Introduction to Machine Learning CS182

Lu Sun

School of Information Science and Technology

ShanghaiTech University

December 21, 2023

Today:

- Recurrent Neural Networks (RNN)

Readings:

- Deep Learning (DL), Chapter 10

Today's Agenda

Recurrent Neural Network (RNN)

- Motivation
- Basic RNN
- Training RNN and LSTM
- Applications in Vision and NLP
- Attention Models

Acknowledgement: Hugo Larochelle's, MehryarMohri@NYU's, YingyuLiang@Princeton's, BhikshaRaj@CMU's&FeifeiLi@Stanford'scourse notes

Recurrent Neural Networks-RNN

- **Motivation**
- Basic RNN
- Training RNN and LSTM
- Applications in Vision and NLP
- Attention Models

Motivation: Sequence Modeling

- Modeling a sequence of tokens
  - Running example: sentences, but it can also be gene or protein sequences, etc.
- Goal: learn/build a good distribution of sentences

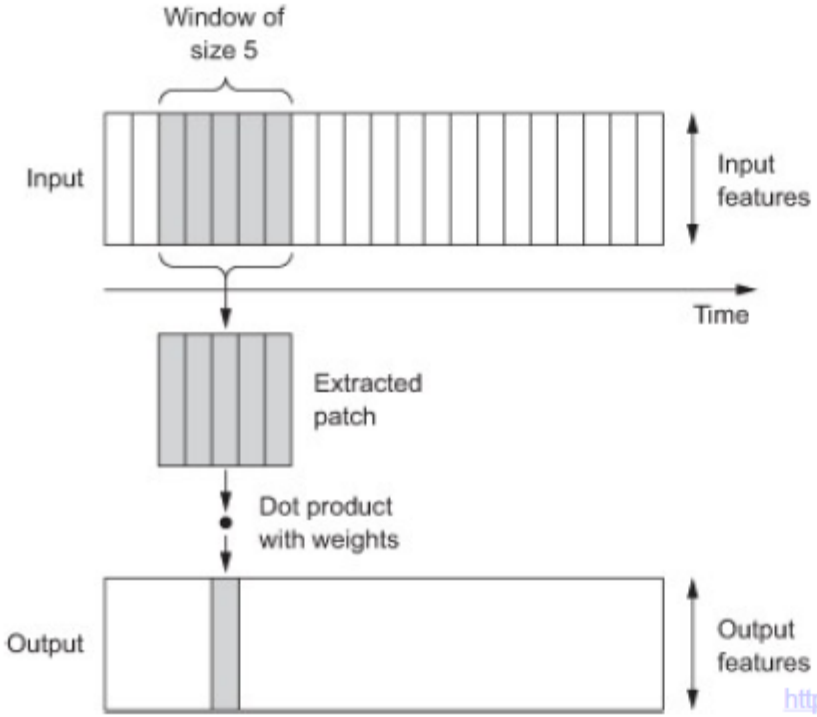
- Inputs: a corpus of sentences $\mathbf{s}^{(1)}, \cdots, \mathbf{s}^{(N)}$
- Output: a distribution $p(\mathbf{s})$

- Common approach: maximum likelihood
  - Assume sentences are independent

$$\max \prod_{i=1}^{N} p(\mathbf{s}^{(i)})$$

Motivation: Sequence Modeling

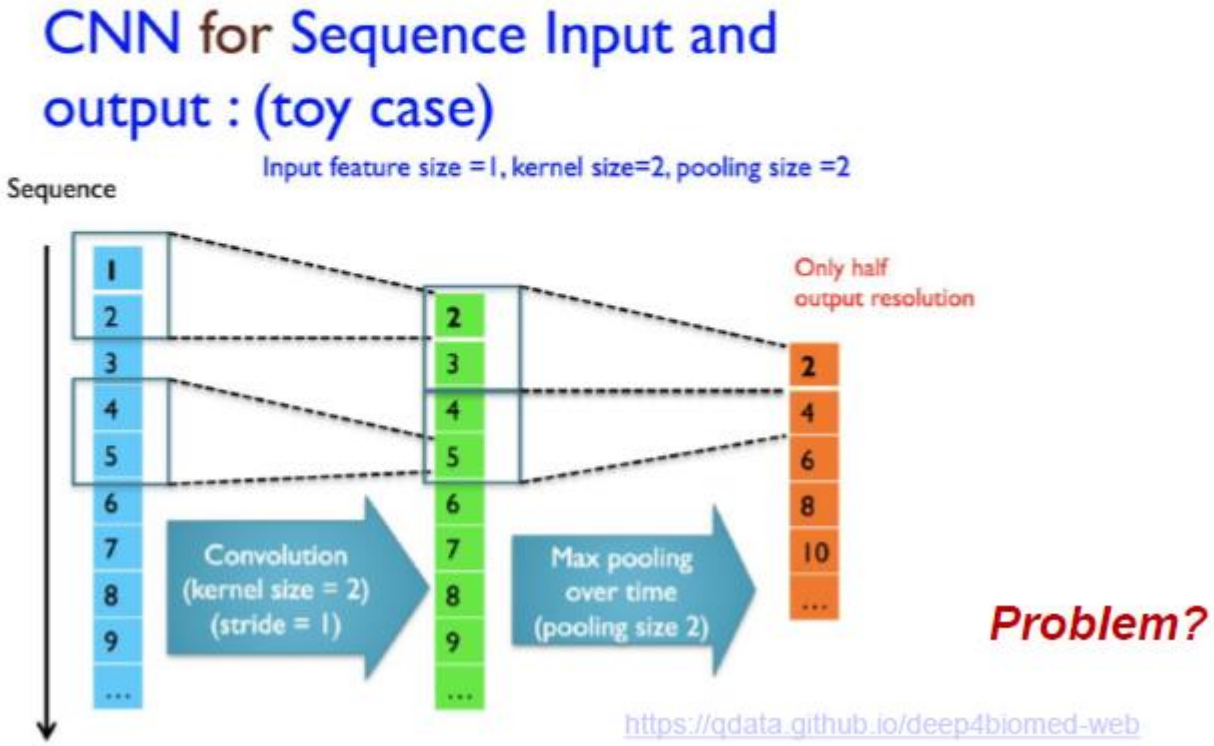
- What is $p(\mathbf{s})$?
- A sentence is a sequence of words $w_1, w_2, \cdots, w_T$.
- Can we use CNN?



https://medium.com/@jon.froiland/convolutional-neural-networks-for-sequence-processing-part-1-420dd9b500

Motivation: Sequence Modeling

- What is $p(\mathbf{s})$?
- A sentence is a sequence of words $w_1, w_2, \cdots, w_T$.
- Can we use CNN?



Motivation: Sequence Modeling

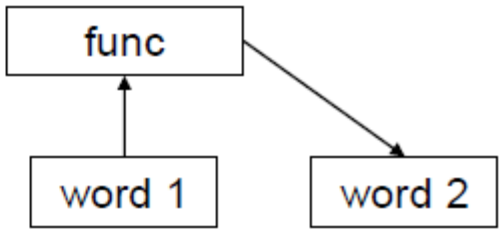
- What is $p(\mathbf{s})$?
- A sentence is a sequence of words $w_1, w_2, \cdots, w_T$.

$$p(\mathbf{s}) = p(w_1, \ldots, w_T) = p(w_1)p(w_2 \mid w_1) \cdots p(w_T \mid w_1, \ldots, w_{T-1}).$$

  - Essentially aim to predict the next word
- Markovian assumption
  - The distribution over the next word depends on the preceding few words. For example,

$$p(w_t \mid w_1, \ldots, w_{t-1}) = p(w_t \mid w_{t-3}, w_{t-2}, w_{t-1}).$$

  - Autoregressive model
    - Memoryless

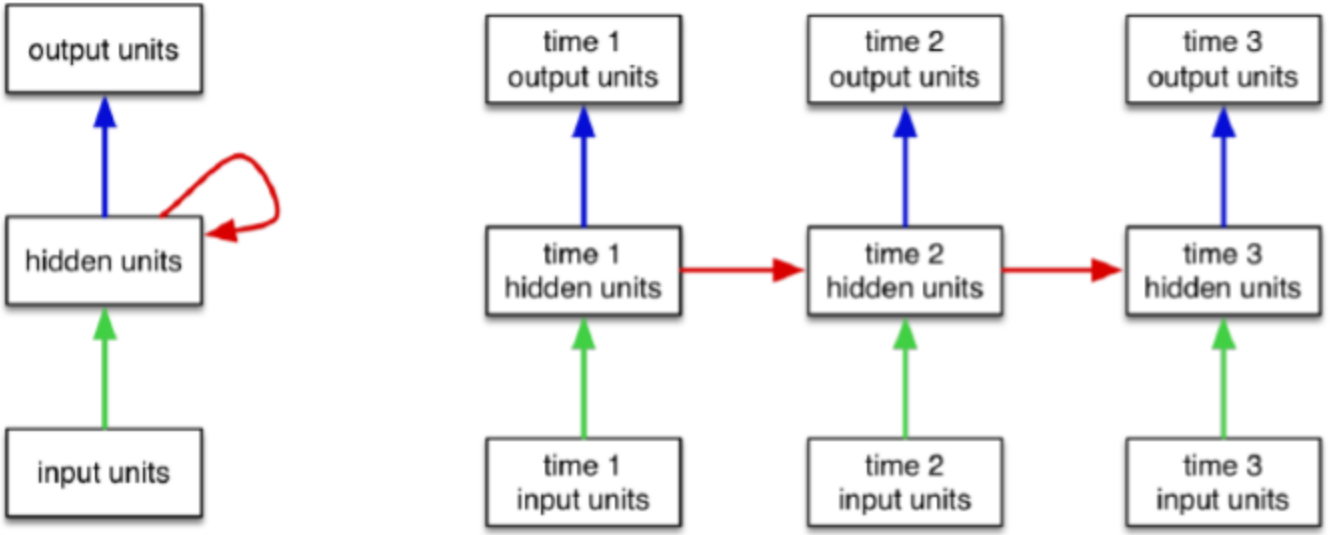


Recurrent Neural Networks-RNN

- Motivation
- **Basic RNN**
- Training RNN and LSTM
- Applications in Vision and NLP
- Attention Models

Recurrent Neural Network

- Recurrent Neural Network as a dynamical system with one set of hidden units feeding into themselves
  - The network's graph has self-loops
- The RNN's graph can be unrolled by explicitly representing the units at all time steps
  - The weights and biases are shared



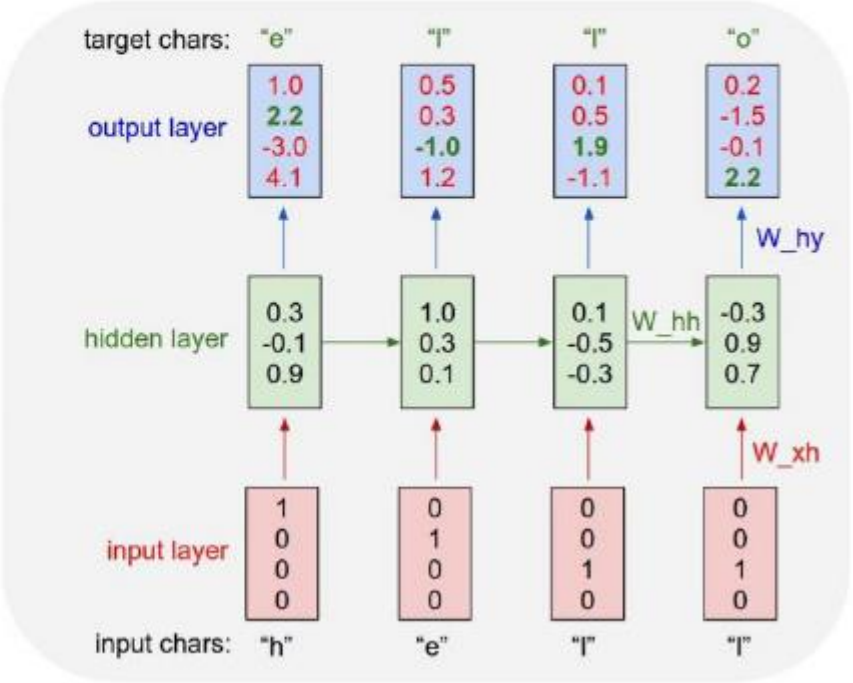
Recurrent Neural Network

- The RNN's graph can be unrolled by explicitly representing the units at all time steps
  - The weights and biases are shared

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **"hello"**



Recurrent Neural Network

- General formulation

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

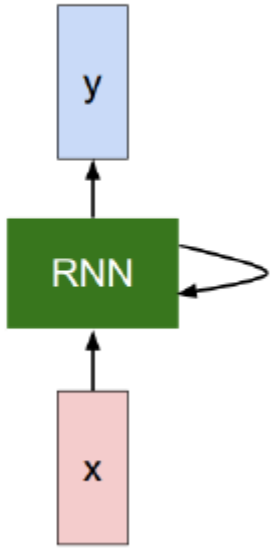
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

input vector at some time step

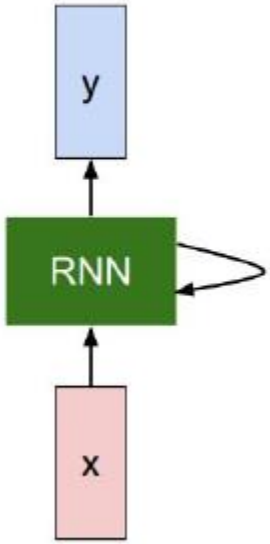


Recurrent Neural Network

- General formulation

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

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

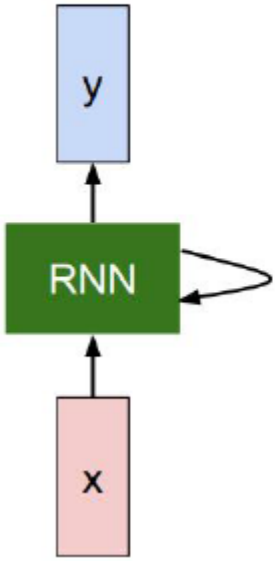


Notice: the same function and the same set of parameters are used at every time step.

(Vanilla) Recurrent Neural Network

- General formulation

The state consists of a single *"hidden"* vector **h**:



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

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

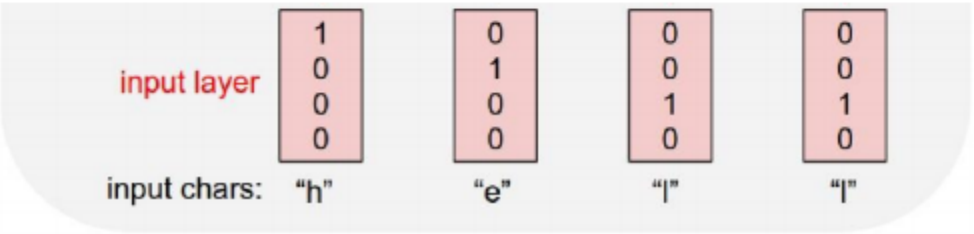
$$y_t = W_{hy}h_t$$

RNN for Language Modeling

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



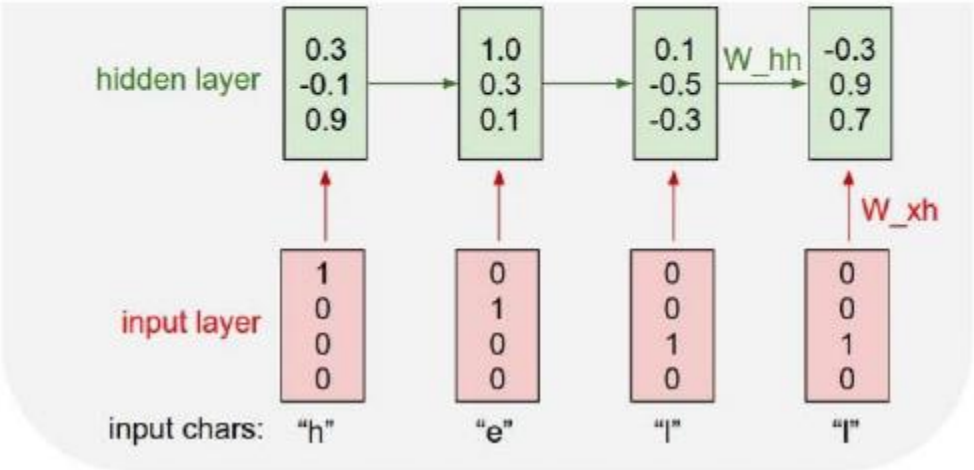
RNN for Language Modeling

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

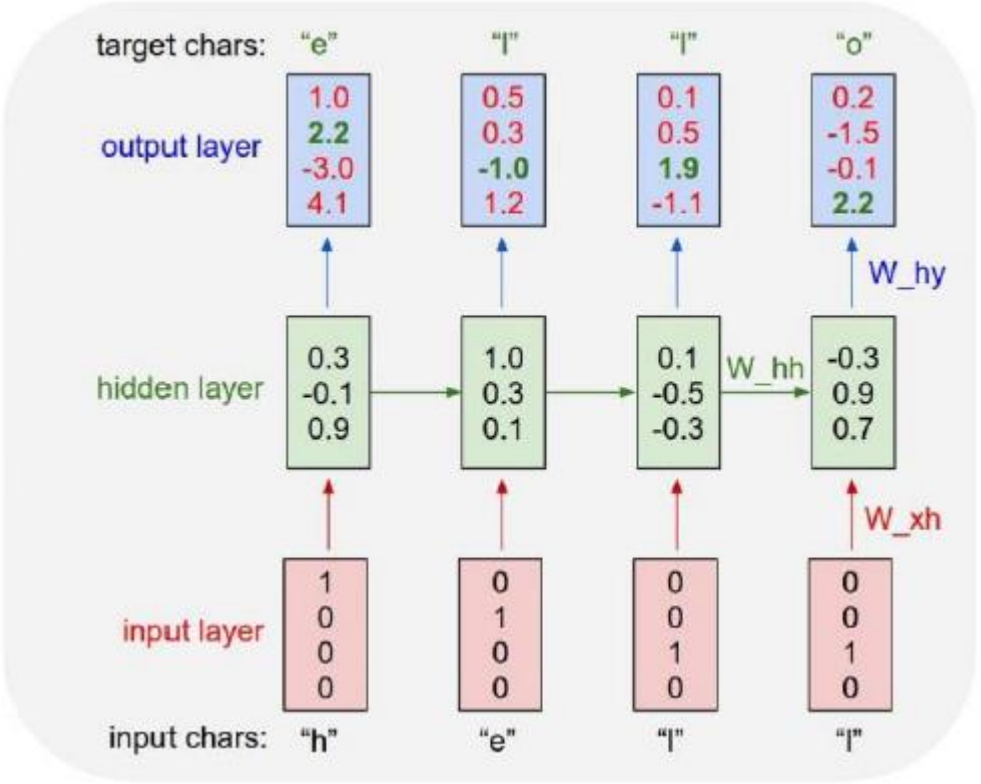


RNN for Language Modeling

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

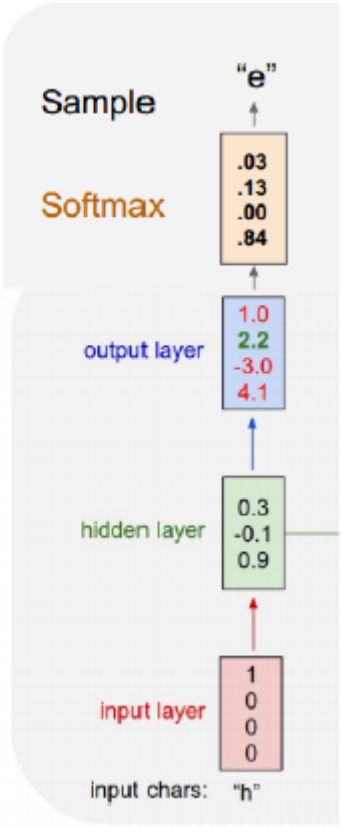


RNN for Language Modeling

Example:
Character-level
Language Model
Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

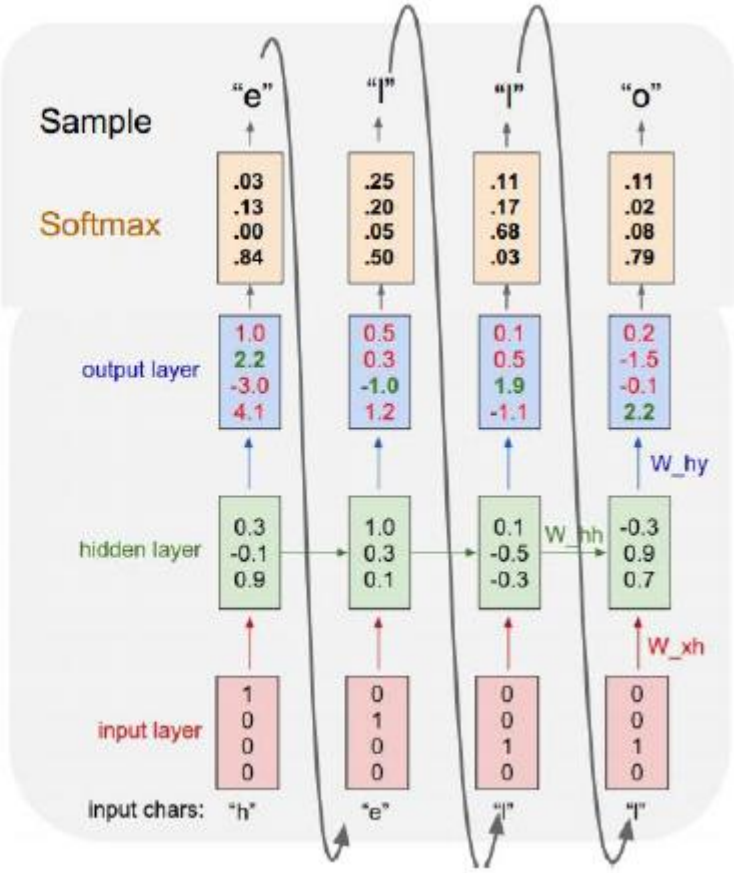


RNN for Language Modeling

Example:
Character-level
Language Model
Sampling

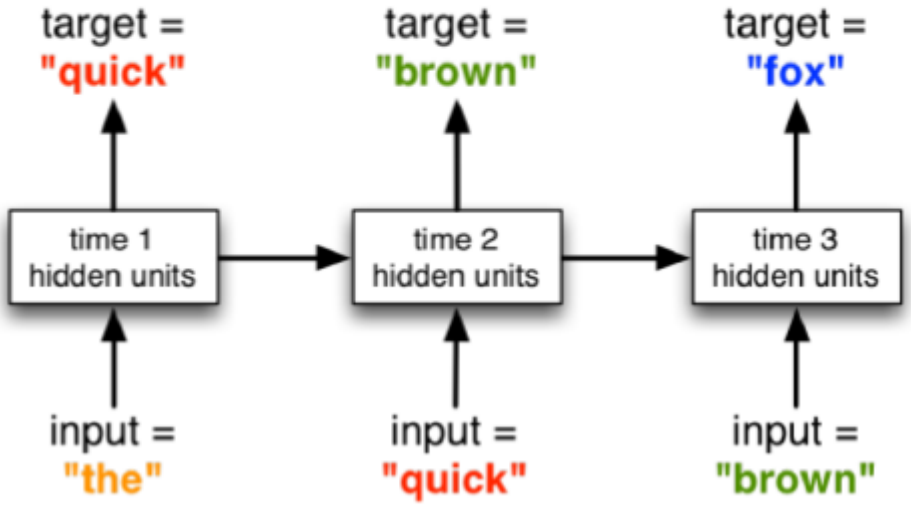
Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



RNN for Language Modeling

- Modeling at word level
  - Each word is represented as an indicator vector
  - The model predicts a distribution over words



RNN for Language Modeling

- Generating from a RNN language model
  - The outputs are fed back to the network

- Training time: the inputs are the token from the training set (teacher forcing).

RNN for Language Modeling

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

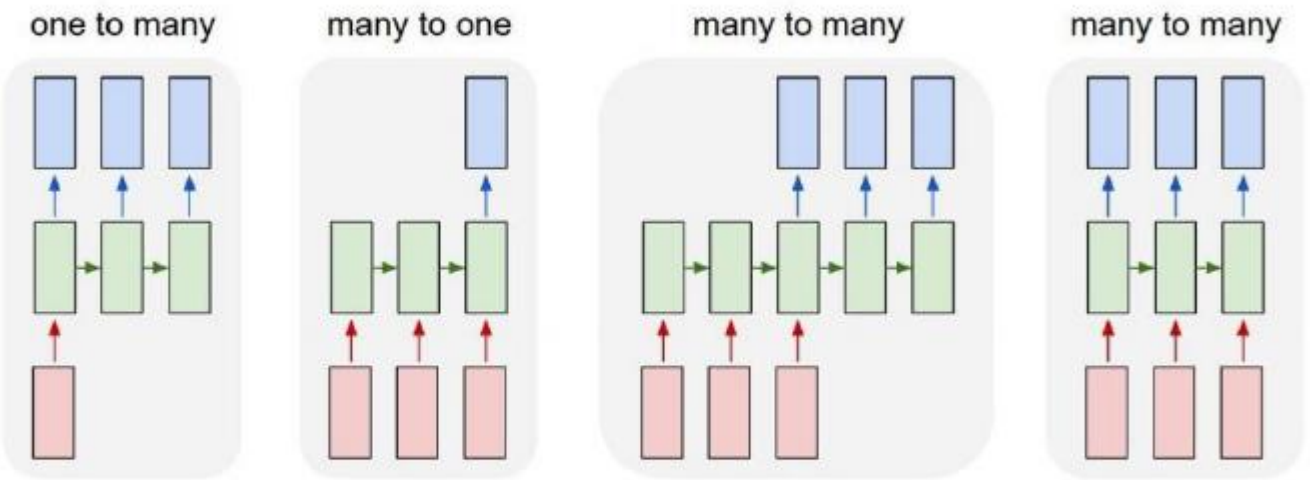
```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

RNN: Model Variants

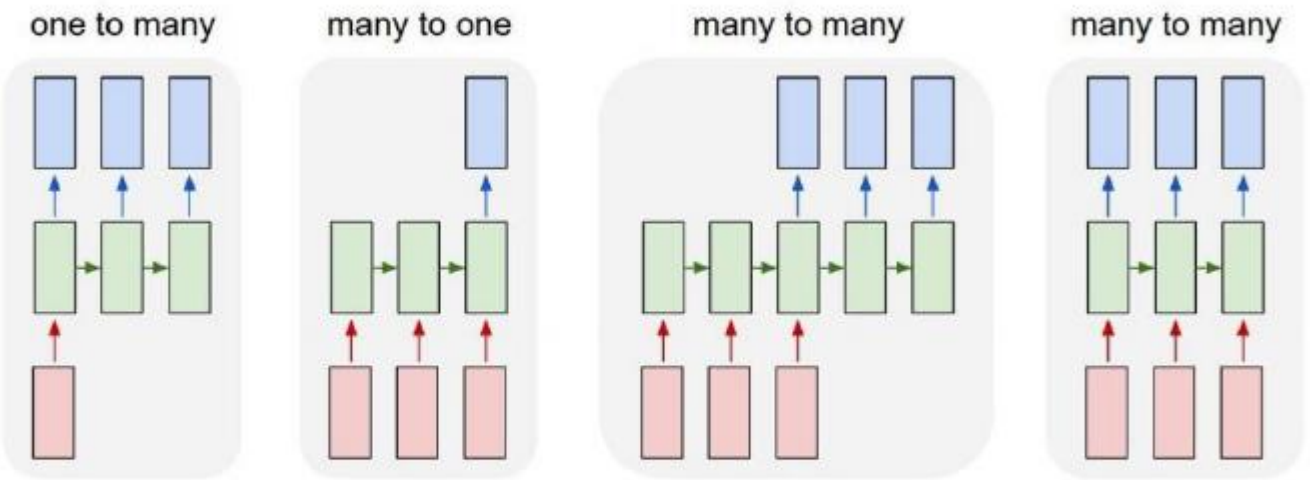
- Recurrent Neural Networks: model variants



e.g. **Image Captioning**
image -> sequence of words

RNN: Model Variants

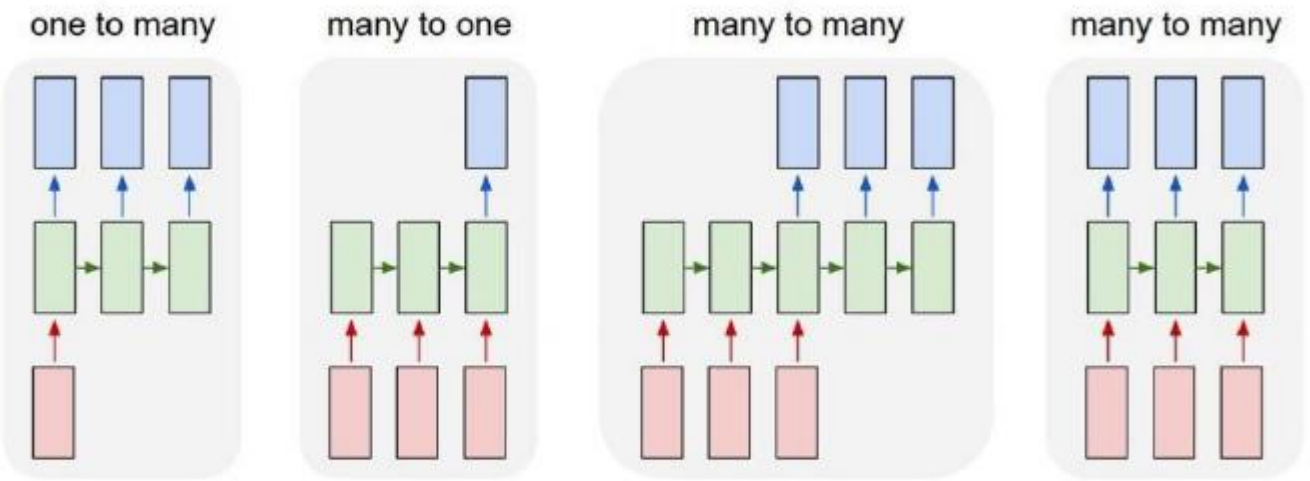
- Recurrent Neural Networks: model variants



e.g. **Sentiment Classification**
sequence of words -> sentiment

RNN: Model Variants

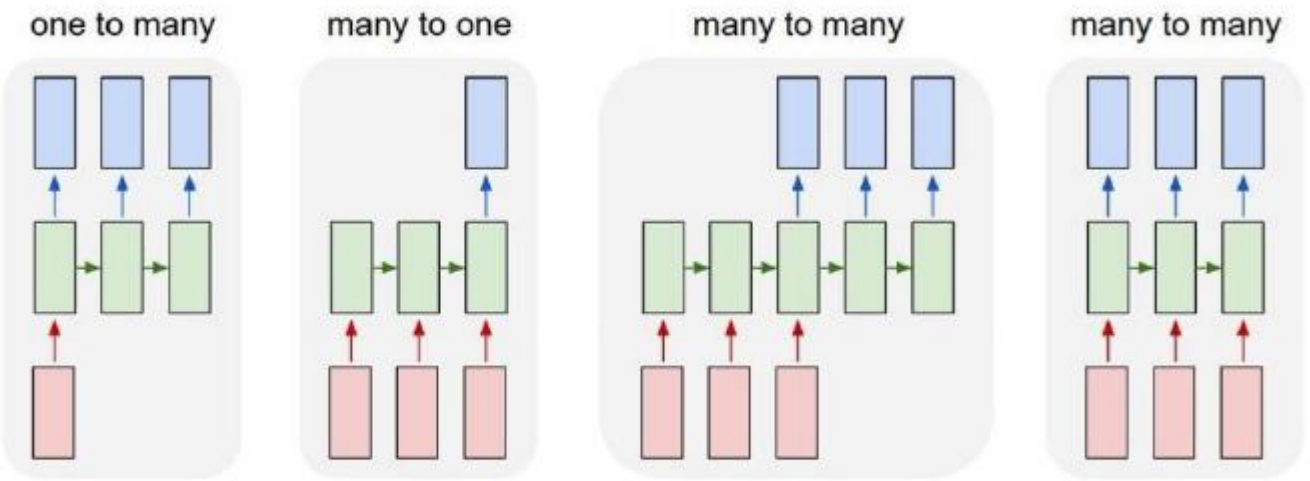
- Recurrent Neural Networks: model variants



e.g. **Machine Translation**
seq of words -> seq of words

RNN: Model Variants

- Recurrent Neural Networks: model variants



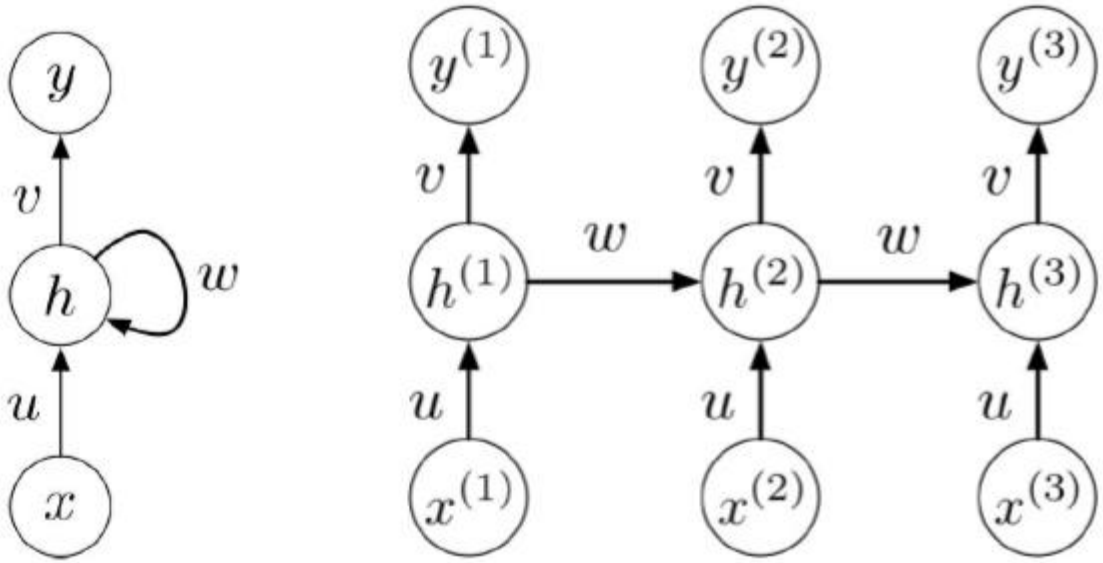
e.g. **Video classification on frame level**

Recurrent Neural Networks-RNN

- Motivation
- Basic RNN
- **Training RNN and LSTM**
- Applications in Vision and NLP
- Attention Models

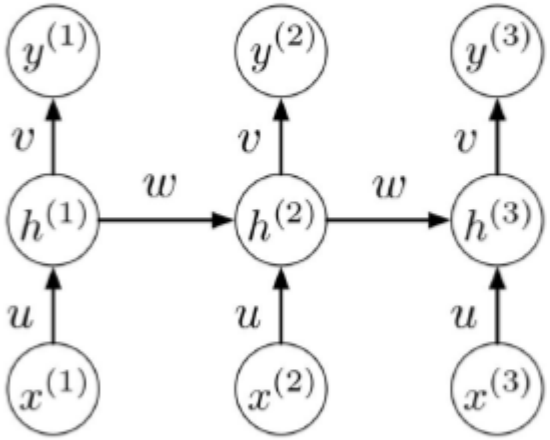
Bachpropagation Through Time (BPTT): Example

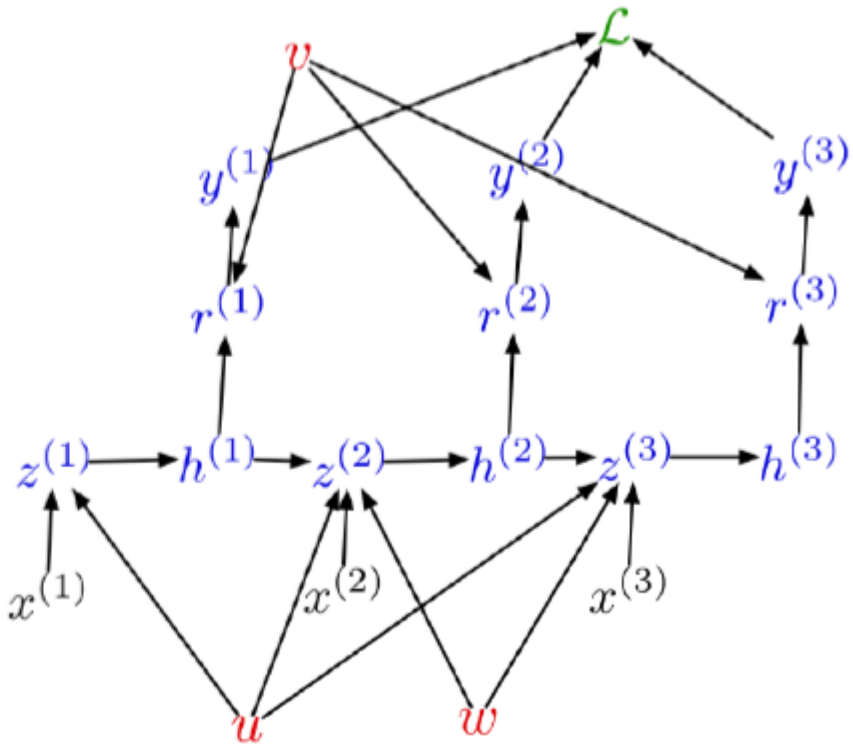
- A simple network
  - Everything is scalar



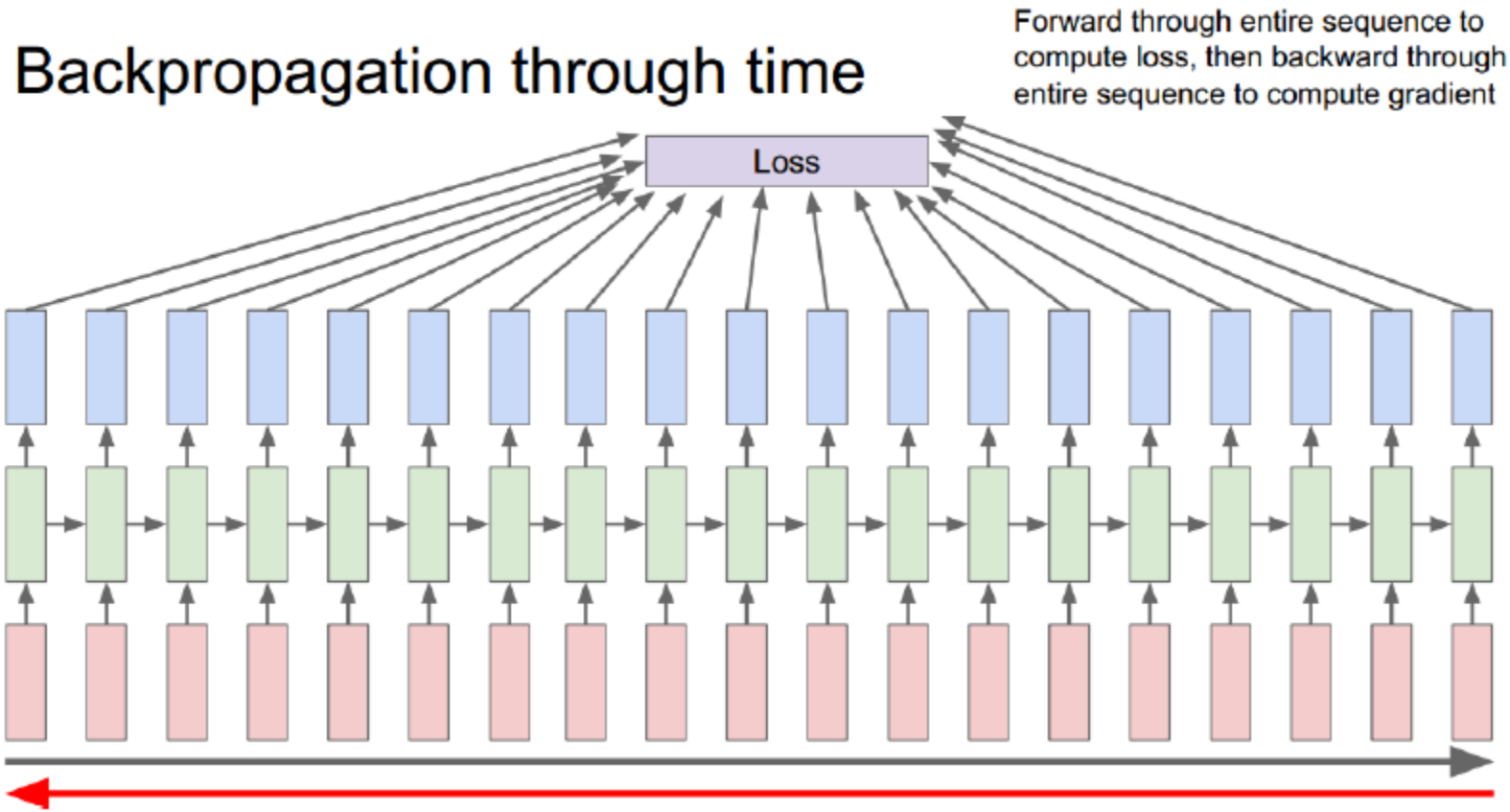
BPTT Example

- A simple network
  - Everything is scalar
  - Unrolled computation graph with shared parameters



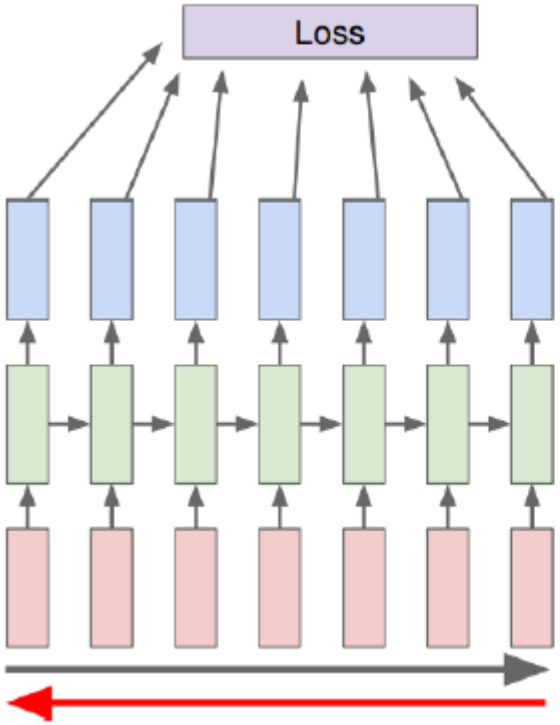


BPTT Example



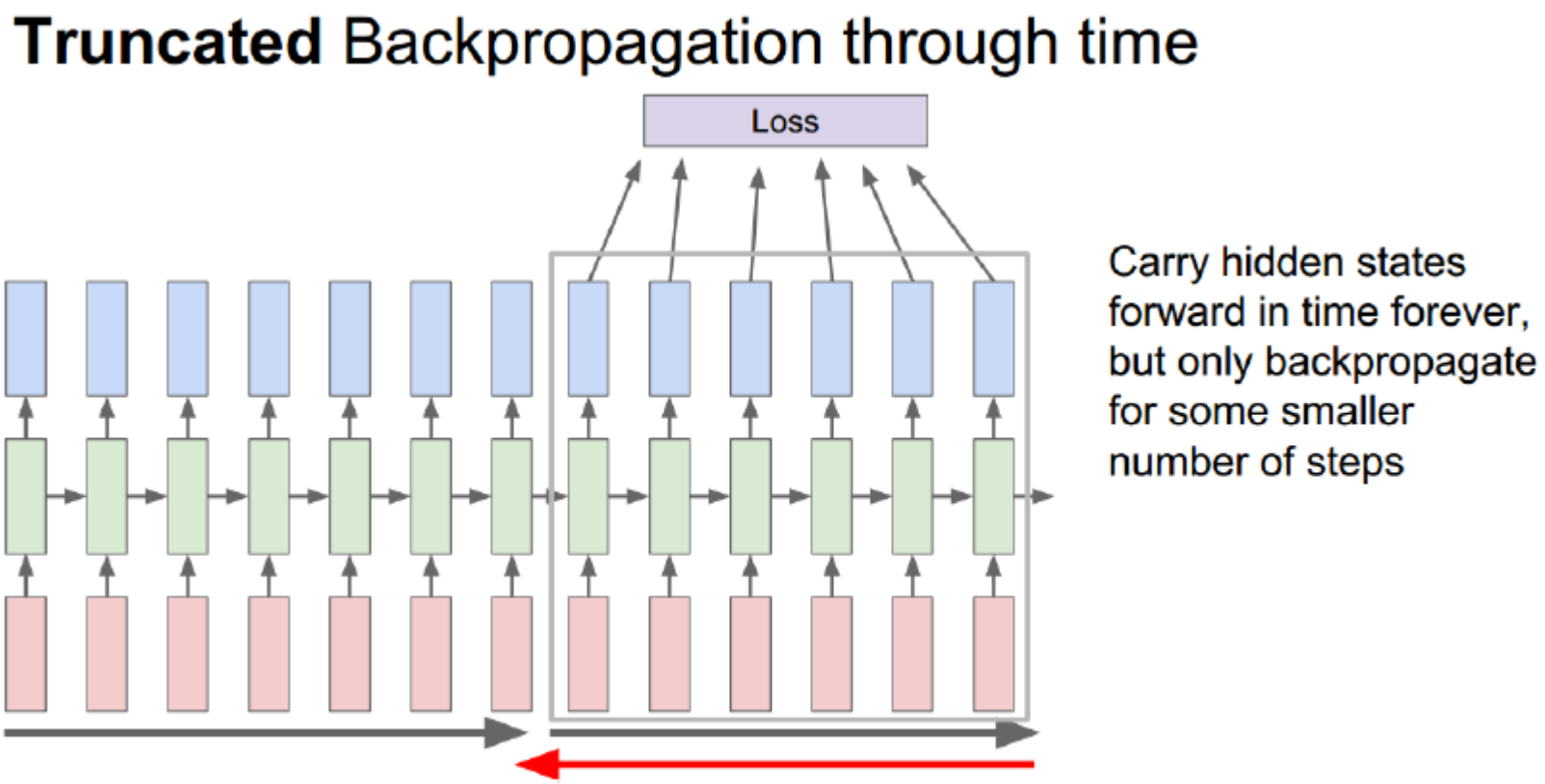
BPTT Example

Truncated Backpropagation through time

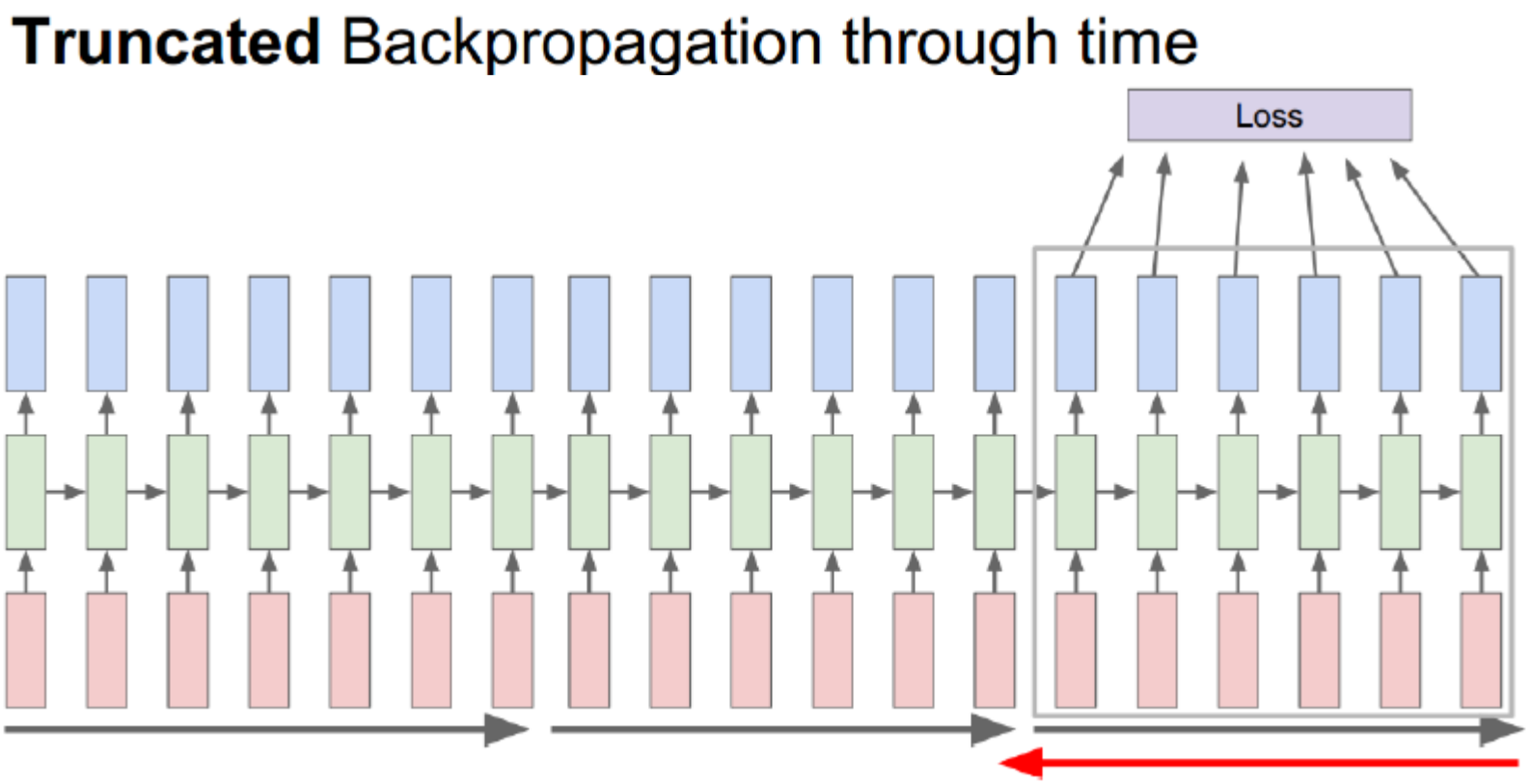


Run forward and backward through chunks of the sequence instead of whole sequence

BPTT Example



BPTT Example

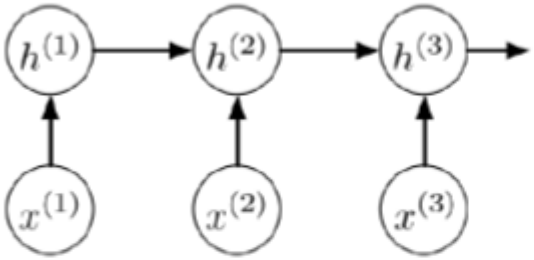


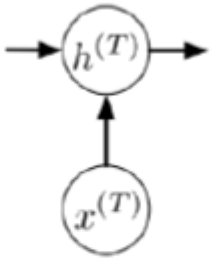
Challenges in Training RNNs

- RNN
  - BP through time is used to compute the gradient descent update
- Problems
  - The updates are mathematically correct, but gradient descent fails because the gradients explode or vanish
  - This limits the scope of the dependencies over time

Why Gradients Explode or Vanish

- Motivating example:
  - Consider a univariate version of the vanilla RNNs





$$z^{(t+1)} = wh^{(t)} + vx^{(t+1)}$$
$$h^{(t)} = \phi(z^{(t)})$$

Backprop updates:

$$\overline{h^{(t)}} = \overline{z^{(t+1)}}\, w$$
$$\overline{z^{(t)}} = \overline{h^{(t)}}\, \phi'(z^{(t)})$$

Applying this recursively:

$$\overline{h^{(1)}} = \underbrace{w^{T-1}\phi'(z^{(2)}) \cdots \phi'(z^{(T)})}_{\text{the Jacobian } \partial h^{(T)}/\partial h^{(1)}} \overline{h^{(T)}}$$

With linear activations:

$$\partial h^{(T)}/\partial h^{(1)} = w^{T-1}$$

Exploding:

$$w = 1.1, T = 50 \quad \Rightarrow \quad \frac{\partial h^{(T)}}{\partial h^{(1)}} = 117.4$$

Vanishing:

$$w = 0.9, T = 50 \quad \Rightarrow \quad \frac{\partial h^{(T)}}{\partial h^{(1)}} = 0.00515$$

Why Gradients Explode or Vanish

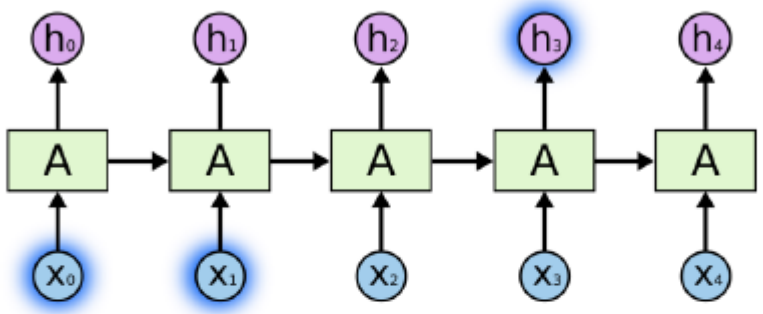
- In the multivariate case, the Jacobians multiply:

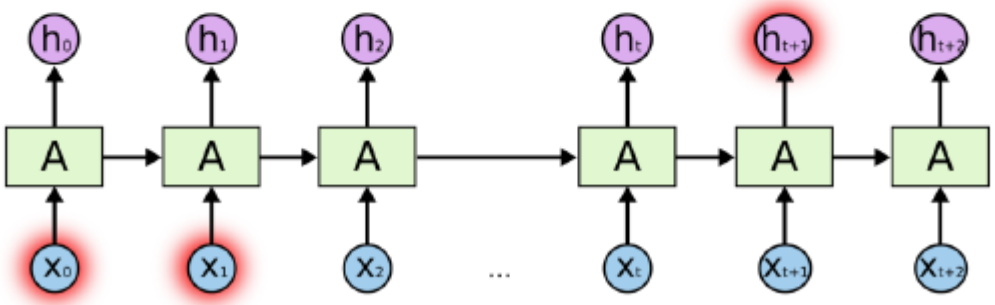
$$\frac{\partial \mathbf{h}^{(T)}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathbf{h}^{(T)}}{\partial \mathbf{h}^{(T-1)}} \cdots \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}}$$

- Contrast this with the forward pass
  - The forward pass has nonlinear activation functions which squash the activations, preventing them from blowing up.
  - The backward pass is linear, so it's hard to keep things stable. There's a thin line between exploding and vanishing.

Vanilla RNN

- Difficulty in modeling long-term dependency



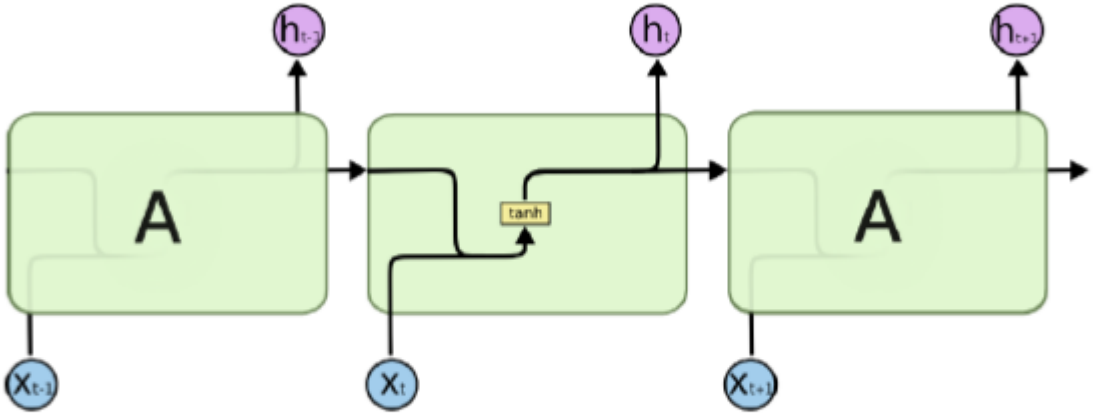


Long-Term Short Term Memory (LSTM)

- Replacing a vanilla RNN neuron by the LSTM unit
- Why it is called LSTM
  - A network's activations are its short-term memory and its weights are its long-term memory
  - The LSTM architecture wants the short-term memory to last for a long time period
- Key idea
  - Composed of memory cells which have controllers that decide when to store or forget information

Standard RNN

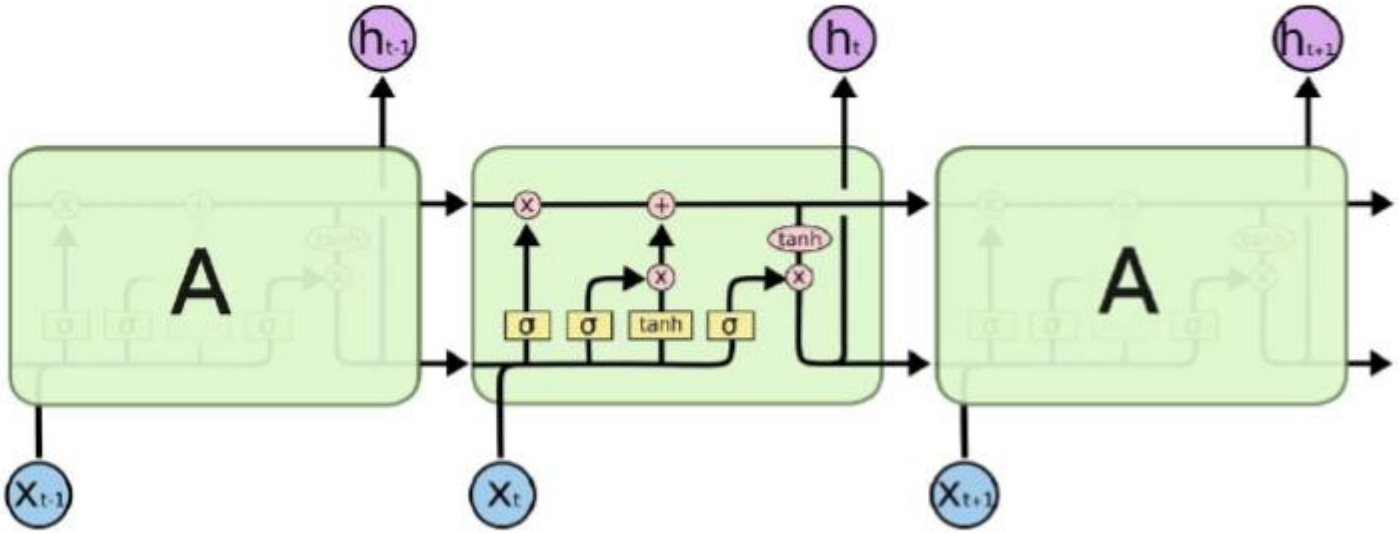
- Recall



- Each recurrent neuron receives past outputs and current input
- Pass through a tanh function

Long-Term Short Term Memory (LSTM)

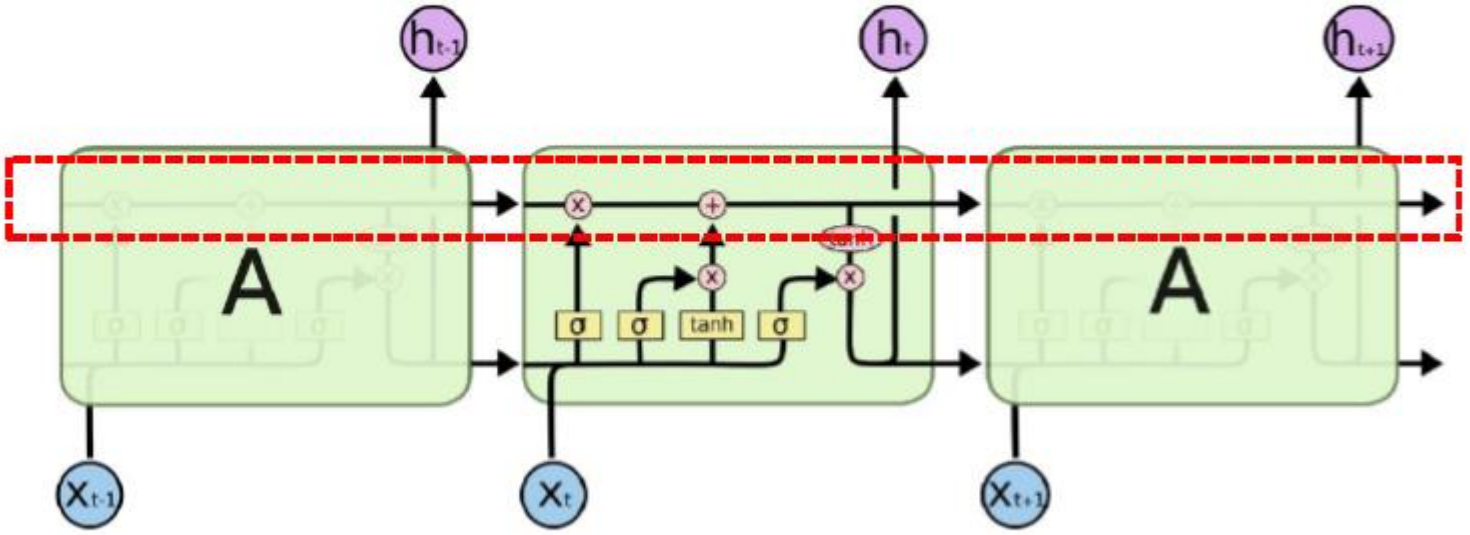
- LSTM uses multiplicative gates that decide if something is important or not



Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation

Long-Term Short Term Memory (LSTM)

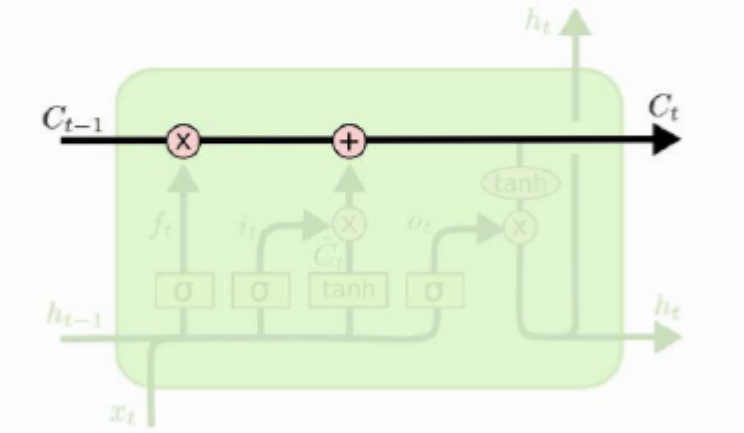
- Key component: a remembered cell state



Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation

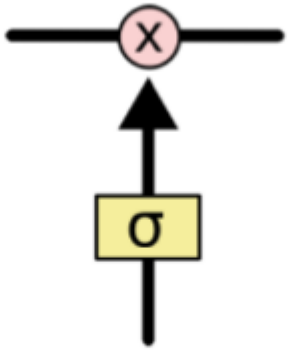
LSTM: Cell State

- A linear history
  - Carries information through
  - Only affected by a gate and addition of current information, which is also gated



LSTM: Gates

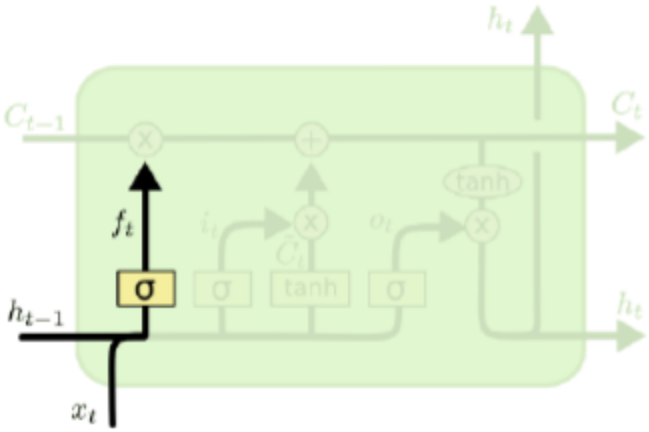
- Gates are simple sigmoid units with output range in (0,1)
- Controls how much of the information will be let through



- Three gates
  - Forget gate
  - Input gate
  - Output gate

LSTM: Forget Gate

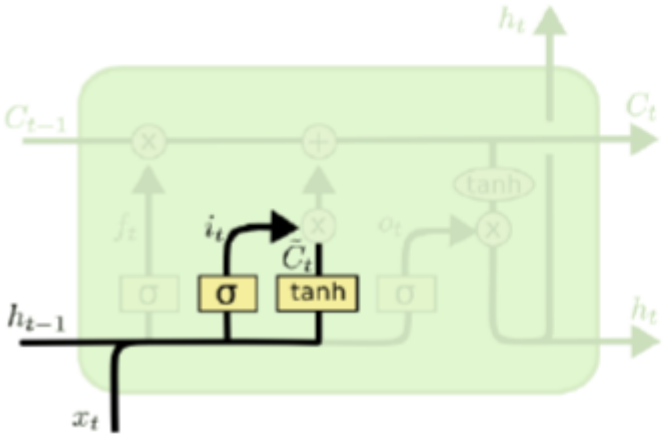
- The first gate determines whether to carry over the history or to forget it
  - Soft decision: how much of the history C_{t-1} to carry over
  - Determined by the current input x_t and the previous state h_{t-1}
  - $\langle h_{t-1}, C_{t-1} \rangle$ can be viewed as partial key-value pairs



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM: Input Gate

- The second gate has two parts
  - A gate that decides if it is worth remembering
  - A nonlinear transformation that extracts new and interesting information from the input
  - Both use the current input and the previous state

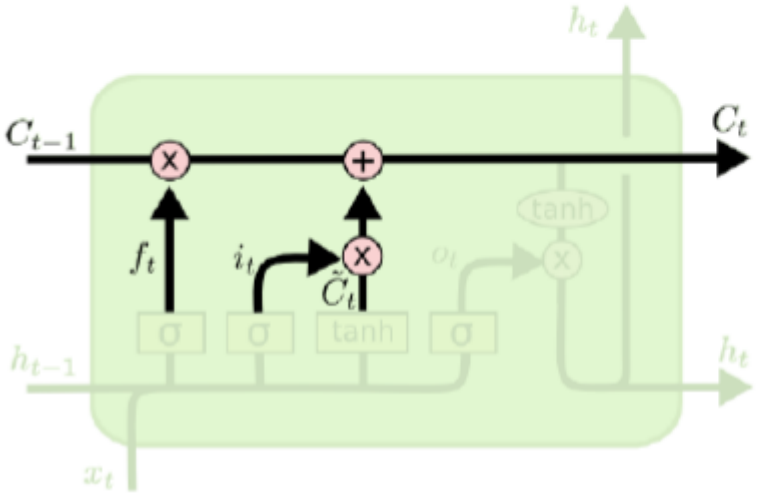


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM: Memory Cell Update

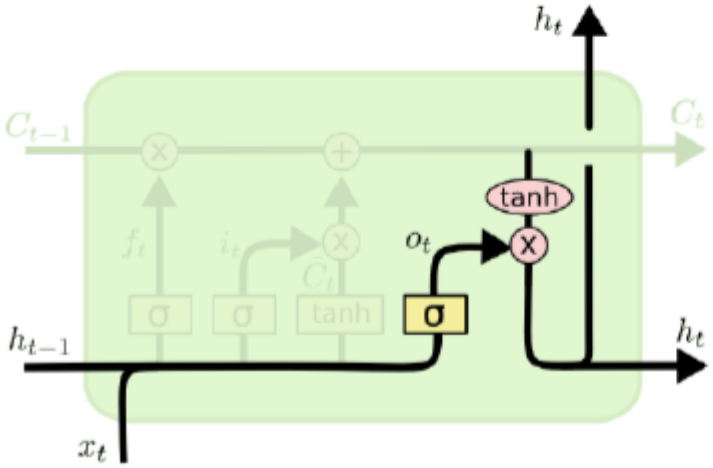
- The output of the second part is added into the current memory cell
  - Can be viewed as value update in a key-value pair
  - The input and state jointly decide how much of history info is kept and how much of embedded input info is added
  - A dynamic mixture of experts at each time step



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM: Output Gate

- The third gate is the output gate
  - To decide if the memory cell contents are worth reporting at this time using the current input and previous state
- The output of the cell or the state
  - A nonlinear transform of the cell values
  - Compress it with tanh to make it in (-1,1)
  - Note the separation of key-value representation

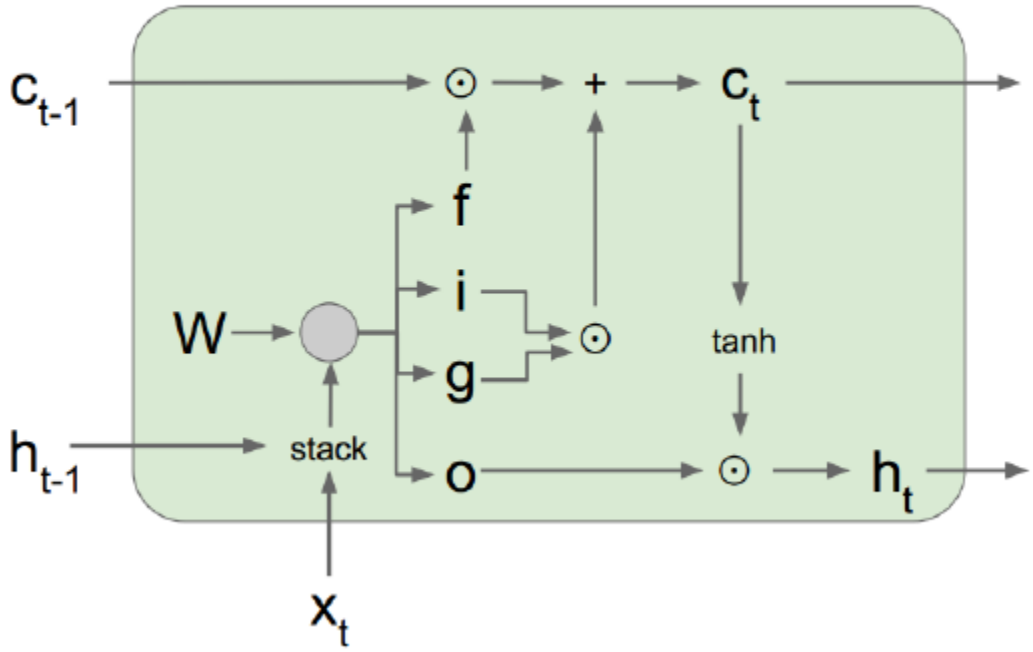


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM

[Hochreiter et al., 1997]



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

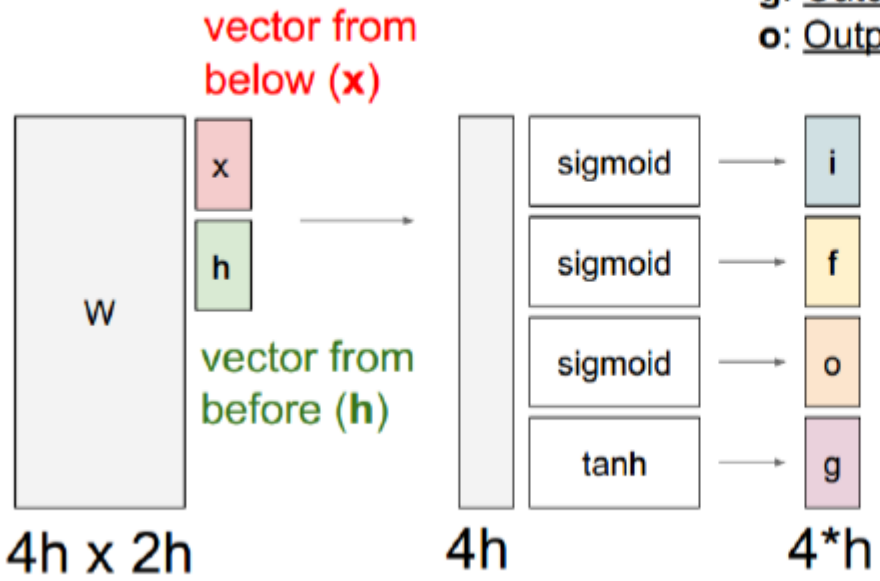
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM

[Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell



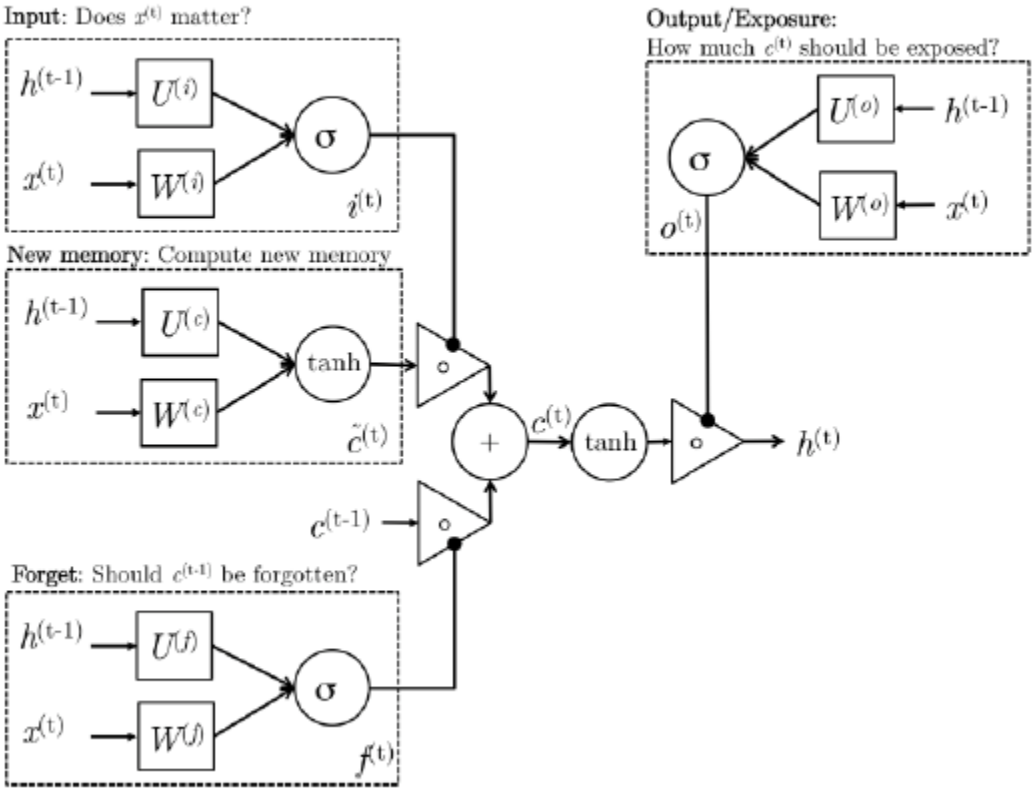
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM: As A Feedforward Layer

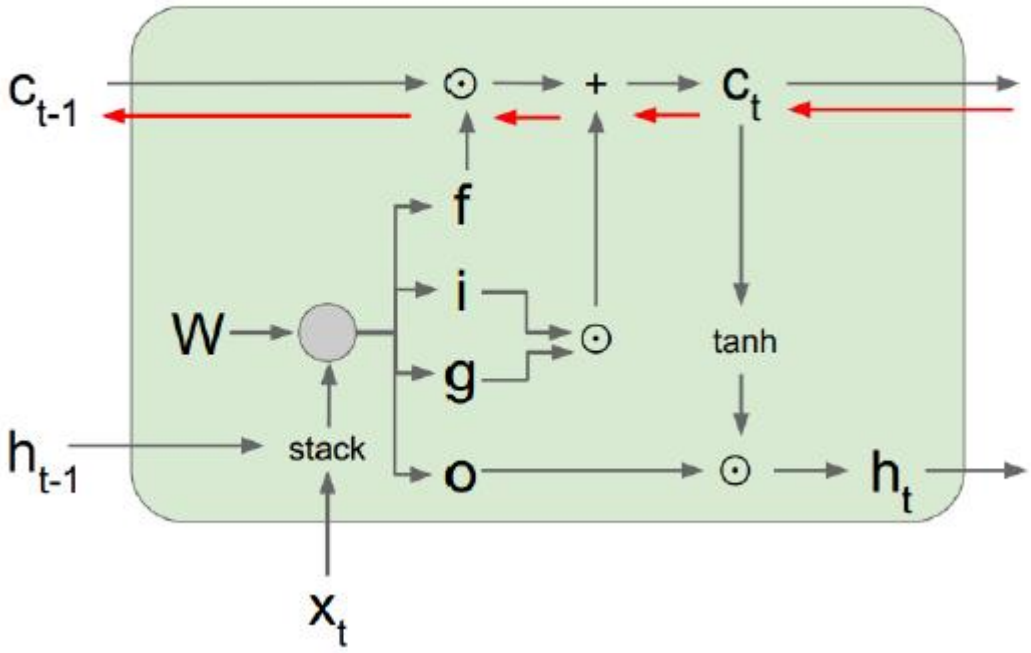
- As a gated feedforward network



Richard Socher's CS224D notes

LSTM: Backpropagation

[Hochreiter et al., 1997]



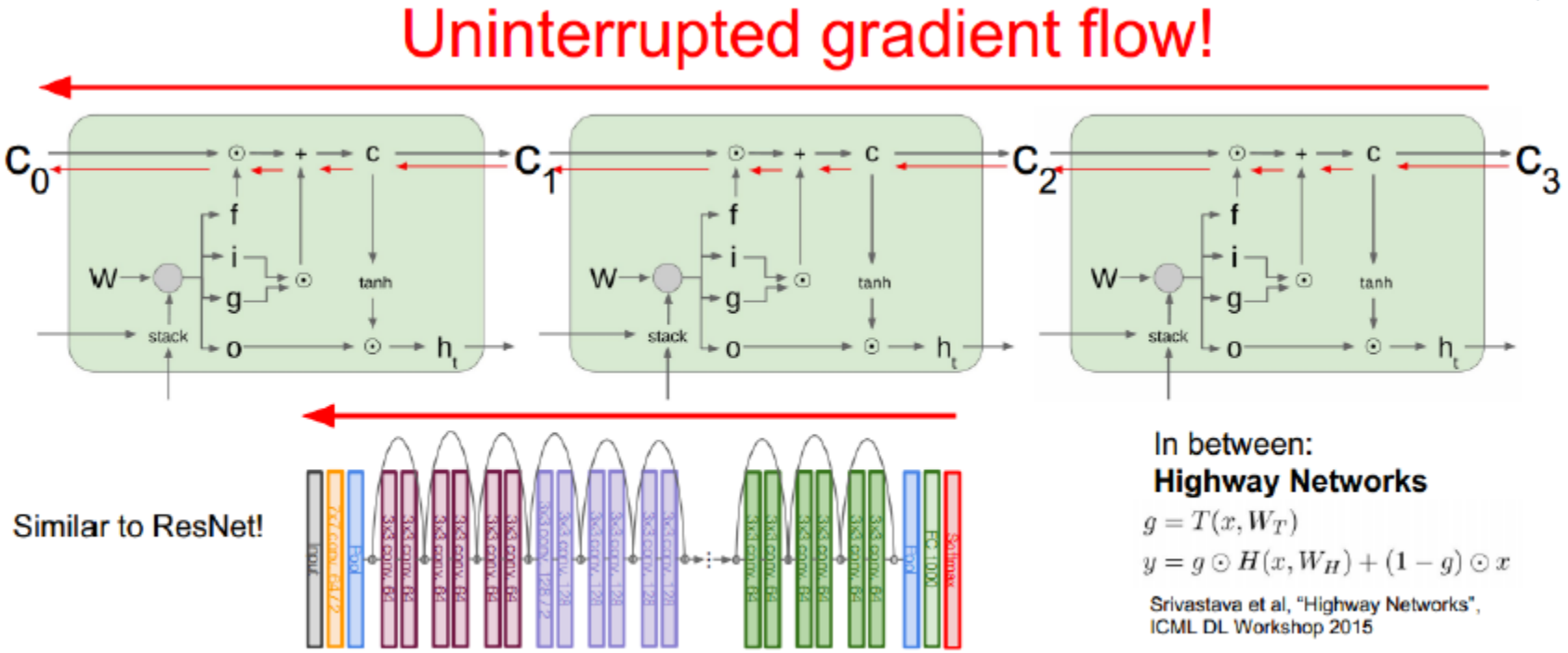
Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

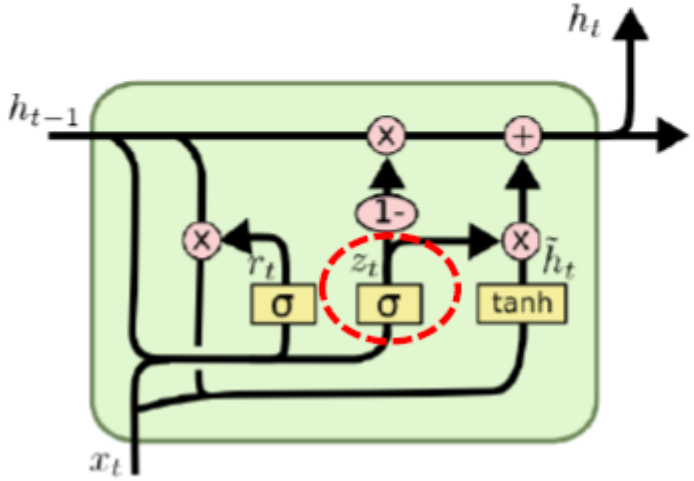
$$h_t = o \odot \tanh(c_t)$$

LSTM: Backpropagation



Gated Recurrent Units (GRU)

- Simplified LSTM
  - Combine the forget and input gates



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Recurrent Neural Networks-RNN

- Motivation
- Basic RNN
- Training RNN and LSTM
- **Applications in Vision and NLP**
- Attention Models

X-to-Sequence Problems

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

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

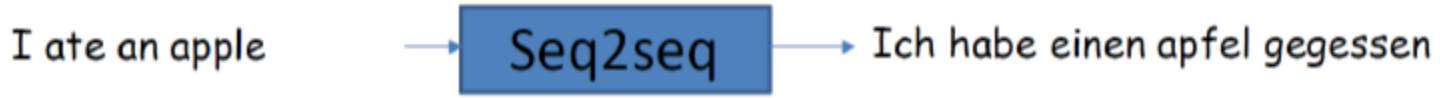
  - Image caption generation

→ 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

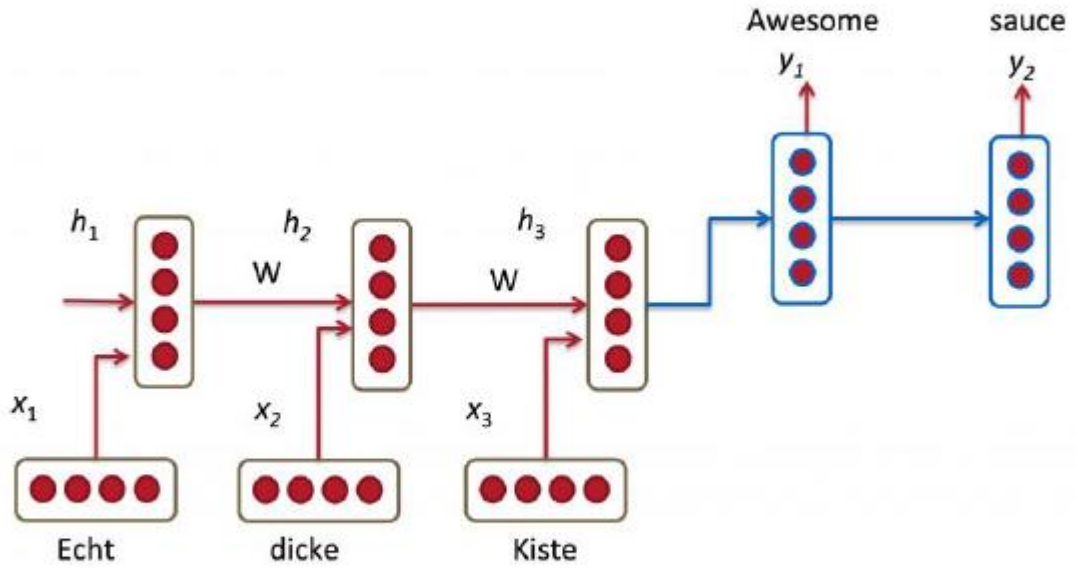
- Task definition



  - 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

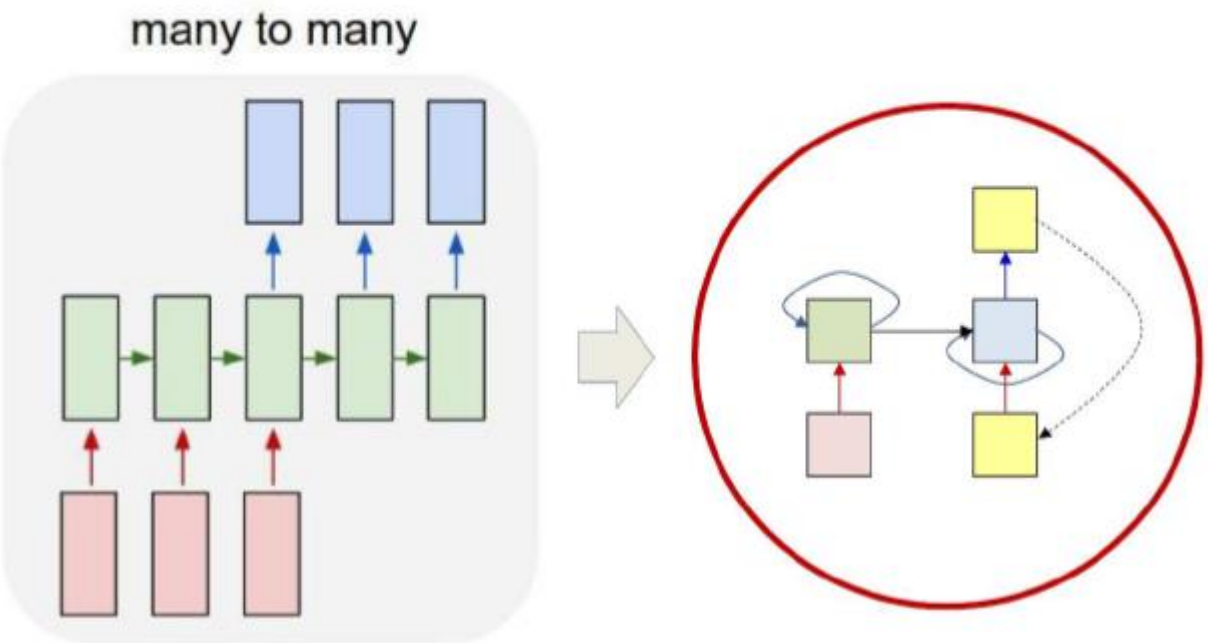
- Delayed sequence to sequence
  - Many to many



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

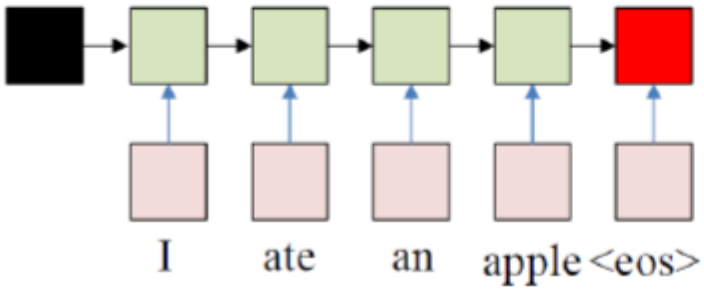
Modeling the Problem

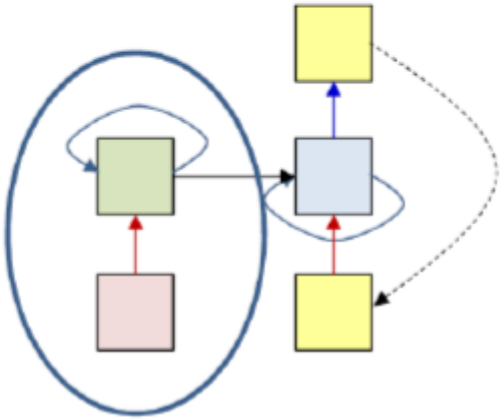
- Delayed sequence to sequence
  - Delayed self-referencing sequence-to-sequence



The “Simple” Translation Model

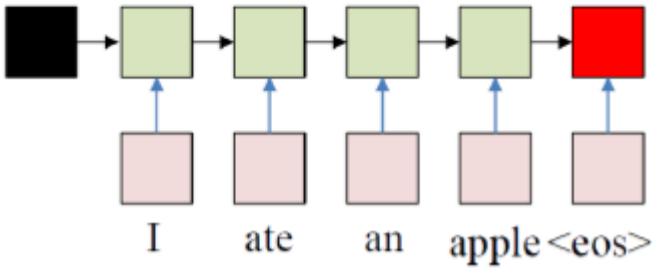
- The input sequence feeds into an recurrent structure
  - The input sequence is terminated by an explicit <eos> symbol
- The hidden activation at the <eos> “stores” all information about the sentence

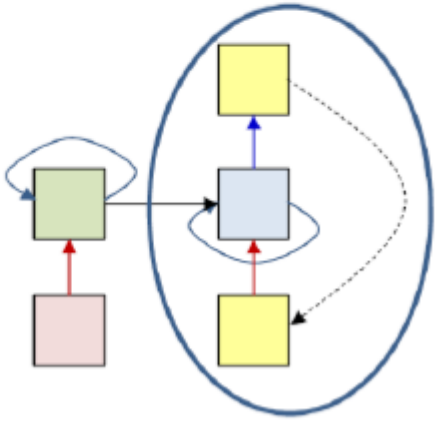




The “Simple” Translation Model

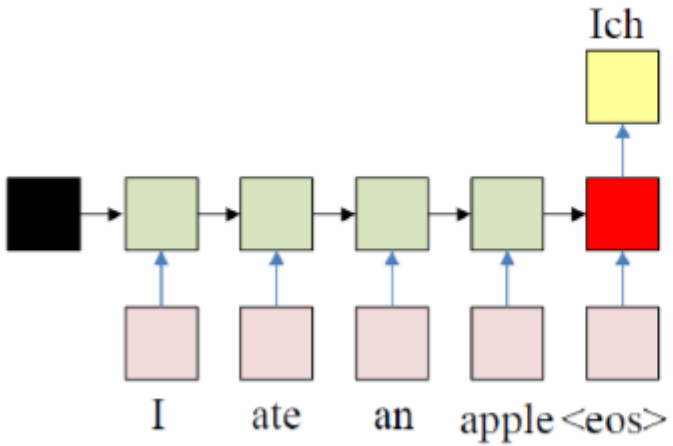
- Subsequently a second RNN uses the hidden activation as initial state to produce a sequence of outputs
  - The output at each time becomes the input at the next time
  - Output production continues until an <eos> is produced





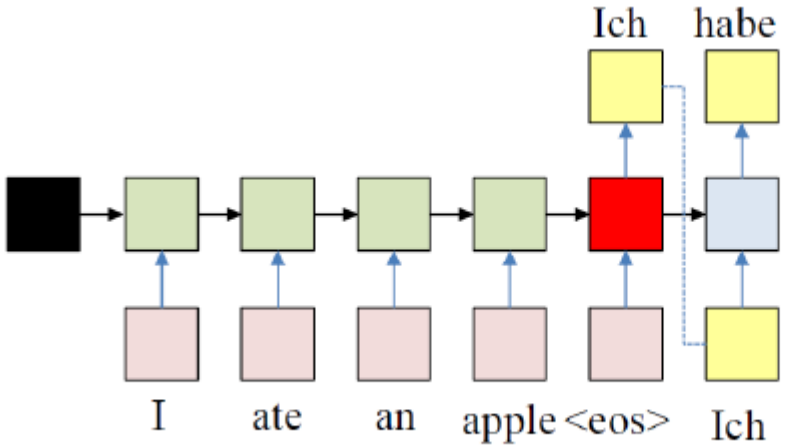
The “Simple” Translation Model

- 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



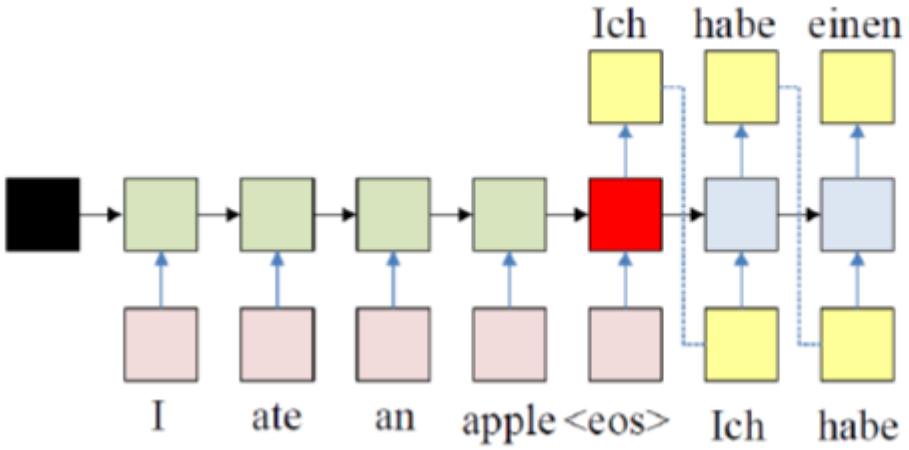
The “Simple” Translation Model

- 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



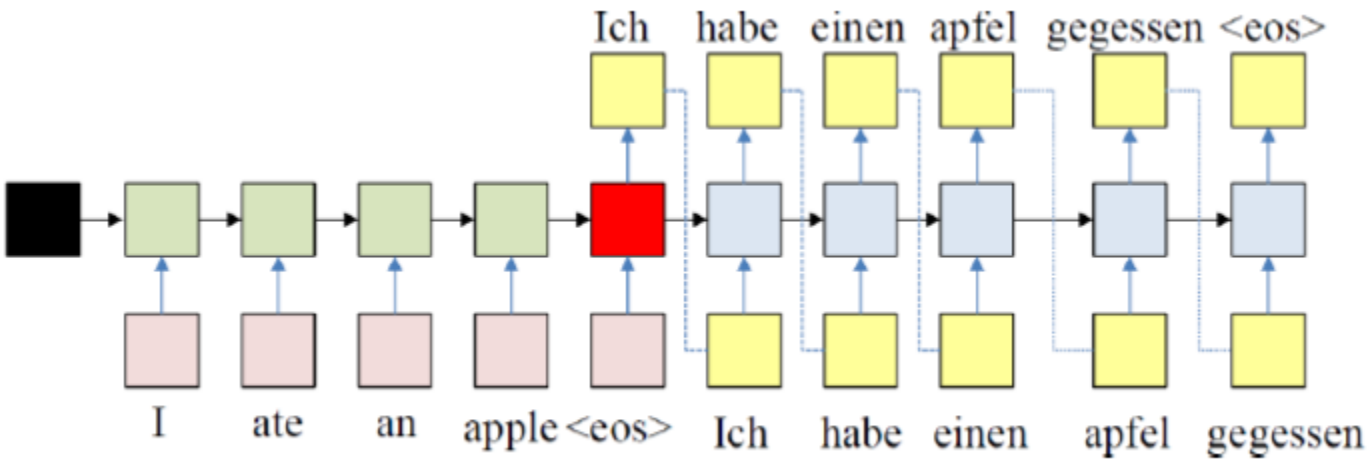
The “Simple” Translation Model

- 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



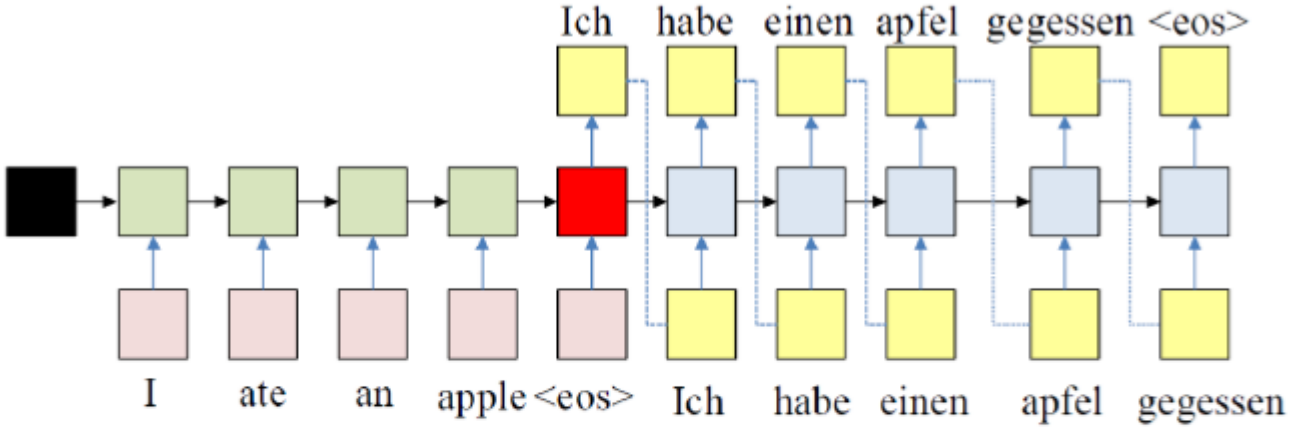
The “Simple” Translation Model

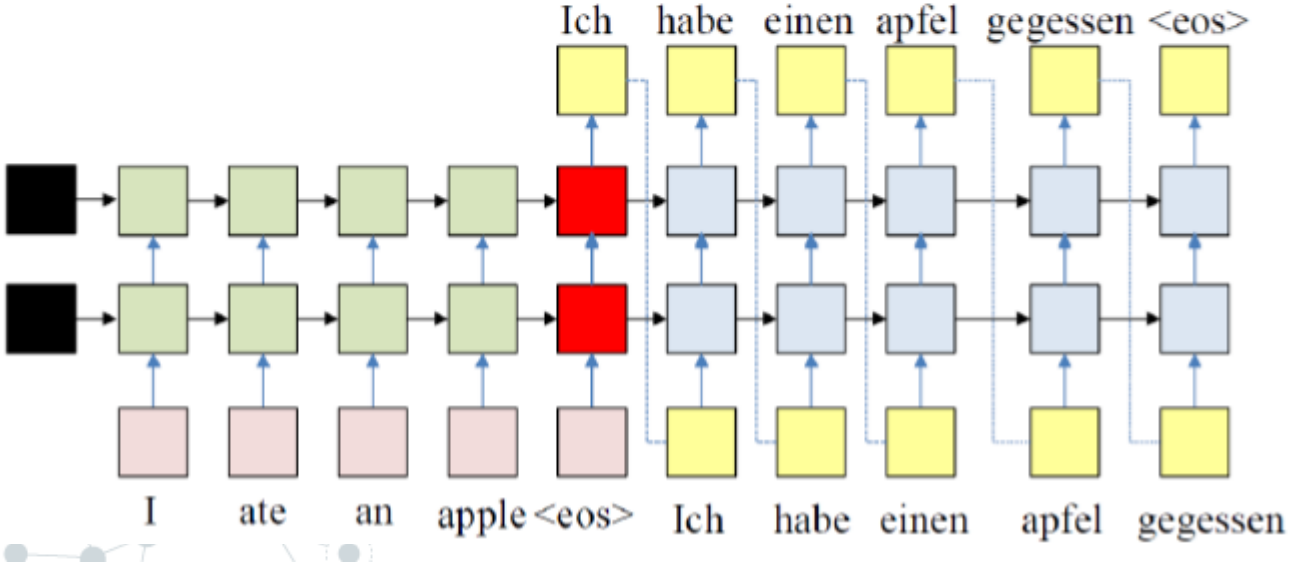
- Subsequently a second RNN uses the hidden activation as initial state to produce a sequence of outputs
  - The output at each time becomes the input at the next time
  - Output production continues until an <eos> is produced



The “Simple” Translation Model

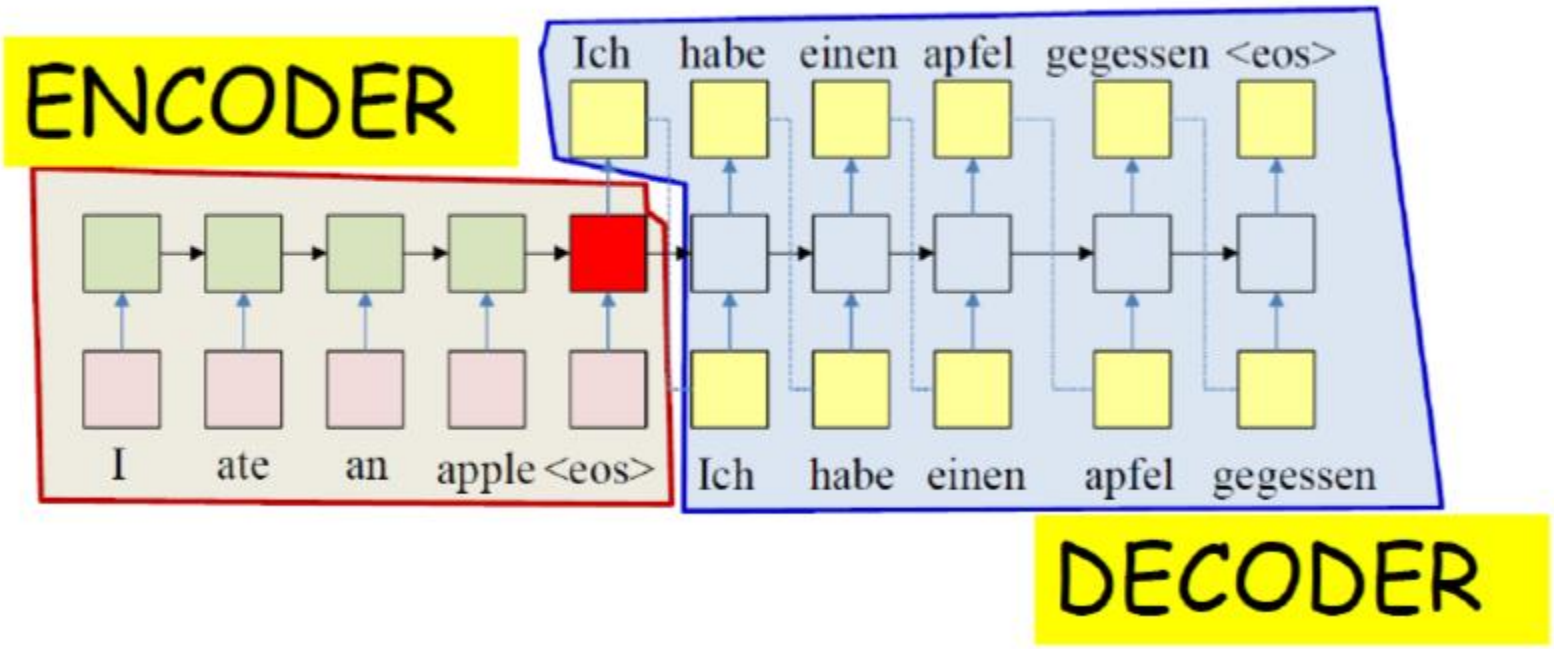
- Such an architecture can be generalized to more layers





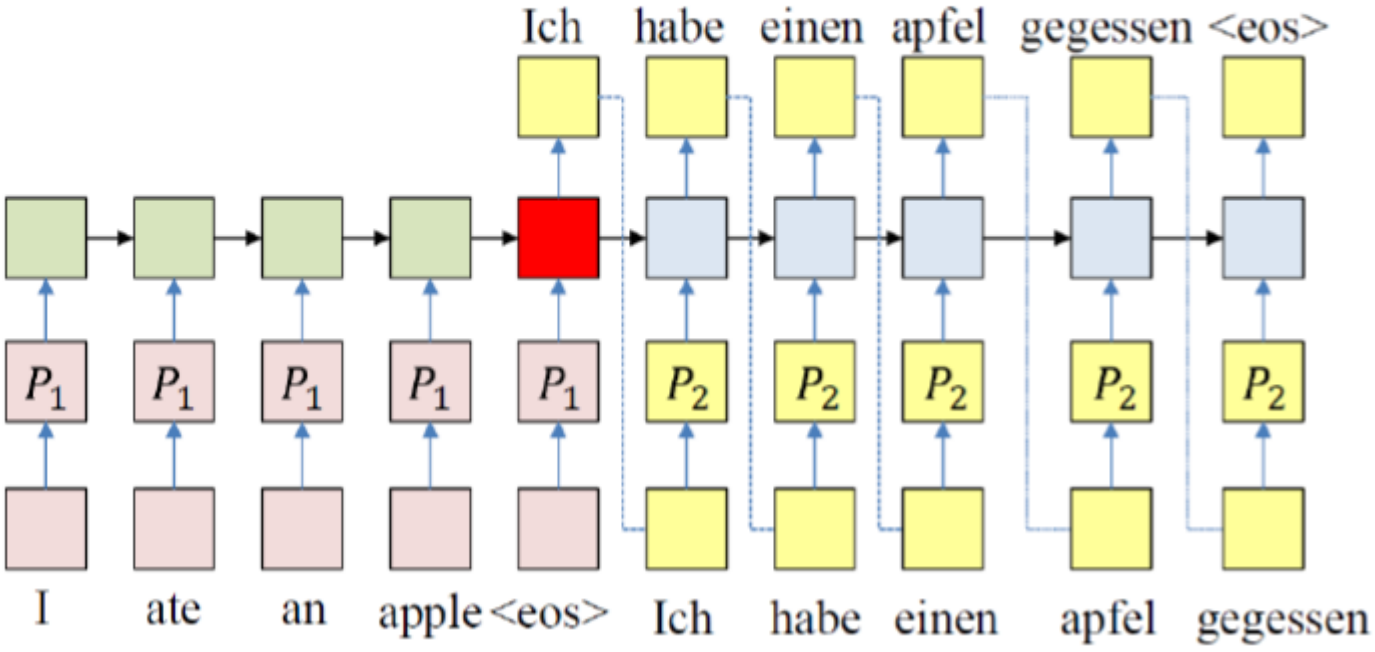
The “Simple” Translation Model

- 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

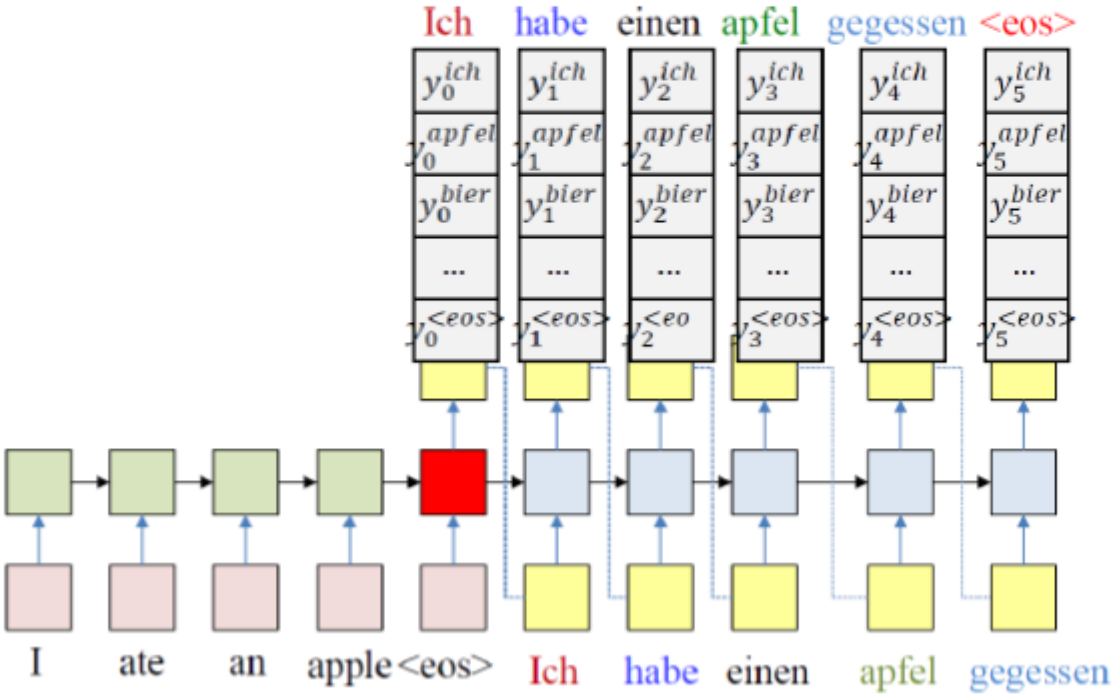


Prediction of Translation Model

- 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>



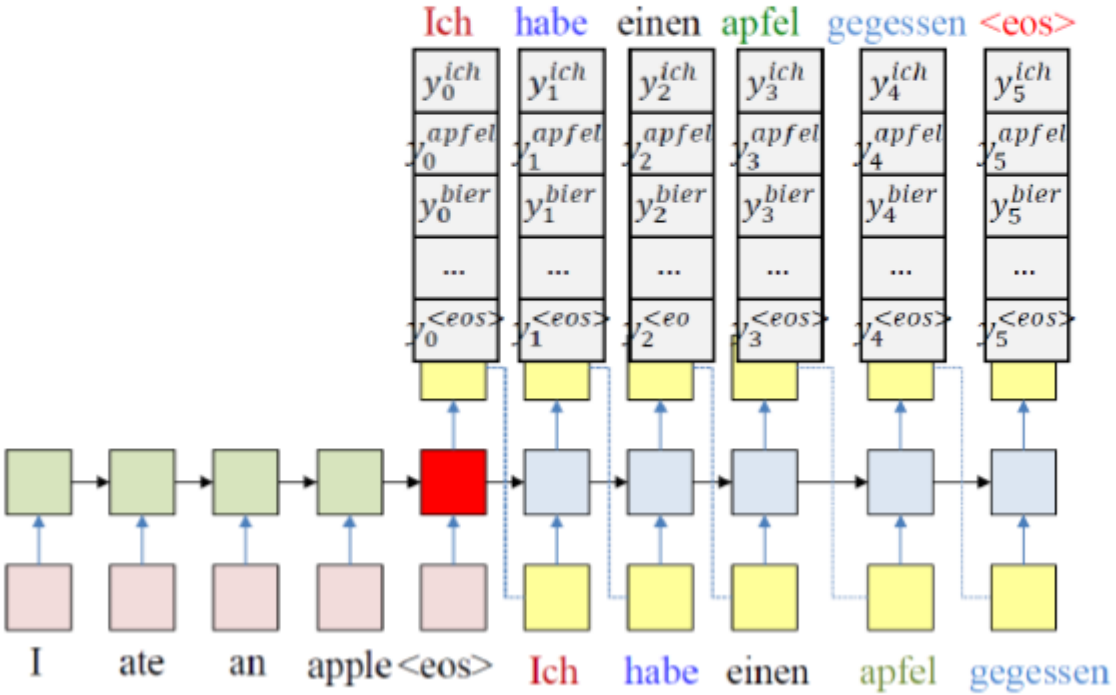
Prediction of Translation Model

- 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

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



Example: Human Machine Conversation

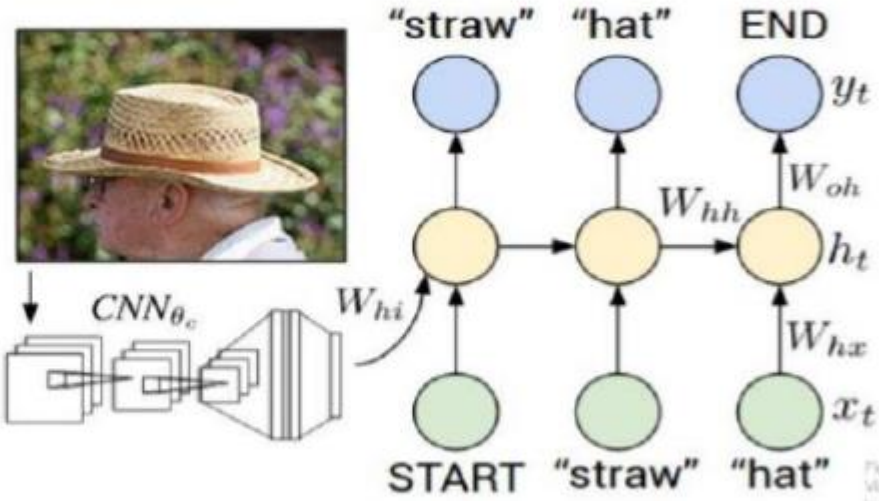
- Task: Human text in, machine response out
- Trained on human-human conversations

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

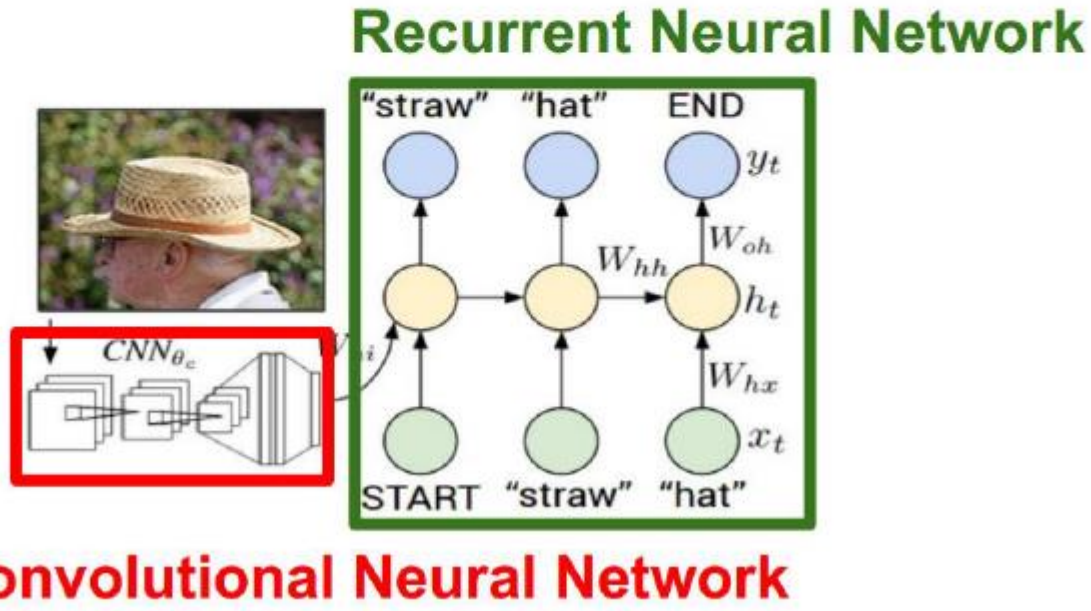
- Generating image captions



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

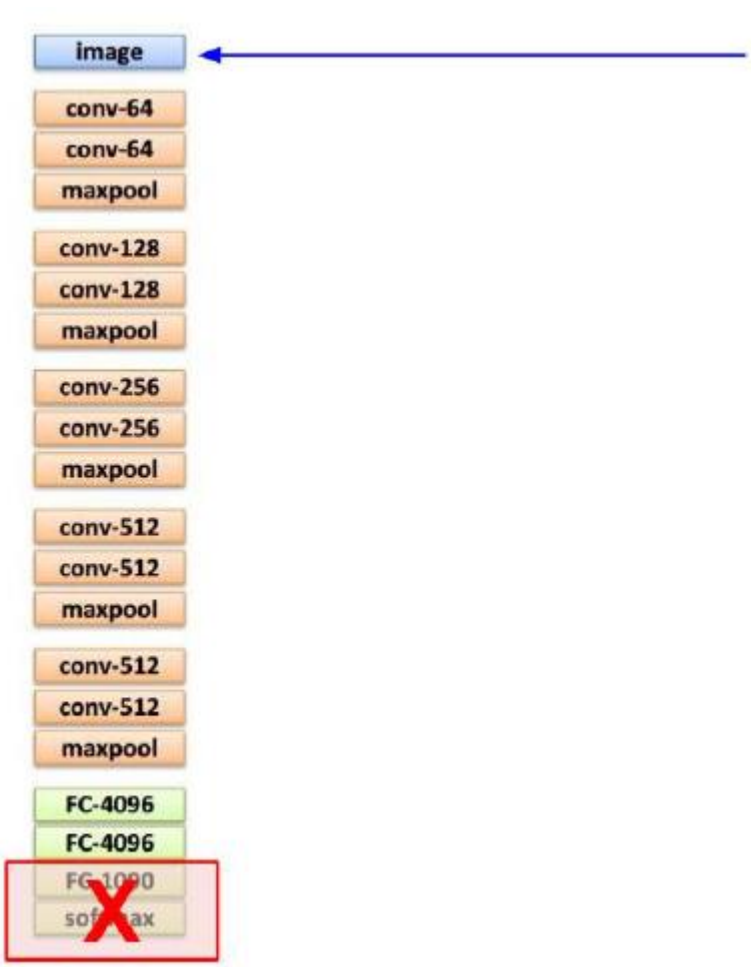
- Initial state is produced by a state-of-the-art CNN-based image classification system
- Subsequent model is the decoder end of a seq-to-seq model



Generating Image Captions

This image is CC0 public domain

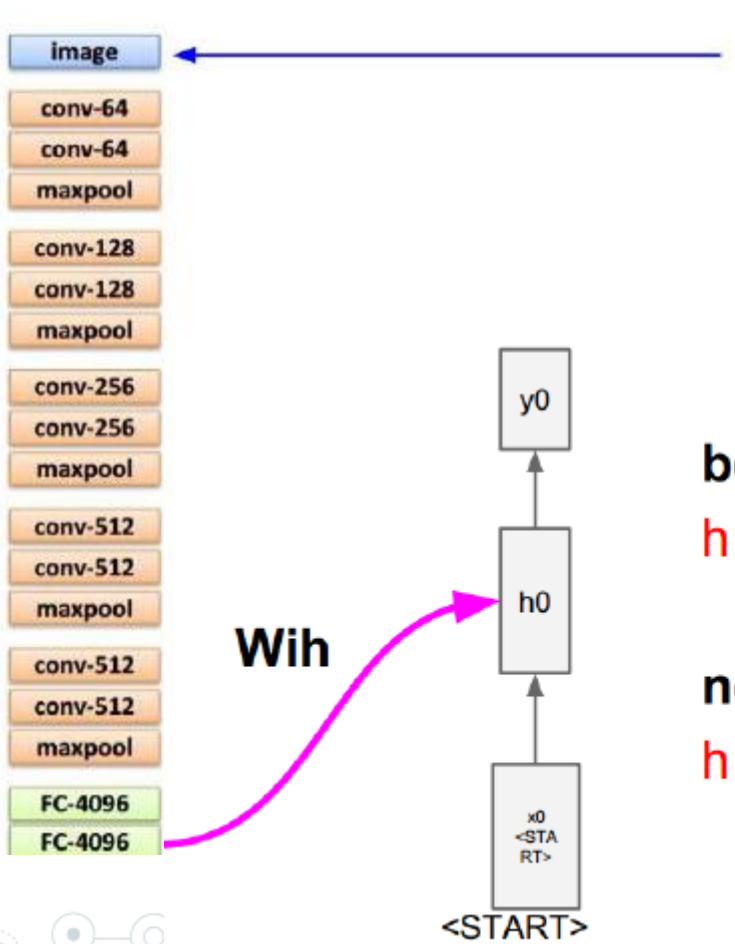
Generating Image Captions



This image is CC0 public domain

test image

Generating Image Captions



test image

This image is CC0 public domain

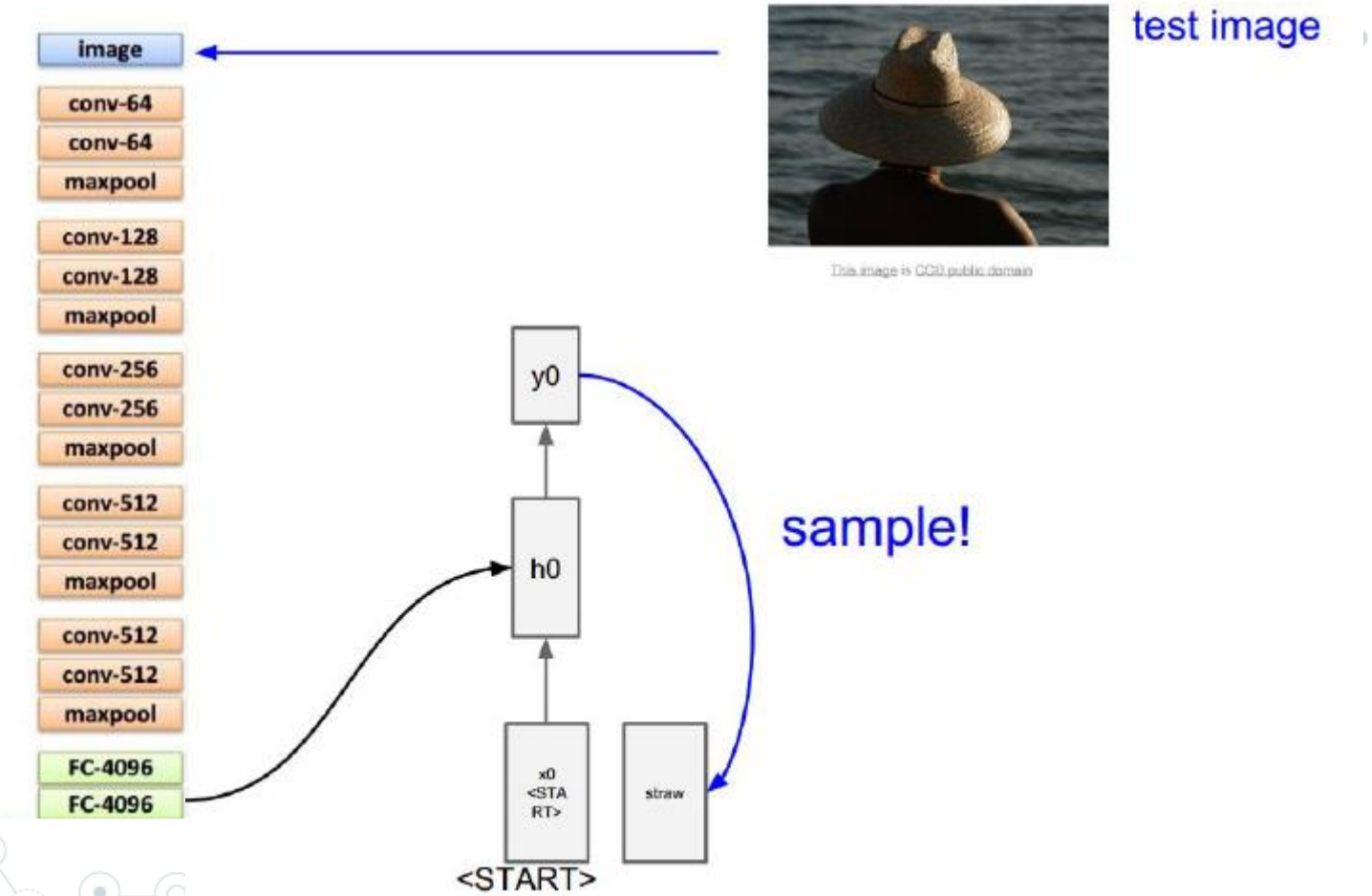
before:

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

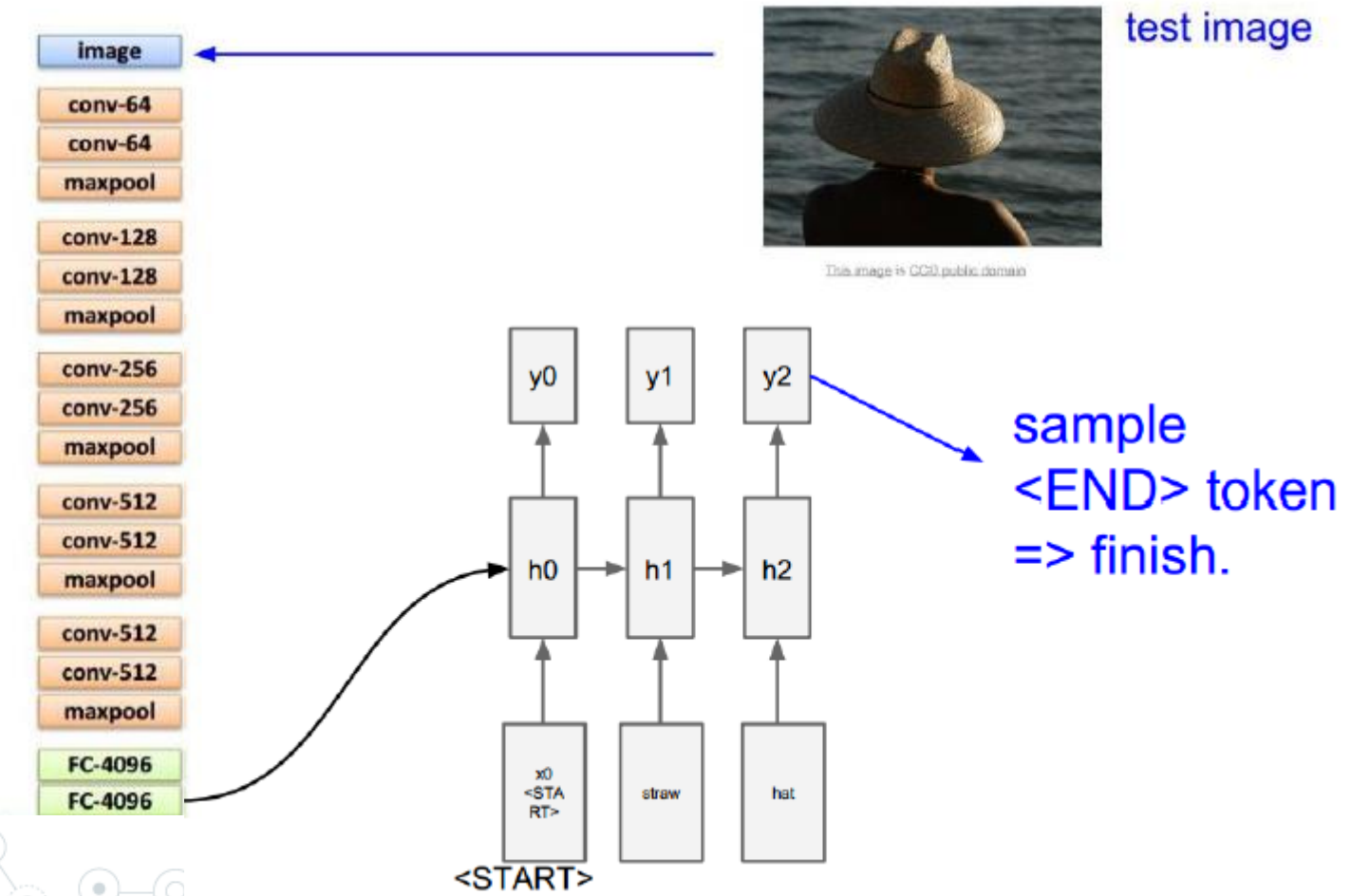
now:

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

Generating Image Captions

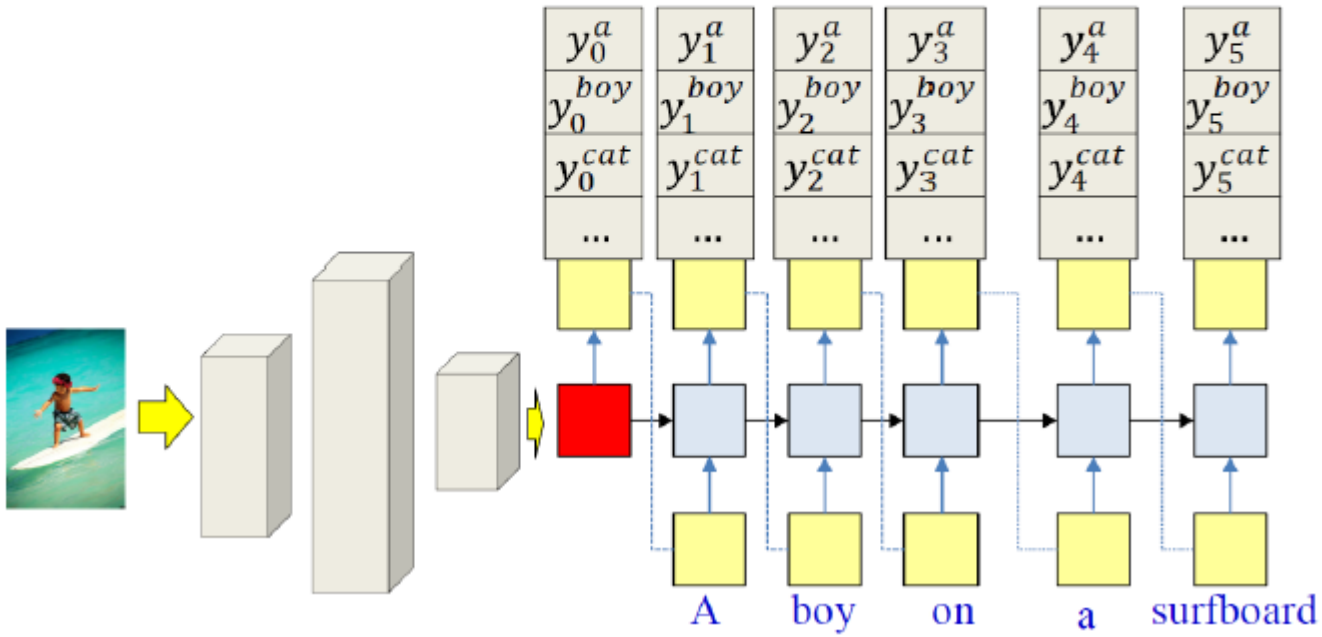


Generating Image Captions



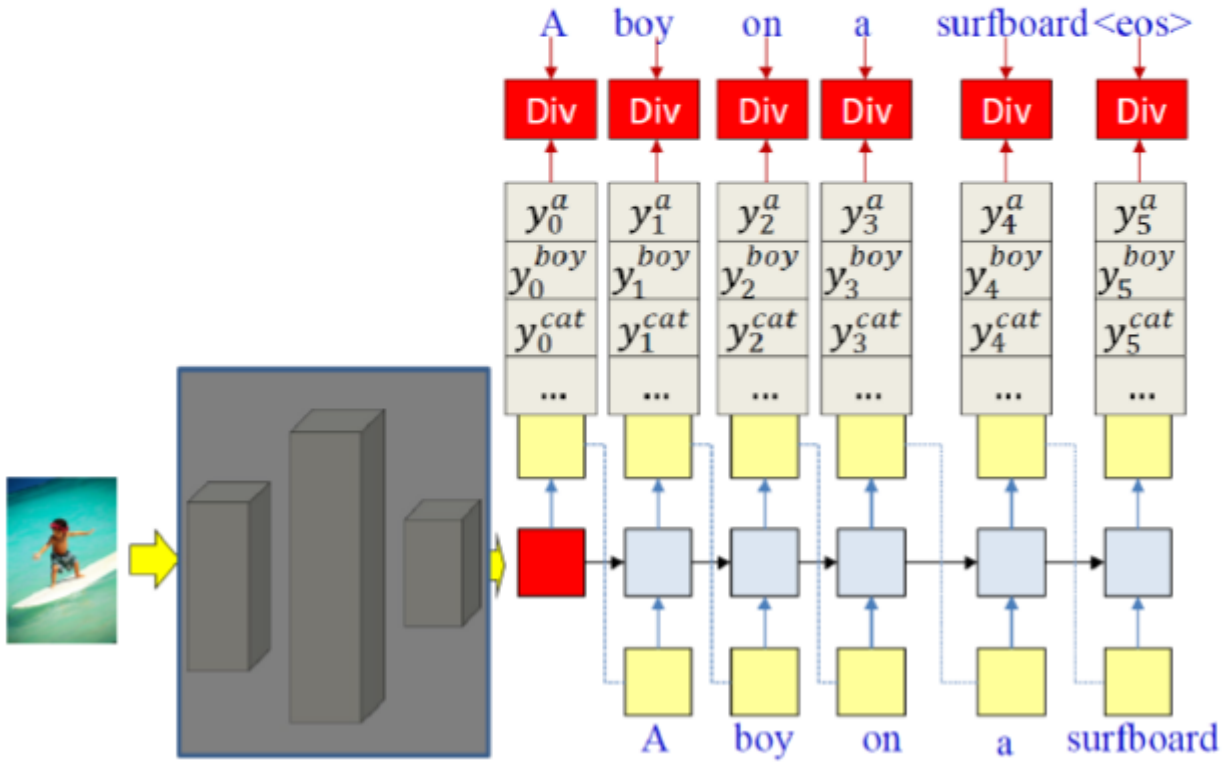
Training the Image-to-Sequence Model

- 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-Sequence 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."

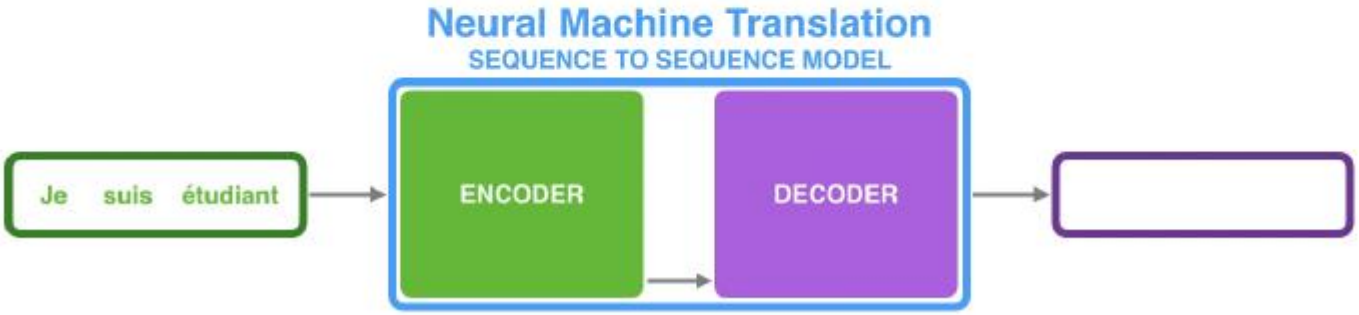
"a horse is standing in the middle of a road."

Recurrent Neural Networks-RNN

- Motivation
- Basic RNN
- Training RNN and LSTM
- Applications in Vision and NLP
- **Attention Models**

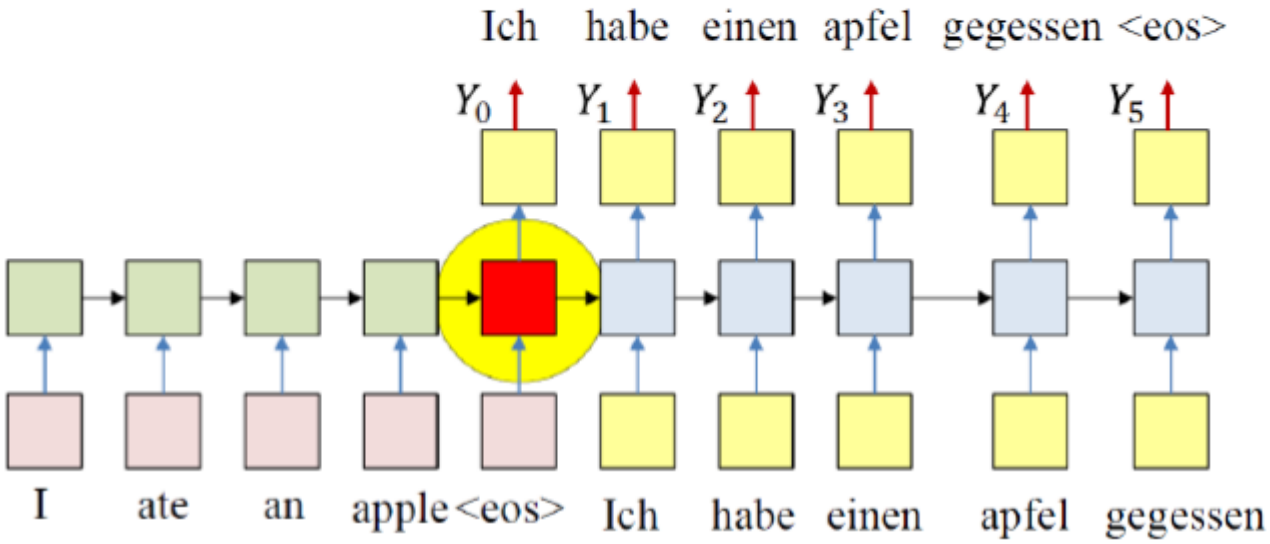
Recap

- RNN models
  - Encoder-decoder architecture



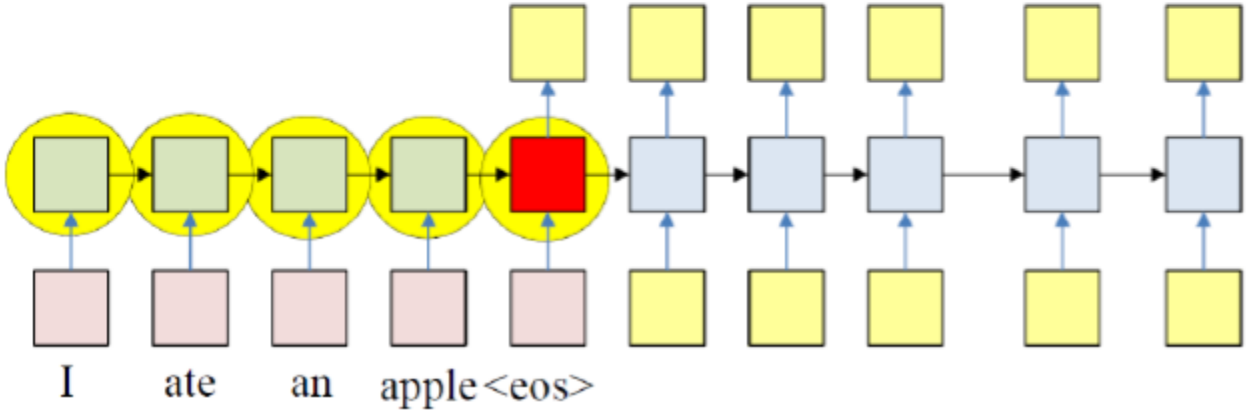
A Problem with This Framework

- 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



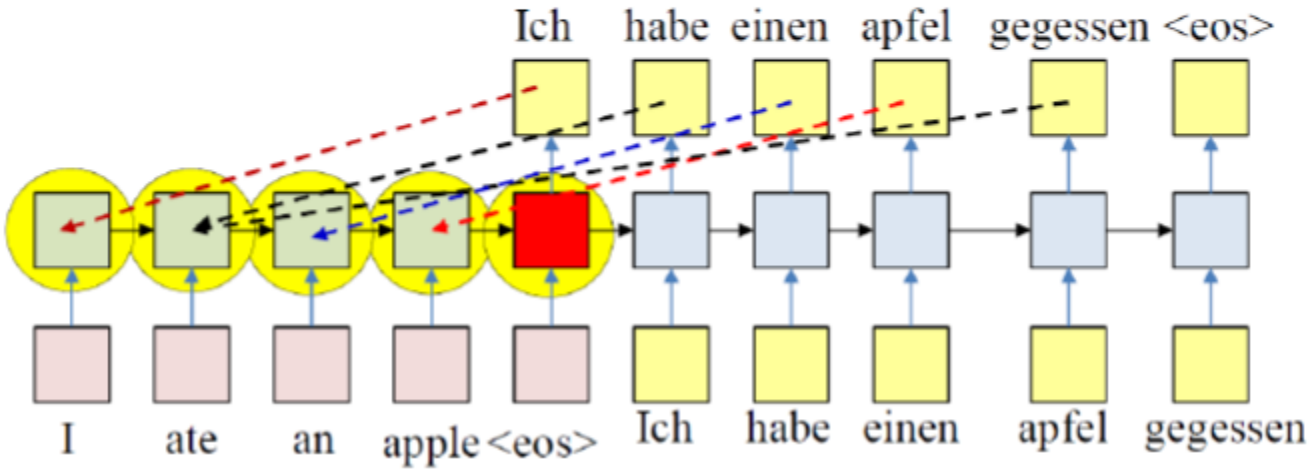
A Problem with This Framework

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



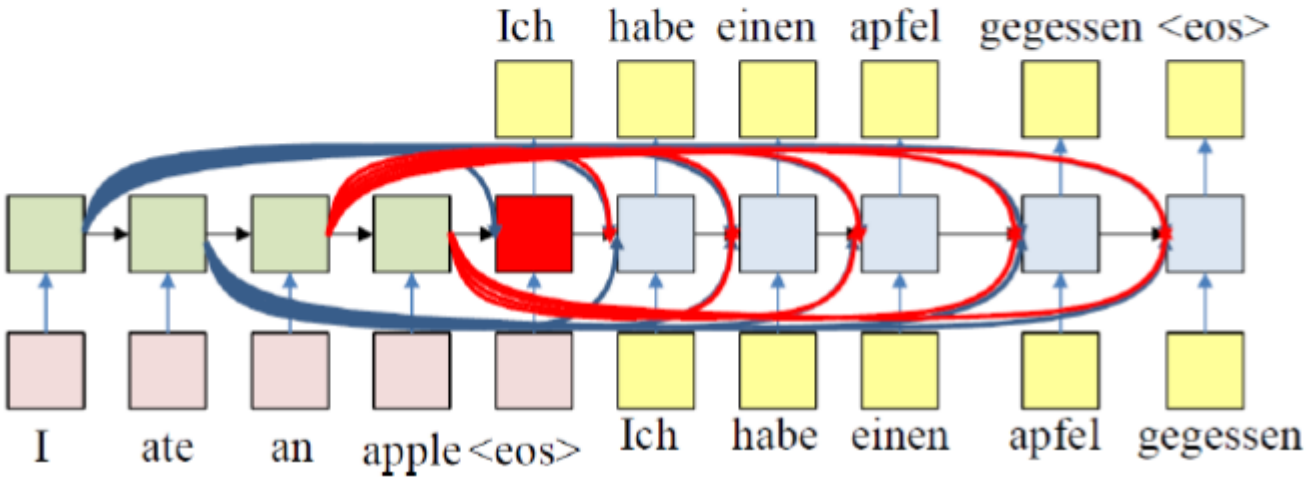
A Problem with This Framework

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



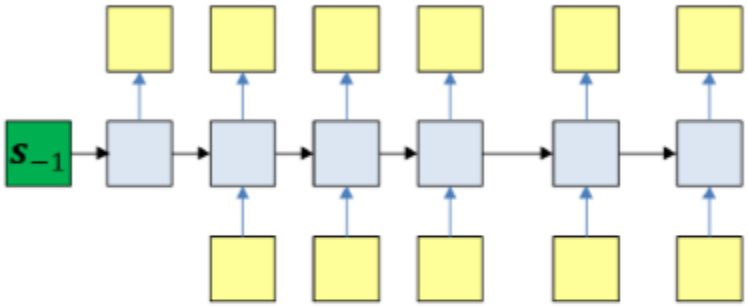
A Problem with This Framework

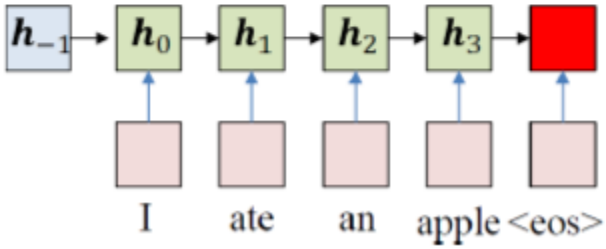
- All latent values carry information
  - Some of which may be diluted downstream
  - Different outputs are related to different inputs
  - Connecting everything to everything is infeasible



Attention Models

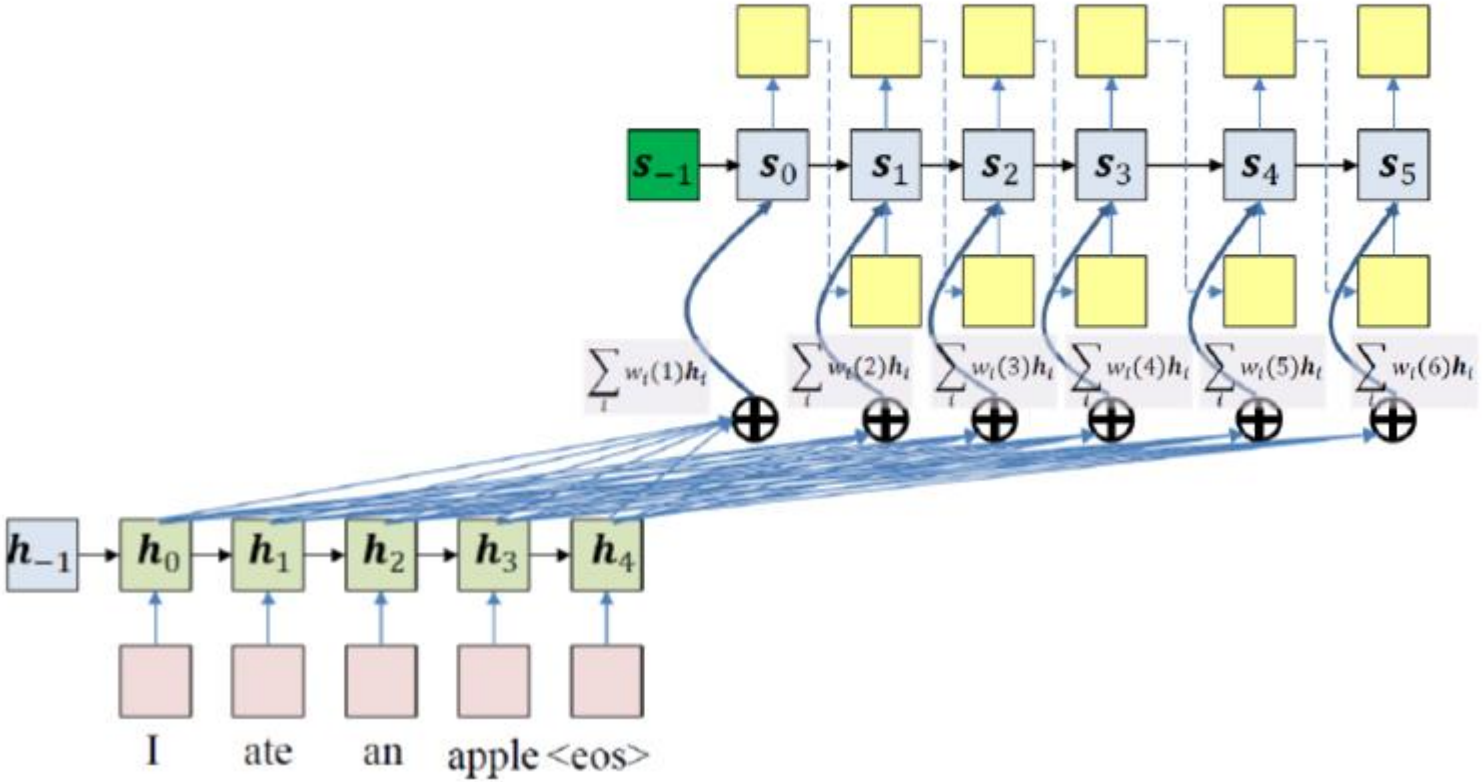
- Separating the encoder and decoder first





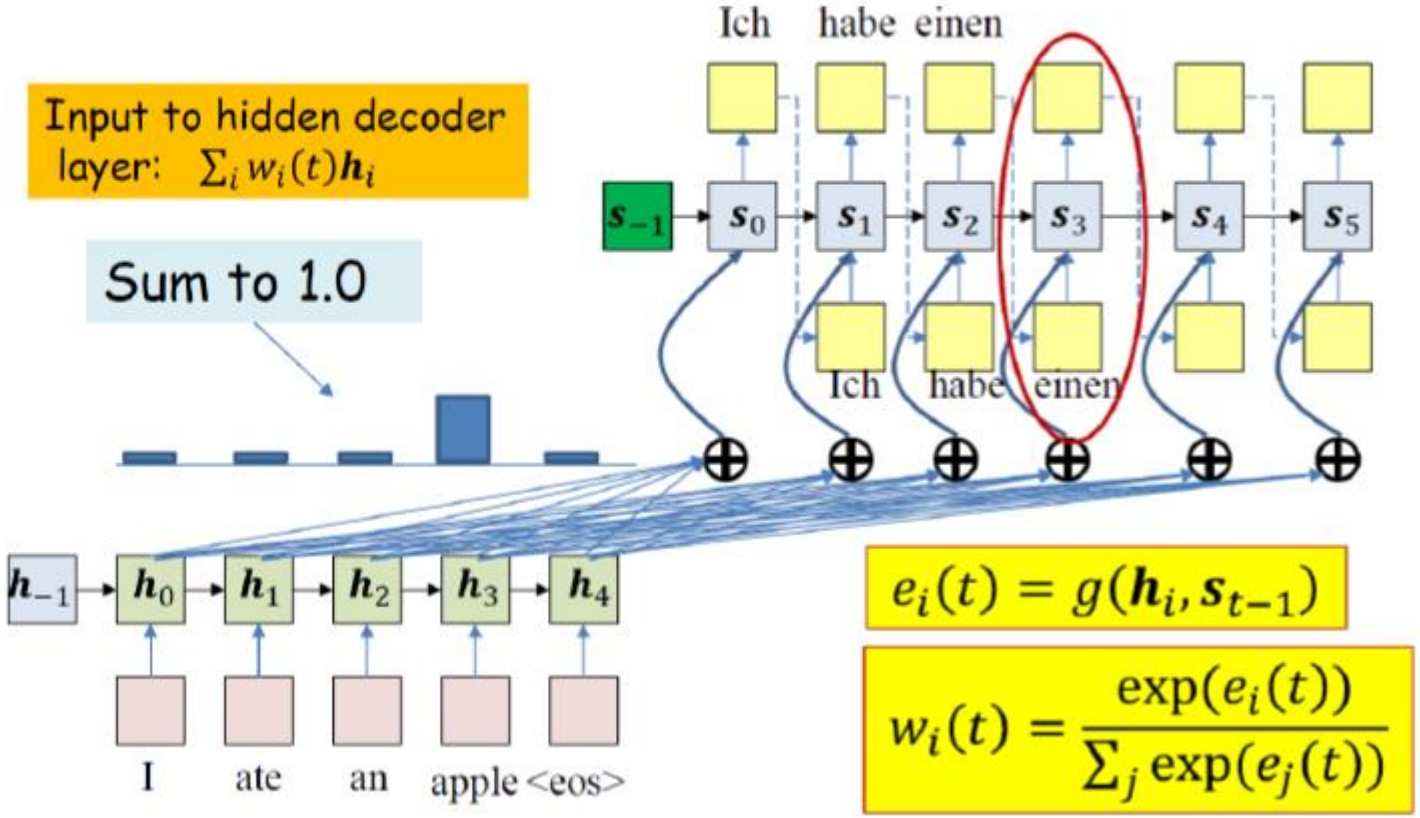
Attention Models

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



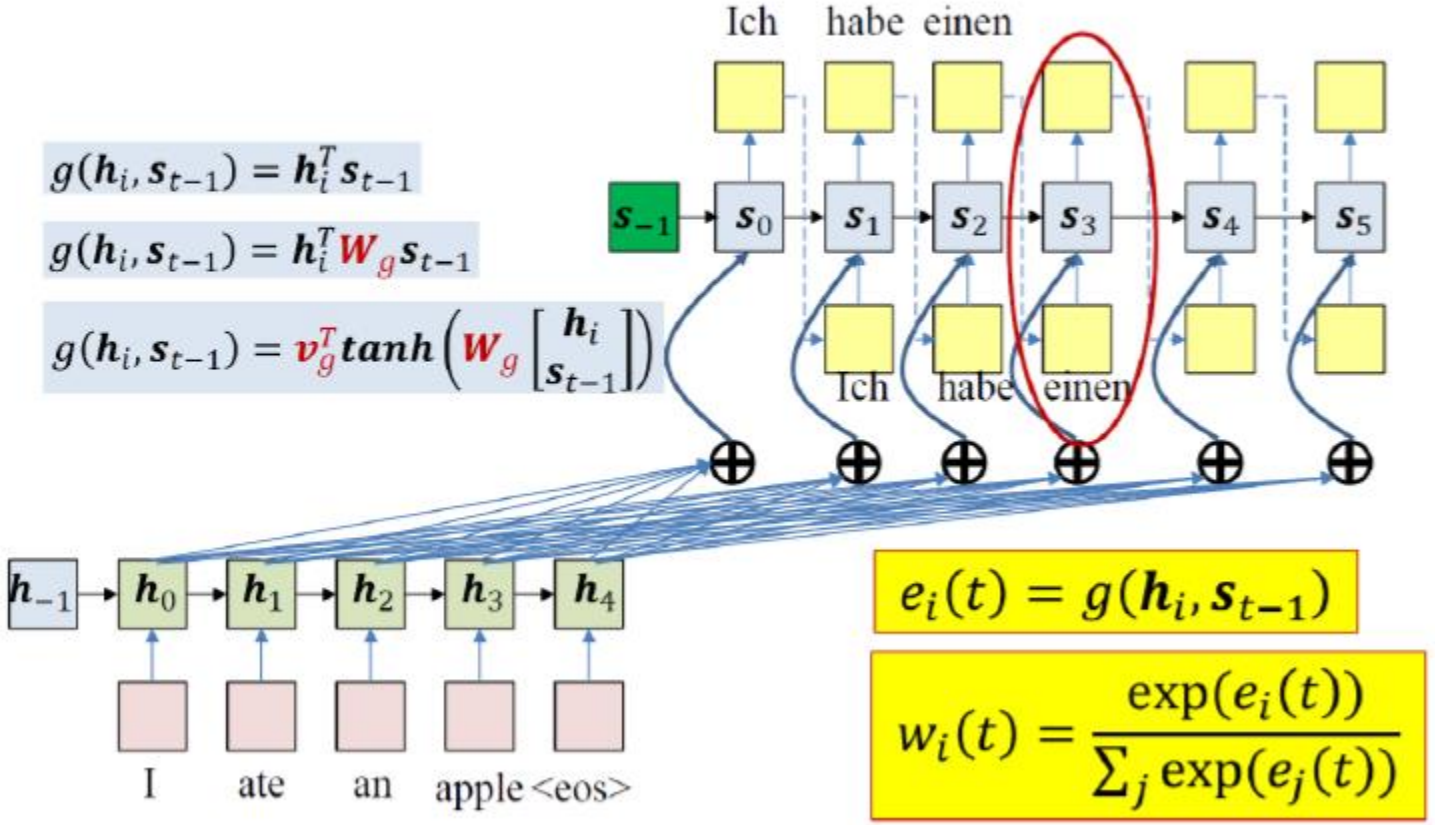
Attention Models

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



Attention Models

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



$$g(h_i, s_{t-1}) = h_i^T s_{t-1}$$

$$g(h_i, s_{t-1}) = h_i^T W_g s_{t-1}$$

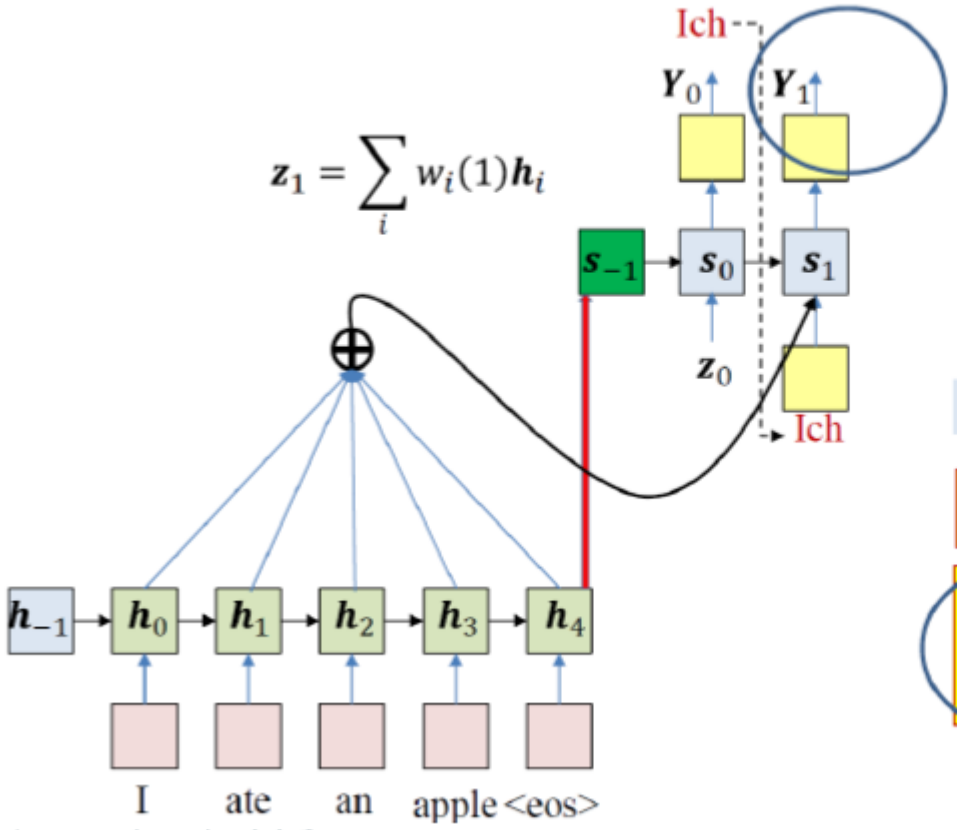
$$g(h_i, s_{t-1}) = v_g^T \tanh\left(W_g \begin{bmatrix} h_i \\ s_{t-1} \end{bmatrix}\right)$$

$$e_i(t) = g(h_i, s_{t-1})$$

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

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

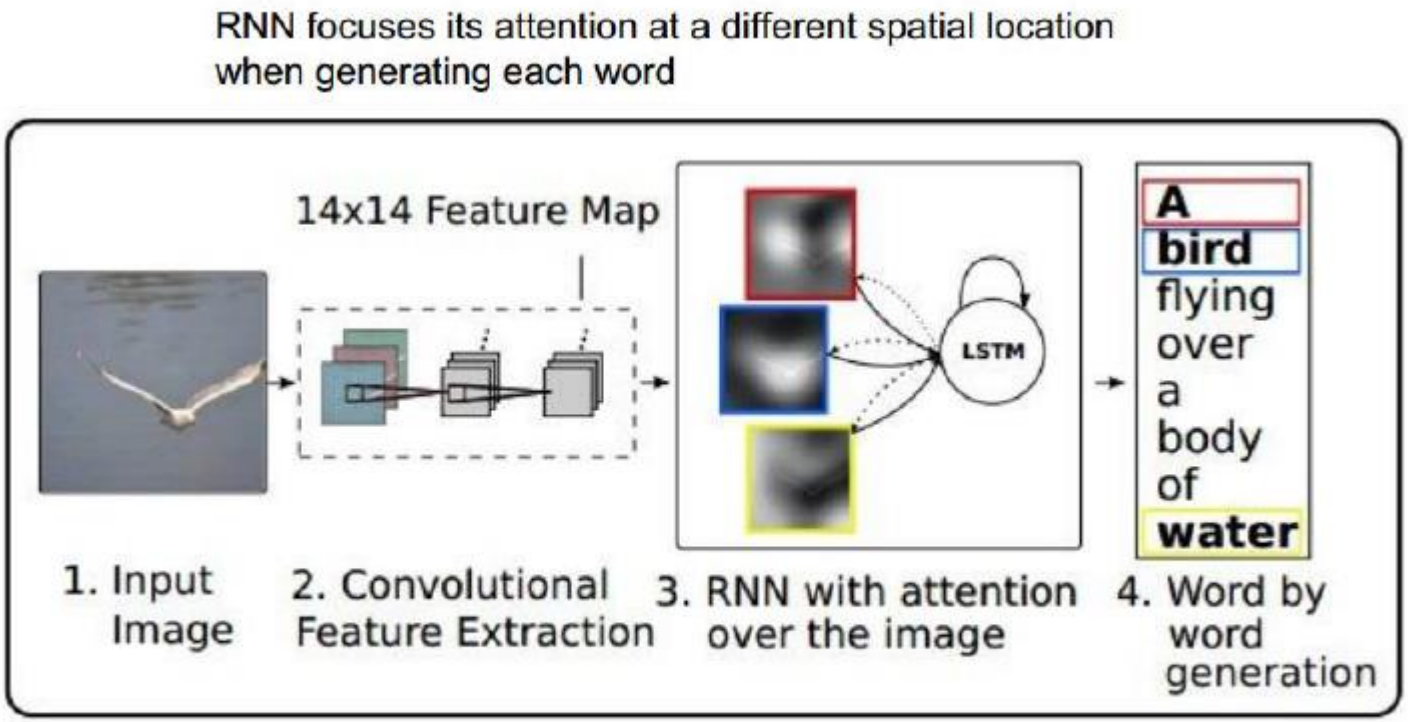


$$g(h_i, s_0) = h_i^T W_g s_0$$

$$e_i(1) = g(h_i, s_0)$$

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

Image Captioning with Attention



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

Image Captioning with Attention

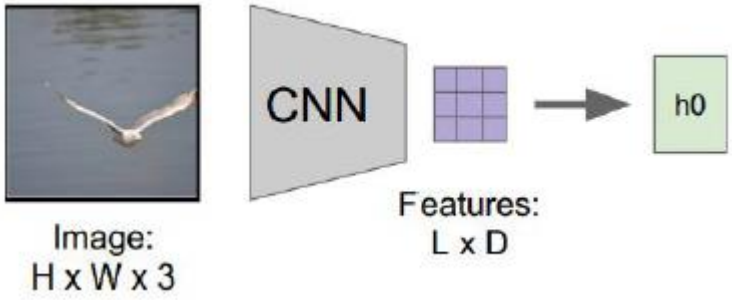
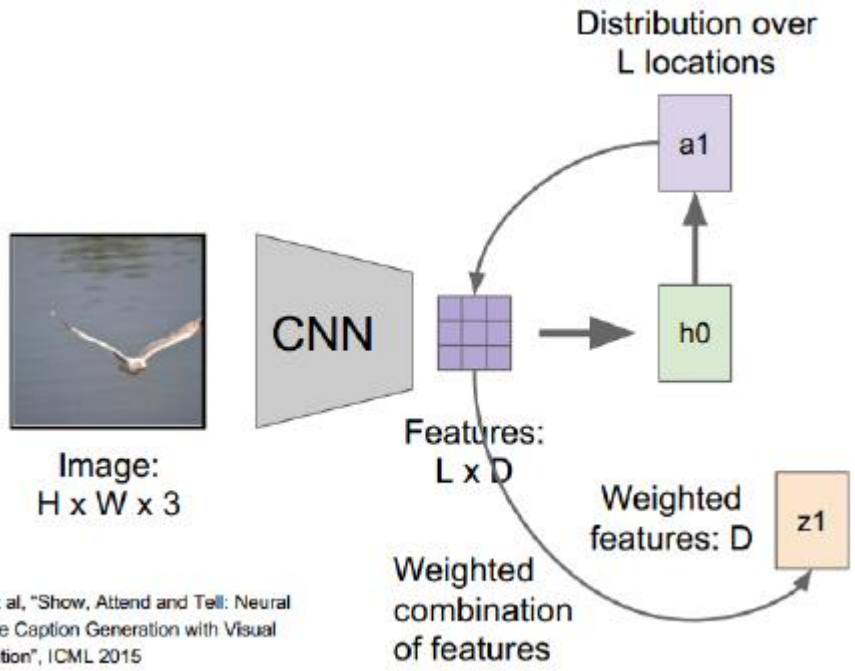
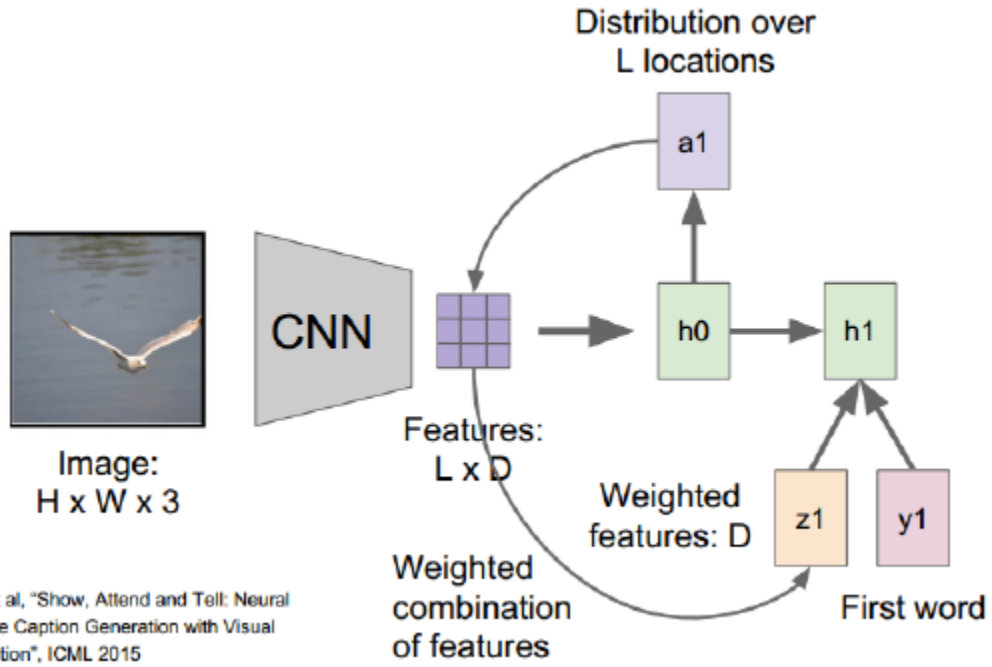


Image Captioning with Attention



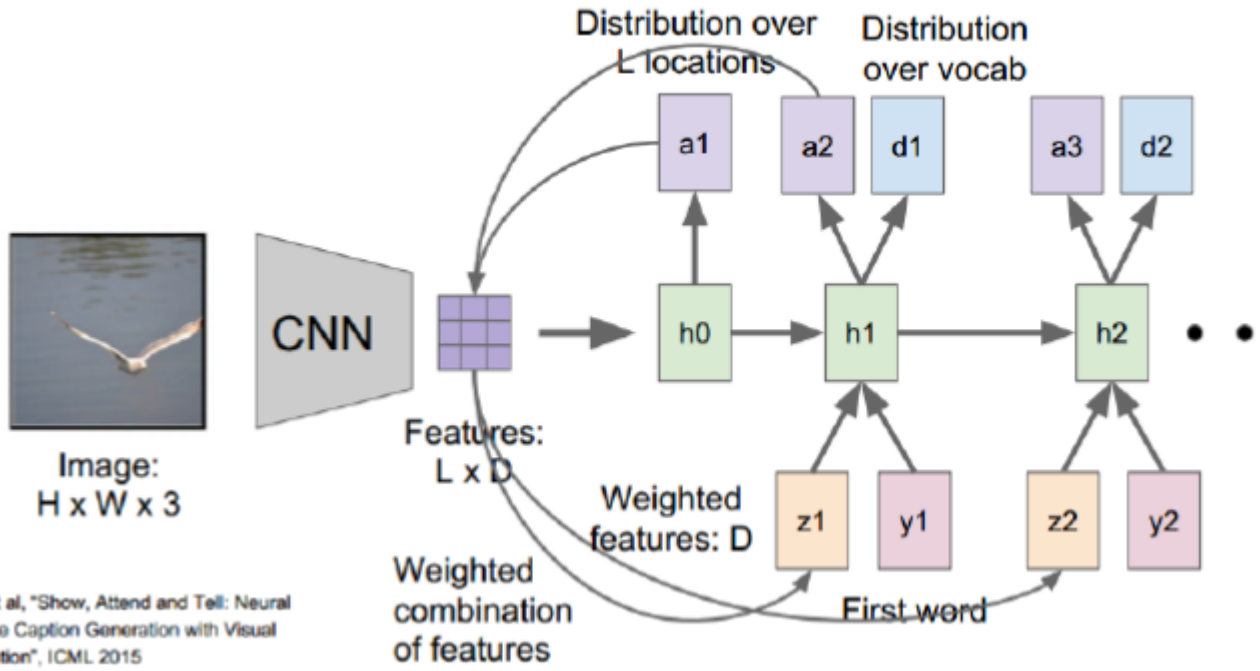
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

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

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