CSE 291 – Intro to Deep RL
Attention and Language Modeling

Prithviraj Ammanabrolu

Thanks to Lillian Weng, Abby Morgan, Jay Alamar, and Brandon Amos.
Some slides were adapted form their courses / blogs.

Model Free vs Model Based RL

- Model-Free RL
  - No model
  - Learn value function (and/or policy) from experience
- Model-Based RL
  - Learn a model from experience
  - Plan value function (and/or policy) from model

Sample Based Planning

- A simple but powerful approach to planning
- Use the model only to generate samples
- Sample experience from model

  $S_{t+1} \sim T_\eta(S_{t+1} \mid S_t , A_t)$

  $R_{t+1} = R_\eta(R_{t+1} \mid S_t , A_t)$

- Apply model-free RL to samples, e.g.: Monte-Carlo control Sarsa Q-learning
- Sample-based planning methods are often more efficient

What is a Model?

- A model M is a representation of an MDP <S, A,T, R>, parametrized by η
- We will assume state space S and action space A are known
- So a model $M = <T_\eta, R_\eta>$ represents state transitions $T_\eta \approx T$ and rewards $R\eta \approx R$

$$S_{t+1} \sim P\eta(S_{t+1} \mid S_t , A_t)$$

$$R_{t+1} = R\eta(R_{t+1} \mid S_t , A_t)$$

- Typically assume conditional independence between state transitions and rewards

$$P\,[S_{t+1}, R_{t+1} \mid S_t , A_t] = P[S_{t+1} \mid S_t , A_t]\, P[R_{t+1} \mid S_t , A_t]$$

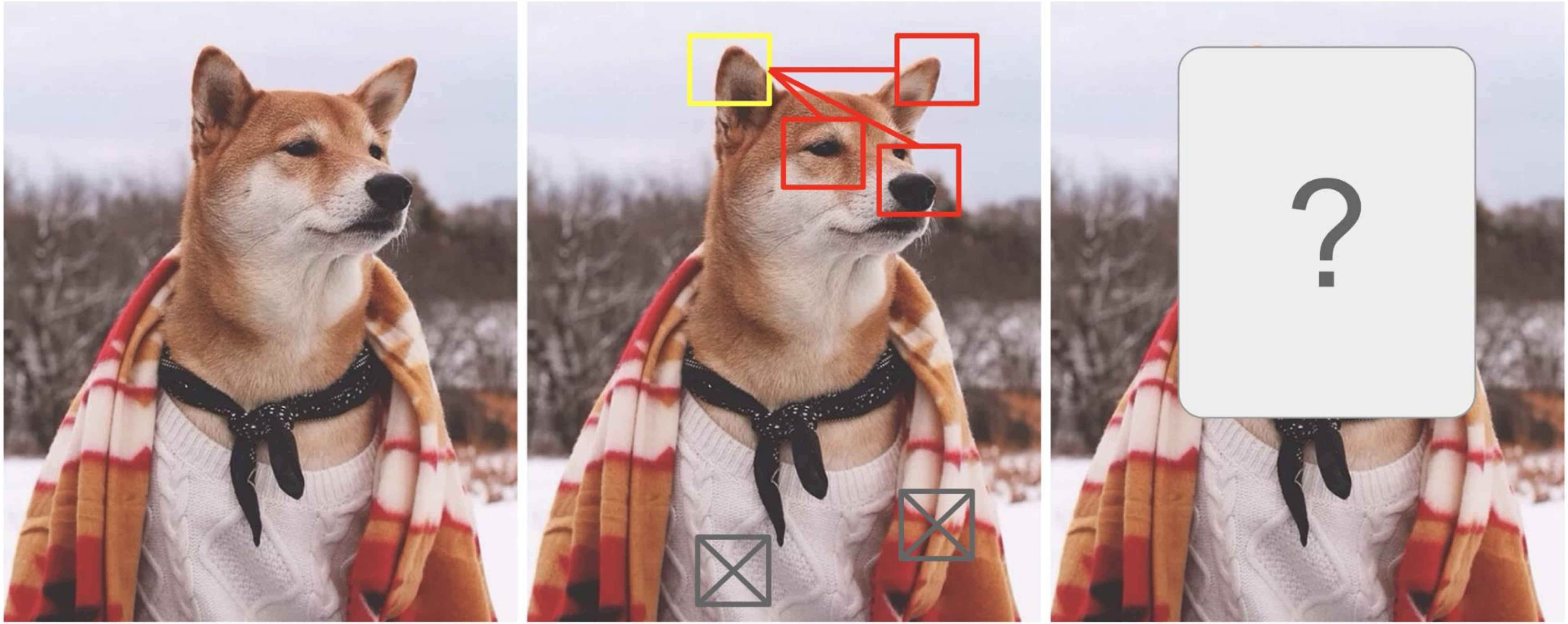
Pros and Cons of MBRL

- Pros
  - Can do all the (self, un) supervised learning tricks to learn from large scale data
  - Can reason about uncertainty
- Cons
  - Need model of T first
  - Will build estimate of value from that
  - Two(+) sources of error

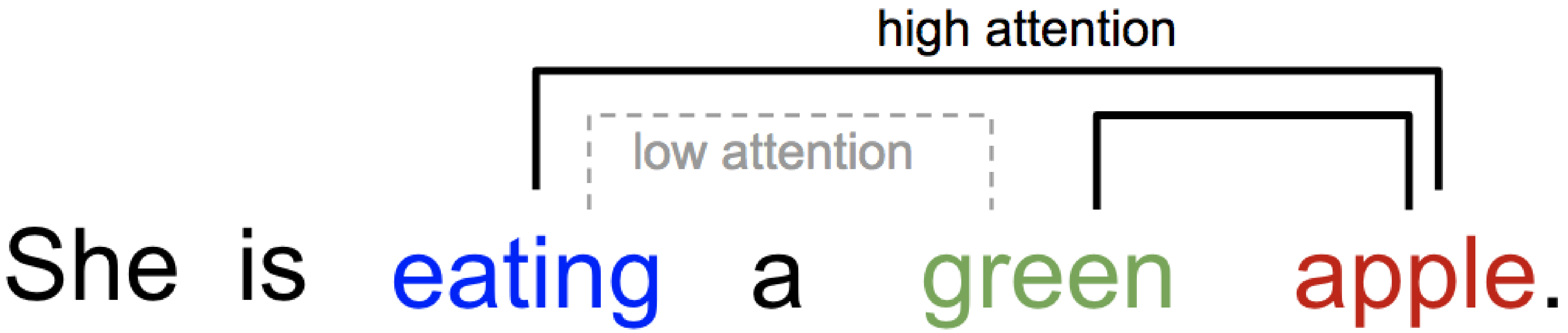
Simultaneous Bottlenecks of Deep RL

- The function approximator needs to be “good” for the task
- CNNs were great for Atari and then Go
- Why did they never work for language?

Pay Attention

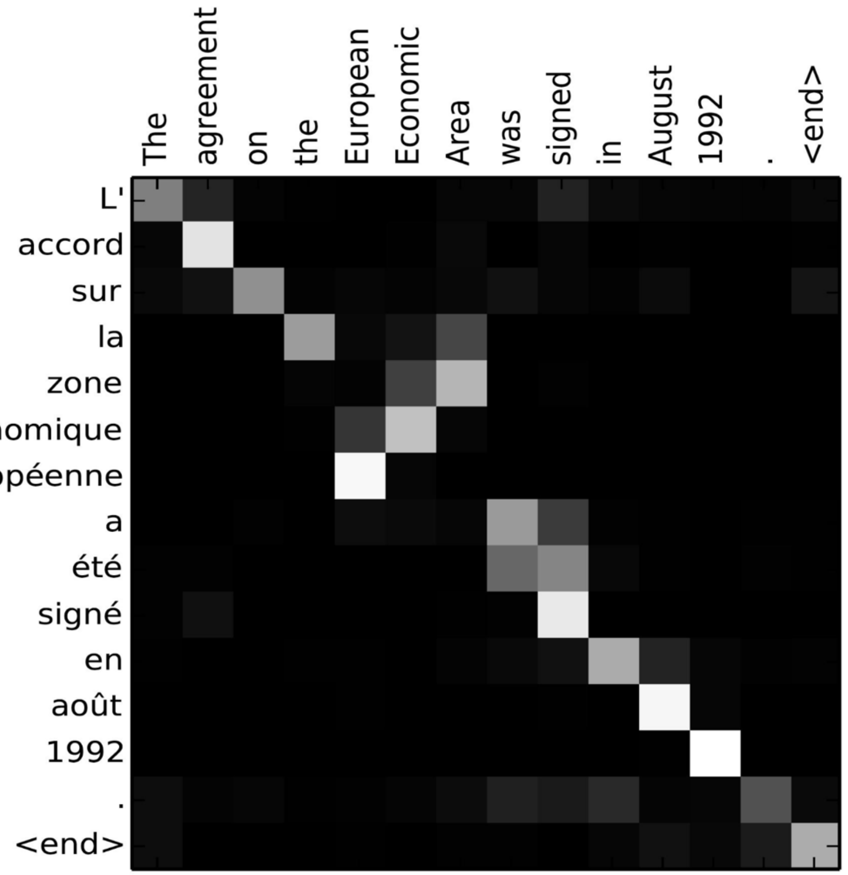


Pay Attention to your Words



Deep Learning Attention

- A vector of importance weights over an input sequence



Attention Alignment

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$
$$\mathbf{y} = [y_1, y_2, \ldots, y_m]$$

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i \quad \text{; Context vector for output } y_t$$

$$\alpha_{t,i} = \text{align}(y_t, x_i) \quad \text{; How well two words } y_t \text{ and } x_i \text{ are aligned.}$$

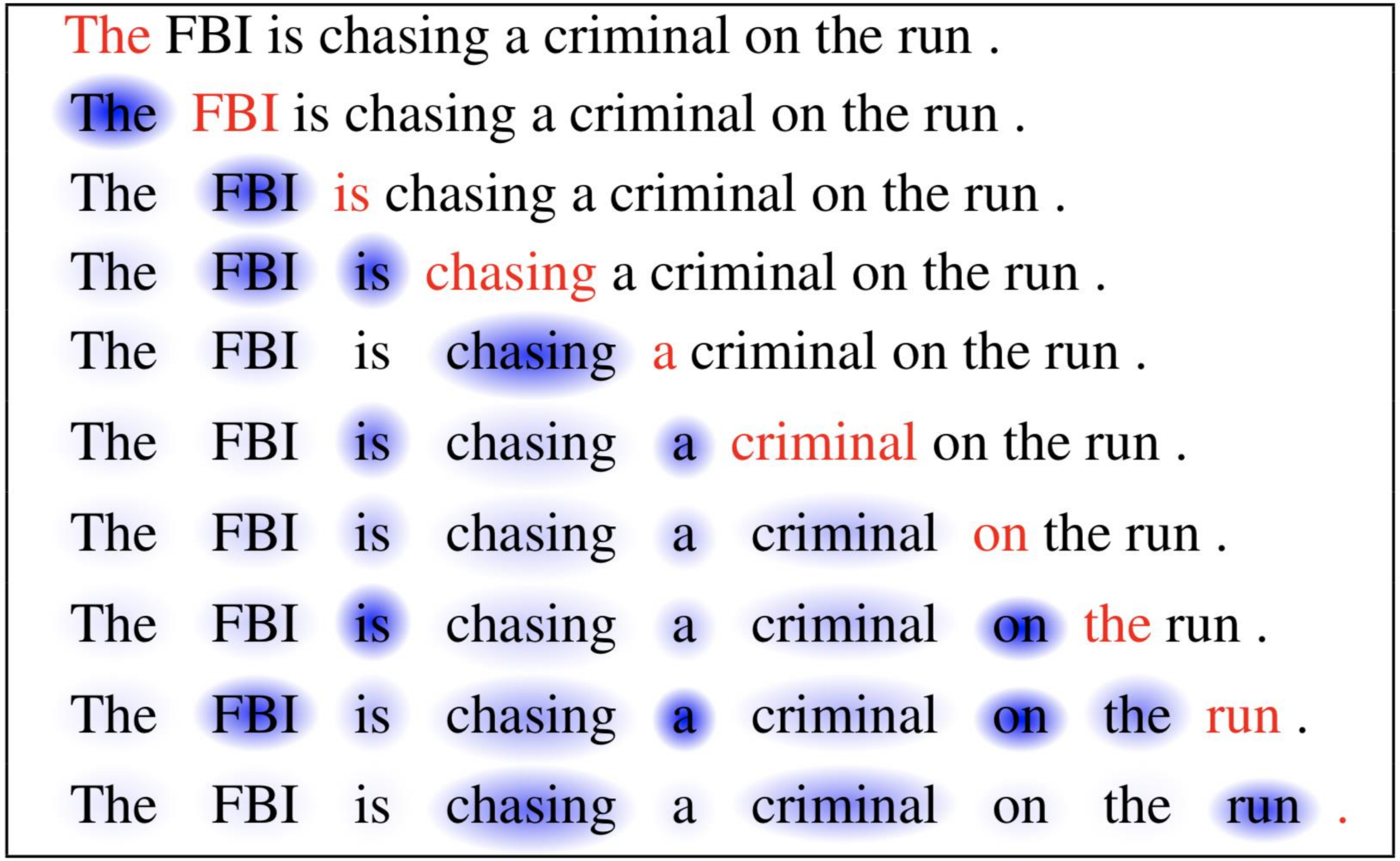
$$= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))} \quad \text{; Softmax of some predefined alignment score..}$$

Types of Attention (pre Vaswani)

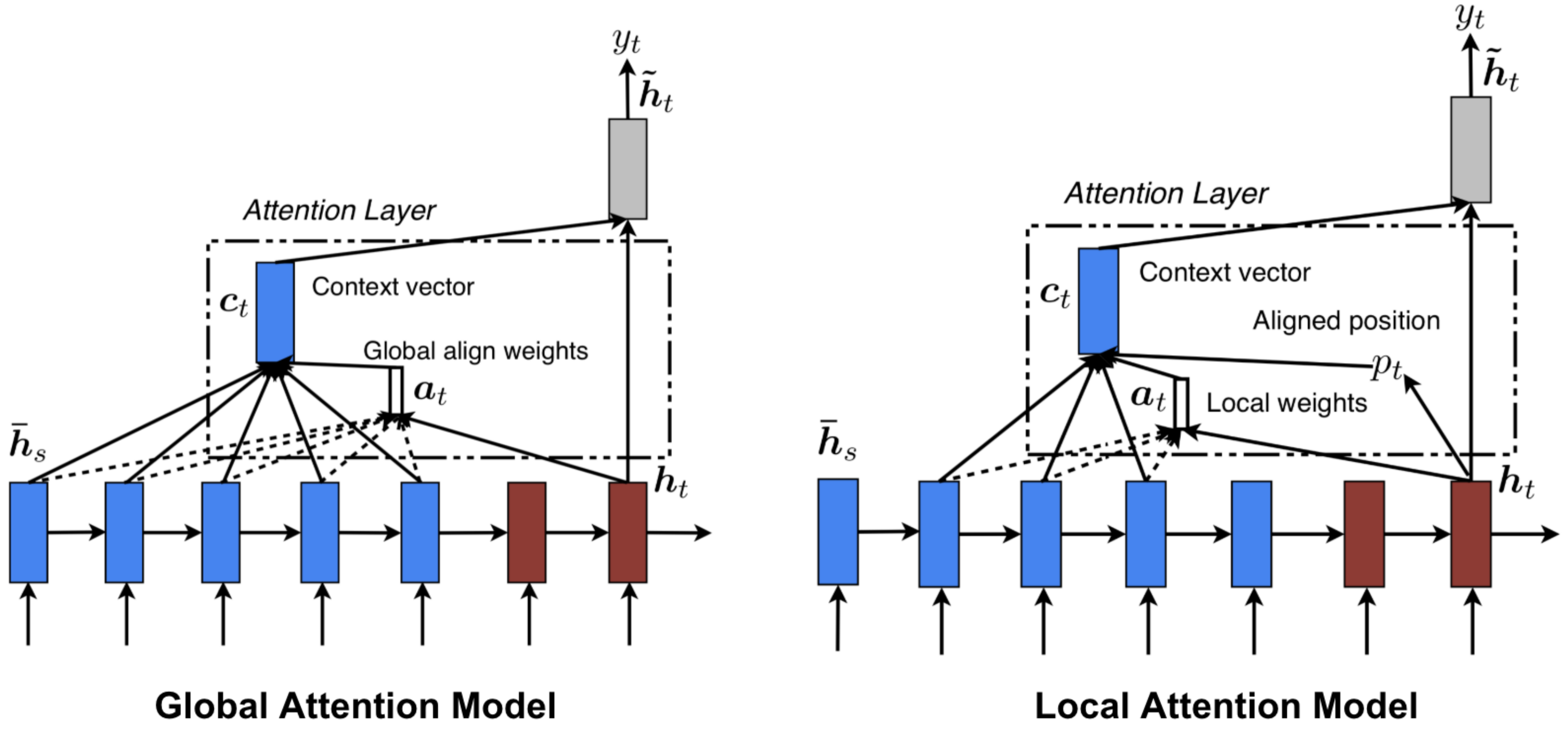
Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_{t-1}; \boldsymbol{h}_i])$	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

Self Attention

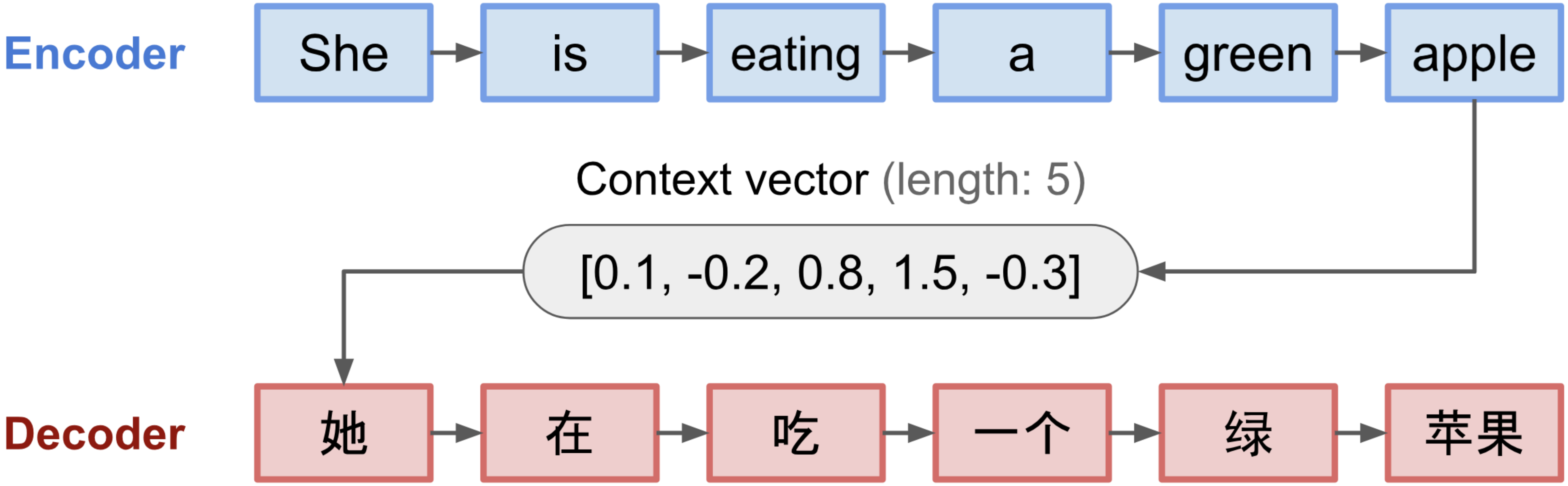
- Different parts of the same sequence attend to each other
- Previously it was all one sequence to another
- Proposed by Cheng et al 2016

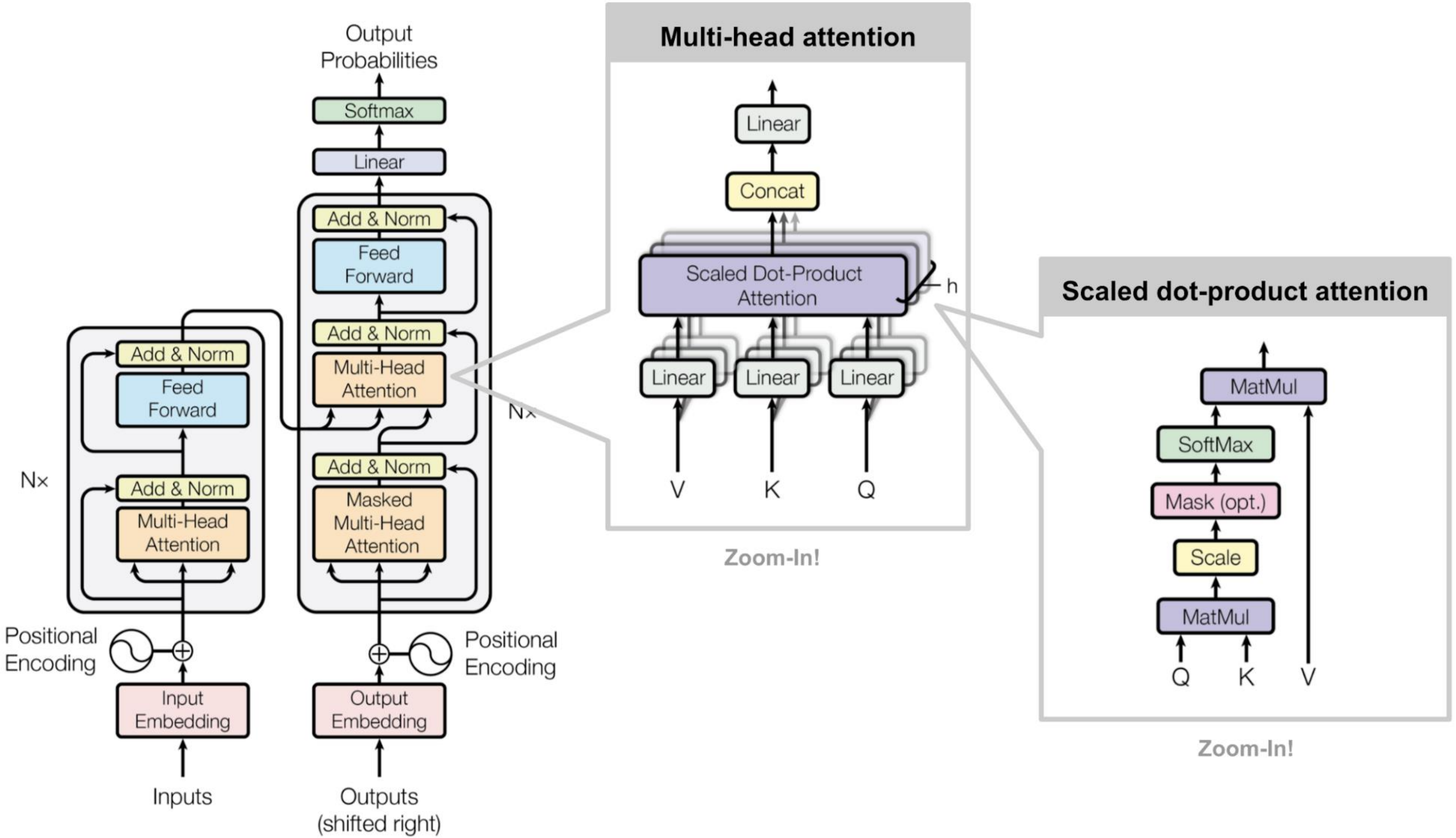


Global vs Local Attention (Luong et al. 2015)



Encoder Decoder RNN Failures



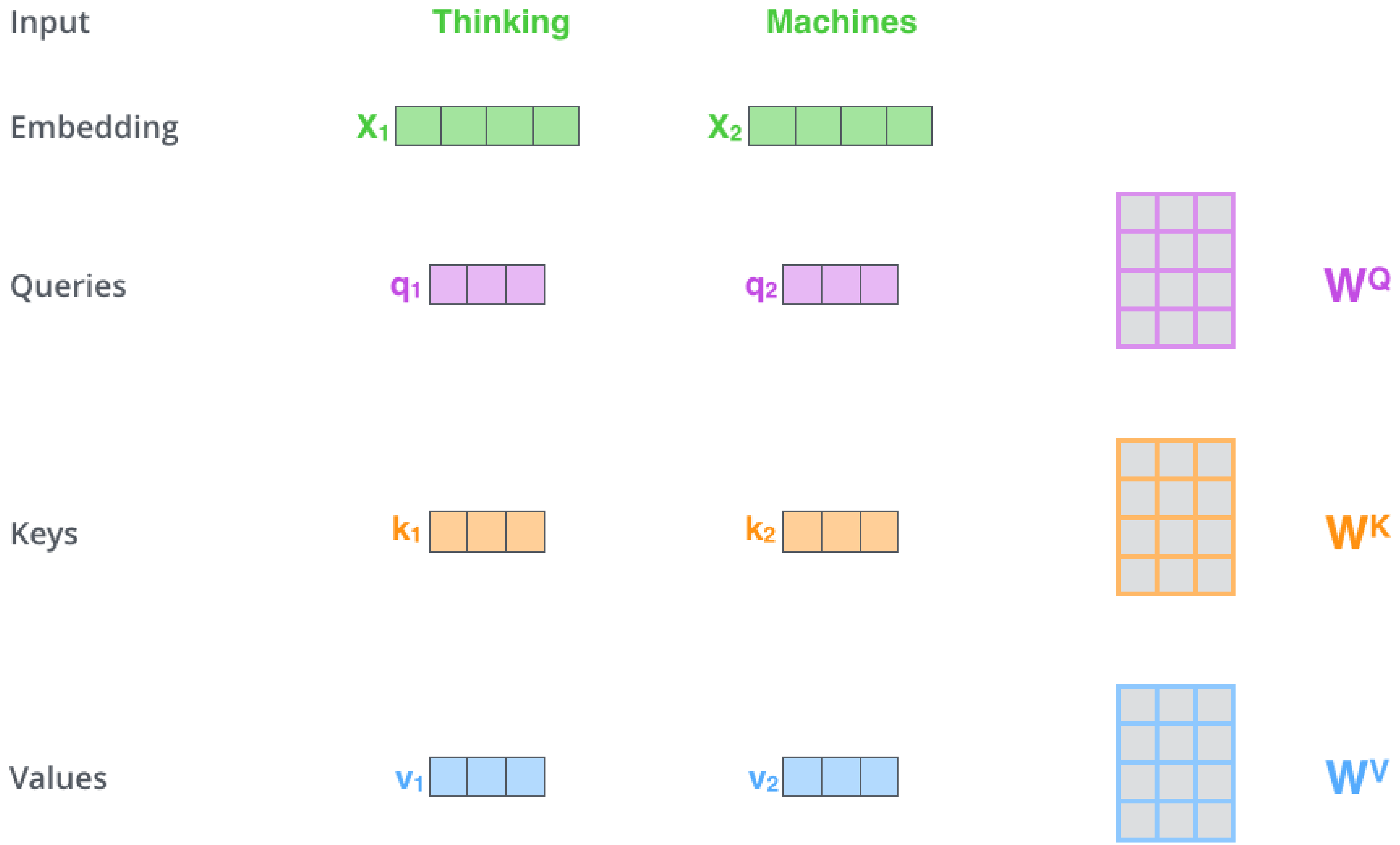


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Multi-head attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q
Zoom-In!
Scaled dot-product attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Zoom-In!

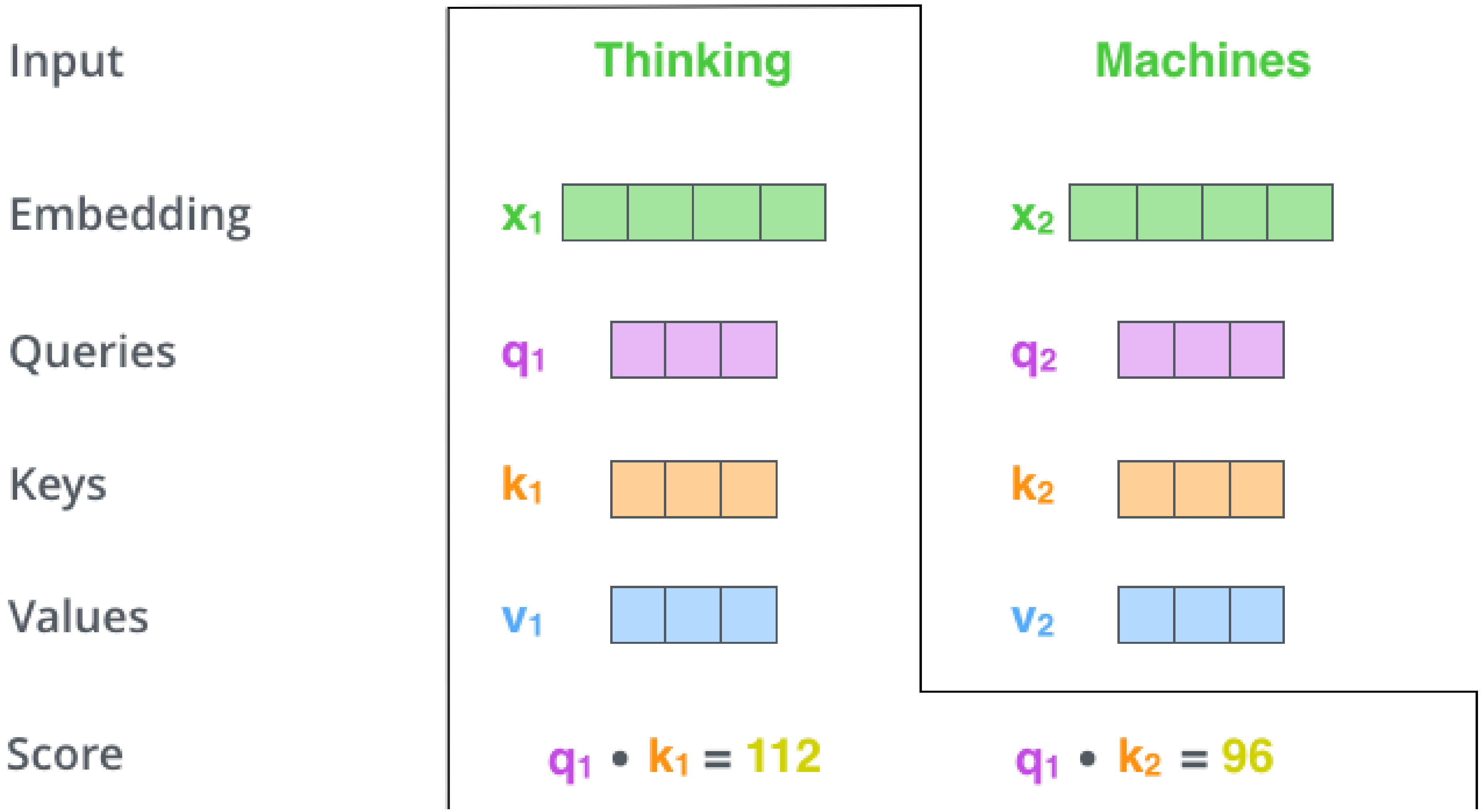
Scaled Dot Product Attention

For each input word we create a query, key, value vector

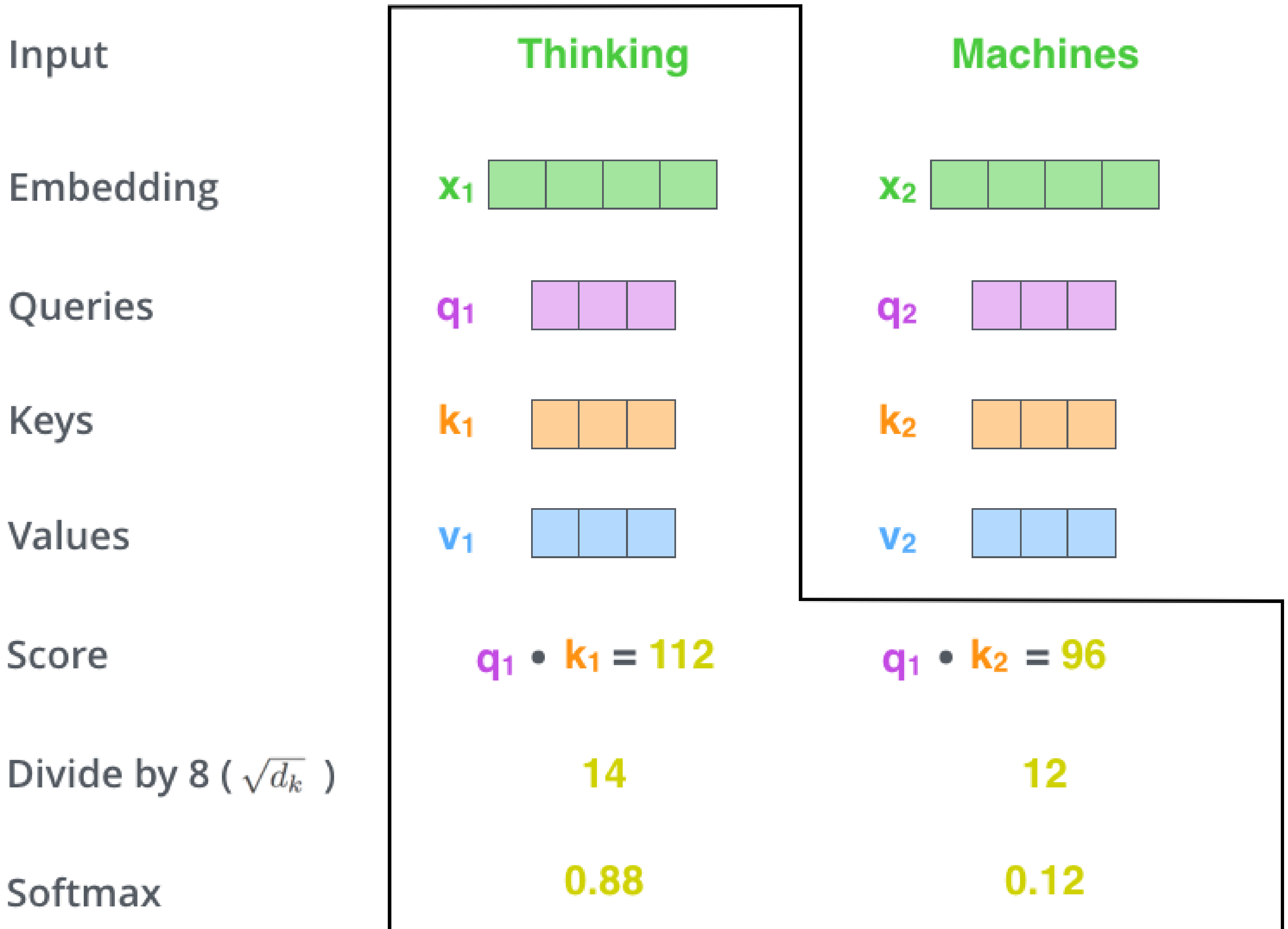
- Query: What are the things I am looking for?
- Key: What are the things that I have?
- Value: What are the things that I will communicate?



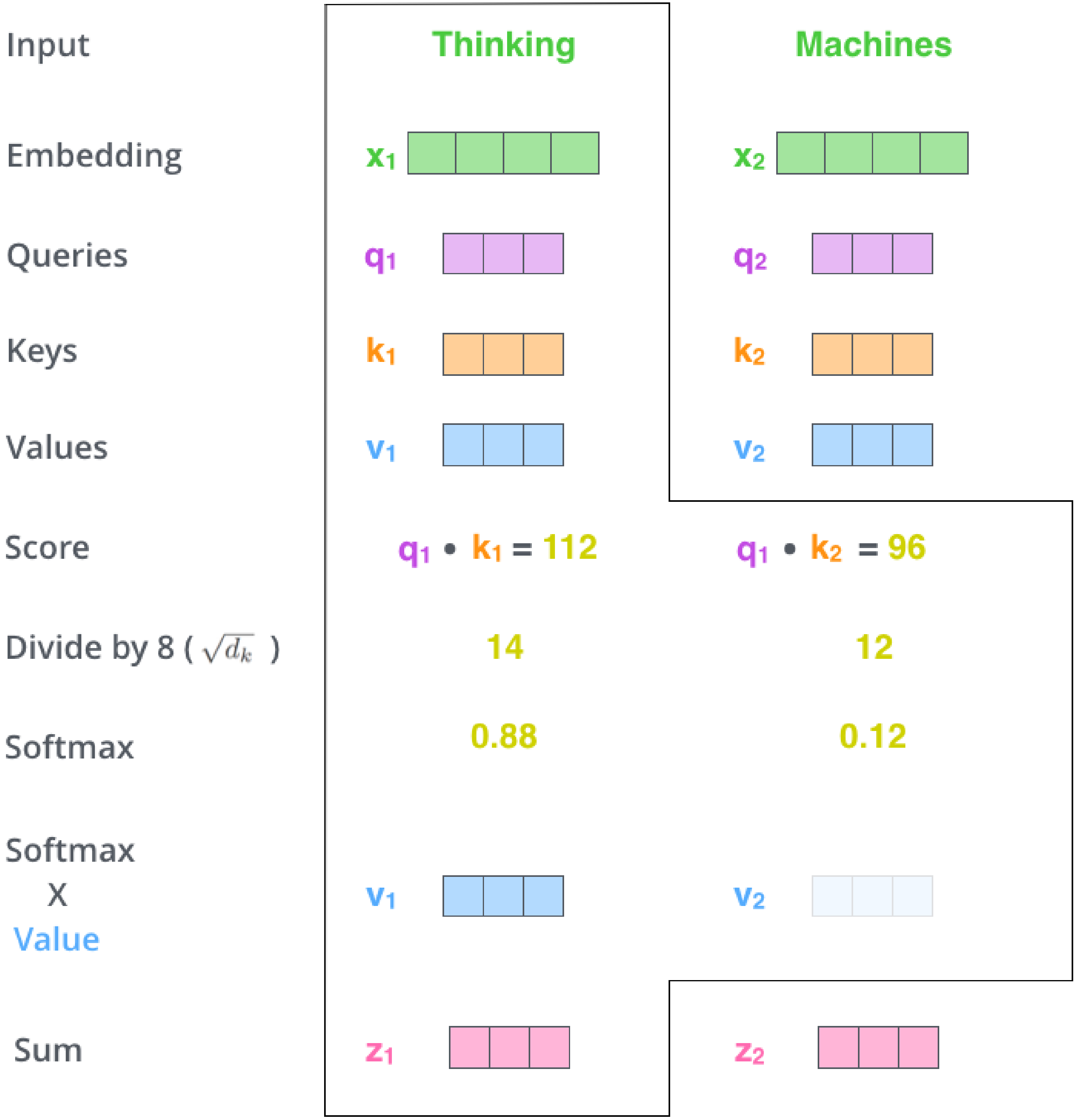
Input
Thinking
Machines
Embedding
X_1
X_2
Queries
q_1
q_2
W^Q
Keys
k_1
k_2
W^K
Values
v_1
v_2
W^V



Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96

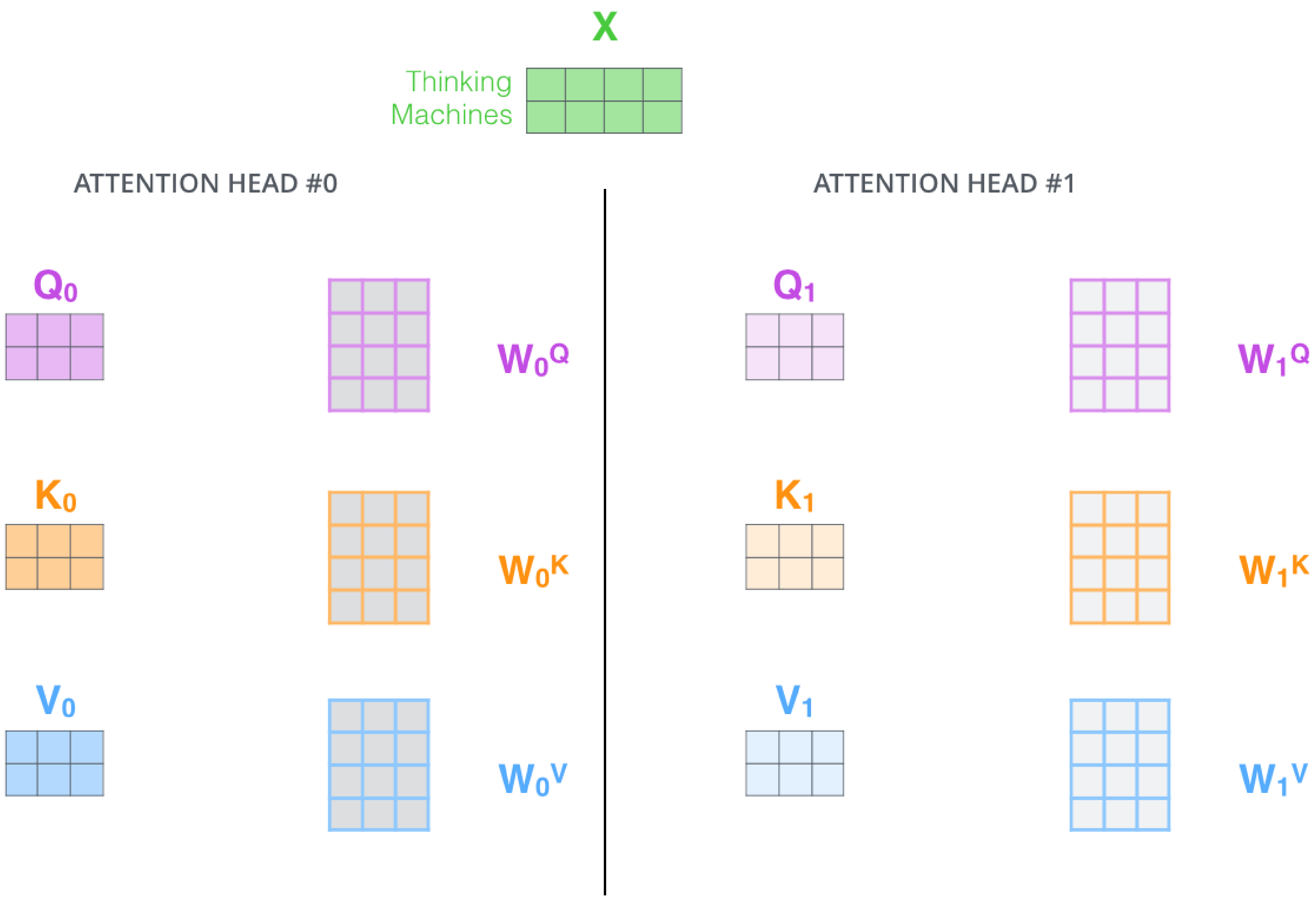


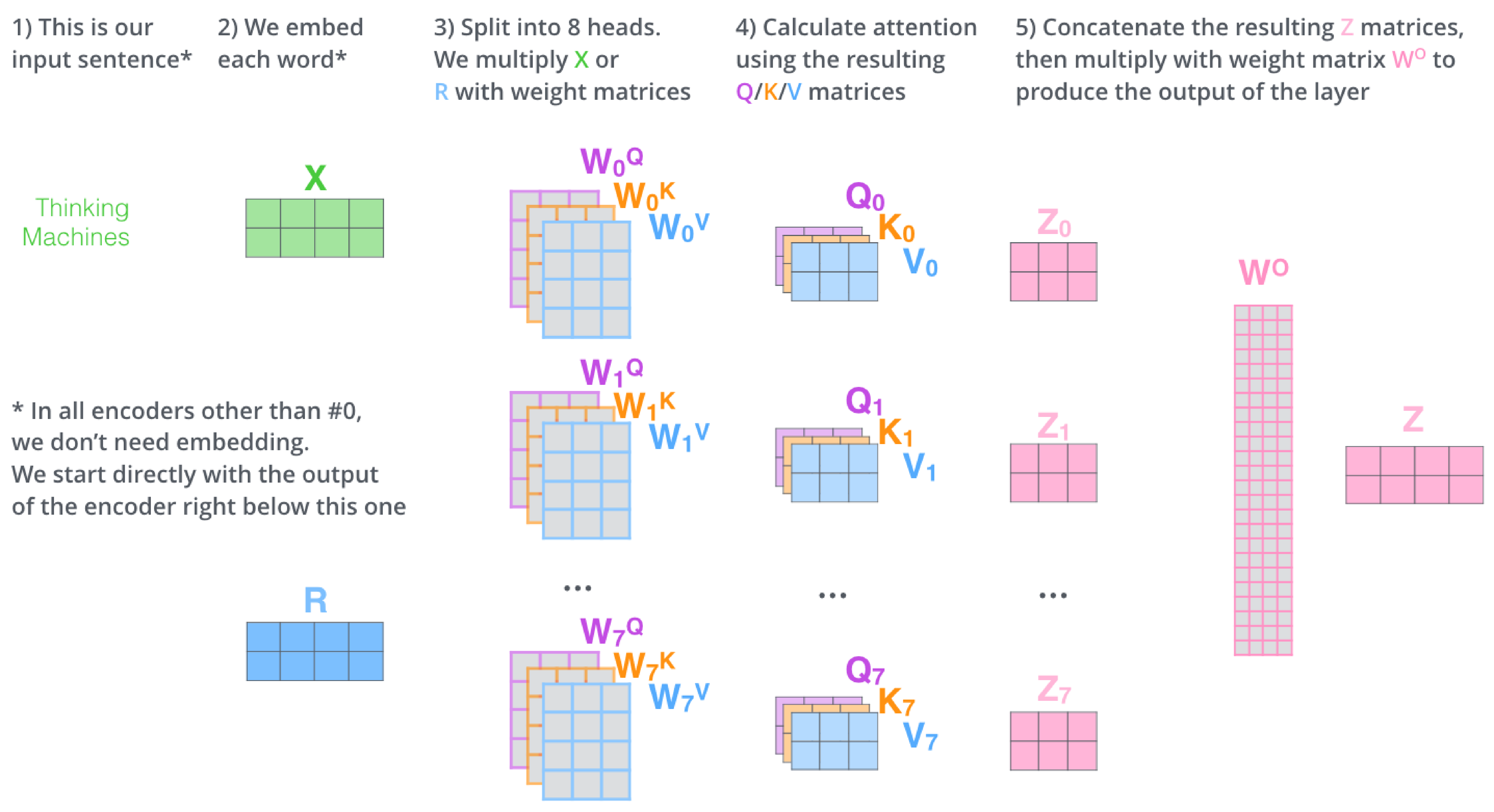
Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \cdot k_1 = 112$
$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12



Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax X Value
v1
v2
Sum
z1
z2

Multi-Head Attention

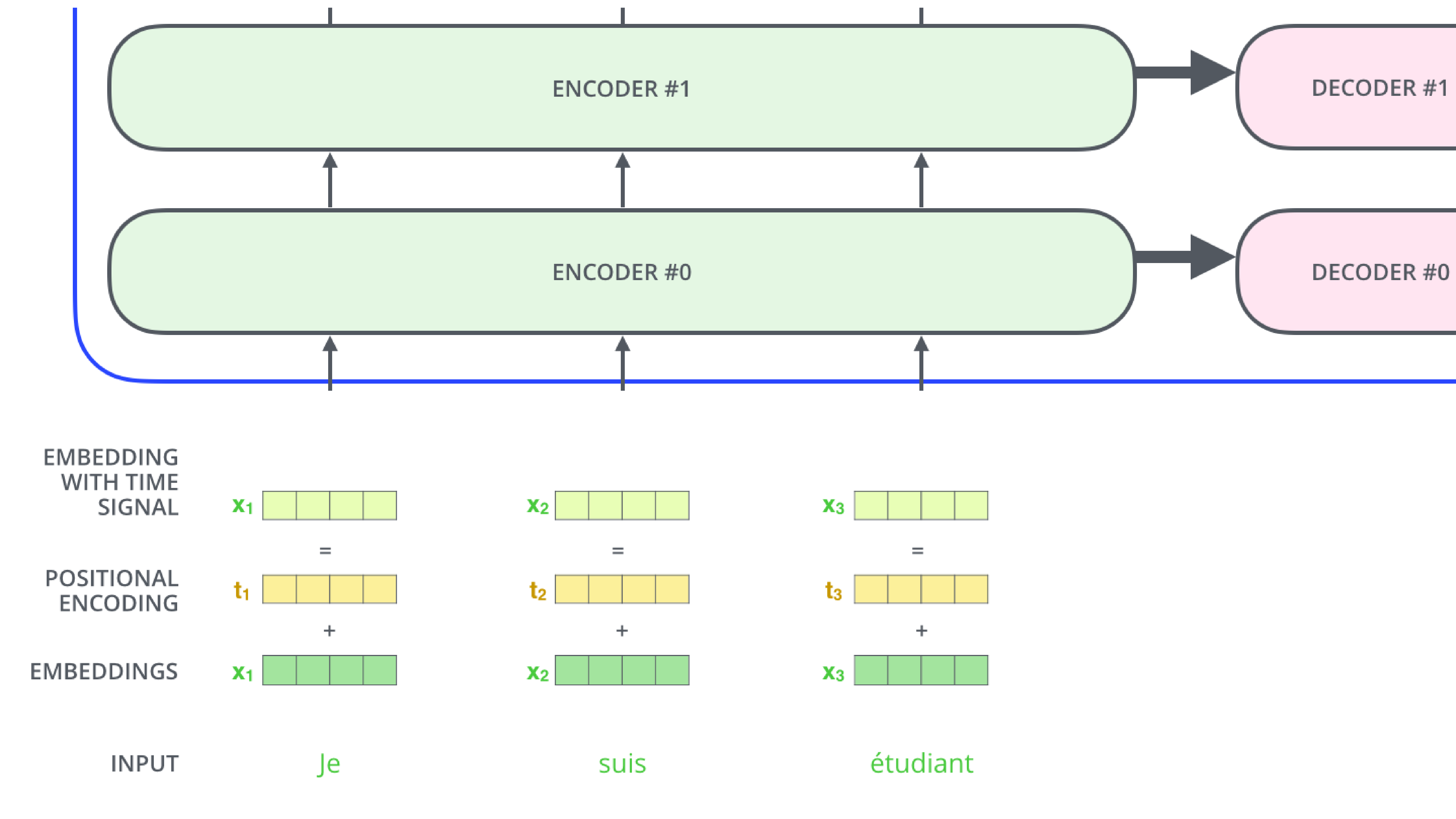



1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W_0^Q
W_0^K
W_0^V
W_1^Q
W_1^K
W_1^V
W_7^Q
W_7^K
W_7^V
Q_0
K_0
V_0
Q_1
K_1
V_1
Q_7
K_7
V_7
Z_0
Z_1
Z_7
W^O
Z

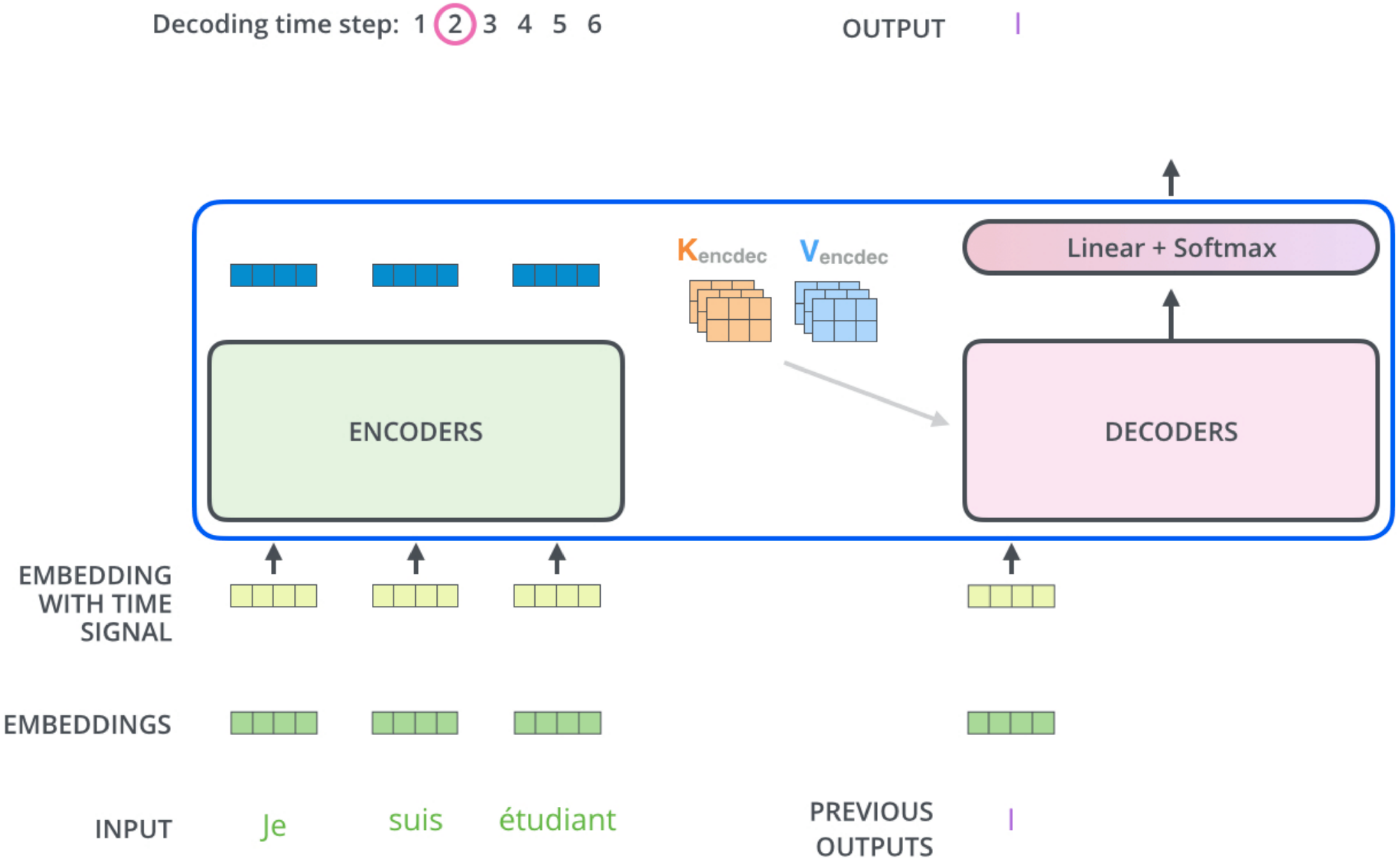
Position Encoding and Tokenizers

Transformer attention as seen so far is position invariant

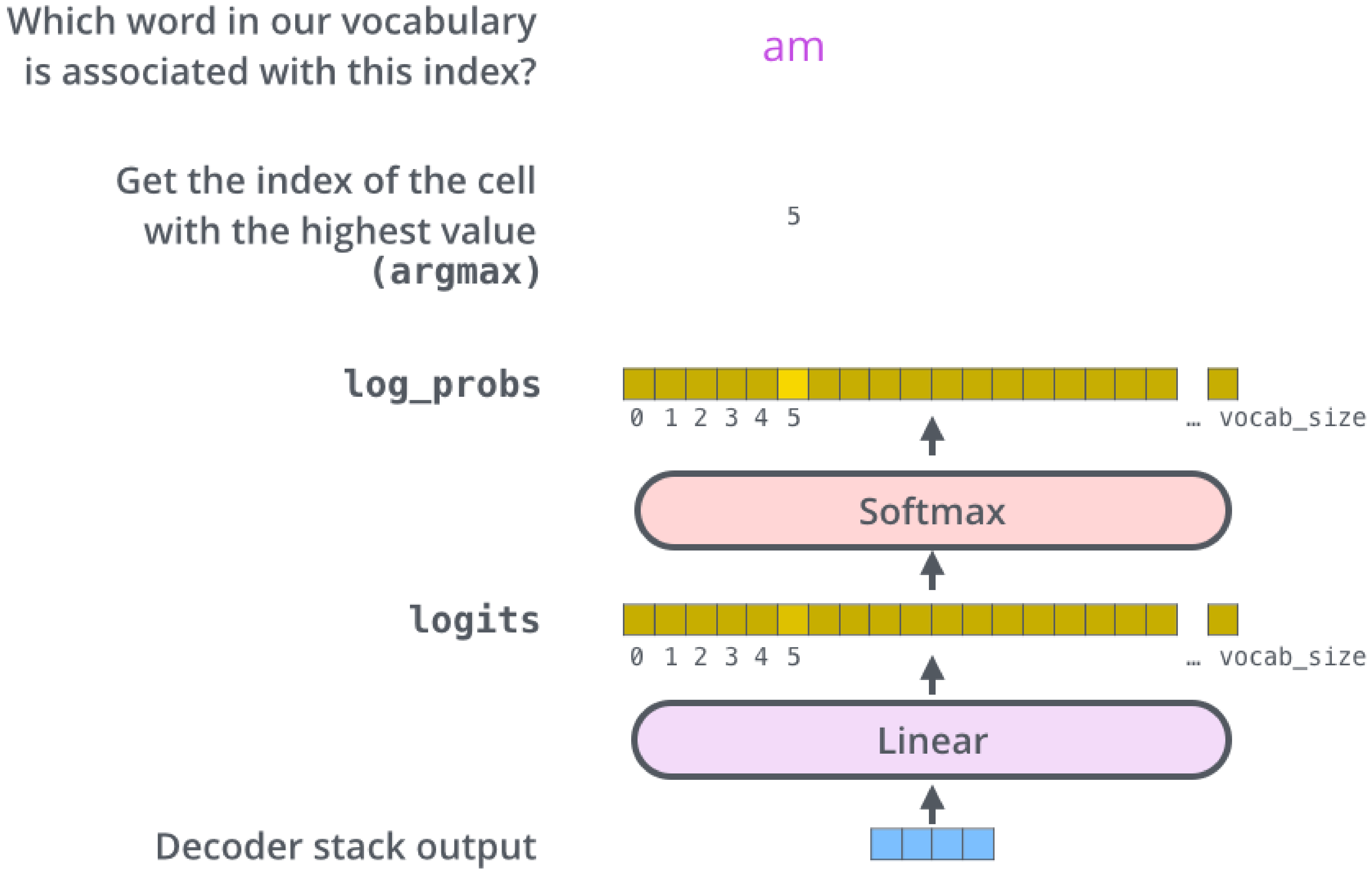
- Position of a word in a sentence matters, how to encode this?
- How to deal with out of vocabulary words? i.e. how to split the input sequence



ENCODER #1
DECODER #1
ENCODER #0
DECODER #0
EMBEDDING WITH TIME SIGNAL
x_1
x_2
x_3
=
POSITIONAL ENCODING
t_1
t_2
t_3
+
EMBEDDINGS
x_1
x_2
x_3
INPUT
Je
suis
étudiant



Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I


Which word in our vocabulary
is associated with this index?
am
Get the index of the cell
with the highest value
(argmax)
5
log_probs
0 1 2 3 4 5
… vocab_size
Softmax
logits
0 1 2 3 4 5
… vocab_size
Linear
Decoder stack output

Tokenization

Word Level

Deep Learning → Deep Learning

Character Level

Deep Learning → D e e p L e a r n i n g

Subword Level

Deep Learning → De ep Learn ##ing

Tokenization

- Subword (Tiktoken Byte Pair Encoding BPE) is the industry standard
- Learned from a representative subset of data

Tokenization (the bane of my existence)

- Many problems you think are LLM limitations are actually (partially) tokenizer issues
- E.g. Is 9.9 greater than 9.11?

    9 . 9 and 9 . 11

    compare initial 9

    compare .

    compare 9 and 11, wait 11 is greater than 9

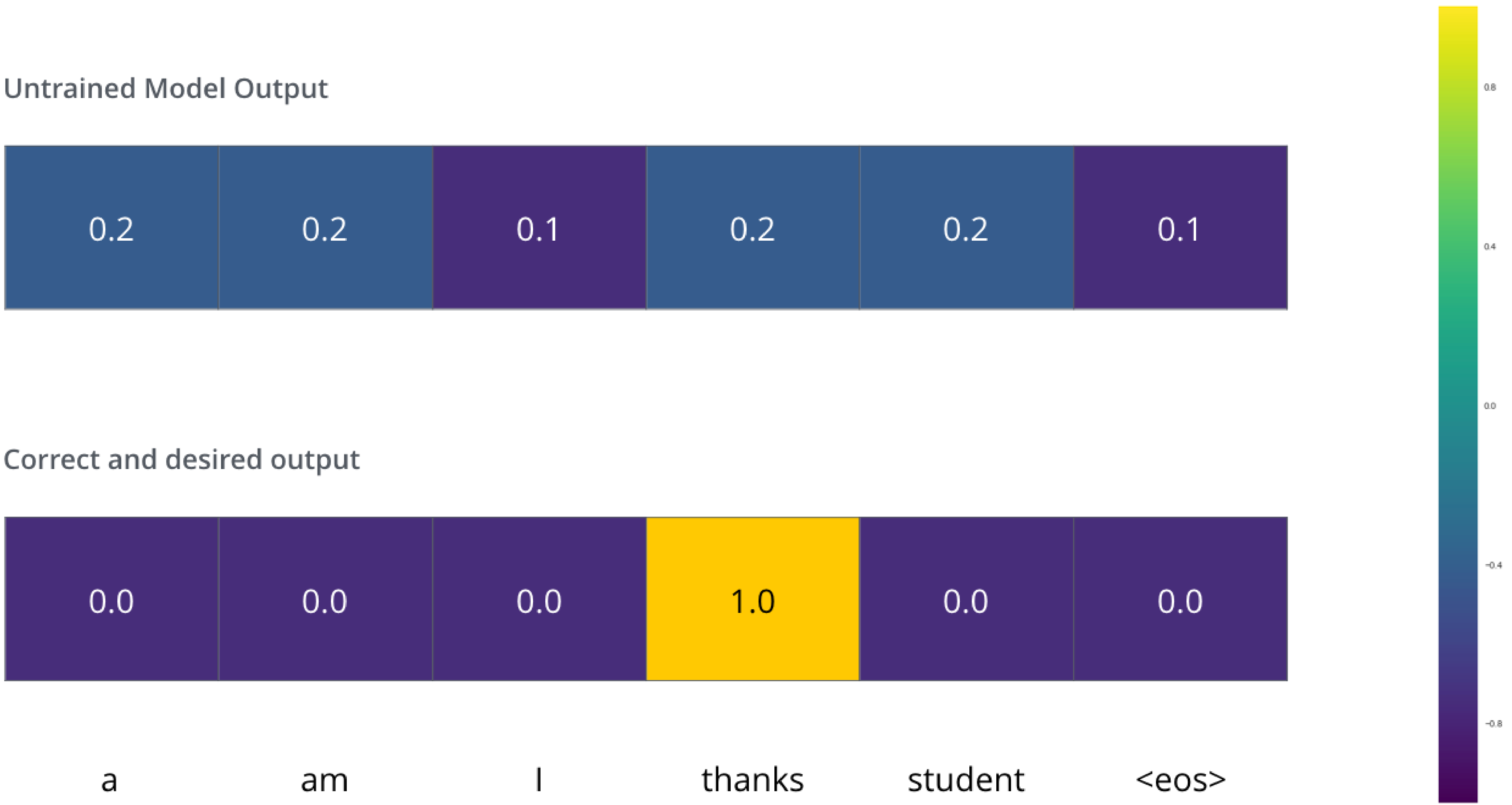
    so 9.9 is not greater than 9.11

How to train this network?

- Language Modeling! Looong history, will not cover here
- Many different forms of objectives

Two popular ones:

- Infill (used for BERT): This is the AI [MASK] Course.
- Next token Prediction (used for GPT): This is the AI Agents ____


Untrained Model Output
0.2
0.2
0.1
0.2
0.2
0.1
Correct and desired output
0.0
0.0
0.0
1.0
0.0
0.0
a
am
I
thanks
student
<eos>
0.8
0.4
0.0
-0.4
-0.8

Transformers

ENCODER ONLY

aka

auto-encoding models

TASKS

- Sentence classification
- Named entity recognition
- Extractive question-answering
- Masked language modeling

EXAMPLES

BERT, RoBERTa, distilBERT

DECODER ONLY

aka

auto-regressive models

TASKS

- Text generation
- Causal language modeling

EXAMPLES

GPT-2, GPT Neo, GPT-3

ENCODER-DECODER

aka

sequence-to-sequence models

TASKS

- Tran
- Summ on
- Gene stion-
  ans g

EXAMPLES

BART, T5, Marian

What does this mean for Deep RL?

- We have a neural net architecture that works well on language!
- We can probably use this as a function approximator for MDPs with language-based state-action spaces
- But how? What even is a language MDP?

In Class Activity

https://github.com/karpathy/minGPT

Pt 1: using mingpt/bpe.py find two *strategies* to generate sequences that look the same to the human eye but return two different BPE tokenid sequences

Pt 2: using mingpt/model.py find how to calculate total trainable parameters as a closed form equation for GPT-2 architecture. What inputs do you need? What changes for other architectures