CS 4803 / 7643: Deep Learning

Topics:

- Recurrent Neural Networks (RNNs)
  - RNN visualizations
  - Image Captioning, Beam Search
  - LSTMs

Dhruv Batra

Georgia Tech

Administrativia

- HW3 Reminder
  - Due: 10/20 11:59pm
  - Theory: Convolutions, Representation Capacity, Double Descent
  - Implementation: Saliency methods (e.g. Grad-CAM) in Python and PyTorch/Captum

- HW2 grades coming soon

Administrativia

- Guest Lecture: Ishan Misra (FAIR)
  - Thurs 10/21
  - Self-Supervised Learning for Vision

http://imisra.github.io/

Administrativia

- Guest Lecture: Michael Auli (FAIR)
  - Tue 10/26
  - Self-Supervised Learning for Speech

https://michaelauli.github.io/

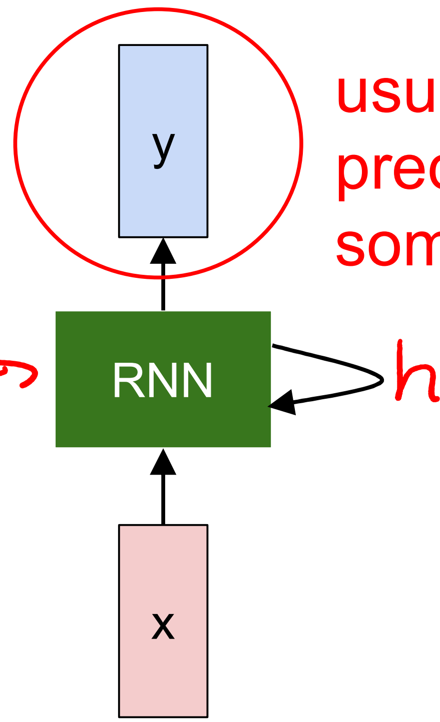
Administrativia

- Guest Lecture: Arjun Majumdar
  - Thurs 10/28
  - Transformers, BERT, ViLBERT

https://arjunmajum.github.io/

Recap from last time

Recurrent Neural Network



usually want to predict a vector at some time steps

"cell"

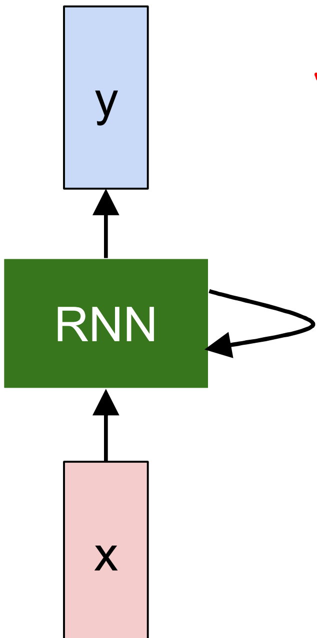
Recurrent unit of computation

cell "hidden" rep. / "state"

$P(y_t \mid x_1 \dots x_t) \approx P(y_t \mid h_t)$

(Vanilla) Recurrent Neural Network

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



$h_t \in \mathbb{R}^{d_2}$

$x \in \mathbb{R}^{d_1}$

"scores" softmax

$$\vec{y}_t = \left(W_{hy} h_t + \vec{b}_y\right)$$

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

$$\mathbb{R}^{512} \ni \vec{h}_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + \vec{b}_h)$$

$$\left(\begin{bmatrix} W_{hh} & W_{xh} \end{bmatrix} \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + \vec{b}_h \right)$$

Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

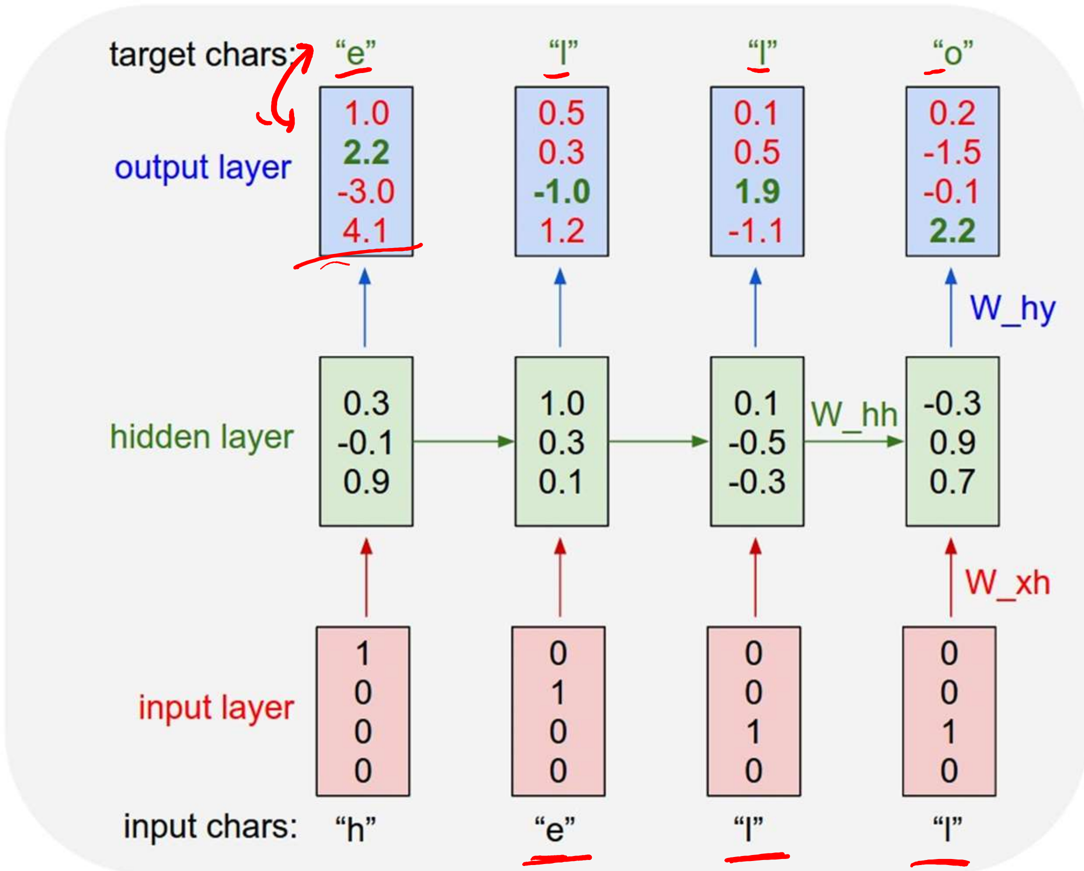
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

$$\min_{\vec{w}} \sum_t -\log P(y_t^{gt} \mid h_t, \vec{w}) = L_1 \quad L_2 \quad L_3 \quad L_4$$

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



$\boxed{x_2 = y_1^{gt}}$ $\quad x_3 = y_2^{gt} \quad - \; - \; - \; -$

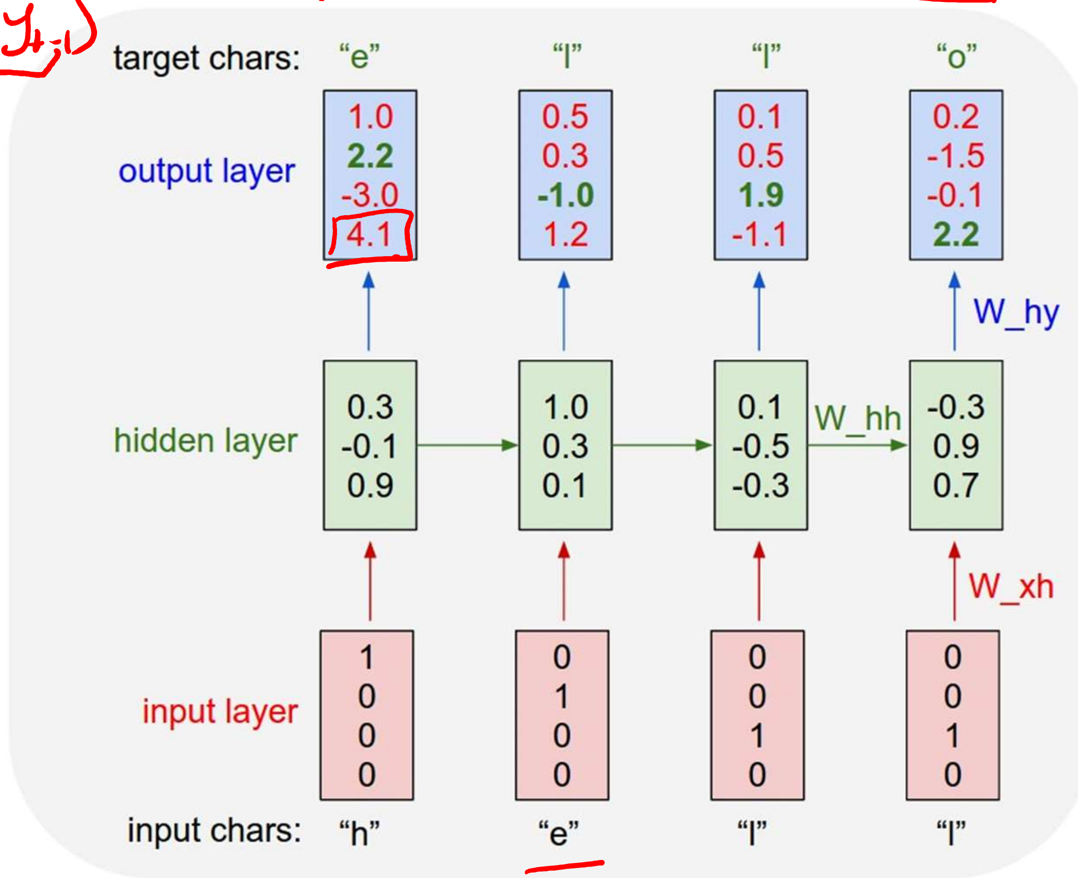
Training Time: $\boxed{\text{MLE}}$ / "Teacher Forcing"

$P(y_1 \ldots y_T) = \prod_t P(y_t \mid y_1 \ldots y_{t-1})$

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
"hello"



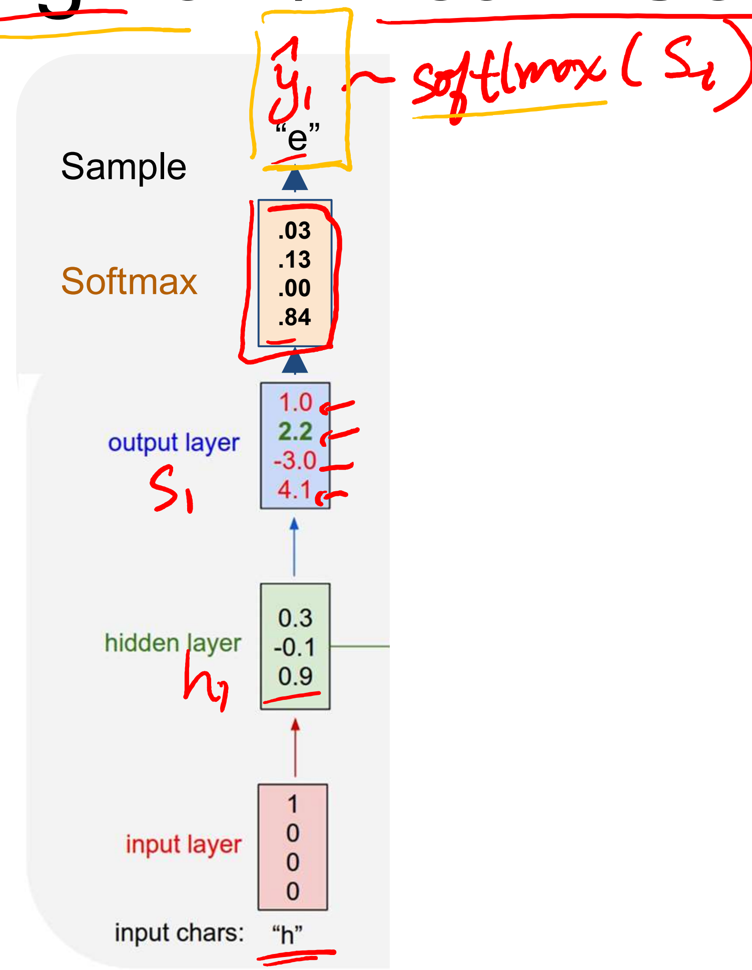
$x_2 = y_1^{96}$

Test Time: Sample / Argmax / Beam Search

Example:
Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

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

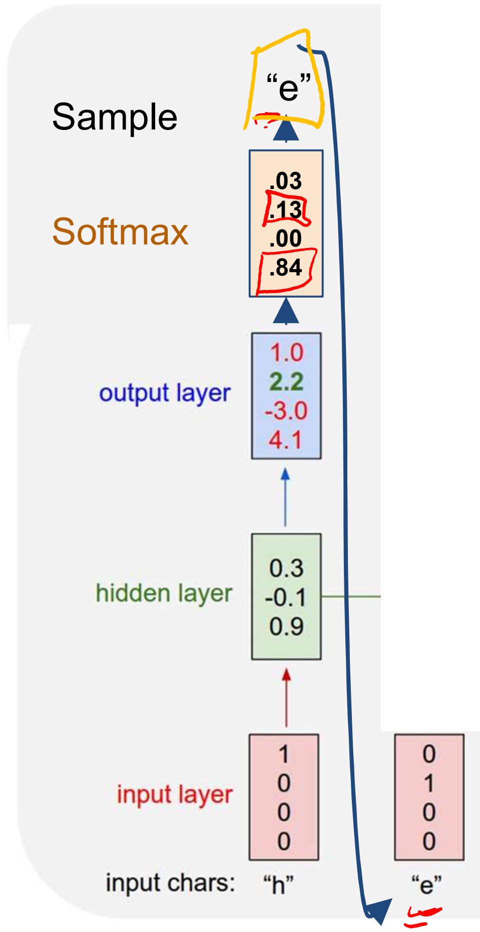


Test Time: Sample / Argmax / Beam Search

Example:
Character-level
Language Model
Sampling

Vocabulary:
[h,e,l,o]

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



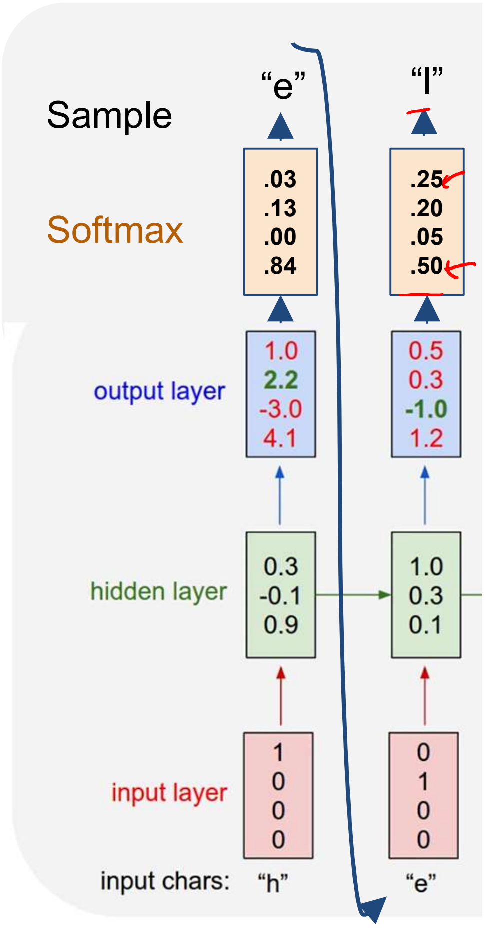
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Test Time: Sample / Argmax / Beam Search

Example:
Character-level
Language Model
Sampling

Vocabulary:
[h,e,l,o]

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

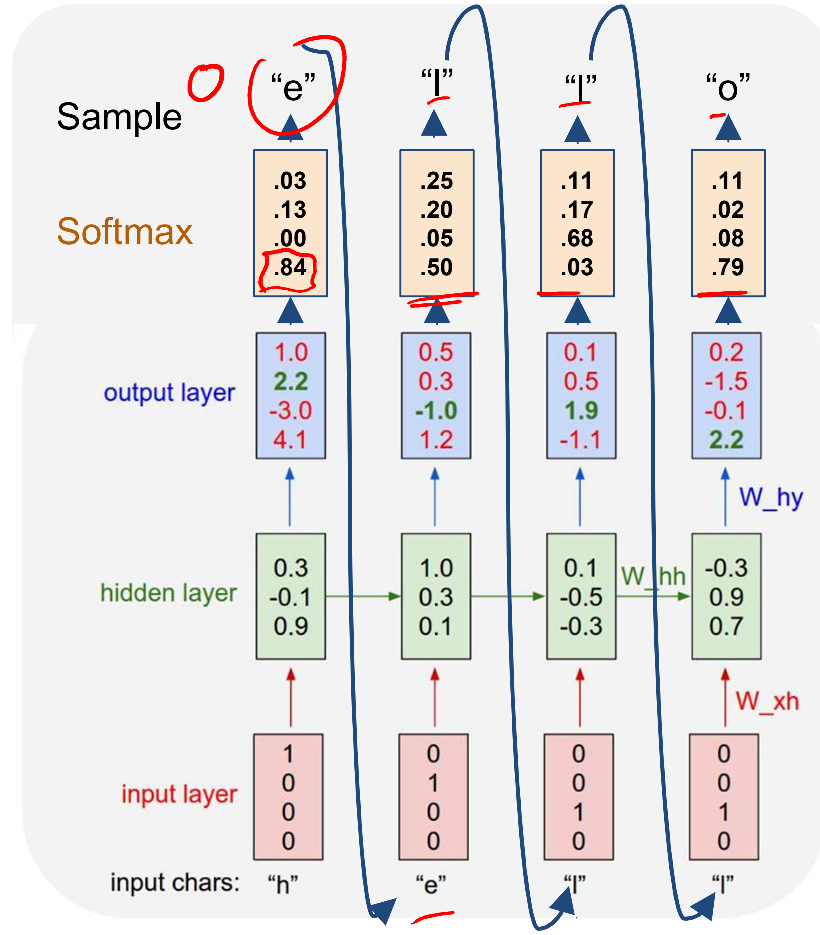


Test Time: Sample / Argmax / Beam Search

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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



h o

Plan for Today

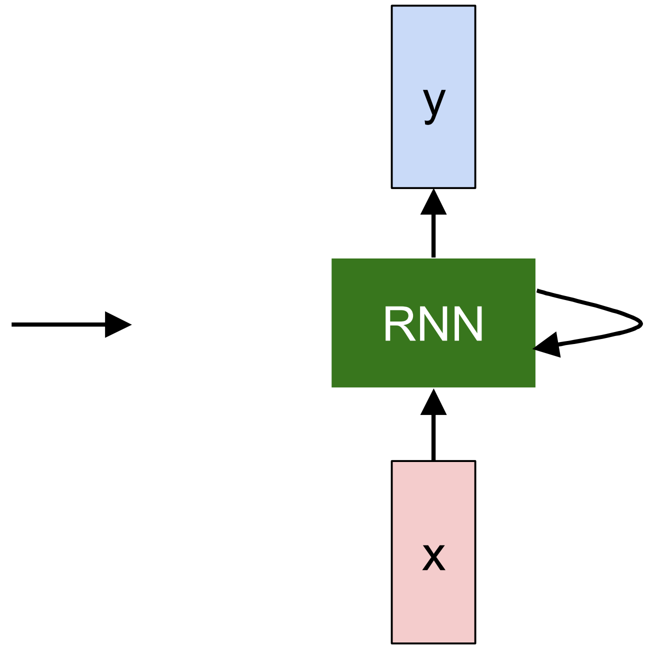
- Recurrent Neural Networks (RNNs)
  - (Finish) Visualization in Character RNNs
  - Inference: Beam Search
    - Example: Image Captioning
  - Multilayer RNNs
  - Problems with gradients in “vanilla” RNNs
  - LSTMs (and other RNN variants)

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.



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

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

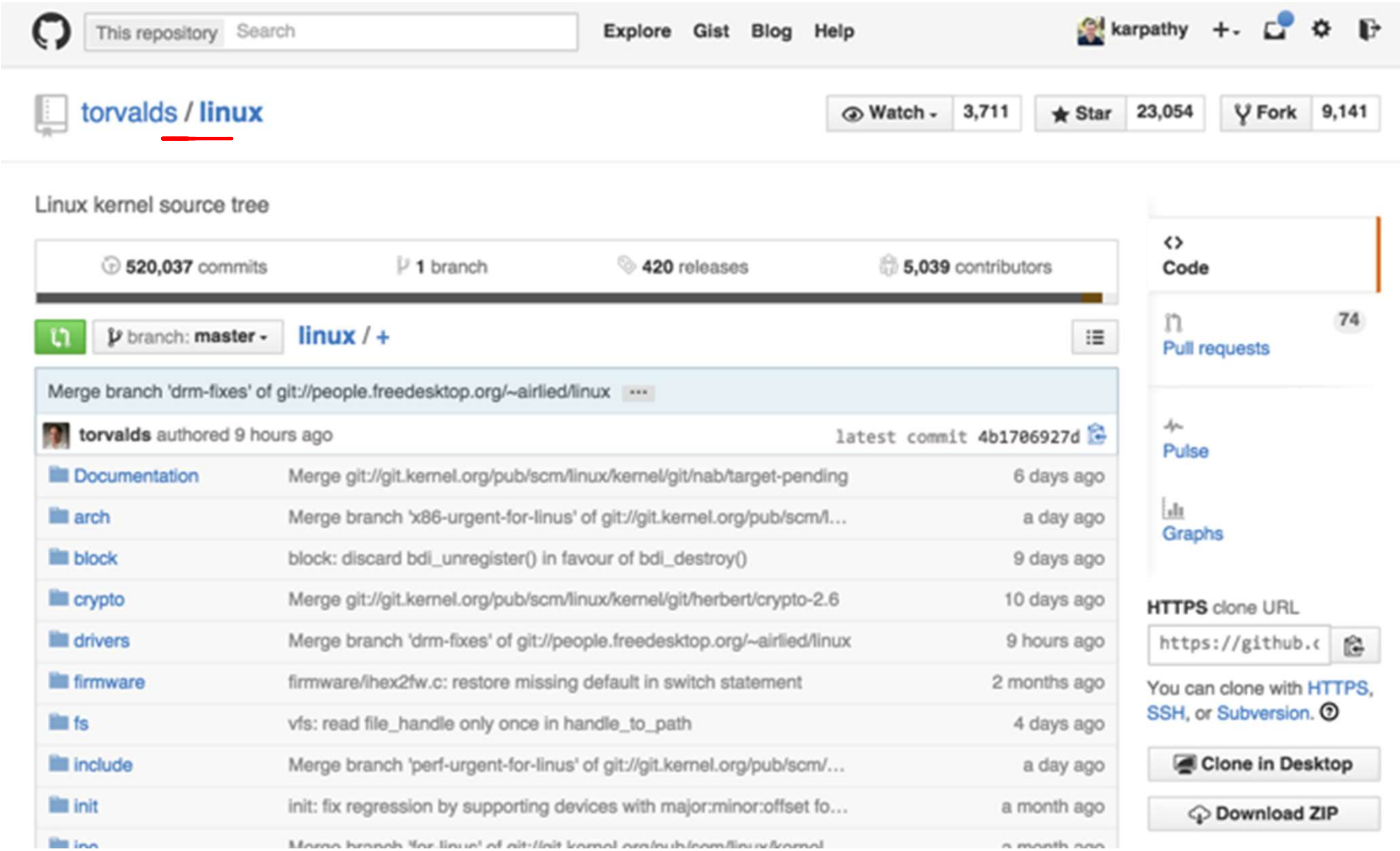
```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

↓ 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.
```



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

```c
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

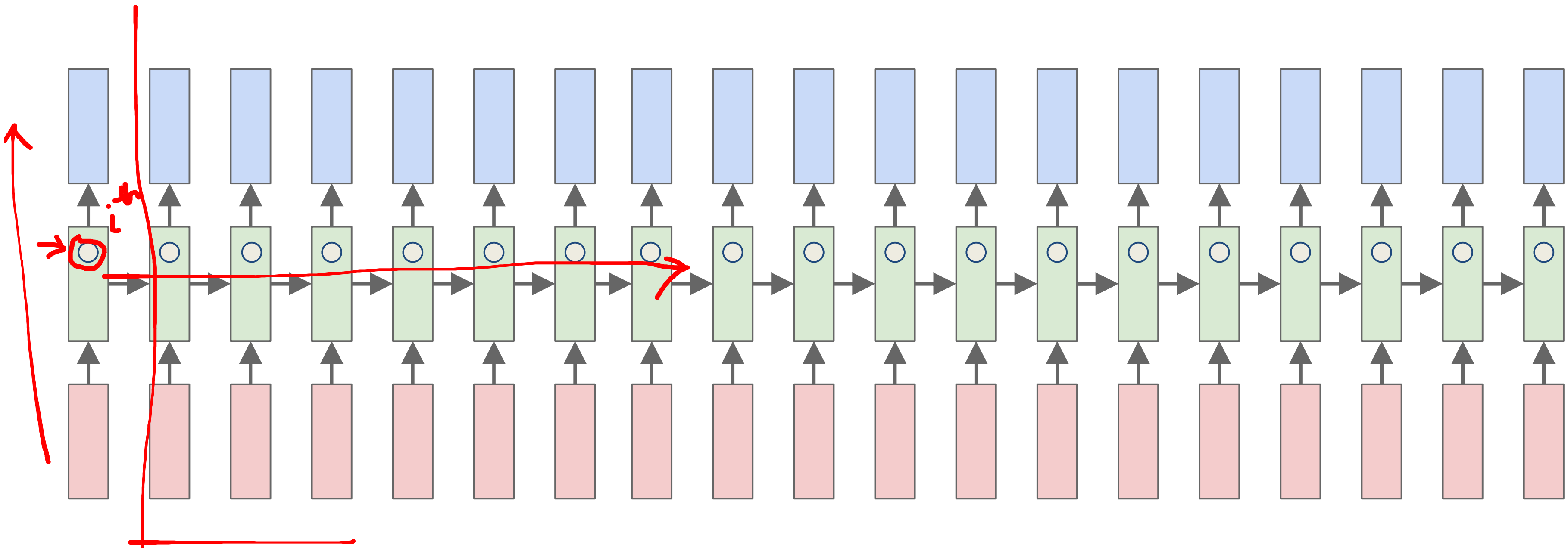
```
/*
 *  Copyright (c) 2006-2010, Intel Mobile Communications.  All rights reserved.
 *
 *   This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 *        This program 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>
```

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

Searching for interpretable cells



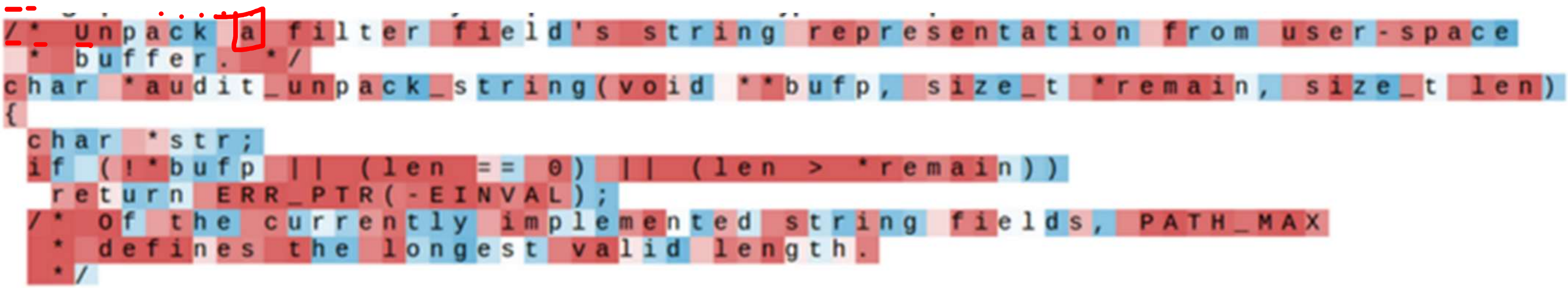
Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

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

h_t

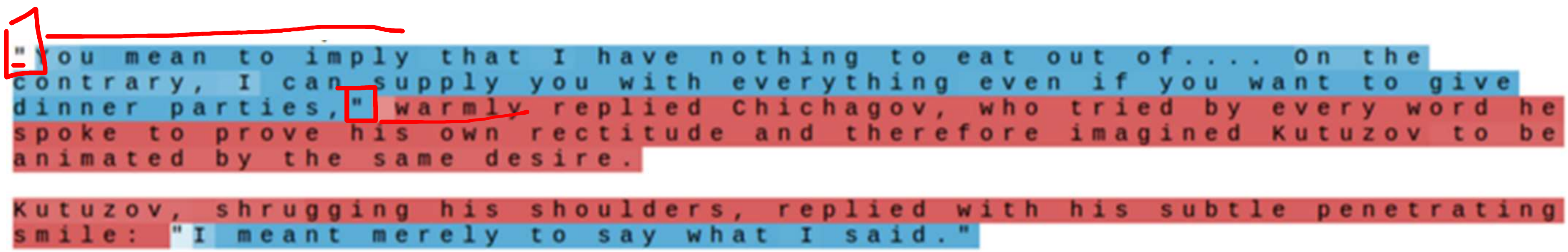
Searching for interpretable cells

$\vec{h}_t$



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

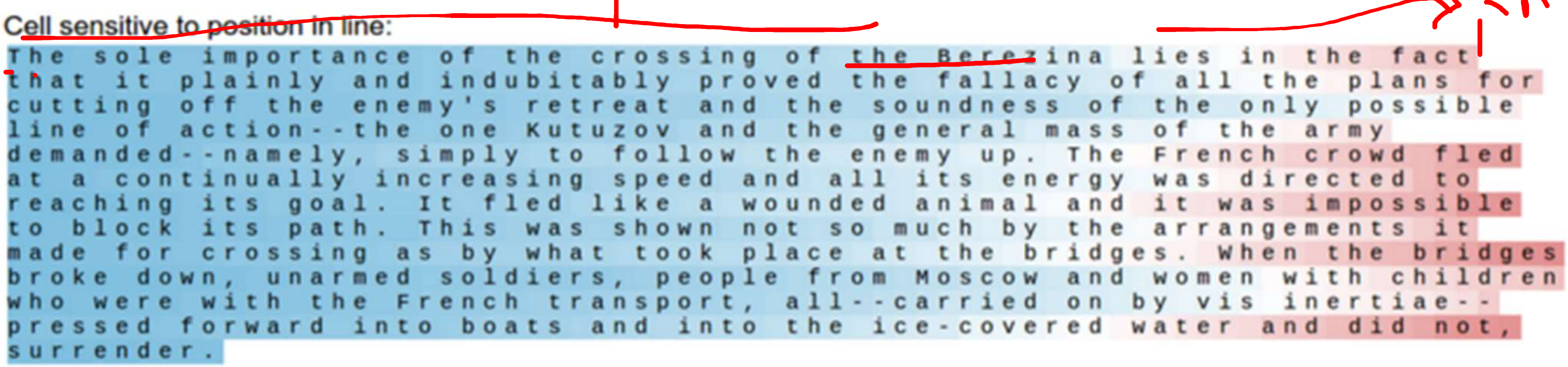
Searching for interpretable cells



quote detection cell

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

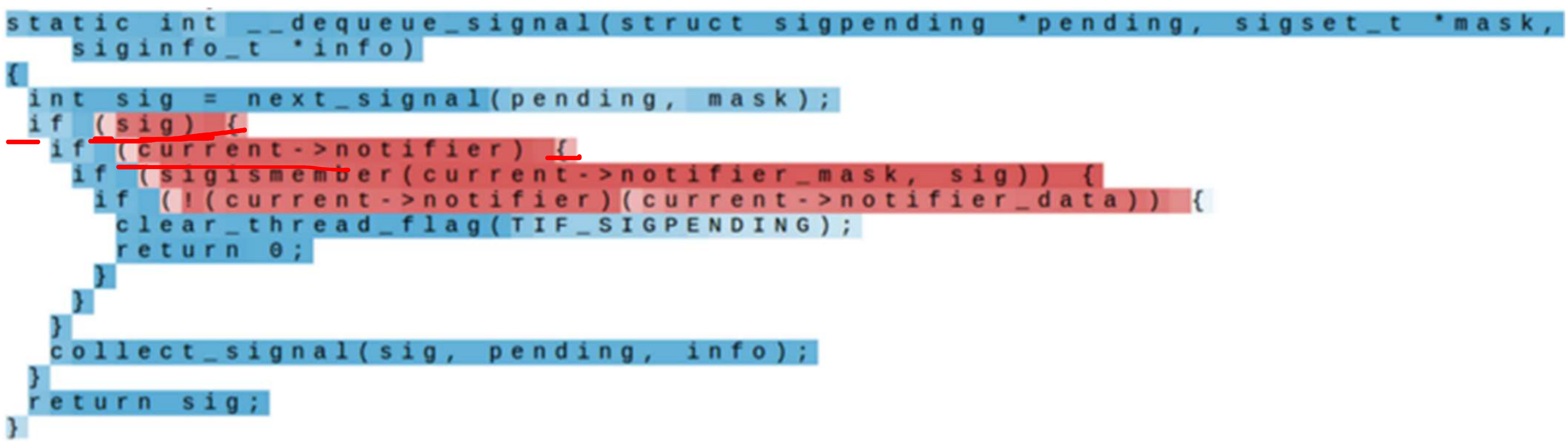
Searching for interpretable cells



line length tracking cell

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

Searching for interpretable cells



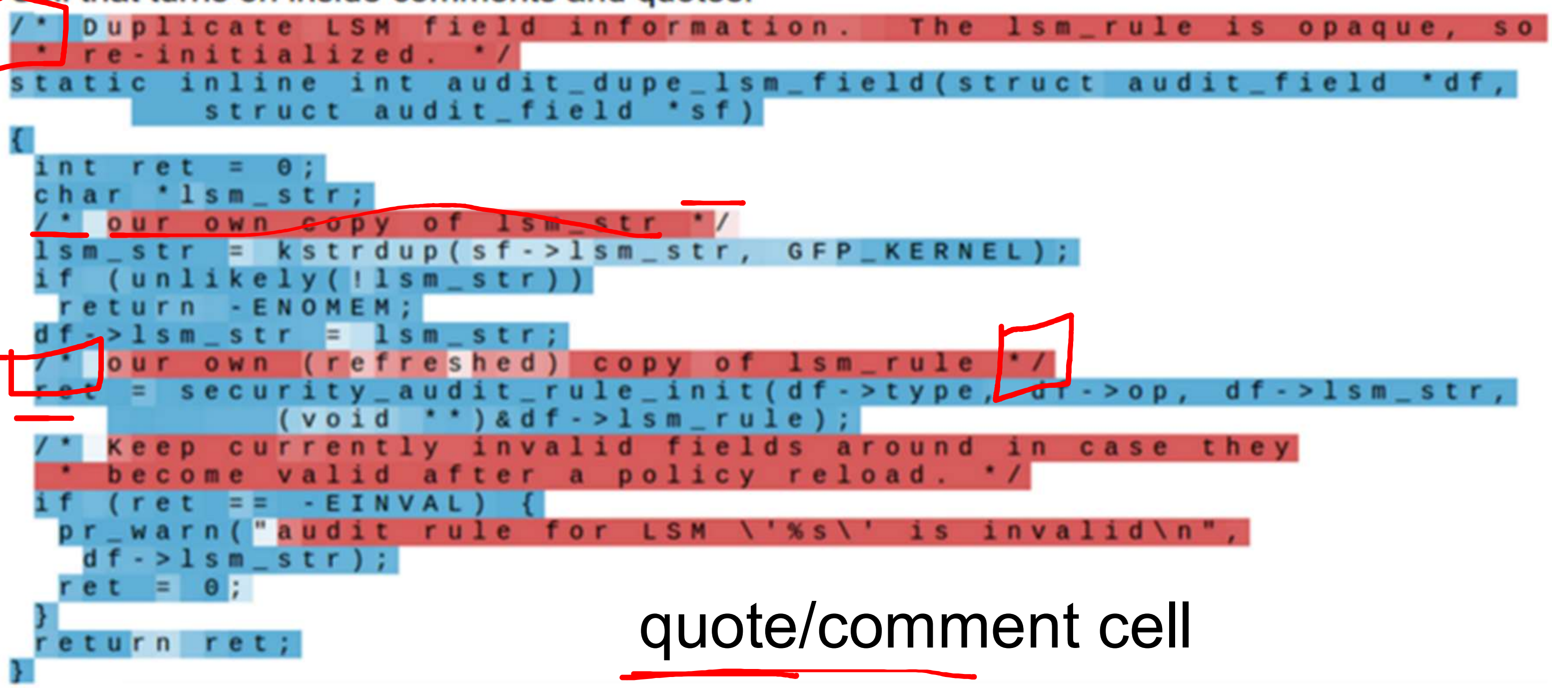
if statement cell

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

//

Searching for interpretable cells

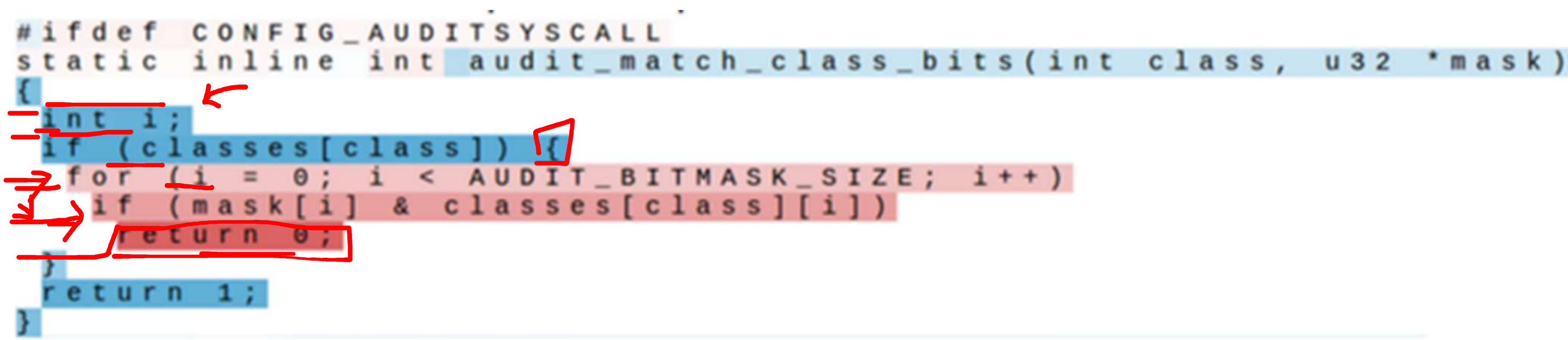
Cell that turns on inside comments and quotes:



quote/comment cell

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

Searching for interpretable cells



code depth cell

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

Plan for Today

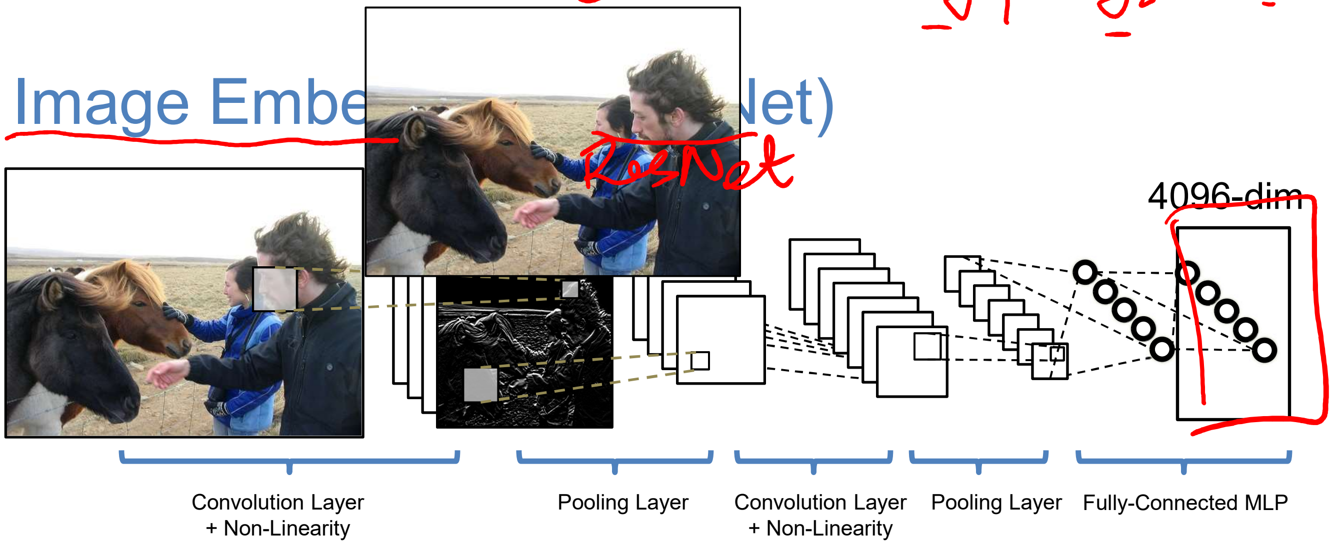
- Recurrent Neural Networks (RNNs)
  - Inference: Beam Search
    - Example: Image Captioning
  - Multilayer RNNs
  - Problems with gradients in "vanilla" RNNs
  - LSTMs (and other RNN variants)

Neural Image Captioning

$x \rightarrow y_1 \; y_2 \; \ldots \; y_T$

Image Embe Net)

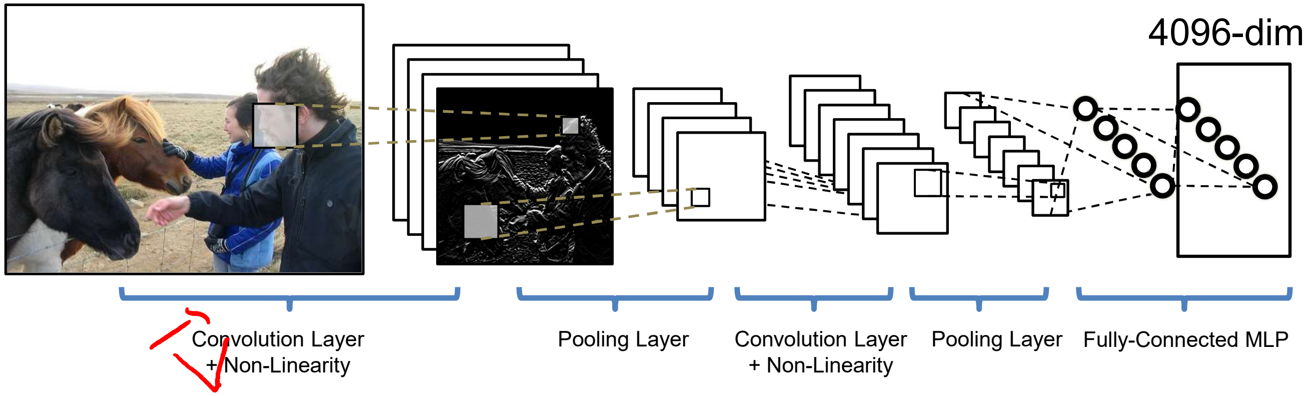
ResNet



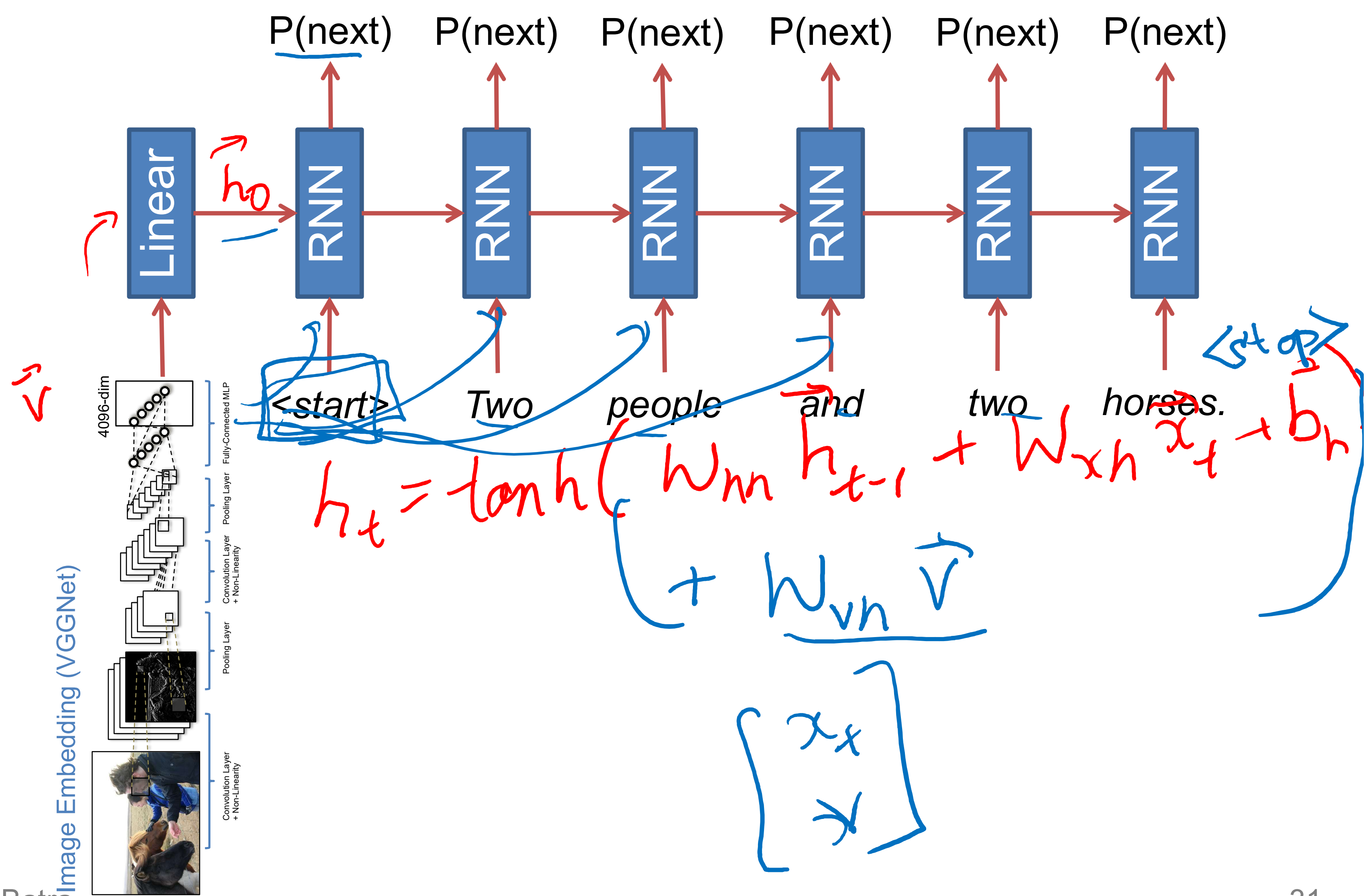
$\vec{v} \in \mathbb{R}^d$ (~~512~~ 4096)

Neural Image Captioning

Image Embedding (VGGNet)



Neural Image Captioning



$$h_t = \tanh\left(W_{hh} h_{t-1} + W_{xh} x_t + b_h + \underline{W_{vh} \vec{v}}\right)$$

$$\begin{bmatrix} x_t \\ v \end{bmatrix}$$

Beam Search Demo

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

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

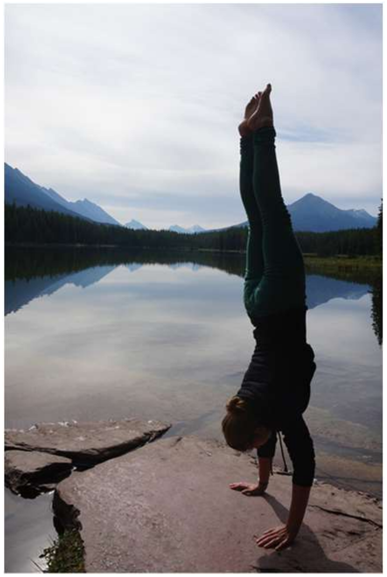
A man riding a dirt bike on a dirt track

Image Captioning: Failure Cases

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

A woman is holding a cat in her hand

A person holding a computer mouse on a desk

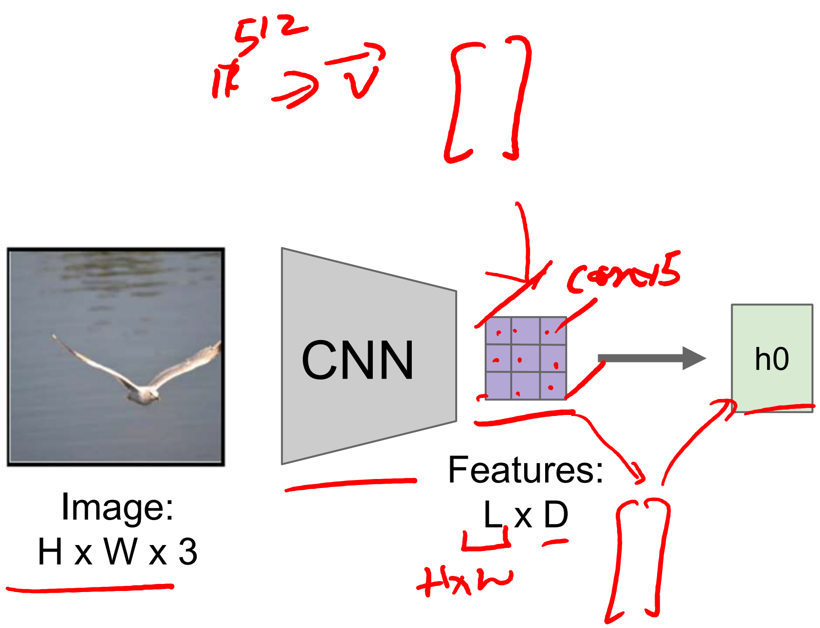


A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

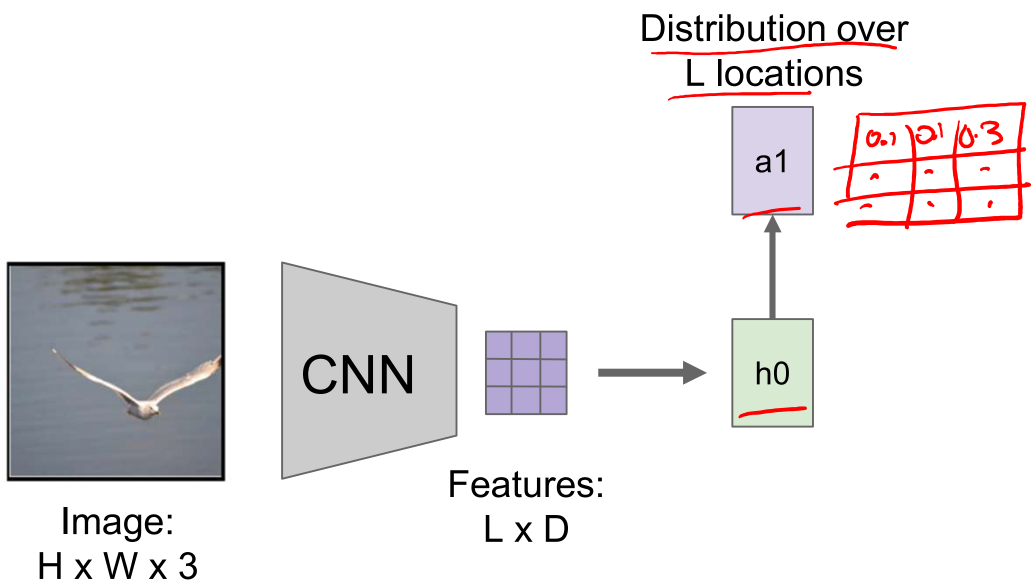
A man in a baseball uniform throwing a ball

Image Captioning with Attention



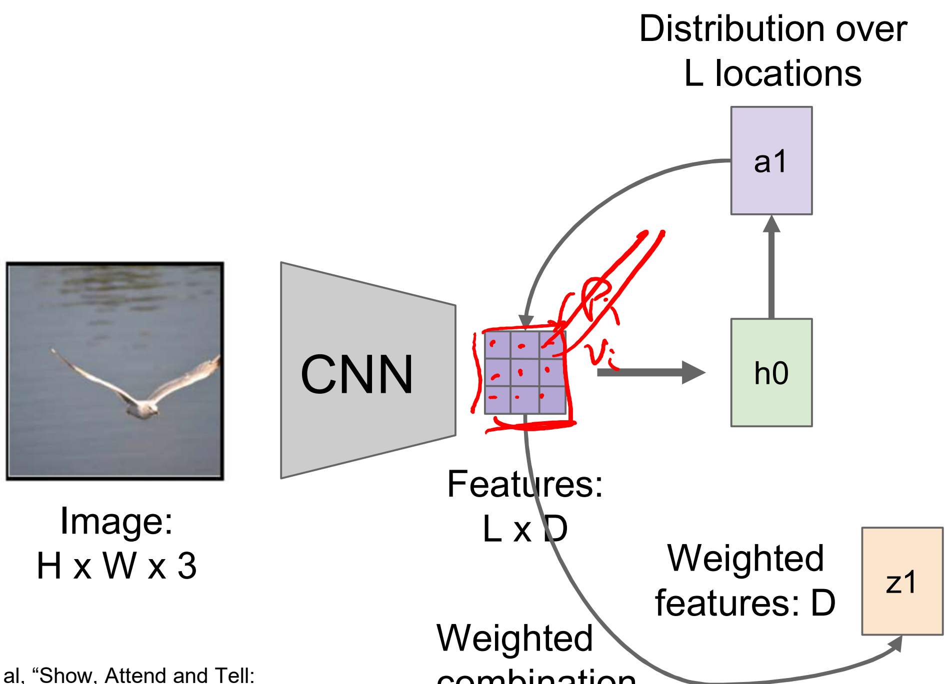
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



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

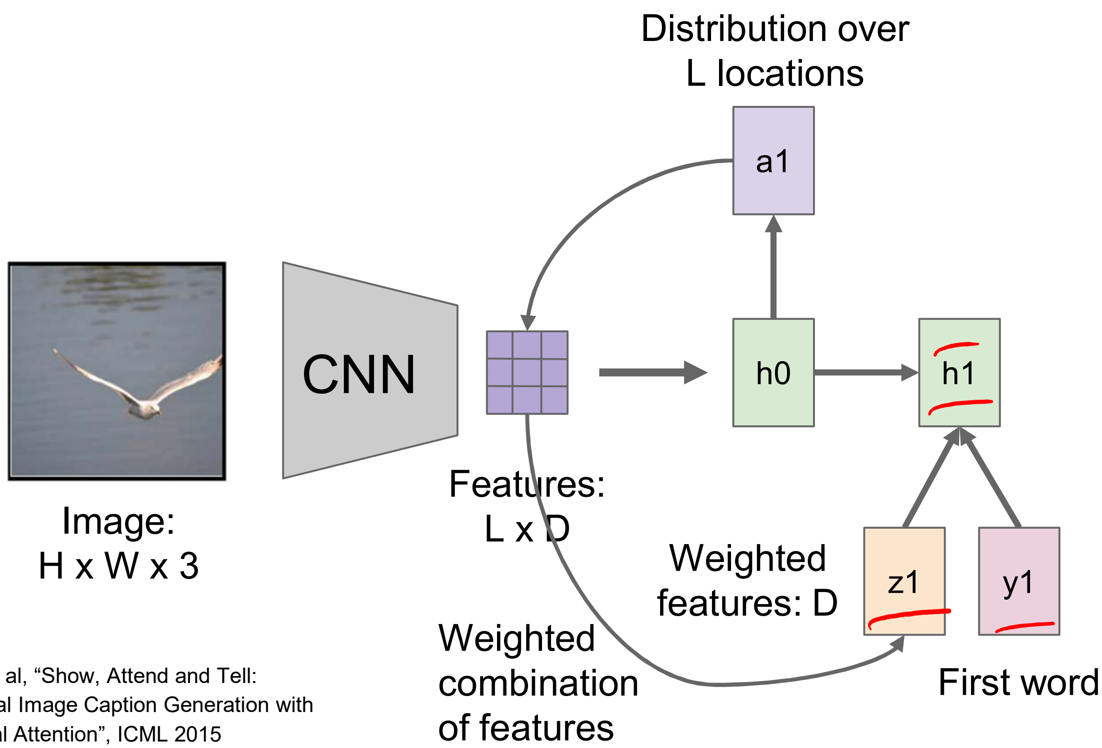
Image Captioning with Attention



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

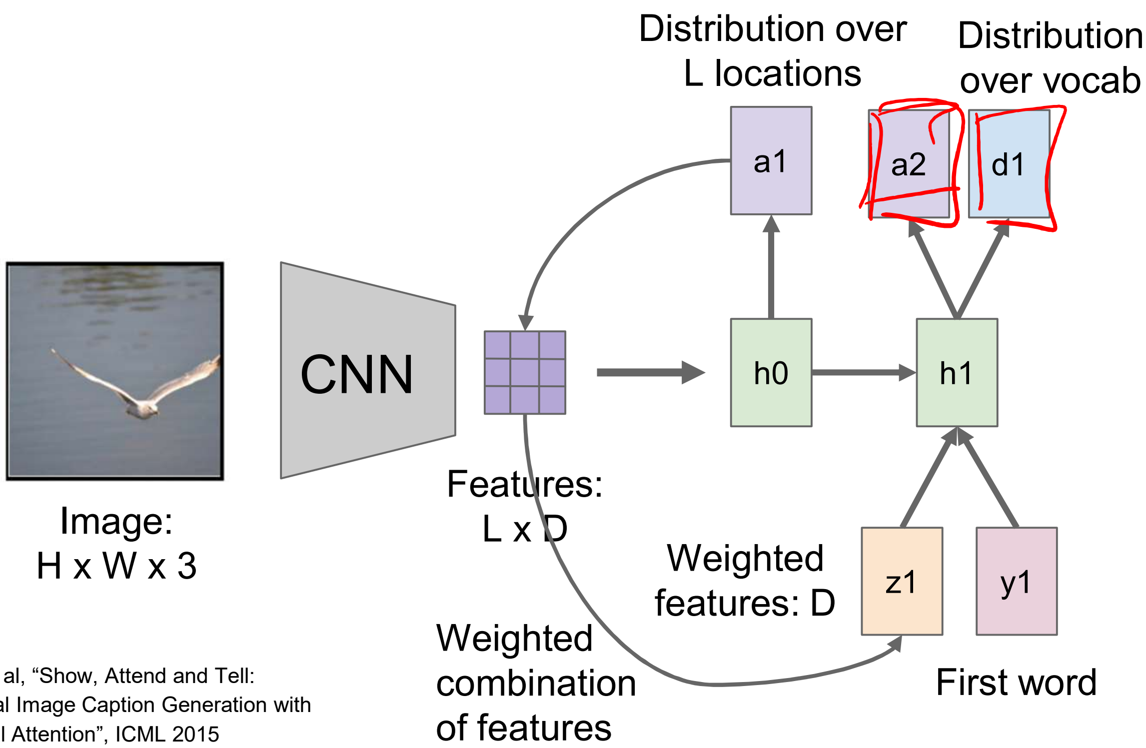
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



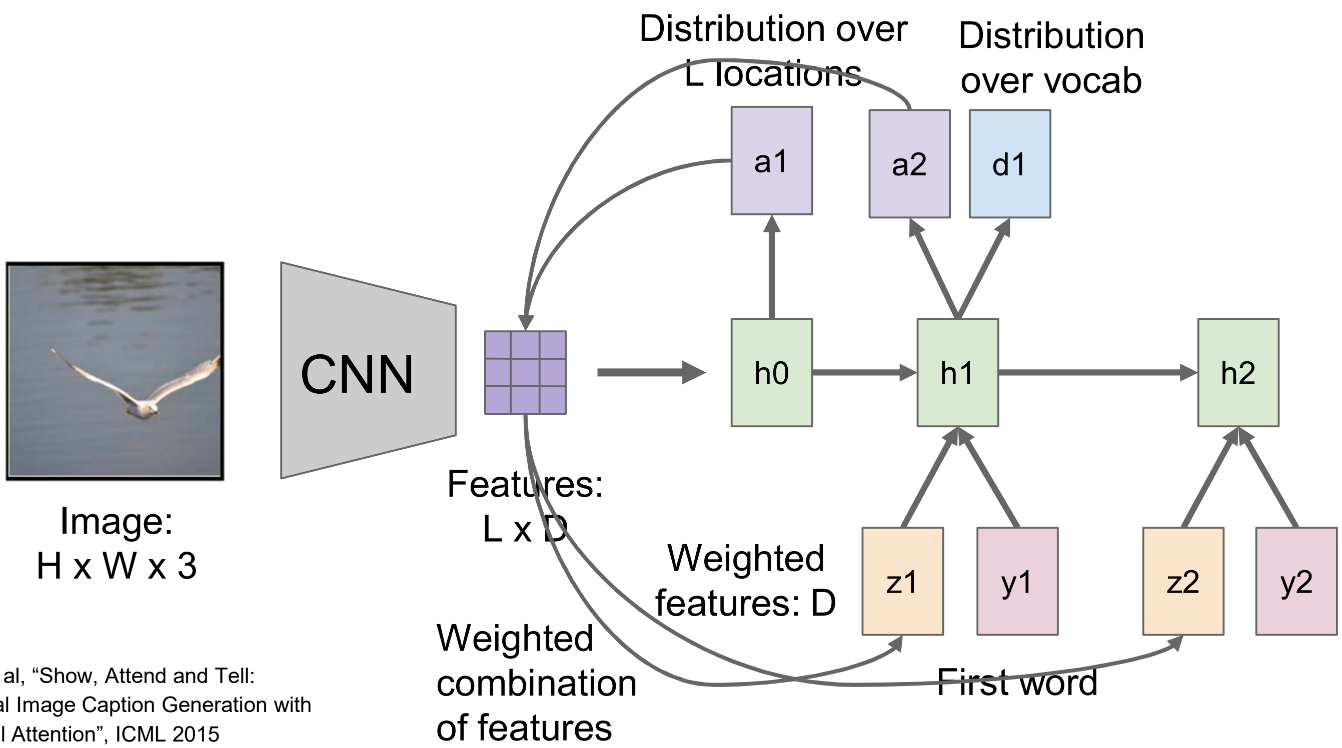
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



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

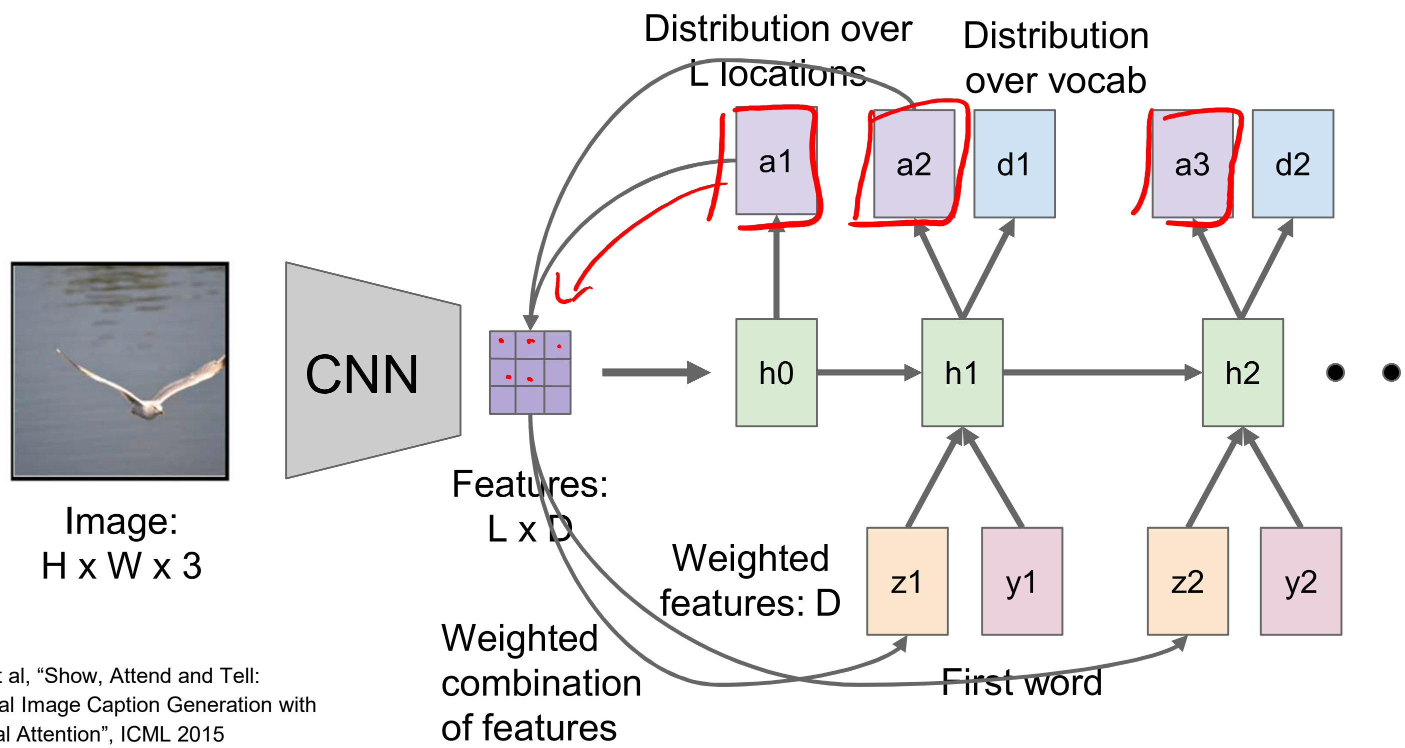
Image Captioning with Attention



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

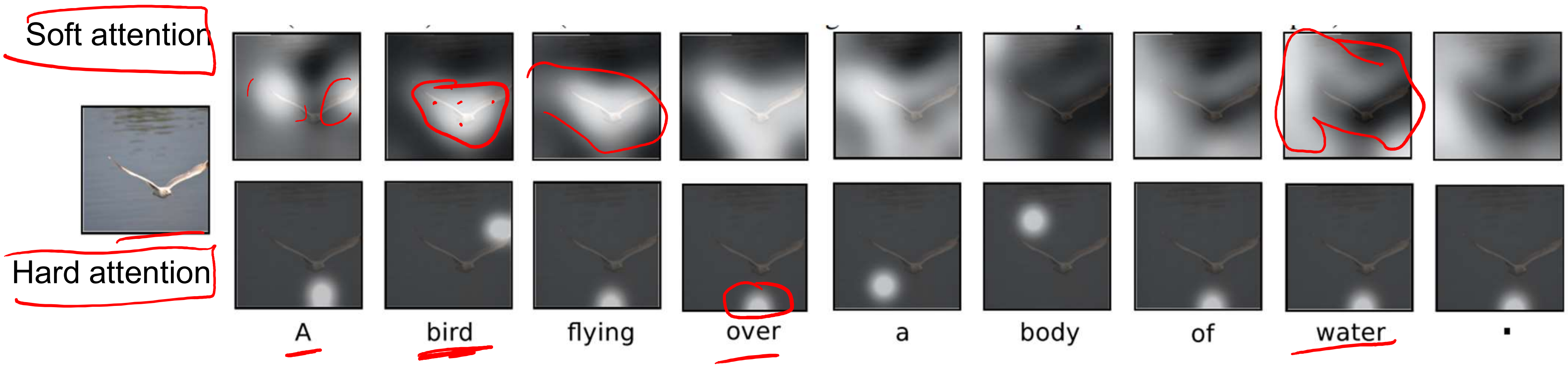
Image Captioning with $\boxed{\text{Attention}}$

where do we
"look at"



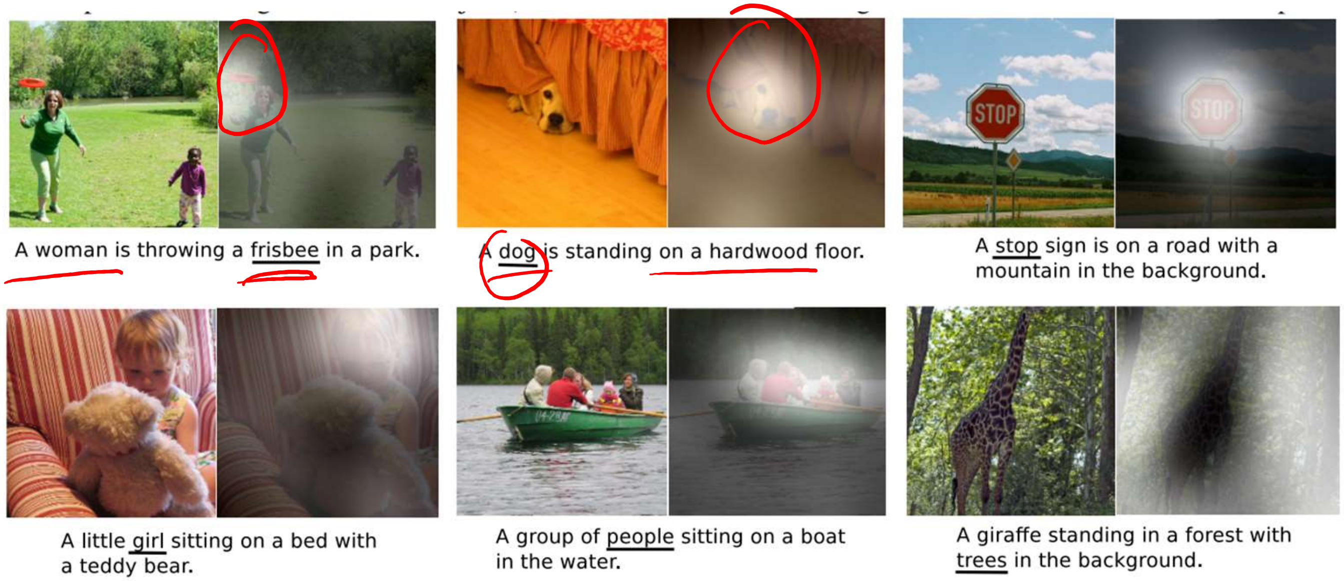
Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Attention



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

Image Captioning with Attention

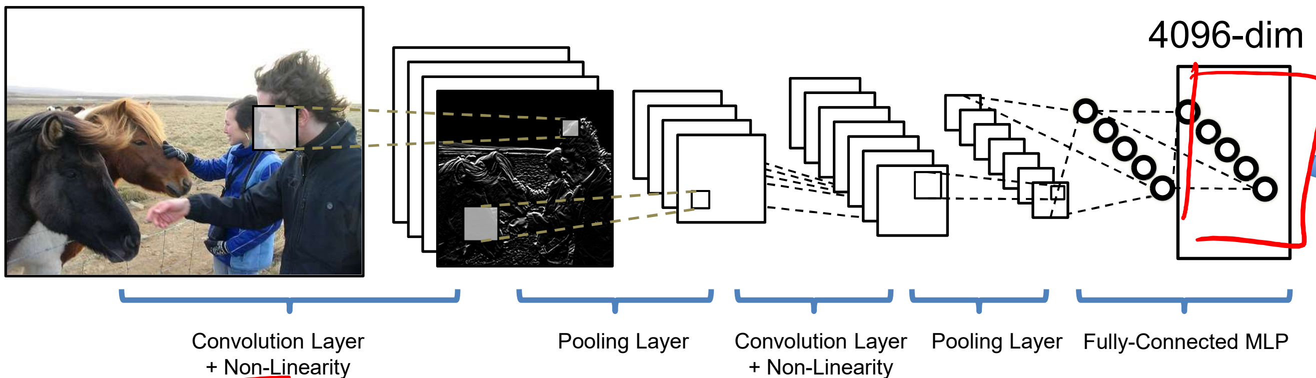


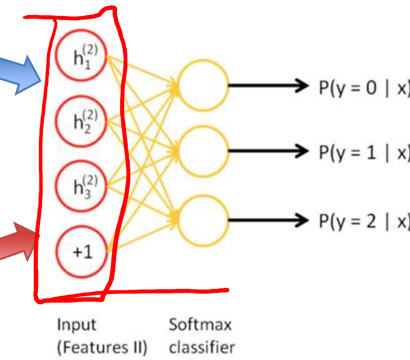
Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Typical VQA Models

Image Embedding (VGGNet)

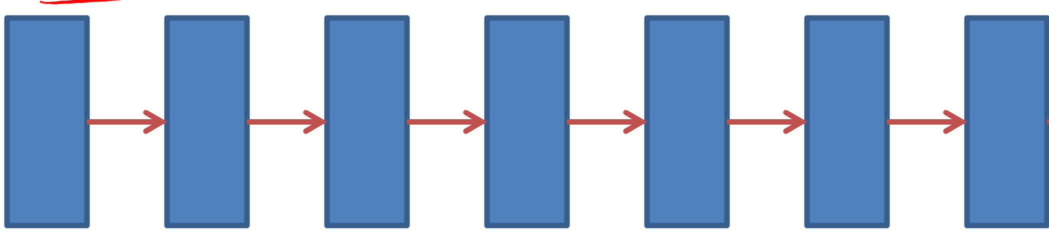
Neural Network
Softmax
over top K answers



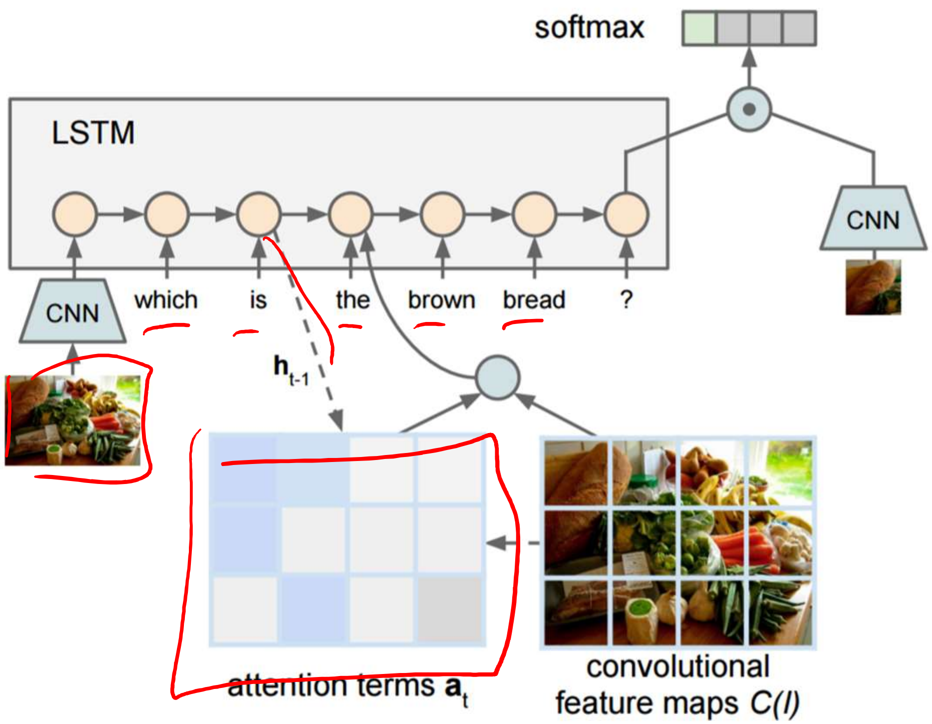


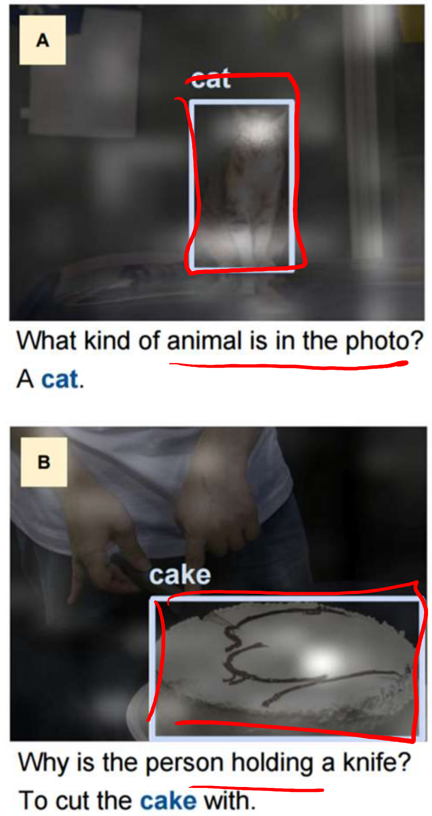
Question Embedding (LSTM)

"How many horses are in this image?"



Visual Question Answering: RNNs with Attention





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

Plan for Today

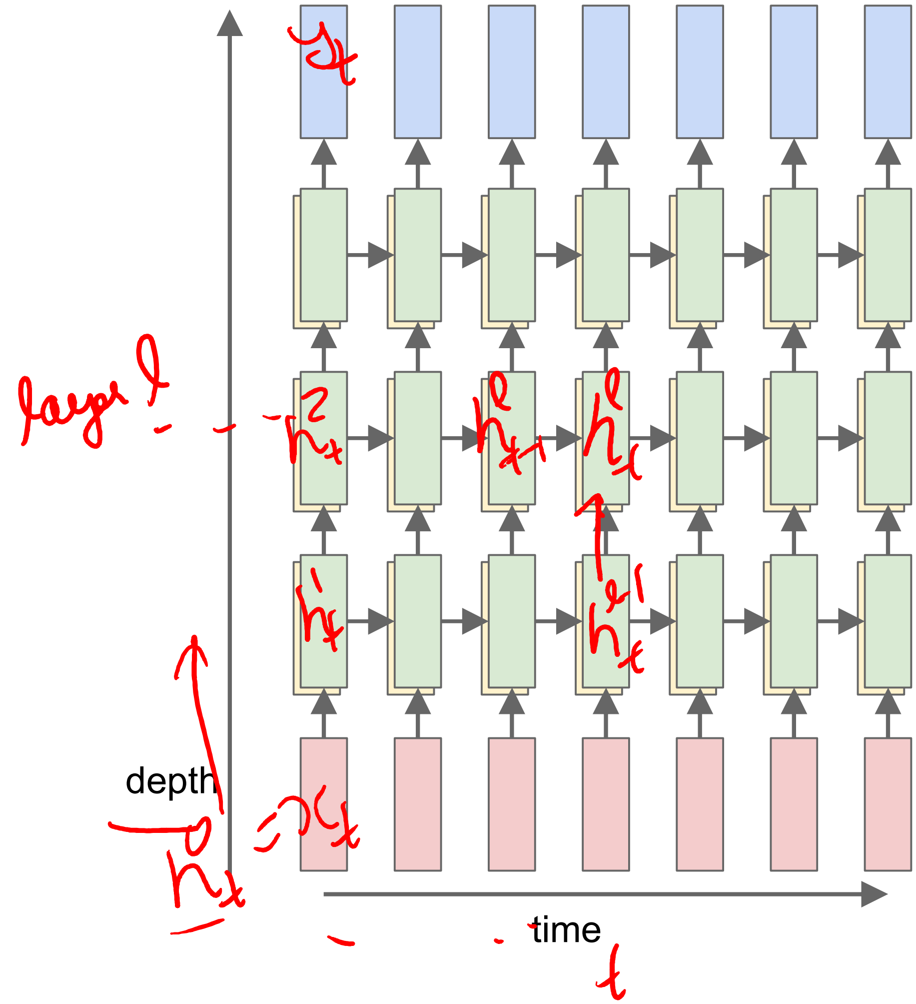
- Recurrent Neural Networks (RNNs)
  - Inference: Beam Search
    - Example: Image Captioning
  - Multilayer RNNs
  - Problems with gradients in “vanilla” RNNs
  - LSTMs (and other RNN variants)

Multilayer RNNs

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

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

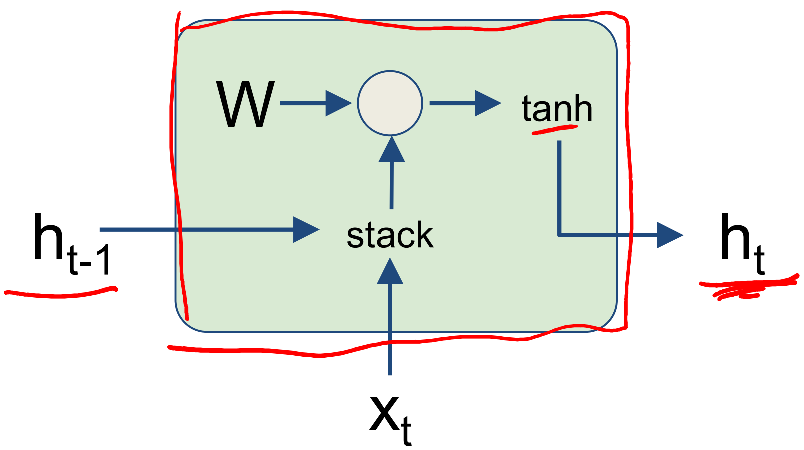
h_t^l



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

Vanilla RNN Gradient Flow

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

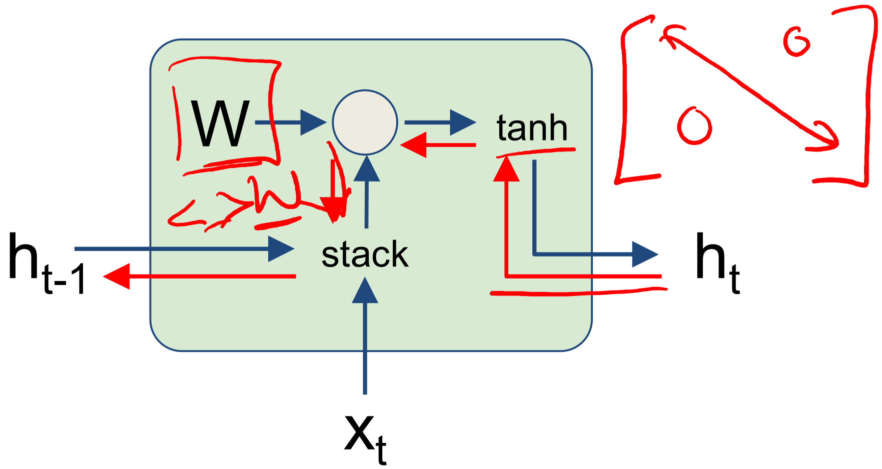


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Vanilla RNN Gradient Flow

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

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

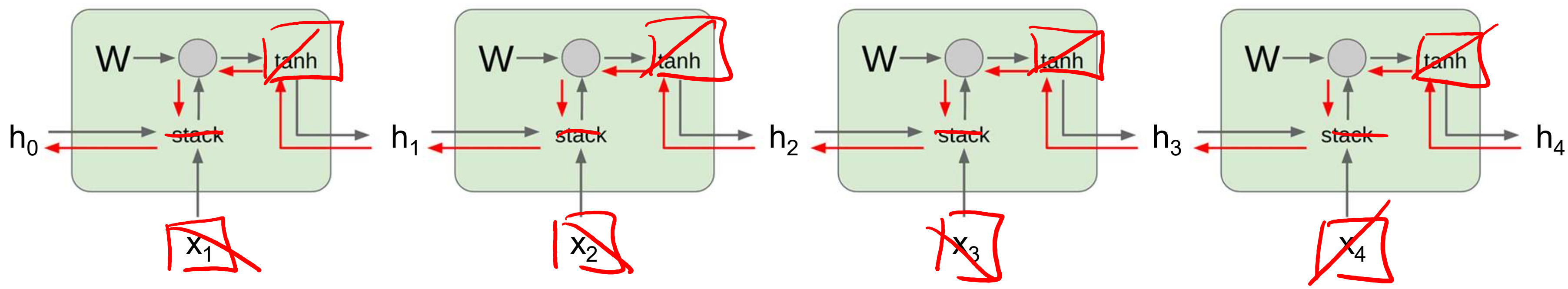


$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix}\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

Vanilla RNN Gradient Flow

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

$h_1 = W h_0$ $\quad h_2 = W h_1$



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

$$\frac{\partial h_T}{\partial h_0} = \boxed{\frac{\partial h_T}{\partial h_{T-1}}} \cdots \frac{\partial h_1}{\partial h_0}$$

$$= W \cdot W \cdot \ldots \cdot W$$

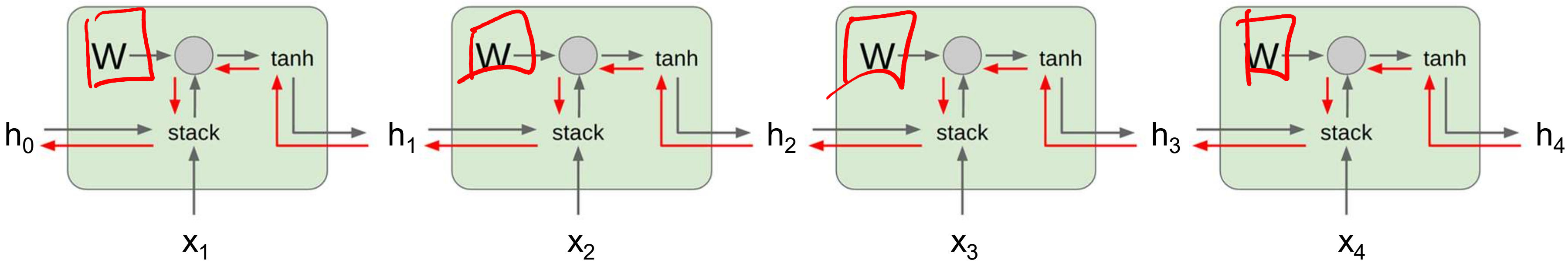
$$= W^{(T+1)}$$

If $h_t \in \mathbb{R}^1$, $W \in \mathbb{R}^1$

$(W)^{T+1}$

Vanilla RNN Gradient Flow

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



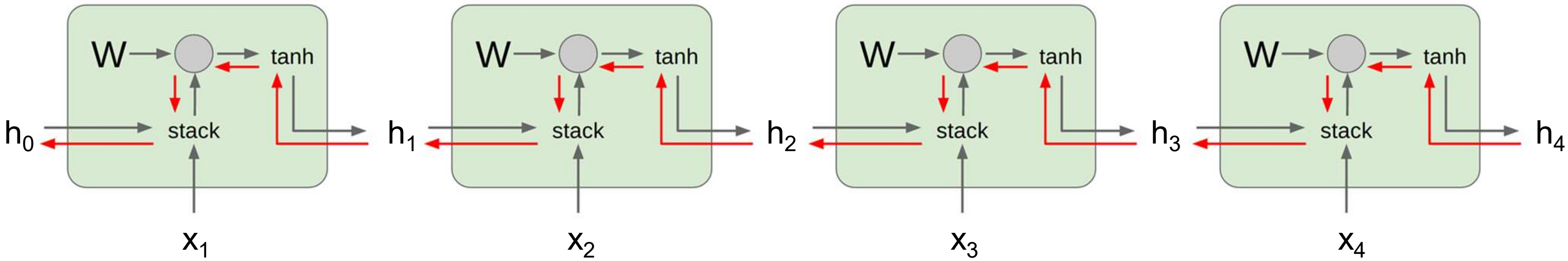
Computing gradient of h_0 involves many factors of W
(and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Vanilla RNN Gradient Flow

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



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

Largest singular value > 1: **Exploding gradients**

Largest singular value < 1: **Vanishing gradients**

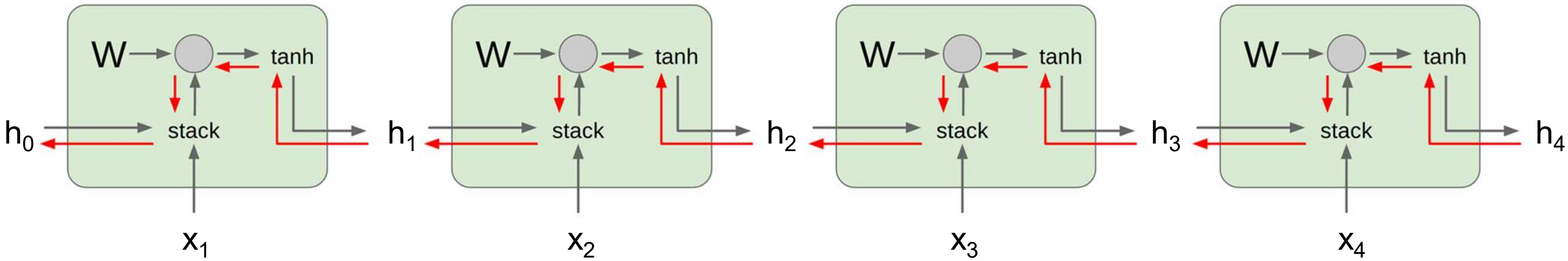
Gradient clipping: Scale gradient if its norm is too big

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

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

Vanilla RNN Gradient Flow

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



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

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

→ Change RNN architecture

Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM

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

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

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

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

Meet LSTMs

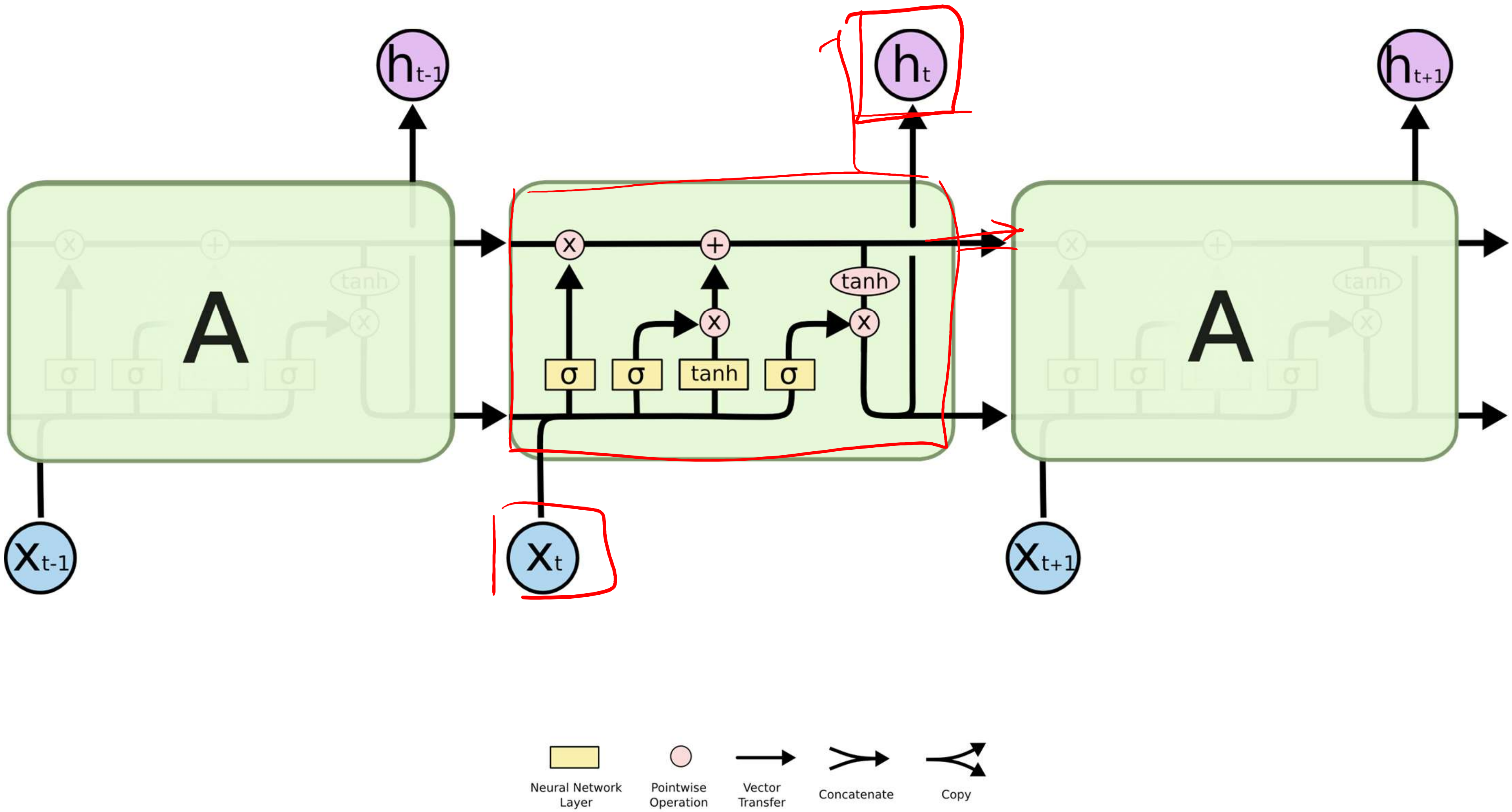


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

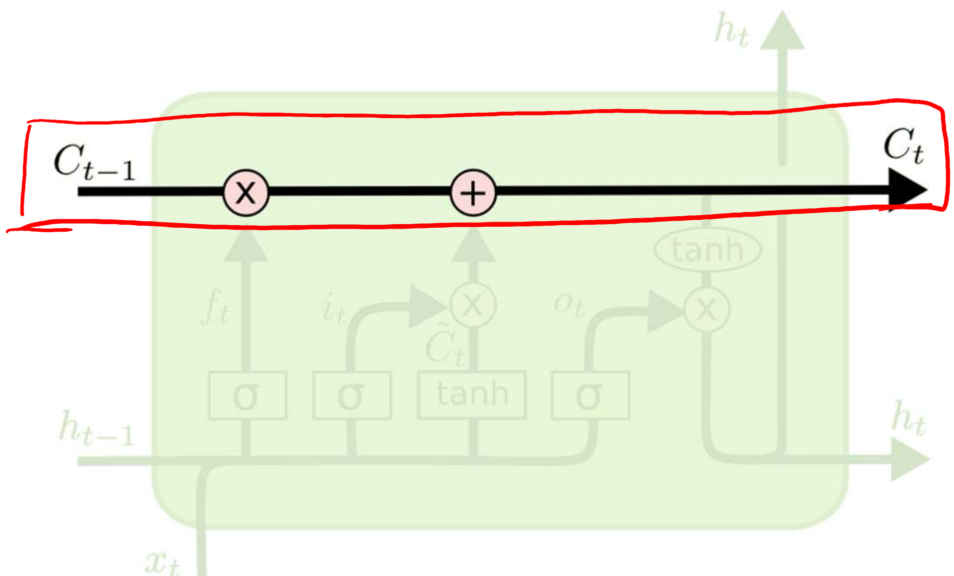
LSTMs Intuition: Memory

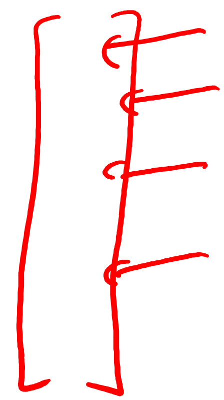
C_t : h_t hidden state

$C_t \in \mathbb{R}^d$ 512

$h_t \in \mathbb{R}^d$

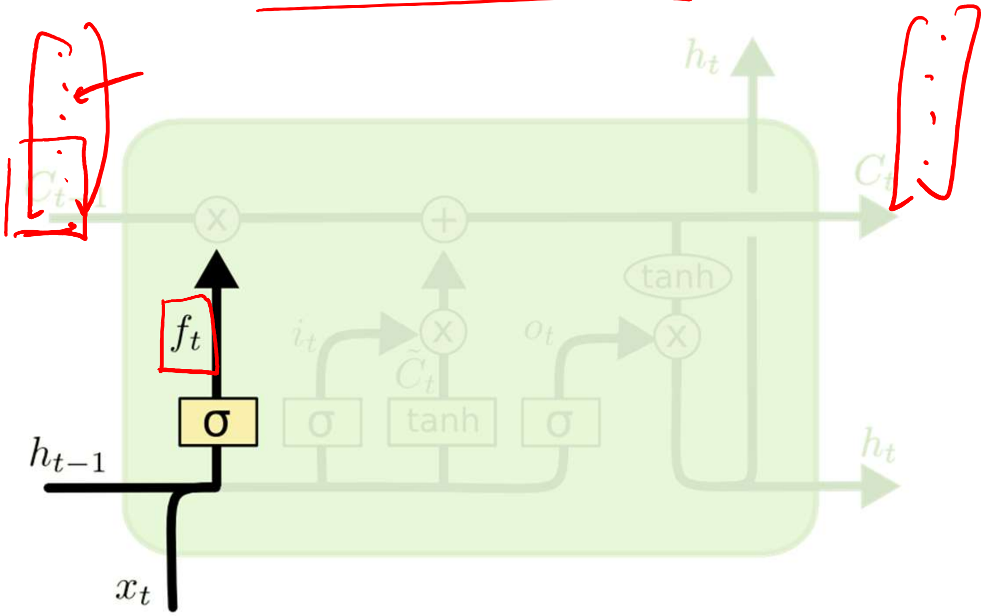
- Cell State / Memory





LSTMs Intuition: Forget Gate

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



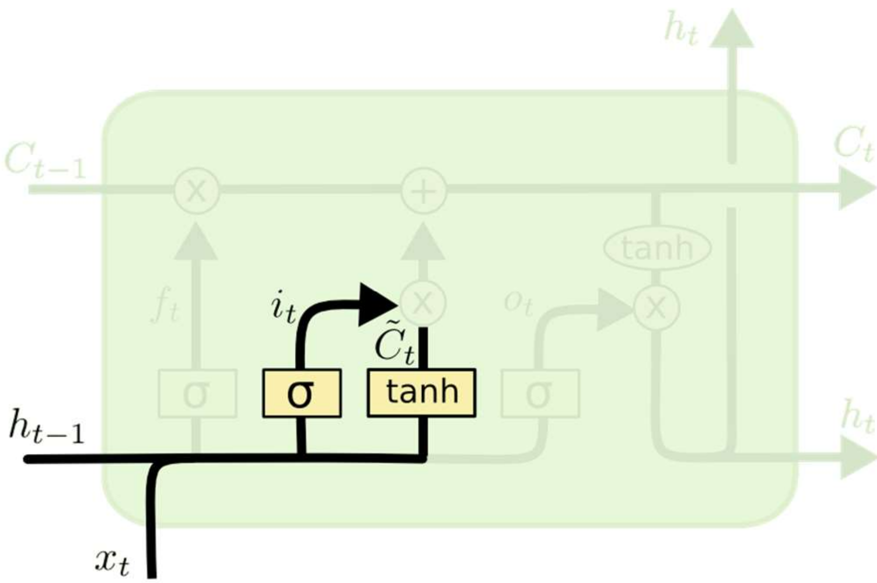
$f_t \in \mathbb{R}^d$

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

LSTMs Intuition: Input Gate

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

$i_t \in \mathbb{R}^d$



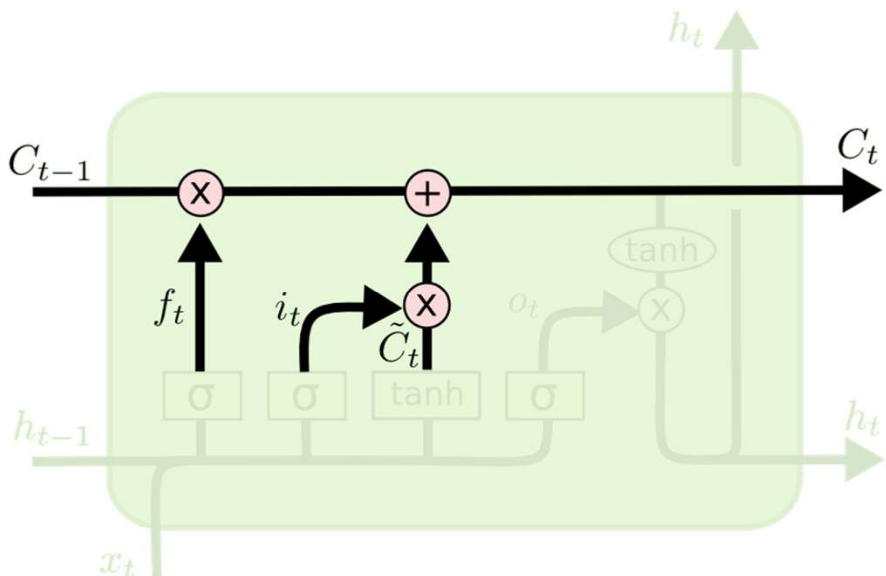
$$i_t = \boxed{\sigma}(\boxed{W_i} \cdot [h_{t-1}, x_t] \; + \; b_i)$$

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

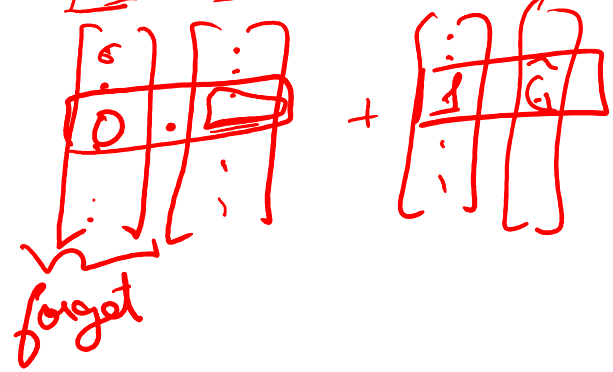
$\tilde{C}$

LSTMs Intuition: Memory Update

- Forget that + memorize this

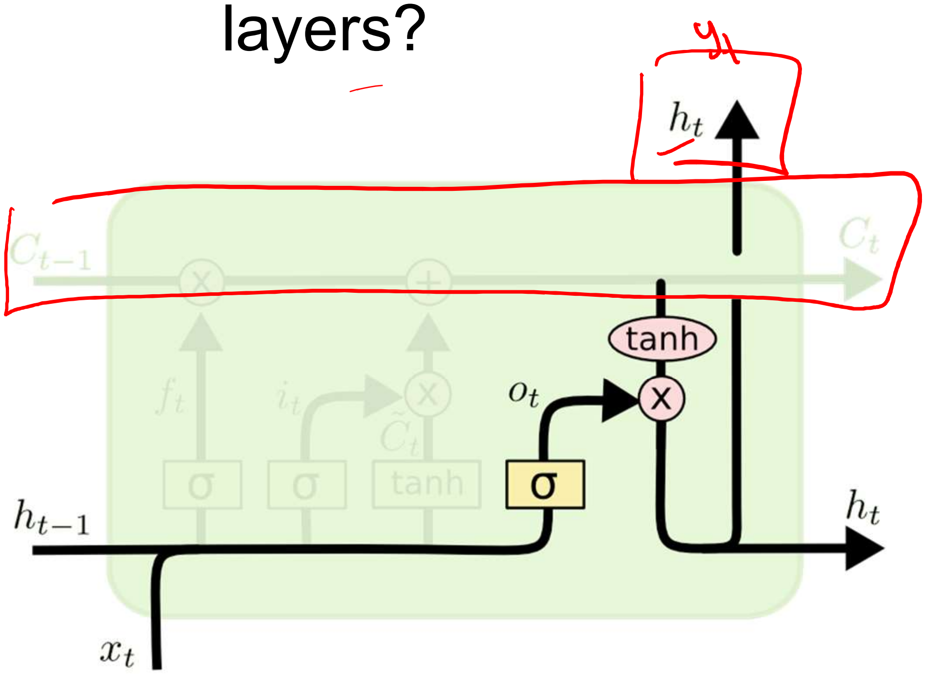


$$C_t = \boxed{f_t} * \underline{C_{t-1}} + \boxed{i_t} * \tilde{C}_t$$



LSTMs Intuition: Output Gate

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

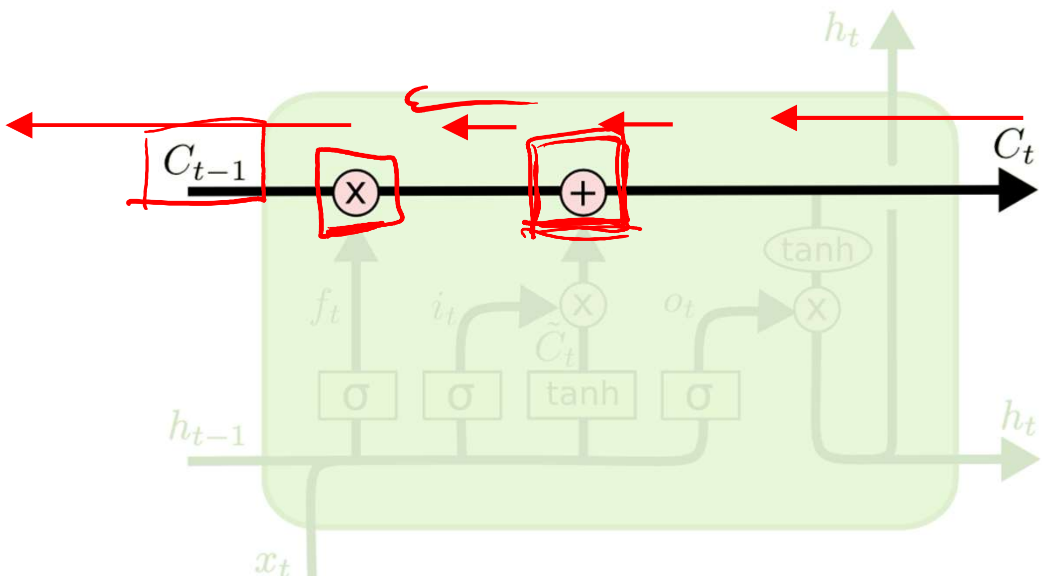


$o_t \in \mathbb{R}^d$

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

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

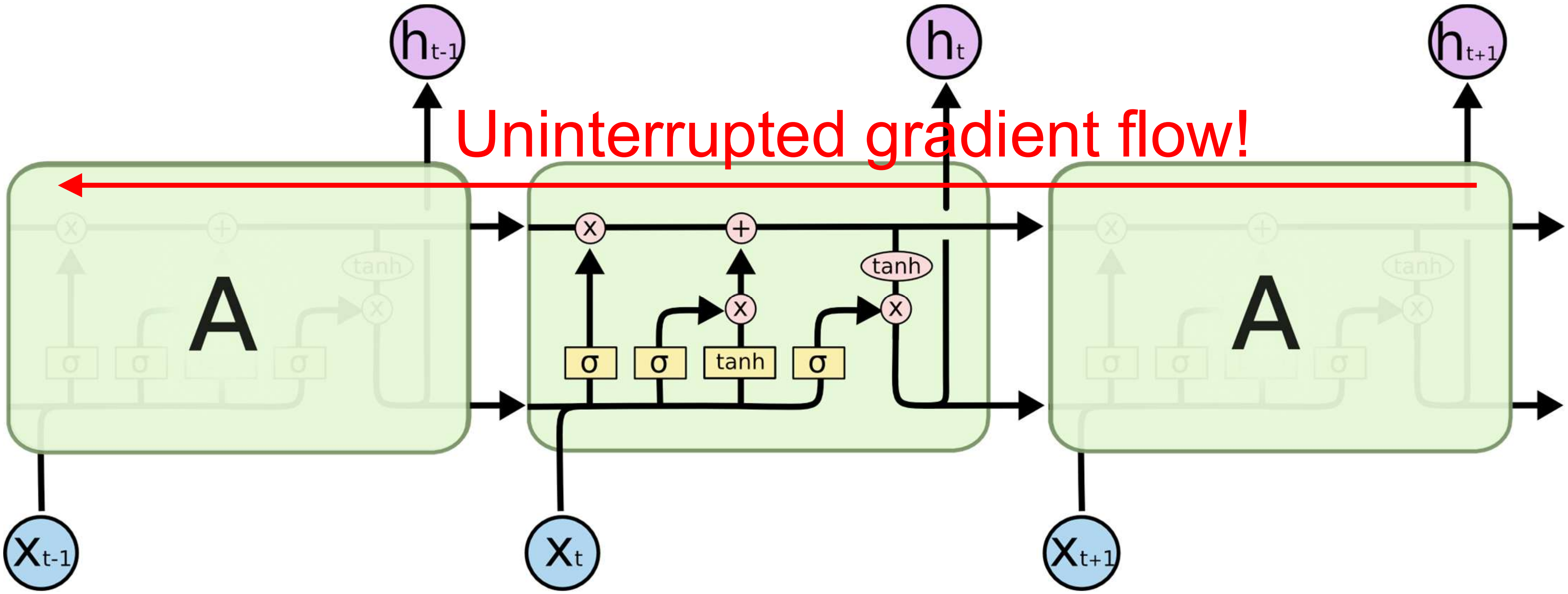
LSTMs Intuition: Additive Updates



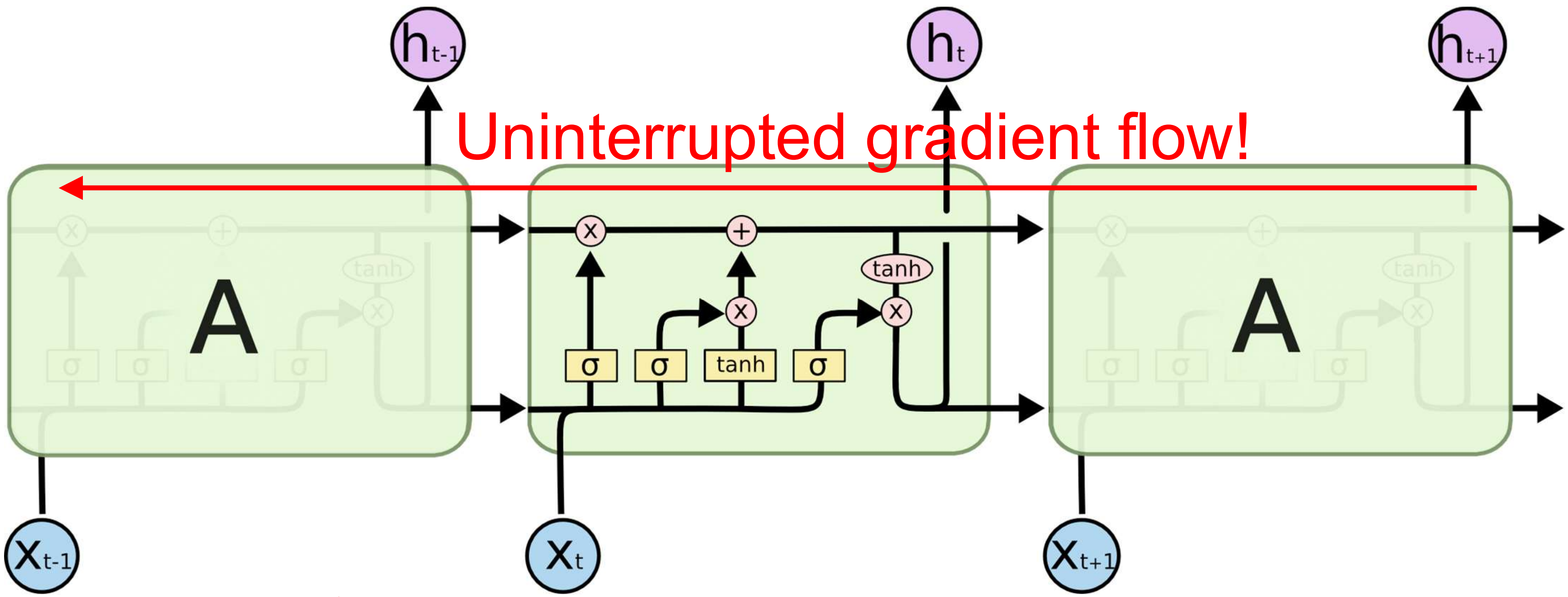
Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

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

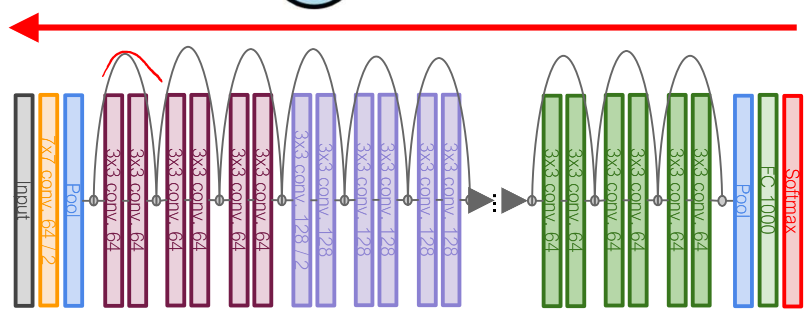
LSTMs Intuition: Additive Updates



LSTMs Intuition: Additive Updates

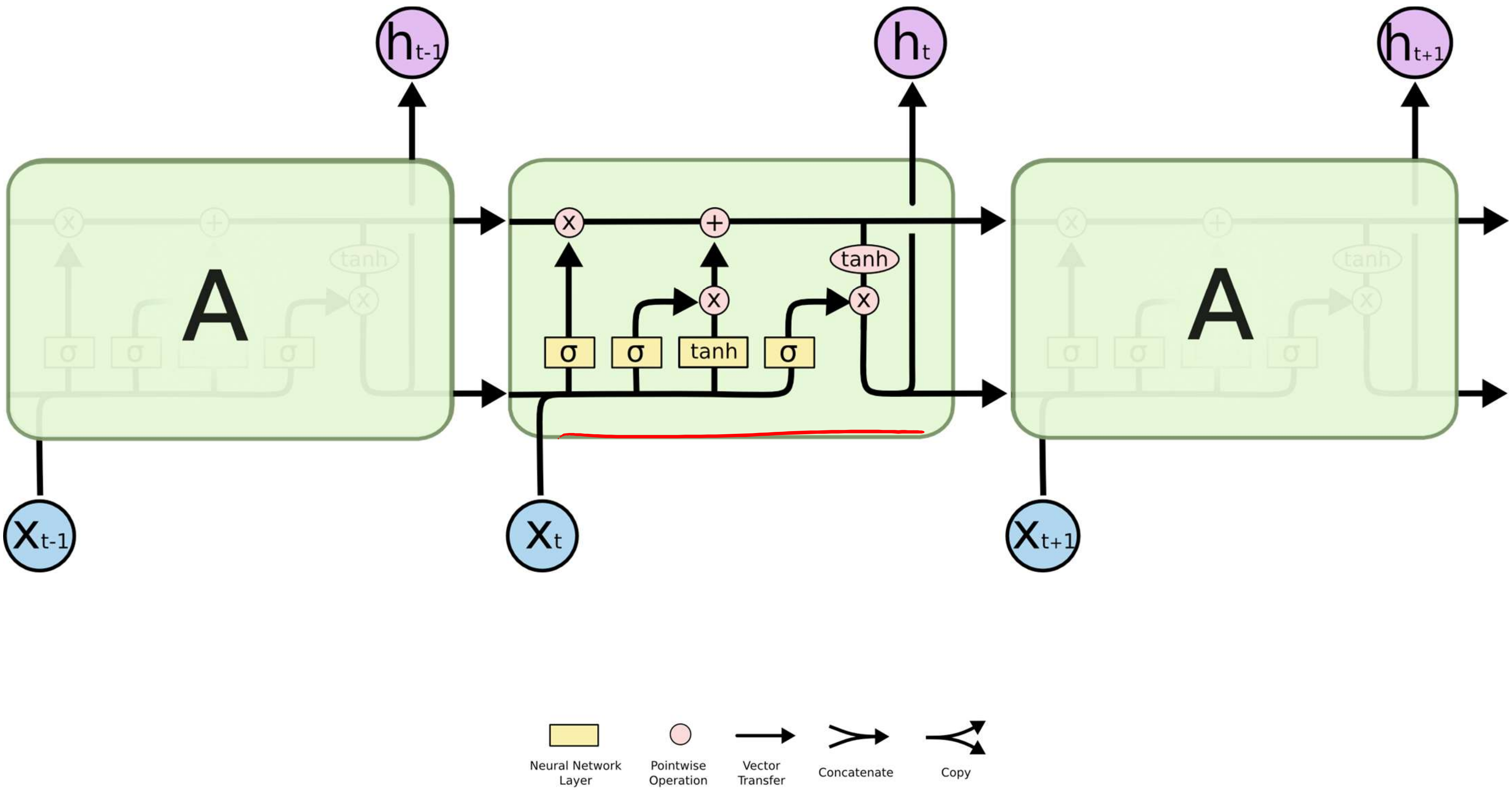


Similar to ResNet!



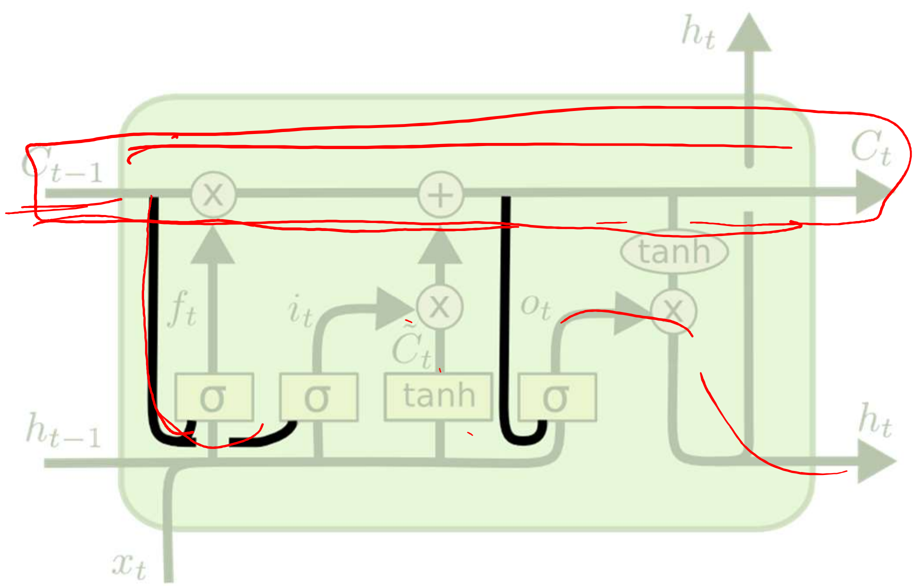
LSTMs

- A pretty sophisticated cell



LSTM Variants #1: Peephole Connections

- Let gates see the cell state / memory



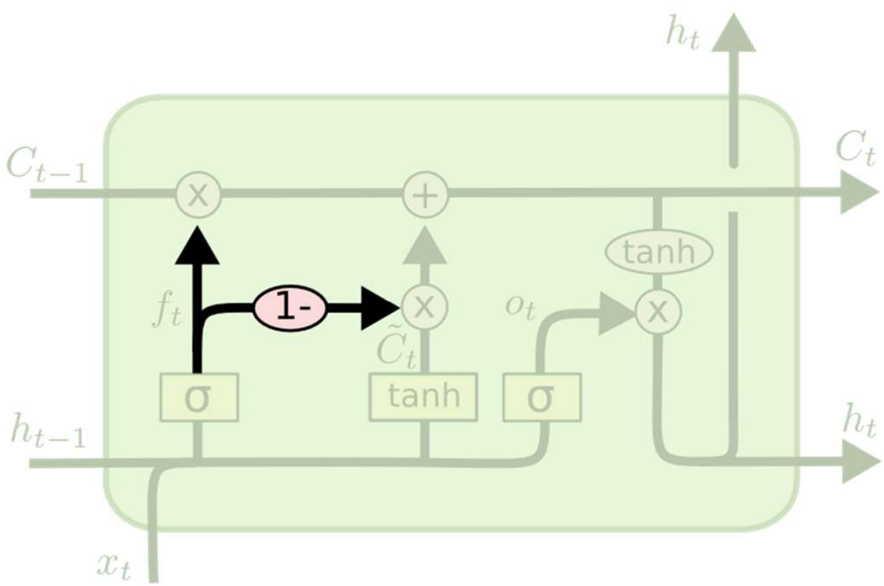
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$

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

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

LSTM Variants #2: Coupled Gates

- Only memorize new if forgetting old

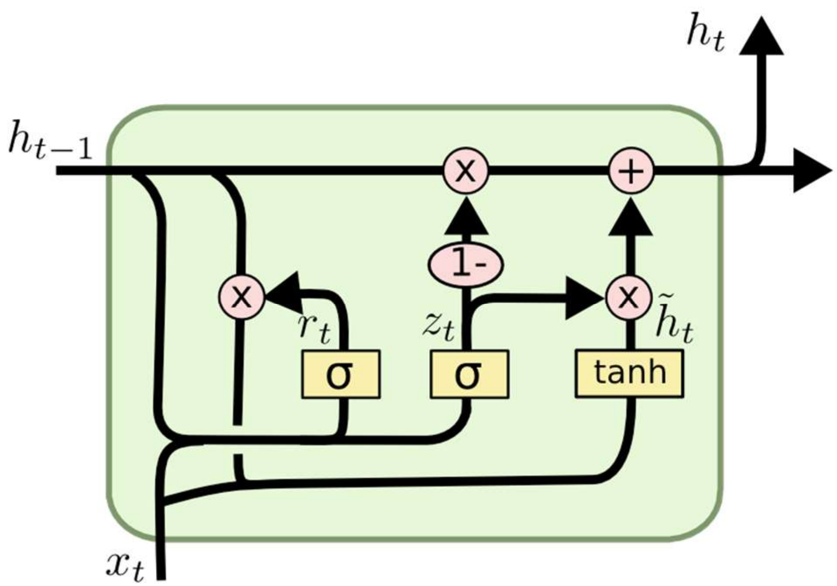


$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

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

LSTM Variants #3: Gated Recurrent Units

- Changes:
  - No explicit memory; memory = hidden output
  - Z = memorize new and forget old



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

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

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

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

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

Other RNN Variants

[*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:

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

Summary

- 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.