

CSE 291 – AI Agents Attention and Language Modeling

Prithviraj Ammanabrolu

Thanks to Lillian Weng, Abby Morgan, Jay Alamar, and Brandon Amos.
Some slides were adapted from 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

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

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

- 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 $\langle S, A, T, R \rangle$, parametrized by η
- We will assume state space S and action space A are known
- So a model $M = \langle T_\eta, R_\eta \rangle$ represents state transitions $T_\eta \approx T$ and rewards $R_\eta \approx R$

$$S_{t+1} \sim P_\eta(S_{t+1} | S_t, A_t)$$

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

- Typically assume conditional independence between state transitions and rewards

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

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

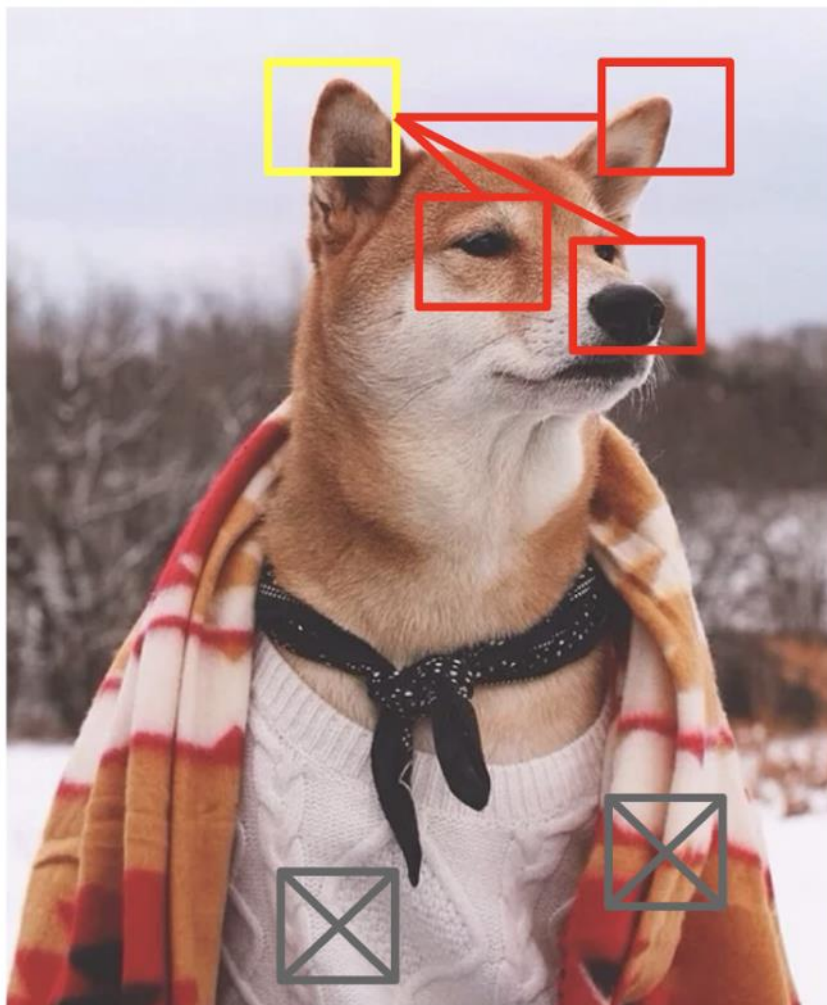
- Cons

- Need model of T first
- Will build estimate of value from that
- Two(+) sources of error

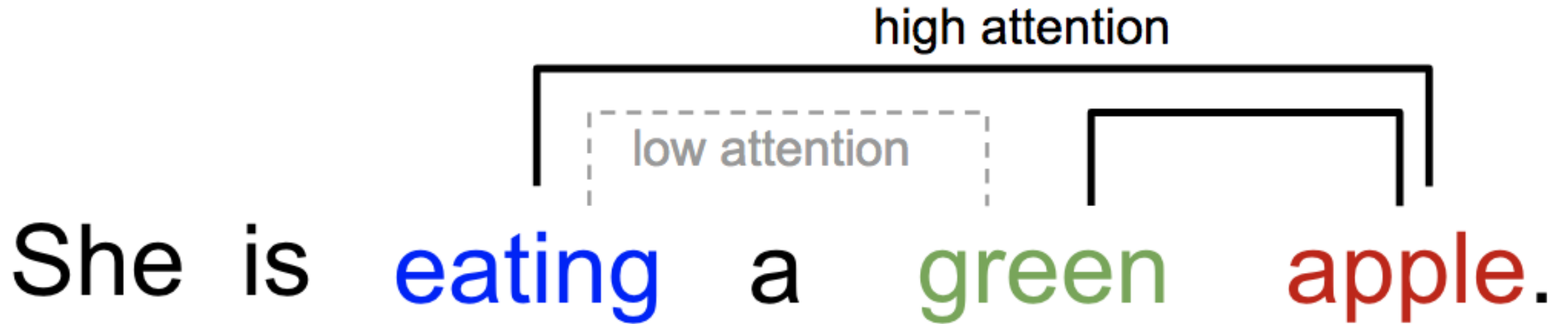
Simultaneous Bottlenecks of Deep RL

- 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



Attention Alignment

$$\mathbf{x} = [x_1, x_2, \dots, x_n]$$

$$\mathbf{y} = [y_1, y_2, \dots, y_m]$$

$$\mathbf{c}_t = \sum_{i=1}^n \alpha_{t,i} \mathbf{h}_i$$

; Context vector for output y_t

$$\alpha_{t,i} = \text{align}(y_t, x_i)$$

; How well two words y_t and x_i are aligned.

$$= \frac{\exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_i))}{\sum_{i'=1}^n \exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_{i'}))}$$

; Softmax of some predefined alignment score..

Types of Attention (pre Vaswani)

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \text{cosine}[\mathbf{s}_t, \mathbf{h}_i]$	<u>Graves2014</u>
Additive(*)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{s}_{t-1}; \mathbf{h}_i])$	<u>Bahdanau2015</u>
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \mathbf{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	<u>Luong2015</u>
General	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^\top \mathbf{W}_a \mathbf{h}_i$ where \mathbf{W}_a is a trainable weight matrix in the attention layer.	<u>Luong2015</u>
Dot-Product	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^\top \mathbf{h}_i$	<u>Luong2015</u>
Scaled Dot-Product(^)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \frac{\mathbf{s}_t^\top \mathbf{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	<u>Vaswani2017</u>

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

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

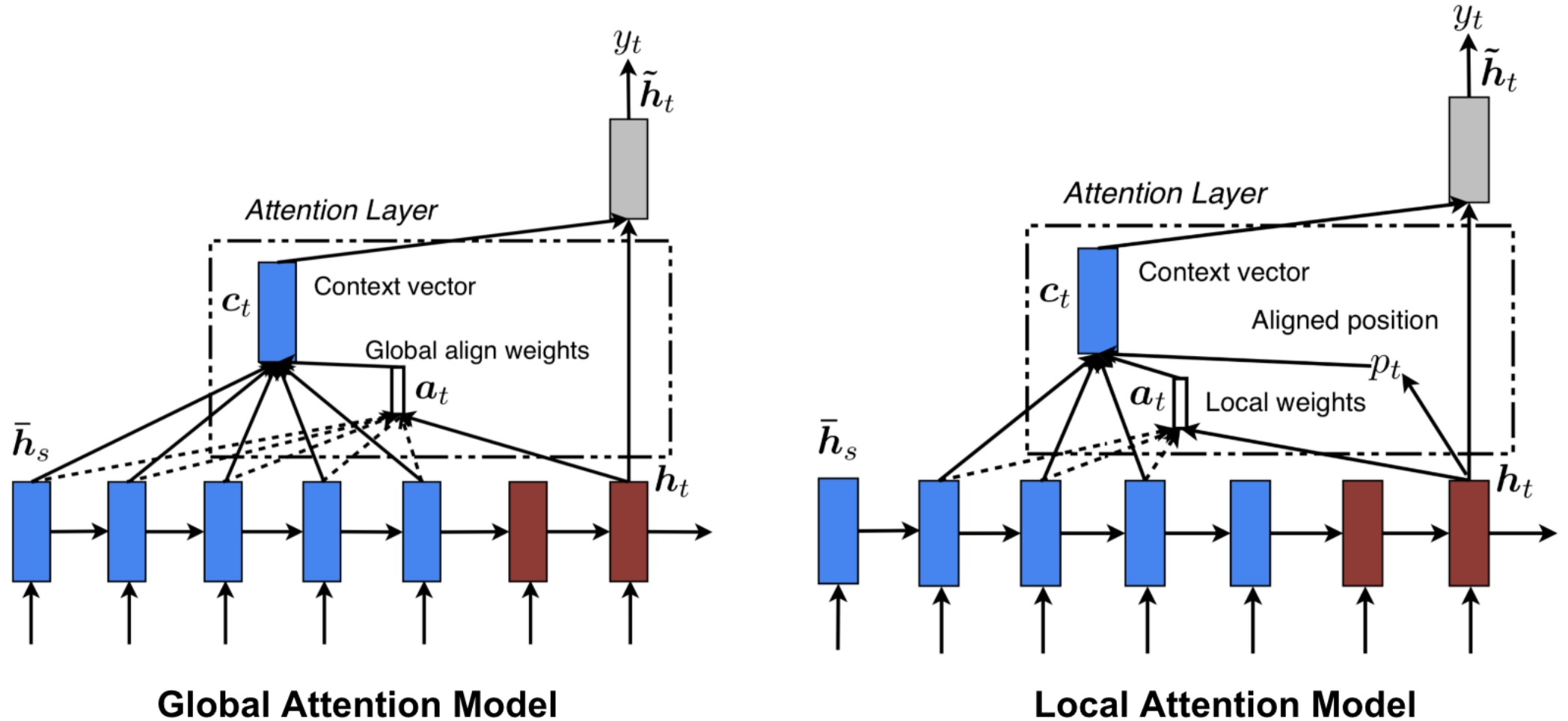
The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

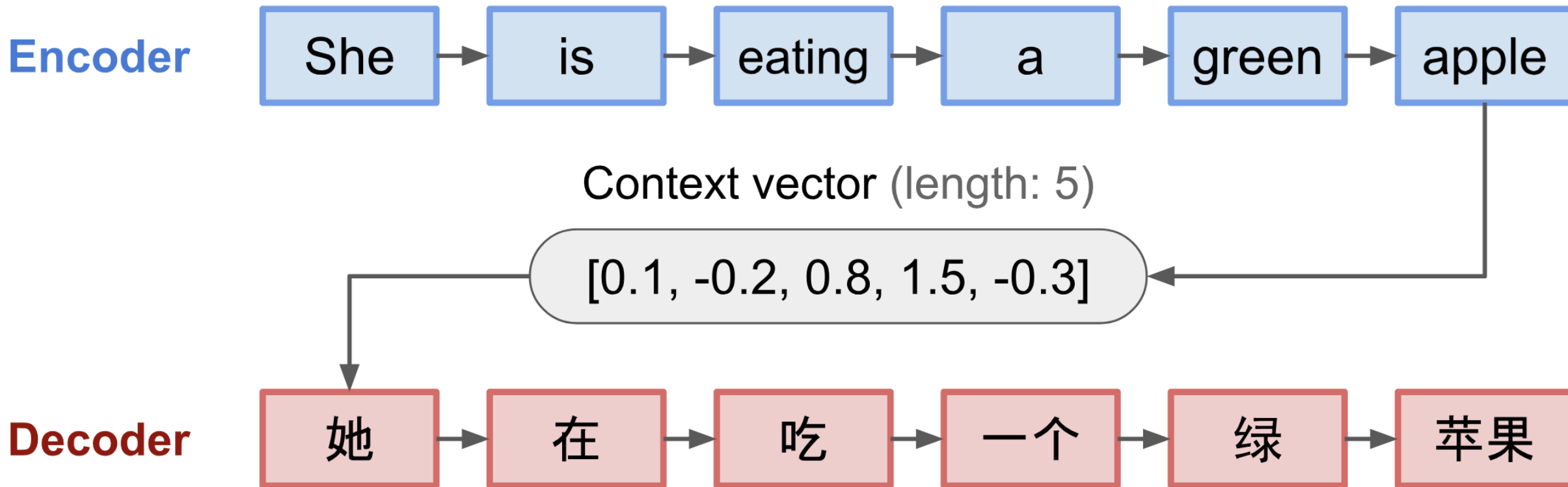
The FBI is chasing a criminal on the run .

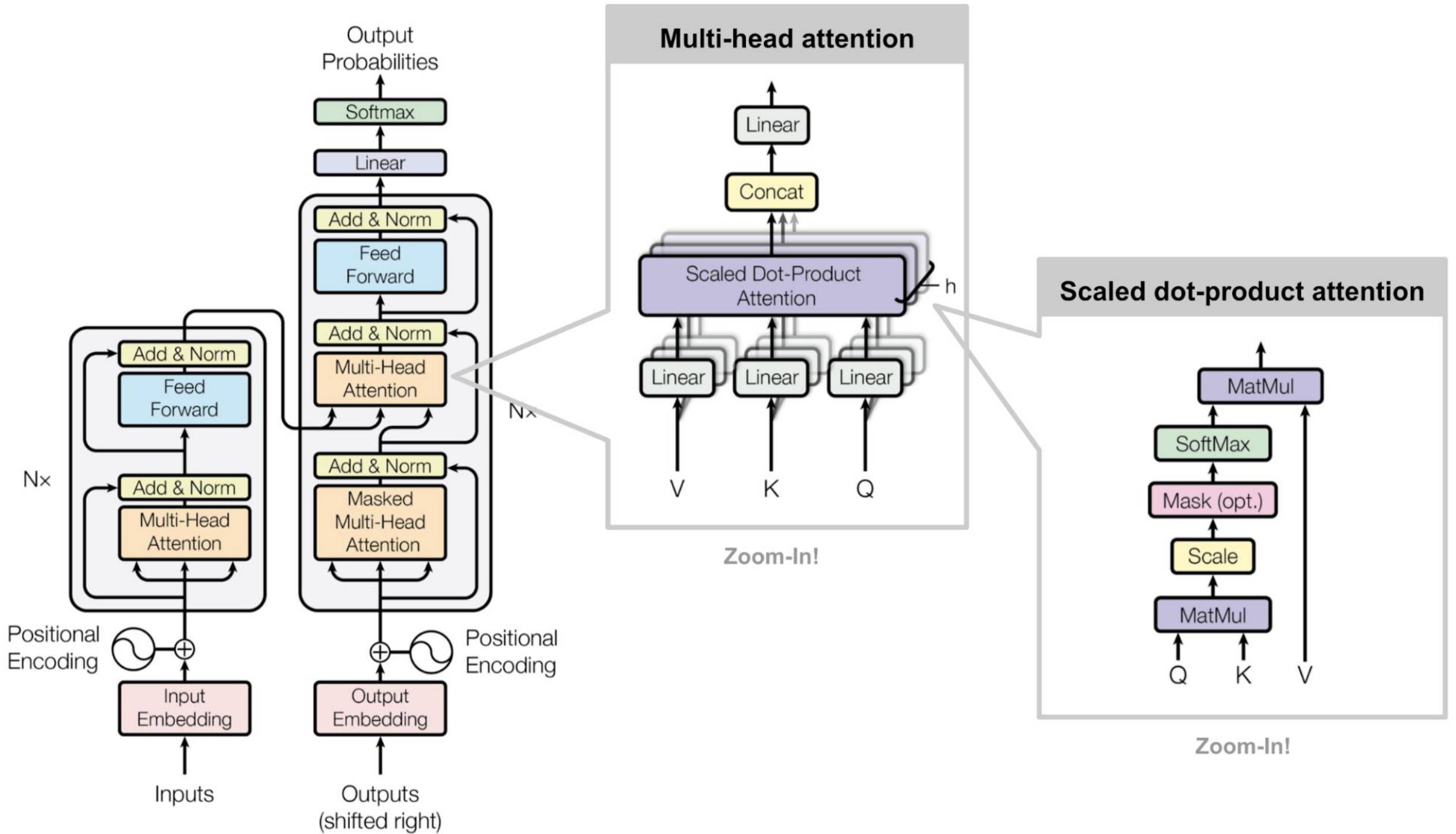
The FBI is chasing a criminal on the run .

Global vs Local Attention (Luong et al. 2015)



Encoder Decoder RNN Failures





Scaled Dot Product Attention

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

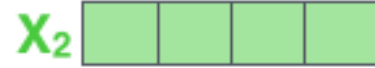
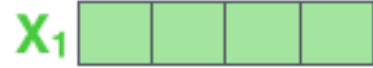
- Query: What are the things I am looking for?
- Key: What are the things that I have?
- Value: What are the things that I will communicate?

Input

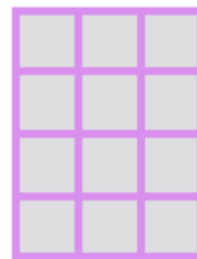
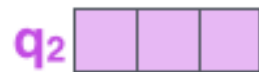
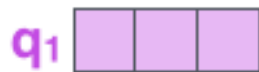
Thinking

Machines

Embedding

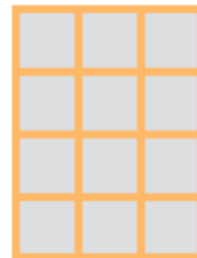
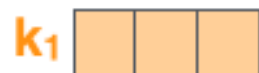


Queries



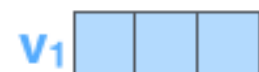
W^Q

Keys



W^K

Values



W^V

Input

Thinking

Machines

Embedding



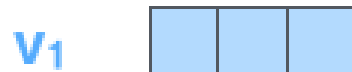
Queries



Keys



Values



Score

$q_1 \cdot k_1 = 112$

$q_1 \cdot k_2 = 96$

Input

Thinking

Machines

Embedding



Queries



Keys



Values



Score

$q_1 \cdot k_1 = 112$

$q_1 \cdot k_2 = 96$

Divide by 8 ($\sqrt{d_k}$)

14

12

Softmax

0.88

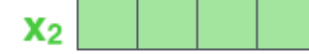
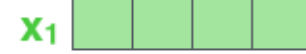
0.12

Input

Thinking

Machines

Embedding



Queries



Keys



Values



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

Softmax

X

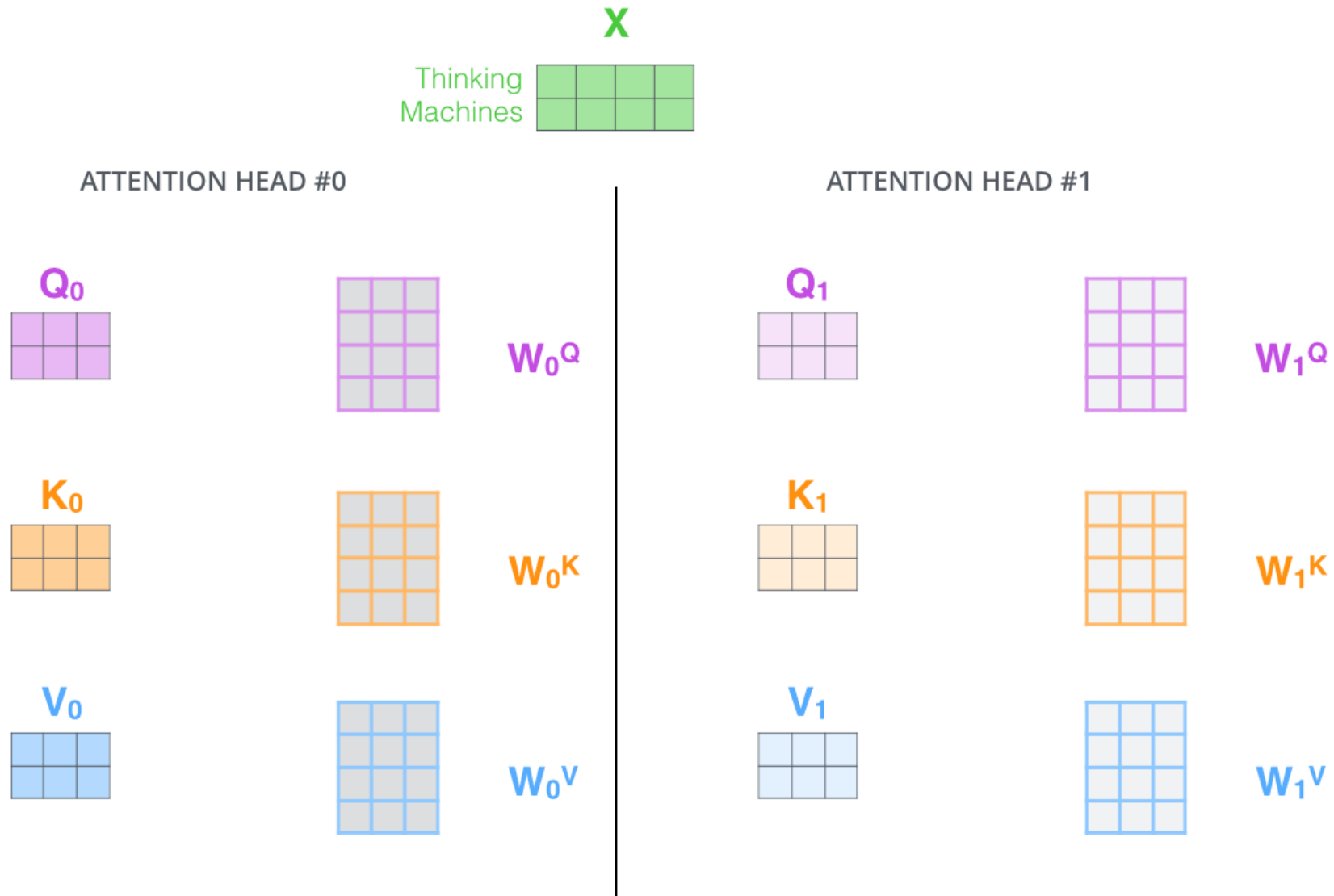
Value



Sum



Multi-Head Attention



1) This is our input sentence*

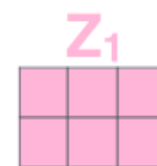
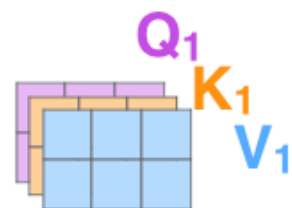
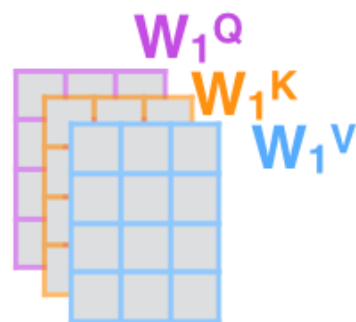
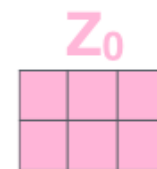
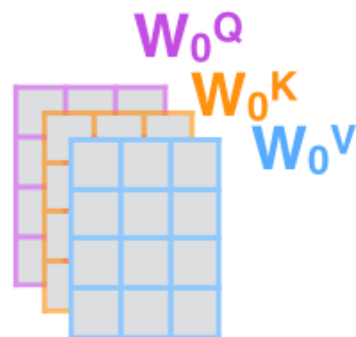
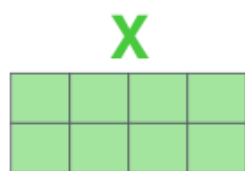
2) We embed each word*

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

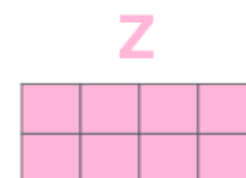
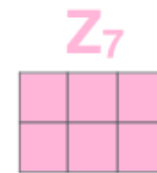
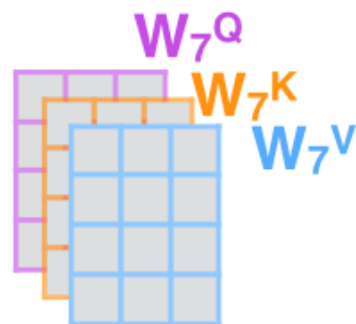
Thinking Machines



...

...

...

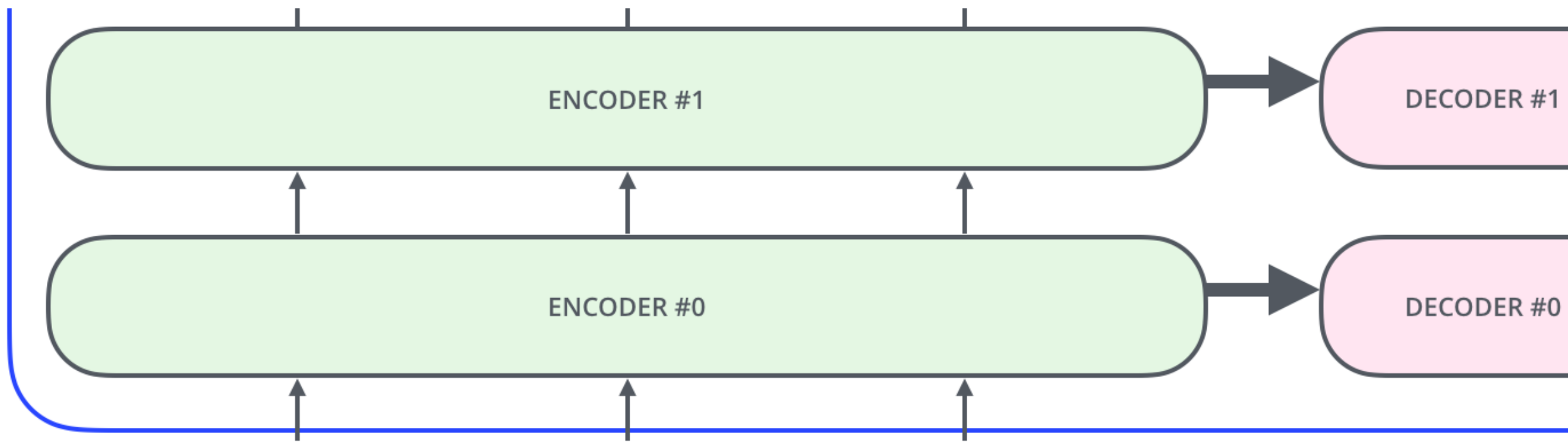


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

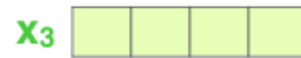
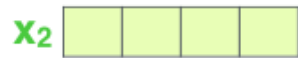
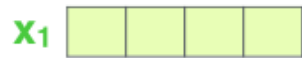
Position Encoding and Tokenizers

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



EMBEDDING
WITH TIME
SIGNAL

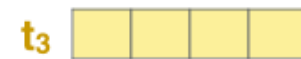
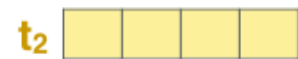
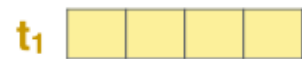


=

=

=

POSITIONAL
ENCODING

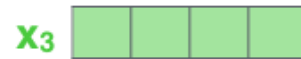
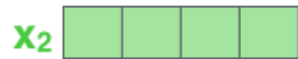
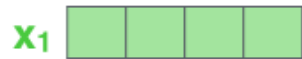


+

+

+

EMBEDDINGS



INPUT

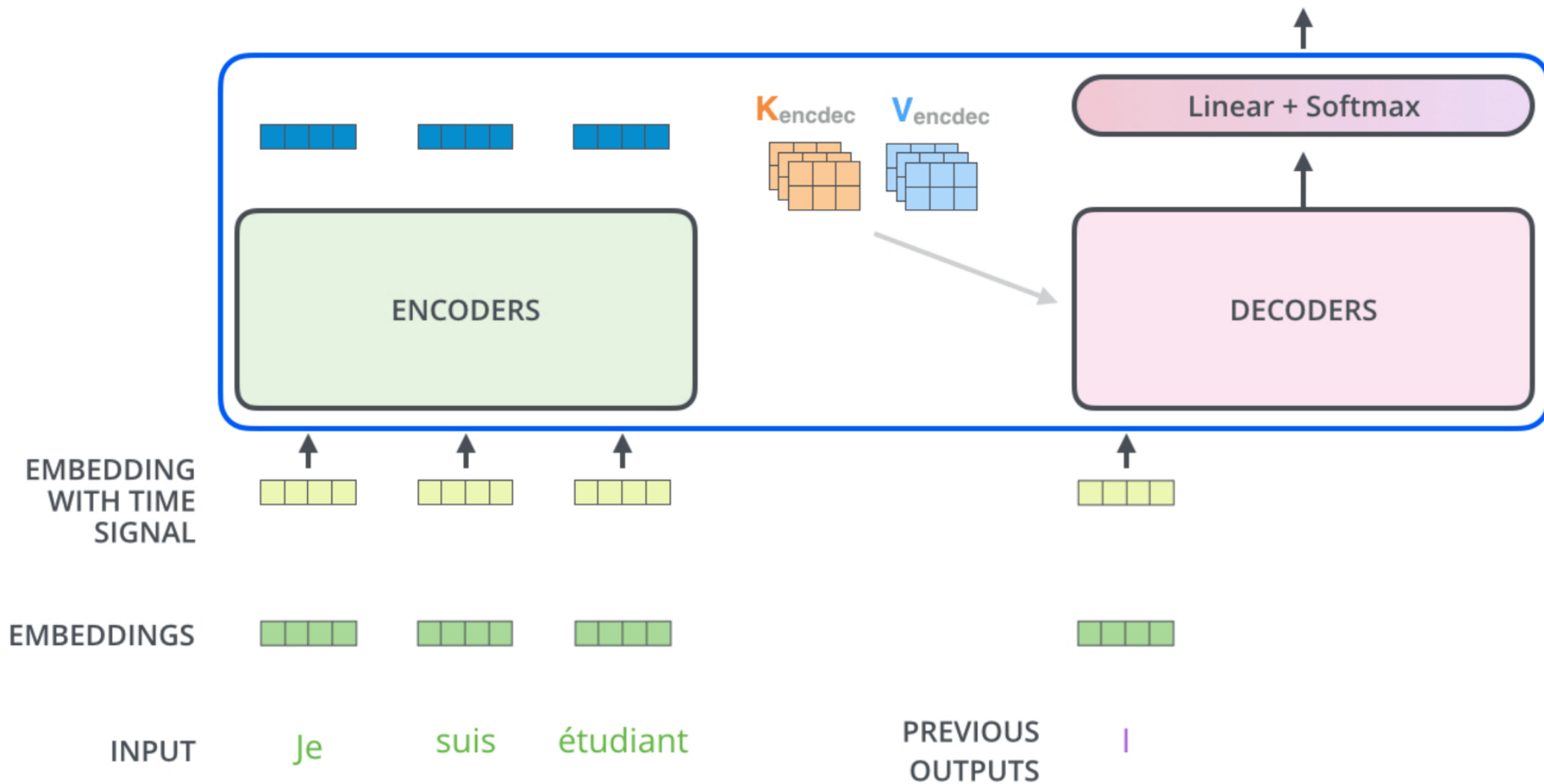
Je

suis

étudiant

Decoding time step: 1 2 3 4 5 6

OUTPUT |



Which word in our vocabulary
is associated with this index?

Get the index of the cell
with the highest value
(**argmax**)

log_probs

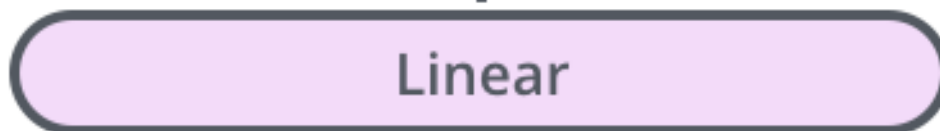


am

5



logits



Decoder stack output



Tokenization

Word Level

Deep Learning → Deep Learning

Character Level

Deep Learning → DeepLearning

Subword Level

Deep Learning → DeepLearn##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



Correct and desired output



a am I thanks student <eos>



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

- Translation
- Summarization
- Generation question-answering



EXAMPLES

BART, T5, Marian

Generating from the logits

- Top-k: select from k most likely tokens each time (k=1 for greedy)
- Sample: pick each token proportional to their probability
- Top-p (aka nucleus sampling): sample from smallest set of tokens that add up to a certain probability mass

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?