ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 17: Attention; Transformers

03/19/2025

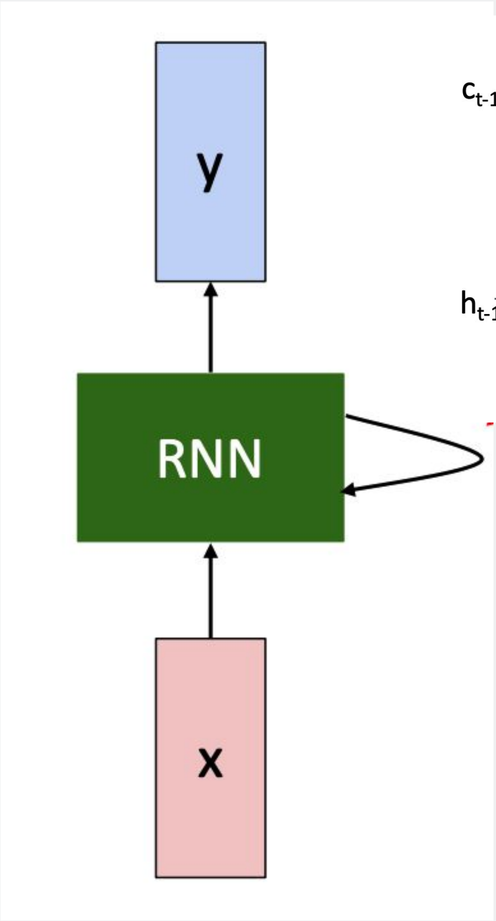
https://deeprob.org/w25/

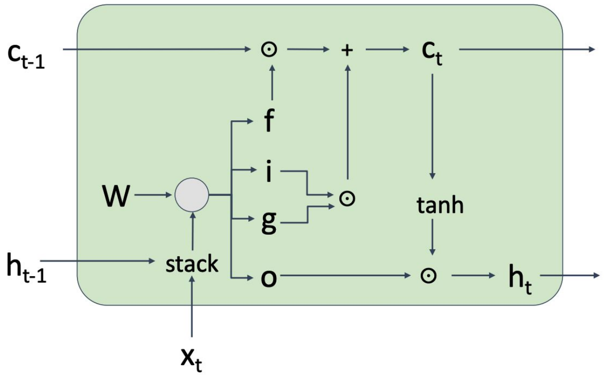
ROBOTICS

Today

- Feedback and Recap (5min)
- Attention (30min)
  - Example 1: Language translation
  - Example 2: video classification
- Transformers (30min)
- Vision Transformers (10min)
- Summary and Takeaways (5min)

Recap





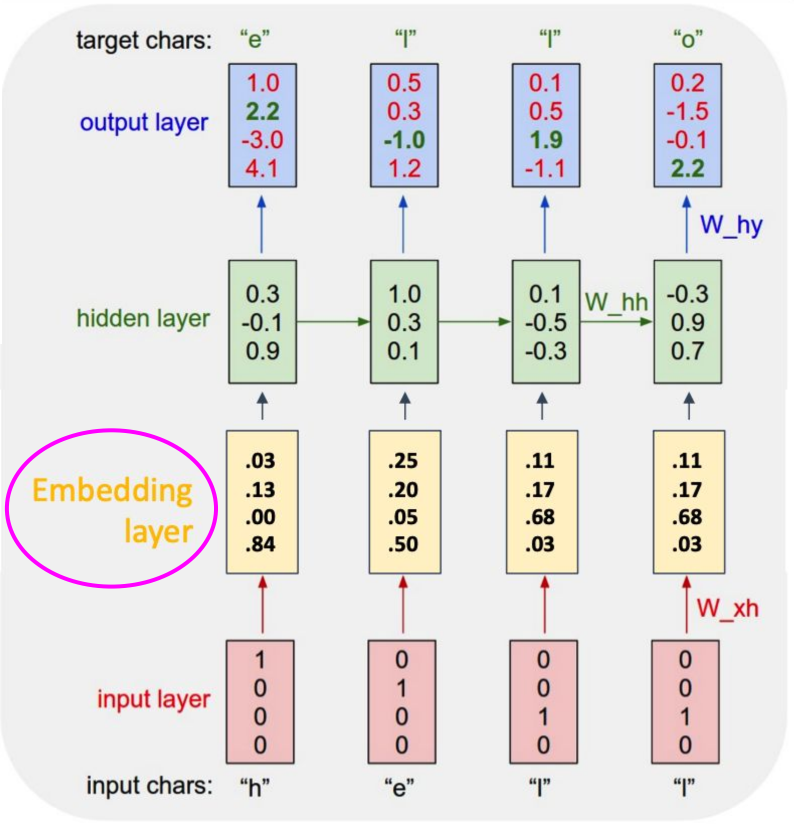
$$h_t = \tanh\left(W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix} + b_h\right)$$

LSTM:

$$\begin{pmatrix}i_t\\ f_t\\ o_t\\ g_t\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix}\left(W\begin{pmatrix}h_{t-1}\\ x_t\end{pmatrix} + b_h\right)$$

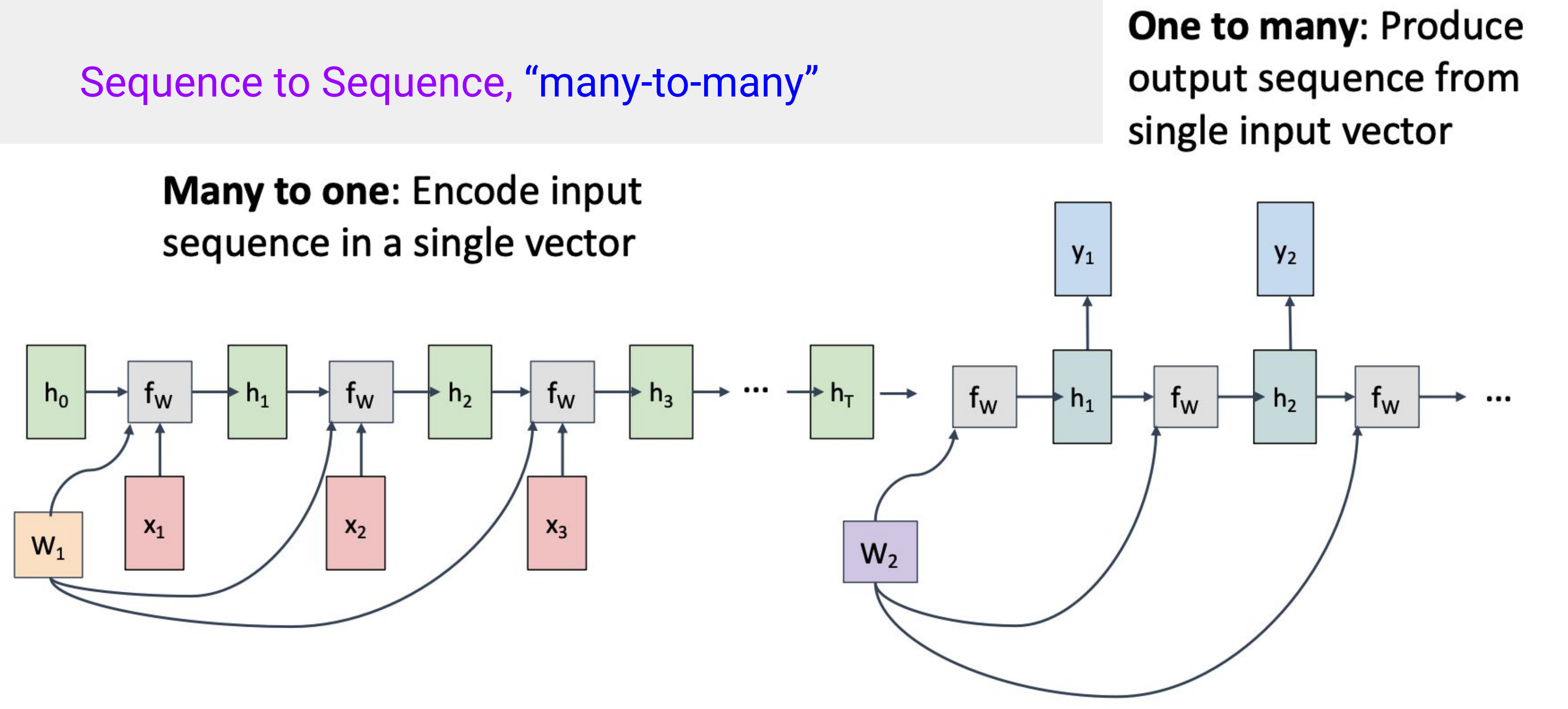
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

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



Recap: Seq2Seq

Sequence to Sequence, "many-to-many"



https://proceedings.neurips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

M ROBOTICS

Limitations of RNNs

- Modeling long-range dependencies limited by vanishing gradient
- Computational and memory efficiency, especially for long sequences
- Parallelization of layers that depend on sequential information

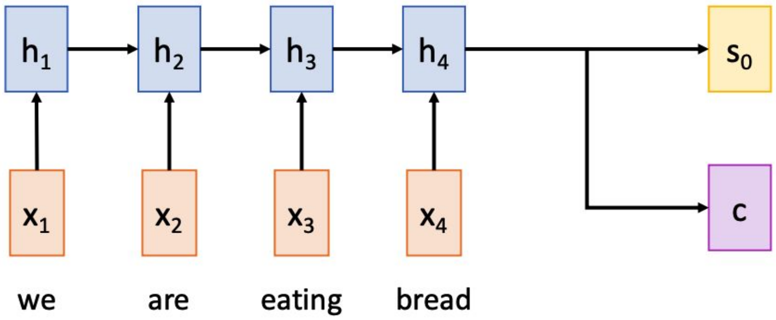
Modeling Long-Range Dependencies

One Example: Language Translation

Input: Sequence $x_1, \dots x_T$
Output: Sequence $y_1, \dots, y_{T'}$

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

From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c=h_T$)

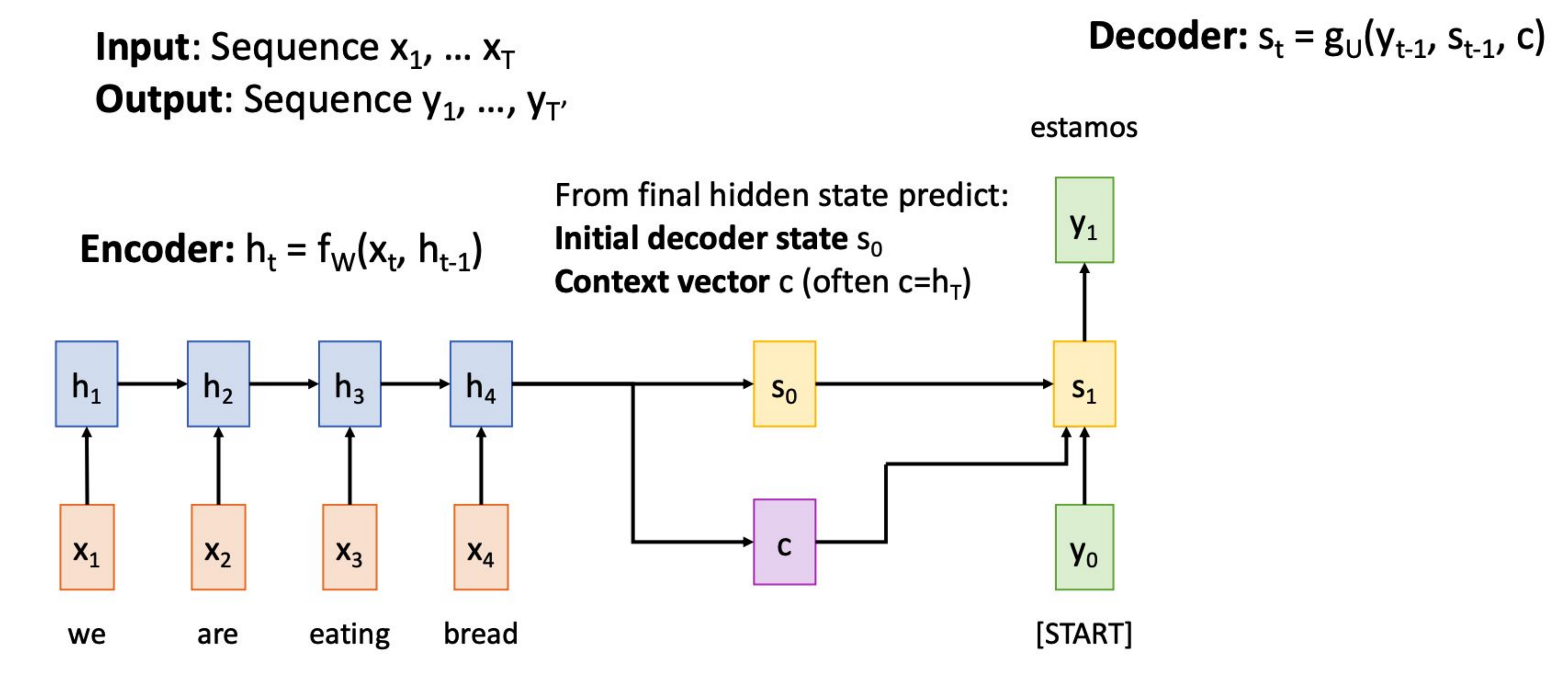


Recall: Seq2Seq + RNN

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014
https://papers.nips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

Modeling Long-Range Dependencies

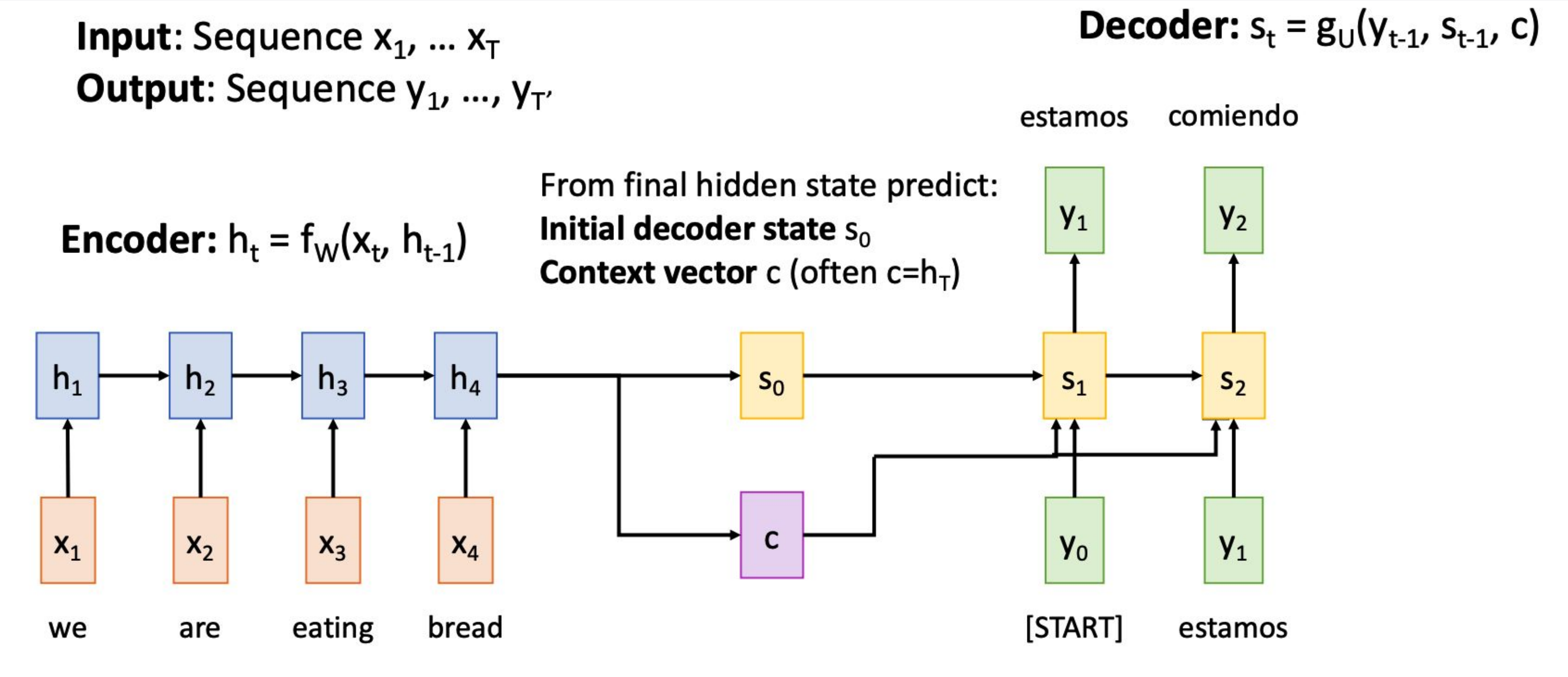
One Example: Language Translation



Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014
https://papers.nips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

Modeling Long-Range Dependencies

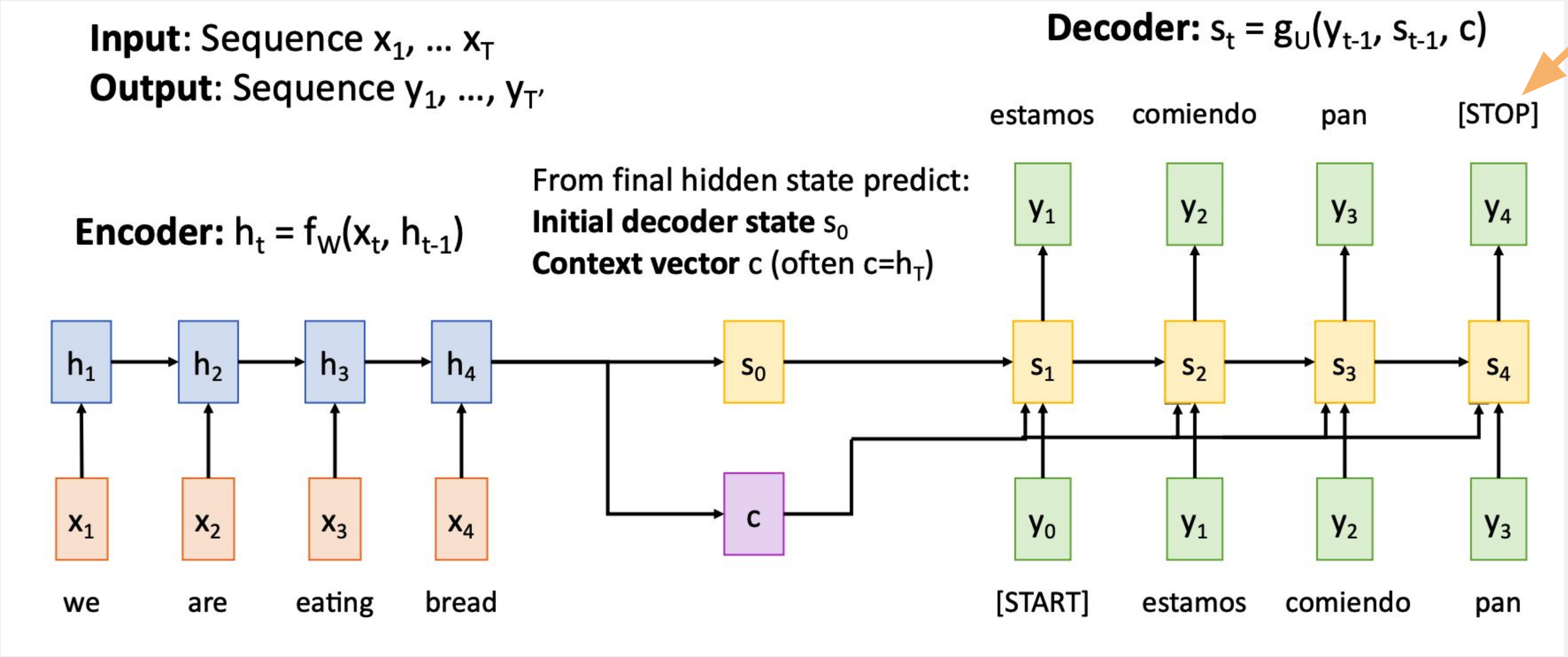
One Example: Language Translation



Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014
https://papers.nips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

Modeling Long-Range Dependencies

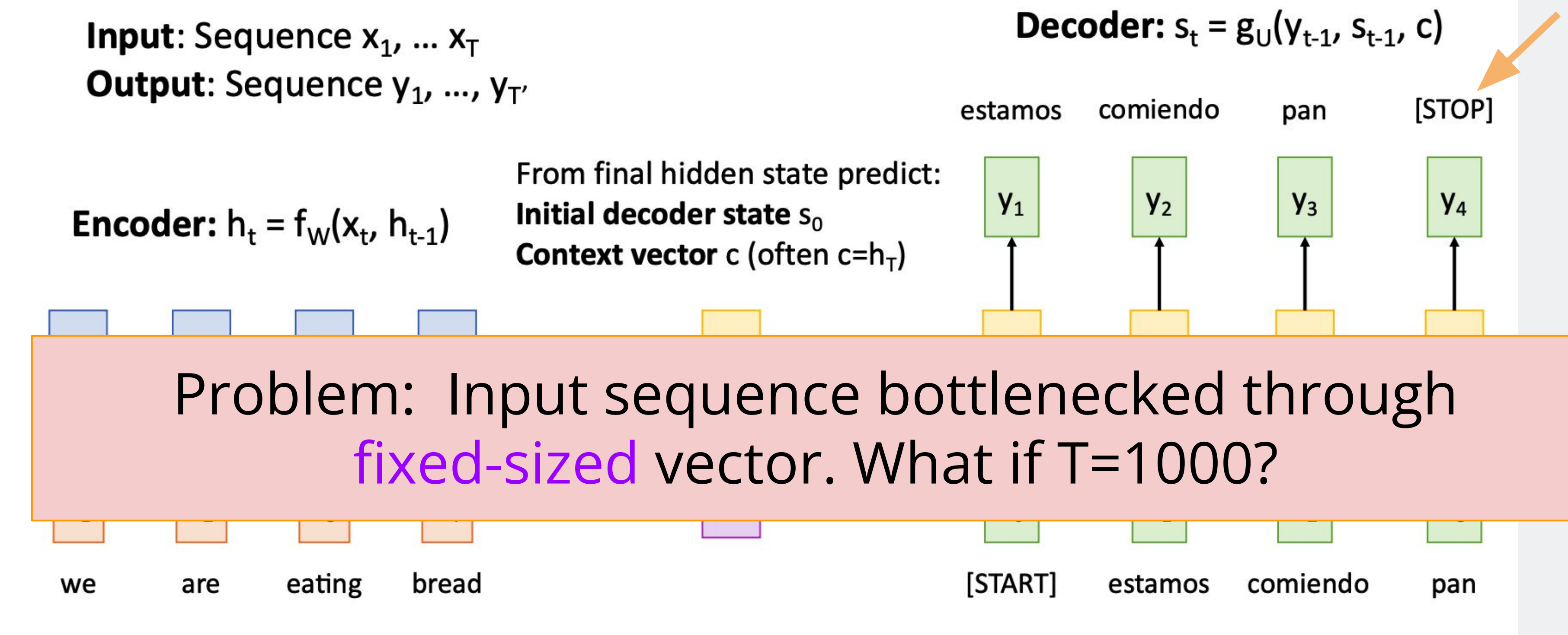
One Example: Language Translation



Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014
https://papers.nips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

Modeling Long-Range Dependencies

One Example: Language Translation



Problem: Input sequence bottlenecked through fixed-sized vector. What if T=1000?

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014
https://papers.nips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

Modeling Long-Range Dependencies

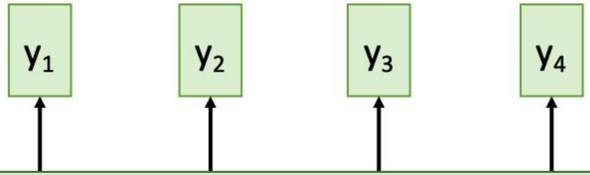
One Example: Language Translation

Input: Sequence $x_1, \ldots x_T$

Output: Sequence $y_1, \ldots, y_{T'}$

Decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c)$

estamos comiendo pan [STOP]



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

From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c=h_T$)

Idea: Instead of than summarizing entire input sequence into one context vector c, let's have decoder compute its own context vector

we are eating bread [START] estamos comiendo pan

Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014
https://papers.nips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

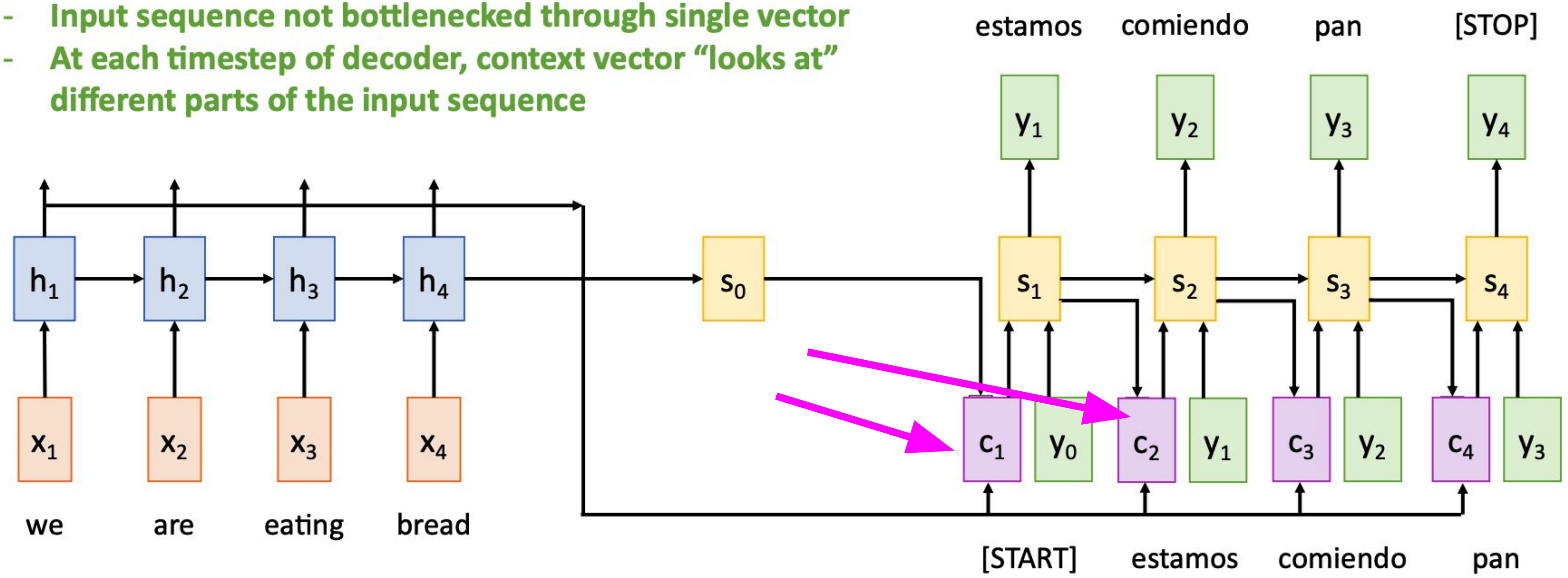
ROBOTICS

Modeling Long-Range Dependencies

One Example: Language Translation

Use a different context vector in each timestep of decoder

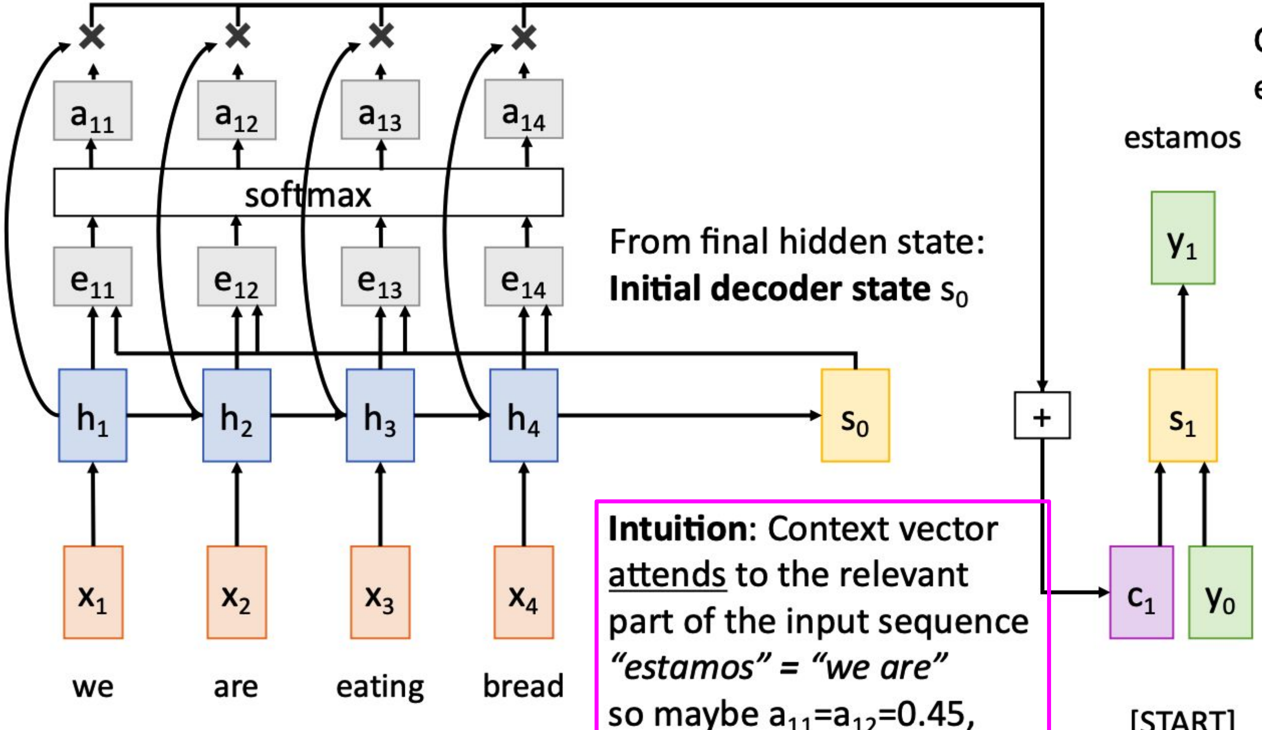
- **Input sequence not bottlenecked through single vector**
- **At each timestep of decoder, context vector "looks at" different parts of the input sequence**



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015
https://arxiv.org/abs/1409.0473

Seq2Seq “Attention”

One Example: Language Translation



From final hidden state: **Initial decoder state s_0**

Intuition: Context vector attends to the relevant part of the input sequence *“estamos” = “we are”* so maybe $a_{11}=a_{12}=0.45$, $a_{13}=a_{14}=0.05$

Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

Normalize alignment scores to get **attention weights**
$0 < a_{t,i} < 1$ $\sum_i a_{t,i} = 1$

Compute context vector as linear combination of hidden states
$c_t = \sum_i a_{t,i} h_i$

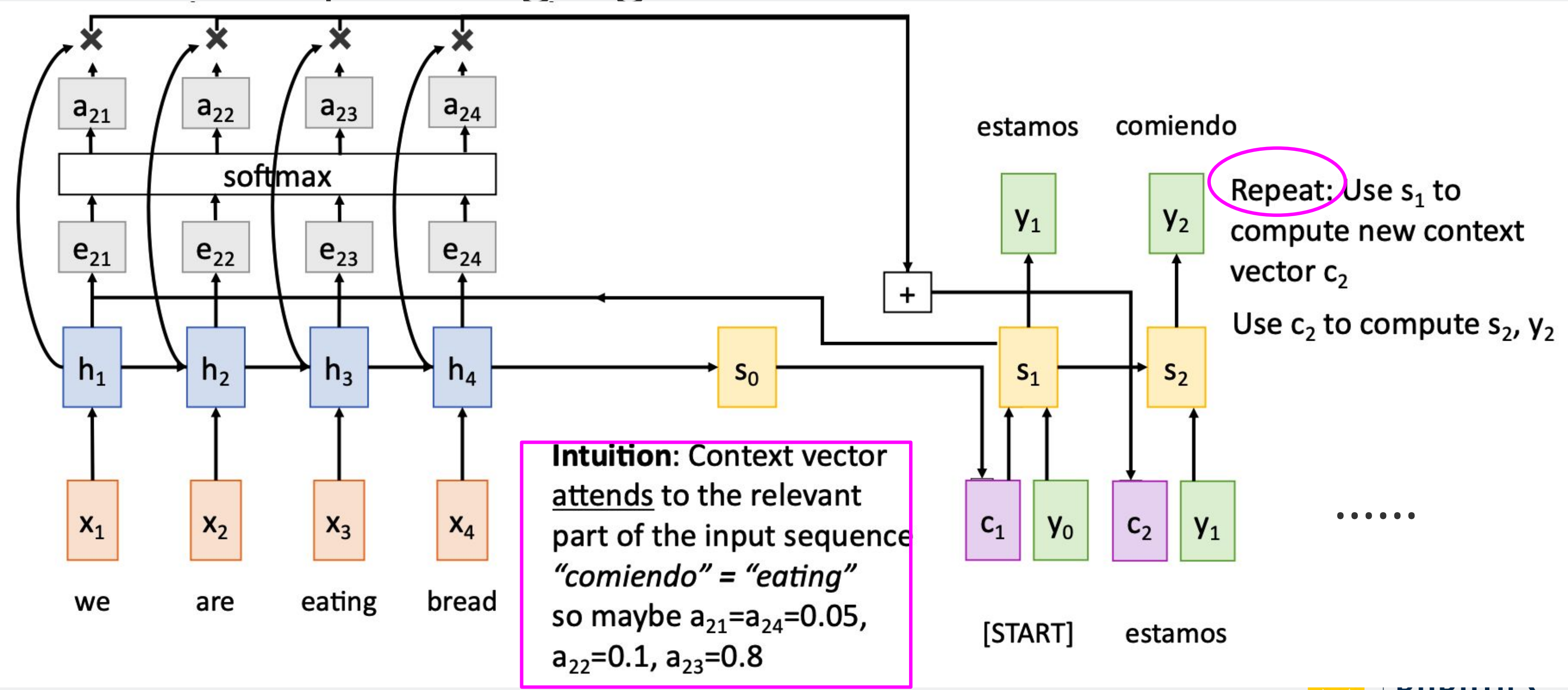
Use context vector in decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

This is all differentiable! Do not supervise attention weights – backprop through everything

ahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015

Seq2Seq "Attention"

One Example: Language Translation



Visualizing Attention (language example)

One Example: Language Translation

Example: English to French translation

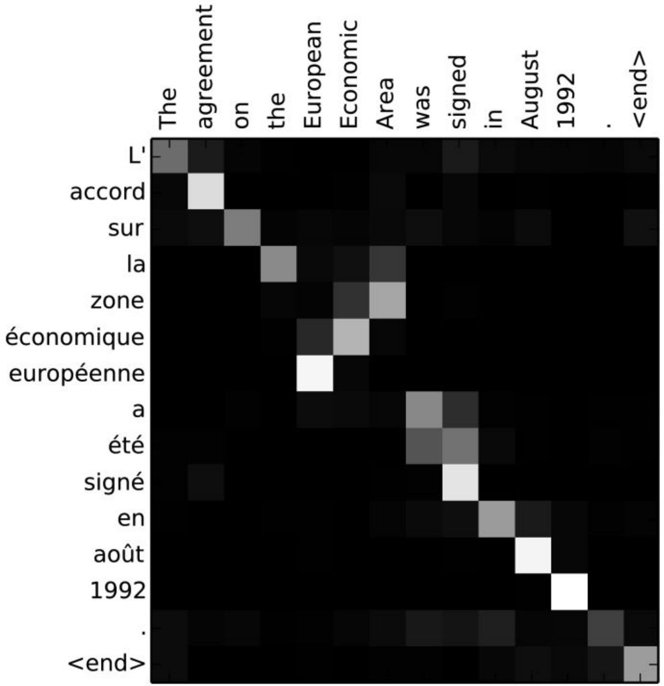
Input: "The agreement on the European Economic Area was signed in August 1992."

Output: "L'accord sur la zone économique européenne a été signé en août 1992."

https://ahaslides.com/0BWPC

Q: what does diagonal weights mean?

Visualize attention weights $a_{t,i}$



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015
https://arxiv.org/abs/1409.0473

Modeling Long-Range Dependencies

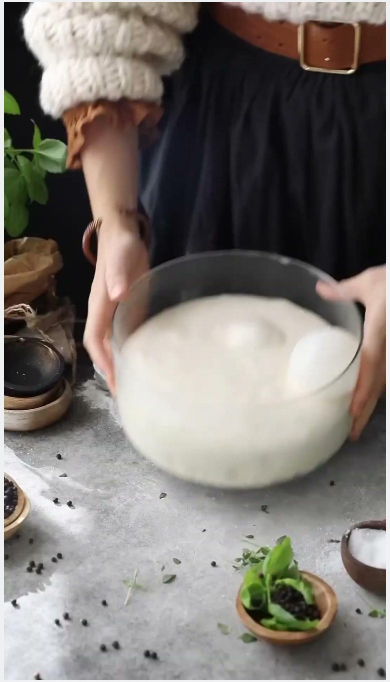
Another Example: Per-frame video classification

Question: what do you think the chef is making?

Source: TurkuazKitchen

Modeling Long-Range Dependencies

Example: Per-frame video classification



Source: TurkuazKitchen

Question: what do you think the chef is making?

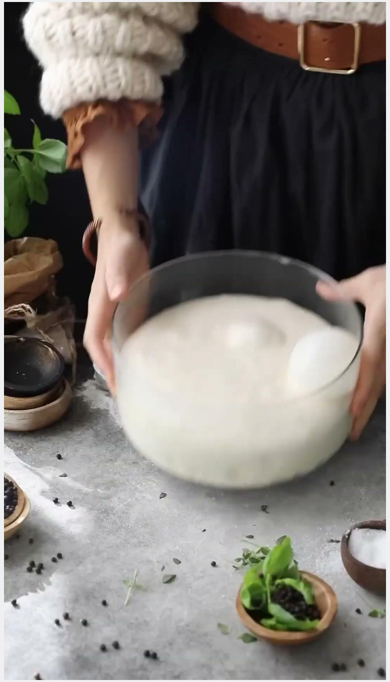
Max-Likelihood:

- Pasta sauce (p=0.5)
- Meatballs (p=0.5)

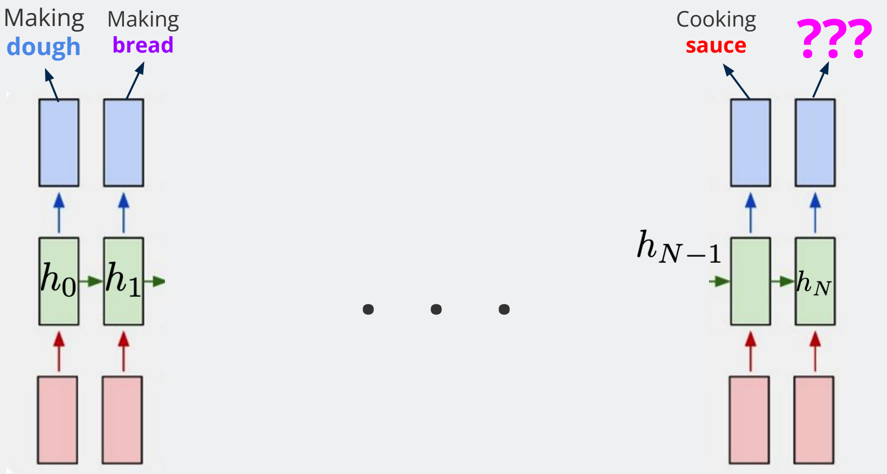
M | ROBOTICS

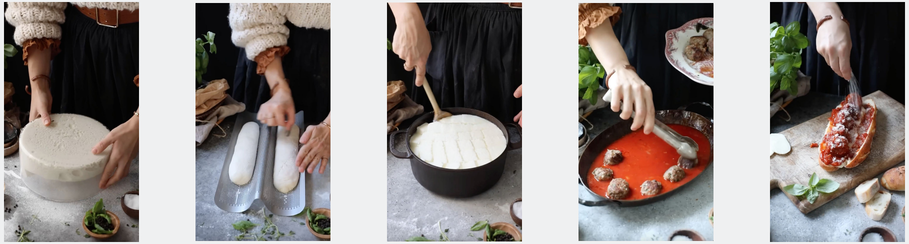
Modeling Long-Range Dependencies

Q: What happens if the hidden state can't encode all previous images' information?



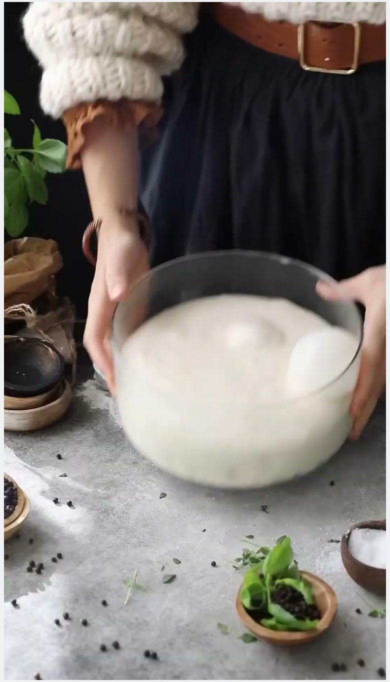
Source: TurkuazKitchen



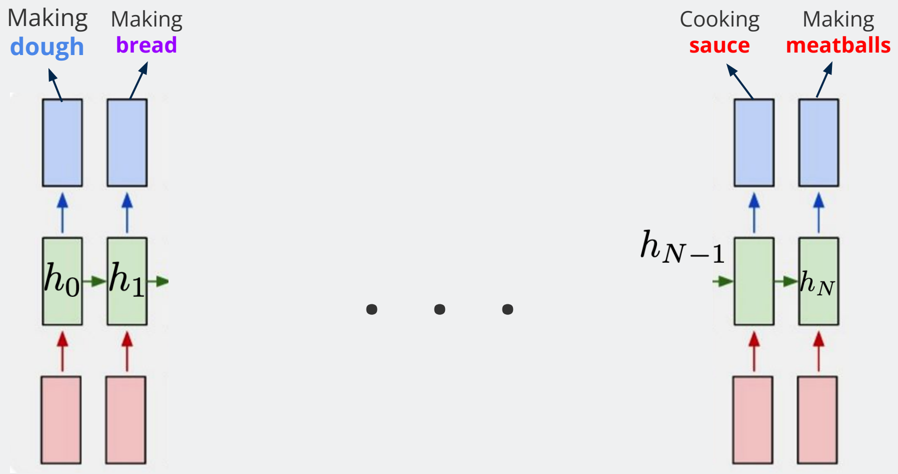


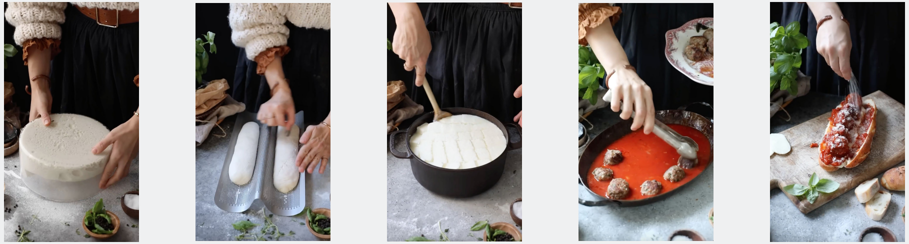
Modeling Long-Range Dependencies

A: The predictions will be biased by local information



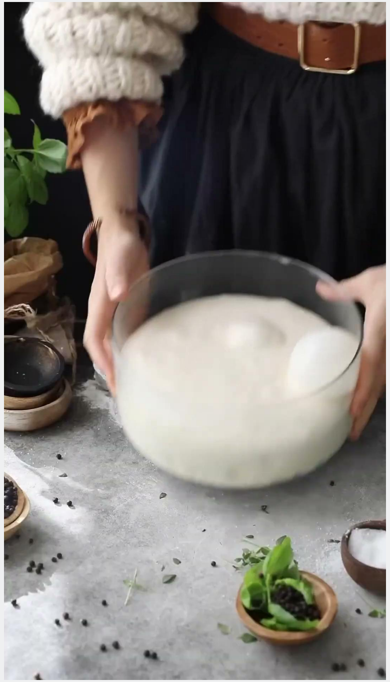
Source: TurkuazKitchen



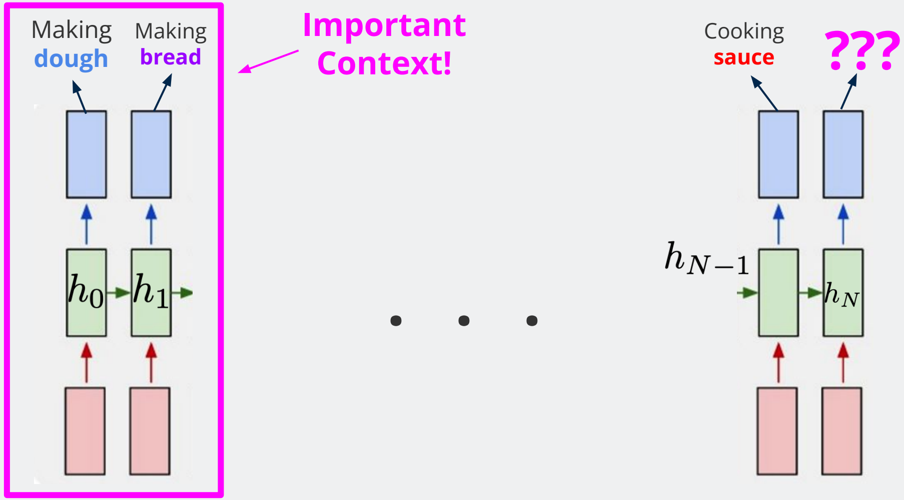


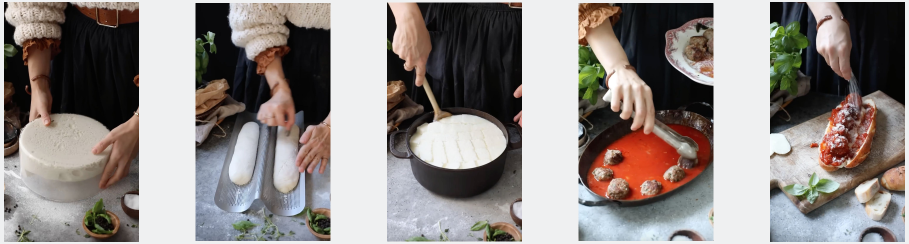
Modeling Long-Range Dependencies

Q: How can we ensure the model has access to earlier information?



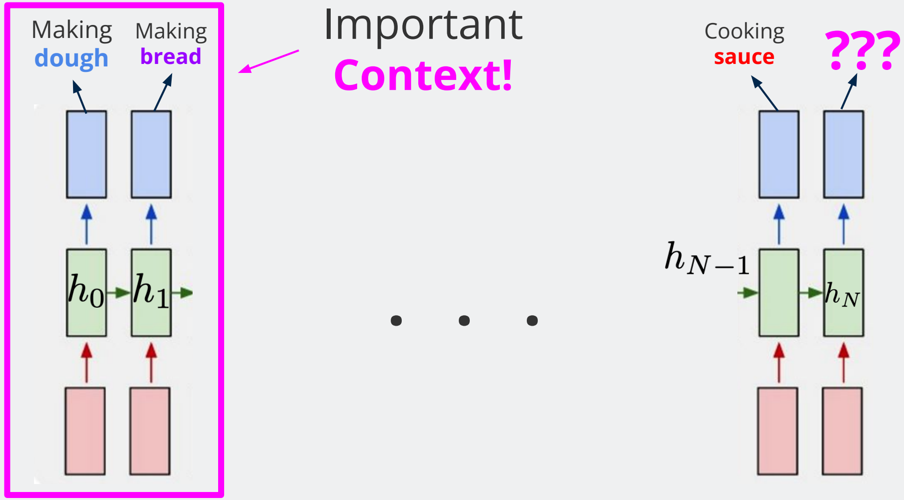
Source: TurkuazKitchen

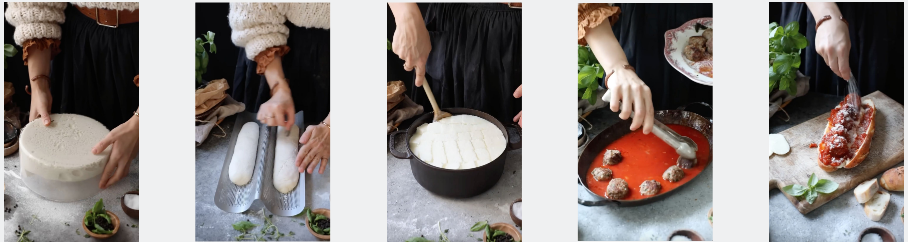




Attention for Modeling Long-Range Dependencies

Q: How can we ensure the model has access to earlier information?

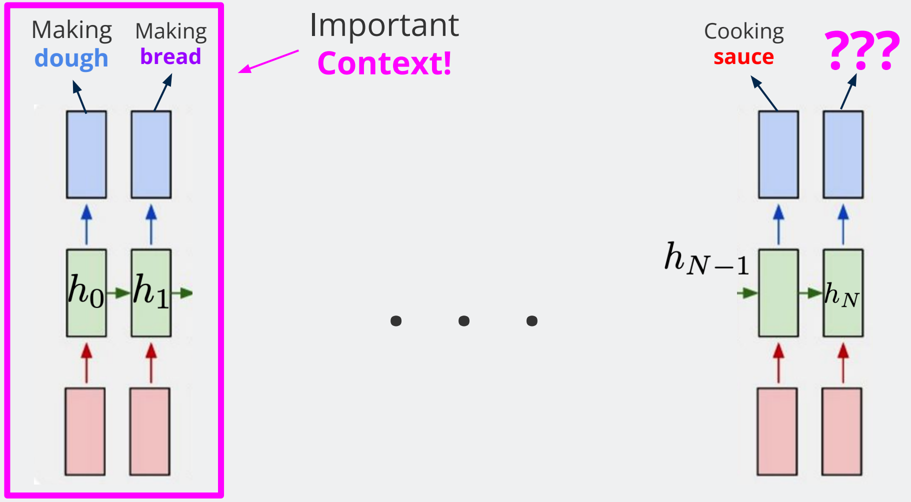


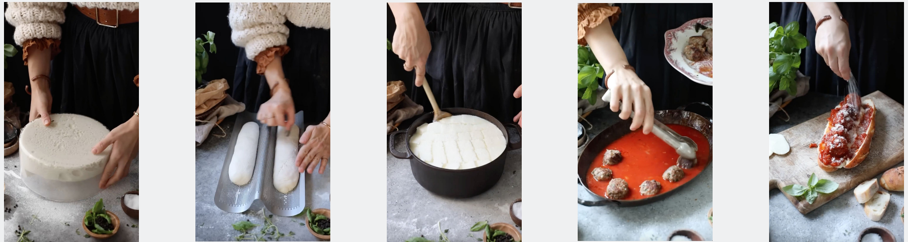


Source: TurkuazKitchen

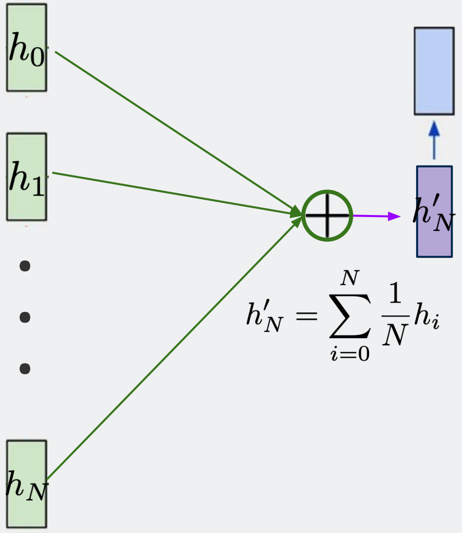
Attention for Modeling Long-Range Dependencies

Q: How can we ensure the model has access to earlier information?





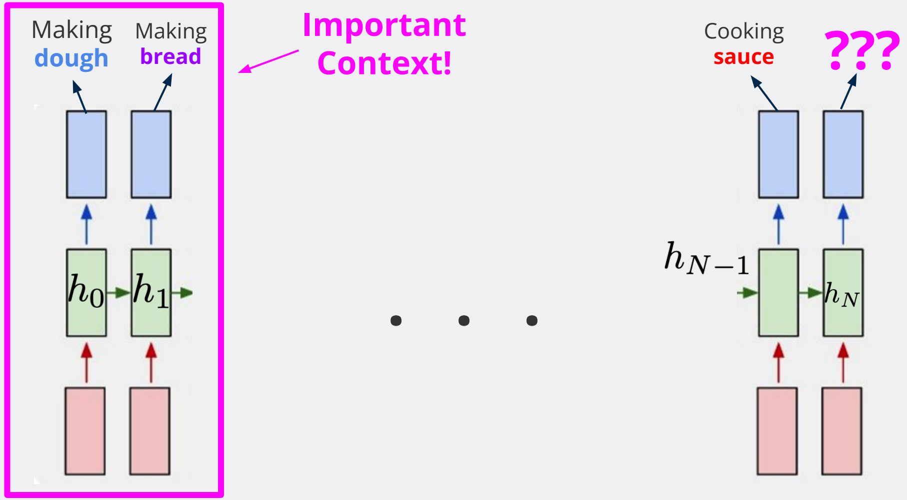
Idea 1: Combine all previous hidden states to provide context before decoding

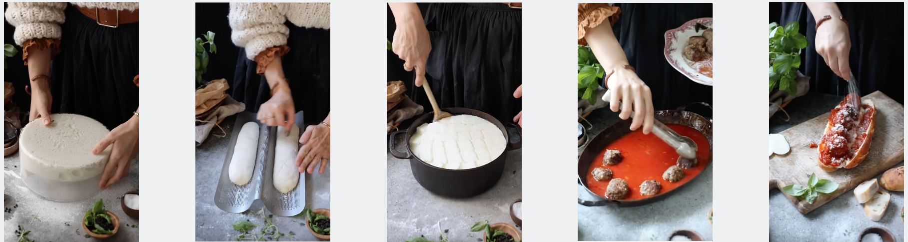


Source: TurkuazKitchen

Attention for Modeling Long-Range Dependencies

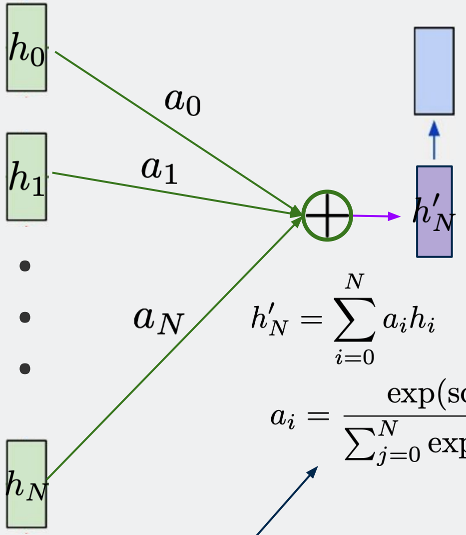
Q: How can we ensure the model has access to earlier information?





Source: TurkuazKitchen

Idea 2: Linearly combine all previous hidden states conditioned on current hidden state



$$h'_N = \sum_{i=0}^{N} a_i h_i$$

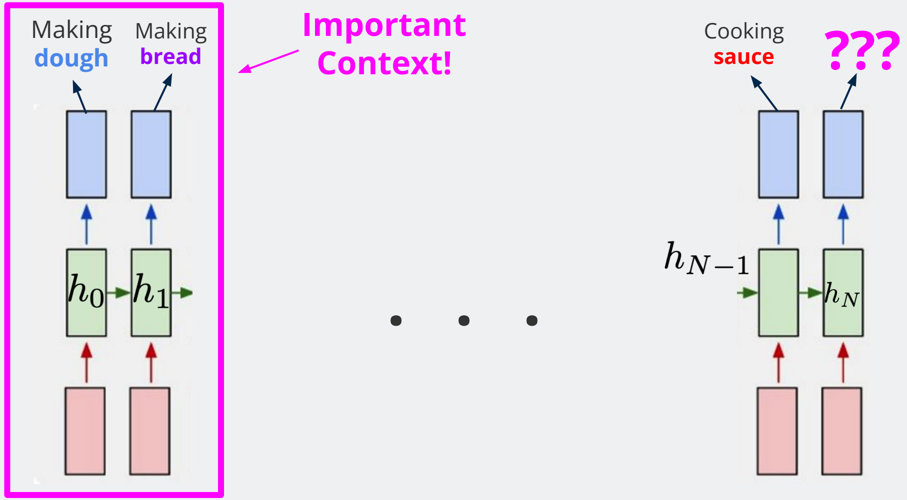
$$a_i = \frac{\exp(\text{score}(h_N, h_i))}{\sum_{j=0}^{N} \exp(\text{score}(h_N, h_j))}$$

Normalize scores to sum to 1

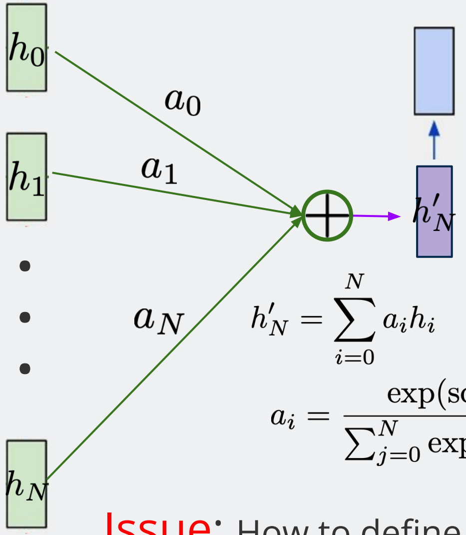
ROBOTICS

Attention for Modeling Long-Range Dependencies

Q: How can we ensure the model has access to earlier information?



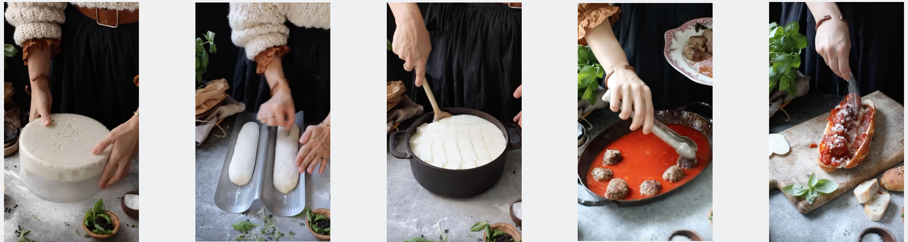
Idea 2: Linearly combine all previous hidden states conditioned on current hidden state



$$h'_N = \sum_{i=0}^{N} a_i h_i$$

$$a_i = \frac{\exp(\text{score}(h_N, h_i))}{\sum_{j=0}^{N} \exp(\text{score}(h_N, h_j))}$$

Issue: How to define the scoring function?

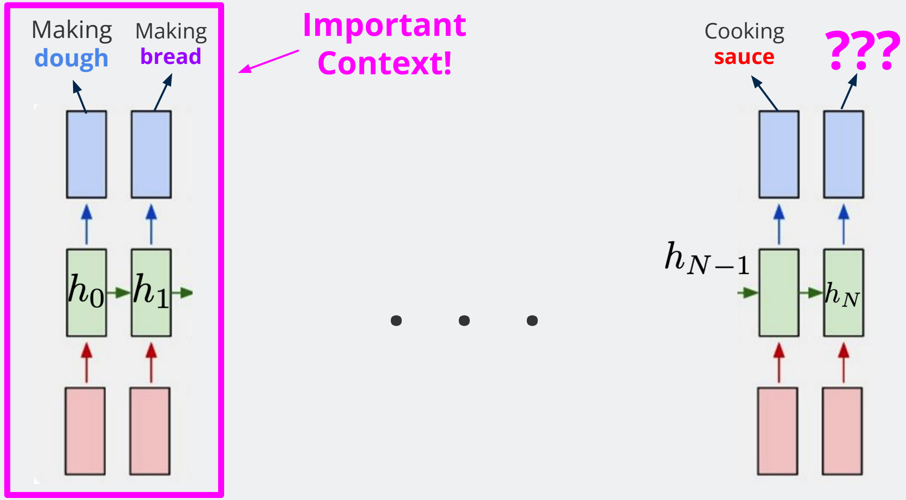


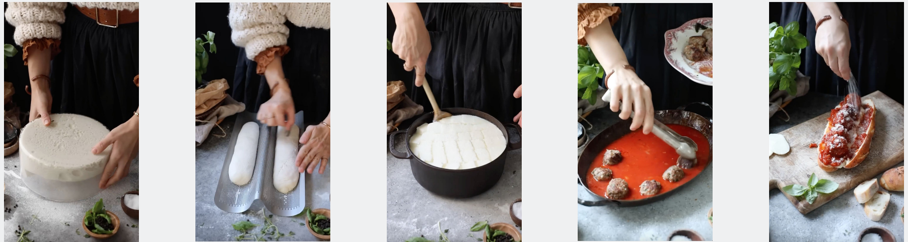
Source: TurkuazKitchen

ROBOTICS

Attention for Modeling Long-Range Dependencies

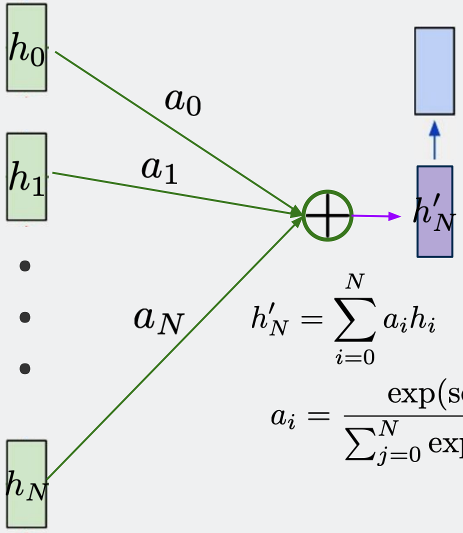
Q: How can we ensure the model has access to earlier information?





Source:
TurkuazKitchen

Idea 3: Train the model to linearly combine all previous hidden states conditioned on

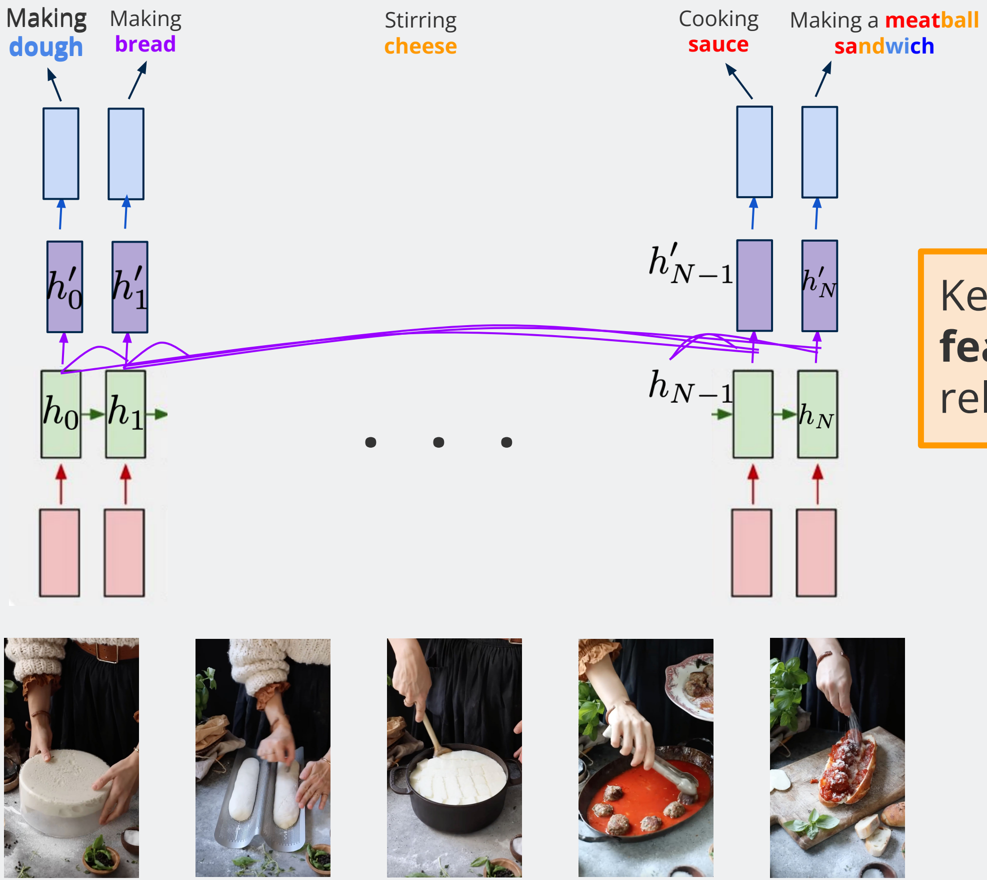


$$h'_N = \sum_{i=0}^{N} a_i h_i$$

$$a_i = \frac{\exp(\text{score}(h_N, h_i))}{\sum_{j=0}^{N} \exp(\text{score}(h_N, h_j))}$$

Use a learned NN to model the score function, thereby decoder learns to model **'attention'**

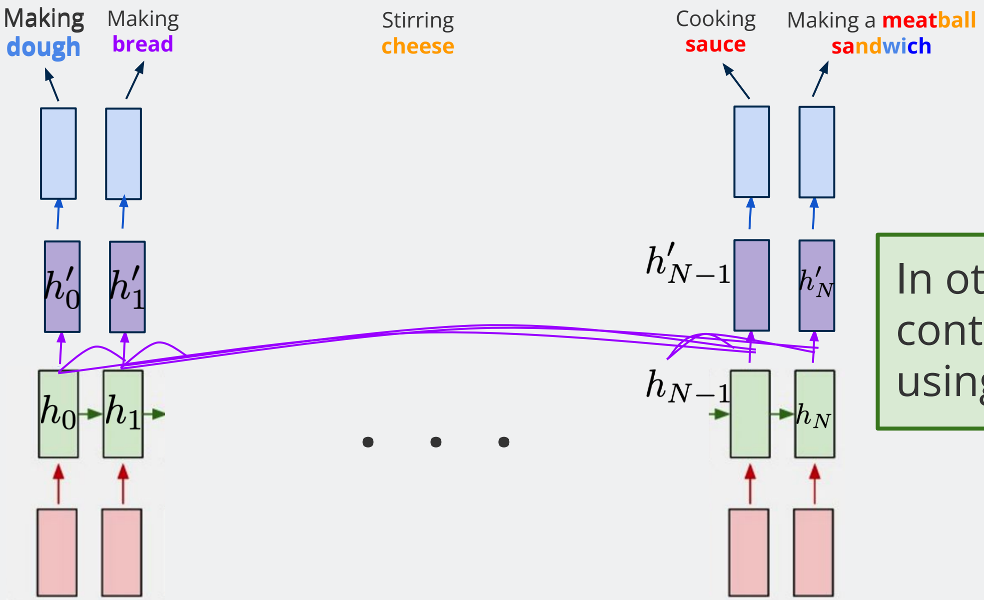
Attention for Modeling Long-Range Dependencies



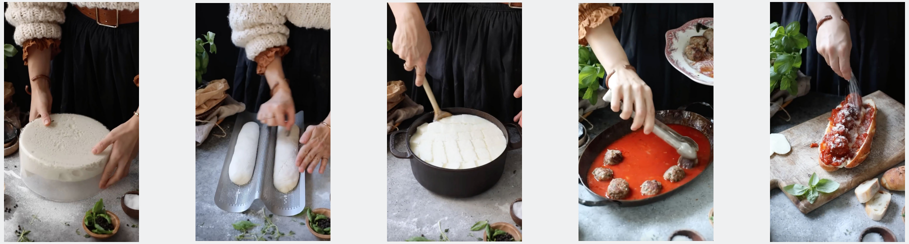
Key idea: **Recombine input features to provide context** relevant to current output token

Source: TurkuazKitchen

Attention for Modeling Long-Range Dependencies

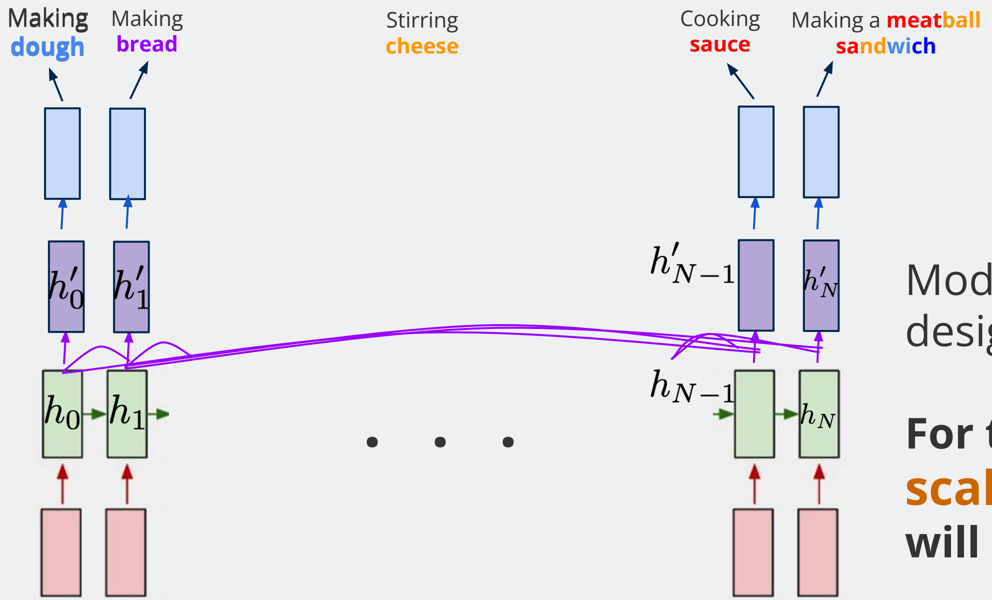


In other words: "**lookup**" the relevant context from past token features using current token features as a 'key'



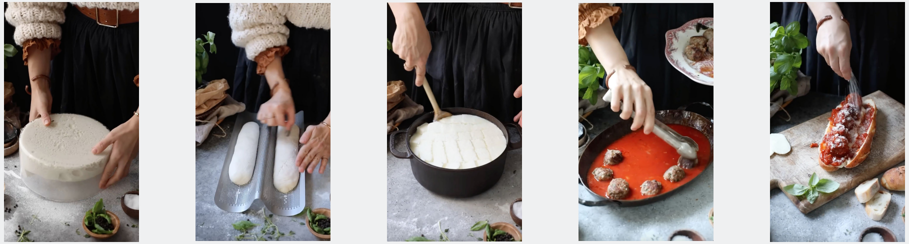
Source: TurkuazKitchen

Attention for Modeling Long-Range Dependencies



Modeling the score function is a design choice

For this course:
scaled dot-product attention
will be used



Source:
TurkuazKitchen

M | ROBOTICS

Introducing Attention and the Transformer Architecture

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

Vaswani et al. NeurIPS’17

https://arxiv.org/abs/1706.03762

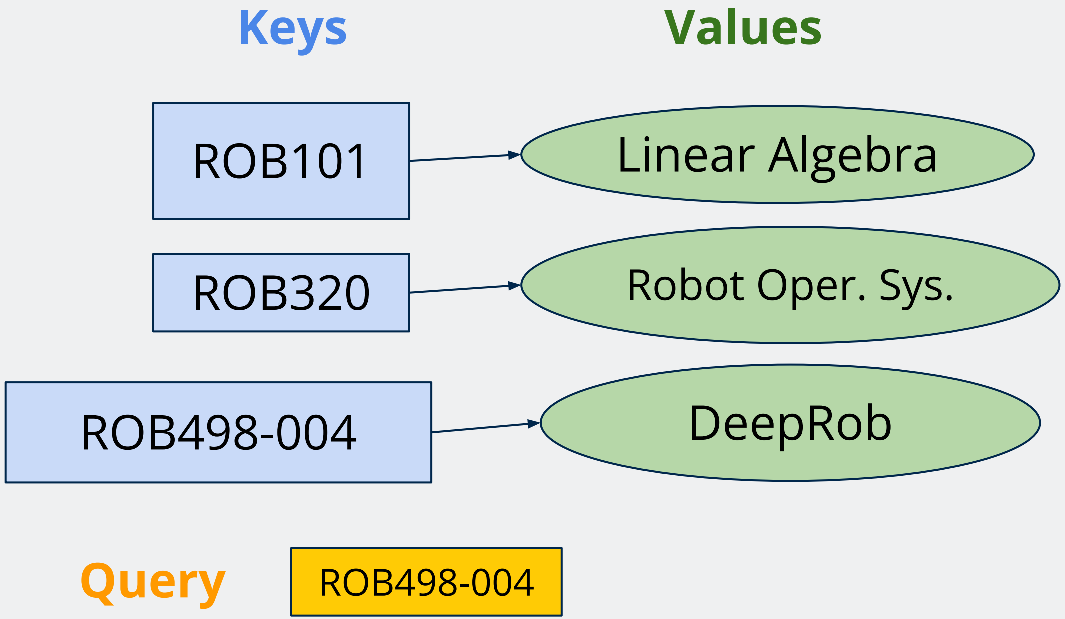
Formalizing Attention: Scaled Dot-Product Attention

Inspiration:

- The attention mechanism is reminiscent of a database system
- We have stored features from past tokens
- We want to 'lookup' information from relevant token features conditioned on our current feature

(Example:)

Database/Lookup Table

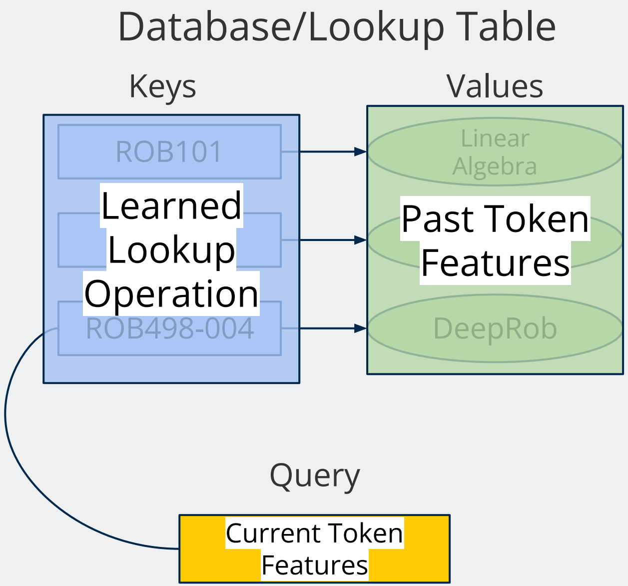


(soon-to-be ROB430 :))

Formalizing Attention: Scaled Dot-Product Attention

Inspiration:

- The attention mechanism is reminiscent of a database system
- We have stored features from past tokens
- We want to 'lookup' information from relevant token features conditioned on our current feature



Intuition:

→ Think of current features as a **query**

→ Past token features as the **values** to search/recombine

→ Values then are the index used to search past tokens

Formalizing Attention: Scaled Dot-Product Attention

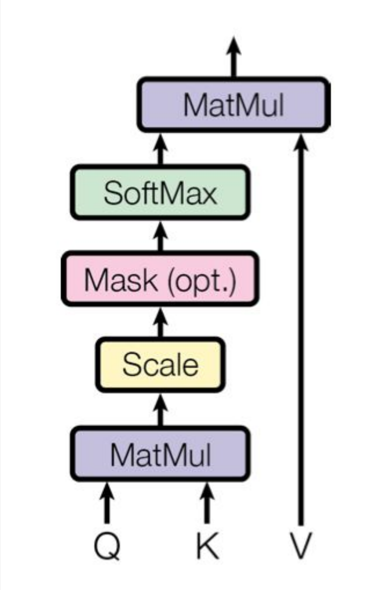
Inspiration:

- The attention mechanism is reminiscent of a database system
- We have stored features from past tokens
- We want to 'lookup' information from relevant token features conditioned on our current feature

Intuition:

➔ Think of current features as a query
➔ Past token features as the values to search/recombine
➔ Values then are the index used to search past tokens

Scaled Dot-Product Attention



Vaswani et al. NeurIPS'17

Formalizing Attention: Scaled Dot-Product Attention

Inputs:

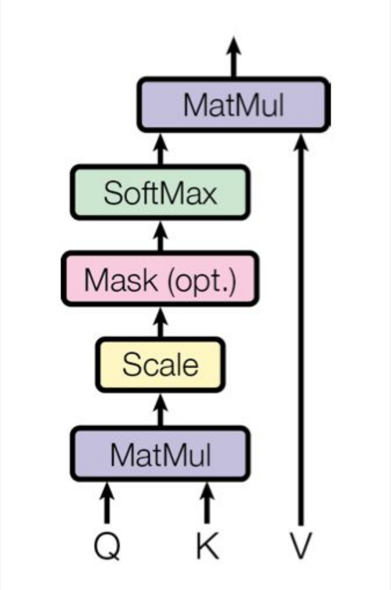
$$Q \in \mathbb{R}^{P \times d_k}$$

$$K \in \mathbb{R}^{N \times d_k}$$

$$V \in \mathbb{R}^{N \times d_v}$$

where, P is the number of input tokens to attend
d_k is the dimension of input token features
d_v is the dimension of output token features
N is the number of context tokens (e.g. sequence length or number of patches for vision transformer)

Scaled Dot-Product Attention



Vaswani et al. NeurIPS’17

Scaled Dot-Product Attention

Inputs:

$$Q \in \mathbb{R}^{P \times d_k}$$

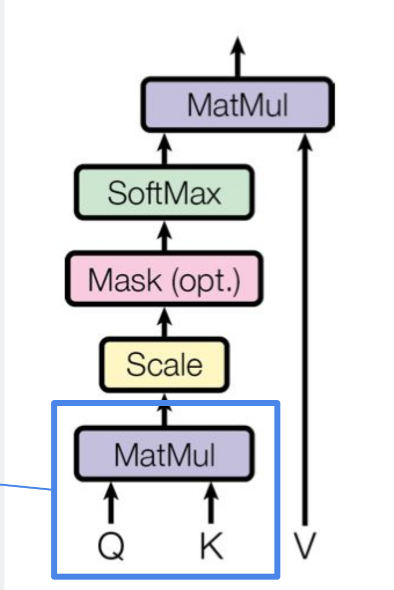
$$K \in \mathbb{R}^{N \times d_k}$$

$$V \in \mathbb{R}^{N \times d_v}$$

$$\boxed{QK^T \in \mathbb{R}^{P \times N}}$$

Each row measures the inner product between a given input token's features and each key feature vector

Scaled Dot-Product Attention



Vaswani et al. NeurIPS'17

Scaled Dot-Product Attention

Inputs:

$$Q \in \mathbb{R}^{P \times d_k}$$

$$K \in \mathbb{R}^{N \times d_k}$$

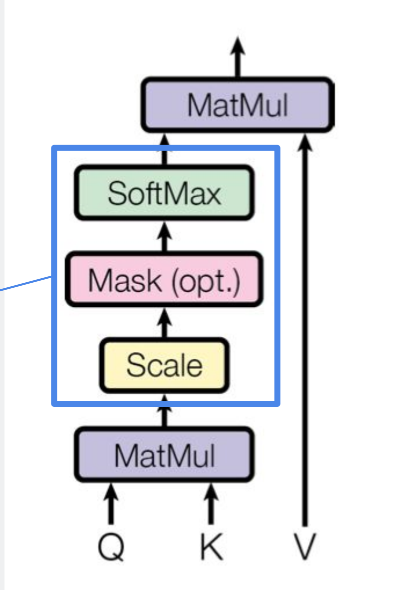
$$V \in \mathbb{R}^{N \times d_v}$$

$$\text{attention} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \in \mathbb{R}^{P \times N}$$

$$\sum_{j=1}^{N} \text{attention}_{i,j} = 1 \qquad \forall i \in 1 \dots P$$

Softmax ensures the attention weights sum to 1 for each input token

Scaled Dot-Product Attention



Vaswani et al. NeurIPS'17

Scaled Dot-Product Attention

Inputs:

$$Q \in \mathbb{R}^{P \times d_k}$$

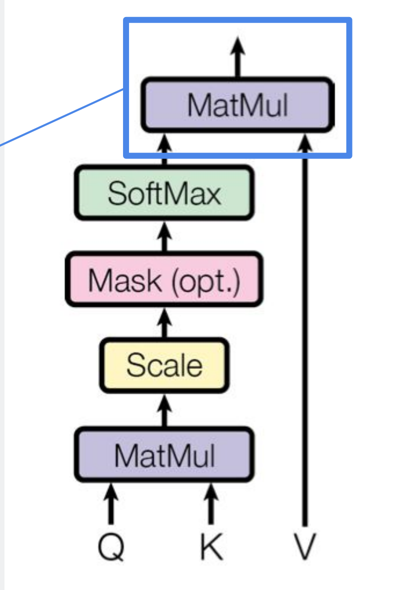
$$K \in \mathbb{R}^{N \times d_k}$$

$$V \in \mathbb{R}^{N \times d_v}$$

$$\boxed{\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \in \mathbb{R}^{P \times d_v}}$$

Output is a linear (re-)combination of the input value tokens based on the pairwise similarity of query and key features

Scaled Dot-Product Attention



Vaswani et al. NeurIPS'17

Scaled Dot-Product Attention

Inputs:

$$Q \in \mathbb{R}^{P \times d_k}$$

$$K \in \mathbb{R}^{N \times d_k}$$

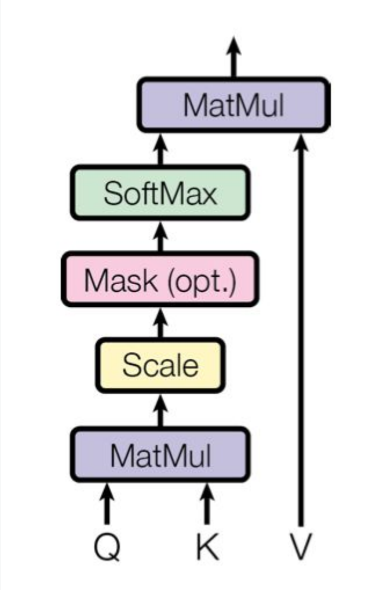
$$V \in \mathbb{R}^{N \times d_v}$$

(Section 3.2.1 paper)

$$\boxed{\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \in \mathbb{R}^{P \times d_v}}$$

Q: Given an input sequence of token features, what are the query, key and value features?

Scaled Dot-Product Attention



Vaswani et al. NeurIPS'17

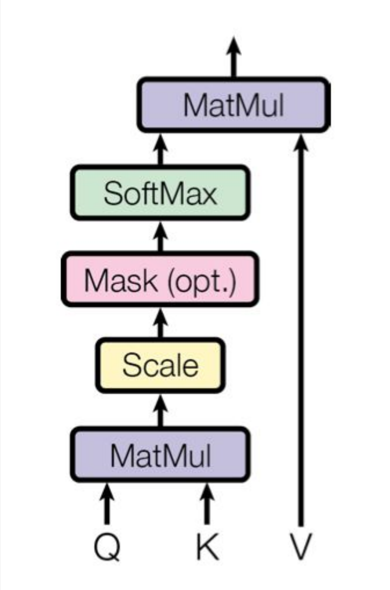
Scaled Dot-Product Attention

Inputs:

$$Q \in \mathbb{R}^{P \times d_k}$$
$$K \in \mathbb{R}^{N \times d_k}$$
$$V \in \mathbb{R}^{N \times d_v}$$

$$\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \in \mathbb{R}^{P \times d_v}$$

Scaled Dot-Product Attention



Vaswani et al. NeurIPS'17

Answer: Given a set of input token features X, the query, key and value features are formed by applying three independent learned linear layers to transform X into Q, K and V respectively

- Effectively allows model to learn the lookup operation
- Referred to as **self-attention**

(P4) `class Attention`

Multi-Head Attention

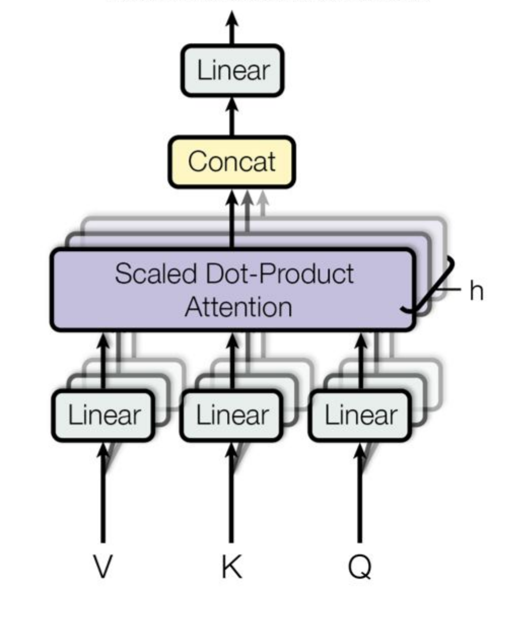
(Section 3.2.2 paper)

In-practice, it is often beneficial to project the input tokens into multiple (h) subspaces before applying scaled dot-product attention

The final output is then the **concatenated** result of each independent attention head

You will experiment with this in project 4 transformer part

Multi-Head Attention



Vaswani et al. NeurIPS'17

(P4) `class MultiHeadAttention`

The Transformer Architecture

Transformer Block:

Input: Set of vectors x

Output: Set of vectors y

Self-attention is the only interaction between vectors!

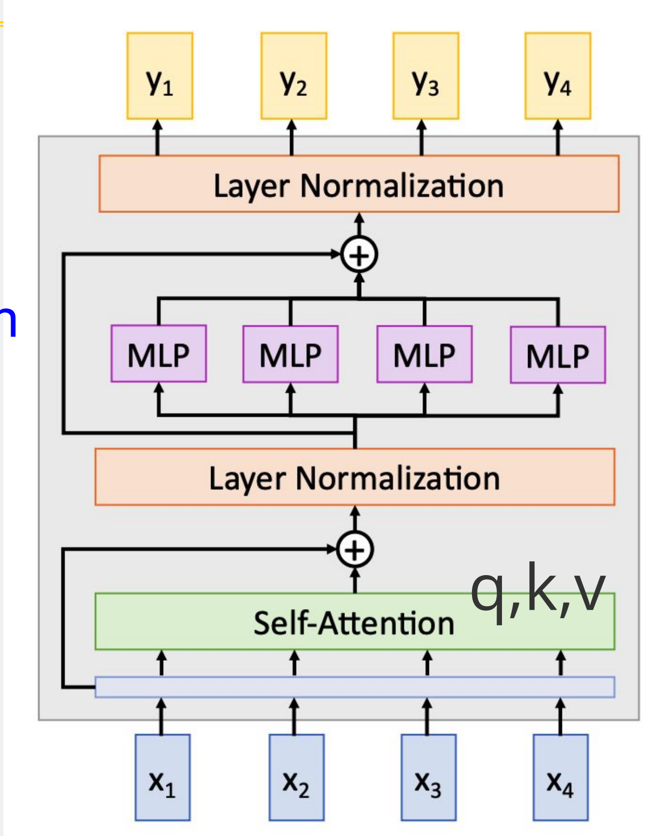
Layer norm and MLP work independently per vector

Highly scalable, highly parallelizable

*Few Parameters!

layernorm1 -> (multihead)self-attention -> layernorm2 -> mlp

with residual connections from input to attention output and from pre-layernorm2 to mlp output



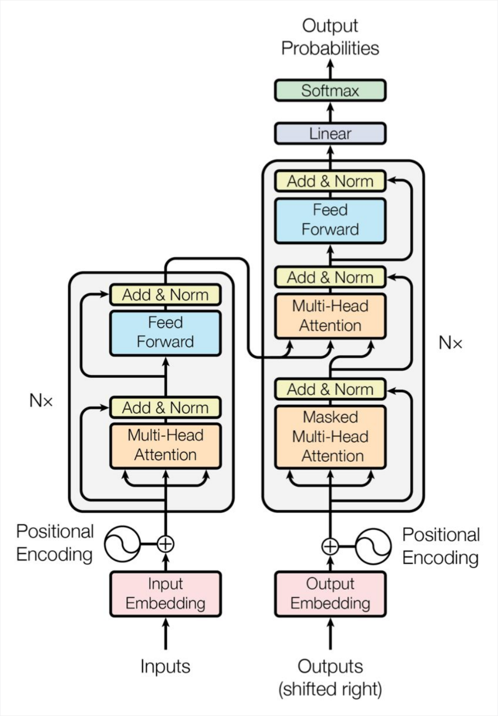
Vaswani et al. NeurIPS'17

(P4) `class TransformerEncoder`

The Transformer Architecture

Other Key Components

- Recurrent connections between sequential blocks to avoid vanishing gradients
- Positional encodings provide the model knowledge of the sequence structure
    - Without positional encodings, model treats natural language as a bag of words (BoW) instead of structured sequence
    - With positional encodings, model can learn in which cases local features are important and in which cases global information is important



Vaswani et al. NeurIPS'17

M | ROBOTICS

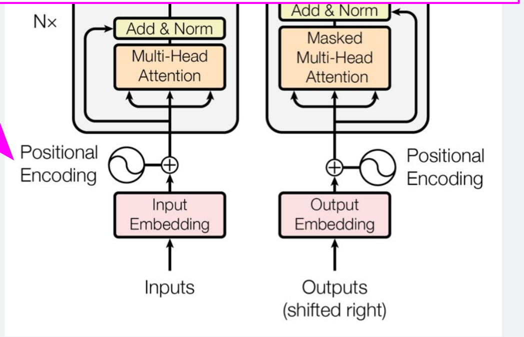
The Transformer Architecture

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

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

- Without positional encodings, model treats natural language as a bag of words (BoW) instead of structured sequence

- With positional encodings, model can learn in which cases local features are important and in which cases global information is important



Vaswani et al. NeurIPS’17

Recall: Three limitations of RNNs

1. Modeling long-range dependencies limited by vanishing gradient

2. Computational and memory efficiency, especially for long sequences

3. Parallelization of layers that depend on sequential information

How does Attention address the limitations of RNNs?

How does Attention Address the Limitations of RNNs

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Vaswani et al. NeurIPS'17

Limitation 1:

- Self-Attention ensures that each output token has access to all previous input tokens (path length of 1)
- Meanwhile for RNNs, the output at token N has a path length of N to interact with input token 1

How does Attention Address the Limitations of RNNs

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Vaswani et al. NeurIPS'17

Limitation 2:

- When sequence length (or restricted context) is much smaller than dimension, self-attention will be at least as efficient as RNN

How does Attention Address the Limitations of RNNs

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Vaswani et al. NeurIPS'17

Limitation 3:

- Self-attention uses constant number of sequential operations while RNN requires N sequential forward propagation steps to generate N-th output

Can we apply Transformers to Images?

Yes!

Idea: Treat the image as a set of patches of pixels

Vision Transformers

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], Lucas Beyer[*], Alexander Kolesnikov[*], Dirk Weissenborn[*], Xiaohua Zhai[*], Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

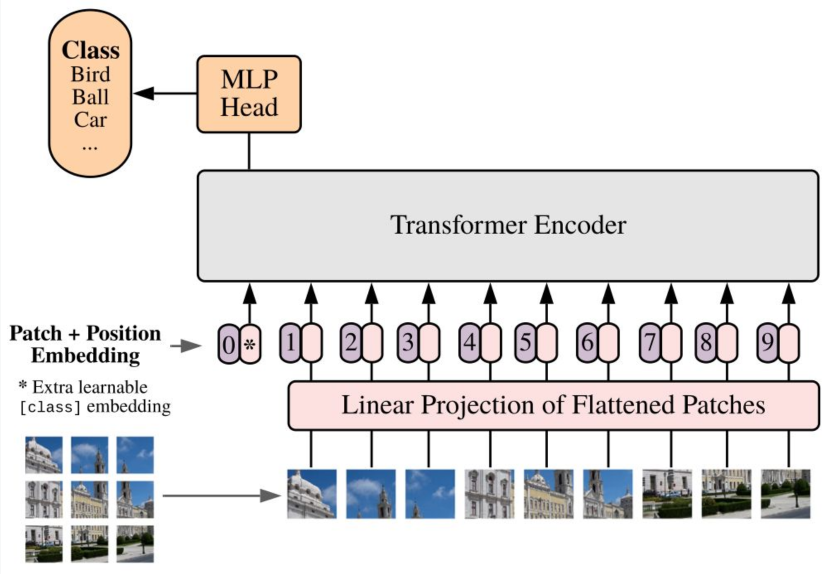
{adosovitskiy, neilhoulsby}@google.com

Dosovitskiy et al. ICLR'21

https://arxiv.org/pdf/2010.11929

Vision Transformers

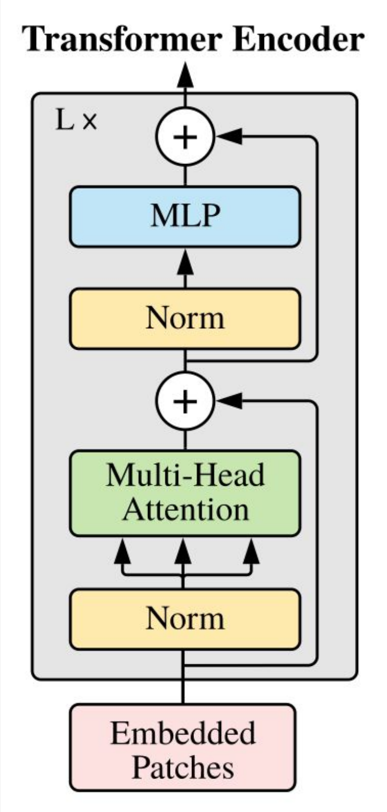
- Convert image into 16x16 patches
  - E.g. (1, 240, 240, 3) -> (1, 15x15, 16x16x3)
- Apply shared linear projection to each patch
  - E.g. (1, 15x15, 16x16x3) -> (1, 15x15, 64)
- Concatenate learnable class token for classifier output
  - E.g. (1, 1+15x15, 64)
- Add position embedding to each patch
  - E.g. (1, 1+15x15, 64) + (1, 1+15x15, 64)



(P4) patchify

Vision Transformer Encoder

- Based on the Transformer encoder
- Sequence of **LNorm->MHSA->LNorm->MLP** with residual skip connections
- For input embedded patches: (1, 1+15x15, D_in)
  - Output: (1, 1+15x15, D_out)
- For final classification decision:
  - Apply MLP and softmax to the class token
  - (1, 1, D_out) -> (1, 1, N_classes)



Dosovitskiy et al. ICLR'21

Reminder

- *P4 Due March 30, 2025*

- We will post a template for poster (“lightning talk”)
  Consider:
  - Topic/Title
  - Aim to reproduce/run 1-2 baseline method by then
  - MUST-DO Goals/Stretch Goals
  - Some Lit Review (“Gaps”)
  - Group member roles/sub-tasks
  - Timeline/Milestone
  - Material (if any)
  - *Bonus: for showcasing on a real robot!*

M | ROBOTICS