Lecture 07: Sequence Models and Attention

Lan Xu
SIST, ShanghaiTech
Fall, 2023

Outline

- RNNs in Vision and NLP
  - X-to-sequence model: NMT and Image Captioning
  - Attention models: NMT and Image Captioning

Acknowledgement: Feifei Li et al's cs231n notes

X-to-Sequence problems

- Sequence or non-sequence in, sequence comes out
  - Machine translation

  - Image caption generation

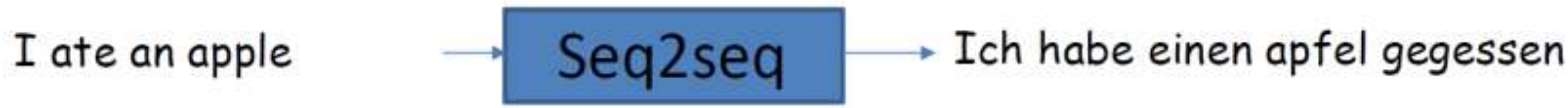
Img2seq

A train on the train tracks.

- No notion of "synchrony" between input and output
  - May even not have a notion of "alignment"

Sequence-to-sequence problem

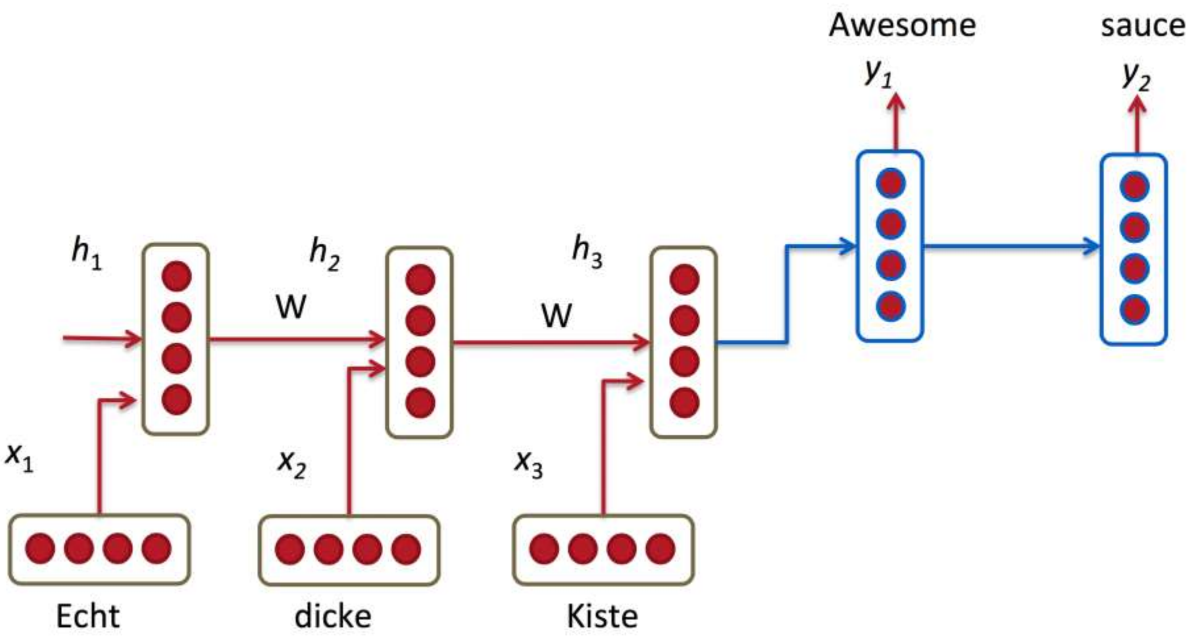
- Task definition



  - A sequence $X_1, \cdots, X_N$ goes in
  - A different sequence $Y_1, \cdots, Y_M$ comes out
  - Example: machine translation
    - The output is in a different language
  - Example: dialog
    - “I have a problem” -> “How may I help you”

Modeling the problem

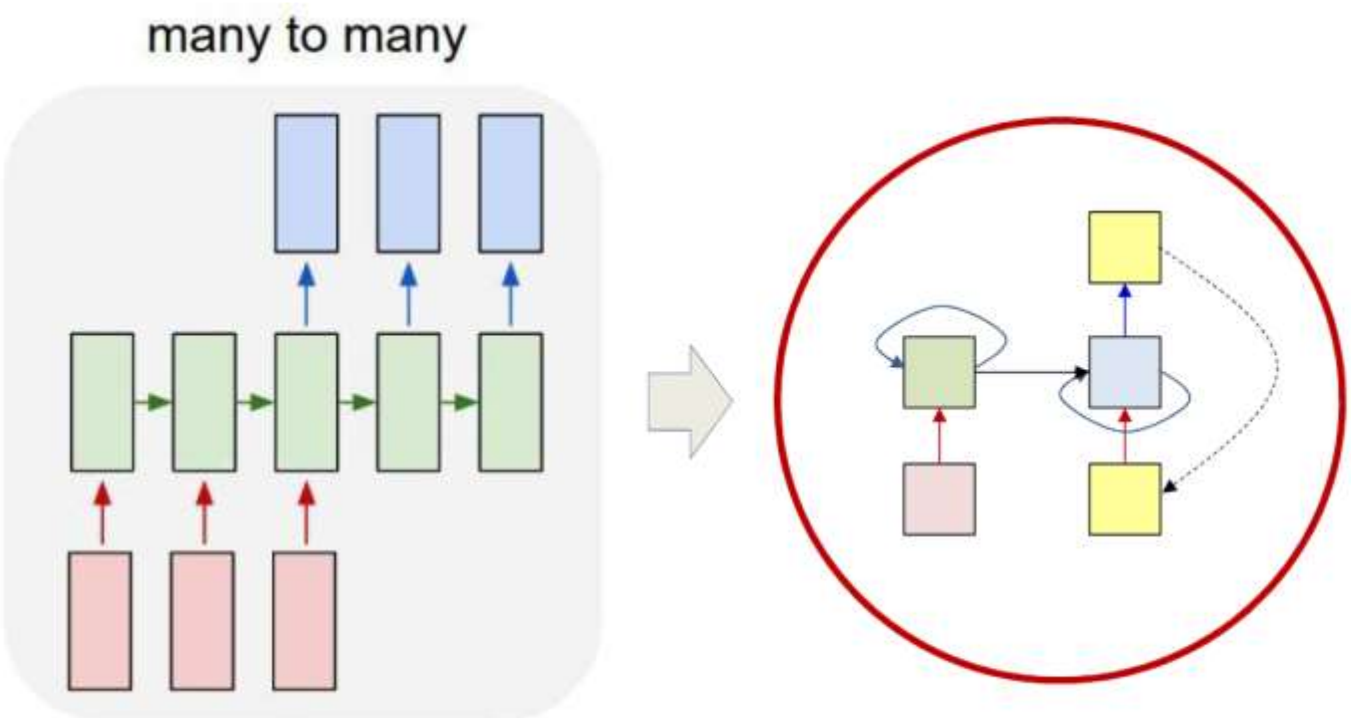
- Delayed sequence to sequence
  - Many to many



http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

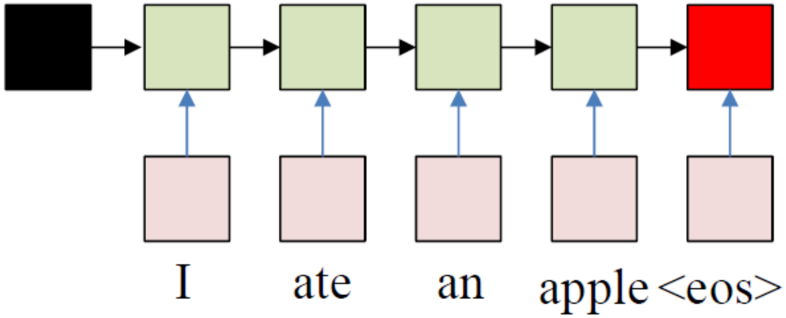
Modeling the problem

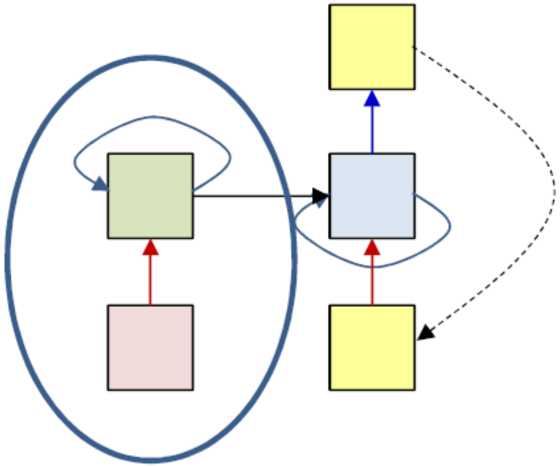
- Delayed sequence to sequence
  - Delayed self-referencing sequence-to-sequence



The “simple” translation model

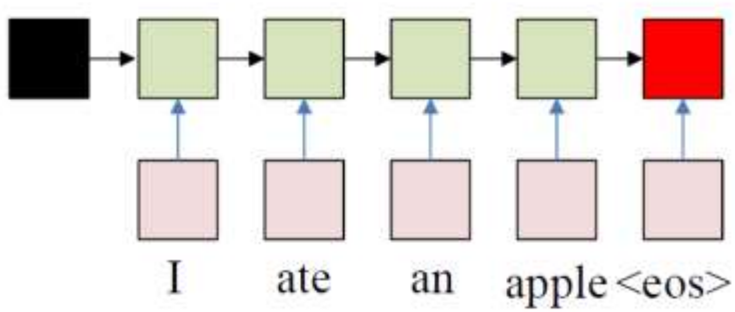
- The input sequence feeds into an recurrent structure
  - The input sequence is terminated by an explicit <eos> symbol
- The hidden activation at the <eos> “stores” all information about the sentence

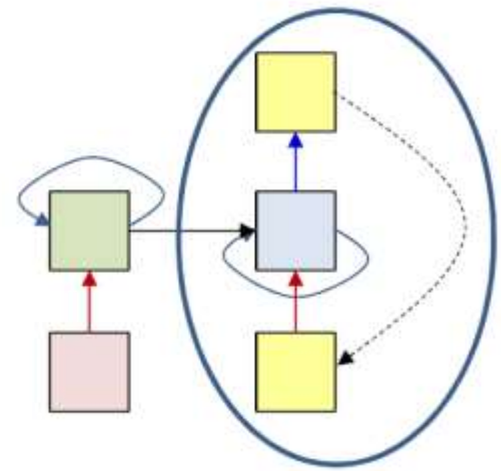




The “simple” translation model

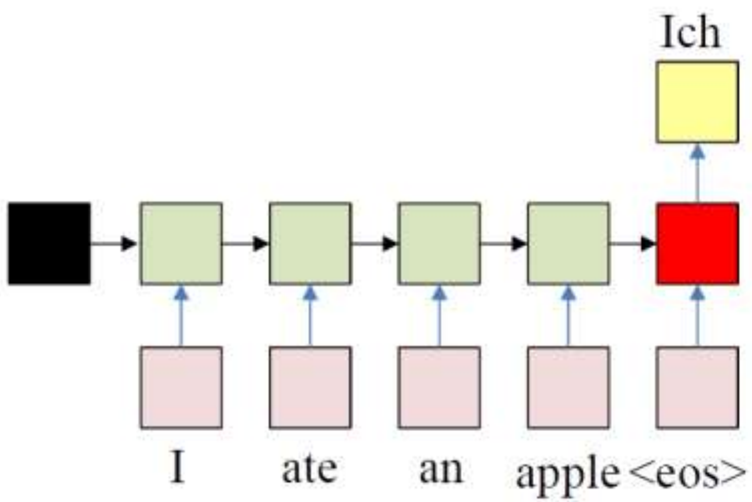
- Subsequently a second RNN uses the hidden activation as initial state to produce a sequence of outputs
  - The output at each time becomes the input at the next time
  - Output production continues until an <eos> is produced





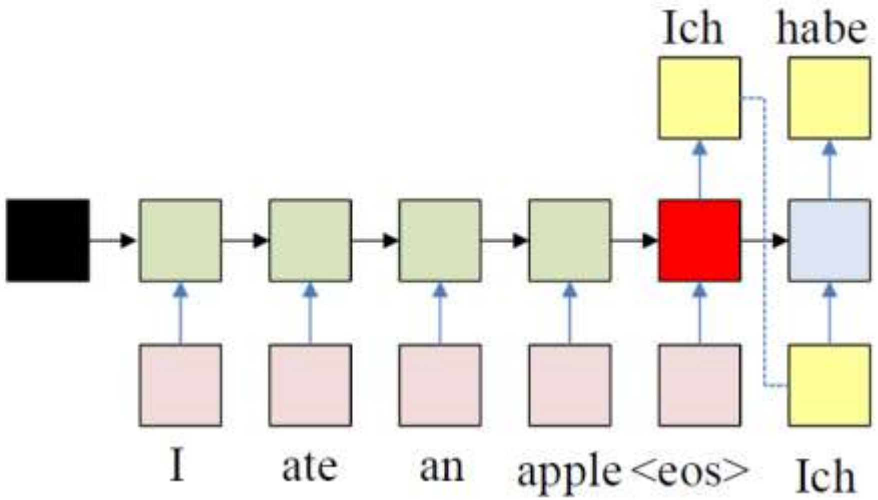
The “simple” translation model

- Subsequently a second RNN uses the hidden activation as initial state to produce a sequence of outputs
  - The output at each time becomes the input at the next time
  - Output production continues until an <eos> is produced



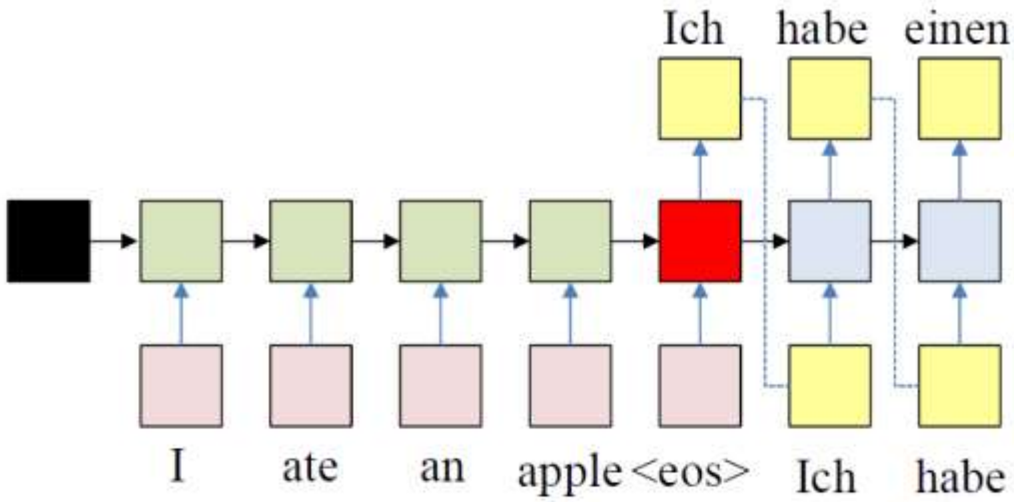
The “simple” translation model

- Subsequently a second RNN uses the hidden activation as initial state to produce a sequence of outputs
  - The output at each time becomes the input at the next time
  - Output production continues until an <eos> is produced



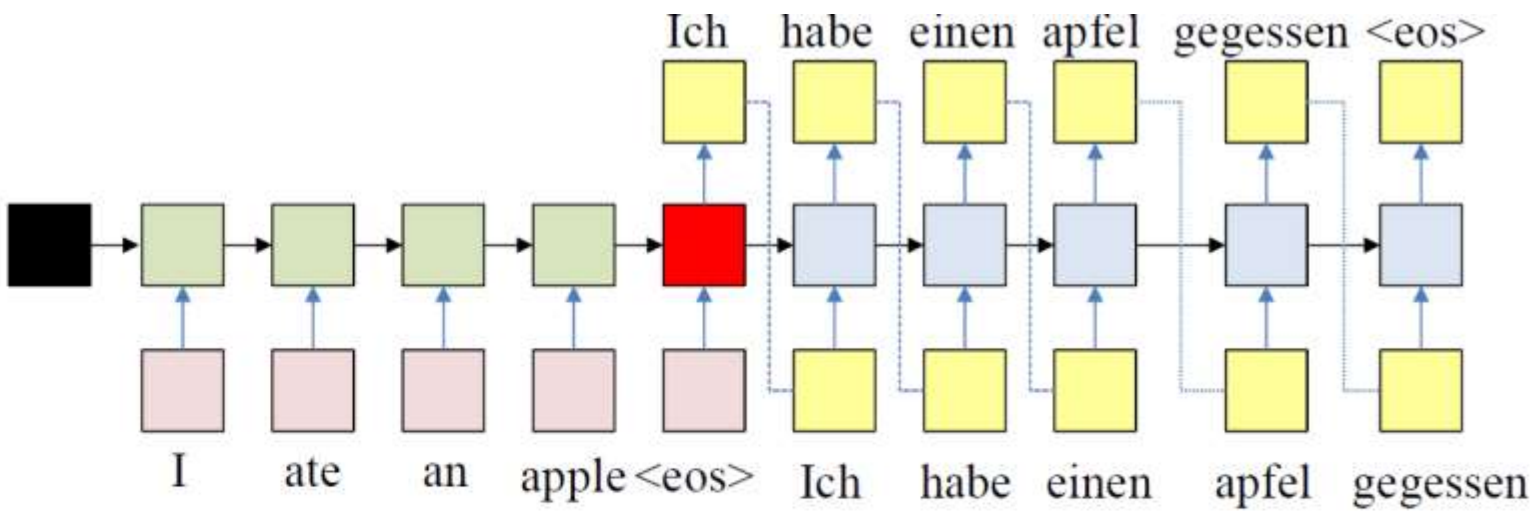
The “simple” translation model

- Subsequently a second RNN uses the hidden activation as initial state to produce a sequence of outputs
  - The output at each time becomes the input at the next time
  - Output production continues until an <eos> is produced



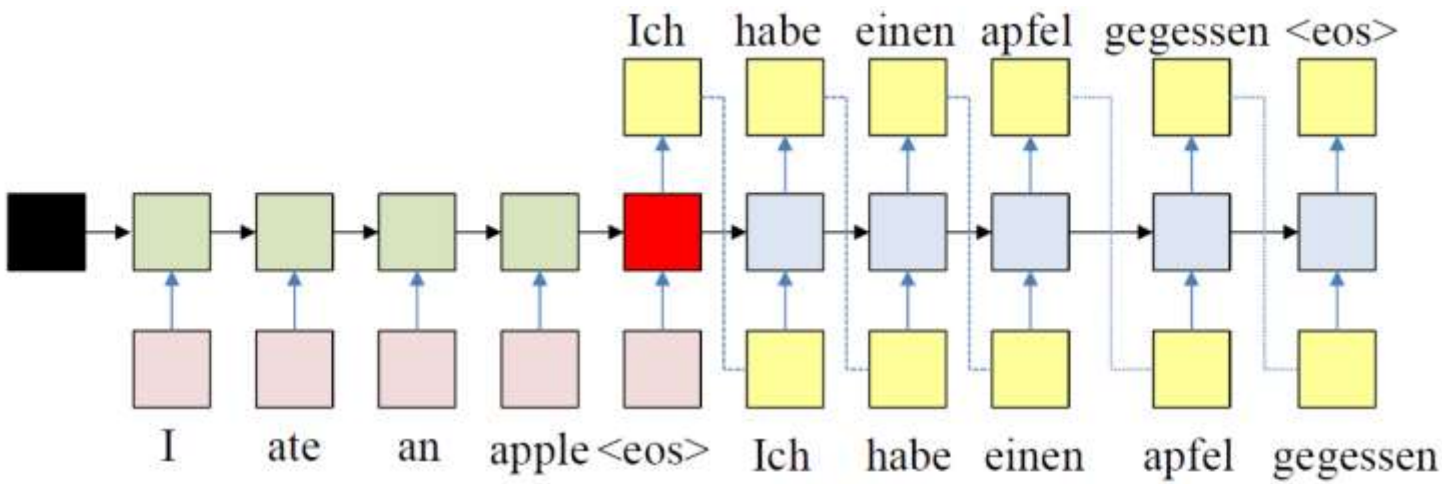
The “simple” translation model

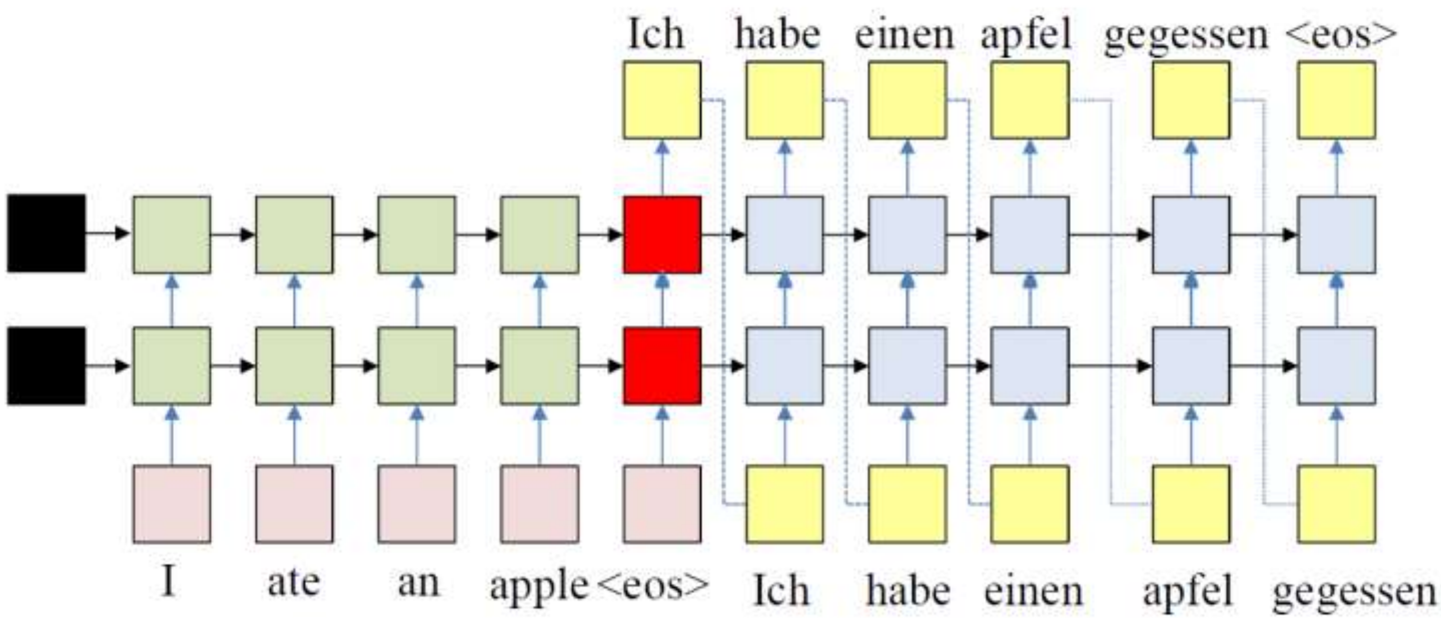
- Subsequently a second RNN uses the hidden activation as initial state to produce a sequence of outputs
  - The output at each time becomes the input at the next time
  - Output production continues until an <eos> is produced



The “simple” translation model

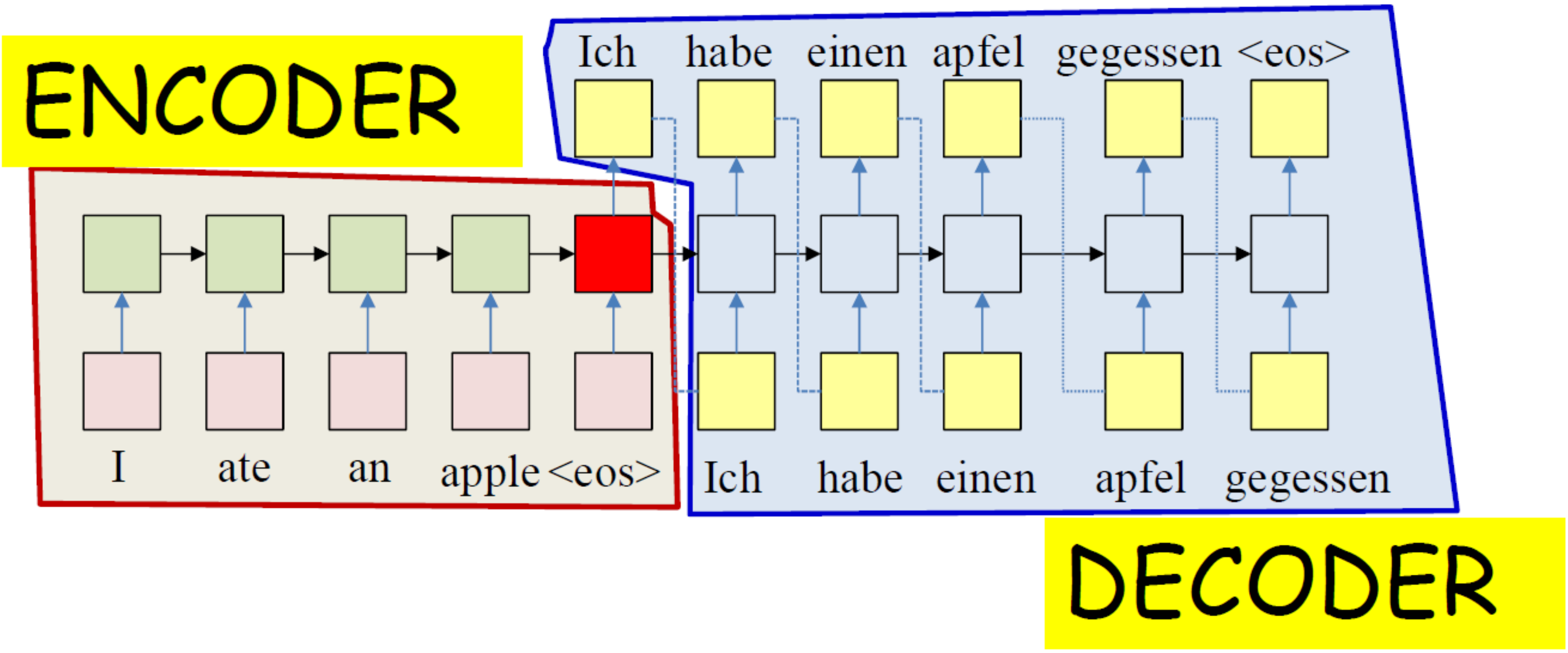
- Such an architecture can be generalized to more layers





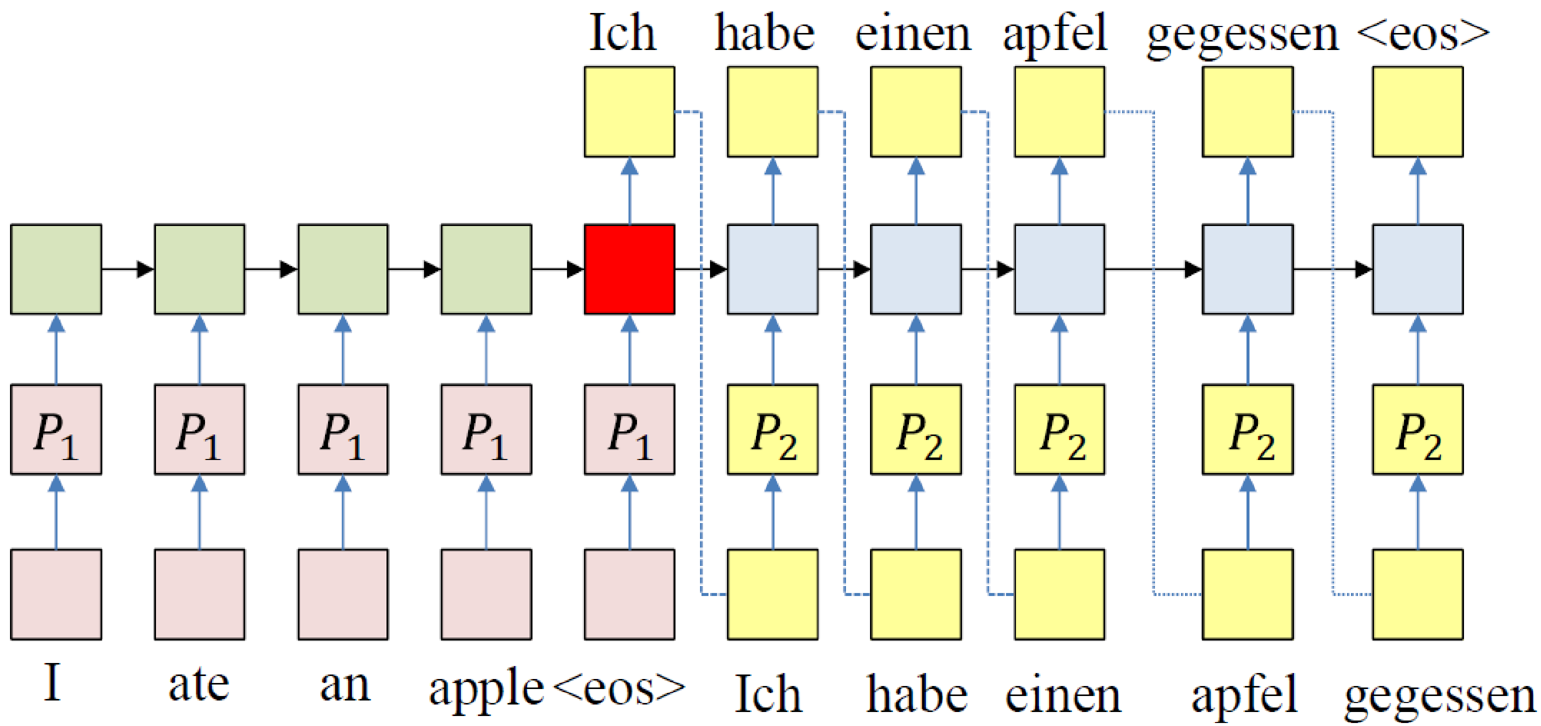
The "simple" translation model

- This is also referred to as an encoder-decoder structure



The “simple” translation model

- A more detailed look
  - Word embedding can be incorporated
  - And will be learned along with the rest of the network

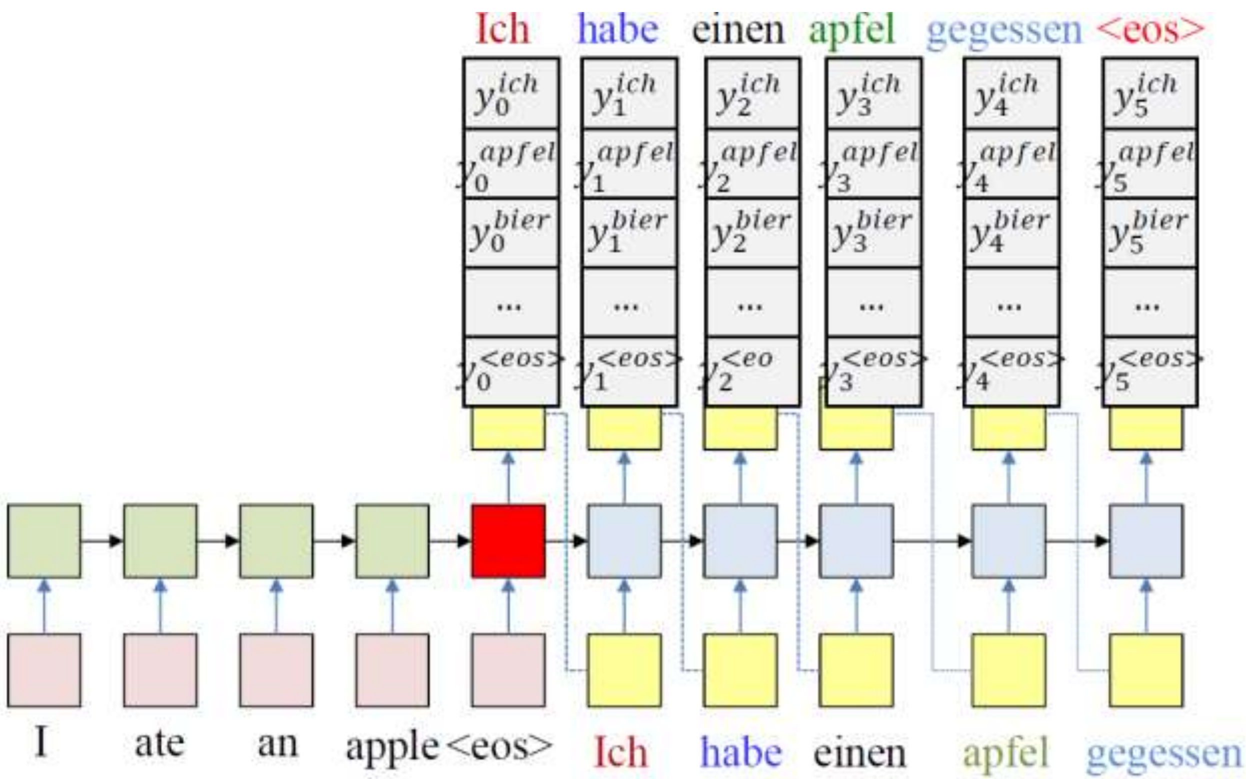


Prediction of translation model

- At each time *k*, the network produces a probability distribution over the output vocabulary

$$y_k^w = P(O_k = w | O_{k-1}, \cdots, O_1, I_1, \cdots, I_N)$$

  - At each time, a word is drawn from the output distribution
  - The drawn word is provided as input to the next time, until <eos>



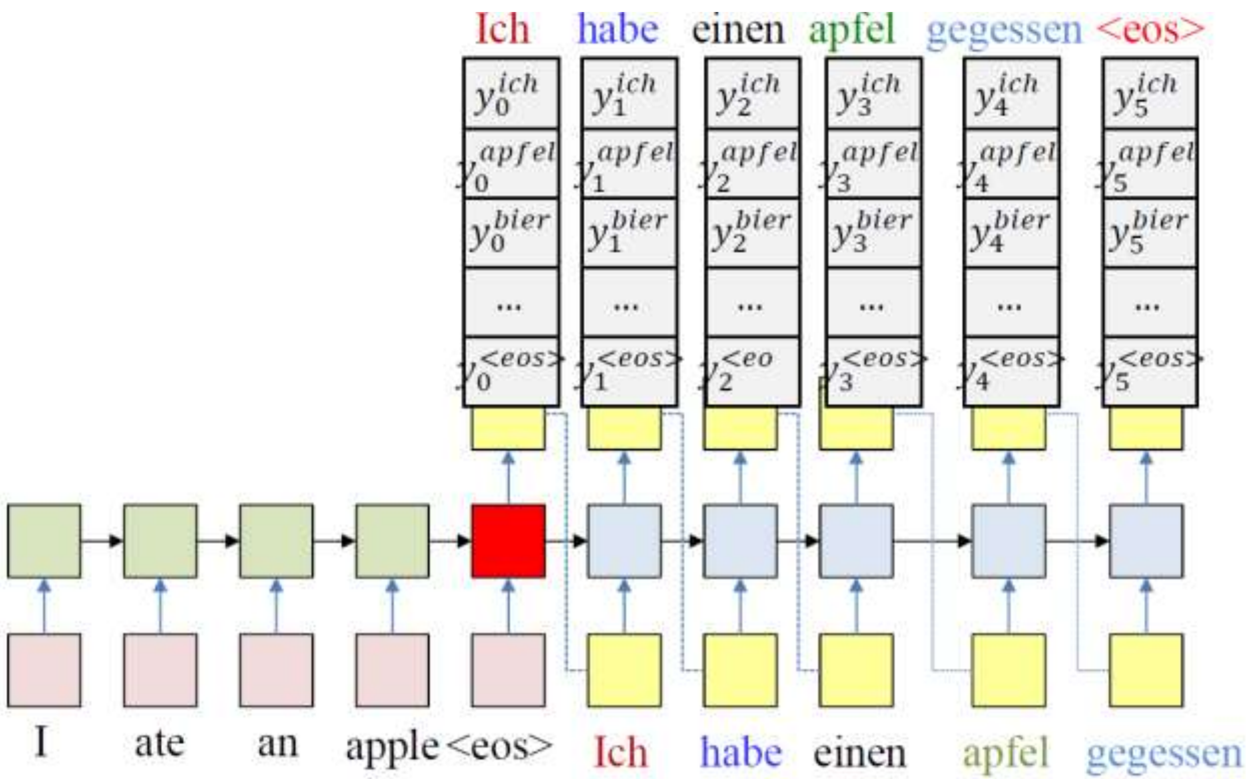
Prediction of translation model

- For a given output sequence $O_1, \cdots, O_L$, its probability is

$$P(O_1, \cdots, O_L | I_1, \cdots, I_N) = y_1^{O_1} y_2^{O_2} \cdots y_L^{O_L}$$

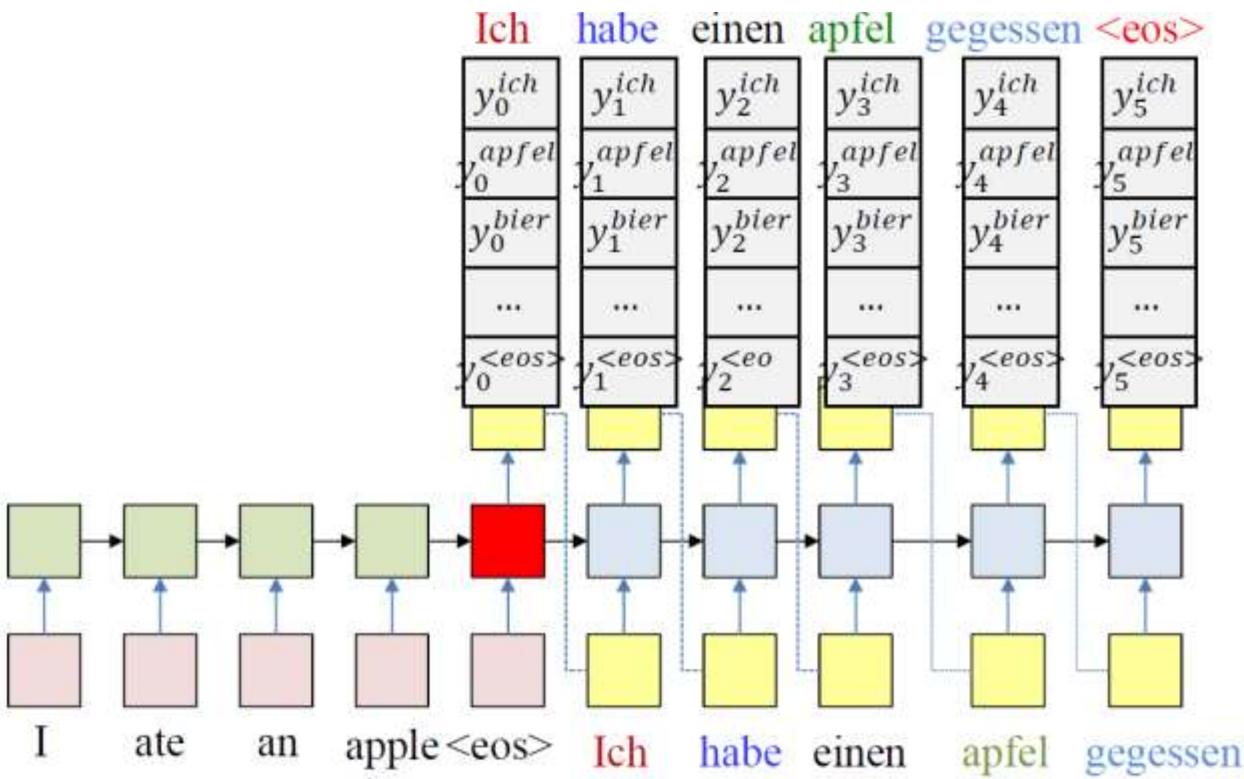
  - The objective of drawing: produce the most probable output

$$\arg\max_{O_1, \cdots, O_L} y_1^{O_1} y_2^{O_2} \cdots y_L^{O_L}$$



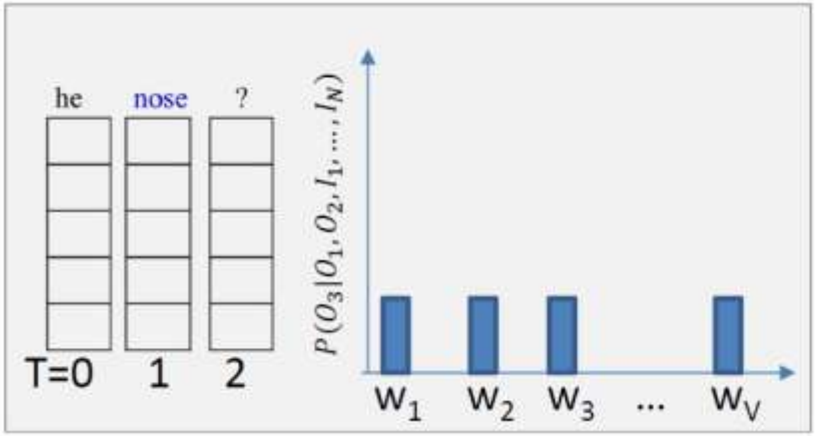
Prediction of translation model

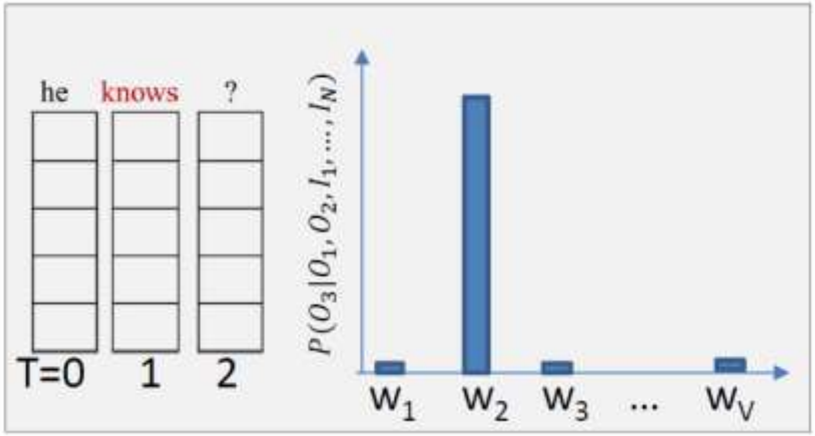
- Solving $\arg\max_{O_1,\cdots,O_L} y_1^{O_1} y_2^{O_2} \cdots y_L^{O_L}$
  - Cannot pick the most likely symbol at each time
  - Reason?
    - Choosing a different, less likely word could cause the probability at the next time to be larger, resulting in a more likely output overall



Why greedy is not good

- Example from speech recognition





"Nose" has highest probability at t=2 and is selected

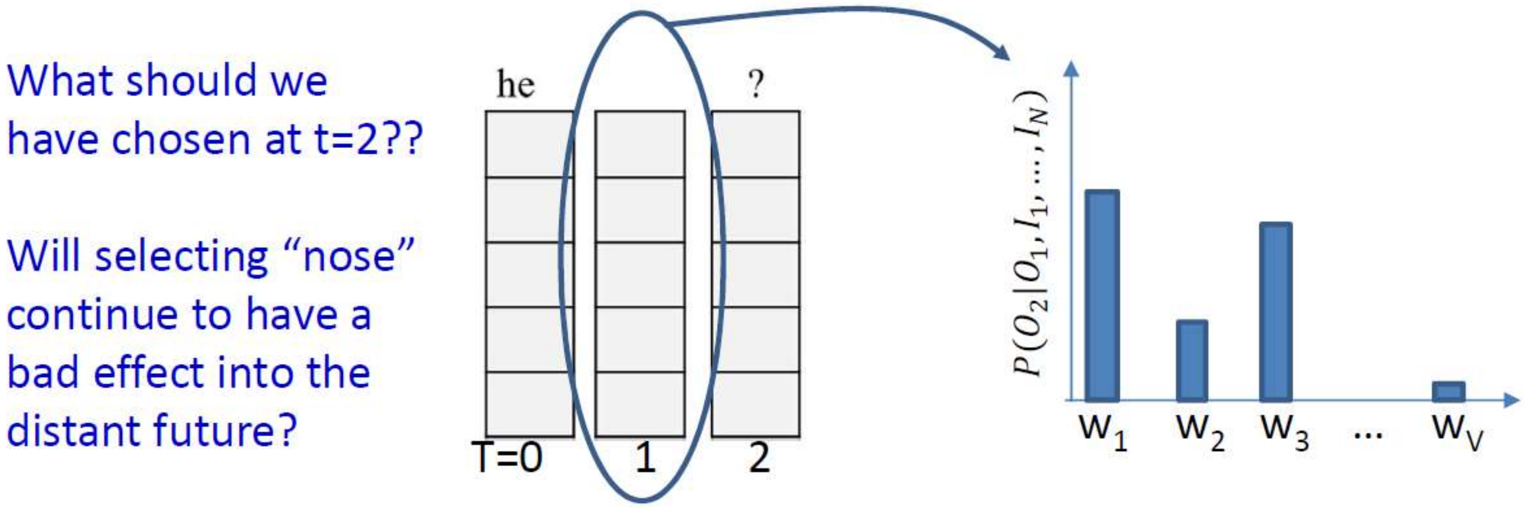
- The model is very confused at t=3 and assigns low probabilities to many words at the next time
- Selecting any of these will result in low probability for the entire 3-word sequence

"Knows" has slightly lower probability than "nose", but is still high and is selected

- "he knows" is a reasonable beginning and the model assigns high probabilities to words such as "something"
- Selecting one of these results in higher overall probability for the 3-word sequence

Why greedy is not good

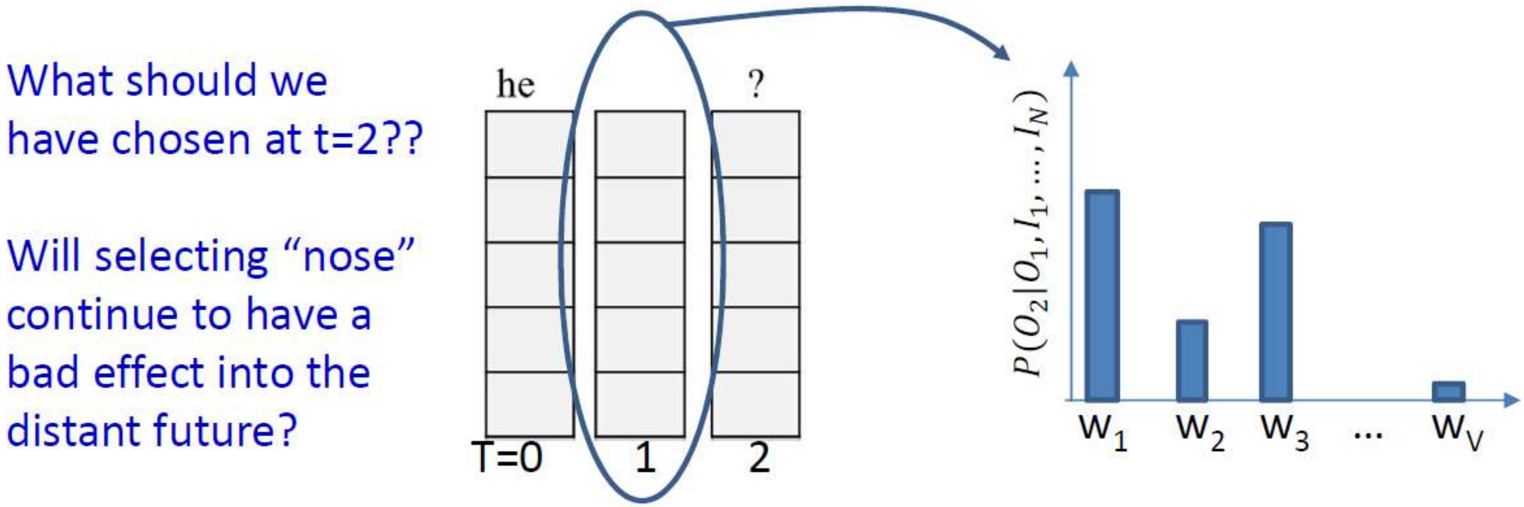
- Example from speech recognition



- Problem: impossible to know a priori which word leads to the more promising future
  - Effect may not be obvious until several words down the line
  - Or the choice of the wrong word early may cumulatively lead to a poorer overall score over time

Why greedy is not good

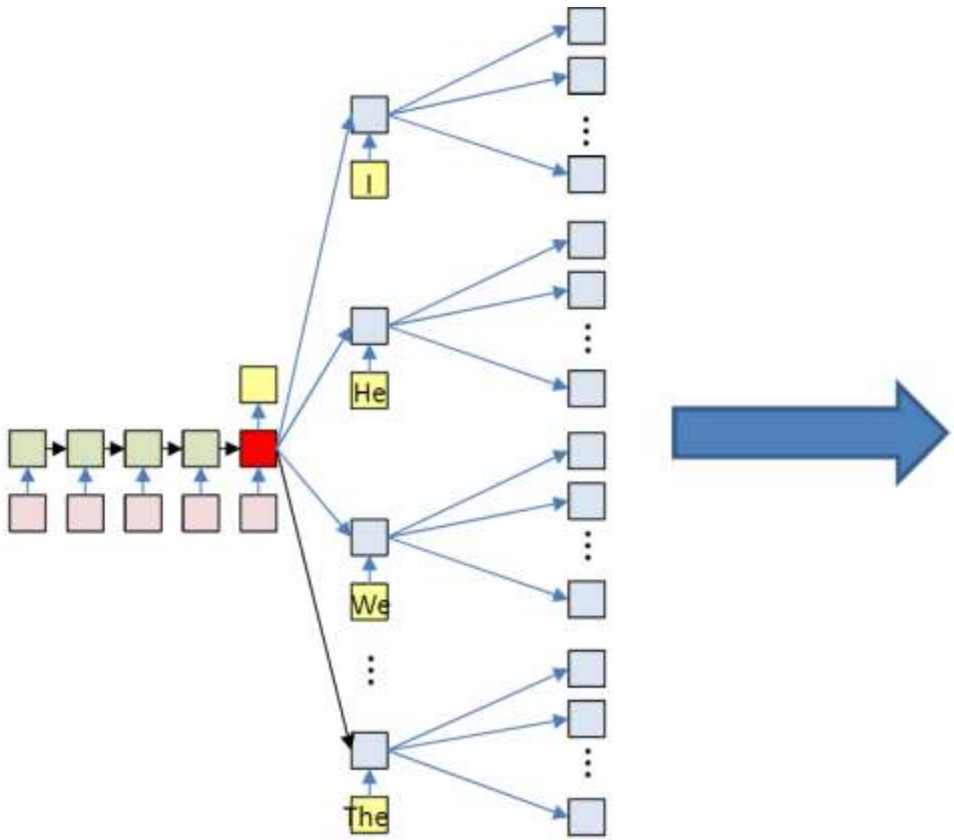
- Example from speech recognition



- Problem: impossible to know a priori which word leads to the more promising future
  - In general, making a poor choice at any time commits us to a poor future
  - Solution: Don't choose

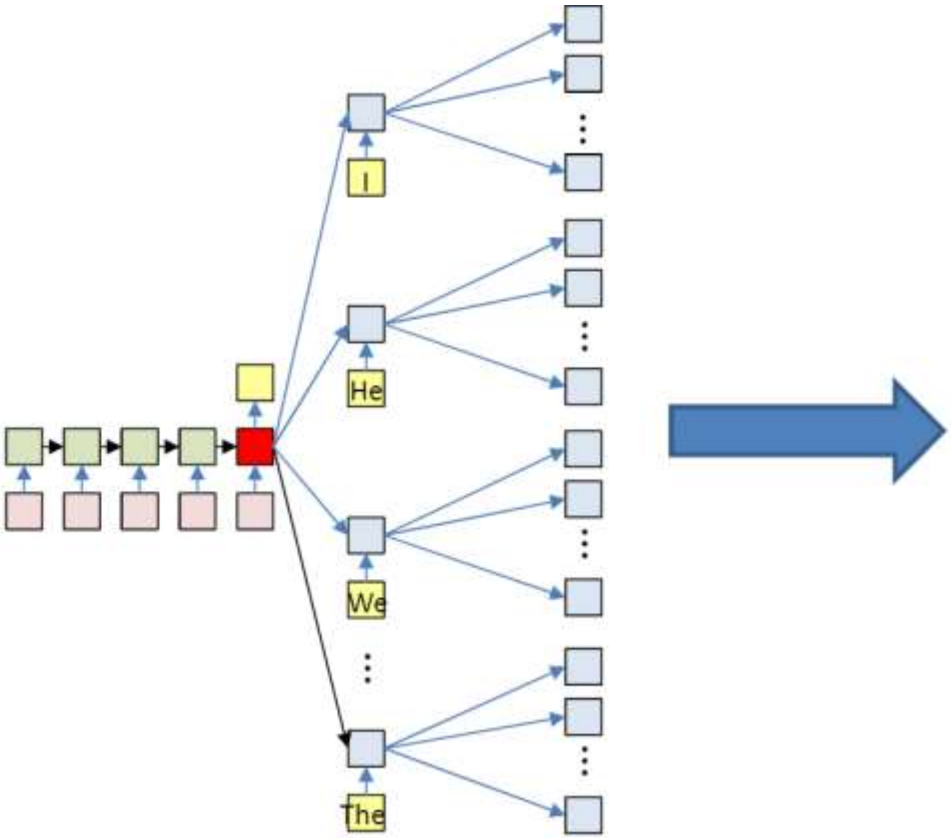
Solution: generate and prune

- Retain all choices and fork the state network
  - With every possible word as input



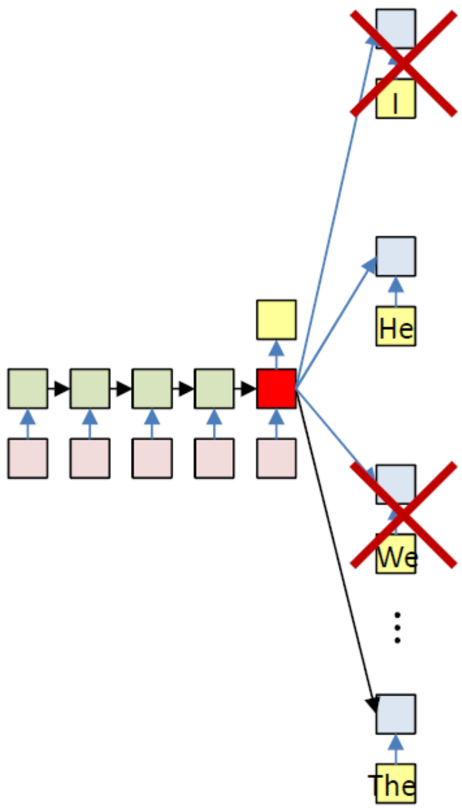
Solution: generate and prune

- Retain all choices and fork the state network
  - However, this will blow up very quickly (exponential growth).



Solution: generate and prune

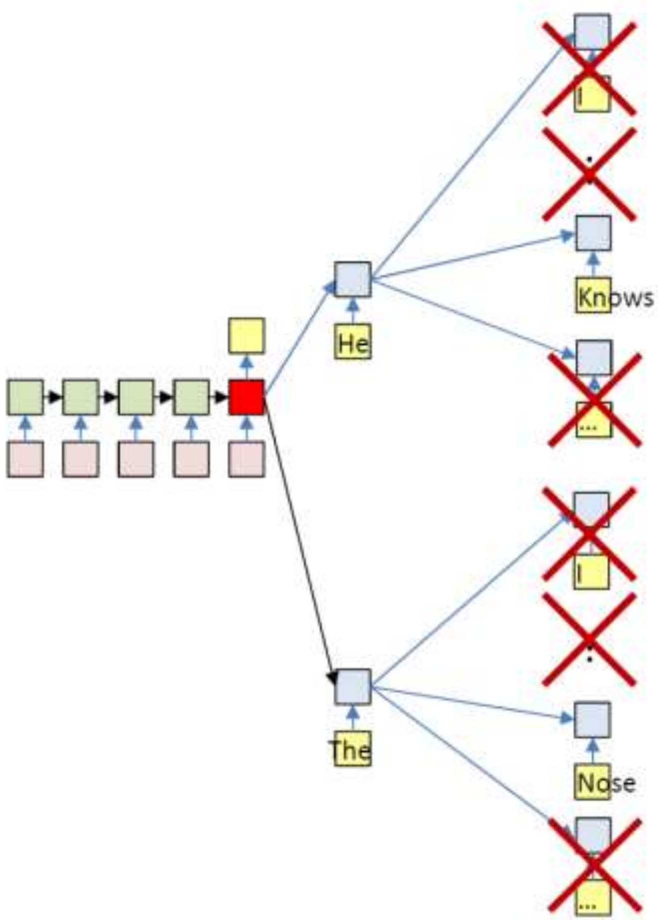
- Prune the state space
  - At each time, retain only the top *K* scoring forks



$$Top_K\, P(O_1|I_1, \ldots, I_N)$$

Solution: generate and prune

- Prune the state space
  - At each time, retain only the top *K* scoring forks



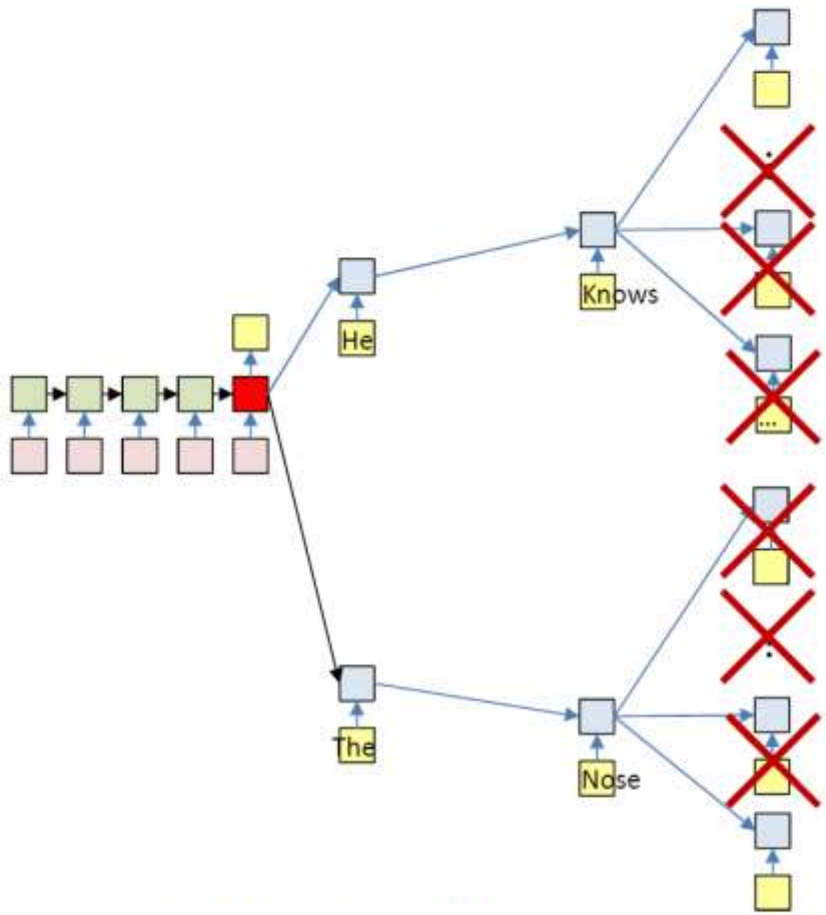
Note: based on product

$$Top_K\, P(O_2 O_1 | I_1, \ldots, I_N)$$

$$= Top_K\, P(O_2 | O_1, I_1, \ldots, I_N) P(O_1 | I_1, \ldots, I_N)$$

Solution: generate and prune

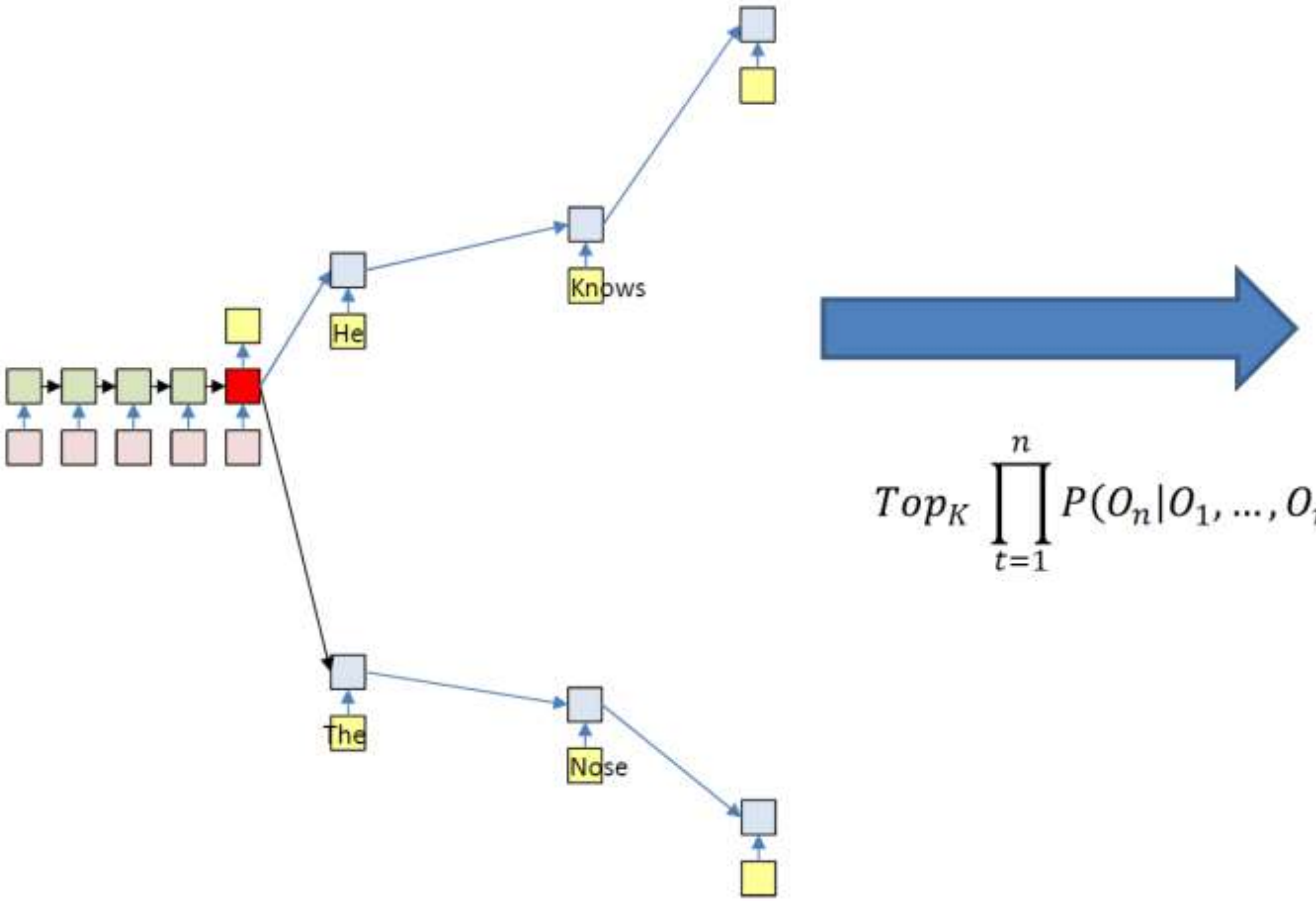
- Prune the state space
  - At each time, retain only the top K scoring forks



$$= Top_K P(O_2|O_1, I_1, \dots, I_N) \times P(O_2|O_1, I_1, \dots, I_N) \times P(O_1|I_1, \dots, I_N)$$

Solution: generate and prune

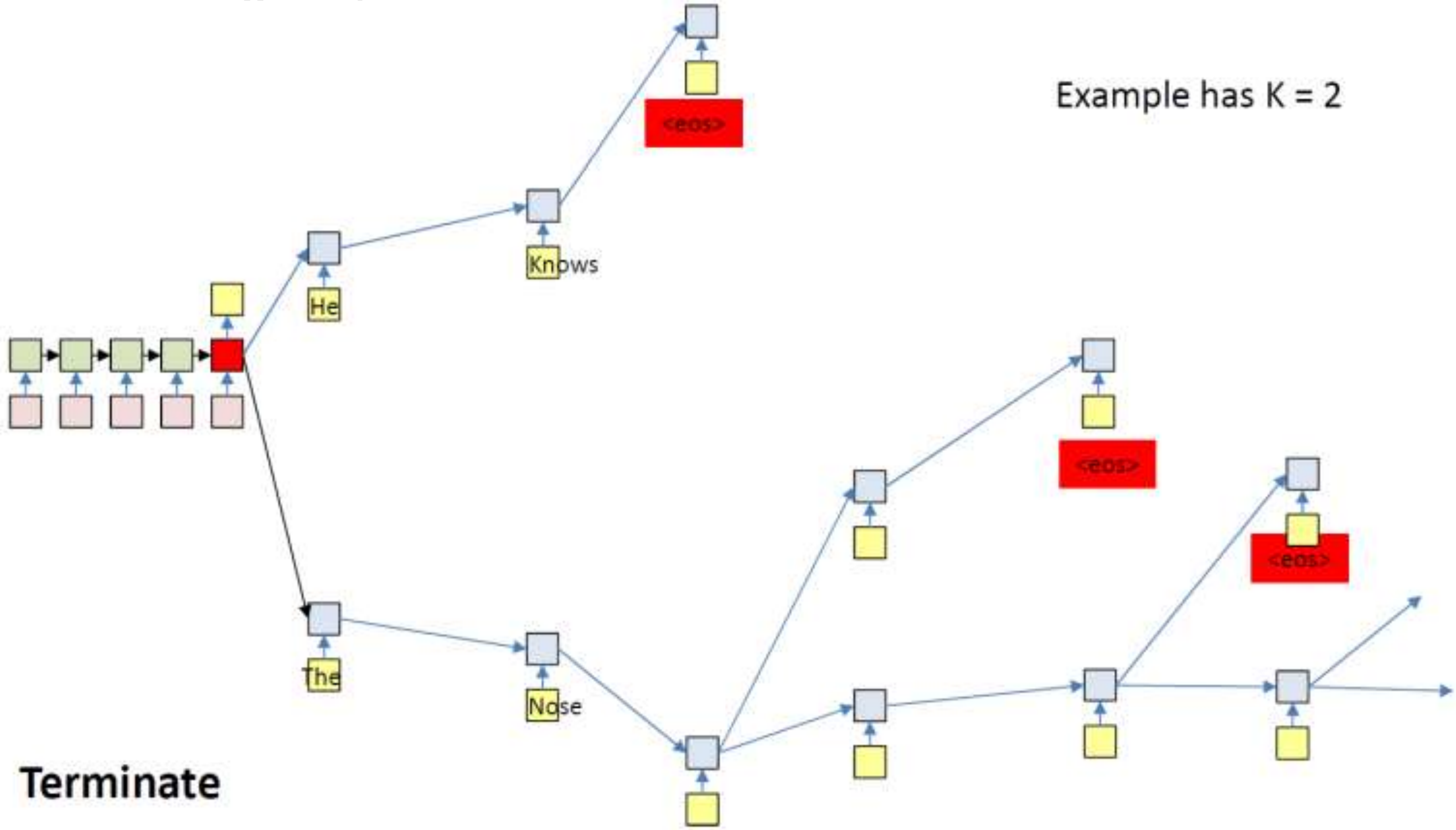
- Prune the state space
  - At each time, retain only the top *K* scoring forks



$$Top_K \prod_{t=1}^{n} P(O_n|O_1, \ldots, O_{n-1}, I_1, \ldots, I_N)$$

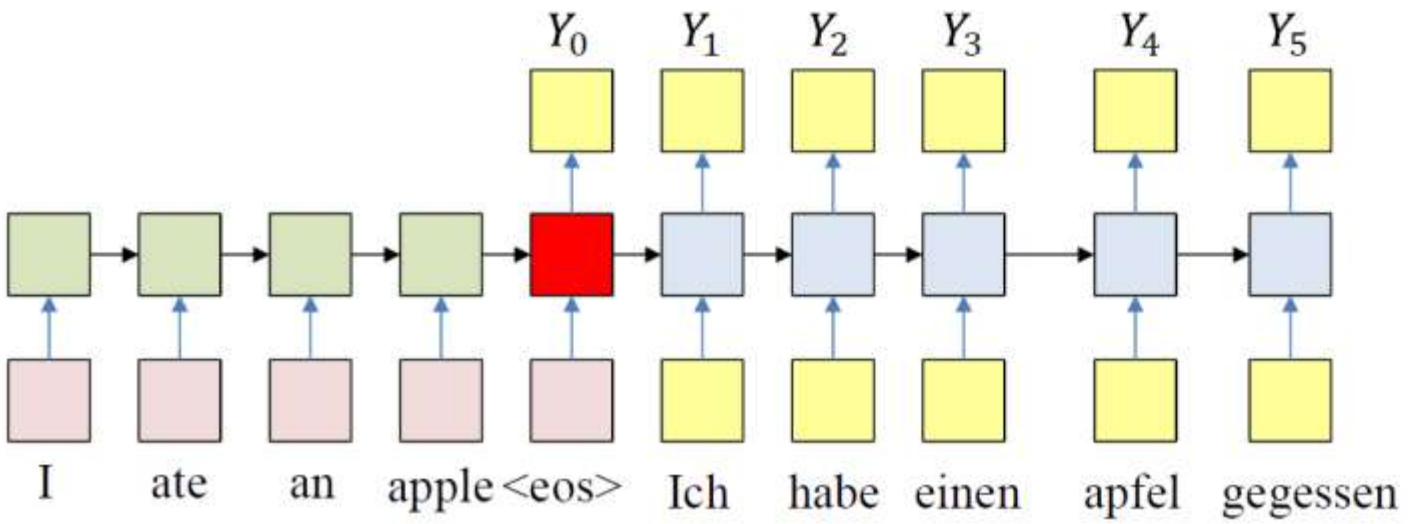
Solution: generate and prune

- Termination
  - Paths cannot continue once the output an <eos>
  - Select the most likely sequence ending in <eos> across all terminating sequences



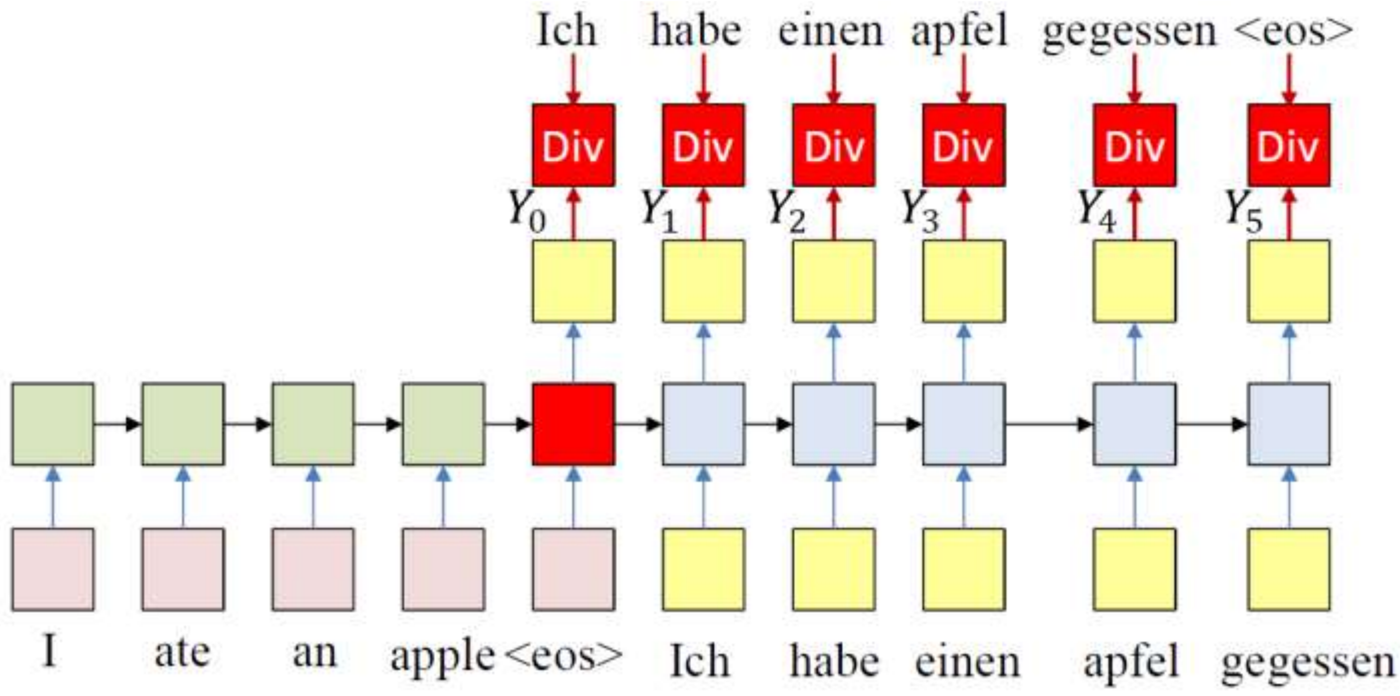
Training the seq-to-seq model

- Forward pass
  - Input the source and target sequences sequentially
  - Output will be a probability distribution over target symbol set



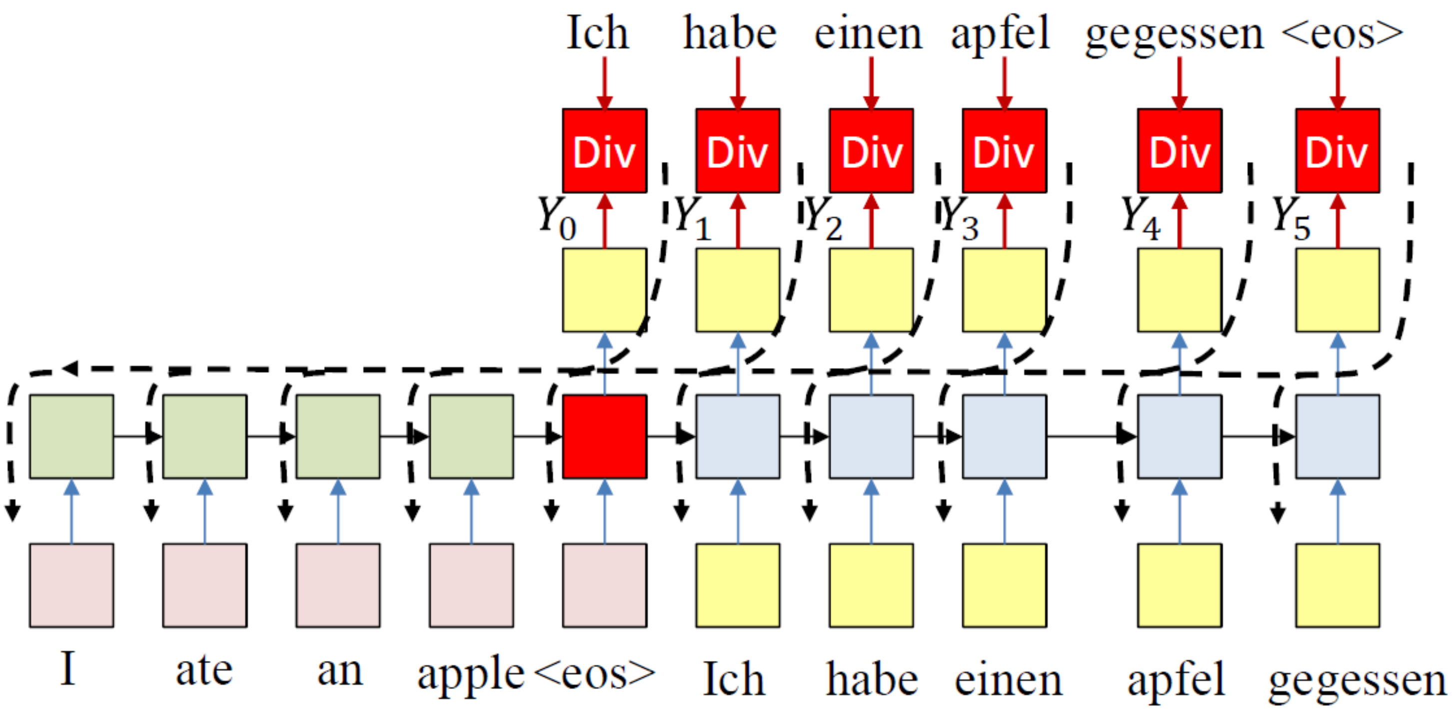
Training the seq-to-seq model

- Backward pass
  - Compute the loss/divergence between the output distribution and target word sequence



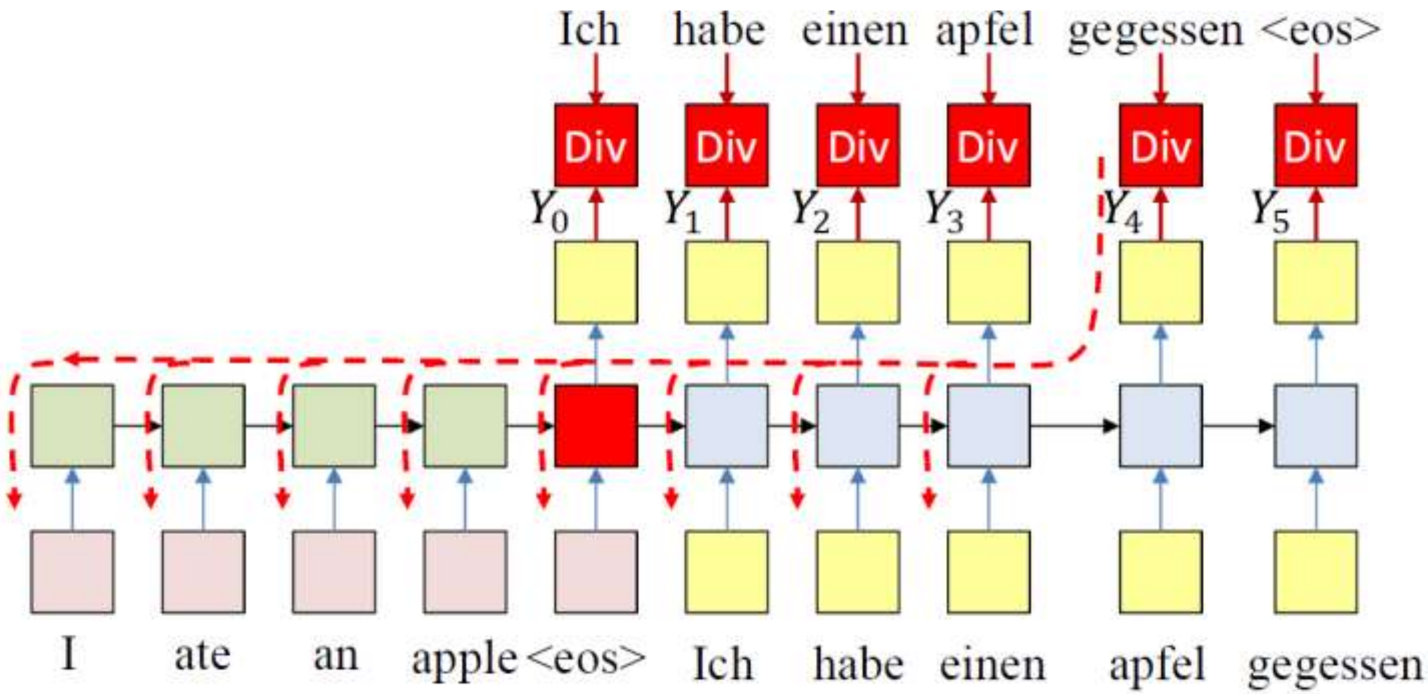
Training the seq-to-seq model

- Backward pass
  - Compute the loss/divergence between the output distribution and target word sequence
  - Backpropagate the derivatives of the divergence through the net



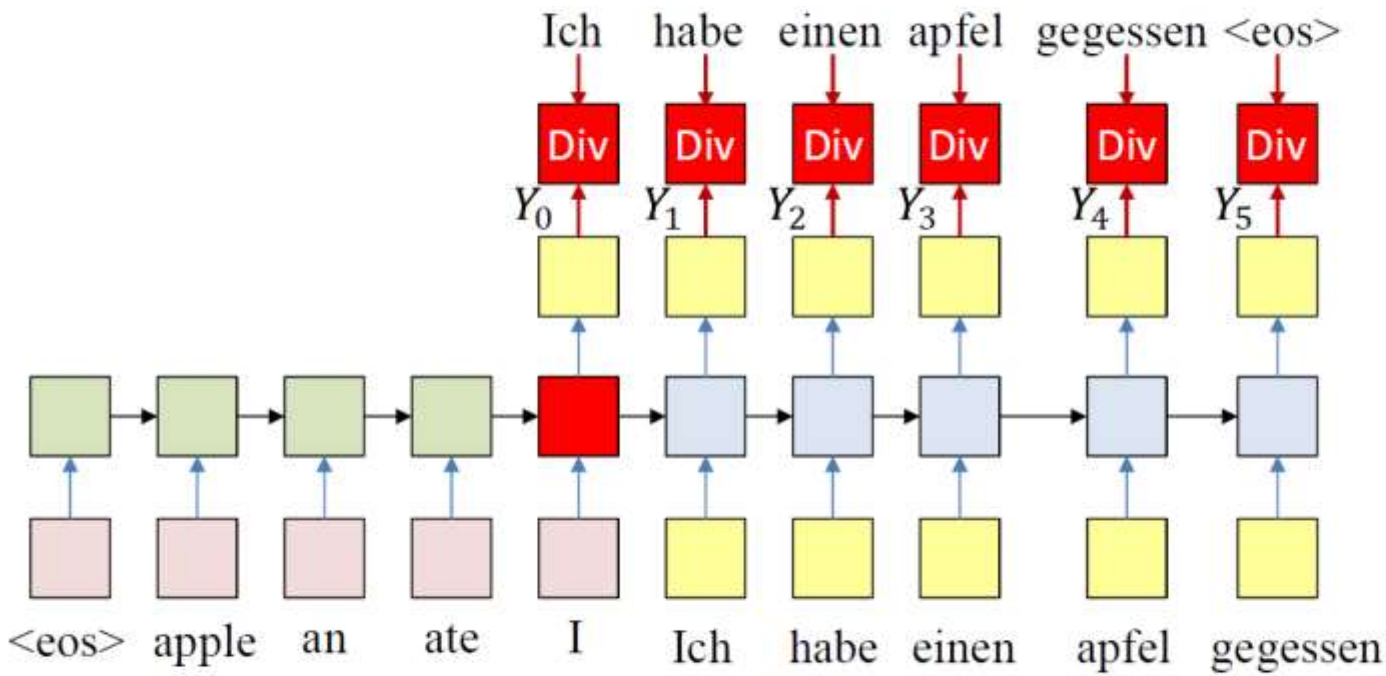
Training the seq-to-seq model

- Backward pass
  - In practice, we may randomly sample words from the output to actually use for the backprop update



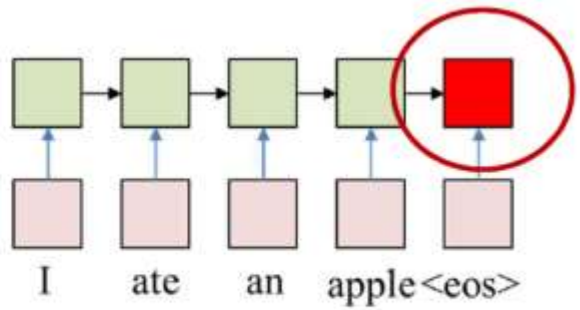
Training the seq-to-seq model

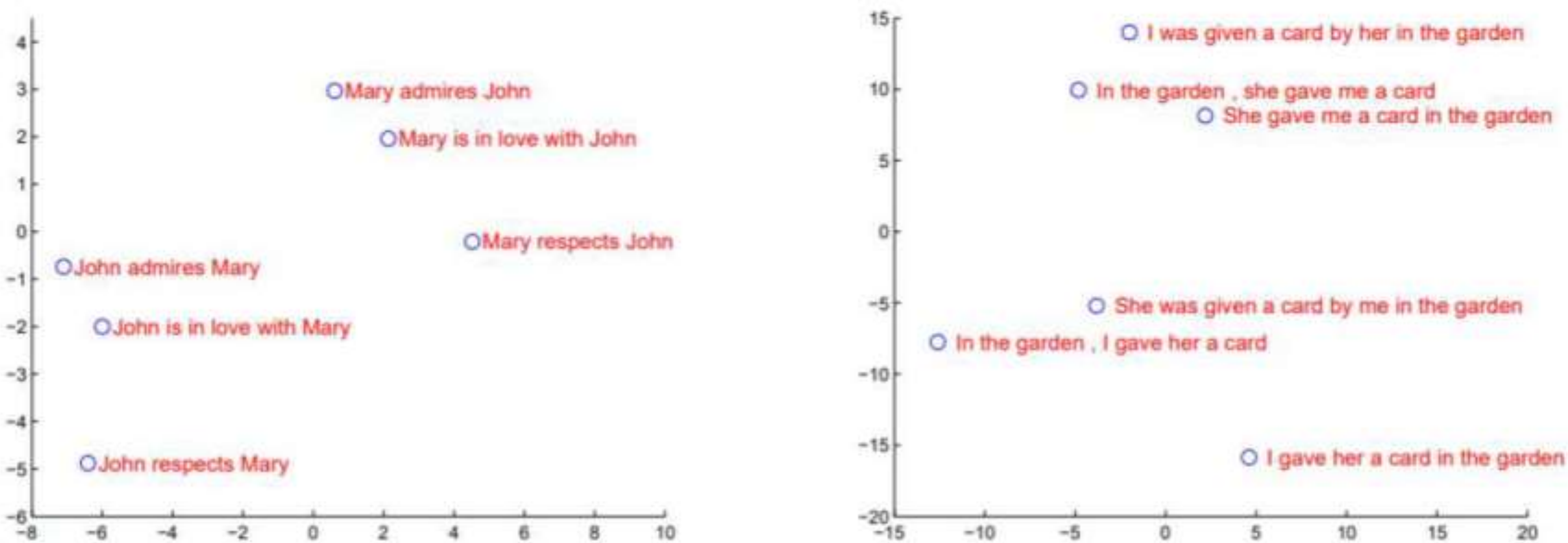
- Trick of the trade
  - The input sequence is fed in reverse order
  - For both training and during actual decode



Machine translation example

- Hidden representation encodes semantics
  - Clustered by semantic meaning





"Sequence-to-sequence learning with neural networks",
Sutskever, Vinyals and Le

Human machine conversation example

- Task: Human text in, machine response out
- Trained on human-human conversations

Machine: *what is the error that you are running please*
Human: *i am seeing an error related to vpn*
Machine: *what is the error message that you are getting when connecting to vpn using network connect ?*
Human: *connection refused or something like that*
Machine: *may i know the version of network connect you connect ?*
Human: *i am not sure i know that*
Machine: *is the network connect prompting for any user certificate*

"A neural conversational model", Orin Vinyals and Quoc Le

Image-to-sequence problem

- Generating image captions

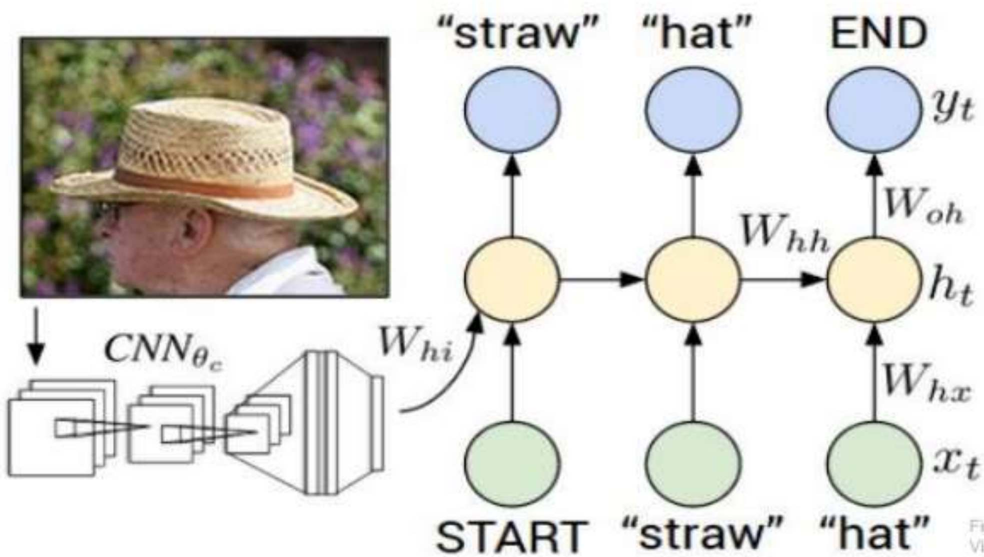
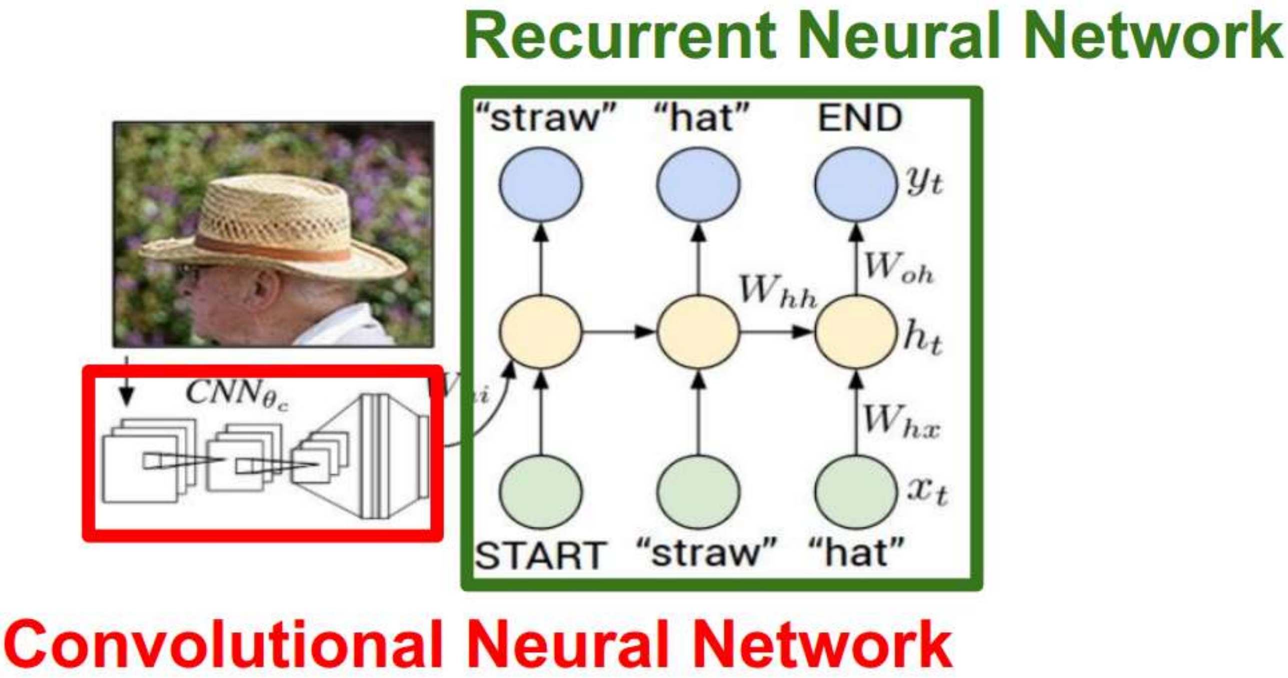


Figure from Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015. Reproduced for educational purposes.

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Image-to-sequence problem

- Initial state is produced by a state-of-the-art CNN-based image classification system
- Subsequent model is the decoder end of a seq-to-seq model

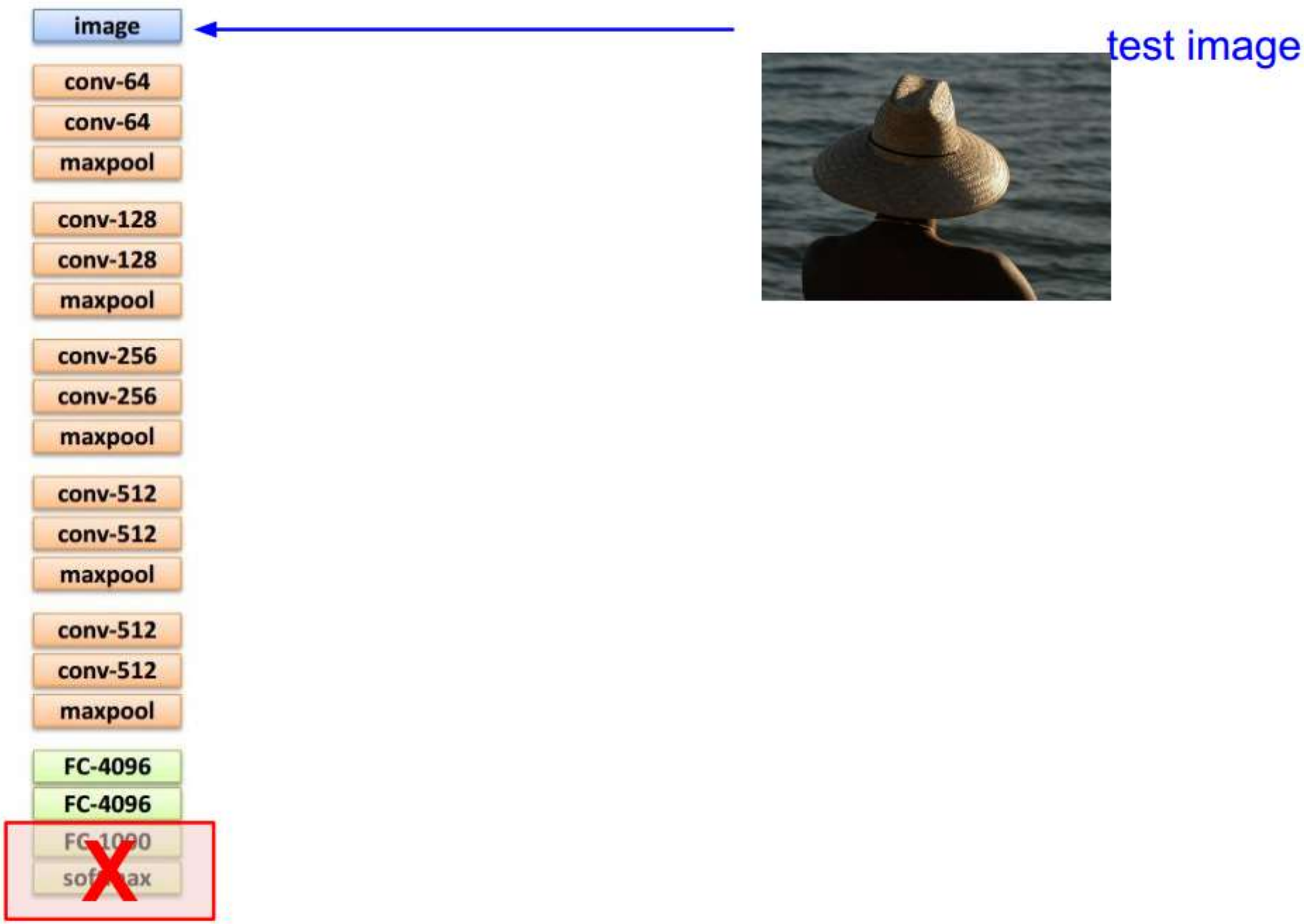


Generating image captions

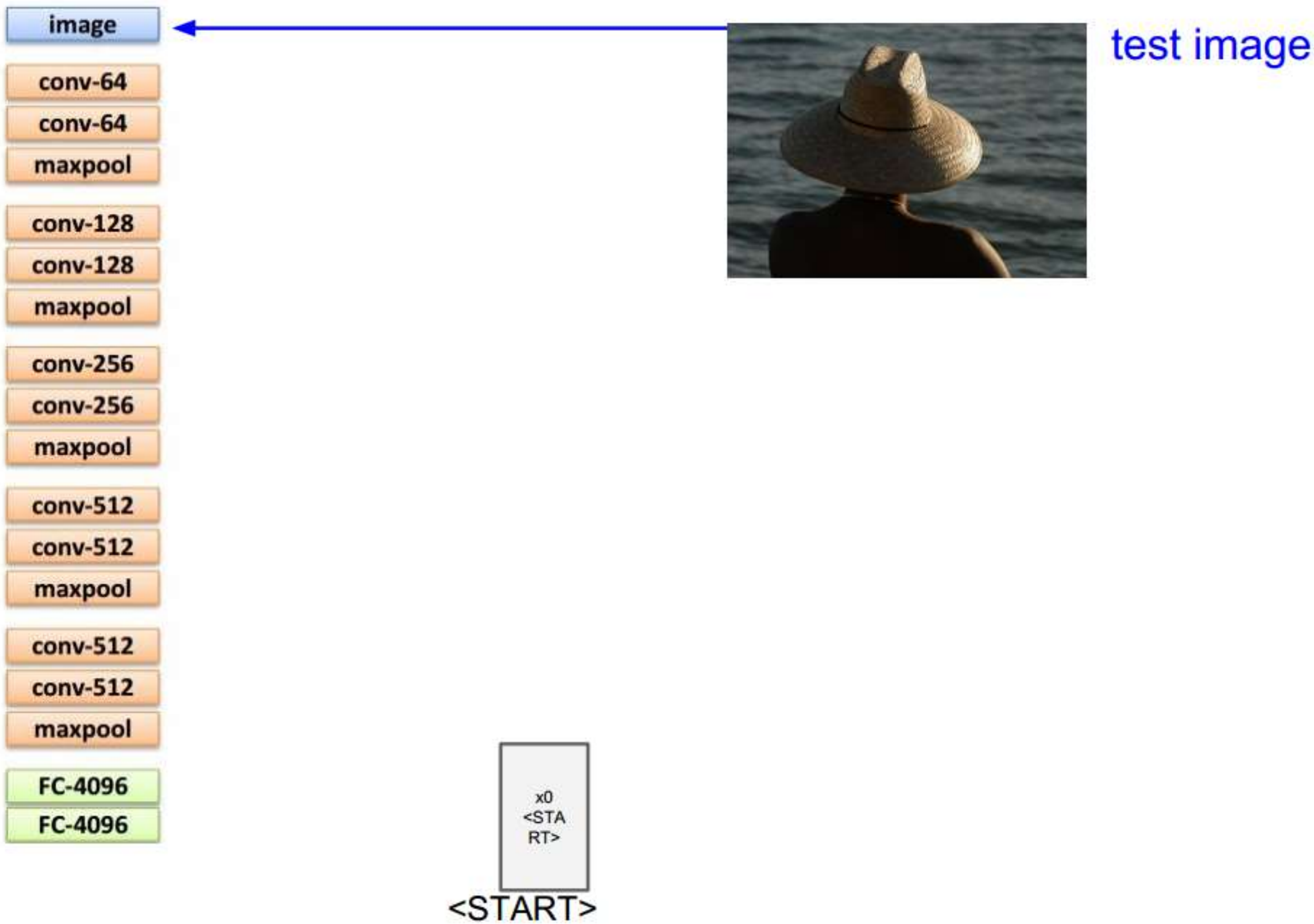
This image is CC0 public domain

test image

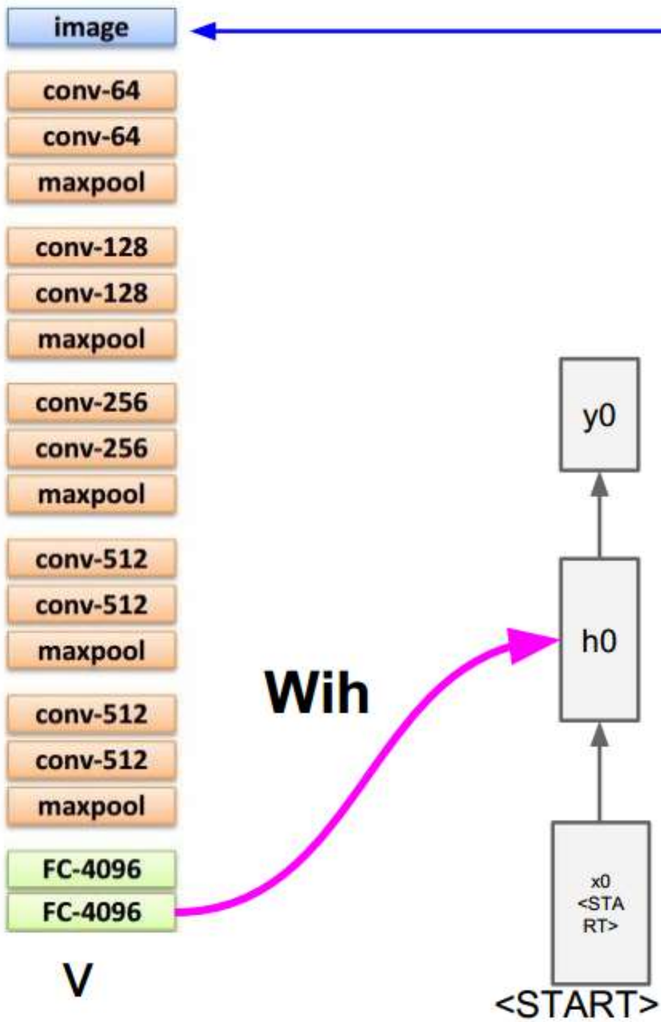
Generating image captions



Generating image captions



Generating image captions



test image

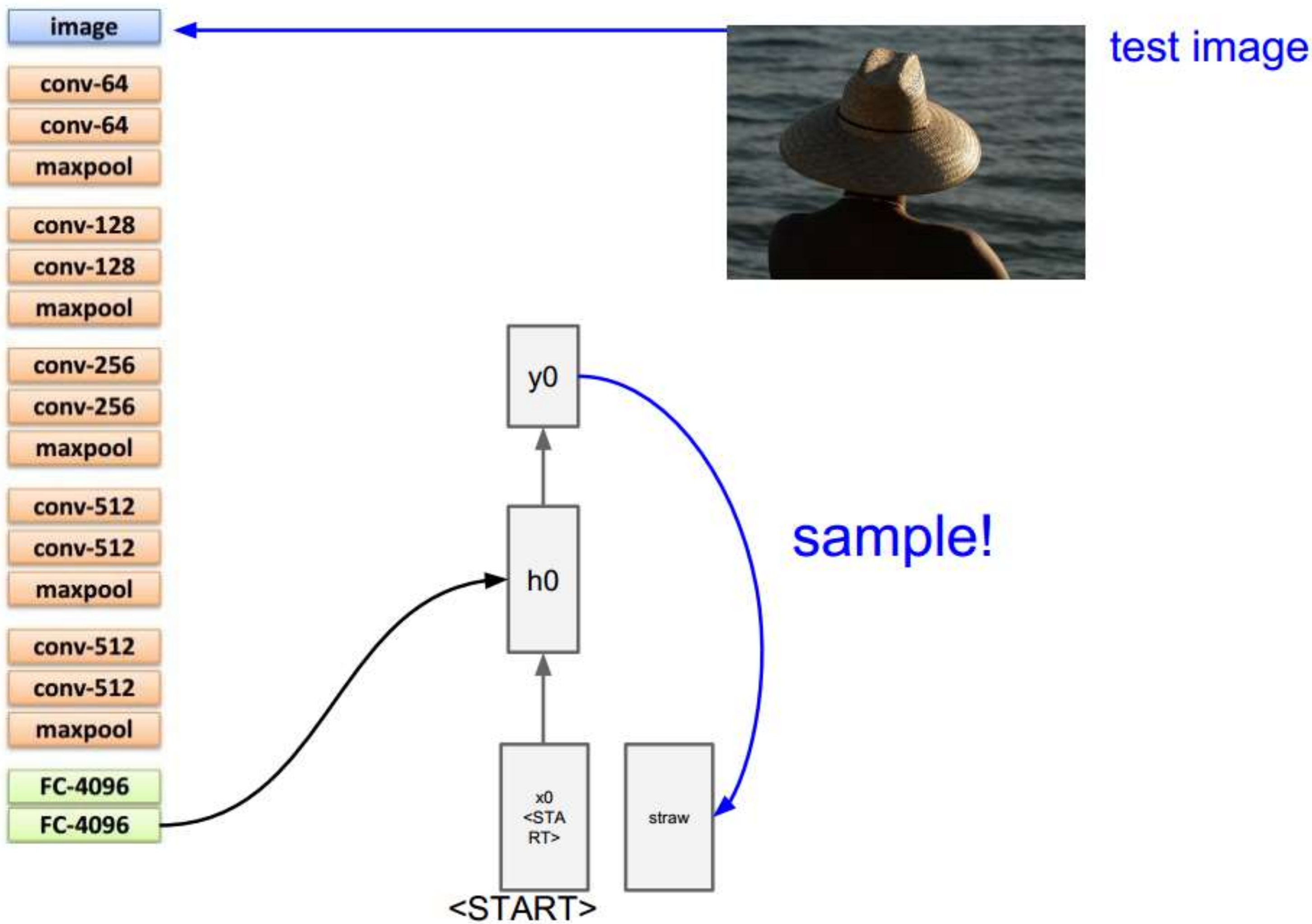
before:

$$h = \tanh(Wxh * x + Whh * h)$$

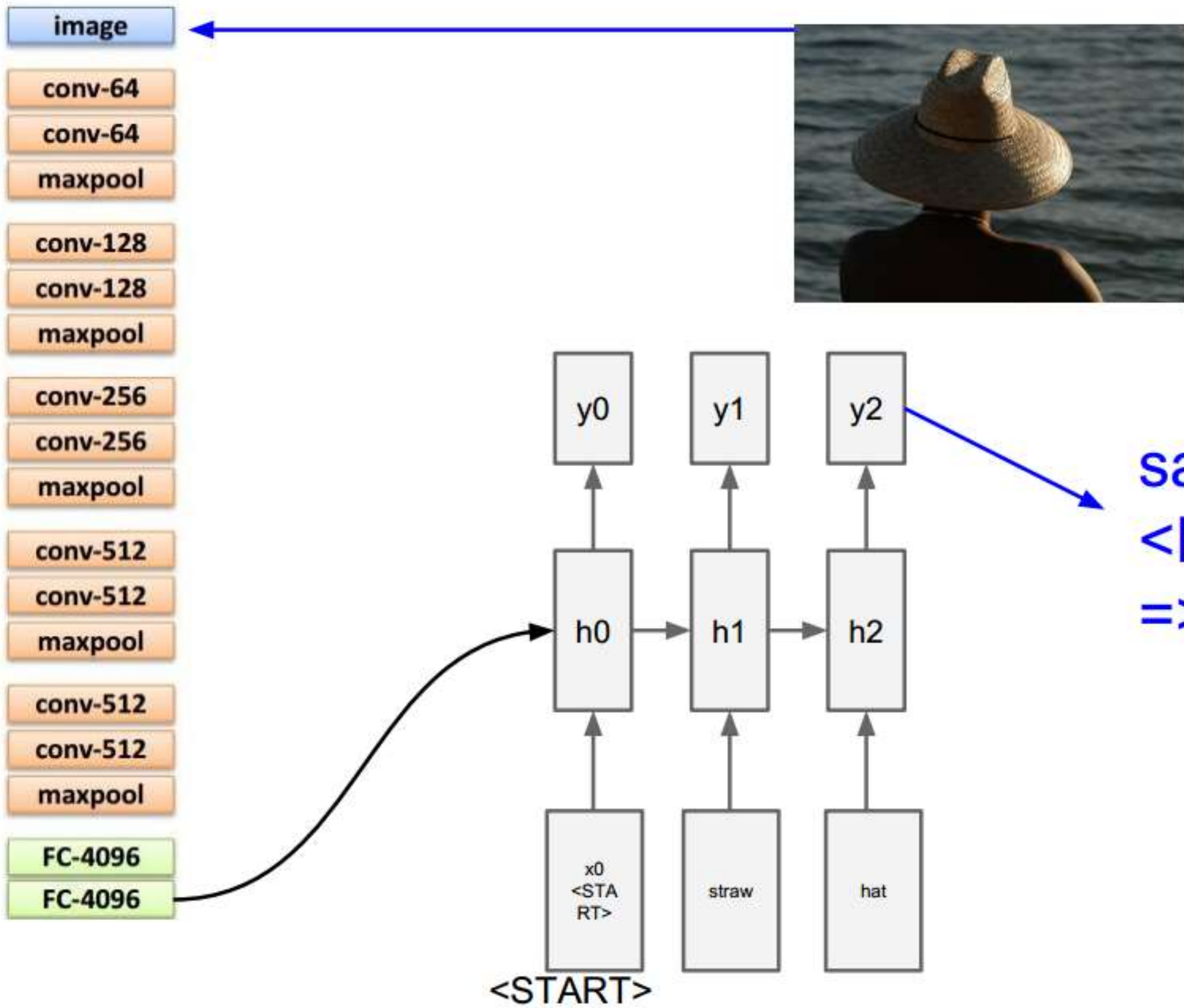
now:

$$h = \tanh(Wxh * x + Whh * h + \mathbf{Wih} * \mathbf{v})$$

Generating image captions



Generating image captions

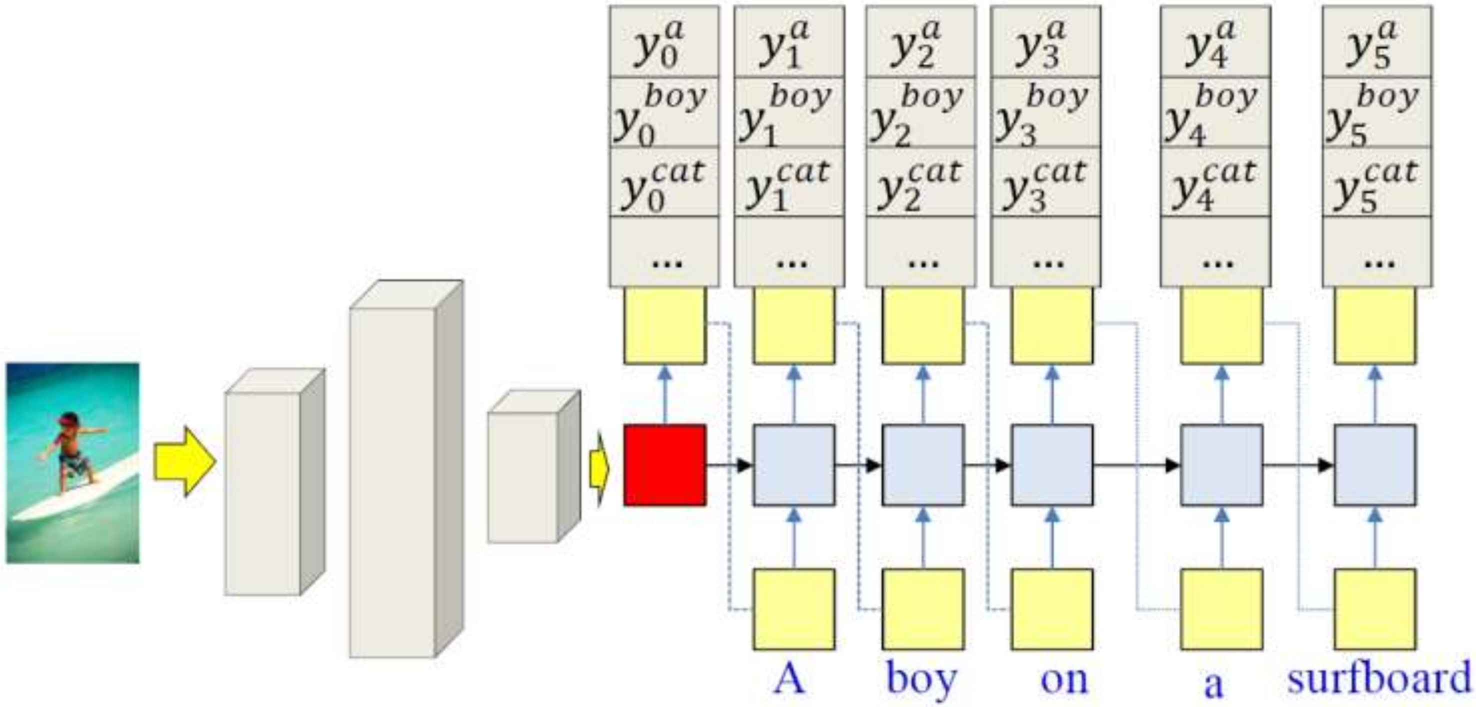


test image

sample
<END> token
=> finish.

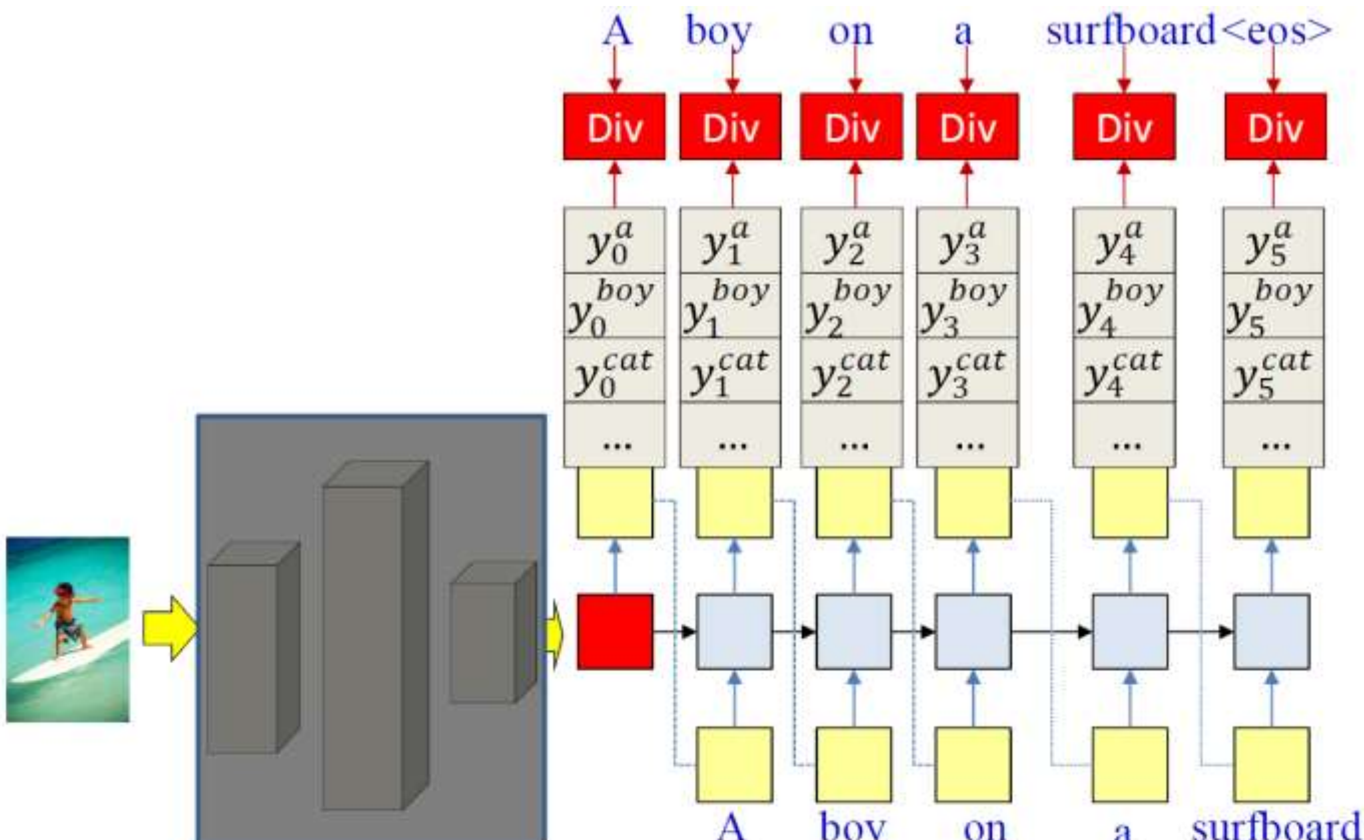
Training the image-to-seq model

- Given a set of (image, caption) pairs
  - The image network is pretrained on a large corpus, e.g., ImageNet
  - Forward pass: produce output distribution given the image and caption



Training the image-to-seq model

- Given a set of (image, caption) pairs
  - Backward pass: compute the loss w.r.t training caption, and backprop derivatives
    - All components, including final layer of the ConvNet, are updated
    - The CNN portions are not modified (transfer learning)



Application: Image Captioning

- Example Results

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

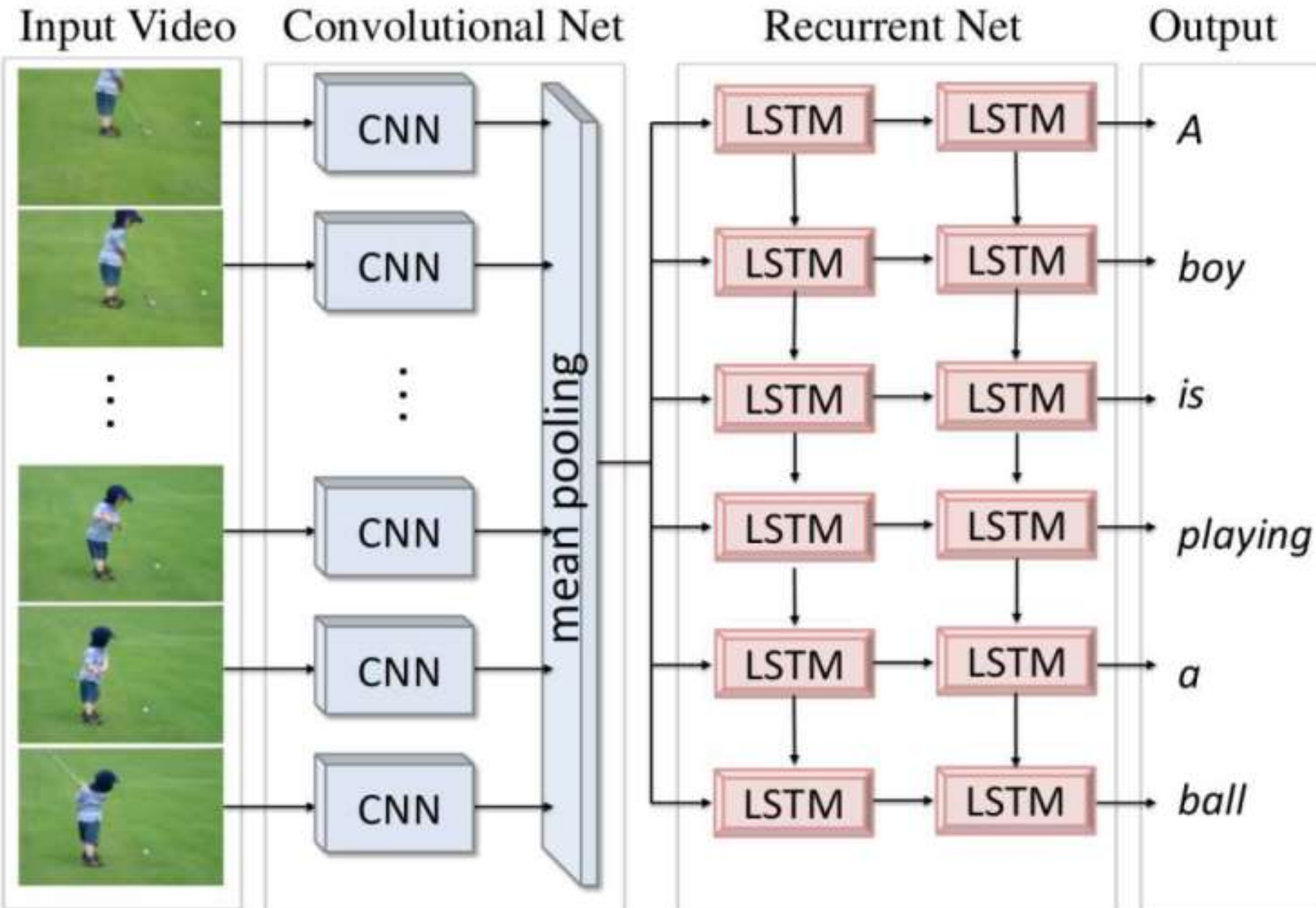
"a young boy is holding a baseball bat."

"a cat is sitting on a couch with a remote control."

"a woman holding a teddy bear in front of a mirror."

"a horse is standing in the middle of a road."

Application: Video Captioning



Translating Videos to Natural Language Using Deep Recurrent Neural Networks
Subhashini Venugopalan, Huijun Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, Kate Saenko

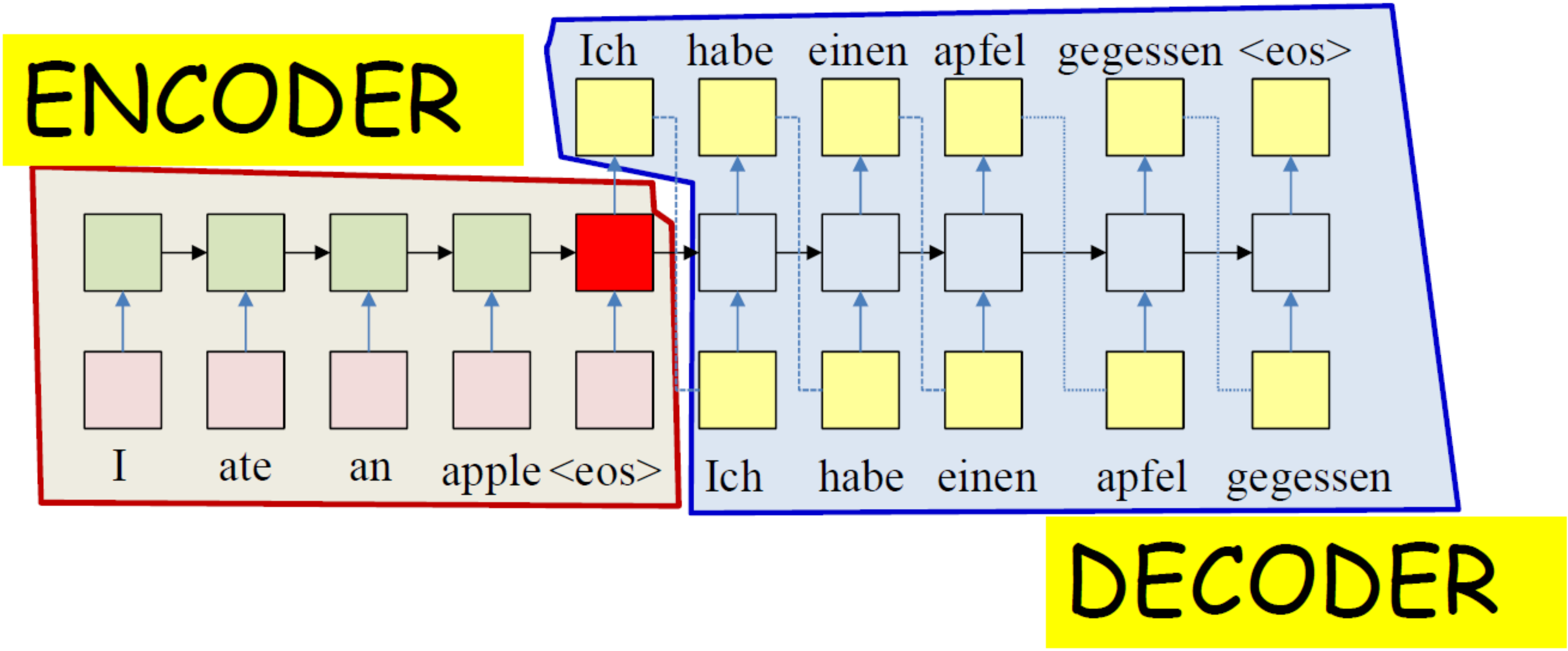
Outline

- RNNs in Vision and NLP
  - X-to-sequence model: NMT and Image Captioning
  - Attention models: NMT and Image Captioning

Acknowledgement: Feifei Li et al's cs231n notes

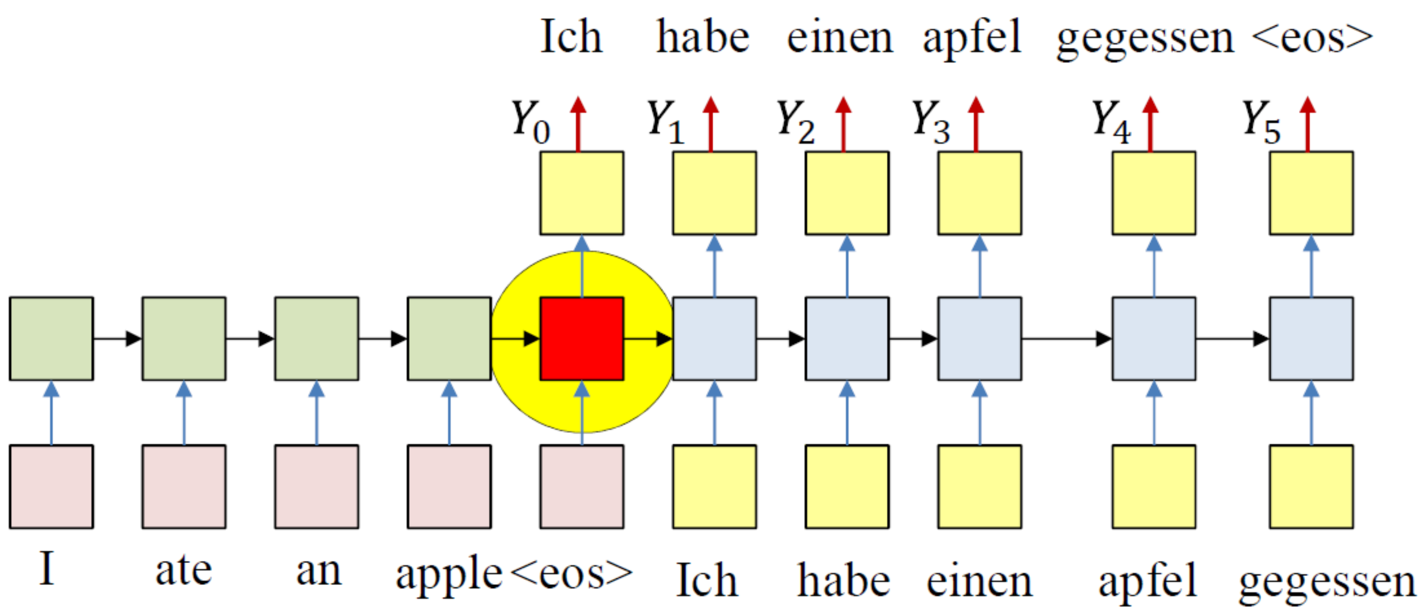
A problem with this framework

- Recall the encoder-decoder structure



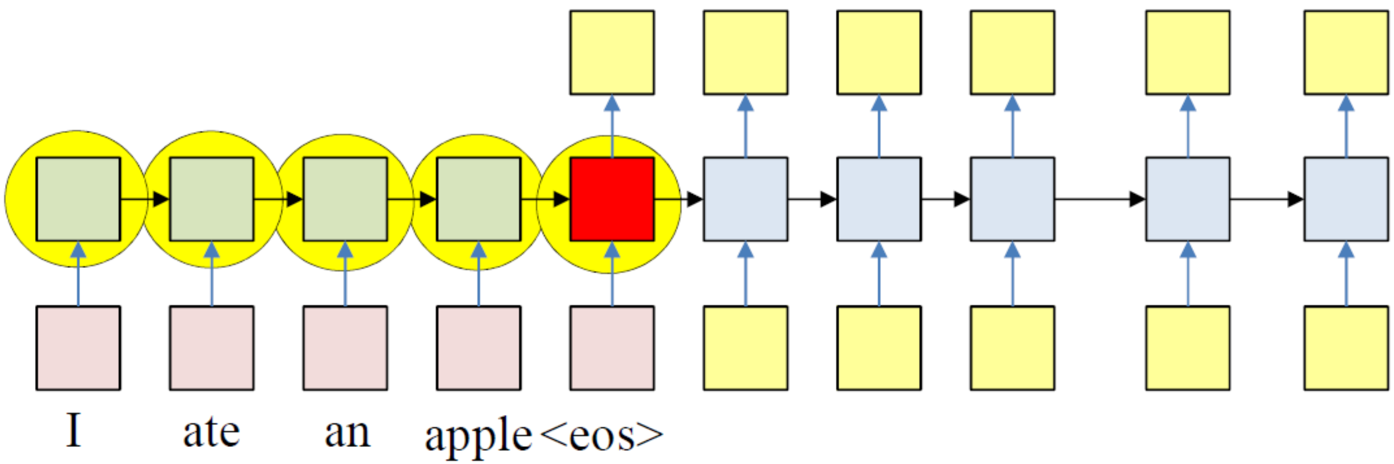
A problem with this framework

- All the information on the input sequence is embedded into a single vector
  - The latent layer at the end of the input sequence
  - This layer is overloaded with information



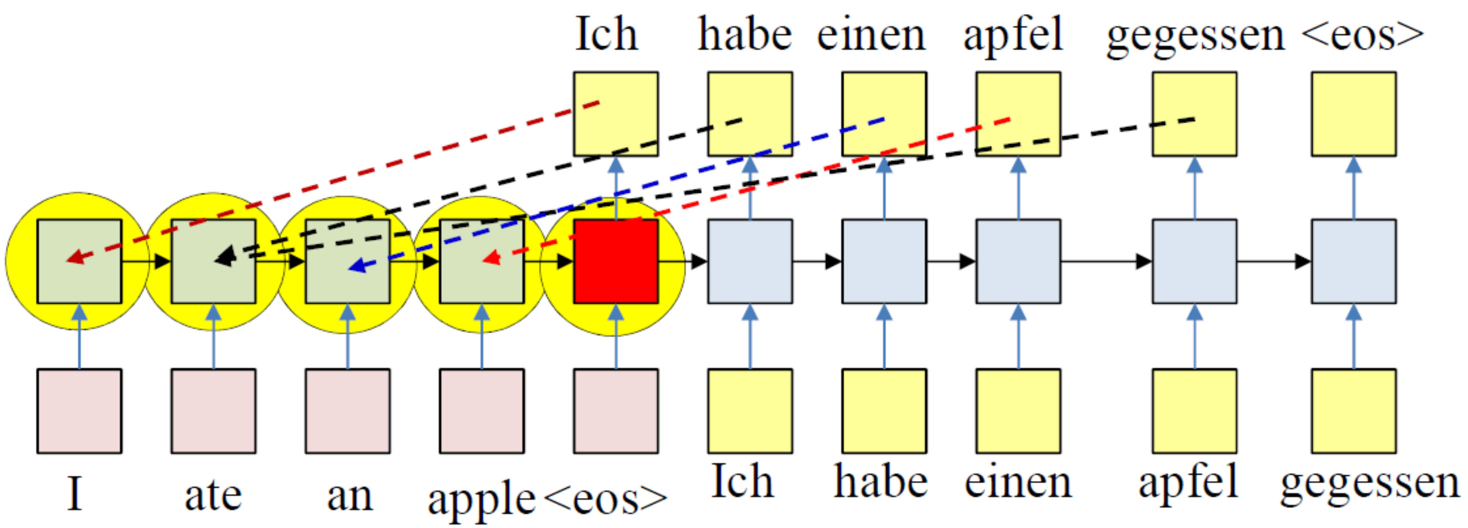
A problem with this framework

- All latent values carry information
  - Some of which may be diluted downstream



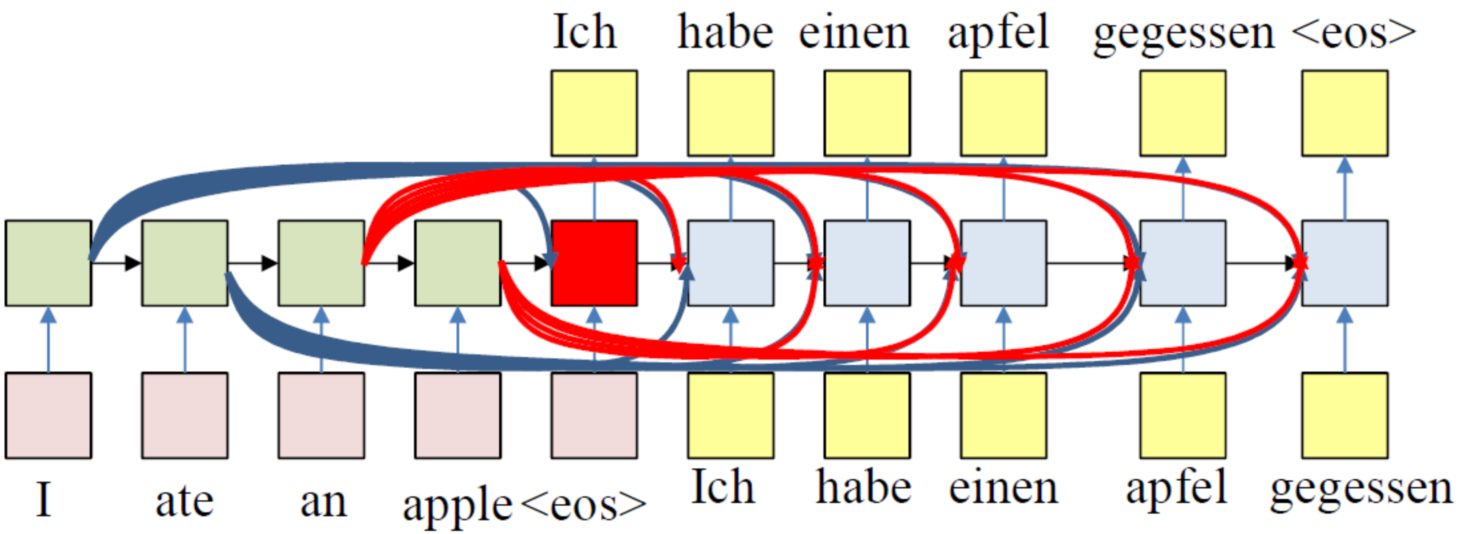
A problem with this framework

- All latent values carry information
  - Some of which may be diluted downstream
  - Different outputs are related to different inputs



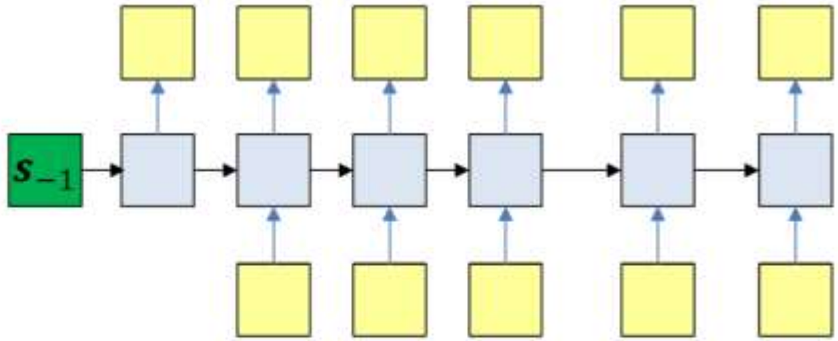
A problem with this framework

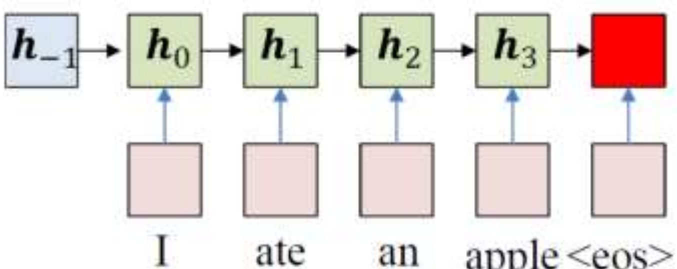
- All latent values carry information
  - Some of which may be diluted downstream
  - Different outputs are related to different inputs
  - Connecting everything to everything is infeasible



Attention models

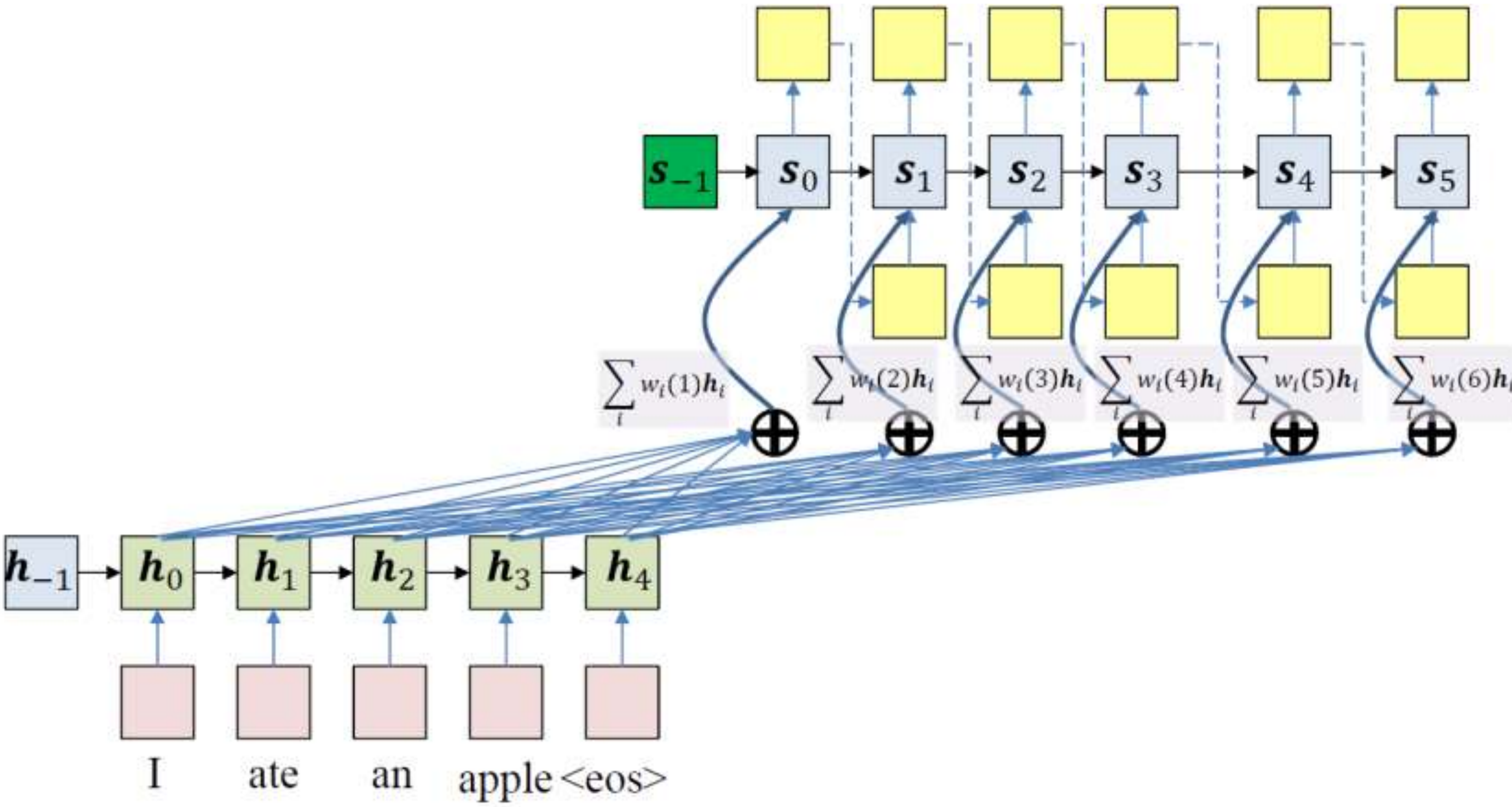
- Separating the encoder and decoder first





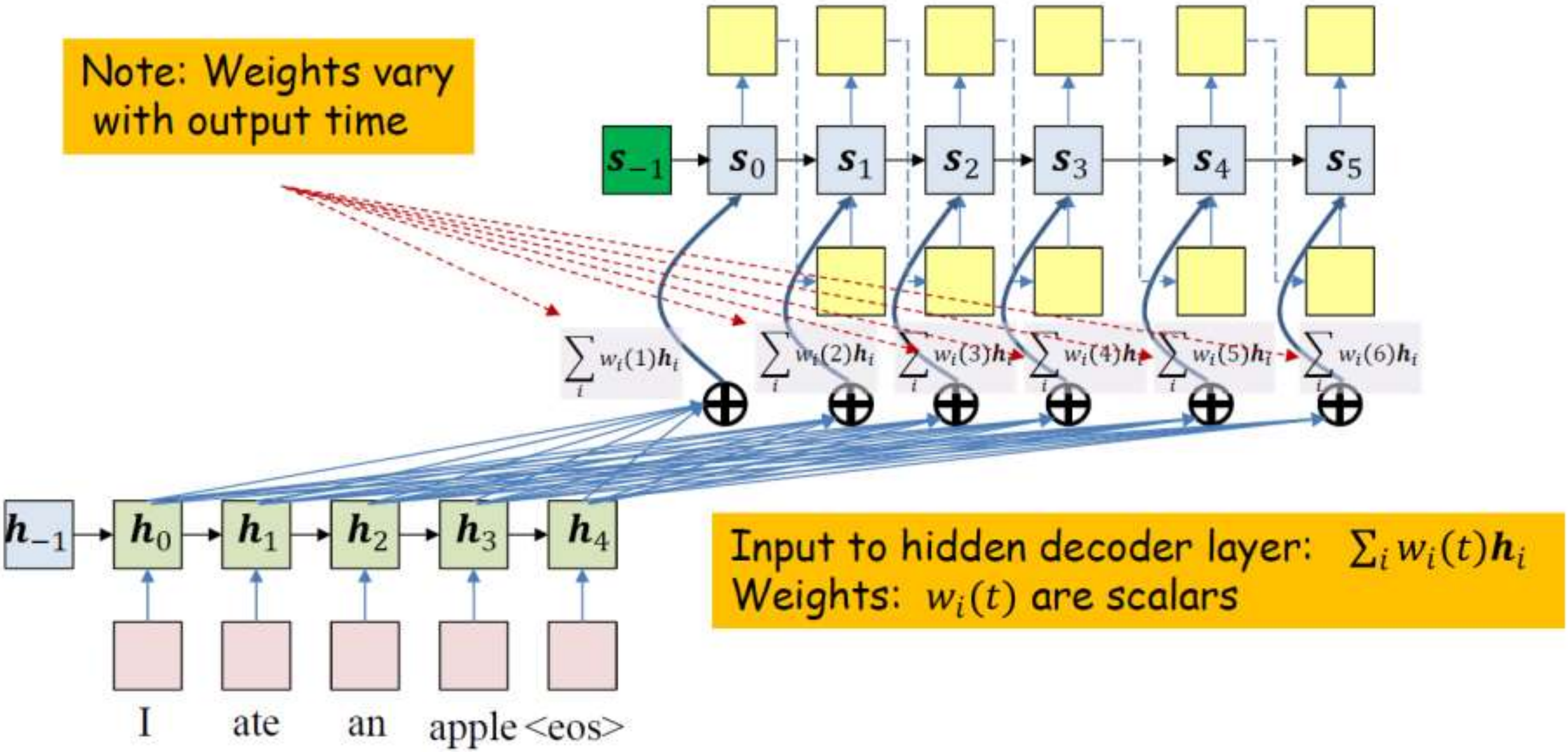
Attention models

- Compute a weighted combination of all the hidden outputs into a single vector



Attention models

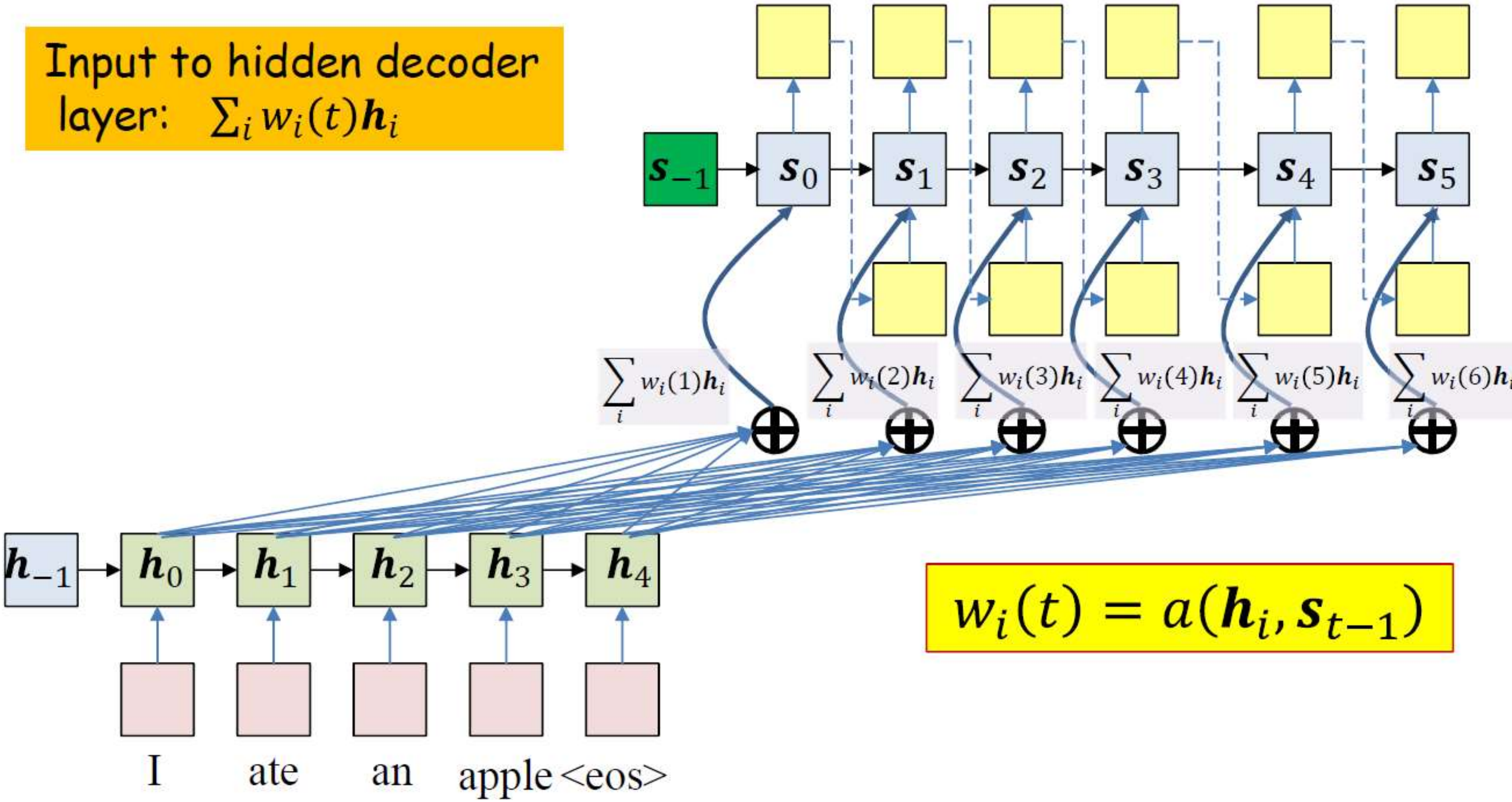
- Compute a weighted combination of all the hidden outputs into a single vector



Attention models

- Require a time-varying weight that specifies relationship of output time to input time
  - Weights are functions of current output state

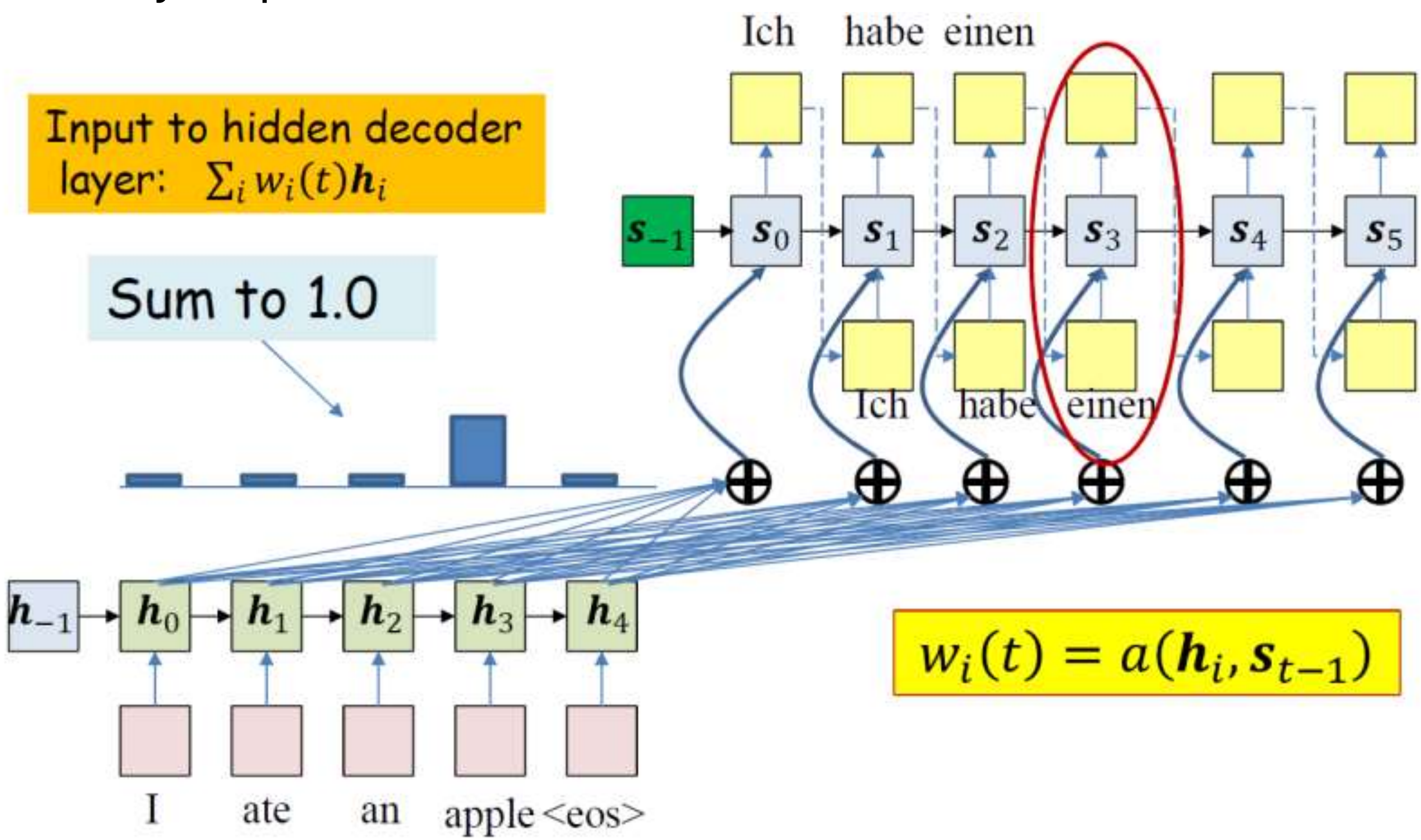
Input to hidden decoder layer: $\sum_i w_i(t)\boldsymbol{h}_i$



$$w_i(t) = a(\boldsymbol{h}_i, \boldsymbol{s}_{t-1})$$

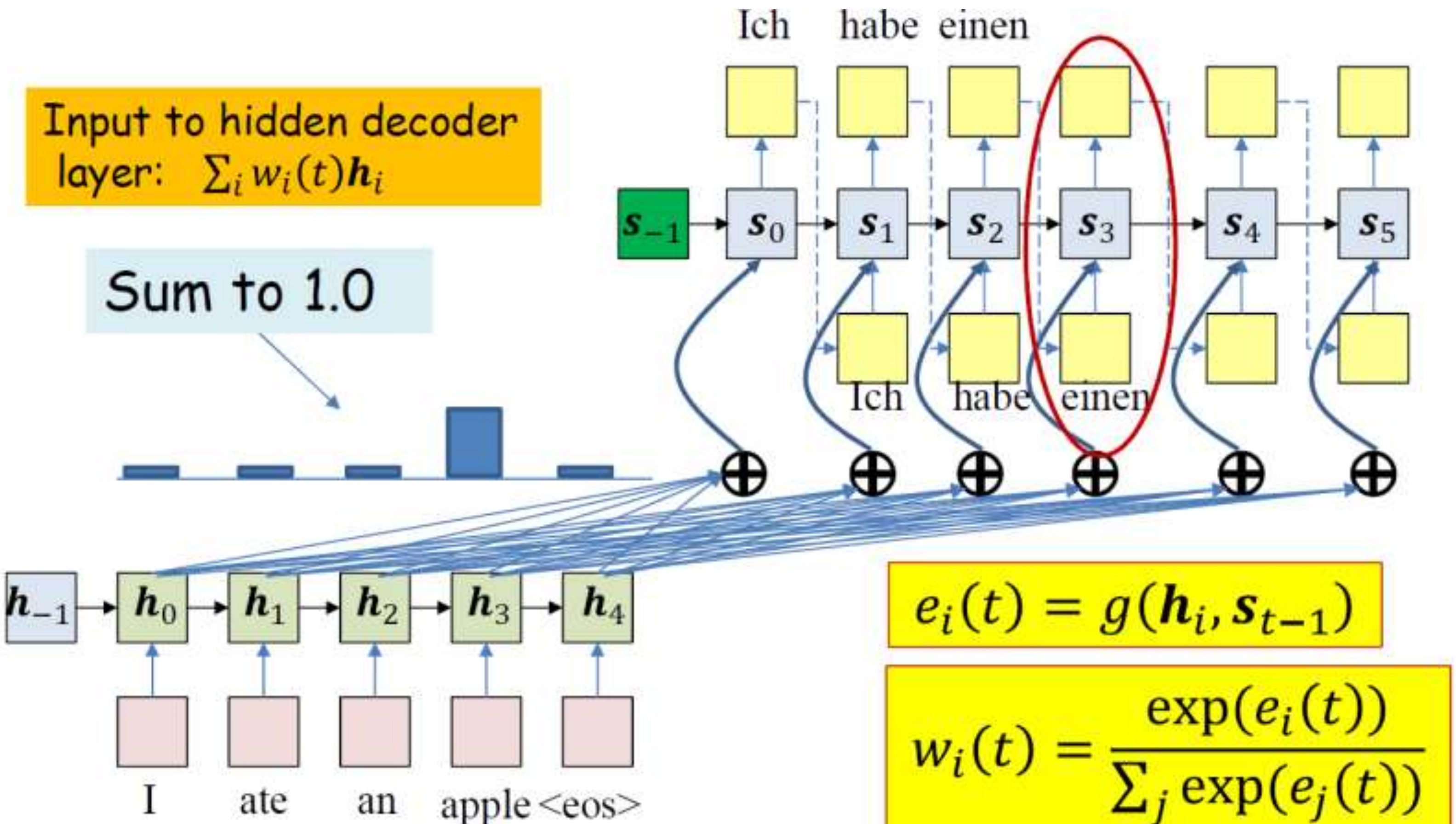
Attention models

- The weights are a distribution over the input
  - Automatically highlight the most important input components for any output



Attention models

- The weights are a distribution over the input
  - A function g() on two hidden states followed by a softmax



$$e_i(t) = g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1})$$

$$w_i(t) = \frac{\exp(e_i(t))}{\sum_j \exp(e_j(t))}$$

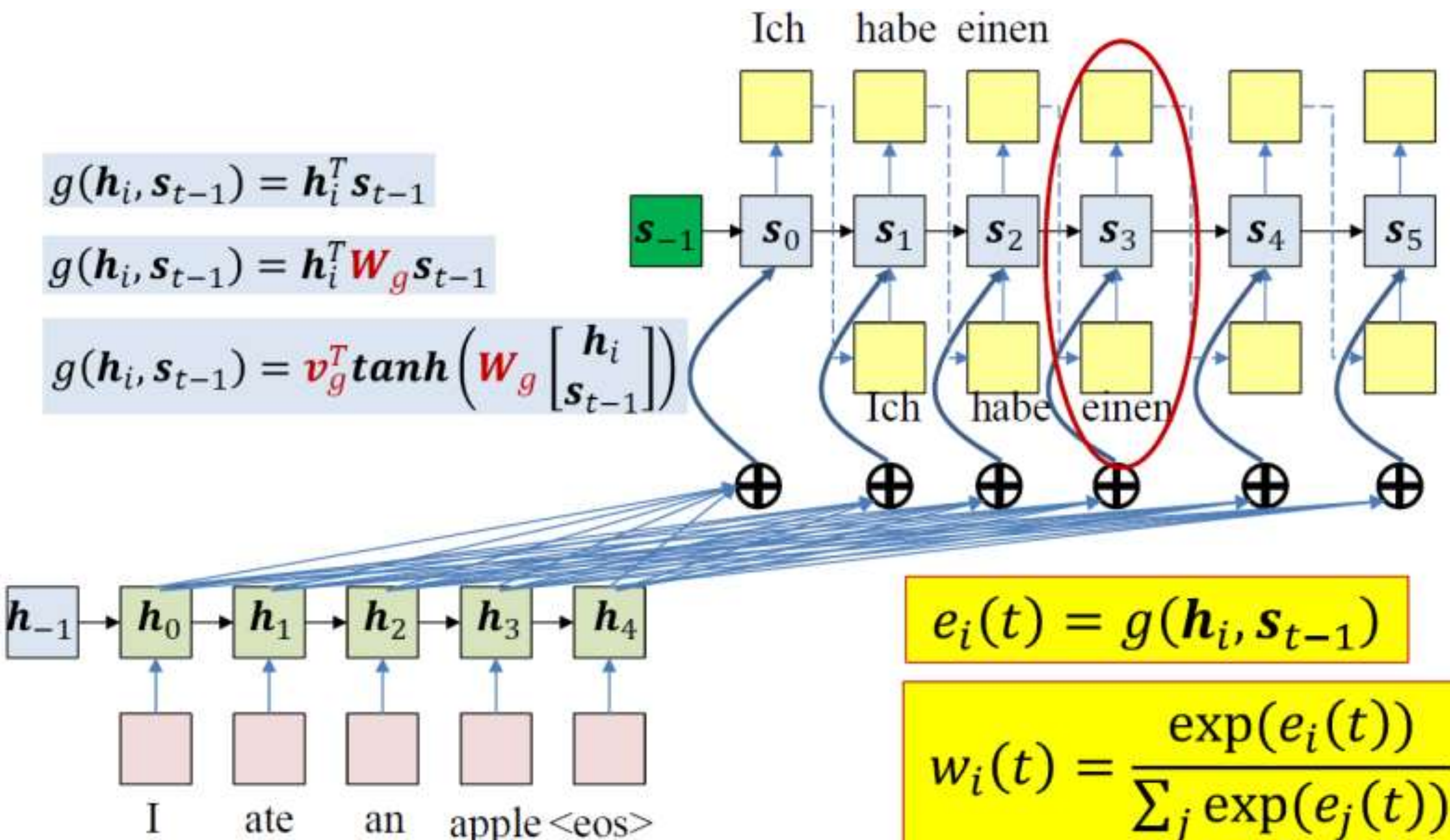
Attention models

- The weights are a distribution over the input
  - Typical options for g() with parameters to be learned

$$g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}) = \boldsymbol{h}_i^T \boldsymbol{s}_{t-1}$$

$$g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}) = \boldsymbol{h}_i^T \boldsymbol{W}_g \boldsymbol{s}_{t-1}$$

$$g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}) = \boldsymbol{v}_g^T \boldsymbol{tanh}\left(\boldsymbol{W}_g \begin{bmatrix} \boldsymbol{h}_i \\ \boldsymbol{s}_{t-1} \end{bmatrix}\right)$$

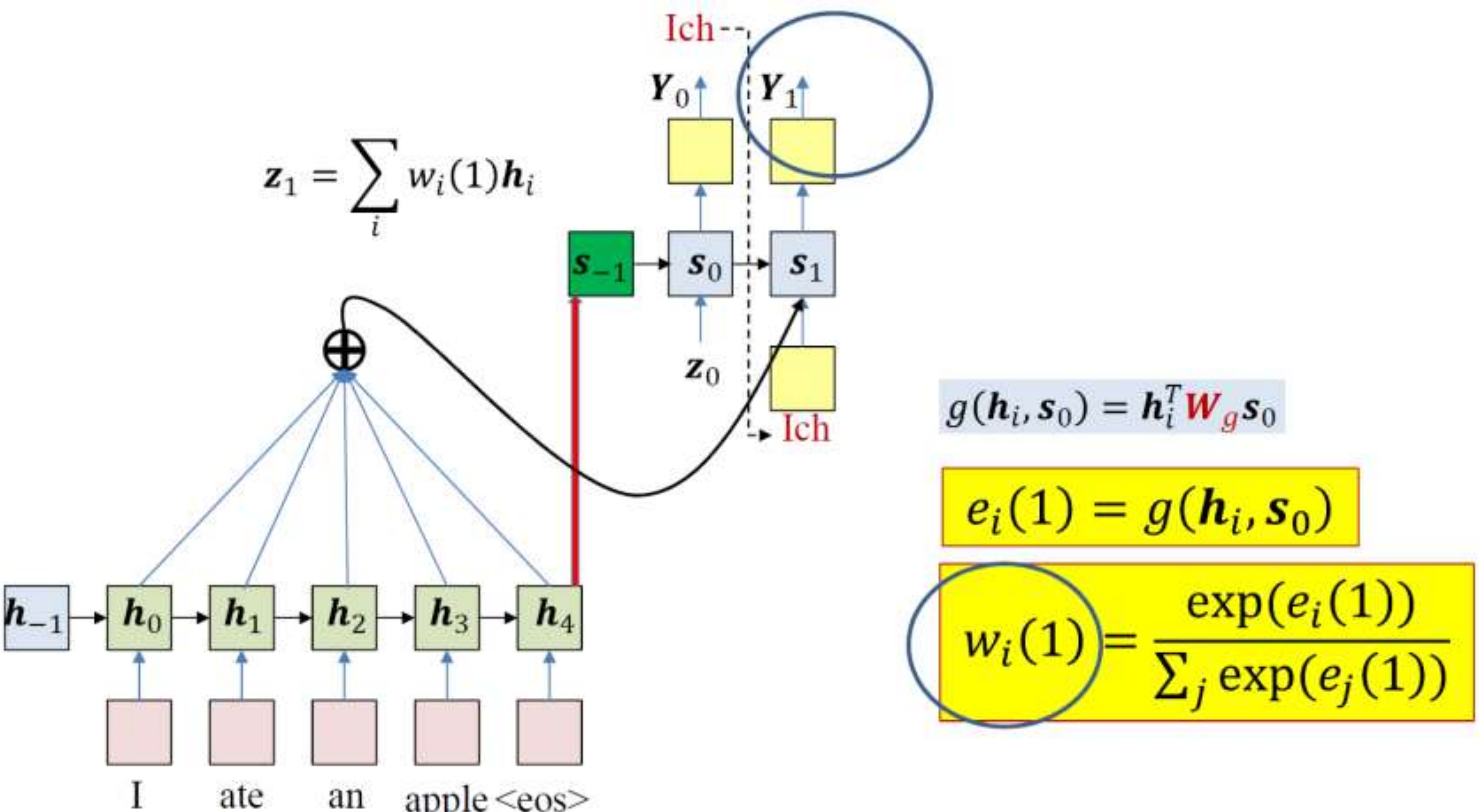


$$e_i(t) = g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1})$$

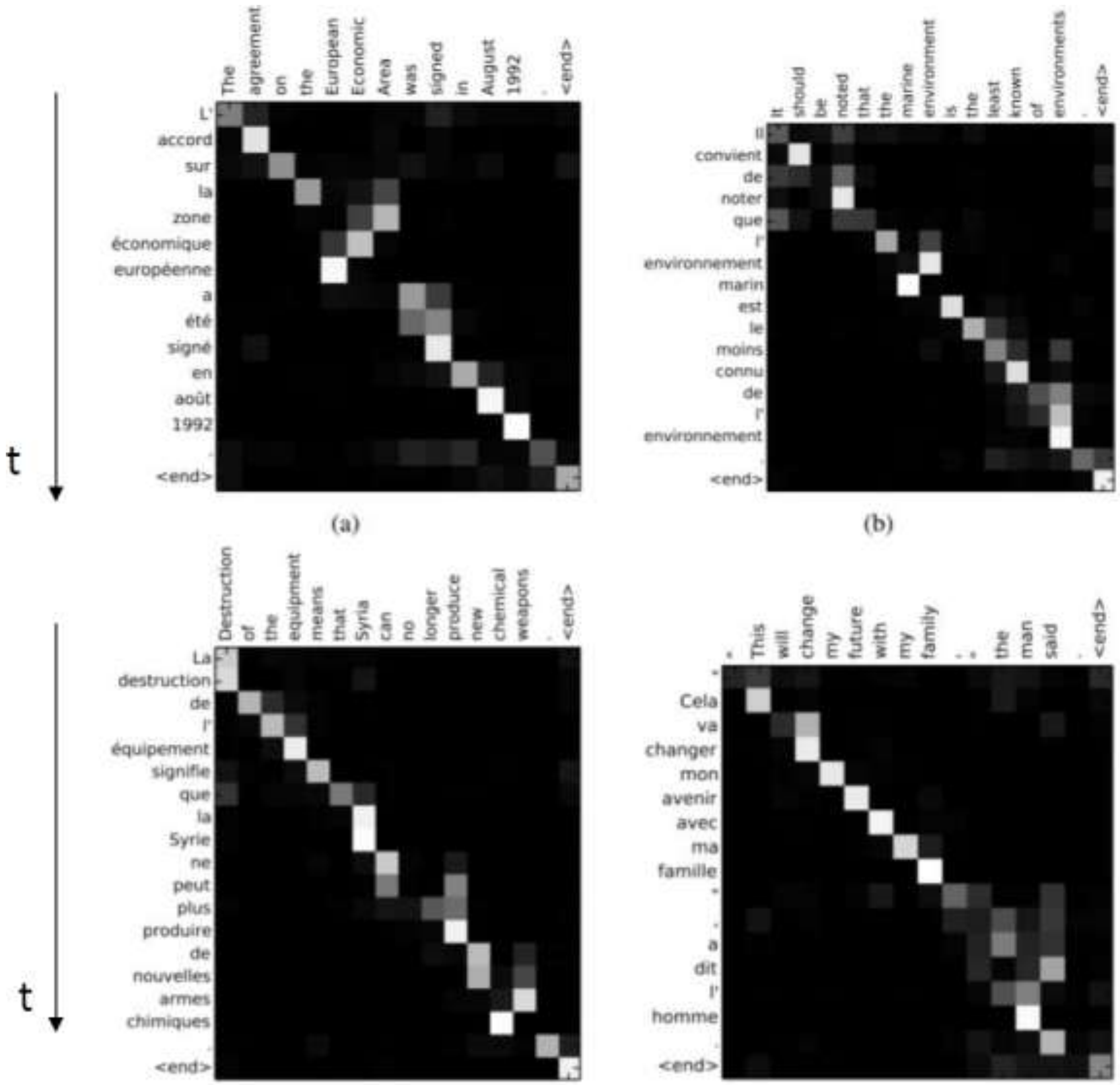
$$w_i(t) = \frac{\exp(e_i(t))}{\sum_j \exp(e_j(t))}$$

What does the attention learn?

- The key component of this model is the attention weight
  - It captures the relative importance of each position in the input to the current output



Attention example



Plot of $w_i(t)$

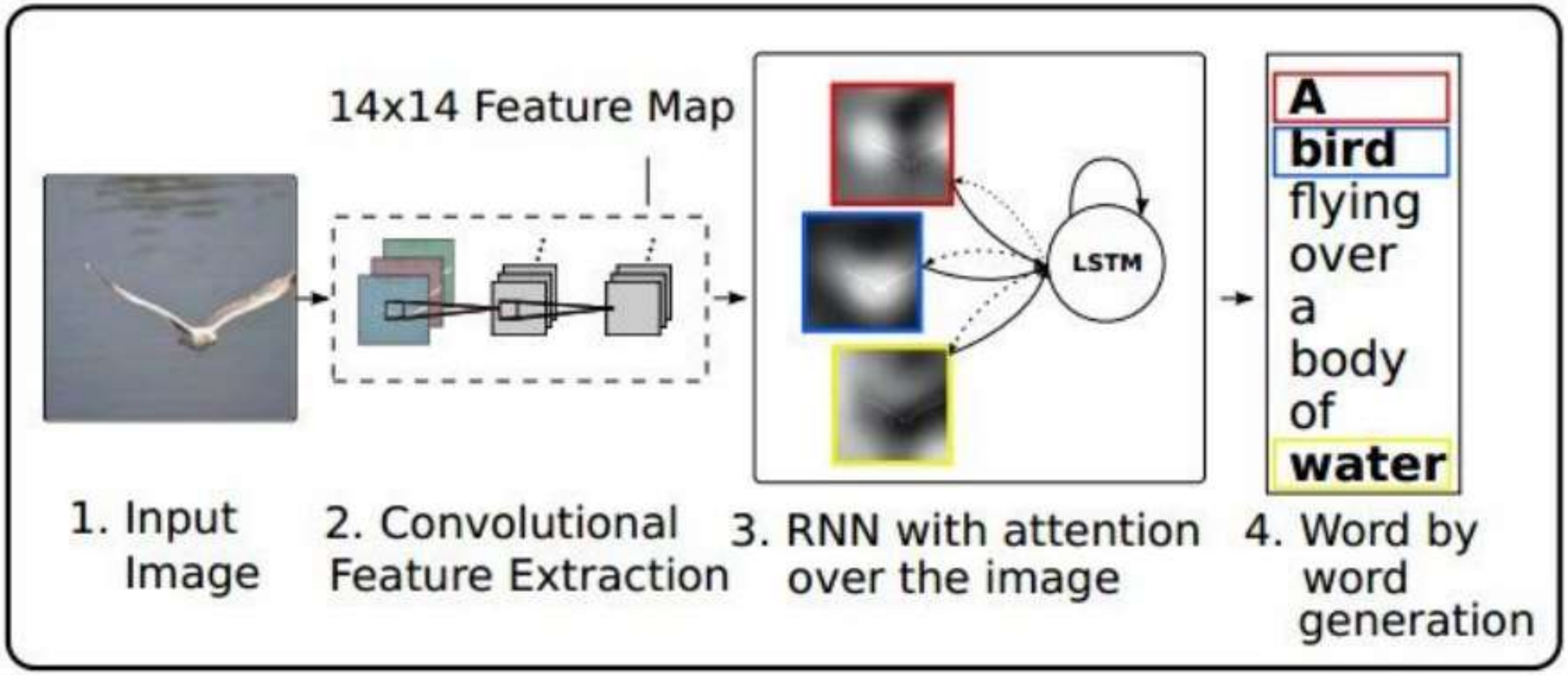
Color shows value (white is larger)

Note how most important input words for any output word get automatically highlighted

The general trend is somewhat linear because word order is roughly similar in both languages

Image Captioning with Attention

RNN focuses its attention at a different spatial location when generating each word



Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention

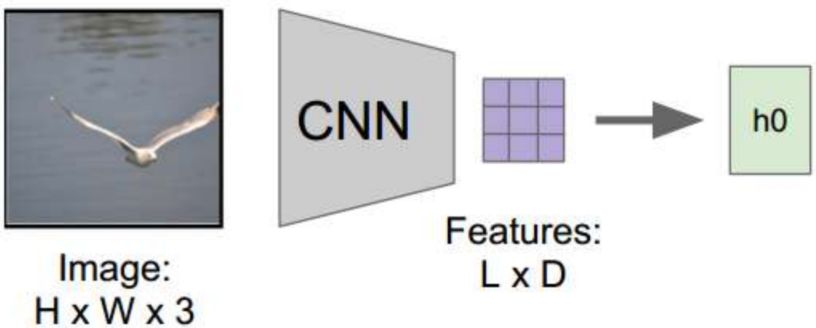
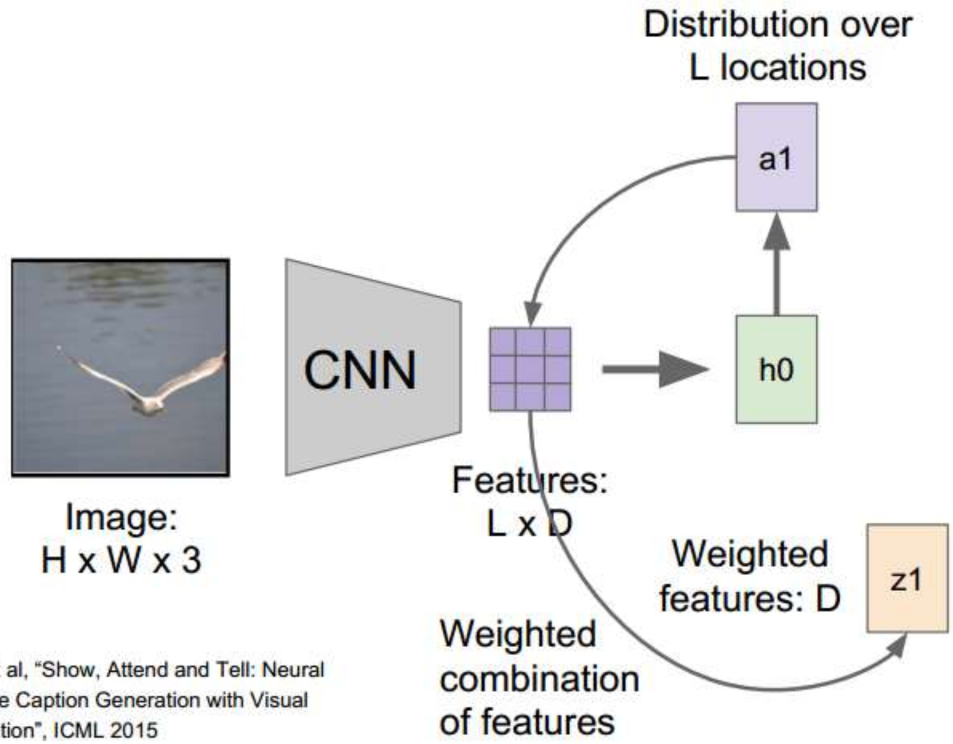


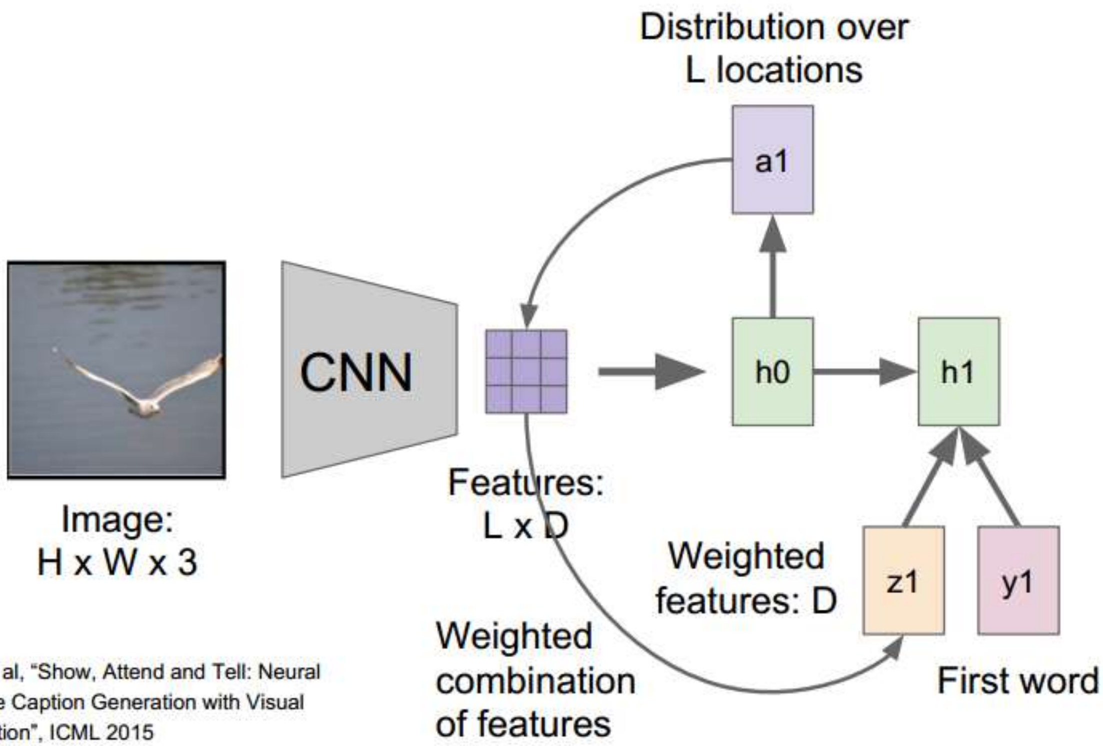
Image Captioning with Attention



$$z = \sum_{i=1}^{L} p_i v_i$$

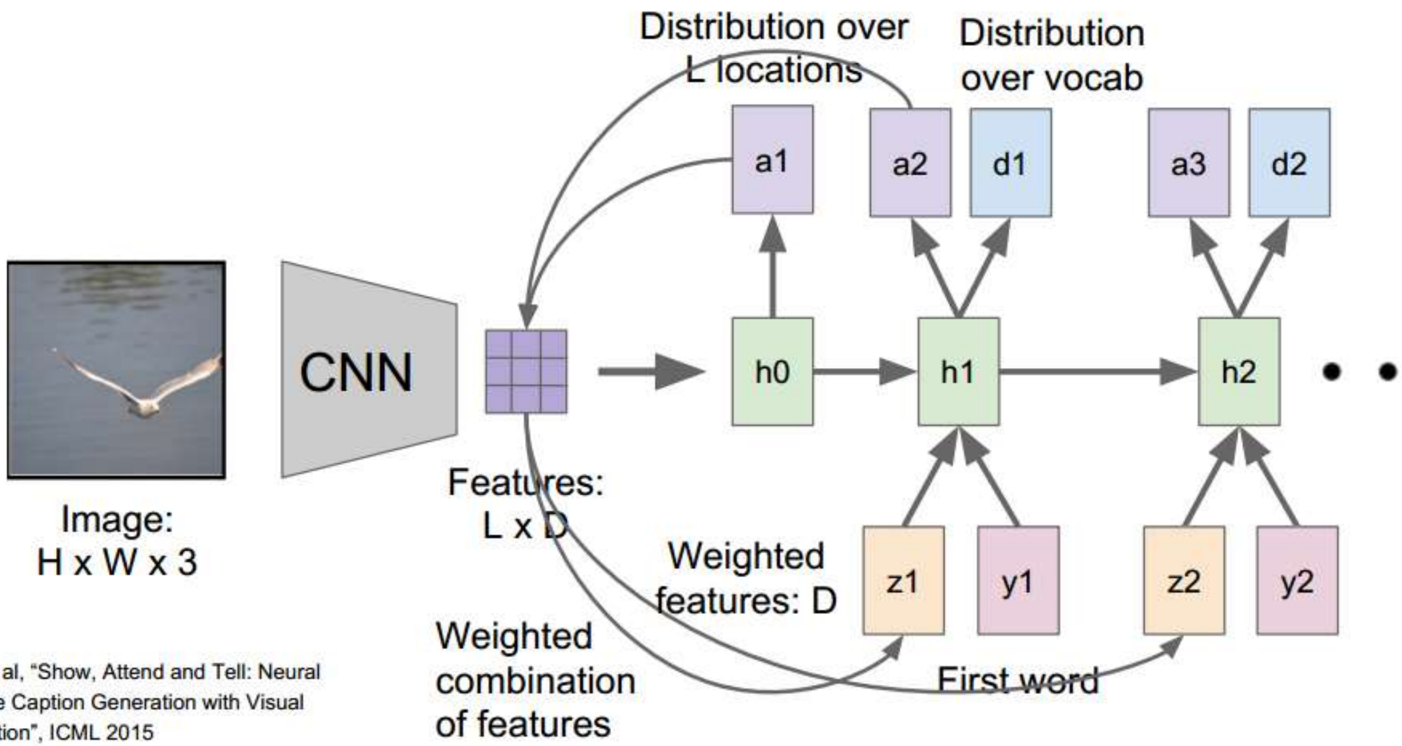
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



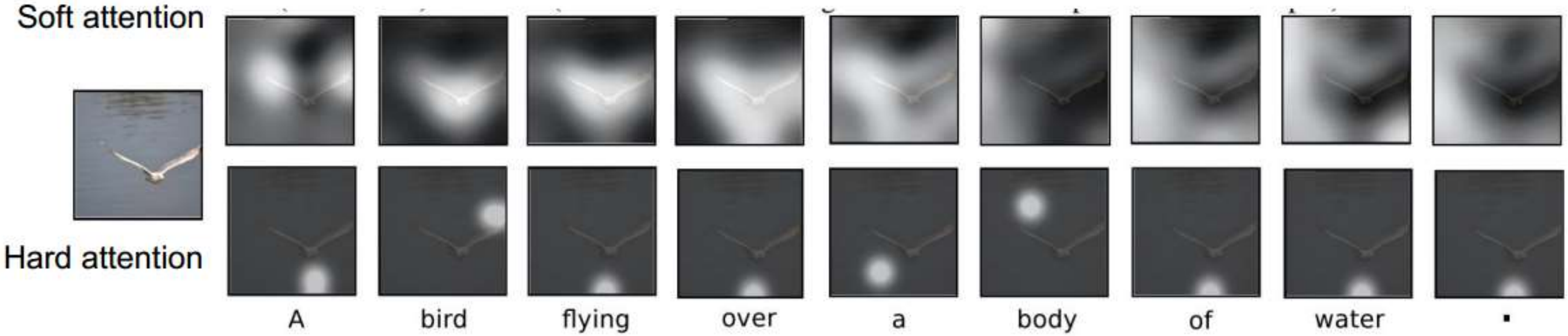
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



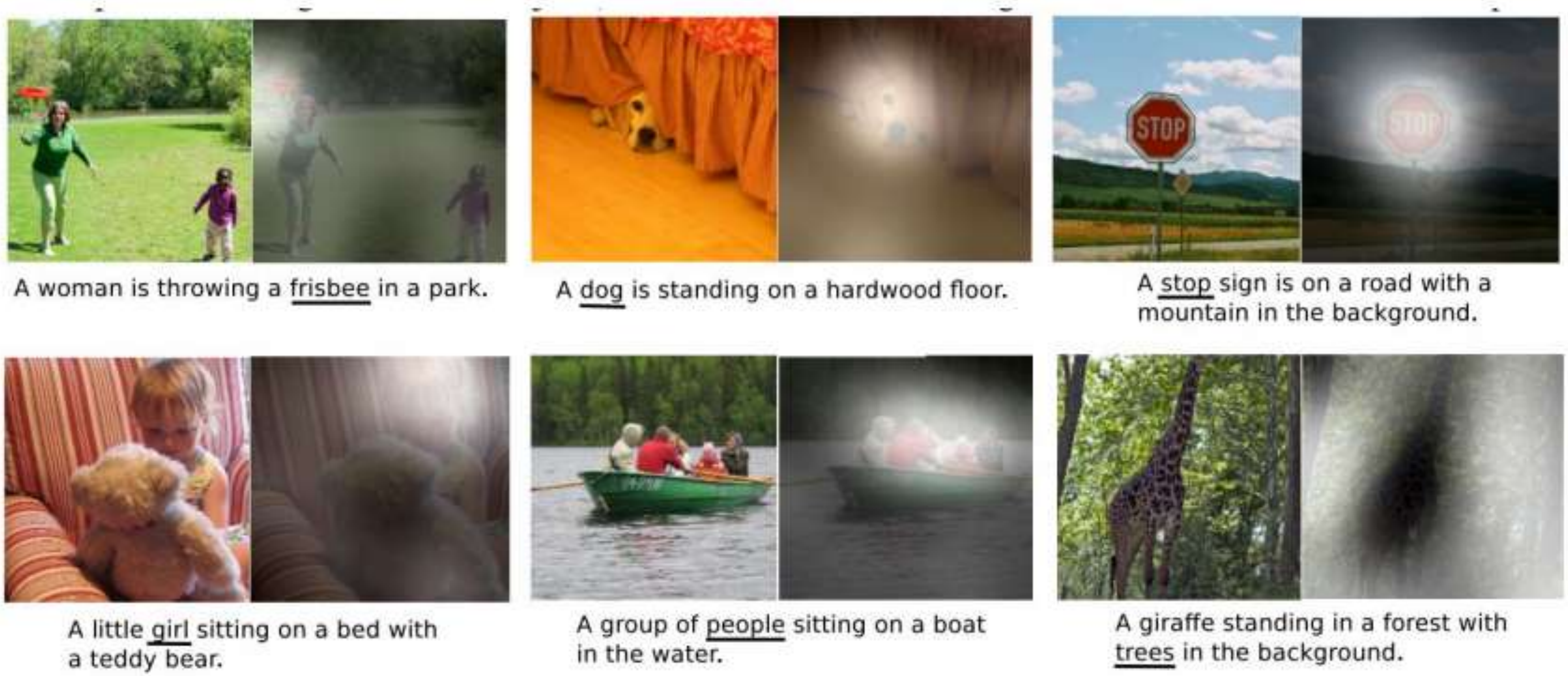
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



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.

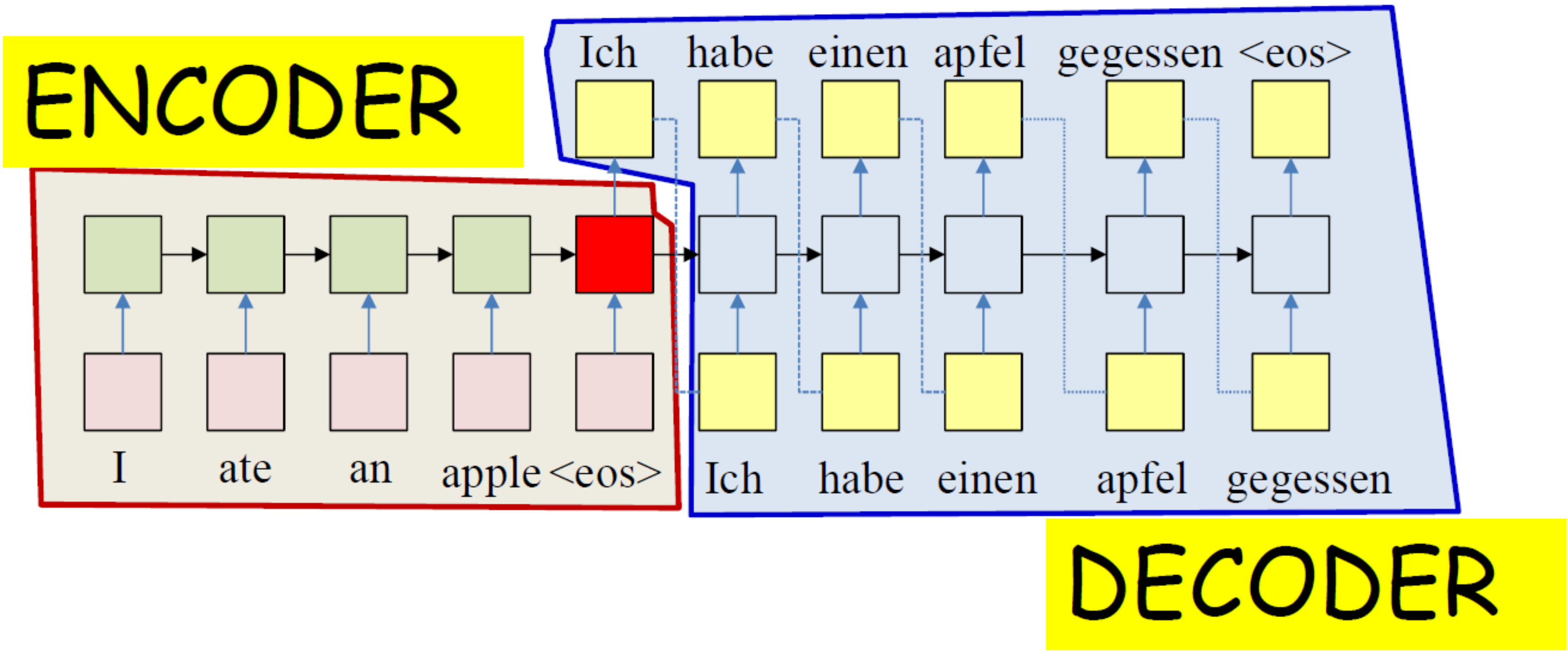
A little girl sitting on a bed with a teddy bear.

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

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

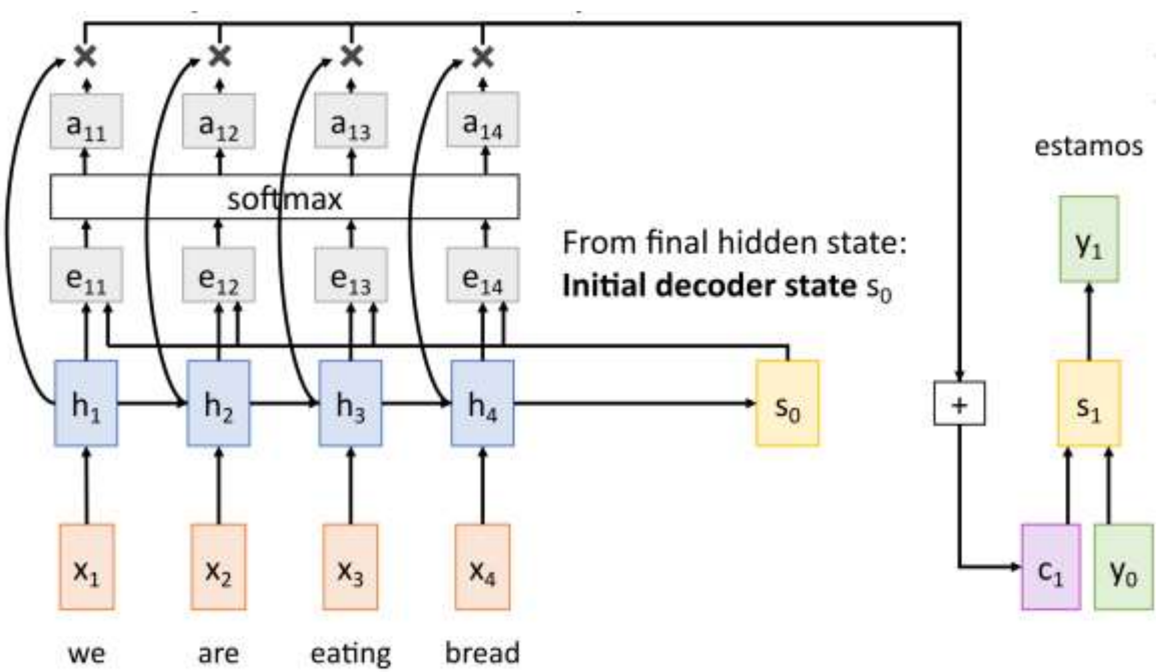
Xu et al, “Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Recall the encoder-decoder structure



NMT with RNN and Attention

- A time-varying weighted combination of all the hidden outputs into a single vector



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

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

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

NMT with RNN and Attention

- At each timestep of decoder, context vector "looks at" different parts of the input sequence.

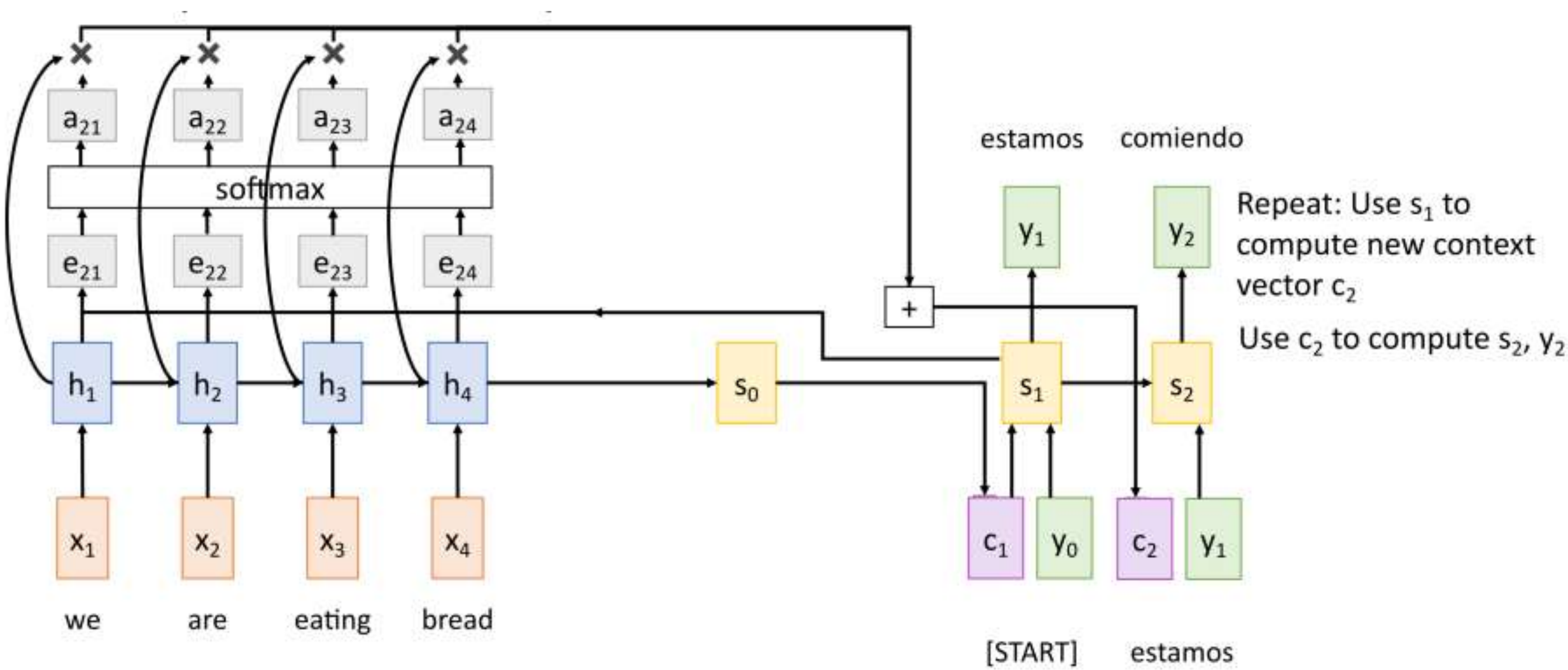


Image Caption with RNN and Attention

- Recall the one using spatial features
- Problem: Input is “bottlenecked” through c

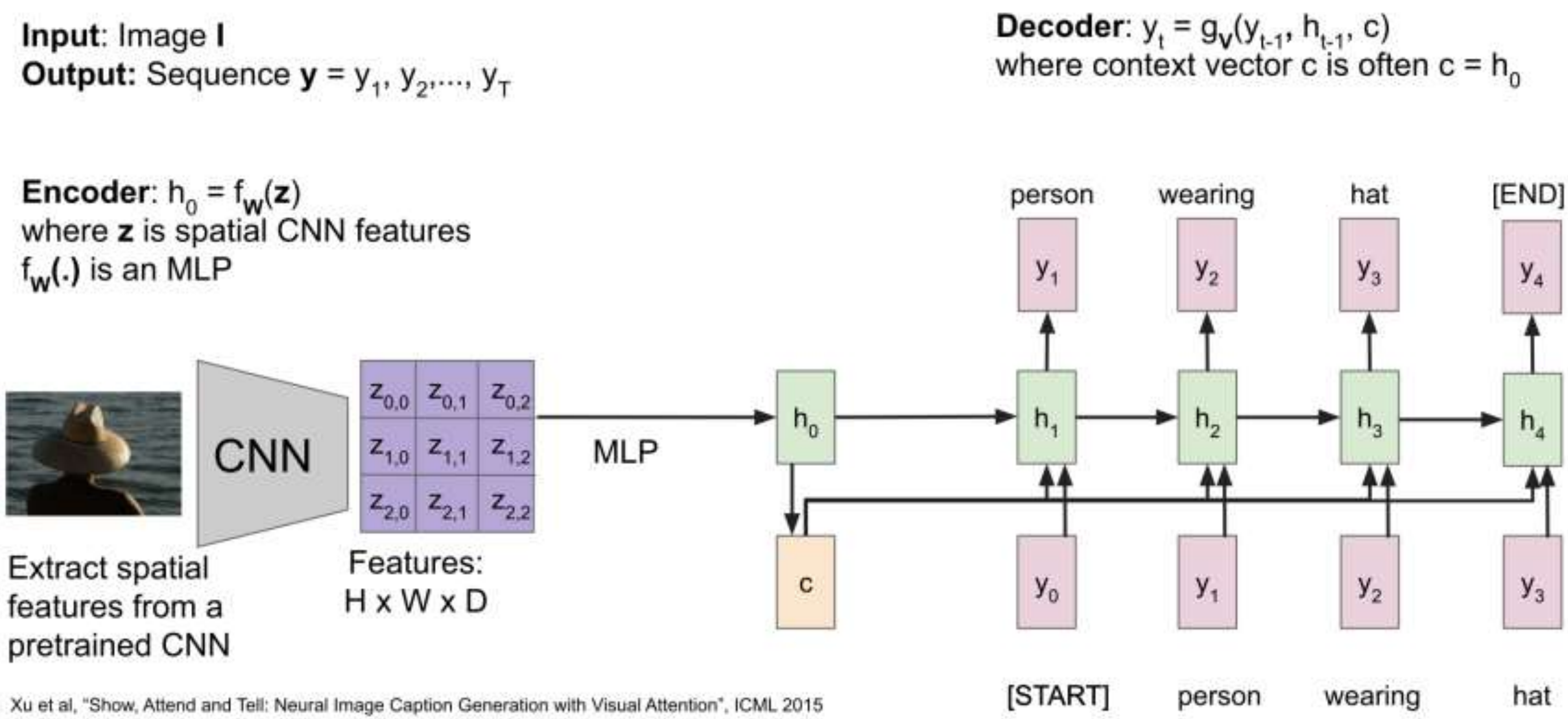
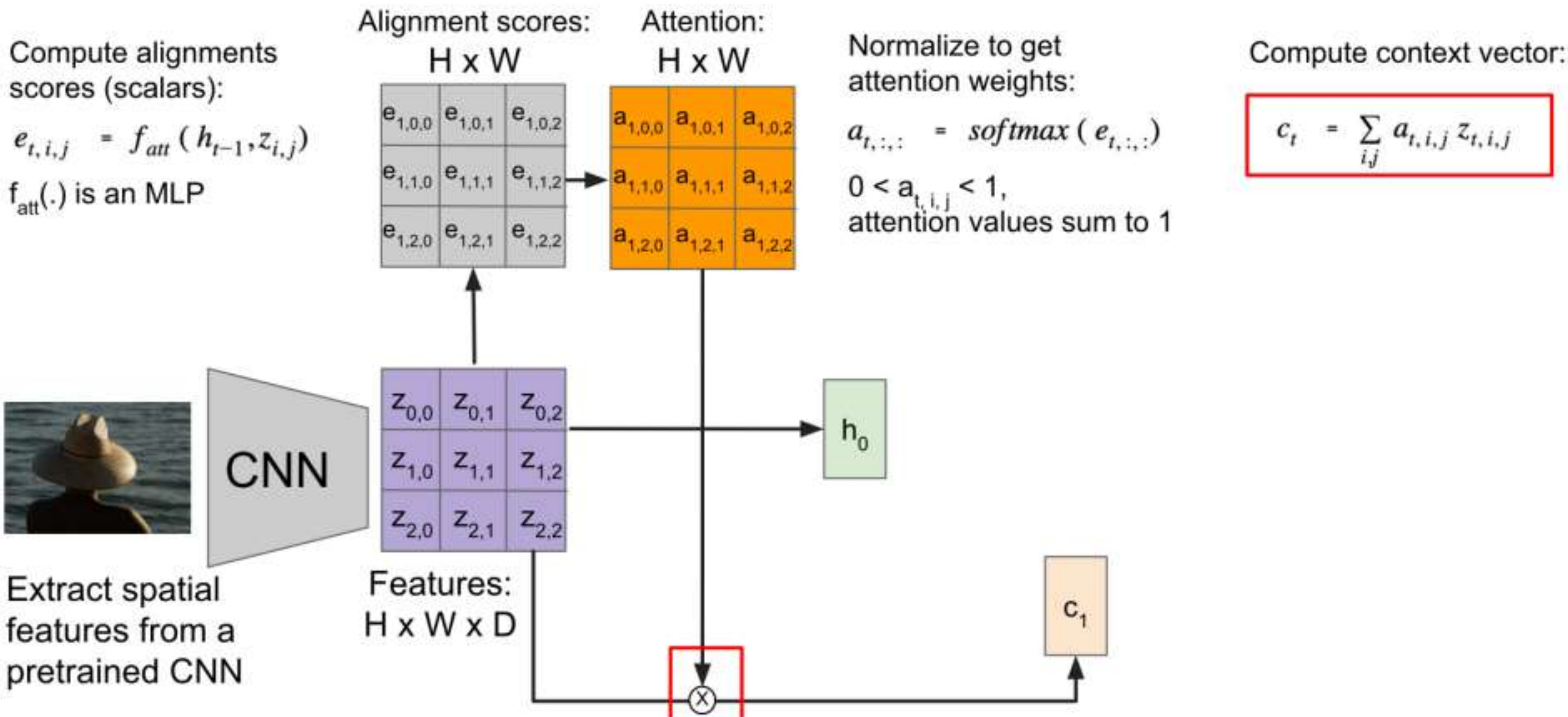


Image Caption with RNN and Attention

- Alignment → Attention



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

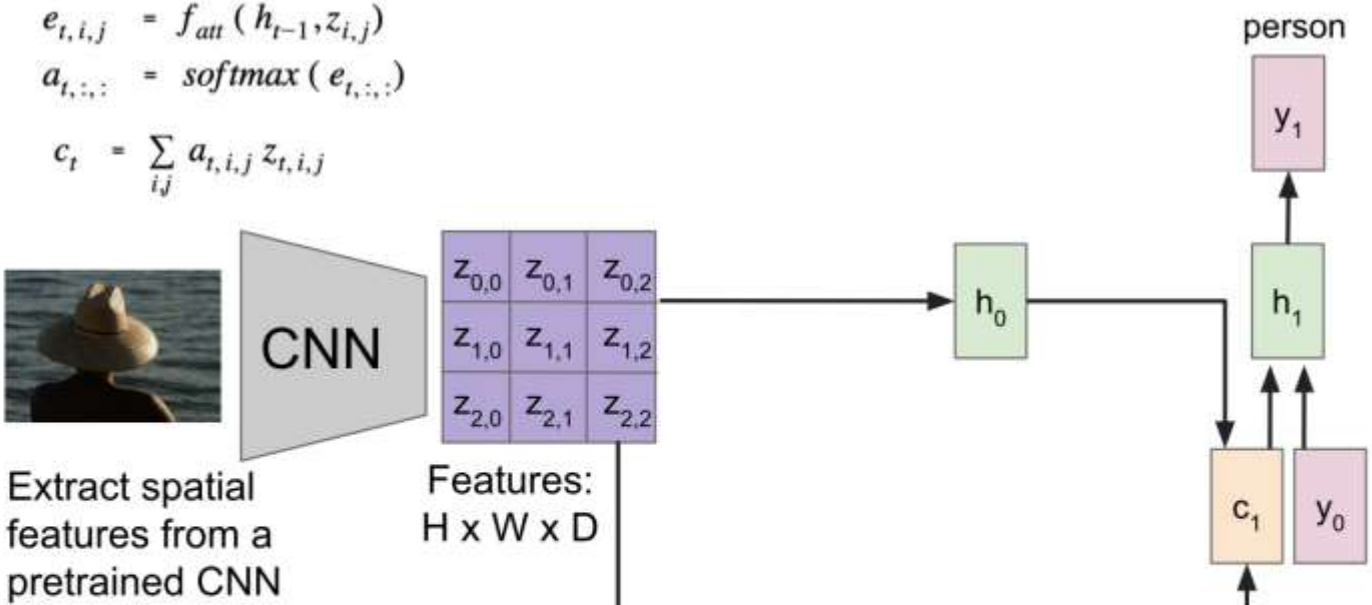
Image Caption with RNN and Attention

- Weighted context → decoder

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

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

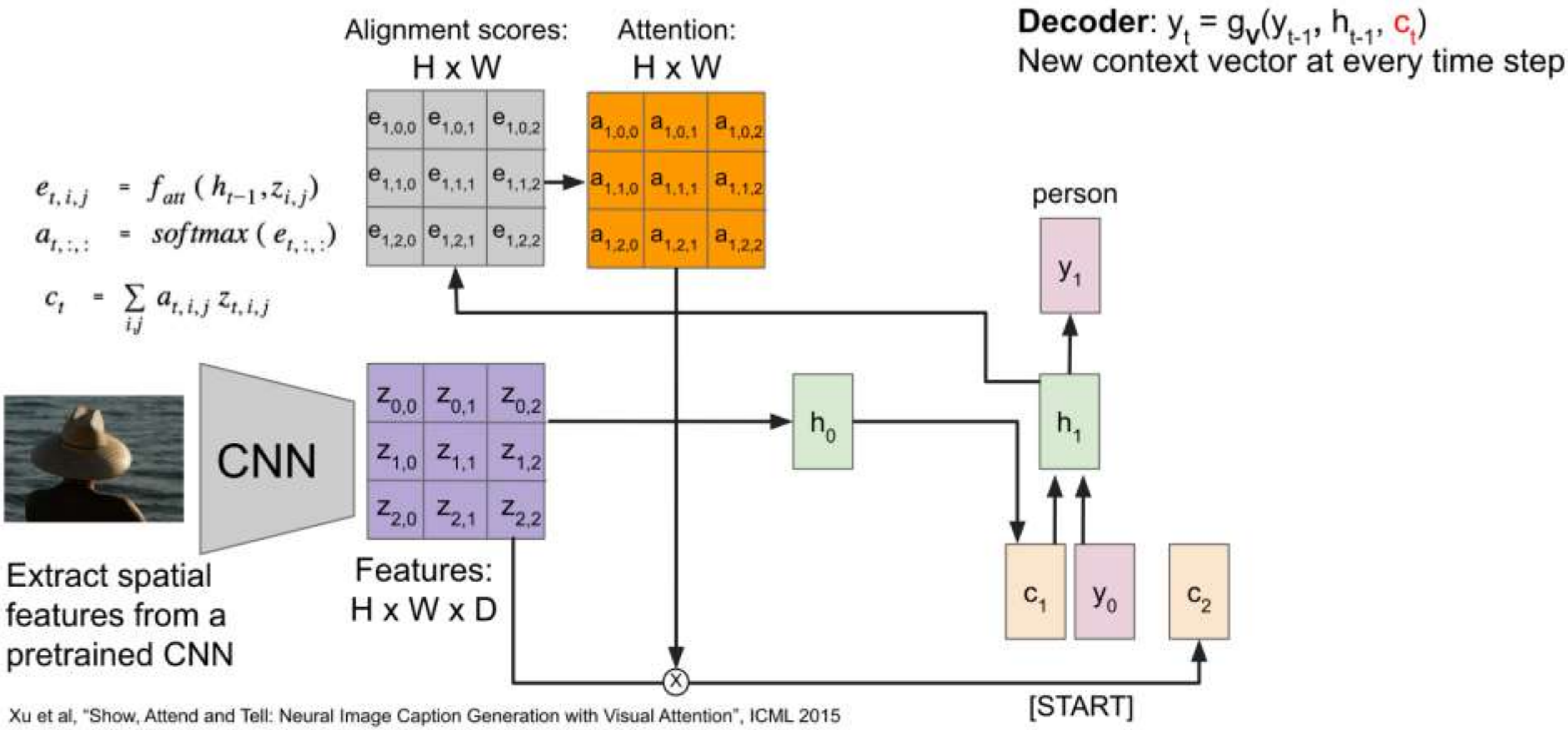
$$e_{t,i,j} = f_{att}(h_{t-1}, z_{i,j})$$
$$a_{t,:,:} = softmax(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} z_{t,i,j}$$



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Caption with RNN and Attention

- Compute the new Alignment & Attention maps



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

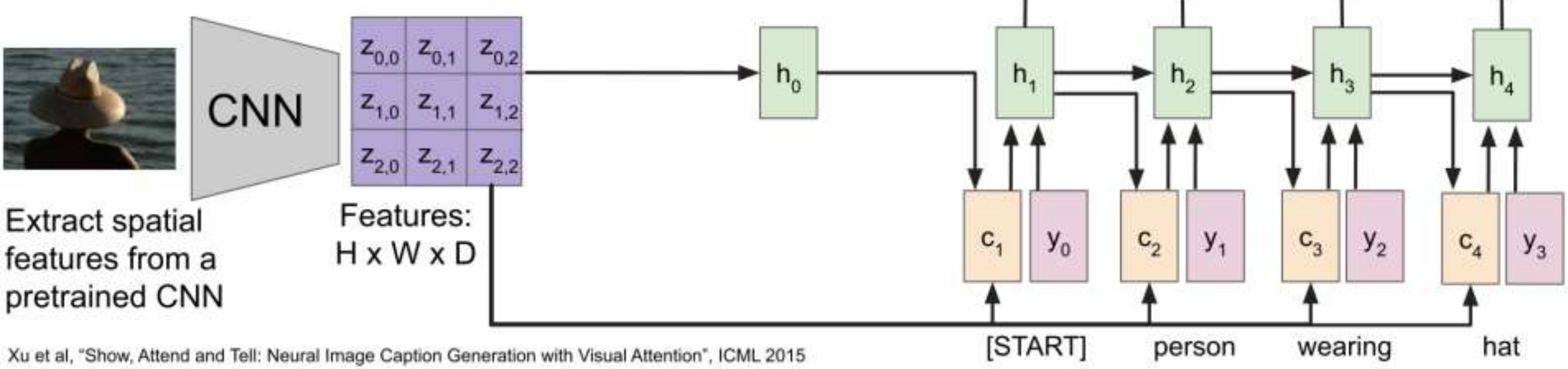
Image Caption with RNN and Attention

- Repeat …

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

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c_t)$
New context vector at every time step

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



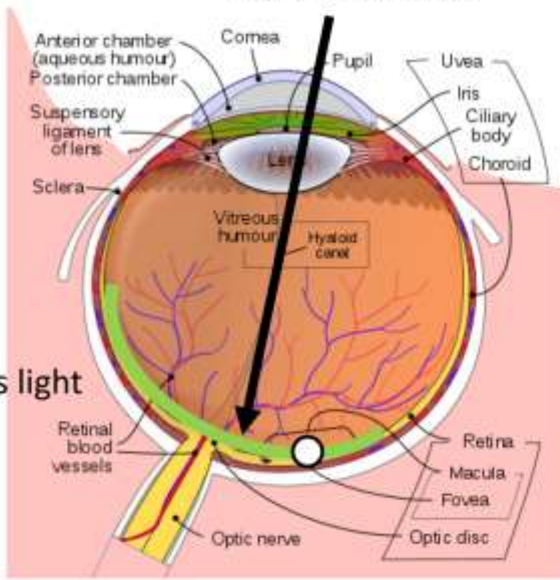
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Attention Mechanism

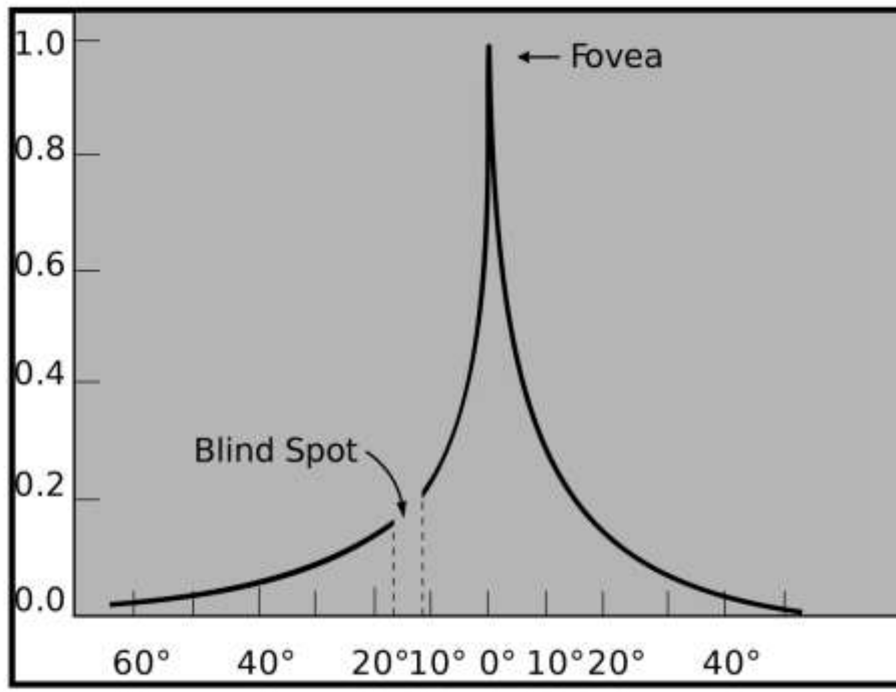
- Human Vision: Fovea

Light enters eye

Retina
detects light



The **fovea** is a tiny region of the retina that can see with high acuity



X, Attend, and Y

"**Show, attend, and tell**" *(Xu et al, ICML 2015)*
Look at image, attend to image regions, produce question

"**Ask, attend, and answer**" *(Xu and Saenko, ECCV 2016)*
"**Show, ask, attend, and answer"** *(Kazemi and Elqursh, 2017)*
Read text of question, attend to image regions, produce answer

---

"**Listen, attend, and spell**" *(Chan et al, ICASSP 2016)*
Process raw audio, attend to audio regions while producing text

"**Listen, attend, and walk**" *(Mei et al, AAAI 2016)*
Process text, attend to text regions, output navigation commands

"**Show, attend, and interact**" *(Qureshi et al, ICRA 2017)*
Process image, attend to image regions, output robot control commands

"**Show, attend, and read**" *(Li et al, AAAI 2019)*
Process image, attend to image regions, output text

Summary

- RNN
  - X-to-Sequence model
  - Attention model
- Next time:
  - General Attention Layer
  - From Attention to Transformer