CS 4803 / 7643: Deep Learning

Topics:

- Recurrent Neural Networks (RNNs)
  - Beam Search
  - LSTMs
- Reinforcement Learning

Dhruv Batra

Georgia Tech

Administrativia

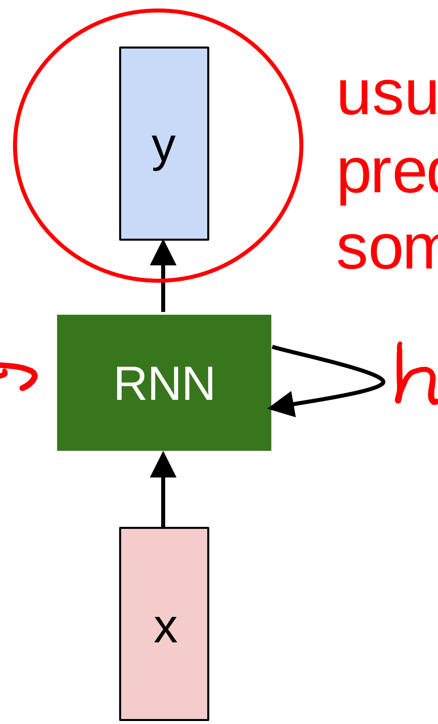
- HW4
  - Due: 10/21, 11:59pm
  - RNNs, LSTMs, Image Captioning, Transformers!

Recap from last time

Recurrent Neural Network

"cell"

Recurrent unit of computation



usually want to predict a vector at some time steps

cell "hidden" / rep.

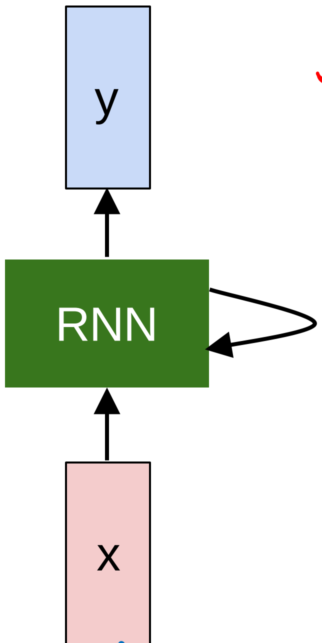
"state"

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

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

(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

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

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

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

$$\left(\begin{bmatrix} W_{hh} & W_{xh} \end{bmatrix} \begin{bmatrix} \cancel{h_{t-2}} \\ h_{t-1} \\ x_t \end{bmatrix} + \vec{b}_h \right)$$

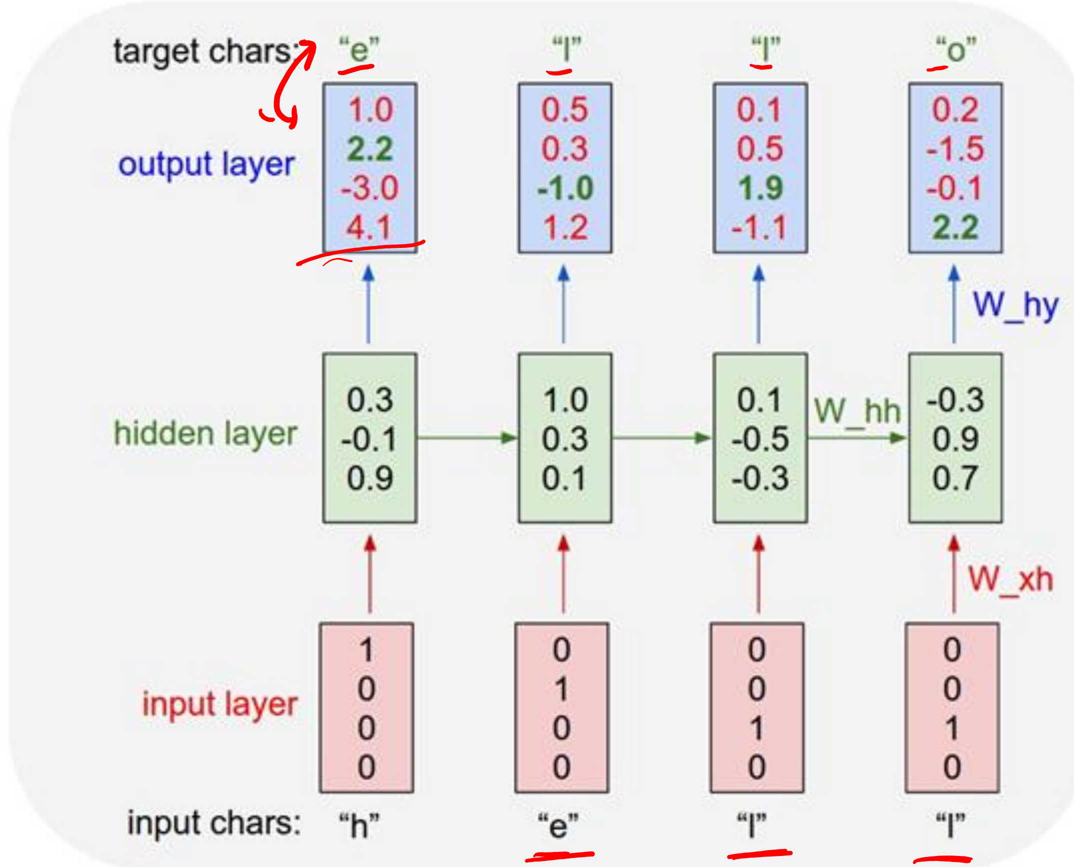
Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

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

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



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

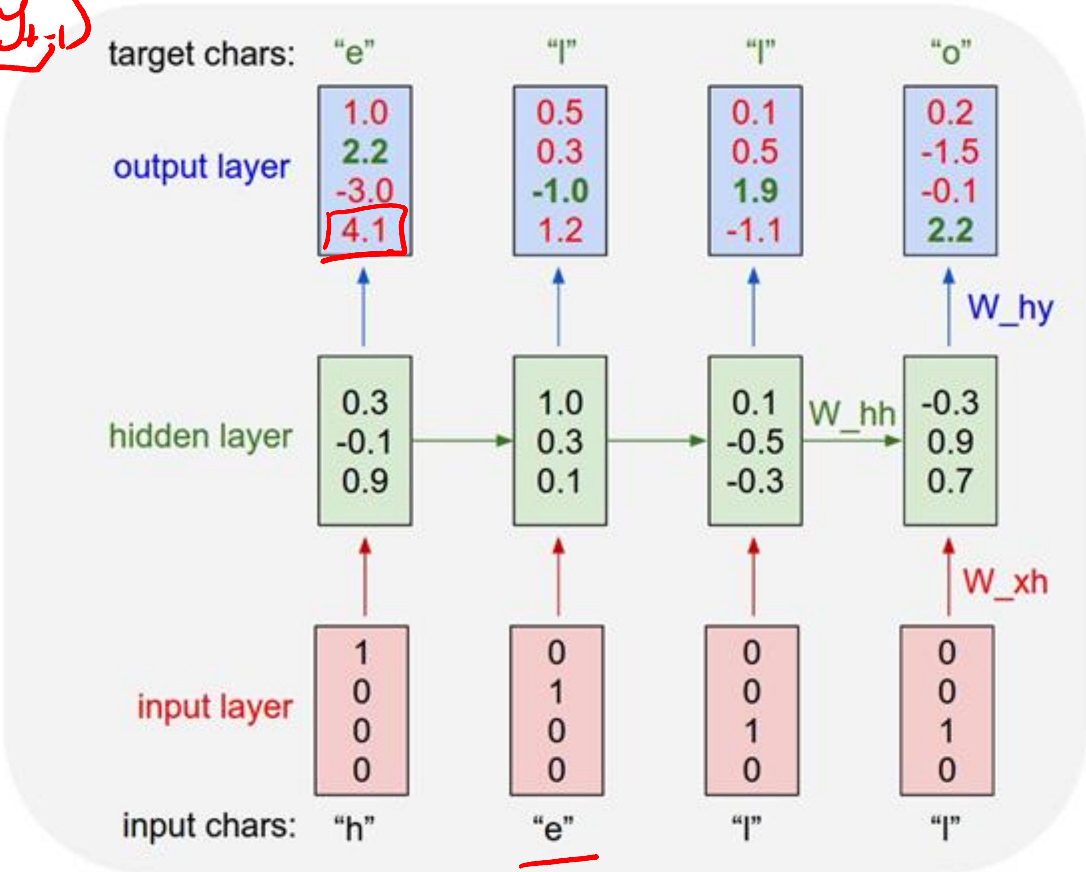
Training Time: $\boxed{\text{MLE}}$ / “Teacher Forcing”

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

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



$x_2 = y_1^{gt}$

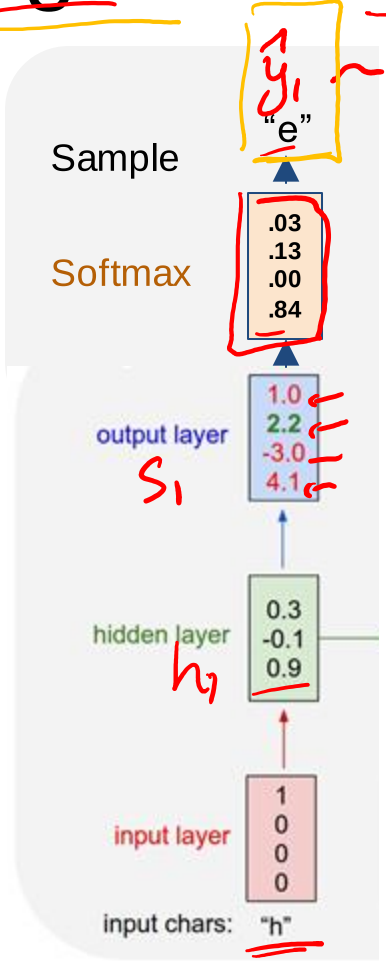
Test Time: Sample / Argmax / Beam Search

$\hat{y}_1 \sim \text{softmax}(S_1)$

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

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

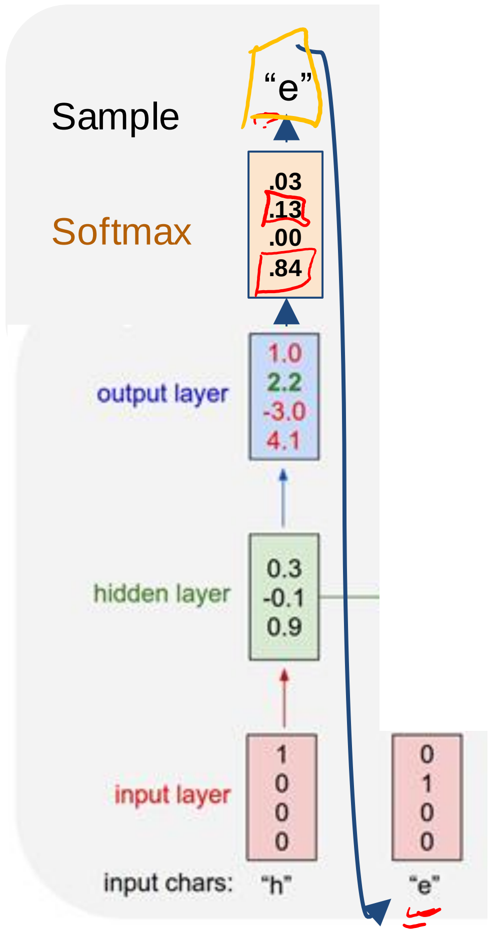


Test Time: Sample / Argmax / Beam Search

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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

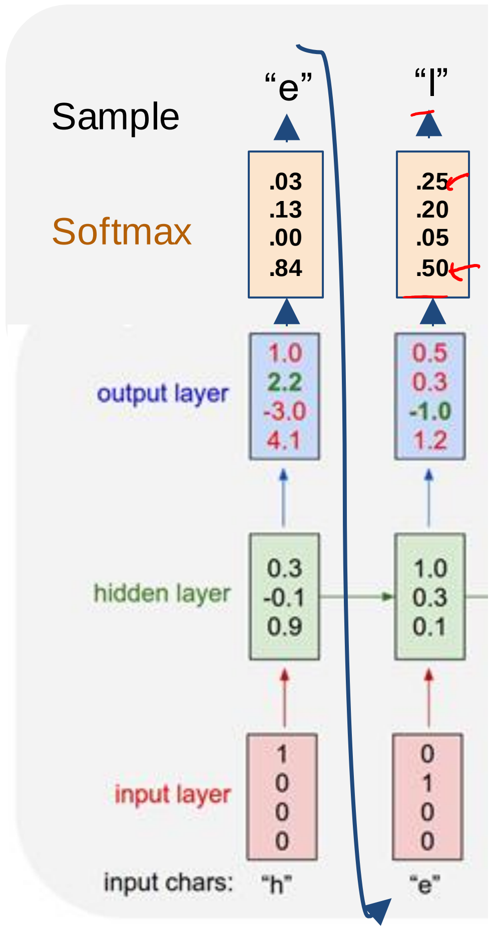


Test Time: Sample / Argmax / Beam Search

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

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

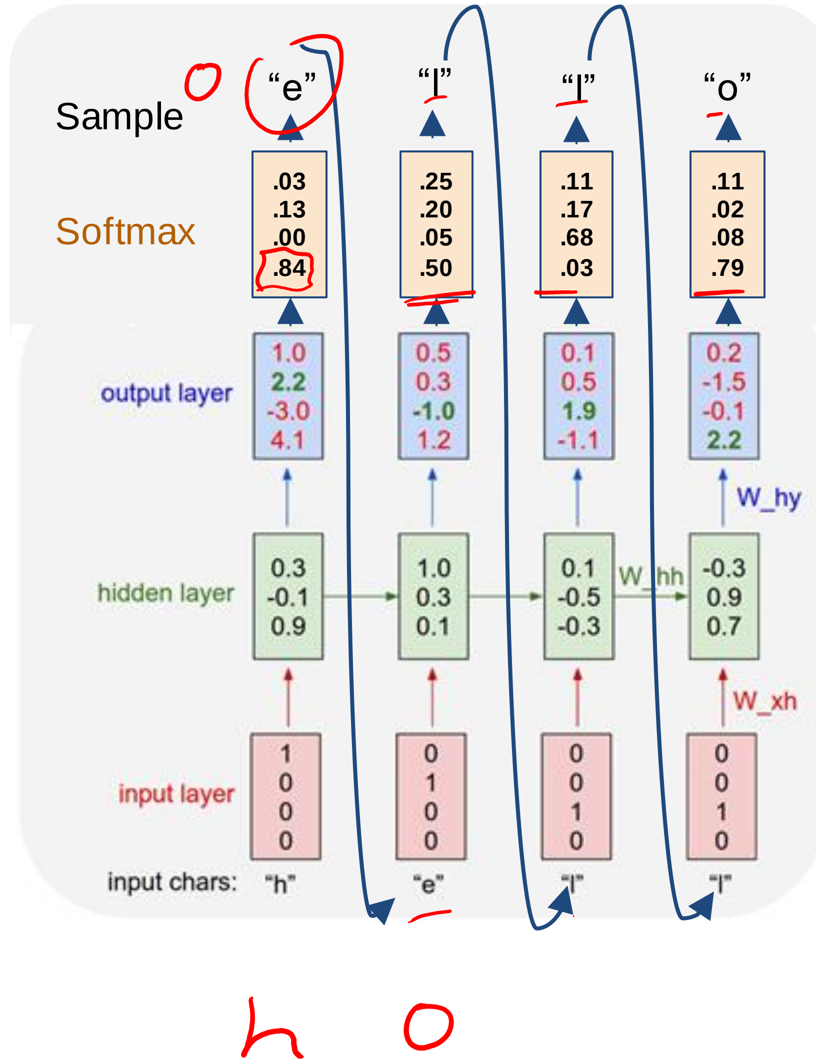


Test Time: Sample / Argmax / Beam Search

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

Plan for Today

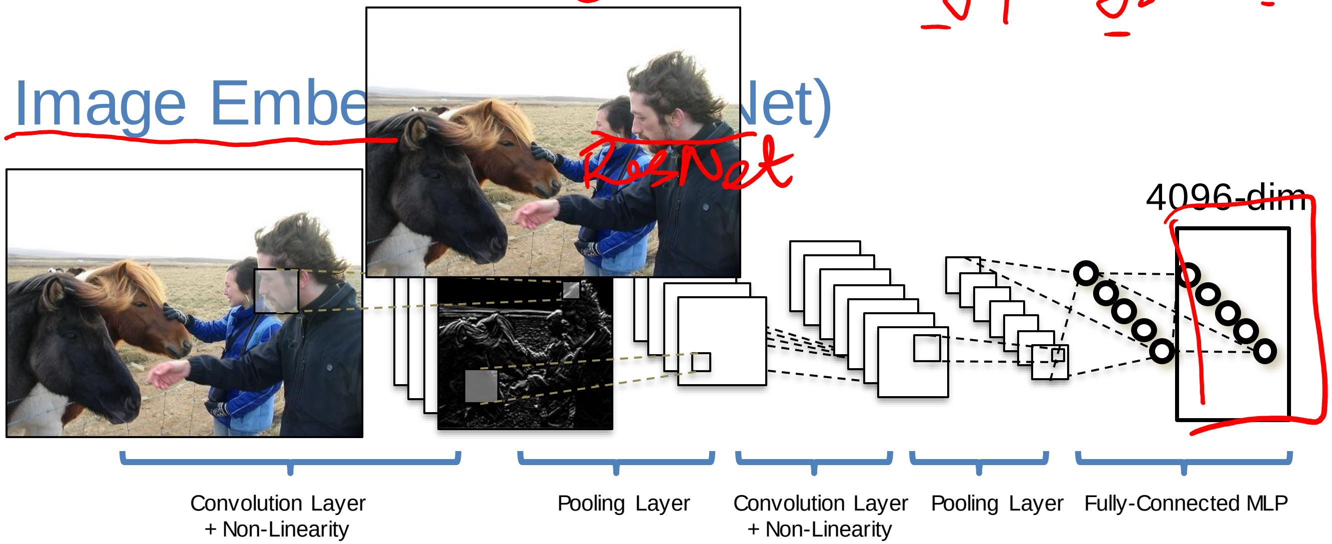
- Recurrent Neural Networks (RNNs)
  - Inference: Beam Search
    - Example: Image Captioning
  - Multilayer RNNs
  - Problems with gradients in “vanilla” RNNs
  - LSTMs (and other RNN variants)

Neural Image Captioning

$\vec{x}$ $\longrightarrow$ y_1 y_2 . . . . y_T

Image Embe… Net)

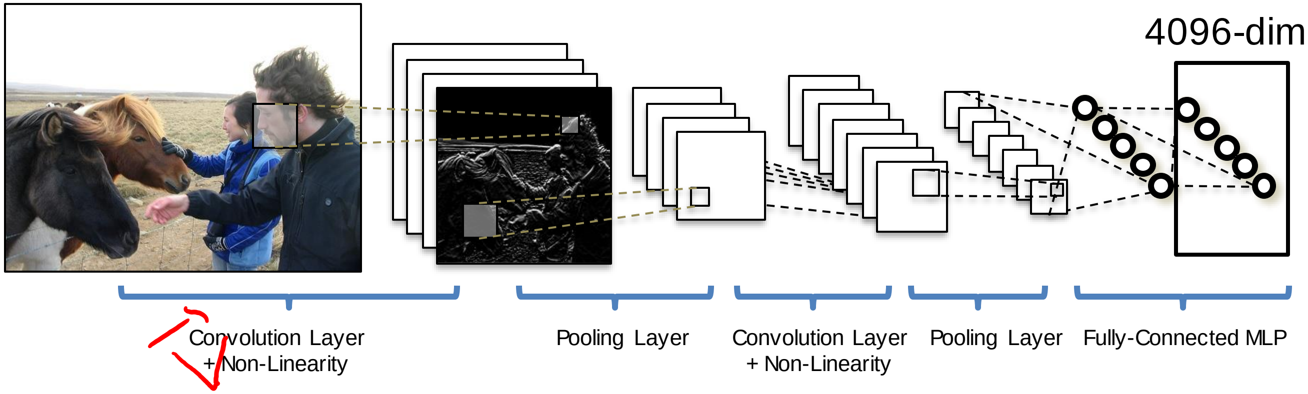
ResNet



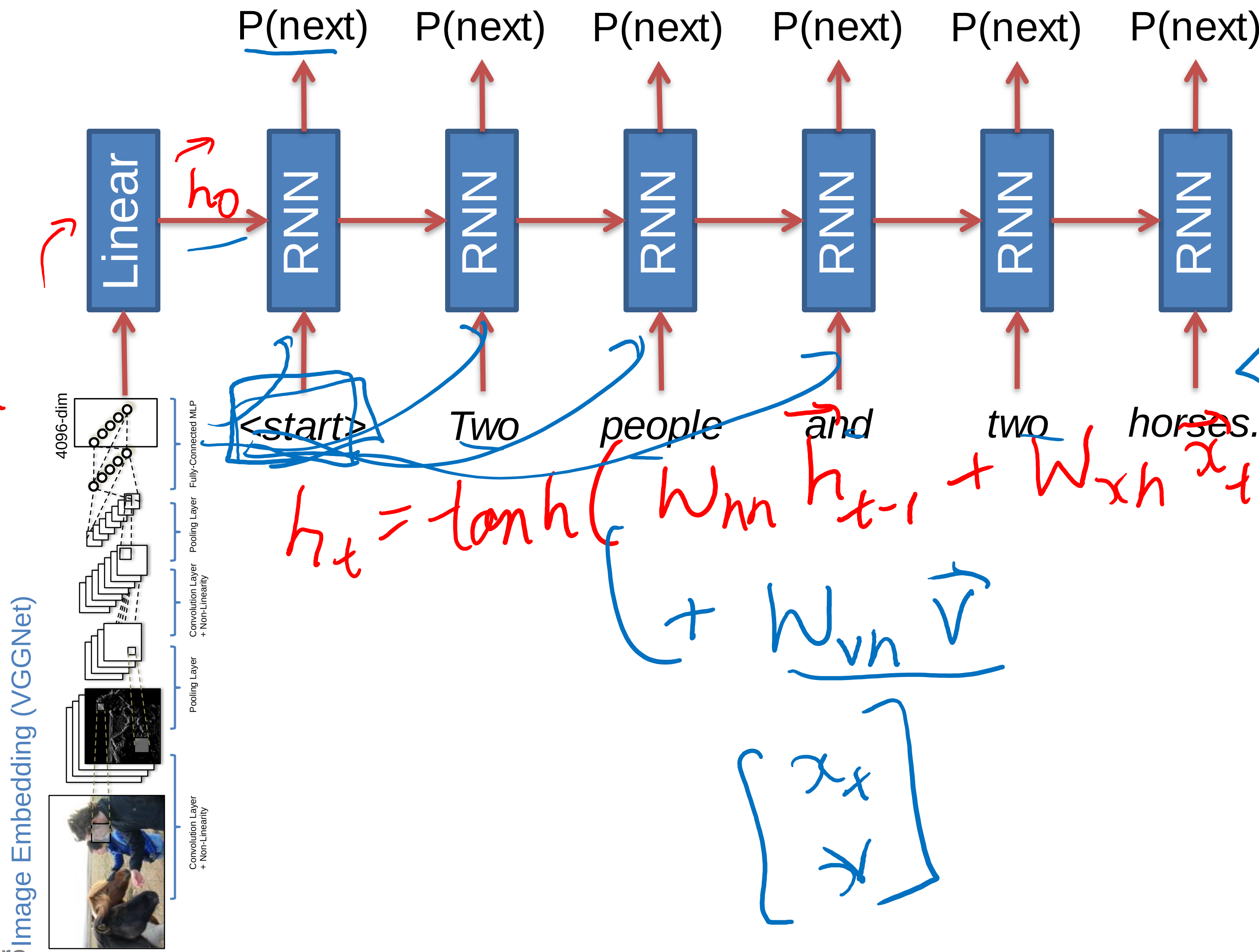
$\vec{v} \in \mathbb{R}^d$ (4096)

Neural Image Captioning

Image Embedding (VGGNet)



Neural Image Captioning



$\vec{v}$ $\vec{h_0}$ <stop>

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

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

Beam Search Demo

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

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

Image Captioning: Example Results

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

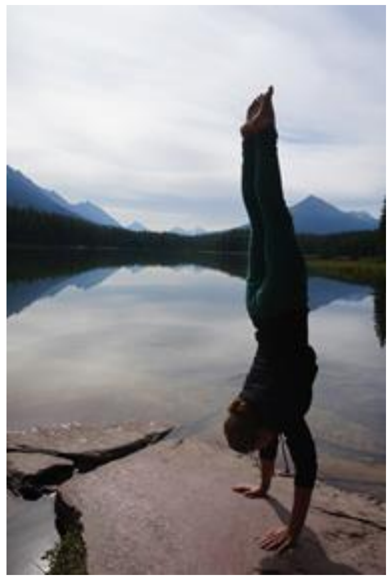
A man riding a dirt bike on a dirt track

Image Captioning: Failure Cases

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

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Plan for Today

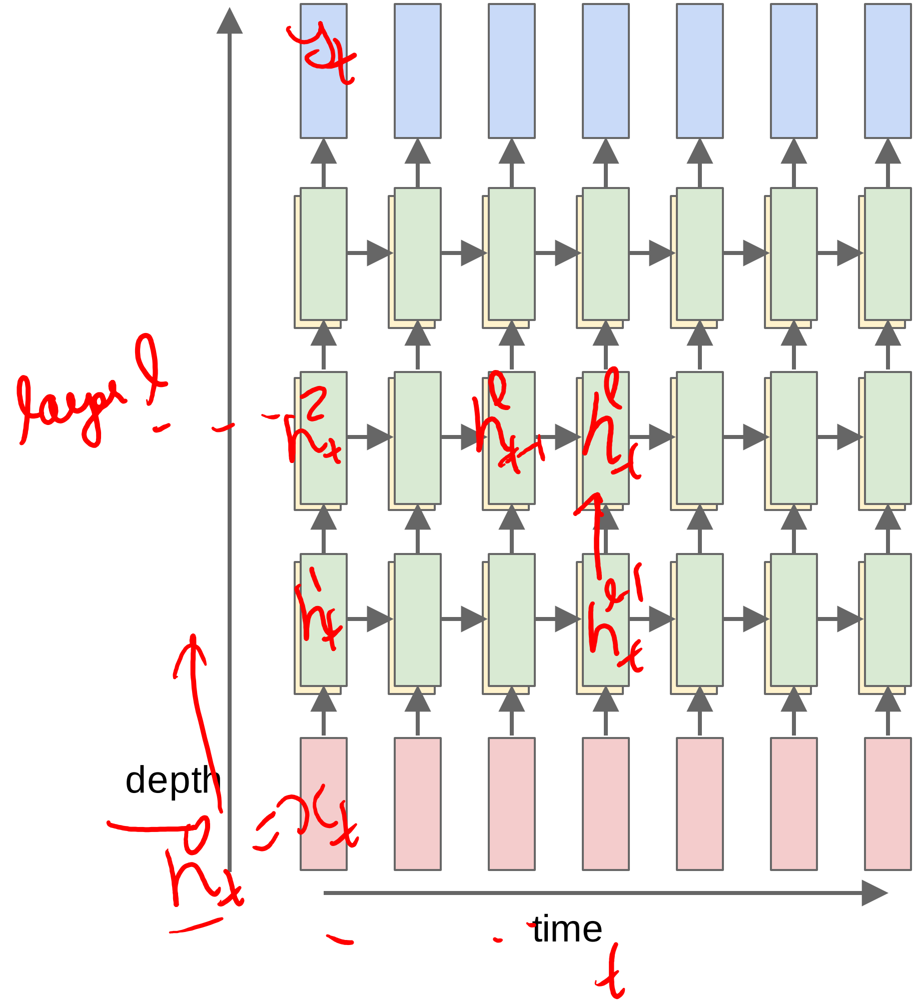
- Recurrent Neural Networks (RNNs)
  - Inference: Beam Search
    - Example: Image Captioning
  - Multilayer RNNs
  - Problems with gradients in “vanilla” RNNs
  - LSTMs (and other RNN variants)

Multilayer RNNs

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

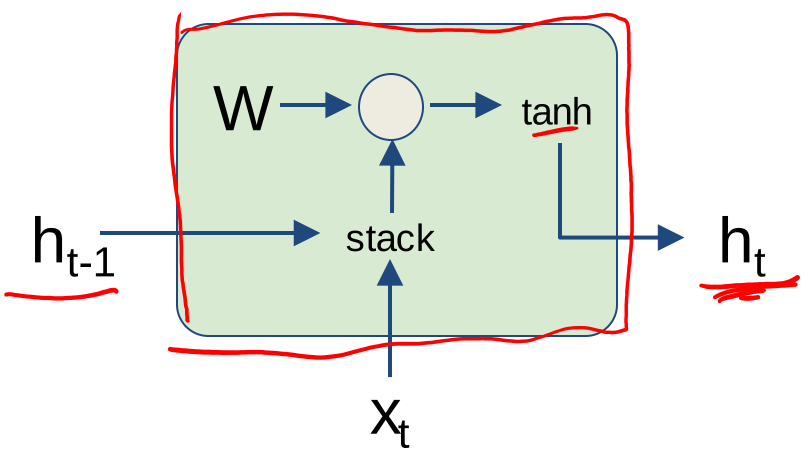
$h \in \mathbb{R}^n$ W^l $[n \times 2n]$

h_t^l



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

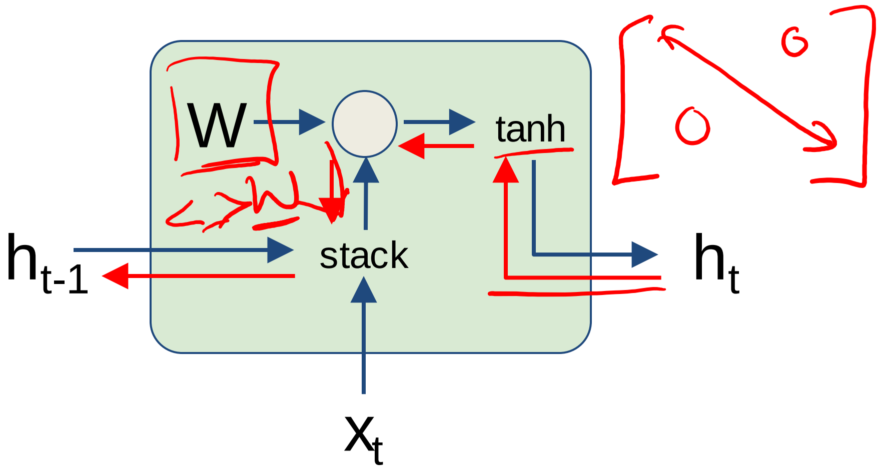


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

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})

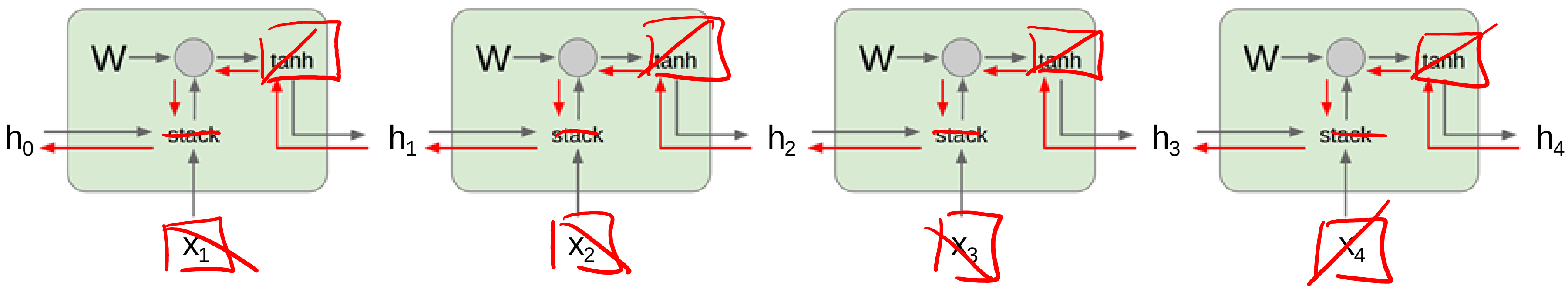


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

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

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



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

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

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

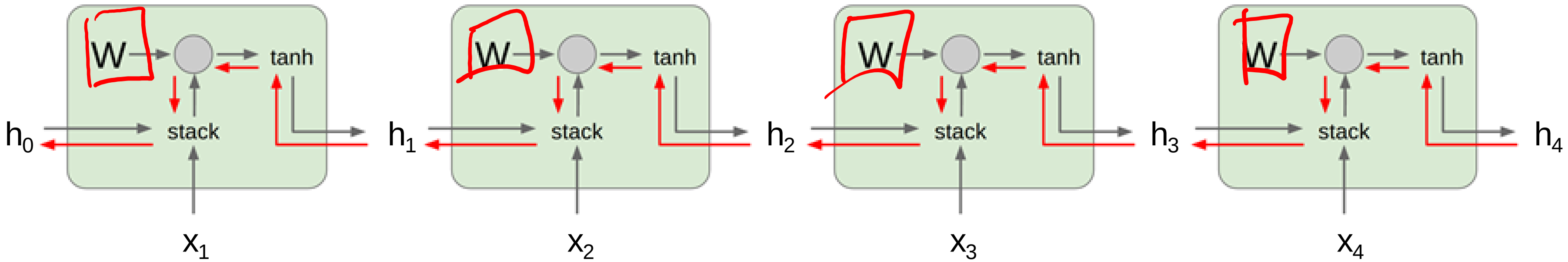
$$= W^{(T+1)}$$

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



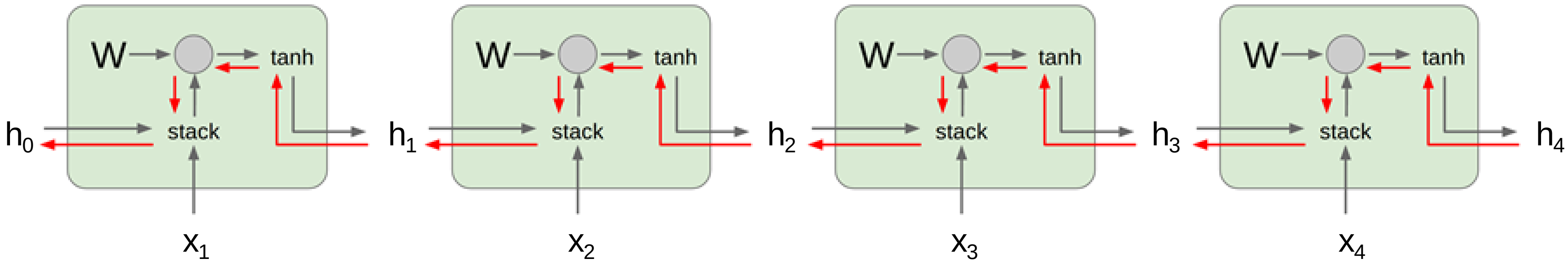
Computing gradient of h_0 involves many factors of W (and repeated tanh)

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

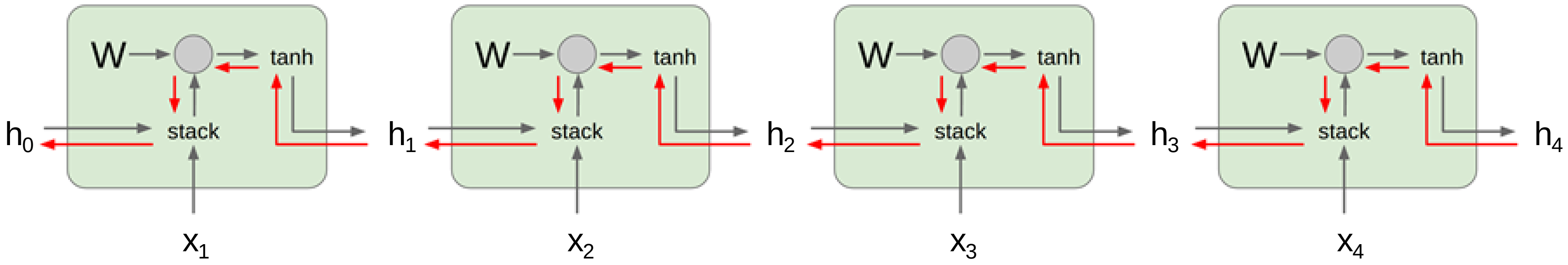
Largest singular value < 1:
Vanishing gradients

Gradient clipping: Scale gradient if its norm is too big

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

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

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

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

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

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

Meet LSTMs

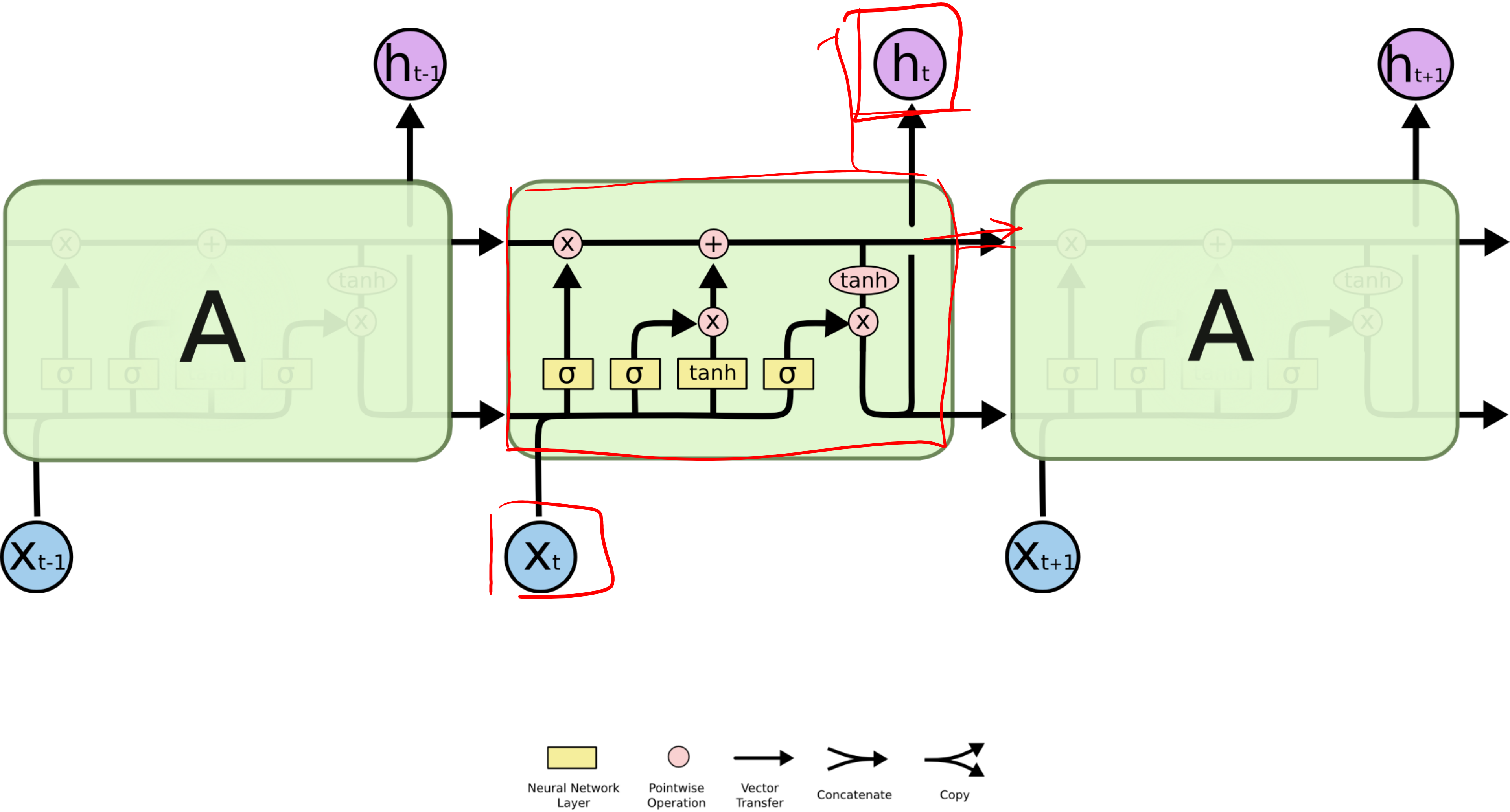


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

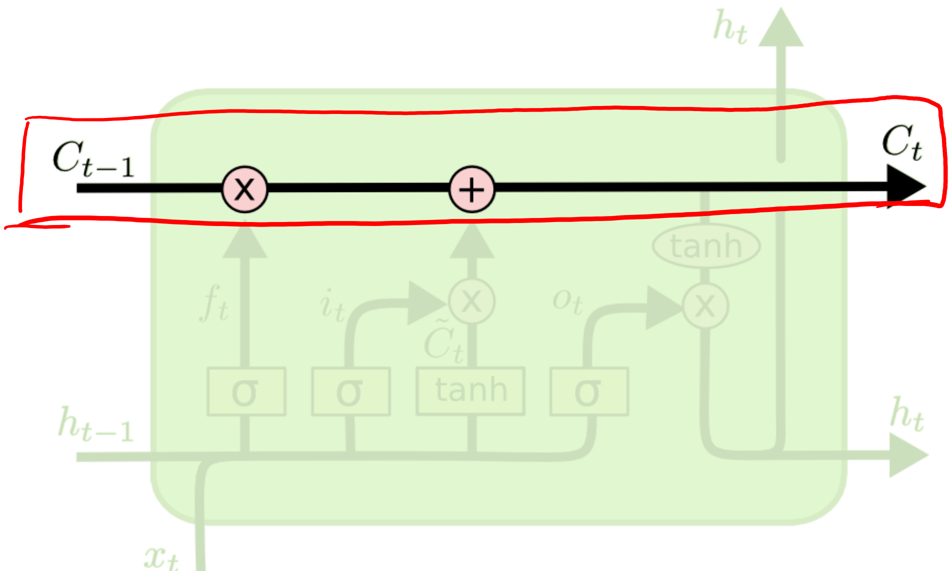
LSTMs Intuition: Memory

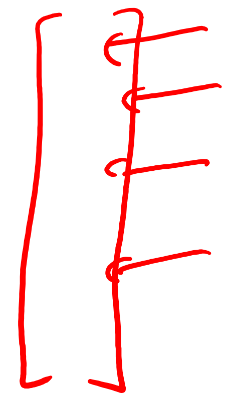
- Cell State / Memory

C_t : h_t hidden state

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

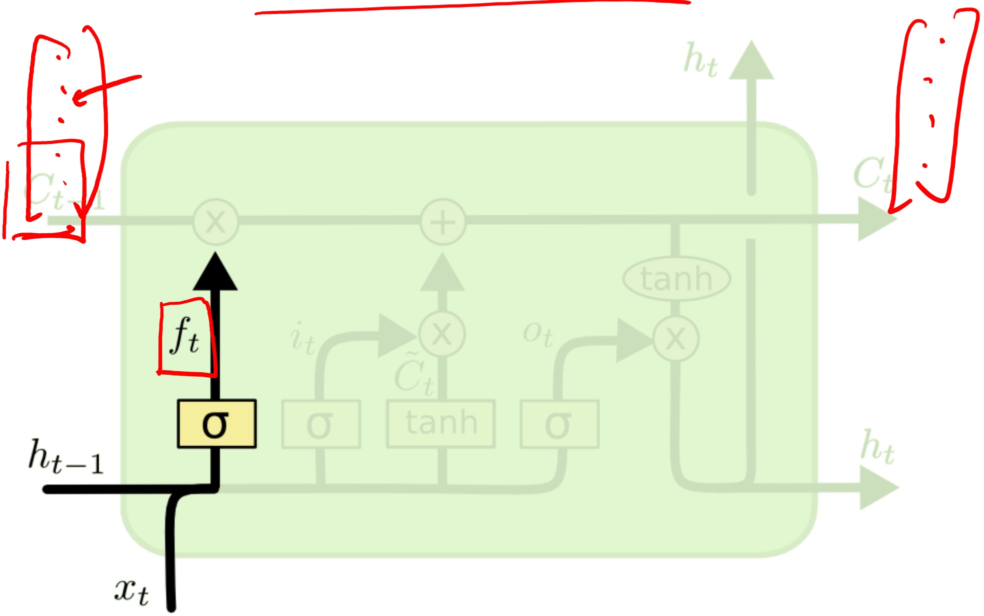
$h_t \in \mathbb{R}^d$





LSTMs Intuition: Forget Gate

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



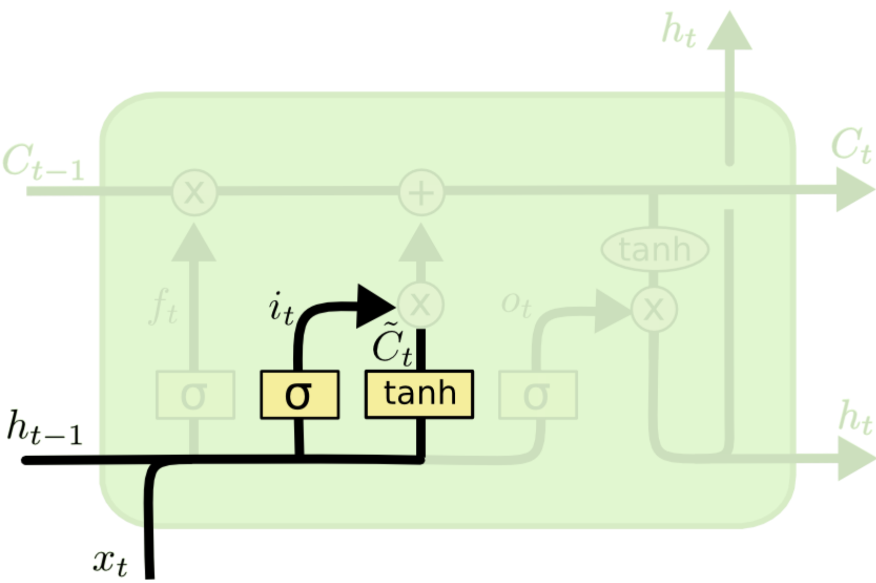
$f_t \in \mathbb{R}^d$

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

LSTMs Intuition: Input Gate

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

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



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

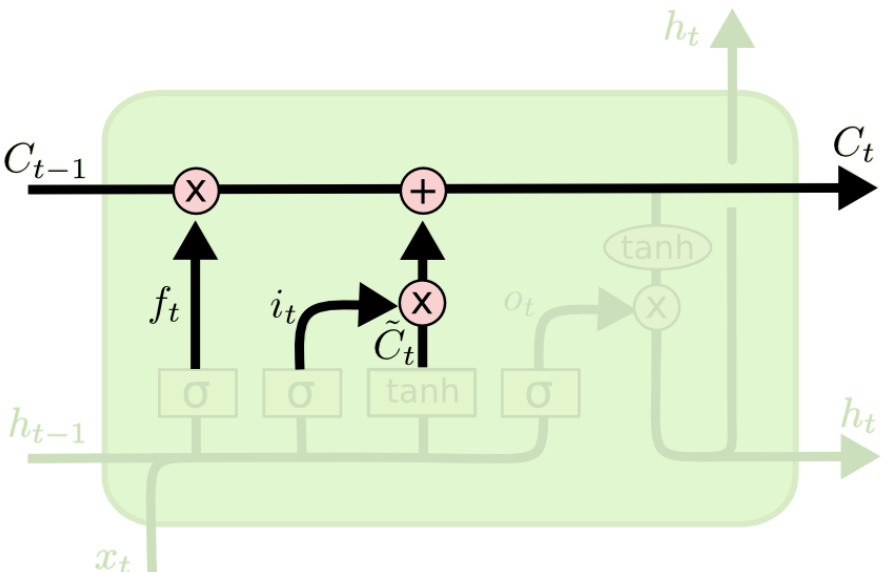
$$\boxed{\tilde{C}_t} = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$\tilde{C}$

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

LSTMs Intuition: Memory Update

- Forget that + memorize this



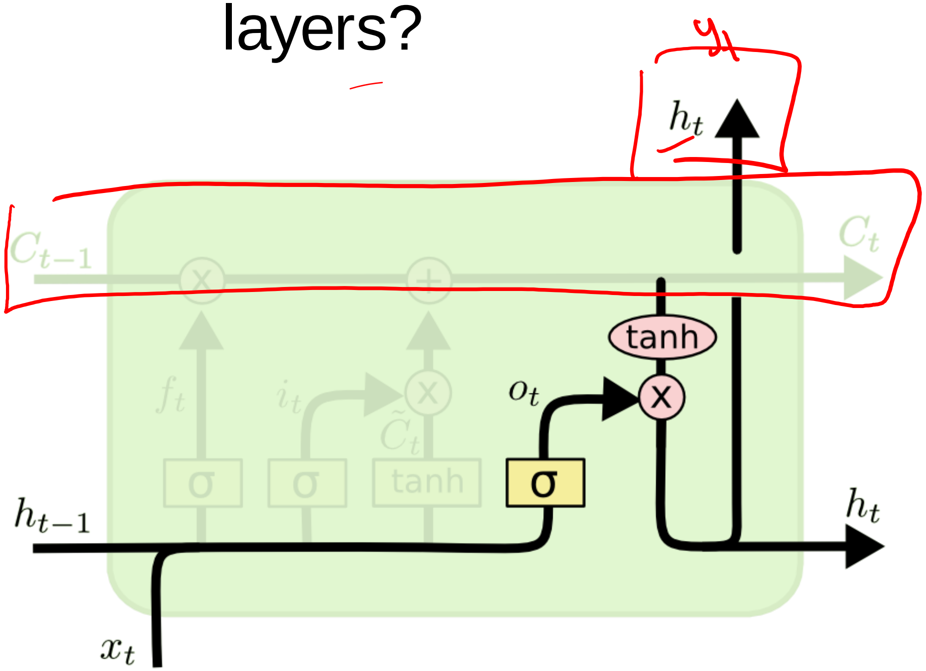
$$C_t = \boxed{f_t} * C_{t-1} + \boxed{i_t} * \tilde{C}_t$$

forget

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

LSTMs Intuition: Output Gate

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

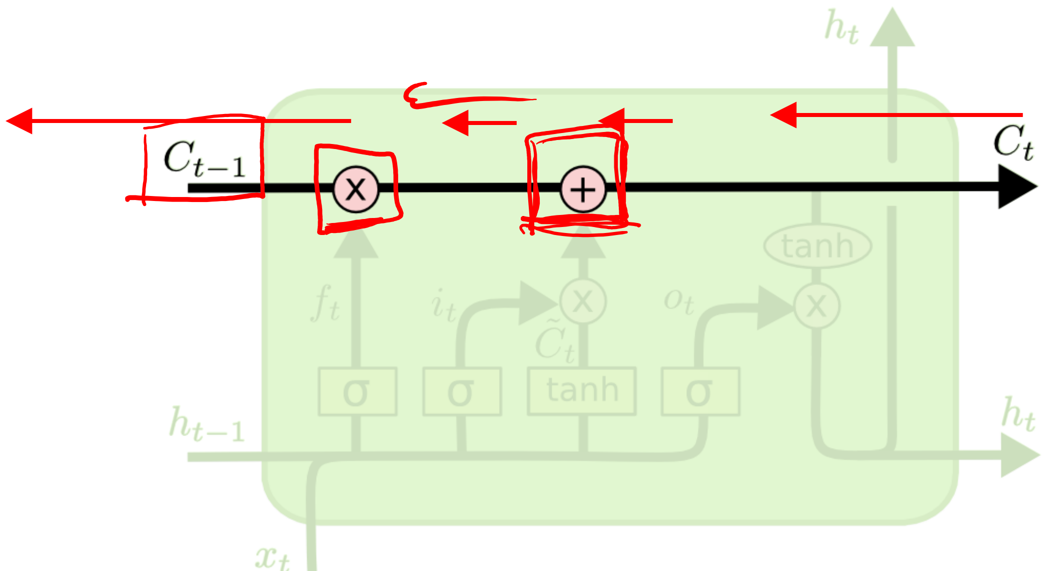


$o_t \in \mathbb{R}^d$

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

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

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

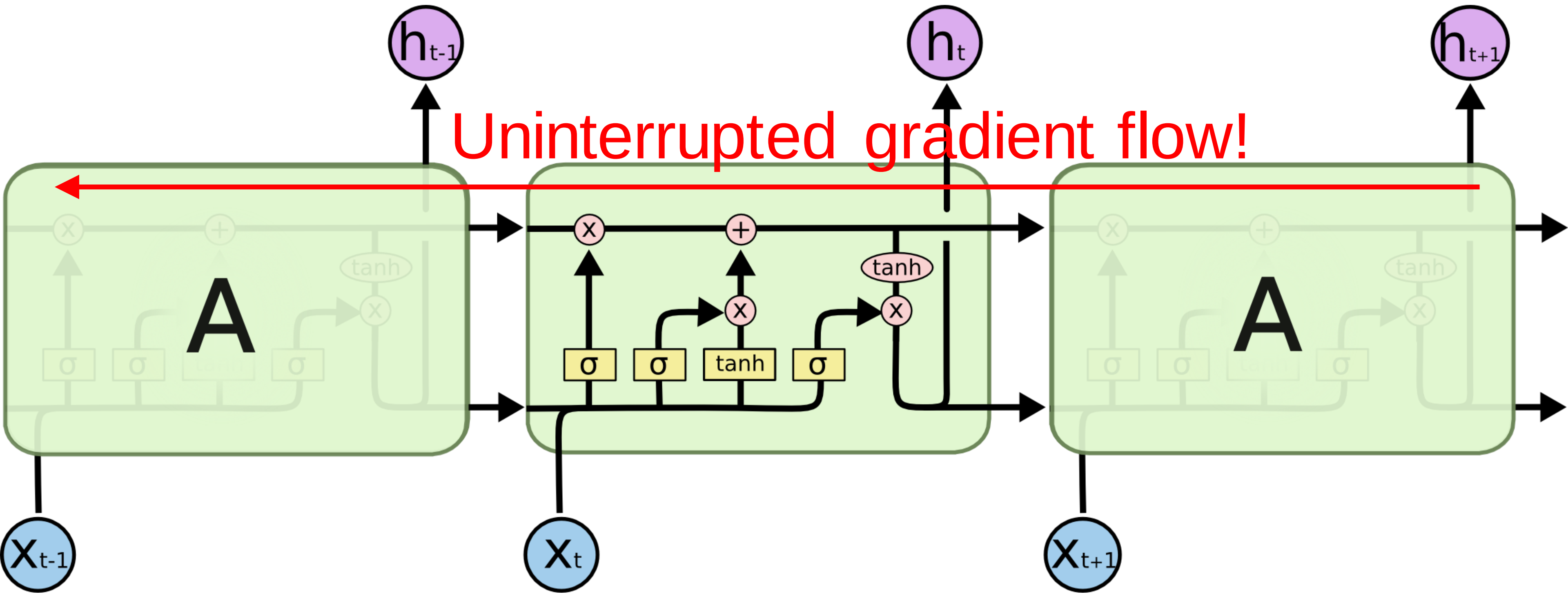
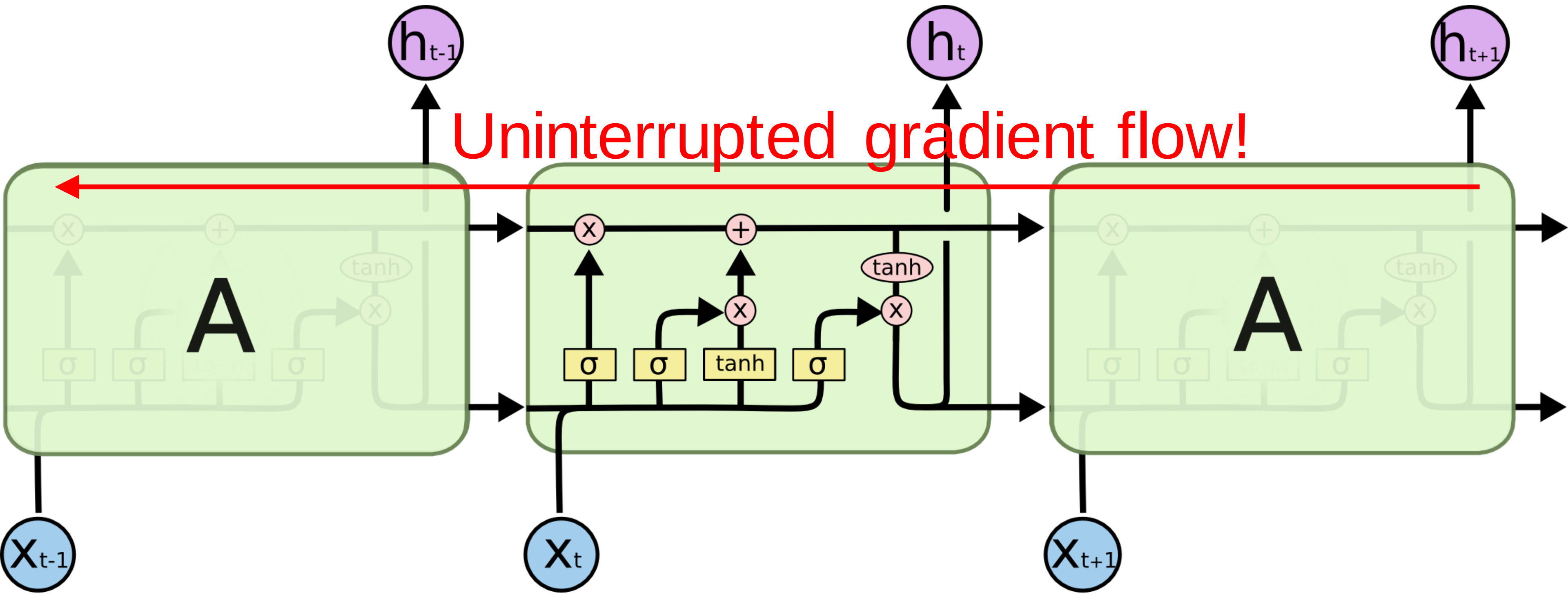


 Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Additive Updates



Similar to ResNet!

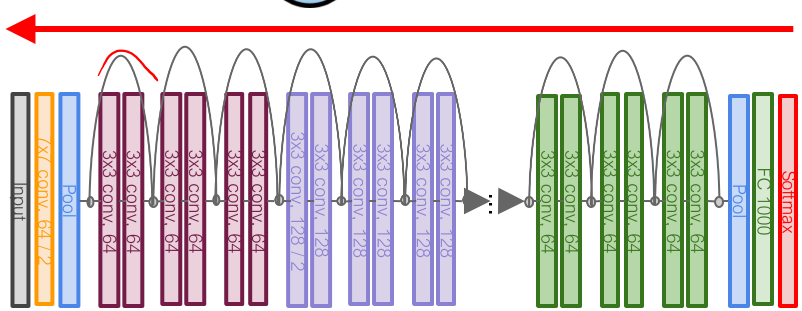


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs

- A pretty sophisticated cell

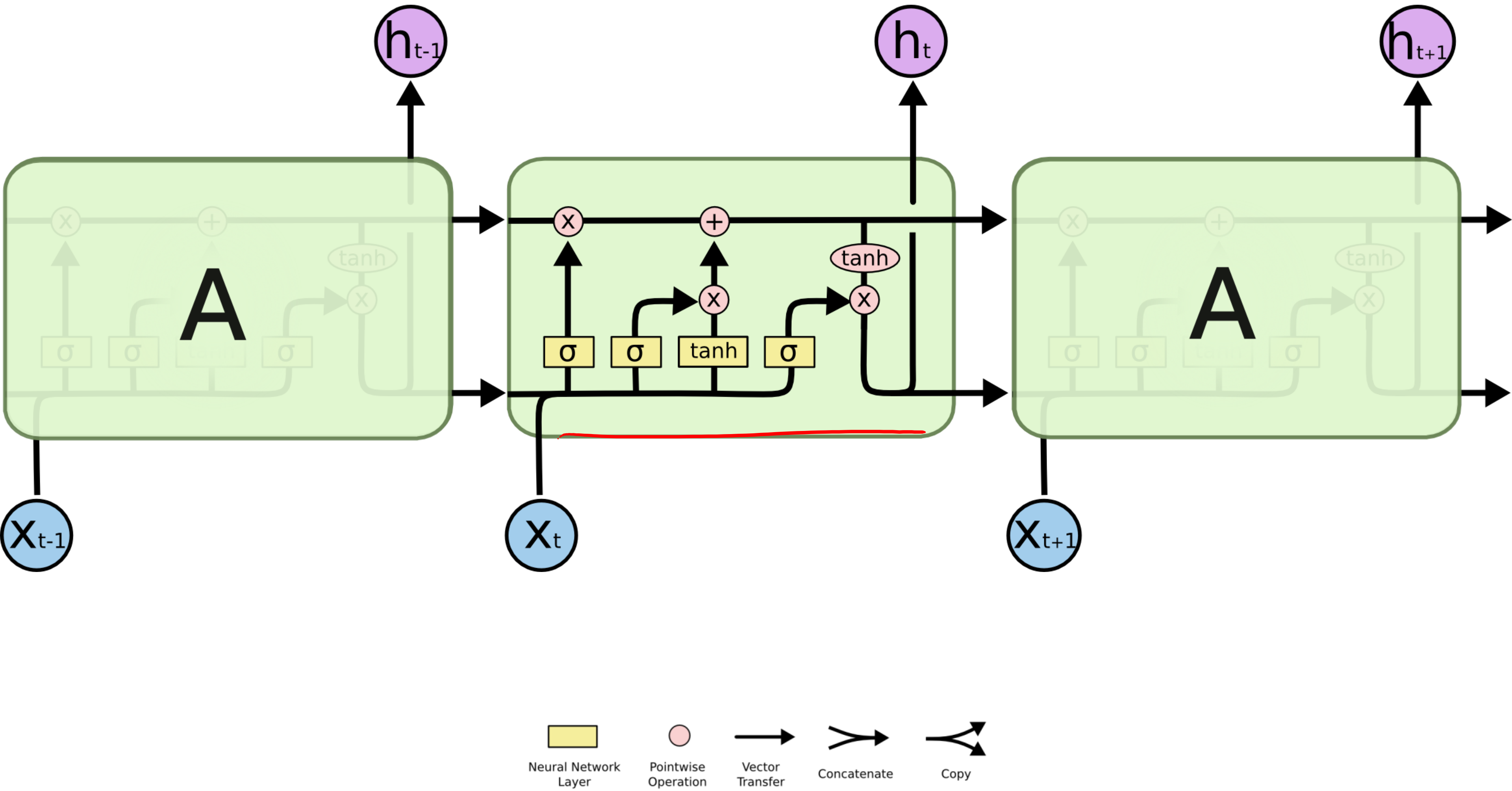
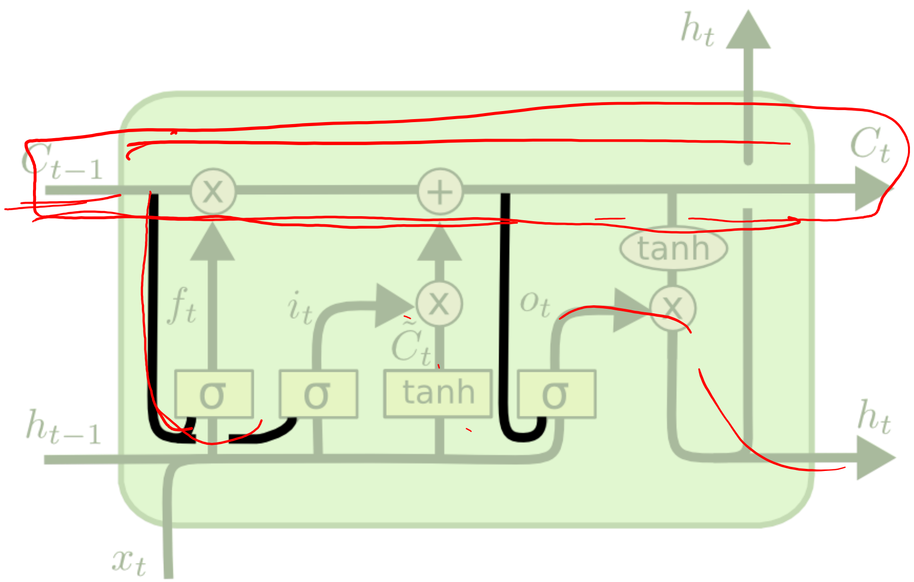


 Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTM Variants #1: Peephole Connections

- Let gates see the cell state / memory

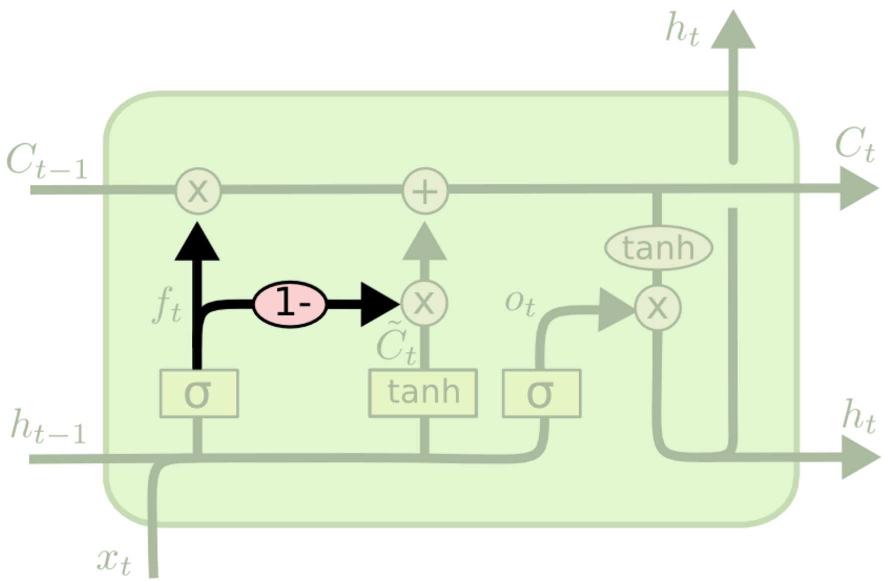


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] + b_o\right)$$

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

LSTM Variants #2: Coupled Gates

- Only memorize new if forgetting old

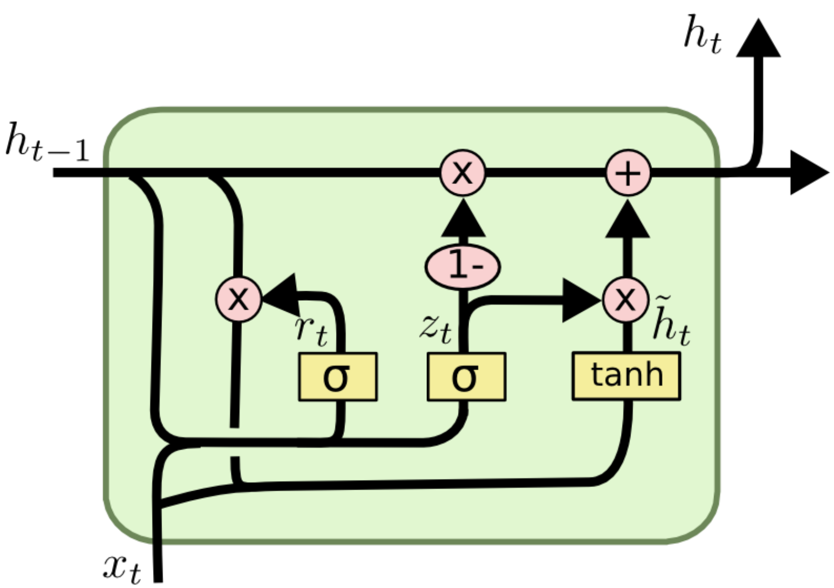


$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

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.

Administrativia

- Guest Lecture: Nirbhay Modhe
  - Reinforcement Learning

https://arjunmajum.github.io/