Recurrent Neural Nets
&
Visual Captioning

Lecture 17

Slides from: Dhruv Bhatra, Fei-Fei Li, Justin Johnson, Serena Yeung, Andrej Karpathy

Recurrent Neural Nets

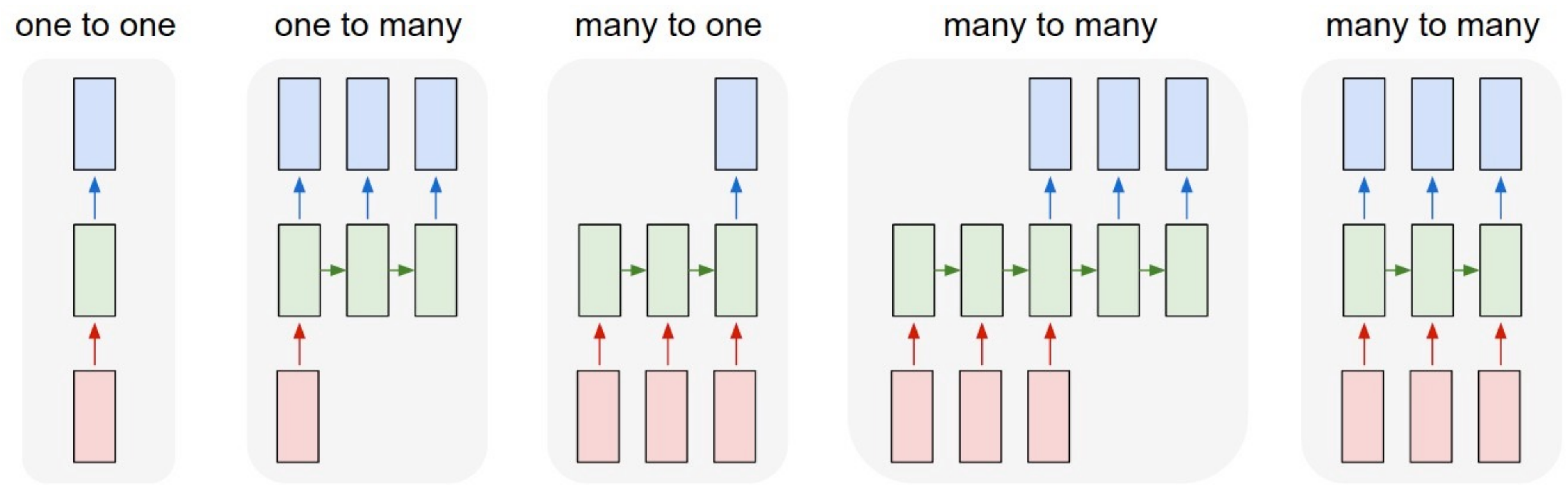
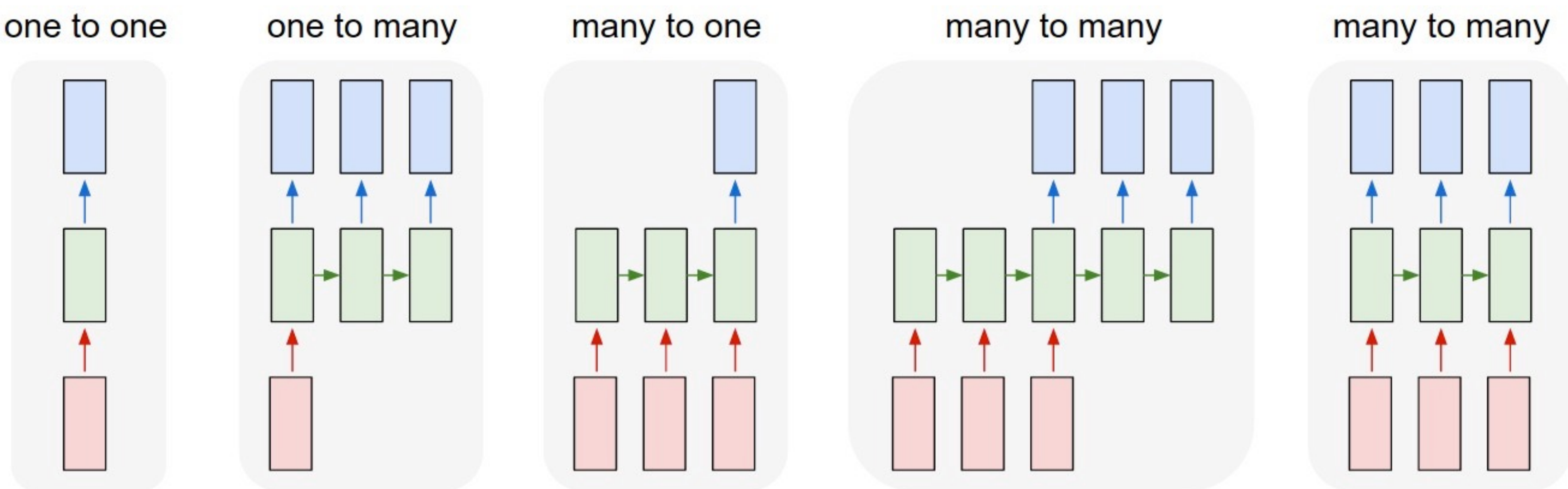


Image Credit: Andrej Karpathy

Recurrent Neural Nets



Input: No sequence

Output: No sequence

Example: “standard” classification / regression problems

Input: No sequence

Output: Sequence

Example: Im2Caption

Input: Sequence

Output: No sequence

Example: sentence classification, multiple-choice question answering

Input: Sequence

Output: Sequence

Example: machine translation, video captioning, open-ended question answering, video question answering

Synonyms

- Recurrent Neural Networks (RNNs)

- Types:
  - "Vanilla" RNNs
  - Long Short Term Memory (LSTMs)
  - Gated Recurrent Units (GRUs)
  - …

- Algorithms
  - BackProp Through Time (BPTT)

What's wrong with MLPs/ConvNets?

- Problem 1: Can't model sequences
  - Fixed-sized Inputs & Outputs
  - No temporal structure

- Problem 2: Pure feed-forward processing
  - No "memory", no feedback

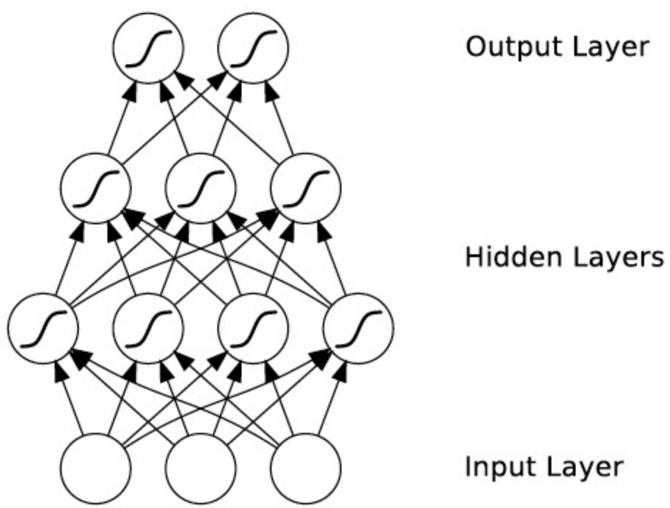


Image Credit: Alex Graves, book

Sequences are everywhere…

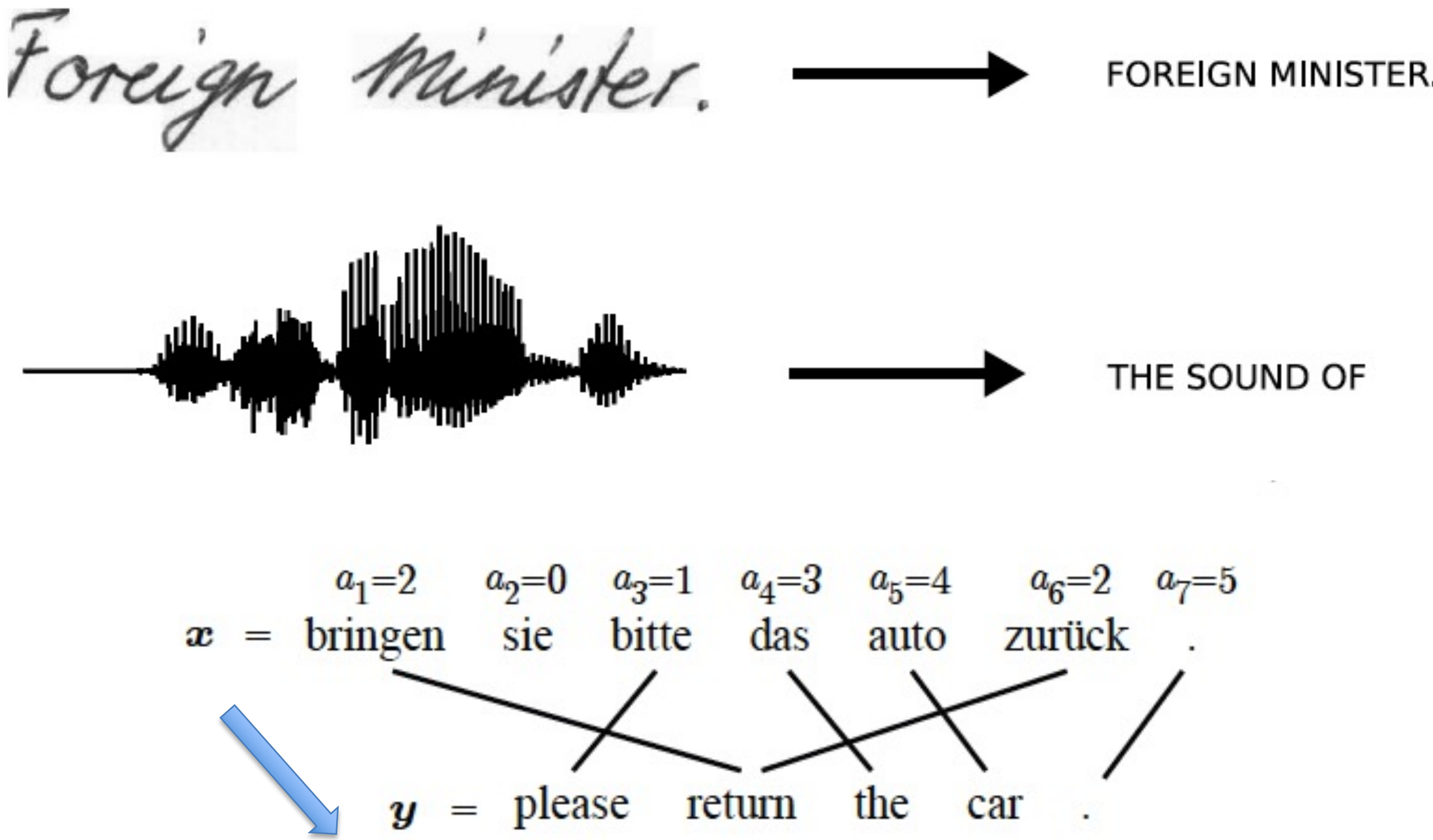
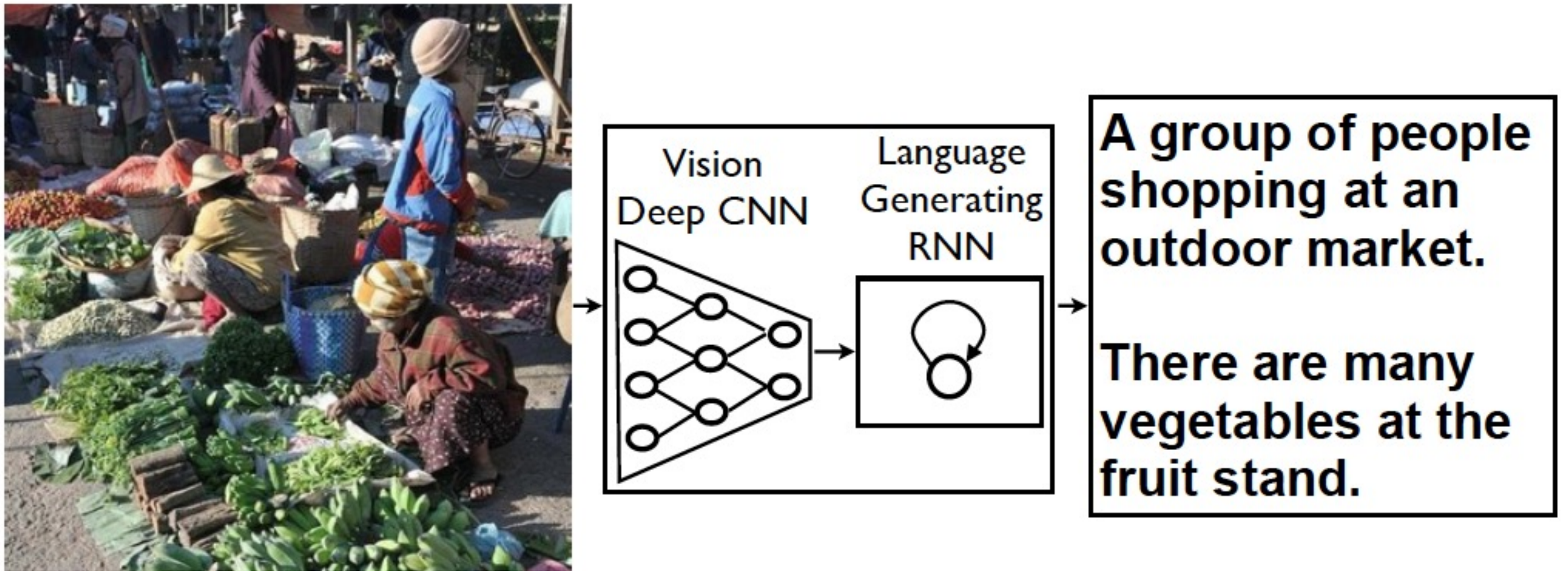
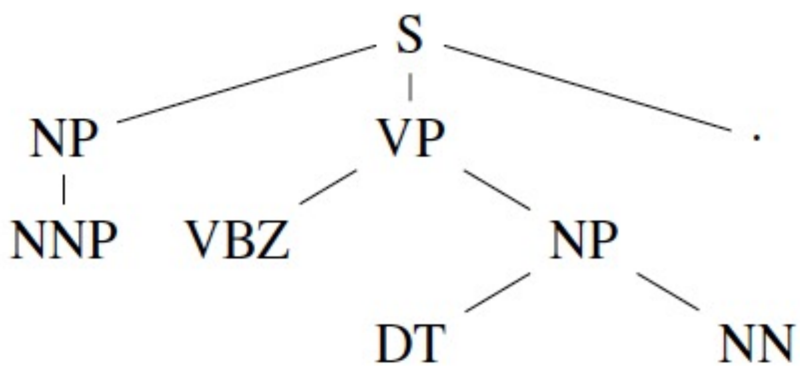


Image Credit: Alex Graves and Kevin Gimpel

Even where you might not expect a sequence…



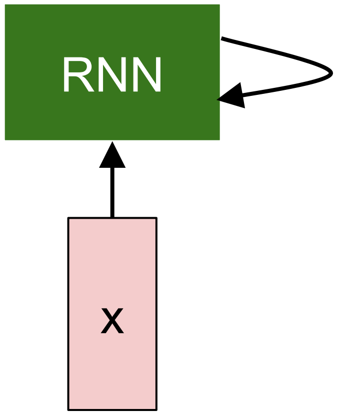
John has a dog . $\rightarrow$



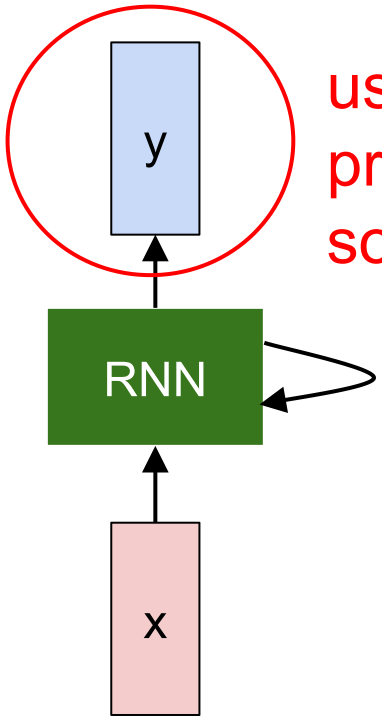
John has a dog . $\rightarrow$ (S (NP NNP $)_{NP}$ (VP VBZ (NP DT NN $)_{NP}$ $)_{VP}$. $)_{S}$

Image Credit: Vinyals et al.

Recurrent Neural Network



Recurrent Neural Network



usually want to predict a vector at some time steps

Recurrent Neural Network

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

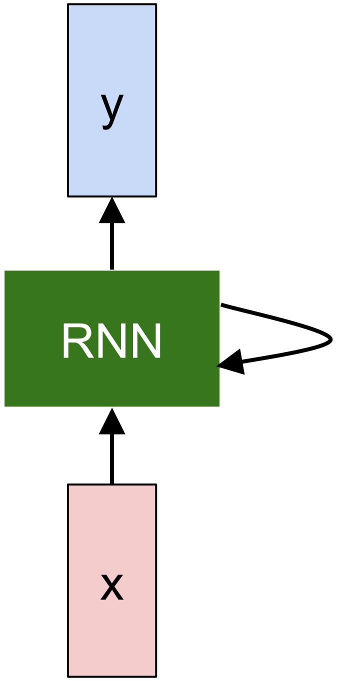
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W

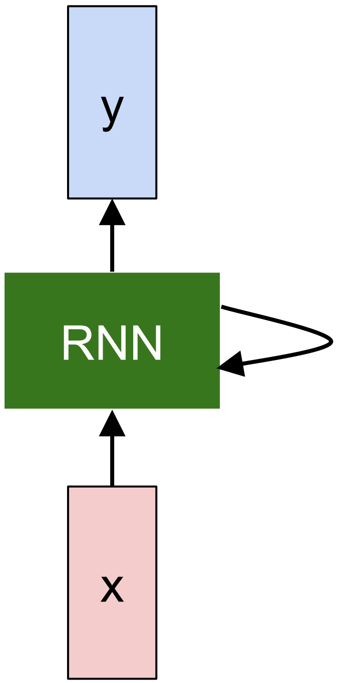


Recurrent Neural Network

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

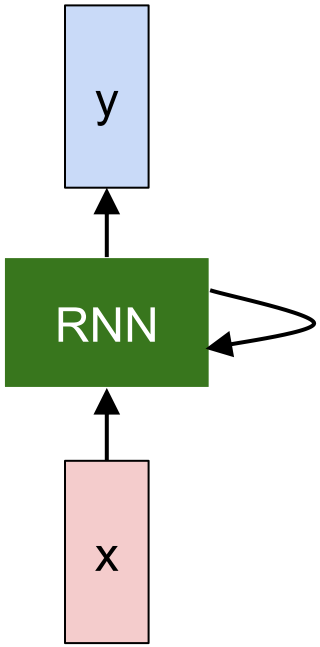
$$h_t = f_W(h_{t-1}, x_t)$$

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



(Vanilla) Recurrent Neural Network

The state consists of a single *“hidden”* vector **h**:

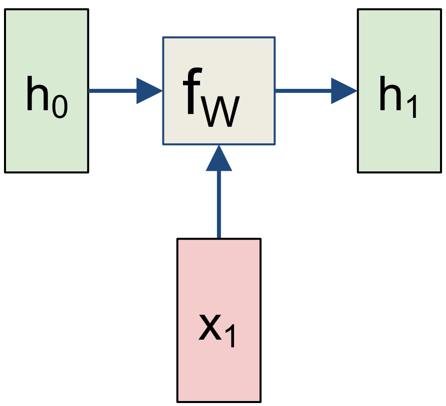


$$y_t = W_{hy} h_t + b_y$$

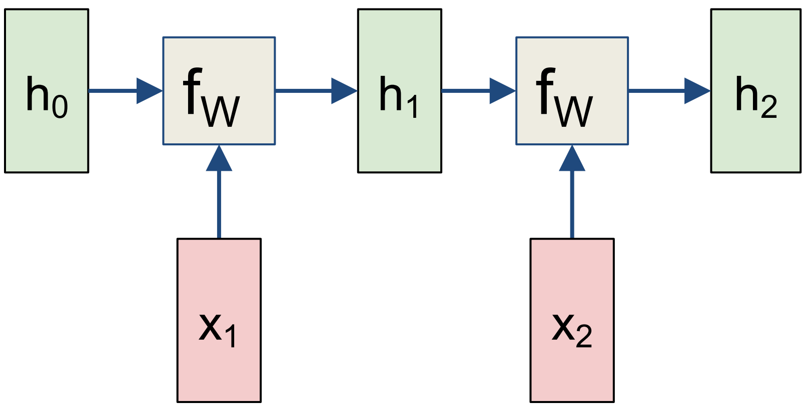
$$h_t = f_W(h_{t-1}, x_t)$$

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

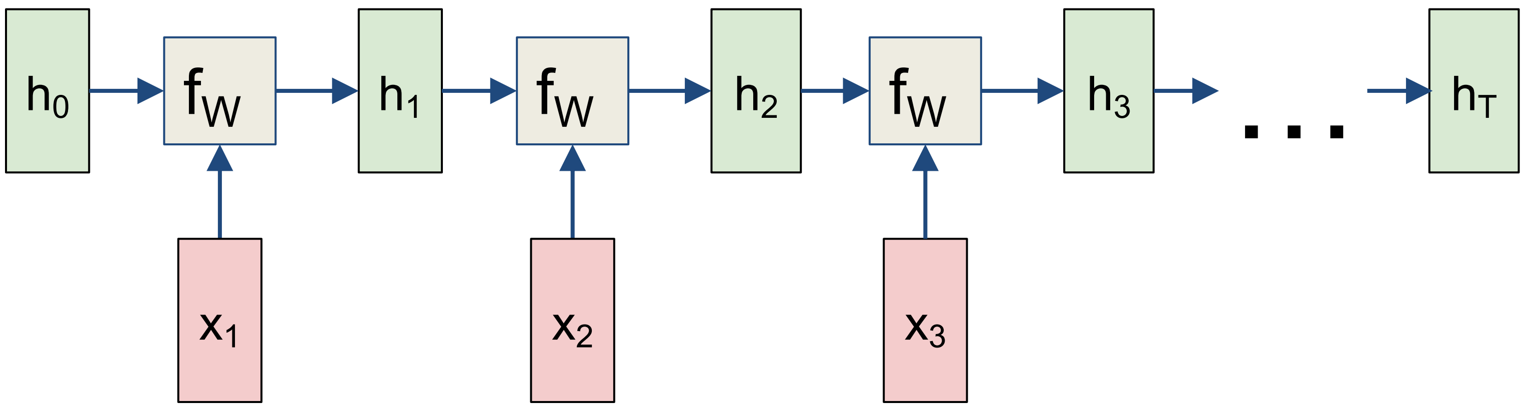
RNN: Computational Graph



RNN: Computational Graph

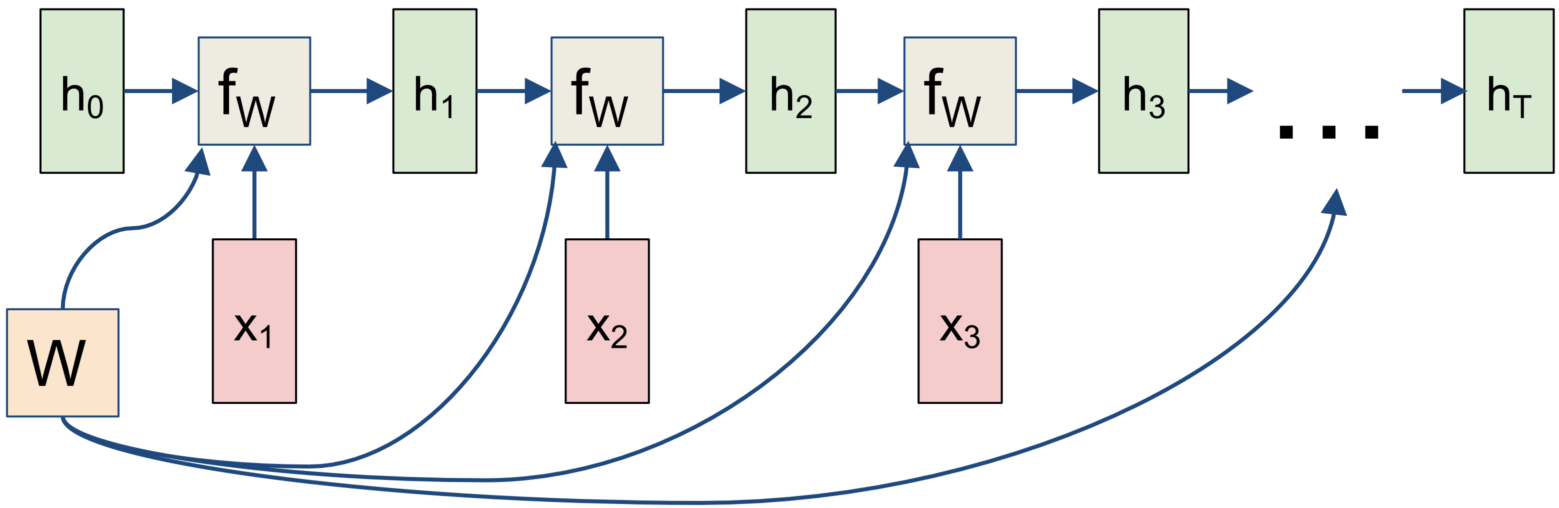


RNN: Computational Graph

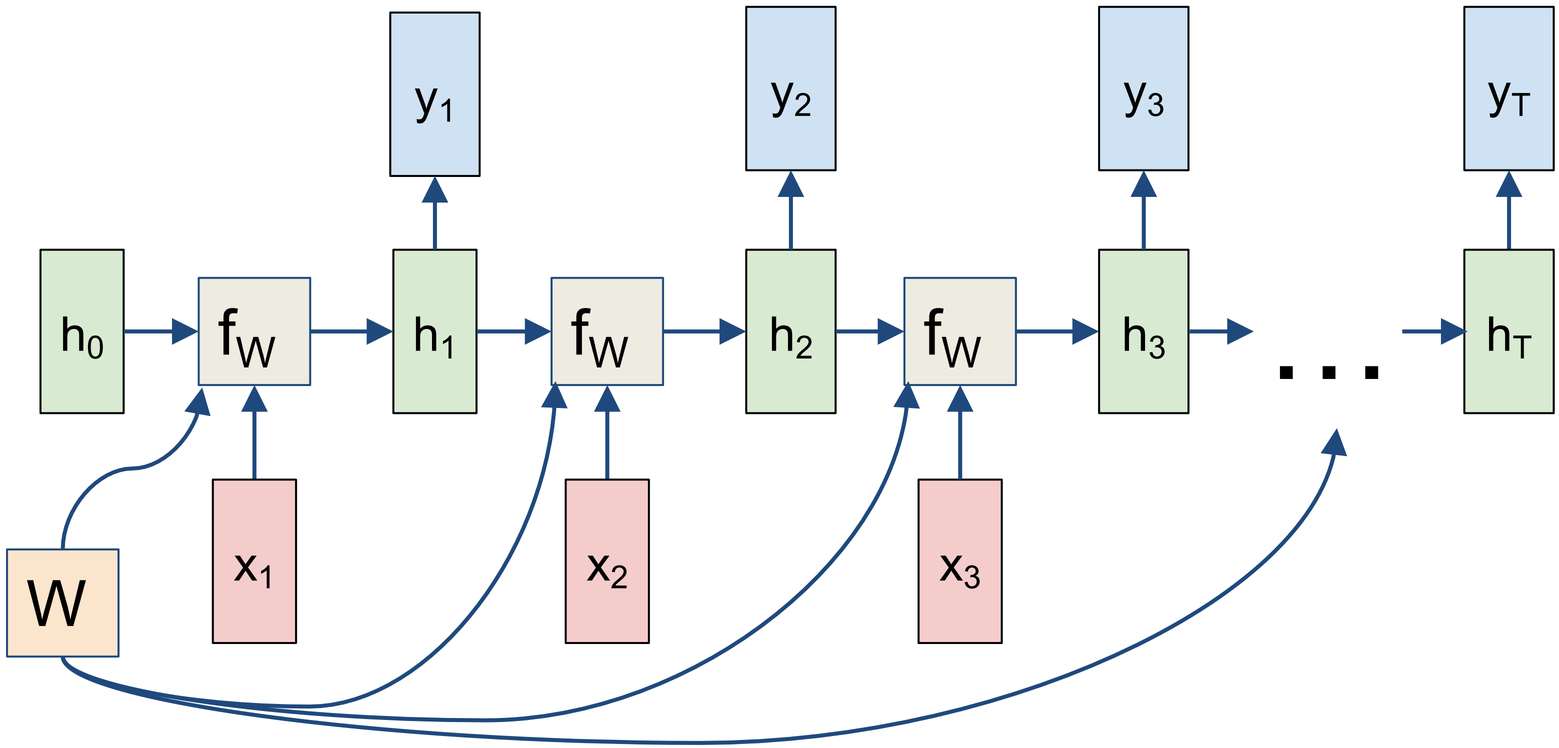


RNN: Computational Graph

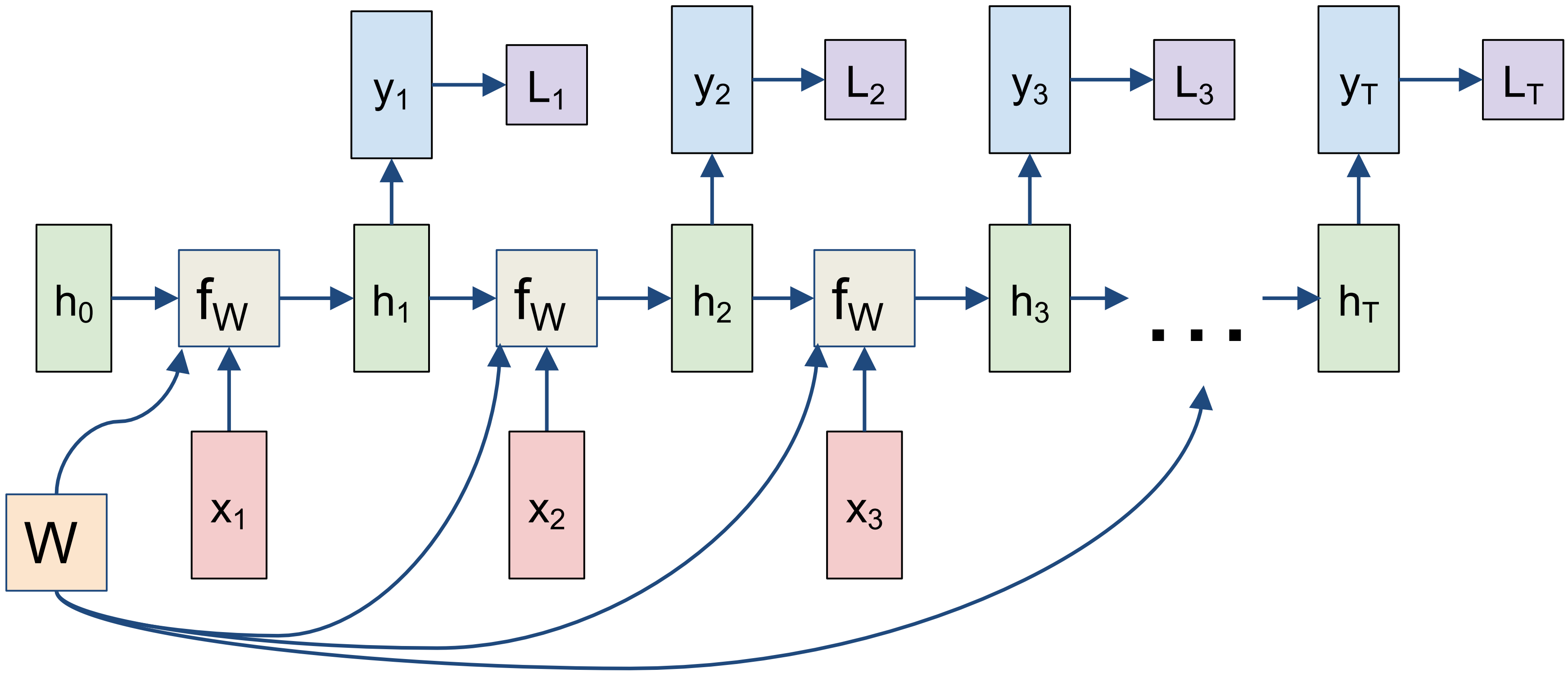
Re-use the same weight matrix at every time-step



RNN: Computational Graph: Many to Many

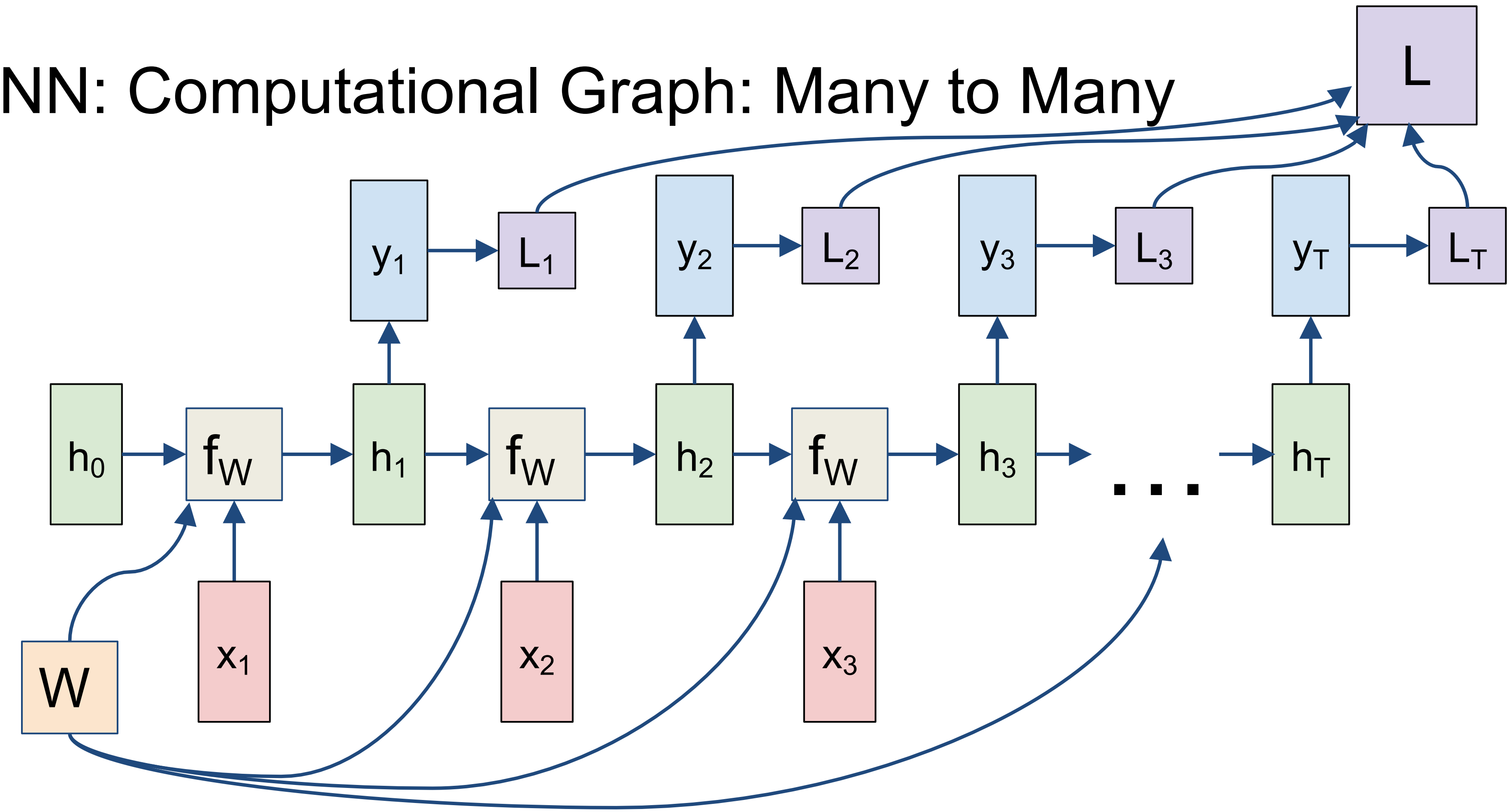


RNN: Computational Graph: Many to Many



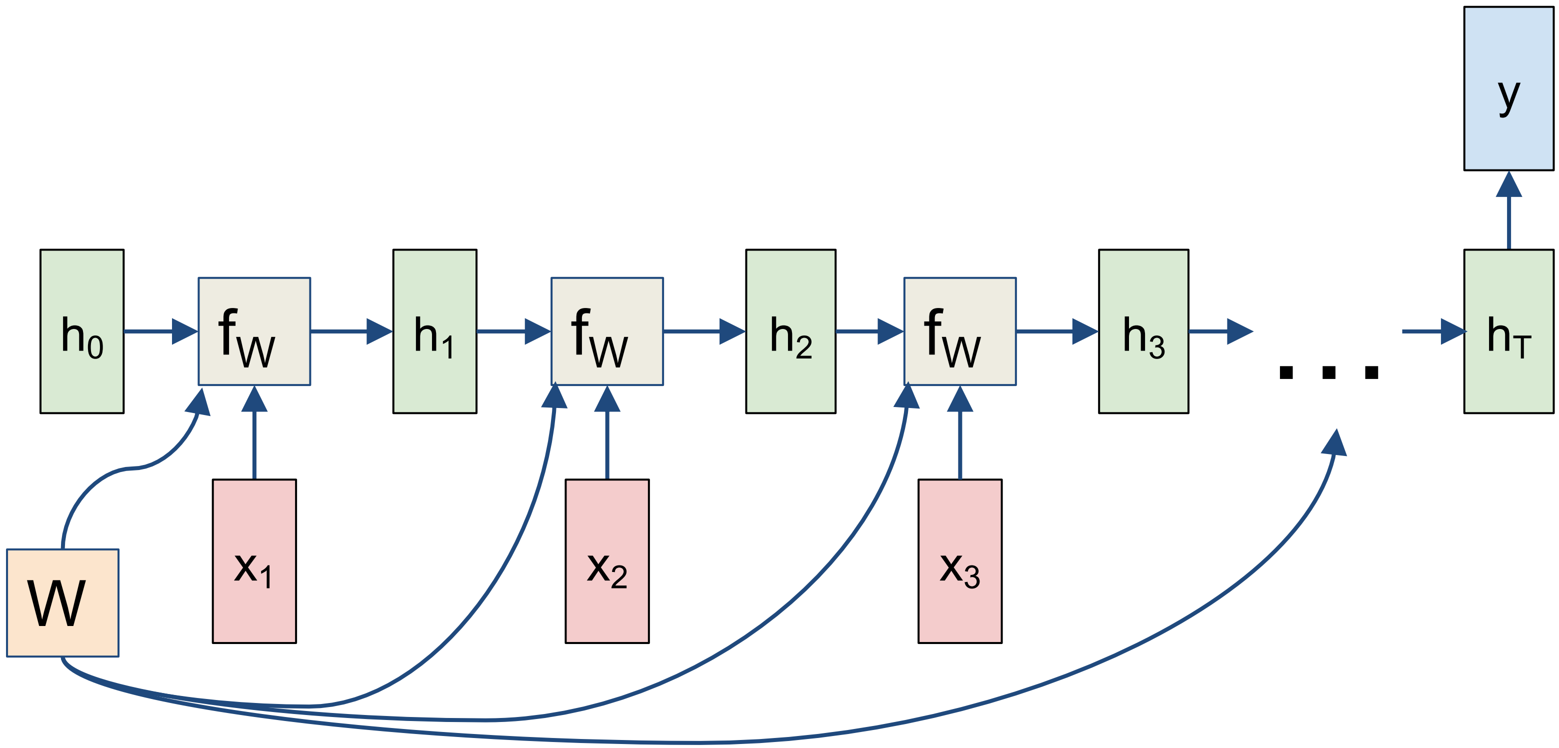
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

RNN: Computational Graph: Many to Many

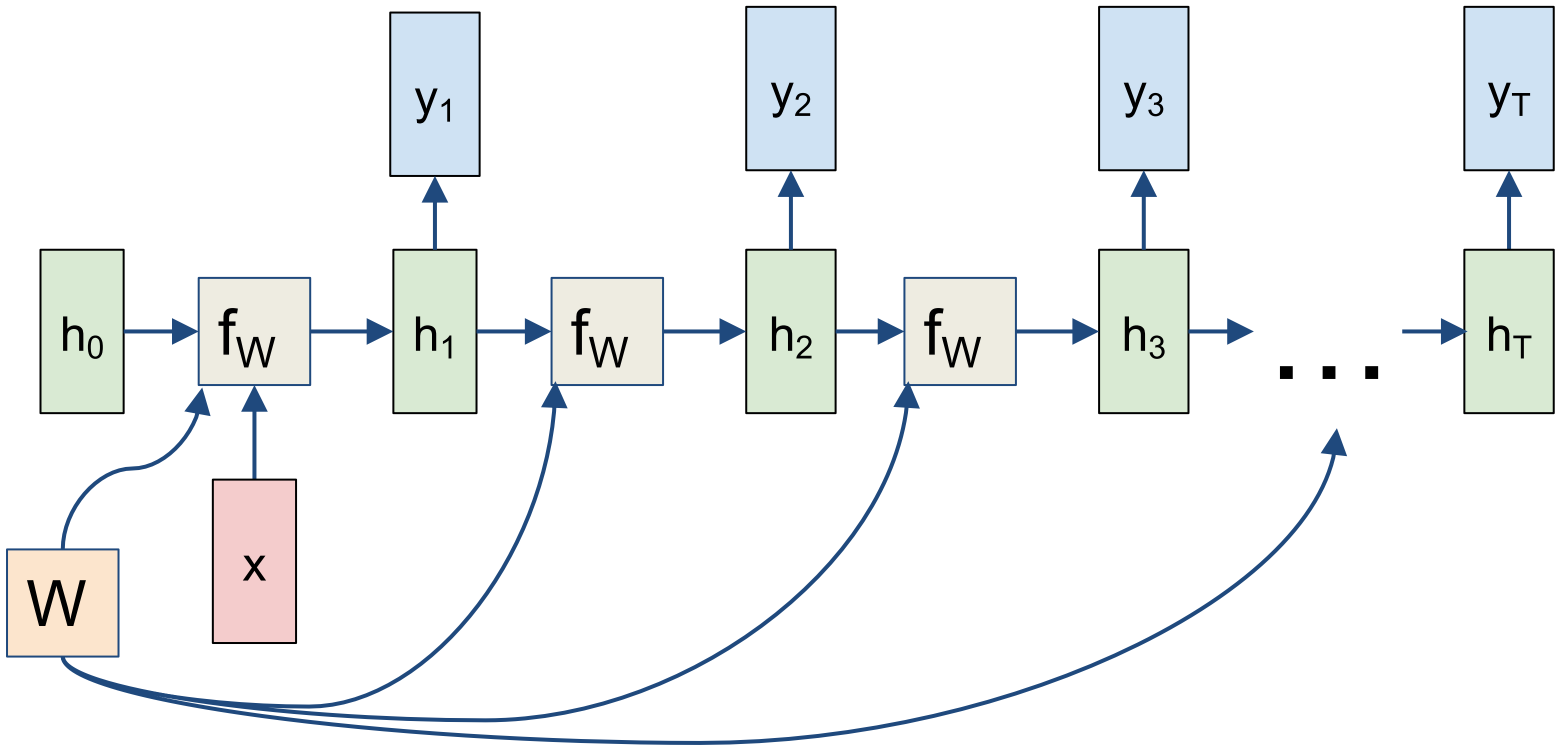


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

RNN: Computational Graph: Many to One

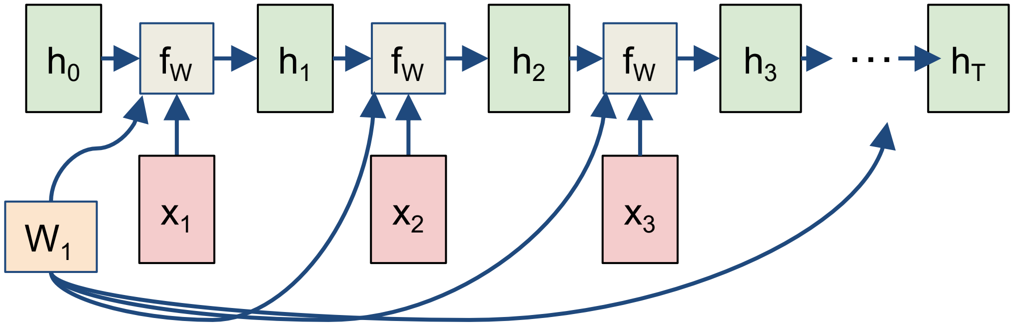


RNN: Computational Graph: One to Many

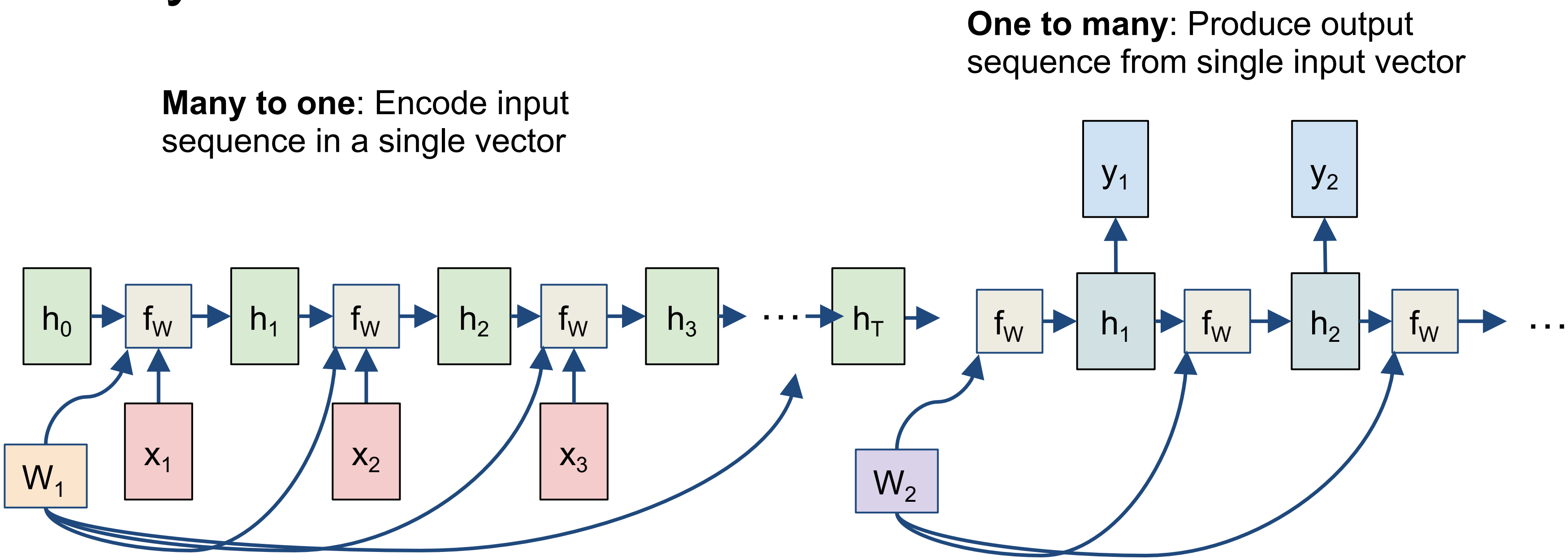


Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

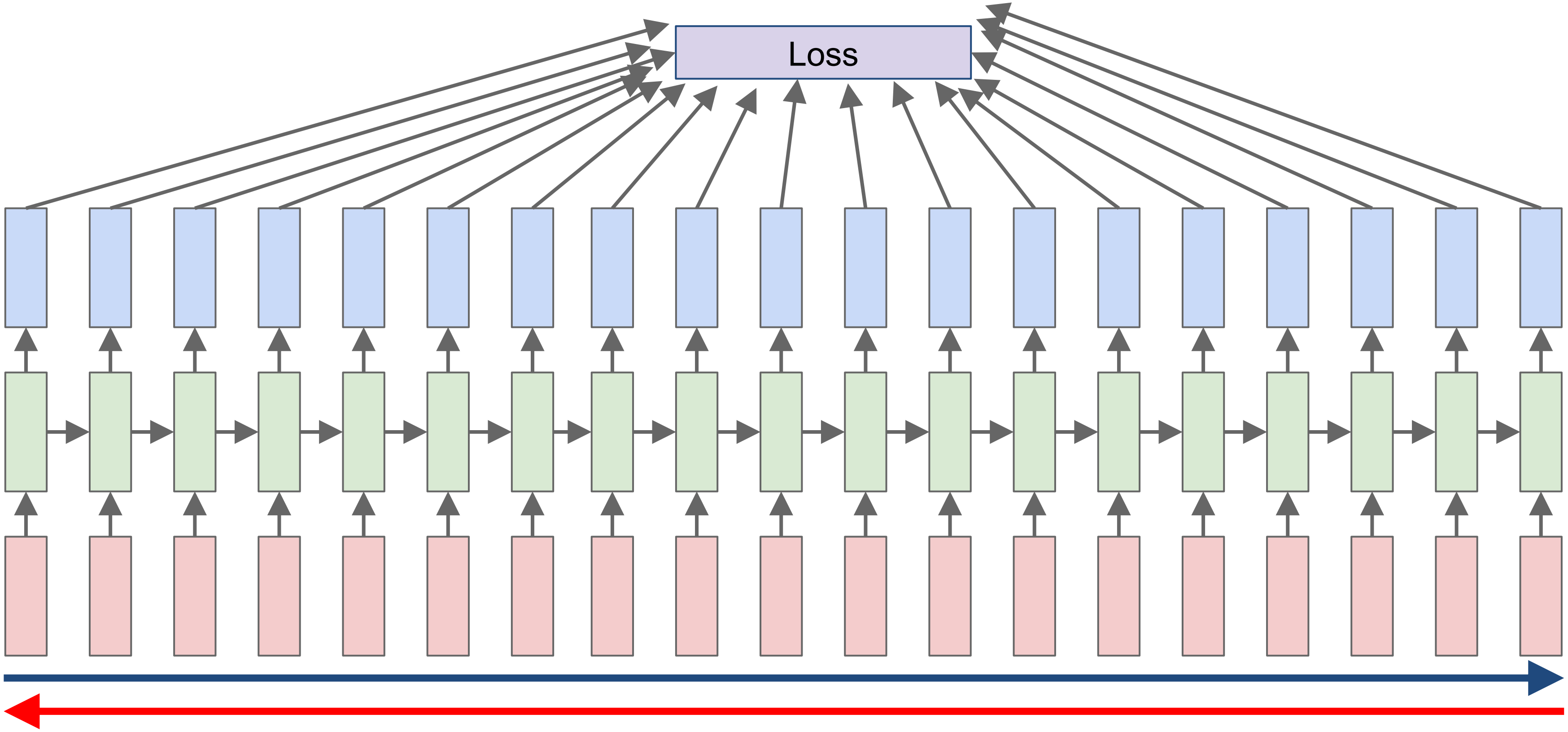


Sequence to Sequence: Many-to-one + one-to-many



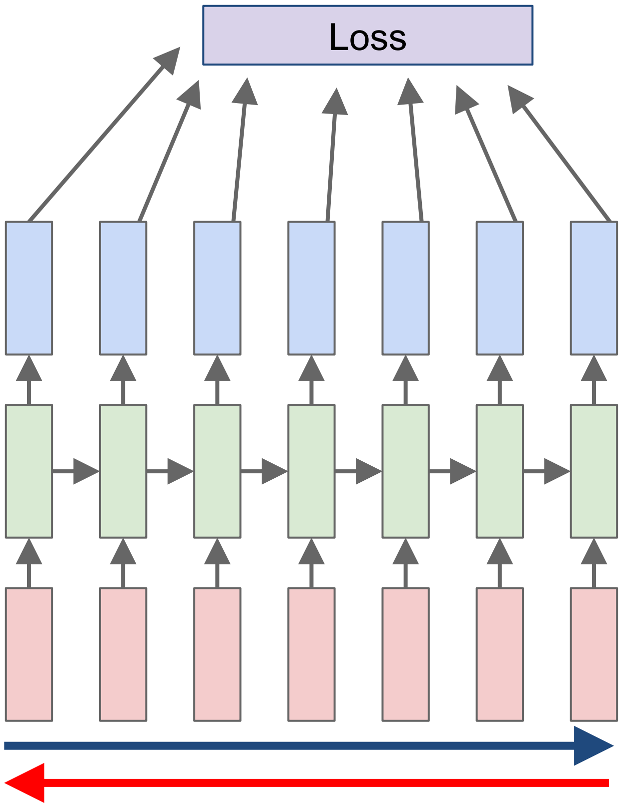
Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



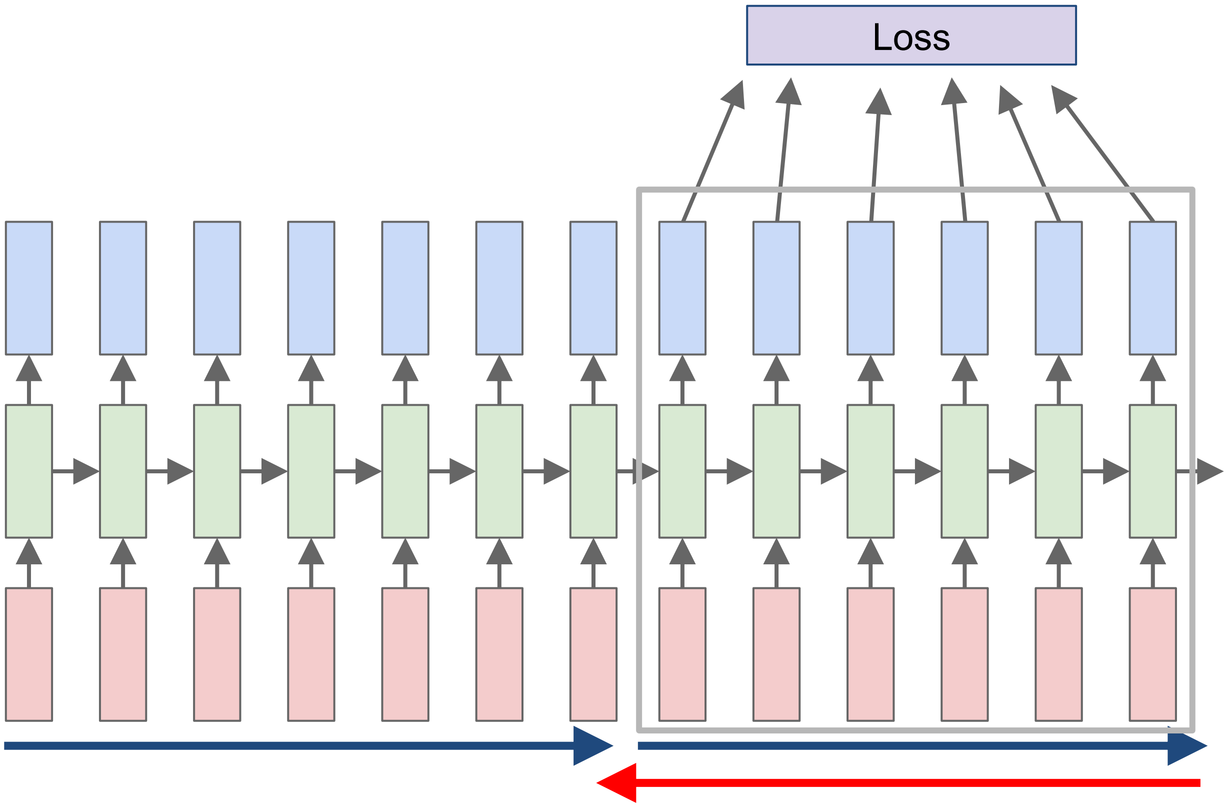
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Truncated Backpropagation through time



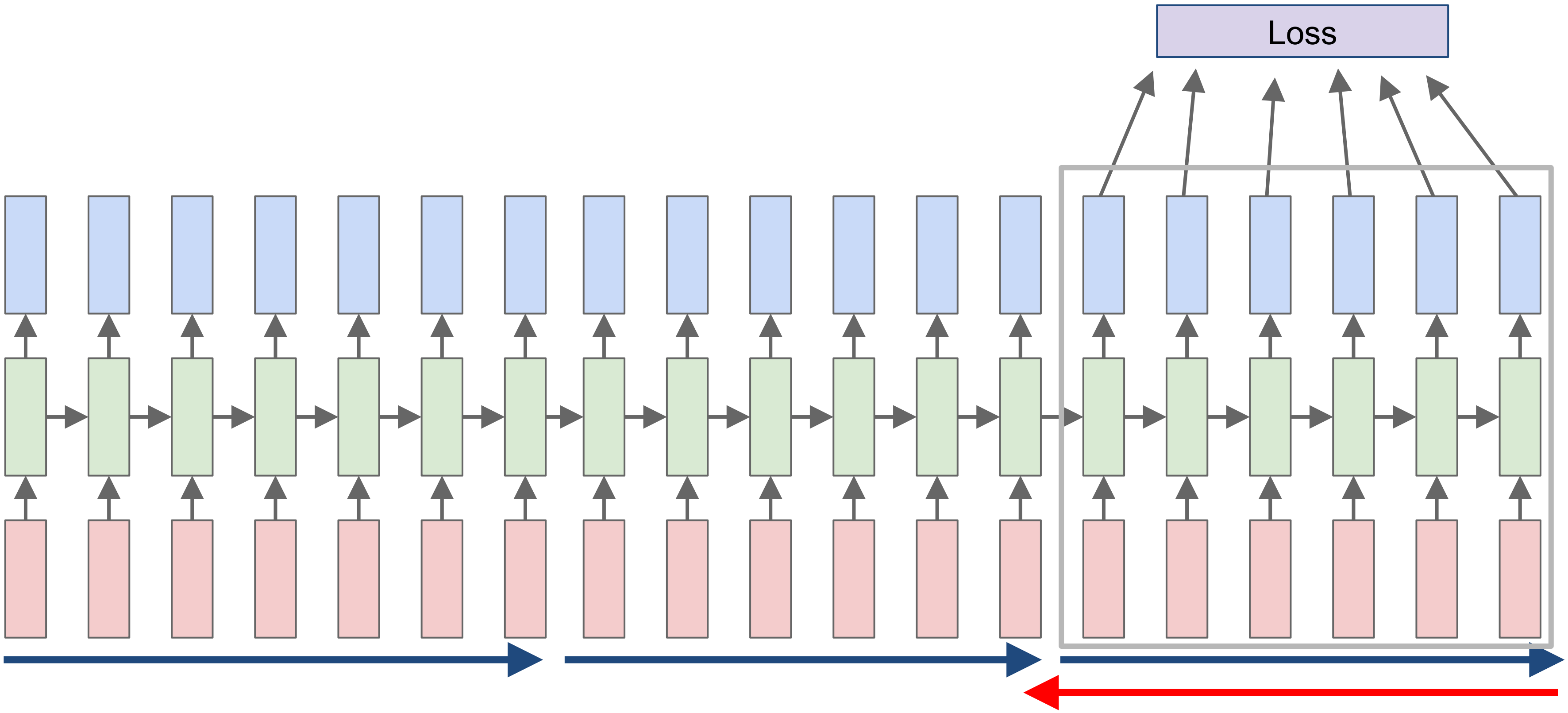
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation through time

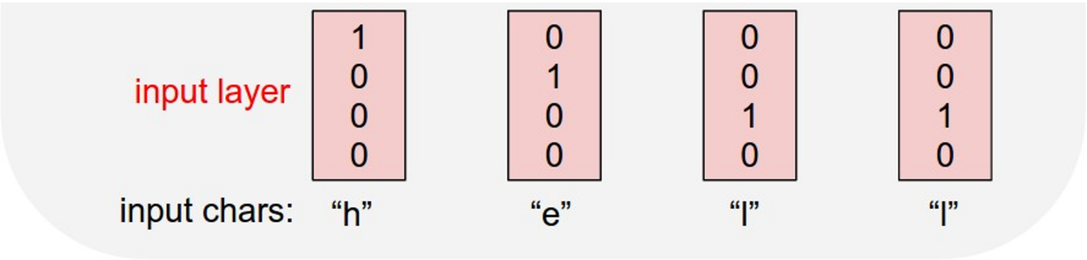


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

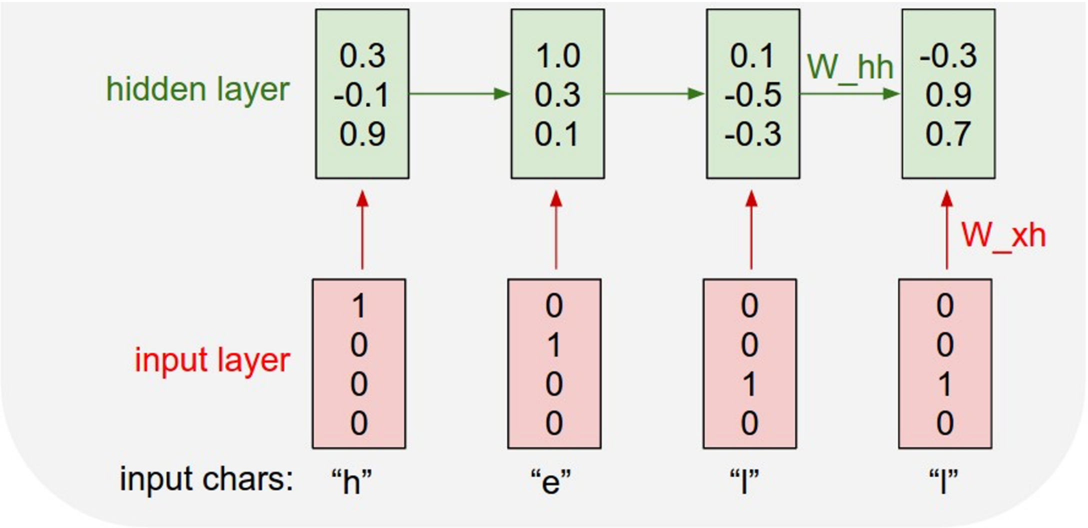


Example: Character-level Language Model

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

Vocabulary: [h,e,l,o]

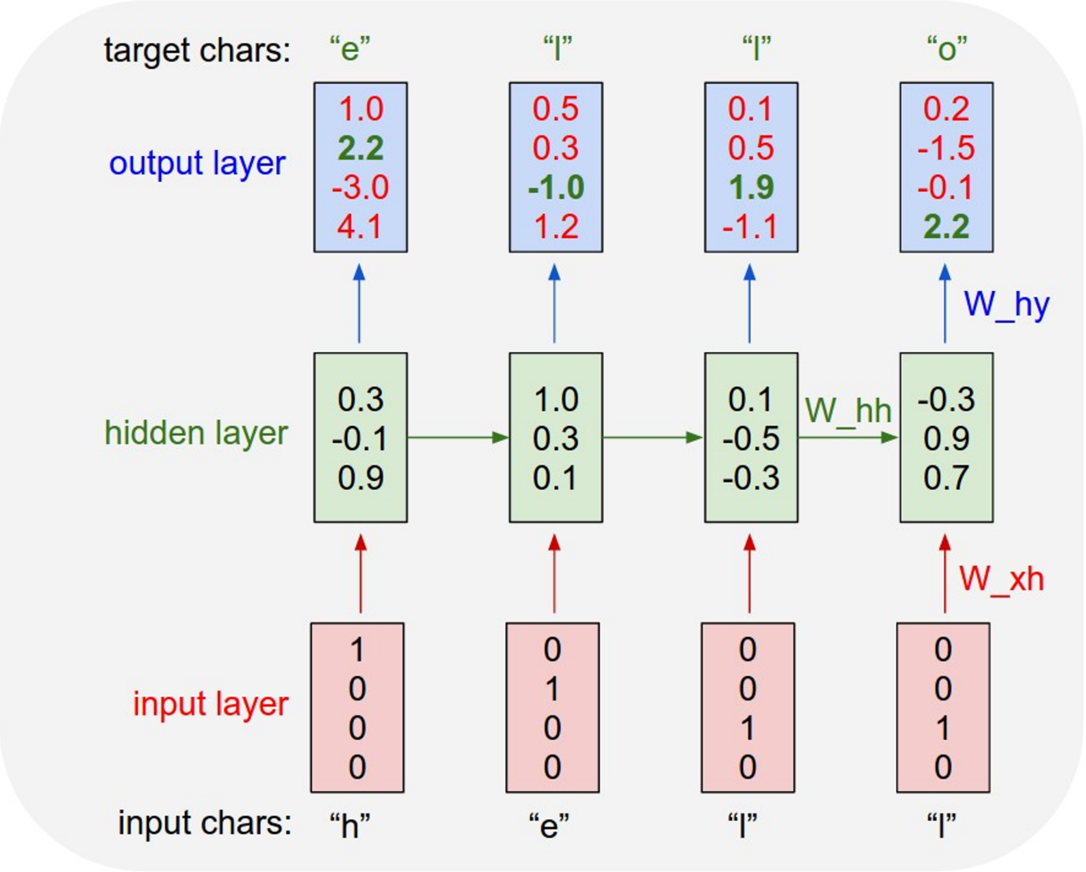
Example training sequence: **“hello”**



Example: Character-level Language Model

Vocabulary: [h,e,l,o]

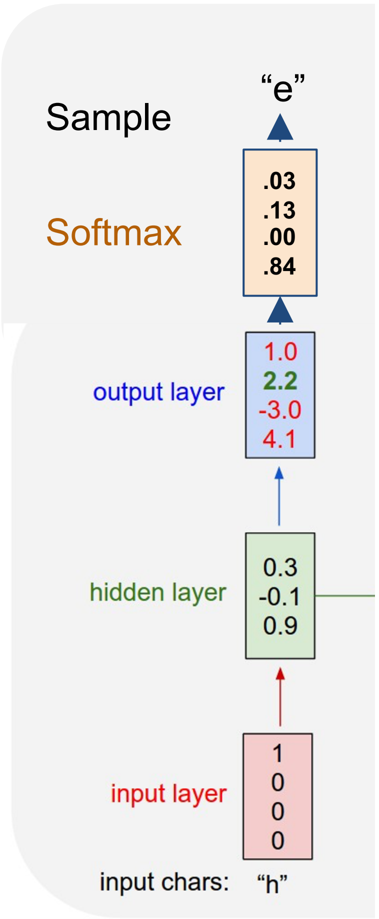
Example training sequence: **“hello”**



Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

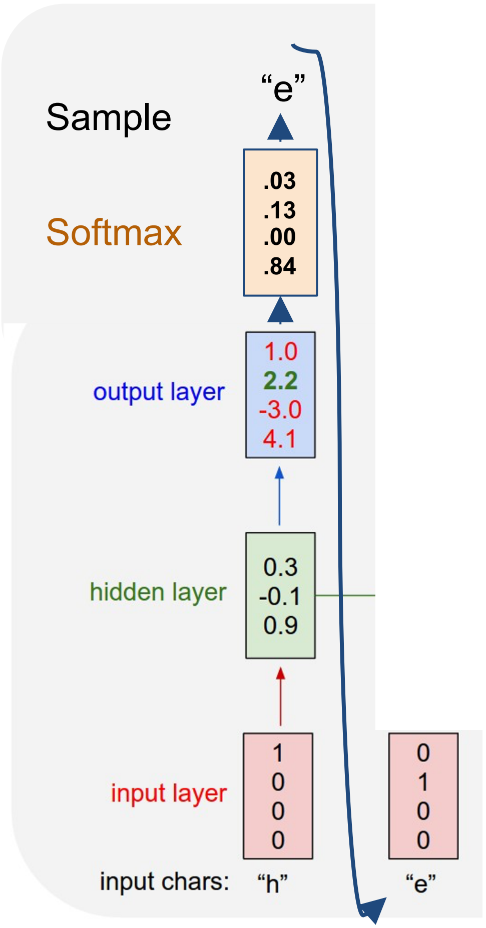
At test-time sample characters one at a time, feed back to model



Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

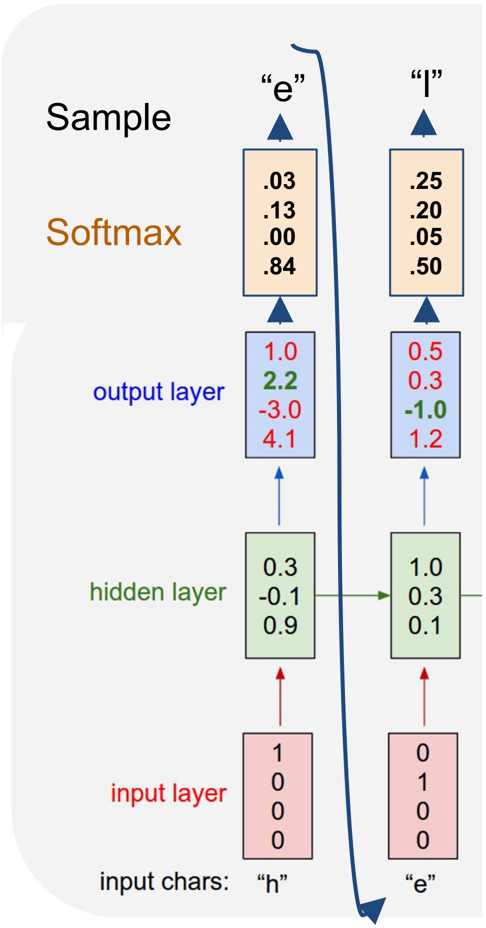
At test-time sample characters one at a time, feed back to model



Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

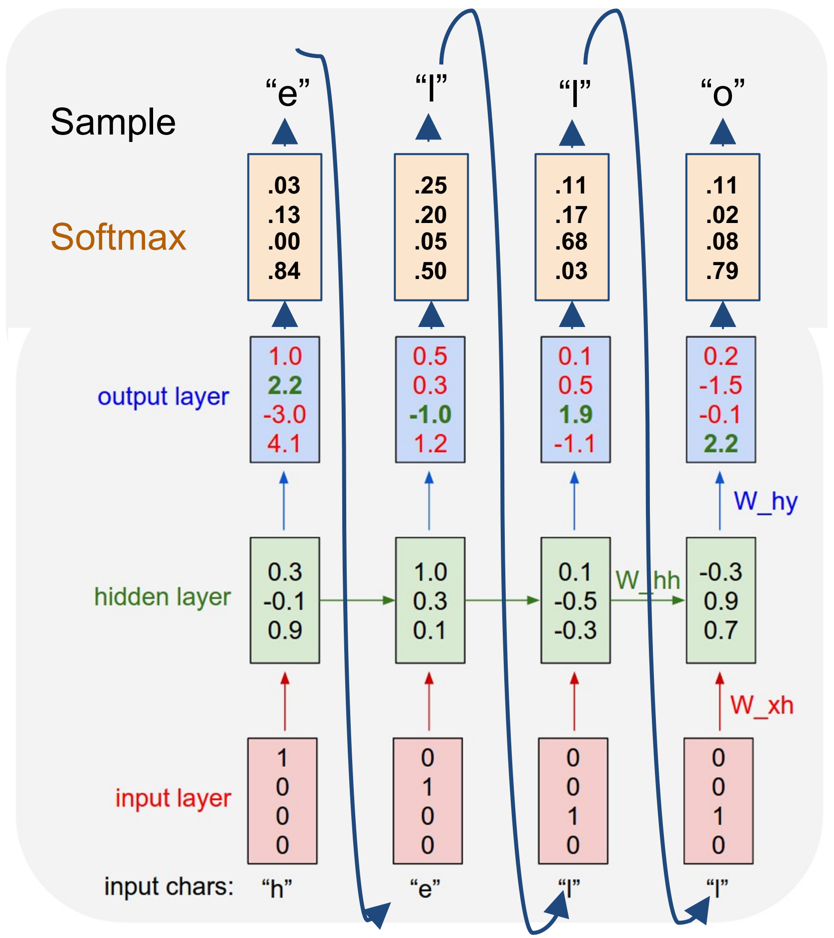
At test-time sample characters one at a time, feed back to model



Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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



min-char-rnn.py gist: 112 lines of Python

```python
"""
Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
BSD License
"""
import numpy as np

# data I/O
data = open('input.txt', 'r').read() # should be simple plain text file
chars = list(set(data))
data_size, vocab_size = len(data), len(chars)
print 'data has %d characters, %d unique.' % (data_size, vocab_size)
char_to_ix = { ch:i for i,ch in enumerate(chars) }
ix_to_char = { i:ch for i,ch in enumerate(chars) }

# hyperparameters
hidden_size = 100 # size of hidden layer of neurons
seq_length = 25 # number of steps to unroll the RNN for
learning_rate = 1e-1

# model parameters
Wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
bh = np.zeros((hidden_size, 1)) # hidden bias
by = np.zeros((vocab_size, 1)) # output bias

def lossFun(inputs, targets, hprev):
  """
  inputs,targets are both list of integers.
  hprev is Hx1 array of initial hidden state
  returns the loss, gradients on model parameters, and last hidden state
  """
  xs, hs, ys, ps = {}, {}, {}, {}
  hs[-1] = np.copy(hprev)
  loss = 0
  # forward pass
  for t in xrange(len(inputs)):
    xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
    xs[t][inputs[t]] = 1
    hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state
    ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
    ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
    loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
  # backward pass: compute gradients going backwards
  dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
  dbh, dby = np.zeros_like(bh), np.zeros_like(by)
  dhnext = np.zeros_like(hs[0])
  for t in reversed(xrange(len(inputs))):
    dy = np.copy(ps[t])
    dy[targets[t]] -= 1 # backprop into y
    dWhy += np.dot(dy, hs[t].T)
    dby += dy
    dh = np.dot(Why.T, dy) + dhnext # backprop into h
    dhraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
    dbh += dhraw
    dWxh += np.dot(dhraw, xs[t].T)
    dWhh += np.dot(dhraw, hs[t-1].T)
    dhnext = np.dot(Whh.T, dhraw)
  for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
    np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
  return loss, dWxh, dWhh, dWhy, dbh, dby, hs[len(inputs)-1]
```

```python
def sample(h, seed_ix, n):
  """
  sample a sequence of integers from the model
  h is memory state, seed_ix is seed letter for first time step
  """
  x = np.zeros((vocab_size, 1))
  x[seed_ix] = 1
  ixes = []
  for t in xrange(n):
    h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
    y = np.dot(Why, h) + by
    p = np.exp(y) / np.sum(np.exp(y))
    ix = np.random.choice(range(vocab_size), p=p.ravel())
    x = np.zeros((vocab_size, 1))
    x[ix] = 1
    ixes.append(ix)
  return ixes

n, p = 0, 0
mWxh, mWhh, mWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
while True:
  # prepare inputs (we're sweeping from left to right in steps seq_length long)
  if p+seq_length+1 >= len(data) or n == 0:
    hprev = np.zeros((hidden_size,1)) # reset RNN memory
    p = 0 # go from start of data
  inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
  targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]

  # sample from the model now and then
  if n % 100 == 0:
    sample_ix = sample(hprev, inputs[0], 200)
    txt = ''.join(ix_to_char[ix] for ix in sample_ix)
    print '----\n %s \n----' % (txt, )

  # forward seq_length characters through the net and fetch gradient
  loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
  smooth_loss = smooth_loss * 0.999 + loss * 0.001
  if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress

  # perform parameter update with Adagrad
  for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
                                [dWxh, dWhh, dWhy, dbh, dby],
                                [mWxh, mWhh, mWhy, mbh, mby]):
    mem += dparam * dparam
    param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update

  p += seq_length # move data pointer
  n += 1 # iteration counter
```

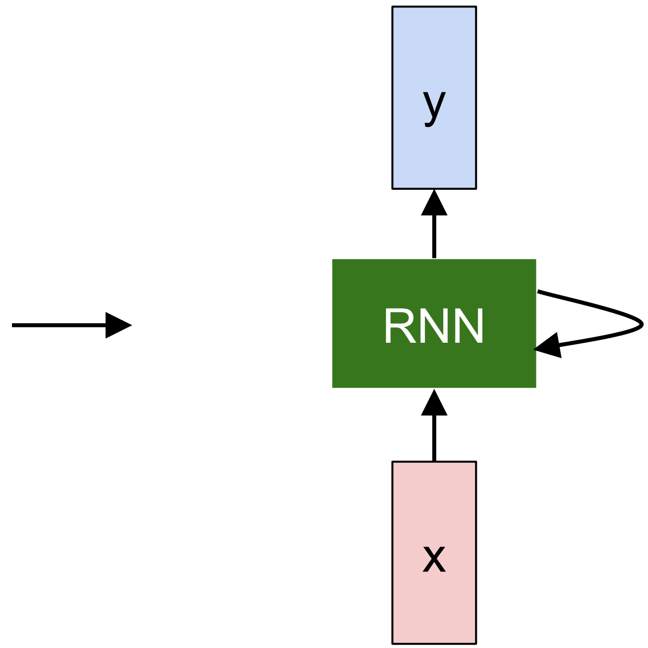
(https://gist.github.com/karpathy/d4dee566867f8291f086)

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
  Pity the world, or else this glutton be,
  To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
  This were to be new made when thou art old,
  And see thy blood warm when thou feel'st it cold.



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.
```

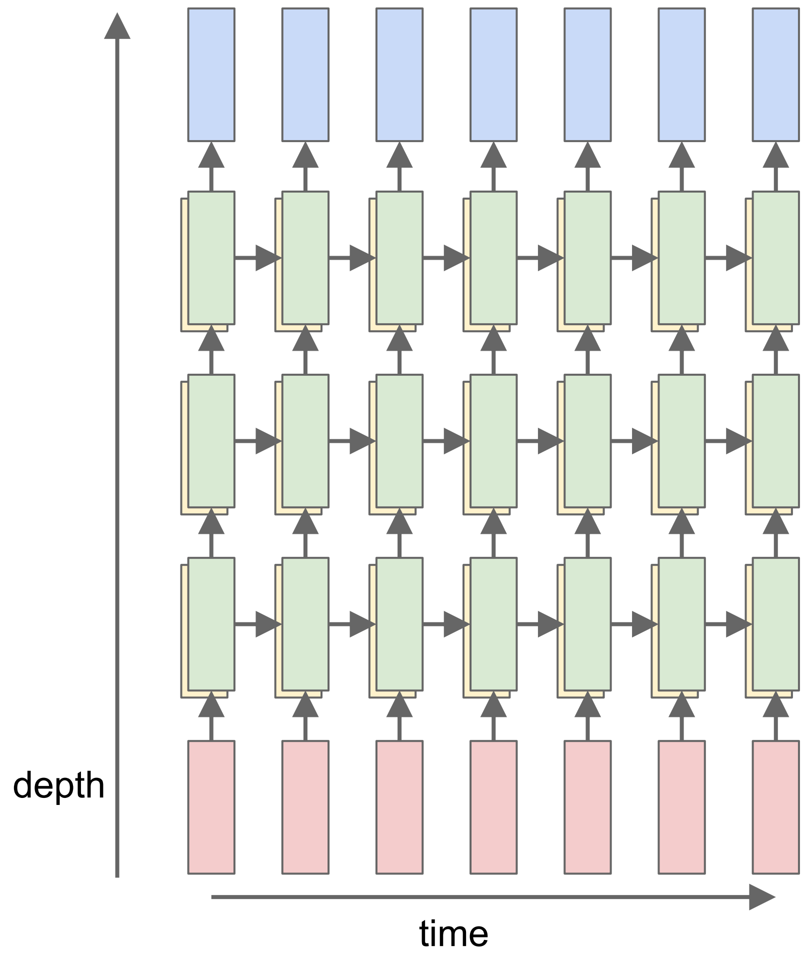
train more

```
"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.
```

Multilayer RNNs

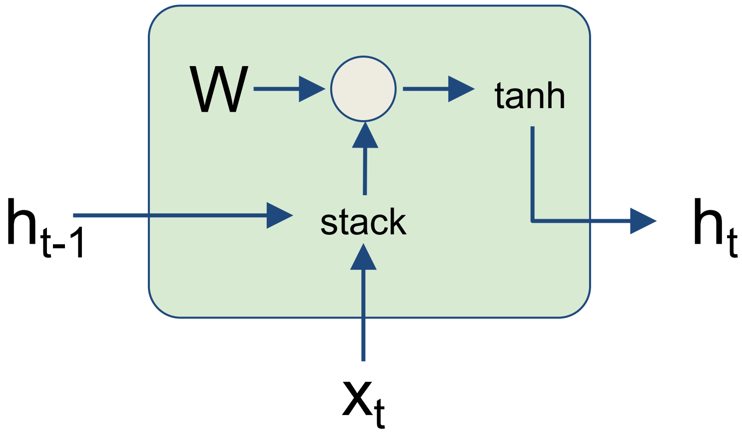
$$\boxed{h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}}$$

$h \in \mathbb{R}^n$ $\qquad W^l \quad [n \times 2n]$



Vanilla RNN Gradient Flow

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013

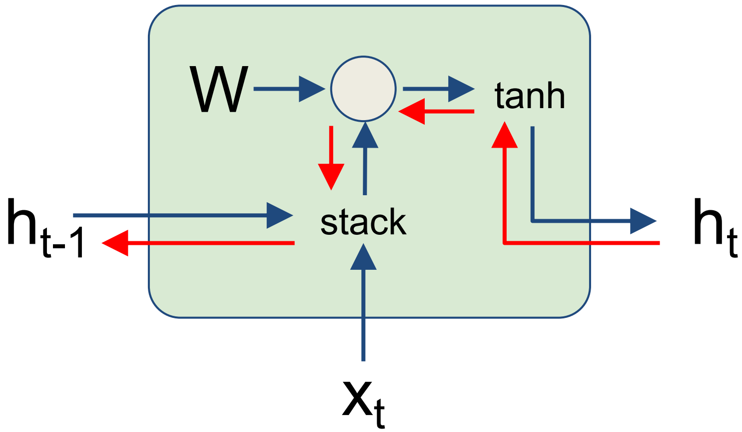


$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

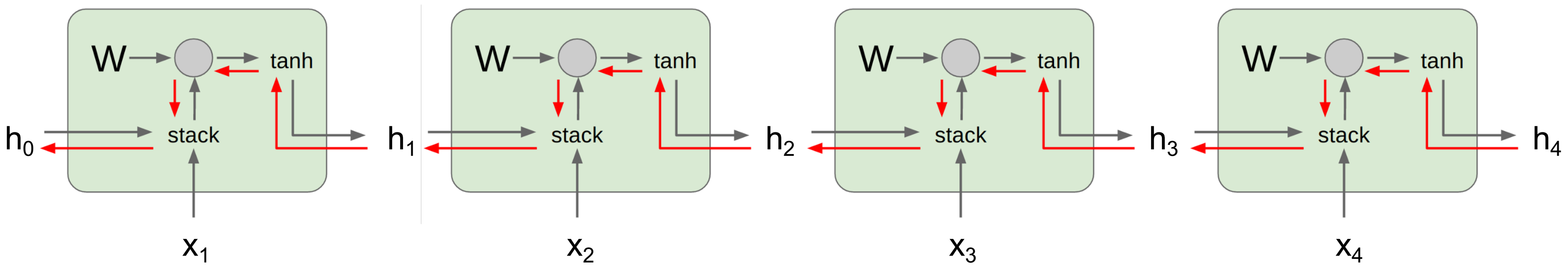
Backpropagation from h_t to h_{t-1} multiplies by W (actually W_{hh}^T)



$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

Vanilla RNN Gradient Flow

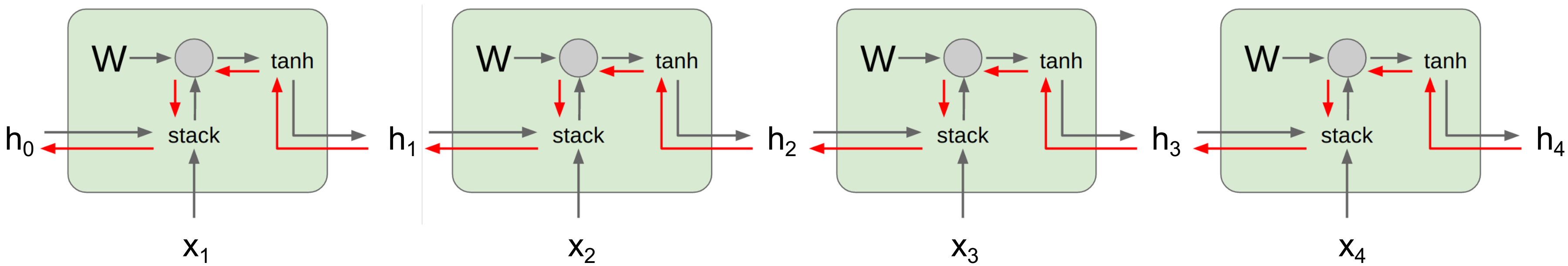
Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Vanilla RNN Gradient Flow

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



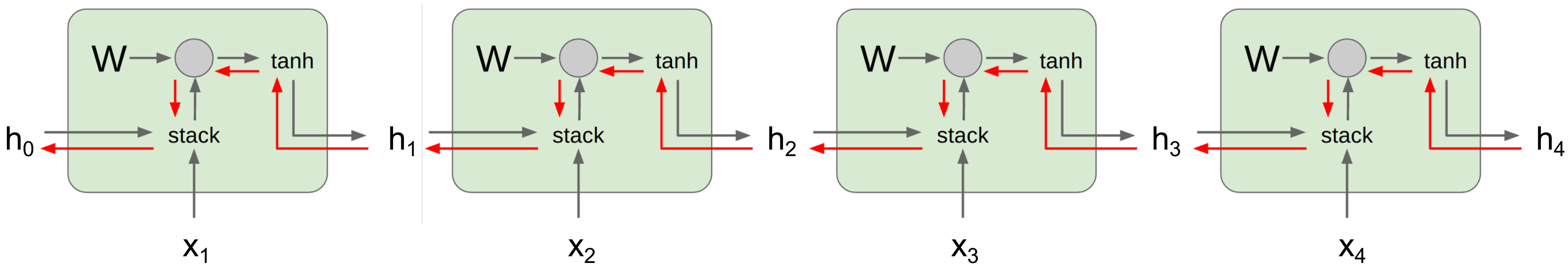
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Vanilla RNN Gradient Flow

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

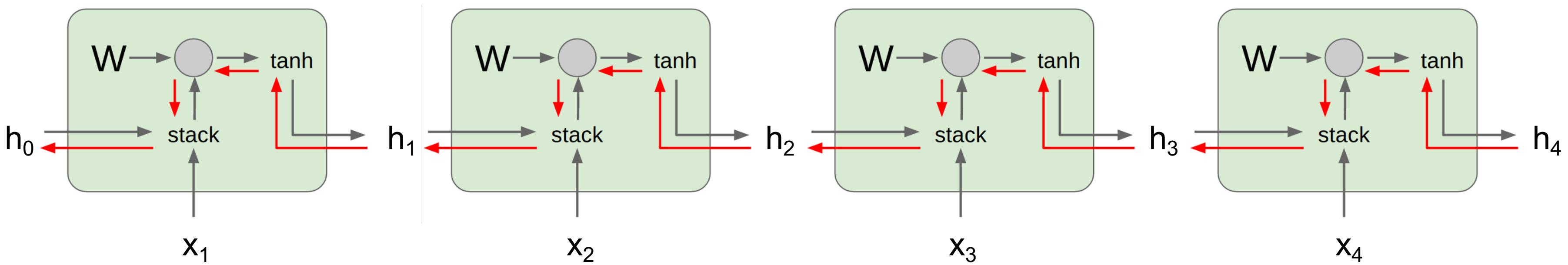
→ **Gradient clipping**: Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

Largest singular value < 1:
Vanishing gradients

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

→ Change RNN architecture

Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}\right)$$

LSTM

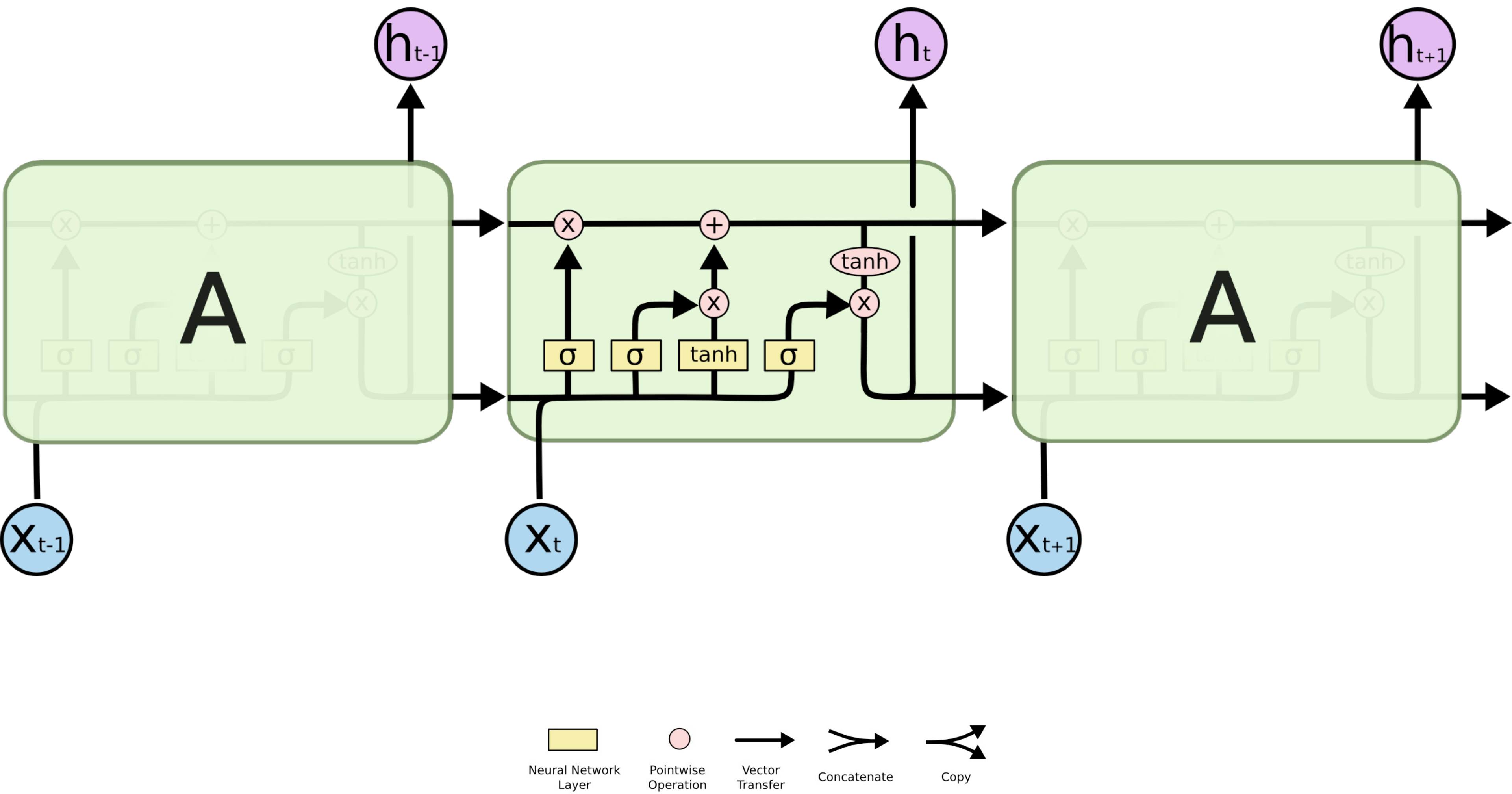
$$\begin{pmatrix}i\\f\\o\\g\end{pmatrix} = \begin{pmatrix}\sigma\\\sigma\\\sigma\\\tanh\end{pmatrix} W \begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}$$

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

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

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

Meet LSTMs



h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTMs Intuition: Memory

- Cell State / Memory

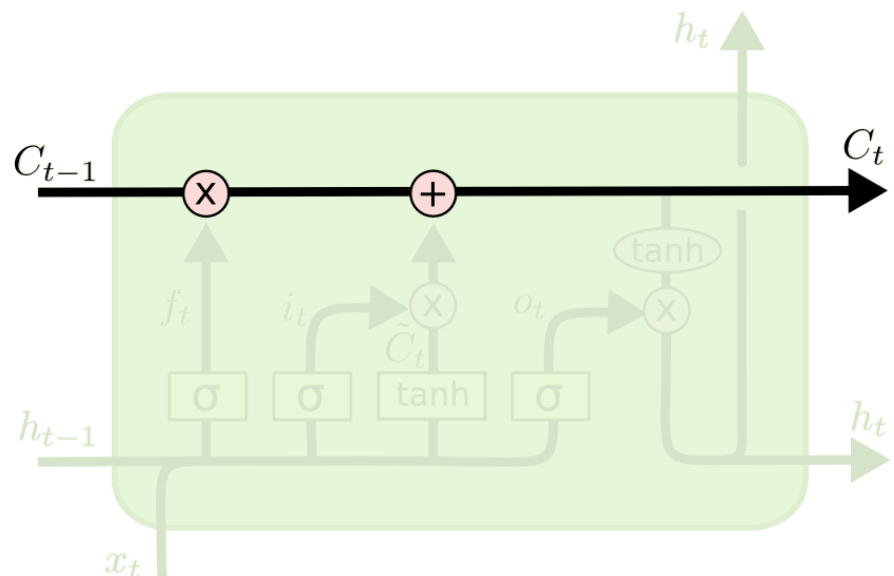
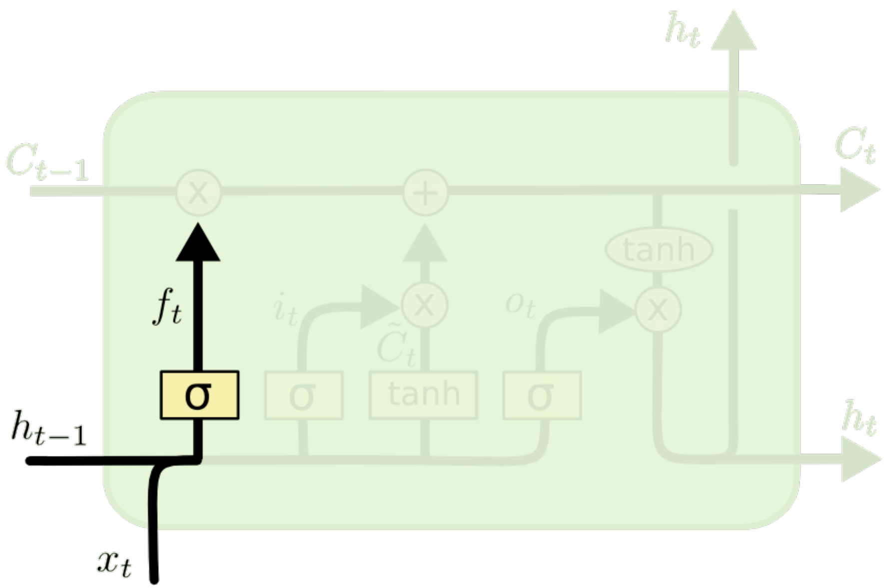


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Forget Gate

- Should we continue to remember this "bit" of information or not?

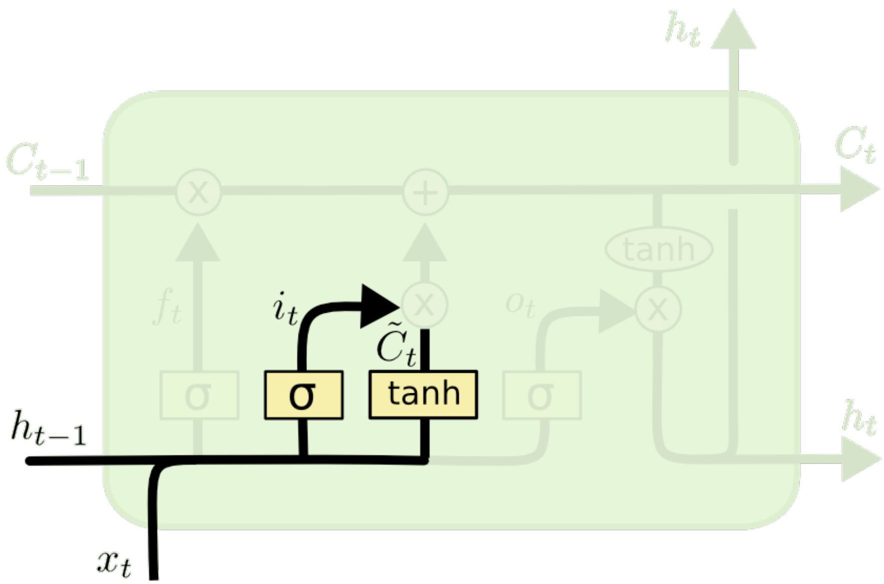


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Input Gate

- Should we update this “bit” of information or not?
  - If so, with what?

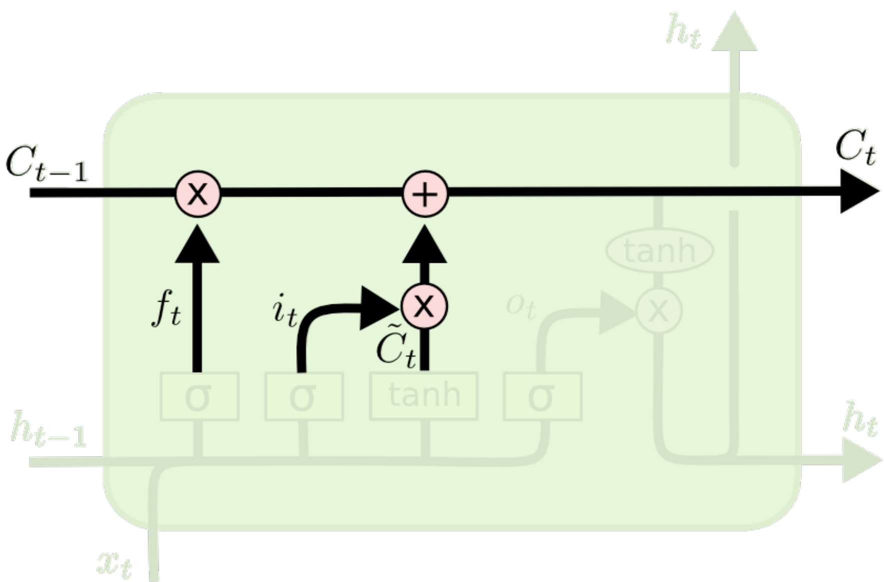


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

LSTMs Intuition: Memory Update

- Forget that + memorize this

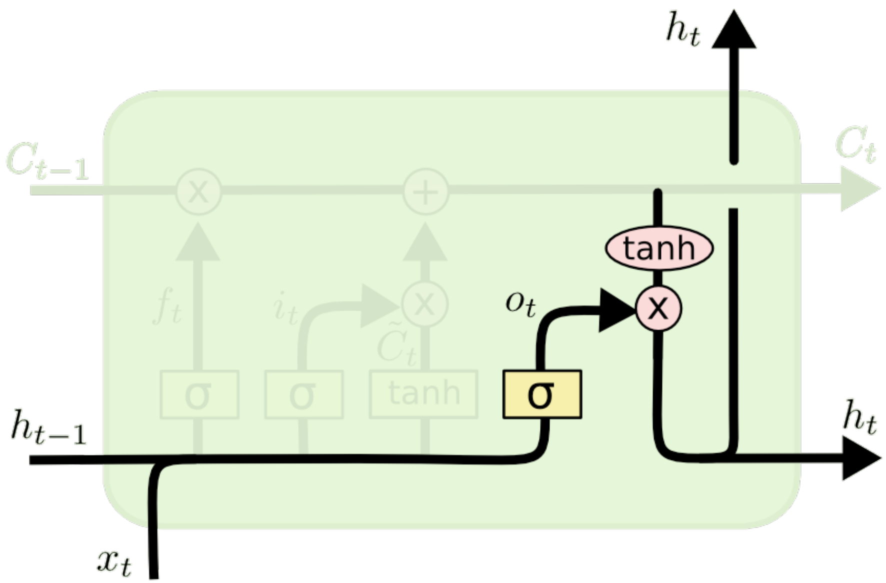


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

 Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Output Gate

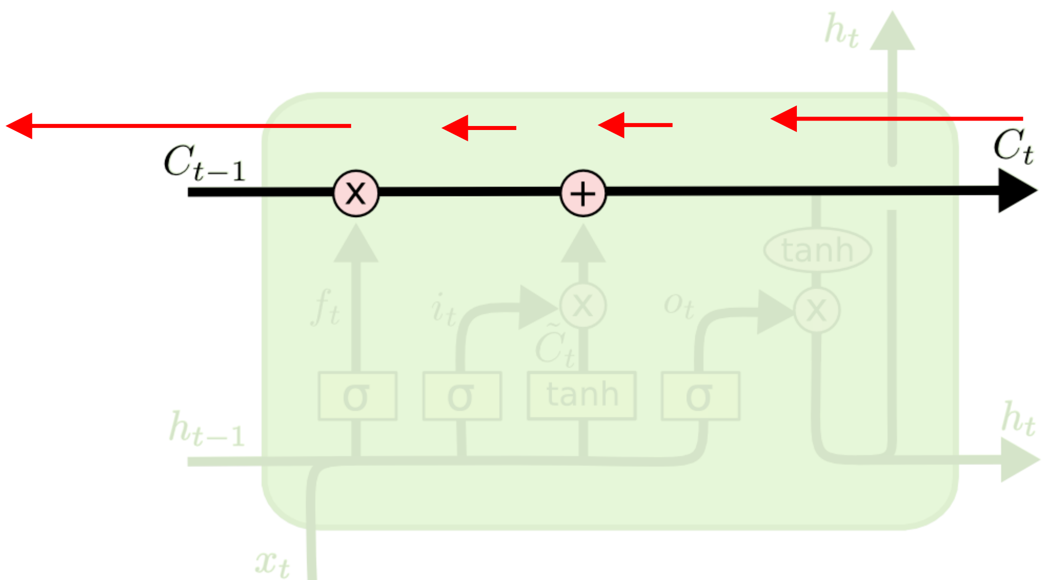
- Should we output this “bit” of information to “deeper” layers?



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

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

LSTMs Intuition: Additive Updates



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

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Additive Updates

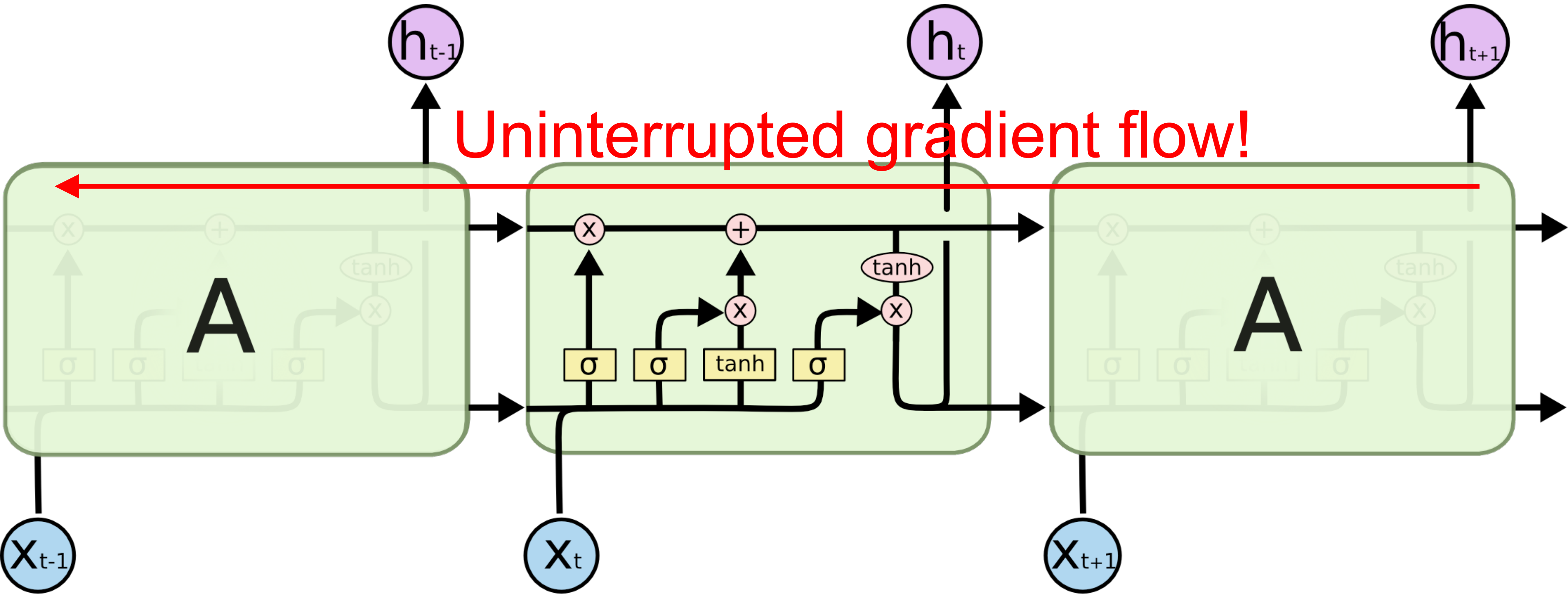
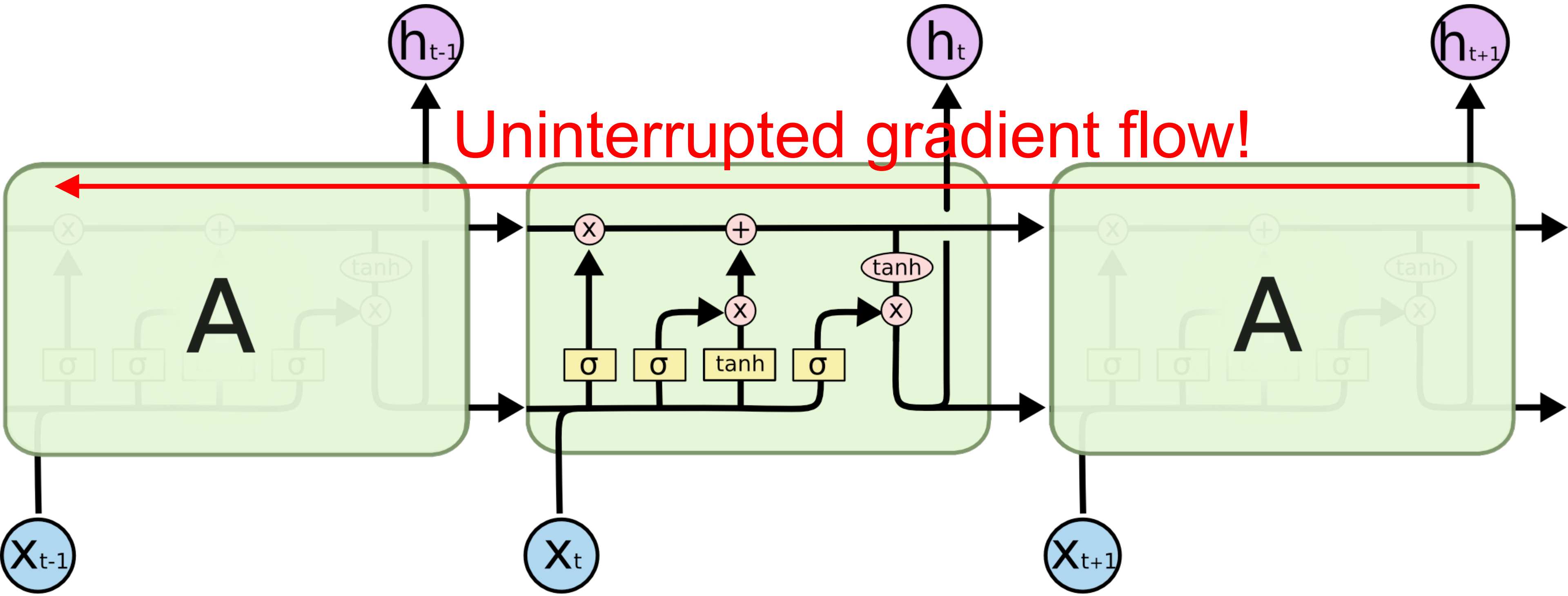
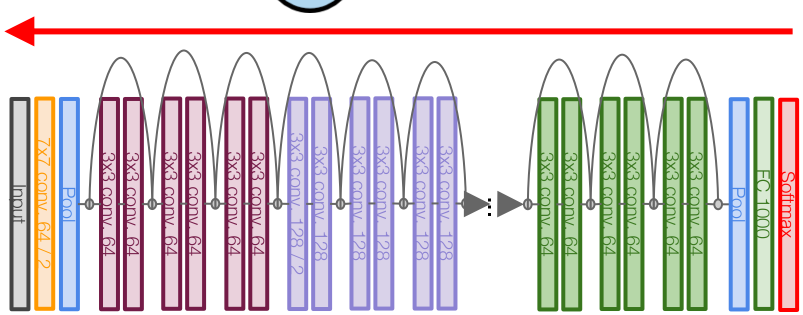


 Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Additive Updates



Similar to ResNet!



LSTMs

- A pretty sophisticated cell

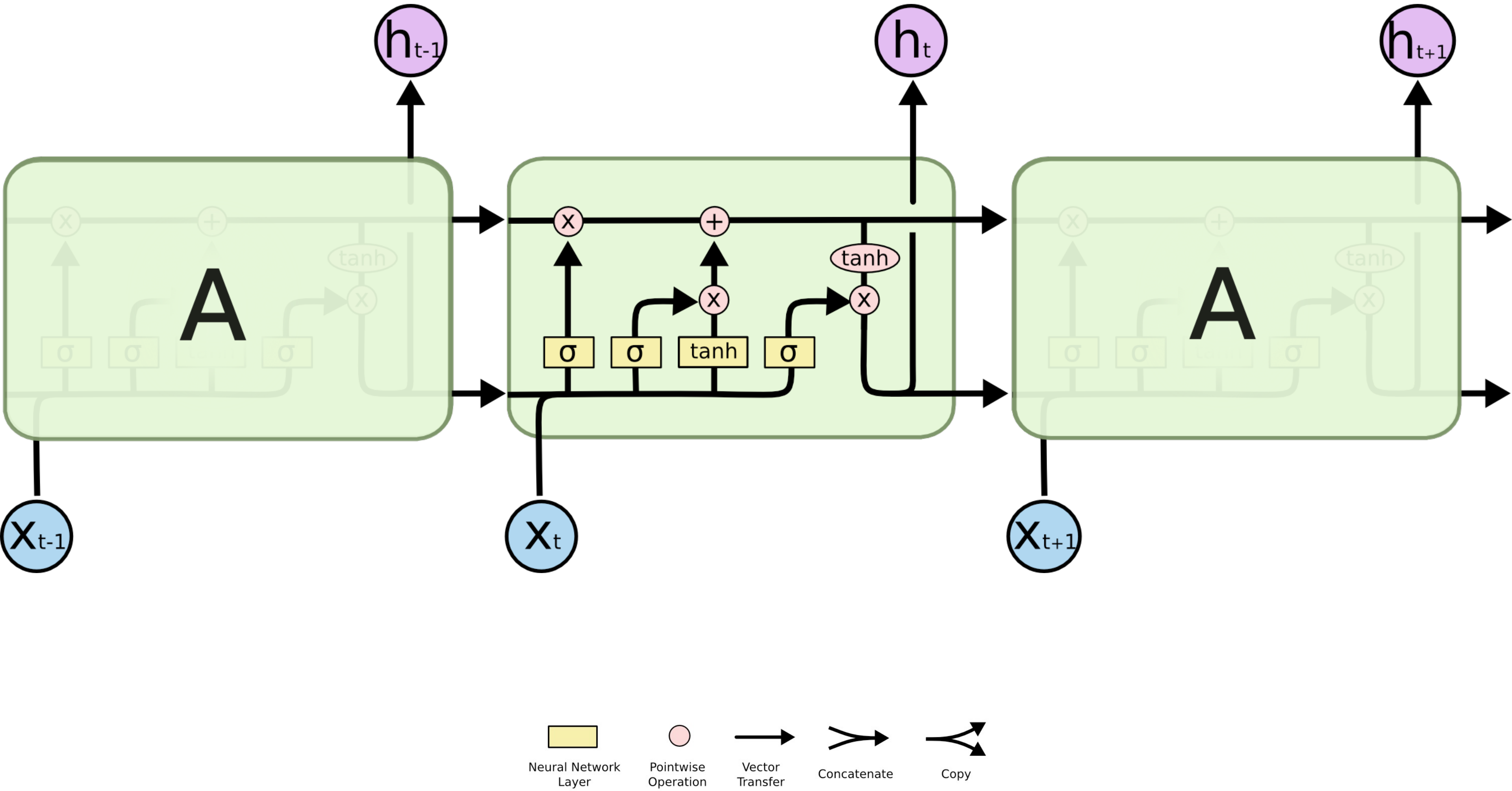
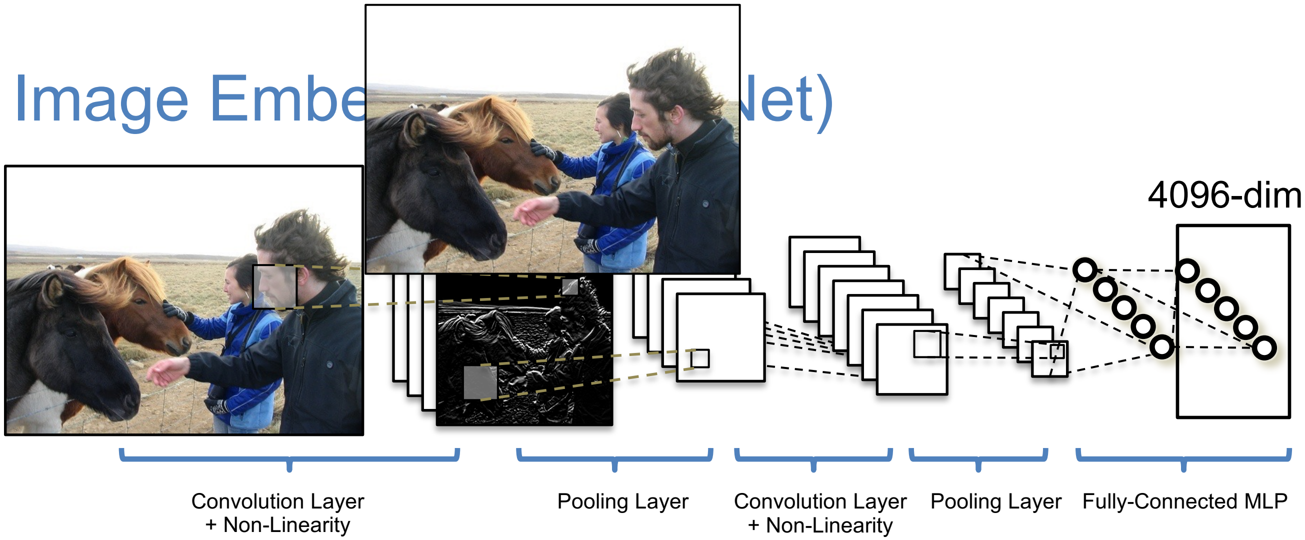


 Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

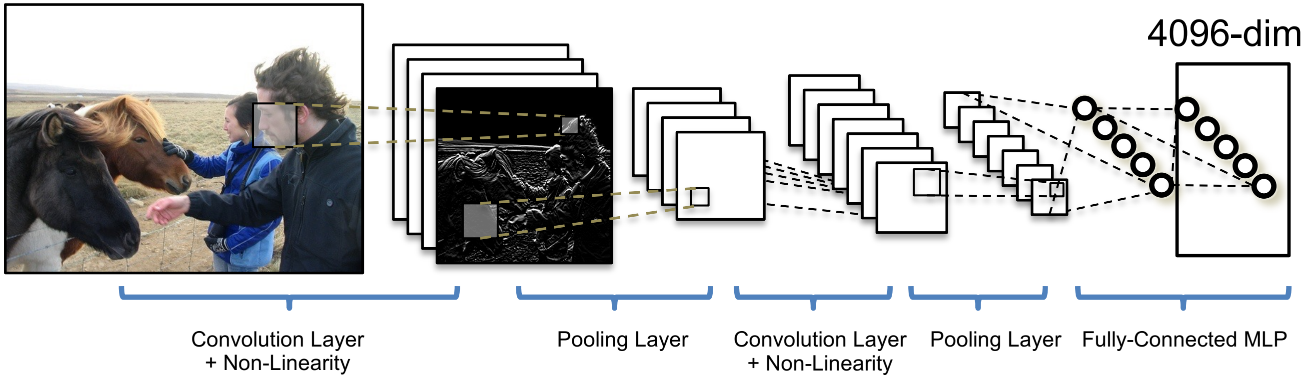
Neural Image Captioning

Image Embe Net)

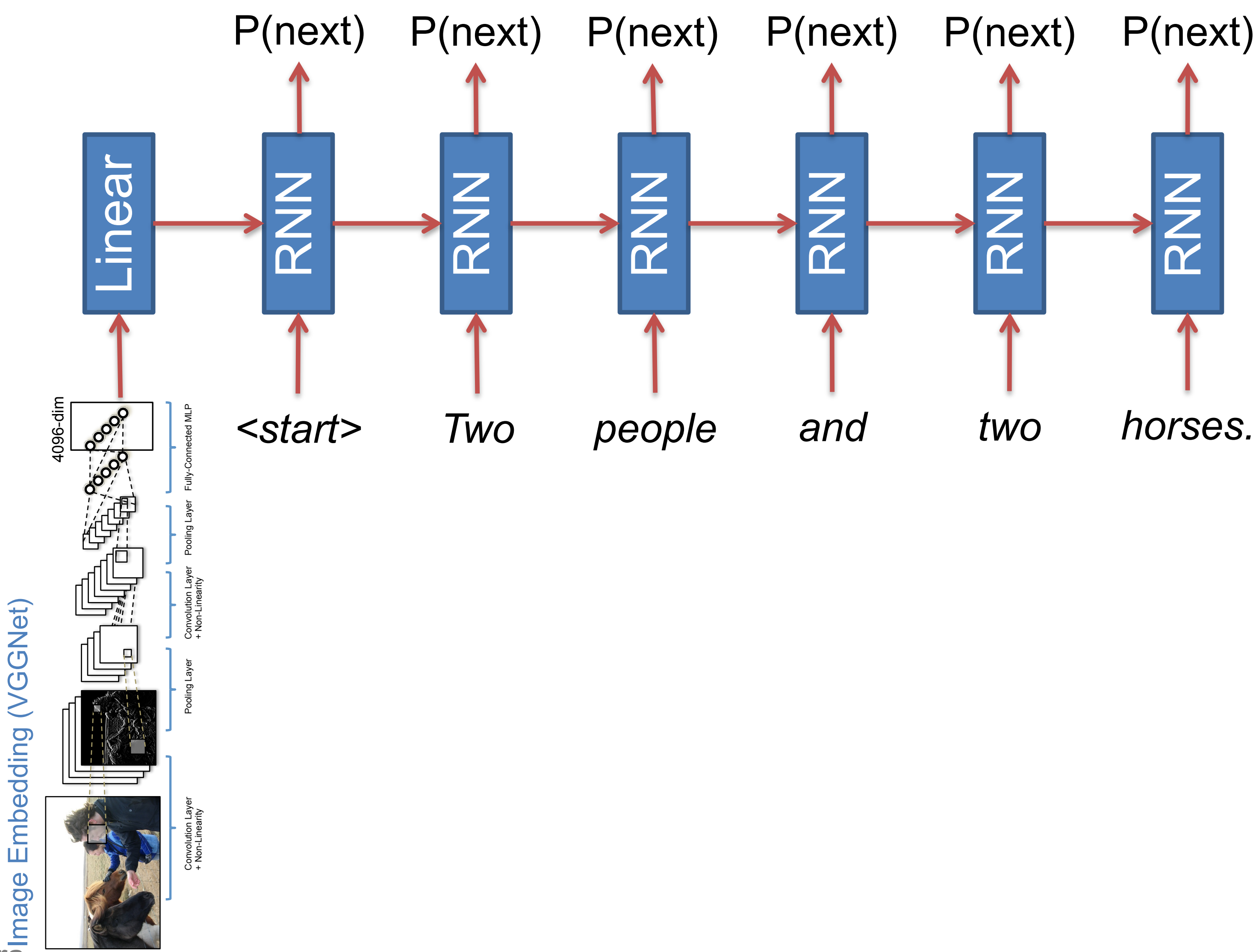


Neural Image Captioning

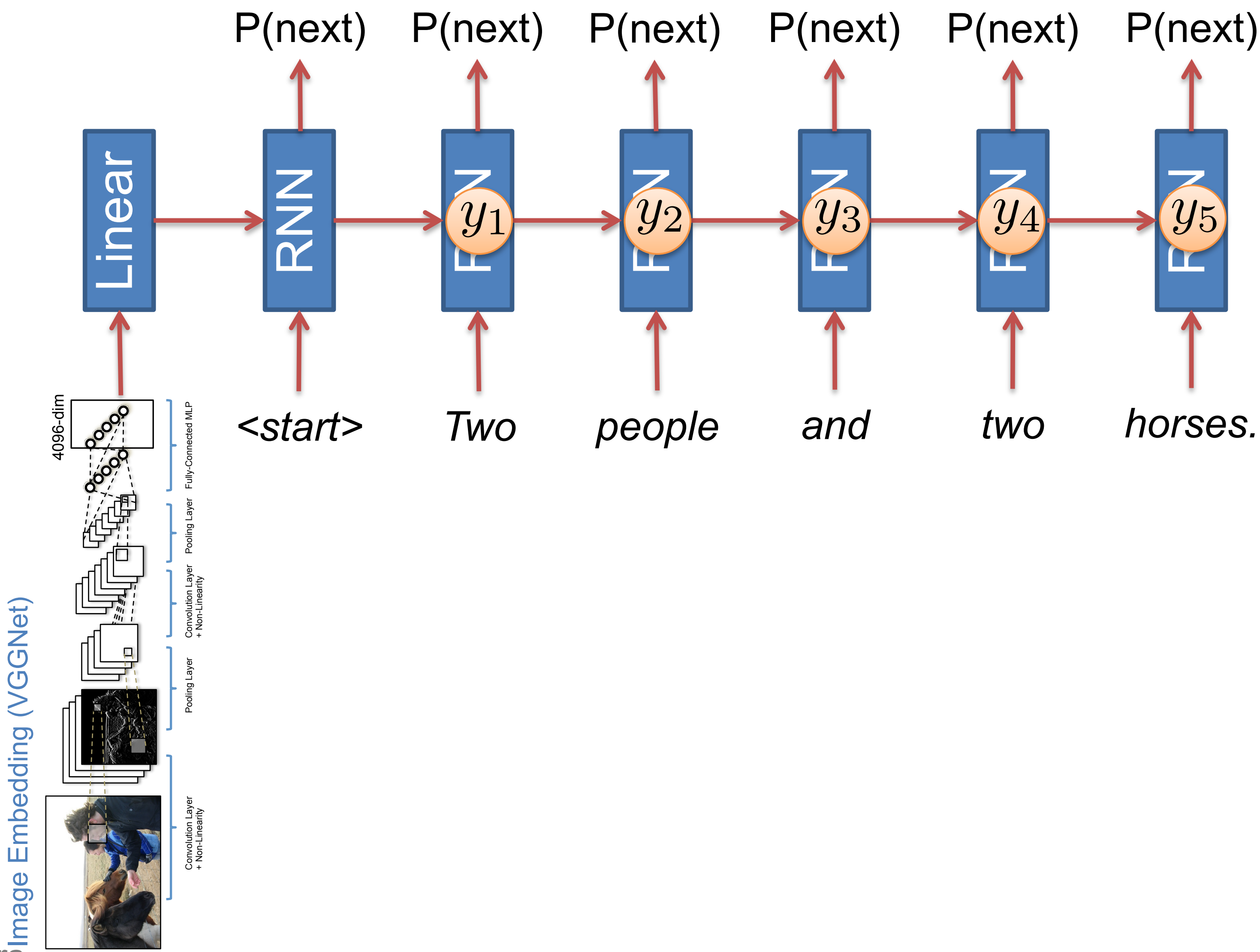
Image Embedding (VGGNet)



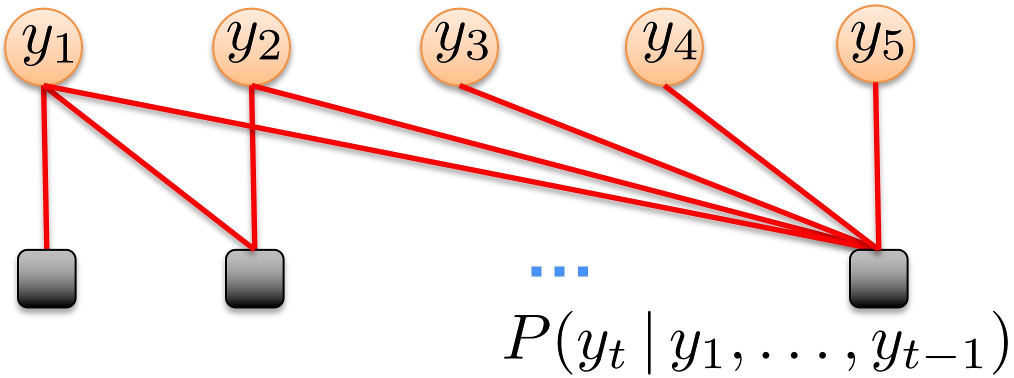
Neural Image Captioning



Neural Image Captioning



Sequence Model Factor Graph



Beam Search Demo

- http://dbs.cloudcv.org/captioning

Image Captioning

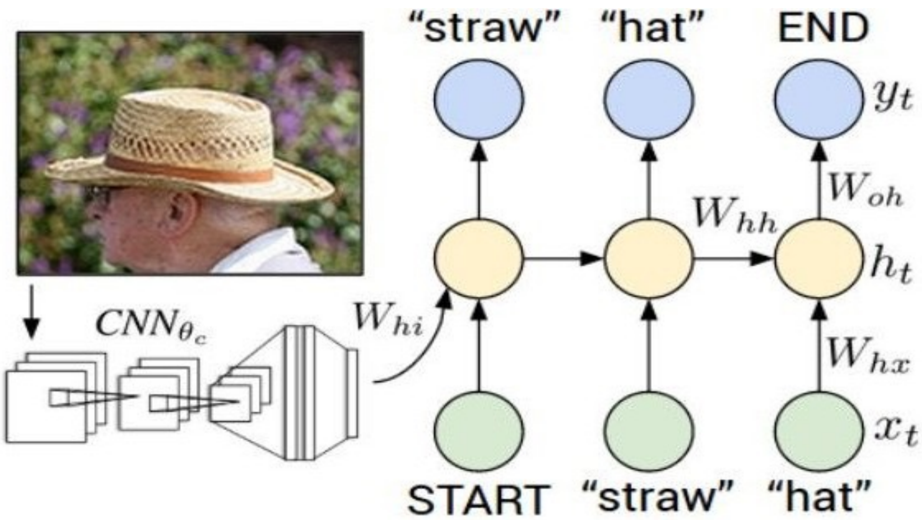
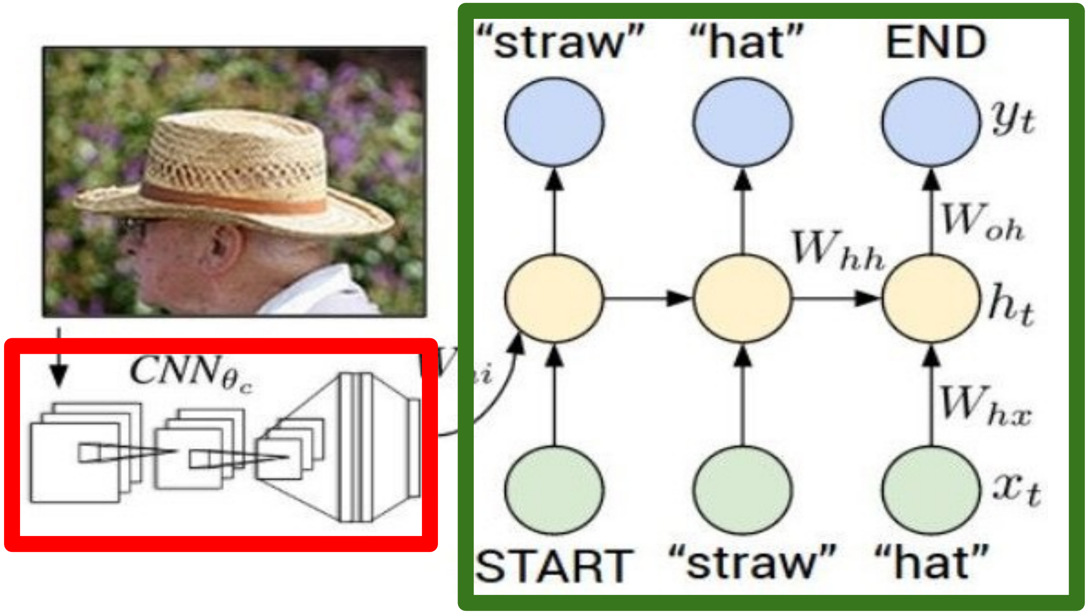


Figure from Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015. Reproduced for educational purposes.

- Many recent works on this:
- Baidu/UCLA: Explain Images with Multimodal Recurrent Neural Networks
- Toronto: Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models
- Berkeley: Long-term Recurrent Convolutional Networks for Visual Recognition and Description
- Google: Show and Tell: A Neural Image Caption Generator
- Stanford: Deep Visual-Semantic Alignments for Generating Image Description
- UML/UT: Translating Videos to Natural Language Using Deep Recurrent Neural Networks
- Microsoft/CMU: Learning a Recurrent Visual Representation for Image Caption Generation
- Microsoft: From Captions to Visual Concepts and Back

Recurrent Neural Network



Convolutional Neural Network

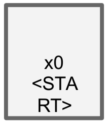
test image


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
FC-1000
softmax

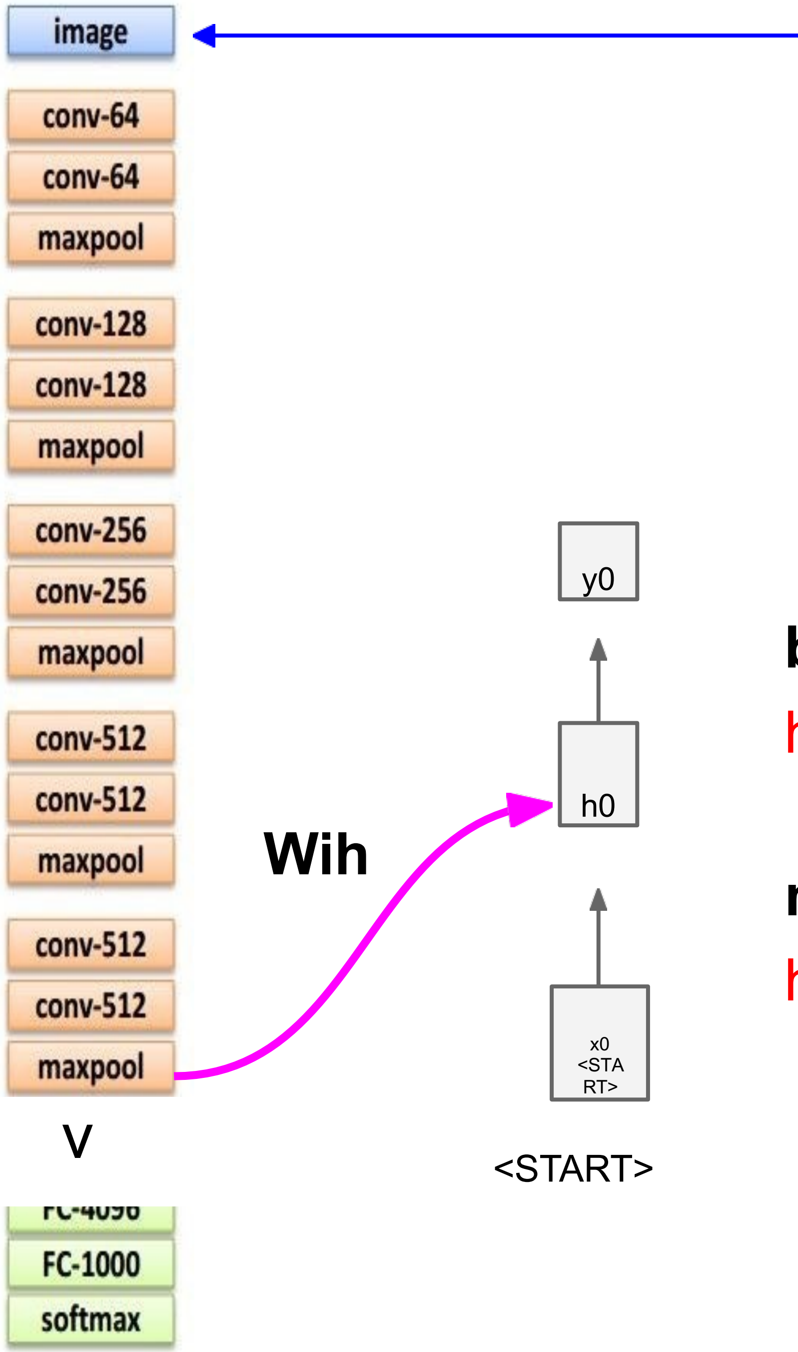
test image


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
X
FC-4096
FC-1000
softmax
test image

test image



<START>



test image

before:

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

now:

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

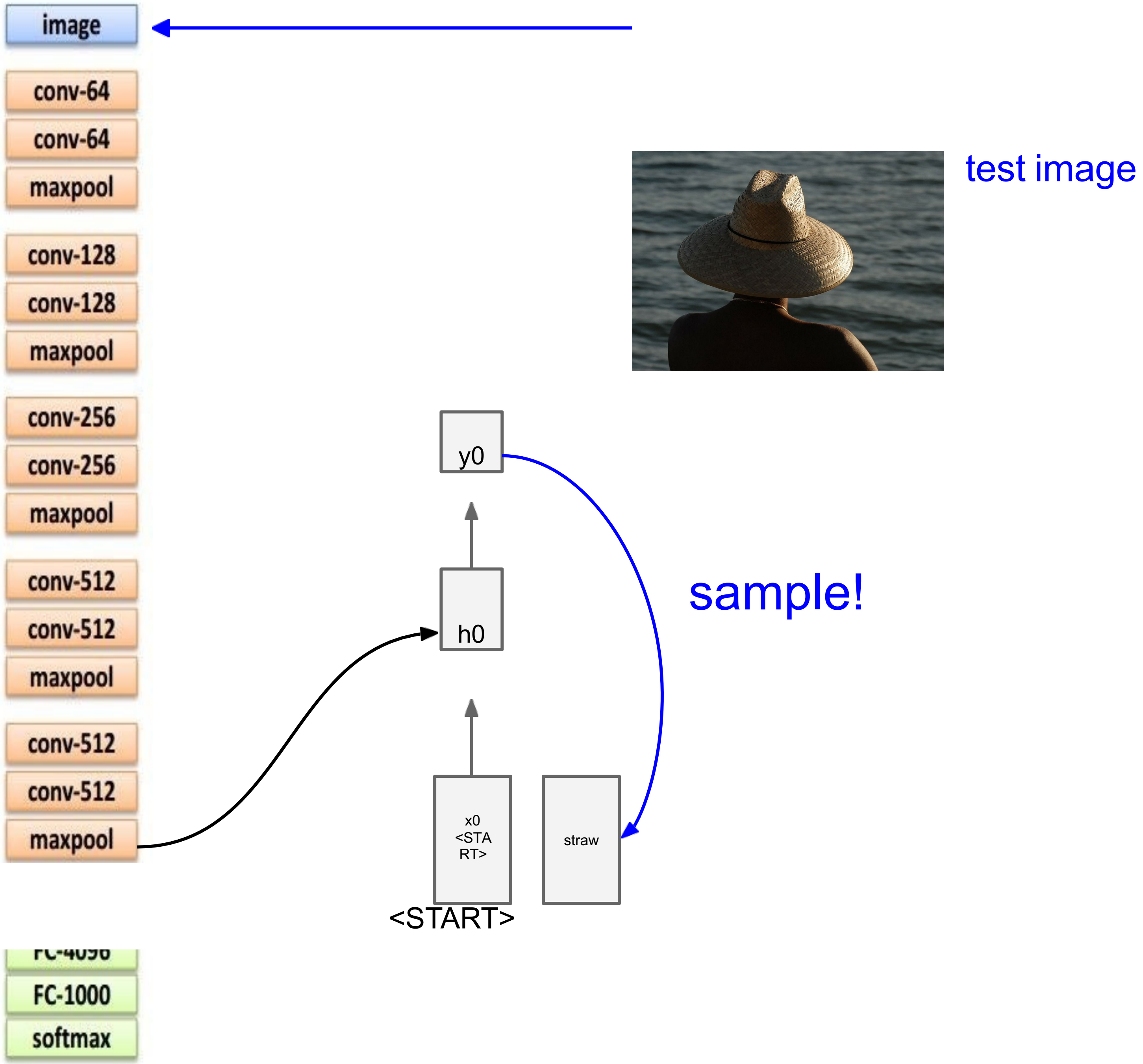


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-1000
softmax
test image
y0
h0
sample!
x0 <START>
straw
<START>

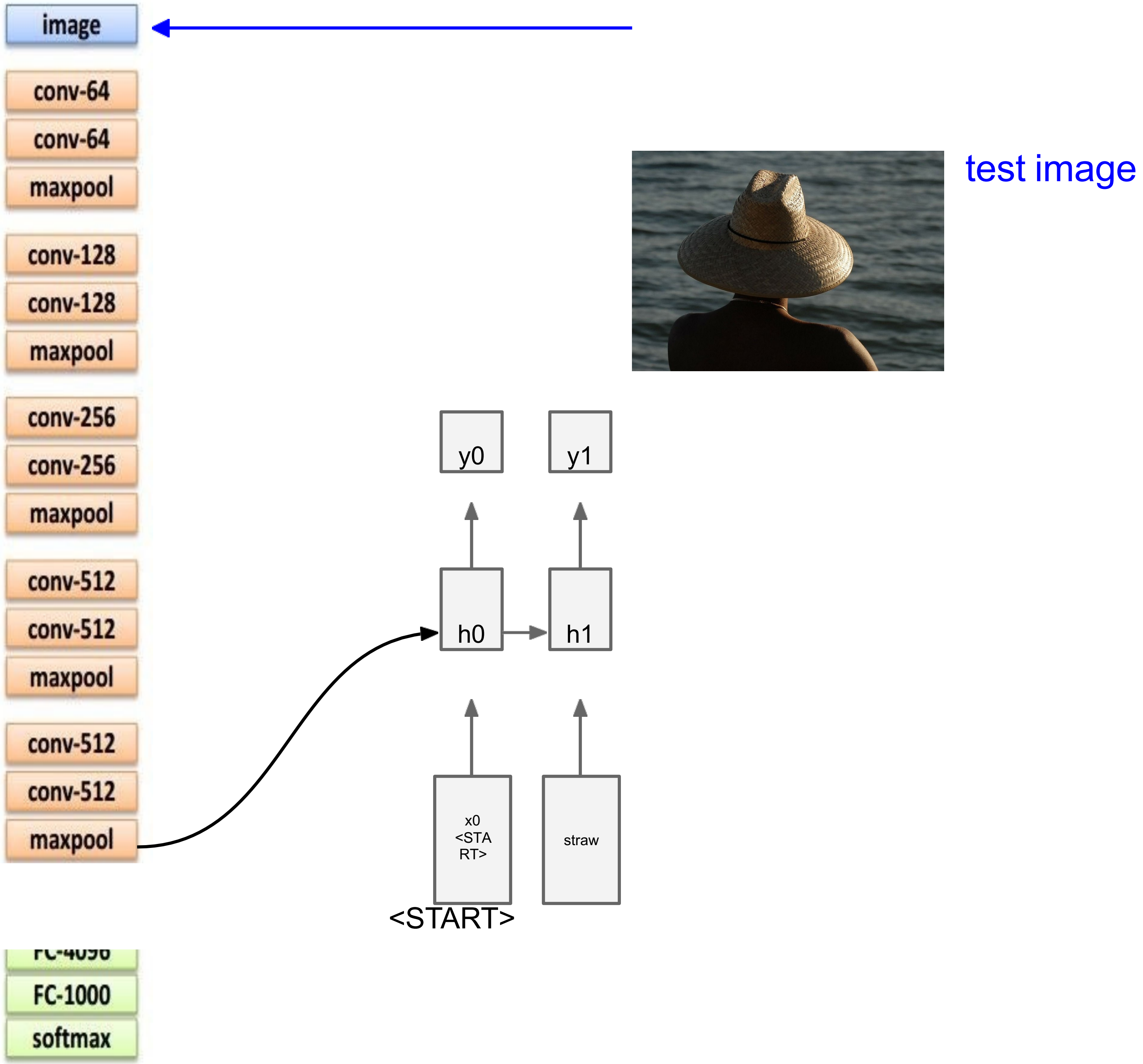


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-1000
softmax
test image
y0
y1
h0
h1
x0
<START>
straw
<START>

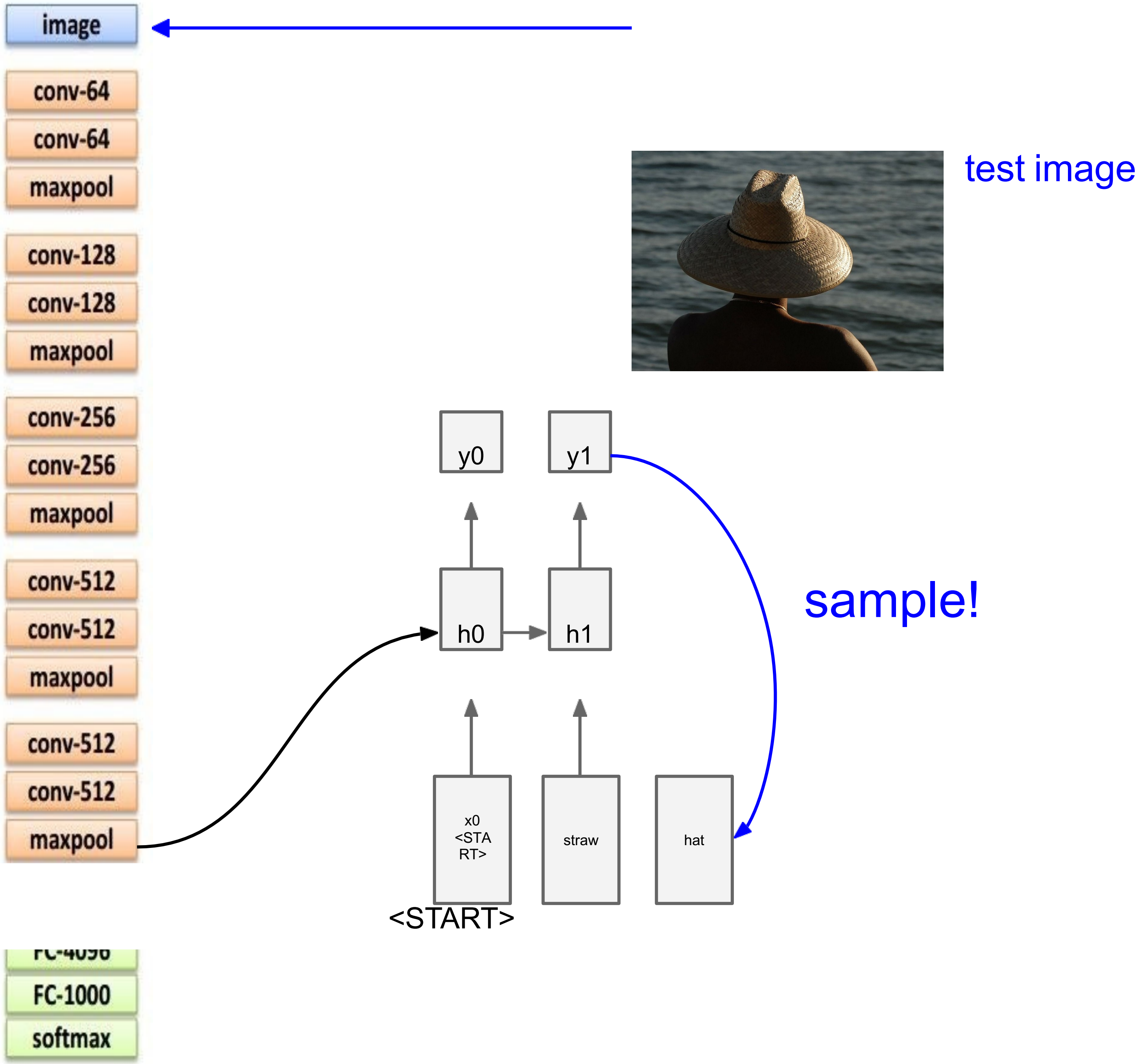


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-1000
softmax
test image
y0
y1
h0
h1
sample!
x0
<STA
RT>
straw
hat
<START>

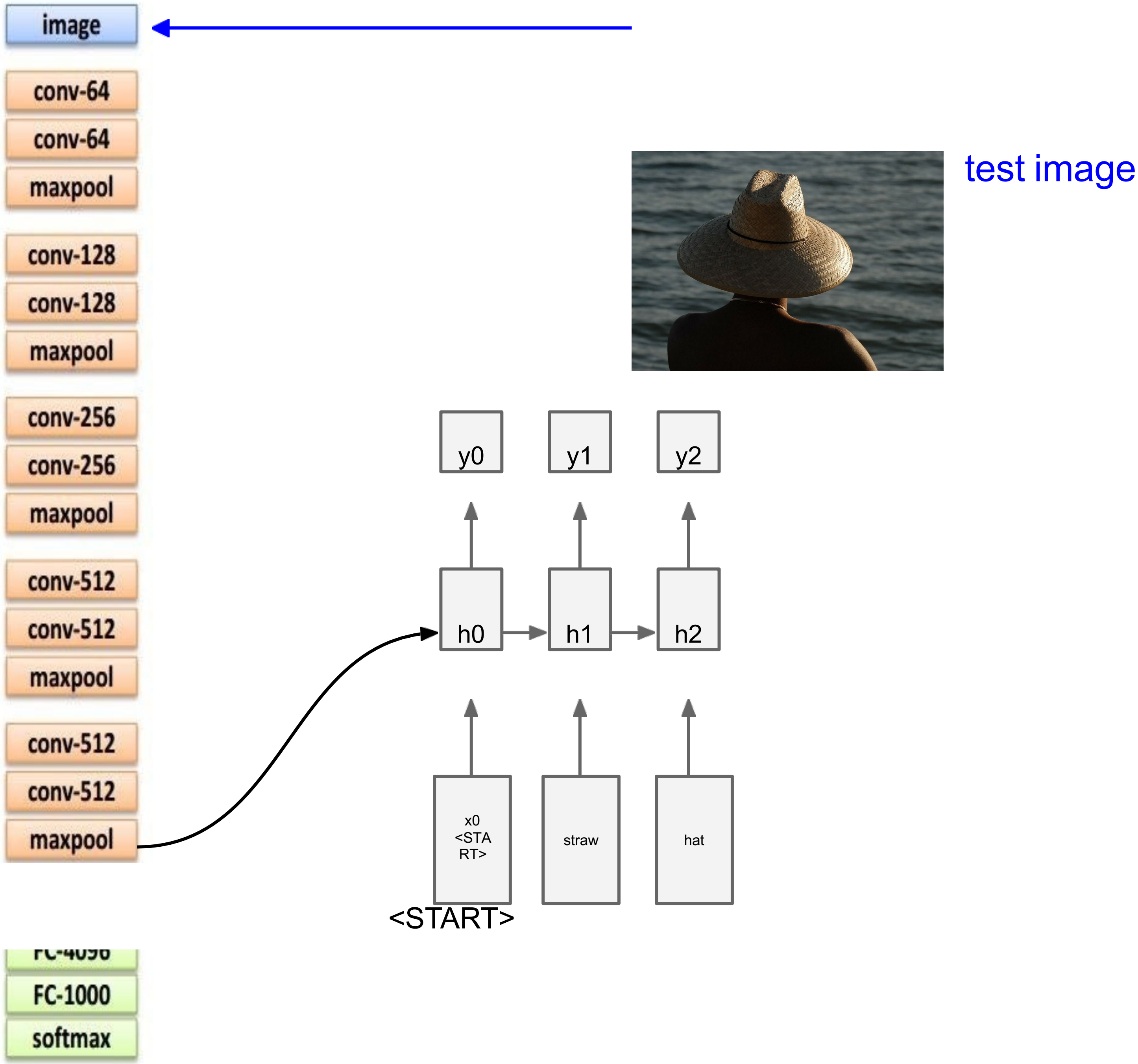


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-1000
softmax
test image
y0
y1
y2
h0
h1
h2
x0
<STA
RT>
straw
hat
<START>

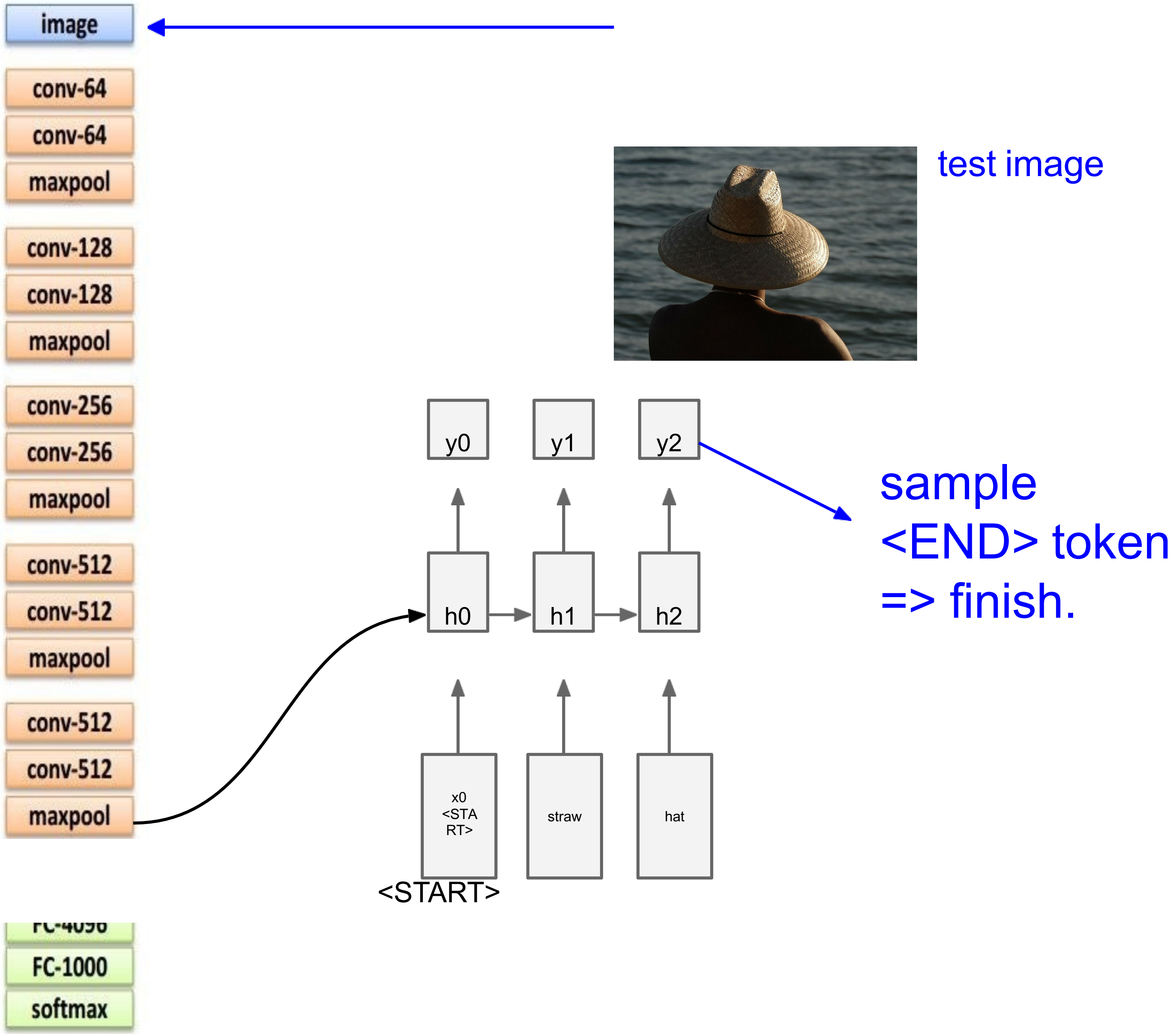


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-1000
softmax
test image
y0
y1
y2
h0
h1
h2
x0 <STA RT>
straw
hat
<START>
sample <END> token => finish.

Captions generated using neuraltalk2
All images are CC0 Public domain:
cat suitcase, cat tree, dog, bear,
surfers, tennis, giraffe, motorcycle

Image Captioning: Example Results

A cat sitting on a suitcase on the floor

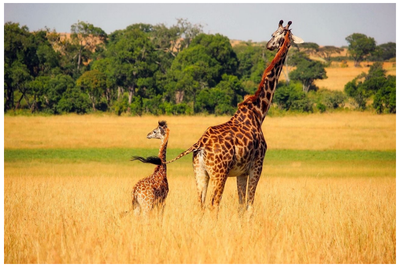
A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court



Two giraffes standing in a grassy field

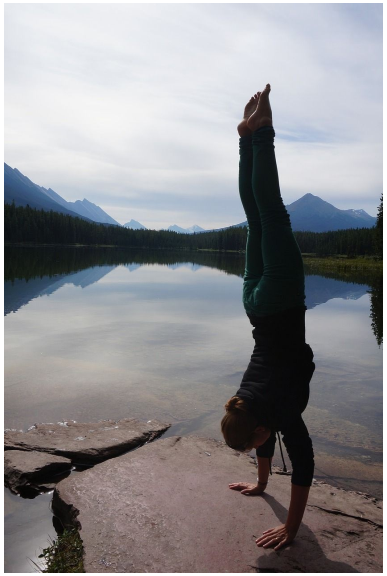
A man riding a dirt bike on a dirt track

Image Captioning: Failure Cases

Captions generated using neuraltalk2
All images are CC0 Public domain: fur coat, handstand, spider web, baseball

A woman is holding a cat in her hand

A person holding a computer mouse on a desk

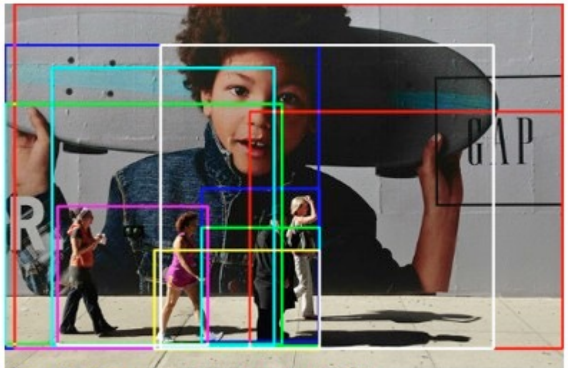


A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

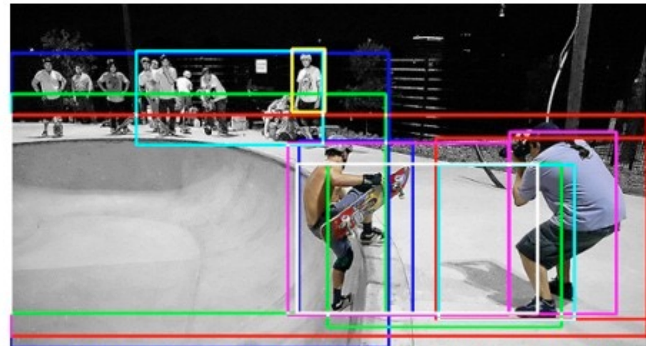
A man in a baseball uniform throwing a ball

More Image Captioning Examples



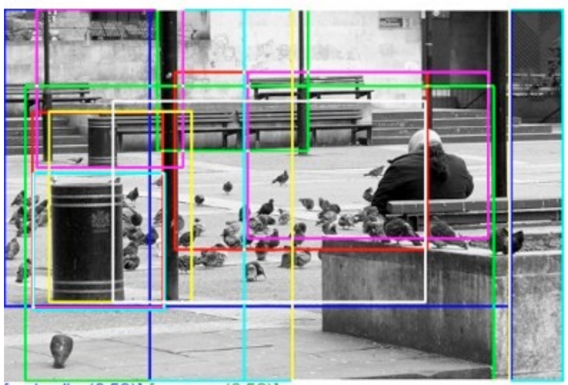
[men (0.59)] [group (0.66)] [woman (0.64)]
[people (0.89)] [holding (0.60)] [playing (0.61)] [tennis (0.69)]
[court (0.51)] [standing (0.59)] [skis (0.58)] [street (0.52)]
[man (0.77)] [skateboard (0.67)]

a group of people standing next to each other
people stand outside a large ad for gap featuring a young boy



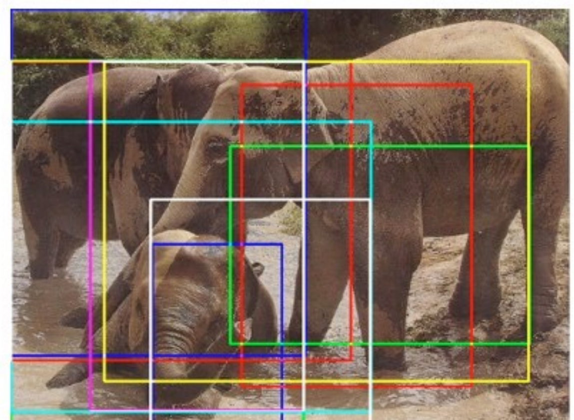
[person (0.55)] [street (0.53)] [holding (0.55)] [group (0.63)] [slope (0.51)]
[standing (0.62)] [snow (0.91)] [skis (0.74)] [player (0.54)]
[people (0.85)] [men (0.57)] [skiing (0.51)]
[skateboard (0.89)] [riding (0.75)] [tennis (0.74)] [trick (0.53)] [skate (0.52)]
[woman (0.52)] [man (0.86)] [down (0.61)]

a group of people riding skis down a snow covered slope
a guy on a skate board on the side of a ramp



[umbrella (0.59)] [woman (0.52)]
[fire (0.96)] [hydrant (0.96)] [street (0.79)] [old (0.50)]
[bench (0.81)] [building (0.75)] [standing (0.57)] [baseball (0.55)]
[white (0.82)] [sitting (0.65)] [people (0.79)] [photo (0.53)]
[black (0.84)] [kitchen (0.54)] [man (0.72)] [water (0.56)]

a black and white photo of a fire hydrant
a courtyard full of poles pigeons and garbage cans also has benches on either side of it one of which shows the back of a large person facin g in the direction of the pigeons



[horse (0.53)] [bear (0.71)] [elephant (0.99)] [elephants (0.95)]
[brown (0.68)] [baby (0.62)] [walking (0.57)] [laying (0.61)]
[man (0.57)] [standing (0.79)] [field (0.65)]
[water (0.83)] [large (0.71)] [dirt (0.65)] [river (0.58)]

a baby elephant standing next to each other on a field
elephants are playing together in a shallow watering hole

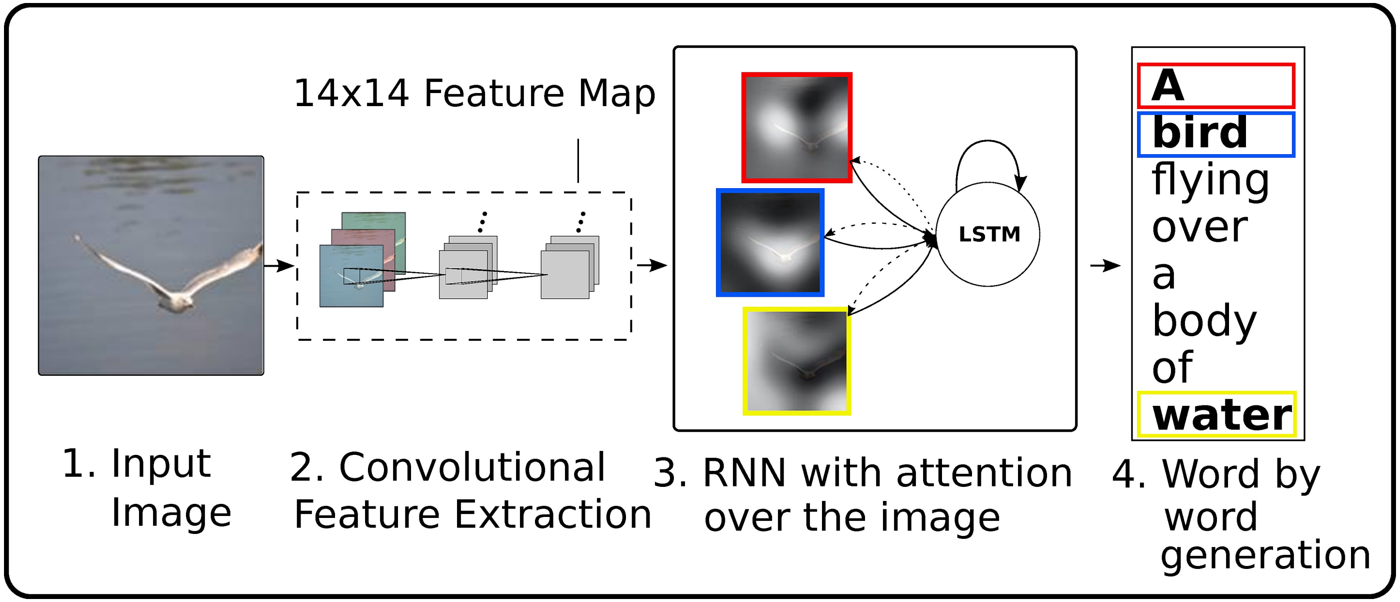
From Captions to Visual Concepts and Back, Hao Fang∗ Saurabh Gupta∗ Forrest Iandola∗ Rupesh K. Srivastava∗, Li Deng Piotr Dollar, Jianfeng Gao Xiaodong He, Margaret Mitchell John C. Platt, C. Lawrence Zitnick, Geoffrey Zweig, CVPR 2015.

Show, Attend and Tell

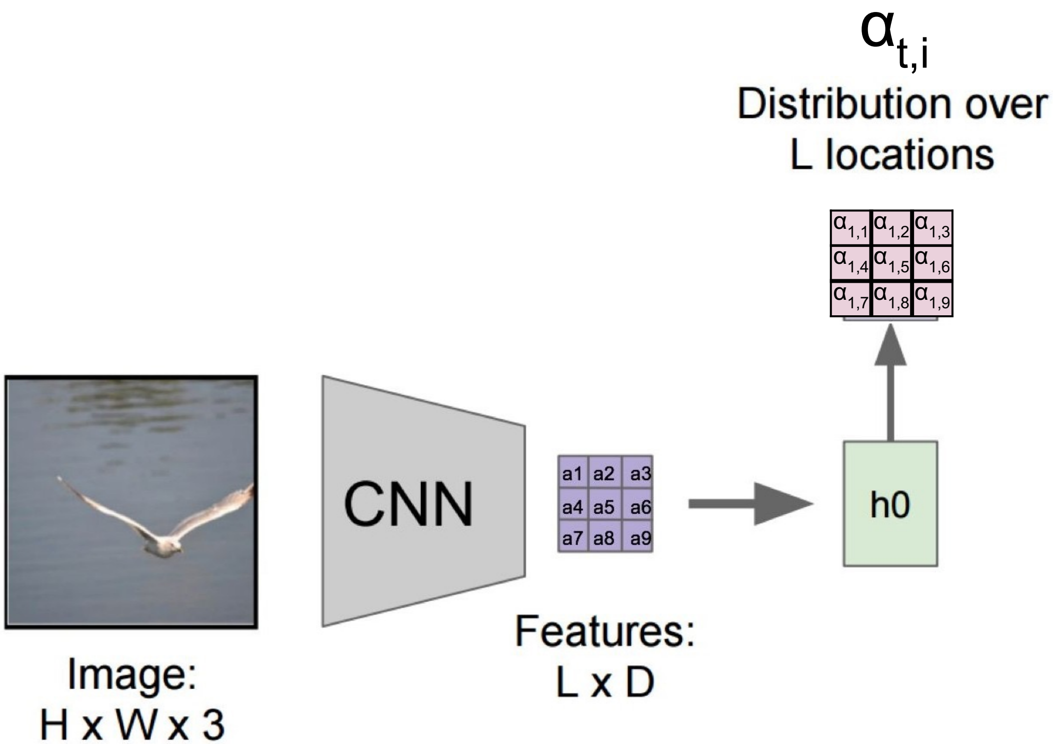
(Xu et al., 2015)

Instead of learning word detectors over image regions, consider learning an **attention model** instead

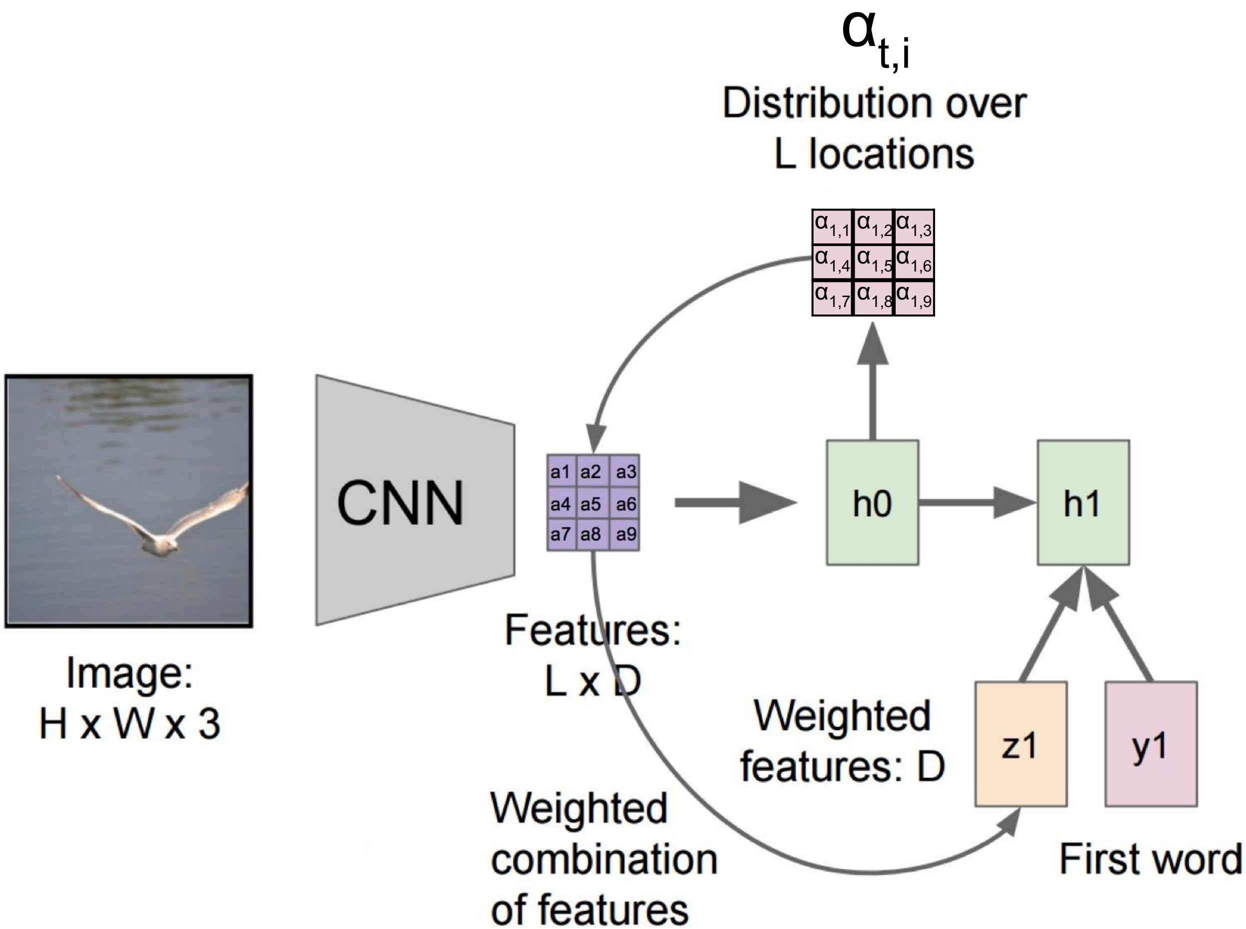
- What is visual attention?
- How to augment Show and Tell with visual attention
- Soft vs. hard attention



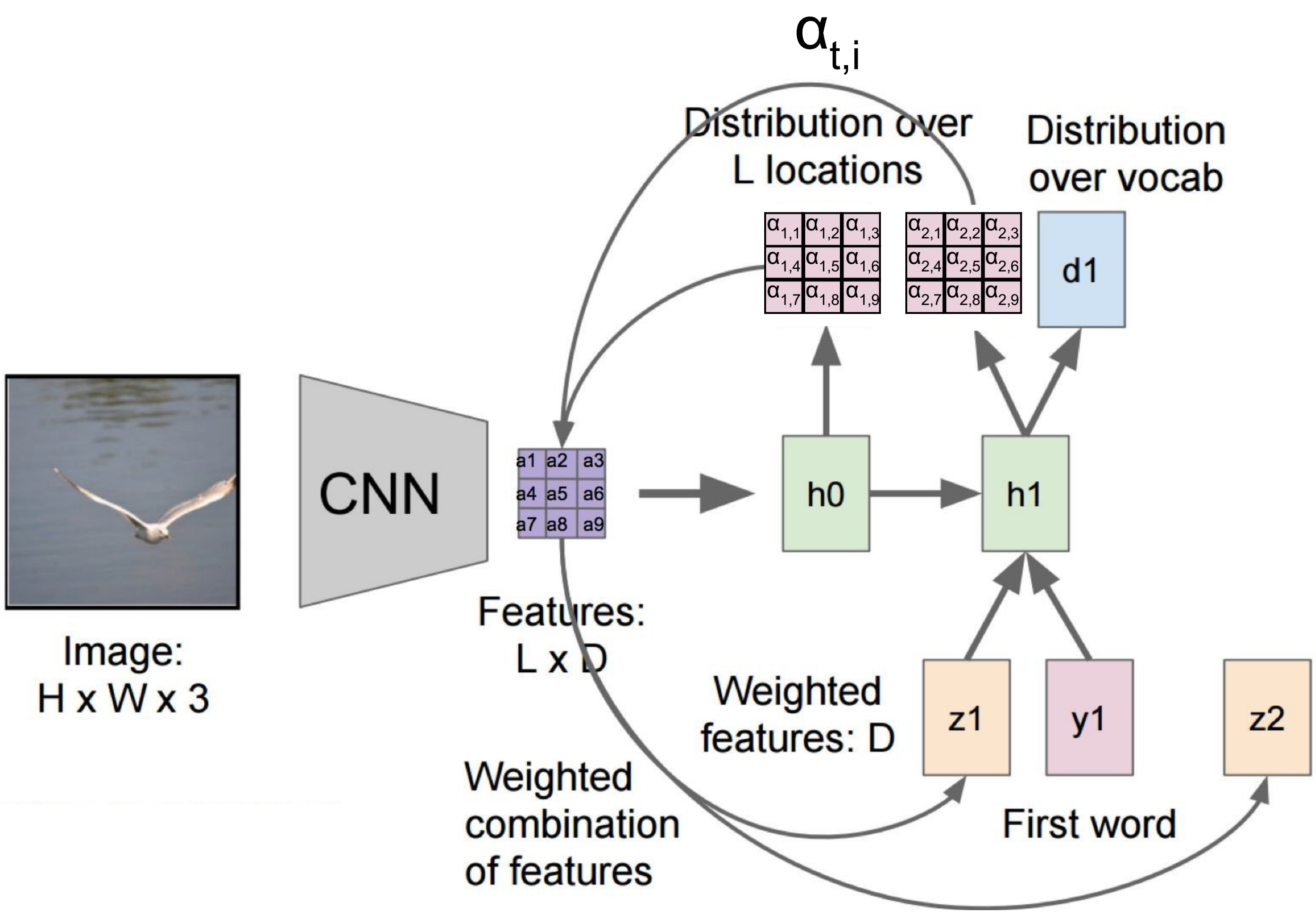
Xu et al., Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015



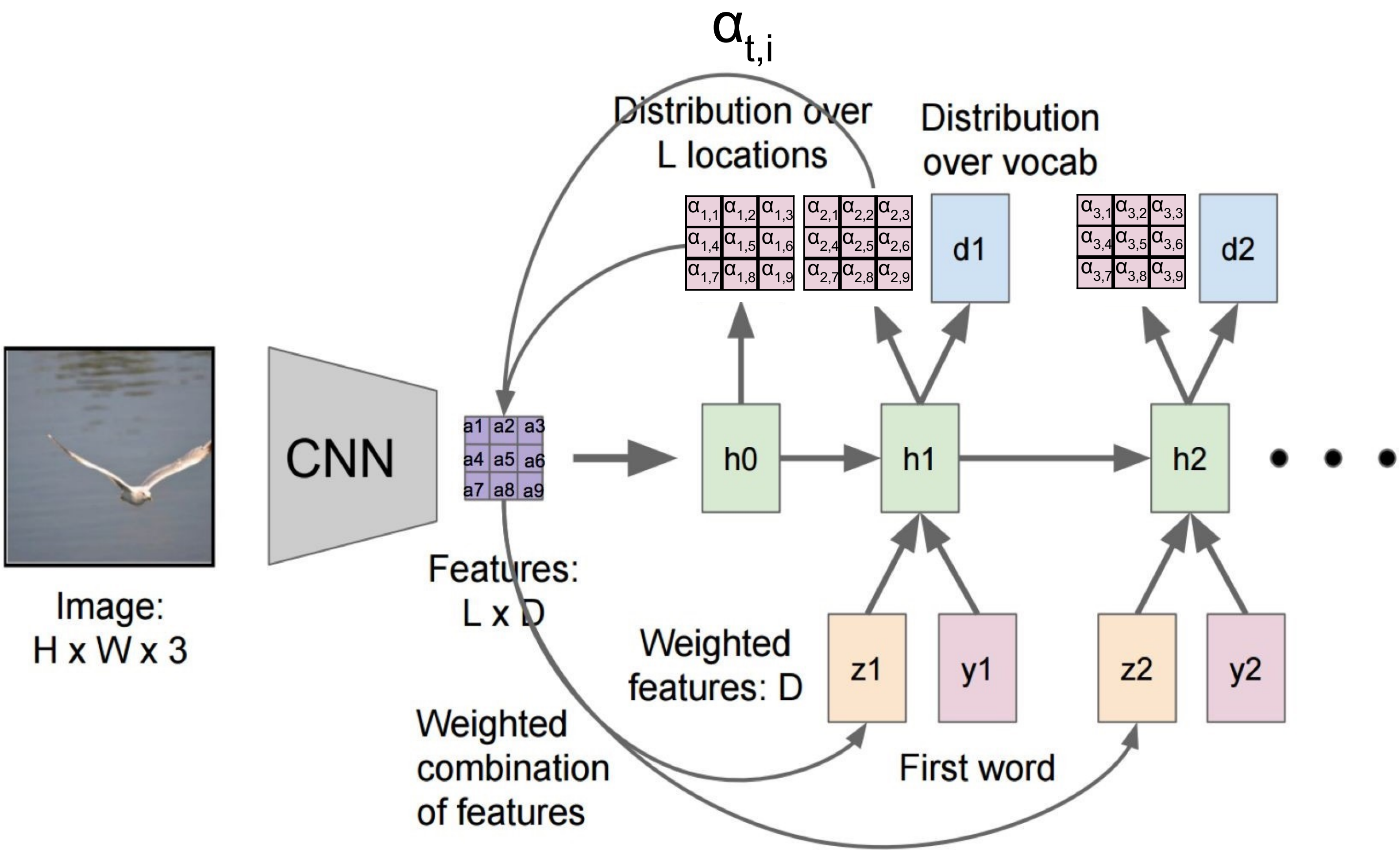
Slides from Stanford 231n
Xu et al., Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015



Slides from [Stanford 231n]
Xu et al., [Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention], ICML 2015



Slides from [Stanford 231n]
Xu et al., [Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention], ICML 2015



Slides from Stanford 231n

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

Soft Attention

$\boldsymbol{z}_t$ is calculated by taking the weighted sum of all feature vectors a

$$z_t = \sum_{i=1}^{L} \alpha_t[i] \cdot a_i$$

- Differentiable
- Deterministic: α_i's assign relative importance to give to location i in blending the a_i's together
- Learned using standard backpropagation

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

Soft Attention: Examples

More examples at [project website]
Xu et al., [Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention], ICML 2015

Hard Attention

At time step *t*, the index into the feature vectors is sampled from the current location distribution vector α_t

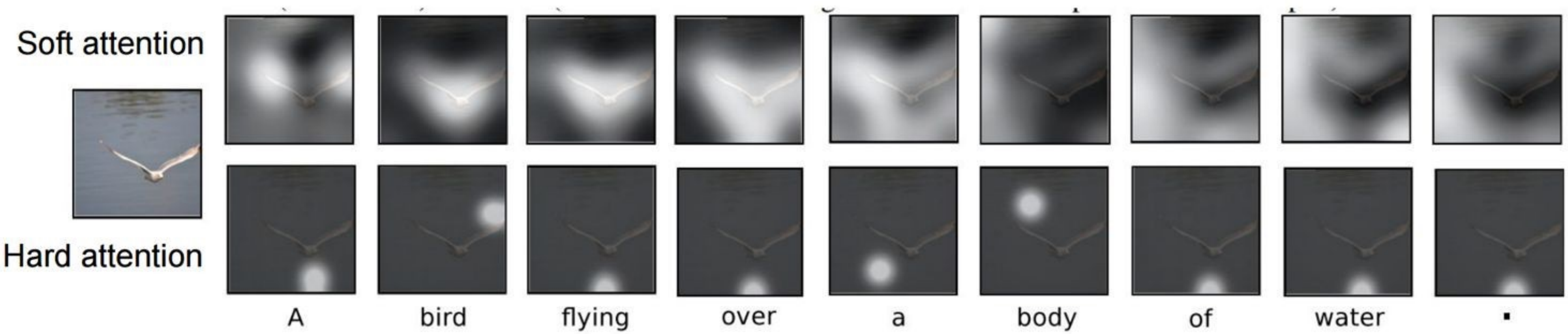
$$k = sample(\alpha_t)$$

$$z_t = a_k$$

- Stochastic: α_i's assign probability that location *i* is the right place to focus for producing the next word
- Focuses on one image region at a time
- Non-differentiable due to sampling
  - Set up as reinforcement learning problem:
    - Action = which area to attend to next
    - Reward = log-likelihood of caption wrt to target sentence

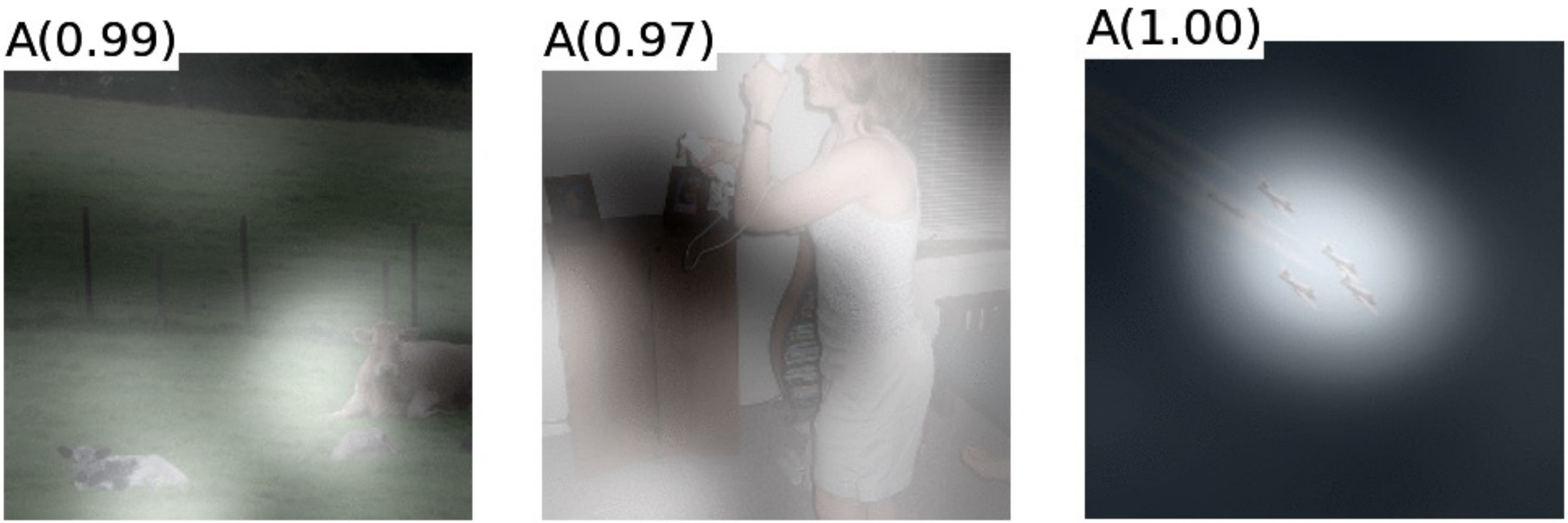
Xu et al., Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

Soft vs. Hard Attention



Xu et al., [Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention](), ICML 2015

Examples



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

How to Evaluate different captions?

1. A woman in a green shirt is getting food ready with a child , while sitting on rocks .
2. A mother and child having a picnic on a big rock with blue utensils .
3. A woman serving food for a little boy outside on a large rock .
4. A woman and a baby eating (having a picnic) .
5. A mother and child picnic on some rocks .

P. Young et al., [From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions](), TACL 2014

[Slide: Narayan-Chen, Xie, Shan]

BLEU (BiLingual Evaluation Understudy)

(Papineni et al., 2002)

- *"The closer a machine translation is to a professional human translation, the better it is."*
- Analyzes co-occurrences of *n*-grams between candidate and reference sentences
  - Modified (clipped) *n*-gram precision
  - Brevity penalty to penalize short candidate sentences
- Has been shown in MT literature to be an insufficient metric (Callison-Burch et al., 2006)
  - Many large variations of a generated sentence can score identically
  - Higher BLEU score is not necessarily indicative of higher human-judged quality

Candidate: `the the the the the the the.`

Reference 1: `The cat is on the mat.`

Reference 2: `There is a cat on the mat.`

Modified Unigram Precision $= 2/7$.

Papineni et al., BLEU: a Method for Automatic Evaluation of Machine Translation, ACL 2002

[Slide: Narayan-Chen, Xie, Shan]

Reference captions:

1. Latino man holding sign on the sidewalk outside promoting Quiznos-Subs .
2. A man is holding an advertisement for Quiznos Subs .
3. A man is holding a Quiznos sign next to a street .
4. A man is holding a Quiznos Sub sign .

Candidate caption:

❓ Quiznos worker wearing sign .

BLEU-4 = 0.106

P. Young et al., [From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions](), TACL 2014

[Slide: Narayan-Chen, Xie, Shan]

METEOR

(Banerjee & Lavie, 2005)

More flexible MT metric that calculates sentence-level similarity scores as a harmonic mean of unigram precision & recall, based on:

- Exact token matching
- Stemmed tokens
- WordNet synonyms
- Paraphrases

SYSTEM	Jim went home	SYSTEM	Jim walks home
REFERENCE	Joe goes home	REFERENCE	Joe goes home

Examples from Statistical Machine Translation slides
Banerjee & Lavie, METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments, ACL 2005

[Slide: Narayan-Chen, Xie, Shan]

CIDEr: Consensus-based Image Description Evaluation

(Vedantam et al., 2015)

- *"Does a caption describe an image as most people tend to describe it?"*
- Automatically evaluate for image I_i how well a candidate sentence c_i matches the **consensus** of a set of image descriptions $S_i = \{s_{i1}, \ldots, s_{im}\}$
  - Intuitively, a measure of consensus should:
    - Encode how often *n*-grams in the candidate sentence are present in the reference sentences
    - *n*-grams not present in the reference sentences should not be in the candidate sentence
    - *n*-grams that commonly occur across all images in the dataset should be given lower weight, since they are likely to be less informative

  In practice: perform a **Term Frequency Inverse Document Frequency (TF-IDF)** (Robertson, 2004) weighting for each *n*-gram

Vedantam et al., [CIDEr: Consensus-based Image Description Evaluation](), CVPR 2015

[Slide: Narayan-Chen, Xie, Shan]

	CIDEr-D	Meteor	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	date
Watson Multimodal[46]	1.123	0.268	0.559	0.773	0.609	0.461	0.344	2016-11-16
MSM@MSRA[29]	1.049	0.266	0.552	0.751	0.588	0.449	0.343	2016-10-25
G-RMI(PG-SPIDEr-TAG)[17]	1.042	0.255	0.551	0.751	0.591	0.445	0.331	2016-11-11
MetaMind/VT_GT[25]	1.042	0.264	0.55	0.748	0.584	0.444	0.336	2016-12-01
ATT-IMG (MSM@MSRA)[5]	1.023	0.262	0.551	0.752	0.59	0.449	0.34	2016-06-13
G-RMI (PG-BCMR)[16]	1.013	0.257	0.55	0.754	0.591	0.445	0.332	2016-10-30
DONOT_FAIL_AGAIN[13]	1.01	0.262	0.542	0.734	0.564	0.425	0.32	2016-11-22
DLTC@MSR[12]	1.003	0.257	0.543	0.74	0.575	0.436	0.331	2016-09-04
Postech_CV[38]	0.987	0.255	0.539	0.743	0.575	0.431	0.321	2016-06-13
feng[15]	0.986	0.255	0.54	0.743	0.578	0.434	0.323	2016-11-06
...								
Human[21]	0.854	0.252	0.484	0.663	0.469	0.321	0.217	2015-03-23

According to CIDEr, humans are in 38th place!! 😱

[MSCOCO challenge leaderboard]

[Slide: Narayan-Chen, Xie, Shan]