Lecture 14: Recurrent Neural Networks III: Sequence Models in NLP and Vision

Lan Xu

SIST, ShanghaiTech

Fall, 2021

Outline

- **RNNs in Vision and NLP**

  - ☐ X-to-sequence model: NMT and Image Captioning
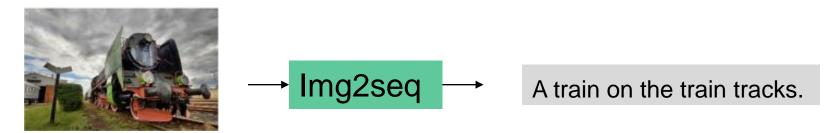
  - ☐ Attention models: NMT and Image Captioning

Acknowledgement: Feifei Li et al's cs231n notes

X-to-Sequence problems

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

I ate an apple → Seq2seq → Ich habe einen apfel gegessen

  - □ 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"
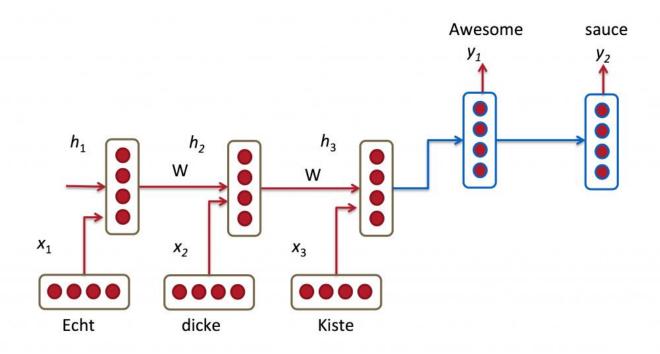
Sequence-to-sequence problem

- **Task definition**

I ate an apple $\longrightarrow$ Seq2seq $\longrightarrow$ Ich habe einen apfel gegessen

- ☐ 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**
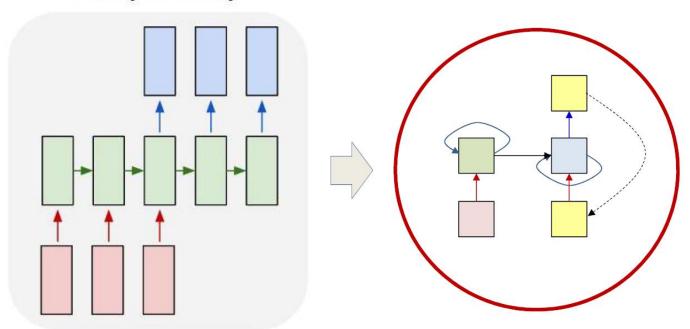 - Many to many

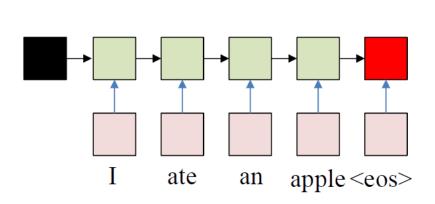http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/
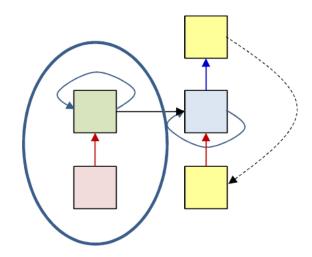
Modeling the problem

■ Delayed sequence to sequence

  ☐ Delayed self-referencing sequence-to-sequence

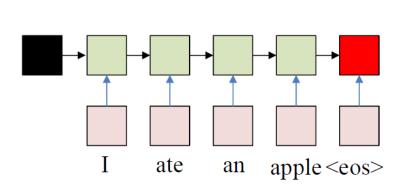many to many

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

I   ate   an   apple <eos>
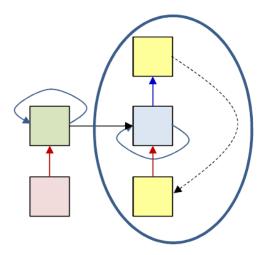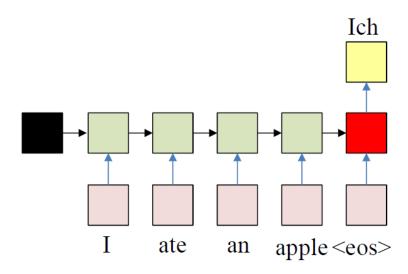
The "simple" translation model

- 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
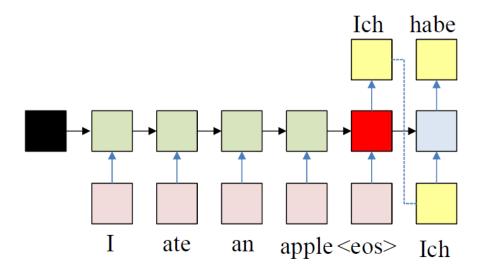
The "simple" translation model

- 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
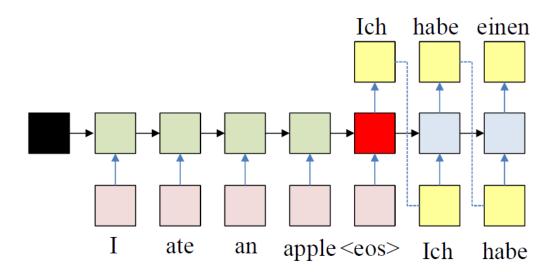
The "simple" translation model

- 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
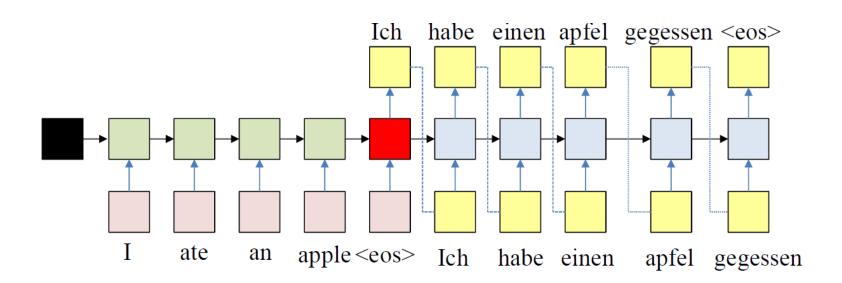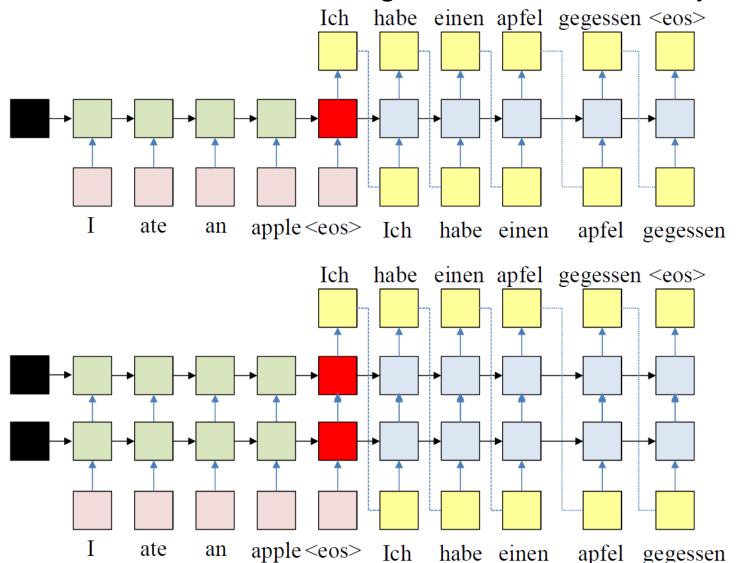
The "simple" translation model

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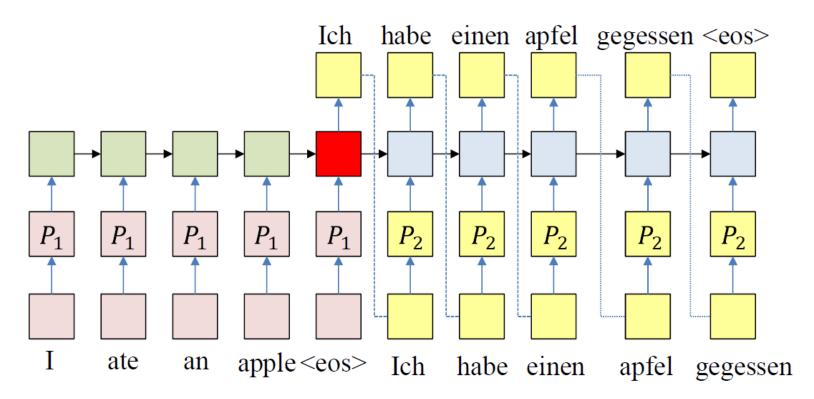
- This is also referred to as an encoder-decoder structure

The "simple" translation model

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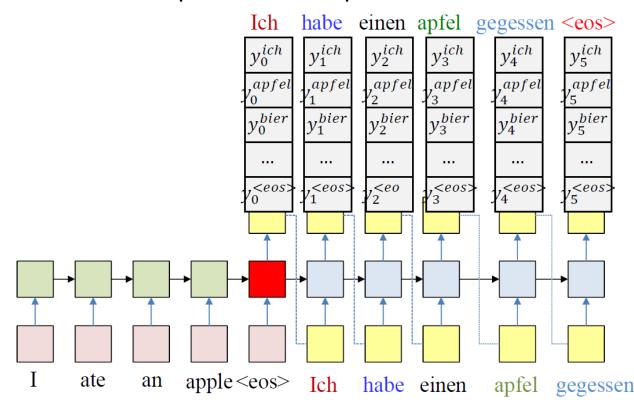

Lan Xu – CS 280 Deep Learning

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

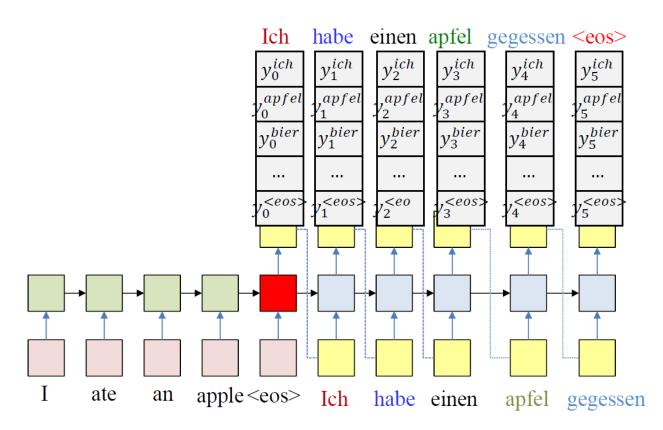 □ The drawn word is provided as input to the next time, until <eos>

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

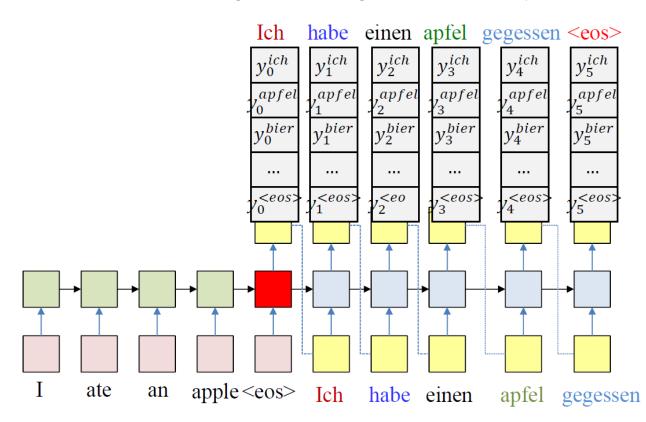$$\arg\max_{O_1, \cdots, O_L} y_1^{O_1} y_2^{O_2} \cdots y_L^{O_L}$$

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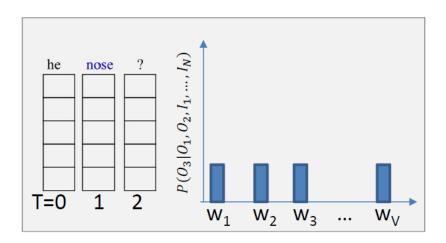
- **Solving** $\arg\max_{O_1,\cdots,O_L} y_1^{O_1} y_2^{O_2} \cdots y_L^{O_L}$
  - ☐ Cannot pick the most likely symbol at each time
  - ☐ Reason?
    - Choosing a different, less likely word could cause the probability at the next time to be larger, resulting in a more likely output overall

Why greedy is not good
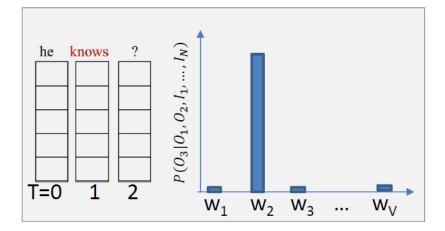
- **Example from speech recognition**

"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

What should we have chosen at t=2??

Will selecting "nose" continue to have a bad effect into the distant future?

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

w_1 w_2 w_3 ... w_V

T=0 1 2

- 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
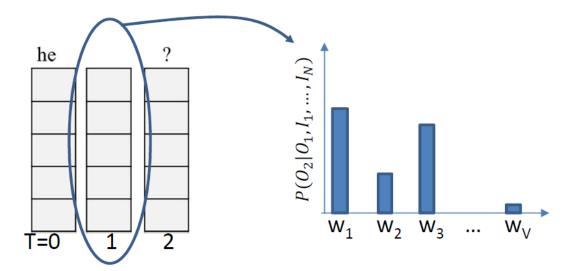
Why greedy is not good
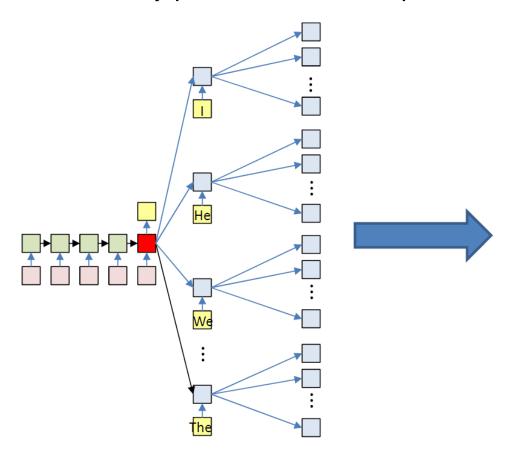
- **Example from speech recognition**

What should we have chosen at t=2??

Will selecting "nose" continue to have a bad effect into the distant future?

he   ?   T=0   1   2

$P(O_2|O_1, I_1, \cdots, I_N)$

$w_1 \quad w_2 \quad w_3 \quad \cdots \quad w_V$

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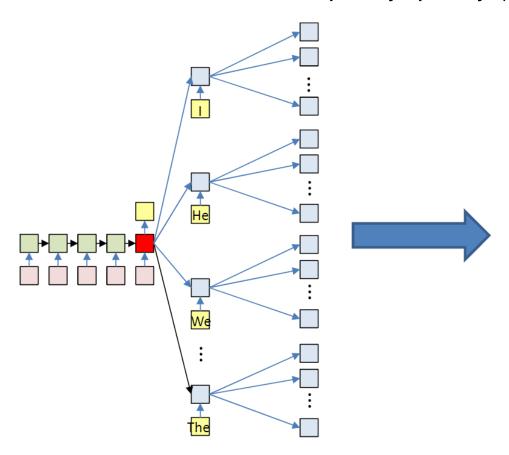
- **Retain all choices and fork the state network**
  - □ With every possible word as input

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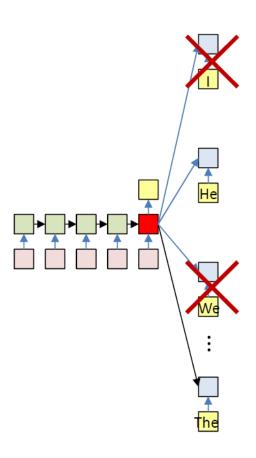
- Retain all choices and fork the state network
  - However, this will blow up very quickly (exponential growth).
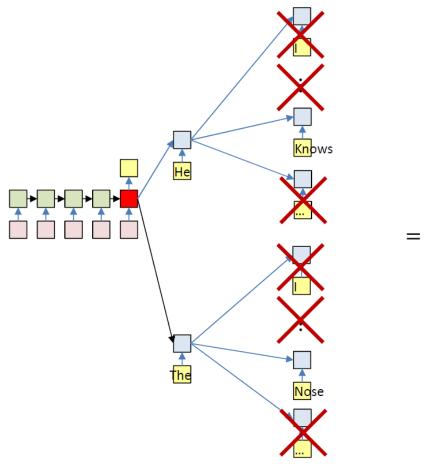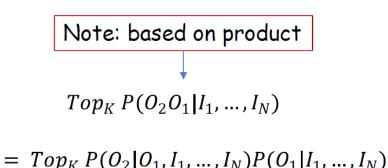
Solution: generate and prune

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

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

Solution: generate and prune

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

Note: based on product

$$Top_K \, P(O_2O_1|I_1, \ldots, I_N)$$

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

Solution: generate and prune
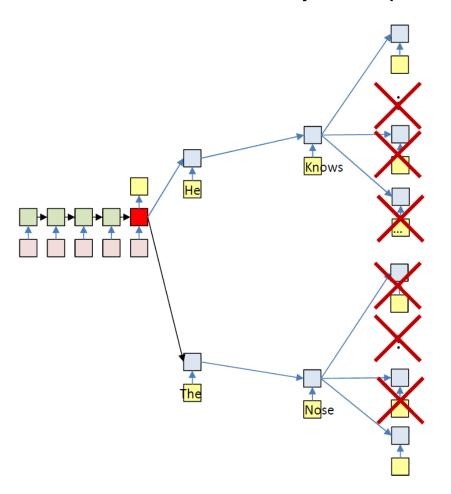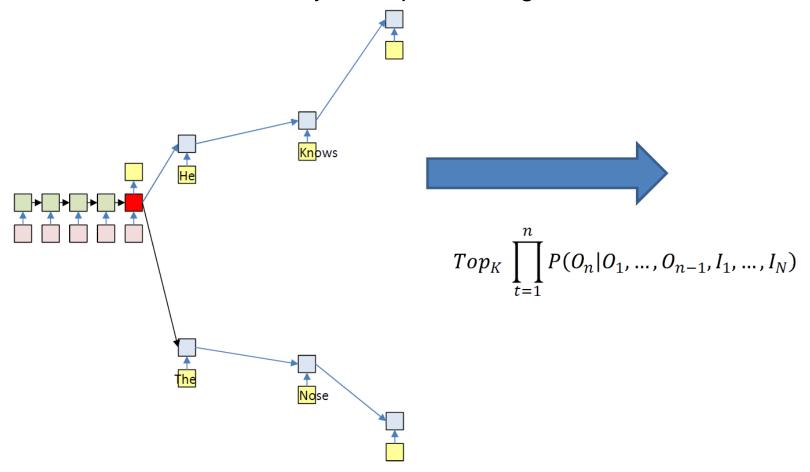
- Prune the state space
  - At each time, retain only the top *K* scoring forks

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

Solution: generate and prune

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

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

Solution: generate and prune
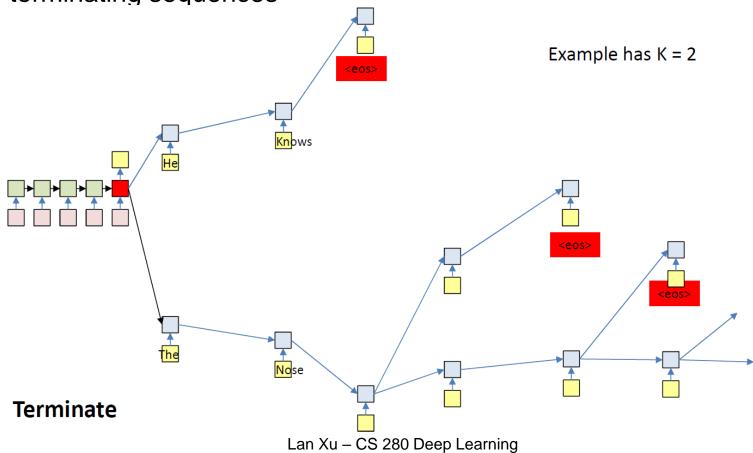
- **Termination**
  - Paths cannot continue once the output an <eos>
  - Select the most likely sequence ending in <eos> across all terminating sequences

Example has K = 2

Terminate

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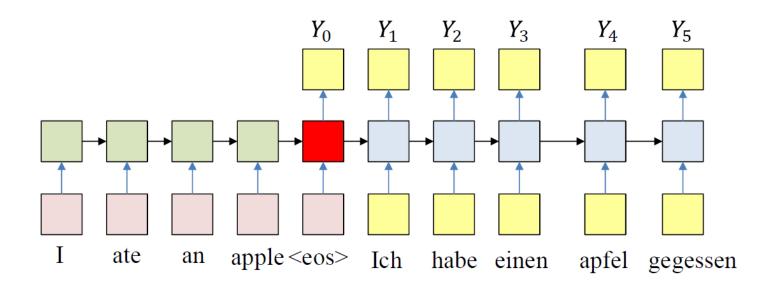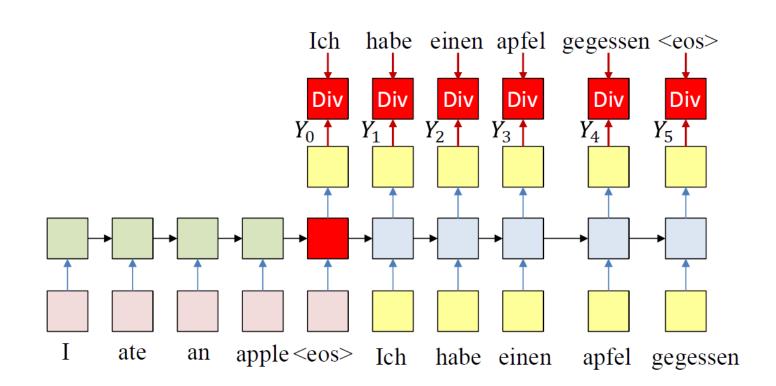
- Forward pass
  - Input the source and target sequences sequentially
  - Output will be a probability distribution over target symbol set
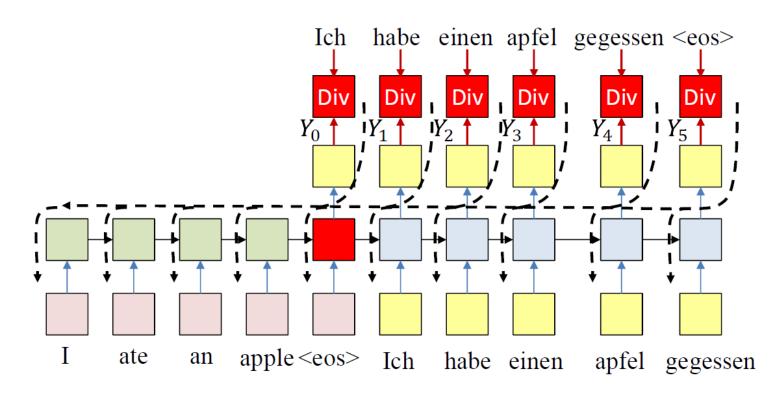
Training the seq-to-seq model

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

Training the seq-to-seq model
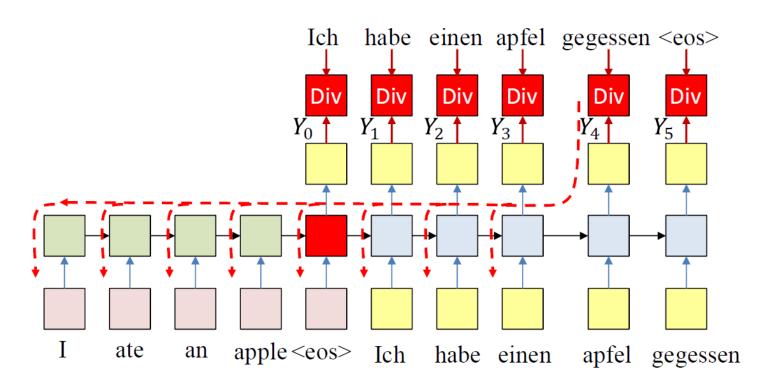
- **Backward pass**
  - ☐ Compute the loss/divergence between the output distribution and target word sequence
  - ☐ Backpropagate the derivatives of the divergence through the net

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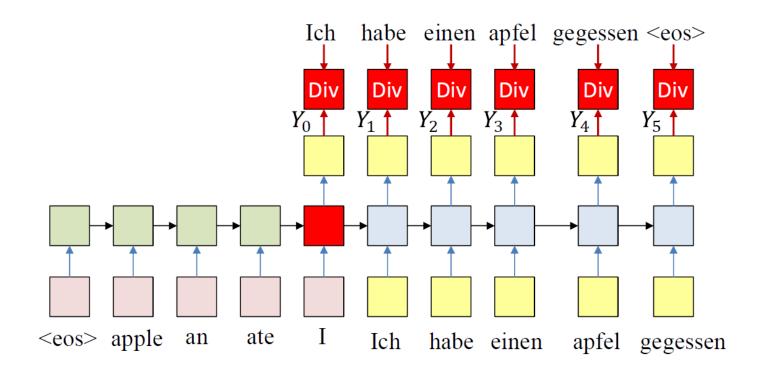
- **Backward pass**
  - In practice, we may randomly sample words from the output to actually use for the backprop update

Training the seq-to-seq model
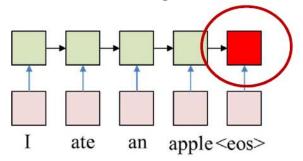
- **Trick of the trade**
  - ☐ The input sequence is fed in reverse order
  - ☐ For both training and during actual decode
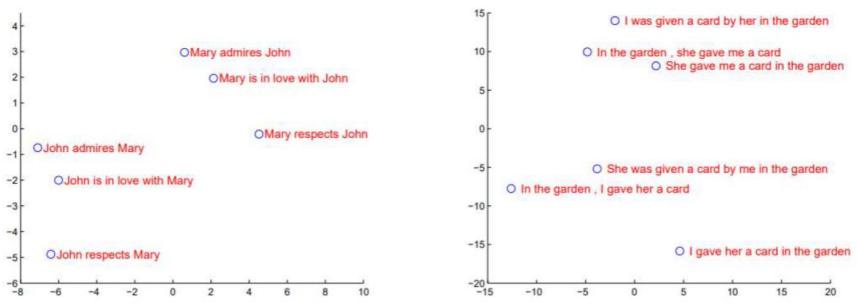
Machine translation example

- Hidden representation encodes semantics
  - Clustered by semantic meaning

I   ate   an   apple <eos>

"Sequence-to-sequence learning with neural networks",

Sutskever, Vinyals and Le   **34**

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 ?*
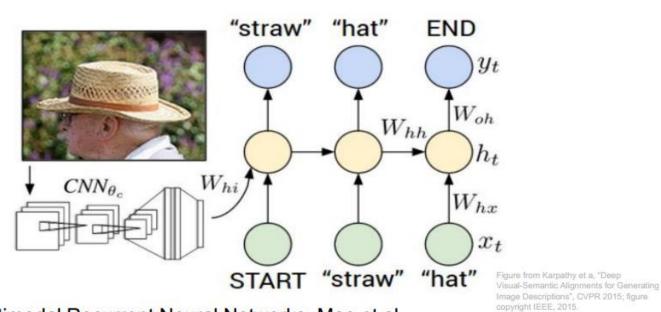
Human: *i am not sure i know that*

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
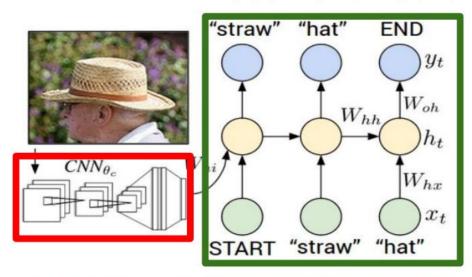
Image-to-sequence problem

- 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
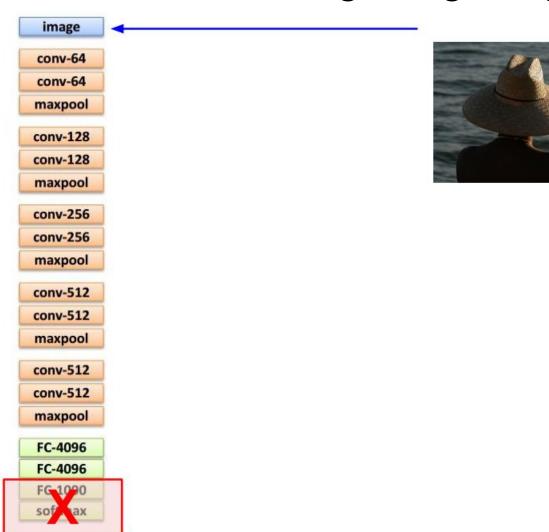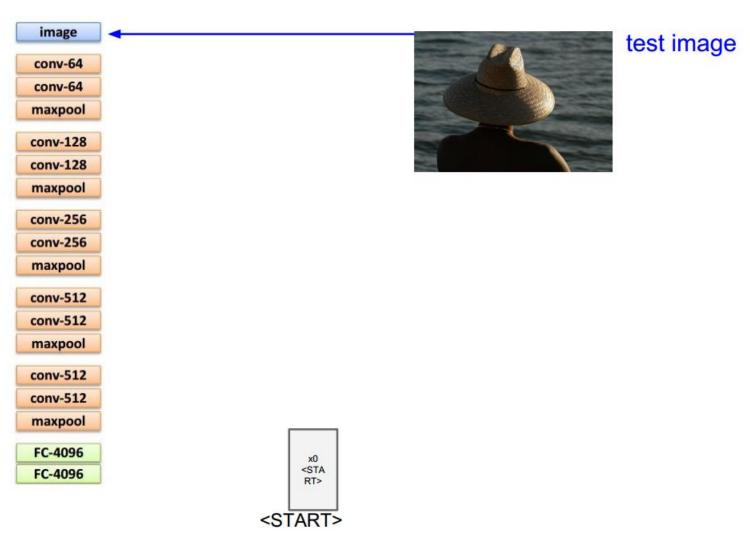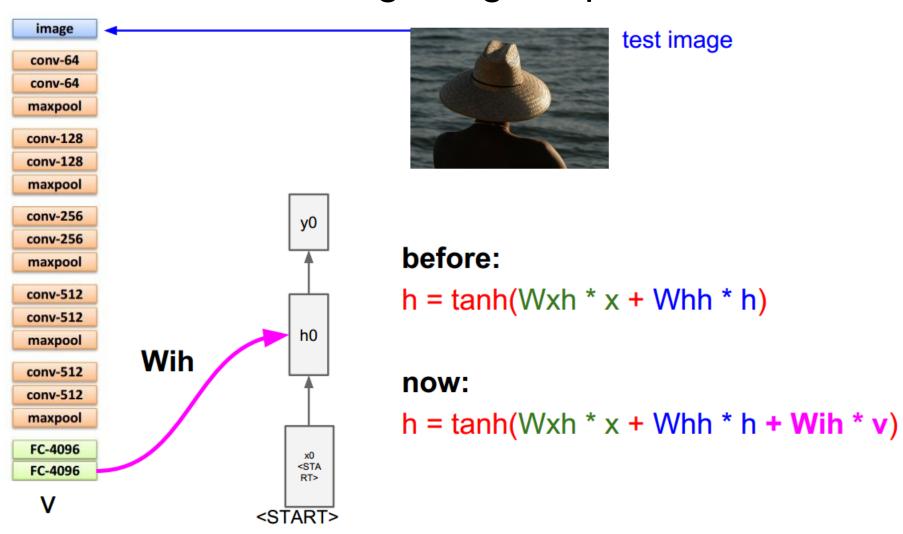
test image

This image is CC0 public domain

Generating image captions

image

| conv-64 |
| conv-64 |
| maxpool |

| conv-128 |
| conv-128 |
| maxpool |

| conv-256 |
| conv-256 |
| maxpool |

| conv-512 |
| conv-512 |
| maxpool |

| conv-512 |
| conv-512 |
| maxpool |

| FC-4096 |
| FC-4096 |
| FC-1000 |
| softmax |

test image

Generating image captions

test image

image

| conv-64 |
| conv-64 |
| maxpool |

| conv-128 |
| conv-128 |
| maxpool |

| conv-256 |
| conv-256 |
| maxpool |

| conv-512 |
| conv-512 |
| maxpool |

| conv-512 |
| conv-512 |
| maxpool |

| FC-4096 |
| FC-4096 |

x0
<START>

<START>

Generating image captions

image
conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096

V

Wih

test image

y0

h0

x0
<STA
RT>

<START>

before:

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

now:

h = tanh(Wxh * x + Whh * h + Wih * v)

Generating image captions

Lan Xu – CS 280 Deep Learning

Generating image captions

Training the image-to-seq model

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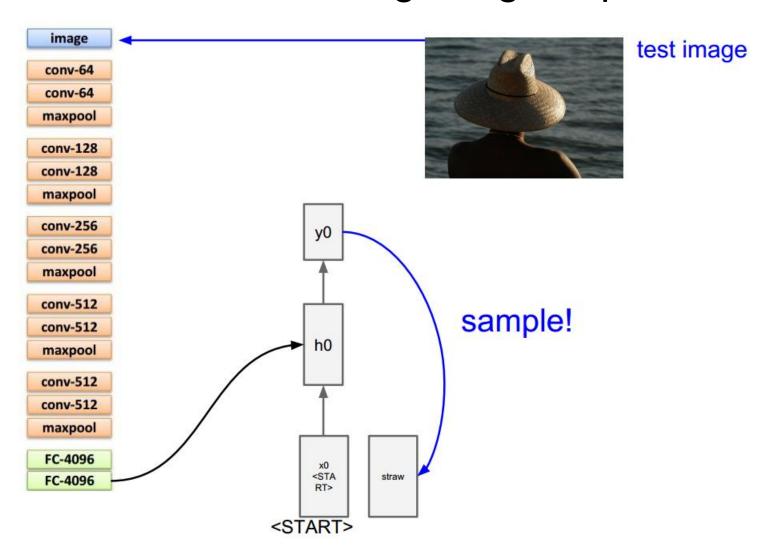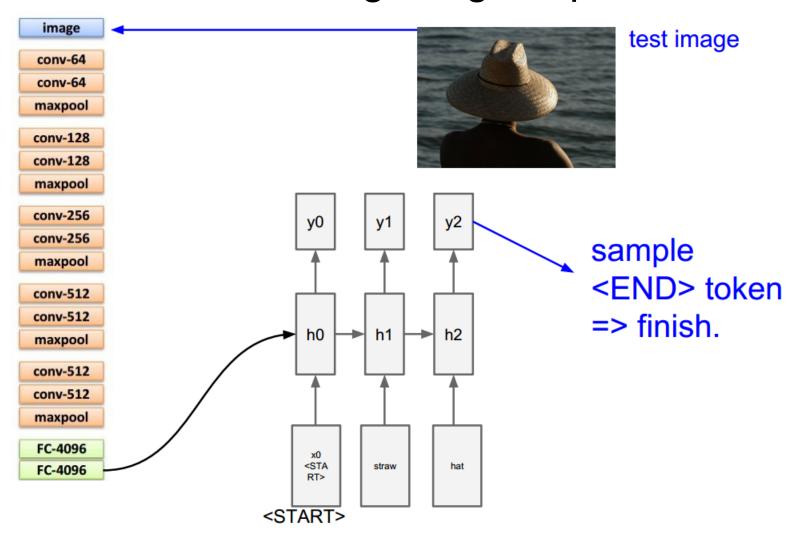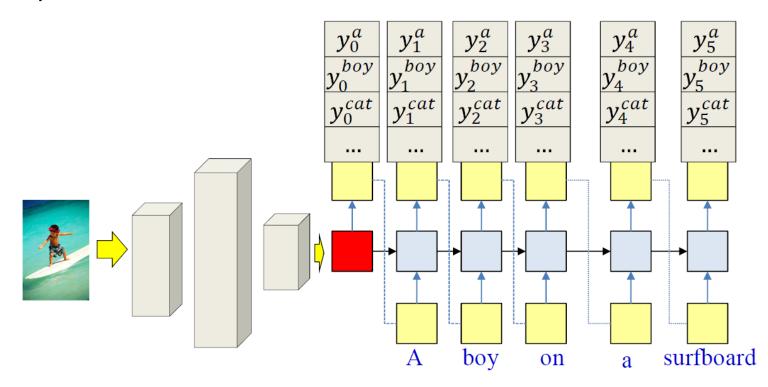 - The image network is pretrained on a large corpus, e.g., ImageNet
  - Forward pass: produce output distribution given the image and caption

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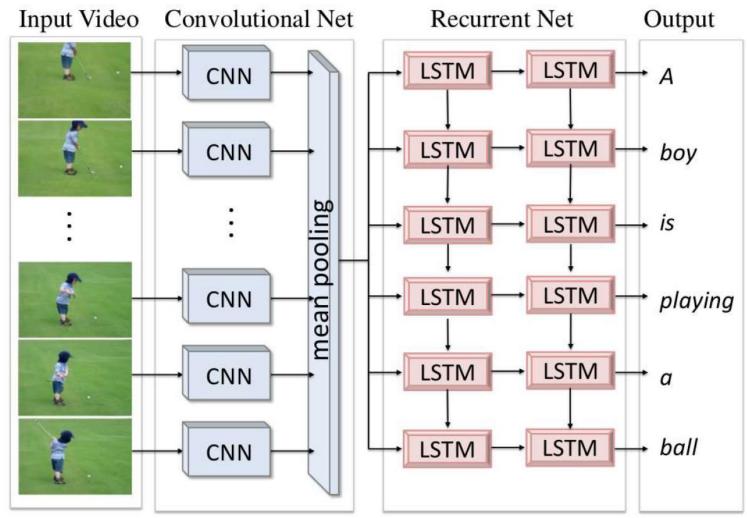
"a woman holding a teddy bear in front of a mirror."

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

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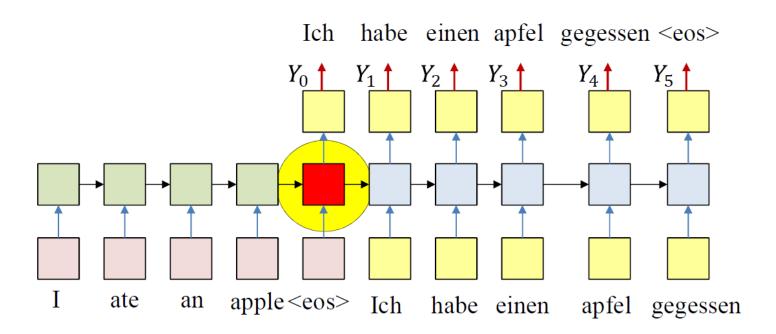
Translating Videos to Natural Language Using Deep Recurrent Neural Networks
Subhashini Venugopalan, Huijun Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, Kate Saenko

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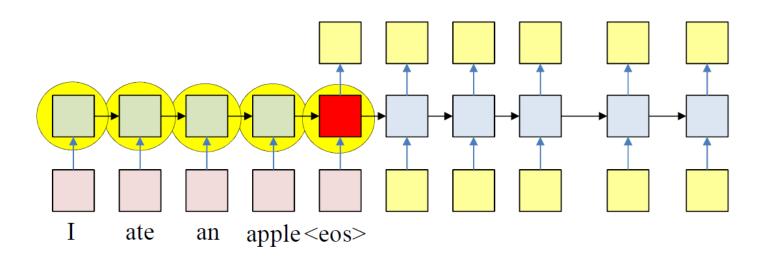
- All the information on the input sequence is embedded into a single vector
  - The latent layer at the end of the input sequence
  - This layer is overloaded with information
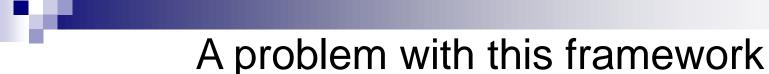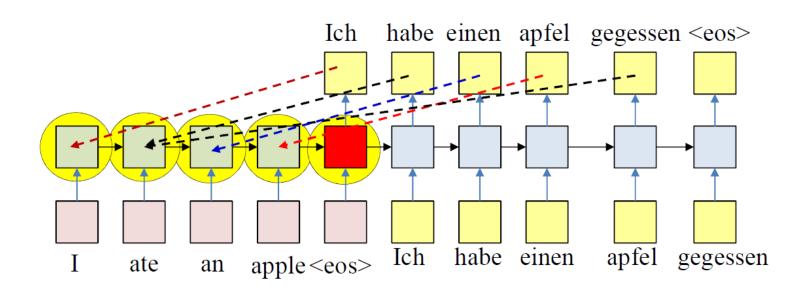
A problem with this framework

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

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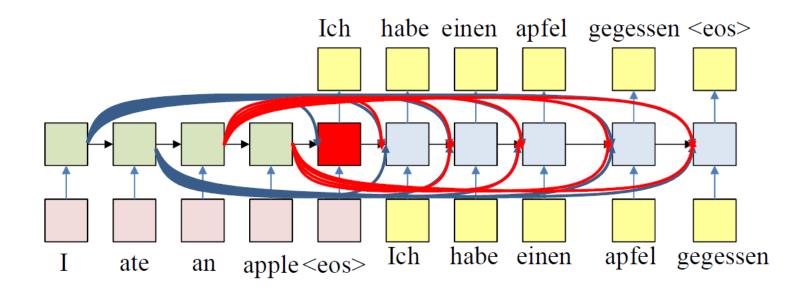
- All latent values carry information
  - Some of which may be diluted downstream
  - Different outputs are related to different inputs

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
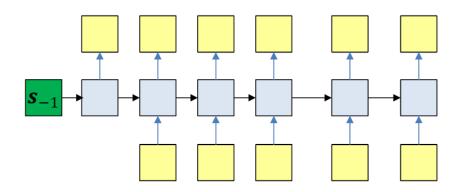
Outline
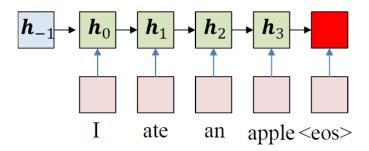
- **RNNs in Vision and NLP**

  - ☐ X-to-sequence model: NMT and Image Captioning

  - ☐ Attention models: NMT and Image Captioning

Acknowledgement: Feifei Li et al's cs231n notes

Attention models
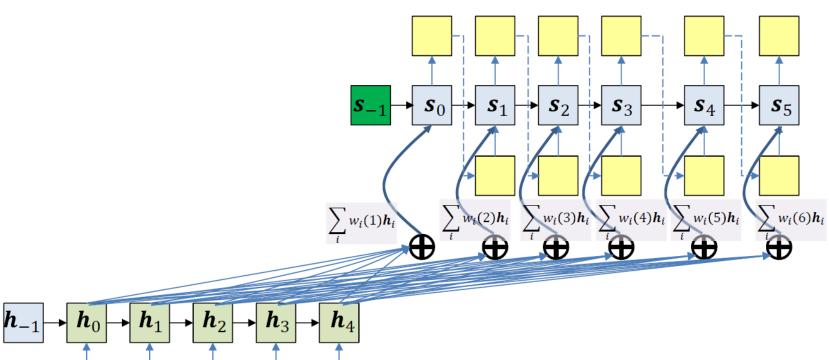
- Separating the encoder and decoder first
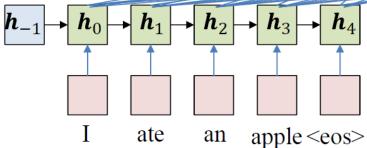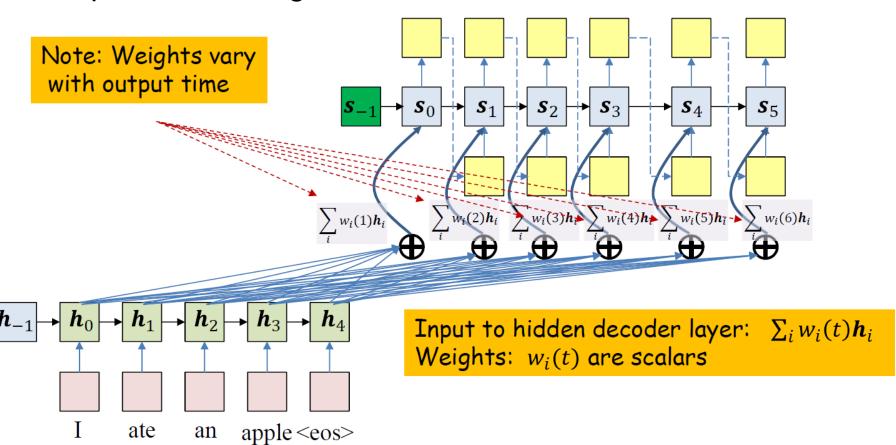
Lan Xu – CS 280 Deep Learning

Attention models

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

Lan Xu – CS 280 Deep Learning

Attention models

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

Note: Weights vary with output time

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

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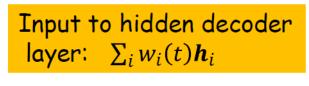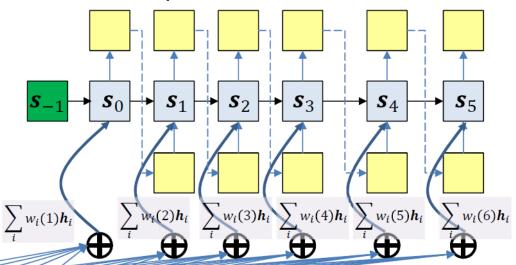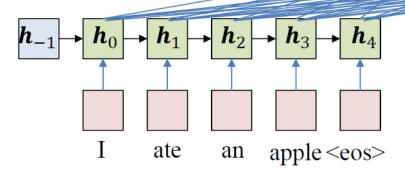
- Require a time-varying weight that specifies relationship of output time to input time
  - Weights are functions of current output state

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

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

I   ate   an   apple <eos>

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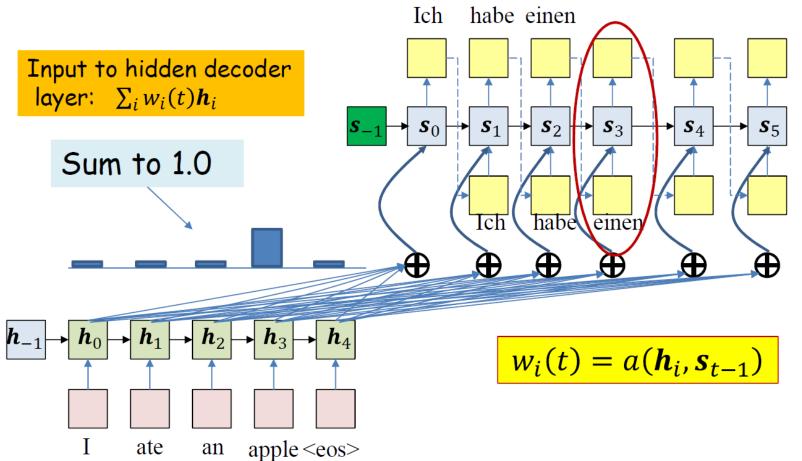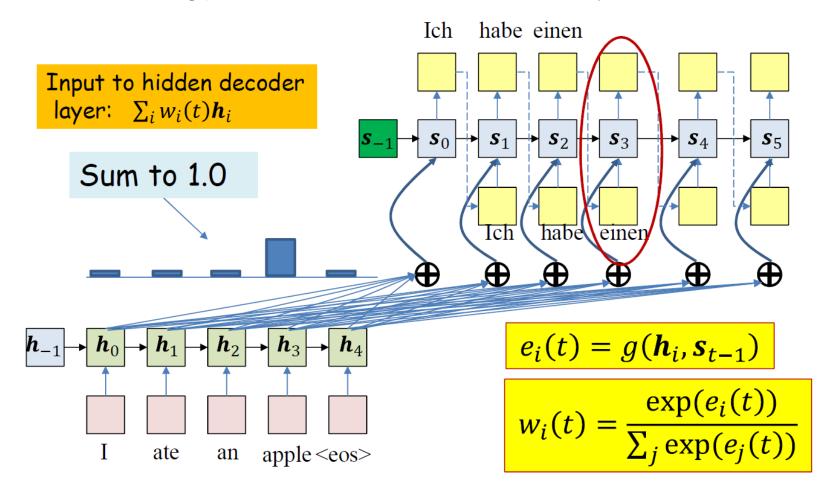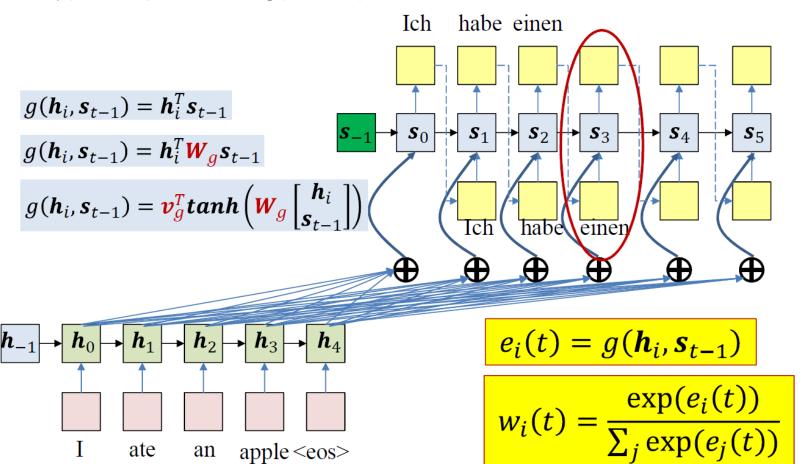
- The weights are a distribution over the input
  - Automatically highlight the most important input components for any output

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

Sum to 1.0

Ich   habe   einen

$\boldsymbol{s}_{-1}$ → $\boldsymbol{s}_0$ → $\boldsymbol{s}_1$ → $\boldsymbol{s}_2$ → $\boldsymbol{s}_3$ → $\boldsymbol{s}_4$ → $\boldsymbol{s}_5$

Ich   habe   einen

$\boldsymbol{h}_{-1}$ → $\boldsymbol{h}_0$ → $\boldsymbol{h}_1$ → $\boldsymbol{h}_2$ → $\boldsymbol{h}_3$ → $\boldsymbol{h}_4$

I   ate   an   apple <eos>

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

Attention models

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

Ich   habe   einen

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

Sum to 1.0

s_{-1} s_0 s_1 s_2 s_3 s_4 s_5

Ich   habe   einen

h_{-1} h_0 h_1 h_2 h_3 h_4

I   ate   an   apple <eos>

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

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

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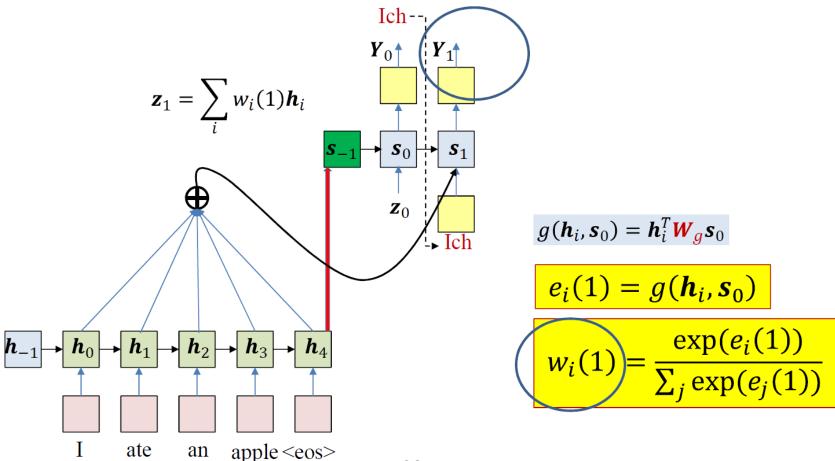
- The weights are a distribution over the input
  - Typical options for g() with parameters to be learned

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

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

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

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

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

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

$$z_1 = \sum_i w_i(1)h_i$$

$$g(h_i, s_0) = h_i^T W_g s_0$$
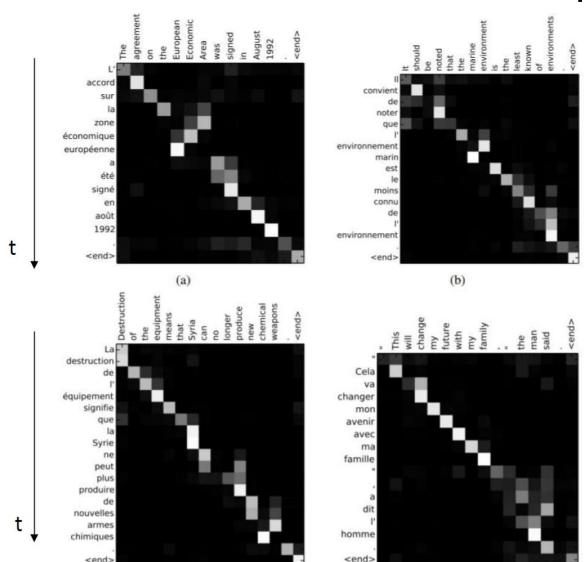
$$e_i(1) = g(h_i, s_0)$$

$$w_i(1) = \frac{\exp(e_i(1))}{\sum_j \exp(e_j(1))}$$
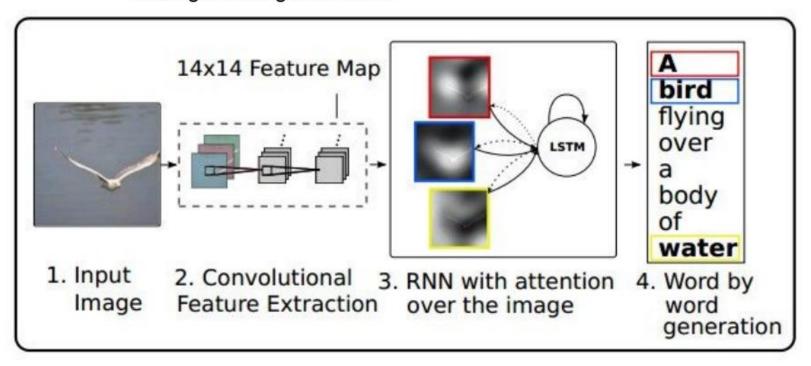
Attention example

(a)

(b)

Plot of $w_i(t)$
Color shows value (white is larger)

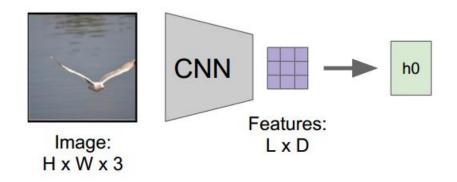Note how most important input words for any output word get automatically highlighted

The general trend is somewhat linear because word order is roughly similar in both languages
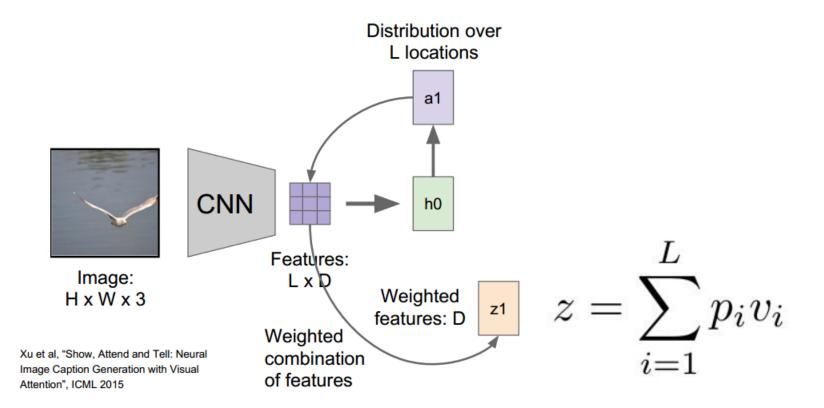
Image Captioning with Attention

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

14x14 Feature Map

1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

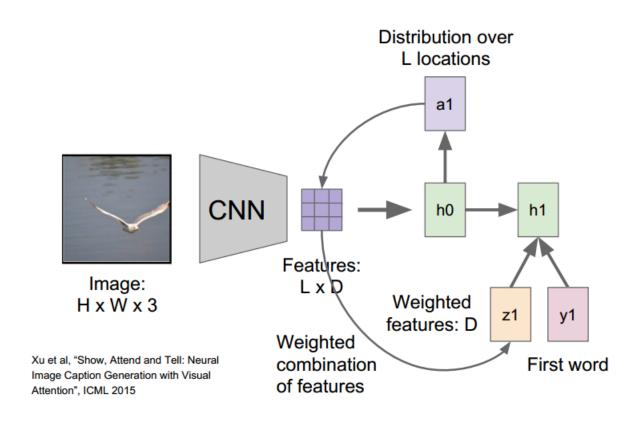A bird flying over a body of water

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015
Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Benchio, 2015. Reproduced with permission.

Image Captioning with Attention

Image:
H x W x 3

CNN

Features:
L x D

h0

Image Captioning with Attention

Distribution over
L locations

a1

CNN

h0

Image:
H x W x 3

Features:
L x D

Weighted
features: D

z1

Weighted
combination
of features

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

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

Image Captioning with Attention

Distribution over L locations

a1

CNN

Image: H x W x 3

Features: L x D

Weighted combination of features

Weighted features: D

h0 → h1

z1   y1

First word

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

Image Captioning with Attention

Distribution over L locations

Distribution over vocab

Features: L x D

Weighted features: D

Weighted combination of features

First word

Image: H x W x 3

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

Image Captioning with Attention

Image Captioning with Attention

A woman is throwing a <u>frisbee</u> in a park.

A <u>dog</u> is standing on a hardwood floor.

A <u>stop</u> sign is on a road with a mountain in the background.

A little <u>girl</u> sitting on a bed with a teddy bear.

A group of <u>people</u> sitting on a boat in the water.

A giraffe standing in a forest with <u>trees</u> in the background.

Summary

- ## RNN
  - ☐ X-to-Sequence model
  - ☐ Attention model
- ## Next time:
  - ☐ Generative models
  - ☐ Deep unsupervised learning