بسم الله الرحمن الرحیم

یادگیری عمیق

جلسه ۱۹ و ۲۰

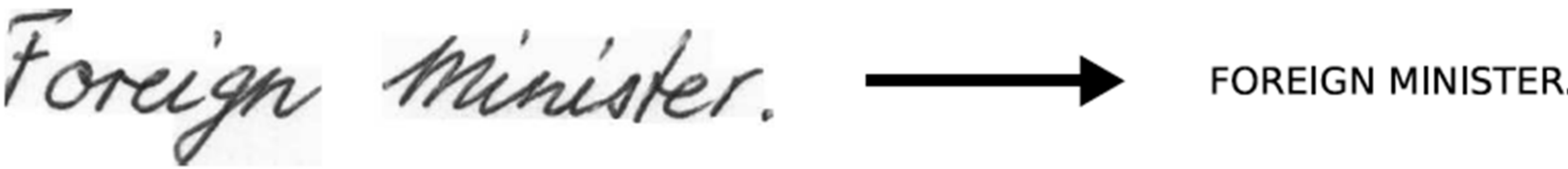
شبکه‌های عصبی بازگشتی

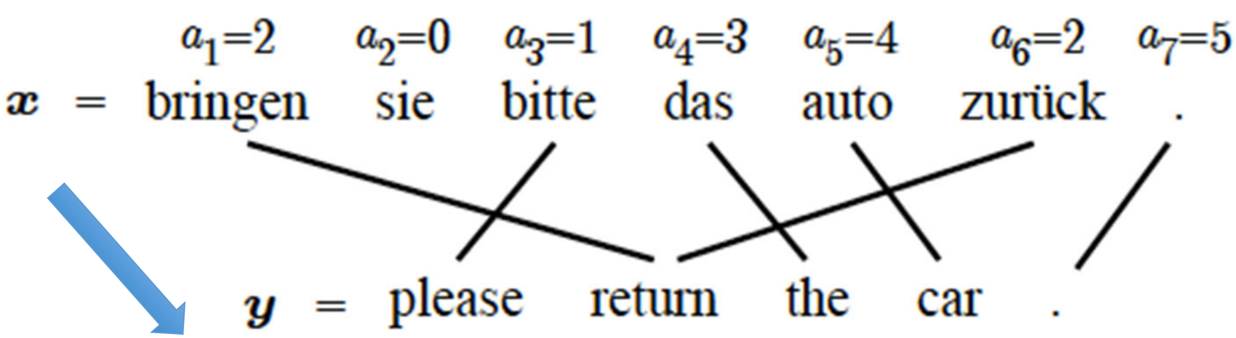
Recurrent Neural Networks (RNN)

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

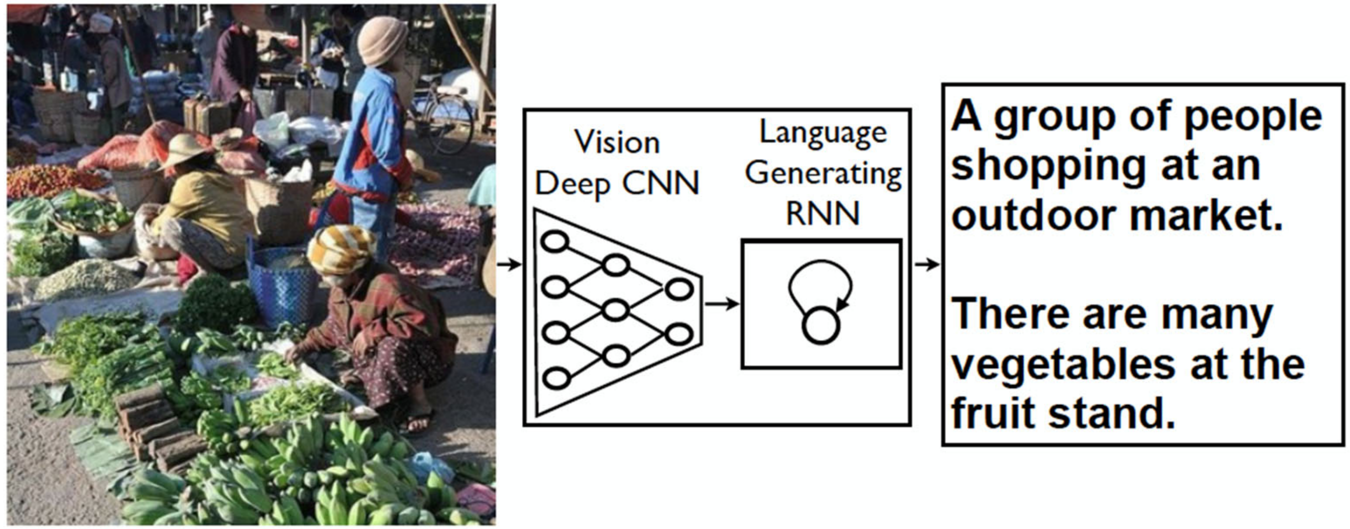
http://courses.fouladi.ir/deep

Sequences are everywhere…

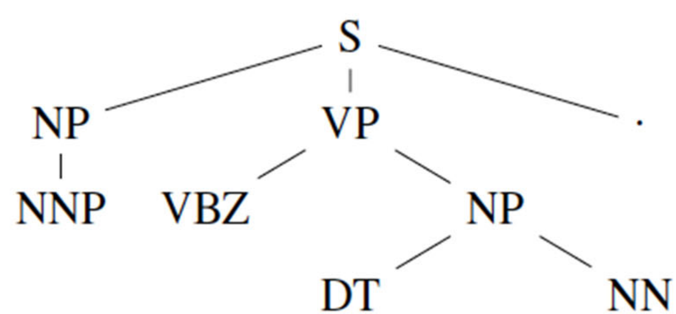




Even where you might not expect a sequence…



John has a dog . →



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

شبکه‌های عصبی بازگشتی

۱

مقدمات

دنباله‌ها

SEQUENCES

دنباله: داده‌های بعدی به داده‌های قبلی وابستگی دارند.

هدف: پیش‌بینی اینکه چه چیزی بعداً می‌آید؟

$$\Pr(x) = \prod_i \Pr(x_i | x_1, \dots, x_{i-1})$$

در نظر گرفتن تکه‌های x_i $\Leftarrow$
تعداد پارامتر کمتر + مدل‌سازی ساده‌تر + امکان تعمیم به طول دلخواه

معمولاً به‌جای طول دلخواه، یک قاب (frame) به‌طول T برداشته می‌شود:

$$\Pr(x) = \prod_i \Pr(x_i | x_{i-T}, \dots, x_{i-1})$$

دنباله‌ها

ویژگی‌ها

SEQUENCES

- داده‌های داخل یک دنباله، iid نیستند.
  - یعنی مستقل و دارای توزیع یکسان نیستند.
- کلمه‌ی بعدی، وابسته به کلمه‌های قبلی است.
  - به‌صورت ایده‌آل، به همه‌ی کلمه‌های قبلی وابسته است.
- برای تحلیل دنباله نیازمند **مضمون** (context) و نیز **حافظه** (memory) هستیم.

مدل کردن مضمون و حافظه

مثال (۱ از ۲)

Modelling Context and Memory

I am Bond, James

McGuire
Bond
tired
am
!

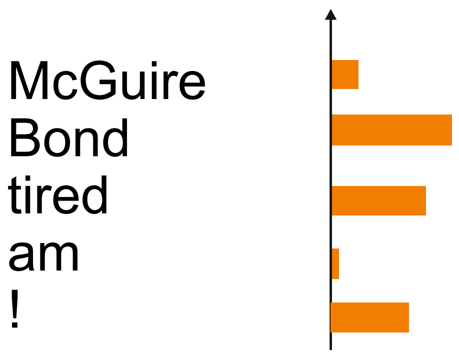
کلمه‌ی بعدی کدام باید باشد؟

مدل کردن مضمون و حافظه

مثال (۲ از ۲)

Modelling Context and Memory

I am Bond, James $\boxed{\text{Bond}}$



Ref: https://uvadlc.github.io/

مدل کردن مضمون و حافظه

بردارهای تک–داغ

ONE-HOT VECTORS

$$x_i \equiv \text{one-hot vector}$$

برداری که همه‌ی عناصر آن صفر است، غیر از یک مقدار 1 برای بعد فعال آن

برای مثال: اگر ۱۲ کلمه در یک دنباله داشته باشیم، ۱۲ بردار تک–داغ خواهیم داشت.

Vocabulary	One-Hot Vectors									
I	I	1	I	0	I	0	I	0	I	0
am	am	0	am	1	am	0	am	0	am	0
Bond	Bond	0	Bond	0	Bond	1	Bond	0	Bond	0
James	James	0	James	0	James	0	James	1	James	0
tired	tired	0	tired	0	tired	0	tired	0	tired	1
,	,	0	,	0	,	0	,	0	,	0
McGuire	McGuire	0	McGuire	0	McGuire	0	McGuire	0	McGuire	0
!	!	0	!	0	!	0	!	0	!	0

پس از ایجاد بردارهای تک–داغ، یک روش جاسازی (مثل Word2Vec یا GloVE) اعمال می‌شود.

Ref: https://uvadlc.github.io/

مدل زبان

LANGUAGE MODEL

مدل زبان
Language Model

مدل زبان احتمال هر دنباله از کلمات آن زبان را بیان می‌کند.

$$P(w_1, w_2, \dots, w_n) \;\; = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \times \cdots \times P(w_n|w_1, w_2, \dots, w_{n-1})$$

کاربردها:

ترتیب‌دهی کلمات
Word Ordering

p(the cat is small) > p(small the is cat)
مورد استفاده در ترجمه‌ی ماشینی

گزینش کلمات
Word Choice

p(walking home after school) > p(walking house after school)
p(he likes apple) > p(he licks apple)
مورد استفاده در بازشناسی گفتار / تولید گفتار

MEMORY

حافظه، بازنمایی گذشته است.

اطلاعات در گام زمانی t با استفاده از پارامتر θ بر روی یک فضای نهفته‌ی c_t افکنده می‌شود.

از اطلاعات افکنده شده، از لحظه‌ی t در لحظه‌ی $t+1$ استفاده‌ی مجدد می‌شود:

$$c_{t+1} = h(x_{t+1}, c_t; \theta)$$

پارامتر بازگشتی θ برای تمام گام‌های زمانی به‌اشتراک گذاشته می‌شود:

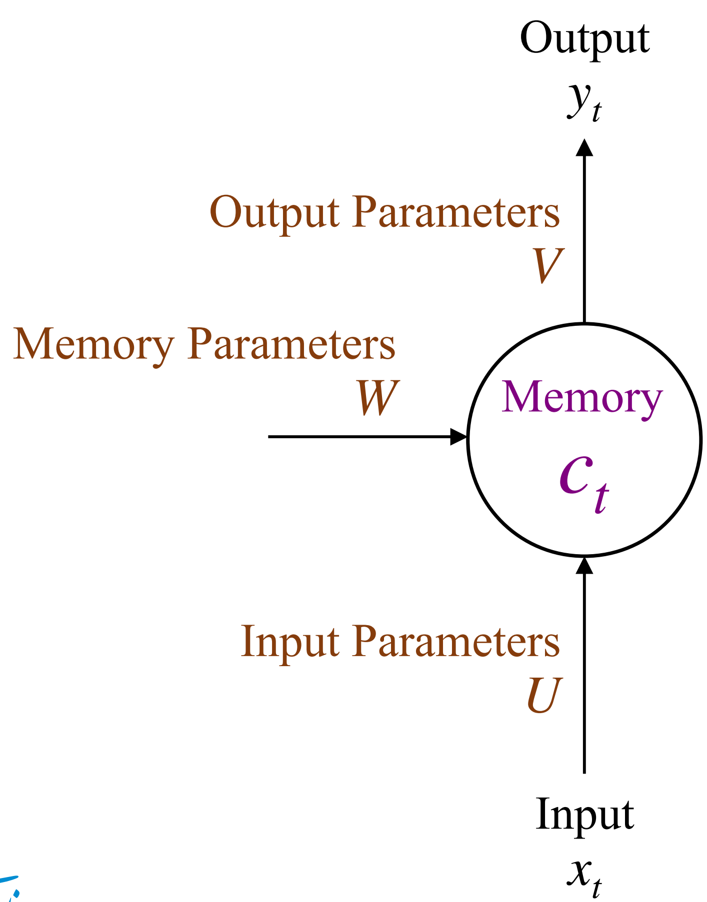
$$c_{t+1} = h(x_{t+1}, h(x_t, h(x_{t-1}, \dots h(x_1, c_0; \theta); \theta); \theta); \theta)$$

مدل کردن مضمون و حافظه

مدل کردن حافظه در قالب یک گراف

Memory as a Graph

سادە‌ترین مدل

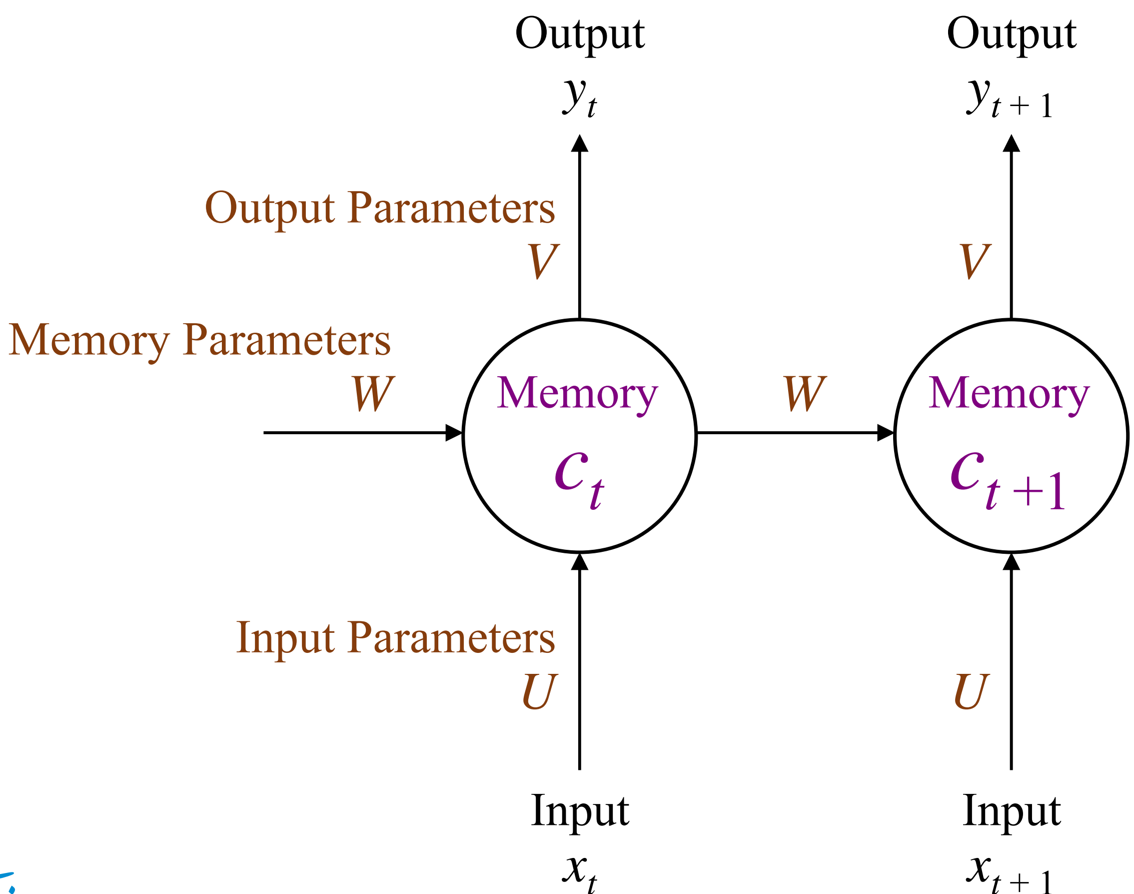


مدل کردن مضمون و حافظه

مدل کردن حافظه در قالب یک گراف

[Memory as a Graph](https://uvadlc.github.io/)

سادهترین مدل



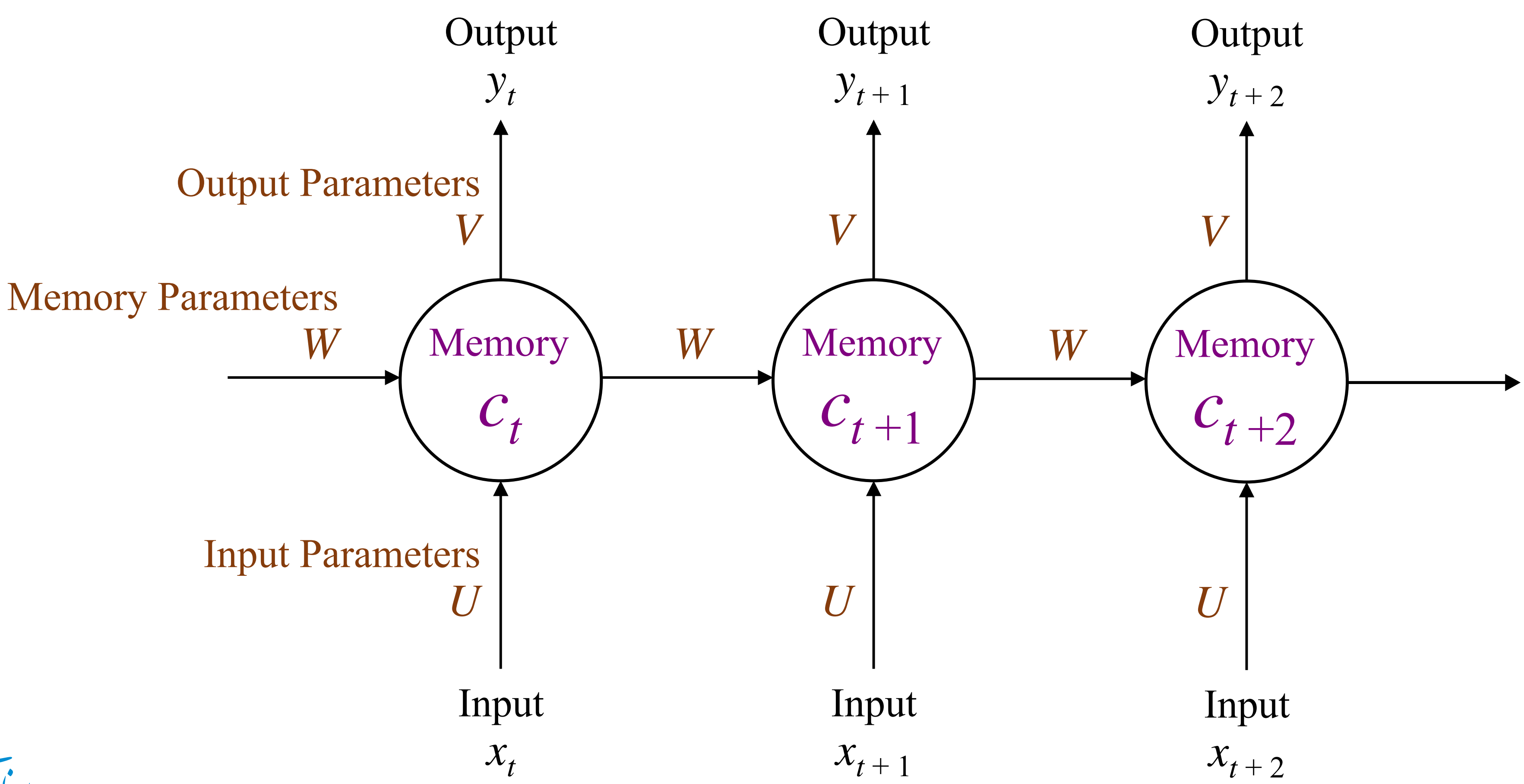
Ref: https://uvadlc.github.io/

مدل کردن مضمون و حافظه

مدل کردن حافظه در قالب یک گراف

Memory as a Graph

سادەترین مدل



Ref: https://uvadlc.github.io/

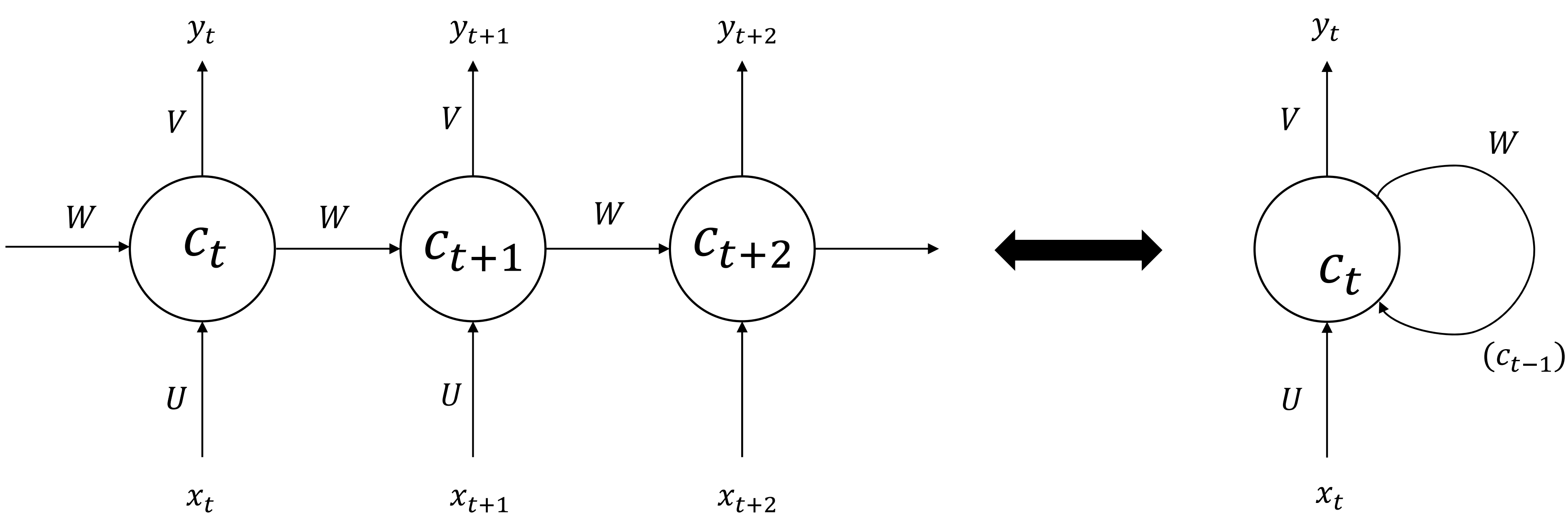
مدل کردن مضمون و حافظه

تا کردن حافظه

Folding the Memory

شبکه‌ی باز شده / تا نشده
Unrolled / Unfolded Network

شبکه‌ی تا شده
Folded Network



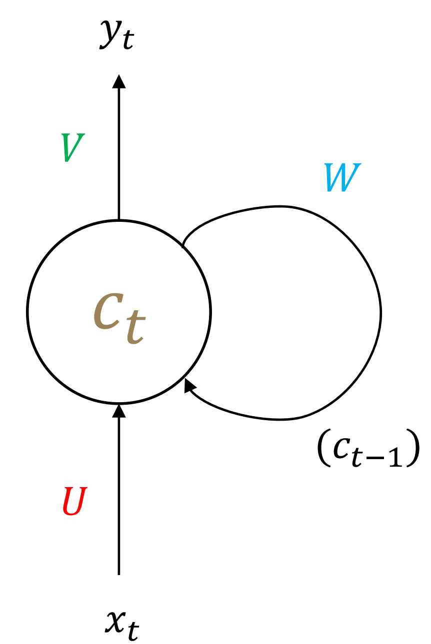
Ref: https://uvadlc.github.io/

شبکه‌ی عصبی بازگشتی

Recurrent Neural Network (RNN)

شبکه‌ی عصبی بازگشتی، با استفاده از دو معادله تعریف می‌شود:

$$c_t = \tanh(U\, x_t + W\, c_{t-1})$$
$$y_t = \mathrm{softmax}(V\, c_t)$$



Ref: https://uvadlc.github.io/

شبکه‌ی عصبی بازگشتی

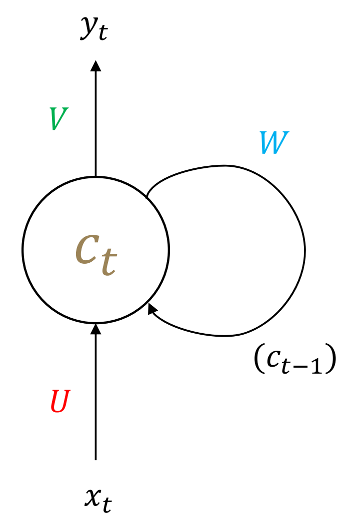
مثال

Recurrent Neural Network (RNN)

- Vocabulary of 5 words
- A memory of 3 units [Hyperparameter that we choose like layer size]
  - c_t: [3×1], W: [3×3]
- An input projection of 3 dimensions
  - U: [3×5]
- An output projections of 10 dimensions
  - V: [10×3]

$$c_t = \tanh(U\,x_t + W\,c_{t-1})$$

$$y_t = \mathrm{softmax}(V\,c_t)$$



$$U \cdot x_{t=4} = \begin{bmatrix} 0.1 & -0.3 & 1.2 & 0.6 & -0.8 \\ -0.2 & 0.4 & 0.5 & 0.9 & -0.1 \\ -0.1 & 0.2 & -0.7 & -0.8 & 0.3 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.9 \\ -0.8 \end{bmatrix} = U^{(4)}$$

Ref: https://uvadlc.github.io/

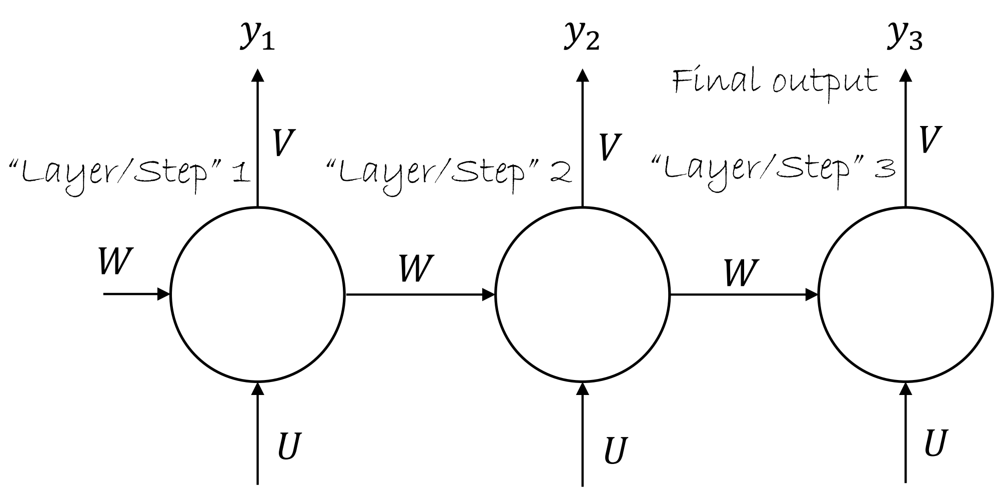
شبکه‌ی عصبی بازگشتی

مقایسه‌ی شبکه‌ی بسته شده با شبکه‌ی چندلایه

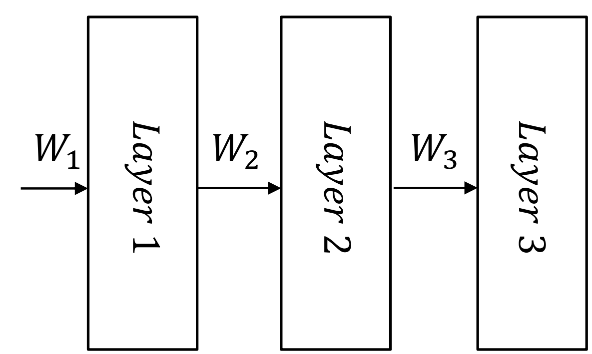
Rolled Network vs. Multi-Layer Network

تفاوت‌ها:

- در شبکه‌ی بسته شده، گام‌ها به‌جای لایه‌ها نقش بازی می‌کنند.
- در شبکه‌ی بسته شده، پارامترهای گام مشترک است؛ در حالی که در شبکه‌ی چندلایه، پارامترها متفاوت است.



3-gram unrolled recurrent network



3-layer neural network

Ref: https://uvadlc.github.io/

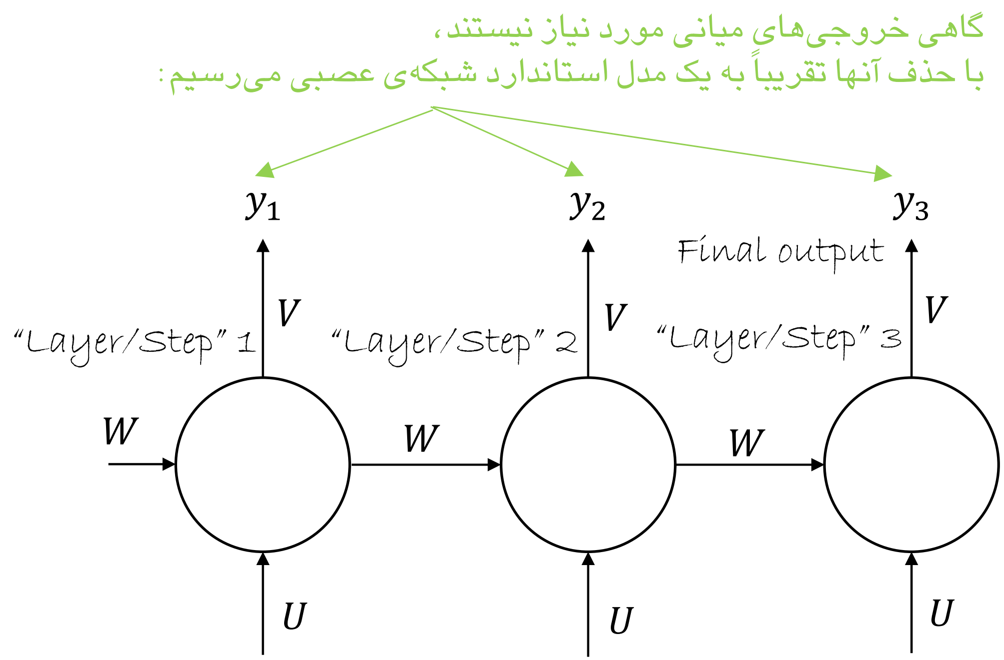
شبکه‌ی عصبی بازگشتی

مقایسه‌ی شبکه‌ی بسته شده با شبکه‌ی چندلایه

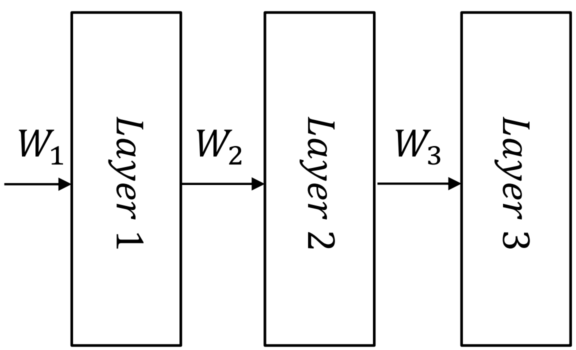
ROLLED NETWORK VS. MULTI-LAYER NETWORK

تفاوت‌ها:

- در شبکه‌ی بسته شده، گام‌ها به‌جای لایه‌ها نقش بازی می‌کنند.
- در شبکه‌ی بسته شده، پارامترهای گام مشترک است؛ در حالی که در شبکه‌ی چندلایه، پارامترها متفاوت است.



3-gram unrolled recurrent network



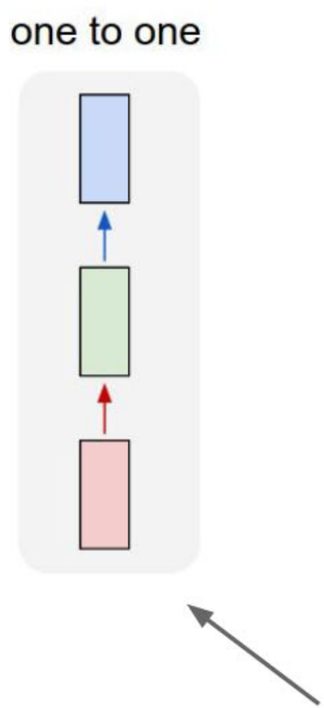
3-layer neural network

Ref: https://uvadlc.github.io/

پردازش دنباله‌ها با استفاده از شبکه‌های عصبی بازگشتی

یک بردار به یک بردار

RECURRENT NEURAL NETWORKS: PROCESS SEQUENCES



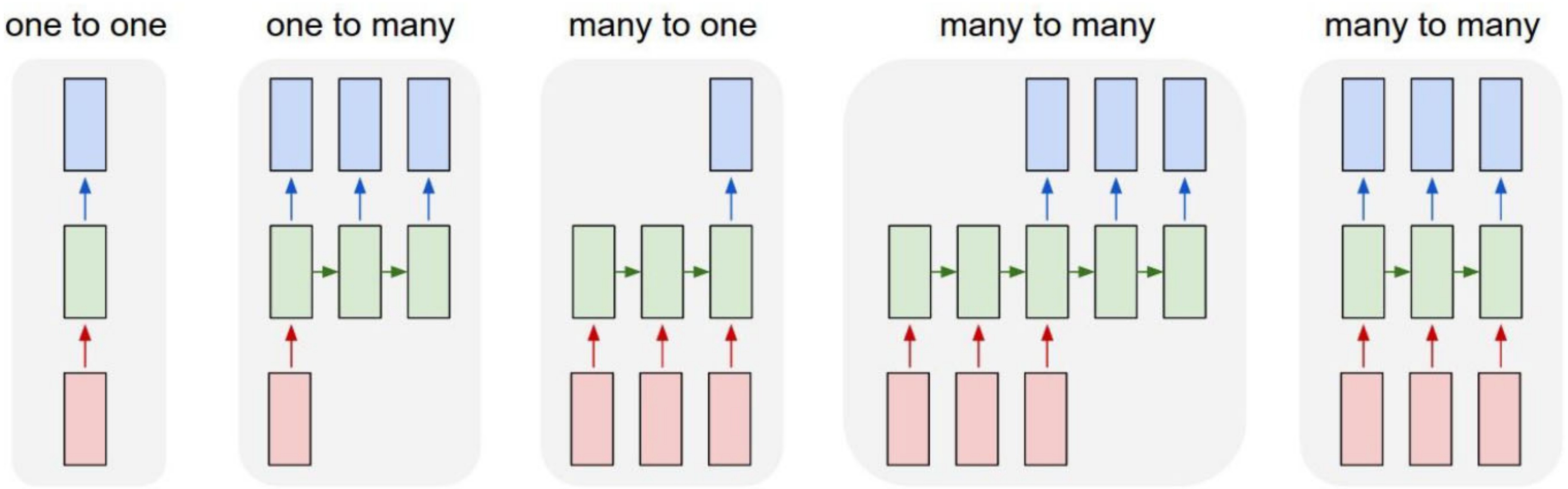
Vanilla Neural Networks

شبکه‌های عصبی ساده:

یک بردار به یک بردار

پردازش دنباله‌ها با استفاده از شبکه‌های عصبی بازگشتی

یک بردار به چند بردار



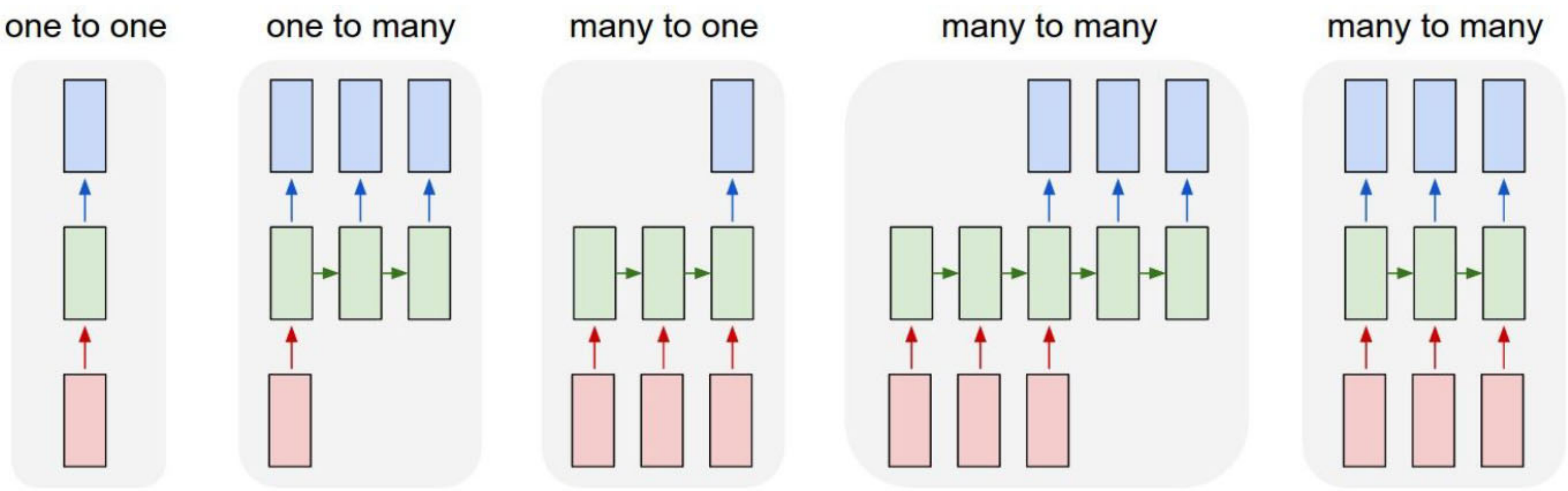
e.g. **Image Captioning**
image -> sequence of words

مانند: عنوان‌گذاری تصاویر
تصویر ← دنباله‌ی کلمات

Ref: http://cs231n.stanford.edu/

پردازش دنباله‌ها با استفاده از شبکه‌های عصبی بازگشتی

چند بردار به یک بردار

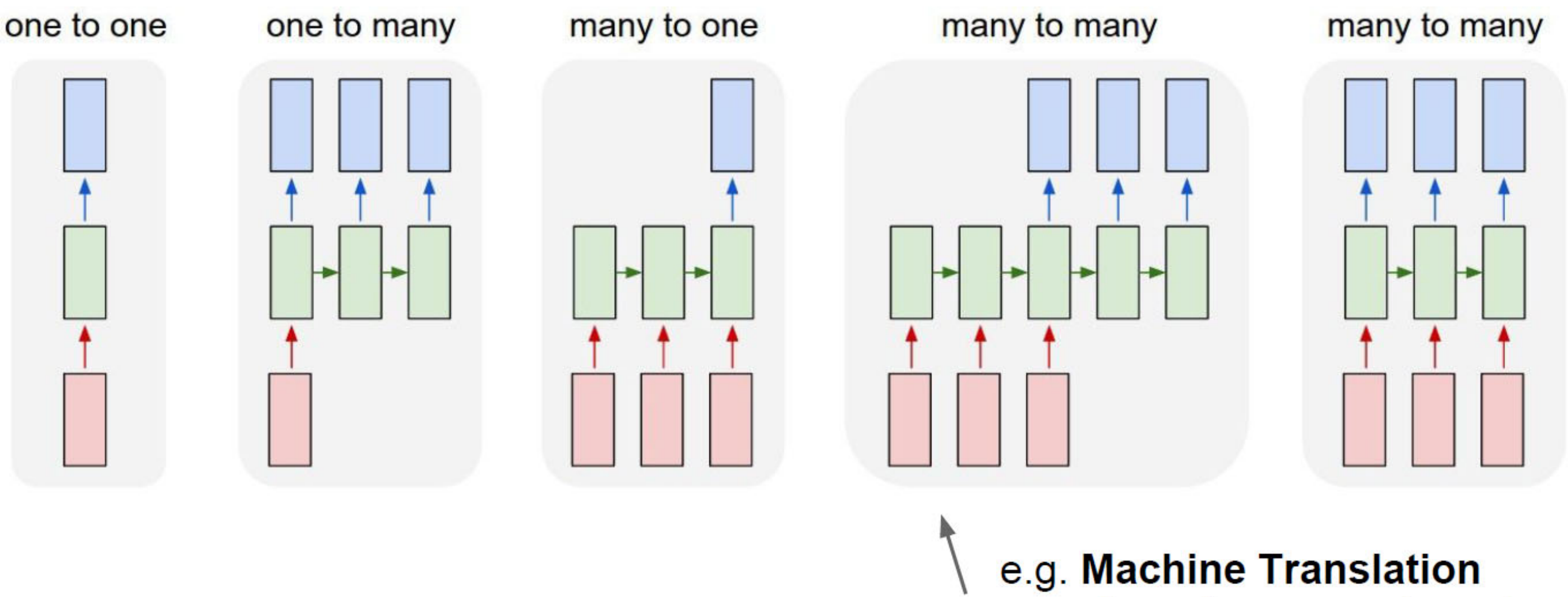


e.g. **Sentiment Classification**
sequence of words -> sentiment

مانند: طبقه‌بندی احساسات (نظرات)
دنباله‌ی کلمات ← احساس (نظر)

پردازش دنباله‌ها با استفاده از شبکه‌های عصبی بازگشتی

چند بردار به چند بردار



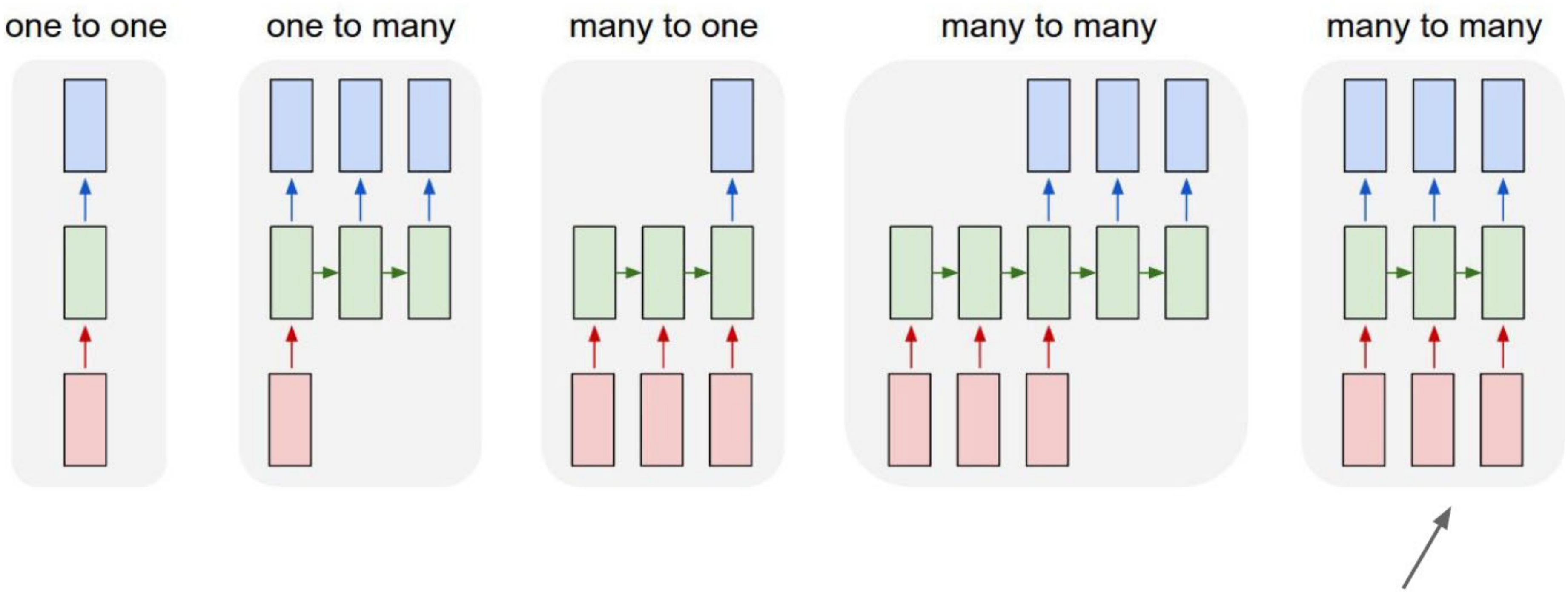
e.g. **Machine Translation**
seq of words -> seq of words

مانند: ترجمه‌ی ماشینی
دنباله‌ی کلمات ← دنباله‌ی کلمات

Ref: http://cs231n.stanford.edu/

پردازش دنباله‌ها با استفاده از شبکه‌های عصبی بازگشتی

چند بردار به چند بردار

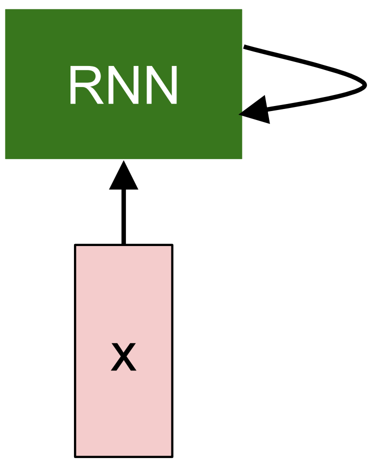


e.g. **Video classification on frame level**

مانند: طبقه‌بندی ویدئوها در سطح فریم

شبکه‌های عصبی بازگشتی

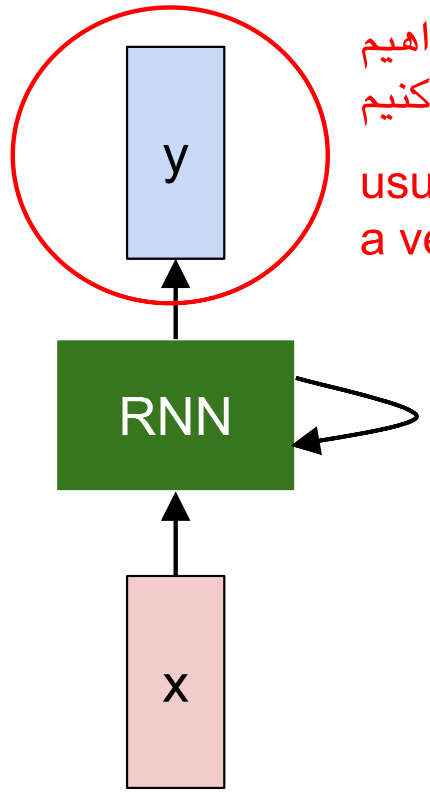
Recurrent Neural Network



Ref: http://cs231n.stanford.edu/

شبکه‌های عصبی بازگشتی

Recurrent Neural Network



معمولاً می‌خواهیم
یک بردار را در چند گام زمانی پیش‌بینی کنیم

usually want to predict
a vector at some time steps

شبکه‌های عصبی بازگشتی

RECURRENT NEURAL NETWORK

می‌توانیم یک دنباله از بردارهای **x** را با اعمال یک **فرمول بازگشتی** در هر گام زمانی پردازش کنیم.

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

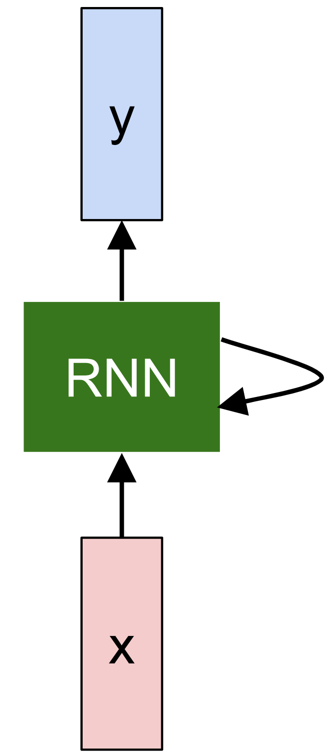
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W



Ref: http://cs231n.stanford.edu/

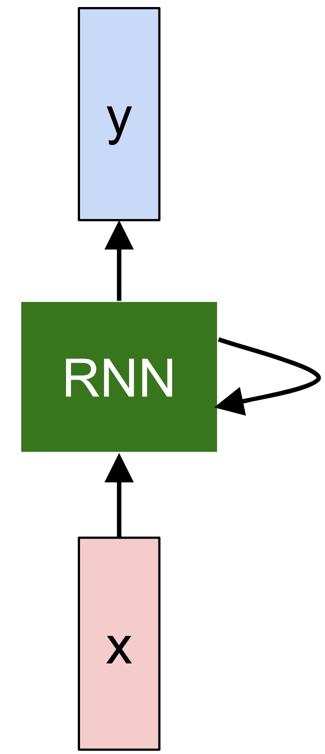
شبکه‌های عصبی بازگشتی

Recurrent Neural Network

می‌توانیم یک دنباله از بردارهای **x** را با اعمال یک **فرمول بازگشتی** در هر گام زمانی پردازش کنیم.

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.

توجه: تابع یکسان و مجموعه پارامترهای یکسانی در هر گام زمانی استفاده می‌شوند.

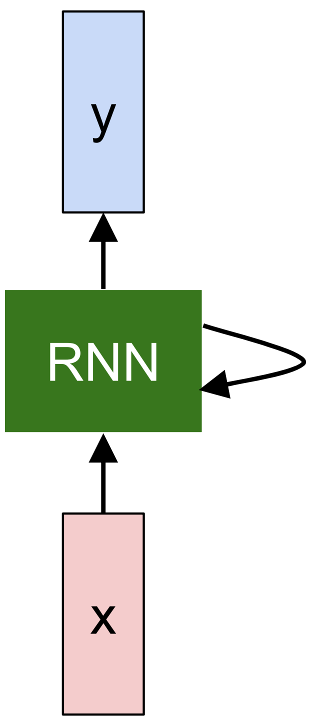
شبکه‌های عصبی بازگشتی

مدل ساده

(Vanilla) Recurrent Neural Network

حالت از یک بردار تنهای پنهان **h** تشکیل شده است.

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



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

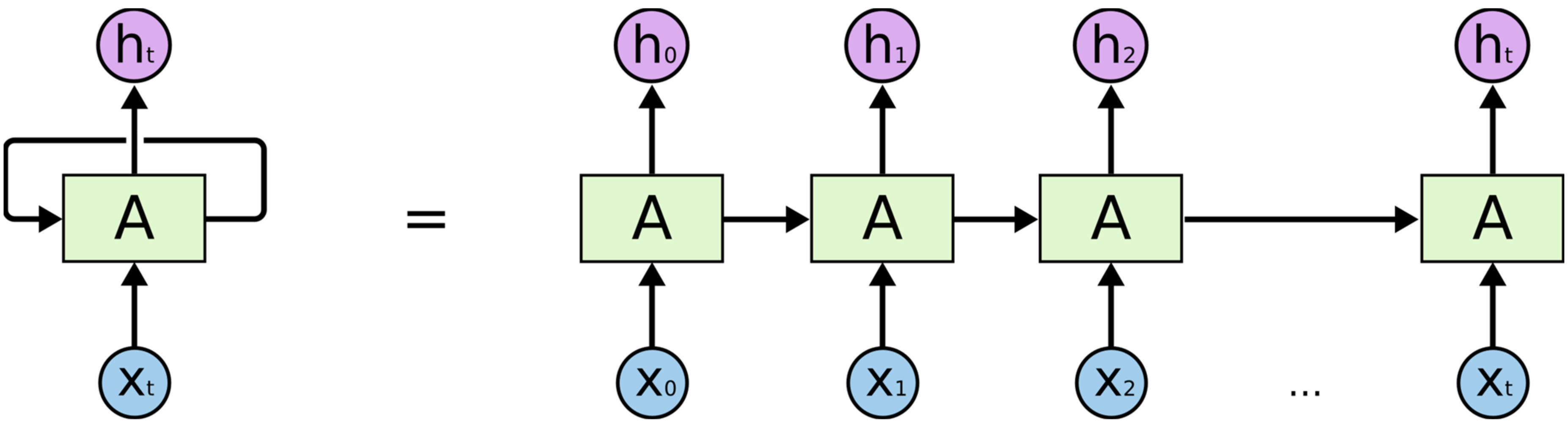
$$h_t = f_W(h_{t-1}, x_t)$$

مانند

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

شبکه‌های عصبی بازگشتی

باز کردن مدل ساده

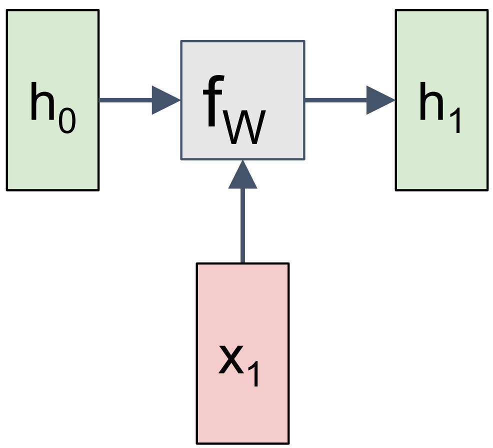


Ref: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

شبکه‌های عصبی بازگشتی

گراف محاساباتی

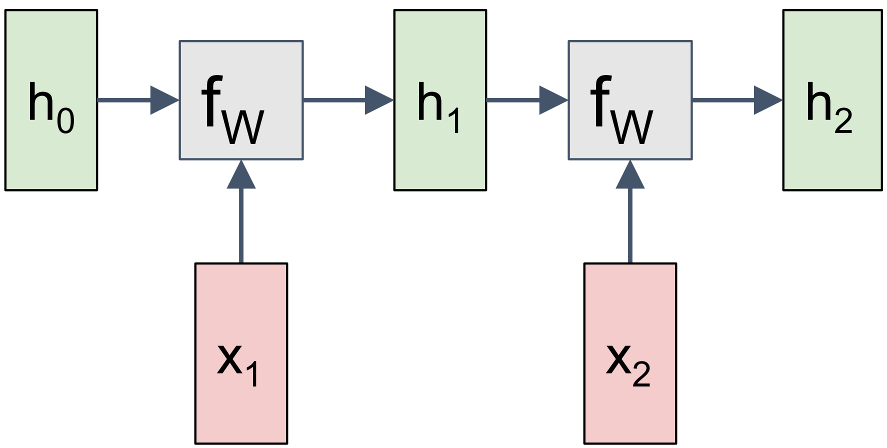
RNN: Computational Graph



شبکه‌های عصبی بازگشتی

گراف محاسباتی

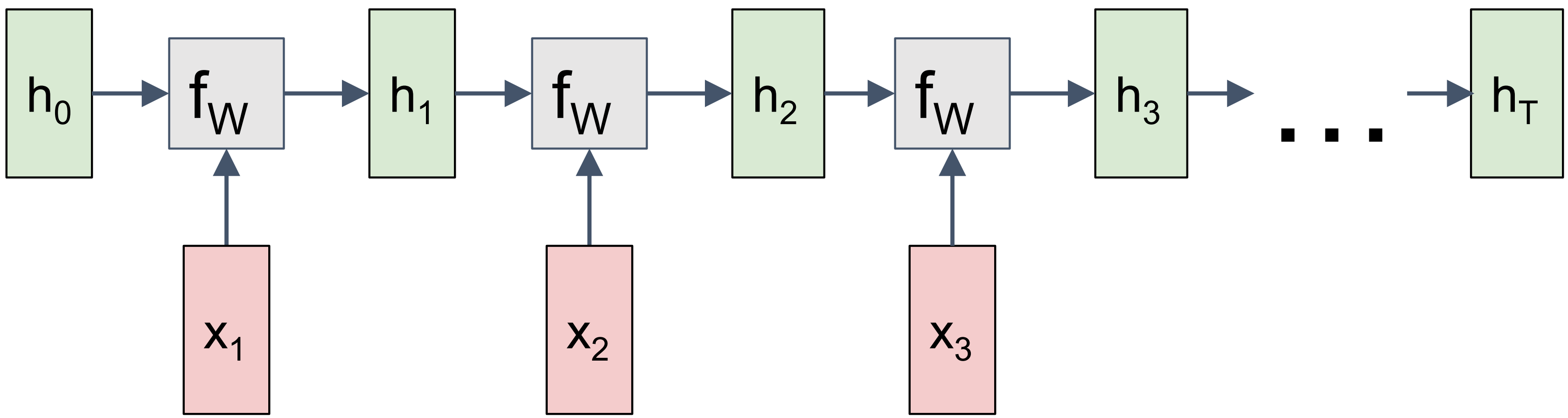
RNN: Computational Graph



شبکه‌های عصبی بازگشتی

گراف محاسباتی

RNN: Computational Graph

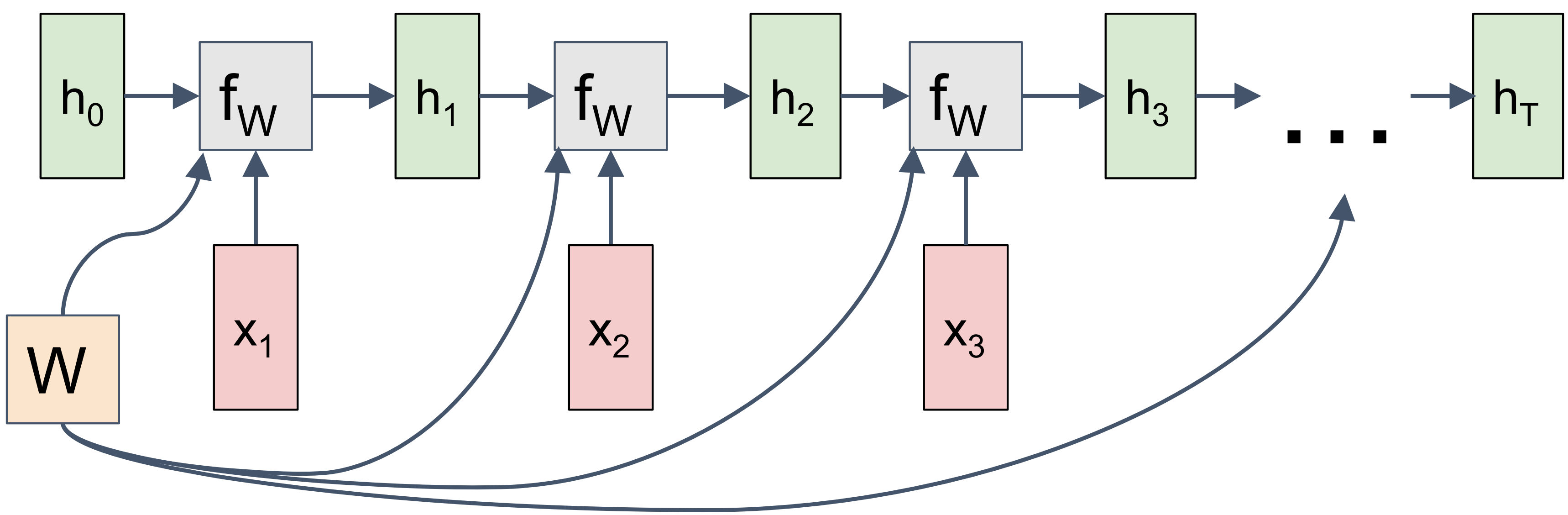


شبکه‌های عصبی بازگشتی

گراف محاسباتی: استفاده‌ی مجدد از یک ماتریس وزن یکسان در همه‌ی گام‌های زمانی

RNN: Computational Graph

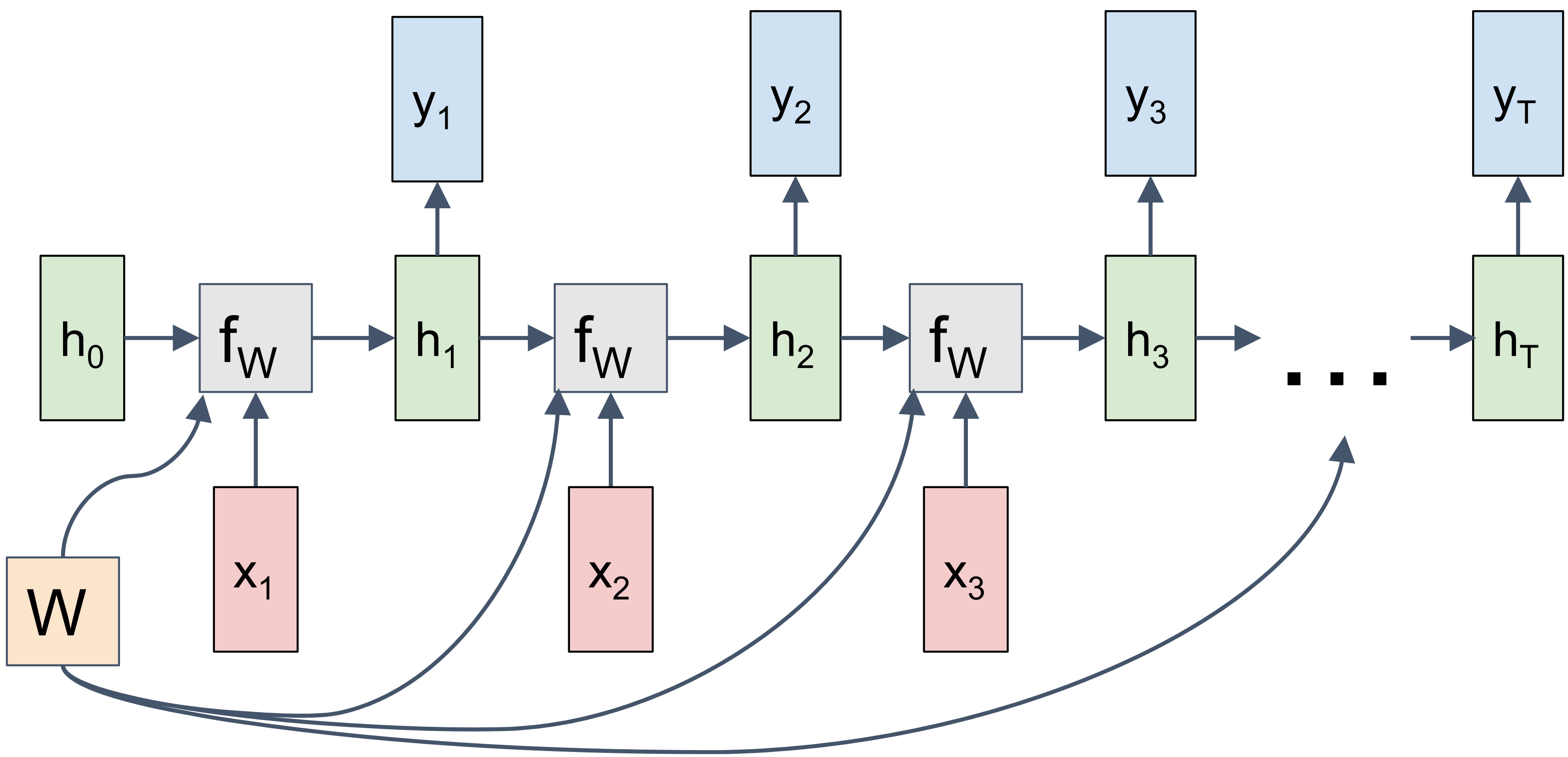
Re-use the same weight matrix at every time-step



شبکه‌های عصبی بازگشتی

گراف محاسباتی: چند به چند

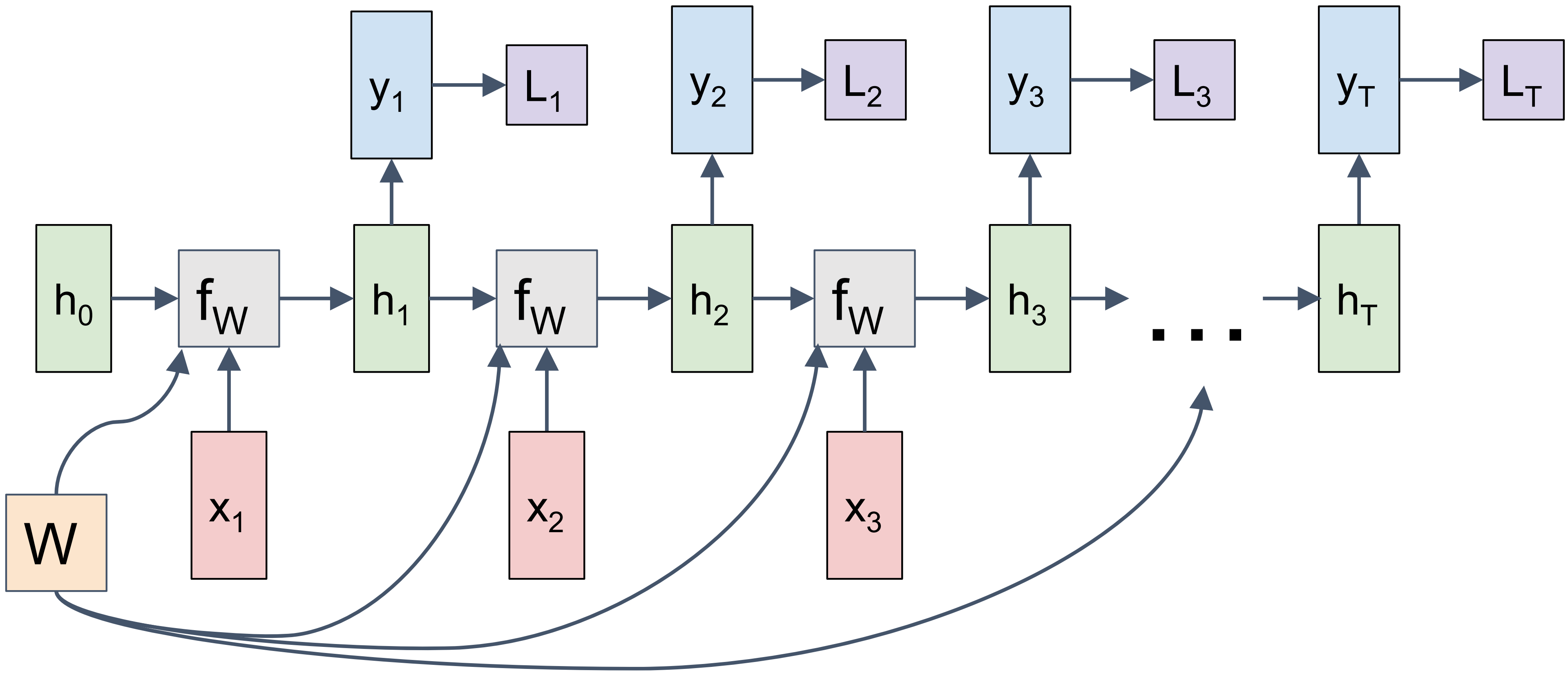
RNN: Computational Graph: Many to Many



شبکه‌های عصبی بازگشتی

گراف محاسباتی: چند به چند: محاسبه‌ی اتلاف‌ها

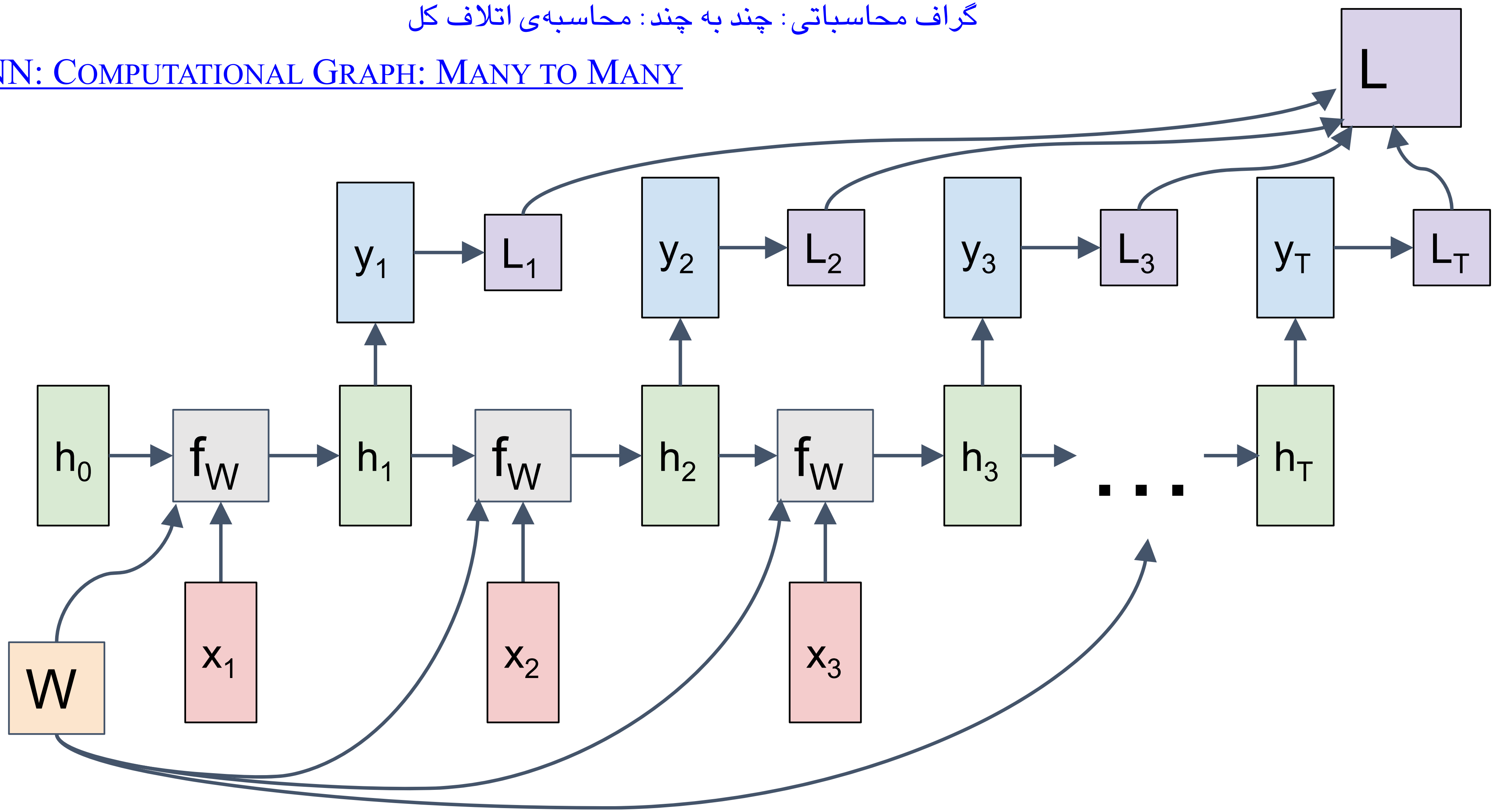
RNN: Computational Graph: Many to Many



شبکه‌های عصبی بازگشتی

گراف محاسباتی: چند به چند: محاسبه‌ی اتلاف کل

RNN: Computational Graph: Many to Many

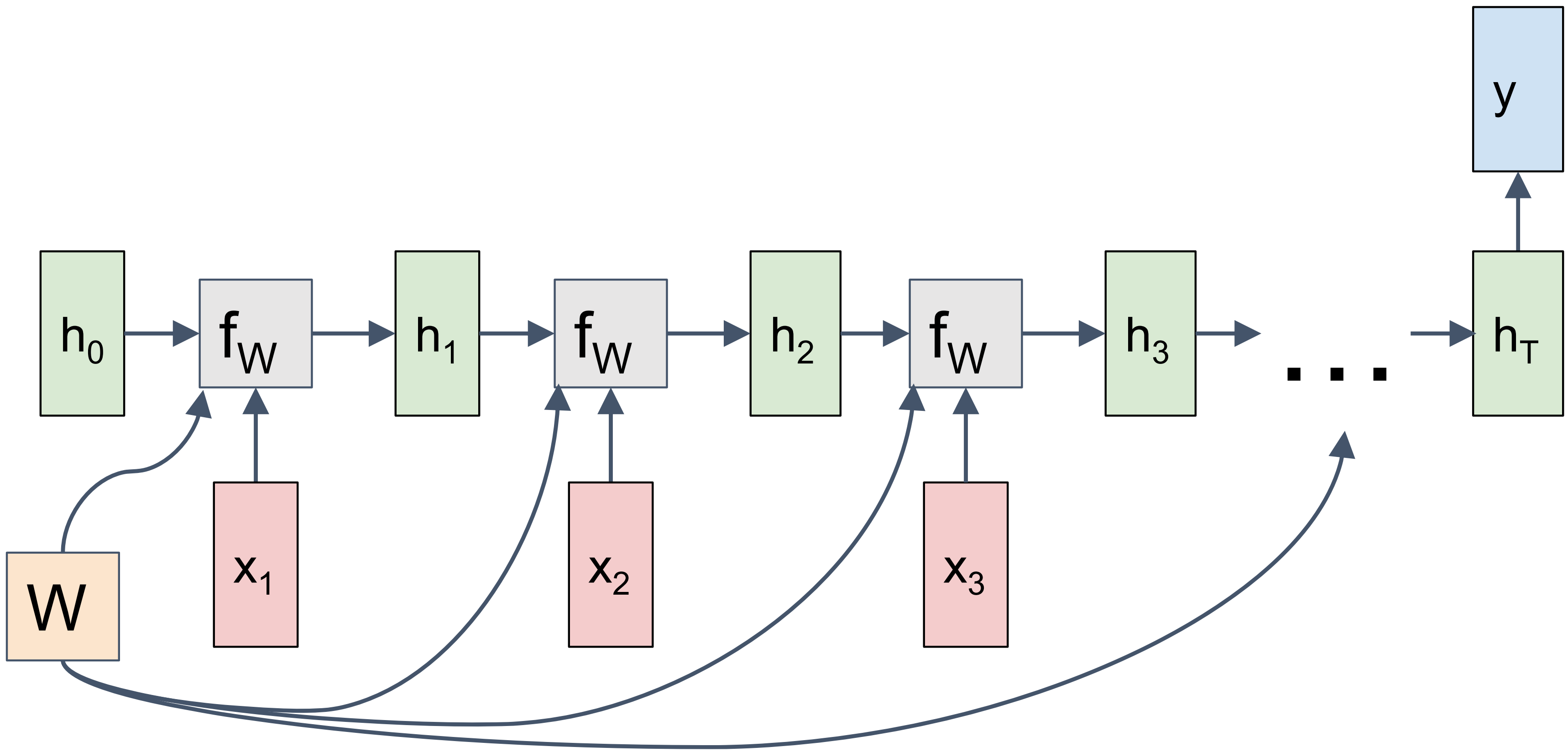


Ref: http://cs231n.stanford.edu/

شبکه‌های عصبی بازگشتی

گراف محاسباتی: چند به یک

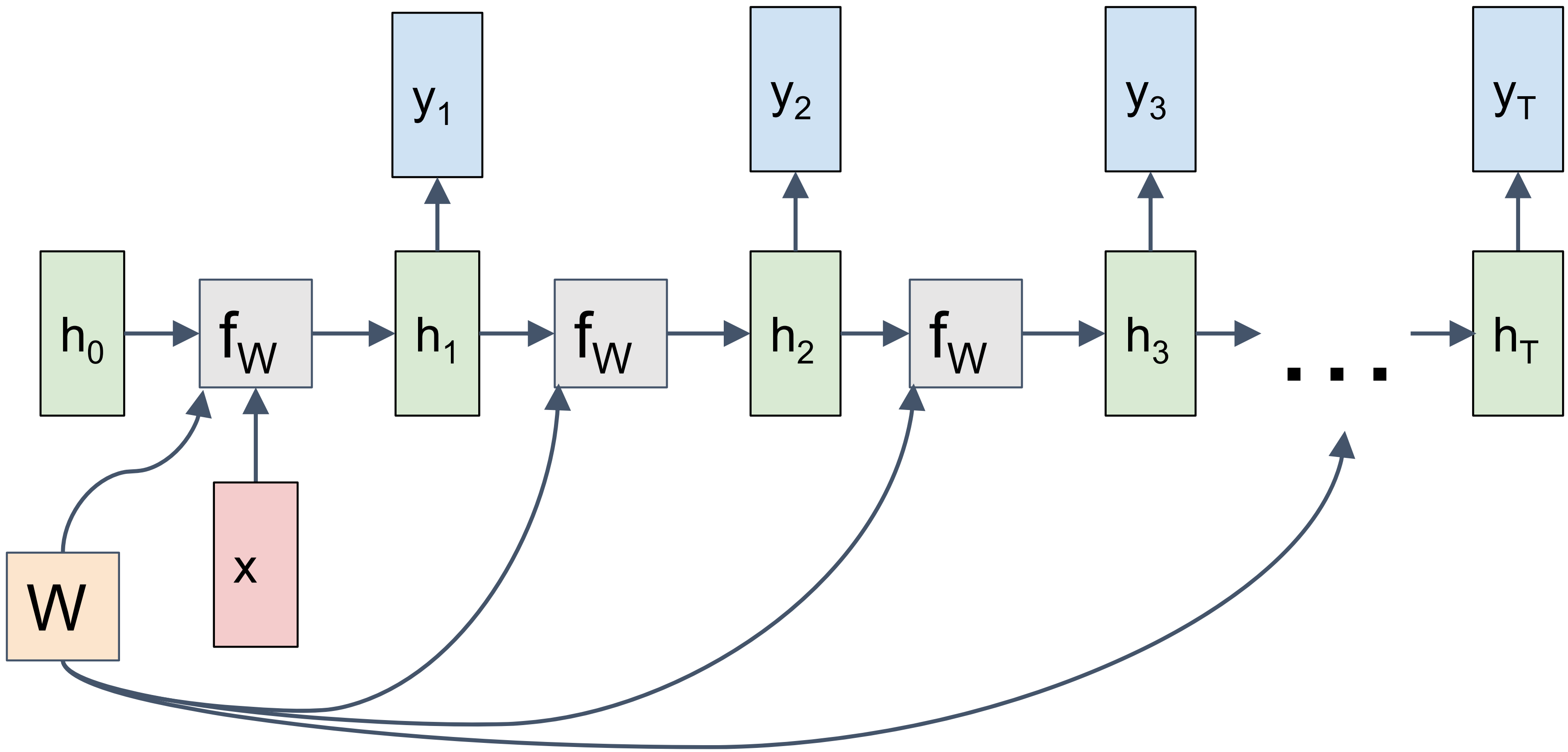
RNN: Computational Graph: Many to One



شبکه‌های عصبی بازگشتی

گراف محاسباتی: یک به چند

RNN: Computational Graph: One to Many



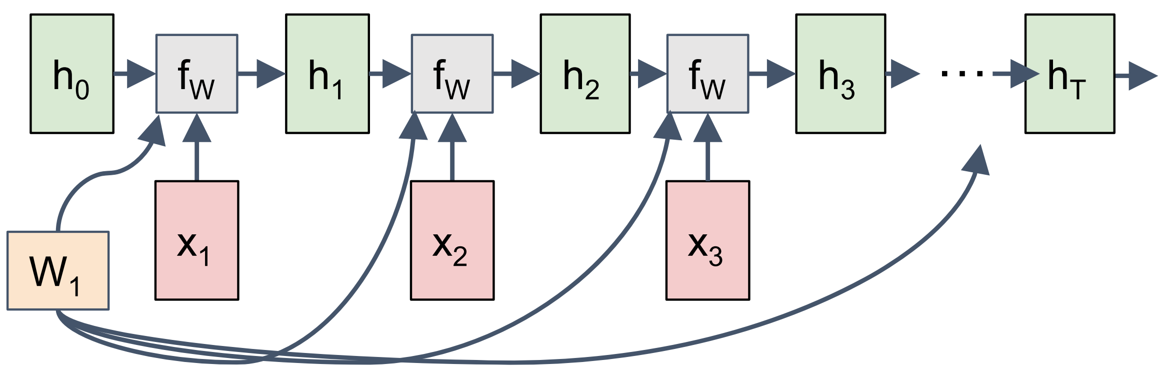
شبکه‌های عصبی بازگشتی

دنباله به دنباله: چند به یک + یک به چند

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

چند به یک:
کدگذاری دنباله‌ی ورودی به یک بردار تنها

Many to one:
Encode input sequence in a single vector



Ref: http://cs231n.stanford.edu/

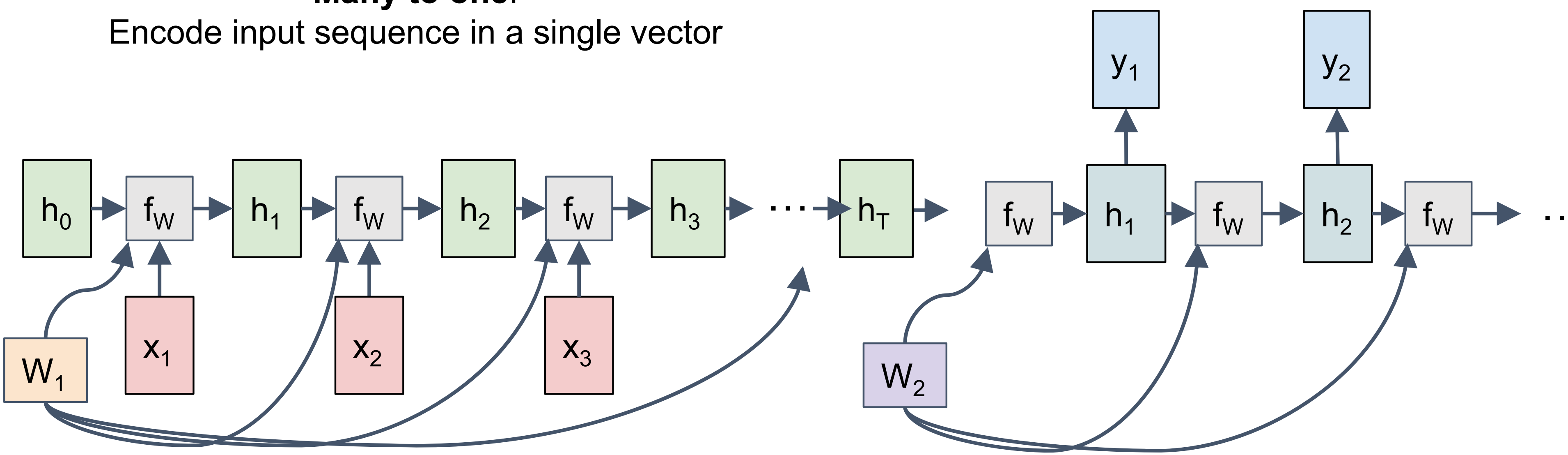
شبکه‌های عصبی بازگشتی

دنباله به دنباله: چند به یک + یک به چند

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

چند به یک:
کدگذاری دنباله‌ی ورودی به یک بردار تنها

Many to one:
Encode input sequence in a single vector



یک به چند:
تولید دنباله‌ی خروجی از روی یک بردار ورودی تنها

One to many:
Produce output sequence from single input vector

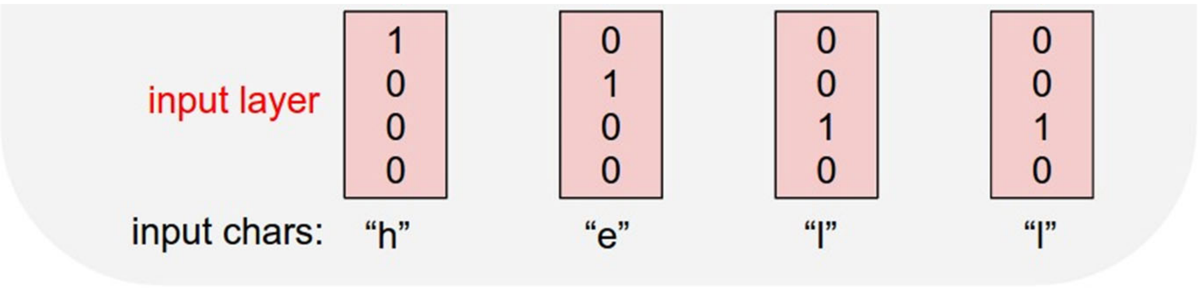
شبکه‌های عصبی بازگشتی

مثال : مدل زبان در سطح کاراکتر

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence: **“hello”**



هدف : پیش‌بینی کاراکتر بعدی با داشتن دنباله‌ای از کاراکترها

شبکه‌های عصبی بازگشتی

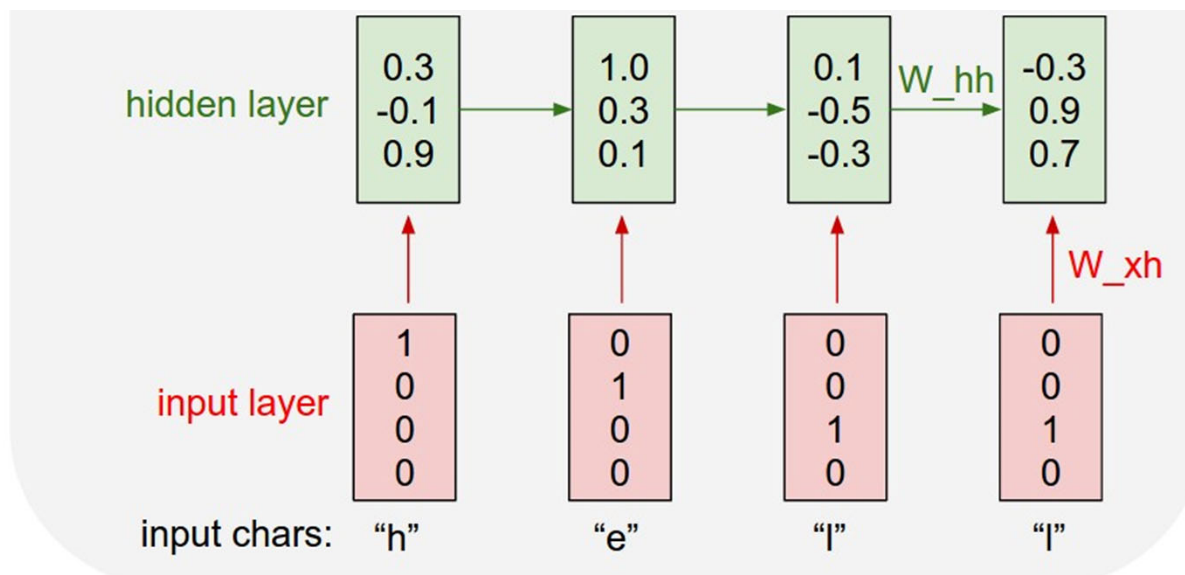
مثال: مدل زبان در سطح کاراکتر

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

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



هدف: پیش‌بینی کاراکتر بعدی با داشتن دنباله‌ای از کاراکترها

Ref: http://cs231n.stanford.edu/

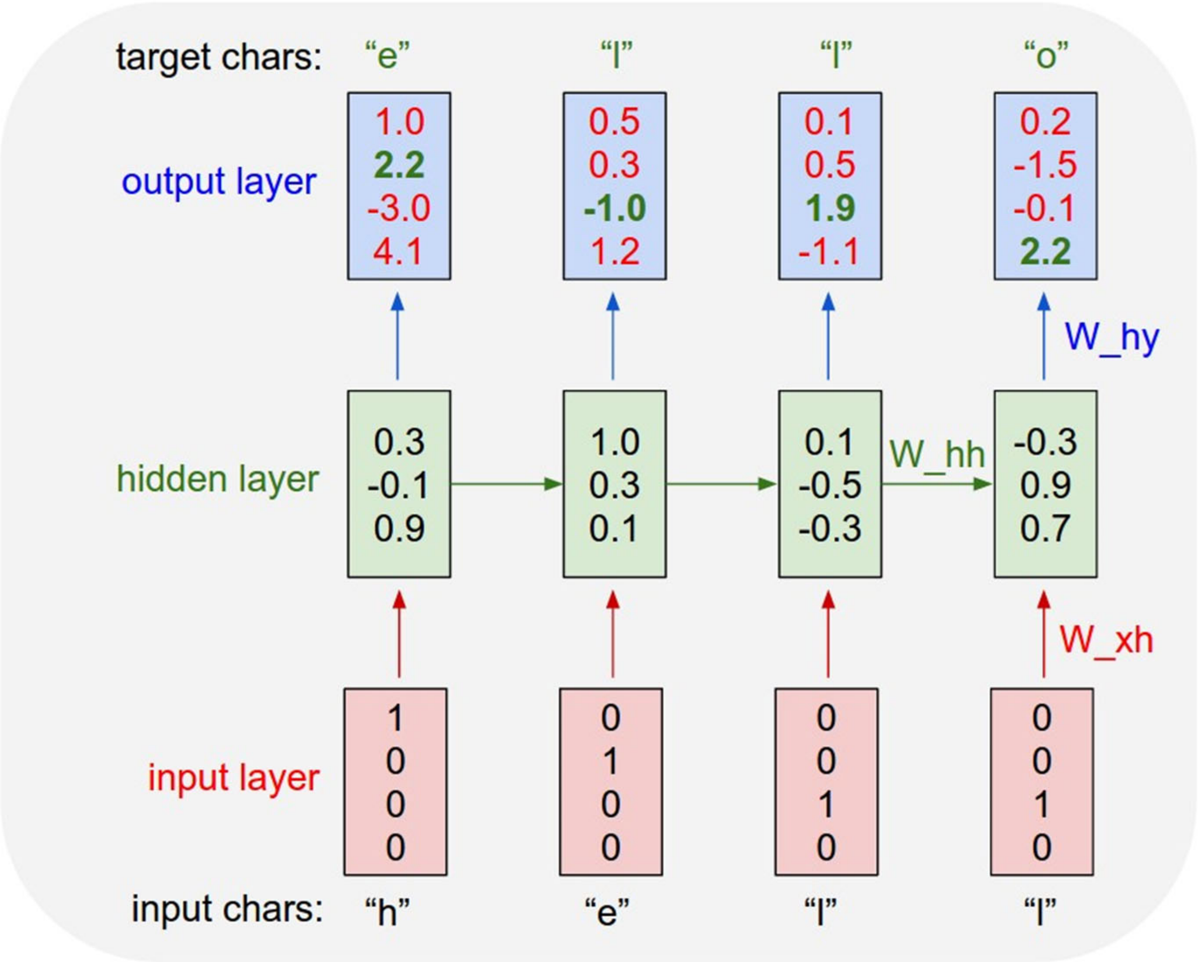
شبکه‌های عصبی بازگشتی

مثال: مدل زبان در سطح کاراکتر

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



هدف: پیش‌بینی کاراکتر بعدی با داشتن دنباله‌ای از کاراکترها

Ref: http://cs231n.stanford.edu/

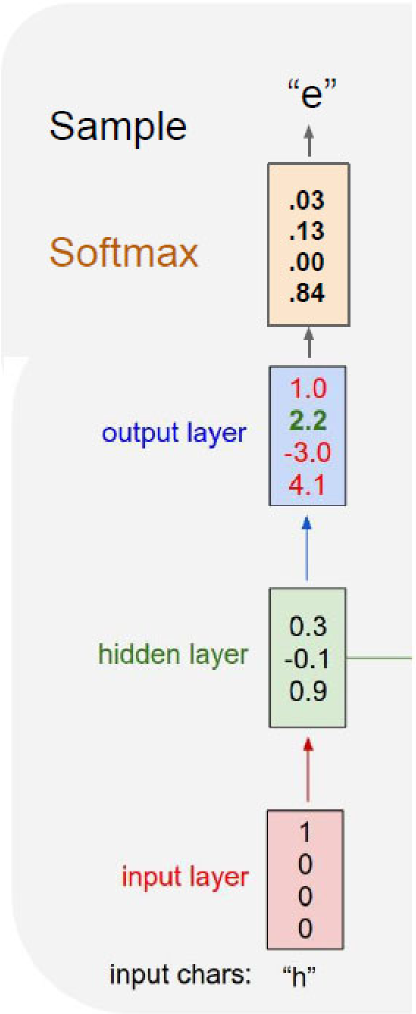
شبکه‌های عصبی بازگشتی

مثال: مدل زبان در سطح کاراکتر

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

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



در زمان آزمایش، در هر لحظه یکی از کاراکترهای نمونه به مدل فیدبک می‌شود.

Ref: http://cs231n.stanford.edu/

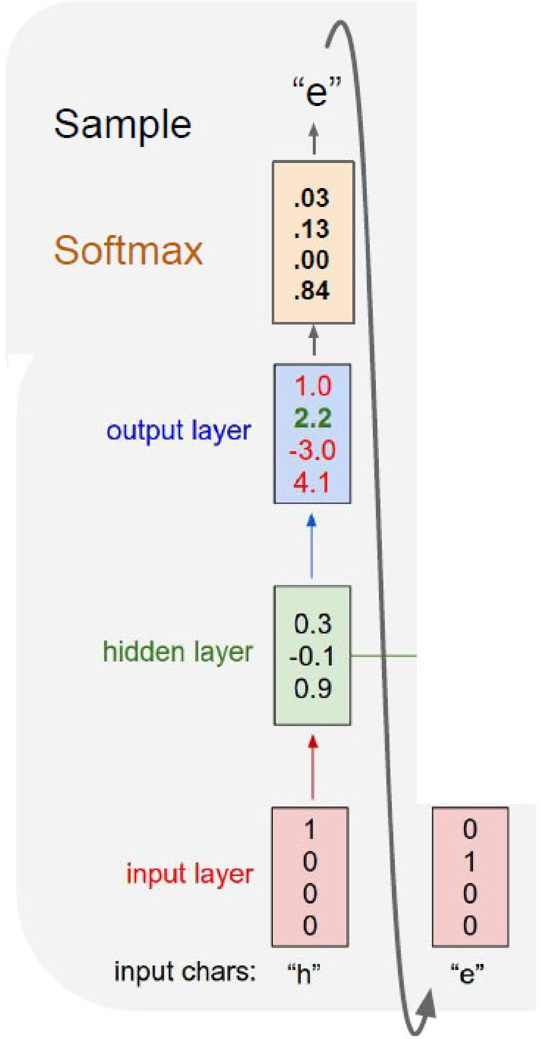
شبکه‌های عصبی بازگشتی

مثال: مدل زبان در سطح کاراکتر

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

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



در زمان آزمایش، در هر لحظه یکی از کاراکترهای نمونه به مدل فیدبک می‌شود.

Ref: http://cs231n.stanford.edu/

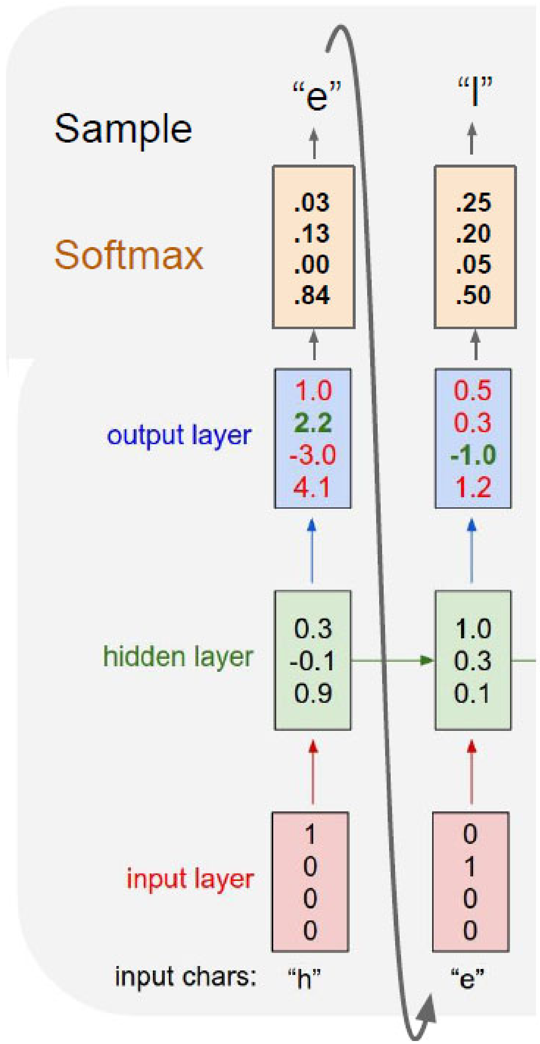
شبکه‌های عصبی بازگشتی

مثال: مدل زبان در سطح کاراکتر

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

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



در زمان آزمایش، در هر لحظه یکی از کاراکترهای نمونه به مدل فیدبک می‌شود.

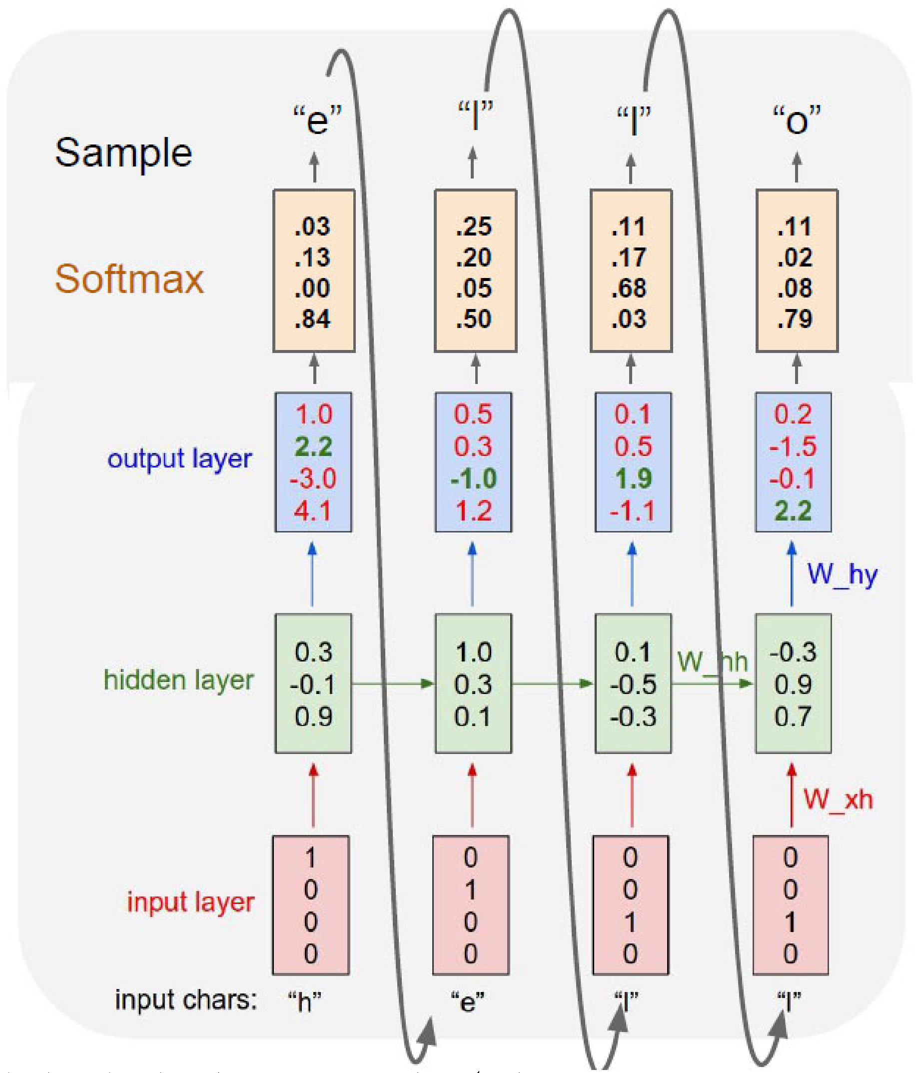
شبکه‌های عصبی بازگشتی

مثال: مدل زبان در سطح کاراکتر

Example:
Character-level Language Model

Vocabulary:
[h,e,l,o]

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



در زمان آزمایش، در هر لحظه یکی از کاراکترهای نمونه به مدل فیدبک می‌شود.

۲

نمونه کاربردهای شبکه‌های عصبی بازگشتی

یادگیری مدل زبان

مثال

```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]

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)

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

یادگیری مدل زبان

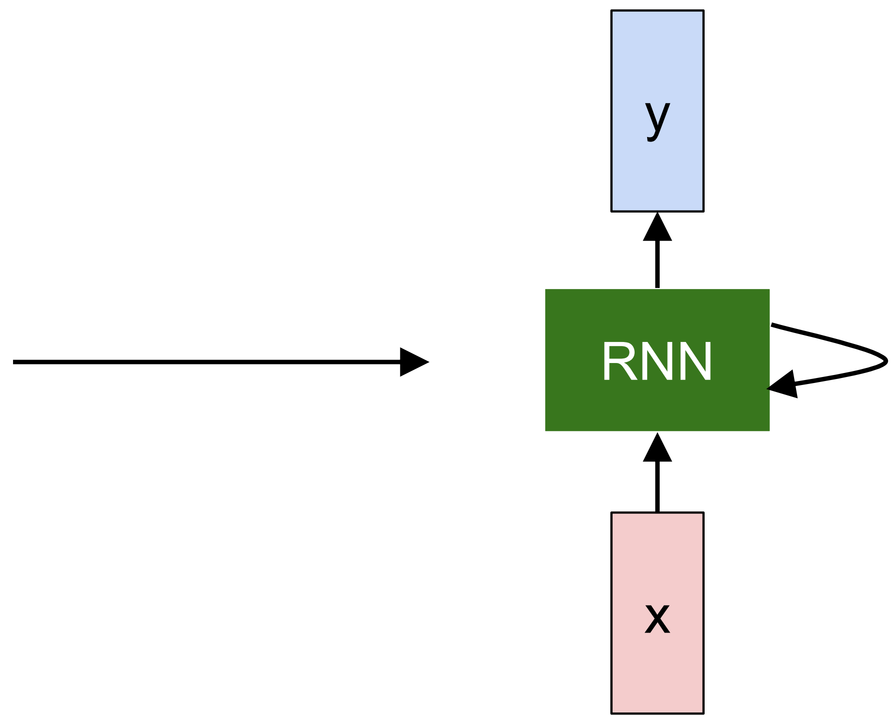
مثال

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.



Ref: http://cs231n.stanford.edu/

یادگیری مدل زبان

مثال

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.
```


Ref: http://cs231n.stanford.edu/

یادگیری مدل زبان

مثال

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

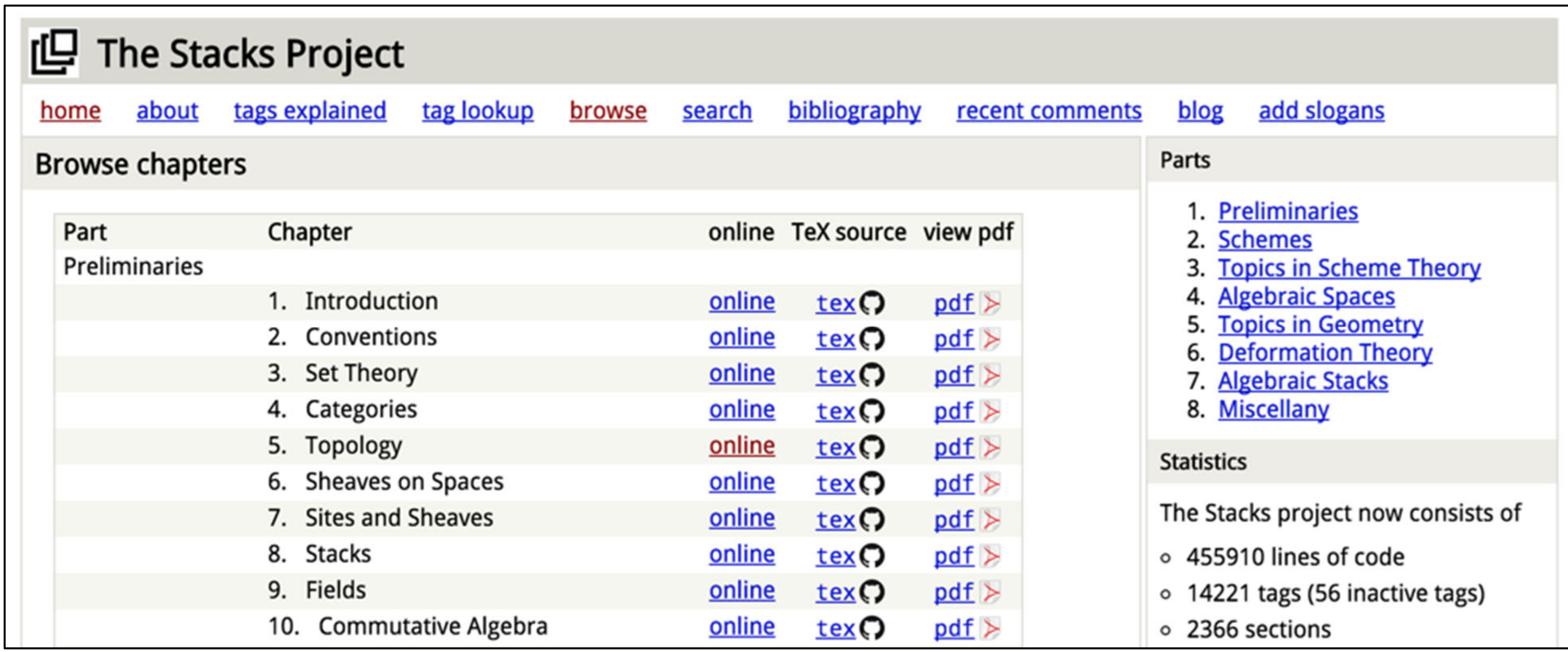
```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```


Ref: http://cs231n.stanford.edu/

یادگیری مدل زبان

مثال: پروژه‌ی استکس

[The Stacks Project: open source algebraic geometry textbook]


The Stacks Project

home about tags explained tag lookup browse search bibliography recent comments blog add slogans

Browse chapters

Part	Chapter	online	TeX source	view pdf
Preliminaries				
	1. Introduction	online	tex	pdf
	2. Conventions	online	tex	pdf
	3. Set Theory	online	tex	pdf
	4. Categories	online	tex	pdf
	5. Topology	online	tex	pdf
	6. Sheaves on Spaces	online	tex	pdf
	7. Sites and Sheaves	online	tex	pdf
	8. Stacks	online	tex	pdf
	9. Fields	online	tex	pdf
	10. Commutative Algebra	online	tex	pdf

Parts

1. Preliminaries
2. Schemes
3. Topics in Scheme Theory
4. Algebraic Spaces
5. Topics in Geometry
6. Deformation Theory
7. Algebraic Stacks
8. Miscellany

Statistics

The Stacks project now consists of

- 455910 lines of code
- 14221 tags (56 inactive tags)
- 2366 sections

LaTeX source

http://stacks.math.columbia.edu/

یادگیری مدل زبان

مثال

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

یادگیری مدل زبان

مثال

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

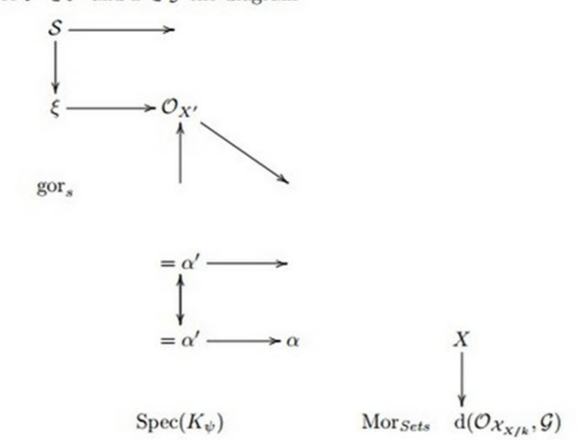
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \ \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

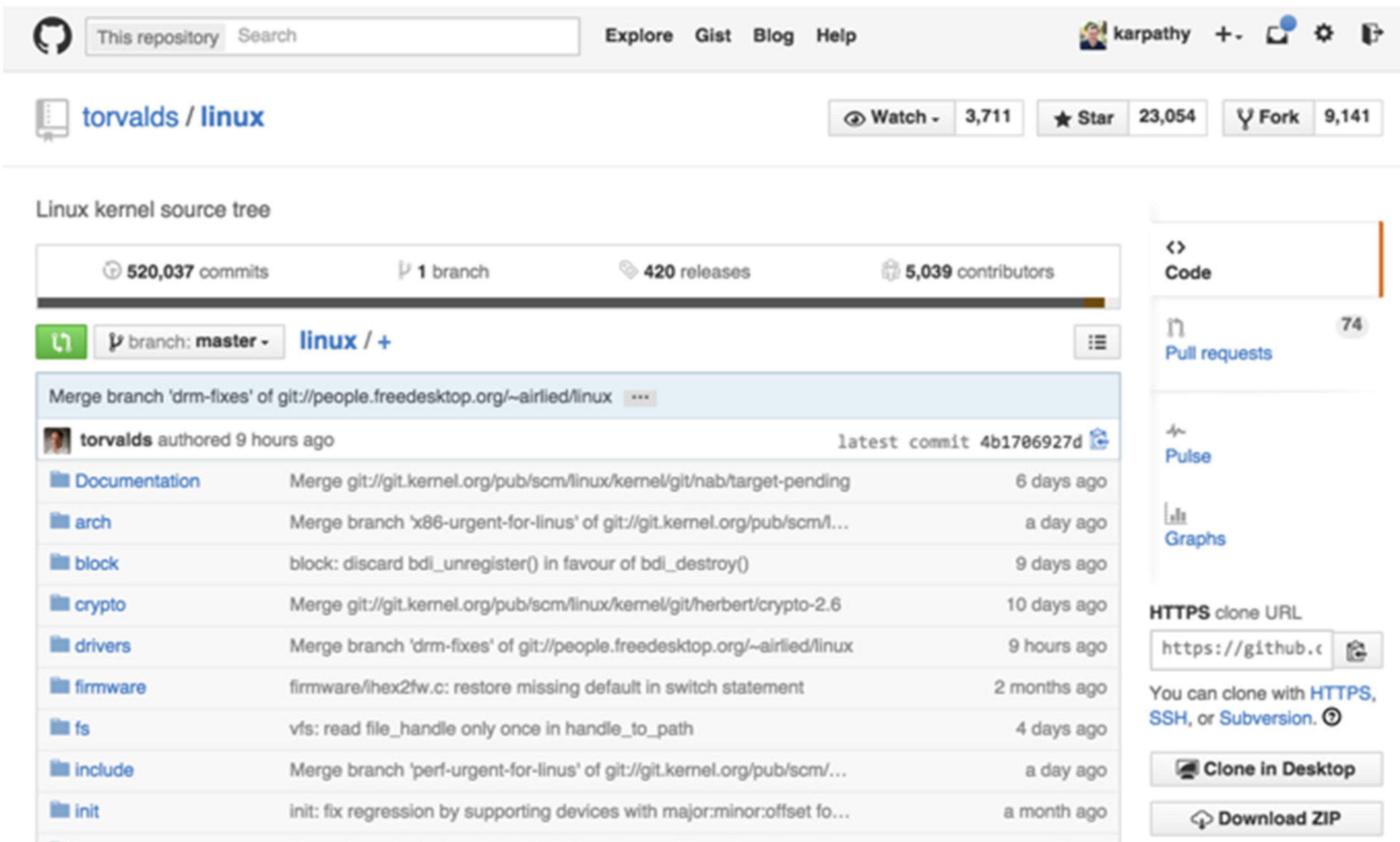
If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Prepared by Kazim Fouladi | Spring 2021 | 2nd Edition

Ref: http://cs231n.stanford.edu/

یادگیری مدل زبان

مثال



Prepared by Kazim Fouladi | Spring 2021 | 2nd Edition
کاظم فولادی

Ref: http://cs231n.stanford.edu/

یادگیری مدل زبان

مثال

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Generated C code

یادگیری مدل زبان

مثال

```
/*
 *  Copyright (c) 2006-2010, Intel Mobile Communications.  All rights reserved.
 *
 *   This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 *        This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.   See the
 *
 *  GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *    along with this program; if not, write to the Free Software Foundation,
 *  Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>
```

یادگیری مدل زبان

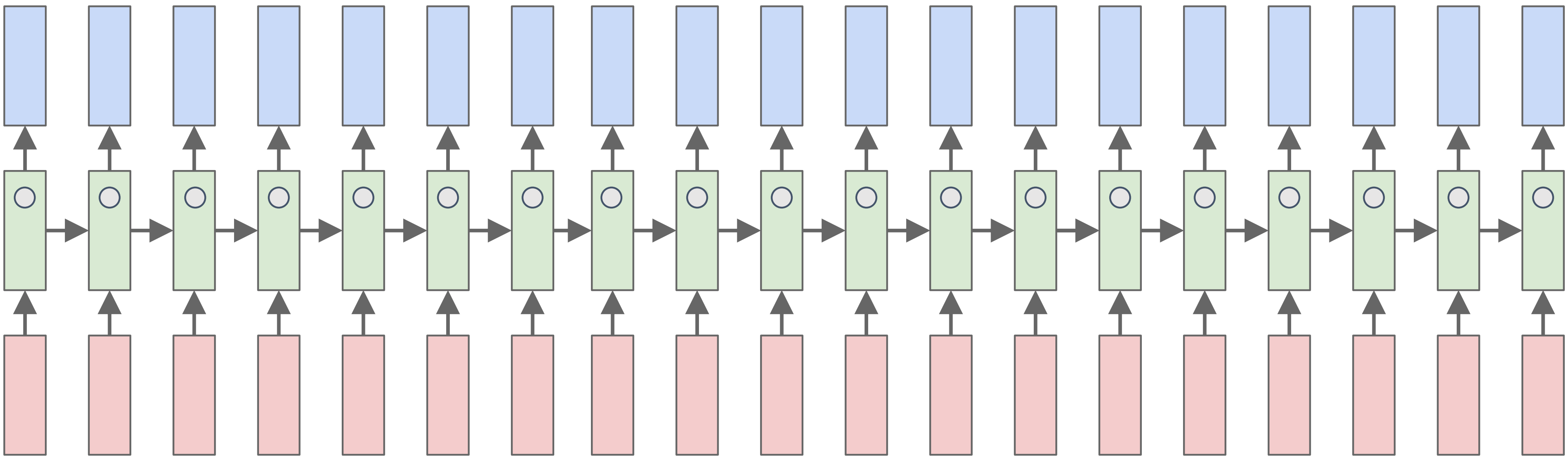
مثال

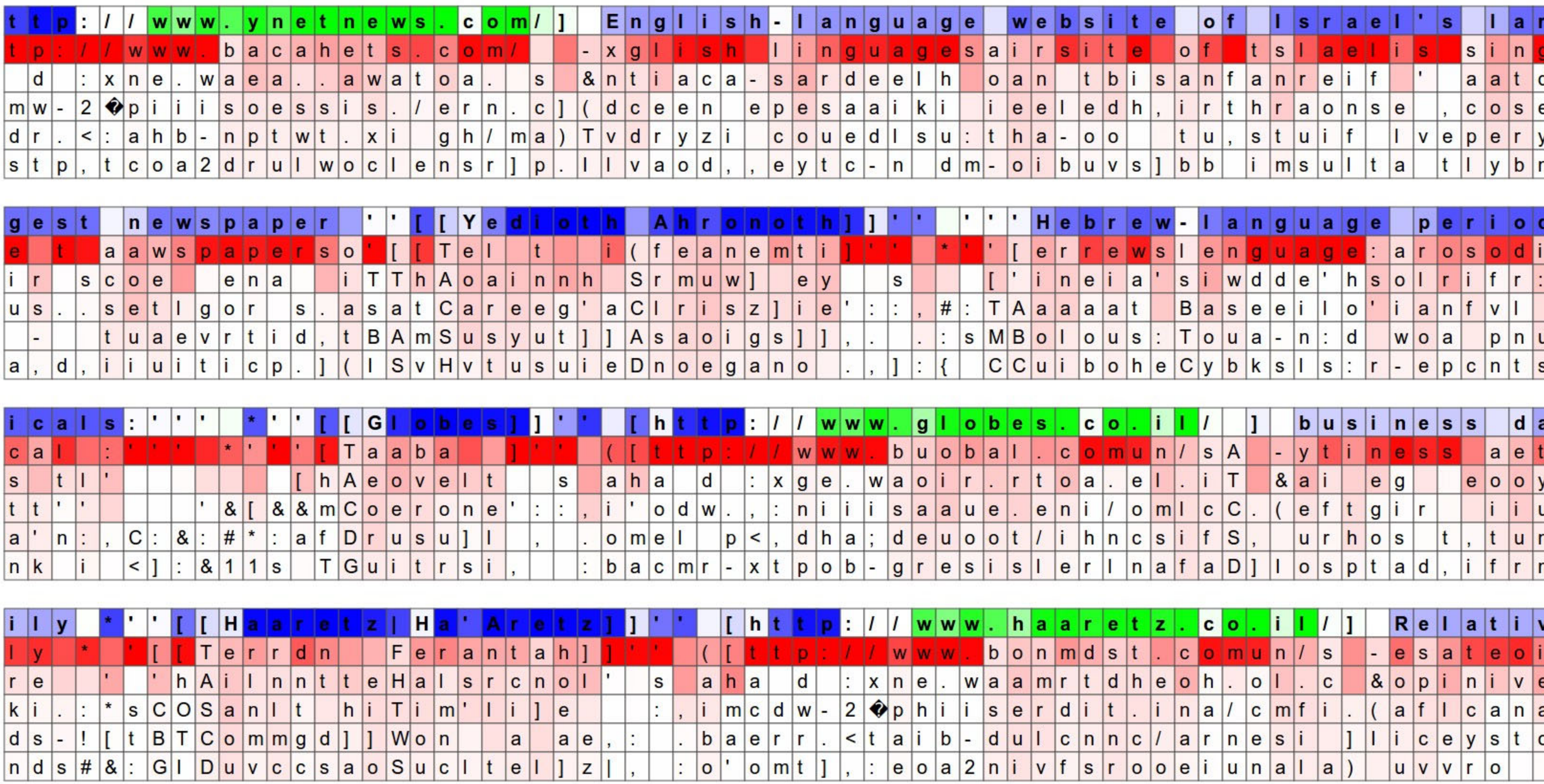
```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)     (func)

#define SWAP_ALLOCATE(nr)     (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0));   \
  if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
          pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
  PUT_PARAM_RAID(2, sel) = get_state_state();
  set_pid_sum((unsigned long)state, current_state_str(),
           (unsigned long)-1->lr_full; low;
}
```


Ref: http://cs231n.stanford.edu/

جستجو به‌دنبال سلول‌های تفسیرپذیر

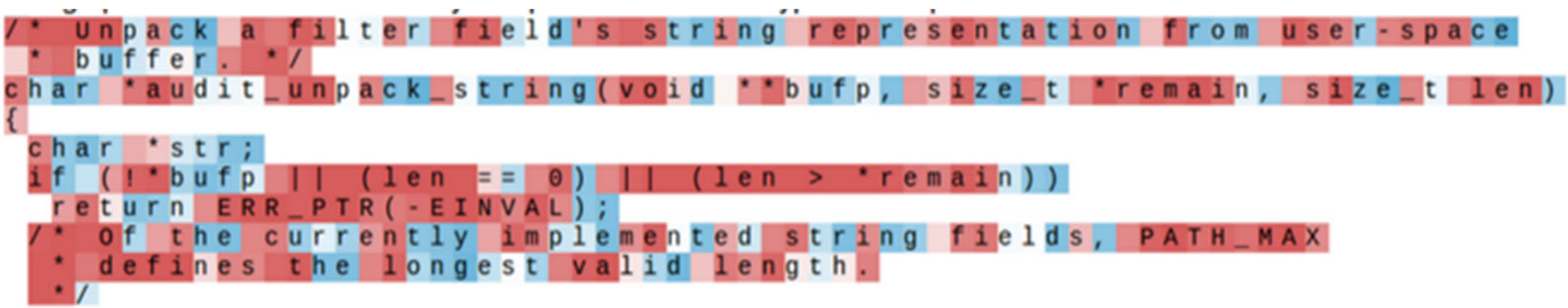
Searching for Interpretable Cells




ttp://www.ynetnews.com/] English-language website of Israel's lar
gest newspaper '' [[Yedioth Ahronoth]] '' ''' Hebrew-language period
icals:''' *'' [[Globes]] '' [http://www.globes.co.il/] business da
ily *''[[Haaretz|Ha'Aretz]]'' [http://www.haaretz.co.il/] Relativ

جستجو به‌دنبال سلول‌های تفسیرپذیر

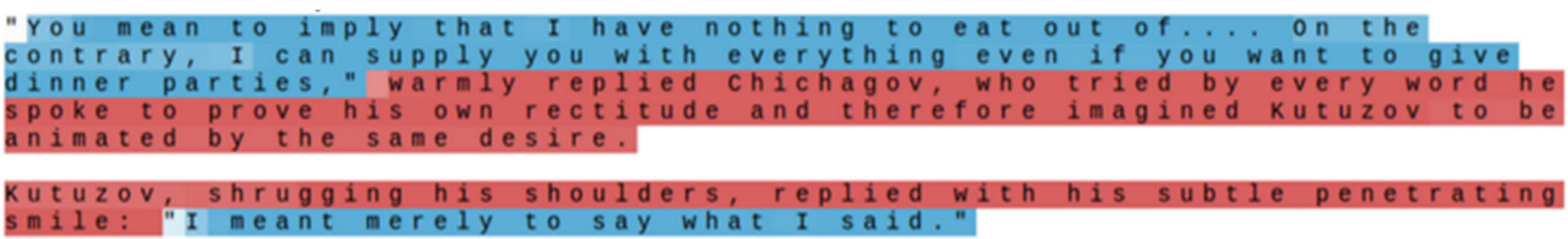
Searching for Interpretable Cells



Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

جستجو به‌دنبال سلول‌های تفسیرپذیر

SEARCHING FOR INTERPRETABLE CELLS



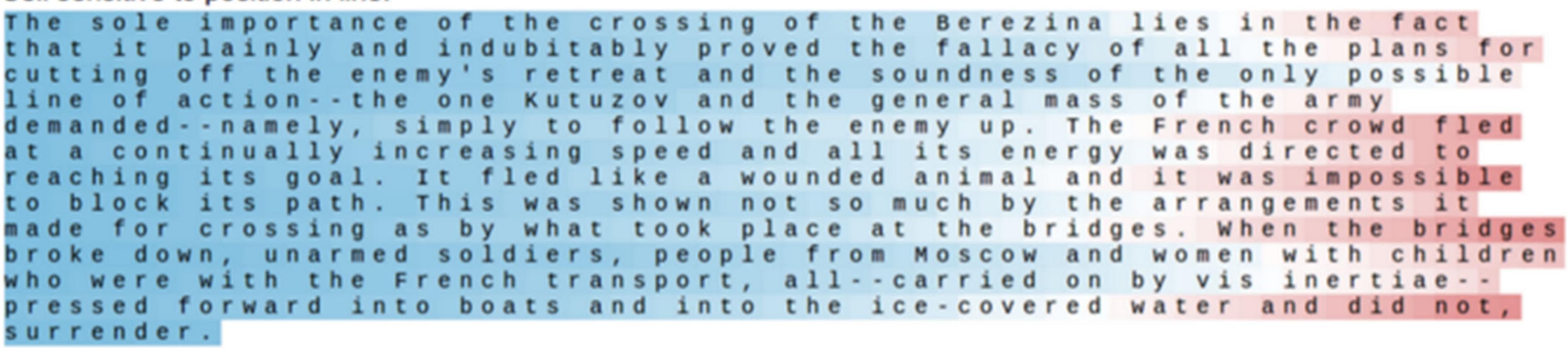
quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

جستجو به‌دنبال سلول‌های تفسیرپذیر

Searching for Interpretable Cells

Cell sensitive to position in line:

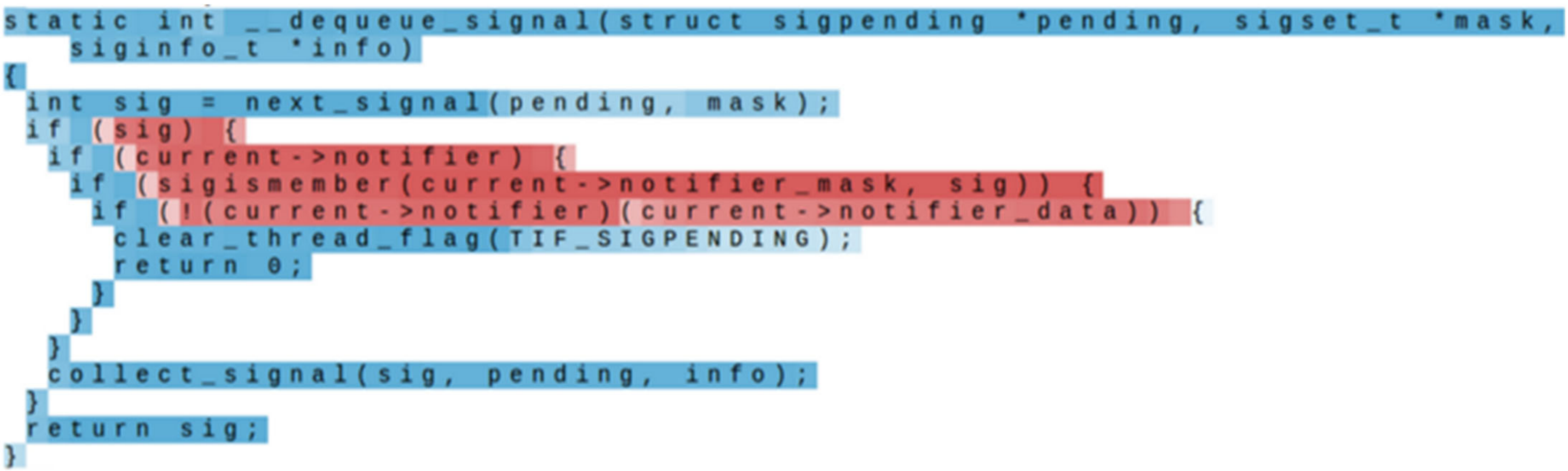


line length tracking cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

جستجو به‌دنبال سلول‌های تفسیرپذیر

Searching for Interpretable Cells



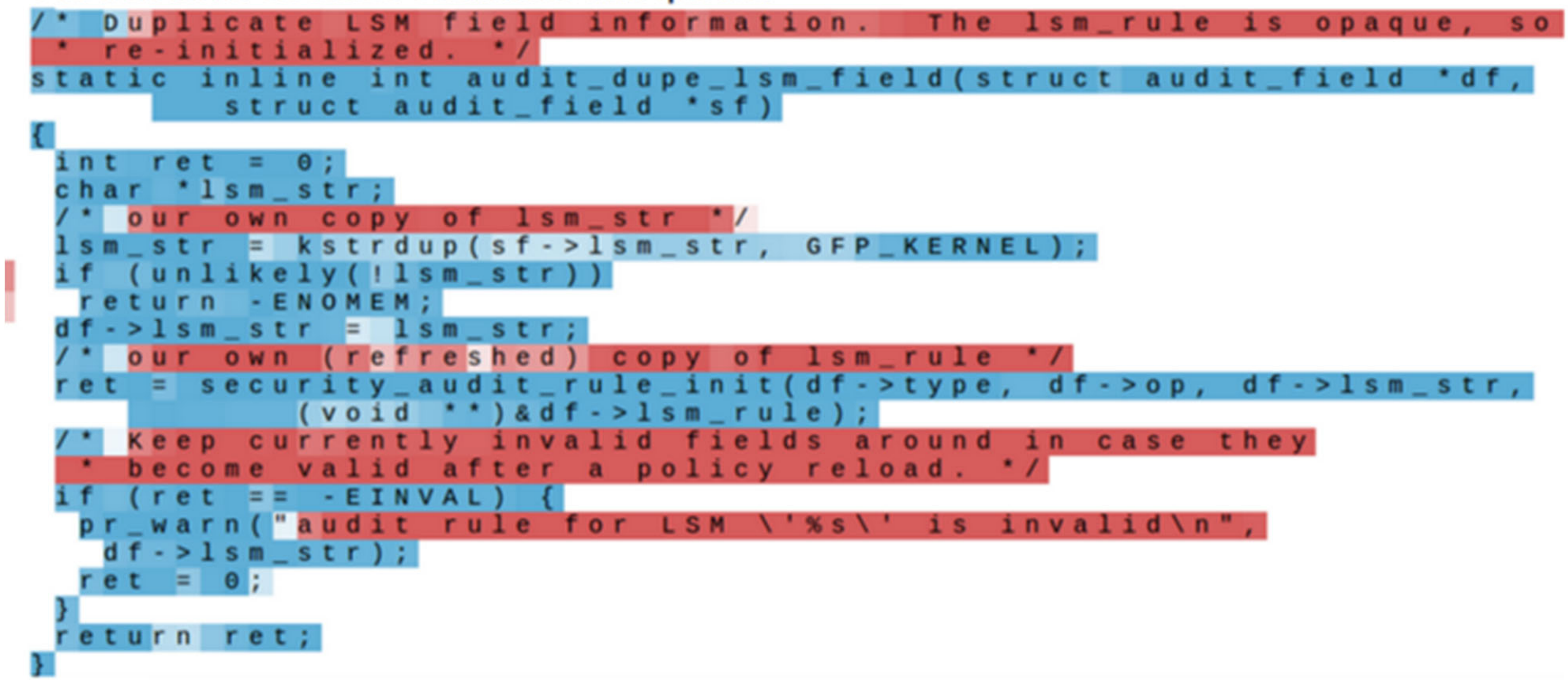
if statement cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

جستجو به‌دنبال سلول‌های تفسیرپذیر

Searching for Interpretable Cells

Cell that turns on inside comments and quotes:



quote/comment cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Ref: http://cs231n.stanford.edu/

جستجو به‌دنبال سلول‌های تفسیرپذیر

SEARCHING FOR INTERPRETABLE CELLS

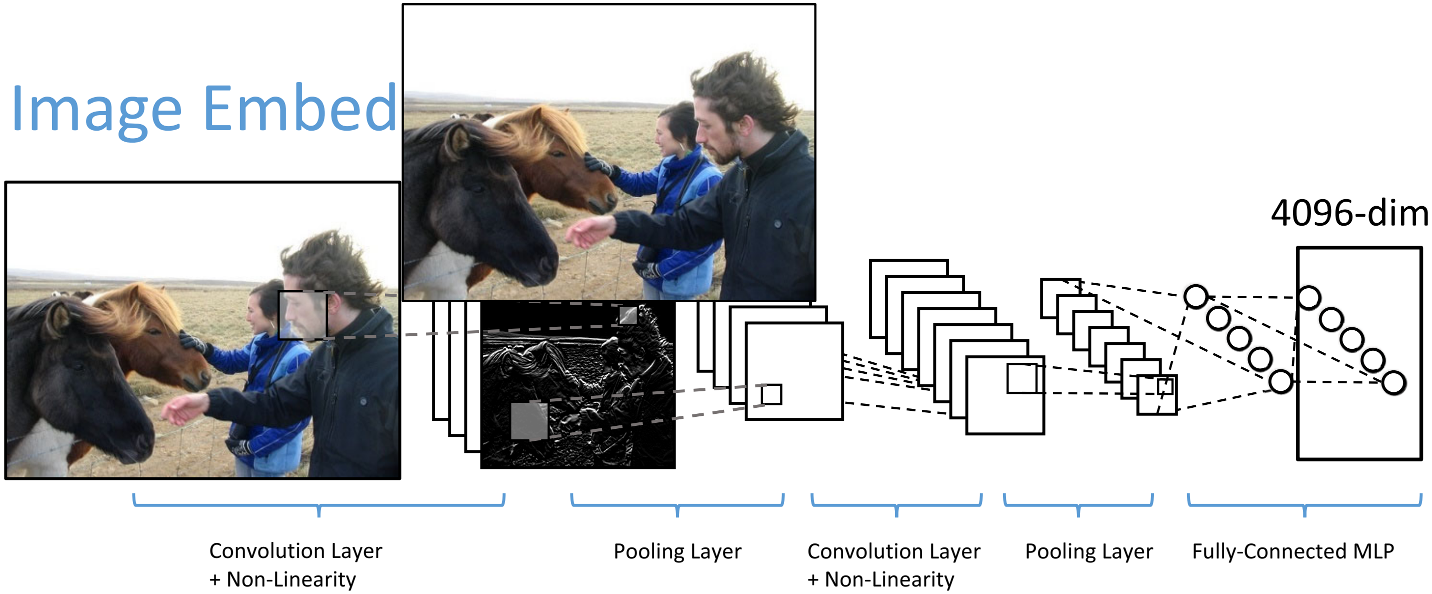
```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

code depth cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

عنوان‌گذاری عصبی تصاویر

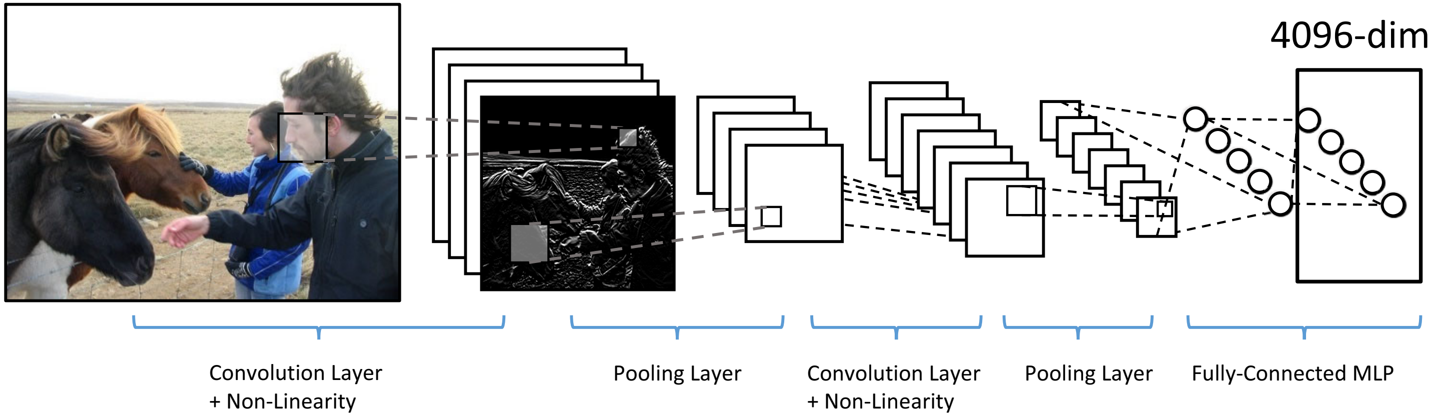
Neural Image Captioning



Ref: http://cs231n.stanford.edu/

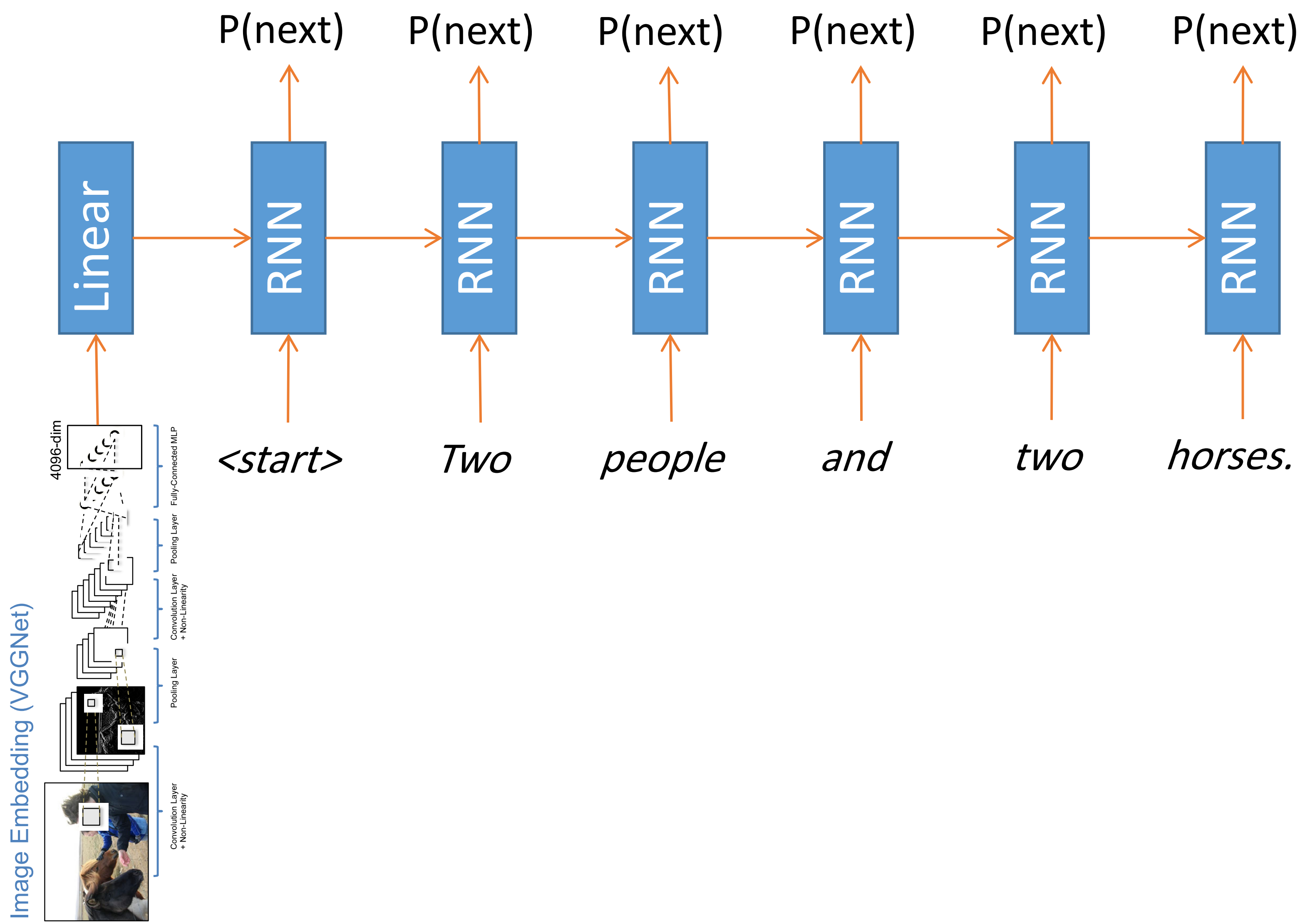
عنوان‌گذاری عصبی تصاویر

Image Embedding (VGGNet)



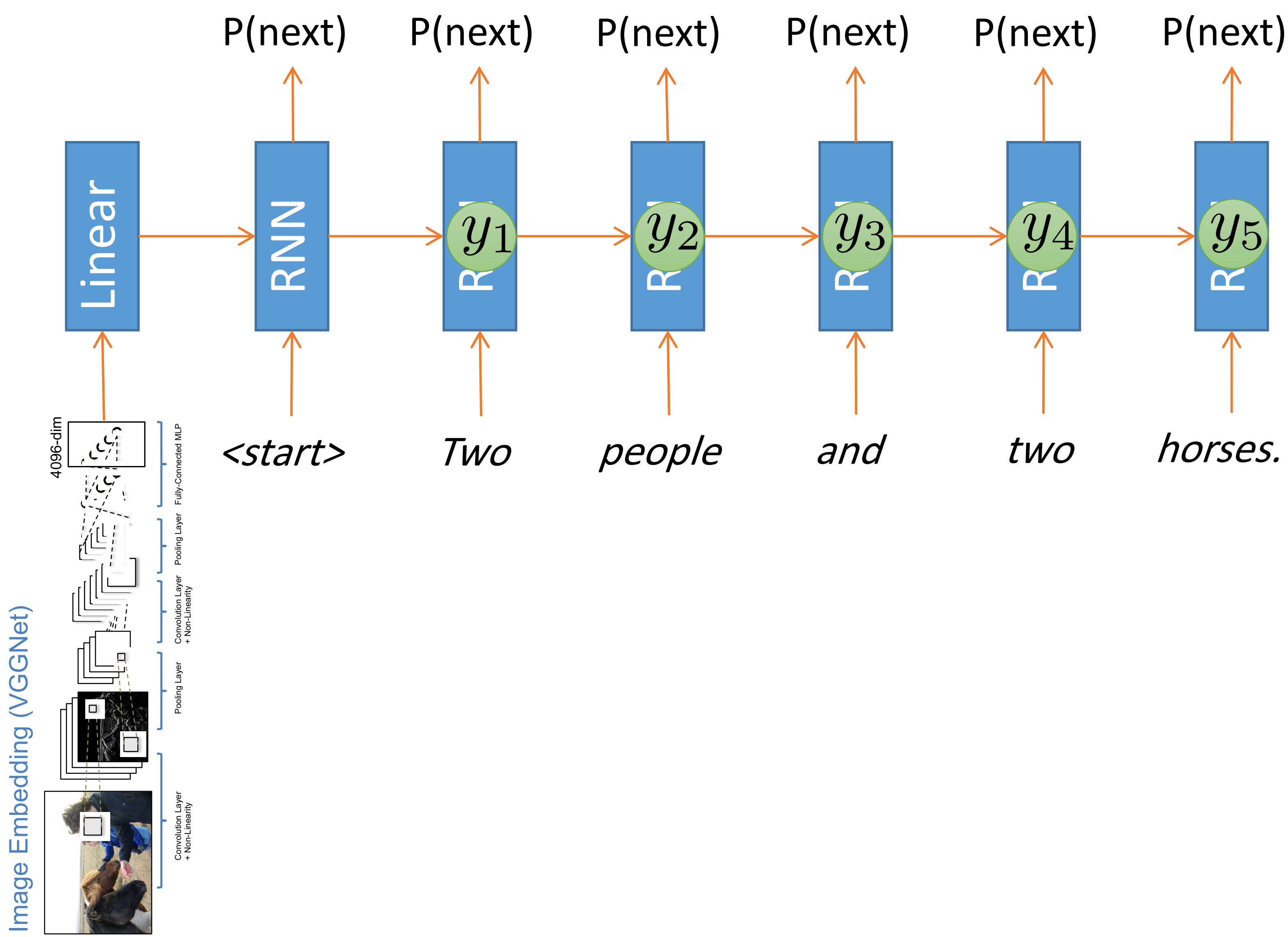
Ref: http://cs231n.stanford.edu/

عنوان‌گذاری عصبی تصاویر



Ref: http://cs231n.stanford.edu/

عنوان‌گذاری عصبی تصاویر

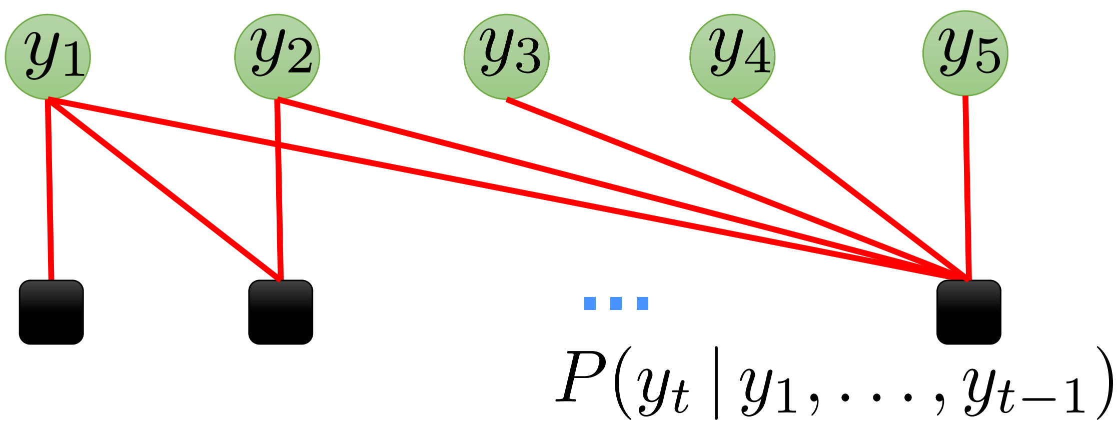


Ref: http://cs231n.stanford.edu/

عنوان‌گذاری عصبی تصاویر

گراف مدل فاکتور دنباله

SEQUENCE MODEL FACTOR GRAPH



http://dbs.cloudcv.org/captioning&mode=interactive

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر

IMAGE CAPTIONING

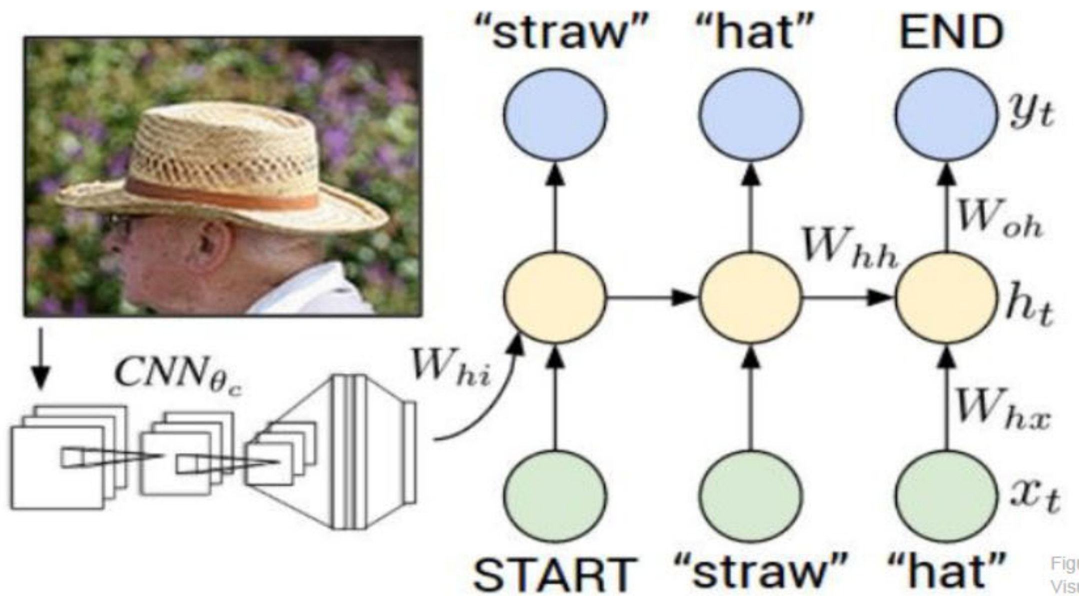
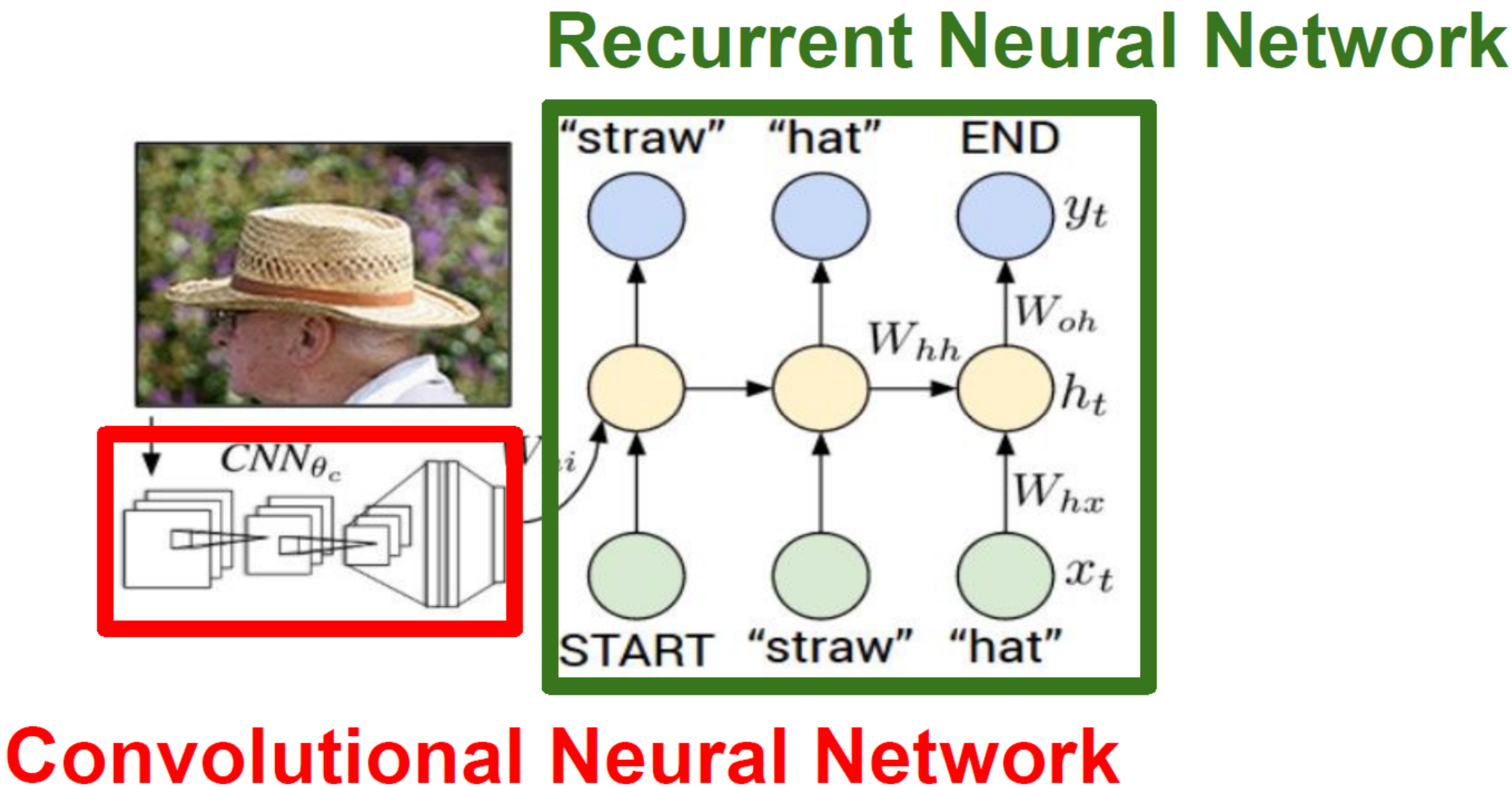


Figure from Karpathy et a, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015. Reproduced for educational purposes.

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

عنوان‌گذاری روی تصاویر

IMAGE CAPTIONING



عنوان‌گذاری روی تصاویر

test image

This image is CC0 public domain

عنوان‌گذاری روی تصاویر

test image

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر

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

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر

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

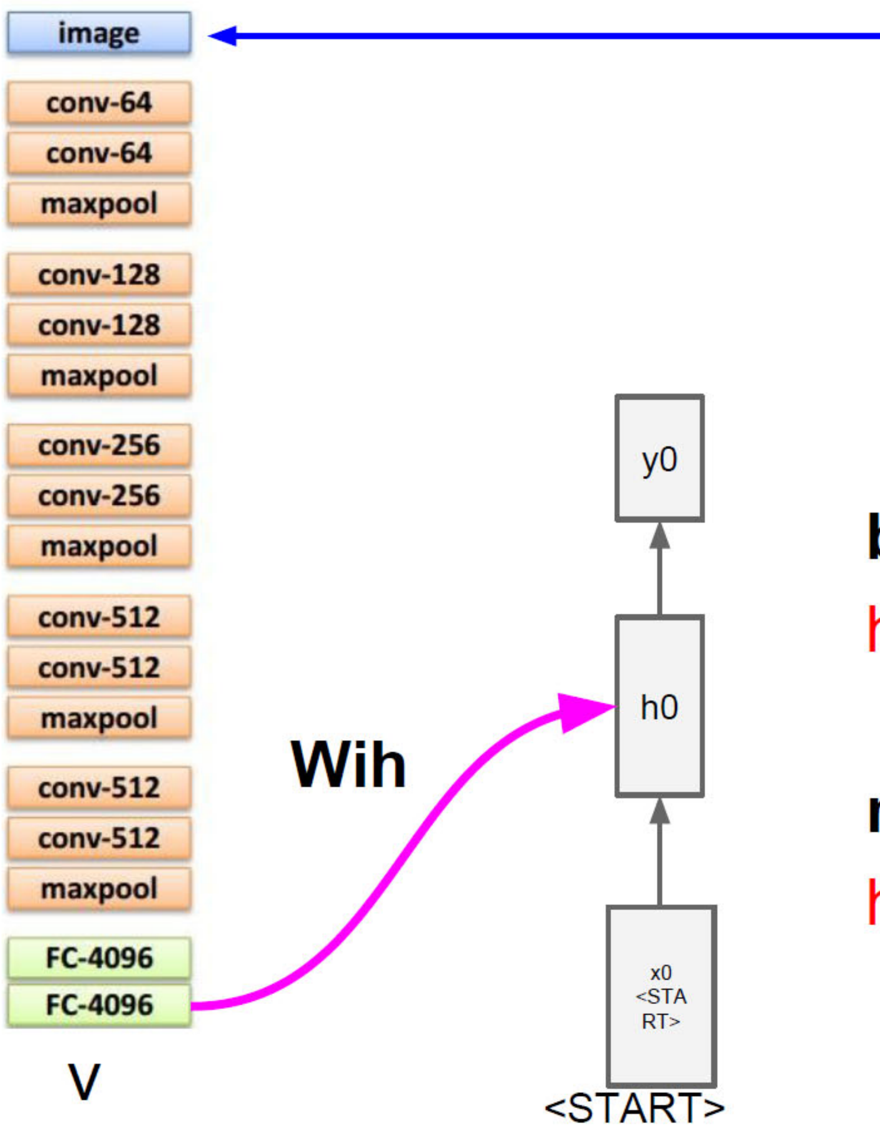
test image

x0
<START>

<START>

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر



test image

before:

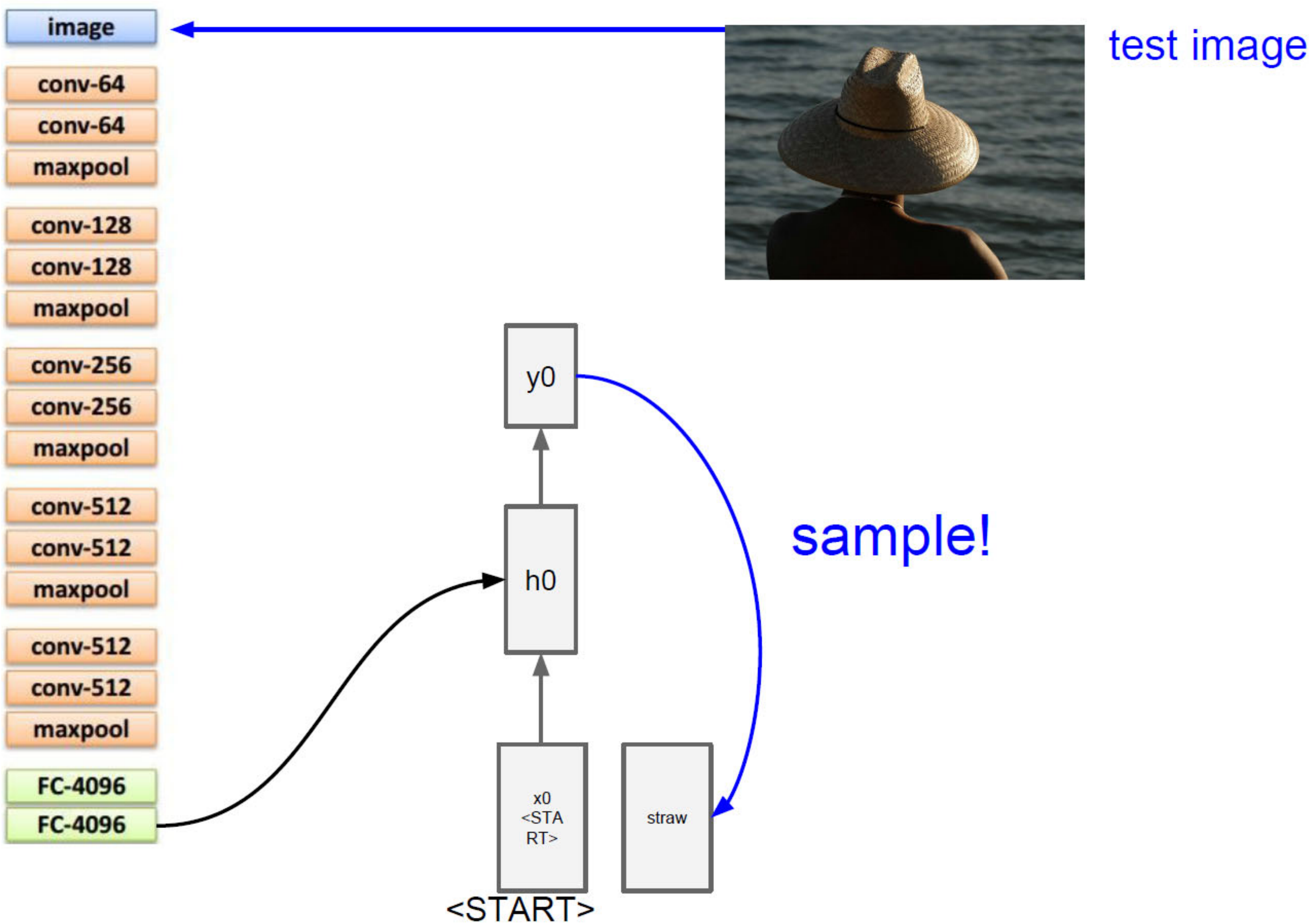
h = tanh(Wxh * x + Whh * h)

now:

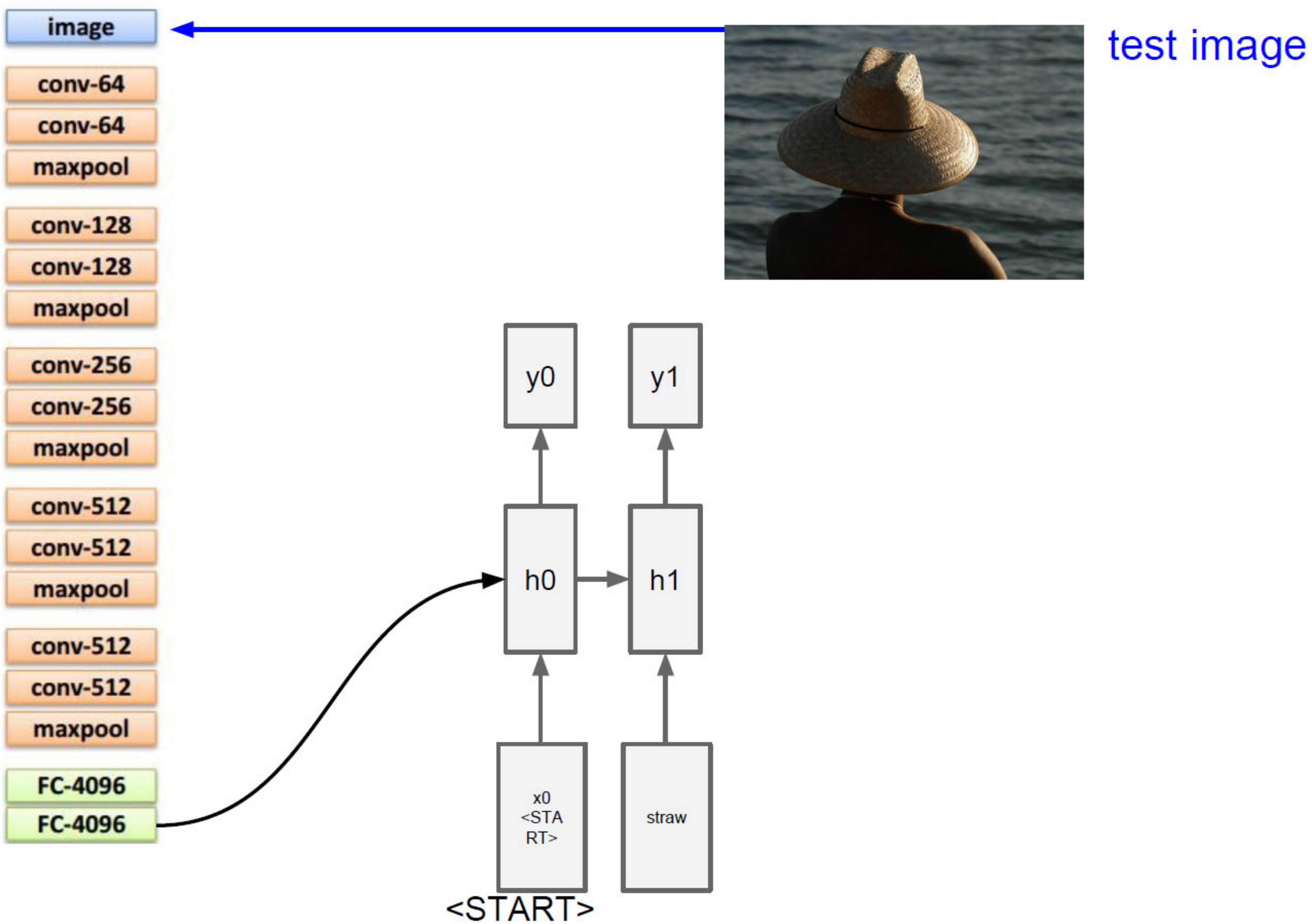
h = tanh(Wxh * x + Whh * h + **Wih * v**)

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر

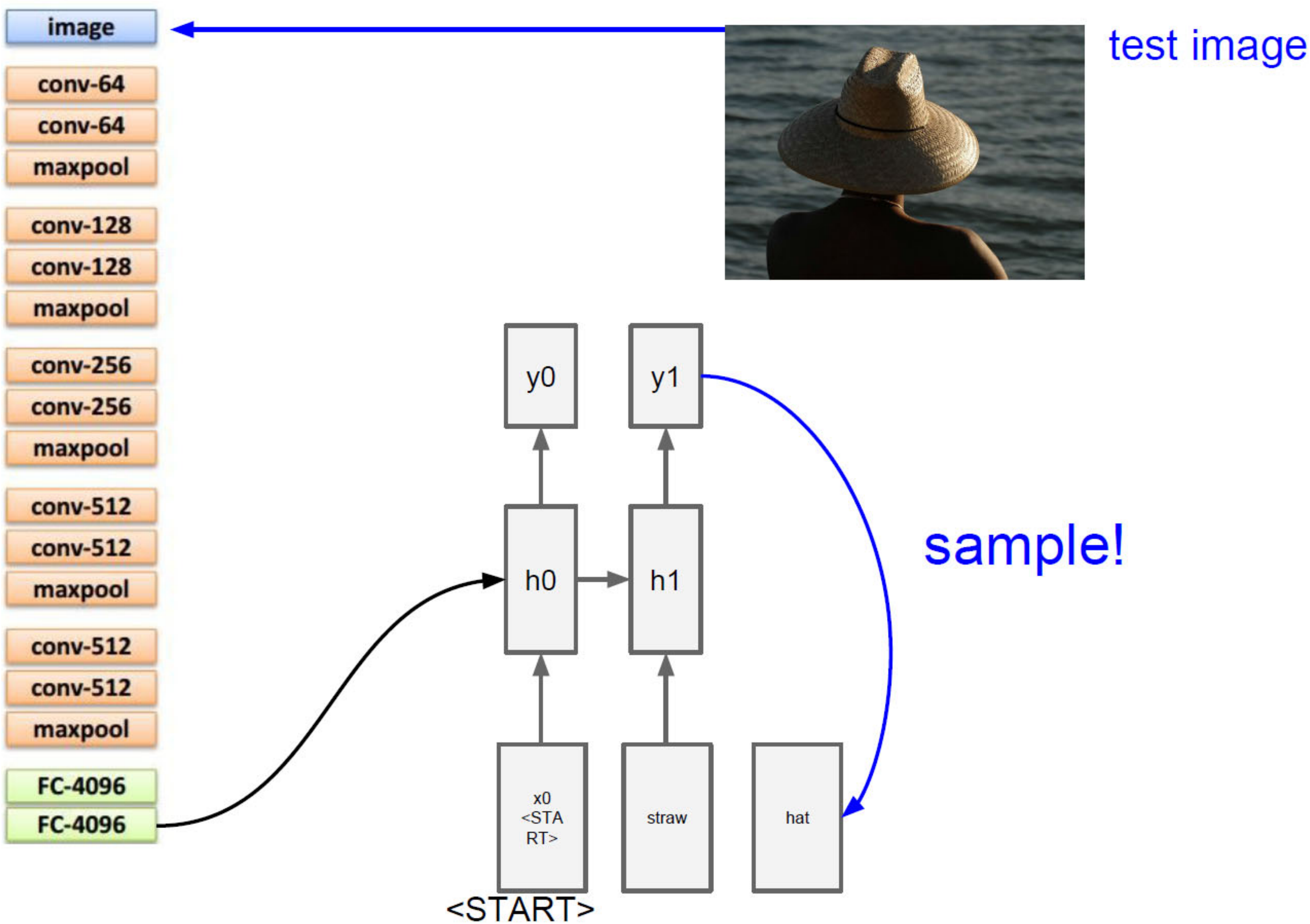


عنوان‌گذاری روی تصاویر

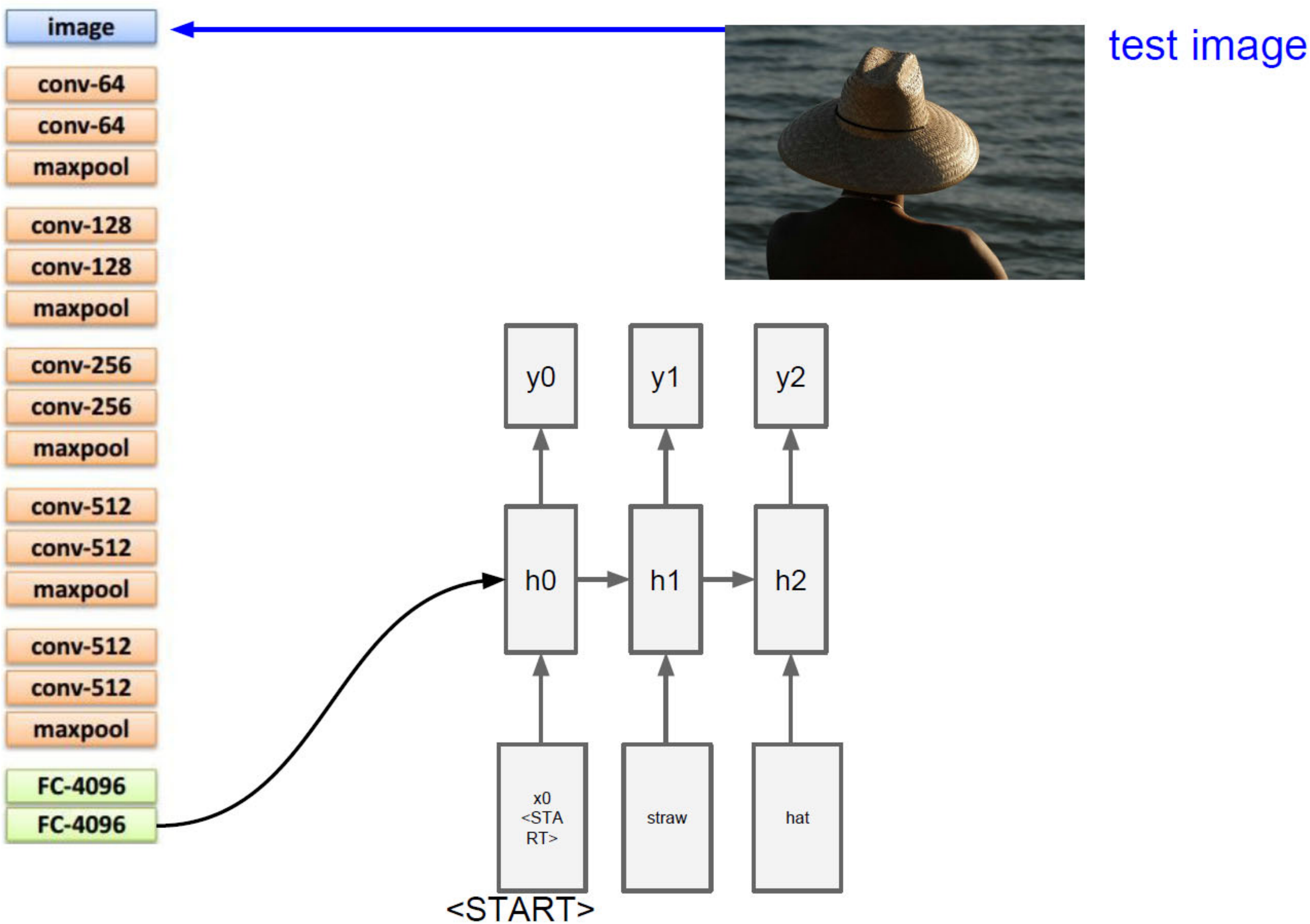


Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر

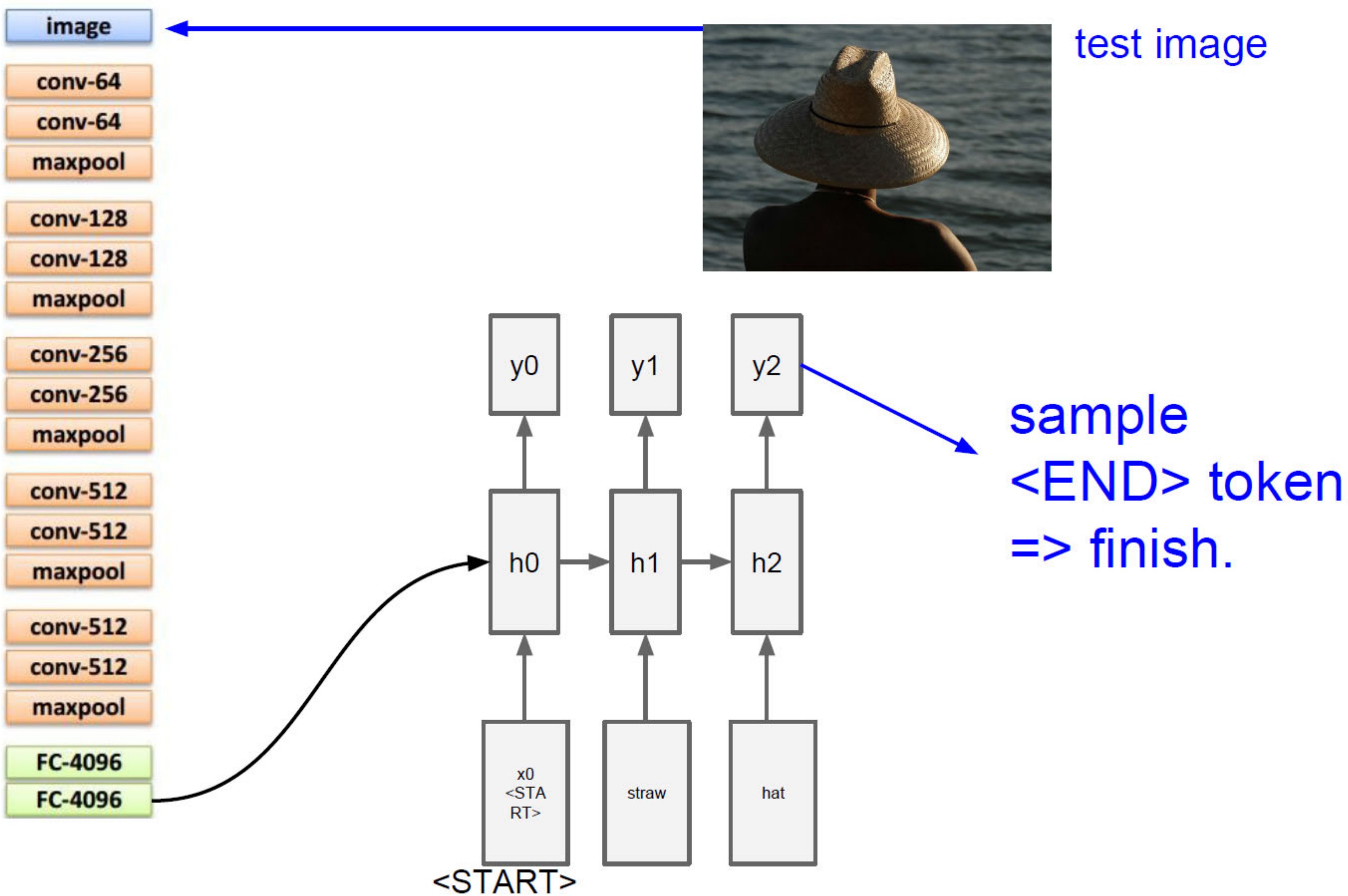


عنوان‌گذاری روی تصاویر



Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر



Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر

نتایج نمونه

Image Captioning: Example Results

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

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

عنوان‌گذاری روی تصاویر

نمونه‌های از موارد شکست

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 bird is perched on a tree branch

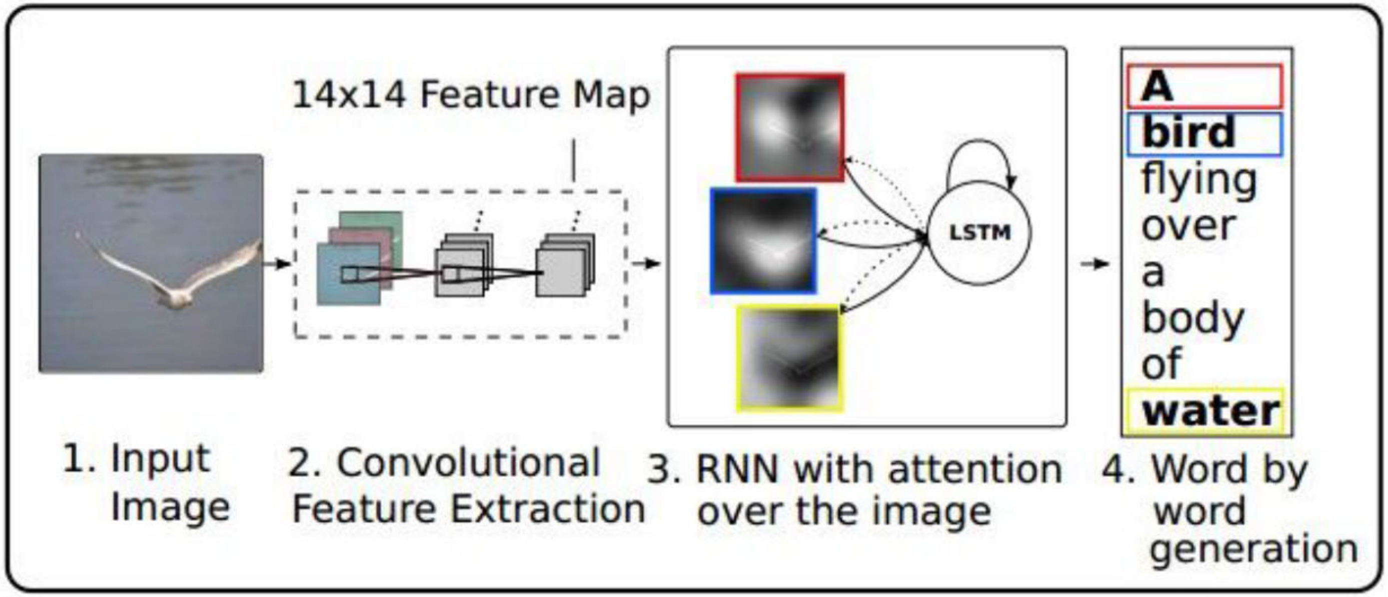
A man in a baseball uniform throwing a ball

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر با استفاده از توجه

Image Captioning with Attention

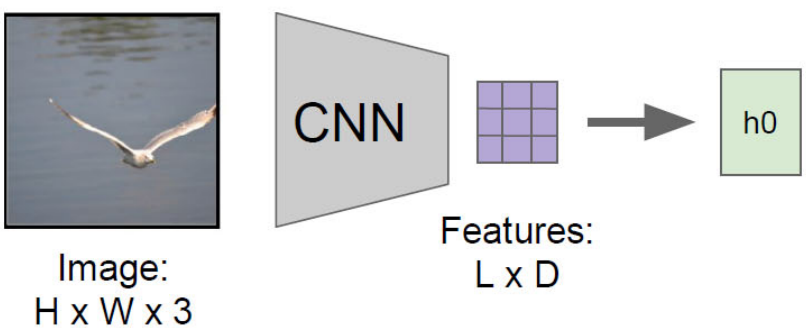
RNN focuses its attention at a different spatial location when generating each word



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

عنوان‌گذاری روی تصاویر با استفاده از توجه

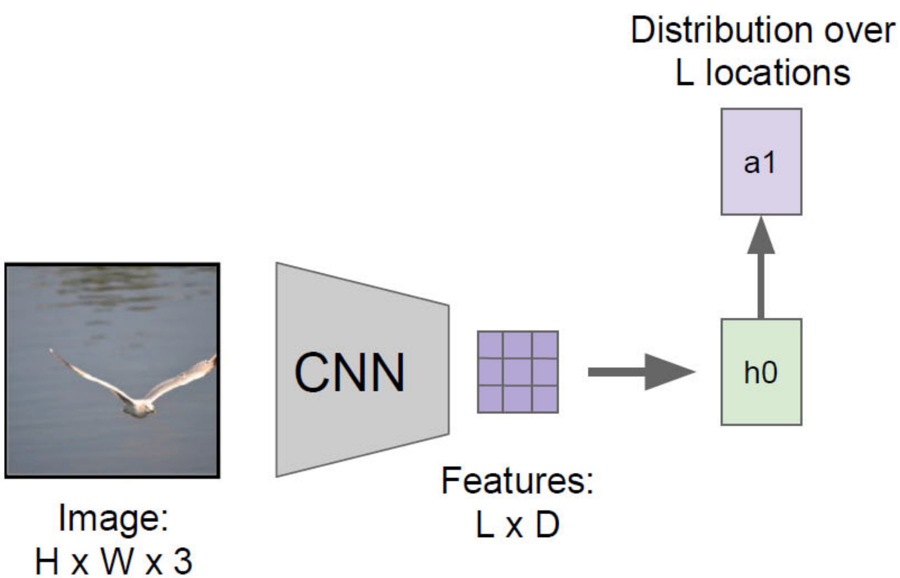
Image Captioning with Attention



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

عنوان‌گذاری روی تصاویر با استفاده از توجه

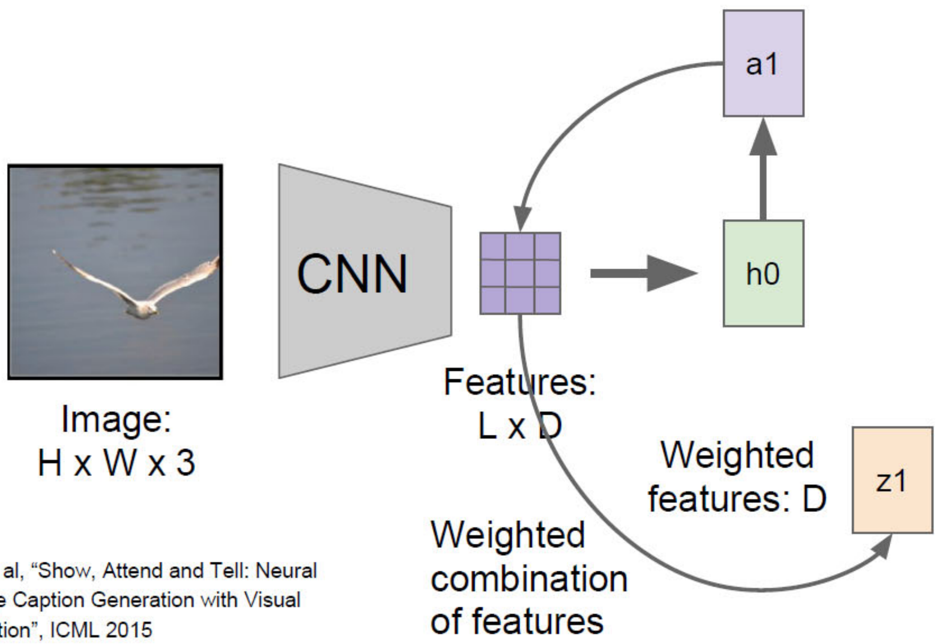
Image Captioning with Attention



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

عنوان‌گذاری روی تصاویر با استفاده از توجه

Image Captioning with Attention



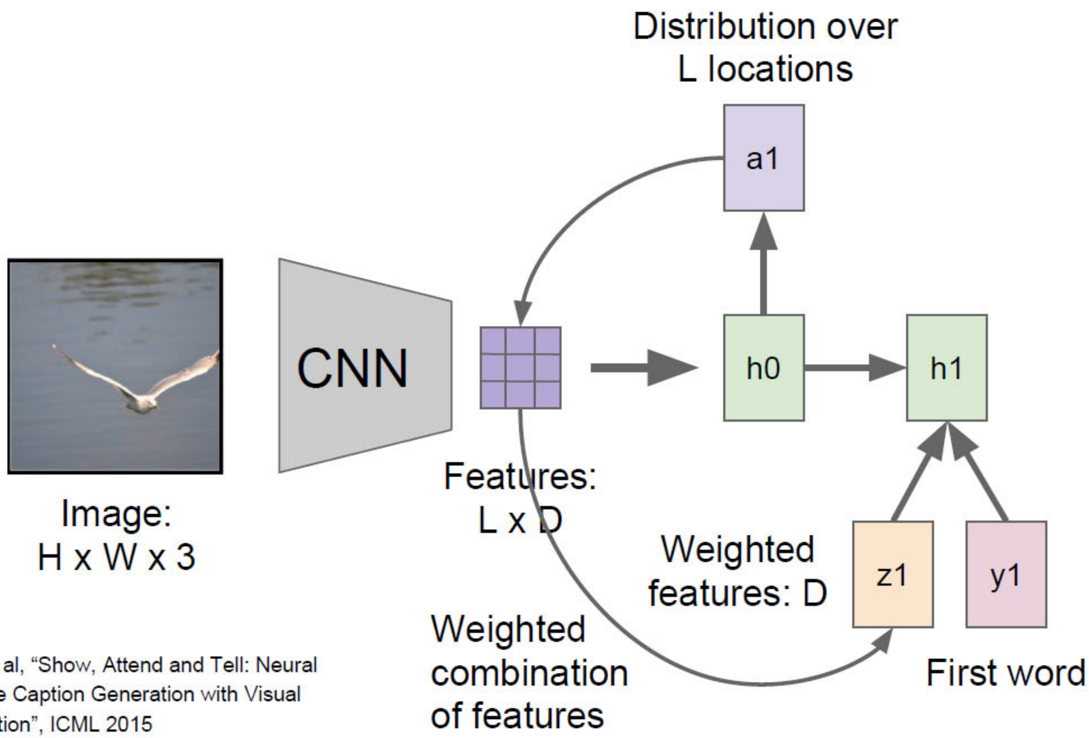
$$z = \sum_{i=1}^{L} p_i v_i$$

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

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر با استفاده از توجه

Image Captioning with Attention

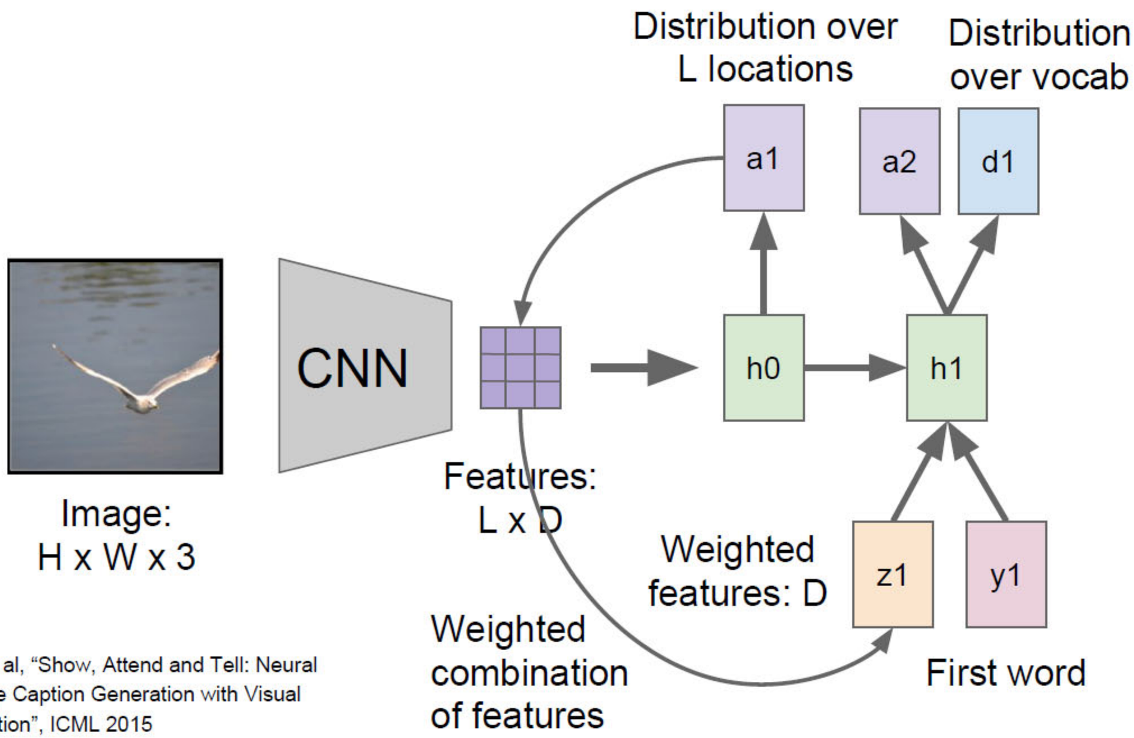


Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Ref: http://cs231n.stanford.edu/

عنوان‌گذاری روی تصاویر با استفاده از توجه

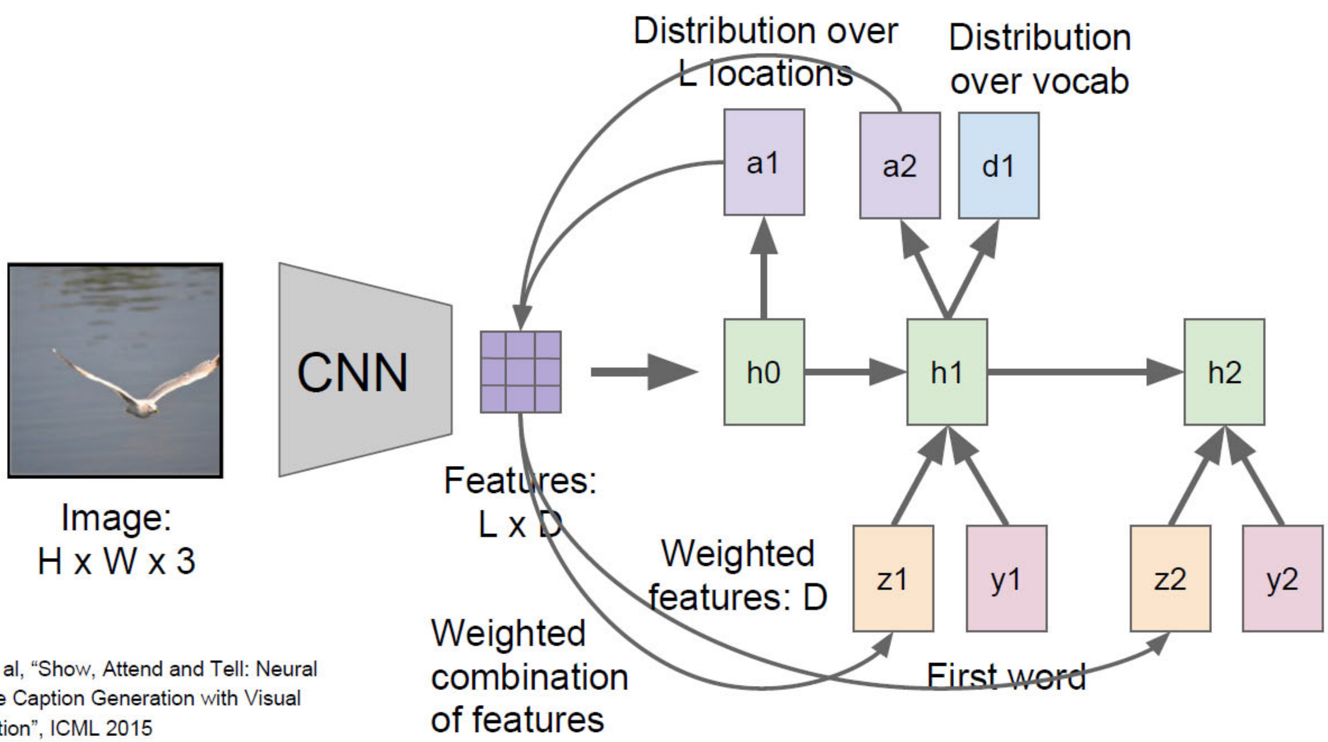
Image Captioning with Attention



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

عنوان‌گذاری روی تصاویر با استفاده از توجه

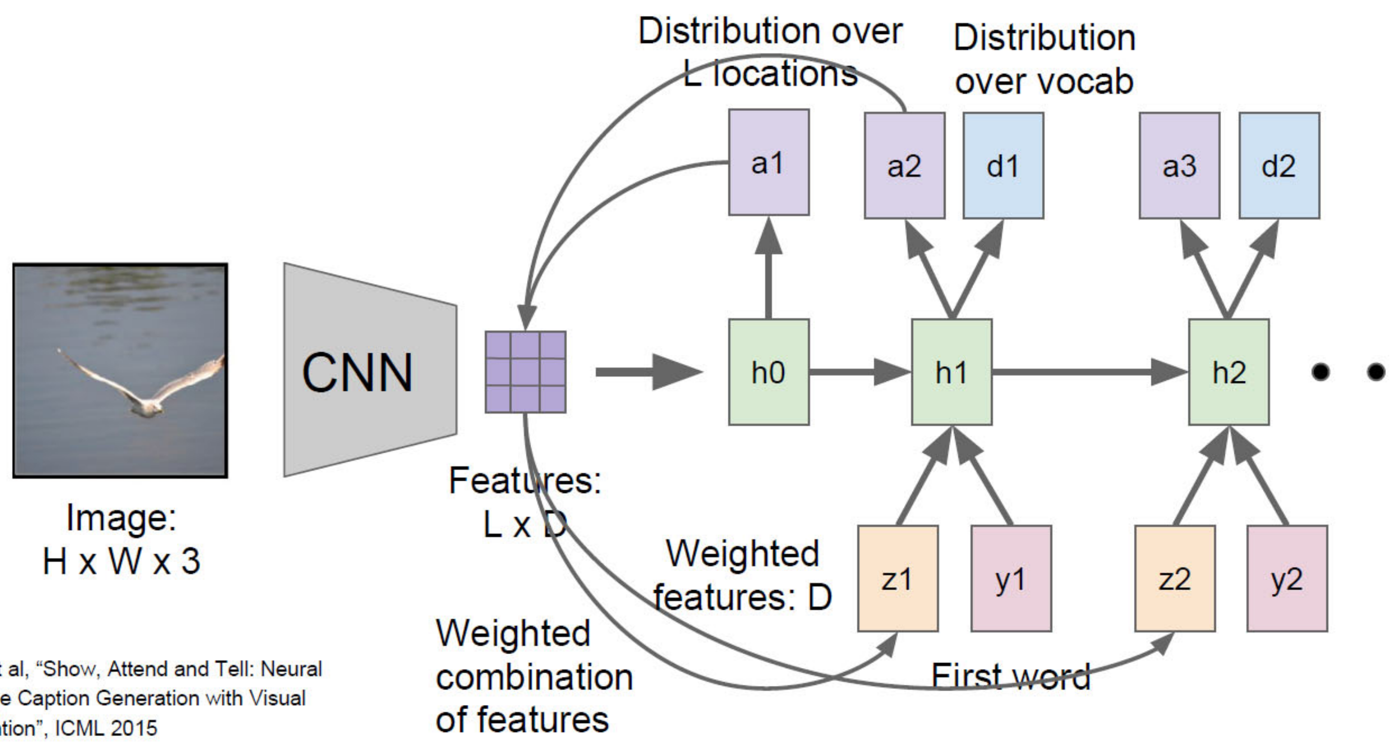
Image Captioning with Attention



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

عنوان‌گذاری روی تصاویر با استفاده از توجه

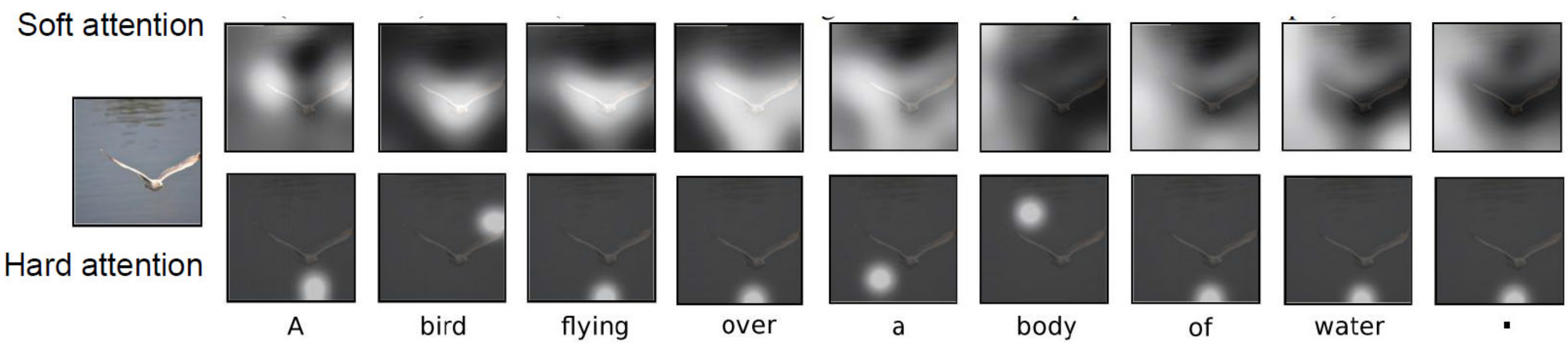
Image Captioning with Attention



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

عنوان‌گذاری روی تصاویر با استفاده از توجه

Image Captioning with Attention

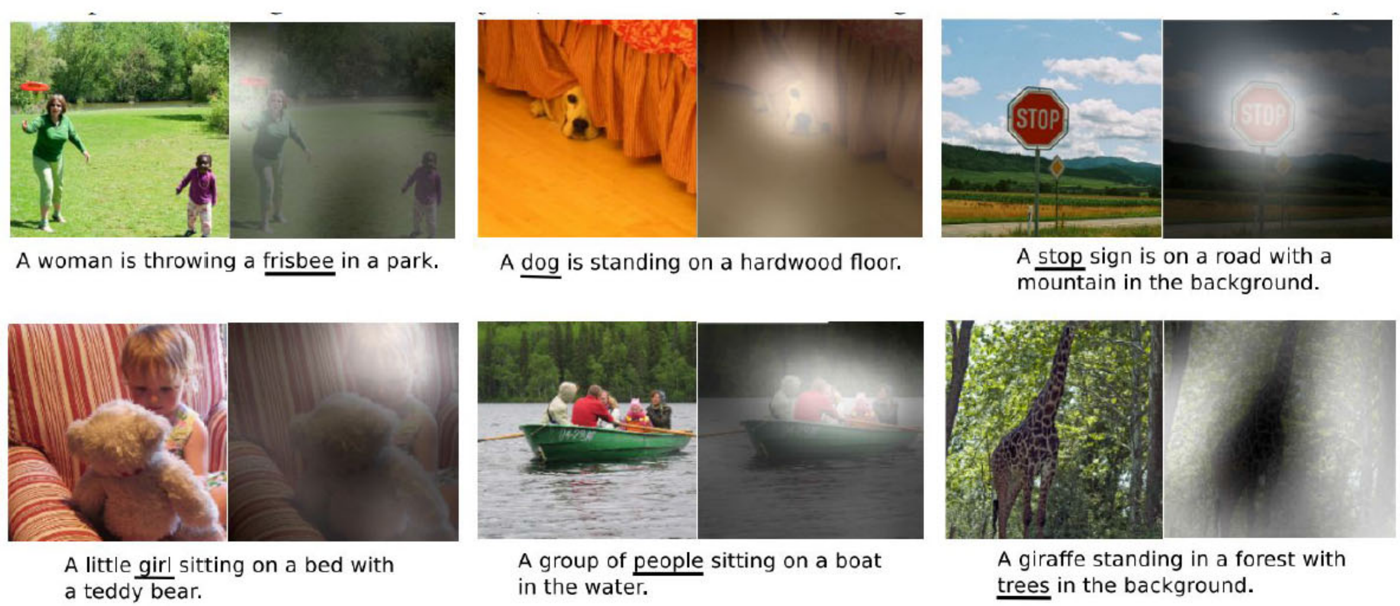


Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

عنوان‌گذاری روی تصاویر با استفاده از توجه

نمونه‌هایی از نتایج

Image Captioning with Attention



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

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

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

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

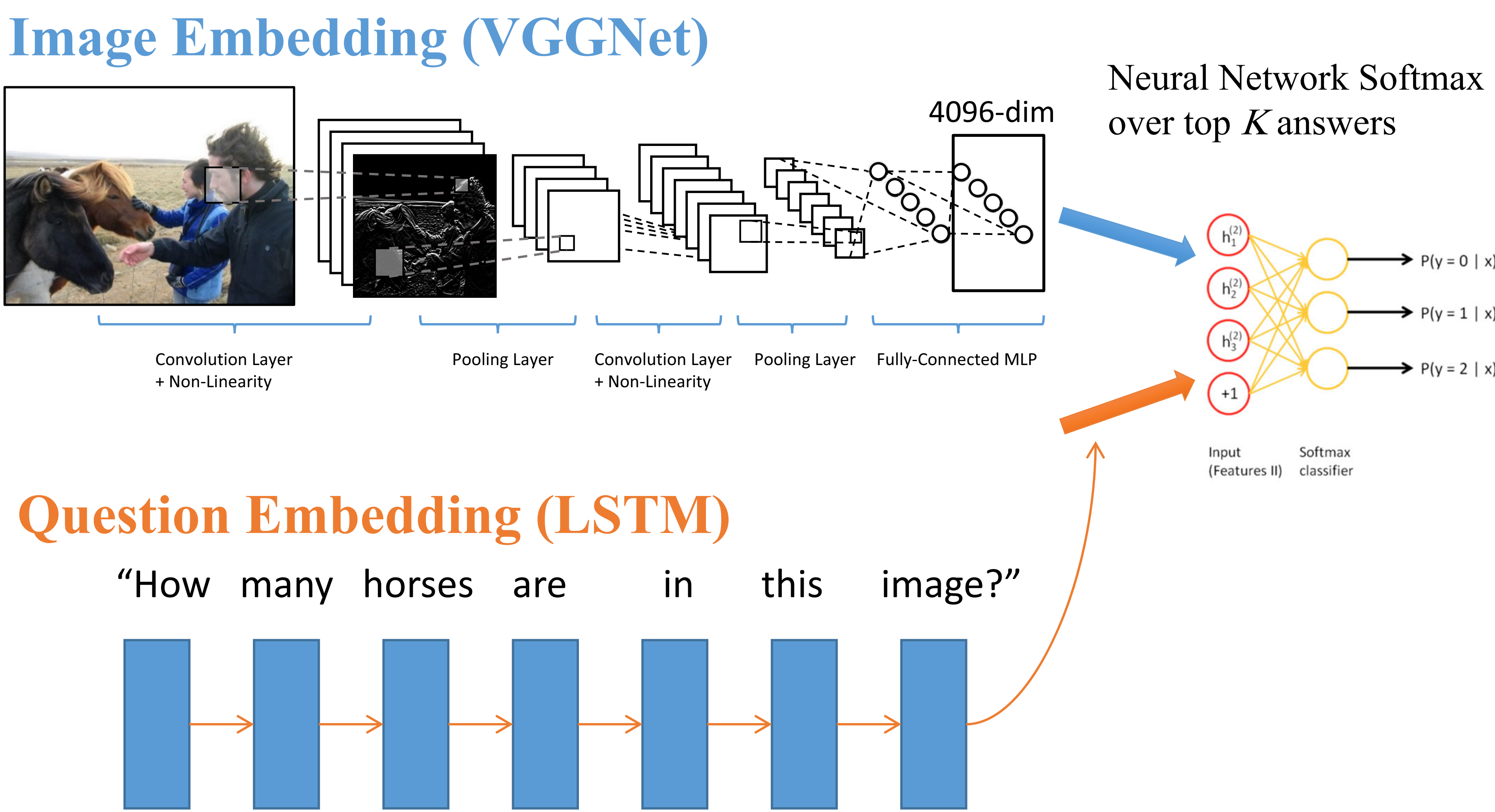
A giraffe standing in a forest with trees in the background.

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

Ref: http://cs231n.stanford.edu/

مدل‌های نوعی پاسخ به پرسش دیداری

Typical VQA Models



Ref: http://cs231n.stanford.edu/

پاسخ به پرسش دیداری

Visual Question Answering

Q: What endangered animal is featured on the truck?

A: **A bald eagle.**
A: A sparrow.
A: A humming bird.
A: A raven.

Q: Where will the driver go if turning right?

A: **Onto 24 ¾ Rd.**
A: Onto 25 ¾ Rd.
A: Onto 23 ¾ Rd.
A: Onto Main Street.

Q: Who is under the umbrella?

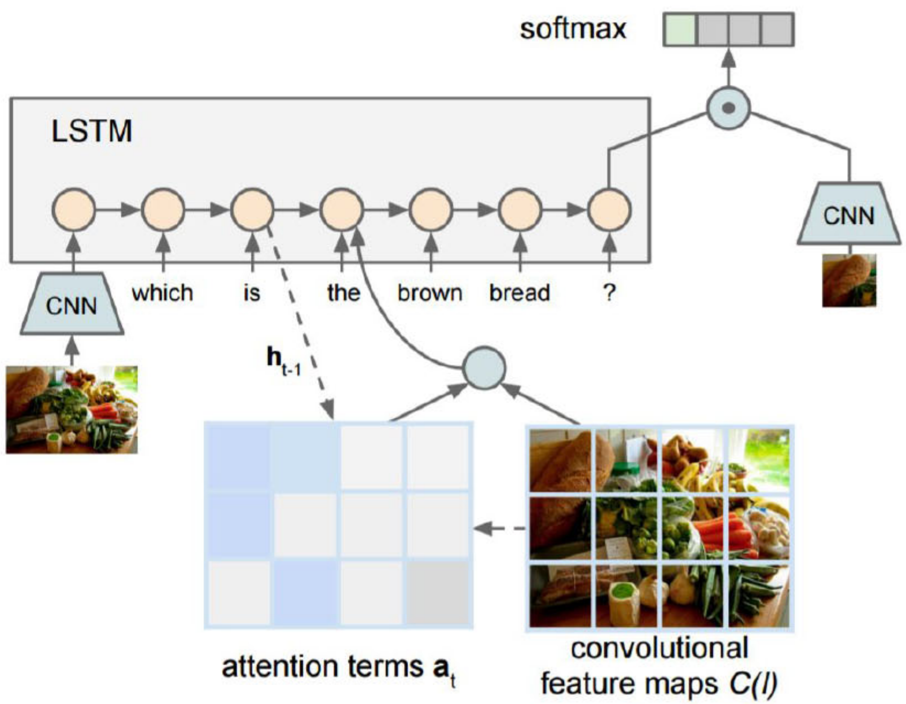
A: **Two women.**
A: A child.
A: An old man.
A: A husband and a wife.

wal et al, "VQA: Visual Question Answering", ICCV 2015
et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016
from Zhu et al, copyright IEEE 2016. Reproduced for educational purposes.

پاسخ به پرسش دیداری

استفاده از شبکه‌های عصبی بازگشتی به‌همراه توجه

Visual Question Answering: RNNs with Attention



What kind of animal is in the photo?
A **cat**.

Why is the person holding a knife?
To cut the **cake** with.

Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016

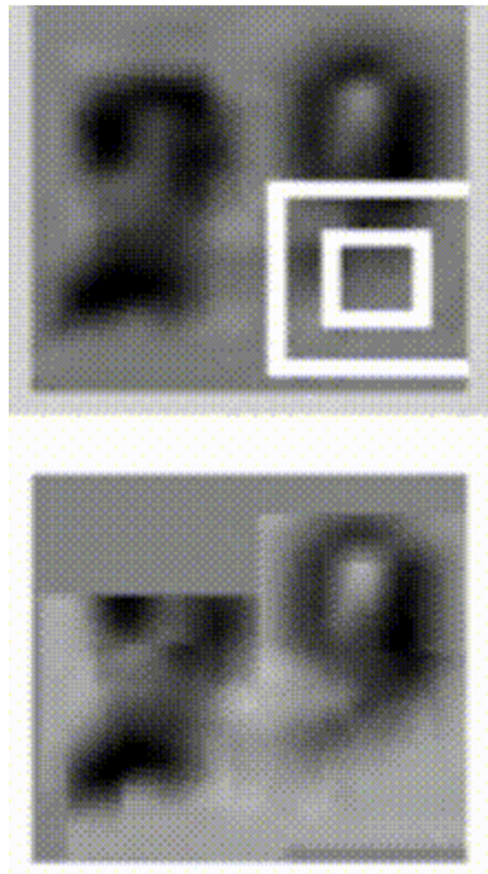
پردازش دنباله‌ای داده‌های غیر دنباله‌ای

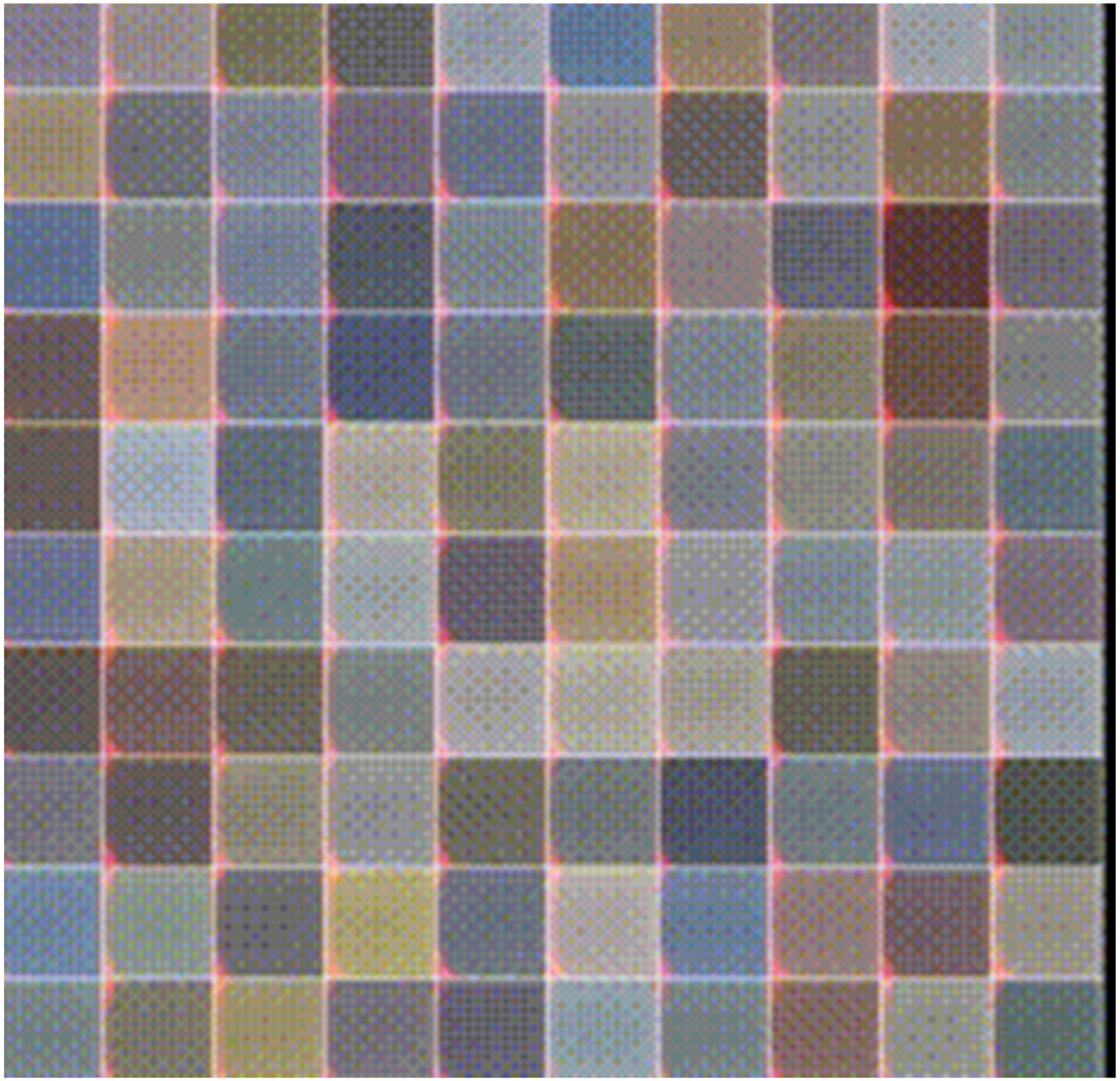
مثال

SEQUENTIAL PROCESSING OF NON-SEQUENCE DATA

Classify images by taking a series of “glimpses”

طبقه‌بندی تصاویر با دریافت یک سری از گلیمپس‌ها





۳

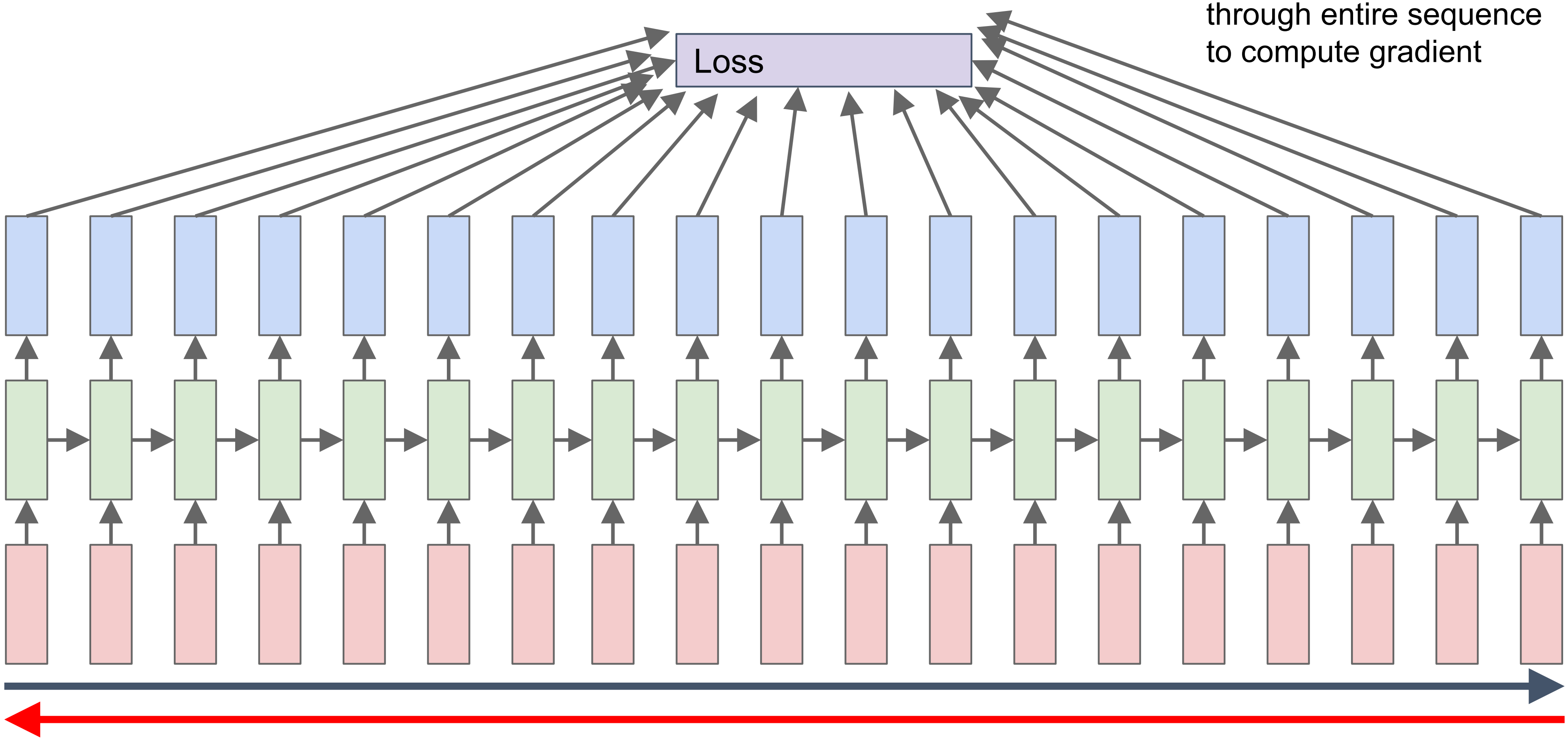
آموزش شبکه‌های عصبی بازگشتی

آموزش شبکه‌های عصبی بازگشتی

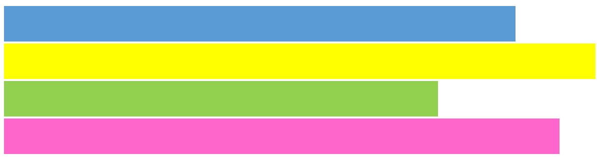
پس‌انتشار در امتداد زمان

Backpropagation Through Time (BPTT)

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



چون دنباله‌های قرار گرفته در یک دسته (batch) ممکن است طول‌های متفاوتی داشته باشند، پردازش دسته‌ای می‌تواند ناکارآمد شود.

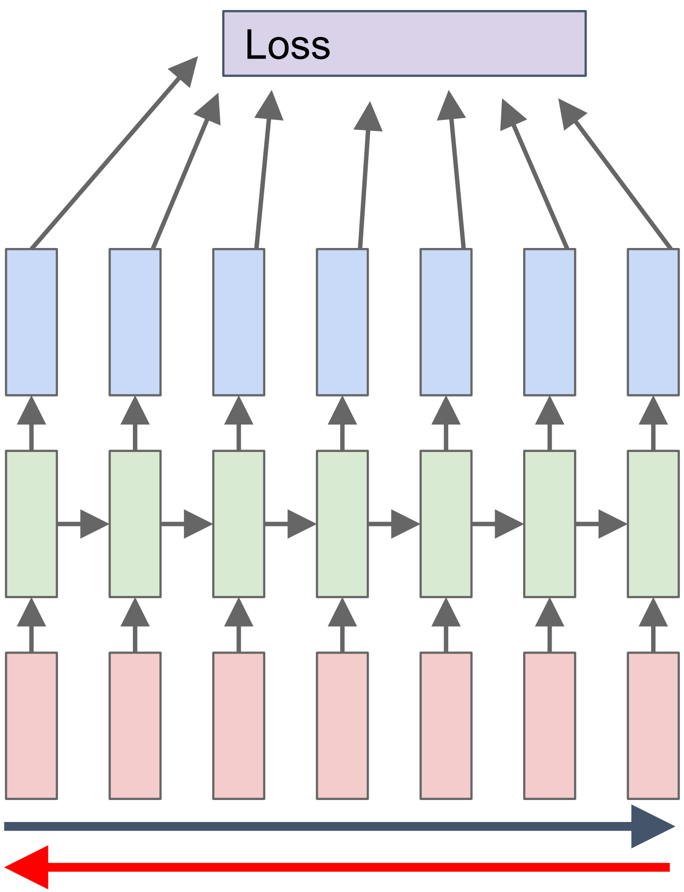


Ref: http://cs231n.stanford.edu/

آموزش شبکه‌های عصبی بازگشتی

پس‌انتشار برش‌یافته در امتداد زمان

Truncated Backpropagation Through Time (TBPTT)



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

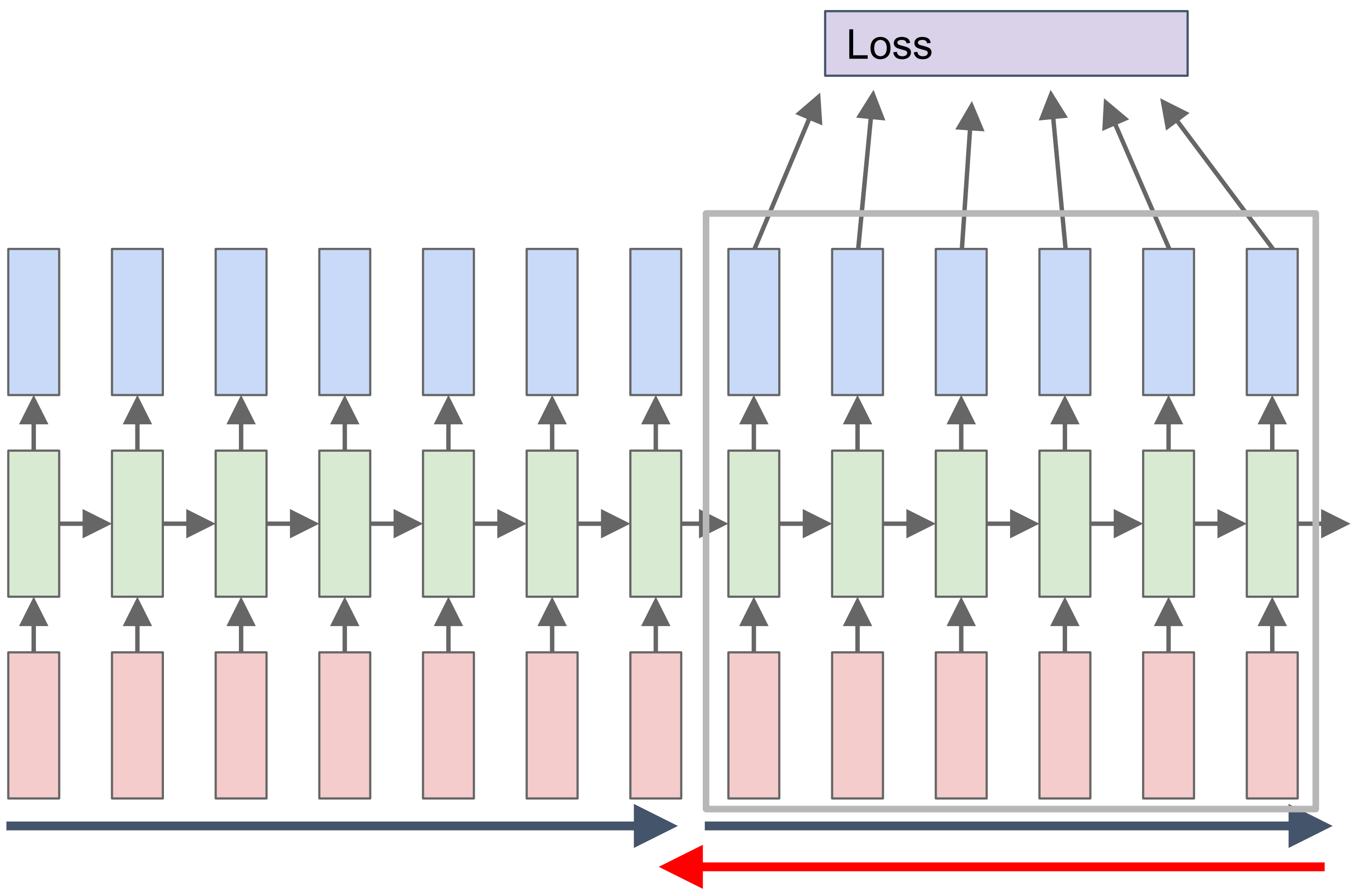
چون دنباله‌های قرار گرفته در یک دسته (batch) طول‌های یکسانی دارند، پردازش دسته‌ای کارآمد می‌شود.

Ref: http://cs231n.stanford.edu/

آموزش شبکه‌های عصبی بازگشتی

پس‌انتشار برش‌یافته در امتداد زمان

TRUNCATED BACKPROPAGATION THROUGH TIME (TBPTT)

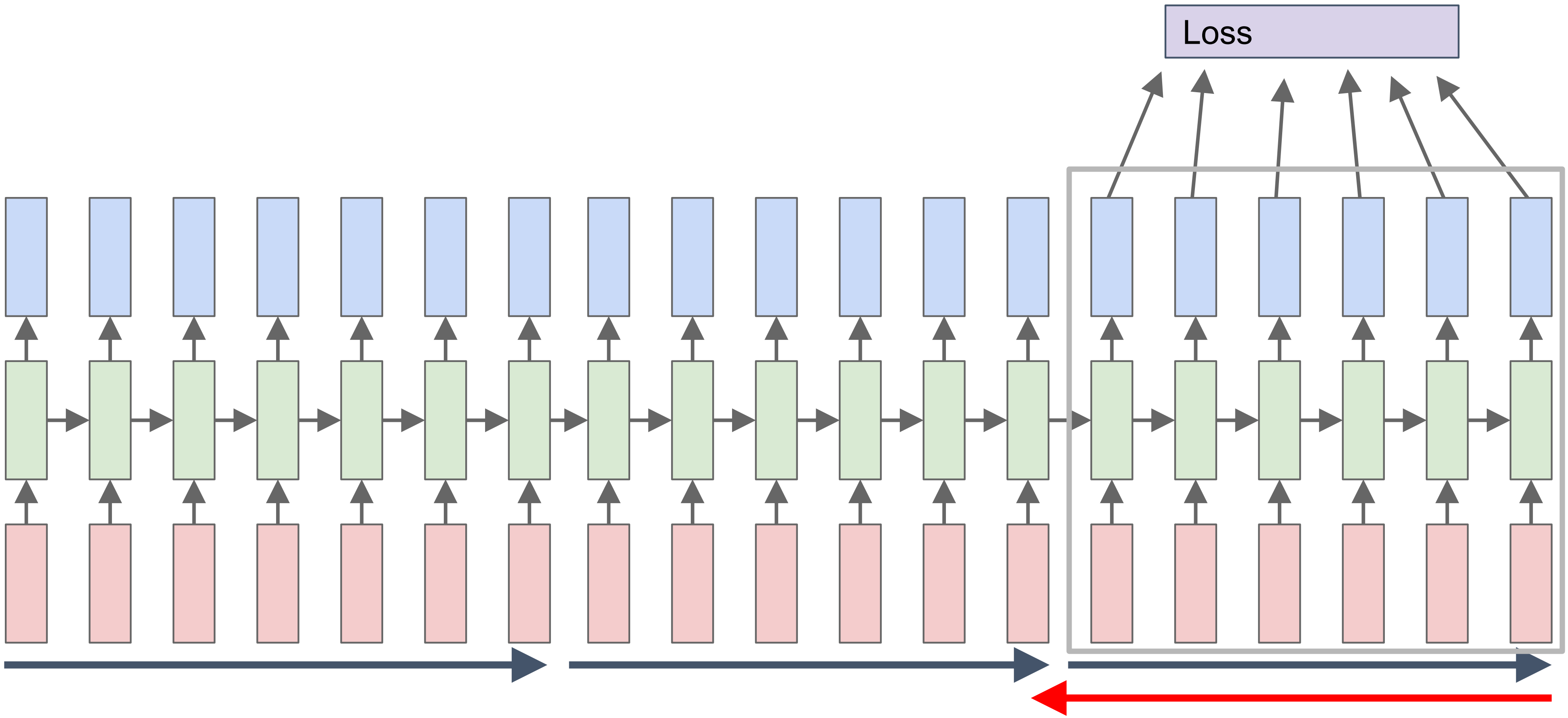


Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

آموزش شبکه‌های عصبی بازگشتی

پس‌انتشار برش‌یافته در امتداد زمان

TRUNCATED BACKPROPAGATION THROUGH TIME (TBPTT)



آموزش شبکه‌های عصبی بازگشتی

TRAINING RECURRENT NEURAL NETWORKS

از تابع اتلاف آنتروپی متقابل استفاده می‌کنیم:
Cross-Entropy Loss

$$P = \prod_{t,k} y_{tk}^{l_{tk}} \quad \Rightarrow \quad \mathcal{L} = -\log P = \sum_t \mathcal{L}_t = -\frac{1}{T} \sum_t l_t \log y_t$$

الگوریتم مورد استفاده: **پس‌انتشار در طول زمان** (Backpropagation Through Time: BPTT)

- مجدداً از قاعده‌ی زنجیره‌ای استفاده می‌شود.
- تنها تفاوت: گرادیان ها بر روی گام‌های زمانی باقی می‌مانند.

Ref: https://uvadlc.github.io/

آموزش شبکه‌های عصبی بازگشتی

پس‌انتشار در طول زمان: مثال (۱ از ۴)

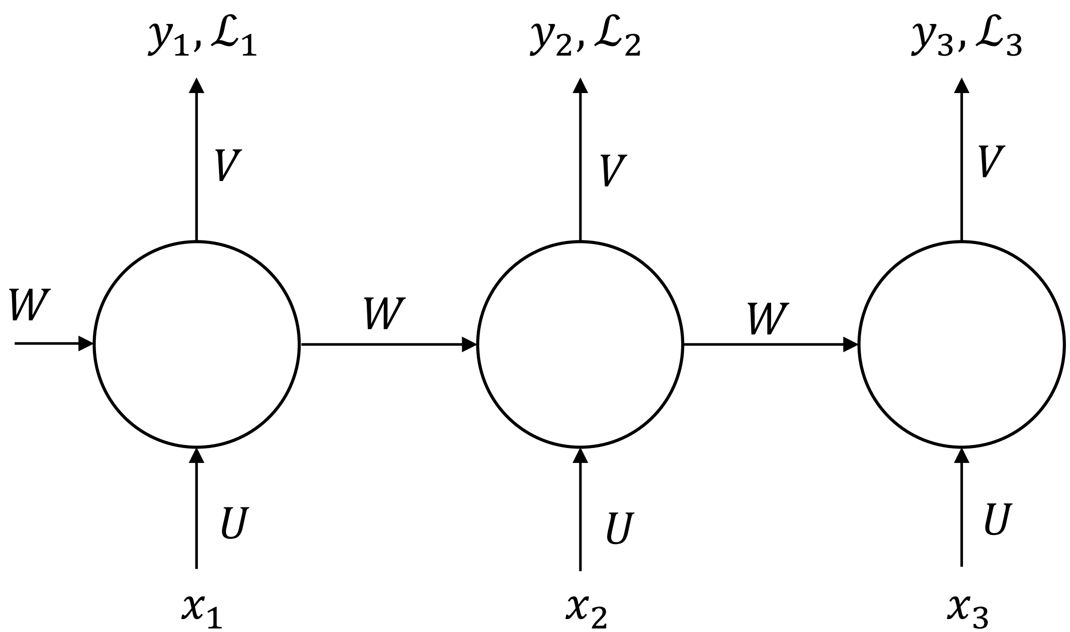
Backpropagation Through Time: BPTT

$$\frac{\partial \mathcal{L}}{\partial V}, \frac{\partial \mathcal{L}}{\partial W}, \frac{\partial \mathcal{L}}{\partial U}$$

برای ساده‌تر کردن، بر روی گام ۳ تمرکز می‌کنیم:

$$\frac{\partial \mathcal{L}_3}{\partial V}, \frac{\partial \mathcal{L}_3}{\partial W}, \frac{\partial \mathcal{L}_3}{\partial U}$$

$$c_t = \tanh(U\,x_t + W c_{t-1})$$
$$y_t = \mathrm{softmax}(V c_t)$$
$$\mathcal{L} = -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t$$



Ref: https://uvadlc.github.io/

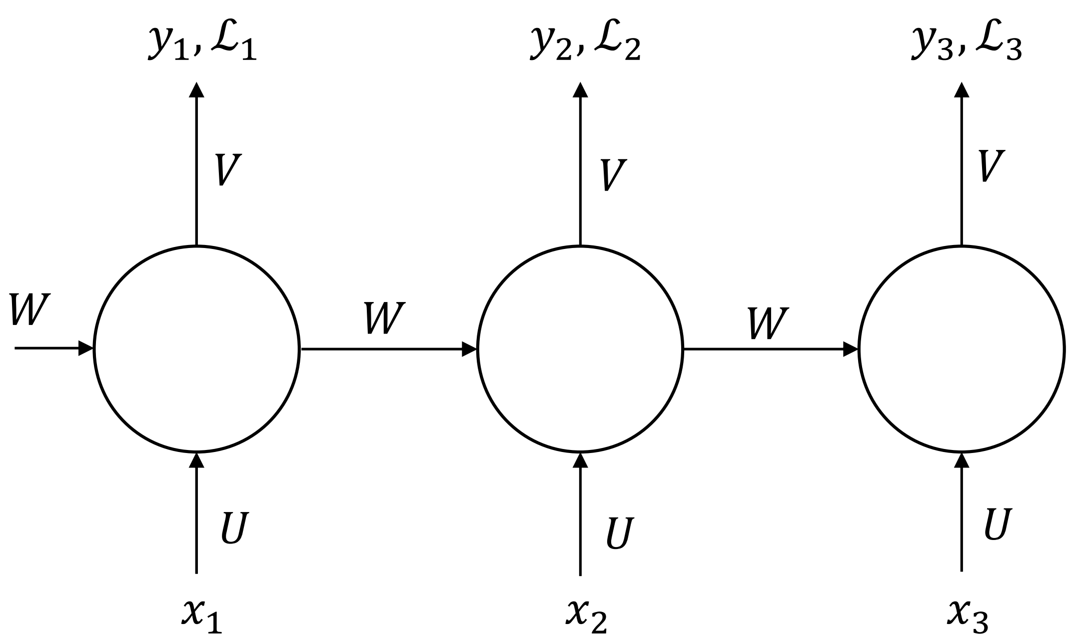
آموزش شبکه‌های عصبی بازگشتی

پس‌انتشار در طول زمان: مثال (۲ از ۴)

Backpropagation Through Time: BPTT

$$\frac{\partial \mathcal{L}_3}{\partial V} = \frac{\partial \mathcal{L}_3}{\partial y_3}\frac{\partial y_3}{\partial V} = (y_3 - l_3)\cdot c_3$$

$$c_t = \tanh(U\,x_t + W c_{t-1})$$
$$y_t = \mathrm{softmax}(V c_t)$$
$$\mathcal{L} = -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t$$



Ref: https://uvadlc.github.io/

آموزش شبکه‌های عصبی بازگشتی

پس‌انتشار در طول زمان: مثال (۳ از ۴)

Backpropagation Through Time: BPTT

$$\frac{\partial \mathcal{L}_3}{\partial W} = \frac{\partial \mathcal{L}_3}{\partial y_3}\frac{\partial y_3}{\partial c_3}\frac{\partial c_3}{\partial W}$$

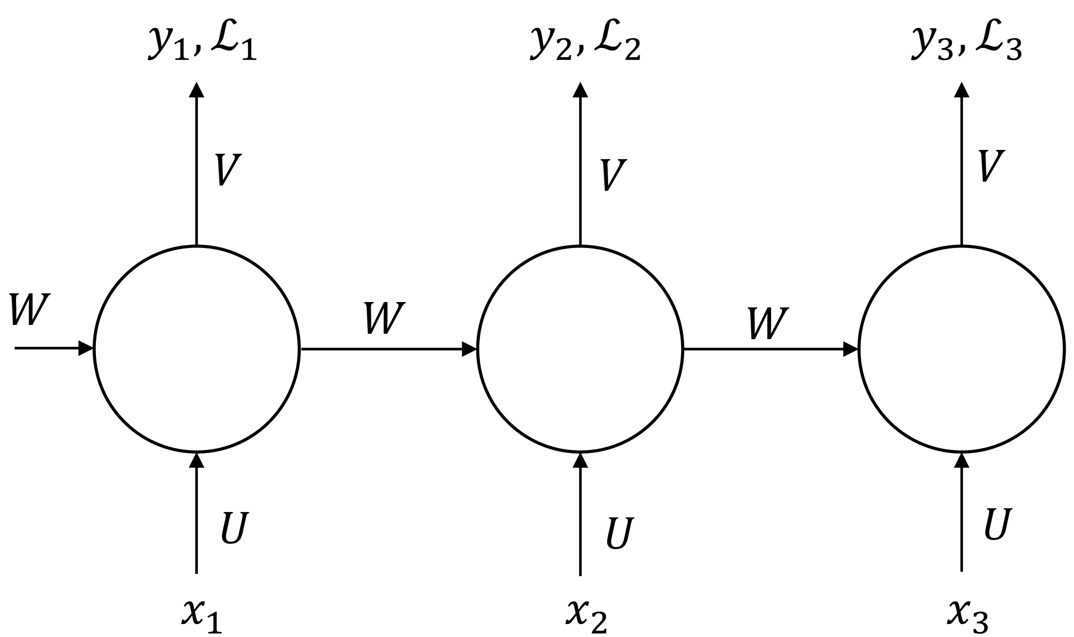
باید رابطه‌ی بین c_3 و W را به‌دست آوریم:

Two-fold: $c_t = \tanh(U\,x_t + W c_{t-1})$

$$\boxed{\frac{\partial\, f(\varphi(x), \psi(x))}{\partial x} = \frac{\partial f}{\partial \varphi}\frac{\partial \varphi}{\partial x} + \frac{\partial f}{\partial \psi}\frac{\partial \psi}{\partial x}}$$

$$\frac{\partial c_3}{\partial W} \propto c_2 + \frac{\partial c_2}{\partial W} \qquad \left(\frac{\partial W}{\partial W} = 1\right)$$

$$\boxed{\begin{aligned} c_t &= \tanh(U\,x_t + W c_{t-1}) \\ y_t &= \mathrm{softmax}(V c_t) \\ \mathcal{L} &= -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t \end{aligned}}$$



آموزش شبکه‌های عصبی بازگشتی

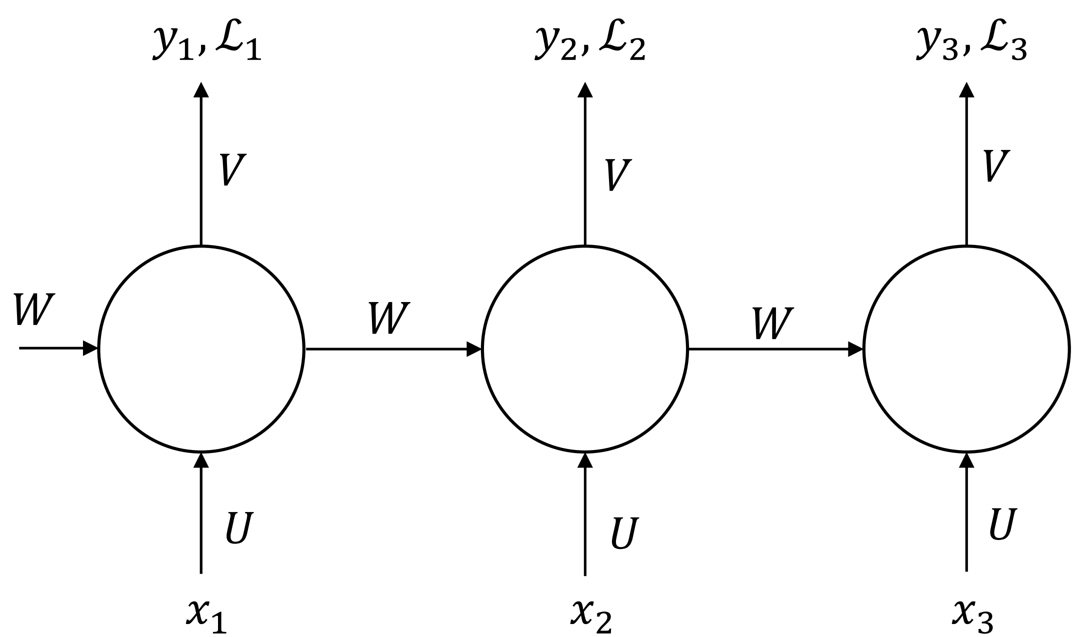
پس‌انتشار در طول زمان: مثال (۴ از ۴)

Backpropagation Through Time: BPTT

به صورت بازگشتی:

$$\left.\begin{aligned}\frac{\partial c_3}{\partial W} &= c_2 + \frac{\partial c_2}{\partial W}\\ \frac{\partial c_2}{\partial W} &= c_1 + \frac{\partial c_1}{\partial W}\\ \frac{\partial c_1}{\partial W} &= c_0 + \frac{\partial c_0}{\partial W}\end{aligned}\right\} \quad \frac{\partial c_3}{\partial W} = \sum_{t=1}^{3} \frac{\partial c_3}{\partial c_t}\frac{\partial c_t}{\partial W} \Rightarrow \boxed{\frac{\partial \mathcal{L}_3}{\partial W} = \sum_{t=1}^{3} \frac{\partial \mathcal{L}_3}{\partial y_3}\frac{\partial y_3}{\partial c_3}\frac{\partial c_3}{\partial c_t}\frac{\partial c_t}{\partial W}}$$

$$\boxed{\begin{aligned} c_t &= \tanh(U\,x_t + W c_{t-1})\\ y_t &= \mathrm{softmax}(V c_t)\\ \mathcal{L} &= -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t\end{aligned}}$$

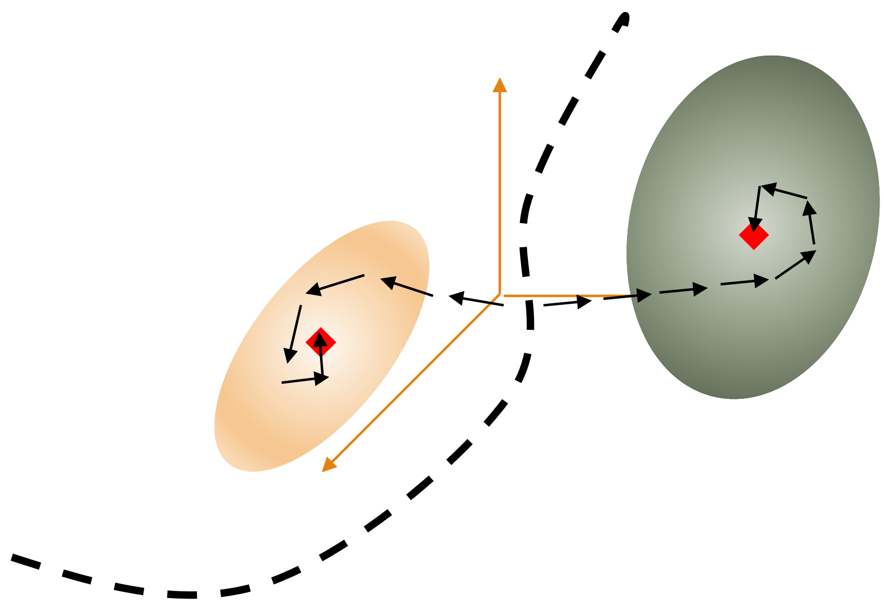


Ref: https://uvadlc.github.io/

آموزش شبکه‌های عصبی بازگشتی

دشواری‌های آموزش

- فضای حافظه‌ی نهفته، از چندین بعد تشکیل شده است.
- یک زیرفضا از فضای حالت حافظه، می‌تواند اطلاعات را ذخیره کند، اگر چند بستر جذب در برخی ابعاد موجود باشد.
- گرادیان‌ها باید در نزدیک بستر جذب قوی باشند.



آموزش شبکه‌های عصبی بازگشتی

دشواری‌های آموزش

آموزش شبکه‌ی عصبی بازگشتی دشوار است، به‌دلیل:

- **اضمحلال گرادیان‌ها (Vanishing Gradients)**
  پس از چند گام زمانی، گرادیان‌ها تقریباً صفر می‌شوند.
- **انفجار گرادیان‌ها (Exploding Gradients)**
  پس از چند گام زمانی، گرادیان‌ها بسیار بزرگ می‌شود.
- **عدم تسخیر وابستگی‌های طولانی-مدت**

فرمول‌بندی جایگزین برای شبکه‌های عصبی بازگشتی

Alternatives Formulation for RNNs

یک فرمول‌بندی جایگزین:

$$c_t = W \cdot \tanh(c_{t-1}) + U \cdot x_t + b$$

$$\mathcal{L} = \sum_t \mathcal{L}_t(c_t)$$

فرمول‌بندی جایگزین برای شبکه‌های عصبی بازگشتی

نگاه دیگری به گرادیان‌ها

Another Look at the Gradients

$$\mathcal{L} = L(c_T(c_{T-1}(\ldots(c_1(x_1, c_0; W); W); W); W)$$

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{t} \frac{\partial \mathcal{L}_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$$

$$\frac{\partial \mathcal{L}}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} = \underbrace{\frac{\partial \mathcal{L}}{\partial c_t} \cdot \frac{\partial c_t}{\partial c_{t-1}} \cdot \frac{\partial c_{t-1}}{\partial c_{t-2}}}_{Rest \,\rightarrow\, \text{short-term factors}} \cdot \underbrace{\ldots \cdot \frac{\partial c_{\tau+1}}{\partial c_\tau}}_{t \gg \tau \,\rightarrow\, \text{long-term factors}} \leq \eta^{t-\tau} \frac{\partial \mathcal{L}_t}{\partial c_t}$$

η determines the norm of the gradients

گرادیان‌های RNN، یک حاصل‌ضرب بازگشتی از $\partial c_t / \partial c_{t-1}$ است.

Ref: https://uvadlc.github.io/

فرمول‌بندی جایگزین برای شبکه‌های عصبی بازگشتی

گرادیان‌های RNN در یک بعد

RNN Gradients in 1D

- $$\underbrace{\frac{\partial \mathcal{L}}{\partial c_t} = \frac{\partial \mathcal{L}}{\partial c_T} \cdot \underset{<1}{\frac{\partial c_T}{\partial c_{T-1}}} \cdot \underset{<1}{\frac{\partial c_{T-1}}{\partial c_{T-2}}} \cdot \ldots \cdot \underset{<1}{\frac{\partial c_{t+1}}{\partial c_{c_t}}}}\quad \frac{\partial \mathcal{L}}{\partial W} \ll 1 \Longrightarrow \text{ Vanishing gradient}$$

- $$\frac{\partial \mathcal{L}}{\partial c_t} = \frac{\partial \mathcal{L}}{\partial c_T} \cdot \underset{>1}{\frac{\partial c_T}{\partial c_{T-1}}} \cdot \underset{>1}{\frac{\partial c_{T-1}}{\partial c_{T-2}}} \cdot \ldots \cdot \underset{>1}{\frac{\partial c_1}{\partial c_{c_t}}}\quad \frac{\partial \mathcal{L}}{\partial W} \gg 1 \Longrightarrow \text{ Exploding gradient}$$

Ref: https://uvadlc.github.io/

فرمول‌بندی جایگزین برای شبکه‌های عصبی بازگشتی

گرادیان‌های RNN در چند بعد

RNN Gradients in n-D

When $c_T \in \mathbb{R}^N$ then $\dfrac{\partial c_t}{\partial c_{t-1}}$ is a Jacobian

- $$\frac{\partial \mathcal{L}}{\partial c_t} = \frac{\partial \mathcal{L}}{\partial c_T} \cdot \underbrace{\frac{\partial c_T}{\partial c_{T-1}}}_{<1} \cdot \underbrace{\frac{\partial c_{T-1}}{\partial c_{T-2}}}_{<1} \cdot \ldots \cdot \underbrace{\frac{\partial c_{t+1}}{\partial c_t}}_{<1} \quad \frac{\partial \mathcal{L}}{\partial \theta} \ll 1 \Longrightarrow$$ Vanishing gradient
- $$\frac{\partial \mathcal{L}}{\partial c_t} = \frac{\partial \mathcal{L}}{\partial c_T} \cdot \underbrace{\frac{\partial c_T}{\partial c_{T-1}}}_{>1} \cdot \underbrace{\frac{\partial c_{T-1}}{\partial c_{T-2}}}_{>1} \cdot \ldots \cdot \underbrace{\frac{\partial c_{t+1}}{\partial c_t}}_{>1} \quad \frac{\partial \mathcal{L}}{\partial \theta} \gg 1 \Longrightarrow$$ Exploding gradient

$$y \in \mathbb{R}^2, x \in \mathbb{R}^3: \ \frac{d\boldsymbol{y}}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial y^{(1)}}{\partial x^{(1)}} & \frac{\partial y^{(1)}}{\partial x^{(2)}} & \frac{\partial y^{(1)}}{\partial x^{(3)}} \\ \frac{\partial y^{(2)}}{\partial x^{(1)}} & \frac{\partial y^{(2)}}{\partial x^{(2)}} & \frac{\partial y^{(2)}}{\partial x^{(3)}} \end{bmatrix}$$

Ref: https://uvadlc.github.io/

فرمول‌بندی جایگزین برای شبکه‌های عصبی بازگشتی

گرادیان‌های RNN در چند بعد

RNN GRADIENTS IN N-D

When $c_T \in \mathbb{R}^N$ then $\frac{\partial c_t}{\partial c_{t-1}}$ is a Jacobian

شعاع طیفی ژاکوبی (= بزرگ‌ترین مقدار ویژه = ρ) پارامتر مهمی است:

- $$\left.\frac{\partial \mathcal{L}}{\partial c_t} = \frac{\partial \mathcal{L}}{\partial c_T} \cdot \underbrace{\frac{\partial c_T}{\partial c_{T-1}}}_{\rho<1} \cdot \underbrace{\frac{\partial c_{T-1}}{\partial c_{T-2}}}_{\rho<1} \cdot \ldots \cdot \underbrace{\frac{\partial c_{t+1}}{\partial c_t}}_{\rho<1} \right\} \quad \frac{\partial \mathcal{L}}{\partial c_t} \ll 1 \Longrightarrow \text{ Vanishing gradient}$$

- $$\left.\frac{\partial \mathcal{L}}{\partial c_t} = \frac{\partial \mathcal{L}}{\partial c_T} \cdot \underbrace{\frac{\partial c_T}{\partial c_{T-1}}}_{\rho>1} \cdot \underbrace{\frac{\partial c_{T-1}}{\partial c_{T-2}}}_{\rho>1} \cdot \ldots \cdot \underbrace{\frac{\partial c_{t+1}}{\partial c_t}}_{\rho>1} \right\} \quad \frac{\partial \mathcal{L}}{\partial c_t} \gg 1 \Longrightarrow \text{ Exploding gradient}$$

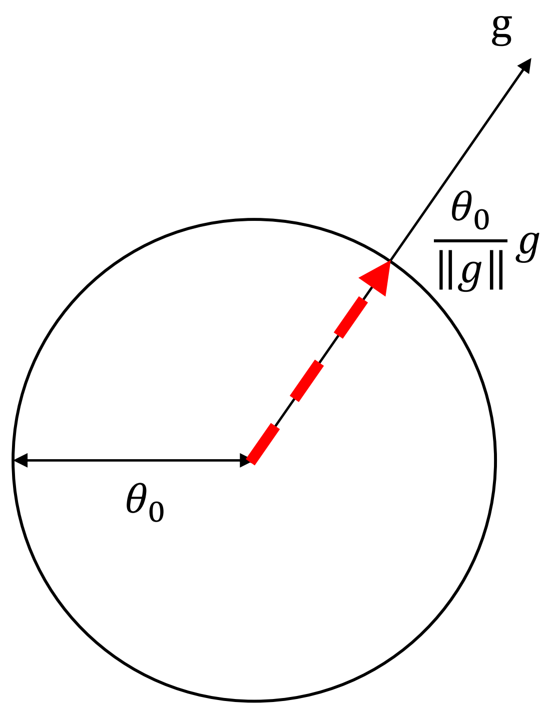
برش گرادیان برای جلوگیری از انفجار گرادیان

Gradient Clipping

مقیاس‌بندی گرادیان با یک مقدار آستانه:

Pseudocode

1. $g \leftarrow \frac{\partial \mathcal{L}}{\partial W}$
2. if $\|g\| > \theta_0$:
   $g \leftarrow \frac{\theta_0}{\|g\|} g$
   else:
   print('Do nothing')



این الگوریتم ساده است، اما به‌خوبی کار می‌کند!

Vanishing Gradients

گرادیان خطا نسبت به سلول میانی:

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{t} \frac{\partial \mathcal{L}_r}{\partial y_t} \frac{\partial y_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$$

$$\frac{\partial c_t}{\partial c_\tau} = \prod_{t \geq k \geq \tau} \frac{\partial c_k}{\partial c_{k-1}} = \prod_{t \geq k \geq \tau} W \cdot \partial \tanh(c_{k-1})$$

Ref: https://uvadlc.github.io/

اضمحلال گرادیان‌ها

Vanishing Gradients

گرادیان خطا نسبت به سلول میانی:

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{t} \frac{\partial \mathcal{L}_r}{\partial y_t} \frac{\partial y_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$$

$$\frac{\partial c_t}{\partial c_\tau} = \prod_{t \geq k \geq \tau} \frac{\partial c_k}{\partial c_{k-1}} = \prod_{t \geq k \geq \tau} W \cdot \partial \tanh(c_{k-1})$$

- For $\mathrm{t} = 1, \mathrm{r} = 2 \Longrightarrow \frac{\partial \mathcal{L}_2}{\partial W} \propto \frac{\partial c_2}{\partial c_1}$
- For $\mathrm{t} = 1, \mathrm{r} = 3 \Longrightarrow \frac{\partial \mathcal{L}_3}{\partial W} \propto \frac{\partial c_3}{\partial c_1} = \frac{\partial c_3}{\partial c_2} \cdot \frac{\partial c_2}{\partial c_1}$
- For $\mathrm{t} = 1, \mathrm{r} = 4 \Longrightarrow \frac{\partial \mathcal{L}_4}{\partial W} \propto \frac{\partial c_4}{\partial c_1} = \frac{\partial c_4}{\partial c_3} \cdot \frac{\partial c_3}{\partial c_2} \cdot \frac{\partial c_2}{\partial c_1}$

اضمحلال گرادیان‌ها

Vanishing Gradients

گرادیان خطا نسبت به سلول میانی:

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{t} \frac{\partial \mathcal{L}_r}{\partial y_t} \frac{\partial y_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$$

$$\frac{\partial c_t}{\partial c_\tau} = \prod_{t \geq k \geq \tau} \frac{\partial c_k}{\partial c_{k-1}} = \prod_{t \geq k \geq \tau} W \cdot \partial \tanh(c_{k-1})$$

وابستگی‌های طولانی–مدت موجب می‌شوند وزن‌ها به‌صورت نمایی کوچک و کوچک‌تر شوند.

Vanishing Gradients

بازمقیاس‌دهی گرادیان‌های مضمحل‌شده راه حل خوبی نیست!

وزن‌ها بین گام‌های زمانی مشترک هستند ⇐ اتلاف‌ها بر روی گام‌های زمانی جمع می‌شوند:

$$\mathcal{L} = \sum_t \mathcal{L}_t \quad \Longrightarrow \quad \frac{\partial \mathcal{L}}{\partial W} = \sum_t \frac{\partial \mathcal{L}_t}{\partial W}$$

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{t} \frac{\partial \mathcal{L}_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W} = \sum_{\tau=1}^{t} \frac{\partial \mathcal{L}_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$$

بازمقیاس‌دهی برای یک گام زمانی، بر همه‌ی گام‌های زمانی تأثیر می‌گذارد

$\Downarrow$

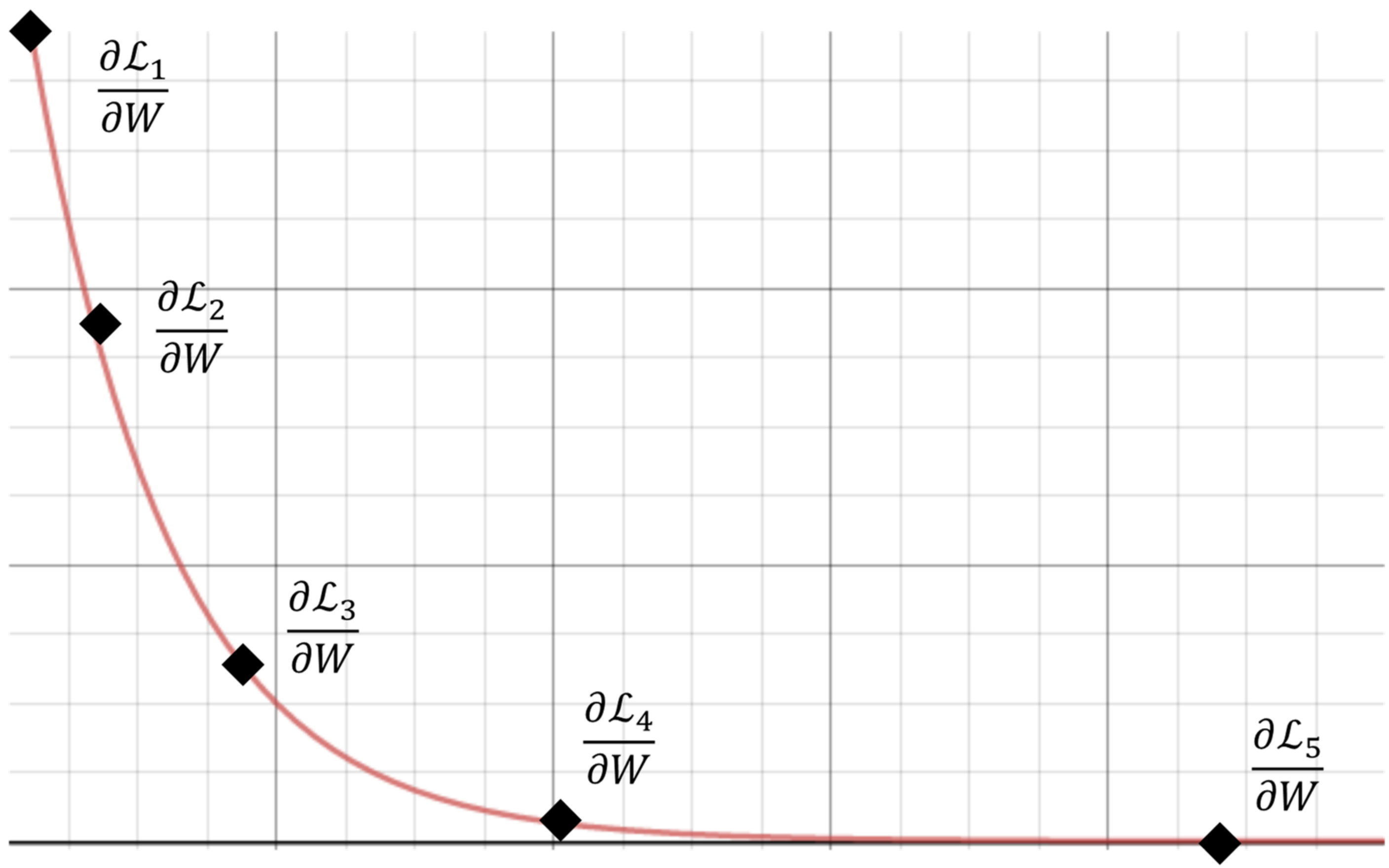
ضریب بازمقیاس‌دهی برای یک گام زمانی، برای گام زمانی دیگر کار نمی‌کند!

اضمحلال گرادیان‌ها

مثال (۱ از ۲)

Vanishing Gradients

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}_1}{\partial W} + \frac{\partial \mathcal{L}_2}{\partial W} + \frac{\partial \mathcal{L}_3}{\partial W} + \frac{\partial \mathcal{L}_4}{\partial W} + \frac{\partial \mathcal{L}_5}{\partial W}$$



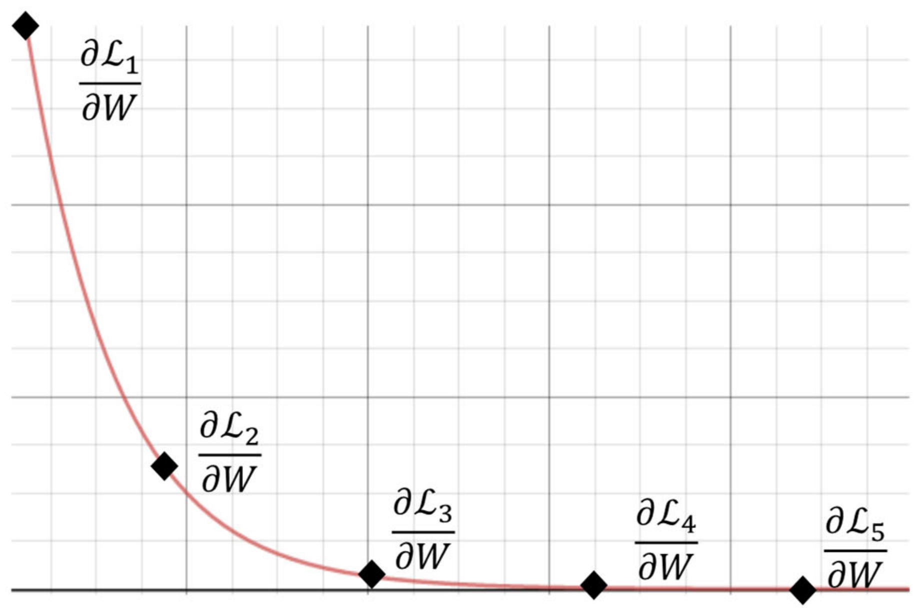
Ref: https://uvadlc.github.io/

اضمحلال گرادیان‌ها

مثال (۲ از ۲)

VANISHING GRADIENTS

- Let's say $\frac{\partial \mathcal{L}_1}{\partial w} \propto 1, \frac{\partial \mathcal{L}_2}{\partial w} \propto 1/10, \frac{\partial \mathcal{L}_3}{\partial w} \propto 1/100, \frac{\partial \mathcal{L}_4}{\partial w} \propto 1/1000, \frac{\partial \mathcal{L}_5}{\partial w} \propto 1/10000$
- $\frac{\partial \mathcal{L}}{\partial w} = \sum_r \frac{\partial \mathcal{L}_r}{\partial w} = 1.1111$
- If $\frac{\partial \mathcal{L}}{\partial w}$ rescaled to 1 → $\frac{\partial \mathcal{L}_5}{\partial w} \propto 10^{-5}$



وابستگی‌های طولانی-مدت قابل چشم‌پوشی است:
(یادگیری فقط بر روی کوتاه-مدت تمرکز می‌کند.)

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}_1}{\partial W} + \frac{\partial \mathcal{L}_2}{\partial W} + \frac{\partial \mathcal{L}_3}{\partial W} + \frac{\partial \mathcal{L}_4}{\partial W} + \frac{\partial \mathcal{L}_5}{\partial W}$$

اضمحلال گرادیان‌ها

رفع مشکل

Fixing Vanishing Gradients

❋ رگولاریزاسیون بر روی وزن‌های بازگشتی (سیگنال خطا را مجبور می‌کند که مضمحل نشود).

$$\Omega = \sum_t \Omega_t = \sum_t \left(\frac{\left| \frac{\partial \mathcal{L}}{\partial c_{t+1}} \frac{\partial c_{t+1}}{\partial c_t} \right|}{\left| \frac{\partial \mathcal{L}}{\partial c_{t+1}} \right|} - 1 \right)^2$$

❋ استفاده از ماژول‌های بازگشتی پیشرفته، مانند:

- **ماژول حافظه‌ی کوتاه‌مدت طولانی** (Long Short-Term Memory Module)
- **ماژول واحد بازگشتی دروازه‌گذاری شده** (Gated Recurrent Unit Module)

اضمحلال گرادیان‌ها

رفع مشکل

Fixing Vanishing Gradients

سیگنال خطا در طول زمان، باید دارای نُرم خیلی بزرگ یا خیلی کوچک نباشد.

راه‌حل: استفاده از یک تابع فعال‌سازی که مشتق آن مساوی با 1 باشد.

$\Downarrow$

گرادیان‌ها نه خیلی بزرگ می‌شوند و نه خیلی کوچک

۴

شبکه‌های عصبی بازگشتی عمیق

DEEP RNNS

Multilayer RNNs

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

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

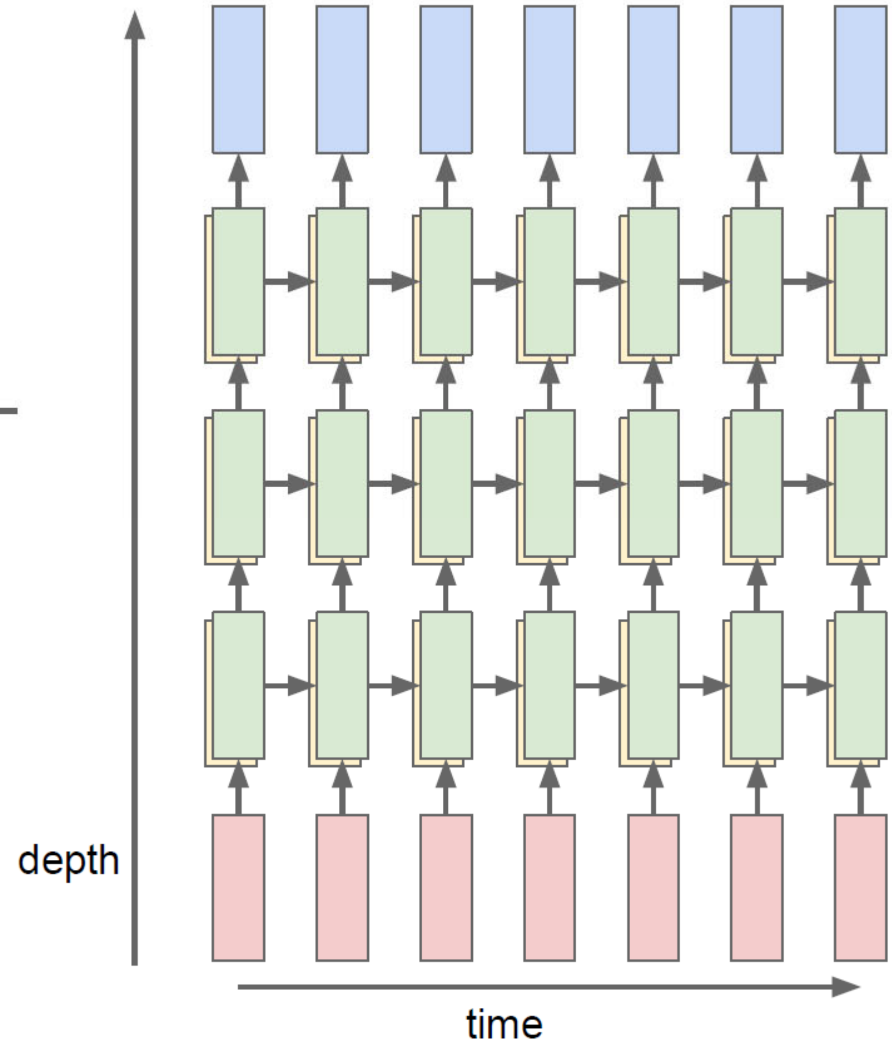
LSTM:

$W^l \ [4n \times 2n]$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

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

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



شبکه‌های عصبی بازگشتی

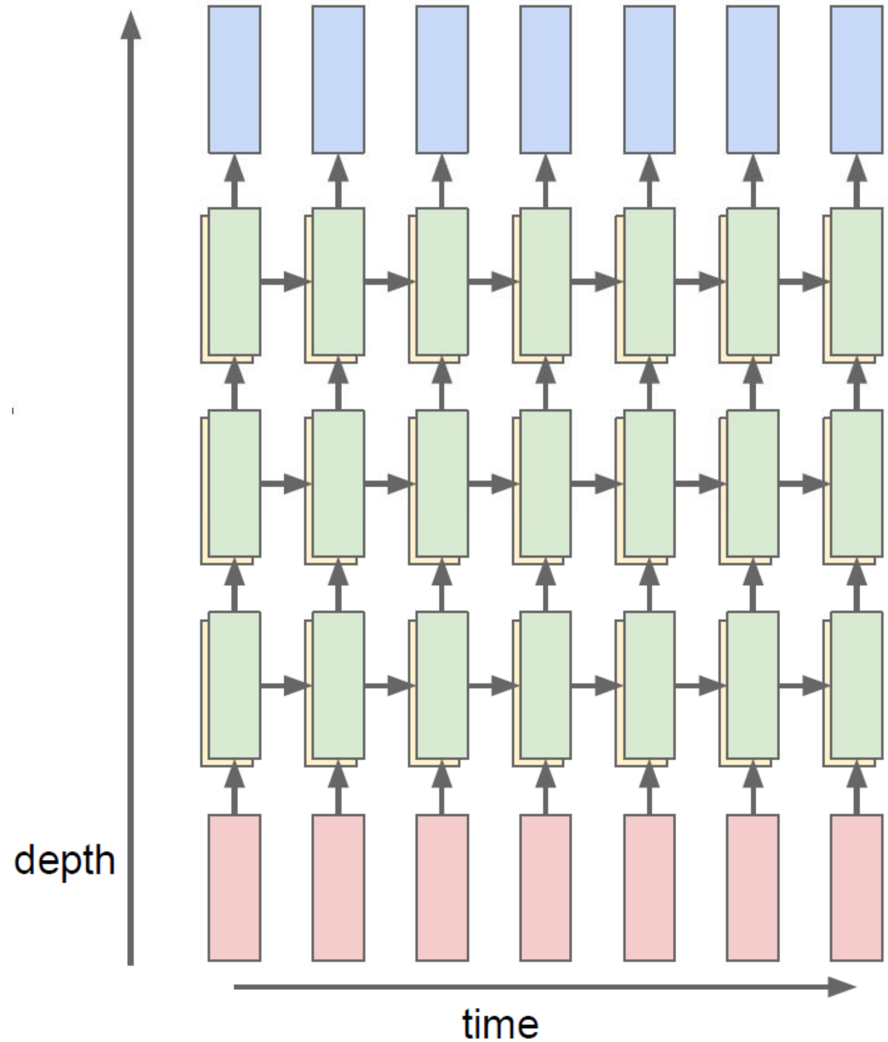
شبکه‌های عصبی بازگشتی چندلایه

MULTILAYER RNNS

Multilayer RNNs

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

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

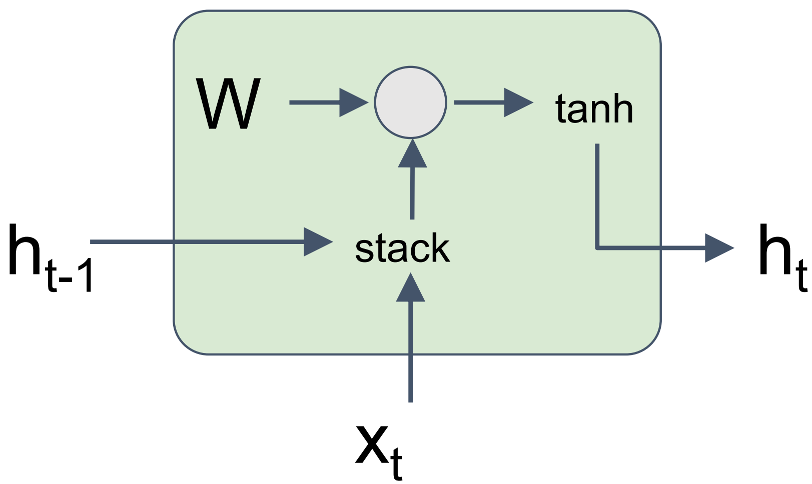


Ref: http://cs231n.stanford.edu/

شبکه‌های عصبی بازگشتی ساده

جریان گرادیان

VANILLA RNN GRADIENT FLOW



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

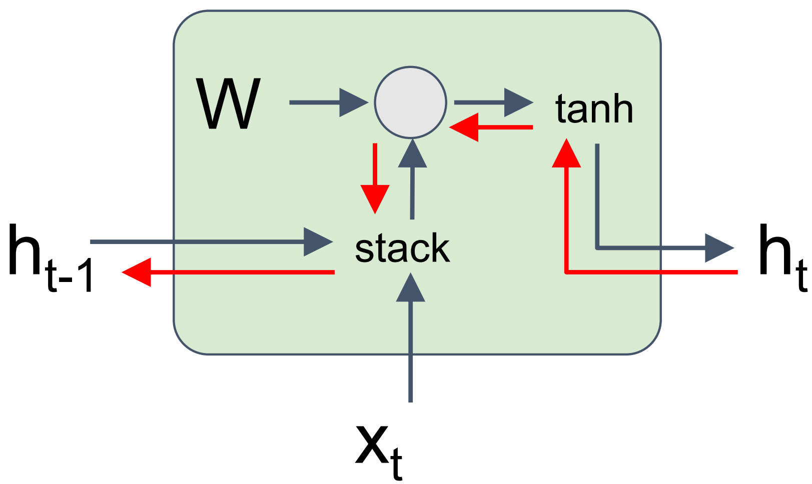
Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013

شبکه‌های عصبی بازگشتی ساده

جریان گرادیان

VANILLA RNN GRADIENT FLOW

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



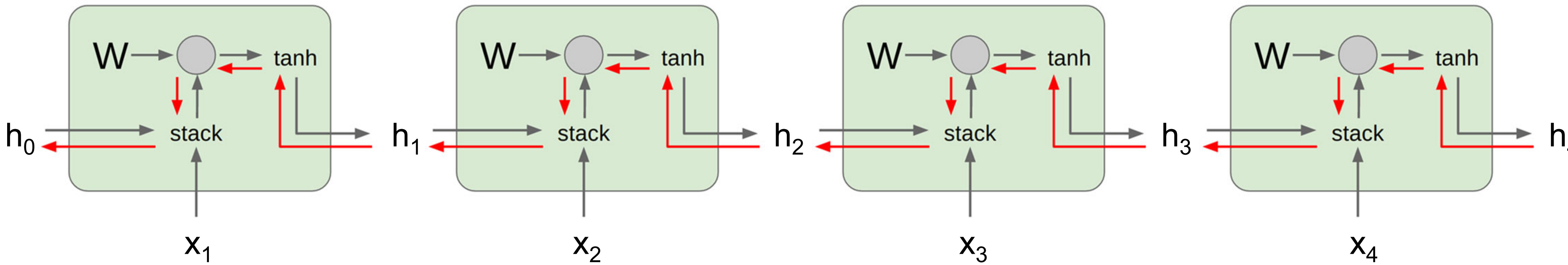
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

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

شبکه‌های عصبی بازگشتی ساده

جریان گرادیان

Vanilla RNN Gradient Flow



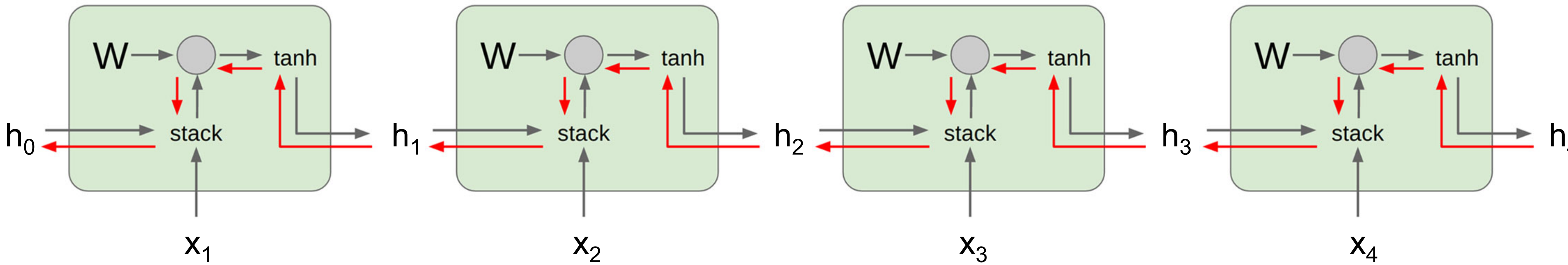
Computing gradient of h_0 involves many factors of W (and repeated tanh)

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

شبکه‌های عصبی بازگشتی ساده

جریان گرادیان

VANILLA RNN GRADIENT FLOW



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

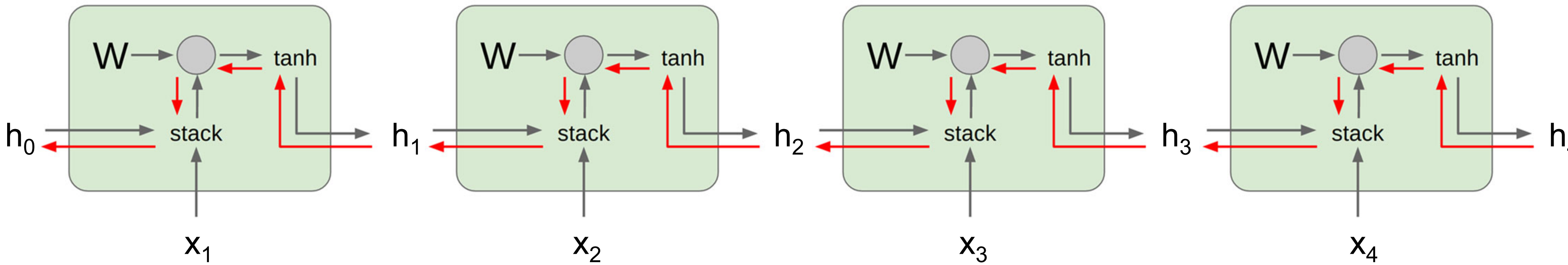
Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

Ref: http://cs231n.stanford.edu/

شبکه‌های عصبی بازگشتی ساده

جریان گرادیان

Vanilla RNN Gradient Flow



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

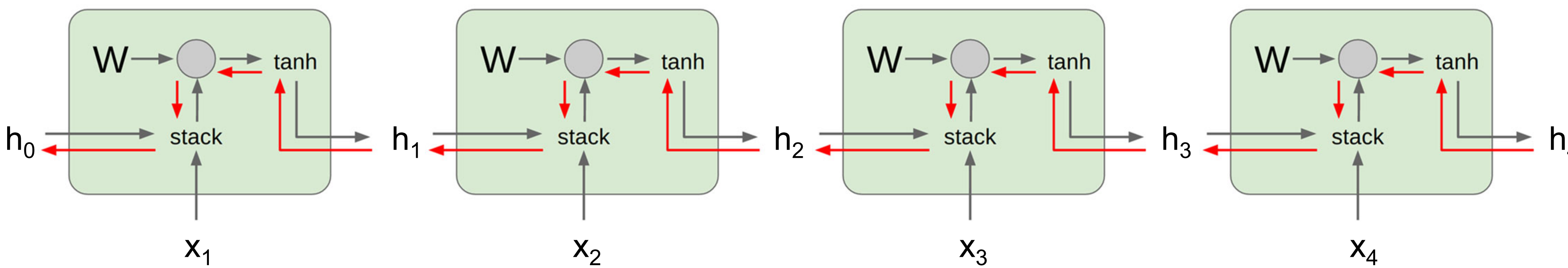
```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

Largest singular value < 1:
Vanishing gradients

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

جریان گرادیان

Vanilla RNN Gradient Flow



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

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

شبکه‌های عصبی بازگشتی

۵

حافظه‌ی کوتاه-مدت طولانی

مقایسه

LONG SHORT-TERM MEMORY (LSTM)

Vanilla RNN

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

LSTM

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

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

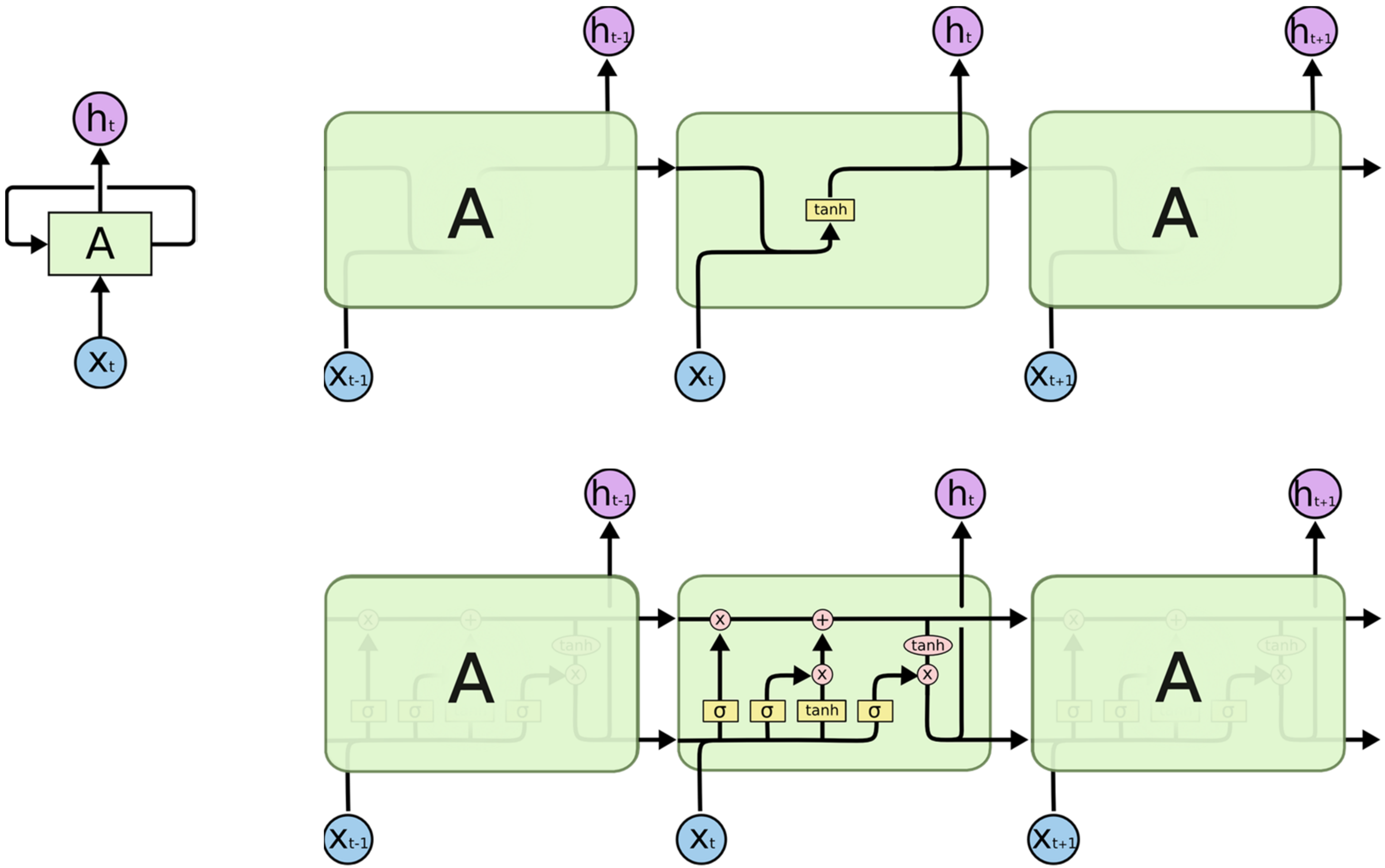
$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation 1997

حافظه‌ی کوتاه-مدت طولانی

مقایسه

LONG SHORT-TERM MEMORY (LSTM)

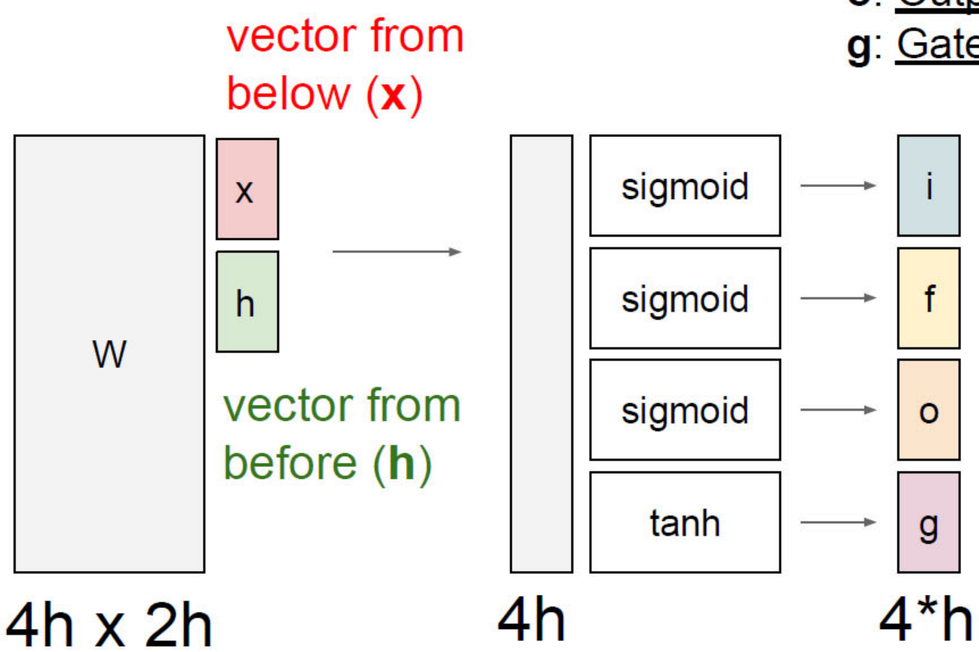


Ref: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

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

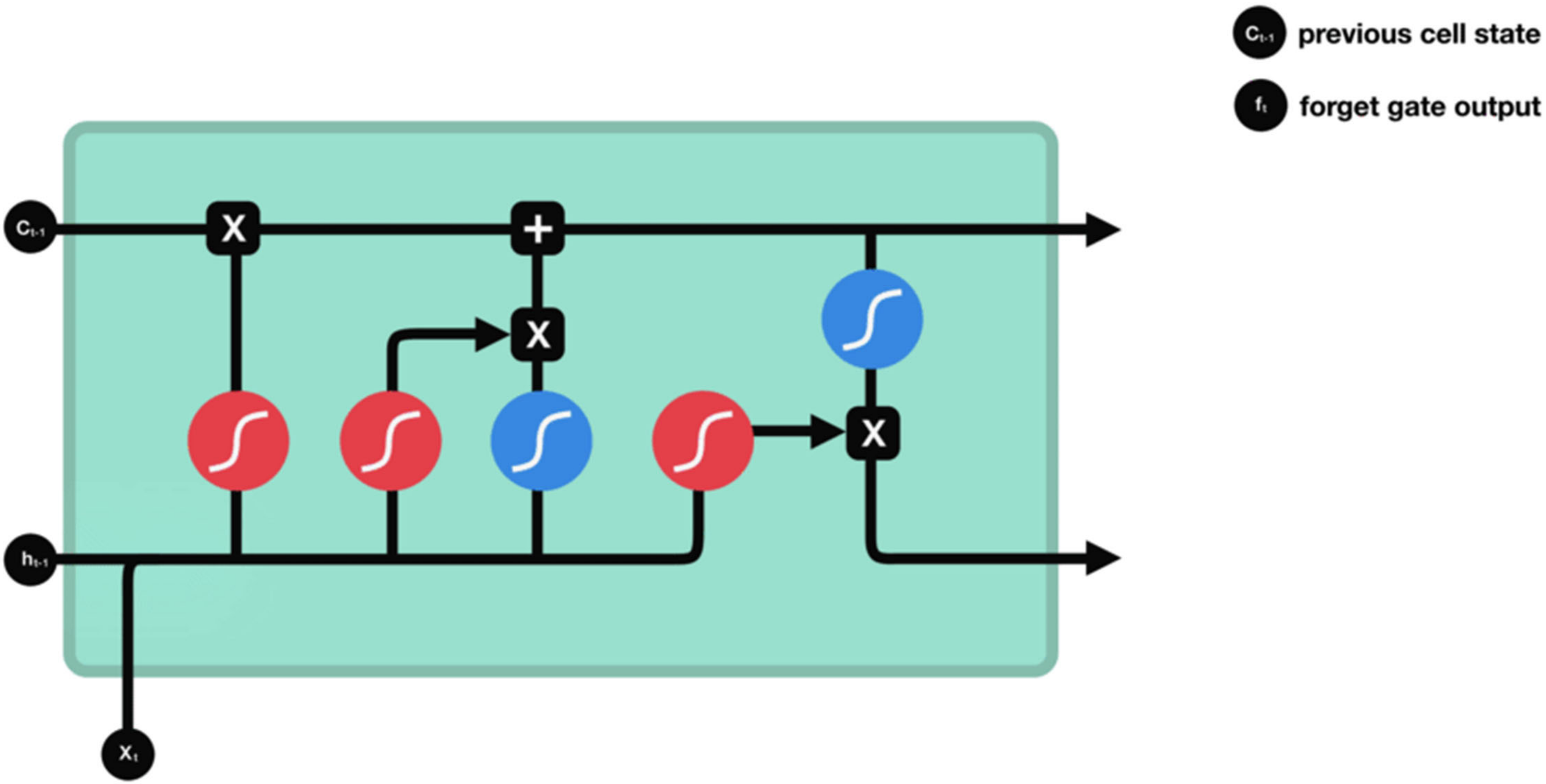


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

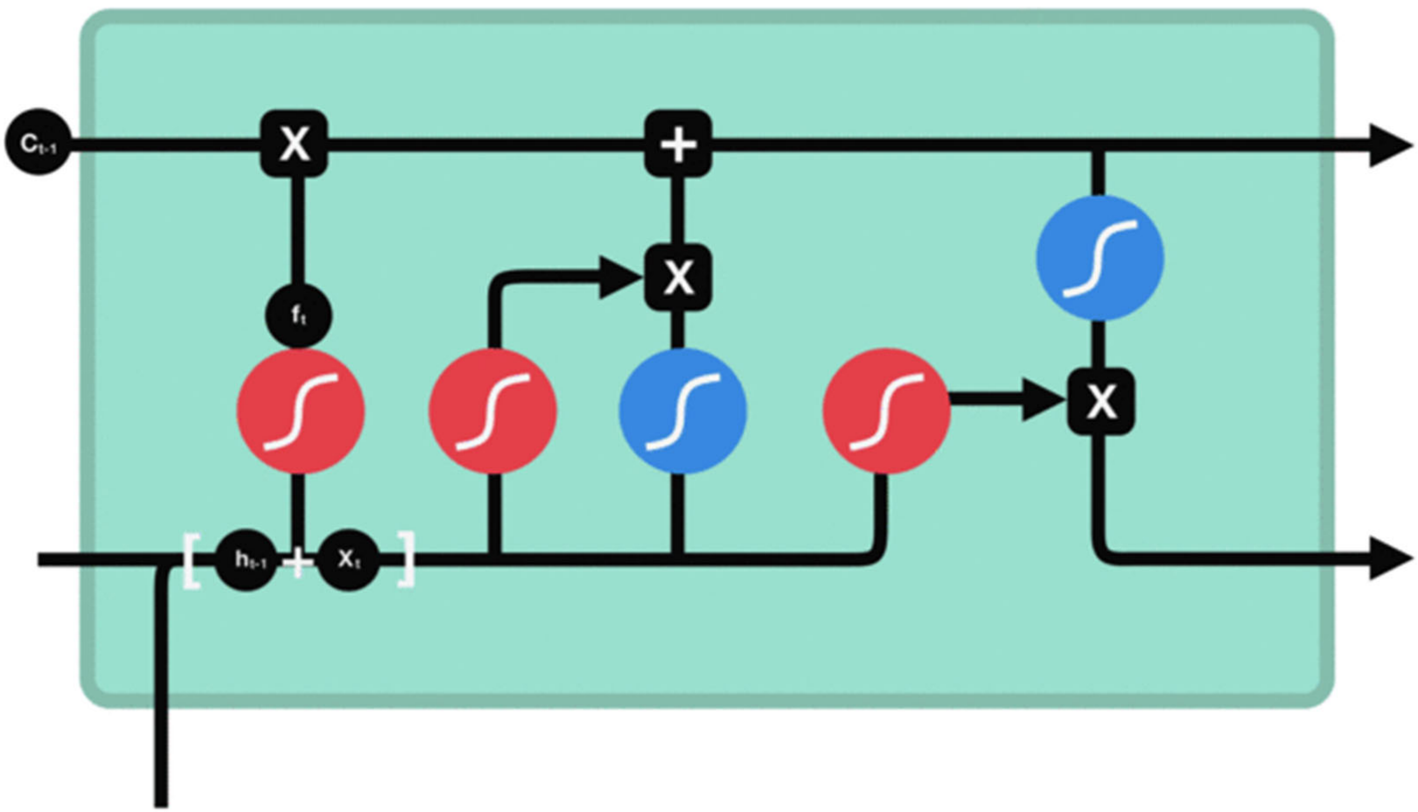
$$c_t = f \odot c_{t-1} + i \odot g$$

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

Ref: http://cs231n.stanford.edu/



c_{t-1}
h_{t-1}
x_t
c_{t-1} previous cell state
f_t forget gate output



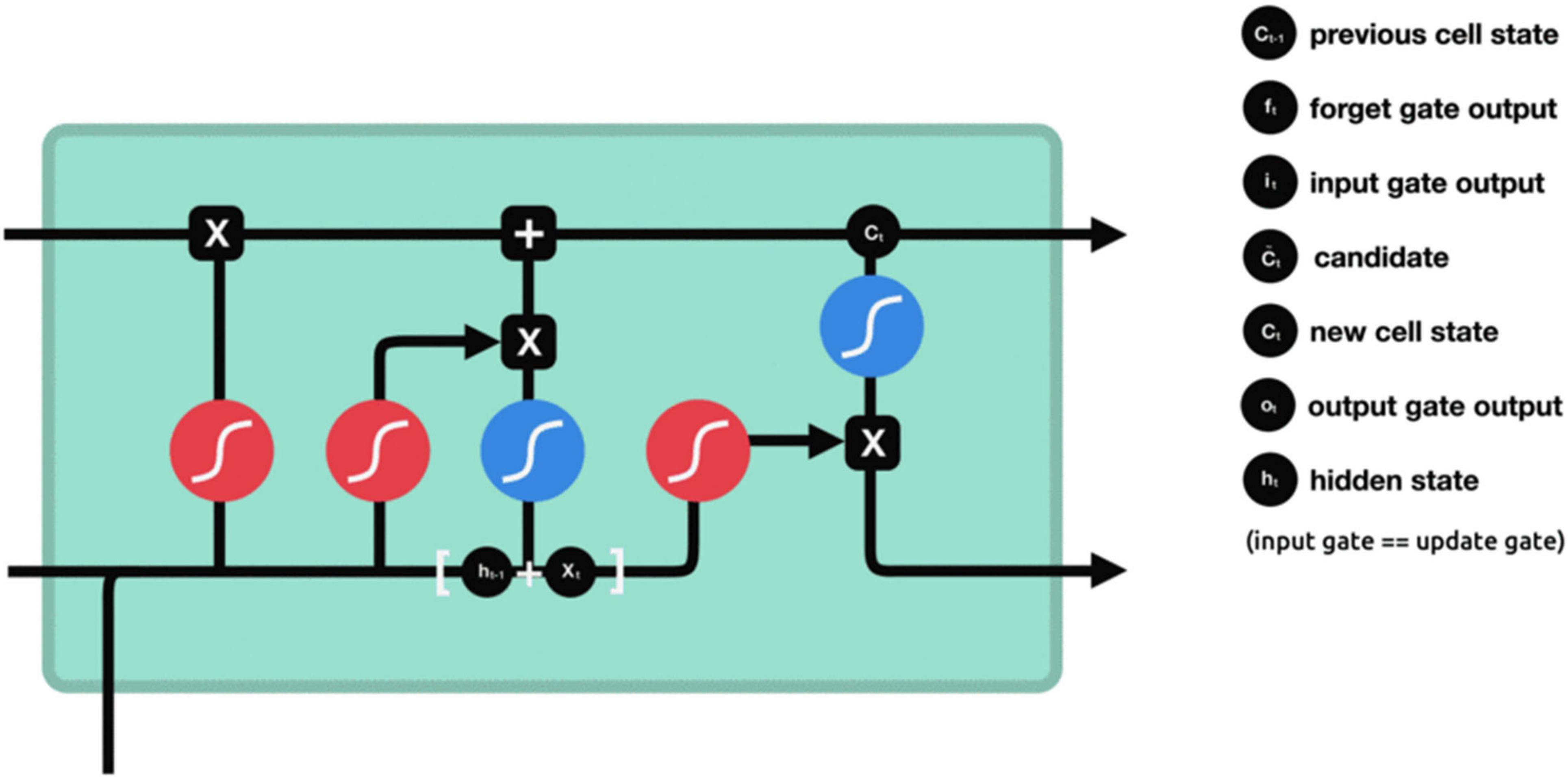
c_{t-1} **previous cell state**

f_t **forget gate output**

i_t **input gate output**

$\tilde{c}_t$ **candidate**

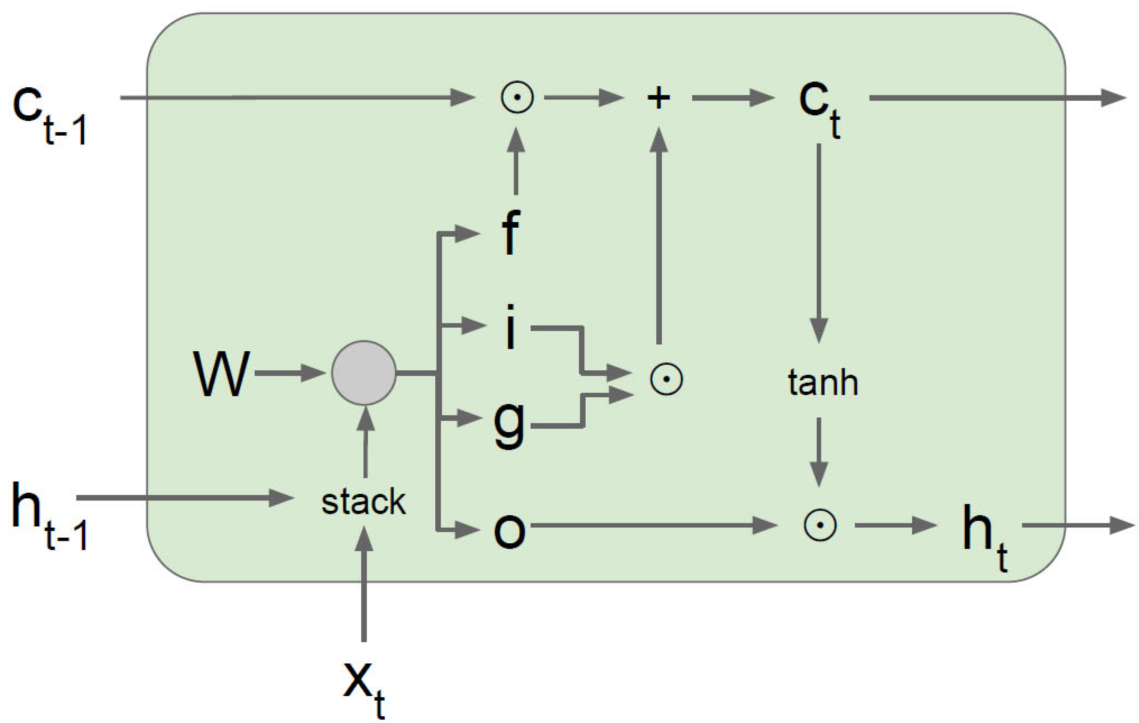
(input gate == update gate)


X
+
X
X
previous cell state
forget gate output
input gate output
candidate
new cell state
output gate output
hidden state
(input gate == update gate)

حافظه‌ی کوتاه-مدت طولانی

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



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

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

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

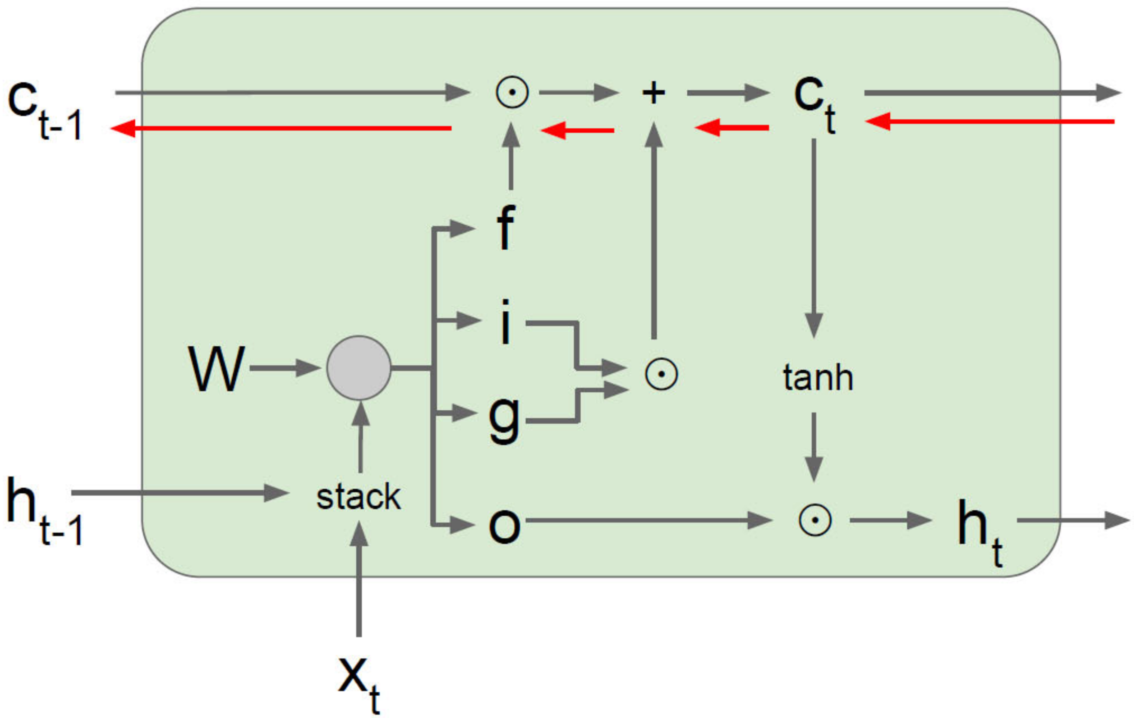
Ref: http://cs231n.stanford.edu/

حافظه‌ی کوتاه-مدت طولانی

جریان گرادیان

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



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

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

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

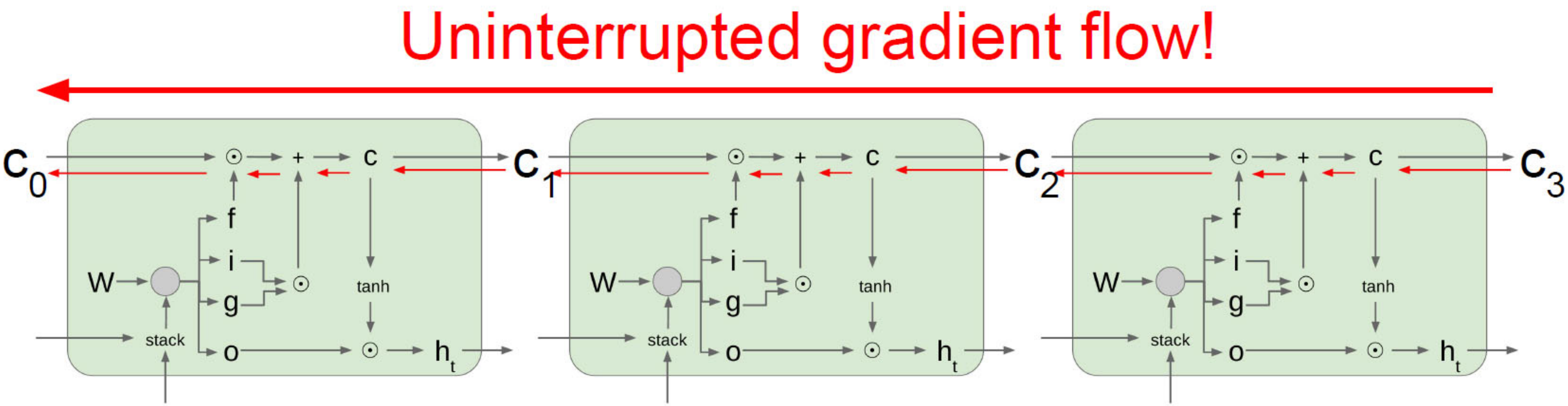
$$h_t = o \odot \tanh(c_t)$$

حافظه‌ی کوتاه-مدت طولانی

جریان گرادیان

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

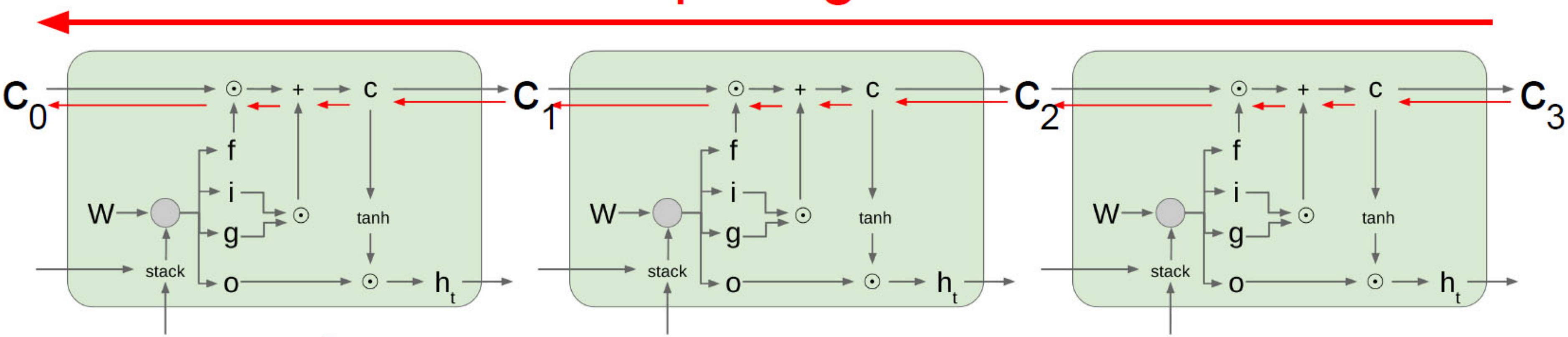


جریان گرادیان

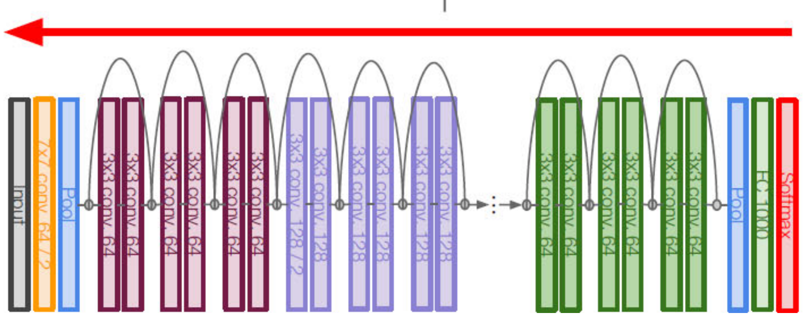
Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

Uninterrupted gradient flow!



Similar to ResNet!



Ref: http://cs231n.stanford.edu/

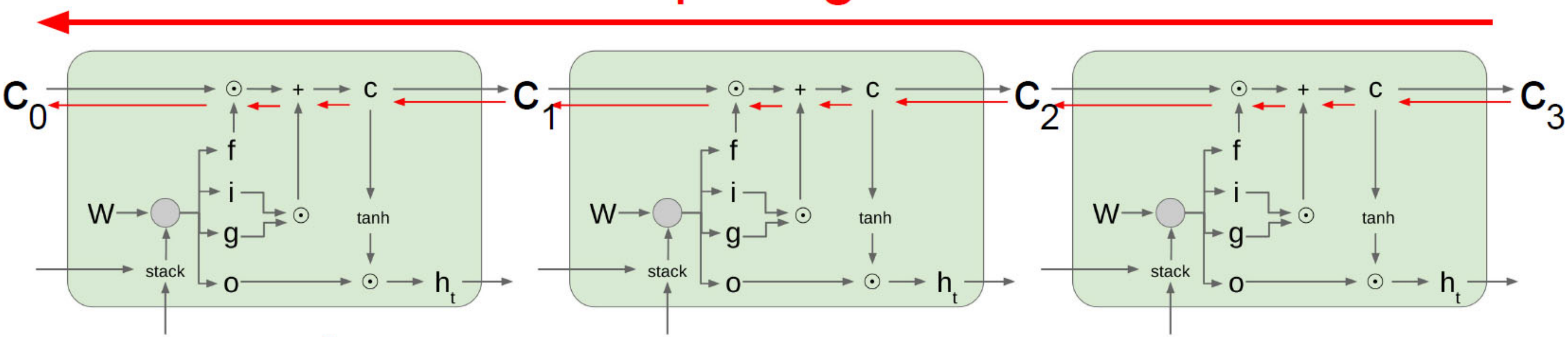
حافظه‌ی کوتاه-مدت طولانی

جریان گرادیان

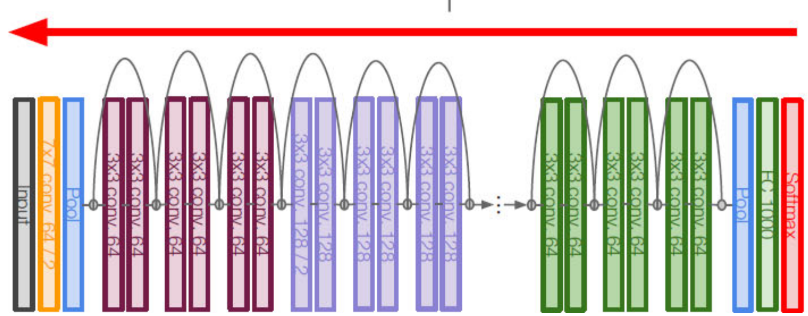
Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]

Uninterrupted gradient flow!



Similar to ResNet!



In between:

Highway Networks

$$g = T(x, W_T)$$

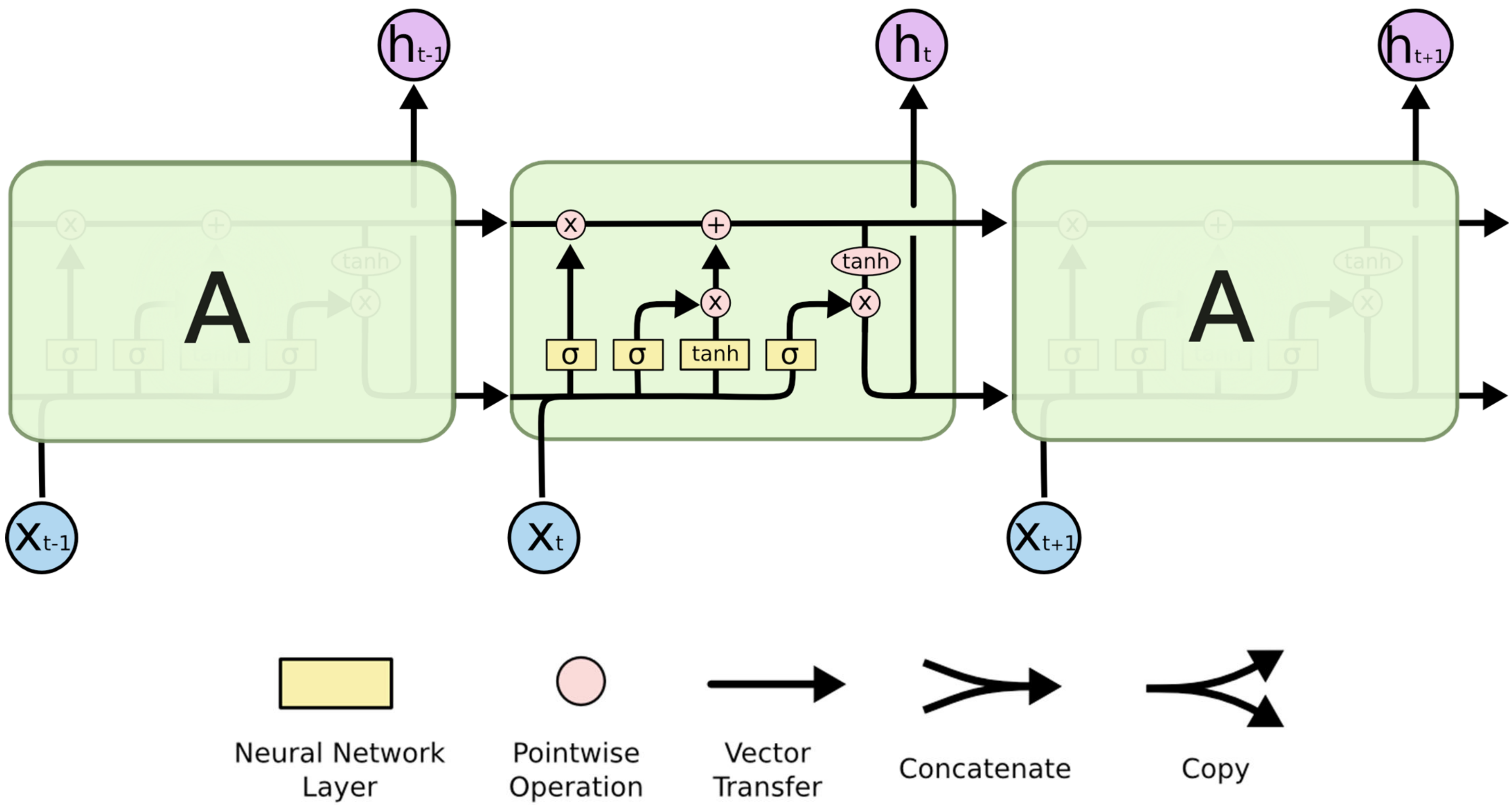
$$y = g \odot H(x, W_H) + (1 - g) \odot x$$

Srivastava et al, “Highway Networks”, ICML DL Workshop 2015

حافظه‌ی کوتاه-مدت طولانی

نشانه‌ها

[Meet LSTMs](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

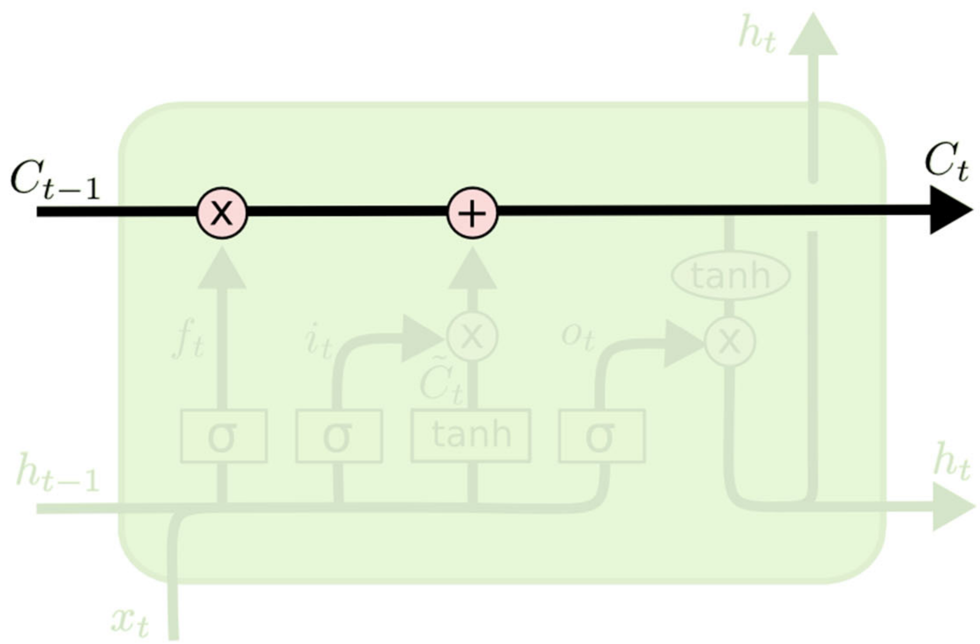


Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM: حافظه

LSTMs Intuition: Memory

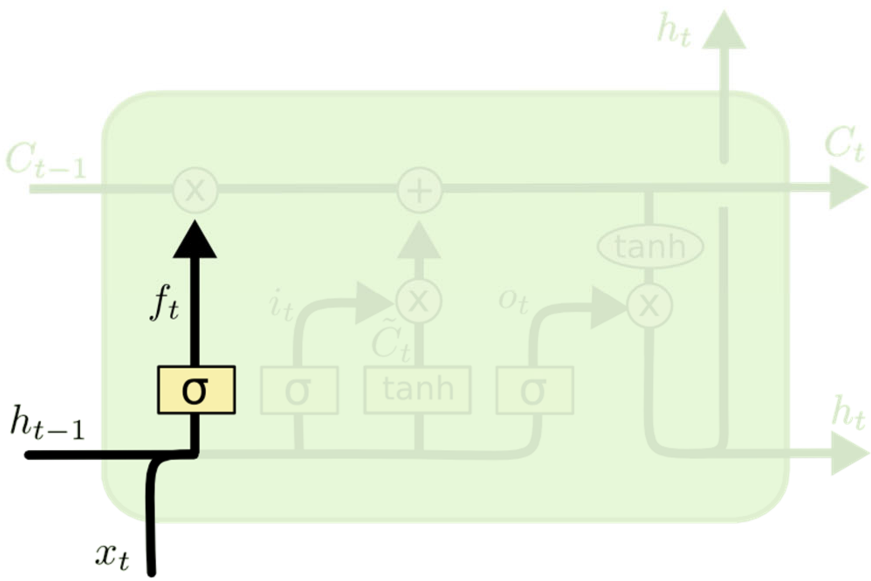


حالت سلول / حافظه
Cell State / Memory

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM : گیت فراموشی

LSTMs Intuition: Forget Gate



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

آیا باید این بیت از اطلاعات را به‌یاد بیاوریم یا خیر؟

Should we continue to remember this “bit” of information or not?

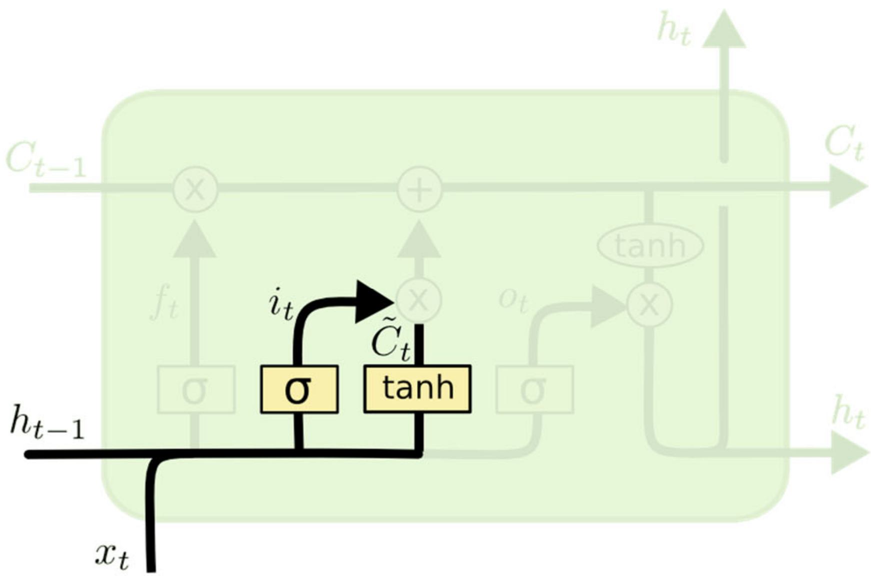
مثال: با دیدن یک فاعل جدید در جمله، می‌خواهیم اطلاعات مربوط به فاعل قبلی (مانند جنسیت و تعداد) را فراموش کنیم.

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM: گیت ورودی

LSTMs Intuition: Input Gate



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

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

آیا باید این بیت از اطلاعات را به‌یاد بیاوریم یا خیر؟
اگر این‌طور است، با کدام ورودی؟

Should we continue to remember this “bit” of information or not?
If so, with what?

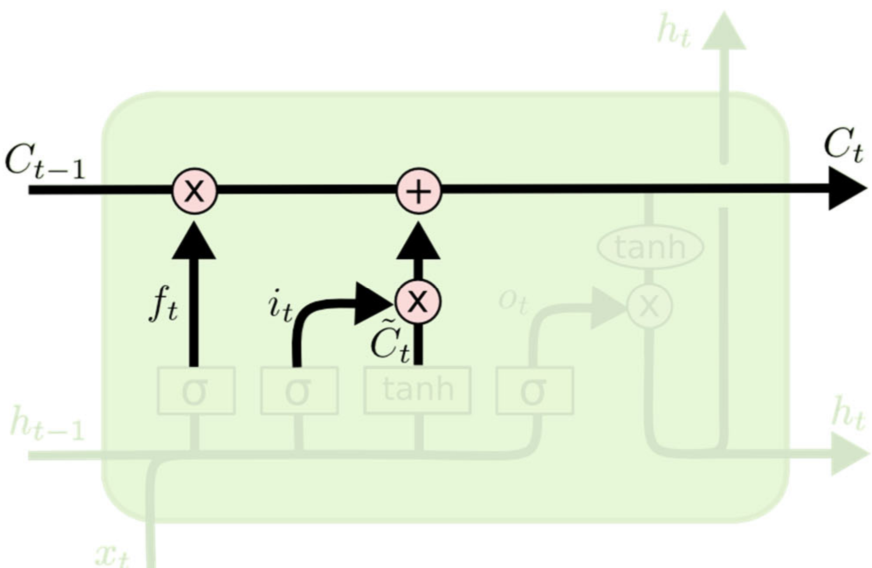
مثال: با دیدن یک فاعل جدید در جمله، می‌خواهیم اطلاعات مربوط به فاعل جدید (مانند جنسیت و تعداد) را به‌خاطر بسپاریم.

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM: به‌روزرسانی حافظه

LSTMs Intuition: Memory Update



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

آن را فراموش کن + این را به‌خاطر بسپار

Forget that + Memorize this

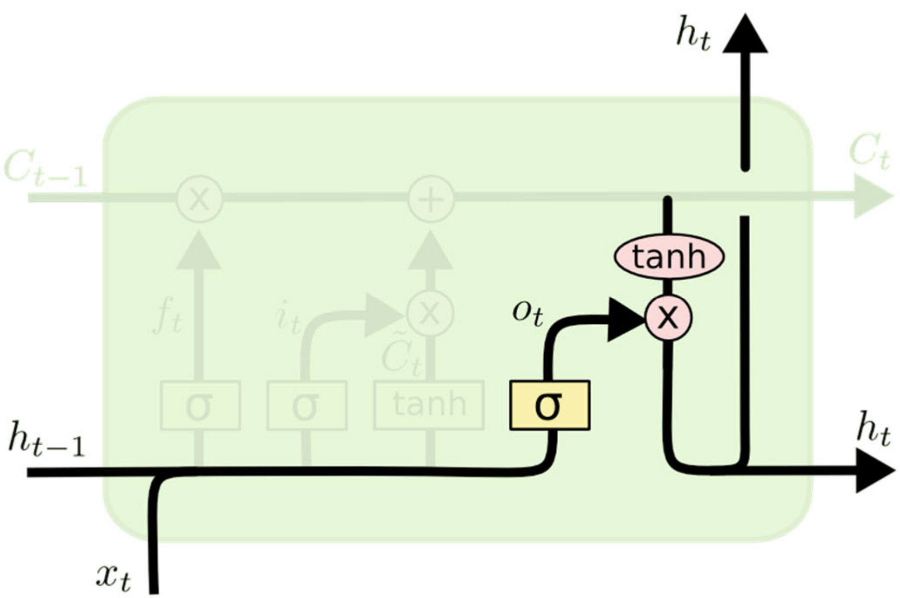
مثال: حذف اطلاعات غیرضروری از حافظه و افزودن اطلاعات مفید جدید به آن

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM: گیت خروجی

LSTMs Intuition: Output Gate



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

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

آیا باید این بیت از اطلاعات را به سمت لایه‌های عمیق‌تر شبکه خارج کنیم؟

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

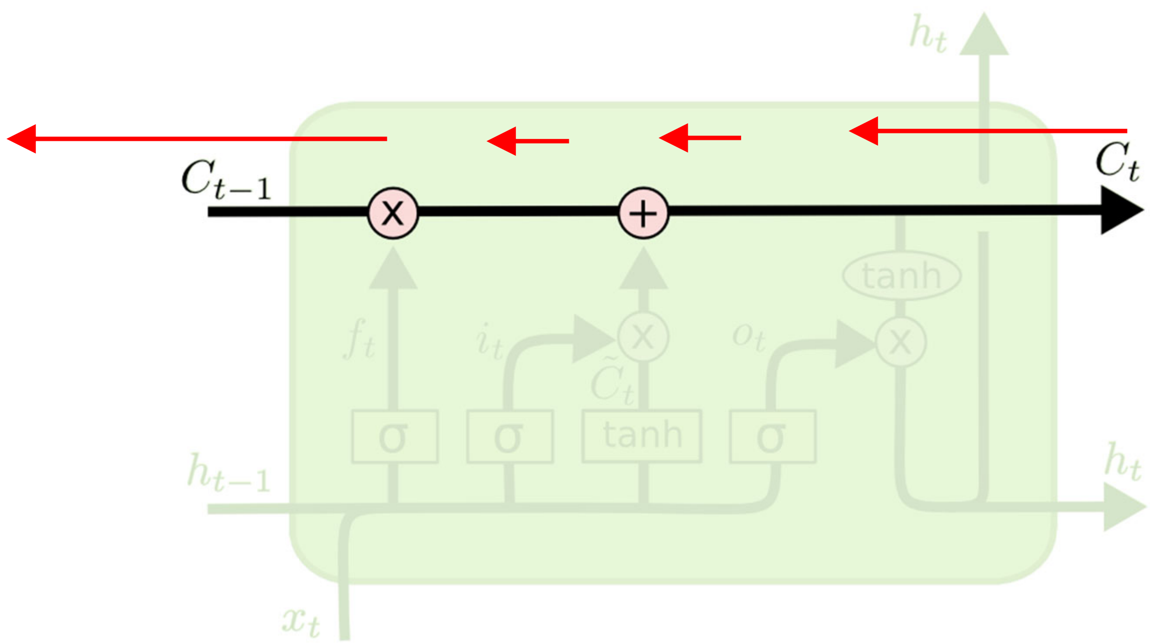
مثال: فرستادن اطلاعات فاعل جمله (مثل جنسیت و تعداد) به خروجی به‌منظور صرف درست فعل پس از آن

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM : به‌روزرسانی‌های جمعی

LSTMs Intuition: Additive Updates



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

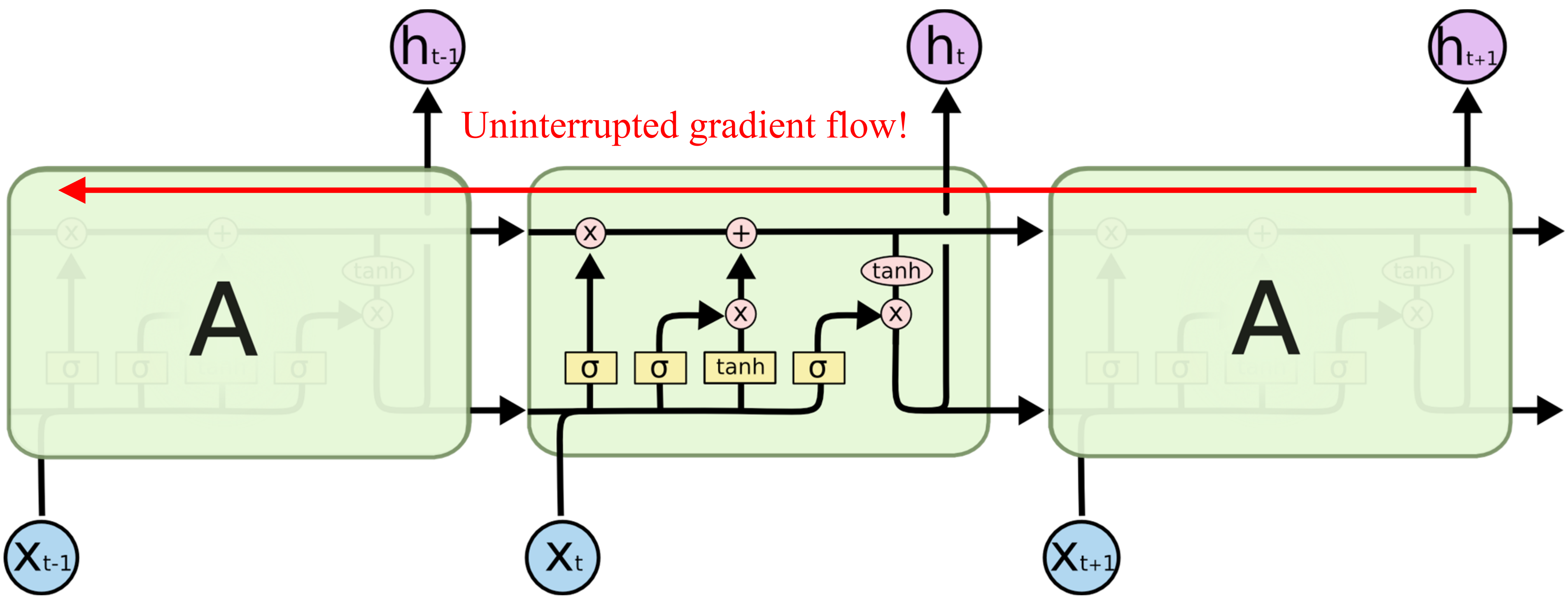
پس‌انتشار از C_t به C_{t-1}
تنها با ضرب عنصربه‌عنصر در f انجام می‌شود،
هیچ ضرب ماتریسی در $\boldsymbol{W}$ نداریم.

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM: به‌روزرسانی‌های جمعی

LSTMs Intuition: Additive Updates

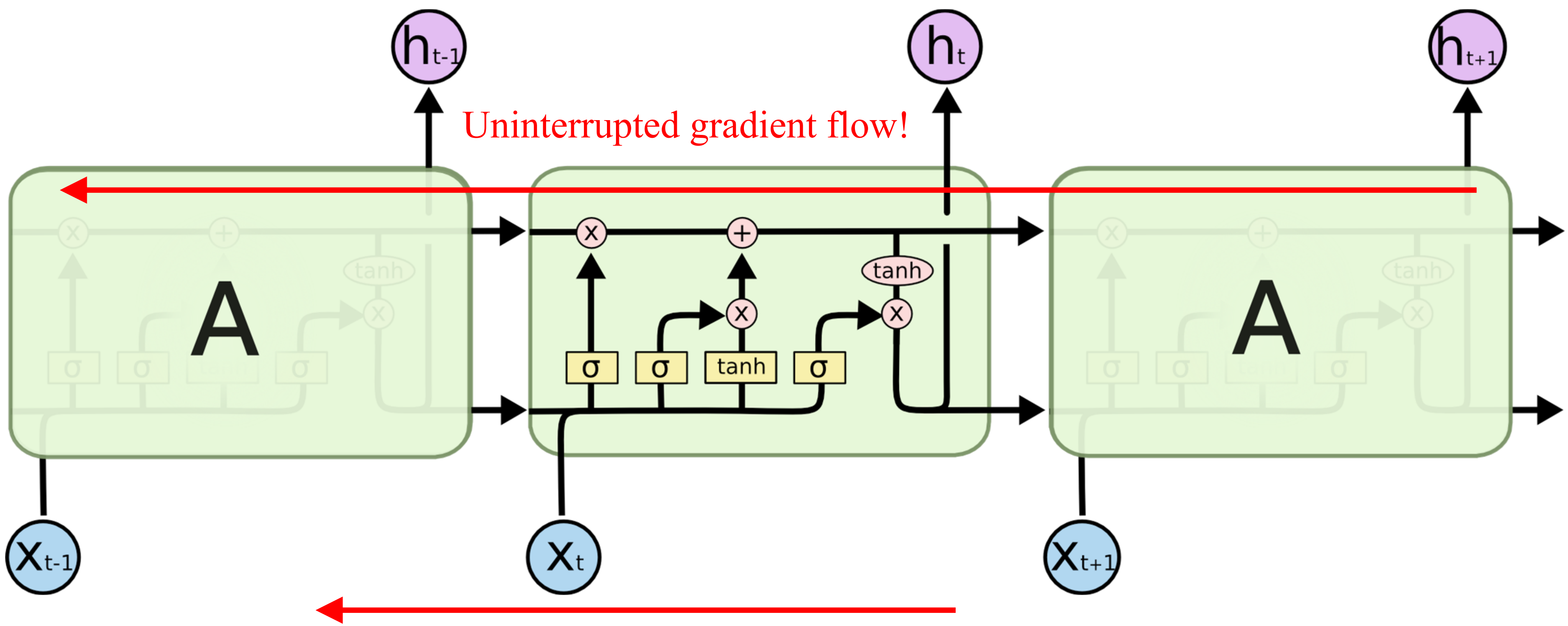


Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

درک شهودی LSTM: به‌روزرسانی‌های جمعی

LSTMs Intuition: Additive Updates



Similar to ResNet!

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

Long Short-Term Memory

$$i = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

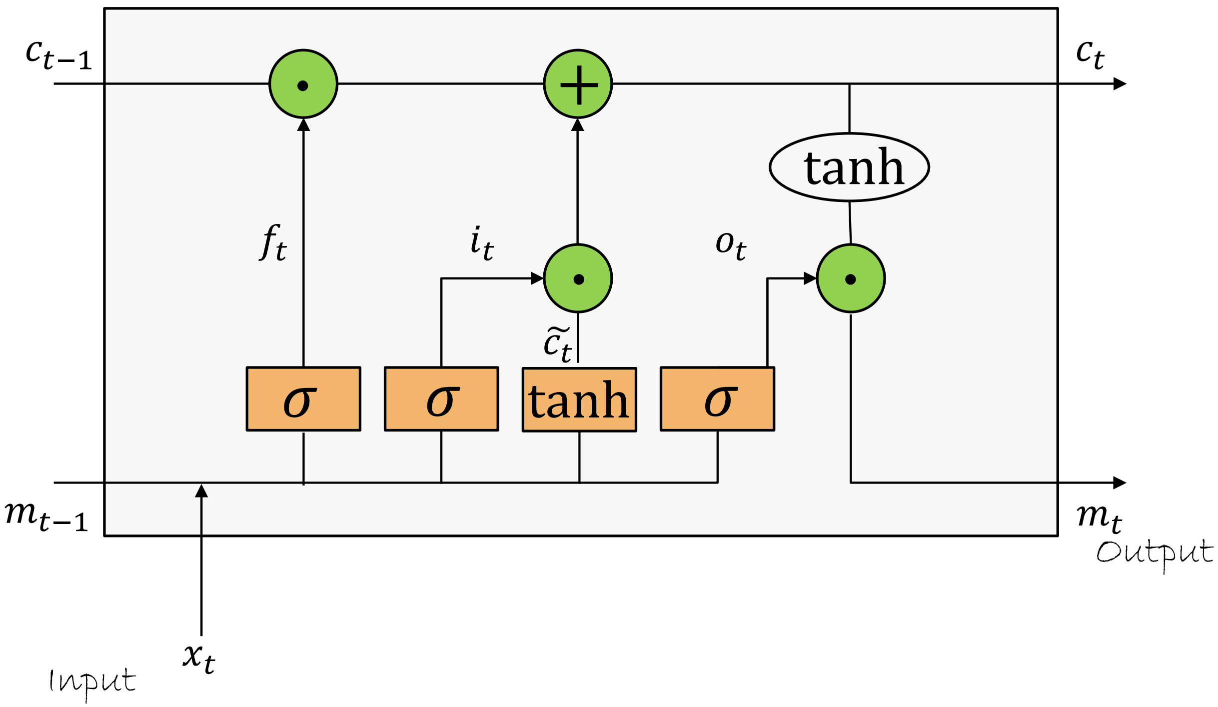
$$f = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c}_t = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \widetilde{c}_t \odot i$$

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



حافظه‌ی کوتاه-مدت طولانی

حالت سلول

CELL STATE

حالت سلول، اطلاعات اساسی را در طول زمان حمل می‌کند.

Cell state line

$$i = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

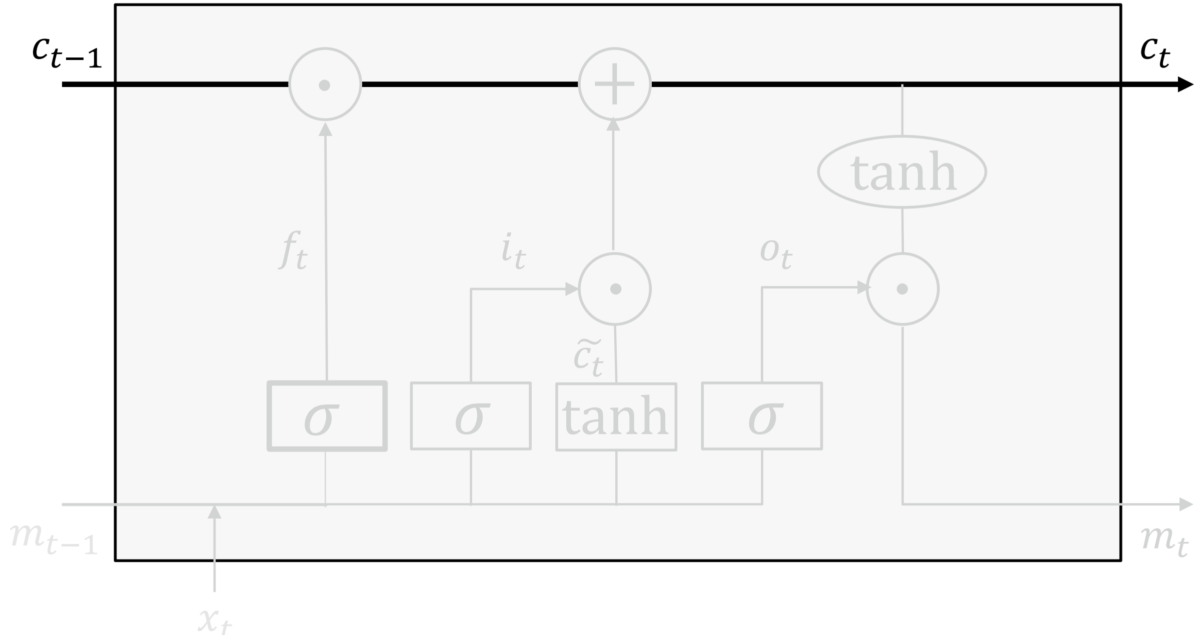
$$f = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c}_t = \tanh\left(x_t U^{(g)} + m_{t-1} W^{(g)}\right)$$

$$c_t = c_{t-1} \odot f + \widetilde{c}_t \odot i$$

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



غیرخطیت‌های LSTM

LSTM NON-LINEARITIES

$\sigma \in (0,1)$ **گیت (دروازه) کنترل (control gate)**: شبیه یک سوئیچ عمل می‌کند.

$\tanh \in (-1,1)$ غیرخطیت بازگشتی

$$i = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

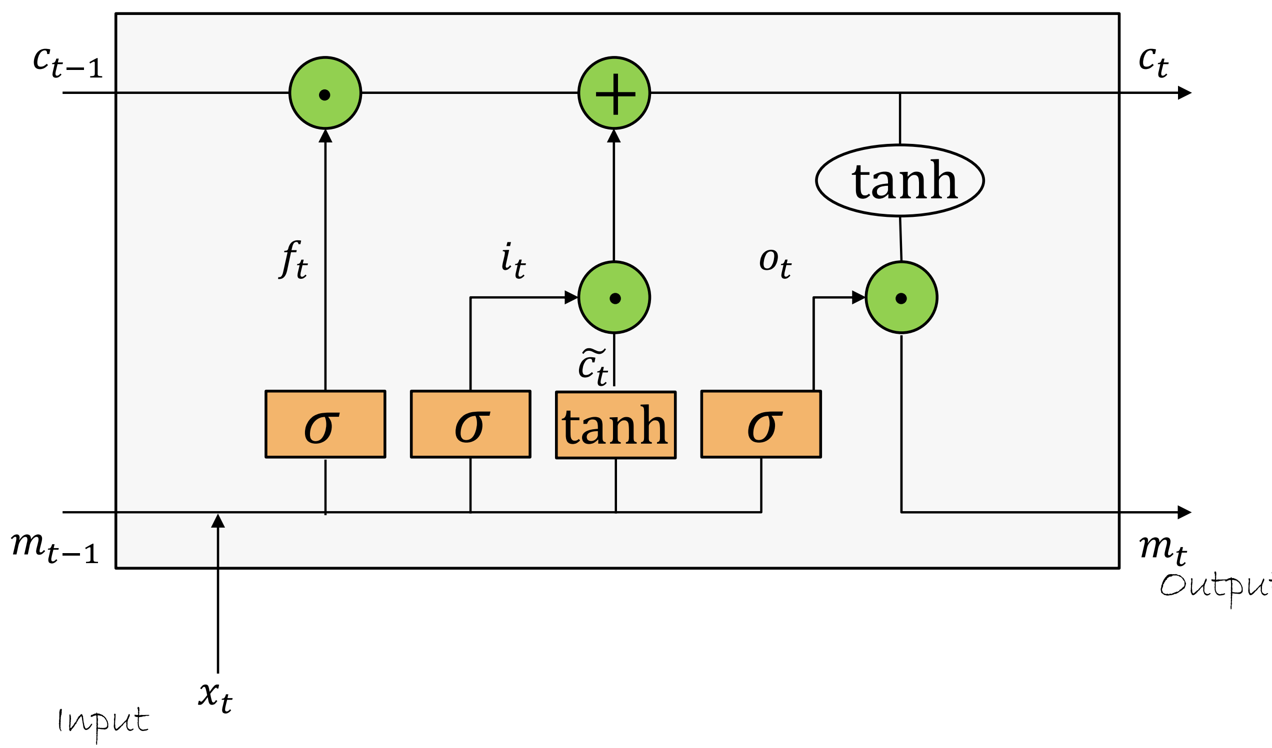
$$f = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\tilde{c}_t = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \tilde{c}_t \odot i$$

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



Ref: https://uvadlc.github.io/

حافظه‌ی کوتاه-مدت طولانی

گام به گام با LSTM: گام ۱

LSTM STEP-BY-STEP: STEP 1

برای مثال: می‌خواهیم یک جمله را مدل کنیم.

باید تصمیم بگیریم برای حافظه‌ی جدید چه چیزی را فراموش کنیم و چه چیزی را به‌خاطر بیاوریم.

سیگموئید = 1 ⇐ به‌یادآوری همه‌چیز
سیگموئید = 0 ⇐ فراموش کردن همه‌چیز

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

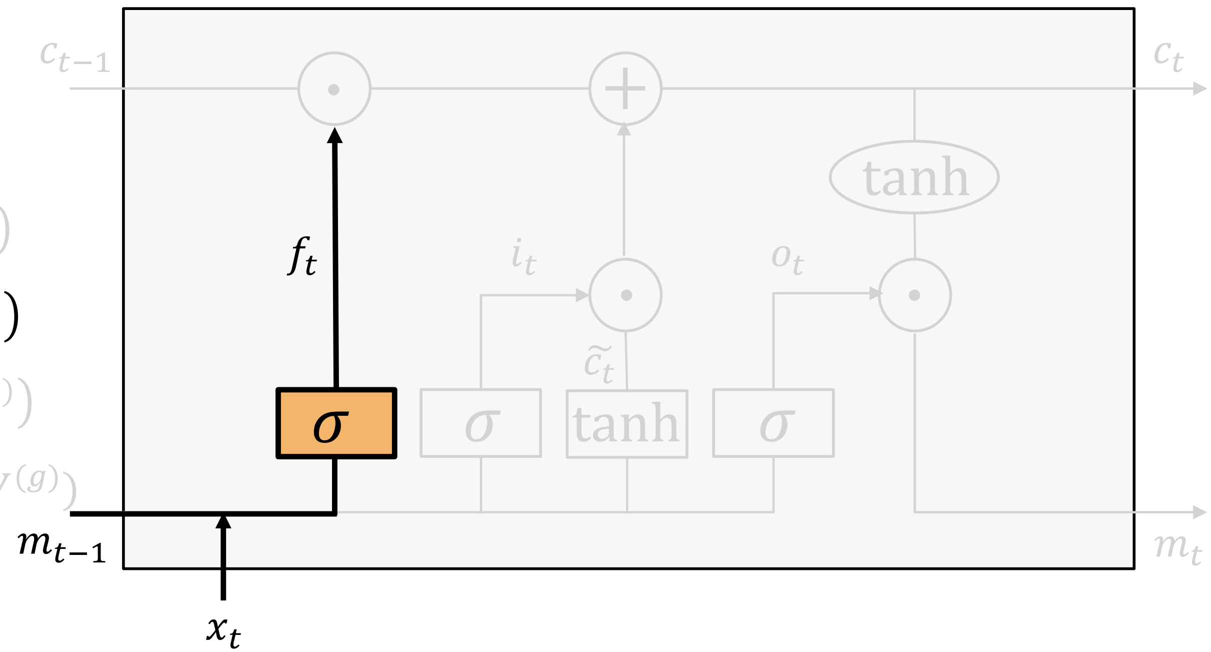
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c}_t = \tanh\left(x_t U^{(g)} + m_{t-1} W^{(g)}\right)$$

$$c_t = c_{t-1} \odot f + \widetilde{c}_t \odot i$$

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



گام به گام با LSTM: گام ۲

LSTM Step-by-Step: Step 2

باید تصمیم بگیریم چه اطلاعات جدیدی باید به حافظه‌ی جدید اضافه شود.

- ورودی i_t را مدوله می‌کنیم.
- حافظه‌های کاندیدای $\widetilde{c}_t$ را تولید می‌کنیم.

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

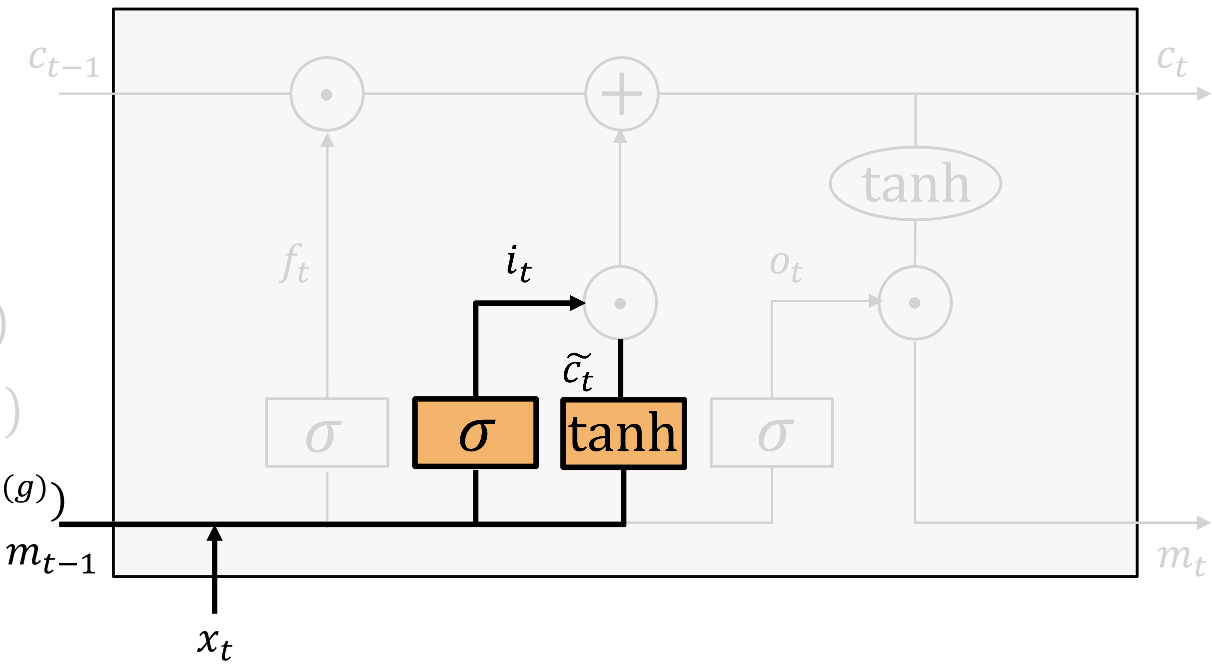
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c}_t = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \widetilde{c}_t \odot i$$

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



Ref: https://uvadlc.github.io/

گام به گام با LSTM: گام ۳

LSTM Step-by-Step: Step 3

حالت فعلی سلول c_t را محاسبه و به‌هنگام می‌کنیم، بر اساس:

- حالت قبلی سلول
- آنچه تصمیم داریم فراموش کنیم
- آنچه به‌عنوان ورودی مجاز می‌دانیم
- حافظه‌های کاندیدا

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

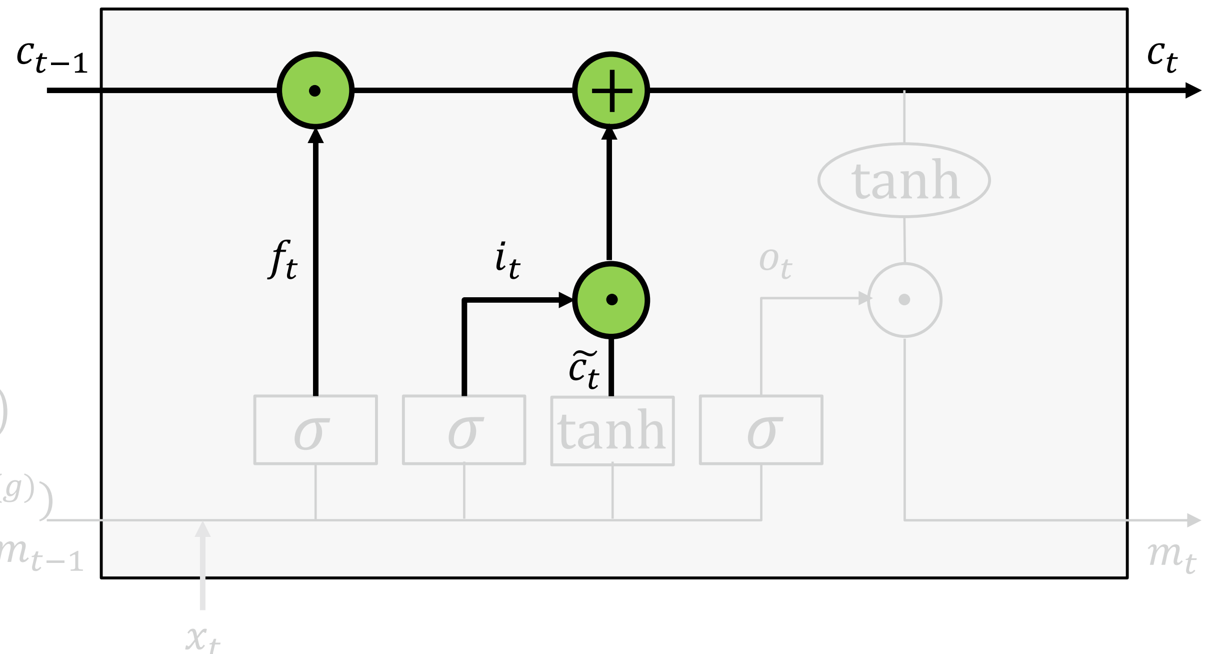
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c}_t = \tanh\left(x_t U^{(g)} + m_{t-1} W^{(g)}\right)$$

$$c_t = c_{t-1} \odot f + \widetilde{c}_t \odot i$$

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



گام به گام با LSTM: گام ۴

LSTM Step-by-Step: Step 4

خروجی را مدوله می‌کنیم؛ حافظه‌ی جدید را تولید می‌کنیم.

اگر حالت سلول حاوی چیز مربوطی است
1 = سیگموئید ⇐

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

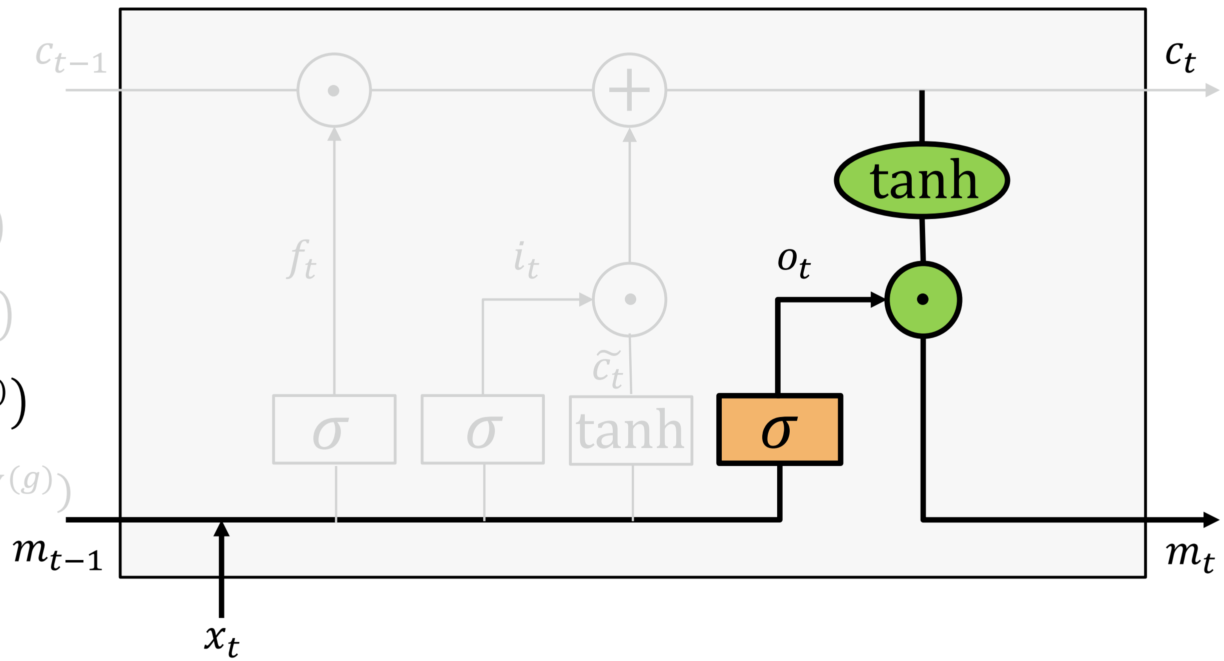
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c_t} = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

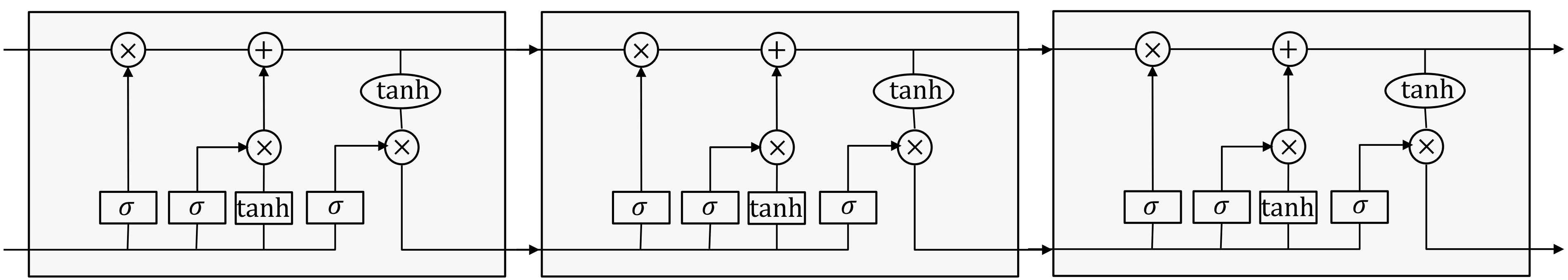
$$c_t = c_{t-1} \odot f + \widetilde{c_t} \odot i$$

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



LSTM Unrolled Network

به‌لحاظ ماکروسکوپی، بسیار شبیه به شبکه‌های عصبی بازگشتی استاندارد است؛
اما موتور آن کمی متفاوت است (پیچیده‌تر)
[زیرا گیت‌های LSTMهای آن وابستگی‌های کوتاه مدت و بلند را تسخیر می‌کند.]



Ref: https://uvadlc.github.io/

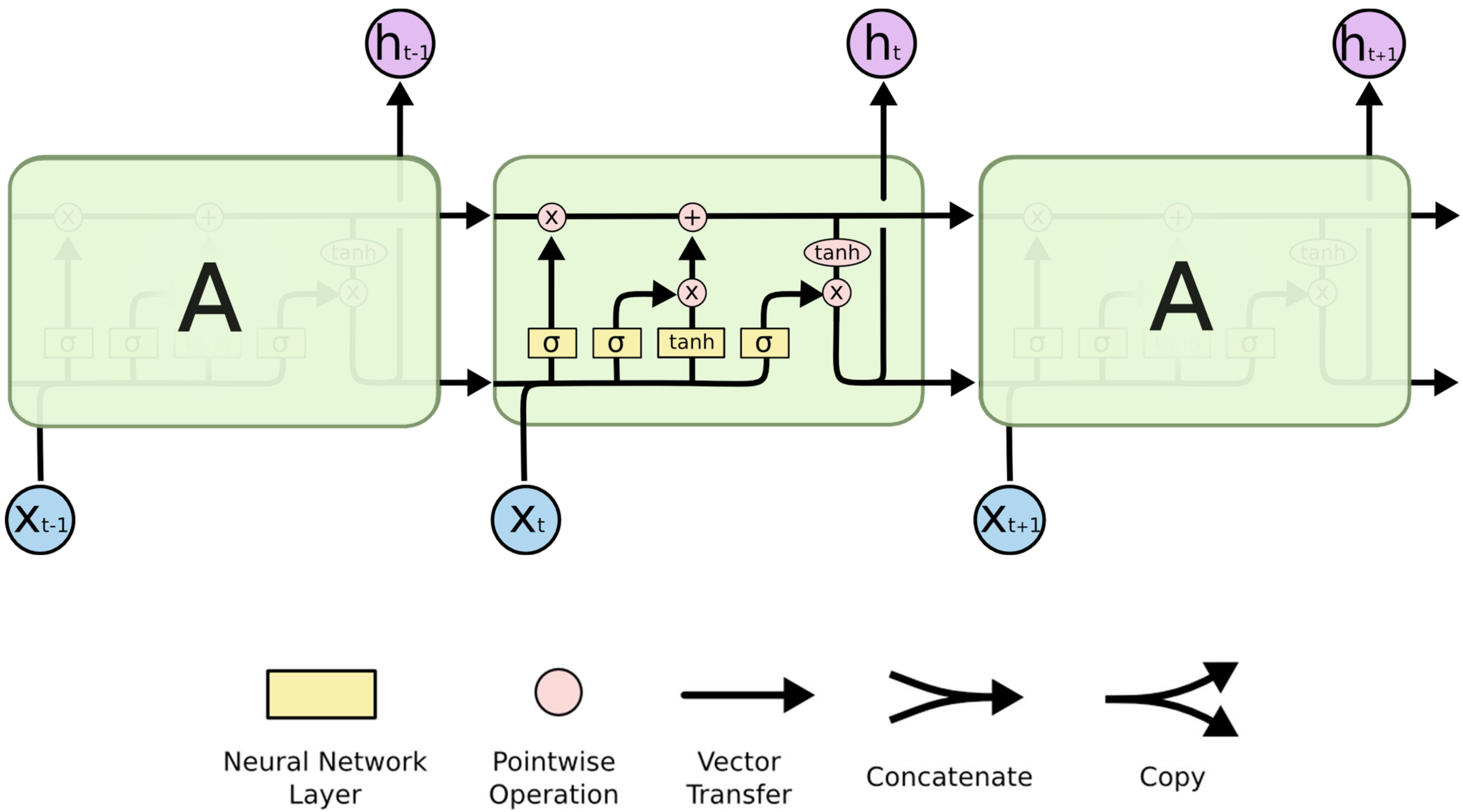
شبکه‌های عصبی بازگشتی

۶

دیگر انواع شبکه‌های عصبی بازگشتی

حافظه‌ی کوتاه-مدت طولانی

LSTMs

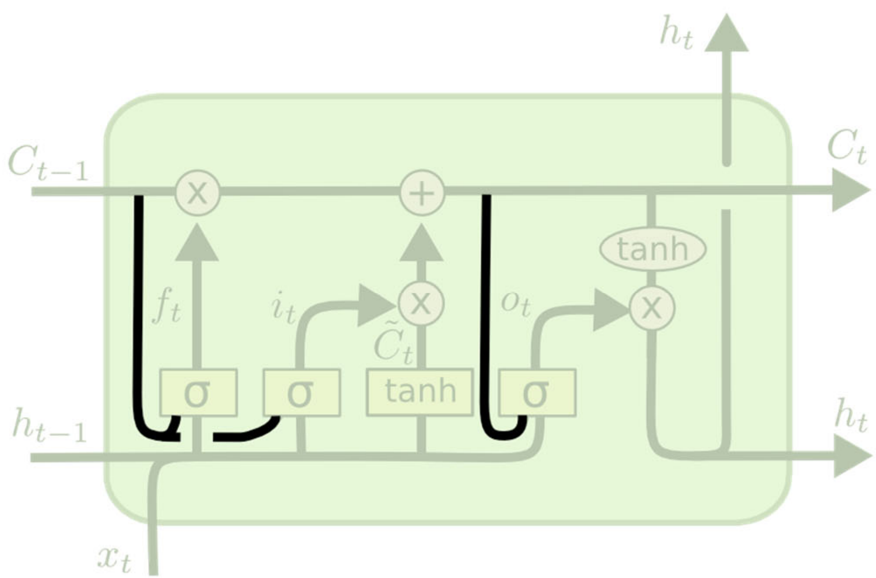


Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

تغییر شماره‌ی ۱ : اتصالات روزنه‌ای

LSTM Variants #1: Peephole Connections



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] + b_o\right)$$

اجازه دادن به گیت‌ها تا بتوانند حالت/ حافظه‌ی سلول را ببینند.

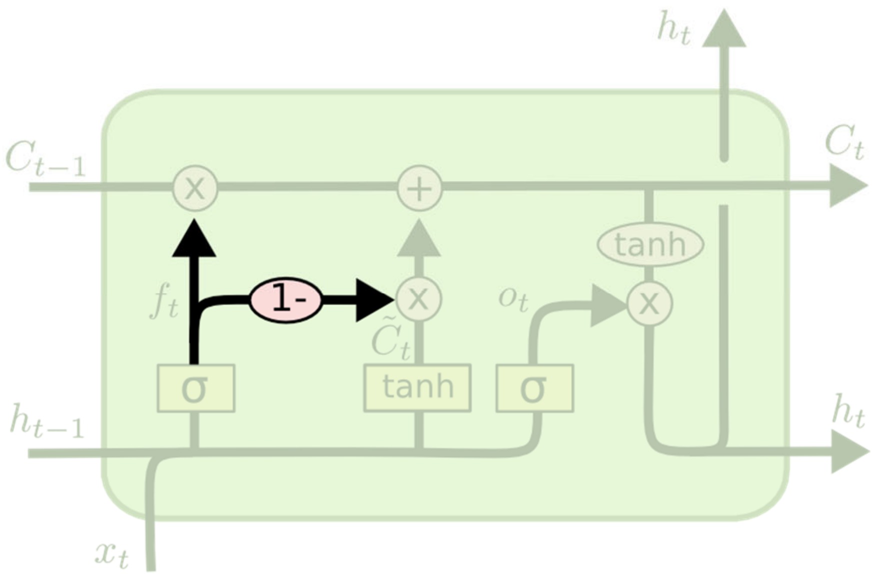
Let gates see the cell state / memory.

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

تغییر شماره‌ی ۲ : گیت‌های جفت‌شده

LSTM Variants #2: Coupled Gates



$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

اطلاعات جدید در صورتی به‌خاطر سپرده می‌شوند که اطلاعات قدیمی فراموش شوند

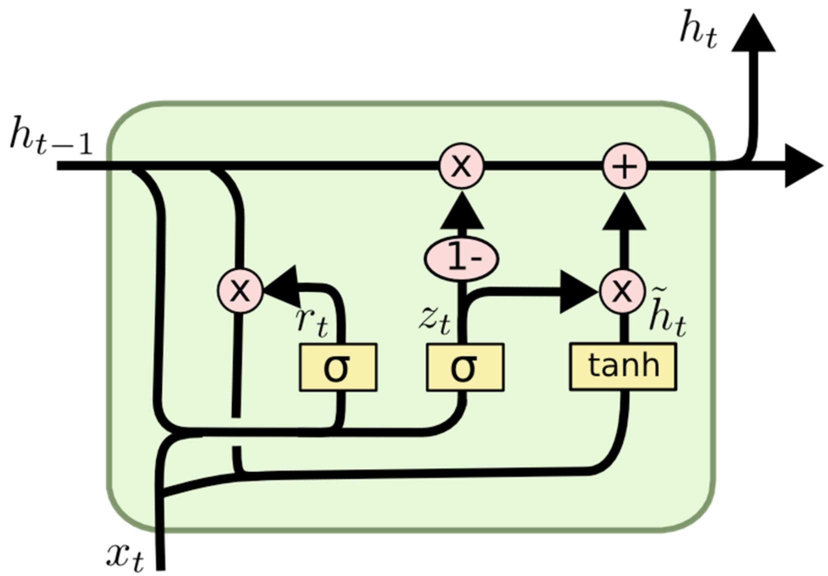
Only memorize new if forgetting old

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

حافظه‌ی کوتاه-مدت طولانی

تغییر شماره‌ی ۳ : واحدهای بازگشتی دروازه‌گذاری شده

LSTM Variants #3: Gated Recurrent Units



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

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

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

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

تغییرات:

حافظه‌ی صریحی وجود ندارد؛ حافظه = خروجی پنهان

z = اطلاعات جدید را به خاطر بسپار و اطلاعات قدیمی را فراموش کن

Changes:

No explicit memory; memory = hidden output

Z = memorize new and forget old

Ref: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

دیگر انواع شبکه‌های عصبی بازگشتی

Other RNN Variants

GRU [*Learning phrase representations using rnn encoder-decoder for statistical machine translation*, Cho et al. 2014]

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

[*LSTM: A Search Space Odyssey*, Greff et al., 2015]

[*An Empirical Exploration of Recurrent Network Architectures*, Jozefowicz et al., 2015]

MUT1:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}h_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}\tanh(h_t) + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

Ref: http://cs231n.stanford.edu/

خلاصه

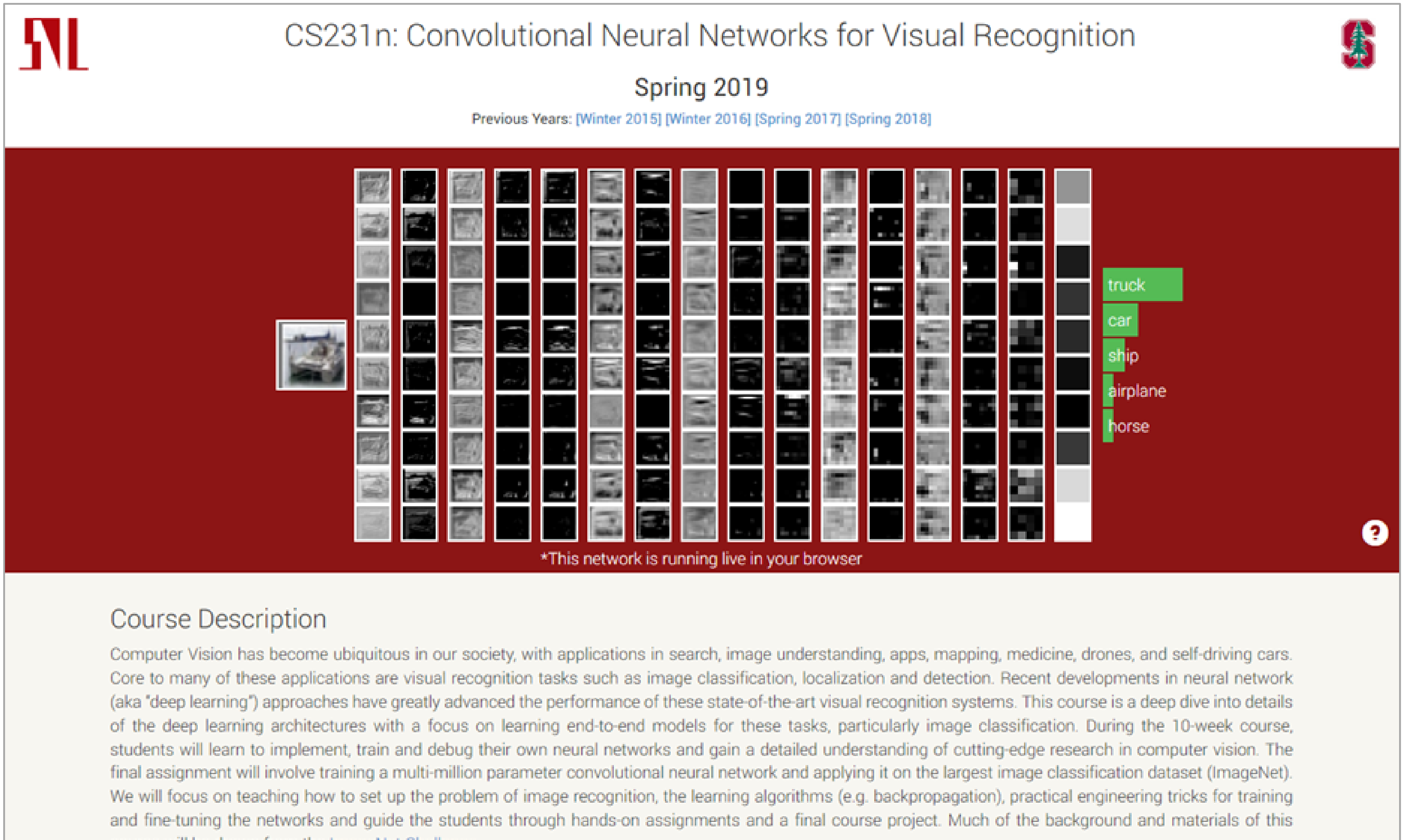
- RNNs allow a lot of flexibility in architecture design.
- Vanilla RNNs are simple but don't work very well.
- Common to use LSTM or GRU:
  their additive interactions improve gradient flow.
- Backward flow of gradients in RNN can explode or vanish.
  - Exploding is controlled with gradient clipping.
  - Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed.

شبکه‌های عصبی بازگشتی

۷

منابع

منبع اصلی



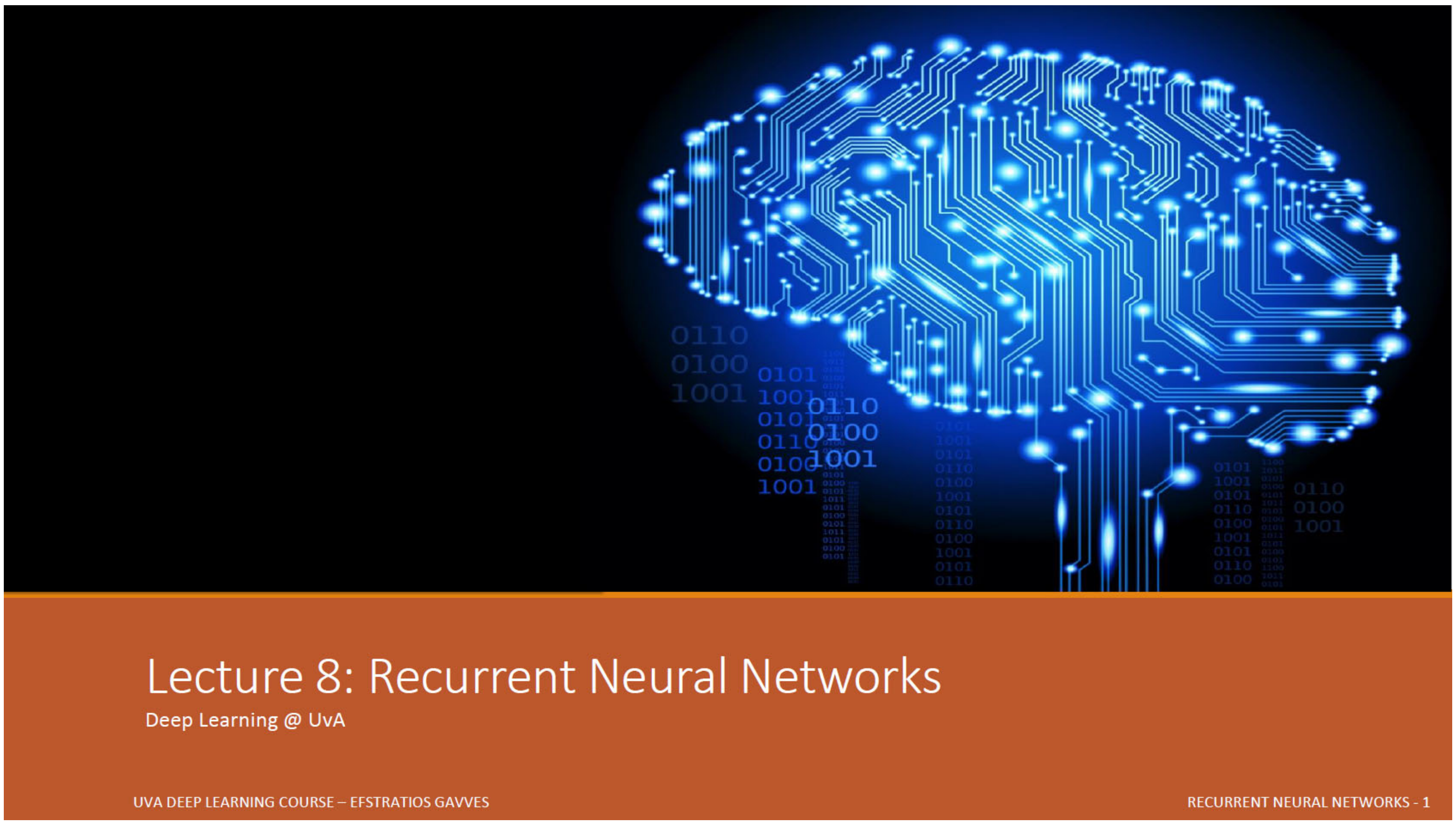
Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network (aka "deep learning") approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of the deep learning architectures with a focus on learning end-to-end models for these tasks, particularly image classification. During the 10-week course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. The final assignment will involve training a multi-million parameter convolutional neural network and applying it on the largest image classification dataset (ImageNet). We will focus on teaching how to set up the problem of image recognition, the learning algorithms (e.g. backpropagation), practical engineering tricks for training and fine-tuning the networks and guide the students through hands-on assignments and a final course project. Much of the background and materials of this course will be drawn from the ImageNet Challenge.

http://cs231n.stanford.edu

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

منبع کمکی



https://uvadlc.github.io/

منبع کمکی

colah's blog   Blog   About   Contact

Understanding LSTM Networks

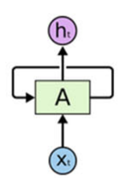
Posted on August 27, 2015

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

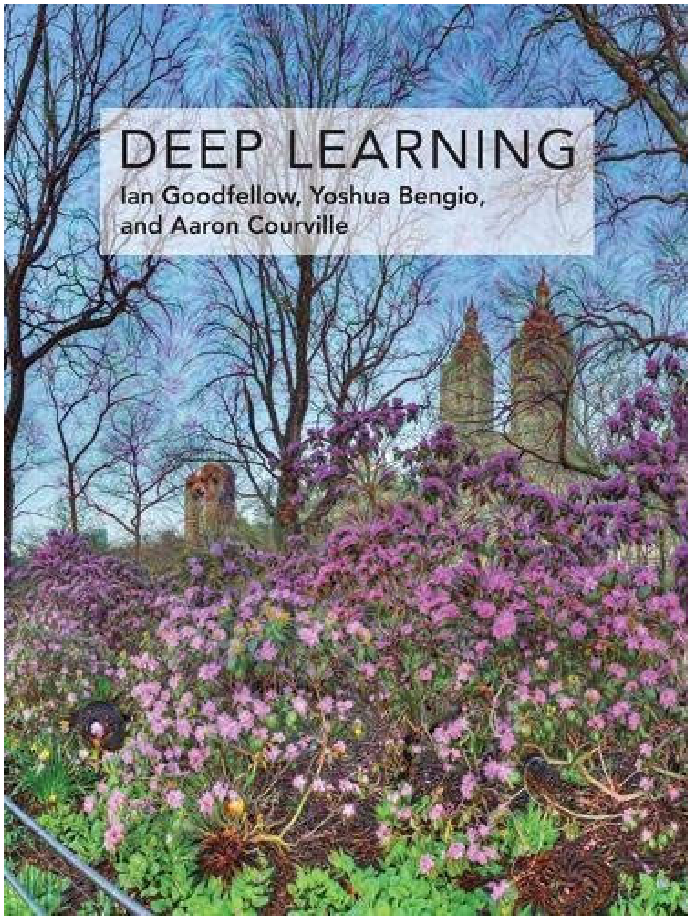
Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.



Recurrent Neural Networks have loops.

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

منبع کمکی



I. Goodfellow, Y. Bengio, A. Courville,
Deep Learning,
MIT Press, 2016.

Chapter 10

Chapter 10

Sequence Modeling: Recurrent and Recursive Nets

Recurrent neural networks or RNNs (Rumelhart *et al.*, 1986a) are a family of neural networks for processing sequential data. Much as a convolutional network is a neural network that is specialized for processing a grid of values $\mathbf{X}$ such as an image, a recurrent neural network is a neural network that is specialized for processing a sequence of values $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}$. Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length.

To go from multi-layer networks to recurrent networks, we need to take advantage of one of the early ideas found in machine learning and statistical models of the 1980s: sharing parameters across different parts of a model. Parameter sharing makes it possible to extend and apply the model to examples of different forms (different lengths, here) and generalize across them. If we had separate parameters for each value of the time index, we could not generalize to sequence lengths not seen during training, nor share statistical strength across different sequence lengths and across different positions in time. Such sharing is particularly important when a specific piece of information can occur at multiple positions within the sequence. For example, consider the two sentences “I went to Nepal in 2009” and “In 2009, I went to Nepal.” If we ask a machine learning model to read each sentence and extract the year in which the narrator went to Nepal, we would like it to recognize the year 2009 as the relevant piece of information, whether it appears in the sixth

373