Deep Generative Models: Recurrent Neural Networks and Attention Mechanisms

Fall Semester 2025

René Vidal
Director of the Center for Innovation in Data Engineering and Science (IDEAS)
Rachleff University Professor, University of Pennsylvania
Amazon Scholar & Chief Scientist at NORCE

Penn
UNIVERSITY of PENNSYLVANIA

Taxonomy of Generative Models

What we've learned:
- Markov Models, HMMs, LDSs, RNNs

What we've learned:
- PPCA
- VAE

Deep Generative Models

- **Autoregressive models** (e.g., PixelCNN)
- **Flow-based models** (e.g., RealNVP)
- **Latent variable models**
  - **Implicit models** (e.g., GANs)
  - **Prescribed models** (e.g., VAEs)
- **Energy-based models**

What we study now:
- Variants of RNN architectures
- Applications

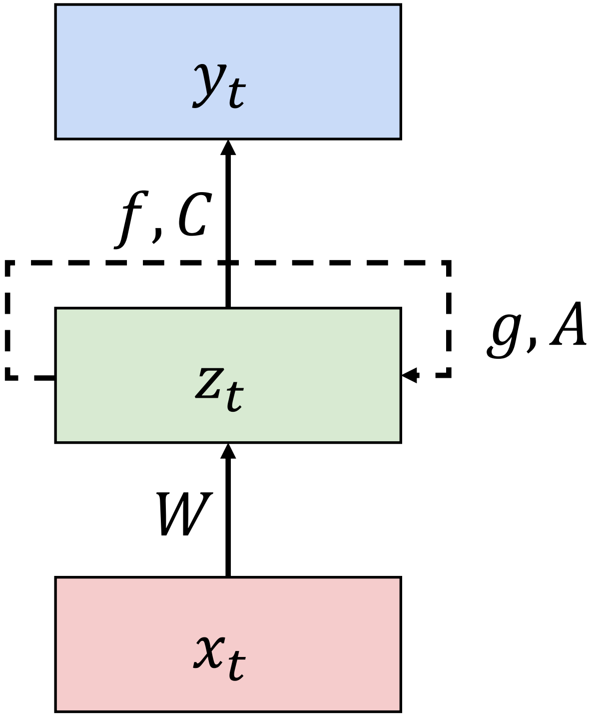
Autoregressive Models

- Many kinds of models
  - Markov Chains
  - Hidden Markov Models
  - Markov Random Fields
  - Linear Dynamical Systems
  - **Recurrent Neural Networks**
  - Transformers

- Last lecture
  - **Model**: Introduced the vanilla RNN architecture
  - **Inference**: Unfolding
  - **Training**: Backpropagation Through Time, Vanishing and Exploding Gradients
  - **Variants of RNNs**: LSTMs, GRUs

This Lecture

- We will continue with Recurrent Neural Networks
  - Sequence to Sequence Models
  - Align and Translate Model
  - Image Captioning



- Generalizations of Vanilla Neural Networks: RNNs can be very flexible, depending on the task!

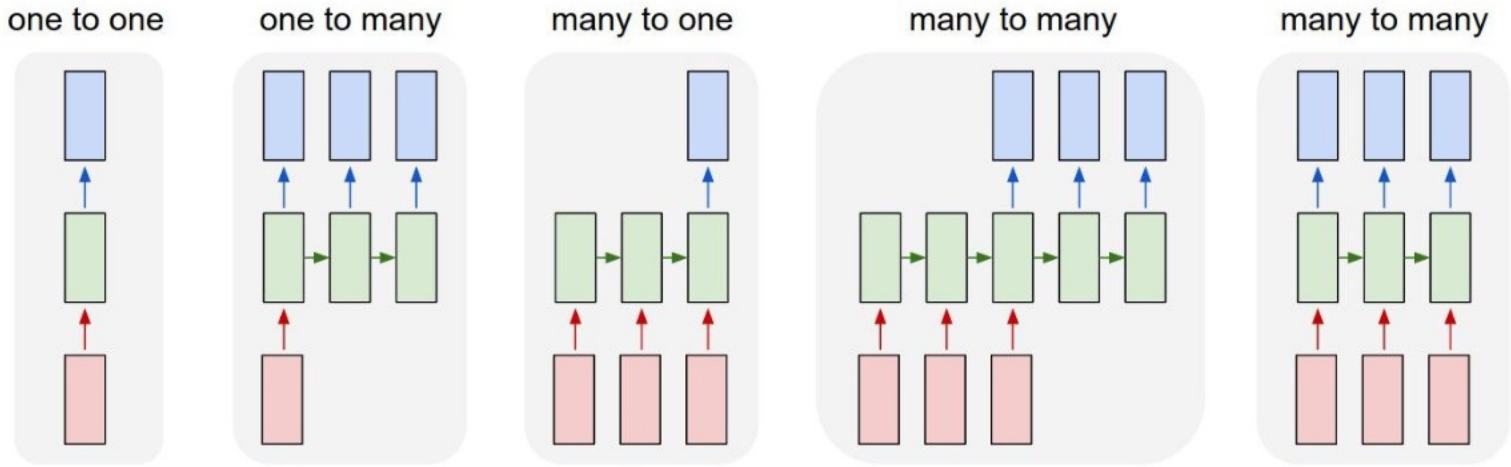


Image Classification | Image Captioning | Action Prediction | Text Translation Video Captioning | Video Segmentation

Timeline in

- In today’s and following lectures, we will see how the attention mechanism emerges into the well-know Transformer architecture today.

2014

Sequence to Sequence Learning with Neural Networks [Sutskever et al. (2014)]

- First application of RNNs to Machine Translation Tasks
- Introducing so-called “encoder-decoder” architectures

2015

Neural Machine Translation by Jointly Learning to Align and Translate [Bahdanau et al. (2015)]

- Using attention mechanism during the decoding process

2016

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention [Xu et al. (2016)]

- First application of attention for image captioning
- Our first multi-modal application in this class!

2017

Attention Is All You Need [Vaswani et al. (2017)]

- Introduced the popular architecture known as Transformer
- Stacking attention layers together

Consider the task of Machine Translation

- Say we are given pairs of sentences, one with English and the other with Spanish
  - Original sentence: "I have a big cat but a small house."
  - Translated sentence: "Tengo un gato grande pero una casa pequeña."

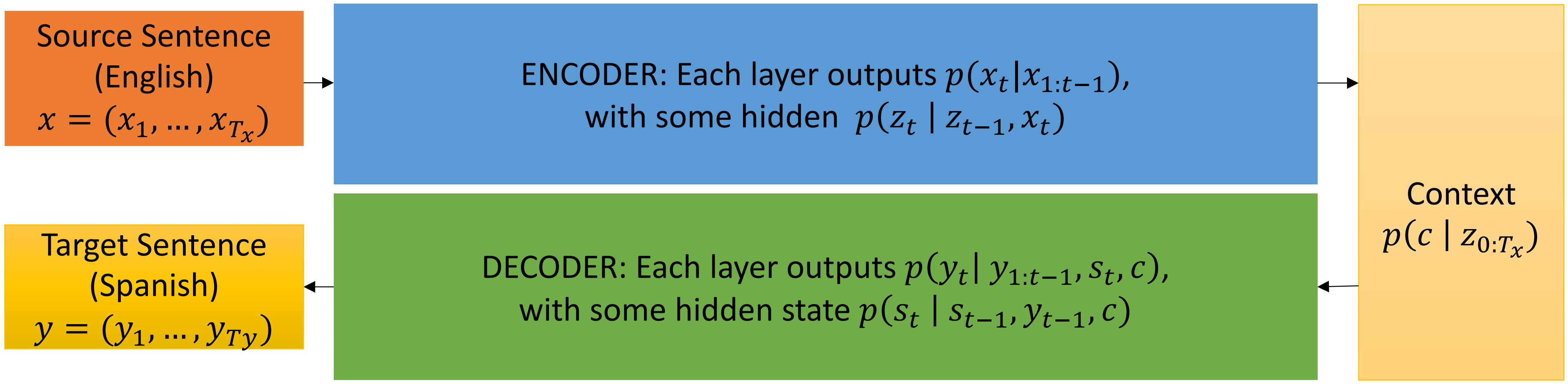
- In **Conditional Language Modeling (CLM)**, we want to compute

$$\hat{y}_{1:T_y} = \underset{y_{1:T_y}}{\operatorname{argmax}} P_\theta(y_{1:T_y} \mid x_{1:T_x})$$

- Here:
  - $\hat{y}_{1:T_y}$ is the target sentence
  - $x_{1:T_x}$ is our original sentence
  - θ is the parameters of our language model
- So, what is our model? And how do we learn θ?

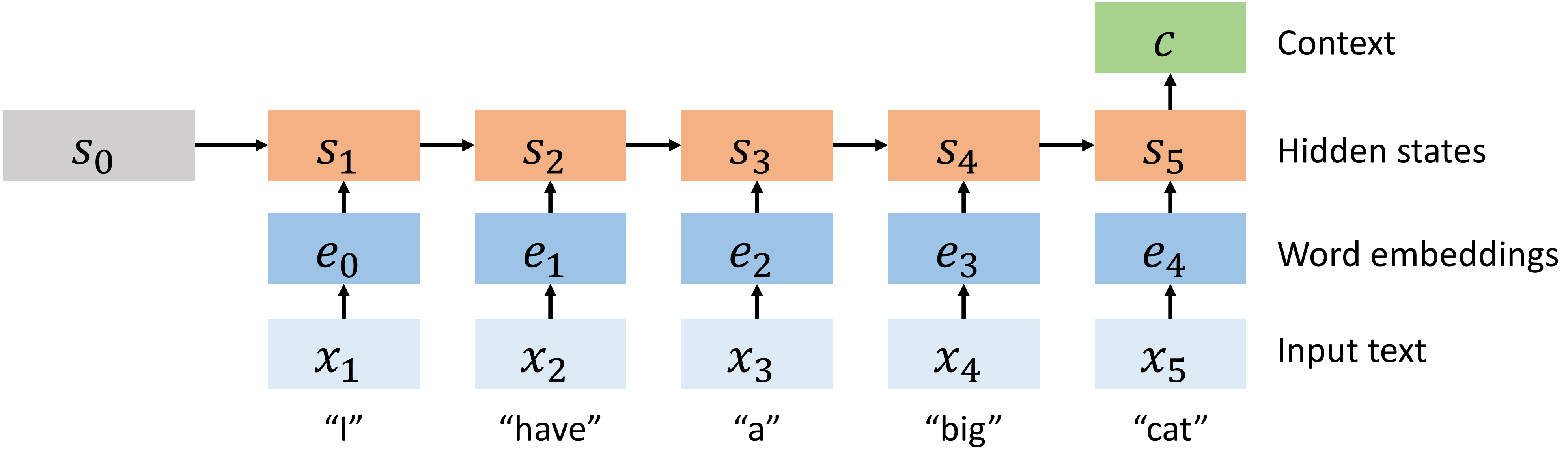
Overview

- The high-level idea is as follows:
  - A RNN allows us to encode our source sentence (English) $x_{1:T}$ to some latent (hidden) space $z_{1:T}$. This latent space encodes then semantics of the source sentence.
  - Once the semantics are captured, we want to decode it into the language we desire, i.e. target sentence (Spanish) $y_{1:T}$.
- A similar structure can be found in VAEs, where we also have an encoder-decoder structure

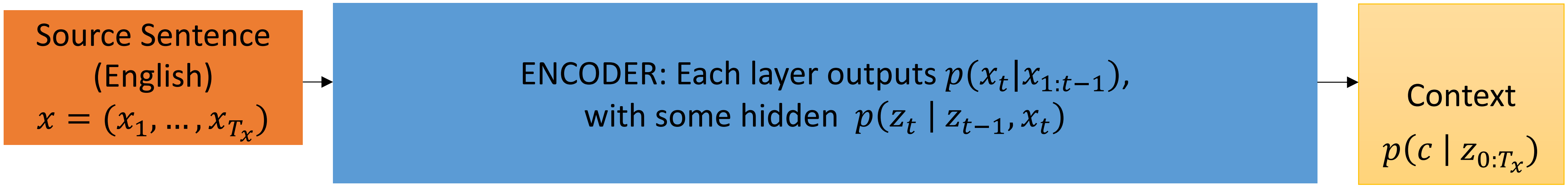


Structure of the Encoder

- Recall the RNN Encoder for Next word Prediction: modify it to produce a context

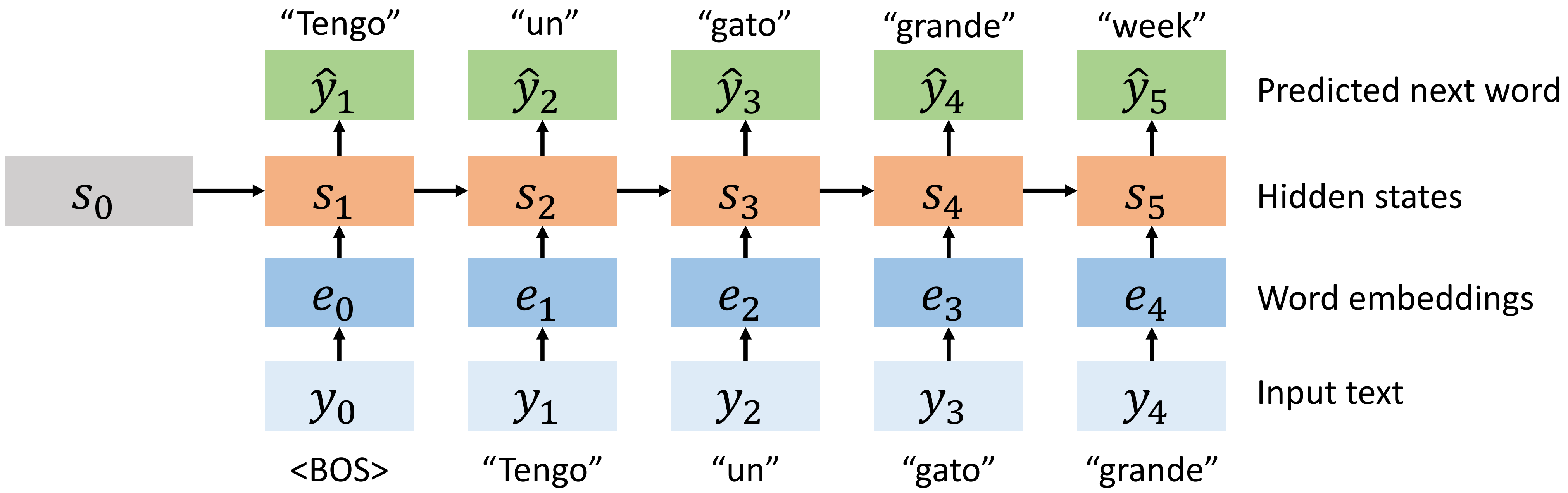


- We do not need a decoder: just summarize input sequence into a context vector

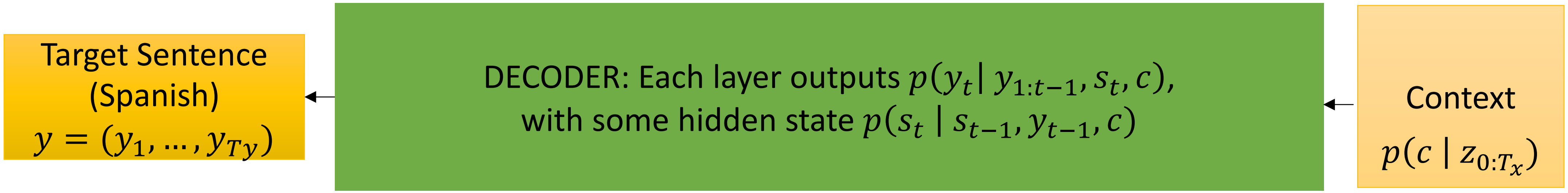


Structure of the Decoder

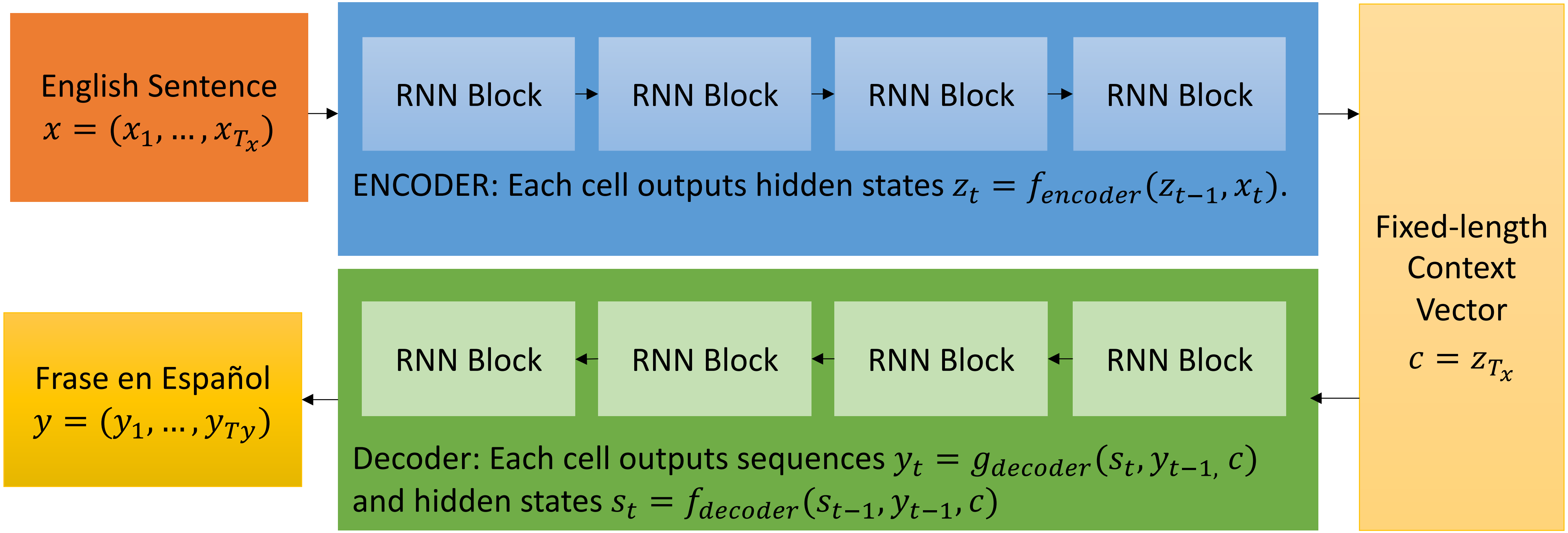
- Recall the RNN Encoder for Next word Prediction



- We now augment it with context



RNN Encoder-Decoder Architecture



- Remarks on Architecture from Sutskever et al. (2014):
  - $f_{encoder}, f_{decoder}, g_{decoder}$ are parameterized by LSTM layers.
  - In theory, the context vector can be the output of a more complex function h that takes in the entire sequence of hidden states, i.e., $c = h(z_{0:T})$. But they found virtually no difference in performance when compared to only using the very last state.
  - $g_{encoder}$ is not needed since we are not "decoding" from the ENCODER block.

Learning and Inference

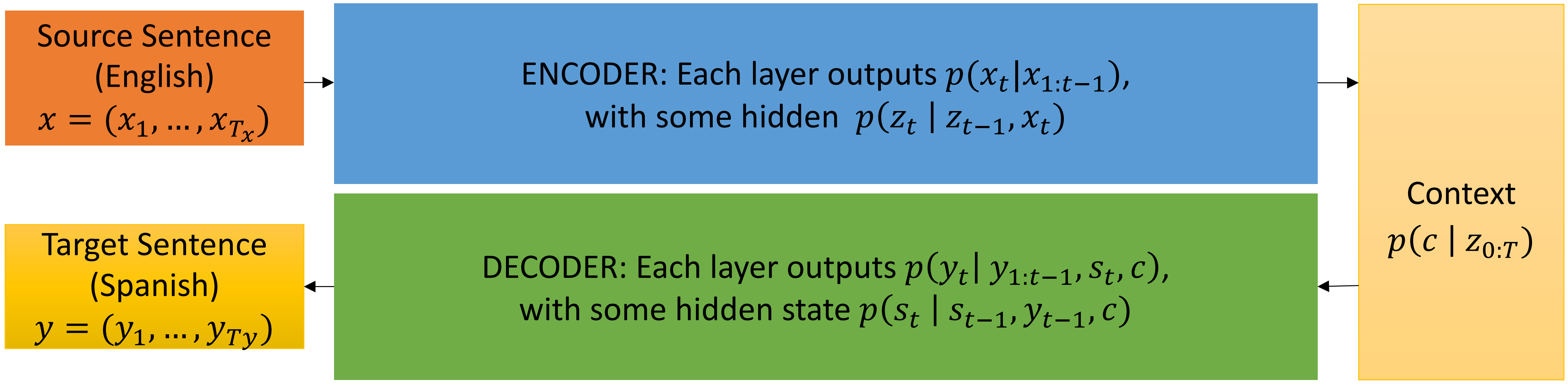
- **Learning:** Suppose we have the N samples $\left\{\left(x_{1:T_x}^{(n)}, y_{1:T_y}^{(n)}\right)\right\}_{n=1}^{N}$ of source-target sentence pairs. Similar to sentence classification, we can train the entire model end-to-end using cross entropy loss

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log P_{\theta}(y_{1:T_y}^{(n)} \mid x_{1:T_x}^{(n)})$$

- **Inference:** To decode, we simply select the target sentence with the highest probability. For a given $x_{1:T_x}$,

$$\begin{aligned} \hat{y}_{1:T_y} &= \mathrm{argmax}_{y_{1:T_y}} P_{\theta}\left(y_{1:T_y} \mid x_{1:T_x} \right) \\ &= \mathrm{argmax}_{y_{1:T_y}} \underbrace{P_{\theta}\left(y_{1:T_y} \mid c \right)}_{\text{Decoder} \to \text{Context}} \underbrace{P_{\theta}(c \mid x_{1:T_x})}_{\text{Context} \leftarrow \text{Encoder}} \end{aligned}$$

Major Flaw in Fixed-context seq2seq Models

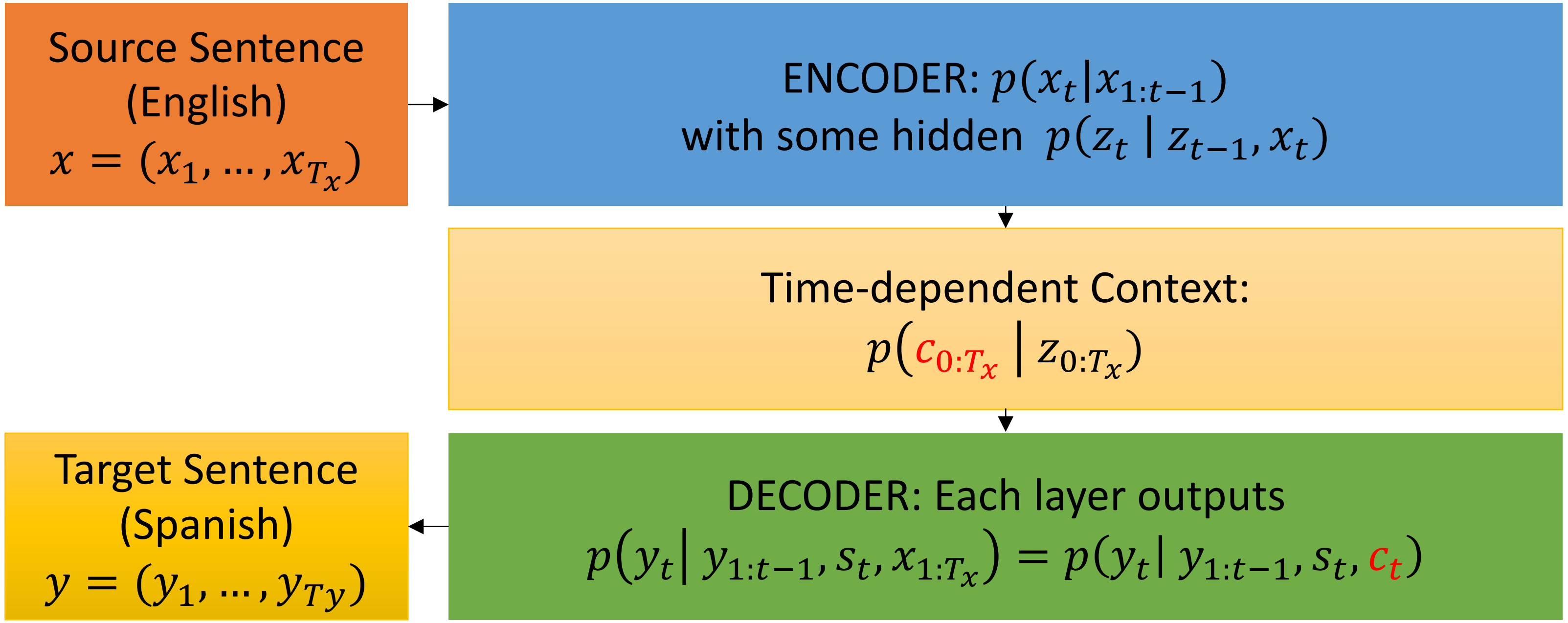


- However, there are obvious flaws to this design:
  - **Encoding**: the context c may not be able to capture earlier parts of the source sentence
  - **Fixed-length Context:** All the information from the source sentence is “jammed” into the single context vector c.

- As a result, this design often fails to capture long range dependences.

Improving seq2seq Models

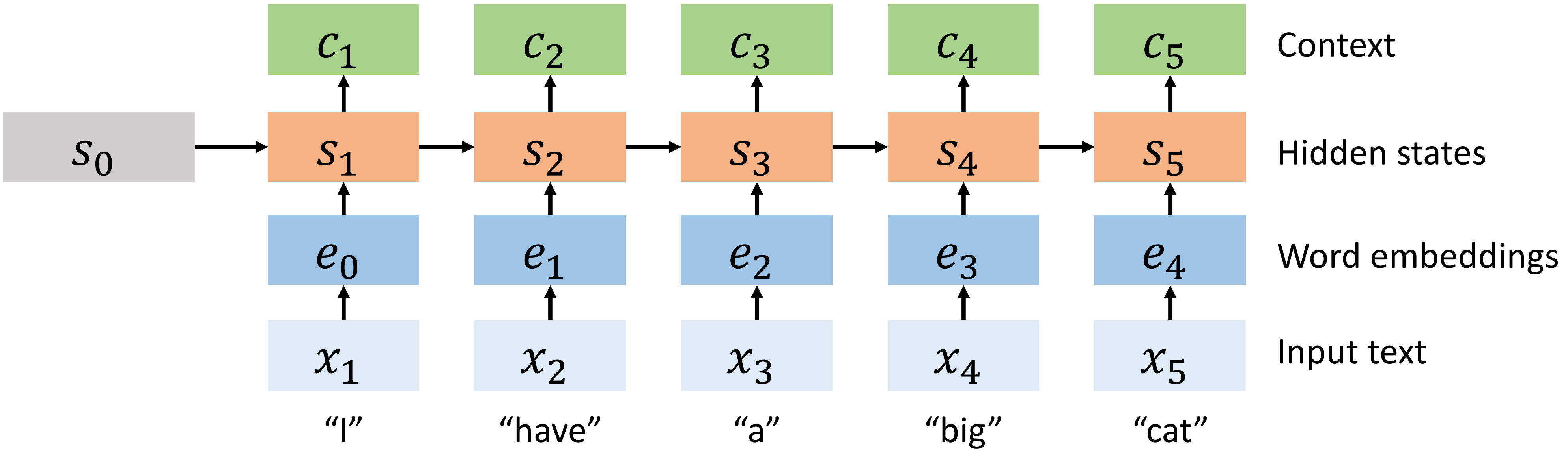
- Q: How can we improve *fixed-context* seq2seq models?
  - A: one possibility is to make the context time-dependent!
  - If our new context can better capture the information from each word, then it should prove long-range dependencies.



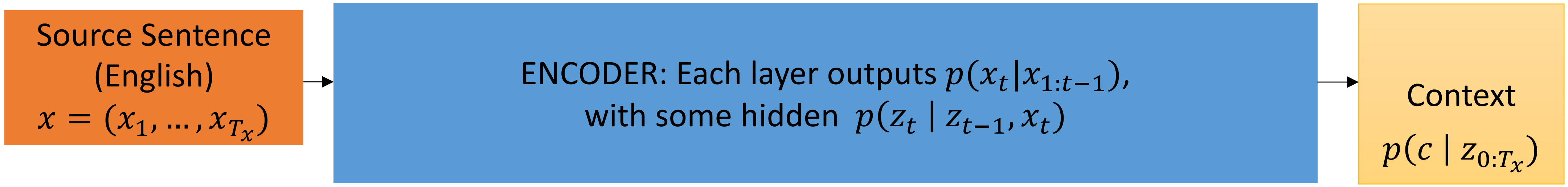
- How should we model the probabilities $p\left(c_{0:T_x} \mid z_{0:T_x}\right)$ and $p(y_t | y_{1:t-1}, s_t, c_t)$?

Structure of the Encoder

- Recall the RNN Encoder for Next word Prediction: modify it to produce a context



- We now augment it with context



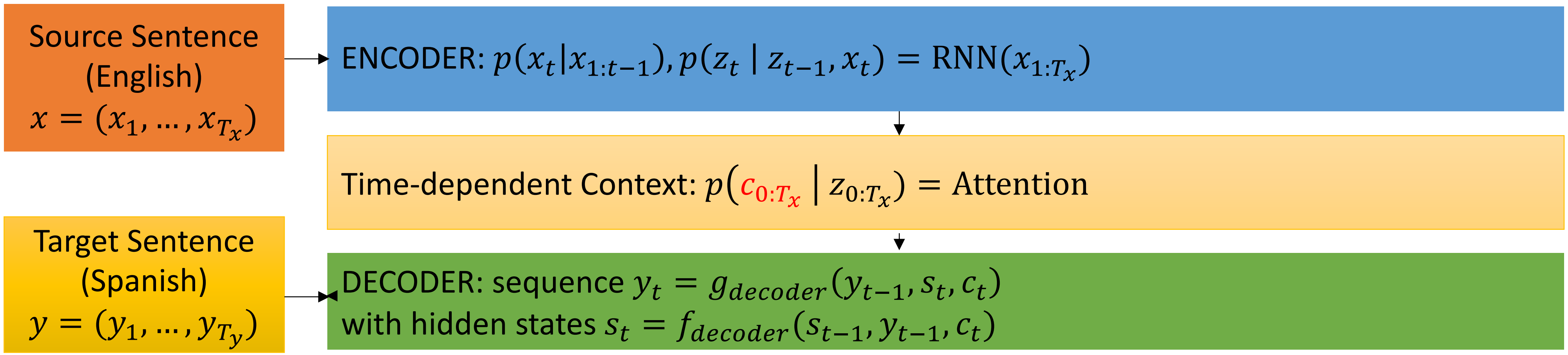
Align and Translate [Bahdanau et al. (2015)]

- **Intuition:** Translation of the word x_t to y_t depends on the contexts of both the source sentence $x_{1:T}$ and target sentence $y_{1:T}$.
  - The latent space should be able to capture what is important

- Take our Spanish example:
  - Original sentence: "I have a big cat but a small house."
  - Translated sentence: "Tengo un gato grande pero una casa pequeña."
  - Notice that the translation doesn't exactly align
  - Hence we need a way to tell the model what part of the sentence to focus on

- **High-Level Idea**: During decoding, each context c_t to be a summary of the sources' hidden states $z_{0:T_x}$ and the target's current hidden states s_t

Align and Translate [Bahdanau et al. (2015)]



- Define the probability of the target word y_t at time t as

$$p\left(y_t \middle| y_{1:t-1}, s_t, x_{1:T_x}\right) = g_{decoder}(y_{t-1}, s_t, c_t)$$

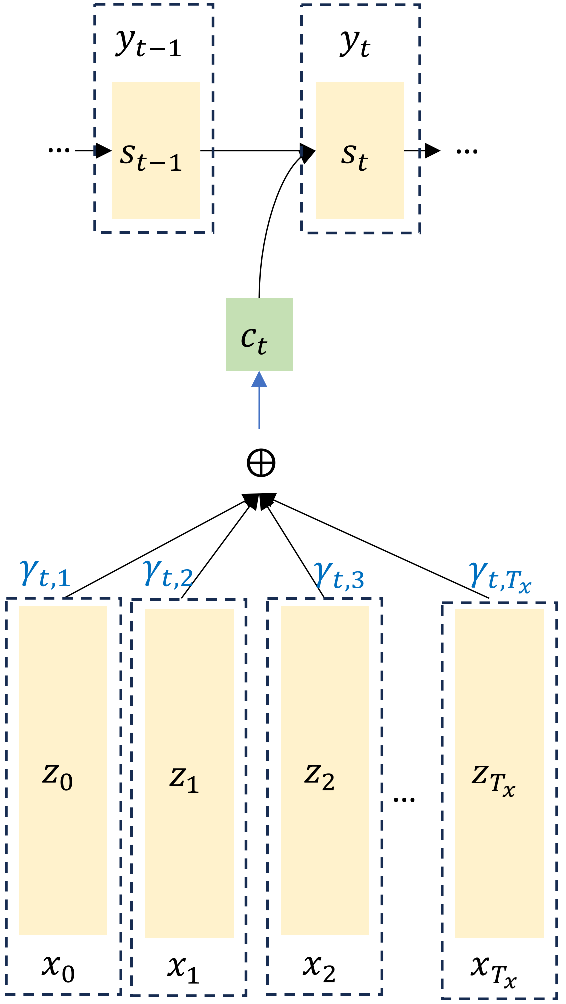
- Here $s_t = f_{decoder}(s_{t-1}, y_{t-1}, c_t)$ is hidden state of the RNN decoder that takes in the previous word y_t, the previous hidden state s_t, and a context vector c_t as input.
  - Similar to before, $f_{decoder}$ and $g_{decoder}$ are functions parameterized by neural networks.

Align and Translate

- Decoder: **context vector** c_t is computed as a weighted sum of the hidden states z_j:

$$c_t = \sum_{j=1}^{T_x} \gamma_{tj} z_j \qquad \gamma_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \qquad e_{tj} = a(s_{t-1}, z_j)$$

Context vector — Weights of hidden states — Alignment model



- Here:
  - c_t is the expected hidden state over all the hidden states with probability γ_{tj}.
  - γ_{tj} is the probability that the target word y_t is aligned to, or translated from, a source word x_j.
  - a is called the **Alignment model**
    - Computes how well the inputs around position j and the output at position t match
    - Typically chosen to be a feedforward neural network

Align and Translate

- In Bahdanau et al. (2015), they made the following design choices:
  - **Encoder**: Using a Bi-directional RNN, compute the *forward and backward* hidden states $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ using input $x = (x_0, \dots, x_T)$. Concatenate them as one encoder hidden state $z_t = \left[\overrightarrow{h_t} \middle| \overleftarrow{h_t}\right]$ (assume they are row vectors). Hidden states are also called *annotations.*
  - **Decoder:** Using a single direction RNN with Attention mechanism and alignment model

$$a(s_{i-1}, z_j) = v_a^{\top} \tanh(W_a s_{i-1} + U_a z_j)$$

- Ultimately, these design choices are flexible and application-dependent.

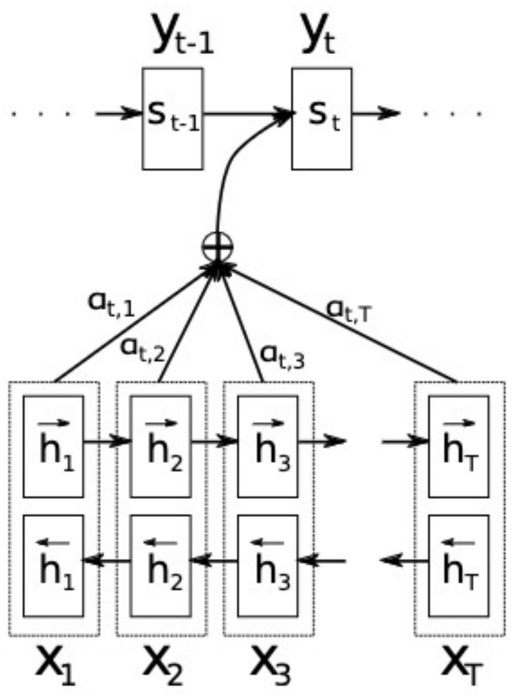


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \dots, x_T)$.

Visualization of Annotations and Alignments

- Correlation between the source sentence (English) and target sentence (French)

- Able to show that some target words “attend” to multiple target words

- Diagonal: x_t matches with y_t
- Cross-Diagonal: context dependent

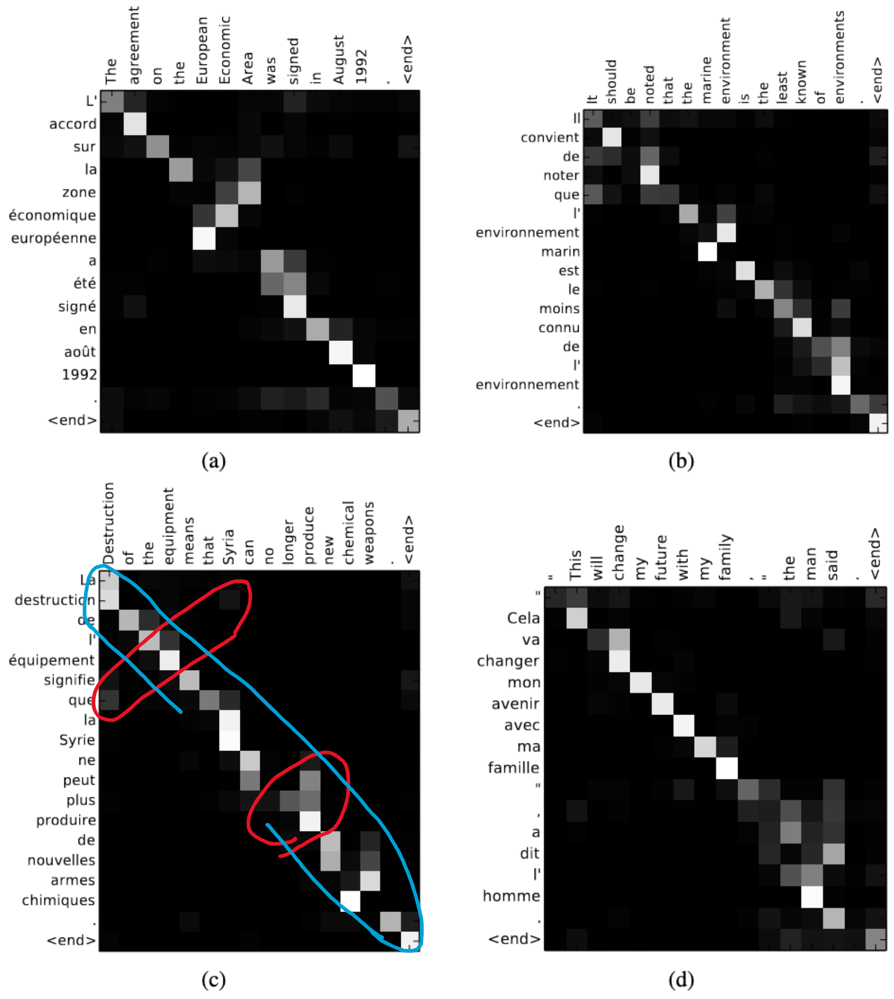


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

Recap

- Today we covered two seq2seq models:
  - Encoder-Decoder with fixed context [Sutskever et al. (2014)]
  - Time-dependent context with Attention Mechanism [Bahdanau et al. (2015)]

- Comparing seq2seq models
  - Bi-directional RNNs instead of LSTMs
  - **Alignment model** instead of single fixed-vector hidden states
  - Have context vector c_t that depends on the timestep

- Next lecture:
  - Using attention mechanism for image captioning
  - Is attention all your need?

Deep Generative Models: VAE+RNN for Image Captioning

Fall Semester 2025

René Vidal

Director of the Center for Innovation in Data Engineering and Science (IDEAS)

Rachleff University Professor, University of Pennsylvania

Amazon Scholar & Chief Scientist at NORCE

Encoder-Decoder Architectures

- Encoder-Decoder Architectures allow us to
  - Learn a meaningful hidden representation for our input
  - Via a Decoder, make use of our hidden representation for downstream tasks

- So far, our main motivation has been driven by Language
  - Machine Translation, Text Summarization, etc

- What about Cross Modalities? Language-to-Vision?

Up Next

- Today we will talk about Image Caption Generation using a combination of Variational Auto Encoders (VAEs) and Recurrent Neural Networks (RNNs)

- Introduced in Xu et al (2016) "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention"

- Task: Given an image, generate a sentence that describes the image
  - Can be seen as a combination of Object Detection and Machine Translation

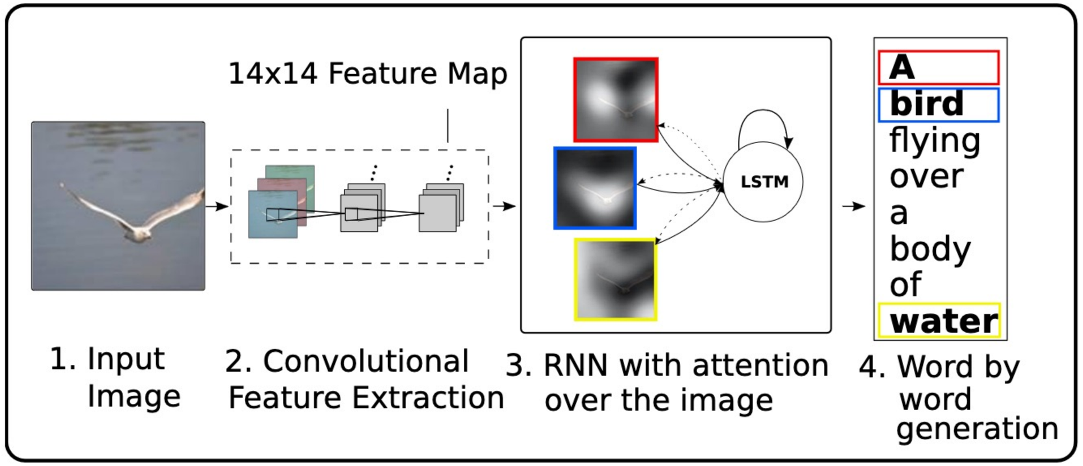
A bird flying over a body of water.

A woman throwing a frisbee in a park.

Task

- Today we will talk about Image Caption Generation using a combination of Variational Auto Encoders (VAEs) and Recurrent Neural Networks (RNNs)

- Our overall pipeline:

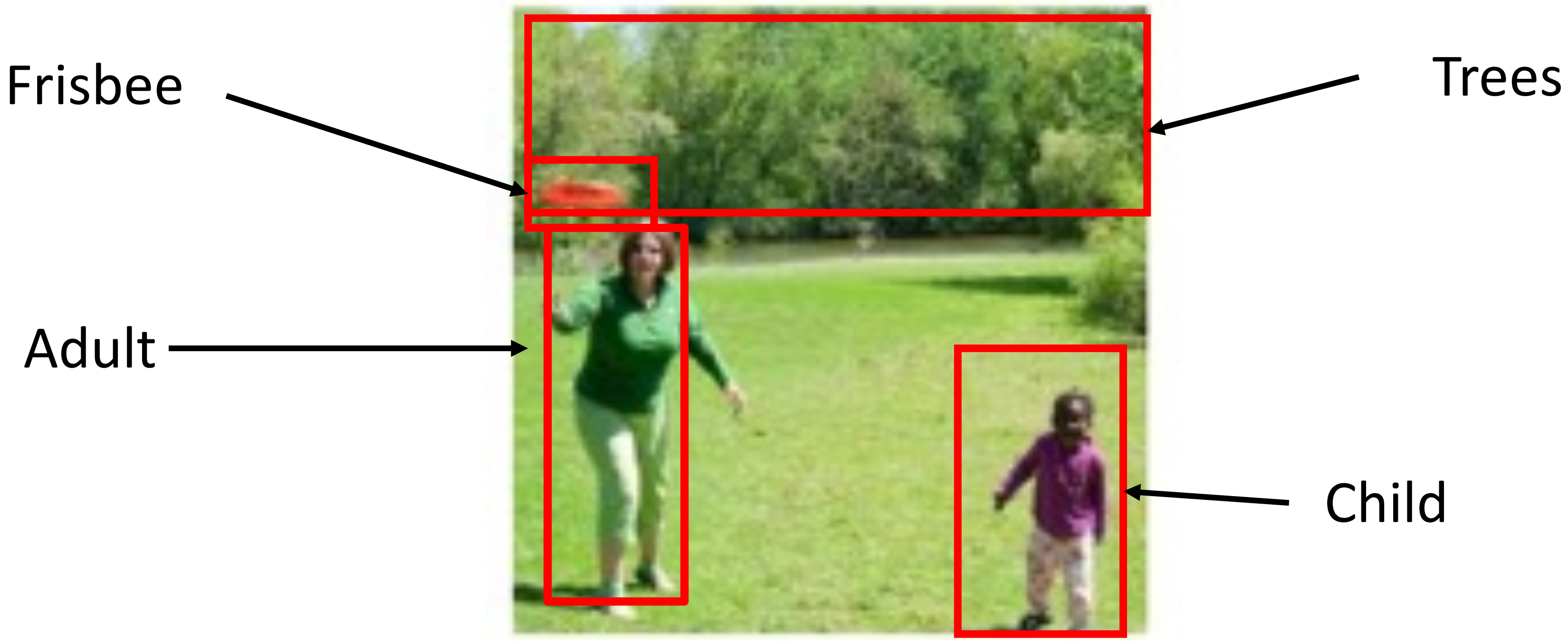


- Similar to any language task, suppose we are given a vocabulary of size K, a sentence of length T can be presented by each word being a one-hot embedding

$$y = \{y_0, \dots, y_T\}, y_t \in \mathbb{R}^K$$

Image Encoder

- An image can have many sources of information



- Ideally our hidden representation should be meaningful, in the sense that it should capture all the semantic parts of the image

Image Encoder: Convolutional Neural Networks

- To capture these meaningful features, we will feed the image through a (pre-trained) Convolutional Neural Network

- Then use the feature vectors x_i of earlier convolutional layers to represent low-level features

- Denote each part by

$$x = [x_1 | \dots | x_L] \in \mathbb{R}^{T_x \times D}$$

where T_x is the number of low-level features of dimension D

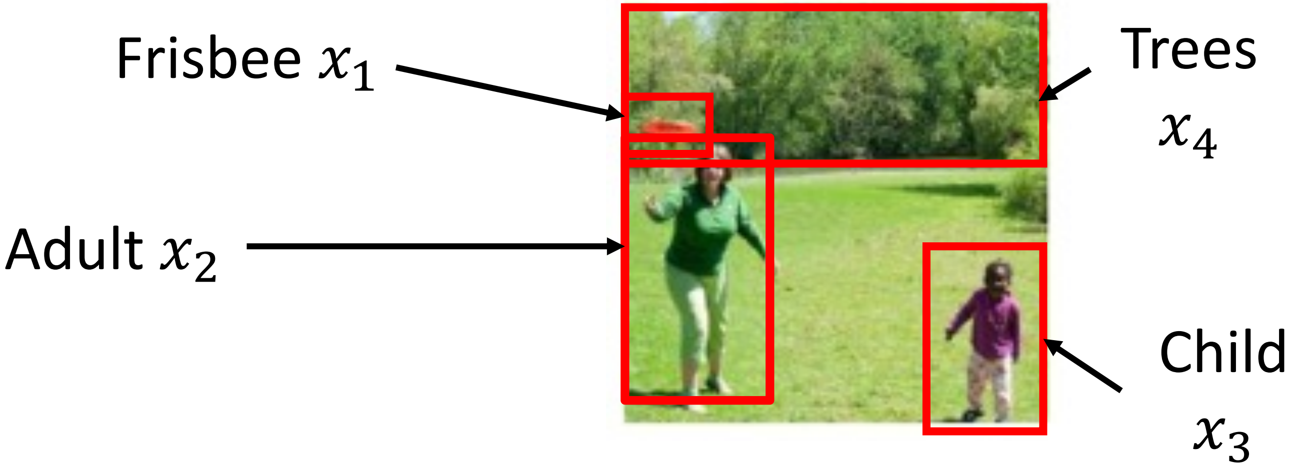
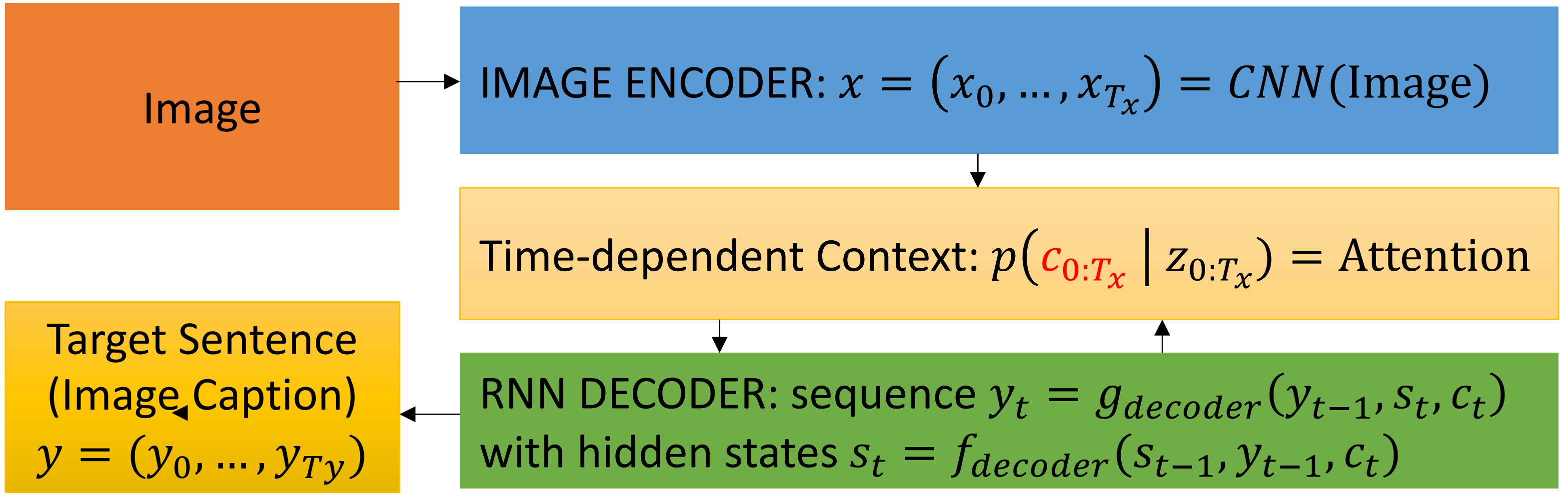


Figure above: In an ideal situation, each semantic part is presented by a low-level feature vector x_i.

Decoder: LSTM with Context



- Similar to Align and Translate, now we have to design the context vectors
  - For image captioning, we will use attention mechanisms to attend to different locations of the image

- So, how is the context vector $\hat{c}_t$ computed using our image features $x_1 \dots x_{T_x}$?

Decoder: Context Vector and Attention

- c_t is a context vector that presents the relevant part of the image input at time t
- There are two ways to compute c_t :
  - Option 1: ϕ = **Hard Attention**: only one of the T_x image locations is chosen
  - Option 2: ϕ = **Soft Attention:** all of them is weighted in some way
- Similar to Align and Translate model, we can define:

A person is standing on a beach with a surfboard.

$$c_t = \phi(x_1, \dots, x_L, \gamma_{t,1}, \dots, \gamma_{t,L})$$

Some function ϕ of using the attention weights and features to combine a context vector.

$$\gamma_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$$

Weights, for which of the L positions to attend to

$$e_{ti} = a(x_i, s_{t-1})$$

“Attention Model”
a multi-layer perceptron

Image Features $x_1, \dots, x_{T_x}$
Decoder’s Hidden Features $s_1, \dots, s_T$

First option for ϕ: Stochastic Hard Attention

- Stochastic Hard Attention implies we use a “on-off” way to choose which location of the image to focus
  - Meaning we can only choose one location each time
- Let $\hat{\gamma}_t \in \{0,1\}^L$ be a *one-hot* location variable that represents where the model decides to focus attention when generating the t^{th} word.
- We can treat the attention locations as intermediate latent random variables

$$p(\hat{\gamma}_{t,i} = 1 \,|\hat{\gamma}_{1:t-1}, x_1, \ldots, x_L) = \gamma_{t,i} \qquad\qquad \hat{c}_t = \sum_{k=1}^{T_x} \hat{\gamma}_{t,k} x_k$$

- This means we can treat γ_t as a categorical distribution:

$$\hat{\gamma}_t \sim \mathrm{Categorical}(\gamma_{t,1}, \ldots, \gamma_{t,T_x})$$

- And we can just sample this distribution during inference to obtain samples for the context $\hat{c}_t$.

Stochastic Hard Attention (Learning)

- While it is intuitive to parameterize $\hat{\gamma}_t \sim \text{Categorical}(\gamma_{t,1}, \dots, \gamma_{t,T_x})$, it raises the question of how to train the entire model end-to-end?
    - This is the same issue we face in VAEs!
    - Hence we can use the **Variational Lower Bound** approach
- To backpropagate through the entire model, we need to define **a variational lower bound** on the marginal log-likelihood $\log p(y_{0:T} \mid x_{1:T_x})$ of observing the sequence of words $y_{0:T}$ given image features x

- Quick Recall: Let X and Z be a random variable, jointly distributed with distribution p_θ. If $p_\theta(X)$ is the marginal distribution of X and $p_\theta(Z|X)$ is the conditional distribution of Z given X. Then for any sample $x \sim p_\theta$ and any distribution q_ψ, we have

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\psi}[\log \frac{p_\theta(x, z)}{q_\psi(z)}]$$

Stochastic Hard Attention (Learning)

- Just like our VAE model, we may now consider our context $p(c)$ as our latent variable. Then we can derive the ELBO.

- Define
  - ψ as the parameters of the encoder $q(c \mid x)$, the distribution of context vectors from CNNs.
  - θ as the parameters of the decoder $p(y \mid c, x)$, the image captioner.
- The Evidence Lower Bound L_s:

$$
\begin{aligned}
L_{\theta,\psi}(c, x, y) &= \sum_c q_\psi(c \mid x) \log p_\theta(y \mid c, x) \\
&\leq \log \sum_c q_\psi(c \mid x) p_\theta(y \mid c, x) && \text{(Jensen's Inequality)} \\
&= \log p_\theta(y \mid x) && \text{(Marginal Log-Likelihood)}
\end{aligned}
$$

Stochastic Hard Attention (Learning)

- Our Lower Bound: $L_{\theta,\psi}(c,x,y) = \sum_c q_\psi(c \mid x) \log p_\theta(y \mid c,x)$
- To learn we will need the gradient. For **both parameter** $W = \{\theta,\psi\}$ in our RNN, we can estimate the gradient using Monte Carlo sampling approximation.

- The exact derivative for the ELBO objective (derivation next slide):

$$\frac{\partial L}{\partial W} = \sum_c q_\psi(c \mid x)\left[\frac{\partial \log p_\theta(y \mid c,x)}{\partial W} + \log p_\theta(y \mid c,x)\frac{\partial \log q_\psi(c \mid x)}{\partial W}\right]$$

- The estimated derivative using Monte Carlo sampling approximation, with $\hat{\gamma}_t \sim \text{Categorical}\left(\gamma_{t,1},\ldots,\gamma_{t,L}\right)$ and $\hat{c}_t = \sum_{k=1}^{T_x} \hat{\gamma}_{t,k} x_k$:

$$\frac{\partial L}{\partial W} = \frac{1}{M}\sum_{m=1}^{M}\left[\frac{\partial \log p_\theta(y \mid \hat{c}^{(m)},x)}{\partial W} + \log p_\theta(y \mid \hat{c}^{(m)},x)\frac{\partial \log q_\psi\left(\hat{c}^{(m)} \mid x\right)}{\partial W}\right]$$

Derivation of the Gradient for Exact ELBO

- $L_{\theta,\psi}(c,x,y) = \sum_c q_\psi(c \mid x) \log p_\theta(y \mid c, x)$

$$\frac{\partial L_{\theta,\psi}(c,x,y)}{\partial W}$$

$$= \sum_c q_\psi(\, c \mid x\,) \frac{\partial \log p_\theta(y|c,x)}{\partial W} + \frac{\partial q_\psi(c \mid x)}{\partial W} \log p_\theta(y \mid c, x) \qquad \text{(chain rule)}$$

$$= \sum_c q_\psi(\, c \mid x\,) \frac{\partial \log p_\theta(y|c,x)}{\partial W} + q_\psi(c \mid x) \frac{\partial \log q_\psi(c \mid x)}{\partial W} \log p_\theta(y \mid c, x)$$

$$= \sum_c q_\psi(\, c \mid x\,) [\frac{\partial \log p_\theta(\, y|c, x\,)}{\partial W} + \frac{\partial \log q_\psi\,(c \mid x)}{\partial W} \log p_\theta(y \mid c, x)]$$

- The third line uses the identity $\frac{\partial q_\psi(c \mid x)}{\partial W} = q_\psi(c \mid x) \frac{\partial \log q_\psi(c \mid x)}{\partial W}$

Second option for ϕ: Deterministic "Soft" Attention

- Recall our three equations:

$$c_t = \phi(x_1, \dots, x_L, \gamma_{t,1}, \dots, \gamma_{t,L}) \qquad \gamma_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \qquad e_{ti} = a(x_i, s_{t-1})$$

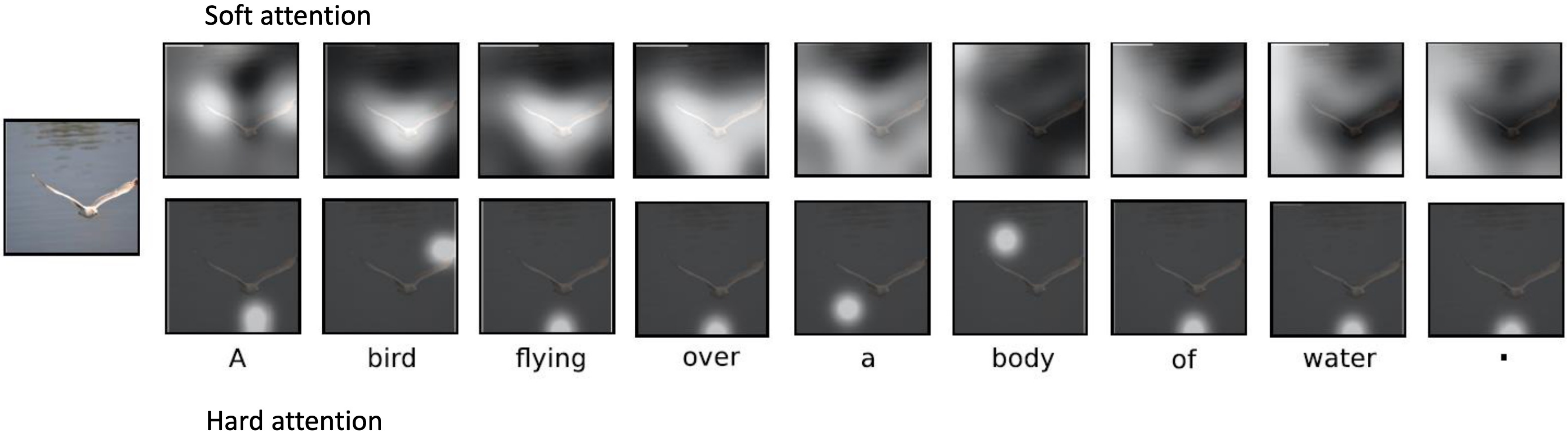
- Hard Attention method requires us to ample the attention location c_t each time
- Instead, we can take the expectation of the context vector c_t directly

$$c_t = \phi(x_1, \dots, x_L, \gamma_{t,1}, \dots, \gamma_{t,L}) = \sum_{i=1}^{T_x} \gamma_{t,i} x_i$$

- Then this would no longer be a "on-off" mechanism, but a weighted sum of low-level features instead.

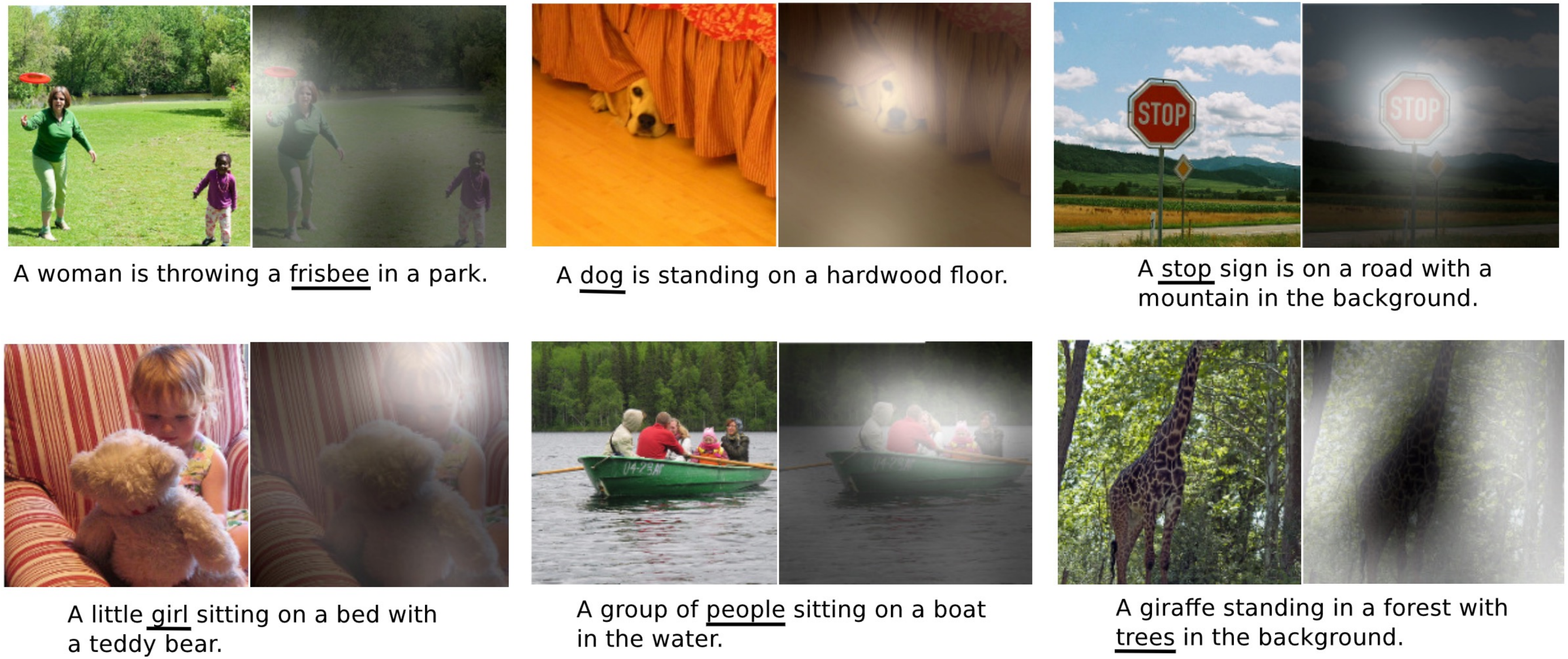
- Lucky for us, this is differentiable end-to-end using cross entropy

Soft Attention vs Hard Attention



Examples of Image Caption Generation

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



Examples of Image Caption Generation

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Wrap-up

- We introduced a Multi-modal Encoder-Decoder architecture method to do image caption
  - Generative: parameterize location variable with categorial variable (Hard Attention), use MCMC to sample and learn the RNN decoder.
  - Discriminative: use weighted sum (Soft Attention) and train everything end-to-end.

- We have shown the brief history of Attention mechanism
  - Sequence to Sequence with Neural Networks for Machine Translation
    - The use of fixed-length single context vector to decode c
  - Align and Translate for Machine Translation
    - The use of multiple time-dependent context vectors c_t
  - Image Captioning
    - Soft and Hard Attention

Why do RNNs fall short? And what can we do?

- Hard to capture long-term dependencies
  - Require modification to architectures
- Training Issues: Vanishing/Exploding Gradients
- Hard to handle varying length sequences
- Sequential nature make them hard to process in parallel

- **Solution to all of this:**
- Let’s not depend on recurrence anymore
- Let’s just rely “Attention” completely to capture global dependencies