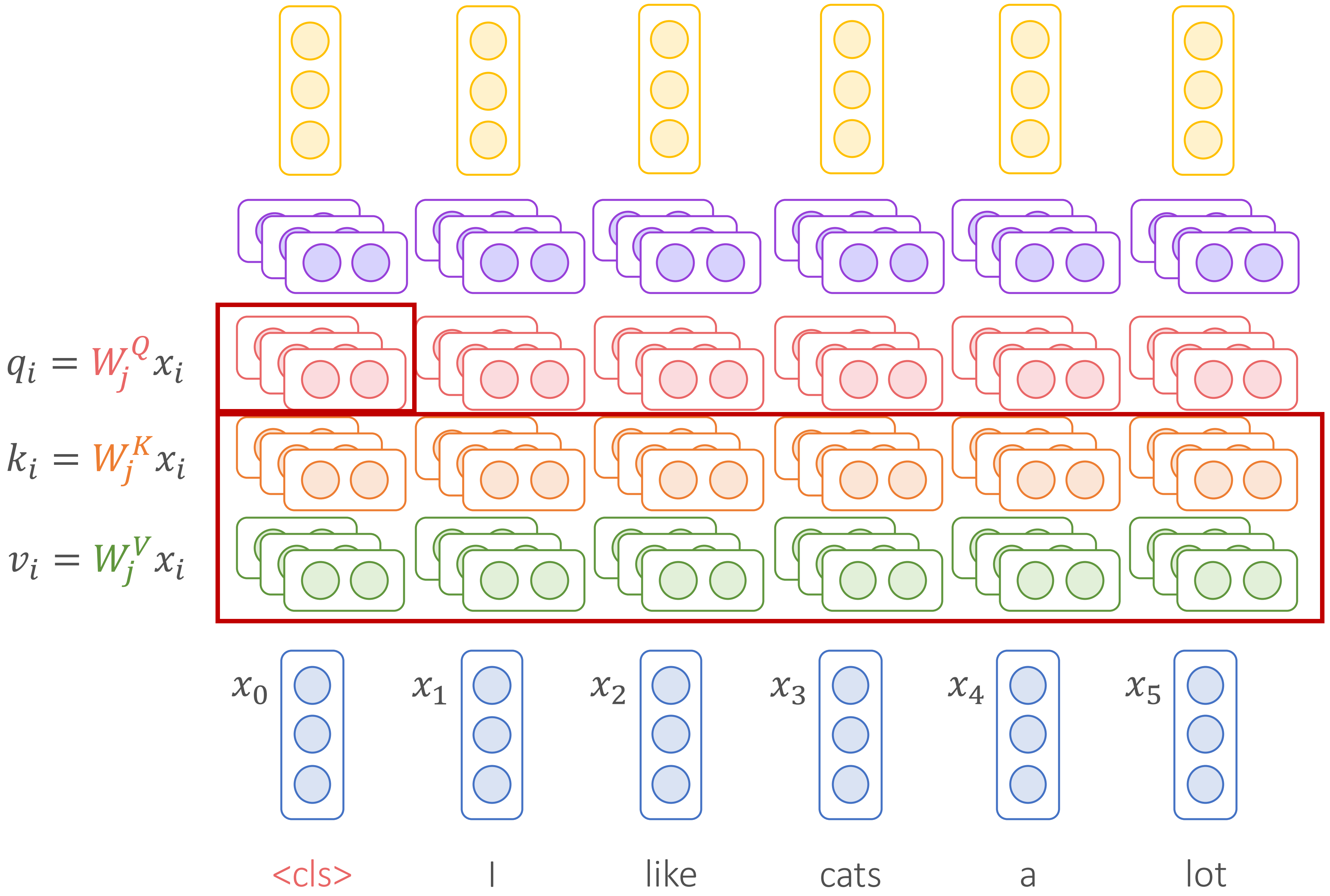
Transformer as Sentence-Level Encoder



Transformer as Sentence-Level Encoder



Transformer as Sentence-Level Encoder

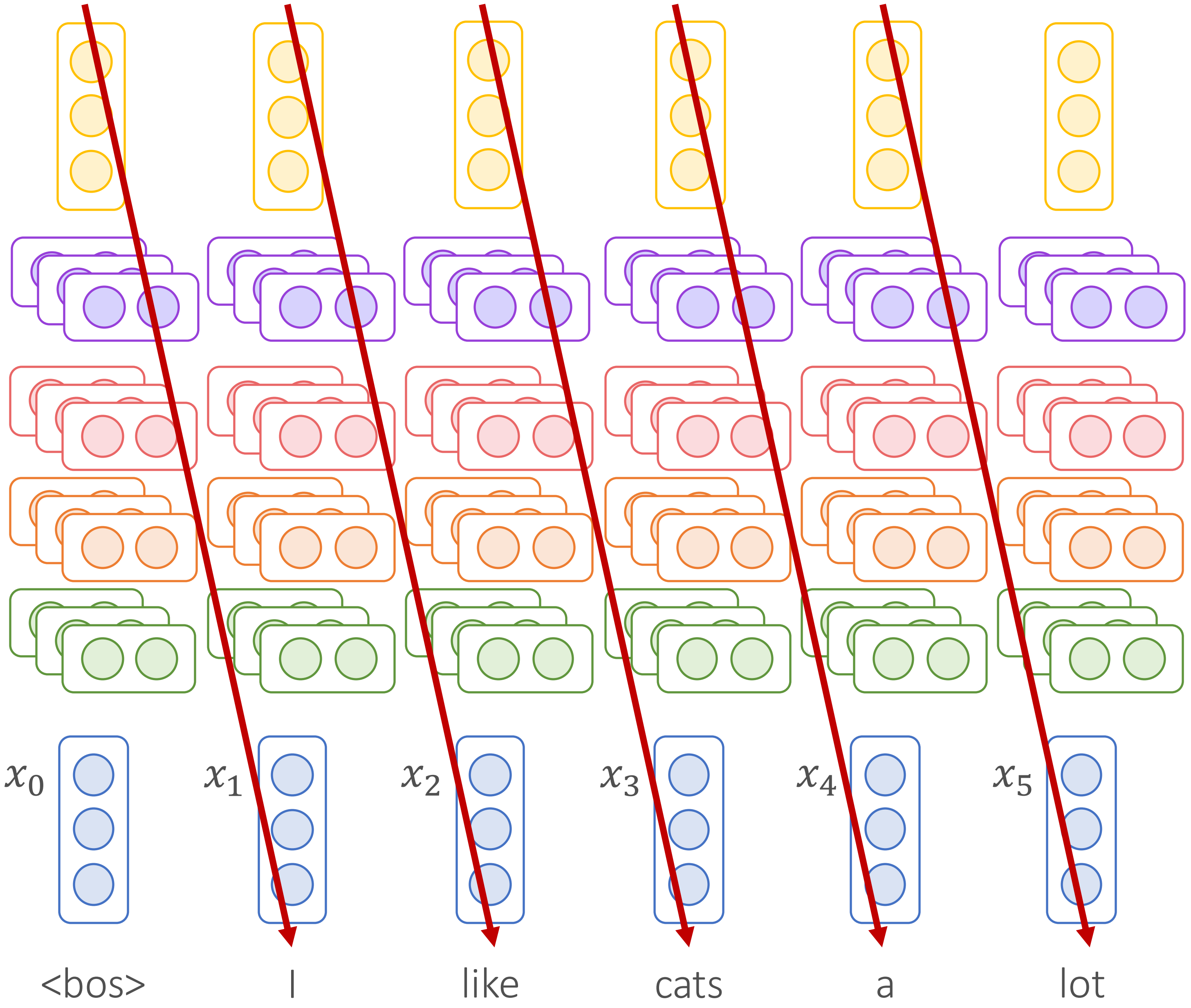


Transformer as Decoder?

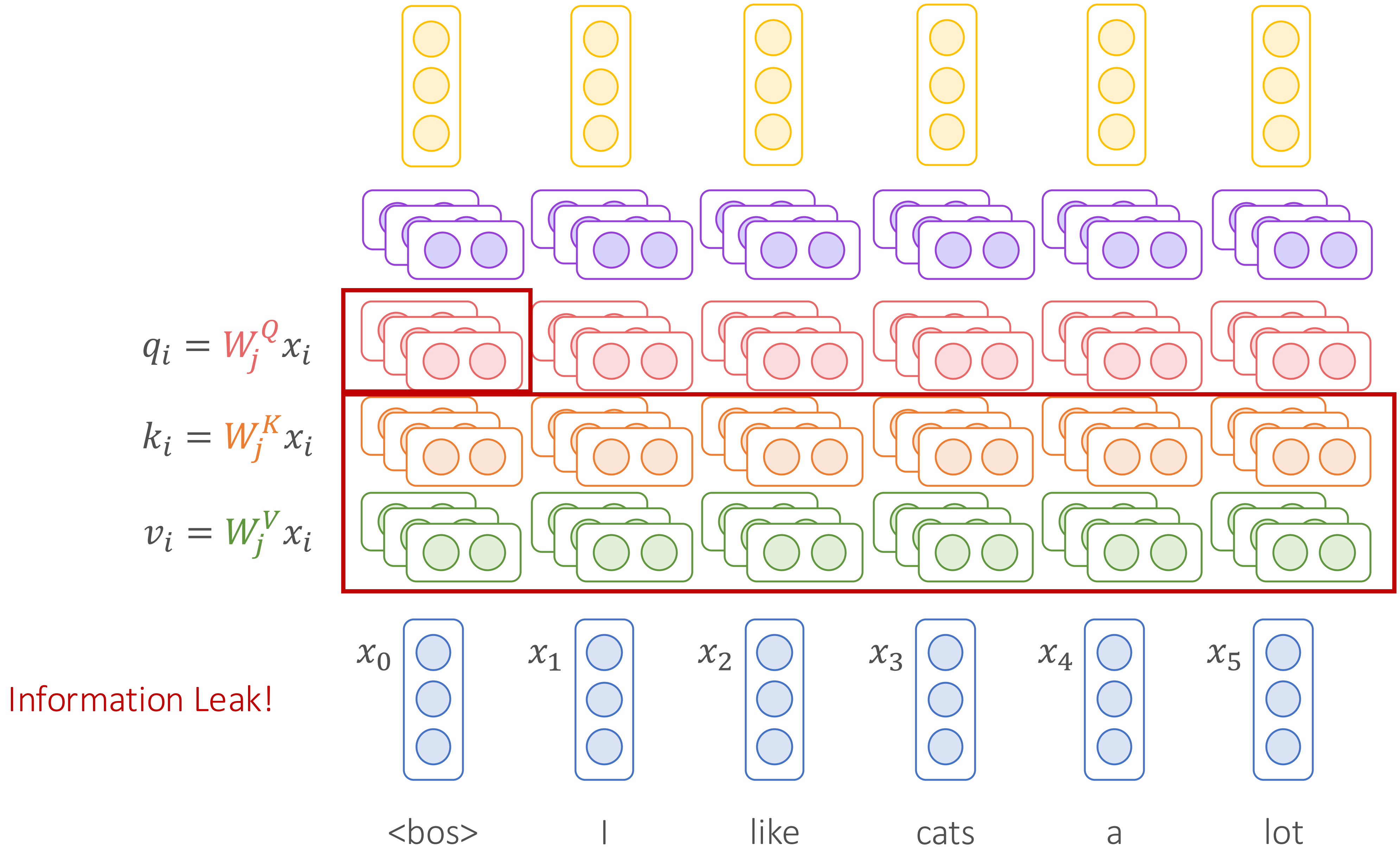
$q_i = W_j^Q x_i$

$k_i = W_j^K x_i$

$v_i = W_j^V x_i$



Transformer as Decoder?



Lecture Plan

- Transformers
  - Attention
  - Self-Attention
  - Transformer Encoder
  - Positional Encoding