CPSC 540: Machine Learning

Recurrent Neural Networks

Winter 2018

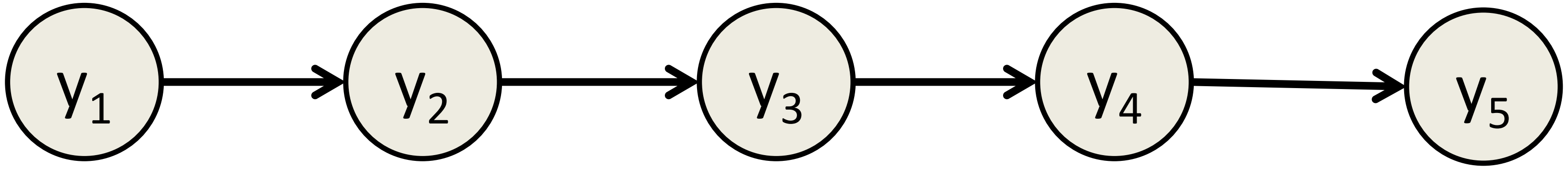
Last Time: Computer Vision CNN “Revolution”

- CNNs are now being used beyond image classification:

- Trend towards end-to-end systems:
  - Neural network does every step, backpropagation refines every step.
- Fully-convolutional networks (FCNs) are a common ingredient.
  - All layers are convolutions, including upsampling “transposed convolutions”.

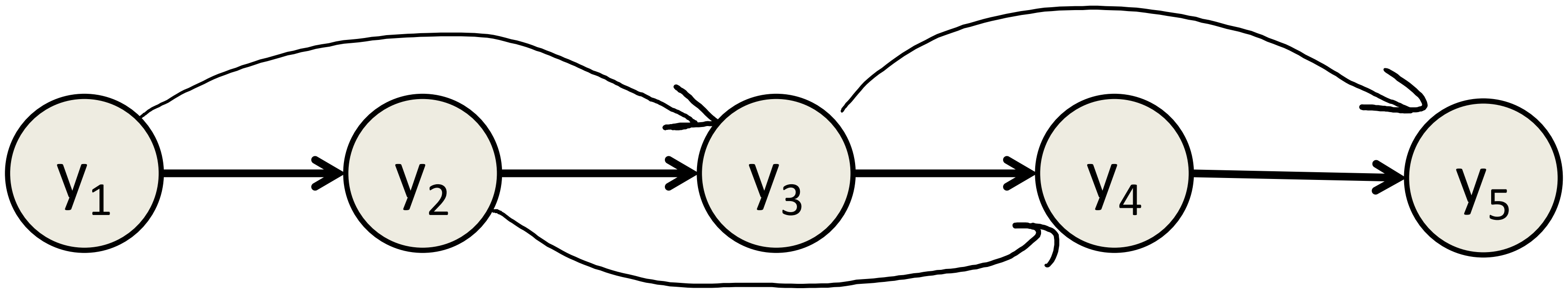
Motivation: Sequence Modeling

- We want to predict the next words in a sequence:
  - “I am studying to become a [?????????????????????????????]”.
- Simple idea: supervised learning to predict the next word.
  - Applying it repeatedly to generate the sequence.
- Simple approaches:
  - Markov chain (doesn’t work well, see “Garkov”).



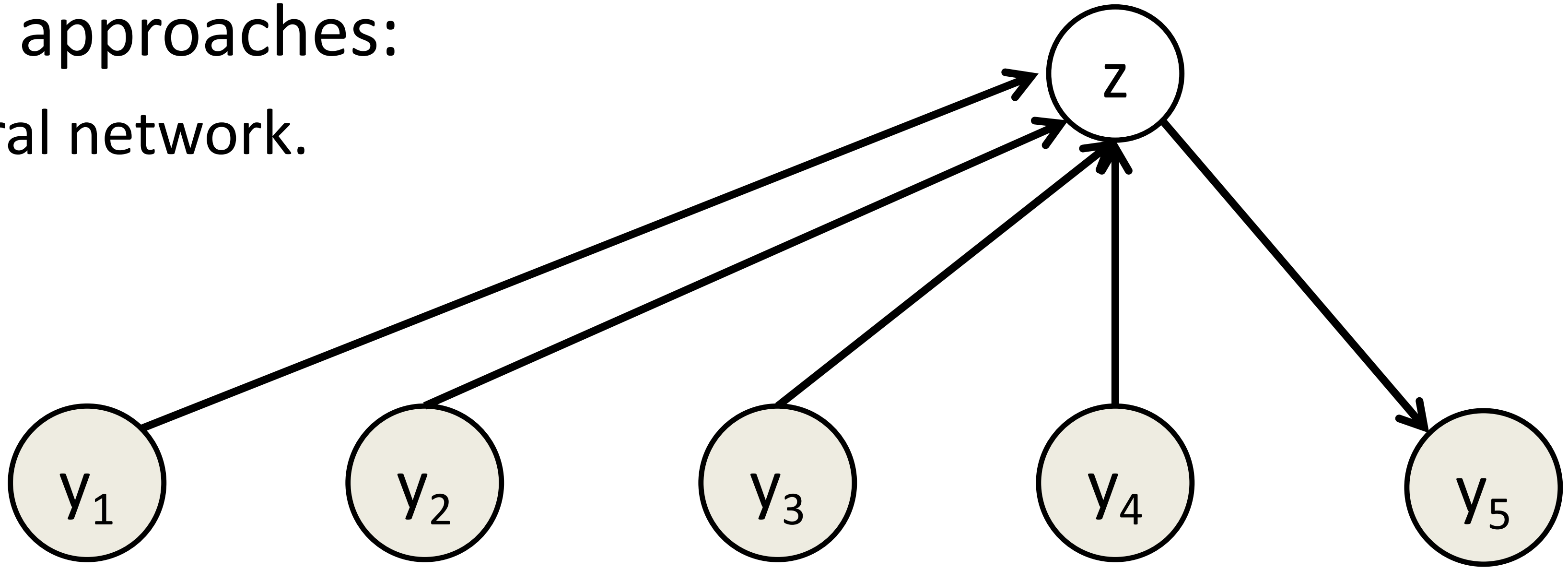
Motivation: Sequence Modeling

- We want to predict the next words in a sequence:
  - "I am studying to become a [????????????????????????????]".
- Simple idea: supervised learning to predict the next word.
  - Applying it repeatedly to generate the sequence.
- Simple approaches:
  - Higher-order Markov chain ("n-gram"):



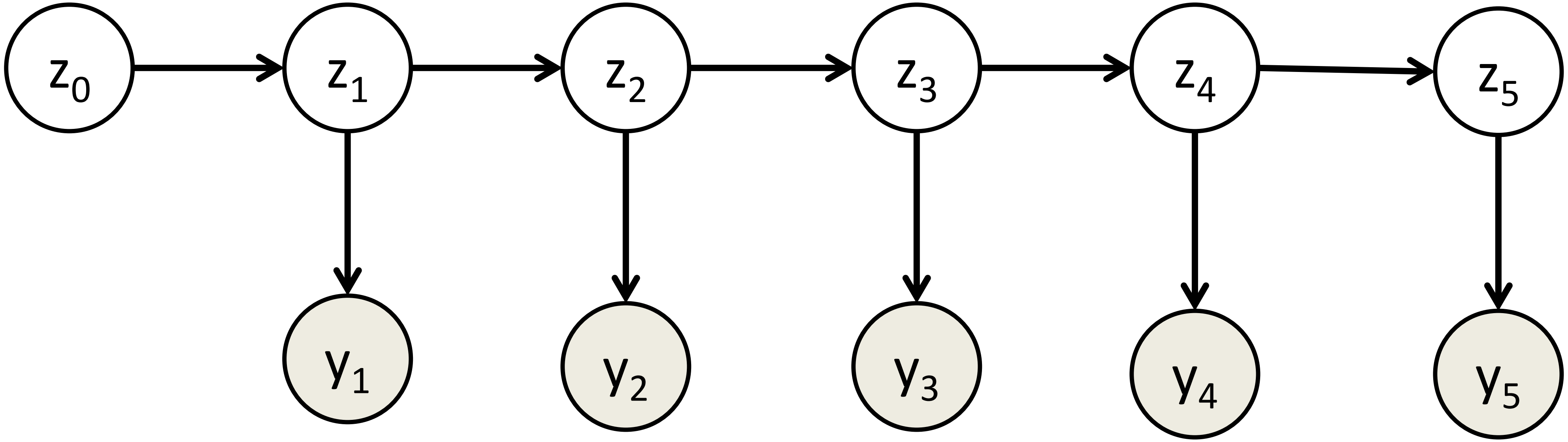
Motivation: Sequence Modeling

- We want to predict the next words in a sequence:
  - "I am studying to become a [??????????????????????????????]".
- Simple idea: supervised learning to predict the next word.
  - Applying it repeatedly to generate the sequence.
- Simple approaches:
  - Neural network.



State-Space Models

- Problem with simple approaches:
  - All information about previous decision must be summarized by x_t.
  - We 'forget' why we predicted x_t when we go to predict x_{t+1}.
- More complex dynamics possible with state-space models:
  - Add hidden states with their own latent dynamics.



Challenges of State-Space Models

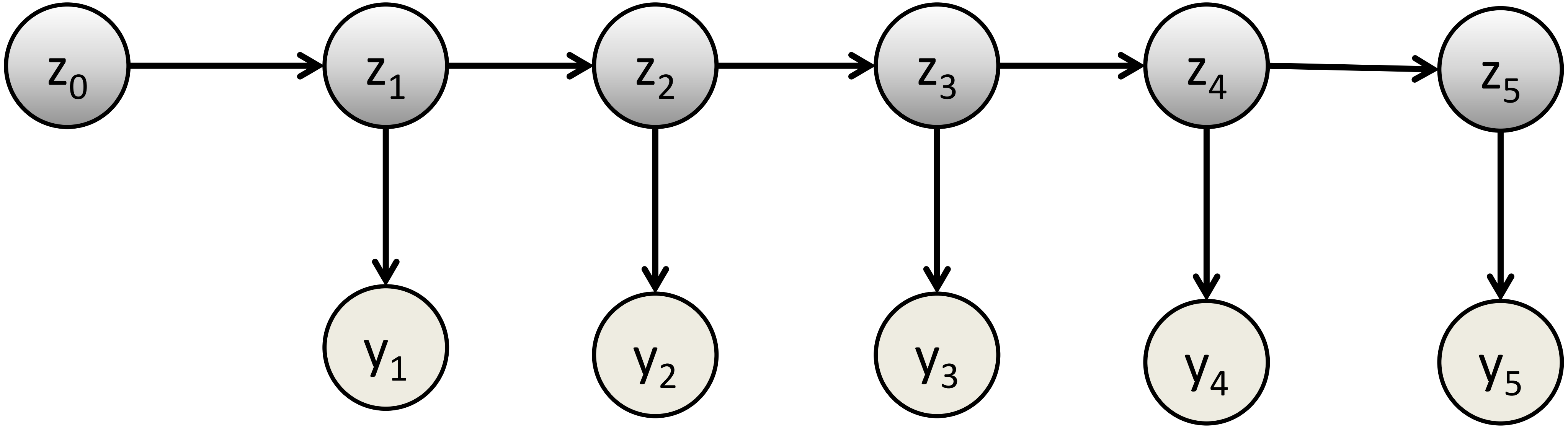
- Problem 1: inference only has closed-form in simple situations.
  - Only 2 cases: Gaussian z *and* y (Kalman filter) or discrete z (HMMs).
  - Otherwise, need to use approximate inference.

- Problem 2: memory is very limited.
  - You have to choose a z_t at time ‘t’.
    - But still need to compress information into a single hidden state.

- Obvious solution:
  - Have multiple hidden z_t at time ‘t’, as we did before.
    - But now inference becomes hard.

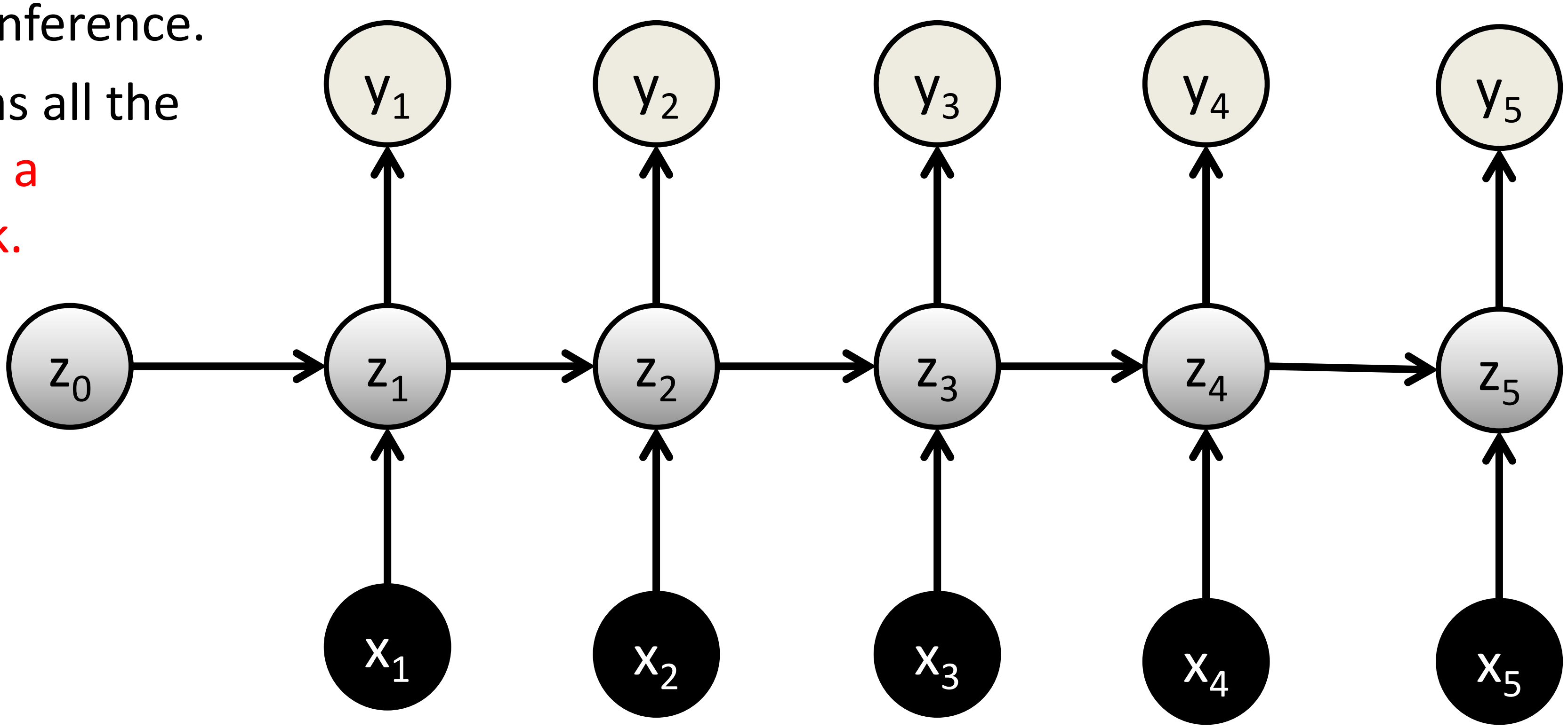
Recurrent Neural Networks

- Recurrent neural networks (RNNs) give solution to inference:
  - At time ‘t’, hidden units are deterministic transformations of time ‘t-1’.
  - Basically turns the problem into a big and structured neural network.



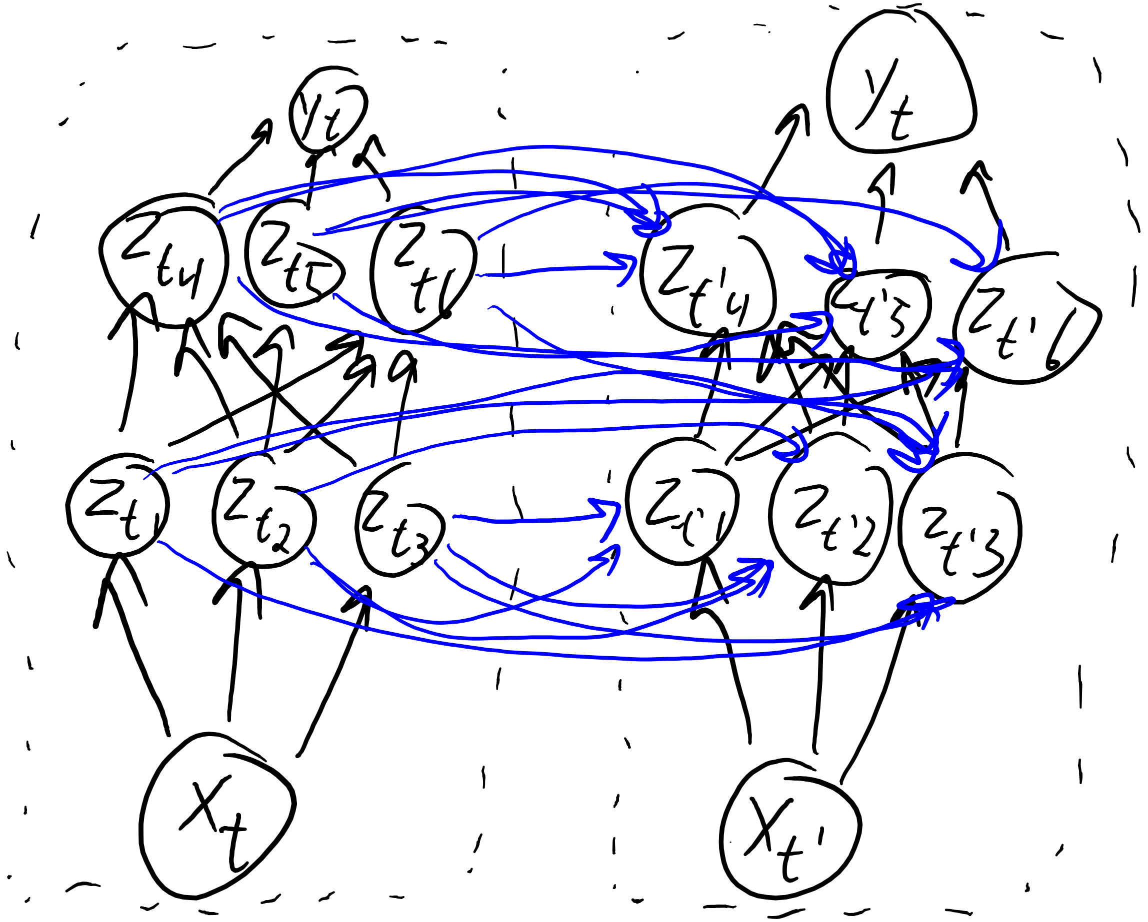
Recurrent Neural Networks

- RNNs can be used to translate input sequence to output sequence:
  - A neural network version of latent-dynamics models.
  - Deterministic transforms mean hidden ‘z’ can be really complicated.
    - But with easy inference.
    - I’m using “z_1” as all the hidden units in a neural network.



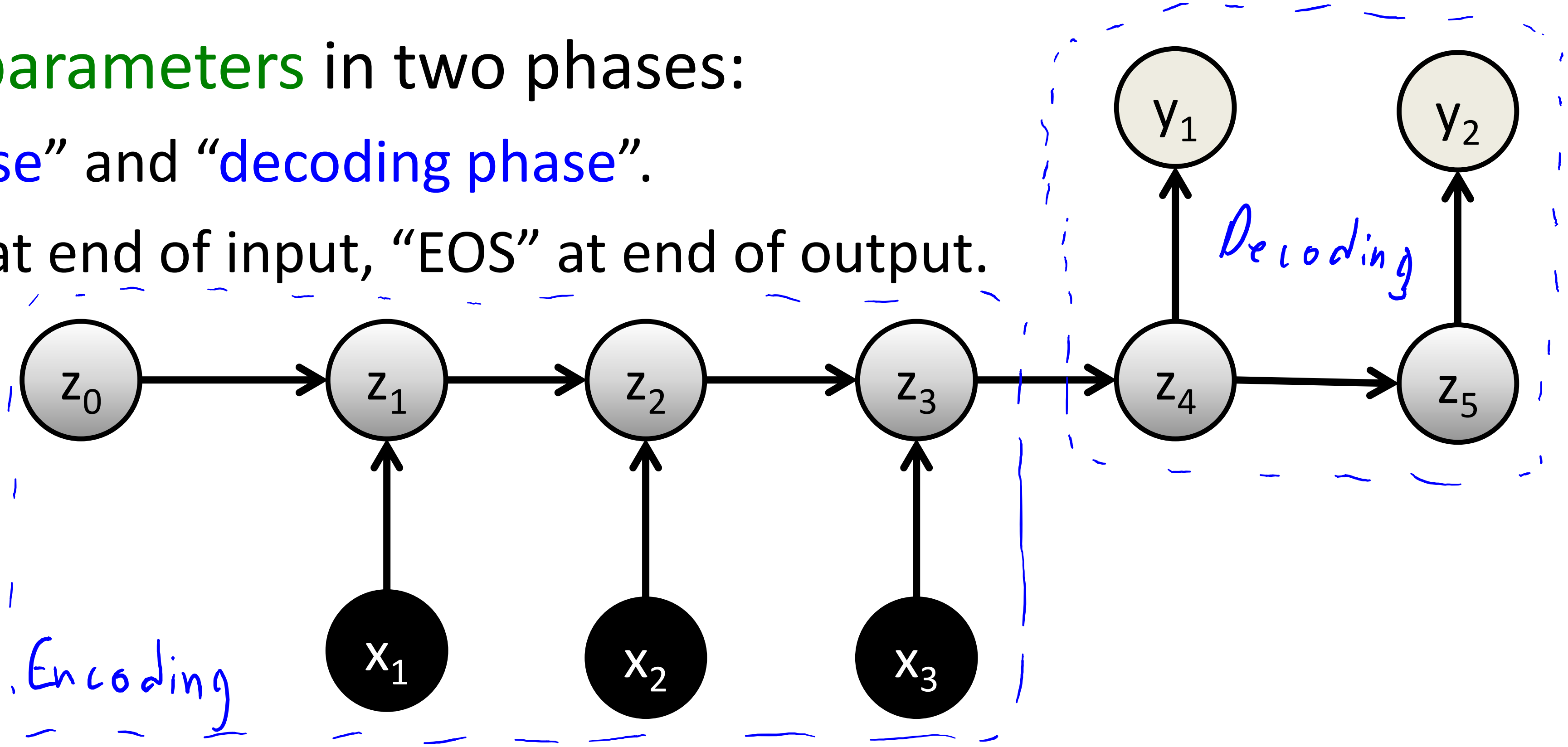
Recurrent Neural Networks

- Can think of each time as implementing the same neural network:
  - But with connections from hidden units at previous time.



Sequence-to-Sequence

- An interesting variation on this for sequences of different lengths:
  - Translate from French sentence ‘x’ to English sentence ‘y’.
- Usually we tie parameters in two phases:
  - “Encoding phase” and “decoding phase”.
  - Special “BOS” at end of input, “EOS” at end of output.



Training Recurrent Neural Networks

- Train using stochastic gradient: "backpropagation through time".
- Similar challenges/heuristics to training deep neural networks:
    - "Exploding/vanishing gradient", initialization is important, slow progress, etc.

- Exploding/vanishing gradient problem is now worse:
  – Parameters are tied across time:
    - Gradient gets magnified or shrunk exponentially at each step.

  – Common solutions:
    - "Gradient clipping": limit gradient to some maximum value.
    - Long Short Term Memory (LSTM): make it easier for information to persist.

Variations on Recurrent Neural Networks

- Bi-directional RNNs: feedforward from past and future.
- Recursive neural networks: consider sequences through non-chain data.
- Graphical models to explicitly encourage output dependencies:
    - https://arxiv.org/abs/1711.04956

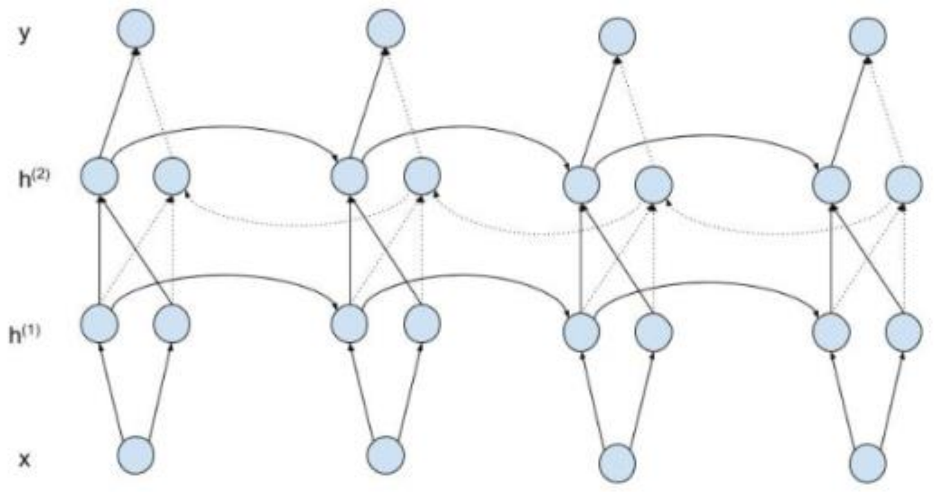
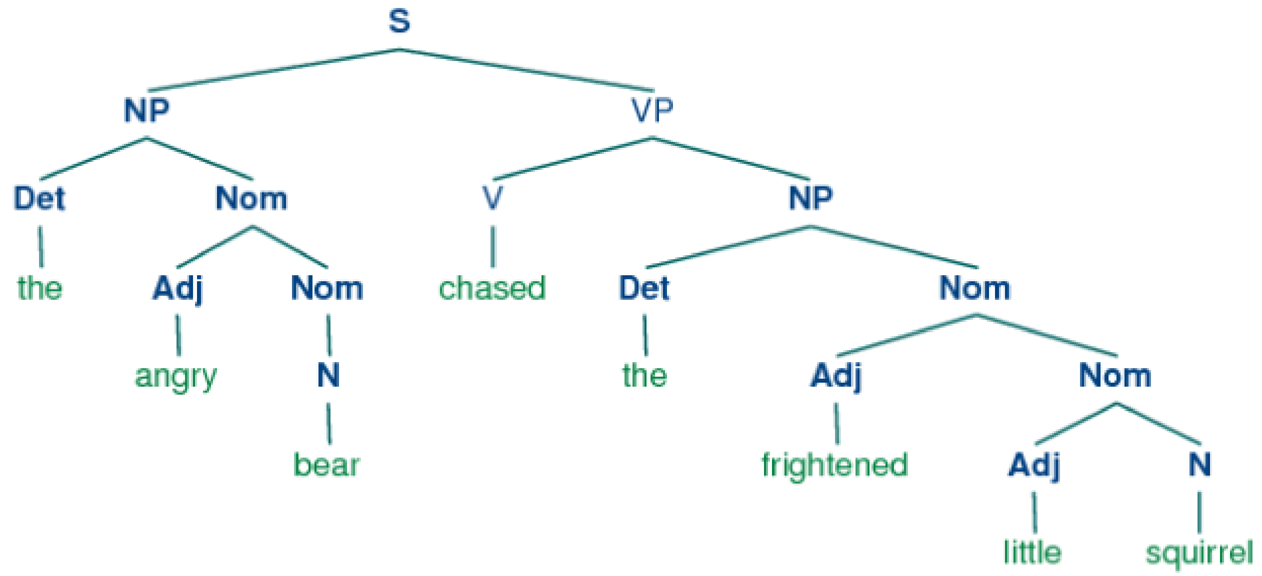


Figure 2: A deep bi-directional RNN with 2 stakced layers



Long Short Term Memory (LSTM)

- Long short term memory (LSTM) models are special case of RNNs:
  - Designed so that model can remember things for a long time.
- LSTMs have been the analogy of convolutions for RNNs:
  - "The trick that makes them work in applications."
- LSTMs are getting impressive performance in various settings:
  - Cursive handwriting recognition.
    - https://www.youtube.com/watch?v=mLxsbWAYIpw
  - Speech recognition.
  - Machine translation.
  - Image and video captioning.

LSTMs for Image Captioning

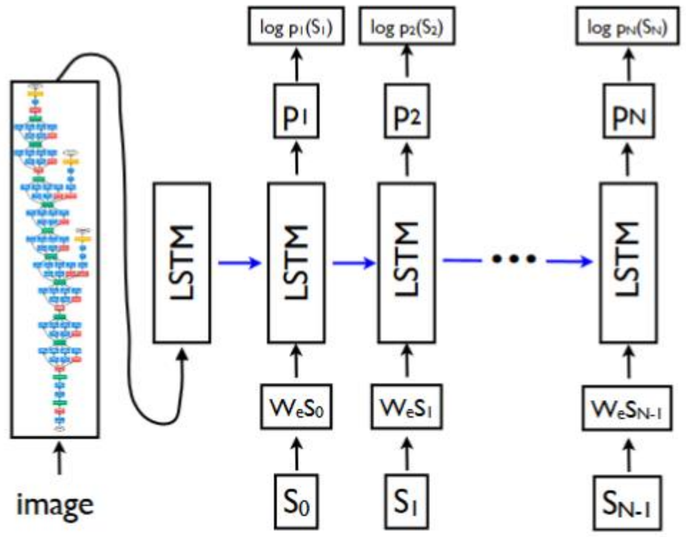
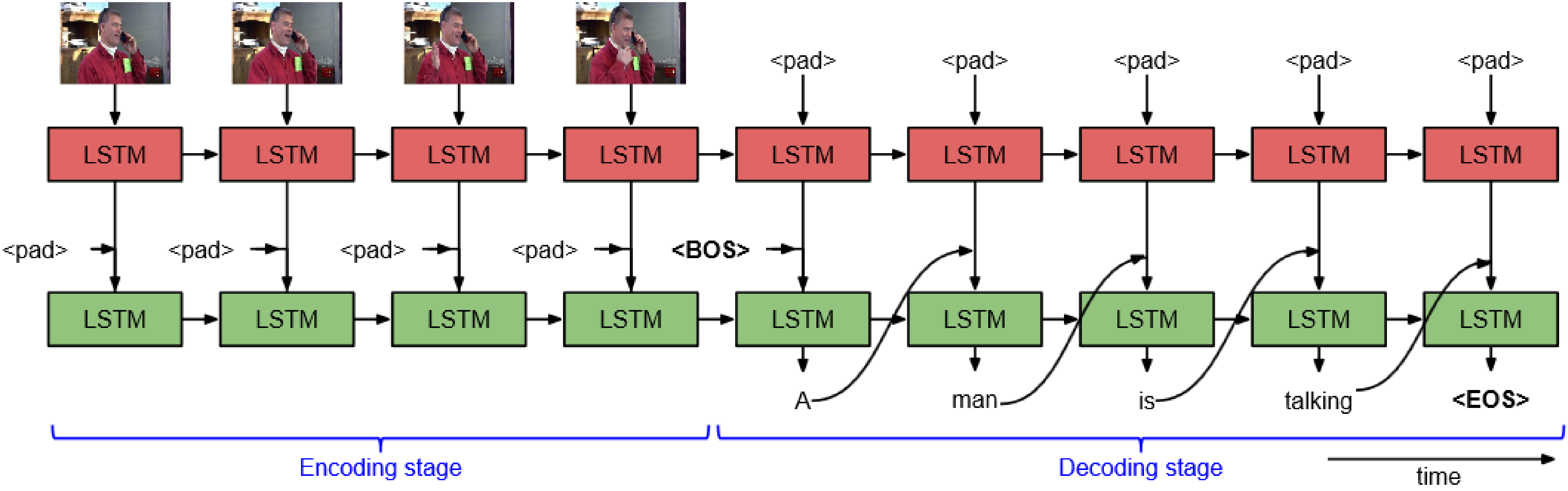


Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

Figure 5. A selection of evaluation results, grouped by human rating.

https://arxiv.org/pdf/1411.4555.pdf

LSTMs for Video Captioning



http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Venugopalan_Sequence_to_Sequence_ICCV_2015_paper.pdf

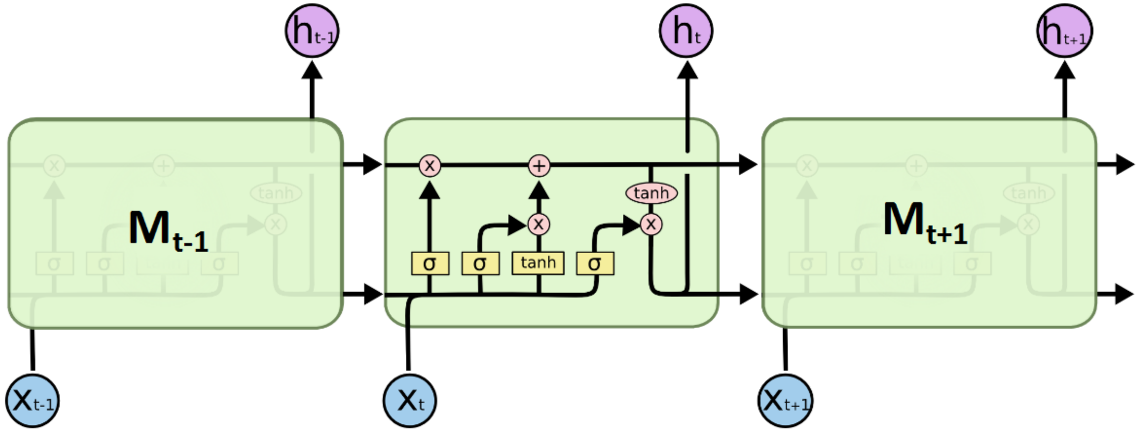
LSTMs for Video Captioning

Figure 3. Qualitative results on MSVD YouTube dataset from our S2VT model (RGB on VGG net). (a) Correct descriptions involving different objects and actions for several videos. (b) Relevant but incorrect descriptions. (c) Descriptions that are irrelevant to the event in the video.

http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Venugopalan_Sequence_to_Sequence_ICCV_2015_paper.pdf

Long Short Term Memory

- In addition to usual hidden values 'z', LSTMs have memory cells 'c':
  - Purpose of memory cells is to remember things for a long time.

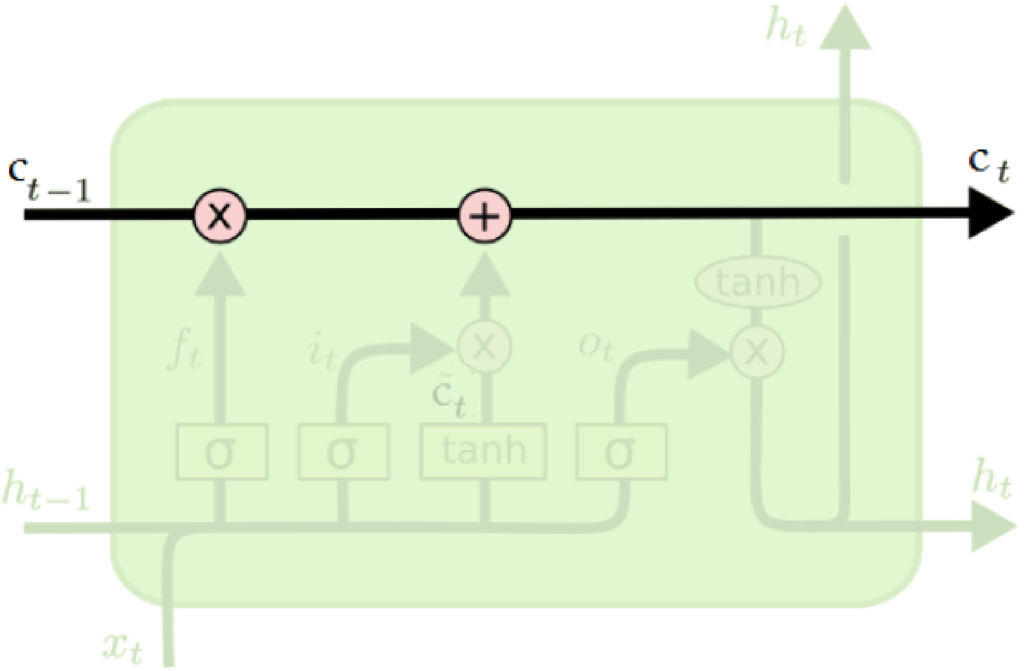


- "Read/write/forget":
  - Information gets into the cell when its input gate is on.
  - Information is read from the cell when the output gate is on.
  - Information is thrown away when the forget gate is off.
- "Gate functions": approximate binary operations (like "write or not").
  - Replace operation by a sigmoid functions to make it continuous/differentiable.

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

The Core Idea Behind LSTMs: Cell State (Memory Cell)

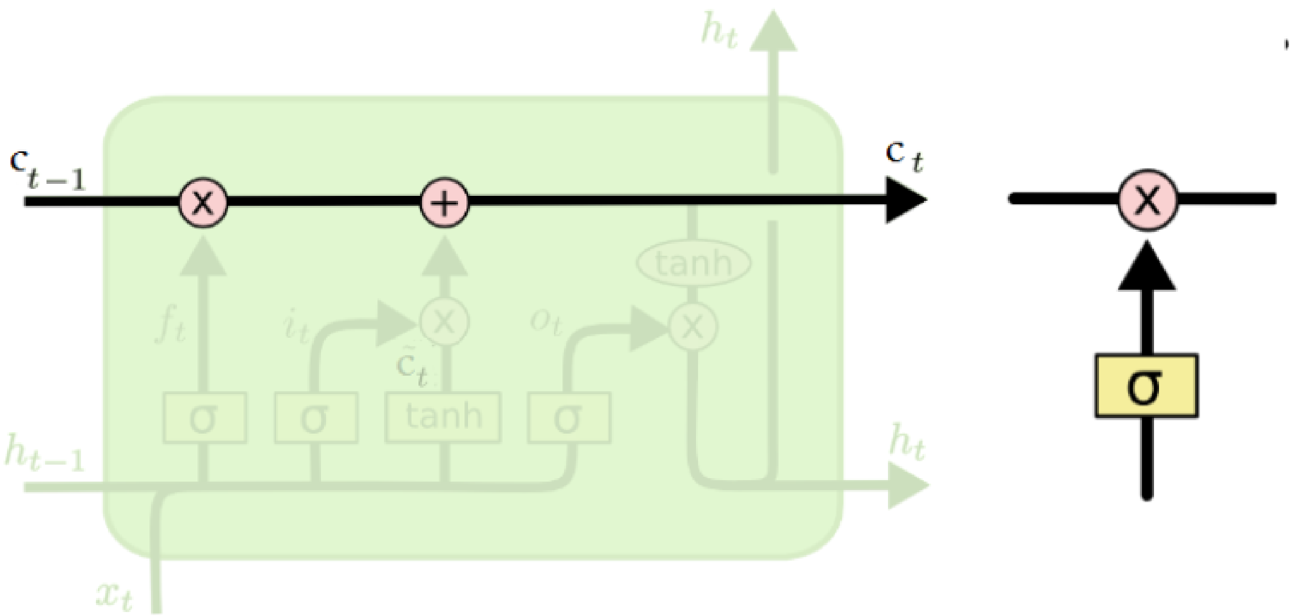
- Information can flow along the **memory cell unchanged**.
- Information can be **removed** or **written** to the **memory cell**, regulated by gates.



Adapted from: *C. Olah*

Gates

- **Gates** are a way to optionally let information through.
  - A **sigmoid layer** outputs number between 0 and 1, **deciding** how much of each component should be let through.
  - A pointwise multiplication operation applies the decision.

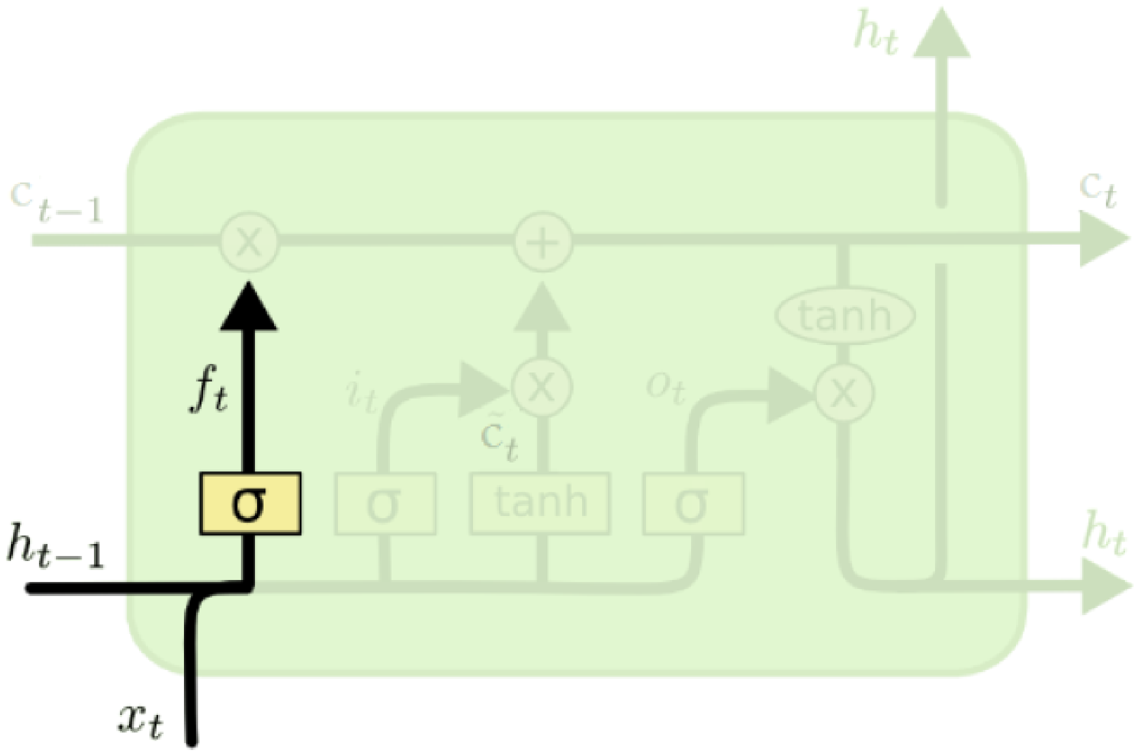


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Adapted from: *C. Olah*

Forget Gate

- A **sigmoid** layer, **forget gate**, **decides** which values of the **memory cell** to **reset**.



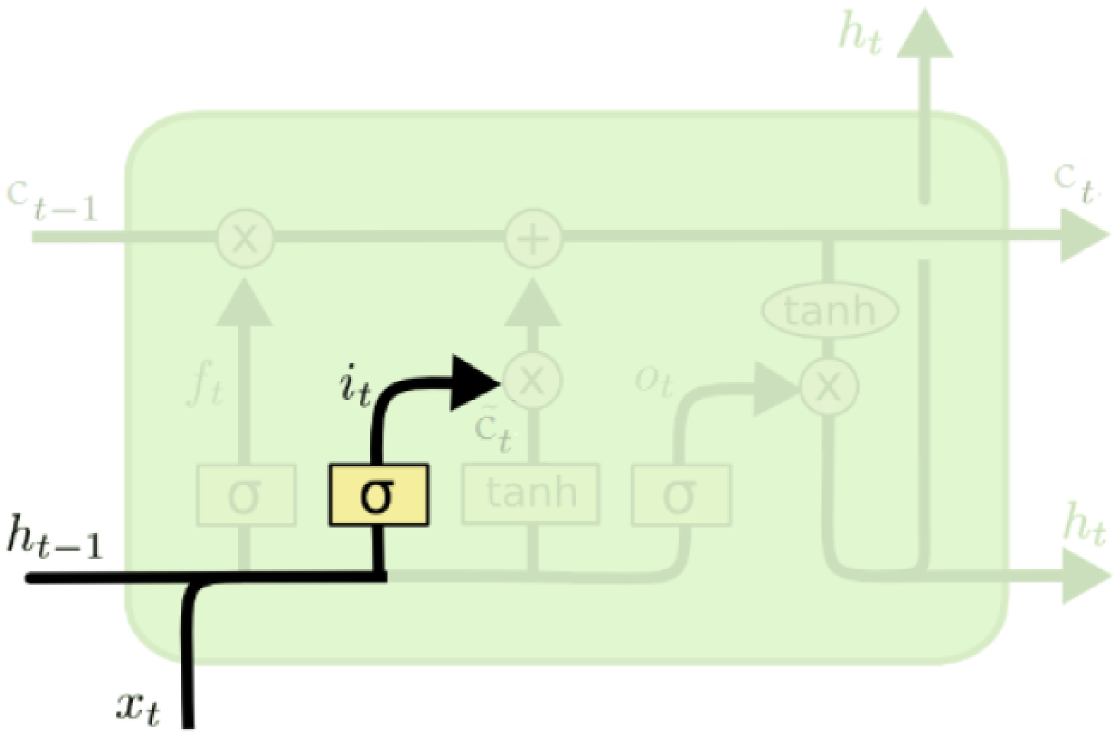
$$\mathbf{f}_t = \sigma(W_f.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

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

Adapted from: *C. Olah*

Input Gate

- A **sigmoid** layer, **input gate**, **decides** which values of the **memory cell** to **write** to.

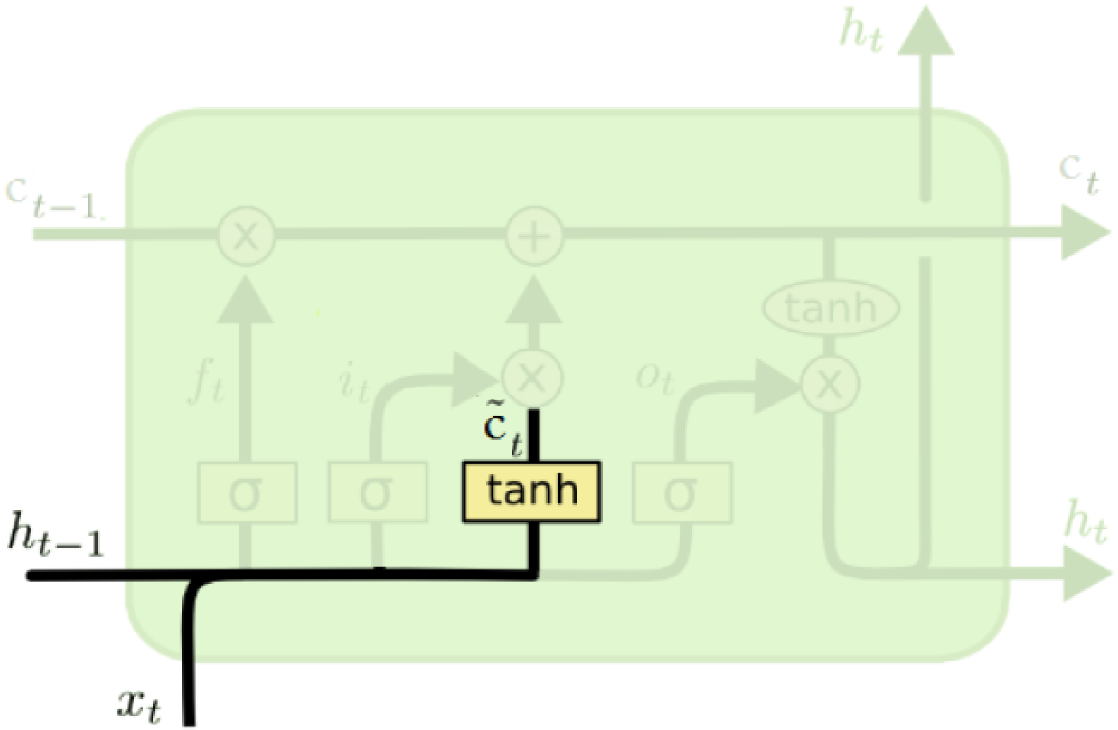


$$\mathbf{i}_t = \sigma(W_i.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

Adapted from: *C. Olah*

Vector of New Candidate Values

- A **Tanh** layer creates a **vector of new candidate values** $\tilde{c}_t$ to **write** to the **memory cell**.

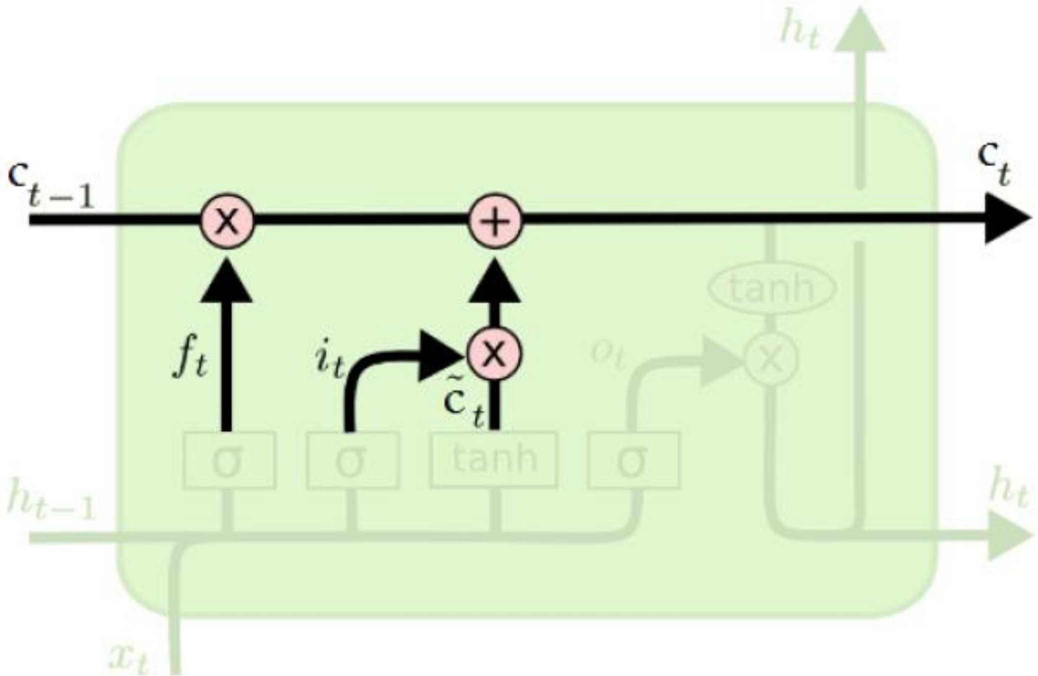


$$\tilde{\mathbf{c}}_t = \text{Tanh}(W_c.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c)$$

Adapted from: *C. Olah*

Memory Cell Update

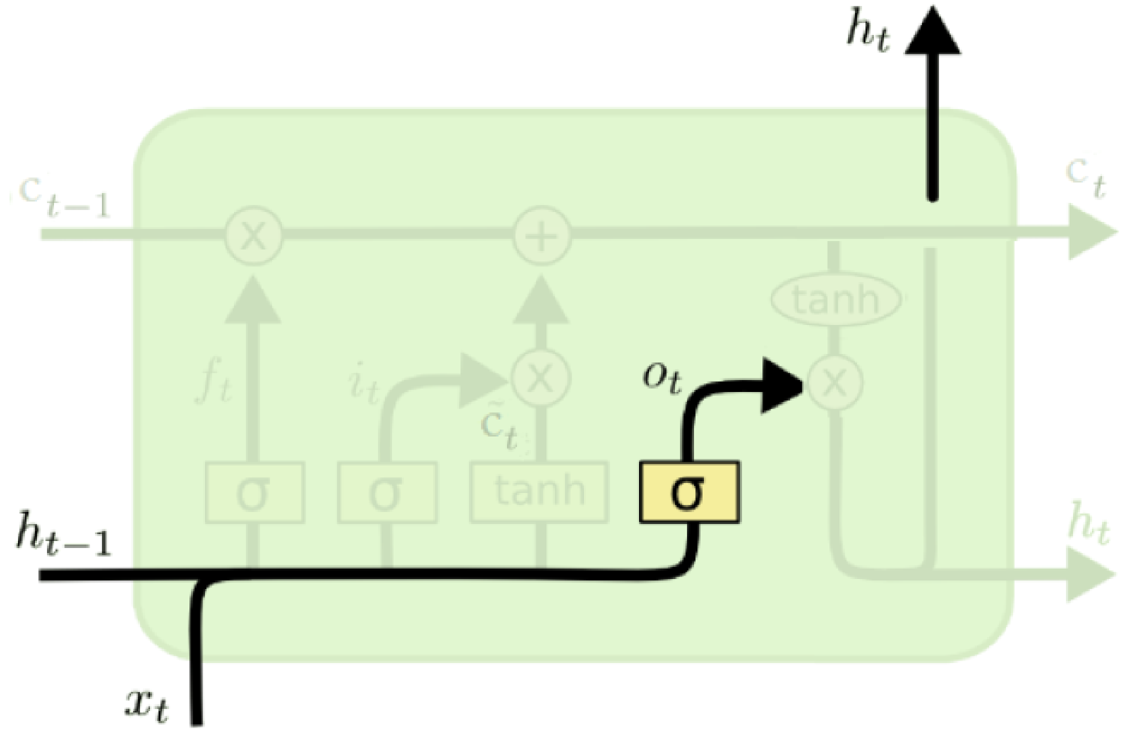
- The previous steps decided which values of the **memory cell** to **reset** and **overwrite**.
- Now the LSTM **applies the decisions** to the **memory cell**.



$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \widetilde{\mathbf{c}}_t$$

Output Gate

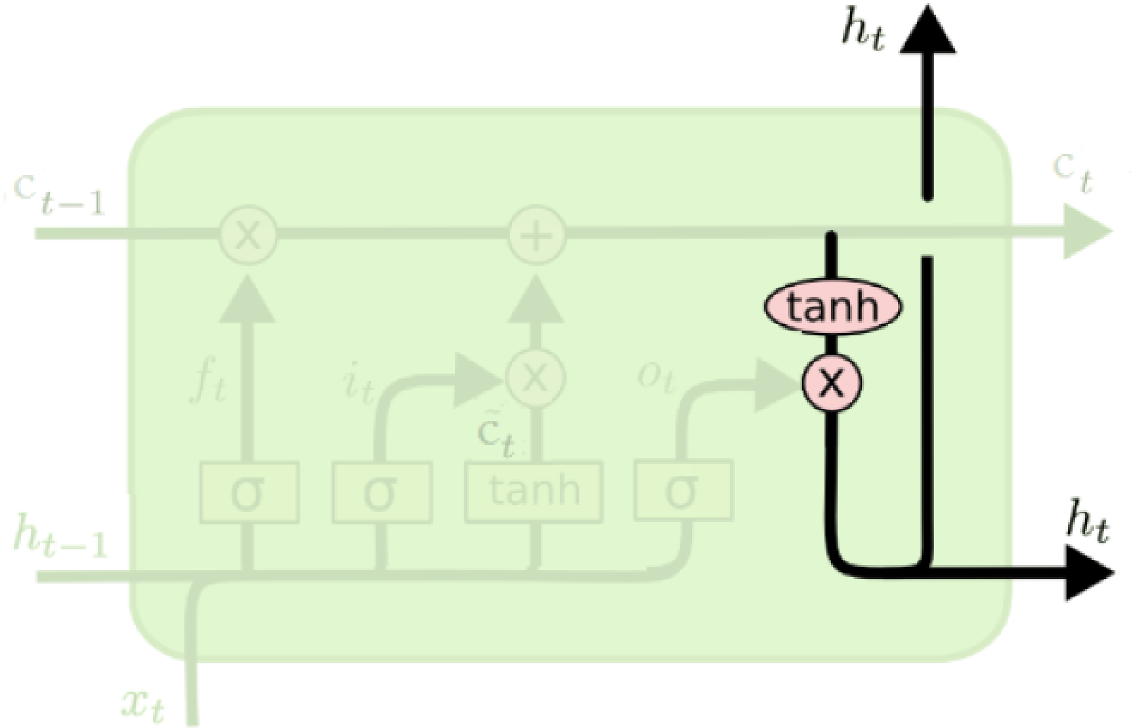
- A **sigmoid** layer, **output gate**, **decides** which values of the **memory cell** to **output**.



$$\mathbf{o}_t = \sigma(W_o.[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

Output Update

- The **memory cell** goes through **Tanh** and is multiplied by the **output gate**.



$$\mathbf{h}_t = \mathbf{o}_t * \text{Tanh}(\mathbf{c}_t)$$

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

Adapted from: *C. Olah*

LSTM Structure

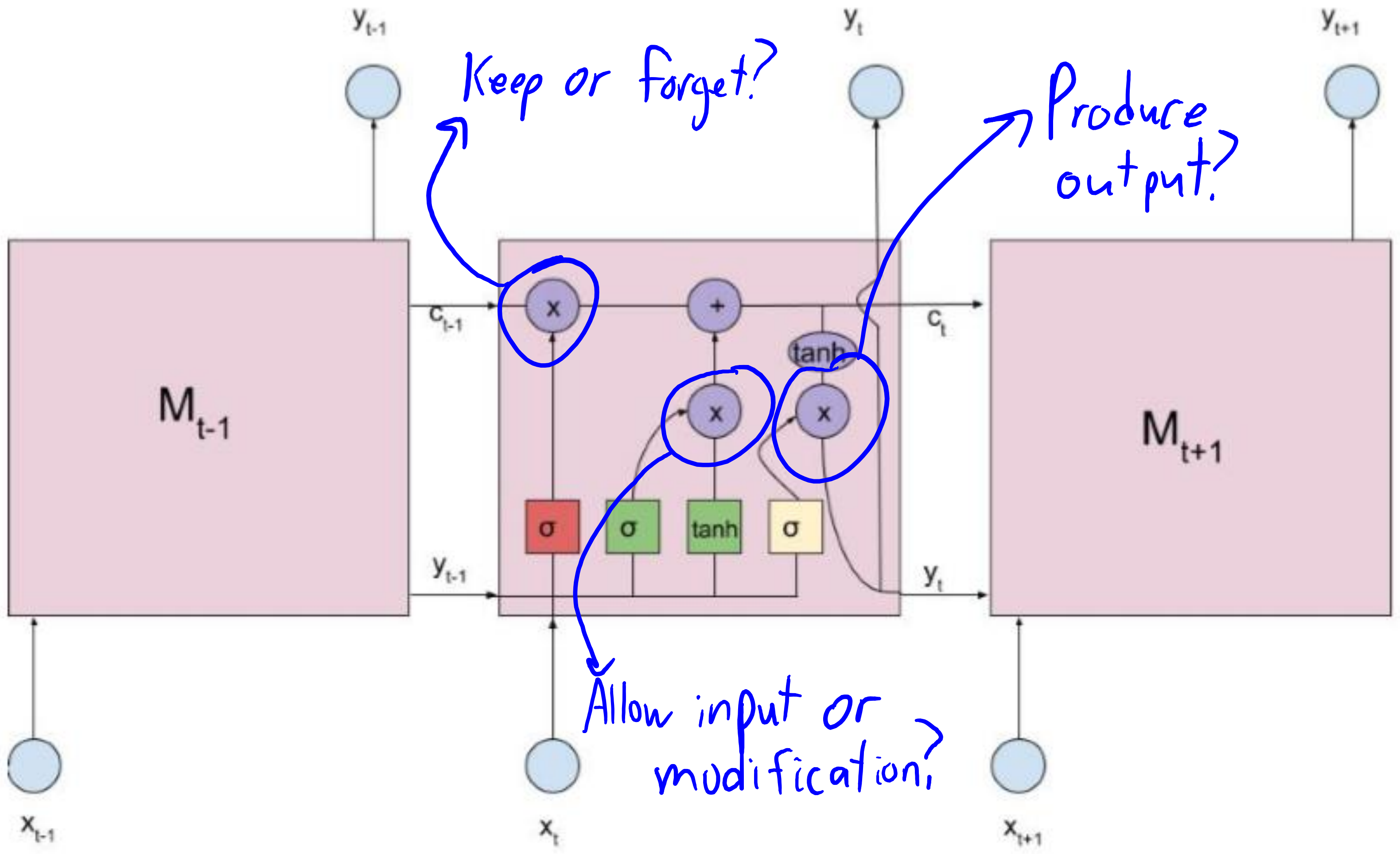


Figure 6: A close look at LSTM structure

Vanilla RNN vs. LSTM

Vanilla Recurrent Neural Network (RNN) has a recurrence of the form

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

Previous layer, same time.
Same layer, previous time.

memory vector c_t^l. At each time step the LSTM can choose to read from, write to, or reset the cell using explicit gating mechanisms. The precise form of the update is as follows:

Input
Forget
Output
Candidate

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

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

Forget times old memory.
Input times candidate.

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

Output times current memory

Here, the sigmoid function sigm and tanh are applied element-wise, and W^l is a $[4n \times 2n]$ matrix.

- Notice that if "f=1" and "i=0", then memory is unchanged.
  - Memory might only change for specific inputs.
- More recent: gated recurrent unit (GRU):
  - Similar performance but a bit simpler.

http://arxiv.org/pdf/1506.02078v2.pdf

Variants on LSTM

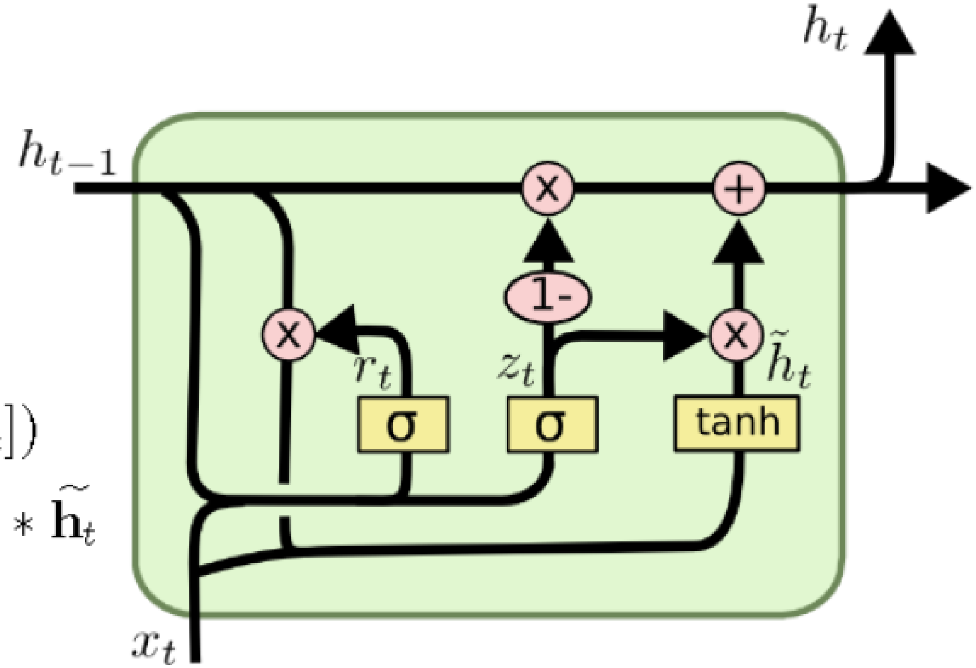
- Gated Recurrent Unit (GRU) [Cho et al., 2014]:
  - Combine the **forget** and **input** gates into a single **update** gate.
  - **Merge the memory cell and the hidden state**.
  - ...

$$z_t = \sigma(W_z.[\mathbf{h}_{t-1}, \mathbf{x}_t])$$

$$r_t = \sigma(W_r.[\mathbf{h}_{t-1}, \mathbf{x}_t])$$

$$\widetilde{\mathbf{h}}_t = \text{Tanh}(W.[r_t * \mathbf{h}_{t-1}, \mathbf{x}_t])$$

$$\mathbf{h}_t = (1 - z_t) * \mathbf{h}_{t-1} + (z_t) * \widetilde{\mathbf{h}}_t$$



Adapted from: *C. Olah*

Residual Connections

- As in ResNets, modern RNNs are including residual connections:

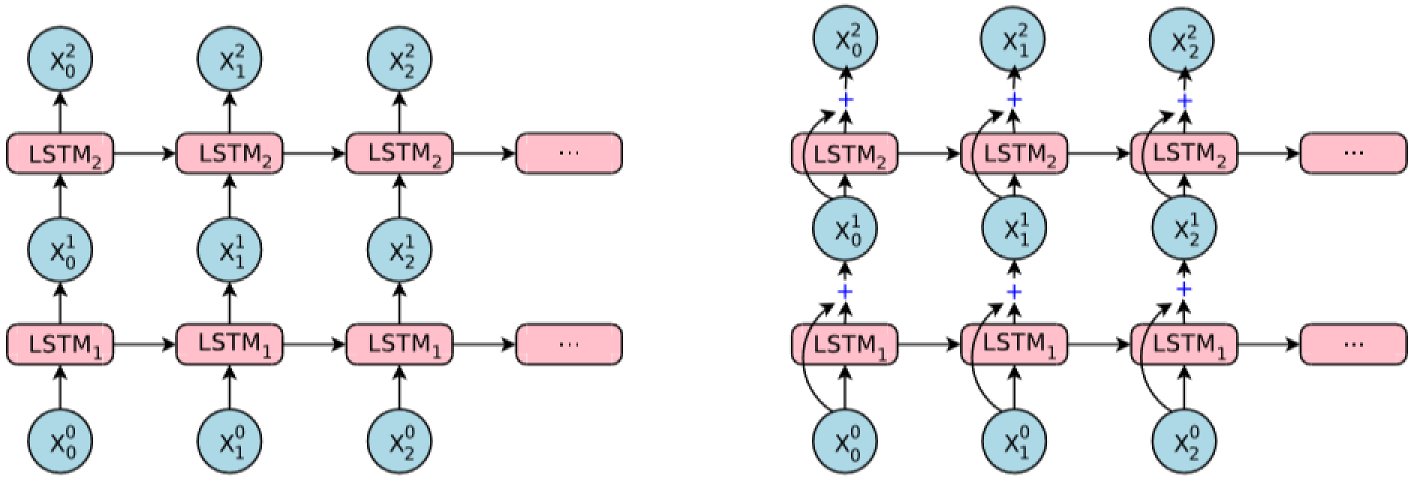


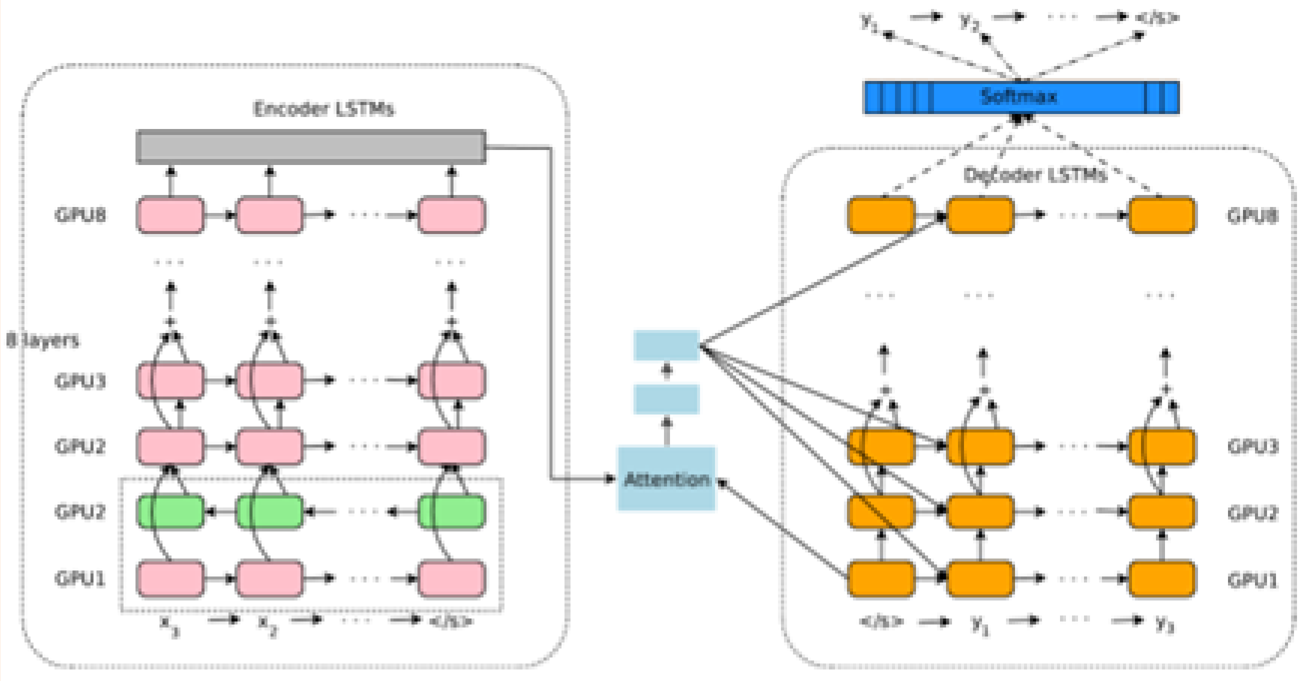
Figure 2: The difference between normal stacked LSTM and our stacked LSTM with residual connections. On the left: simple stacked LSTM layers [41]. On the right: our implementation of stacked LSTM layers with residual connections. With residual connections, input to the bottom LSTM layer ($\mathbf{x_i^0}$'s to $LSTM_1$) is element-wise added to the output from the bottom layer ($\mathbf{x_i^1}$'s). This sum is then fed to the top LSTM layer ($LSTM_2$) as the new input.

- You can also add residual connections across time.
  - Many variations on "skip connections"

https://arxiv.org/pdf/1609.08144.pdf

Attention

- Many recent systems incorporate attention.
  – Google's neural machine translation incorporates.



- Learn to re-weight during decoding to emphasize important parts

https://arxiv.org/pdf/1609.08144.pdf

Attention

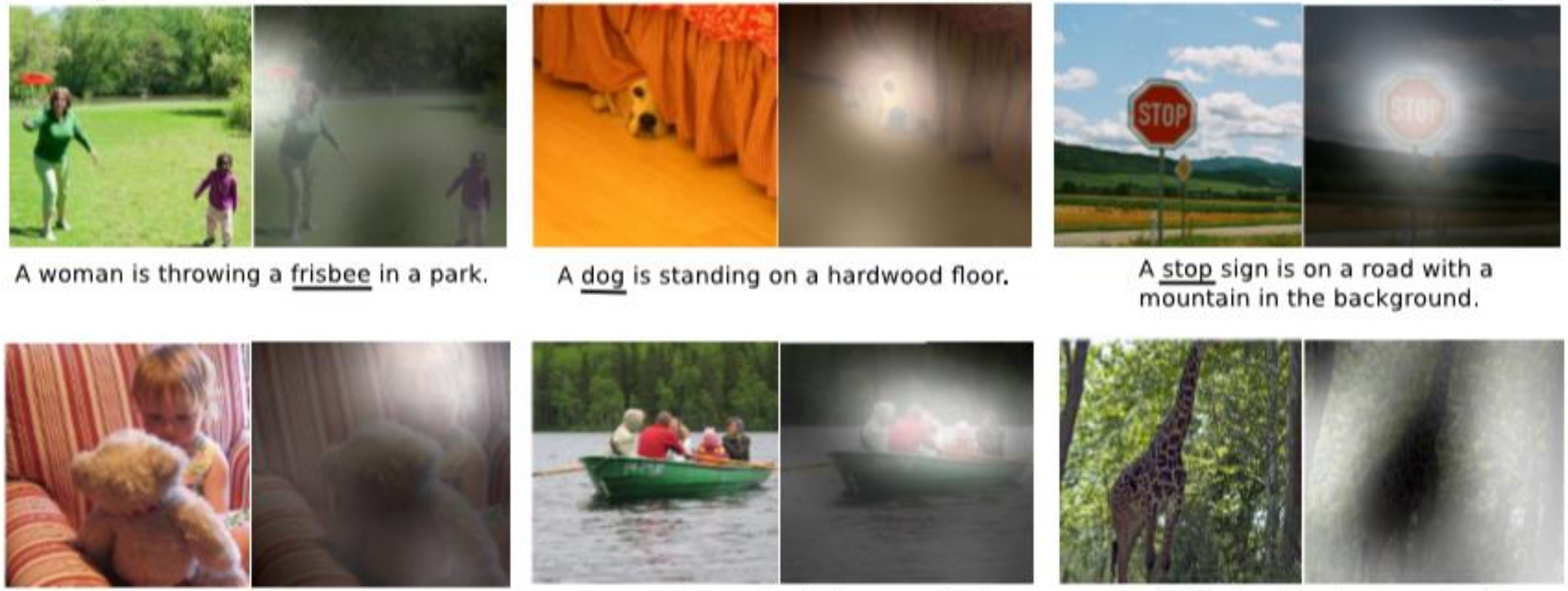
- Attention for language translation:

https://code.facebook.com/posts/1978007565818999/a-novel-approach-to-neural-machine-translation/

Attention

- Attention for image captioning:

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



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.

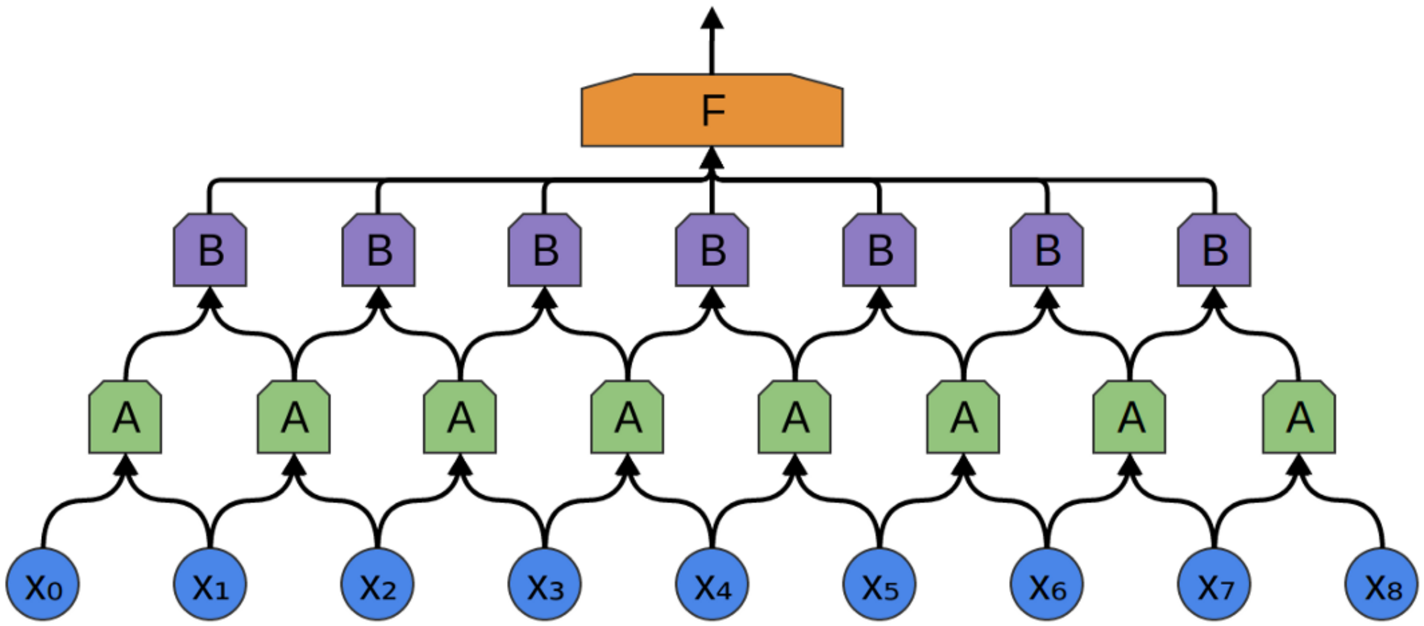
A giraffe standing in a forest with trees in the background.

https://arxiv.org/pdf/1502.03044.pdf

Convolutions for Sequences?

- Should we really be going through a sequence sequentially?
  - What if stuff in the middle is really important, and changes meaning?

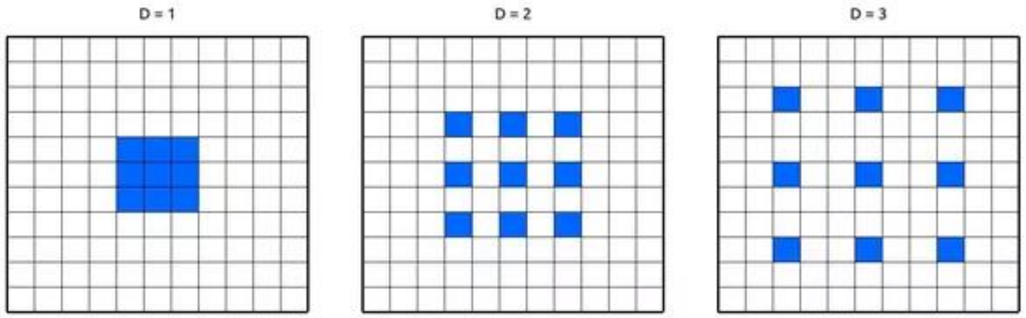
- Recent works have started using convolutions for sequences.



https://medium.com/@TalPerry/convolutional-methods-for-text-d5260fd5675f

Digression: Dilated Convolutions ("a trous")

- Best CNN systems have gradually reduced convolutions sizes.
  - Many modern architectures use 3x3 convolutions, far fewer parameters!
- Sequences of convolutions take into account larger neighbourhood.
  - 3x3 convolution followed by another gives a 5x5 neighbourhood.
  - But need many layers to cover a large area.
- Alternative recent strategy is dilated convolutions ("a trous").



- Not the same as "stride" in a CNN:
  - Doing a 3x3 convolution at all locations, but using pixels that are not adjacent.
  - During upsampling, you can use interpolation to fill the holes.

https://medium.com/@TalPerry/convolutional-methods-for-text-d5260fd5675f

Dilated Convolutions ("a trous")

- Modeling language and music with dilated convolutions:

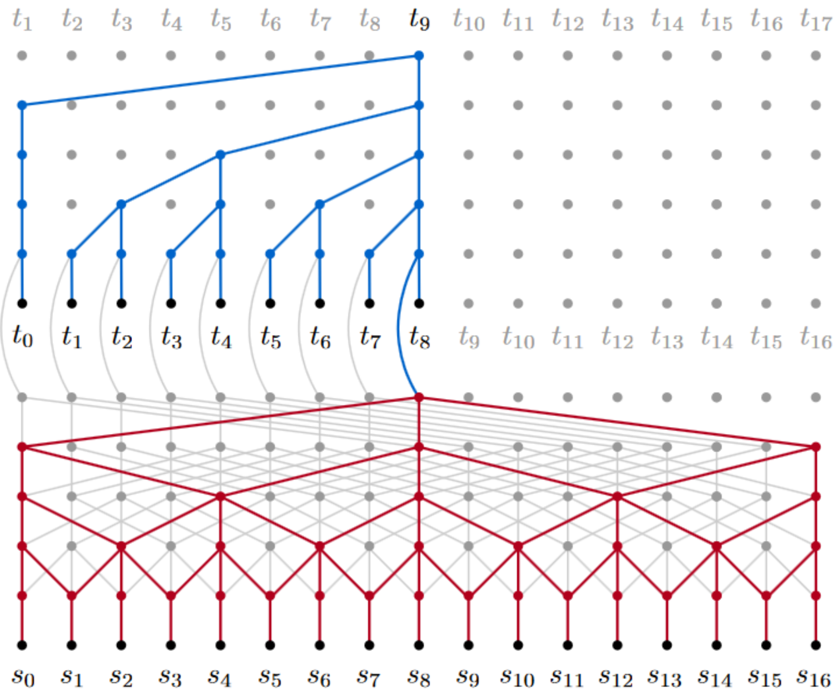
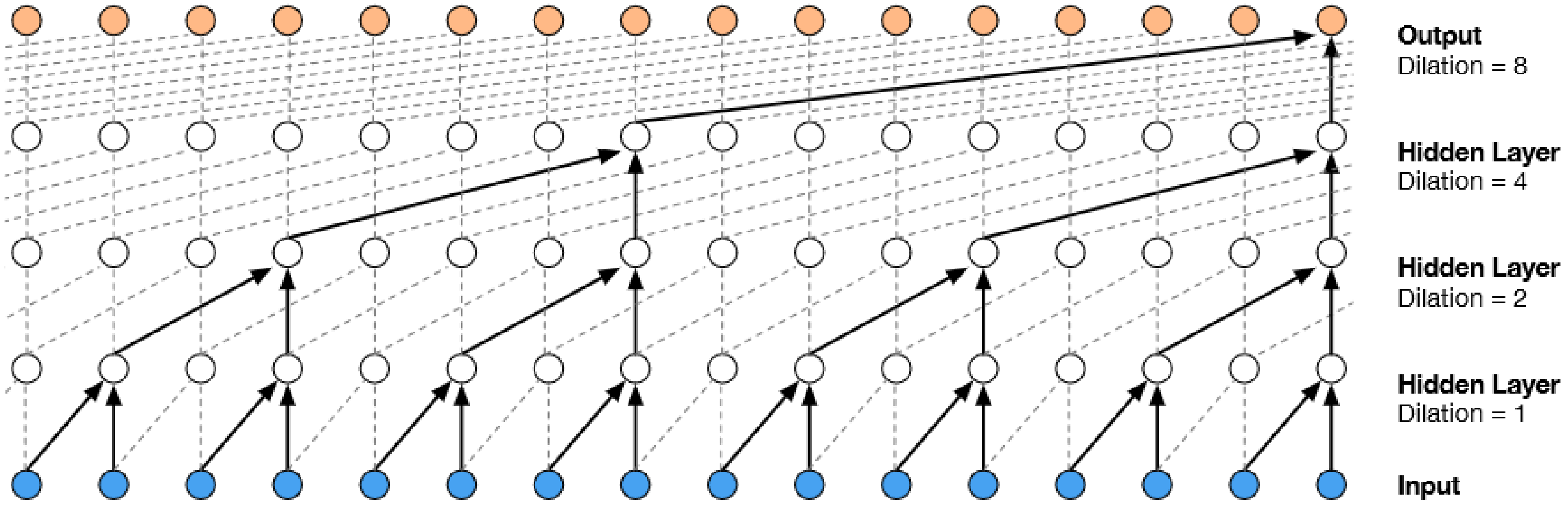


Figure 1. The architecture of the ByteNet. The target decoder (blue) is stacked on top of the source encoder (red). The decoder generates the variable-length target sequence using dynamic unfolding.



https://arxiv.org/pdf/1610.10099.pdf
https://arxiv.org/pdf/1609.03499.pdf

More RNN/CNN Applications

- Generating text:
  - https://pjreddie.com/darknet/rnns-in-darknet
- Fake positive/negative Amazong reviews:
  - https://blog.openai.com/unsupervised-sentiment-neuron
- PDF to LaTeX:

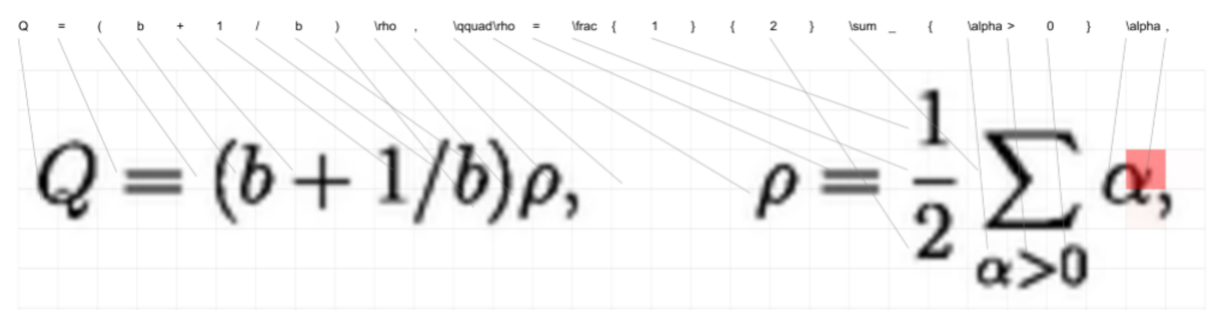


Figure 1: Example of the model generating mathematical markup. The model generates one LaTeX symbol y at a time based on the input image $\mathbf{x}$. The gray lines highlight $H' \times V'$ grid features after the CNN $\mathbf{V}$ and RNN Encoder $\tilde{\mathbf{V}}$. The dotted lines indicate the center of mass of α for each word (only non-structural words are shown). Red cells indicate the relative attention for the last token. See http://lstm.seas.harvard.edu/latex/ for a complete interactive version of this visualization over the test set.

- Lip reading:
  - https://www.youtube.com/watch?v=5aogzAUPilE

https://arxiv.org/pdf/1609.04938v1.pdf

RNNs/CNNs for Poetry

- Generating poetry:

And still I saw the Brooklyn stairs
With the shit, the ground, the golden haze
Of the frozen woods where the boat stood.
When I thought of shame and silence,
I was a broken skull;
I was the word which I called it,
And I saw the black sea still,
So long and dreary and true;
The way a square shook out my ground,
And the black things were worth a power,
To find the world in a world of reason,
And I saw how the mind saw me.

- Image-to-poetry:

A man is sitting on the edge of the waters.
I should see him begin to stand at the throat of the graveyard
and my love is like a stairway in his left arm and a piece of the stairs,
and there is a girl in the doorway and she and I am a good time.
I want to see her the best thing with the footprints in the woods
and the candle shifts back to the shrine and the last late sun
the sky and the candle and the noise of the snow.

Dropout 0.25, Loss 1.1465, 1:16:1, Railroad

A train traveling over a bridge over a river to the end of the street and the sea is a strange street with a cold sun on the street where the sun stands and the sun is still and the sun is still and the sun is gone. The sun is all around me. I am the same as the sun on the street with a strange contract.

A train traveling over a bridge over a river to the graveyard and the barn was a strange street of straw halls and the sun was always sinking in the sun.

I was the one who was still in the street when he was standing in the sun and the sun was still alive.

He was a big smile and I was a child who was a stranger.

- Movie script:
  - https://www.youtube.

https://medium.com/artists-and-machine-intelligence/adventures-in-narrated-reality-6516ff395ba3

RNNs/CNNs for Music and Dance

- Music generation:
  - https://www.youtube.com/watch?v=RaO4HpM07hE

- Text to speech and music waveform generation:
  - https://deepmind.com/blog/wavenet-generative-model-raw-audio

- Dance choreography:
  - http://theluluartgroup.com/work/generative-choreography-using-deep-learning

- Music composition:
  - https://www.facebook.com/yann.lecun/videos/10154941390687143

Neural Turing/Programmers

- Many interesting recent variations on memory/attention.
  - A good place to start is here: https://distill.pub/2016/augmented-rnns

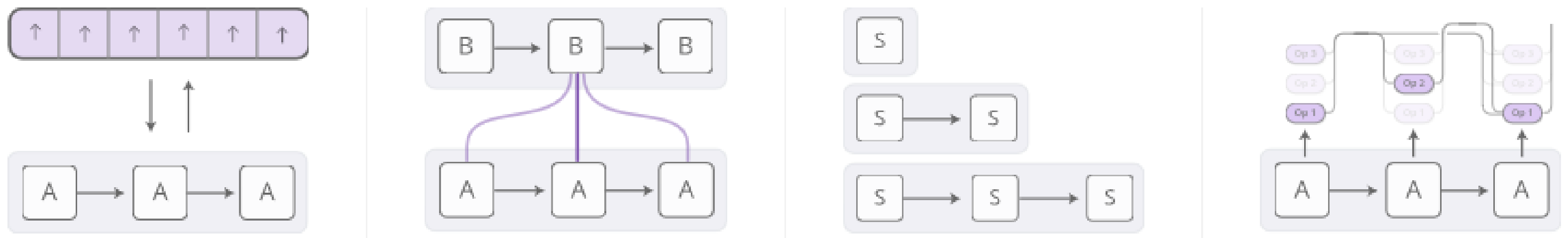
Here is an example of what the system can do. After having been trained, it was fed the following short story containing key events in JRR Tolkien's Lord of the Rings:

Bilbo travelled to the cave.
Gollum dropped the ring there.
Bilbo took the ring.
Bilbo went back to the Shire.
Bilbo left the ring there.
Frodo got the ring.
Frodo journeyed to Mount-Doom.
Frodo dropped the ring there.
Sauron died.
Frodo went back to the Shire.
Bilbo travelled to the Grey-havens.
The End.

After seeing this text, the system was asked a few questions, to which it provided the following answers:

Q: Where is the ring?
A: Mount-Doom
Q: Where is Bilbo now?
A: Grey-havens
Q: Where is Frodo now?
A: Shire

It's probably one of the few technical papers that cite "Lord of the Rings".



Neural Turing Machines have external memory that they can read and write to.

Attentional Interfaces allow RNNs to focus on parts of their input.

Adaptive Computation Time allows for varying amounts of computation per step.

Neural Programmers can call functions, building programs as they run.

https://www.facebook.com/FBAIResearch/posts/362517620591864

Summary

- Recurrent neural networks:
  - Neural networks for model sequenctial inputs and/or sequential outputs.
- Long short term memory:
  - Gating functions which update “memory cells” for long-range interactions.
- Dilated convolutions:
  - Convolutions with holes to model long-term dependencies.
- Next time: why everything we’ve done in 340 and 540 is wrong.