Recurrent Neural Networks

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

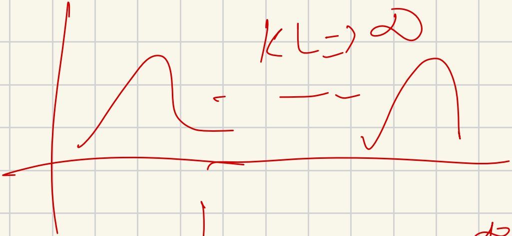
June 10, 2025

Table of Contents

$$P_x(x) = P_z(z) \prod_{i=1}^{N} D\, J(T_\theta(\cdot)^{-1})$$

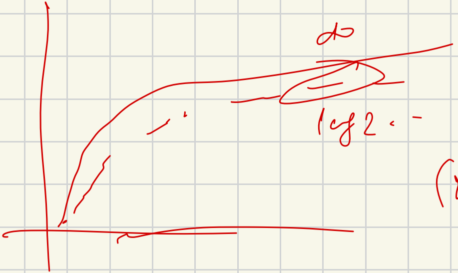
~~log 8~~

β

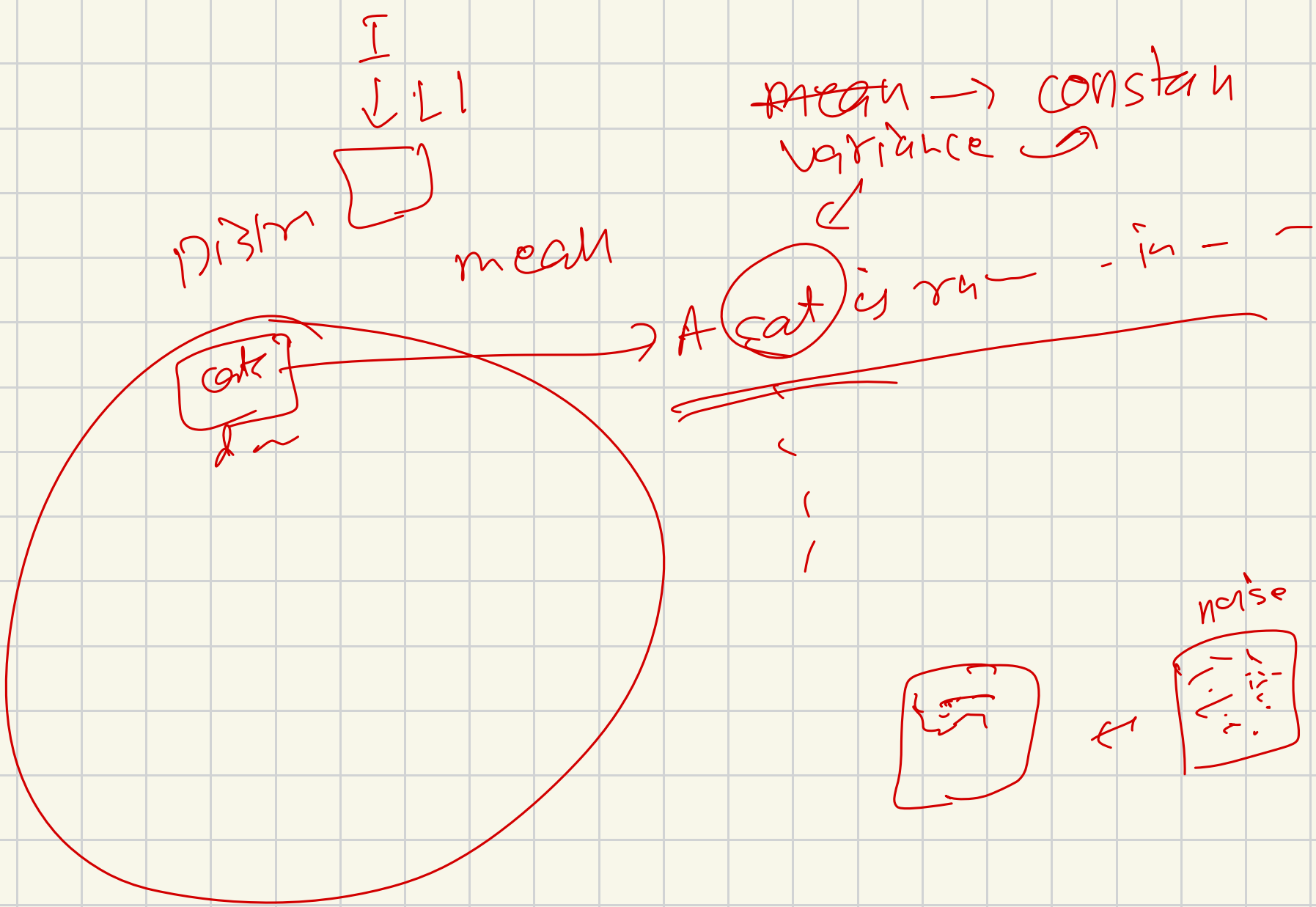


$\overline{KL}$

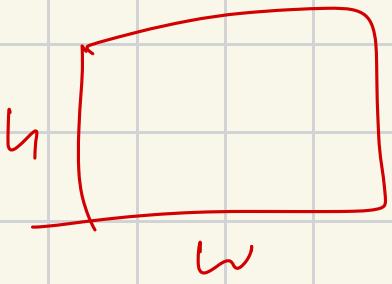
JSD



(noise - dnoise)


I
dish
meal
cat
meal → constant
variance
A cat is ran - in -
noise

CNNs

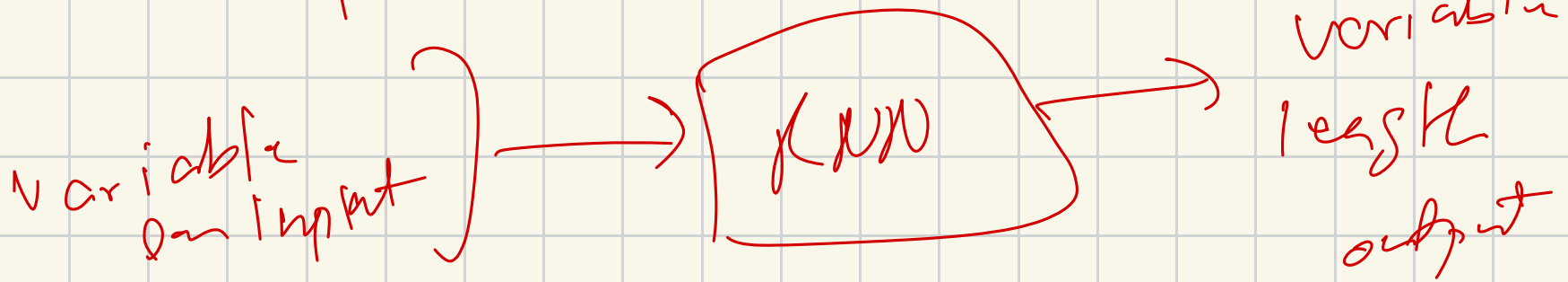


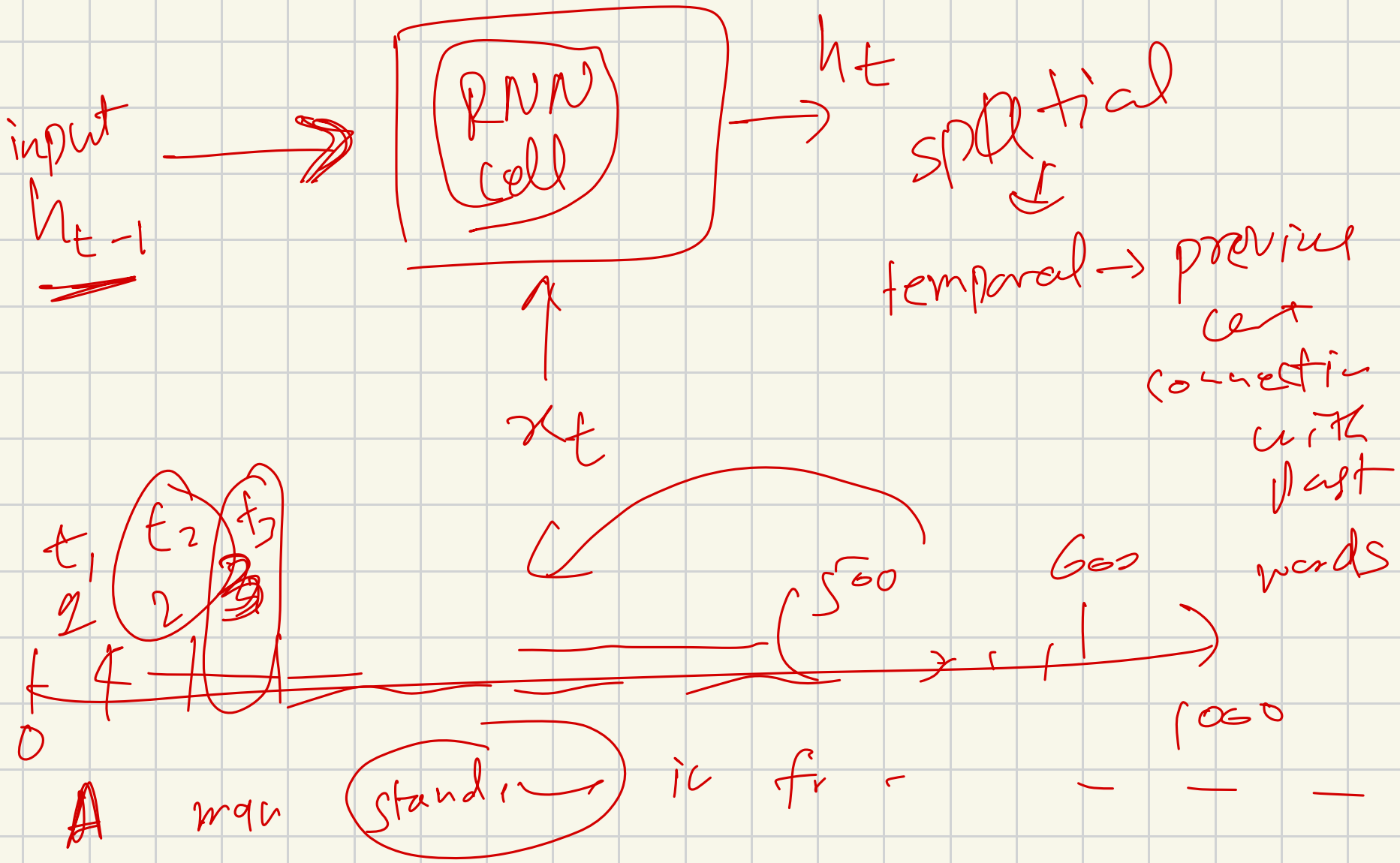
FCN/MLN

input neurons

nn.Linear (input-dim, output)

RNNs.

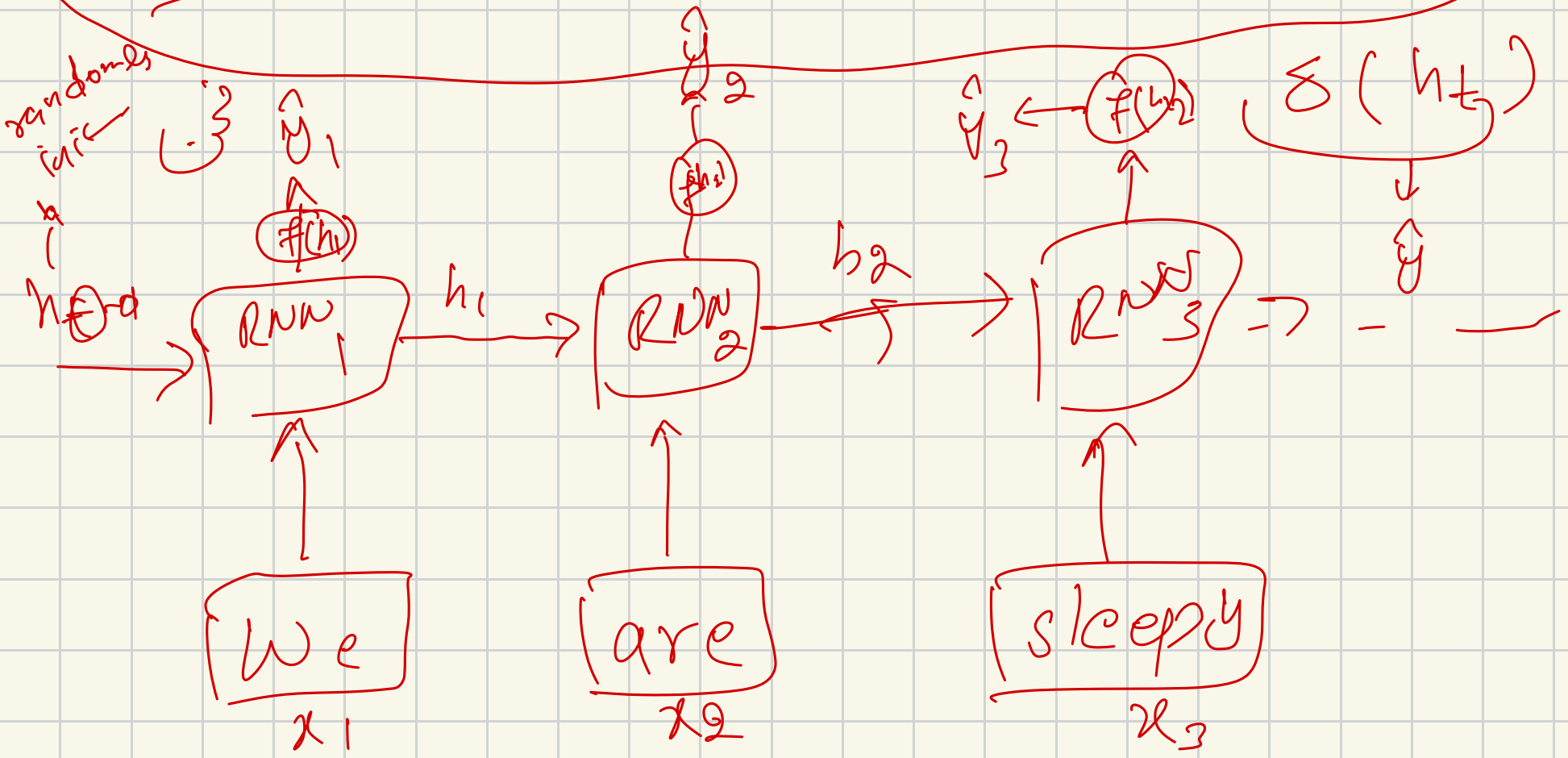




A man standing in fr - - - -

$$h_t = \text{function}(W_h h_{t-1}, W_x x_t + b)$$

$$h_t = \text{activation}(W_h h_{t-1} + W_x x_t + b)$$



Motivation

Recurrent Neural Networks (RNNs) address the limitations of traditional feedforward networks in processing sequential data such as text, speech, and time series. Their ability to capture temporal dependencies makes them essential for various applications:

- **Machine Translation:** Converting sentences between languages.
- **Time-Series Prediction:** Forecasting stock prices using GRU/LSTM architectures.
- **Image Captioning:** Generating textual descriptions from images using CNN-RNN hybrids.

Learning Outcomes

By the end of this section, you will be able to:

- Understand the motivation behind Recurrent Neural Networks (RNNs).
- Implement RNNs for sequence modeling tasks.
- Apply attention mechanisms to improve long-term dependency handling.
- Appreciate the significance of RNNs in advancing machine learning capabilities.

RNNs: Introduction

Definition: Recurrent Neural Networks (RNNs) are neural networks with looping mechanisms that allow them to retain memory of previous inputs.

Core Components:

- **Hidden state:** Represents context from prior time steps.
- **Recurrence relation:** Updates the hidden state using the current input and the previous state:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b)$$

where h_t is the hidden state at time t, x_t is the input at time t, W_h and W_x are weight matrices, b is the bias, and σ is an activation function.

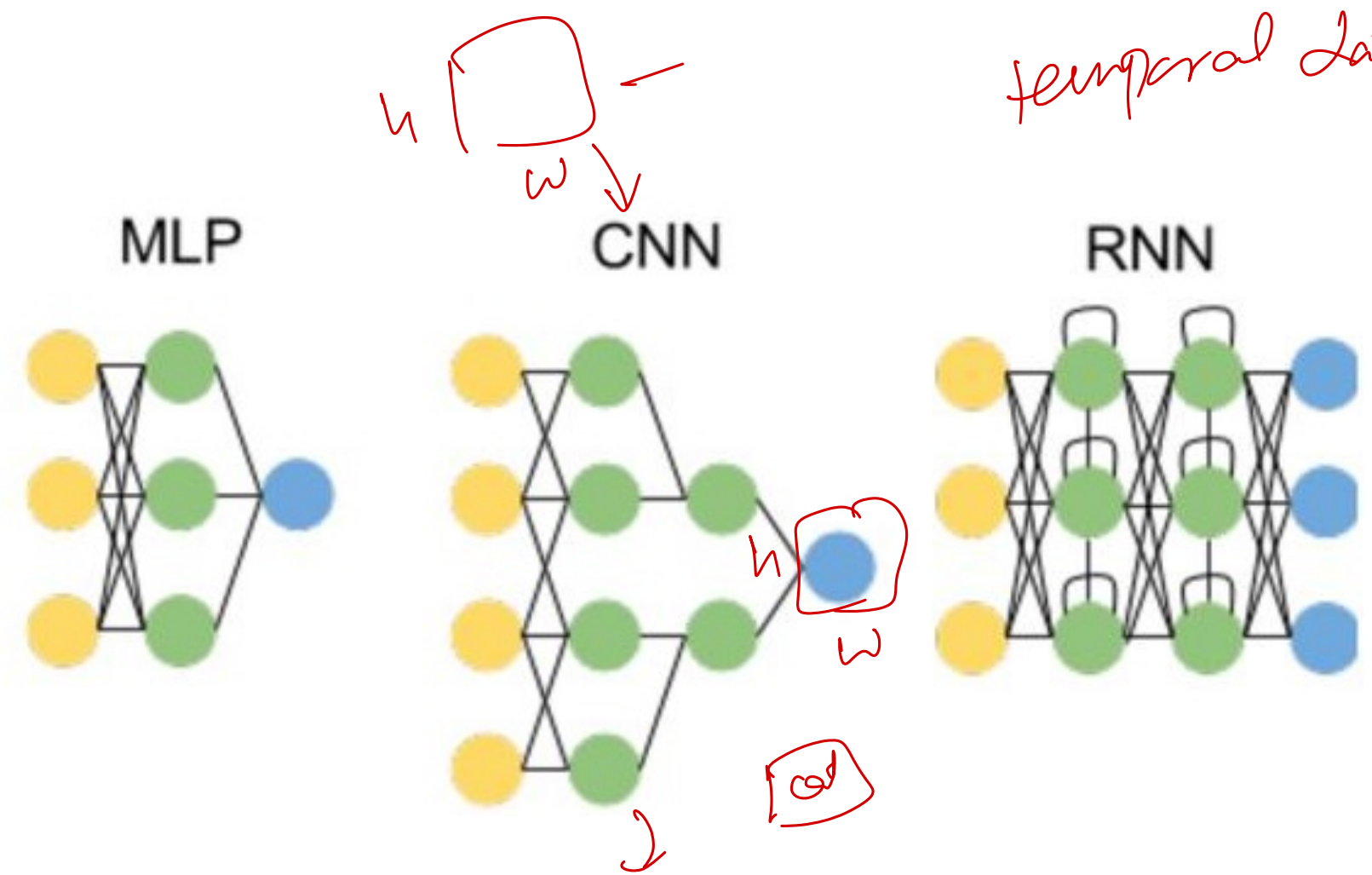
Applications: Sentiment analysis, speech recognition, DNA sequencing, and more.

RNNs: Introduction (cont.)

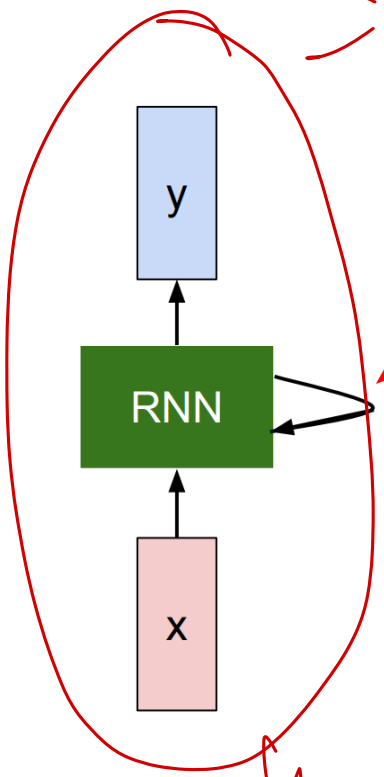
RNN Step-by-Step:

- **Unfolding:** Processes sequences step-by-step, sharing weights across time.
  - **Example:** Predicting the next word in "Apple is __" using stored context.
- **Training:**
  - **Backpropagation Through Time (BPTT):** Adjusts weights by unrolling the network over time.
  - **Challenges:** Vanishing/exploding gradients due to long sequences.

Neural Network Types



Recurrent Neural Networks (RNN)



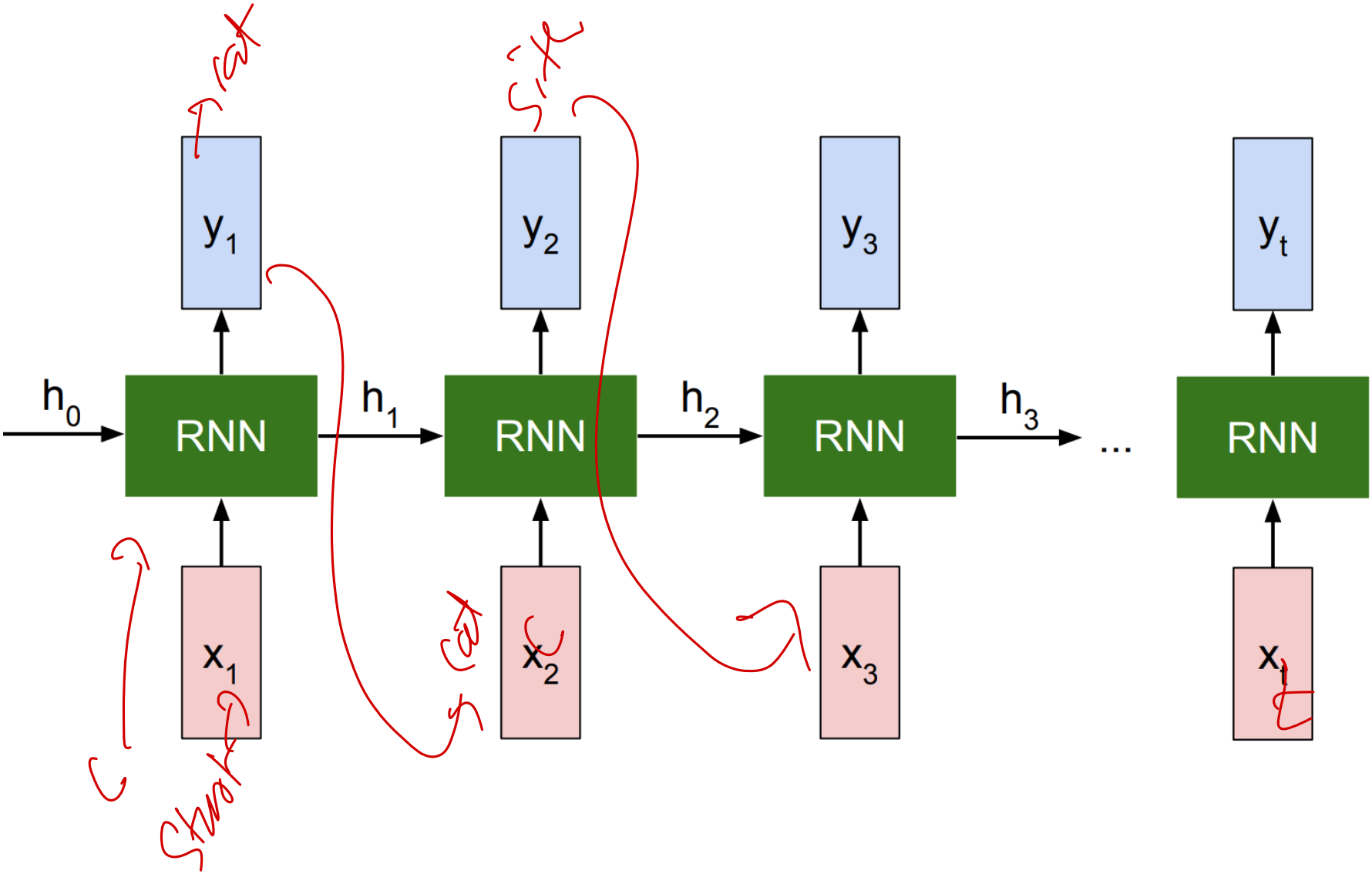
Key idea: RNNs have an "internal state" that is updated as a sequence is processed

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

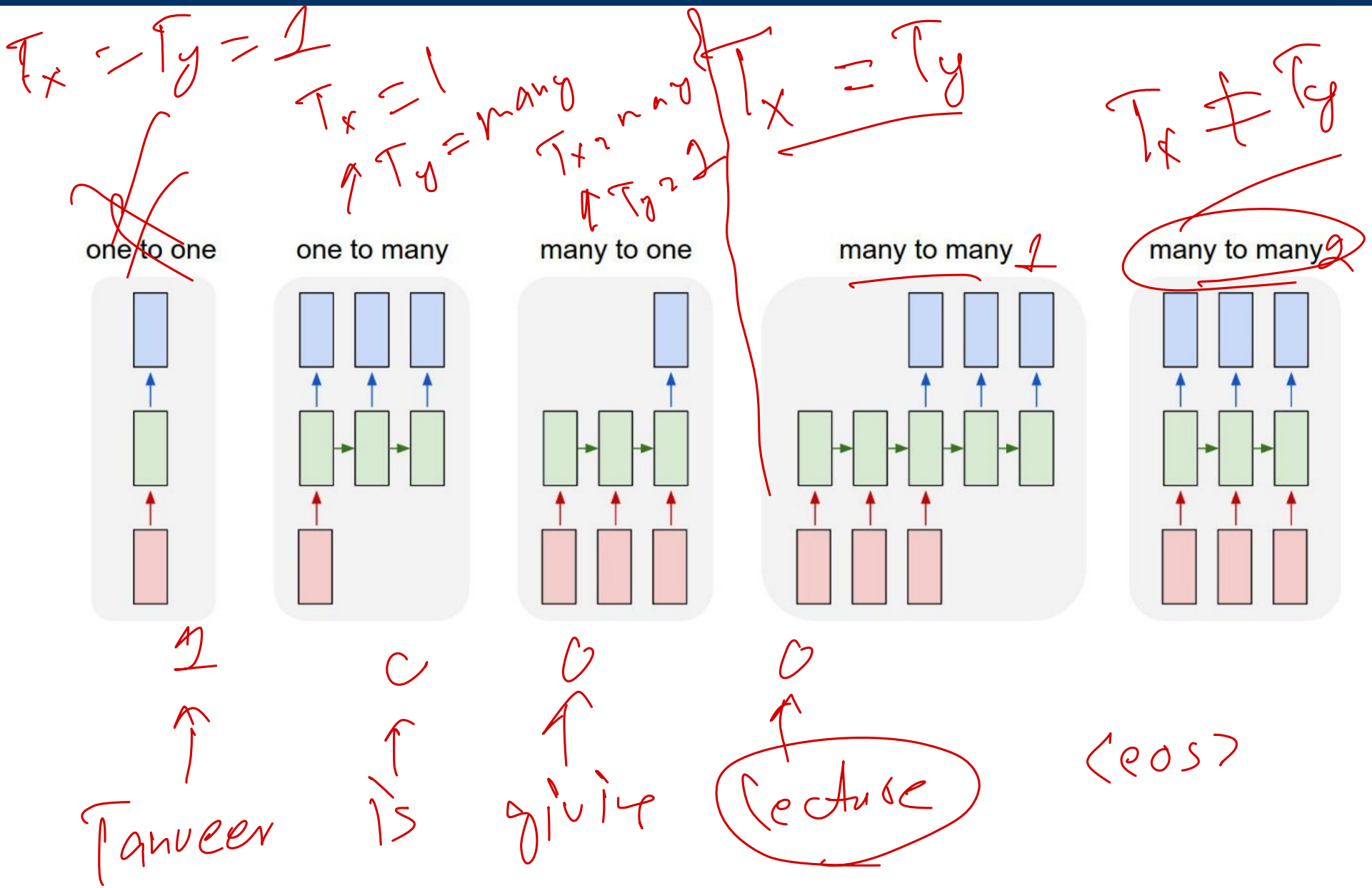
new state | old state | input vector at some time step

some function with parameters W

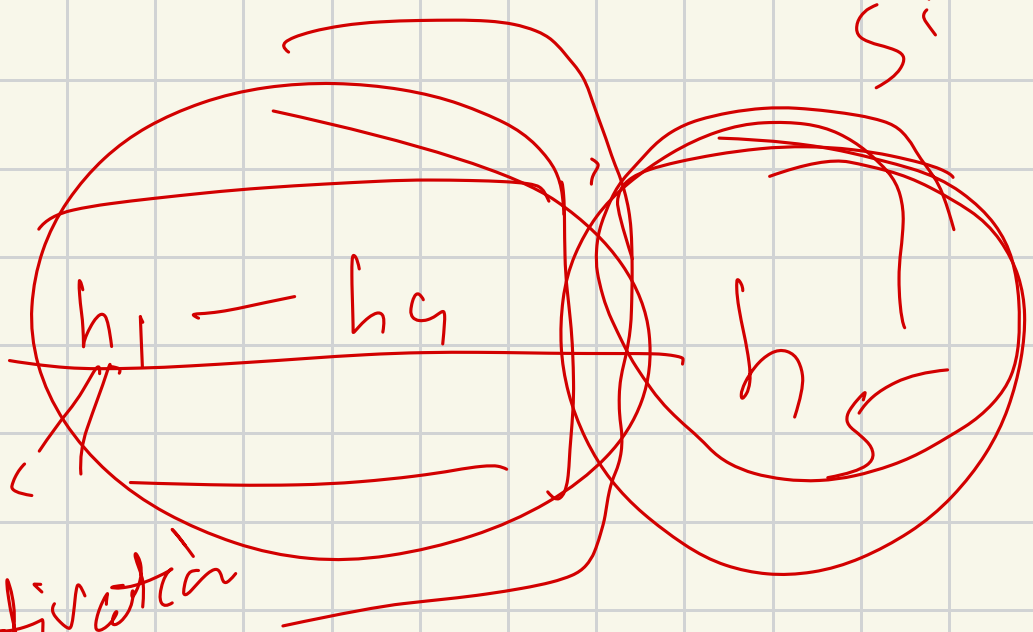
Recurrent Neural Networks (RNN) (cont.)



Recurrent Neural Networks (RNN) (cont.)



$f(y_t \mid y_{t-1} \cdots y_0)$

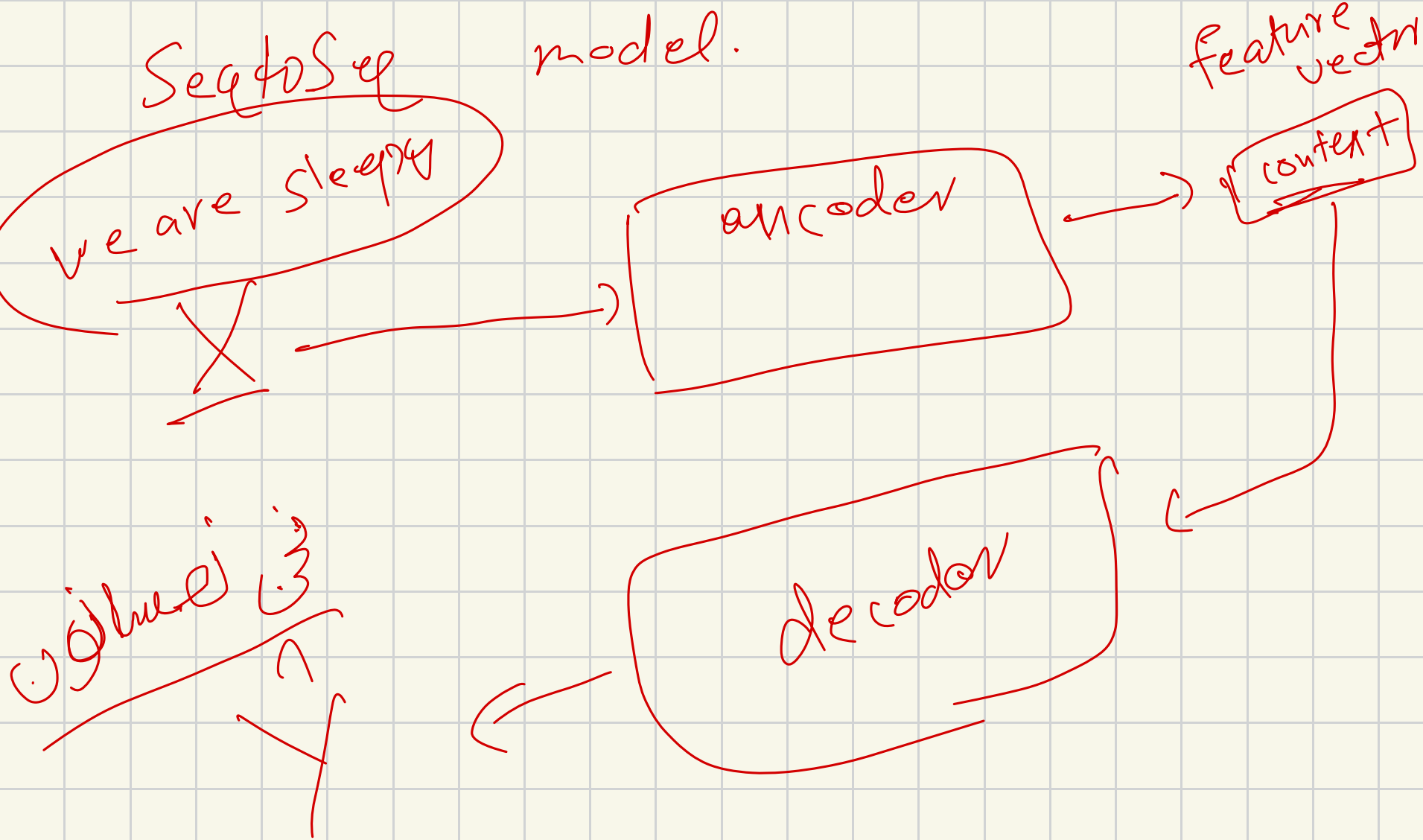


Si

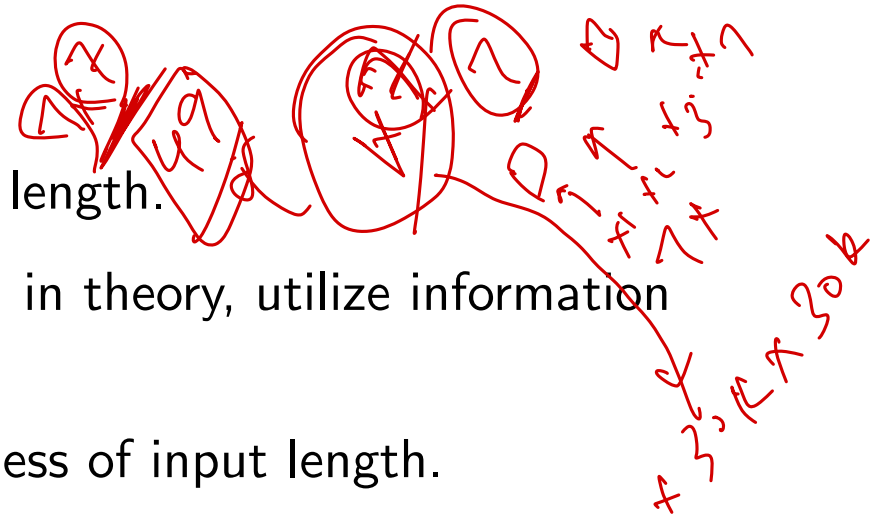
h_{co}

Machine translation.

Seq to Seq model.

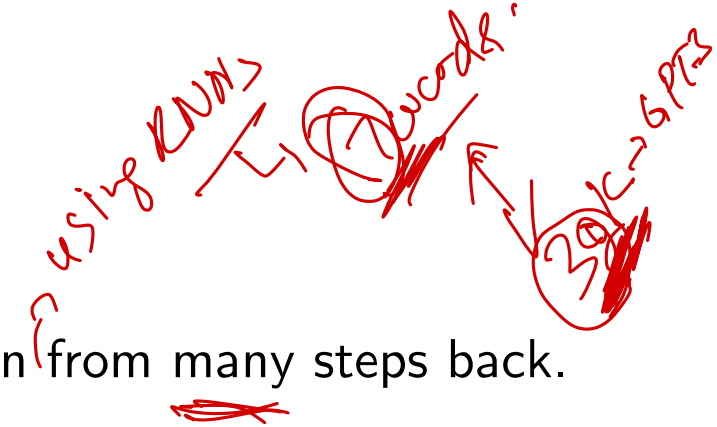


Recurrent Neural Networks (RNN) (cont.)



RNN Advantages:

- Can process input sequences of any length.
- Computation at each time step can, in theory, utilize information from many previous steps.
- Model size remains constant regardless of input length.
- The same weights are applied at every time step, ensuring symmetry in input processing.



RNN Disadvantages:

- Recurrent computation is slow.
- In practice, difficult to access information from many steps back.

Image Captioning

A computer Vision task in which we generate captions for images.

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

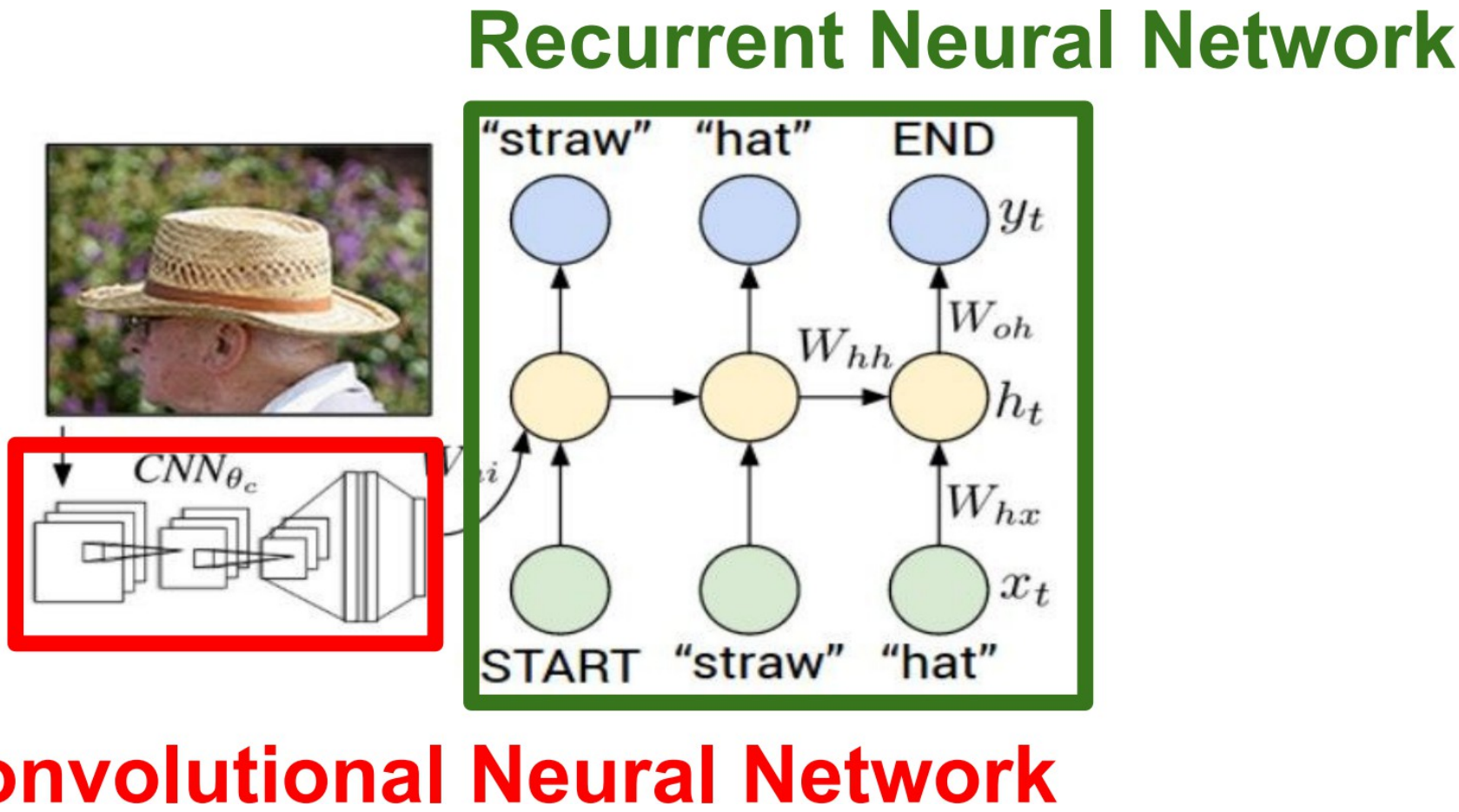
Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

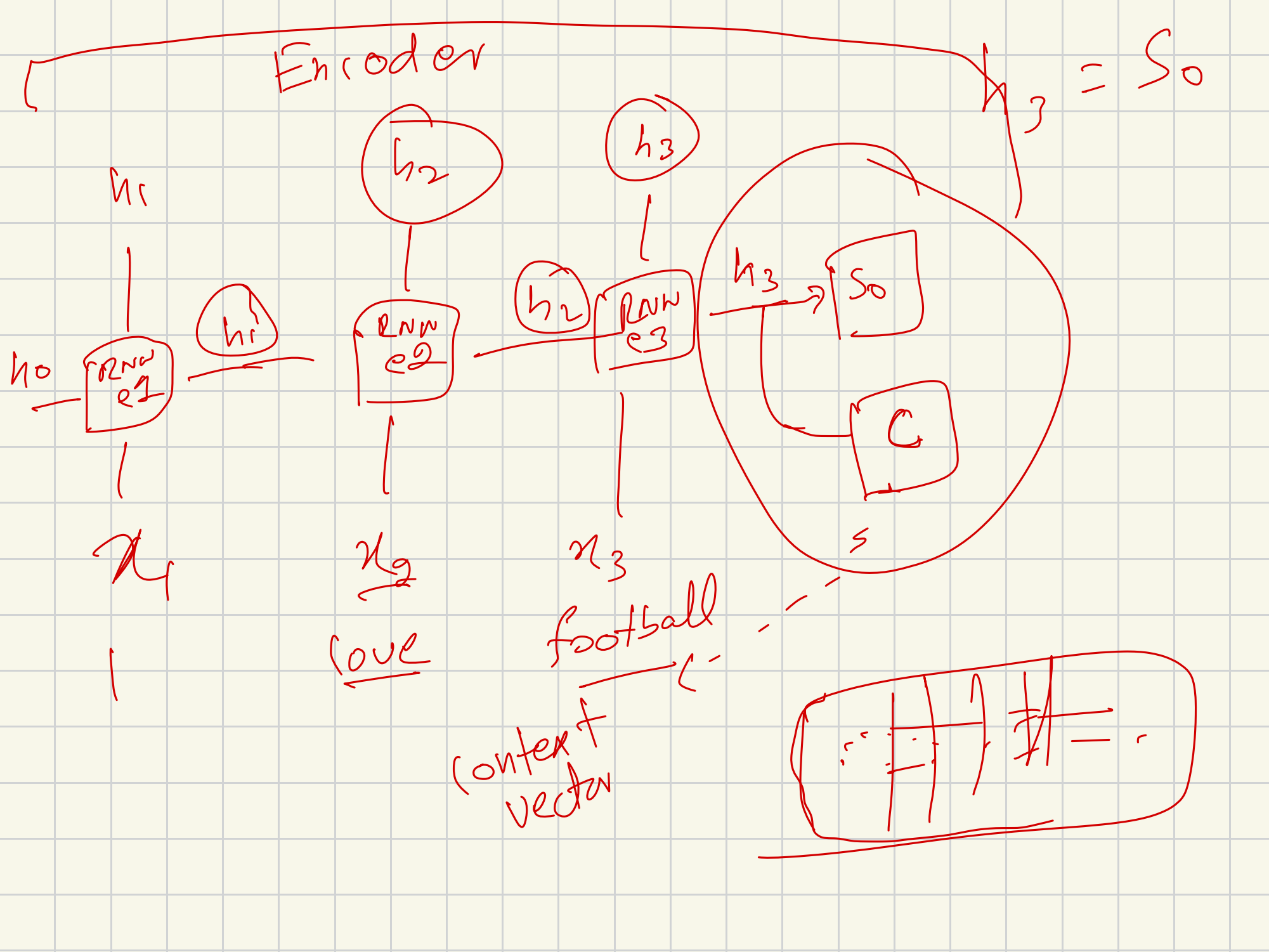
A man riding a dirt bike on a dirt track

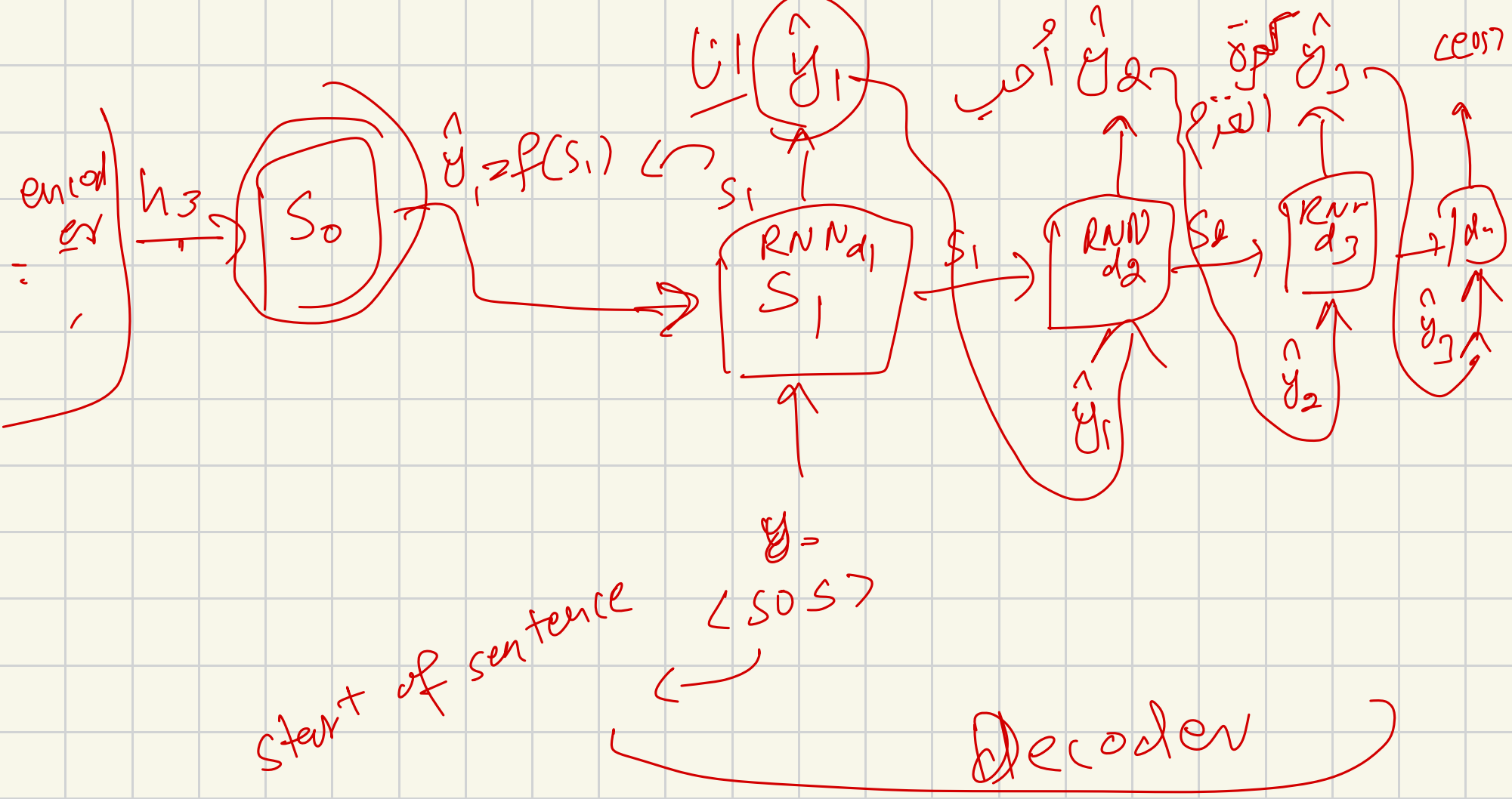
Image Captioning (cont.)



Sequence to Sequence with RNNs

- **Encoder-Decoder Framework:**
  - The encoder processes the input sequence and compresses it into a fixed-length context vector.
  - This context vector summarizes all relevant information from the input.
- **Decoder:**
  - The decoder takes the context vector and generates the output sequence step by step.
  - Each output token is produced based on the context vector and previously generated tokens.
- **With Attention Mechanism:**
  - Instead of relying on a single static context vector, the model computes a dynamic, attention-based context at each decoding step.
  - This allows the decoder to focus on different parts of the input sequence as needed, improving performance on longer or more complex sequences.


Encoder
$h_3 = S_0$
h_1
h_2
h_3
h_0
RNN e1
RNN e2
RNN e3
h_1
h_2
h_3
S_0
C
S
x_1
x_2
x_3
love
football
context vector


encoder
h_3
S_0
$\hat{y}_1 = f(S_1)$
S_1
RNN d_1
S_1
$\hat{y}_1$
$\hat{y}_2$
S_1
RNN d_2
S_2
RNN d_3
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
cross
$\hat{y}_3$
$y =$
<SOS>
start of sentence
Decoder

Sequence to Sequence with RNNs (cont.)

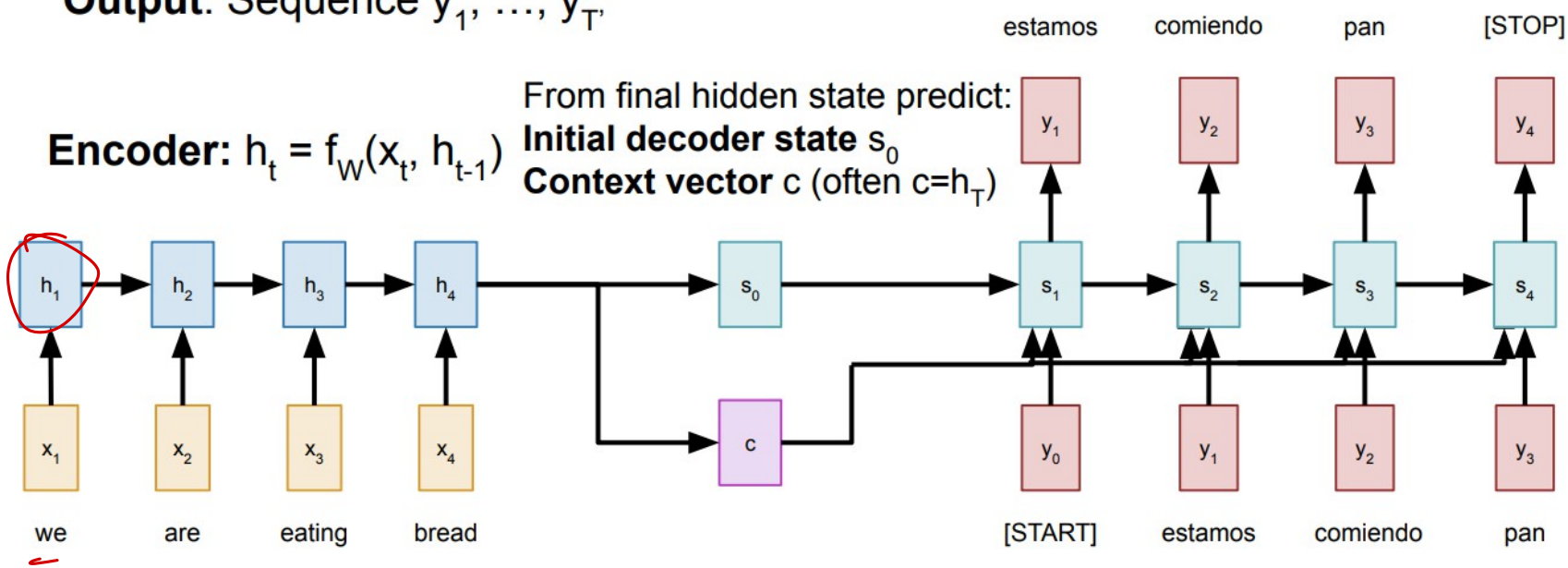
$$h_t = f_a(W_h h_{t-1} + W_x x_t + b)$$

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

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

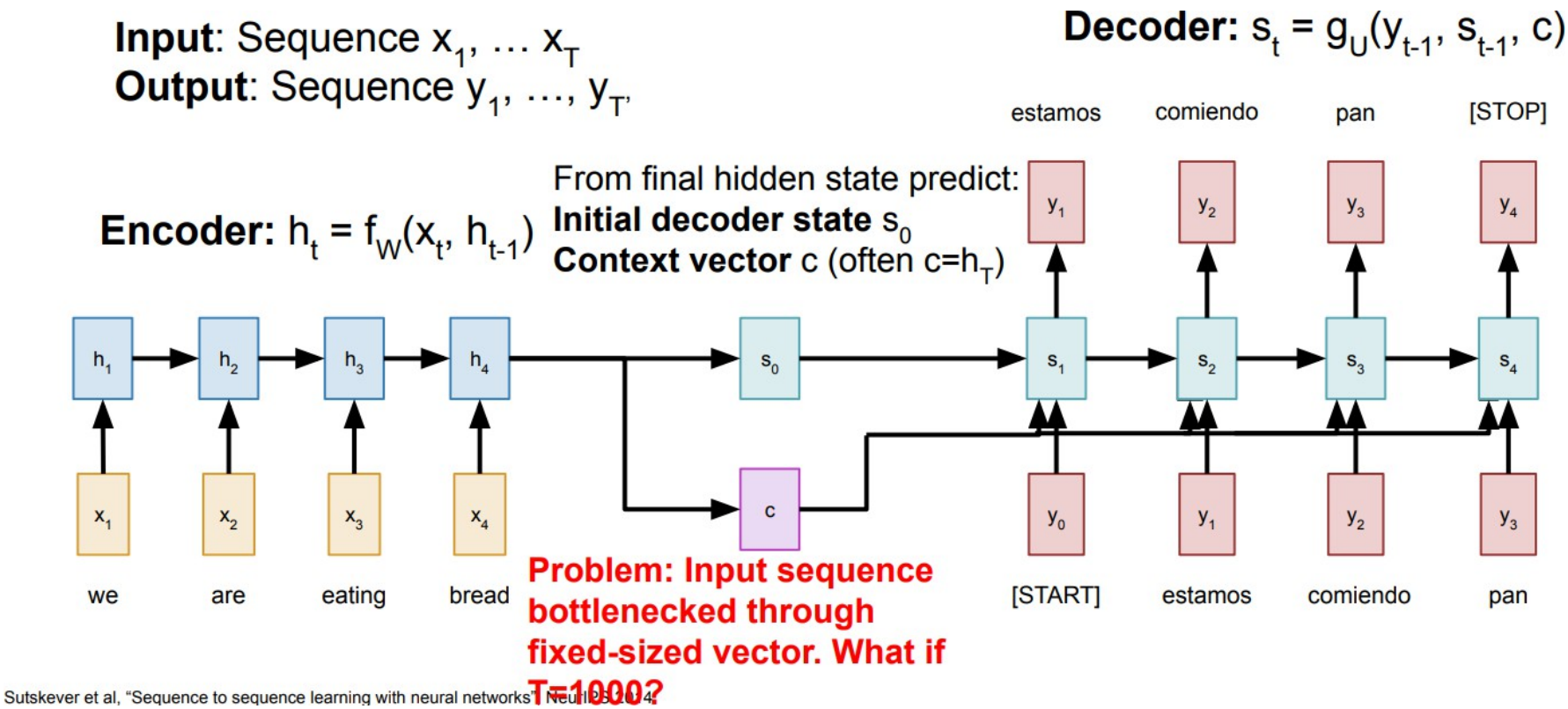
From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c=h_T$)

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



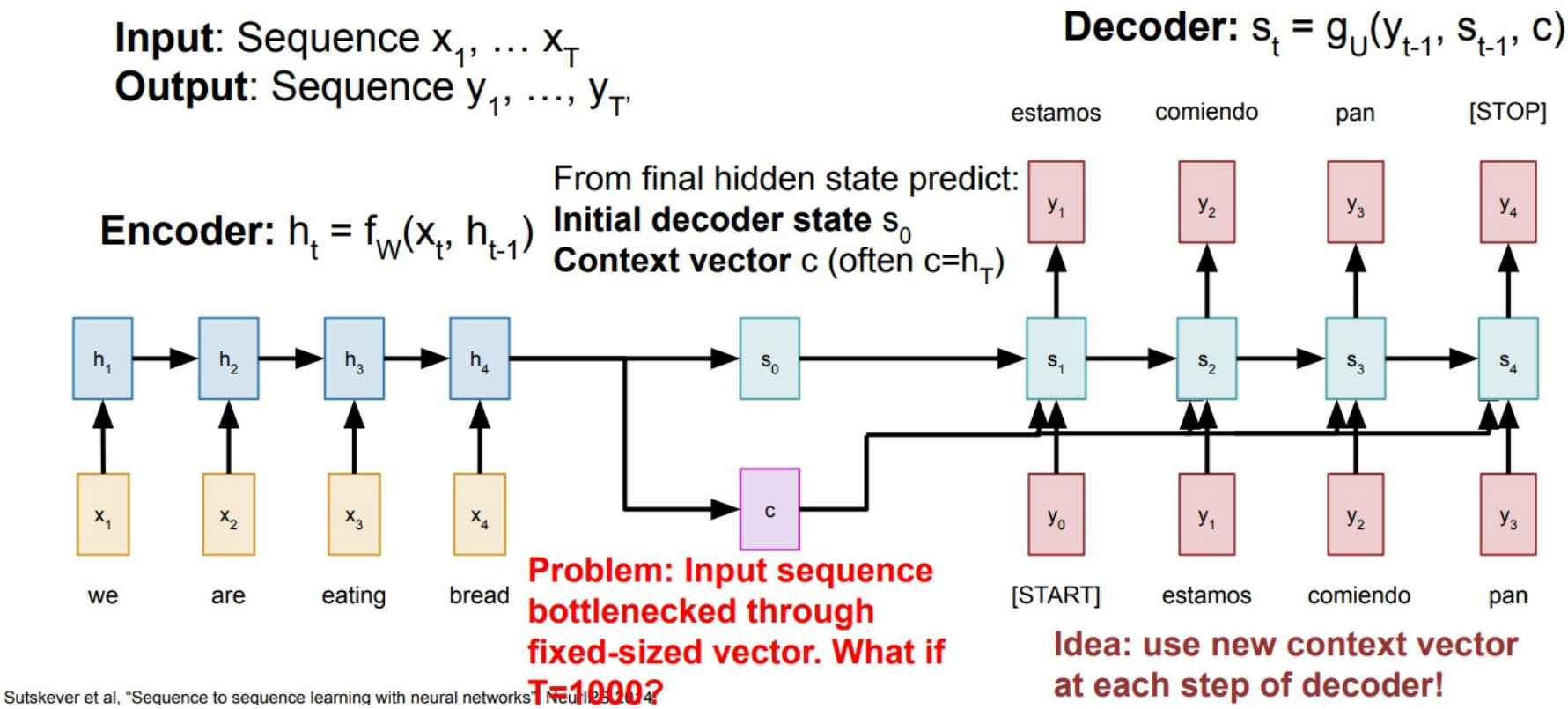
Sutskever et al, "Sequence to sequence learning with neural networks

Sequence to Sequence with RNNs (cont.)



Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Sequence to Sequence with RNNs (cont.)



Sutskever et al, "Sequence to sequence learning with neural networks", NeurIPS 2014

Sequence to Sequence based on sentence length

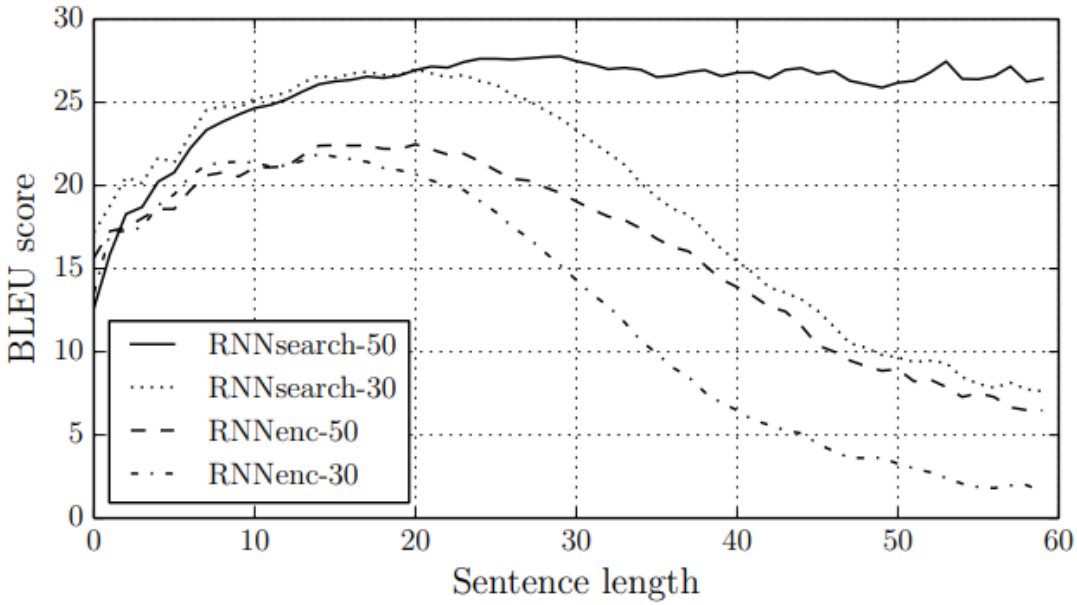


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Sequence to Sequence Calculating H states

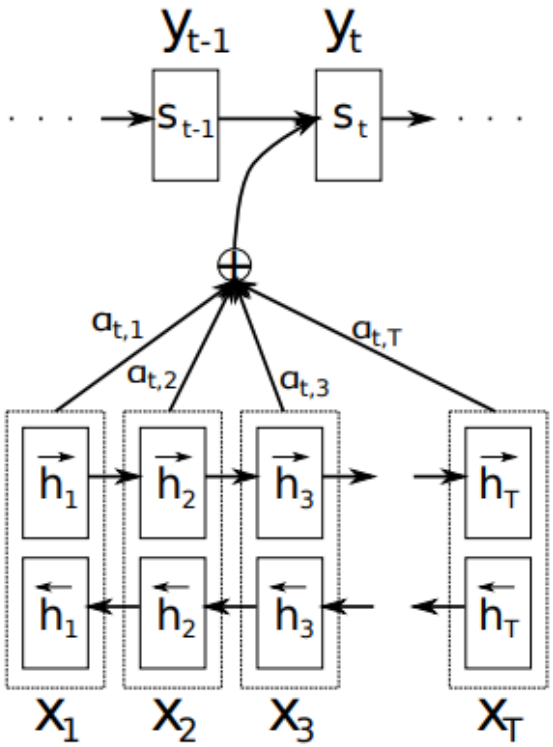
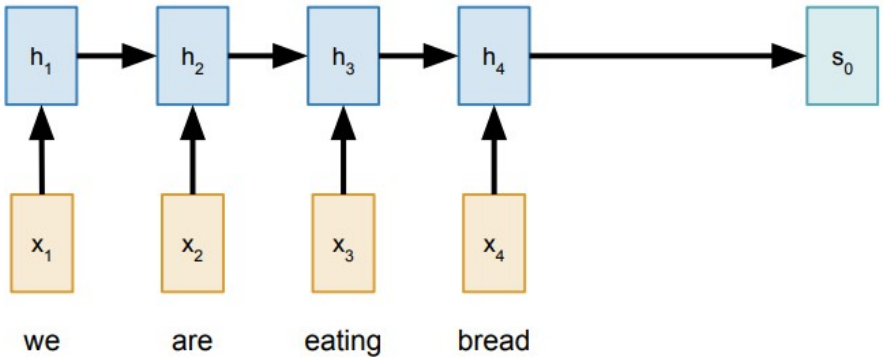


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Sequence to Sequence with RNNs and Attention

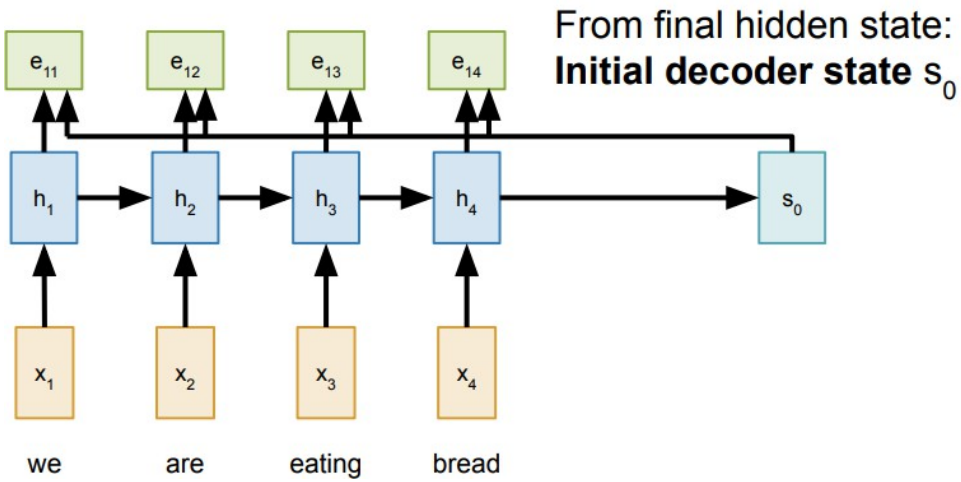
Input: Sequence $x_1, \ldots x_T$
Output: Sequence $y_1, \ldots, y_{T'}$

Encoder: $h_t = f_W(x_t, h_{t-1})$ From final hidden state: **Initial decoder state** s_0

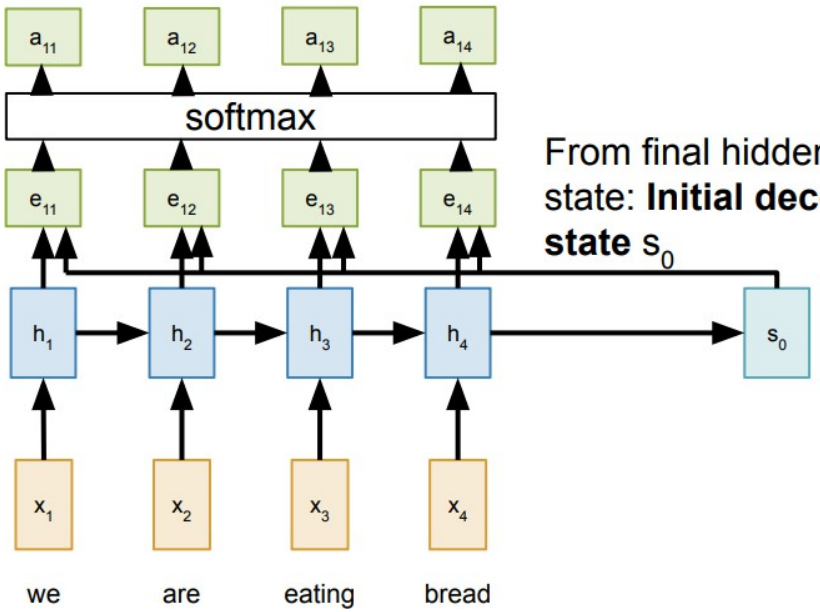


Sequence to Sequence with RNNs and Attention (cont.)

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



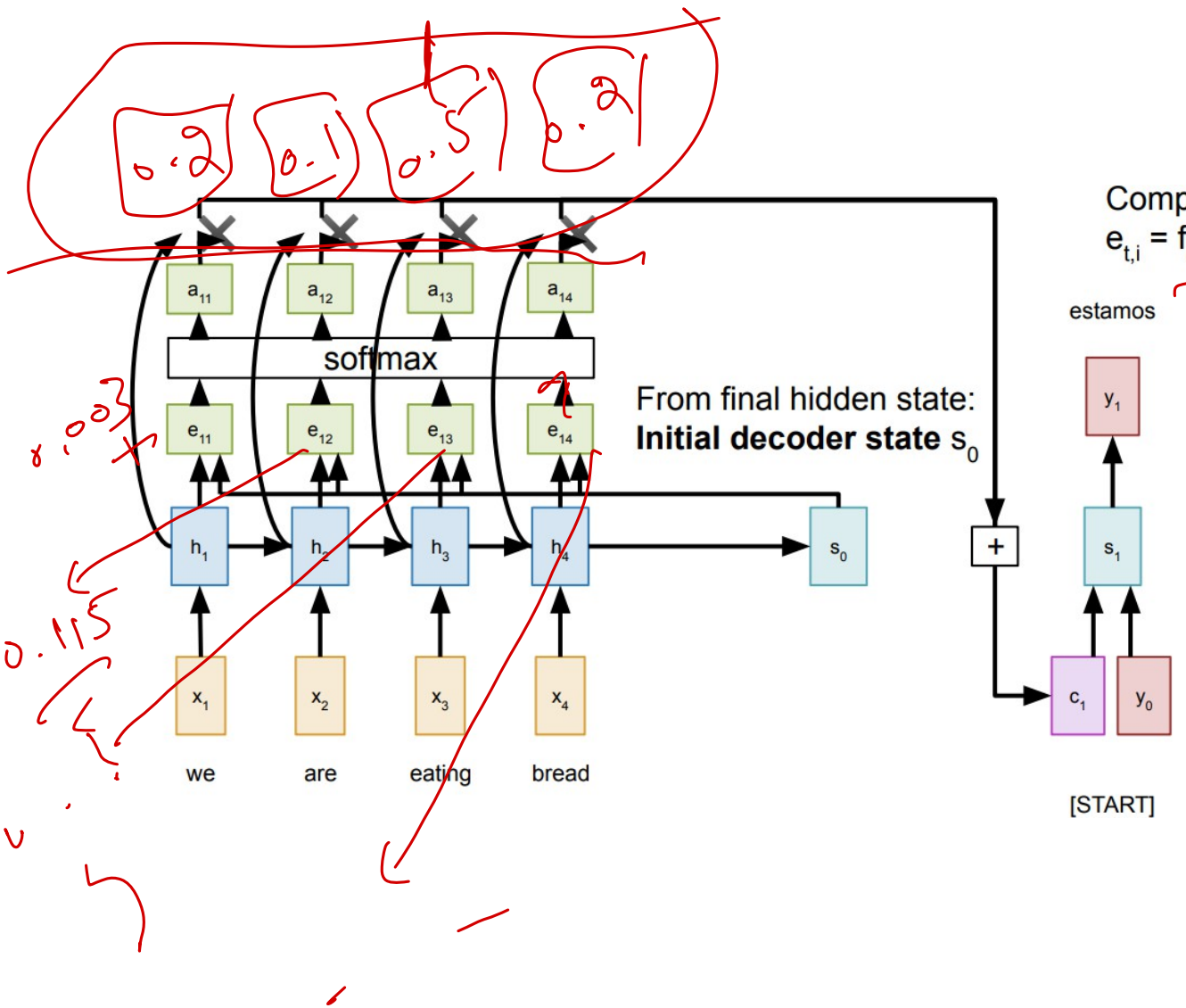
Sequence to Sequence with RNNs and Attention (cont.)



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**
$0 < a_{t,i} < 1 \quad \sum_i a_{t,i} = 1$

Sequence to Sequence with RNNs and Attention (cont.)



Compute (scalar) **alignment scores**

$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

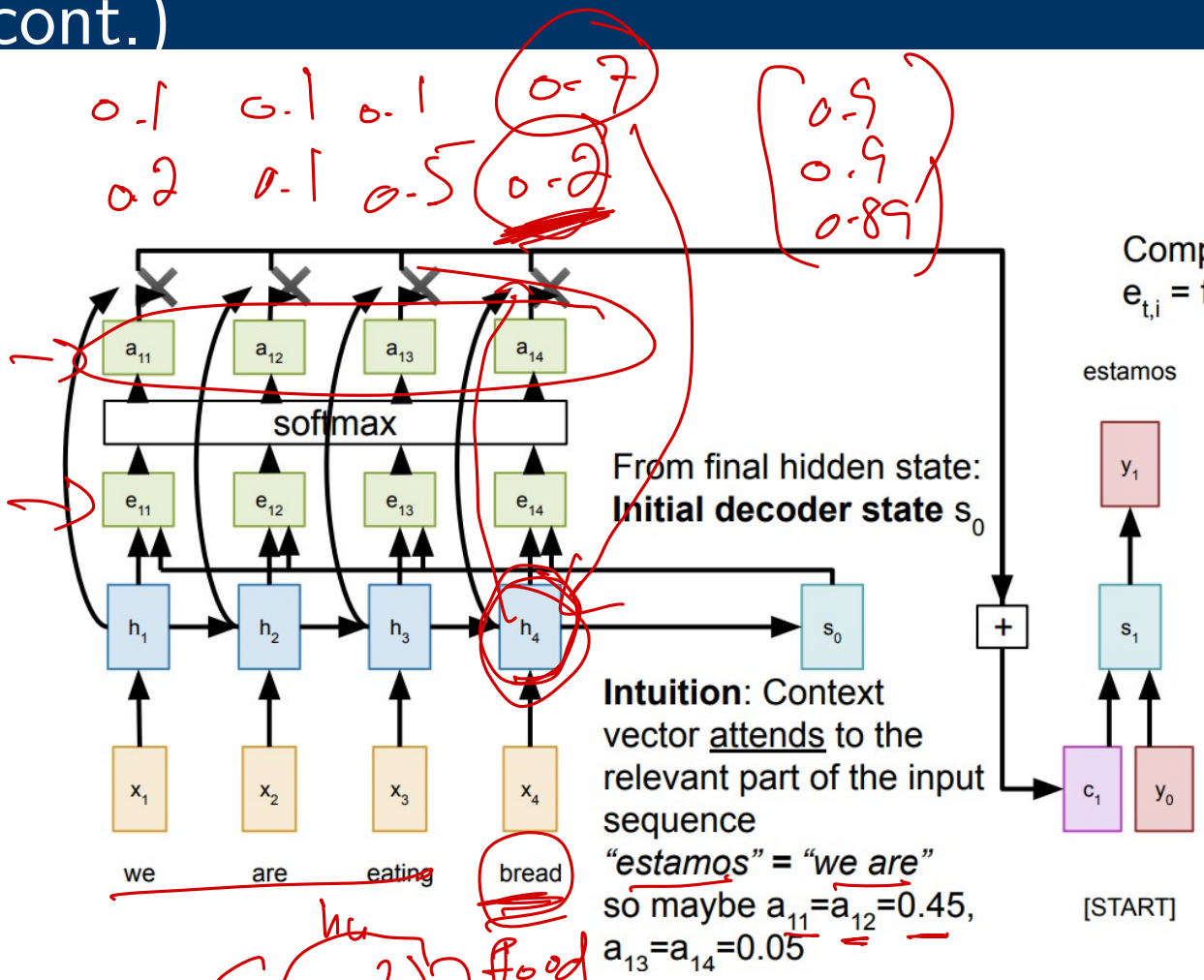
Normalize alignment scores to get **attention weights**

$0 < a_{t,i} < 1 \quad \sum_i a_{t,i} = 1$

Compute context vector as linear combination of hidden states

$c_t = \sum_i a_{t,i} h_i$

Sequence to Sequence with RNNs and Attention (cont.)



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

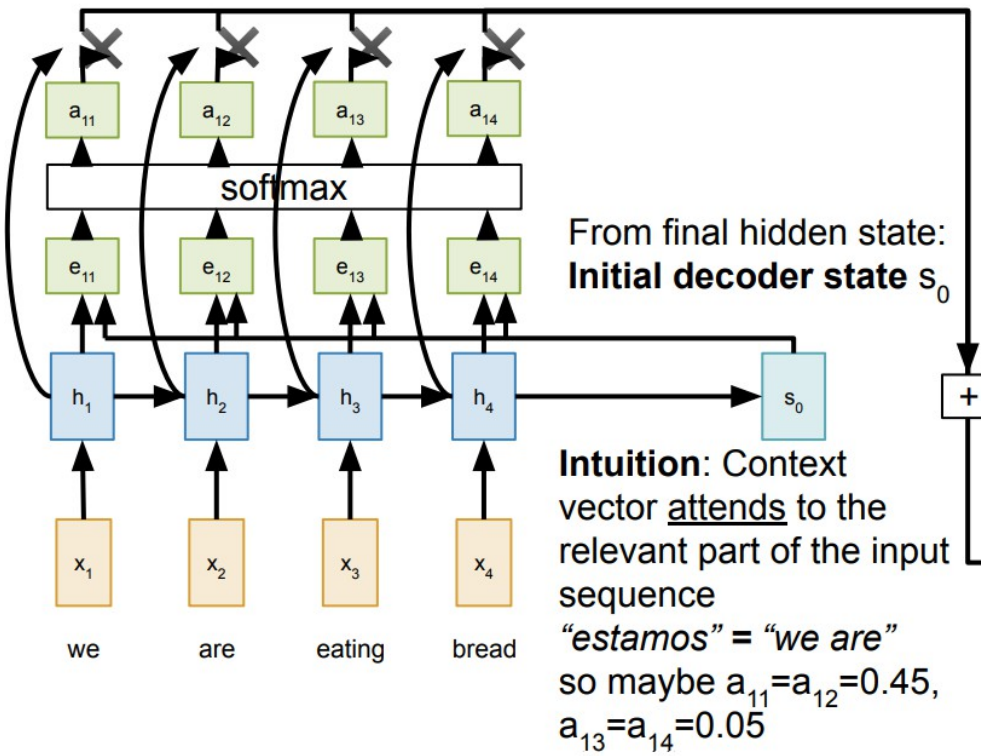
Compute (scalar) **alignment scores**
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ (f_{att} is an MLP)

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

Sequence to Sequence with RNNs and Attention (cont.)



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

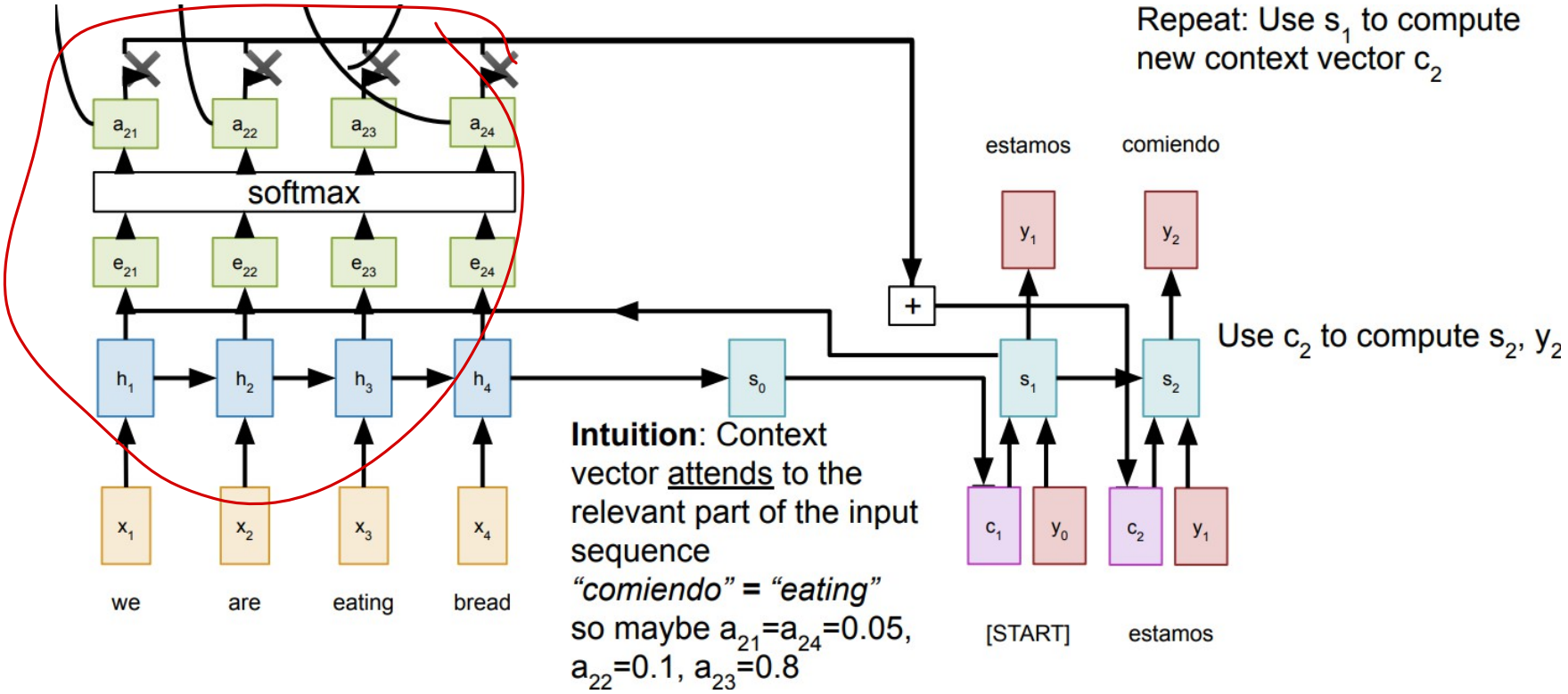
Normalize alignment scores to get **attention weights**
$0 < a_{t,i} < 1 \quad \sum_i a_{t,i} = 1$

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

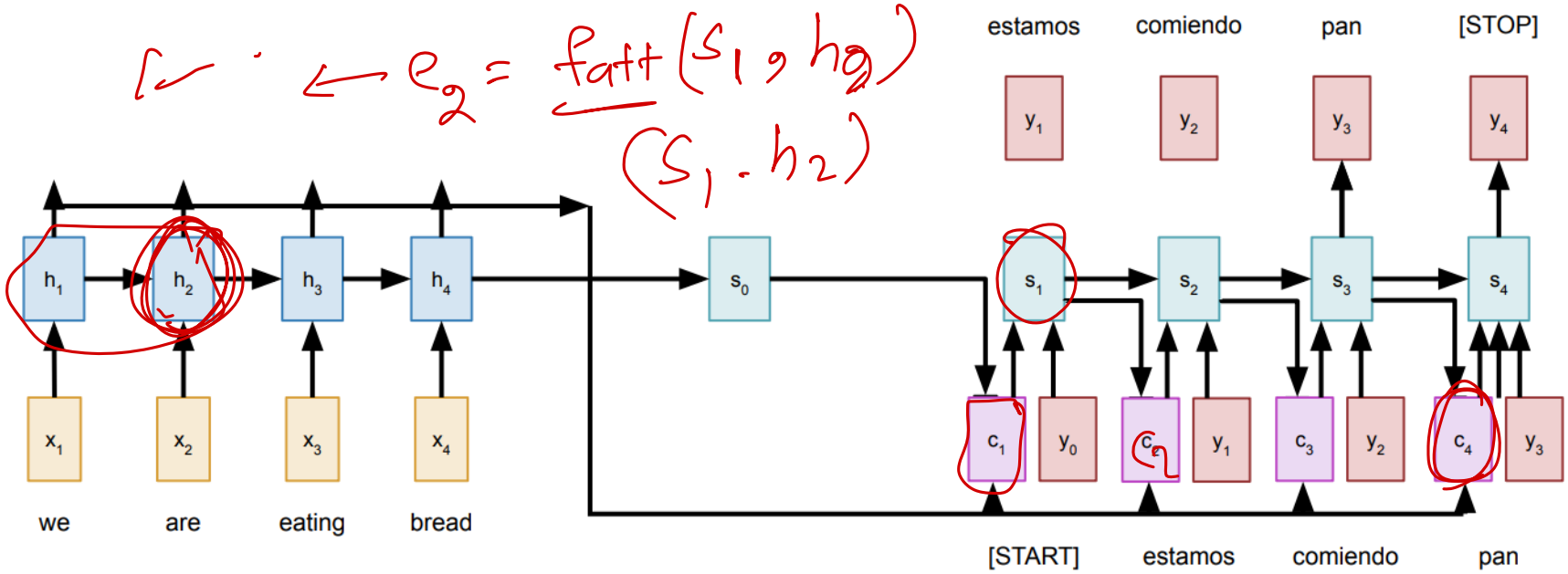
Use context vector in decoder: $s_t = g_U(y_{t-1}, s_{t-1}, c_t)$

This is all differentiable! No supervision on attention weights – backprop through everything

Sequence to Sequence with RNNs and Attention (cont.)



- **Dynamic Context Vector at Each Decoder Timestep:**
  - Instead of using a single, fixed context vector for all decoding steps, a different context vector is computed at each timestep of the decoder.
  - This approach prevents the input sequence from being bottlenecked through a single vector, allowing richer information flow.
  - At every decoding step, the context vector dynamically attends to different parts of the input sequence, enabling the decoder to focus on the most relevant information for generating each output token.



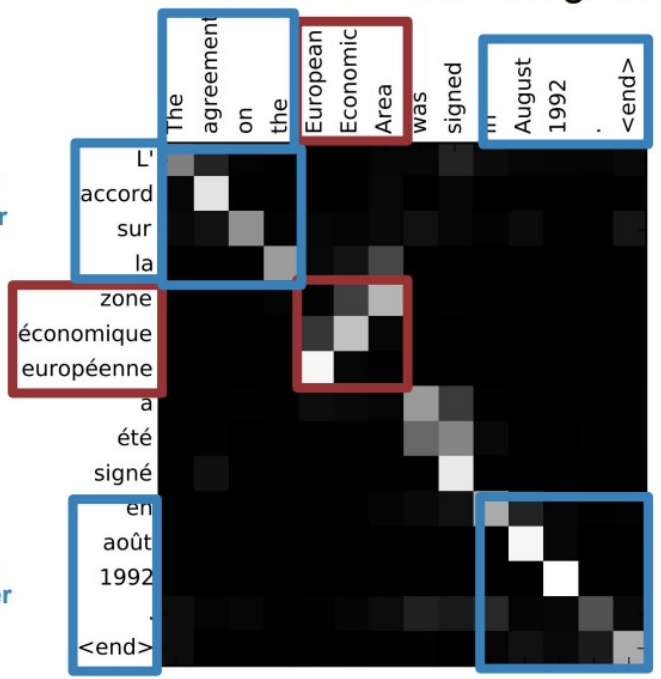
Sequence to Sequence with RNNs and Attention (cont.)

Example: English to French translation

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

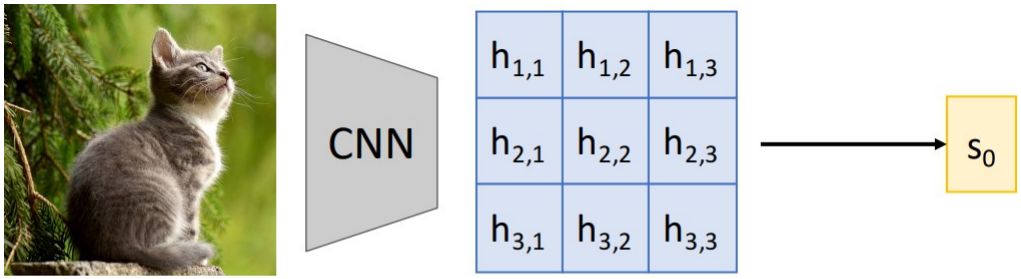
Output: "L'accord sur la zone économique européenne a été signé en août 1992."

Visualize attention weights $a_{t,i}$



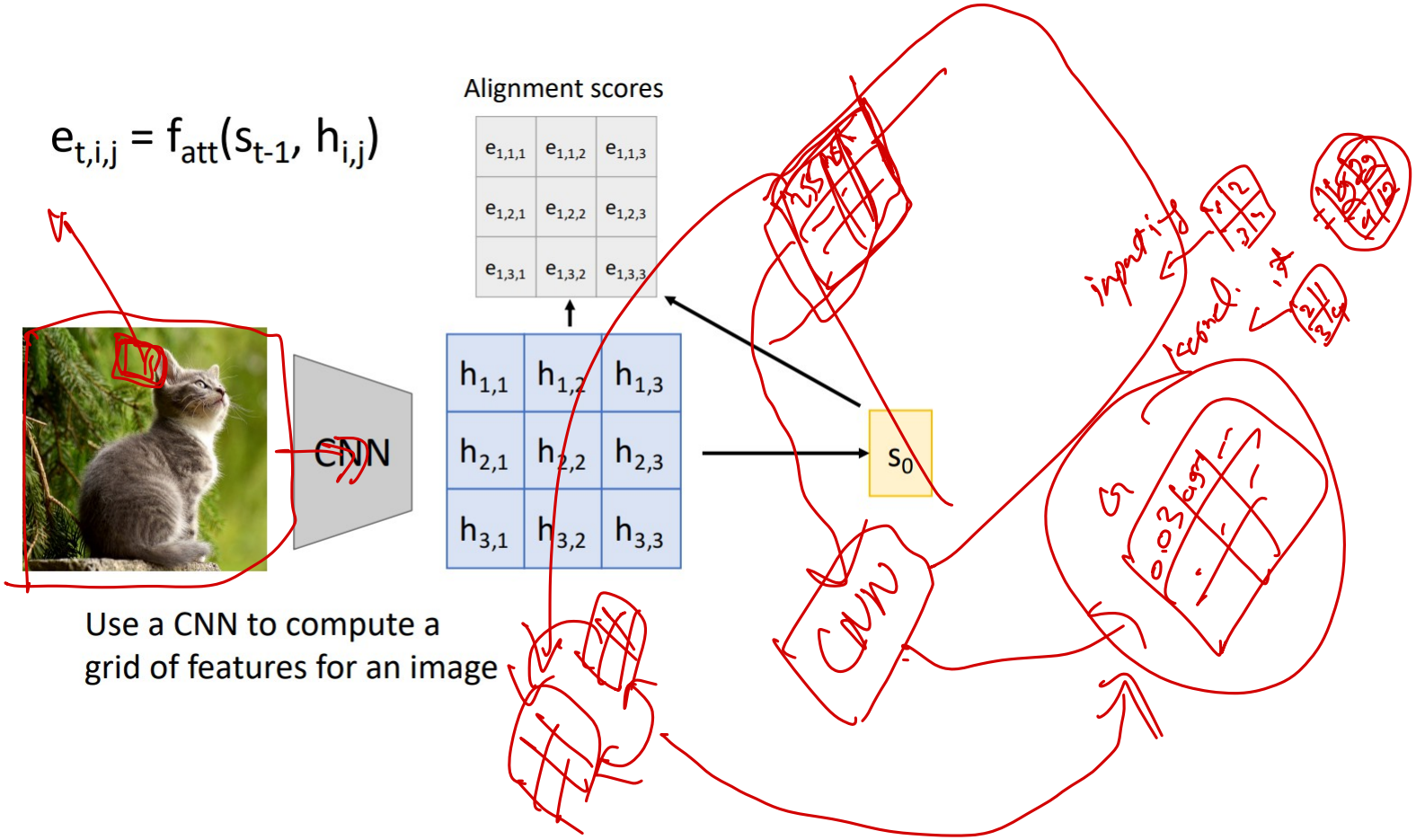
Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

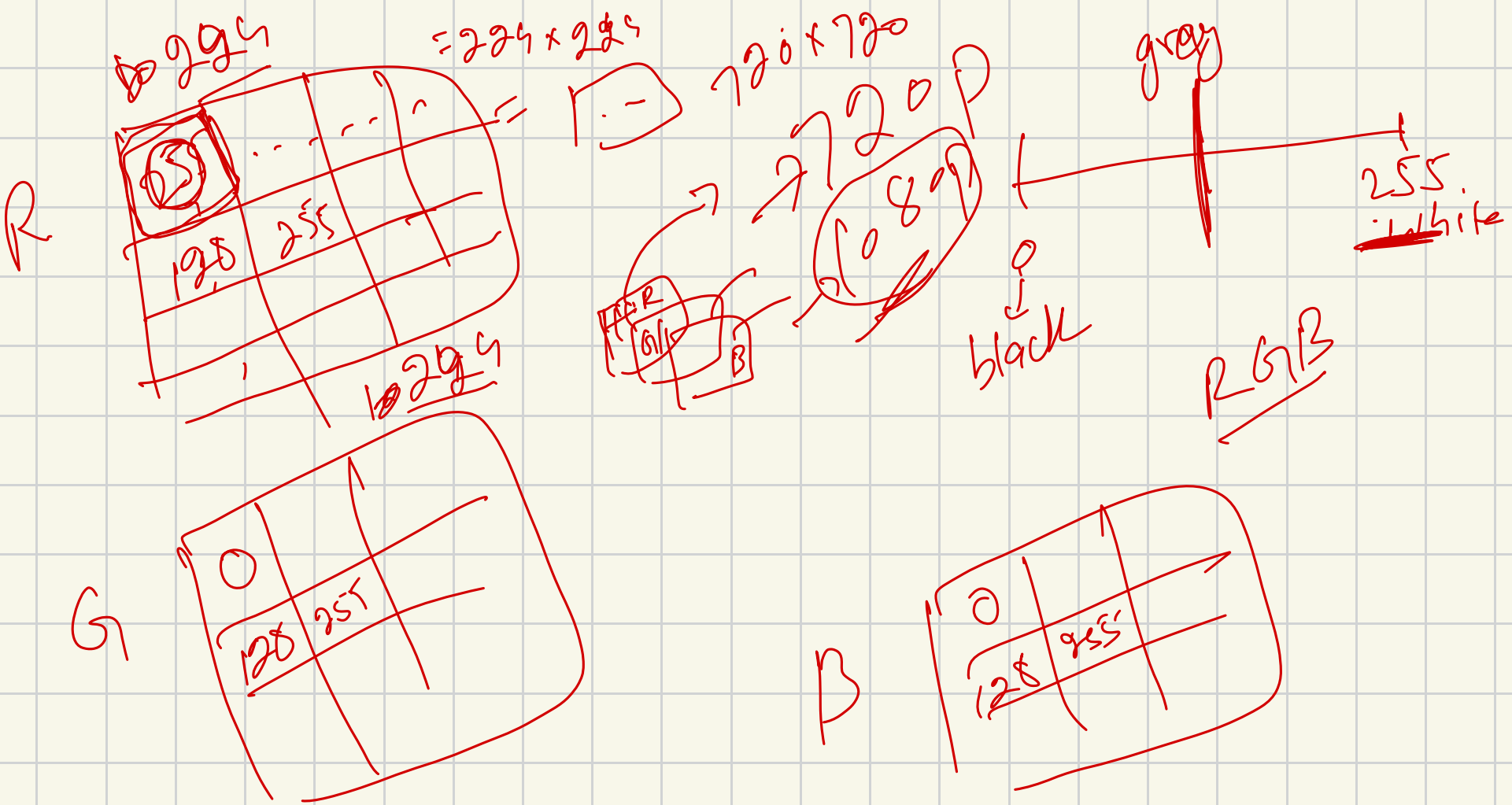
Sequence to Sequence with RNNs and Attention (cont.)

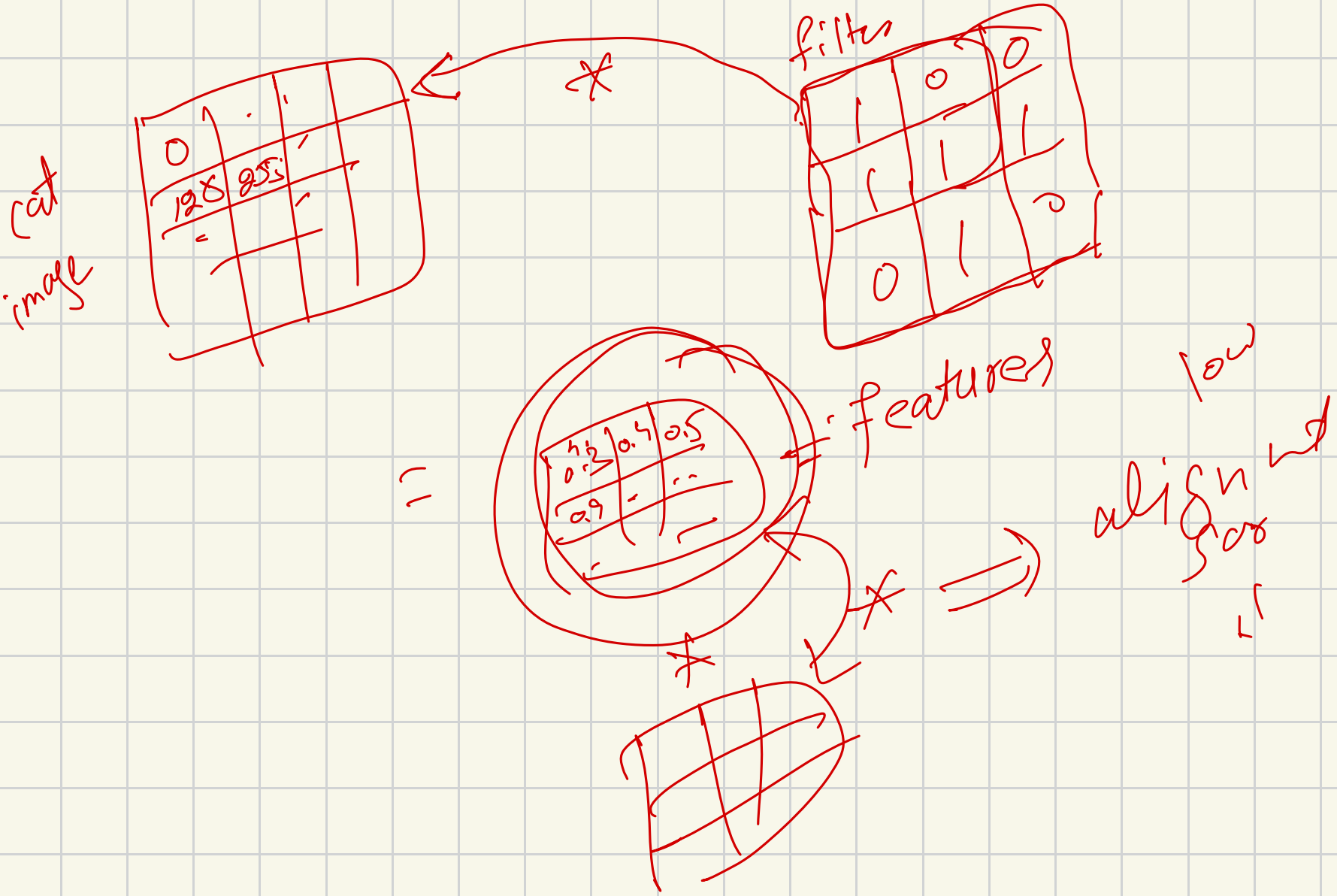


Use a CNN to compute a grid of features for an image

Sequence to Sequence with RNNs and Attention (cont.)




R
G
B
0
128
255
1024
720
1080
gray
black
RGB

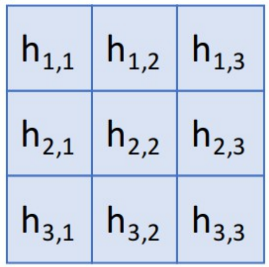

cat image
filter
features
low
aligned

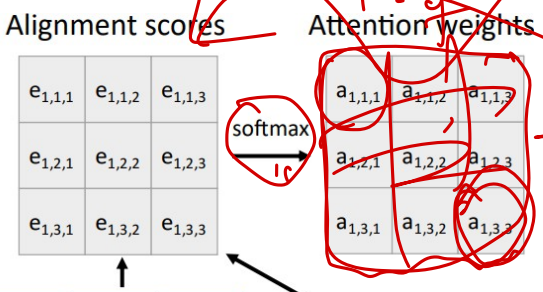
Sequence to Sequence with RNNs and Attention (cont.)

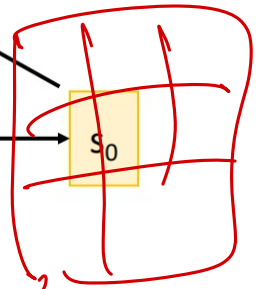
$$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$$
$$a_{t,:,:} = \text{softmax}(e_{t,:,:})$$

CNN

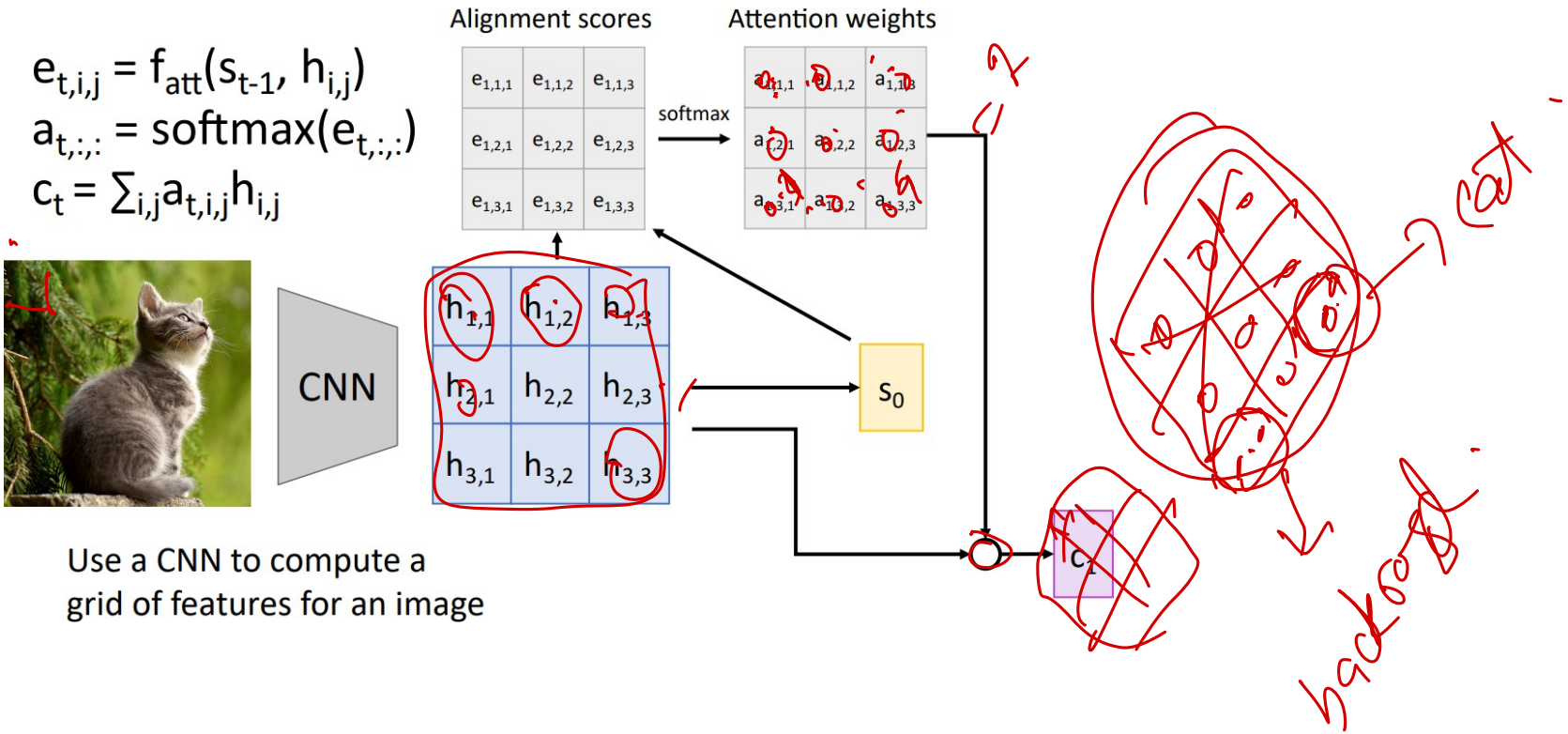
Use a CNN to compute a grid of features for an image



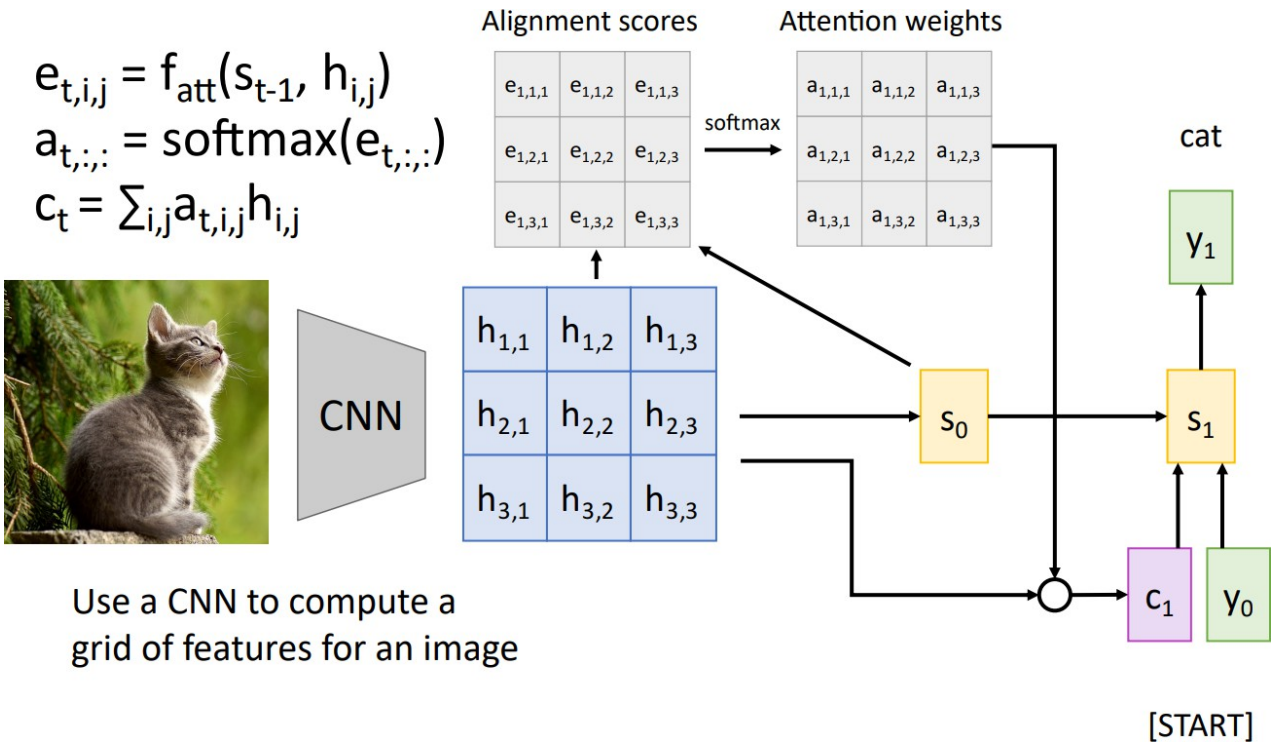




Sequence to Sequence with RNNs and Attention (cont.)

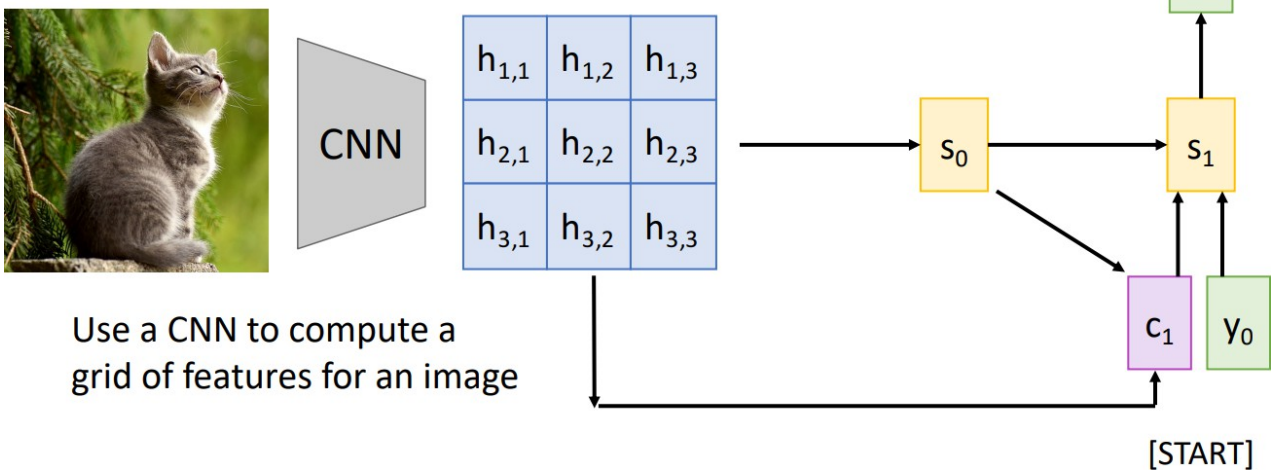


Sequence to Sequence with RNNs and Attention (cont.)

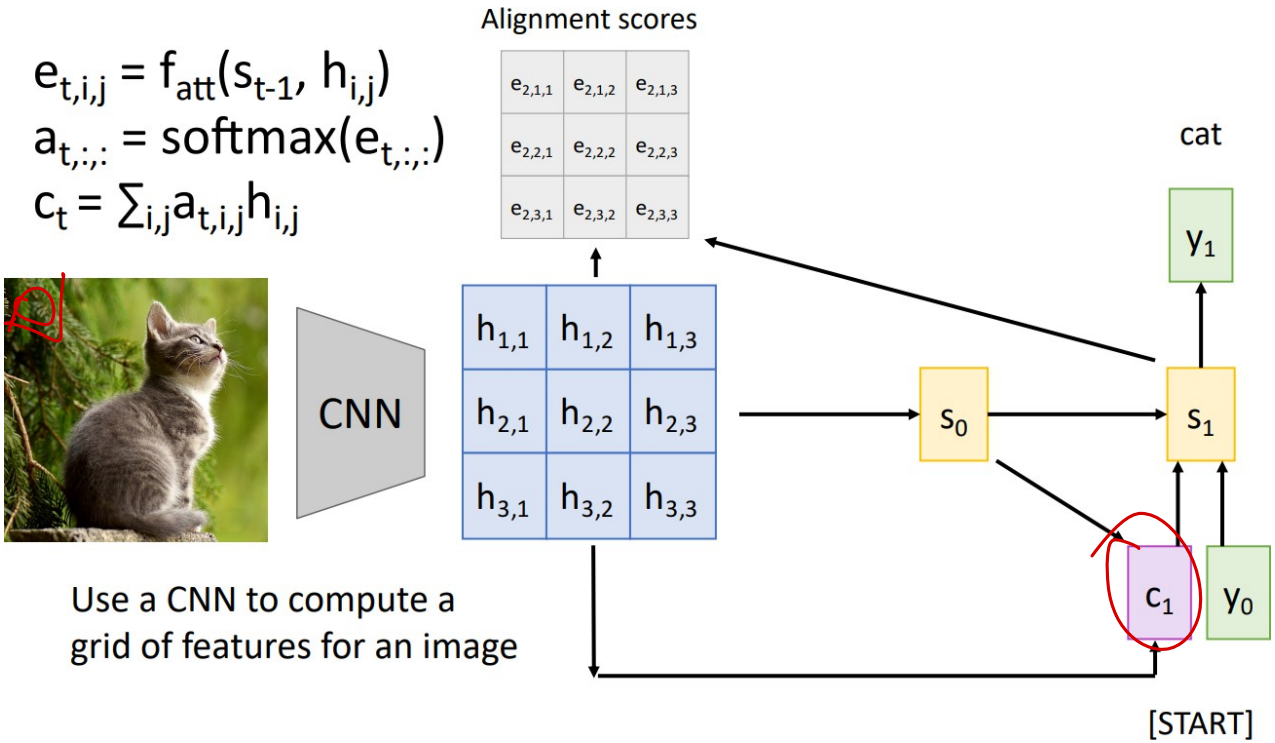


Sequence to Sequence with RNNs and Attention (cont.)

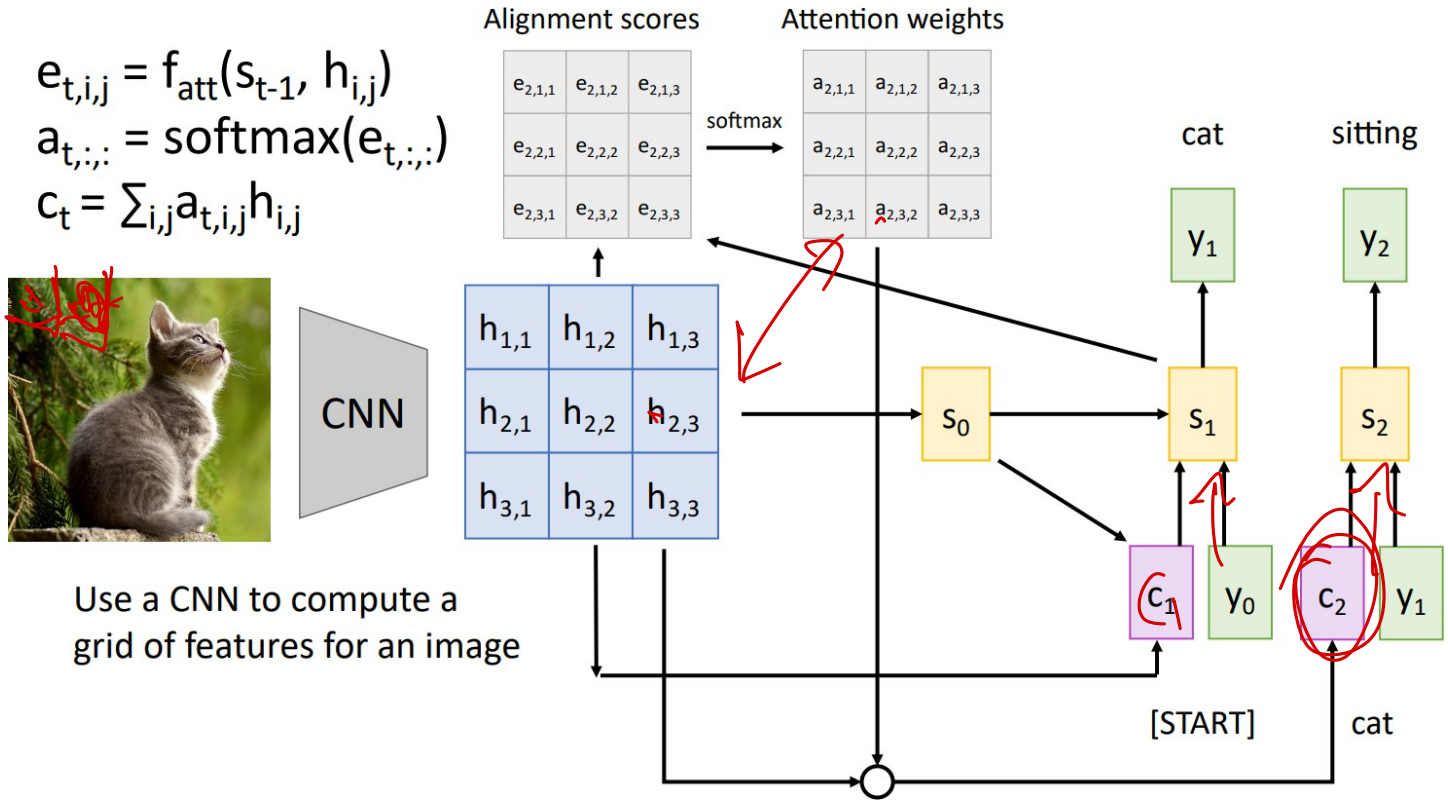
$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$



Sequence to Sequence with RNNs and Attention (cont.)



Sequence to Sequence with RNNs and Attention (cont.)



Sequence to Sequence with RNNs and Attention (cont.)

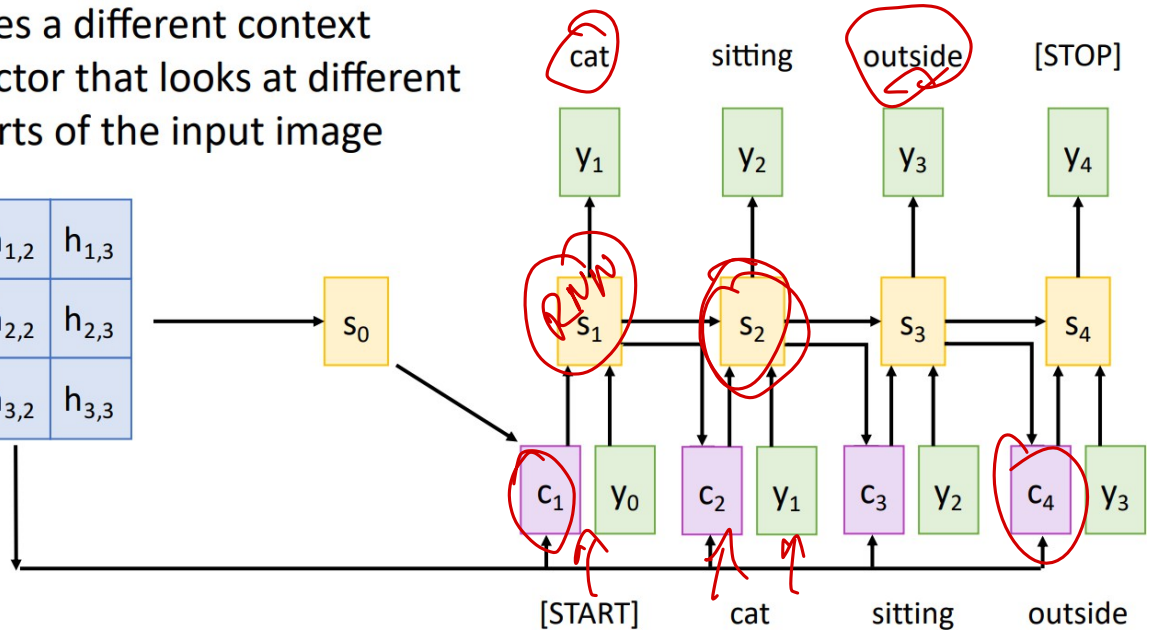
$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

Each timestep of decoder uses a different context vector that looks at different parts of the input image

CNN

$h_{1,1}$ $h_{1,2}$ $h_{1,3}$
$h_{2,1}$ $h_{2,2}$ $h_{2,3}$
$h_{3,1}$ $h_{3,2}$ $h_{3,3}$

Use a CNN to compute a grid of features for an image



Sequence to Sequence with RNNs and Attention (cont.)

A dog is standing on a hardwood floor.

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

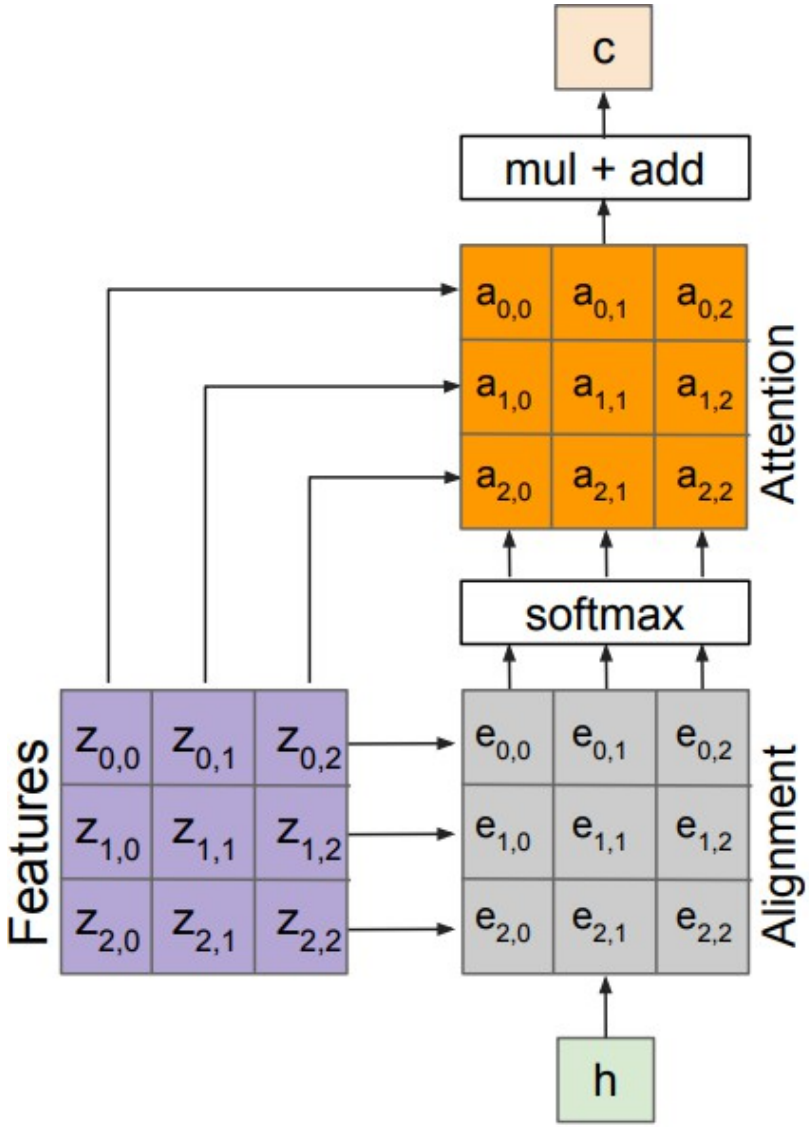
A group of people sitting on a boat in the water.

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

1

[1] Xu et al, "Show, AKend, and Tell: Neural Image CapAon GeneraAon with Visual AKenAon", ICML 2015

Attention we just saw in image captioning



Outputs:
context vector: **c** (shape: D)

Operations:
Alignment: $e_{i,j} = f_{att}(h, z_{i,j})$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $\mathbf{c} = \sum_{i,j} a_{i,j} z_{i,j}$

Inputs:
Features: **z** (shape: H x W x D)
Query: **h** (shape: D)

Limitations and Considerations of RNNs

- **Long-term Dependencies:**
  Basic Recurrent Neural Networks (RNNs) have difficulty capturing dependencies in sequences longer than approximately 10 steps. This limitation arises due to issues like vanishing and exploding gradients, which hinder the learning of long-range relationships in data.
- **Computational Cost:**
  Training RNNs is generally slower compared to more modern architectures such as transformers. This is because RNNs process sequences sequentially, making it challenging to leverage parallel computation effectively.
- **Alternatives:**
  Transformer models and Large Language Models (LLMs) have become popular alternatives. They offer superior parallel processing capabilities and are more effective at modeling long-range dependencies in sequential data.

References

[1] MIT 6.S191: Recurrent Neural Networks (2024).
https://introtodeeplearning.mit.edu/2024/recurrent/

[2] AWS: What is an RNN?
https://aws.amazon.com/what-is/rnn/

[3] V7 Labs: Complete Guide to RNNs.
https://www.v7labs.com/blog/recurrent-neural-network

[4] Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., & Yuille, A. (2014). Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN).
arXiv preprint arXiv:1412.6632.
https://arxiv.org/abs/1412.6632

[5] Elman, J. L. (1990). Finding structure in time.
Cognitive Science, 14(2), 179-211.

References (cont.)

[6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

[7] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*. https://arxiv.org/abs/1406.1078

[8] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*. https://arxiv.org/abs/1409.0473

References (cont.)

[9] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.
International Conference on Machine Learning (ICML).
https://arxiv.org/abs/1502.03044

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need.
Advances in Neural Information Processing Systems (NeurIPS).
https://arxiv.org/abs/1706.03762

[11] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult.
IEEE Transactions on Neural Networks, 5(2), 157-166.

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.