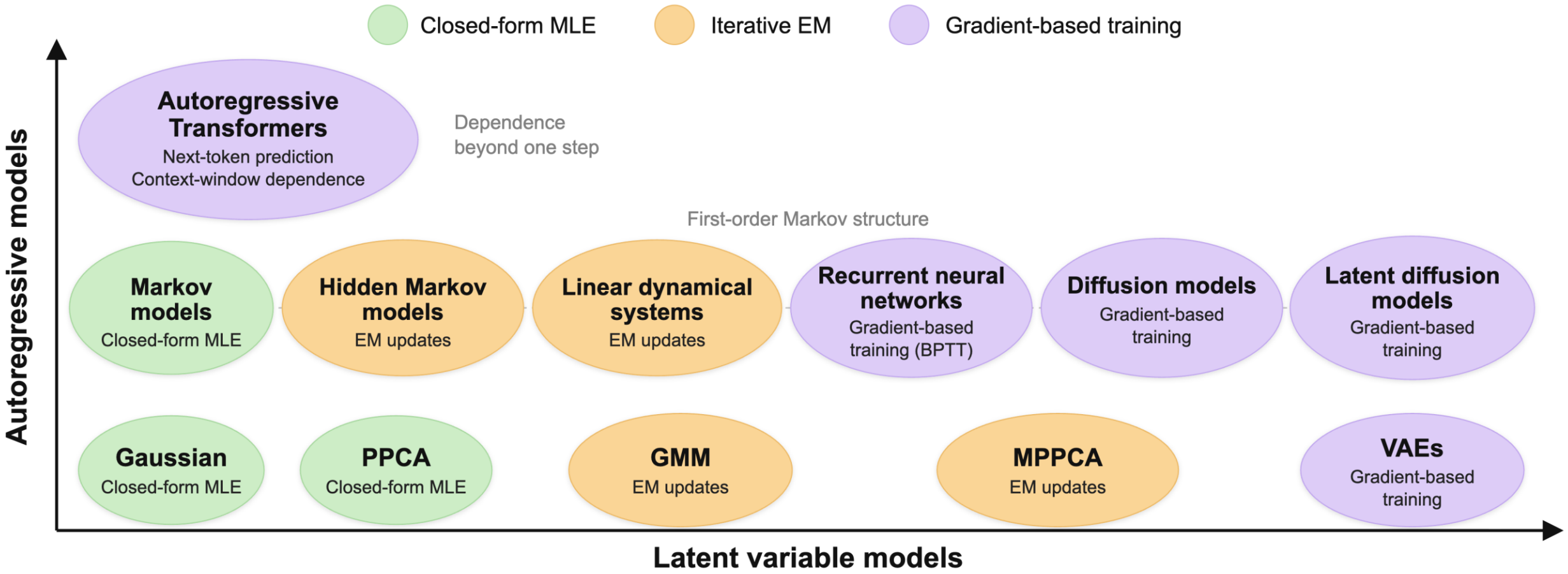
Deep Generative Models: Recurrent Neural Networks and Attention Mechanisms

Fall Semester 2026

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



Penn
University of Pennsylvania

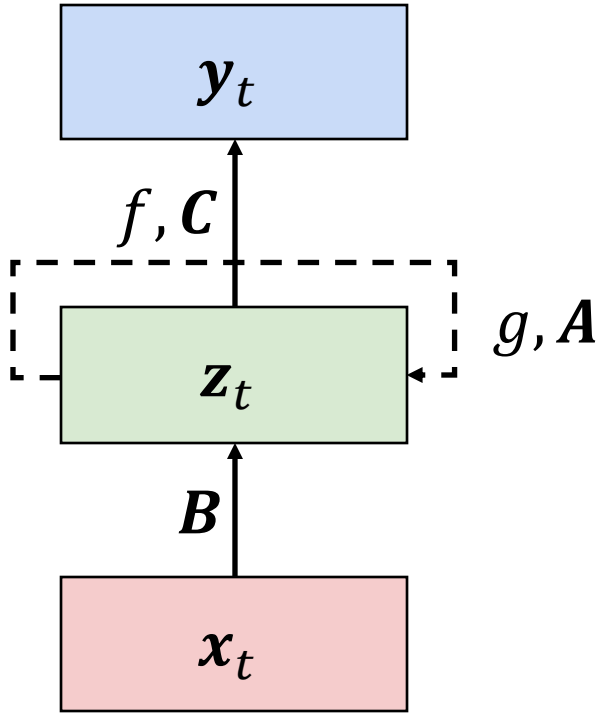
Autoregressive Models

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

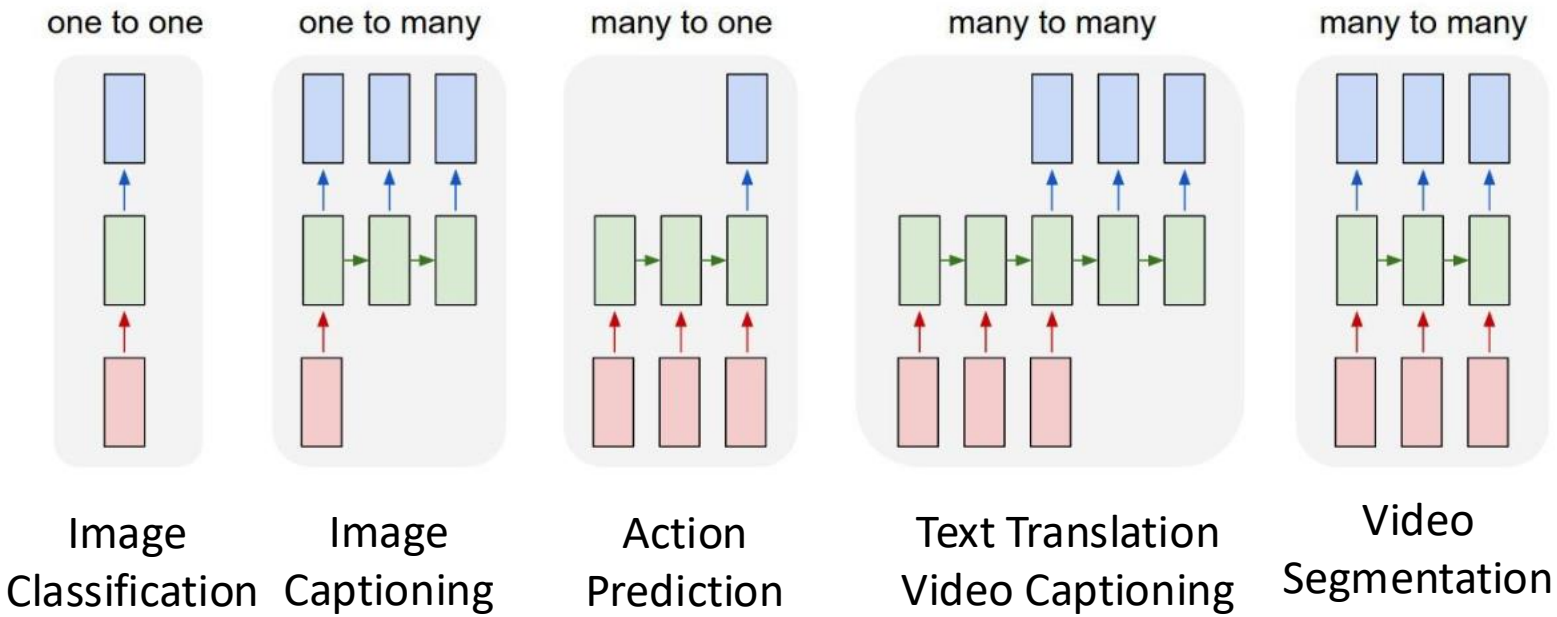
- Last lecture
  - **Model**: Introduced the vanilla RNN architecture
  - **Inference**: Unfolding
  - **Training**: Backpropagation Through Time, Vanishing and Exploding Gradients
  - **Variants of RNNs**: LSTMs, GRUs

This Lecture

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

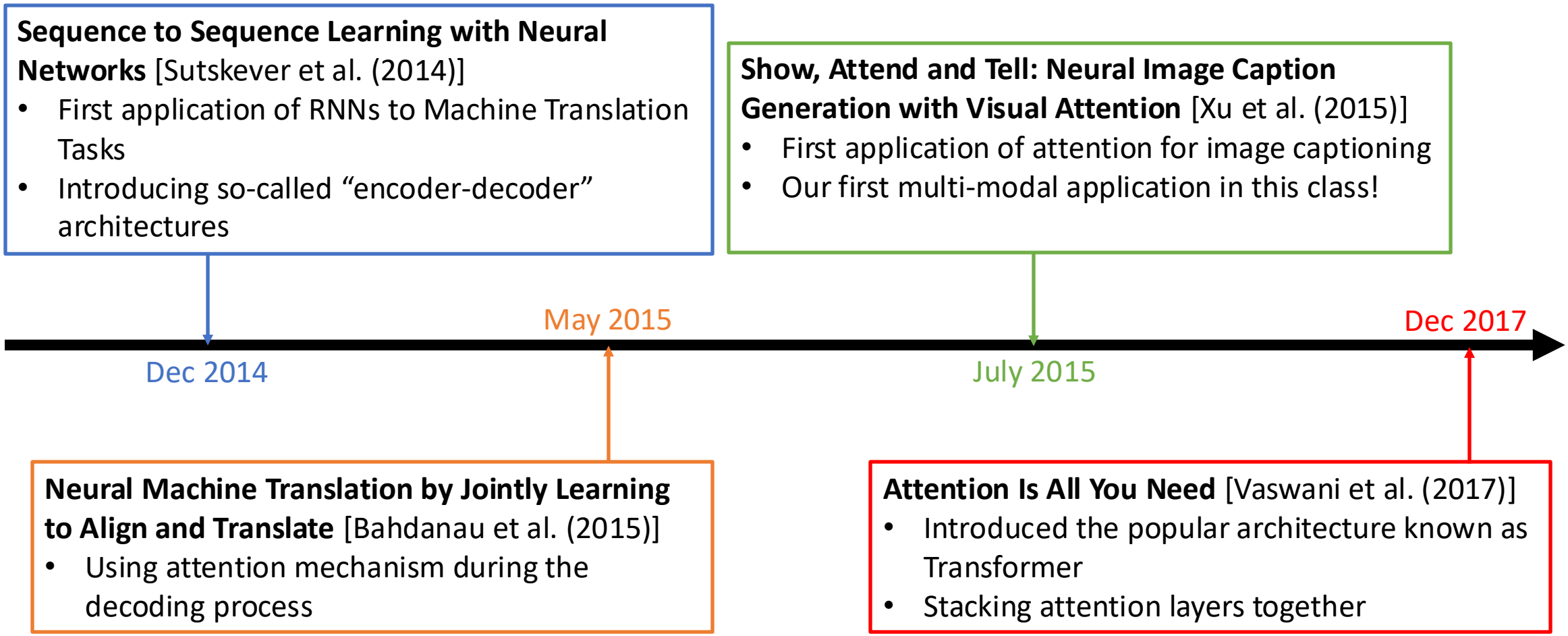


- Generalizations of Vanilla Neural Networks: RNNs can be very flexible, depending on the task!



Timeline of Attention Mechanisms

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



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

$$\hat{\boldsymbol{y}}_{1:T_y} = \underset{\boldsymbol{y}_{1:T_y}}{\operatorname{argmax}}\, p_\theta(\boldsymbol{y}_{1:T_y} \mid \boldsymbol{x}_{1:T_x})$$

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

Consider More General RNN Models

- The RNNs discussed thus far used linear layers and point-wise non-linearities

$$\boldsymbol{z}_t = g(\boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{B}\boldsymbol{x}_t) + \boldsymbol{w}_t$$
$$\boldsymbol{y}_t = f(\boldsymbol{C}\boldsymbol{z}_t) + \boldsymbol{v}_t$$

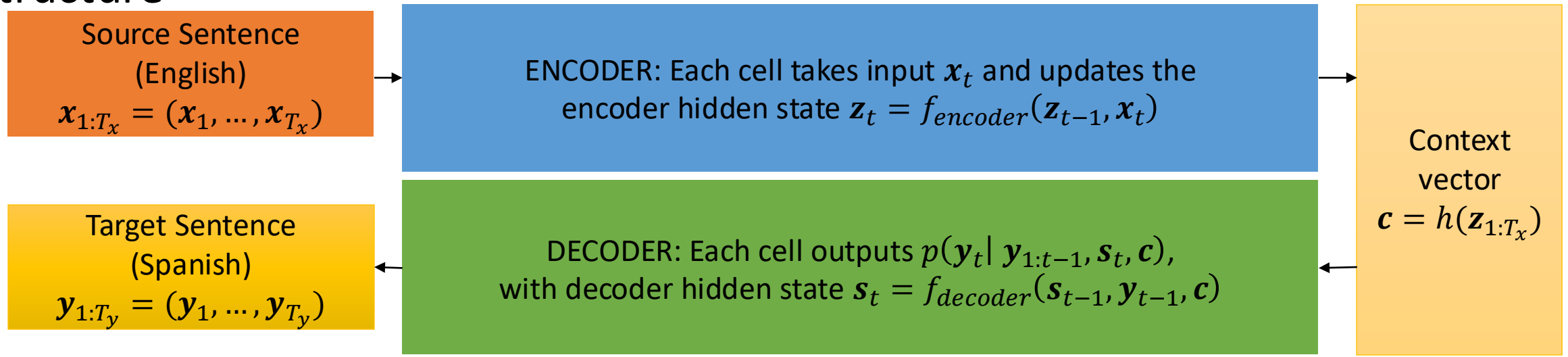
- More generally, we can design RNNs with general non-linear maps (e.g., LSTMs)

$$\boldsymbol{z}_t = f_{enc}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_t)$$
$$\boldsymbol{y}_t = f_{dec}(\boldsymbol{z}_t, \boldsymbol{x}_t)$$

- We remove the noise terms to obtain deterministic "encoders" and "decoders" conditioned on the stochastic input $\boldsymbol{x}_t$, and thus "avoid" the need to do EM or VI for learning the model parameters.

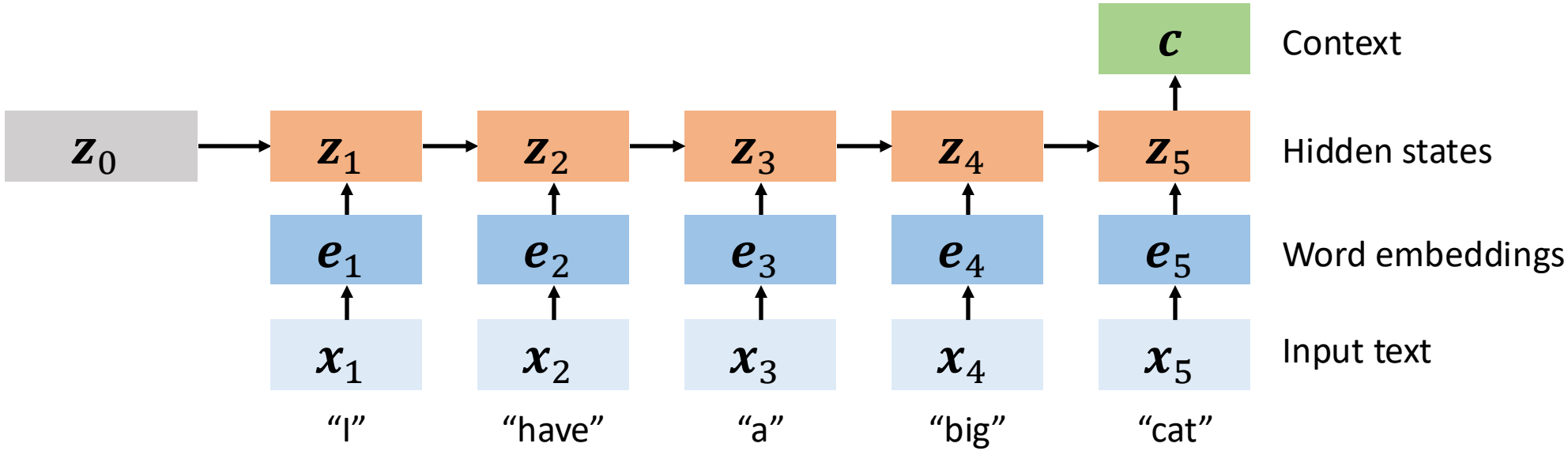
Consider Integrating Multiple RNN Models for MT

- The high-level idea is as follows:
  - An RNN allows us to encode the source (English) sentence $\boldsymbol{x}_{1:T_x}$ into a sequence of hidden states $\boldsymbol{z}_{1:T_x}$. We then summarize these encoder hidden states into a context vector $\boldsymbol{c} = h(\boldsymbol{z}_{1:T_x})$ which represents the source sentence.
  - Once the semantics are captured in $\boldsymbol{c}$, a decoder RNN generates the target (Spanish) sentence $\boldsymbol{y}_{1:T_y}$, with decoder hidden states $\boldsymbol{s}_{1:T_y}$.
  - We condition the decoder on $\boldsymbol{y}_{t-1}$ rather than $\boldsymbol{y}_t$ so that we can use it during inference.
- A similar structure can be found in VAEs, where we also have an encoder-decoder structure



Structure of the Encoder

- Recall RNN Encoder for next word prediction, and modify it to produce a context



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

Source Sentence (English) $x_{1:T_x} = (x_1, \ldots, x_{T_x})$

→

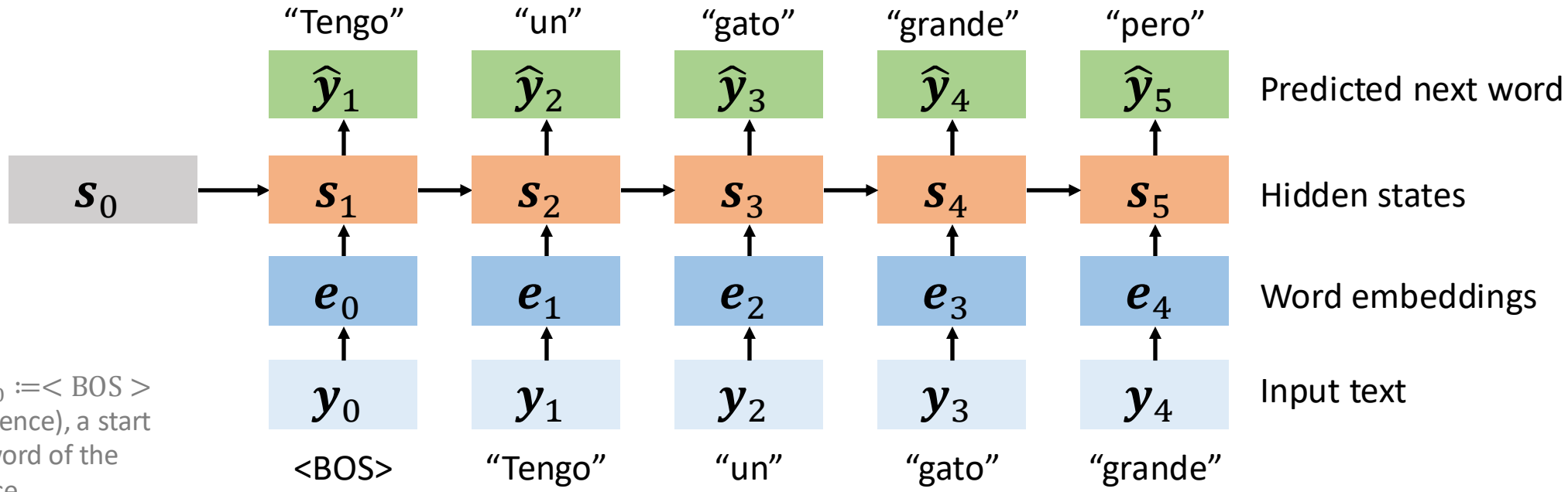
ENCODER: Each cell takes input x_t and updates the encoder hidden statestate $z_t = f_{encoder}(z_{t-1}, x_t)$

→

Context vector $c = h(z_{1:T_x})$

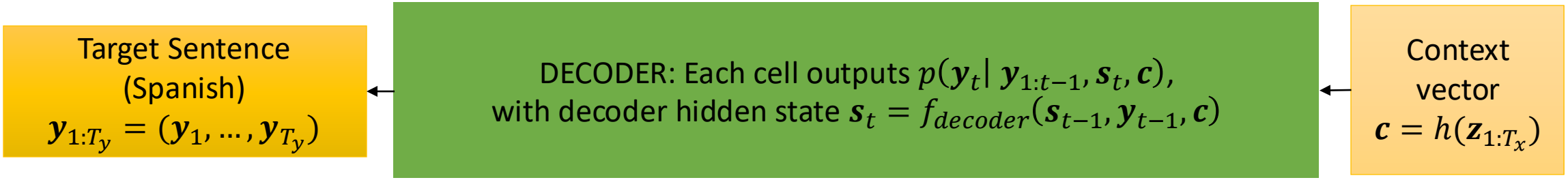
Structure of the Decoder

- Recall RNN for next word prediction

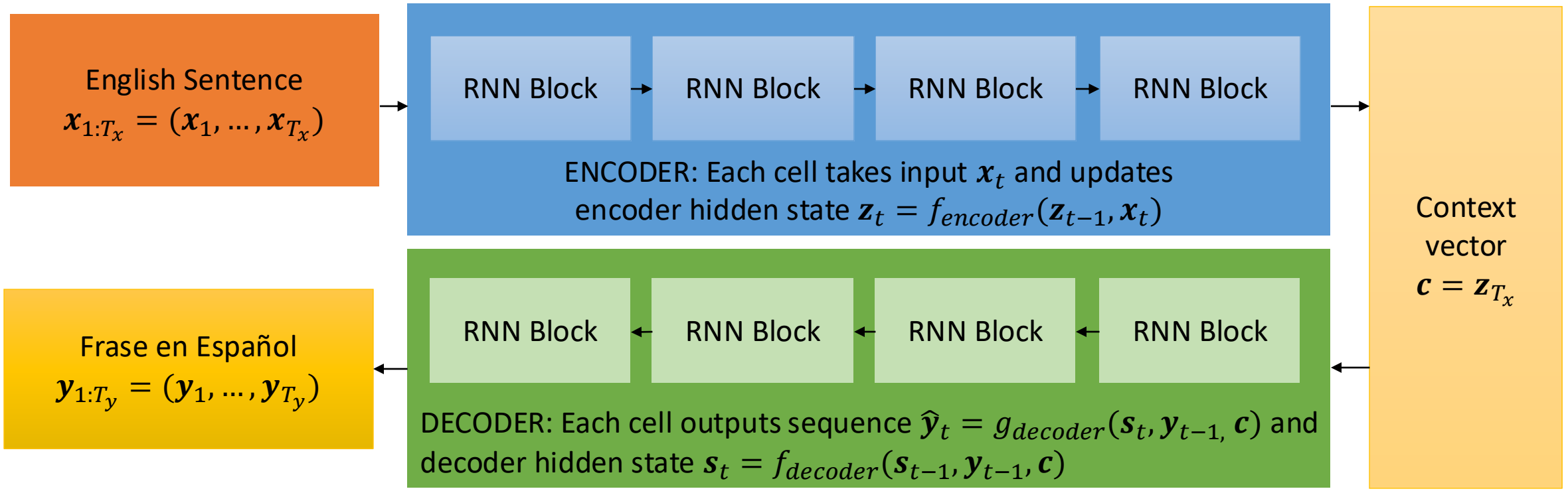


Note: Here, $y_0 :=< \text{BOS} >$ (Begin of Sentence), a start token, not a word of the target sentence.

- We now augment it with context



RNN Encoder-Decoder Architecture



- Remarks on architecture from Sutskever et al. (2014) seq2seq paper:
  - $f_{encoder}$ and $f_{decoder}$ are implemented using LSTM layers; $g_{decoder}$ is an output layer.
  - The fixed context vector is taken directly from the final encoder hidden state, i.e., $\boldsymbol{c} = \boldsymbol{z}_{T_x}$.
  - A $g_{encoder}$ is not needed because the encoder is used only to produce hidden states and the context vector.

Learning and Inference

- **Learning:** Given N samples $\left\{\left(\boldsymbol{x}_{1:T_x}^{(n)}, \boldsymbol{y}_{1:T_y}^{(n)}\right)\right\}_{n=1}^{N}$ of source-target sentence pairs, we learn the encoder-decoder parameters θ by conditional maximum likelihood.
- We compute the context vector independently of the latent states as

$$\boldsymbol{c}^{(n)} = f_{encoder,\theta}\left(\boldsymbol{x}_{1:T_x}^{(n)}\right).$$

- We factorize the conditional model autoregressively:

$$p_\theta\left(\boldsymbol{y}_{1:T_y}^{(n)} \mid \boldsymbol{x}_{1:T_x}^{(n)}\right) = \prod_{t=1}^{T_y} p_\theta\left(\boldsymbol{y}_t^{(n)} \mid \boldsymbol{y}_{1:t-1}^{(n)}, \boldsymbol{c}^{(n)}\right).$$

- Therefore, we maximize the conditional log-likelihood without EM or VI

$$\max_\theta \frac{1}{N}\sum_{n=1}^{N} \log p_\theta(\boldsymbol{y}_{1:T_y}^{(n)} \mid \boldsymbol{x}_{1:T_x}^{(n)}) = \max_\theta \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_y} \log p_\theta\left(\boldsymbol{y}_t^{(n)} \mid \boldsymbol{y}_{1:t-1}^{(n)}, \boldsymbol{c}^{(n)}\right).$$

Learning and Inference

- **Inference:** To decode, we seek the target sentence with the highest probability.
- For a given $\boldsymbol{x}_{1:T_x}$, compute the context vector

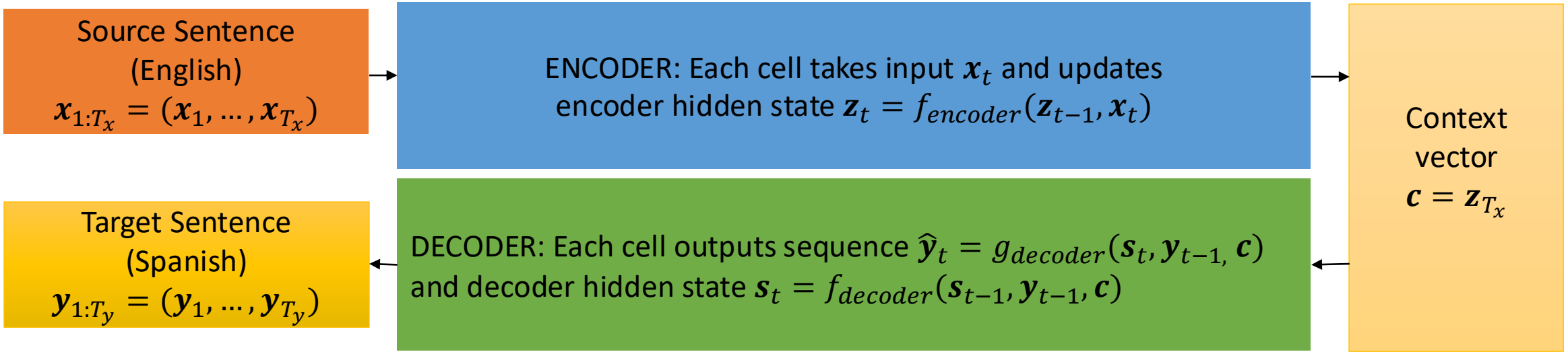
$$\boldsymbol{c} = f_{encoder,\theta}(\boldsymbol{x}_{1:T_x}).$$

- Then

$$\begin{aligned}\hat{\boldsymbol{y}}_{1:T_y} &= \operatorname{argmax}_{\boldsymbol{y}_{1:T_y}} p_\theta\left(\boldsymbol{y}_{1:T_y} \mid \boldsymbol{x}_{1:T_x}\right) \\ &= \operatorname{argmax}_{\boldsymbol{y}_{1:T_y}} \sum_{t=1}^{T_y} \log p_\theta(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{c}).\end{aligned}$$

- Note that exact maximization (e.g., Viterbi++) over all sentences is expensive.
- In practice, we use **greedy decoding** (keep the single best next token at each step) or **beam search** (keep multiple candidate sequences with the highest cumulative log-probability).

Limitations of Fixed-Context seq2seq Models

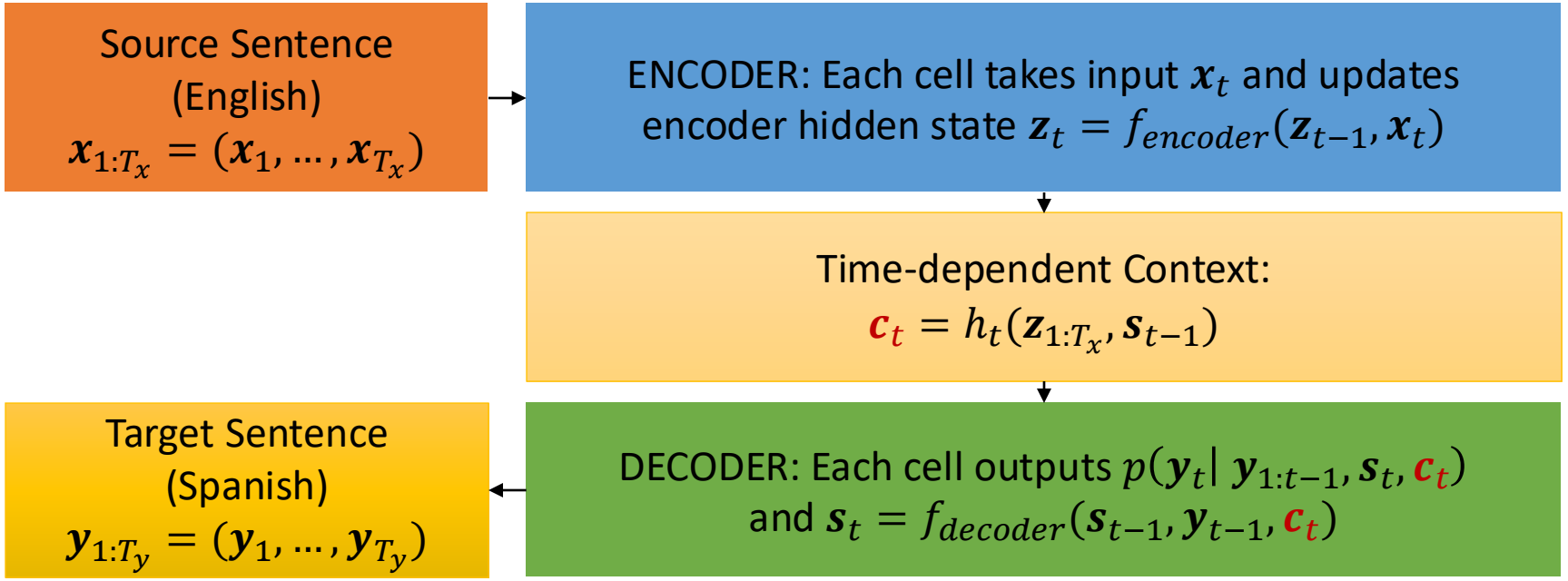


- However, there are obvious limitations to this design:
  - **Encoding**: the context c may not be able to capture earlier parts of the source sentence
  - **Fixed-length Context:** All the information from the source sentence is "jammed" into the single context vector c.

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

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 improve modeling of long-range dependencies.



- How should we calculate $\boldsymbol{c}_t = h_t(\boldsymbol{z}_{1:T_x}, \boldsymbol{s}_{t-1})$ and model $p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{s}_t, \boldsymbol{c}_t)$?

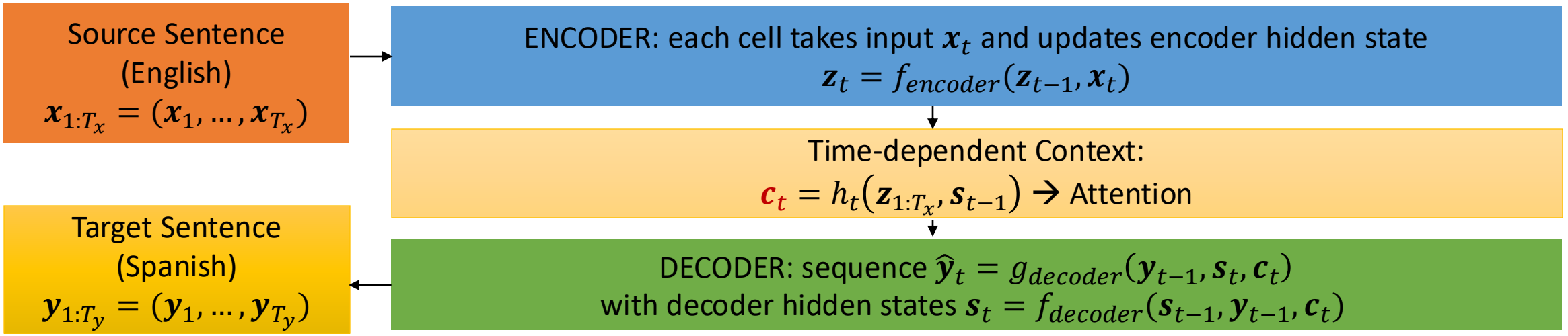
Align and Translate [Bahdanau et al. (2015)]

- **Intuition:** Translation of the word $\boldsymbol{x}_j$ to $\boldsymbol{y}_t$ depends on the contexts of both the source sentence $\boldsymbol{x}_{1:T_x}$ and target sentence $\boldsymbol{y}_{1:T_y}$.
  - The hidden states 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**: At decoding step t, use the previous decoder hidden state $\boldsymbol{s}_{t-1}$ to determine which encoder hidden states $\boldsymbol{z}_{1:T_x}$ are most relevant, and combine them into the context vector $\boldsymbol{c}_t$.

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



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

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

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

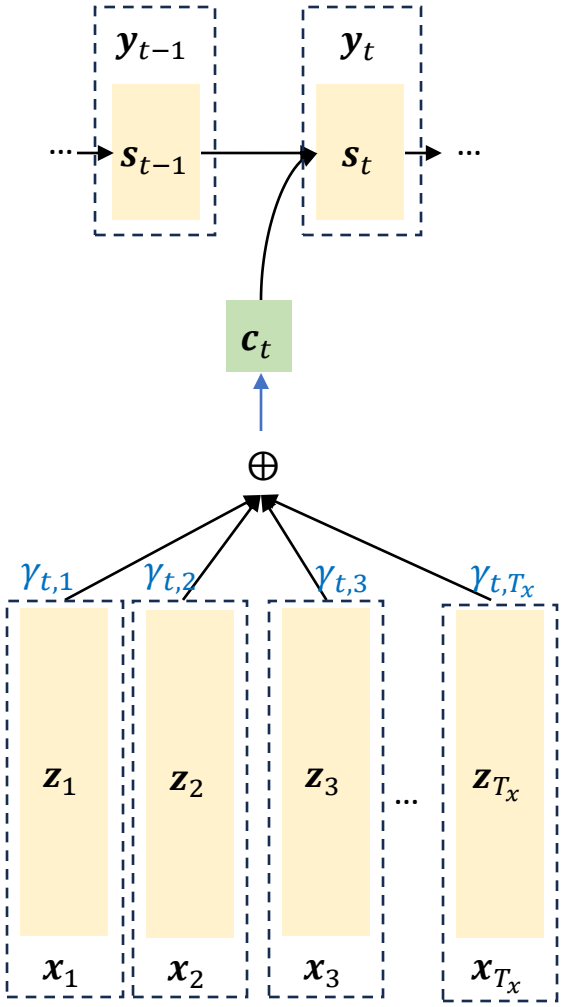
[1] Bahdanau et al. (2015): https://arxiv.org/pdf/1409.0473

Align and Translate

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

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

Context vector Weights of hidden states Alignment model



- Here:
  - c_t is the expected hidden state over all the hidden states with probability $\gamma_{t,j}$.
  - $\gamma_{t,j}$ 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 $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$ using input $\boldsymbol{x}_{1:T} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)$. Concatenate them as one encoder hidden state $\boldsymbol{z}_t = \left[\overrightarrow{\boldsymbol{h}_t} \middle| \overleftarrow{\boldsymbol{h}_t}\right]$ (assume they are row vectors). Hidden states are also called *annotations.*
  - **Decoder:** Using a single direction RNN with Attention mechanism and alignment model

$$a(\boldsymbol{s}_{t-1}, \boldsymbol{z}_j) = \boldsymbol{v}_a^\top \tanh(\boldsymbol{W}_a \boldsymbol{s}_{t-1} + \boldsymbol{U}_a \boldsymbol{z}_j)$$

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

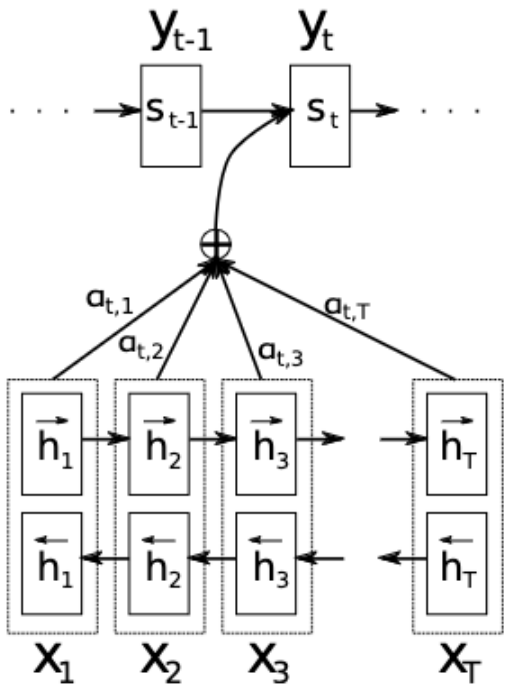


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \dots, x_T)$.

[1] Bahdanau et al. (2015): https://arxiv.org/pdf/1409.0473

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 source words
- Diagonal: $\boldsymbol{x}_t$ matches with $\boldsymbol{y}_t$
- Cross-Diagonal: context dependent

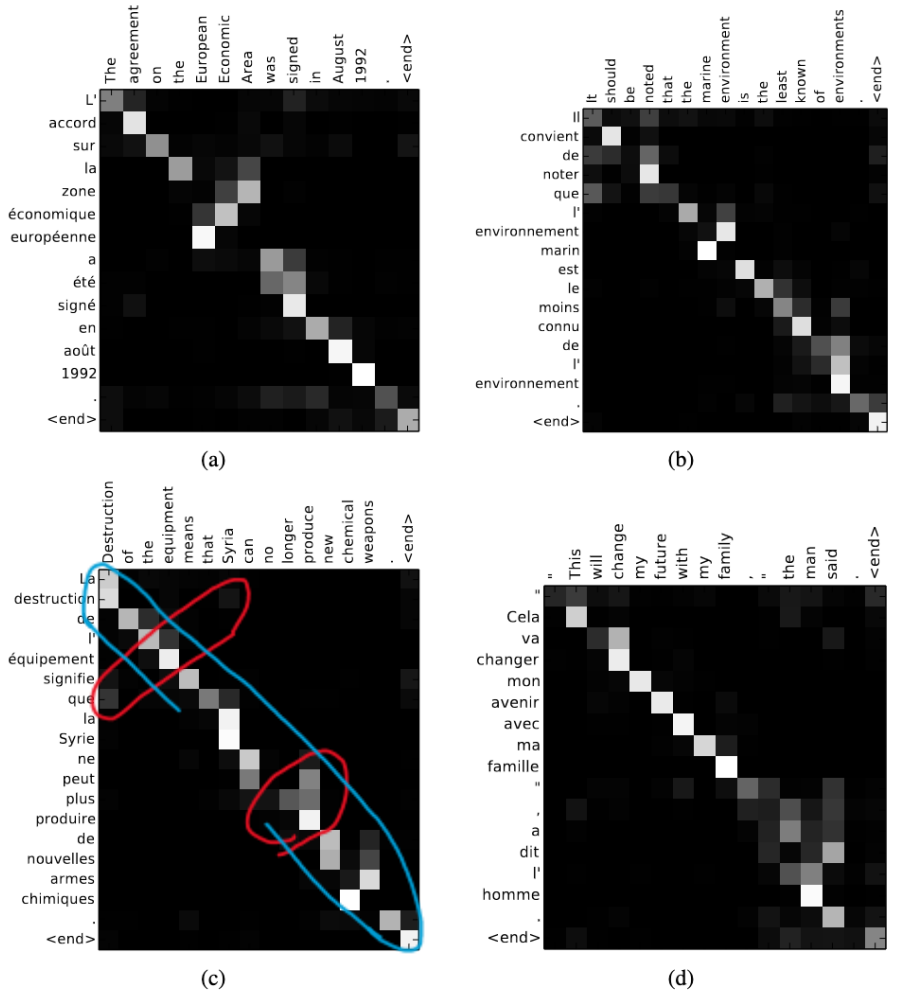


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (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 a unidirectional one
  - **Alignment model** instead of single fixed-vector hidden states
  - Have context vector $\boldsymbol{c}_t$ that depends on the timestep

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

Deep Generative Models: VAE+RNN for Image Captioning

Fall Semester 2026

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 (2015) “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.

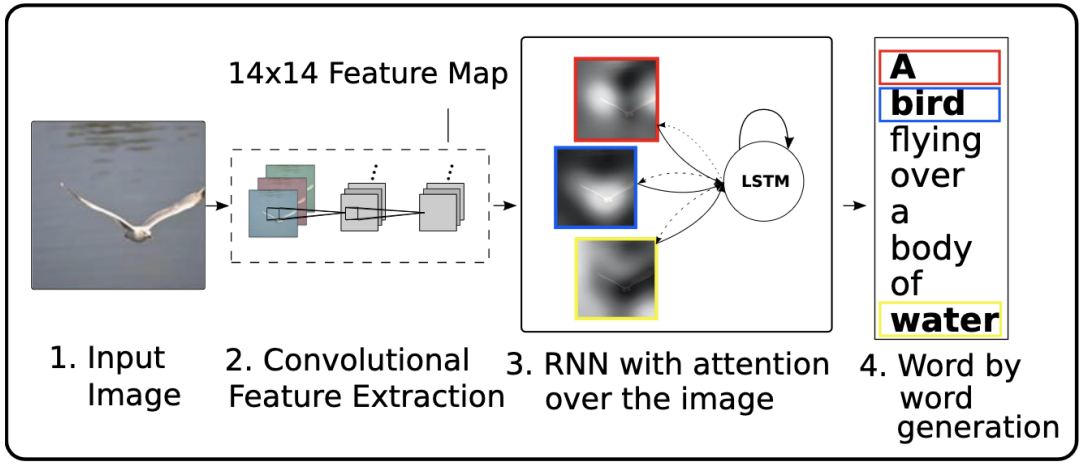
A woman throwing a frisbee in a park.



[1] Xu et al (2015). https://proceedings.mlr.press/v37/xuc15.html

Task

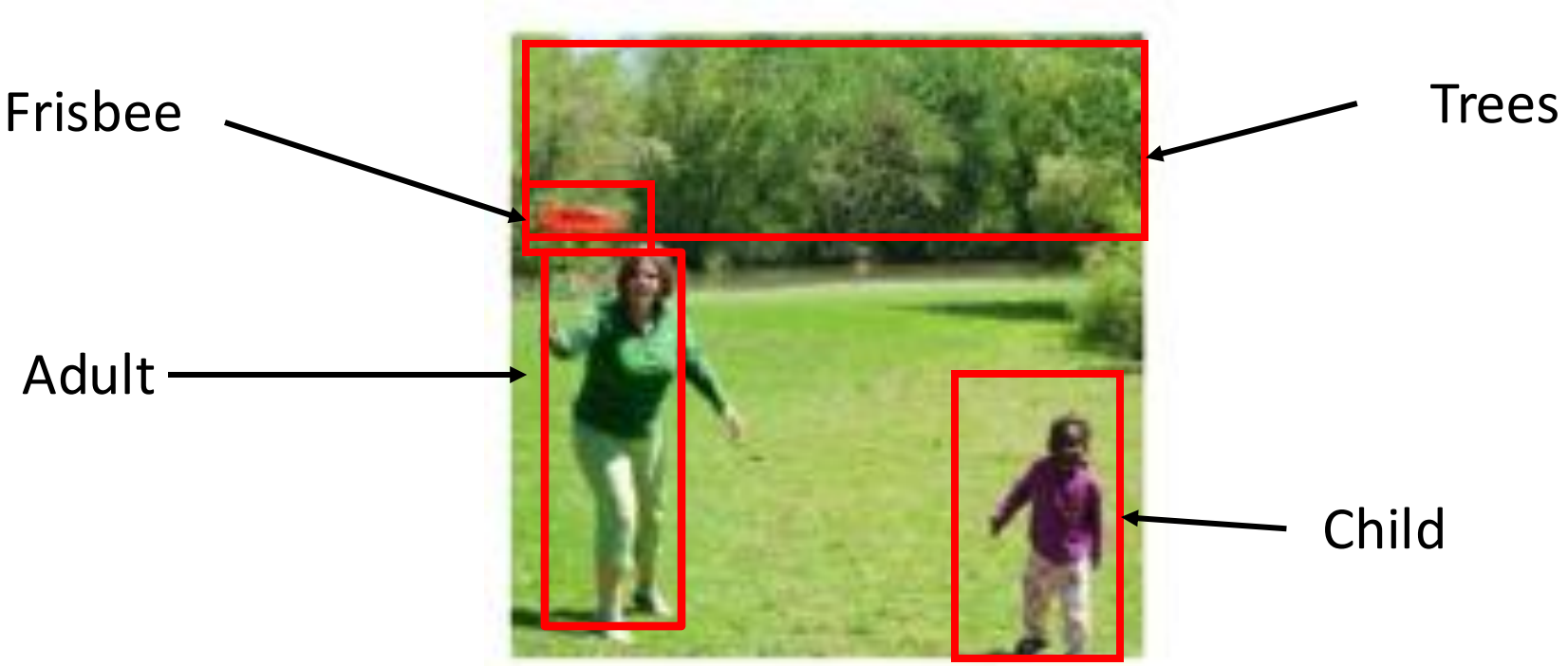
- Today we will talk about Image Caption Generation using a combination of Variational Auto Encoders (VAEs) and Recurrent Neural Networks (RNNs).
- Overall pipeline:



- Similar to any language task, suppose we are given a vocabulary of size $|V|$, a sentence of length T_y can be presented by each word being a one-hot embedding:
  - $\boldsymbol{y}_{1:T_y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T_y}\}, \boldsymbol{y}_t \in \{0,1\}^{|V|}$.

Image Encoder

- An image can have many sources of information.



- Ideally the 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 spatial feature vectors $\boldsymbol{x}_i$ from a convolutional feature map.
- Denote the feature vector at spatial location i by $x_i \in \mathbb{R}^D, i = 1,2, \dots, T_x$.
- Collecting all feature vectors as columns,

$$\boldsymbol{x}_{1:T_x} = [\boldsymbol{x}_1 | \dots | x_{T_x}] \in \mathbb{R}^{D \times T_x},$$

where T_x is the number of spatial locations and D is the dimension of each feature vector.

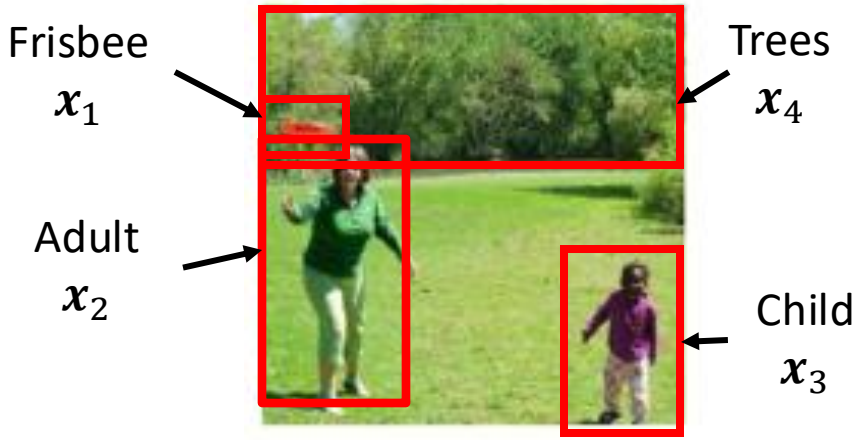
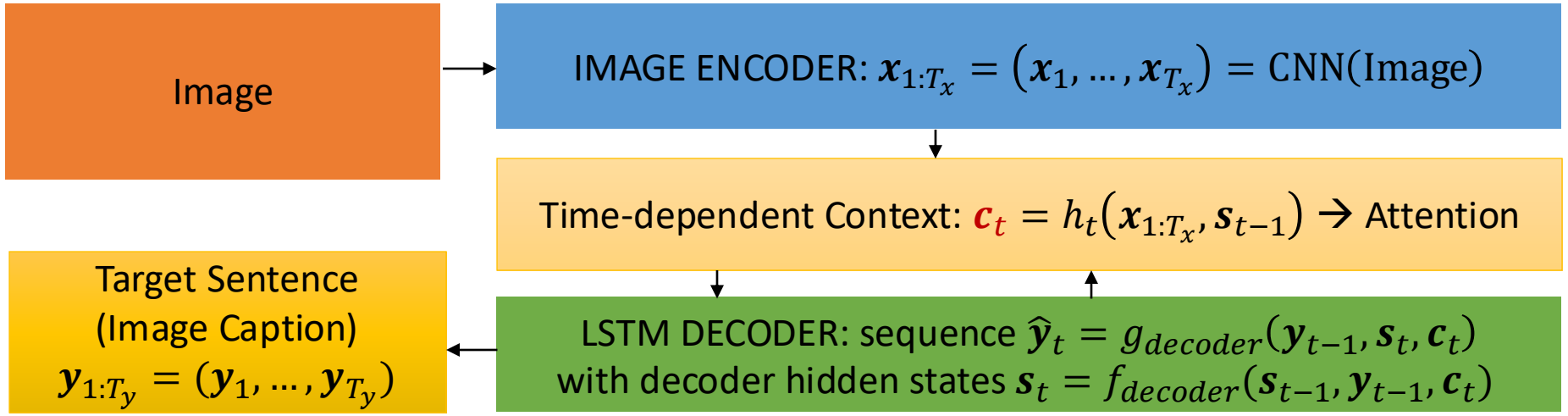


Figure above: In an ideal situation, each semantic part is presented by a low-level feature vector $\boldsymbol{x}_i$.

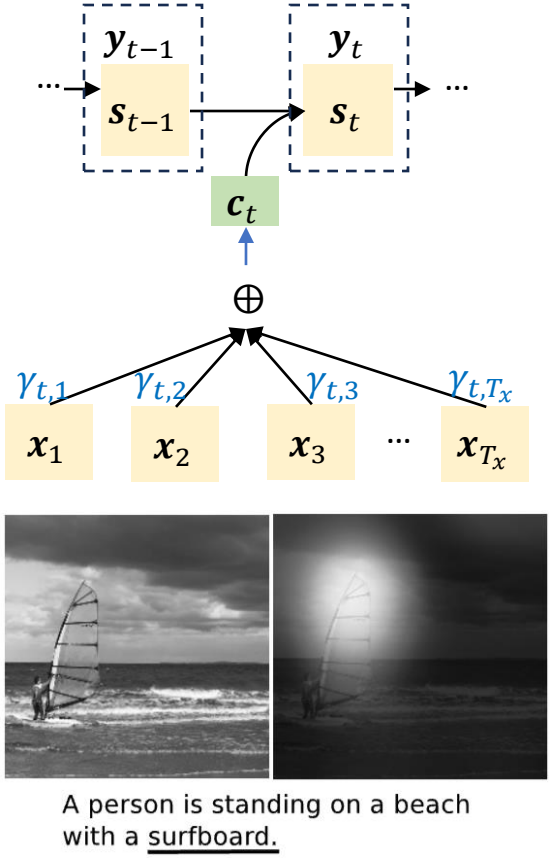
Decoder: LSTM with Context



- Similar to Align and Translate, now we need 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 $\boldsymbol{c}_t = h_t(\boldsymbol{x}_{1:T_x}, \boldsymbol{s}_{t-1})$ computed using attention?

Decoder: Context Vector and Attention



- c_t is a context vector representing 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 selected.
  - Option 2: ϕ = **Soft Attention:** all T_x image locations are combined using attention weights.
- Similar to Align and Translate model, we can define:

$$c_t = \phi(x_1, \ldots, x_{T_x}, \gamma_{t,1}, \ldots, \gamma_{t,T_x})$$

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

$$\gamma_{t,j} = \frac{\exp(e_{t,j})}{\sum_{k=1}^{T_x} \exp(e_{t,k})}$$

Attention weights over the T_x positions

$$e_{t,j} = a(s_{t-1}, x_j)$$

"Attention Model"
a multi-layer perceptron

Image features $x_1, \ldots, x_{T_x}$
Decoder hidden states $s_1, \ldots, s_{T_y}$

Option 1 for ϕ: Stochastic Hard Attention

- Stochastic Hard Attention implies we use an "on-off" way to choose which location of the image to focus.
  - This means we can only choose one location each time.
- Let $\hat{\boldsymbol{\gamma}}_t \in \{0,1\}^{T_x}$ 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

$$p(\hat{\gamma}_{t,i} = 1 \,|\, \hat{\gamma}_{1:t-1}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T_x}) = \gamma_{t,i}\,, \qquad \boldsymbol{c}_t = \sum_{i=1}^{T_x} \hat{\gamma}_{t,i} \boldsymbol{x_i}\,.$$

- This means we can treat $\hat{\boldsymbol{\gamma}}_t$ as a categorical distribution:

$$\hat{\boldsymbol{\gamma}}_t \sim \text{Categorical}(\gamma_{t,1}, \ldots, \gamma_{t,T_x}), \qquad \sum_{i=1}^{T_x} \hat{\gamma}_{t,i} = 1\,.$$

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

Stochastic Hard Attention (Learning)

- While it is intuitive to parameterize $\widehat{\boldsymbol{\gamma}}_t \sim \text{Categorical}(\gamma_{t,1}, \ldots, \gamma_{t,T_x})$, 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 $\log p\left(\boldsymbol{y}_{1:T_y} \,\middle|\, \boldsymbol{x}_{1:T_x}\right)$ of observing the sequence of words $\boldsymbol{y}_{1:T_y}$ given image features $\boldsymbol{x}_{1:T_x}$.
- Quick Recall: Let X and Z be two random variables, 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 $\boldsymbol{x} \sim p_\theta$ and any distribution q_ψ, we have

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

Stochastic Hard Attention (Learning)

- Analogous to latent-variable models such as VAEs, we treat the stochastic attention locations $\widehat{\boldsymbol{\gamma}}_{1:T_y}$ as latent variables. The context vectors $\boldsymbol{c}_t$ are deterministic functions of these latent attention variables and the image features.
- Define
  - ψ as the parameters of the attention model $q_\psi(\widehat{\boldsymbol{\gamma}} \mid \boldsymbol{x})$, which defines a distribution over latent attention locations.
  - θ as the parameters of the caption decoder $p_\theta(\boldsymbol{y} \mid \widehat{\boldsymbol{\gamma}}, \boldsymbol{x})$.
- The Hard-Attention Lower Bound:

$$\mathcal{L}_{\theta,\psi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\widehat{\boldsymbol{\gamma}}} q_\psi(\widehat{\boldsymbol{\gamma}} \mid \boldsymbol{x}) \log p_\theta(\boldsymbol{y} \mid \widehat{\boldsymbol{\gamma}}, \boldsymbol{x})$$

$$\leq \log \sum_{\widehat{\boldsymbol{\gamma}}} q_\psi(\widehat{\boldsymbol{\gamma}} \mid \boldsymbol{x}) p_\theta(\boldsymbol{y} \mid \widehat{\boldsymbol{\gamma}}, \boldsymbol{x}) \qquad \text{(Jensen's Inequality)}$$

$$= \log p_{\theta,\psi}(\boldsymbol{y} \mid \boldsymbol{x}) \qquad \text{(Marginal Log-Likelihood)}$$

Gradient of the Hard-Attention Lower Bound

- The hard-attention lower bound: $\mathcal{L}_{\theta,\psi}(\boldsymbol{x},\boldsymbol{y}) = \sum_{\widehat{\boldsymbol{\gamma}}} q_\psi(\widehat{\boldsymbol{\gamma}}|\,\boldsymbol{x}) \log p_\theta(\boldsymbol{y}\,|\widehat{\boldsymbol{\gamma}},\boldsymbol{x})$.
- For the decoder parameters θ:

$$\nabla_\theta \mathcal{L} = \sum_{\widehat{\boldsymbol{\gamma}}} q_\psi(\widehat{\boldsymbol{\gamma}}|\,\boldsymbol{x}) \nabla_\theta \log p_\theta(\boldsymbol{y}\,|\widehat{\boldsymbol{\gamma}},\boldsymbol{x})\,.$$

- For the attention parameters ψ:

$$\nabla_\psi \mathcal{L} = \sum_{\widehat{\boldsymbol{\gamma}}} q_\psi(\widehat{\boldsymbol{\gamma}}|\,\boldsymbol{x}) \log p_\theta(\boldsymbol{y}\,|\widehat{\boldsymbol{\gamma}},\boldsymbol{x})\, \nabla_\psi \log q_\psi(\widehat{\boldsymbol{\gamma}}|\,\boldsymbol{x}).$$

- Estimate these using Monte Carlo samples:

$$\widehat{\gamma}^{(m)} \sim q_\psi(\widehat{\boldsymbol{\gamma}}|\,\boldsymbol{x}), \qquad m = 1, \ldots, M.$$

M: number of Monte Carlo samples.

- Then

$$\nabla_\theta \mathcal{L} \approx \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log p_\theta(\boldsymbol{y}\,|\widehat{\boldsymbol{\gamma}}^{(m)},\boldsymbol{x}),$$

$$\nabla_\psi \mathcal{L} \approx \frac{1}{M} \sum_{m=1}^{M} \log p_\theta(\boldsymbol{y}\,|\widehat{\boldsymbol{\gamma}}^{(m)},\boldsymbol{x})\, \nabla_\psi \log q_\psi\big(\widehat{\boldsymbol{\gamma}}^{(\boldsymbol{m})}\big|\,\boldsymbol{x}\big).$$

Option 2 for ϕ: Deterministic "Soft" Attention

- Recall our three equations:

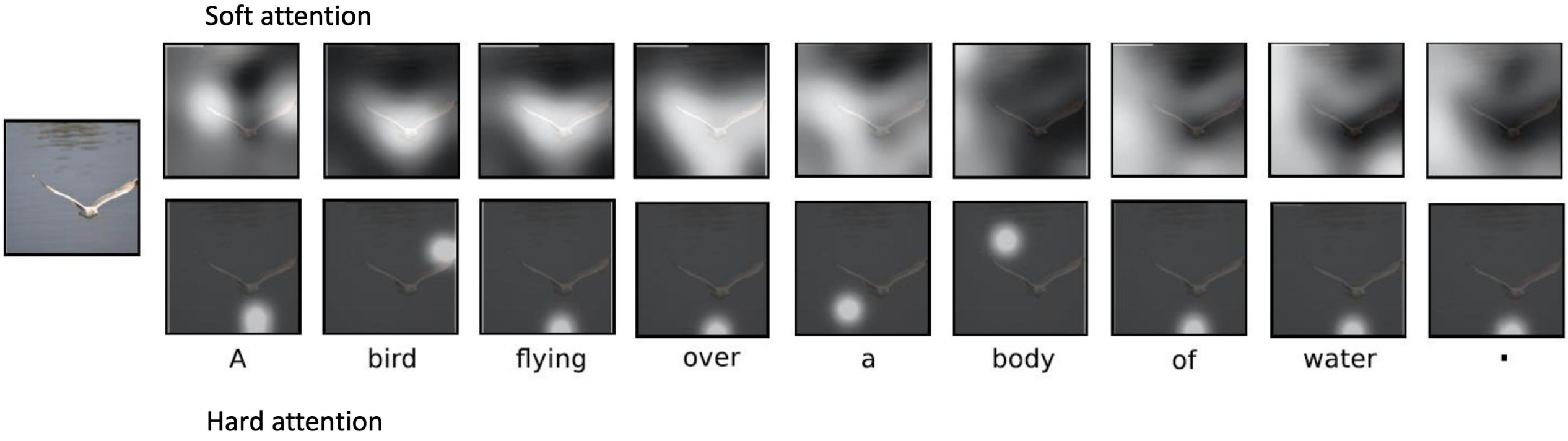
$$\boldsymbol{c}_t = \phi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T_x}, \gamma_{t,1}, \ldots, \gamma_{t,T_x}) \qquad \gamma_{t,j} = \frac{\exp(e_{t,j})}{\sum_{k=1}^{T_x} \exp(e_{t,k})} \qquad e_{t,i} = a(\boldsymbol{x}_i, \boldsymbol{s}_{t-1})$$

- Hard attention method requires us to sample the attention location $\widehat{\boldsymbol{\gamma}}_t$ at each decoding step.
- Soft attention uses the expected context under the attention distribution:

$$\boldsymbol{c}_t = \phi\left(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T_x}, \gamma_{t,1}, \ldots, \gamma_{t,T_x}\right) = \sum_{i=1}^{T_x} \gamma_{t,i} \boldsymbol{x}_i \,.$$

- Then this would no longer be a "on-off" mechanism, but a weighted sum of image feature vectors.
- Lucky for us, because soft attention is deterministic and differentiable, the whole model can be trained end-to-end with backpropagation using the captioning cross-entropy loss.

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)



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



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



A little girl sitting on a bed with a teddy bear.



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



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

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 categorical variable (Hard Attention), use Monte Carlo sampling 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 $\boldsymbol{c}$
  - Align and Translate for Machine Translation
    - The use of multiple time-dependent context vectors $\boldsymbol{c}_t$
  - 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
- 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 on “Attention” completely to capture global dependencies