Advanced Machine Learning
Lecture 18

Recurrent Neural Networks II

23.01.2017

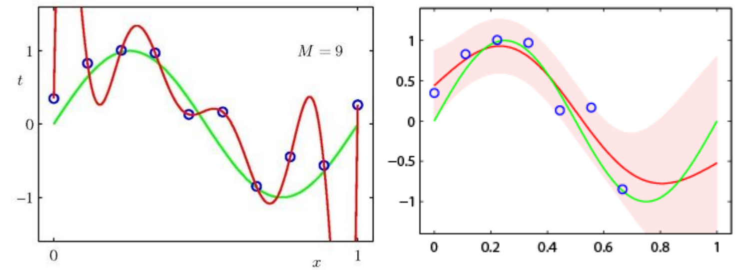
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

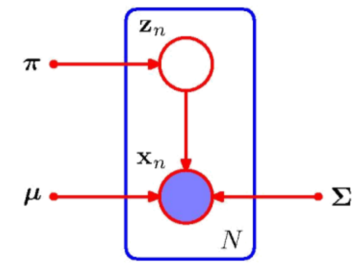
This Lecture: *Advanced Machine Learning*

- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes

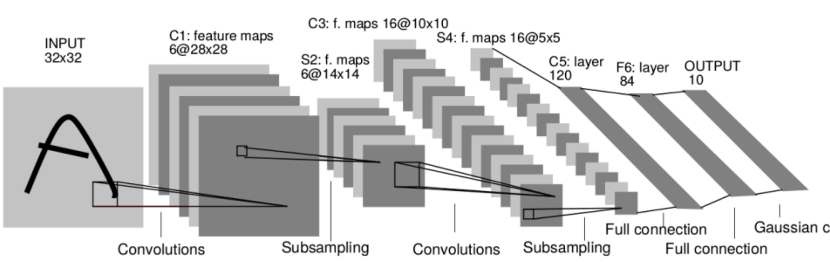
$$f : \mathcal{X} \to \mathbb{R}$$



- Approximate Inference
  - Sampling Approaches
  - MCMC



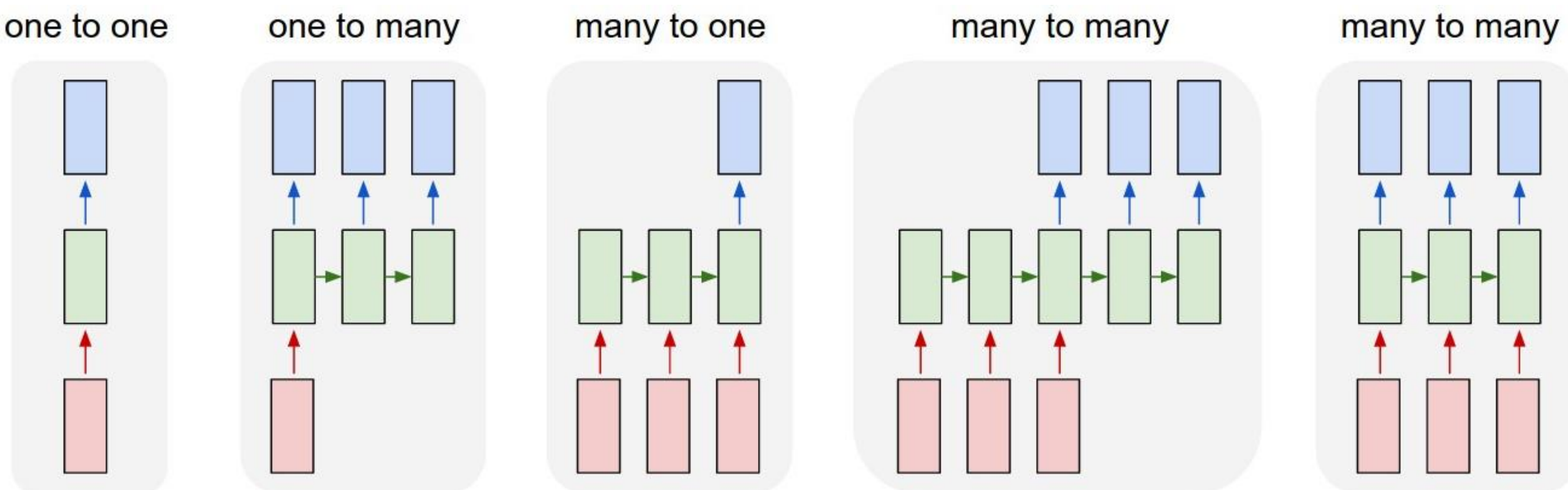
- Deep Learning
  - Linear Discriminants
  - Neural Networks
  - Backpropagation & Optimization
  - CNNs, ResNets, RNNs, etc.



Topics of This Lecture

- Recap: Recurrent Neural Networks (RNNs)
  - Backpropagation through Time (BPTT)
  - Problems with RNN Training
  - Handling Vanishing Gradients
- Improved hidden units for RNNs
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Units (GRU)
- Applications of RNNs

Recurrent Neural Networks

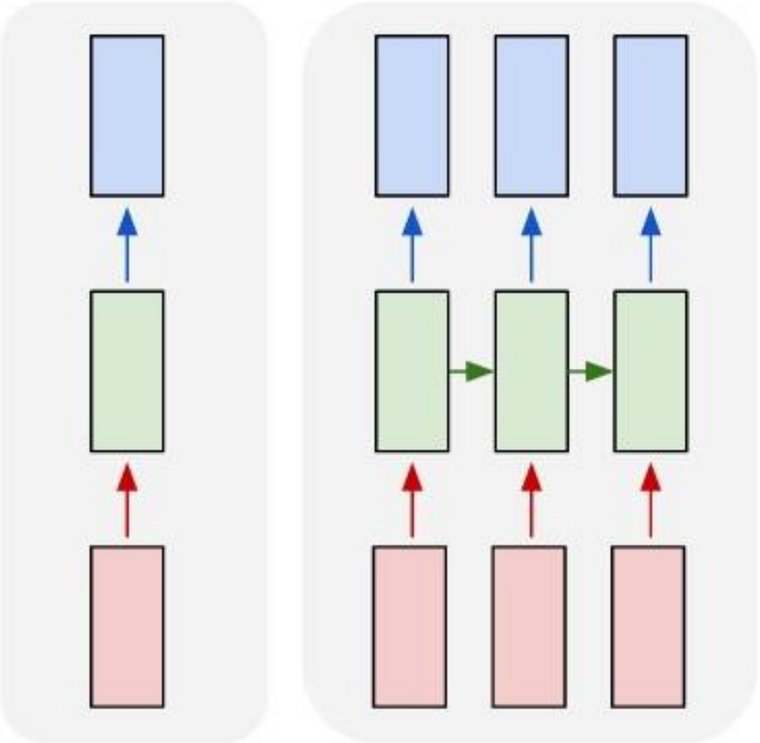


- **Up to now**
  - Simple neural network structure: 1-to-1 mapping of inputs to outputs
- **This lecture: Recurrent Neural Networks**
  - Generalize this to arbitrary mappings

Image source: Andrej Karpathy

Recap: Recurrent Neural Networks (RNNs)

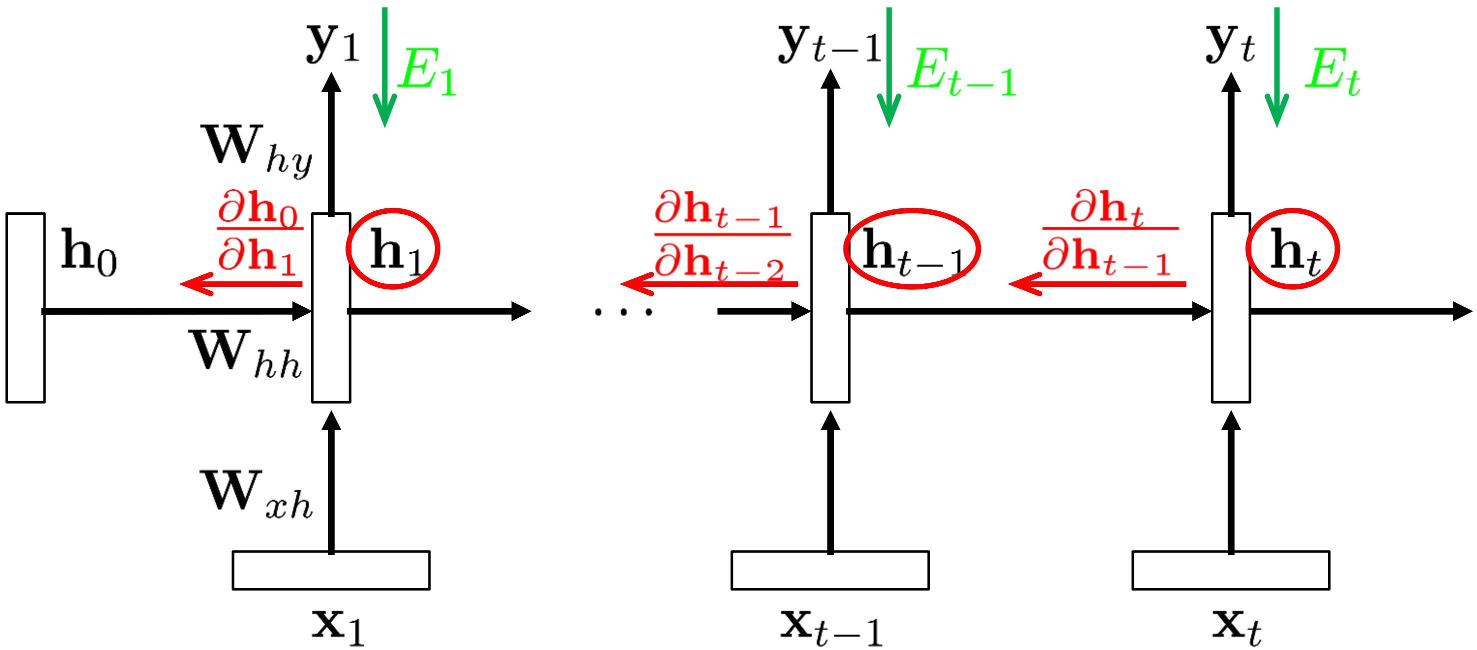
- RNNs are regular NNs whose hidden units have additional connections over time.
  - You can unroll them to create a network that extends over time.
  - When you do this, keep in mind that the weights for the hidden are shared between temporal layers.



- RNNs are very powerful
  - With enough neurons and time, they can compute anything that can be computed by your computer.

Image source: Andrej Karpathy

Recap: Backpropagation Through Time (BPTT)



- **Configuration**

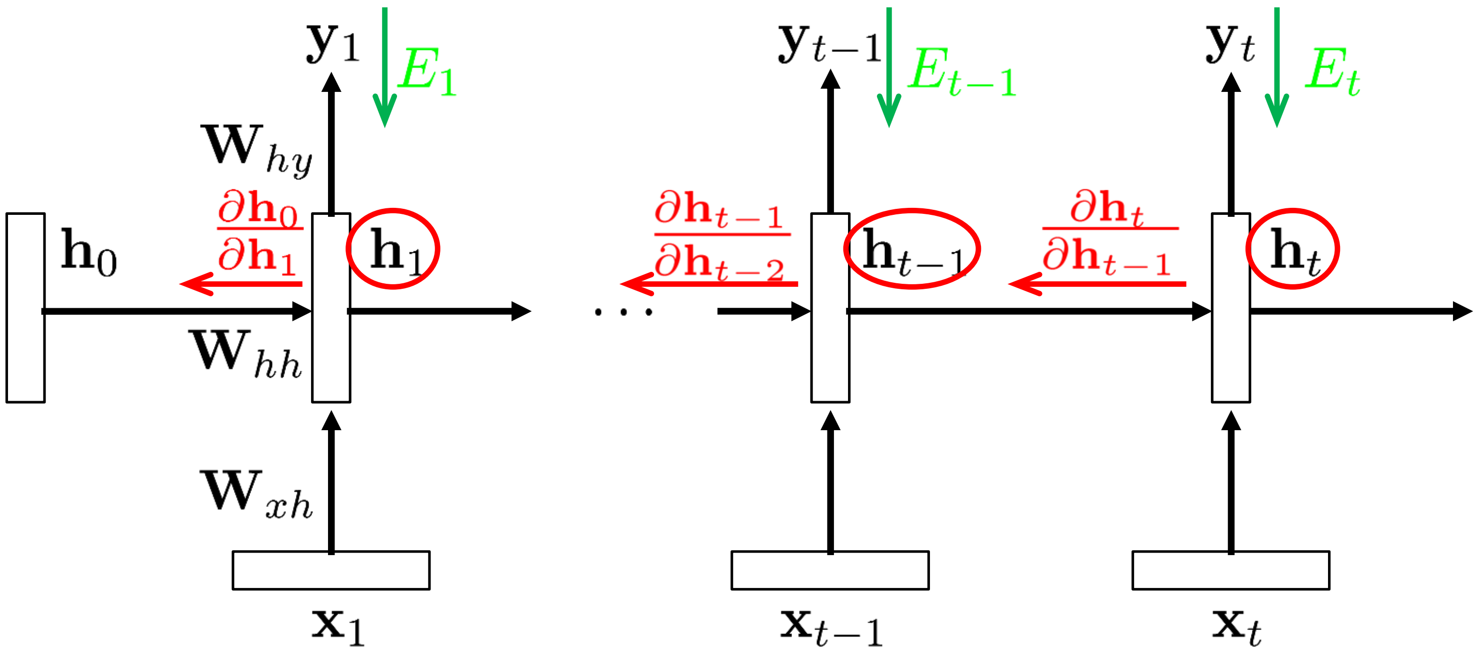
$$\mathbf{h}_t = \sigma\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + b\right)$$

$$\hat{\mathbf{y}}_t = \text{softmax}\left(\mathbf{W}_{hy}\mathbf{h}_t\right)$$

- **Backpropagated gradient**
  - For weight w_{ij}:

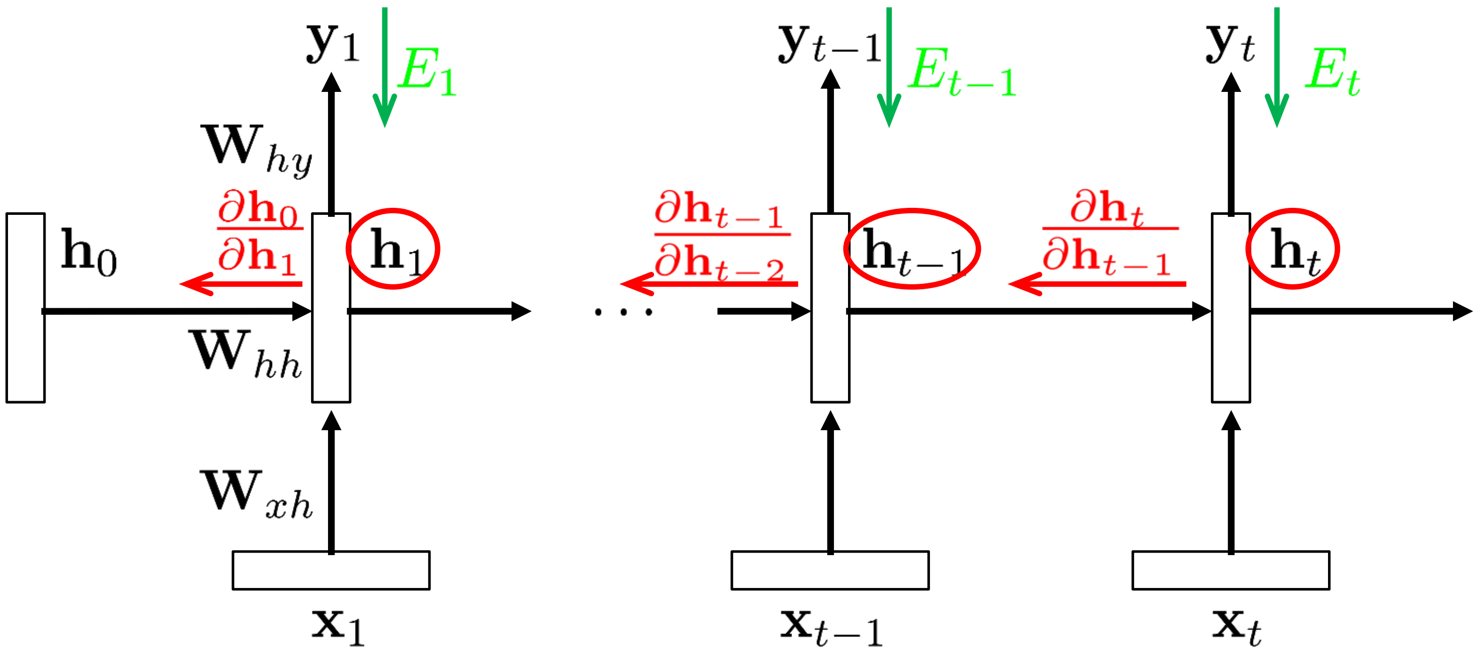
$$\frac{\partial E_t}{\partial w_{ij}} = \sum_{1\le k\le t}\left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial^+ h_k}{\partial w_{ij}}\right)$$

Recap: Backpropagation Through Time (BPTT)



- **Analyzing the terms**
  - For weight w_{ij}:
  $$\frac{\partial E_t}{\partial w_{ij}} = \sum_{1 \le k \le t} \left(\frac{\partial E_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial^+ h_k}{\partial w_{ij}} \right)$$
  - This is the "immediate" partial derivative (with $\mathbf{h}_{k\text{-}1}$ as constant)

Recap: Backpropagation Through Time (BPTT)



- **Analyzing the terms**
  - For weight w_{ij}:
  $$\frac{\partial E_t}{\partial w_{ij}} = \sum_{1 \leq k \leq t} \left(\frac{\partial E_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial^+ h_k}{\partial w_{ij}} \right)$$
  - Propagation term:
  $$\frac{\partial h_t}{\partial h_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

Recap: Exploding / Vanishing Gradient Problem

- **BPTT equations:**

$$\frac{\partial E_t}{\partial w_{ij}} = \sum_{1 \leq k \leq t} \left(\frac{\partial E_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial^+ h_k}{\partial w_{ij}} \right)$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{hh}^\top diag\left(\sigma'(\mathbf{h}_{i-1})\right)$$

$$= \left(\mathbf{W}_{hh}^\top\right)^l$$

(if t goes to infinity and $l = t - k$.)

$\Rightarrow$ We are effectively taking the weight matrix to a high power.

- The result will depend on the eigenvalues of $\mathbf{W}_{hh}$.
  - Largest eigenvalue > 1 $\Rightarrow$ Gradients *may* explode.
  - Largest eigenvalue < 1 $\Rightarrow$ Gradients *will* vanish.
  - This is very bad...

Recap: Gradient Clipping

- **Trick to handle exploding gradients**
  - **If the gradient is larger than a threshold, clip it to that threshold.**

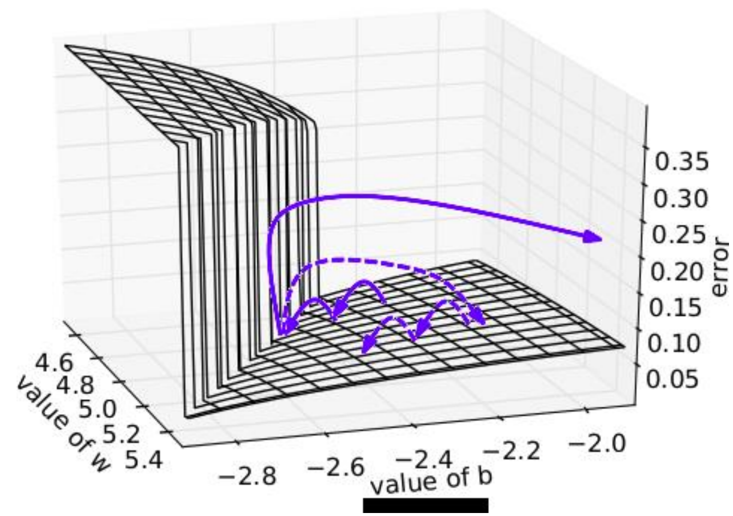
Algorithm 1 Pseudo-code

$$\hat{g} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$$

if $\|\hat{g}\| \geq threshold$ **then**

$$\hat{g} \leftarrow \frac{threshold}{\|\hat{g}\|} \hat{g}$$

end if



  - **This makes a big difference in RNNs**

Handling Vanishing Gradients

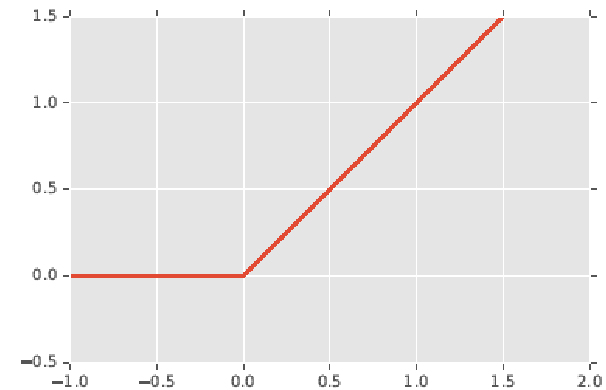
- Vanishing Gradients are a harder problem
  - They severely restrict the dependencies the RNN can learn.
  - The problem gets more severe the deeper the network is.
  - It can be very hard to diagnose that Vanishing Gradients occur (you just see that learning gets stuck).

- Ways around the problem
  - Glorot/He initialization (more on that in Lecture 21)
  - ReLU
  - More complex hidden units (LSTM, GRU)

ReLU to the Rescue

- **Idea**
  - **Initialize $\mathbf{W}_{hh}$ to identity matrix**
  - **Use Rectified Linear Units (ReLU)**

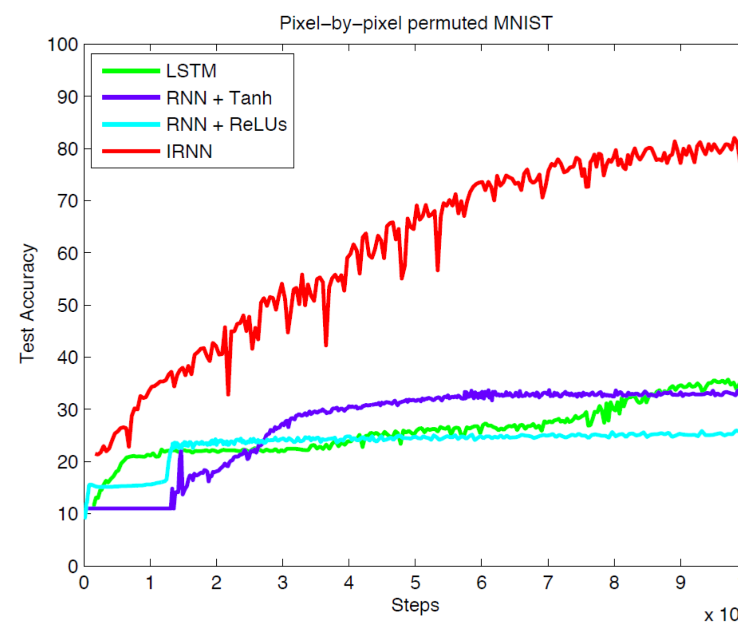
$$g(a) = \max\{0, a\}$$



- **Effect**
  - **The gradient is propagated with a constant factor**

$$\frac{\partial g(a)}{\partial a} = \begin{cases} 1, & a > 0 \\ 0, & \text{else} \end{cases}$$

  - **⇒Huge difference in practice!**



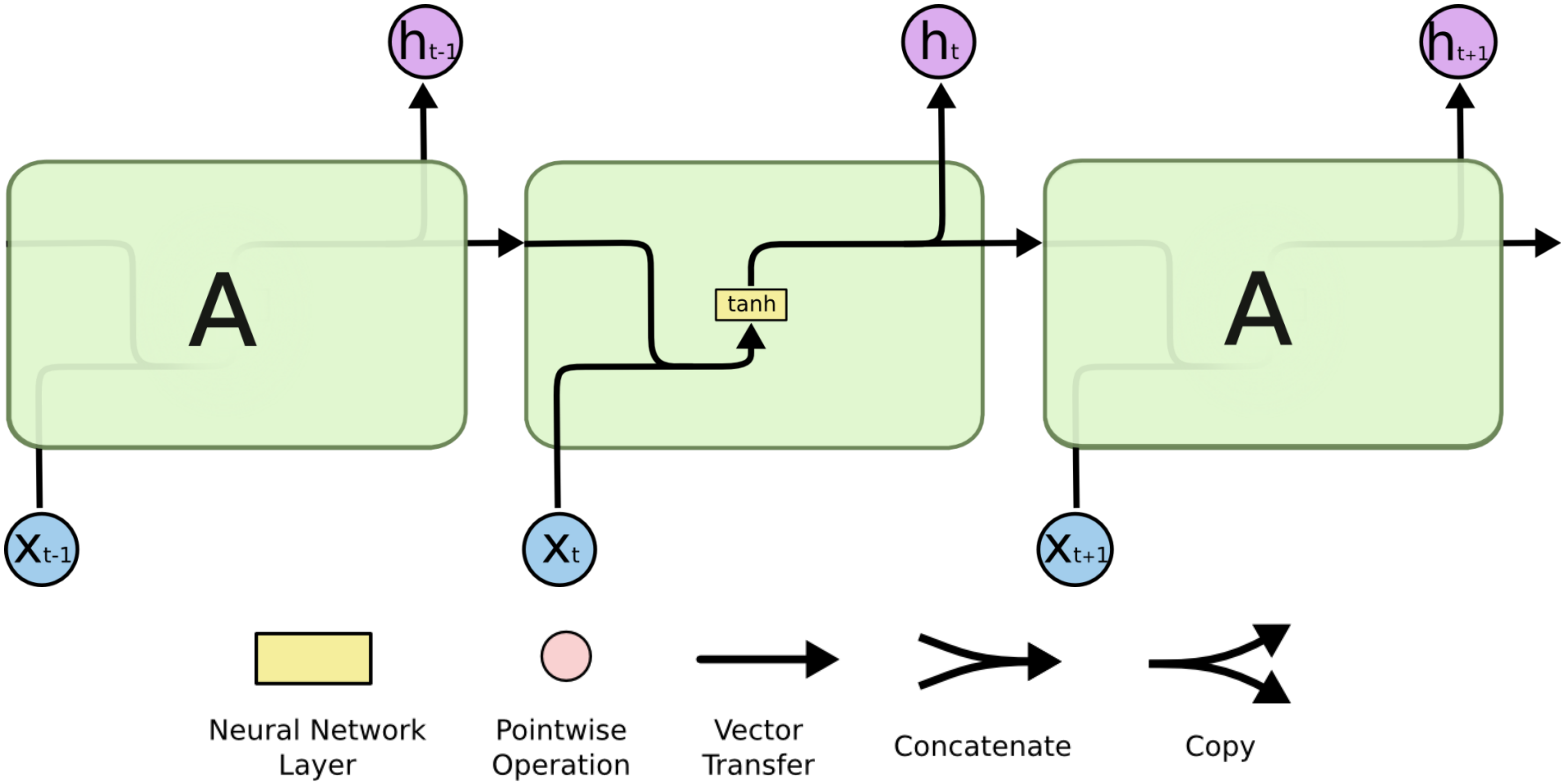
Topics of This Lecture

- Recap: Recurrent Neural Networks (RNNs)
  - Backpropagation through Time (BPTT)
  - Problems with RNN Training
  - Handling Vanishing Gradients
- **Improved hidden units for RNNs**
  - **Long Short-Term Memory (LSTM)**
  - **Gated Recurrent Units (GRU)**
- Applications of RNNs

More Complex Hidden Units

- Target properties
  - Want to achieve constant error flow through a single unit
  - At the same time, want the unit to be able to pick up long-term connections or focus on short-term ones, as the problem demands.
- Ideas behind LSTMs
  - Take inspiration from the design of memory cells
  - Keep around memories to capture long distance dependencies
  - Allow error messages to flow at different strengths depending on the inputs

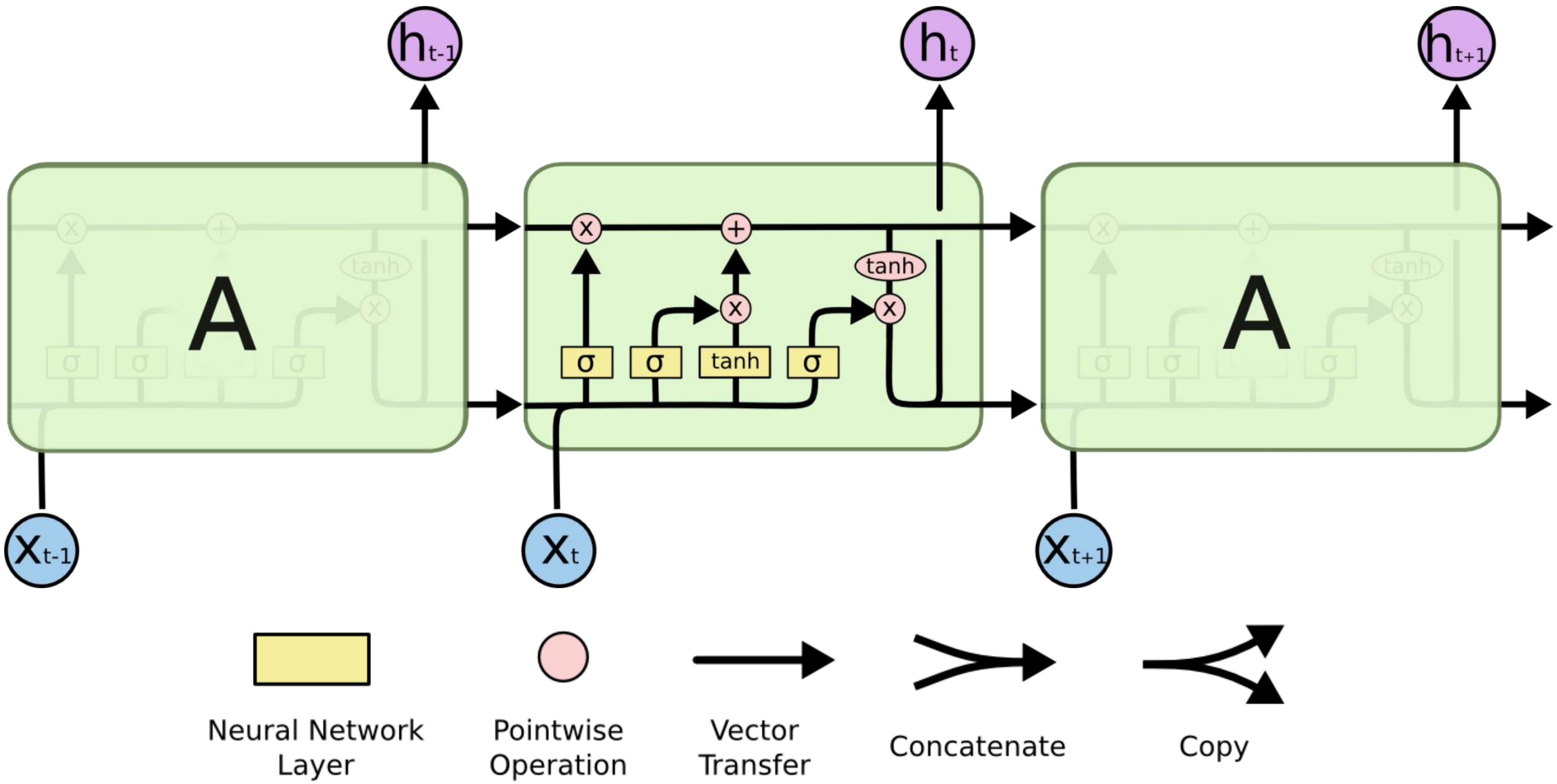
Long Short-Term Memory



- **RNNs can be seen as chains of repeating modules**
  - **In a standard RNN, the repeating module has a very simple structure (e.g., a tanh)**

Image source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory

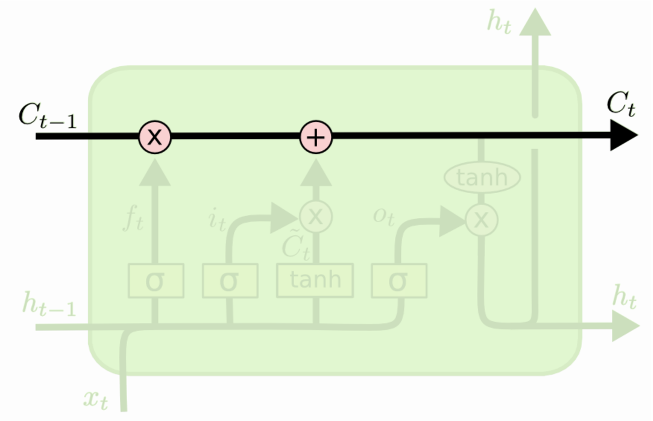


- **LSTMs**
  - **Repeating modules have 4 layers, interacting in a special way.**

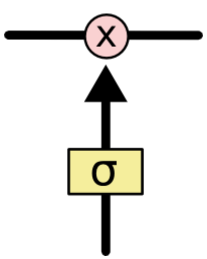
Image source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs: Core Ideas

- Cell state
  - This is the key to LSTMs.
  - It acts like a conveyor belt, information can flow along it unchanged.



- Gates
  - The cell state can be modified through gates.
  - Structure: sigmoid net layer + pointwise multiplication
  - The sigmoid outputs values between 0 and 1
    - 0: Let nothing through
    - 1: Let everything through
  - The gate layers are learned together with all other parameters.



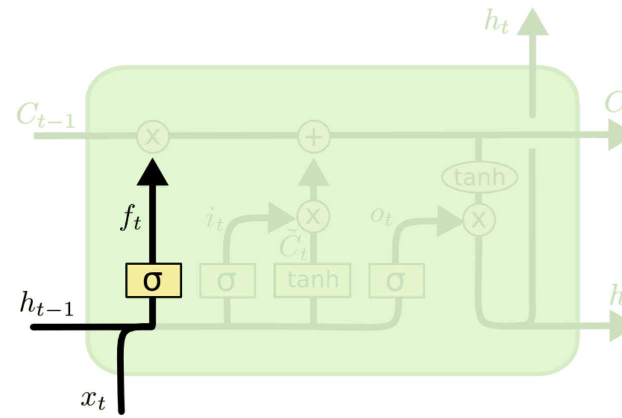
Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Elements of LSTMs

- **Forget gate** layer
  - Look at $\mathbf{h}_{t\text{-}1}$ and $\mathbf{x}_t$ and output a number between 0 and 1 for each dimension in the cell state $\mathbf{C}_{t\text{-}1}$.

    0: completely delete this,

    1: completely keep this.



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

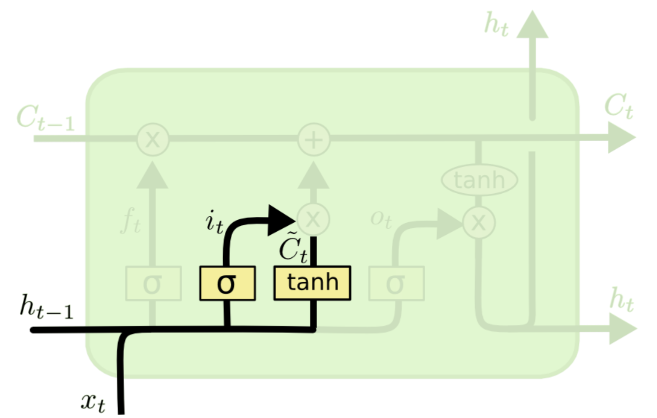
- **Example**
  - Task: try to predict the next word
  - Cell state could include the gender of the present subject

  ⇒ When we see a new subject, want to forget the gender of the old subject.

Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Elements of LSTMs

- **Update gate** layer
  - Decide what information to store in the cell state.
  - Sigmoid network (**input gate** layer) decides which values are updated.
  - tanh layer creates a vector of new candidate values $\tilde{C}_t$ that could be added to the state.



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

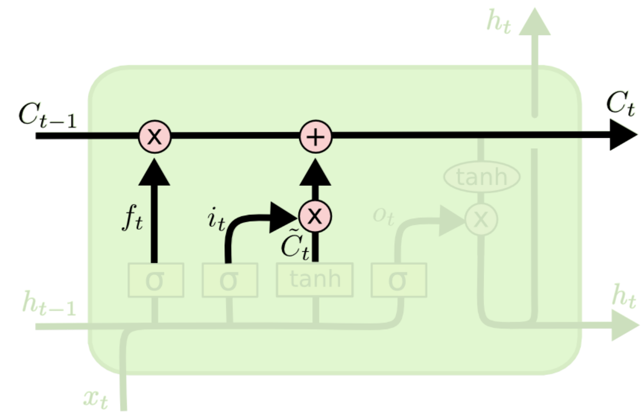
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- In the example
  - Add the gender of the new subject to the cell state.

Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Elements of LSTMs

- **Updating the state**
  - **Multiply the old state by f_t, forgetting the things we decided to forget.**
  - **Then add $i_t * \tilde{C}_t$, the new candidate values, scaled by how much we decided to update each value.**



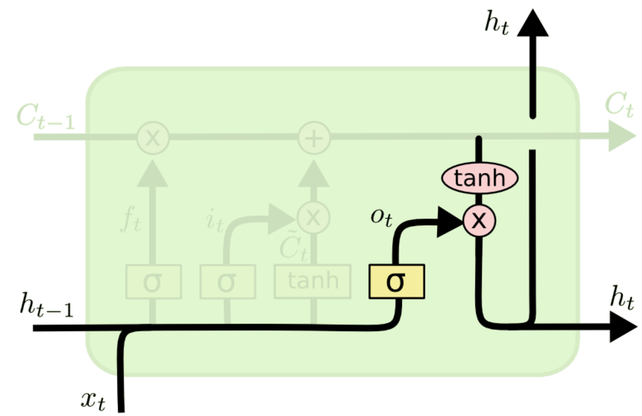
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- **In the example**
  - **Combined effect: replace the old gender by the new one.**

Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Elements of LSTMs

- **Output gate layer**
  - Output is a filtered version of our gate state.
  - First, apply sigmoid layer to decide what parts of the cell state to output.
  - Then, pass the cell state through a tanh (to push the values to be between -1 and 1) and multiply it with the output of the sigmoid gate.



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

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

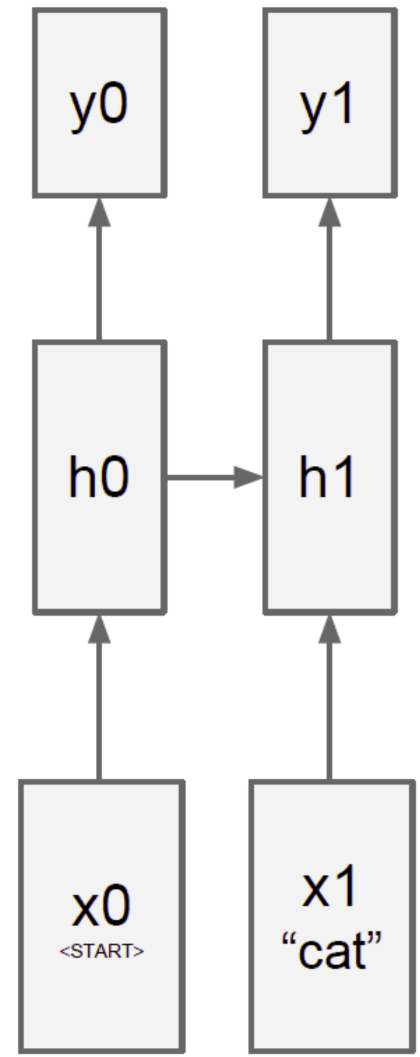
- **In the example**
  - Since we just saw a subject, might want to output information relevant to a verb (e.g., whether the subject is singular or plural).

Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN vs. LSTM

- LSTM just changes the form of the equation for h such that:
  1. More expressive multiplicative interactions become possible
  2. Gradients flow nicer
  3. The network can explicitly decide to reset the hidden state

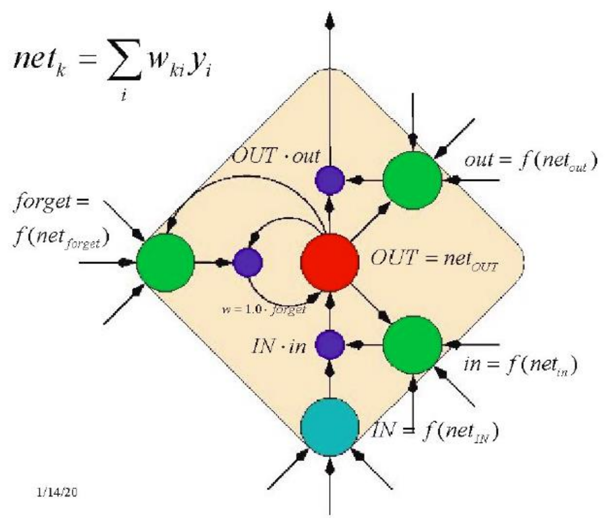
- Those changes have a huge effect in practice
  - LSTMs perform much better than regular RNNs
  - Many applications have become possible with LSTMs that weren't feasible before.

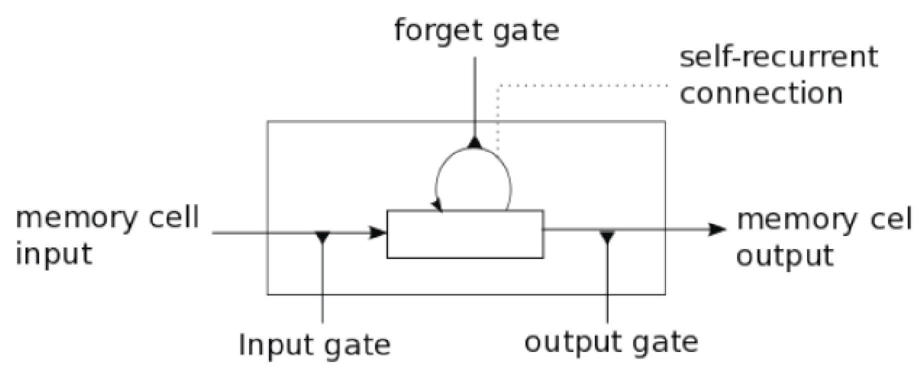


LSTMs in Practice

- LSTMs are currently highly en vogue
  - Popular default model for most sequence labeling tasks.
  - Very powerful, especially when stacked and made even deeper.
  - Most useful if you have lots and lots of data.
  - ⇒ Very active research field

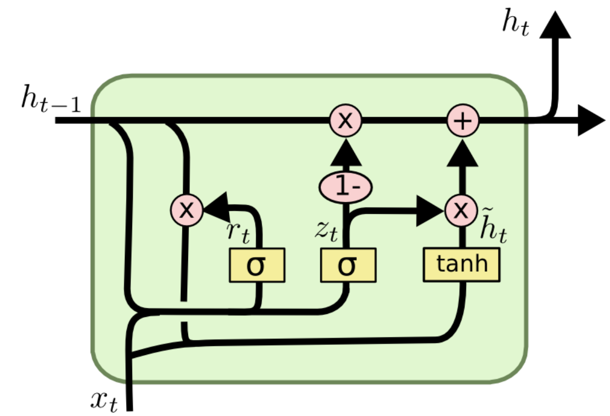
- Here are also some other ways of illustrating them





Extension: Gated Recurrent Units (GRU)

- **Simpler model than LSTM**
  - Combines the forget and input gates into a single update gate z_t.
  - Similar definition for a reset gate r_t, but with different weights.
  - In both cases, merge the cell state and hidden state.



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

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

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

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

- **Empirical results**
  - Performance similar to LSTM (no clear winner yet)
  - But GRU has fewer parameters.

Source: Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRUs: Intuition

- **Effects**
  - **If reset is close to 0, ignore previous hidden state.**

  $\Rightarrow$ Allows model to drop information that is irrelevant in the future.

  - **Update gate z controls how much of past state should matter now.**

  $\Rightarrow$ If z is close to 0, then we can copy information in that unit through many time steps!

  $\Rightarrow$ Less vanishing gradients!

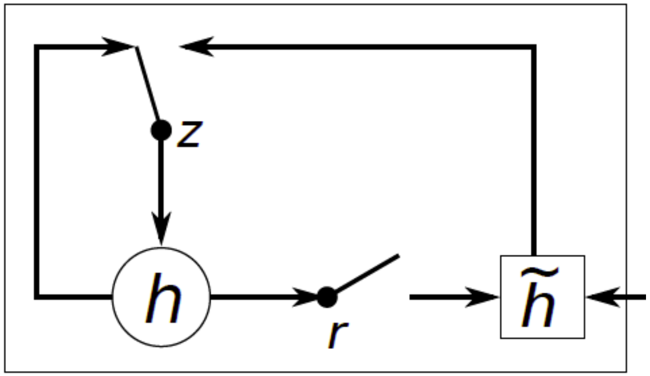
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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

GRUs: Intuition



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

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

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

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

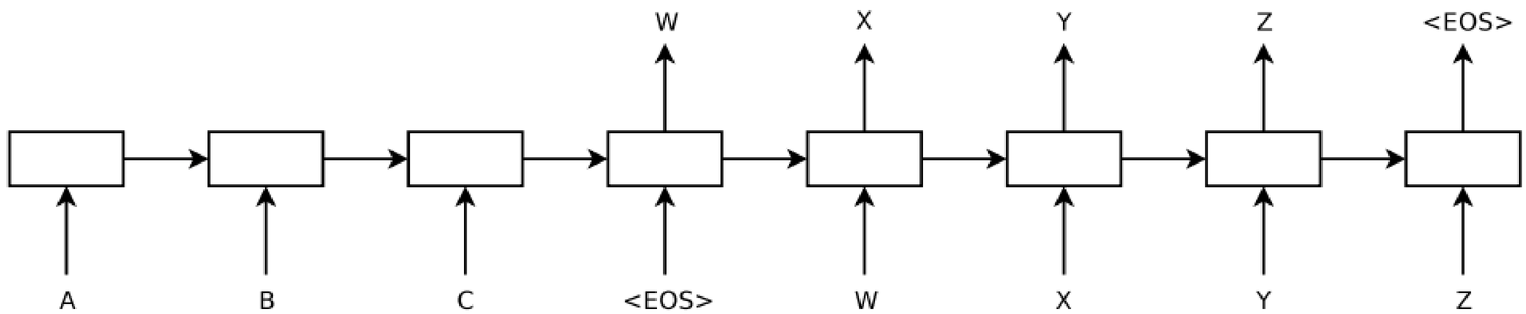
- **Typical learned behaviors**
  - **Units with short-term dependencies often have active reset gate**
  - **Units with long-term dependencies have inactive update gates.**

Topics of This Lecture

- Recap: Recurrent Neural Networks (RNNs)
  - Backpropagation through Time (BPTT)
  - Problems with RNN Training
  - Handling Vanishing Gradients
- Improved hidden units for RNNs
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Units (GRU)
- **Applications of RNNs**

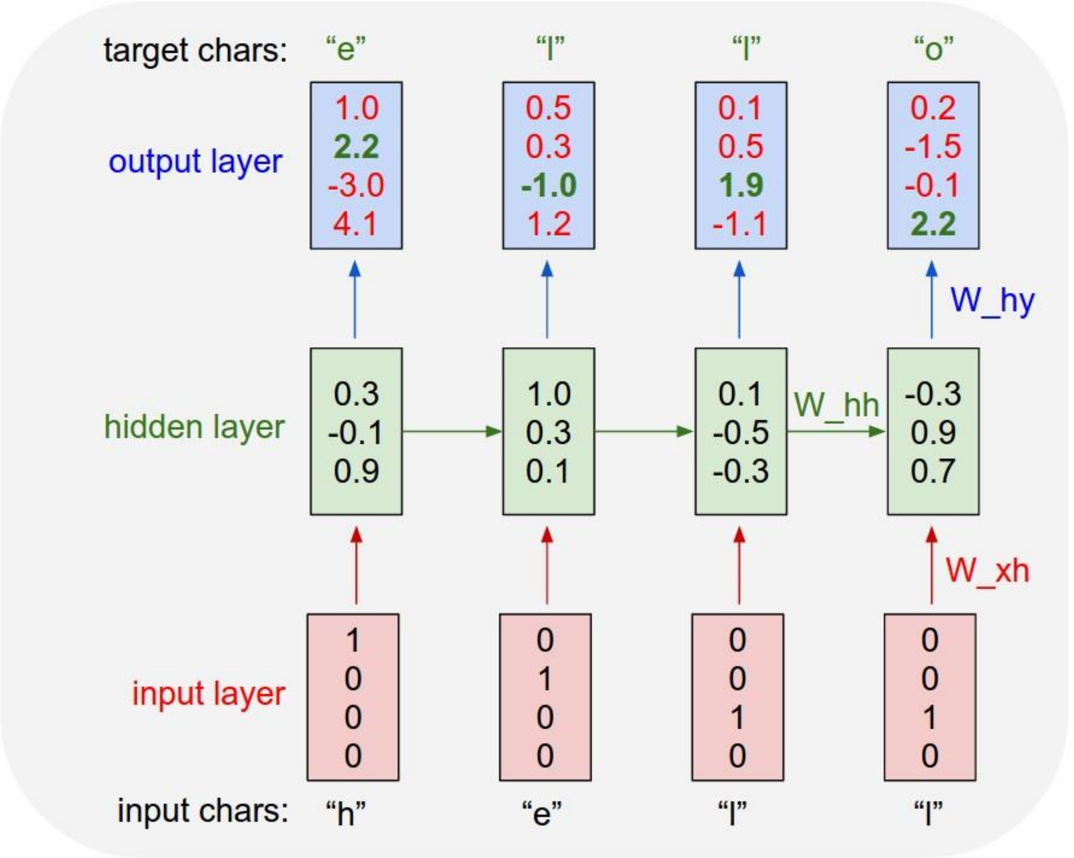
Applications

- **Machine Translation [Sutskever et al., 2014]**



Application: Character-Level Language Model

- **Setup**
  - RNN trained on huge amounts of text
  - Task: model the prob. distribution of the next character in the sequence.



- **Main advantage of RNN here**
  - RNN can learn varying amount of context

Language Model Results

```
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.
```

- **Example: Generating Shakespeare**
  - **Trained on all works of Shakespeare (4.4 MB of data)**
  - **Using a 3-Layer RNN with 512 hidden units per layer**

Language Model Results

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

- **Example: Generating Wikipedia pages**
  - **Trained on 100MB of Wikipedia data**
  - **Using an LSTM**

Language Model Results

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

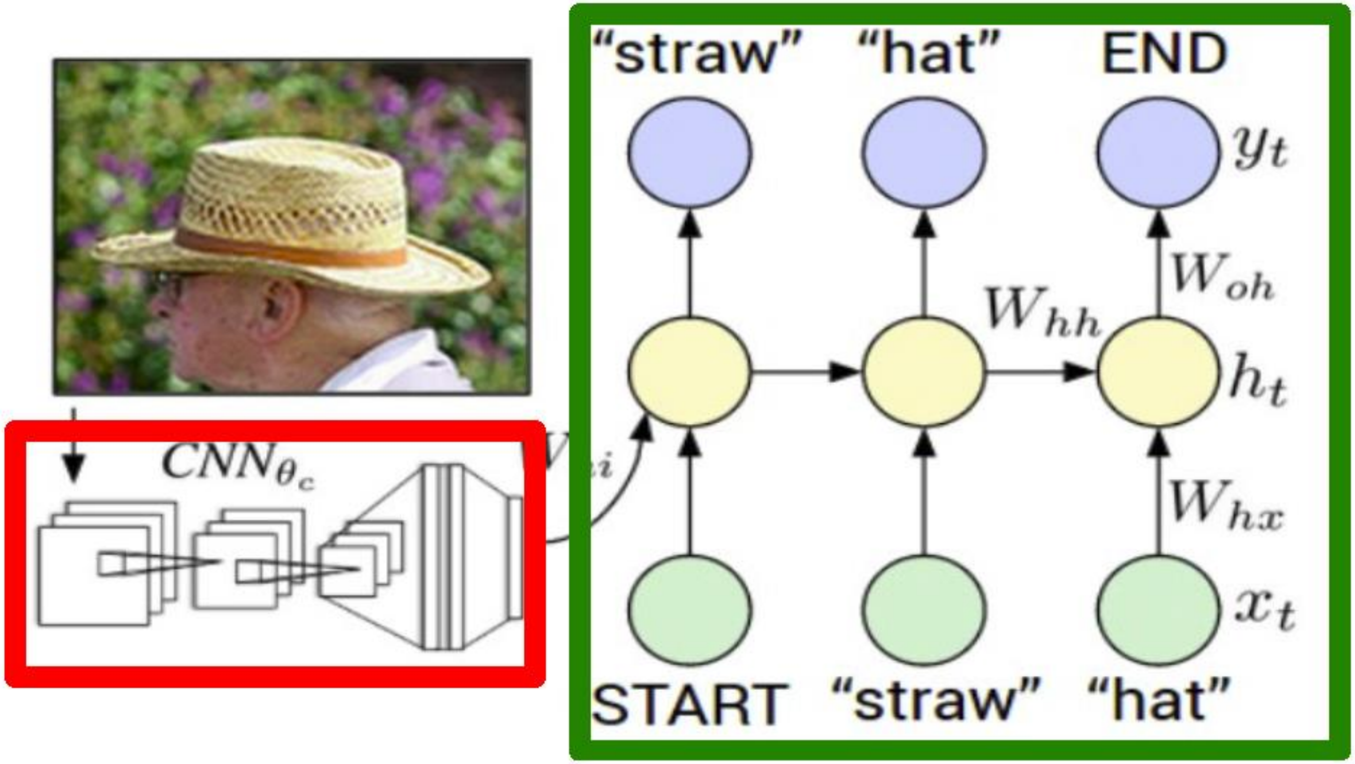
- **Example: Hallucinating Algebraic Geometry**
  - **Trained on an Algebraic Geometry book**
  - **Using a multilayer LSTM**

Language Model Results

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
```

- **Example: Hallucinating C Code**
  - Trained on the Linux source code (474MB from github)
  - Using a large 3-layer LSTM

Applications: Image Tagging



- **Simple combination of CNN and RNN**
  - **Use CNN to define initial state h_0 of an RNN.**
  - **Use RNN to produce text description of the image.**

Applications: Image Tagging

- **Setup**
  - Train on corpus of images with textual descriptions
  - E.g. Microsoft CoCo
    - 120k images
    - 5 sentences each

a man riding a bike on a dirt path through a forest.
bicyclist raises his fist as he rides on desert dirt trail.
this dirt bike rider is smiling and raising his fist in triumph.
a man riding a bicycle while pumping his fist in the air.
a mountain biker pumps his fist in celebration.

Results: Image Tagging

a group of people standing around a room with remotes
logprob: -9.17

a young boy is holding a baseball bat
logprob: -7.61

a cow is standing in the middle of a street
logprob: -8.84

Spectacular results!

Results: Image Tagging

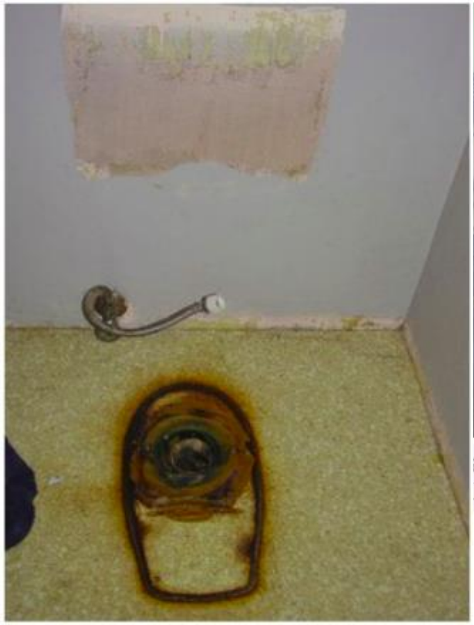
a baby laying on a bed with a stuffed bear
logprob: -8.66

a young boy is holding a baseball bat
logprob: -7.65

a cat is sitting on a couch with a remote control
logprob: -12.45

- **Wrong, but one can still see why those results were selected...**

Results: Image Tagging



a toilet with a seat up in a bathroom
logprob: -13.44

a woman holding a teddy bear in front of a mirror
logprob: -9.65

a horse is standing in the middle of a road
logprob: -10.34

- **Not sure what happened here...**

Fun Application: Image to Story

Later on the eighth day , Billy was a friend of a man who lived on his own . He did n't know how far away they were , and if he was to survive the fall . His mind raced , trying not to show any signs of weakness . The wind ruffled the snow and ice in the snow . He had no idea how many times he was going to climb into the mountains . He told me to stay on the ground for a while , but if I find out what s going on , we should go on foot . Sam and Si Lei joined us in the army .

- **Example: Generating a story from an image**
  - **Trained on corpus of adventure novels**

Source: Ryan Kiros http://www.cs.toronto.edu/~rkiros/adv_L.html

More Results

Having lain on the bed , I did n't know what to say . He turned his attention to the room and saw a large room . The room was furnished with a single bed , a dresser and a large bed with a table in the center of the room . It was a long time ago . The room was designed with the most powerful and efficient ones . As far as I m concerned , it was a long time ago . On the other side of the room was a beautiful picture of a woman who had been abducted by the fireplace and their own personal belongings in order to keep it safe , but it didn t take too long . Feeling helpless , he turned his attention back to me . ``

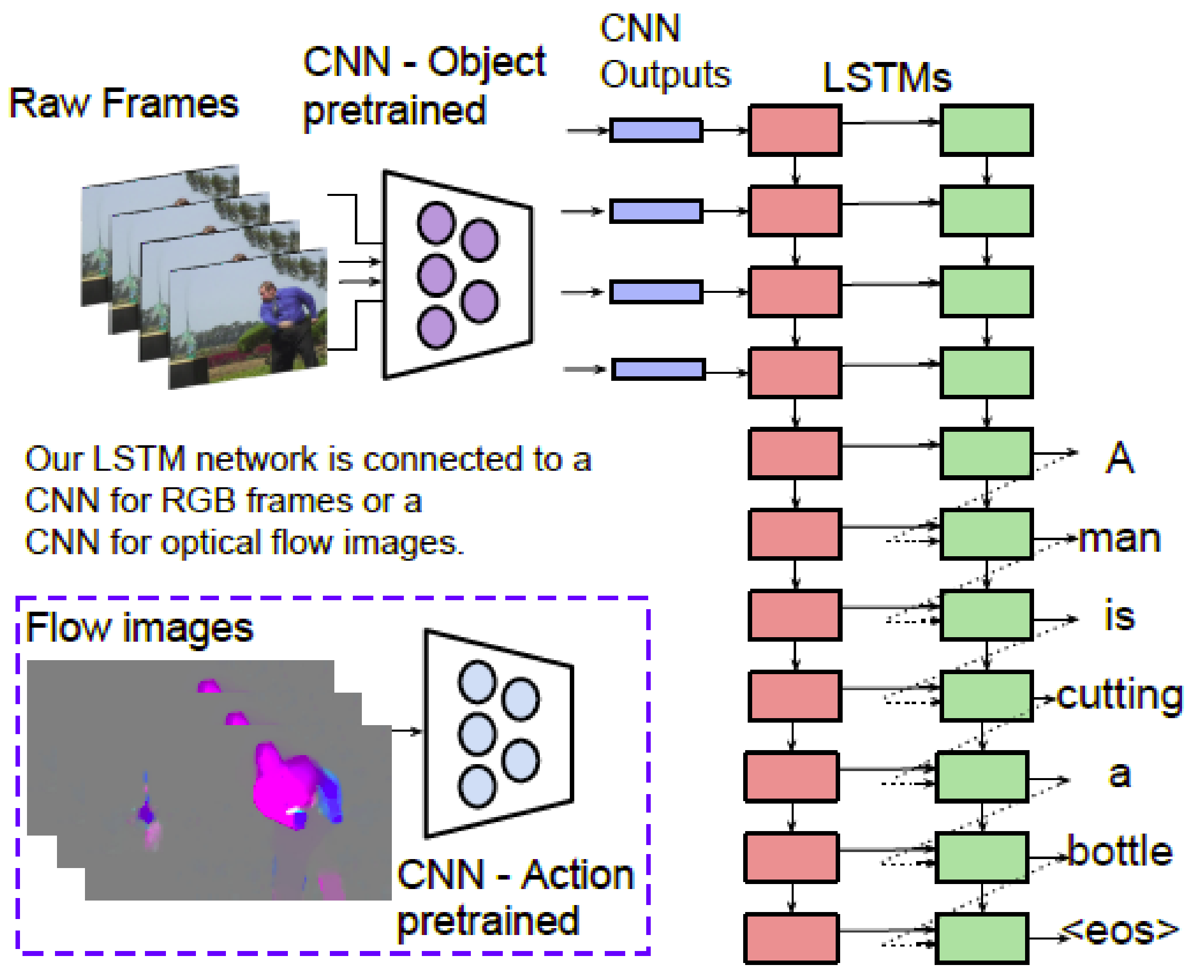
Source: Ryan Kiros http://www.cs.toronto.edu/~rkiros/adv_L.html

More Results

Only Prince Darin knew how to run from the mountains , and once more , he could see the outline of a rider on horseback . The wind ruffled his hair in an attempt to locate the forest . He hadn t been in such a state of mind before , but it was a good thing . All of them seemed to be doing the same thing . They did n't know where they came from . The wind blew up the mountain peaks and disappeared into the sky , leaving trails behind the peaks of the mountains on Mount Fuji .

Source: Ryan Kiros http://www.cs.toronto.edu/~rkiros/adv_L.html

Application: Video to Text Description



Source: Subhashini Venugopalan, ICCV'15

Video-to-Text Results

Source: Subhashini Venugopalan, ICCV'15

References and Further Reading

- RNNs
  - R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, JMLR, Vol. 28, 2013.
  - A. Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks, blog post, May 2015.
- LSTM
  - S. Hochreiter , J. Schmidhuber, Long short-term memory, Neural Computation, Vol. 9(8): 1735-1780, 1997.
  - A. Graves, Generating Sequences With Recurrent Neural Networks, ArXiV 1308.0850v5, 2014.
  - C. Olah, Understanding LSTM Networks, blog post, August 2015.