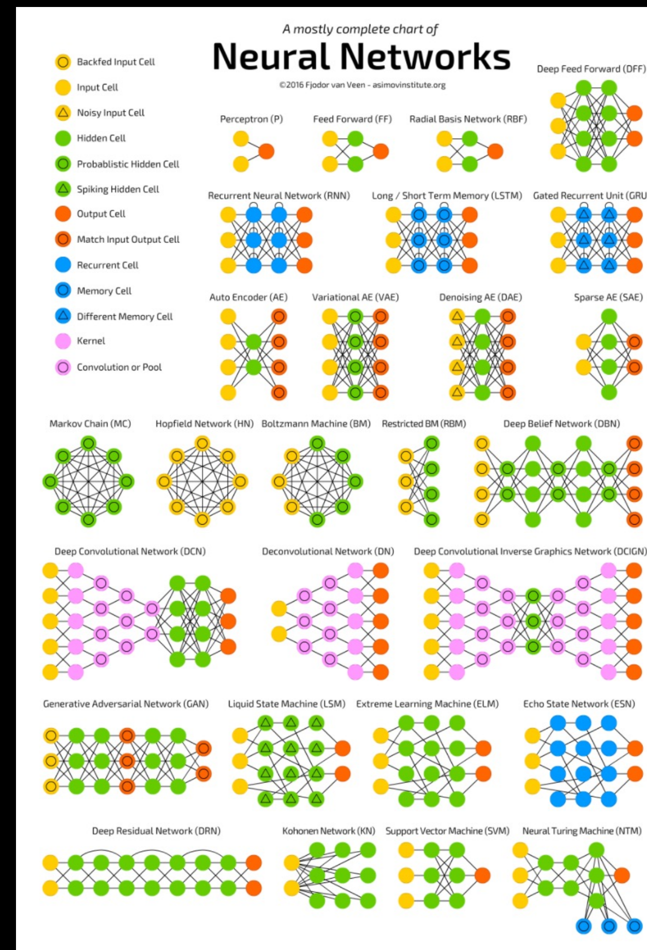
Recurrent Neural Networks

CSCI 4360/6360 Data Science II

The Neural Network Zoo

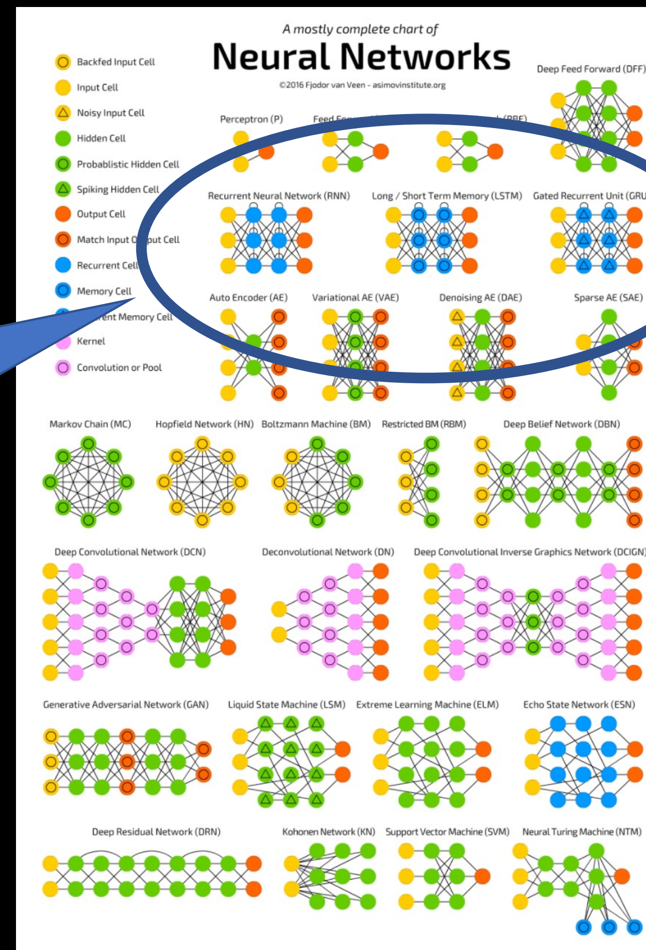
- http://www.asimovinstitute.org/neural-network-zoo/



The Neural Network Zoo

- http://www.asimovinstitute.org/neural-network-zoo/

Today



Modeling Sequences

- Input:

$$X = [\vec{x}_1, \vec{x}_2, ..., \vec{x}_T]$$

- Output:

$$Y = [\vec{y}_1, \vec{y}_2, ..., \vec{y}_N]$$

T and *N* not necessarily equal

Dimensions of *X* and *Y* not necessarily equal

Language Translation

Weather and Climate Forecasting

Automated Driving

Other “long-distance” time series data

Something we’ve seen before

Linear Dynamical Models

- Two main components (using notation from Hyndman 2006):

Appearance Model

$$y_t = Cx_t + u_t$$

State Model

$$x_t = Ax_{t-1} + Wv_t$$

Autoregressive Models

- This is the definition of a 1st-order autoregressive (AR) process!

$$x_t = Ax_{t-1} + Wv_t$$

- Each observation (x_t) is a function of previous observations, plus some noise

- **Markov model!**

Autoregressive Models

- AR models can have higher orders than 1
- Each observation is dependent on the previous *d* observations

$$x_t = A_1 x_{t-1} + A_2 x_{t-2} + \ldots + A_d x_{t-d} + W v_t$$

Autoregressive Models

- Concrete, *a priori* definition of what is important
  - n^{th}-order Markov process
  - n+1 terms and larger are explicitly ignored
- No concept of *attention*
  - All *n* terms receive equal "attention" (computationally, if not also statistically)
  - Are you devoting equal time reading every word on this slide?
- Cannot handle *variable-length inputs*, nor *variable-length outputs*
  - Contrast with CNNs: all input images have to be the same size (usually)
  - Contrast with [insert deep network of choice]: all outputs are the same, given any input

Attention

- Some things are more important than others

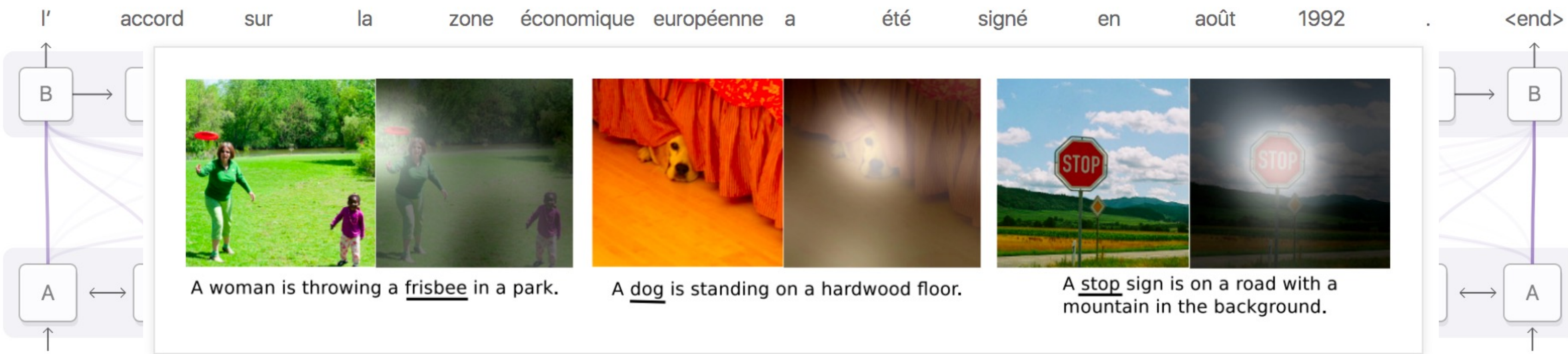
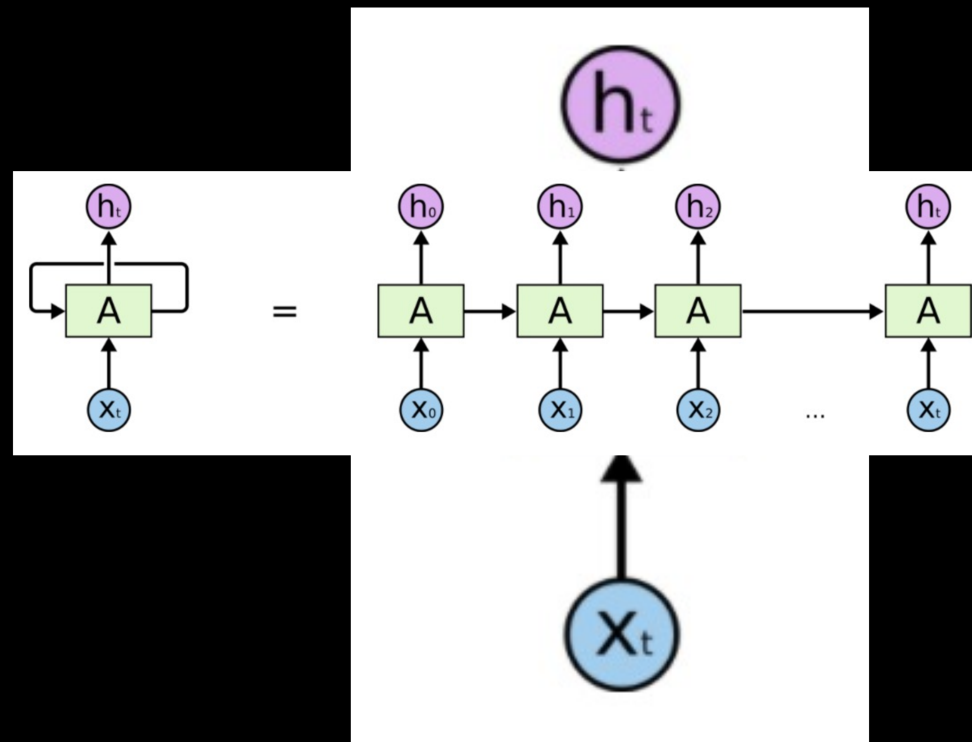


Figure from [3]

Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

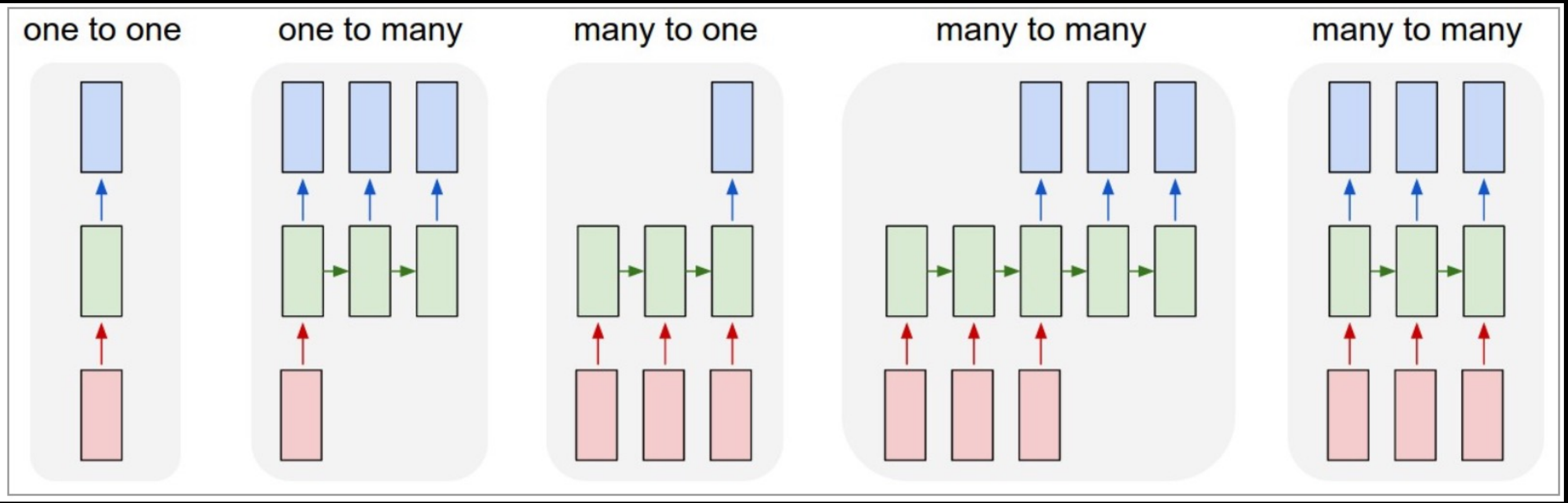
Recurrent Neural Networks

- In short, recurrent neural networks (RNNs) break the typical “directed acyclic” pedagogy of deep networks by introducing self-loops
  - Allows information to persist through multiple iterations
- We can get around problems introduced by loops by “unrolling” the loops
  - This permits backprop to work as usual



Recurrent Neural Network

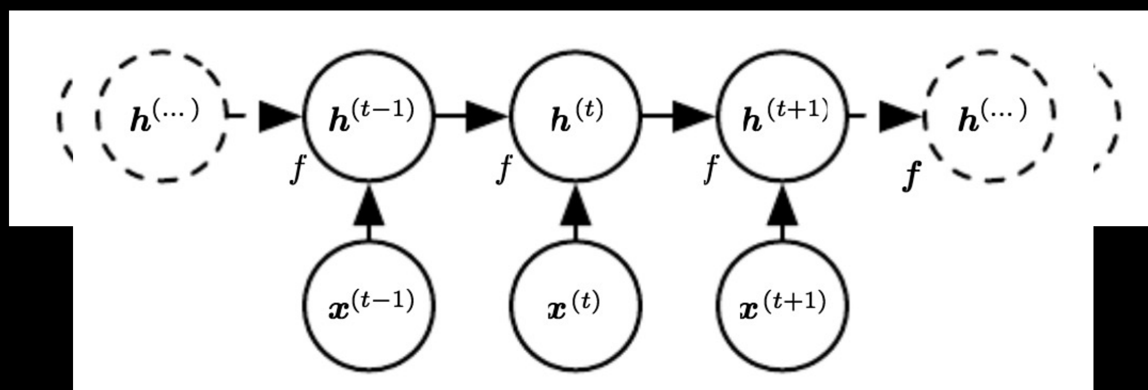
- “List” structure intrinsically handles variable-length data



- Think: convolution, but over time instead of space

Recurrent Neural Networks

- Use the same "parameter sharing" as CNNs
  - And linear dynamical systems!



- f maps each time point to the next
- Also updates internal state h

Recurrent Neural Networks

- Four main equations at each time point

$$\vec{a}^{(t)} = \vec{b} + W\vec{h}^{(t-1)} + U\vec{x}^{(t)}$$
$$\vec{h}^{(t)} = \sigma(\vec{a}^{(t)})$$
$$\vec{o}^{(t)} = \vec{c} + V\vec{h}^{(t)}$$
$$\hat{\vec{y}}^{(t)} = \phi(\vec{o}^{(t)})$$

Bias term

Internal RNN state

Input

Update internal state

Activation

Weights for

Bias term

Internal RNN state

Predicted output

Final layer activation

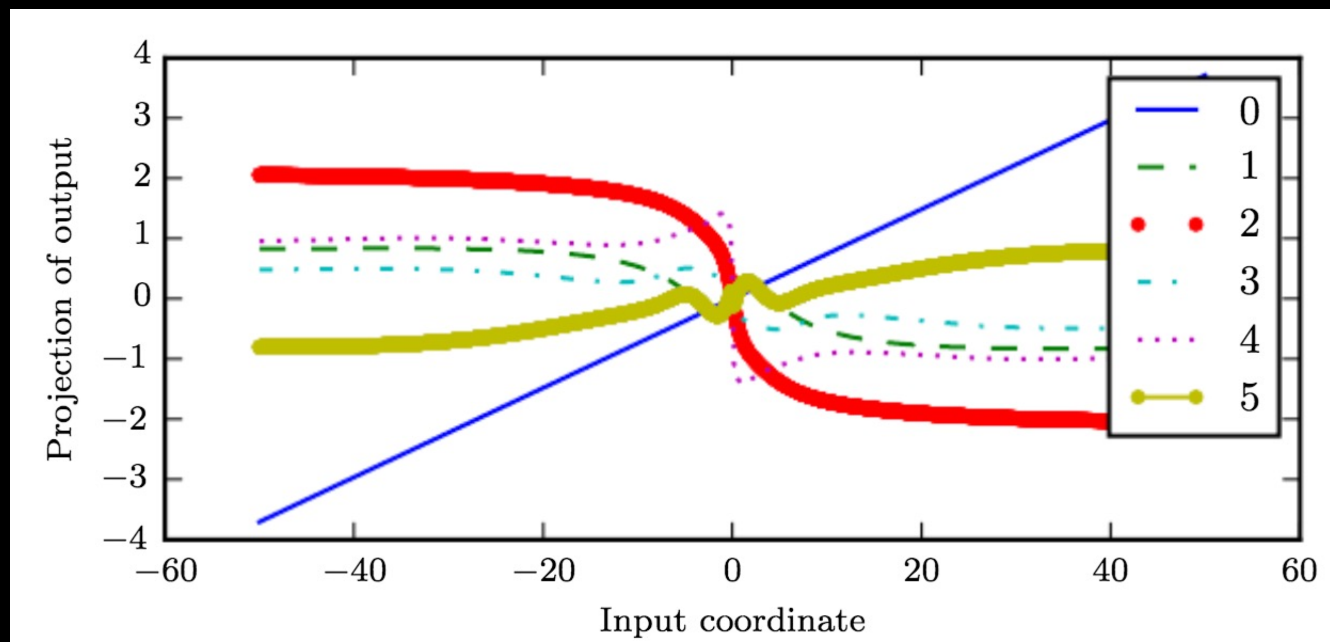
connections

Recurrent Neural Networks

- RNNs are great for modeling sequences, but by themselves cannot capture *attention*
- **Long-term dependencies require an explicit "memory"**

Long-term Dependencies

- RNNs *compose* the same activation function repeatedly
  - Think: recurrence relations
- Results in highly nonlinear behavior



Long-term Dependencies

- Put another way, recall the internal state update:

$$\vec{h}^{(t)} = W^T \vec{h}^{(t-1)}$$

- Where have we seen this before…

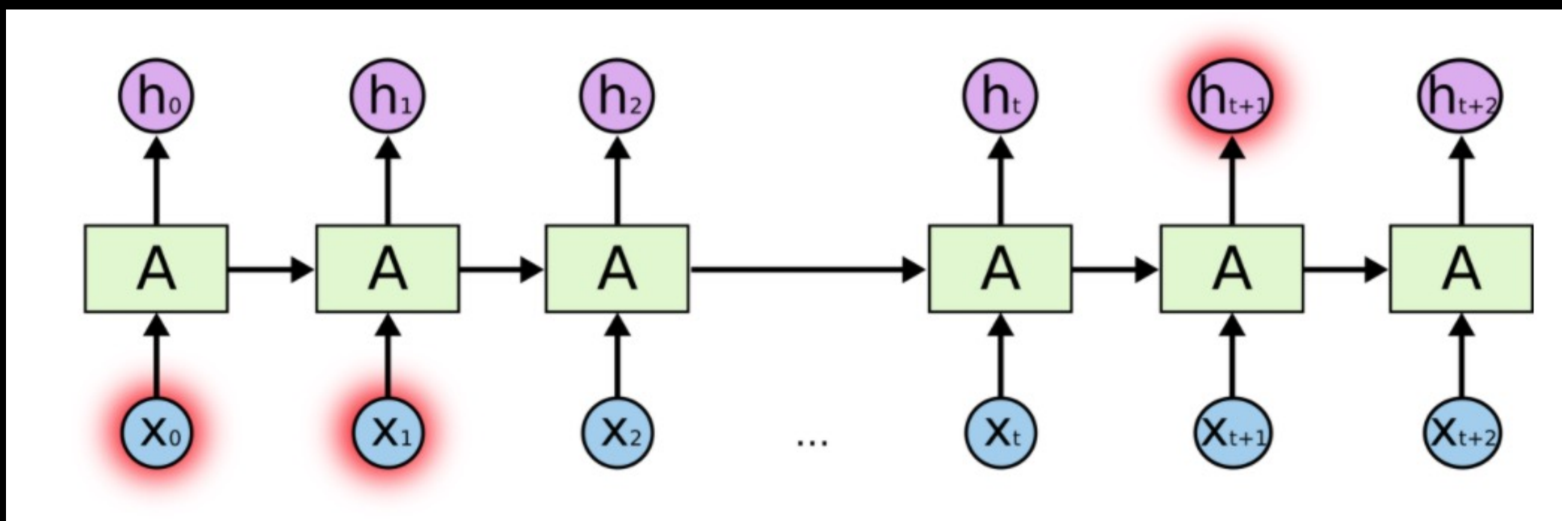
$$\vec{h}^{(t)} = (W^t)^T \vec{h}^{(0)} \qquad W = X \Lambda X^T$$

$$\boxed{\vec{h}^{(t)} = X^T \Lambda^t X \vec{h}^{(0)}}$$

- Eigenvalues are raised to the power *t*, decaying any eigenvalue < 1
- **Any component of $h^{(0)}$ not aligned with largest eigenvalue will be discarded**

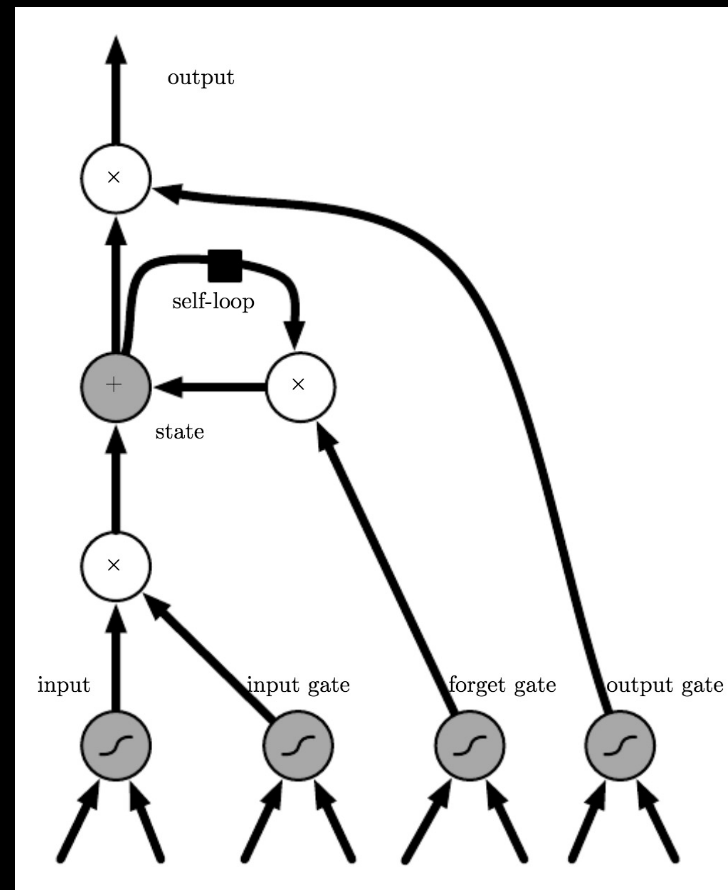
Long-term Dependencies

- "I grew up in France… I speak fluent **French**."



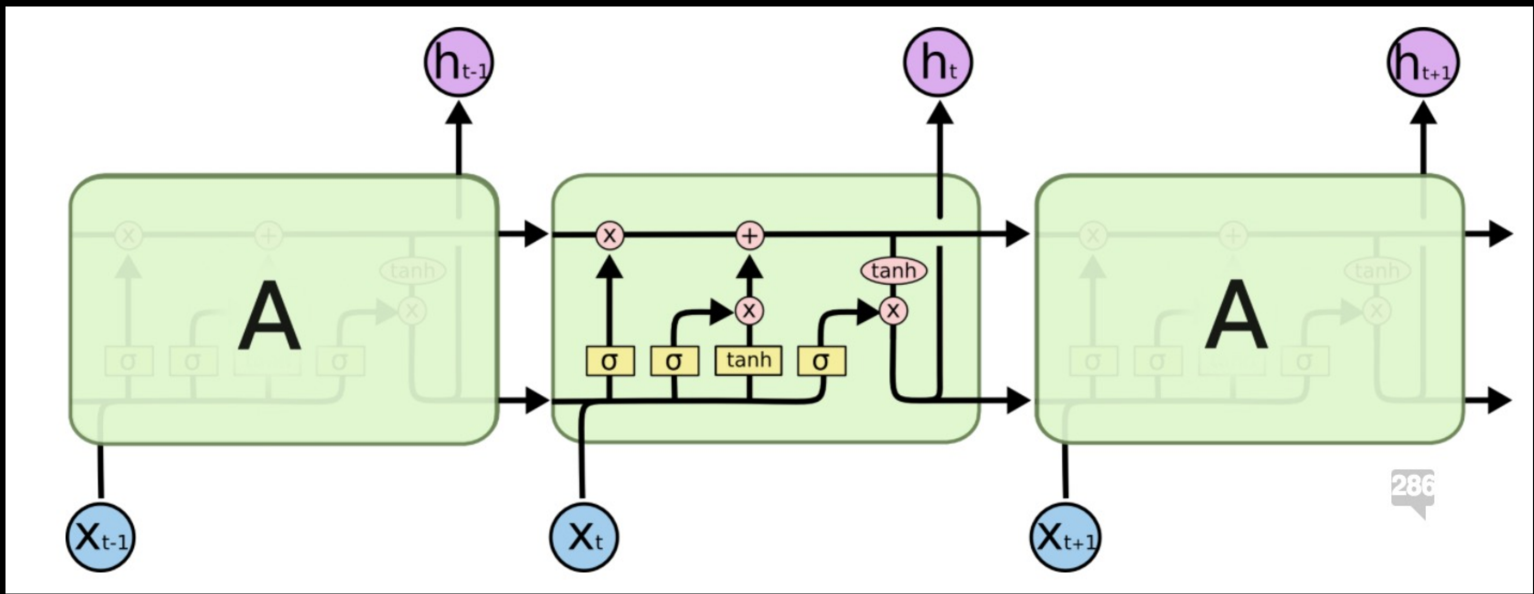
Long-Short Term Memory

- Or “LSTM”
- A variant of the *gated* RNN
- Each hidden state comprises a **forget** gate
  - Determines what to “remember” and what to discard
  - Functions on self-loop input



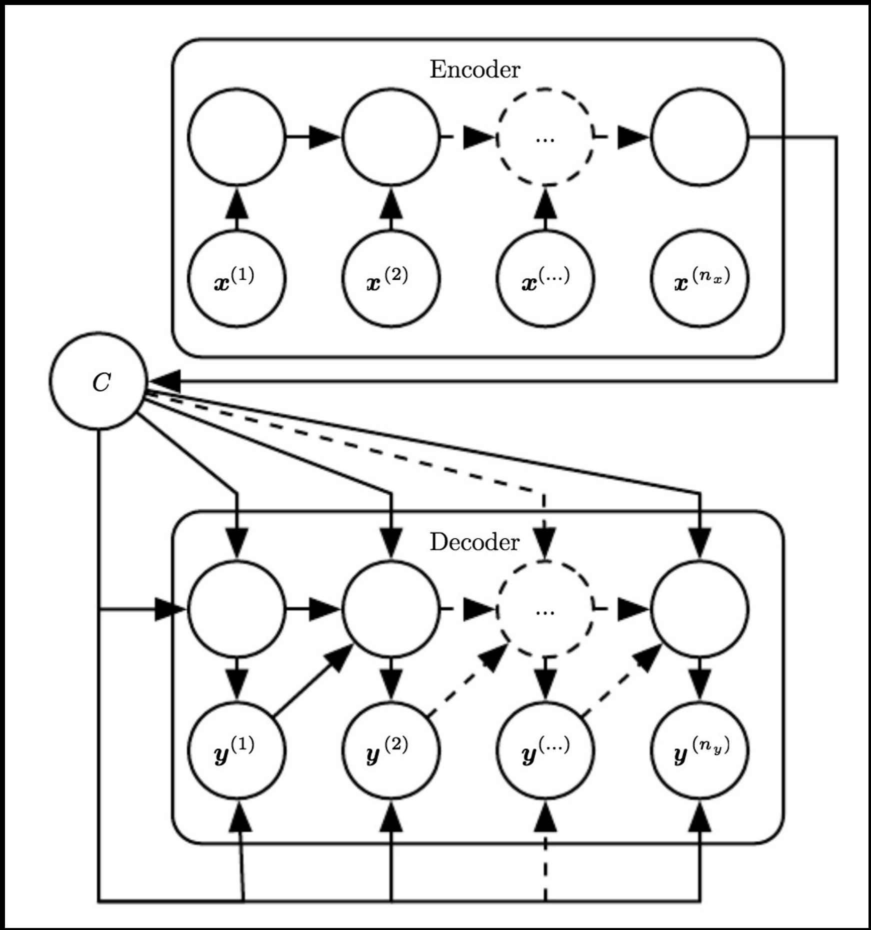
LSTM versus “vanilla” RNN

- A “vanilla” RNN contains only a single activation
- LSTMs have four interacting layers in each step



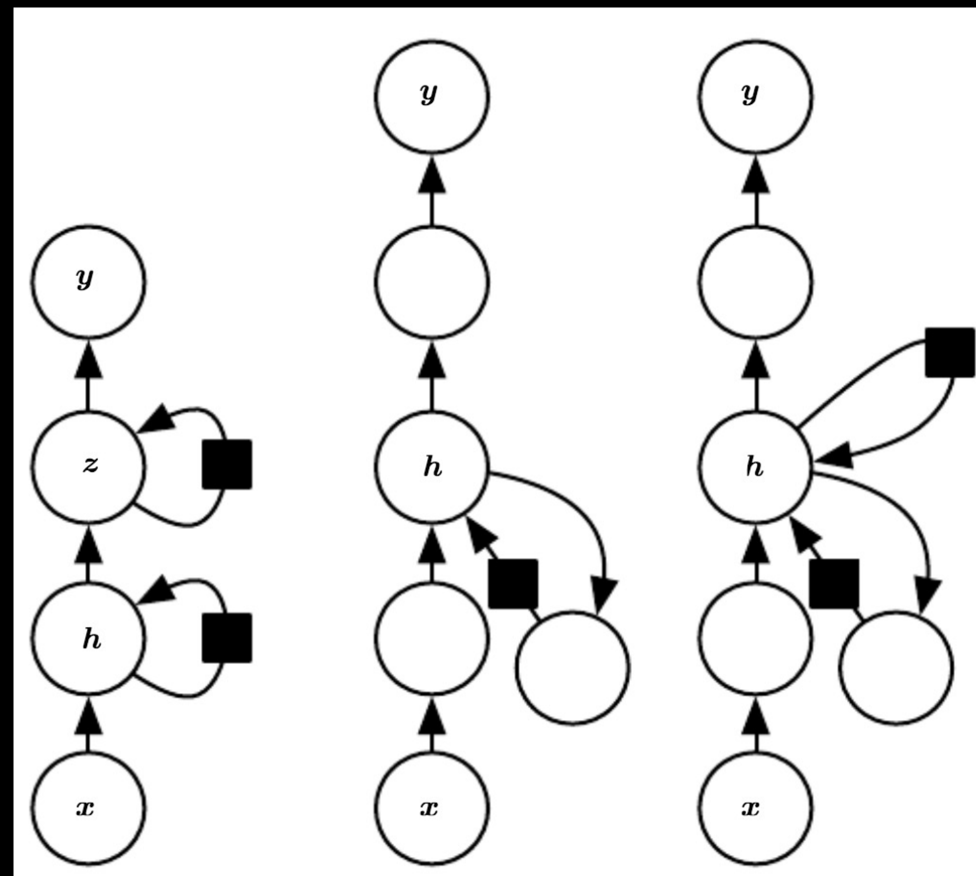
Encoder-Decoder Networks

- Maps input to output sequences
  - Each mapping not necessarily of equal length!
- *C* is a "semantic summary"
  - Think: input "subspace"
- Have to ensure *C* is of sufficient dimensionality to represent input space



Deep Recurrent Networks

- Each recurrent state can feed into a series of hidden states
- Analogous to hidden markov models (HMMs) with attention and nearly infinite support for hidden states



Conclusions

- Recurrent neural networks
  - A generalization of convolution (or is a convolution a generalization of recurrence?): uses same **parameter-sharing** idea
  - Introduces self-loops, but over discrete intervals: loops can be “unrolled” so backpropagation can still be used as normal
  - Still have trouble with long-term dependencies, such as language translation (vanishing / exploding gradient)
- Long-short term memory
  - Introduce a series of gates within the self-loops
  - Gates determine what to remember, what to discard
  - No ill-conditioned gradients
- Attention + Encoder-Decoder Networks
  - Starting to see the foundations for modern Transformers

References

- Deep Learning Book, Chapter 10: "Sequence Modeling: Recurrent and Recursive Nets", http://www.deeplearningbook.org/contents/rnn.html
- "Attention and Augmented Recurrent Neural Networks", https://distill.pub/2016/augmented-rnns/
- "Understanding LSTM Networks" https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- "The Unreasonable Effectiveness of Recurrent Neural Networks" https://karpathy.github.io/2015/05/21/rnn-effectiveness/