

CMPT 733

# Further Topics in Deep Learning

Sequence learning, Sentiment analysis, Word2Vec, DL-Vis

Steven Bergner

22 March 2021

# Overview

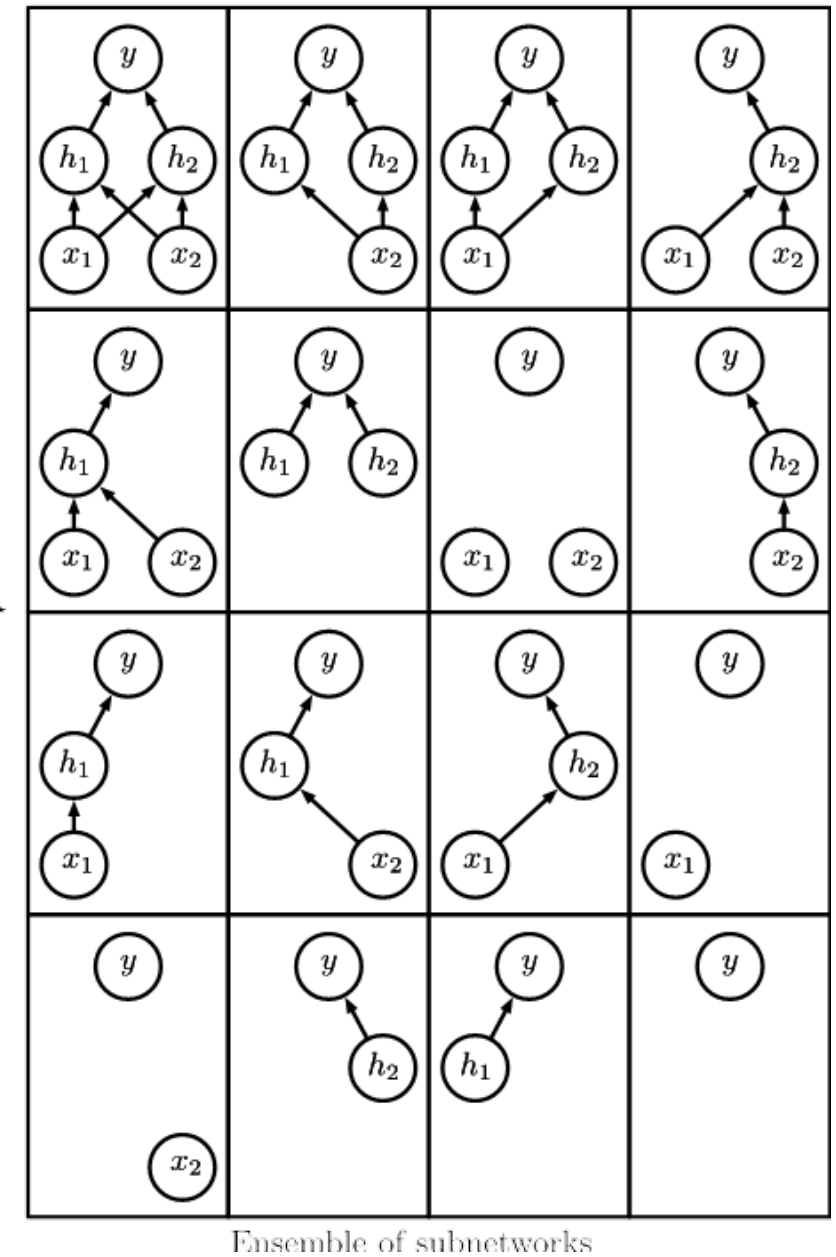
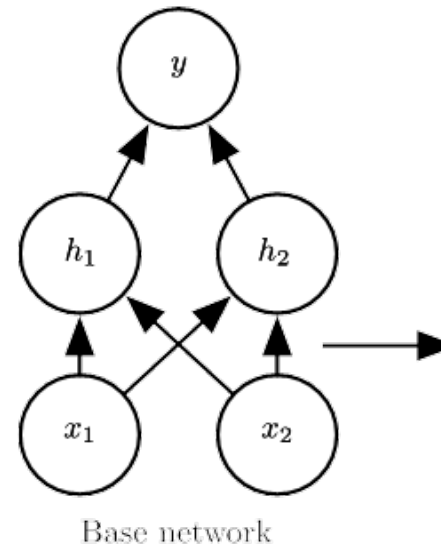
- Recap: Overfitting remedies
- Deep learning for sequences
- Natural language processing, e.g.
  - Sentiment analysis
  - Word embeddings
- Visualization for Deep Learning

# Strategies against Overfitting (continued)

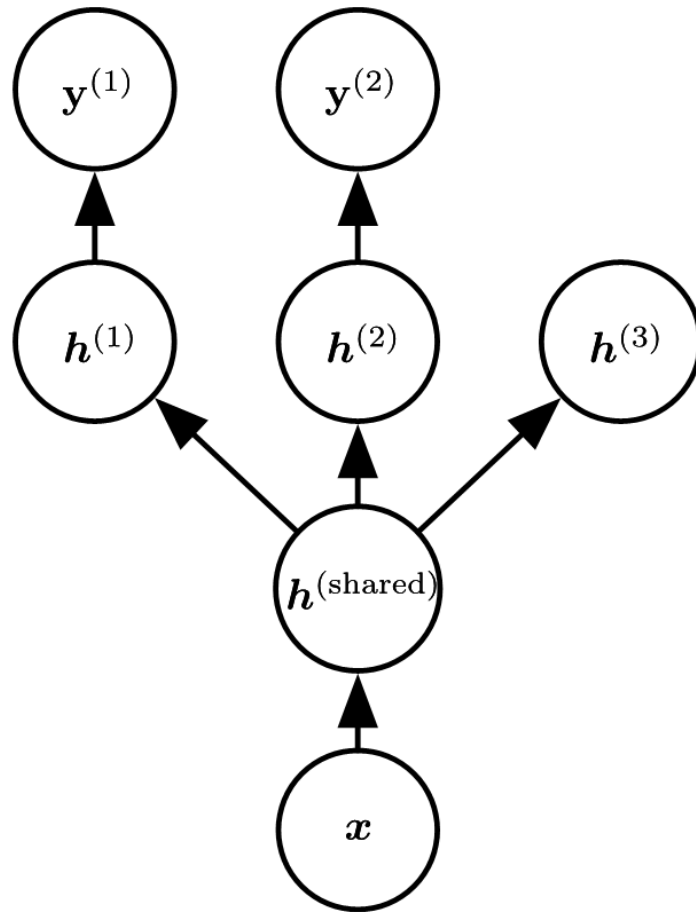
Lower generalization error  
without impacting training error

# Dropout

- Random sample of connection weights is set to zero
- Train different network model each time
- Learn more robust, generalizable features



# Multitask learning



- Shared parameters are trained with more data
- Improved generalization error due to increased statistical strength
- Missing components of  $y$  are masked from the loss function

# Components of popular architectures

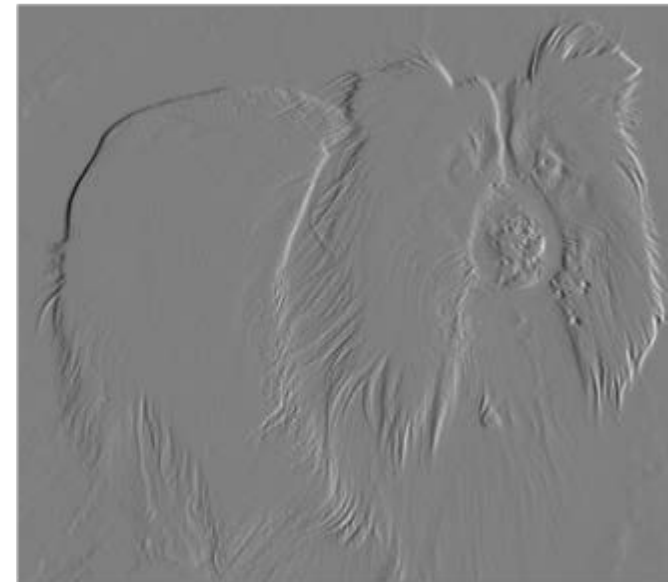
# Convolution as edge detector



Input

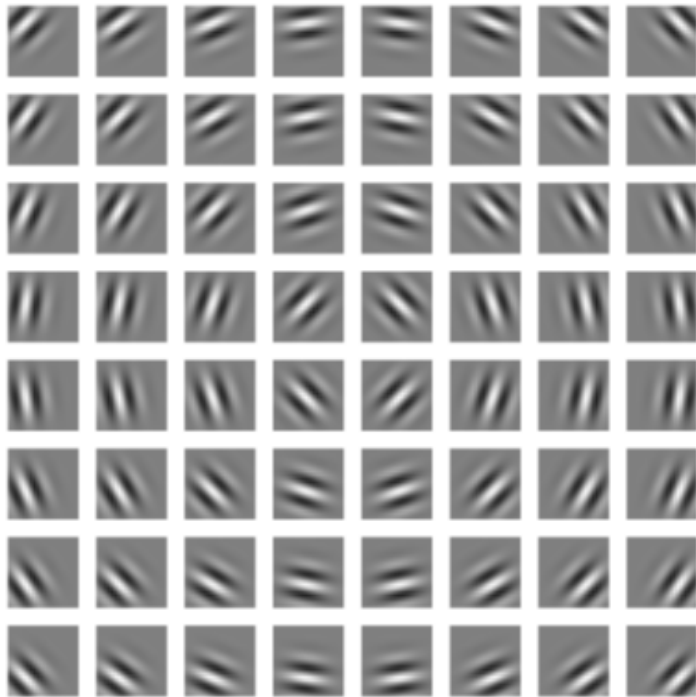
1	-1
---	----

Kernel

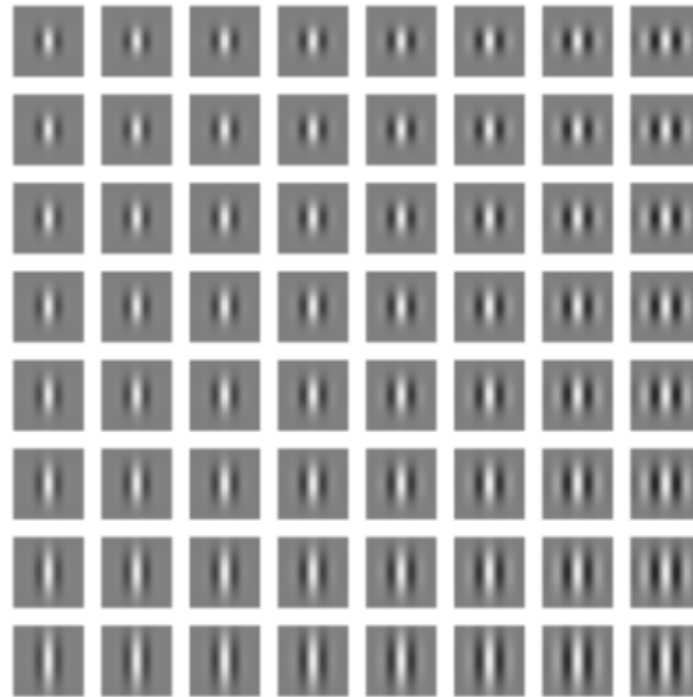


Output

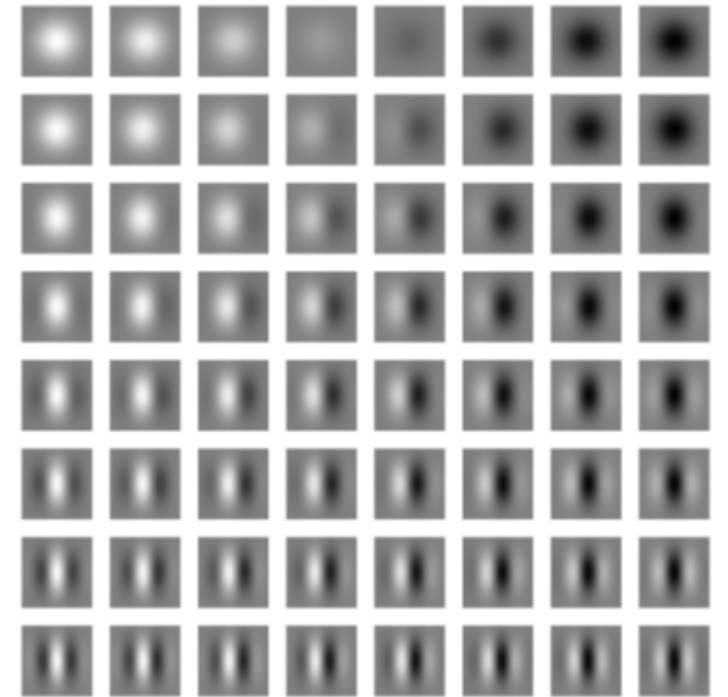
# Gabor wavelets (kernels)



Directional second  
derivative



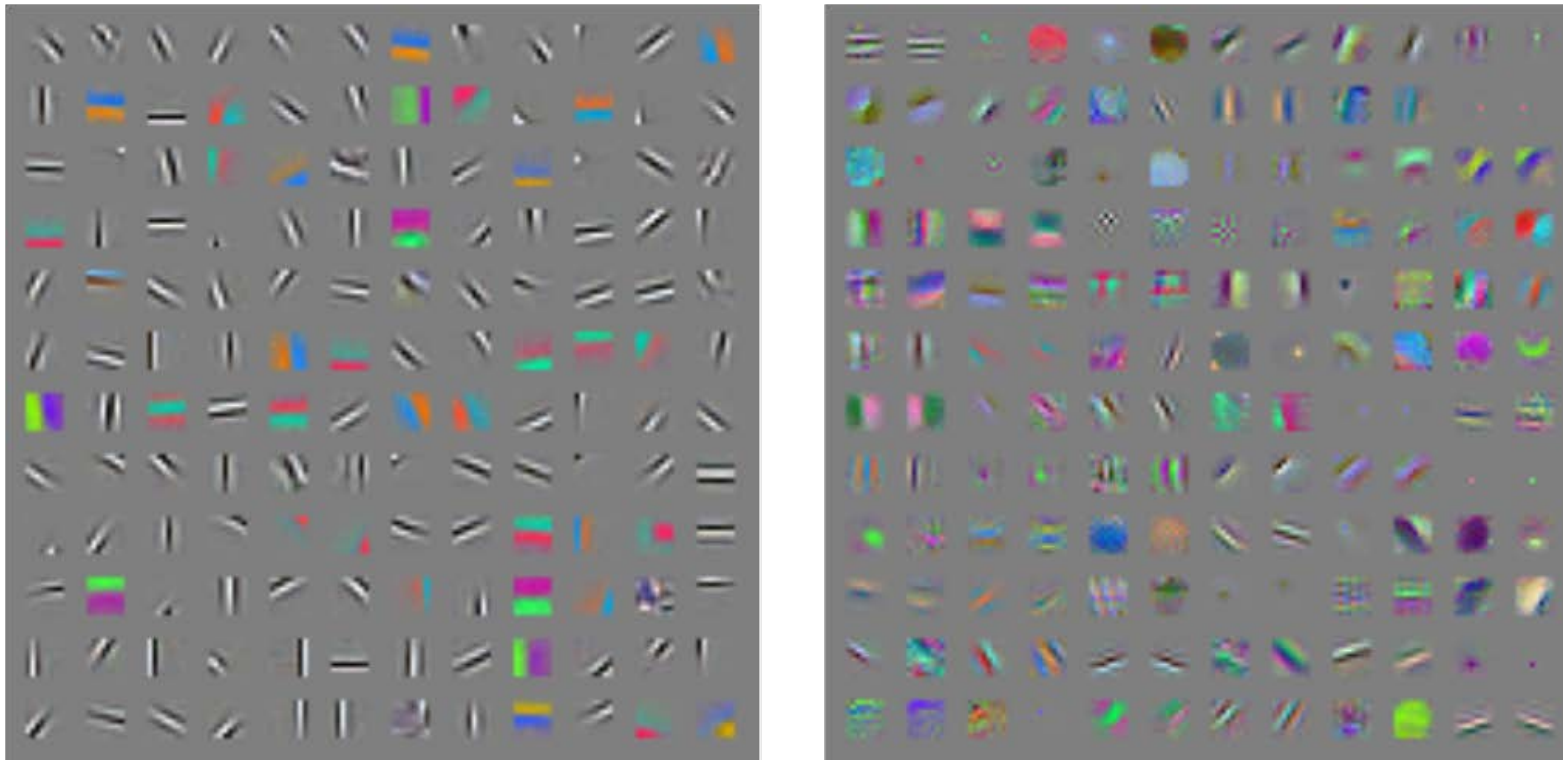
Second derivative  
(curvature)



Local average, first  
derivative

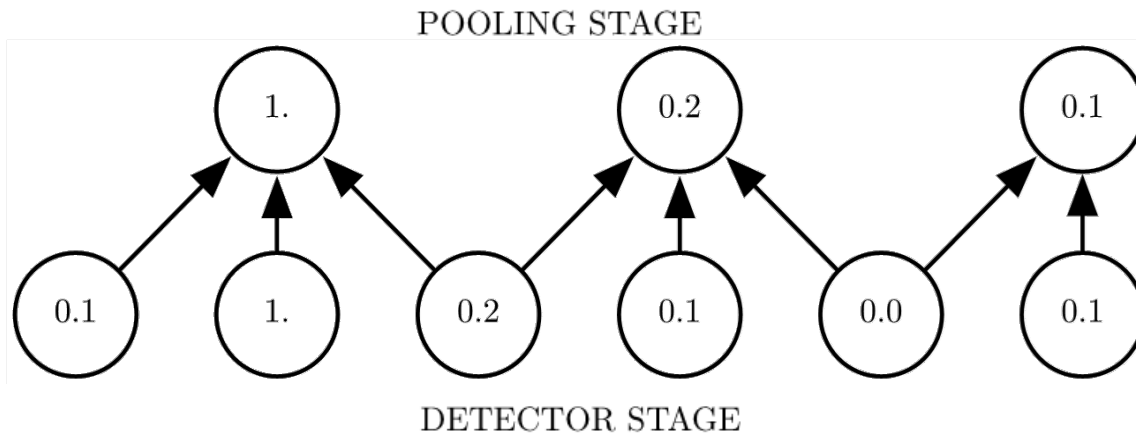


# Gabor-like learned kernels

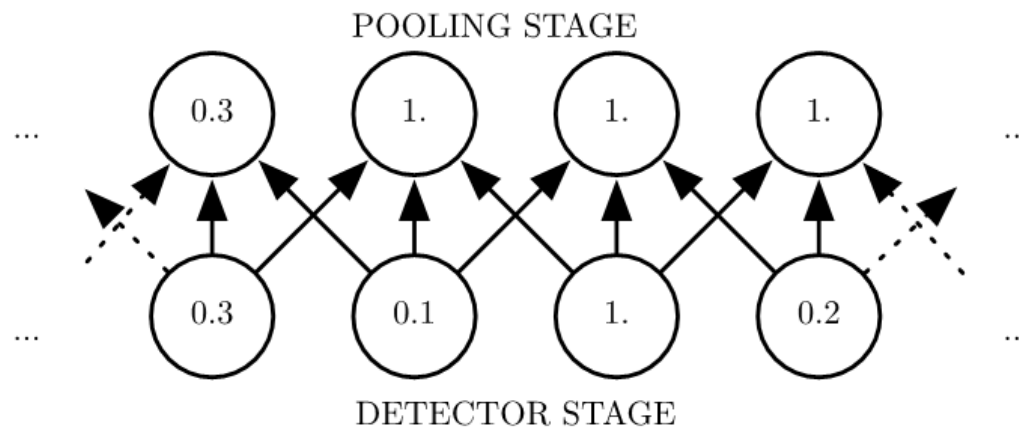


- Features extractors provided by pretrained networks

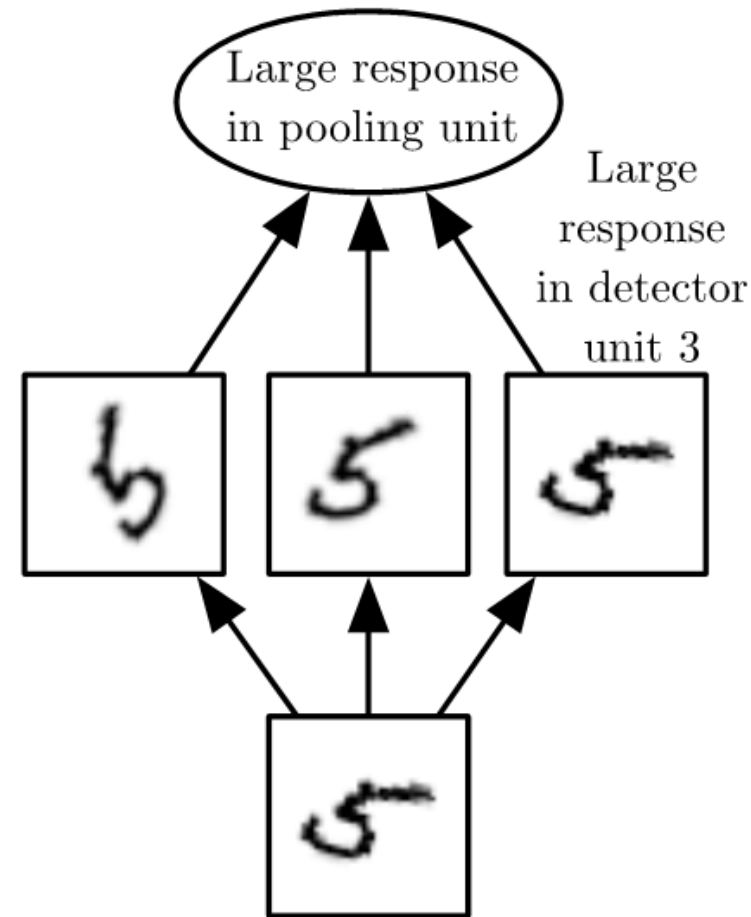
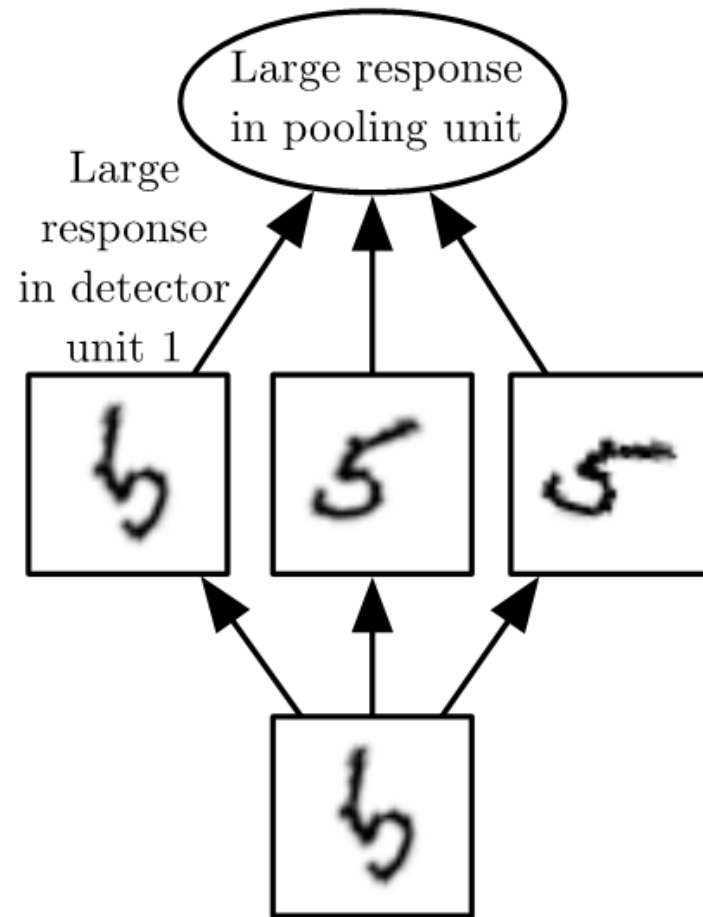
# Max pooling translation invariance



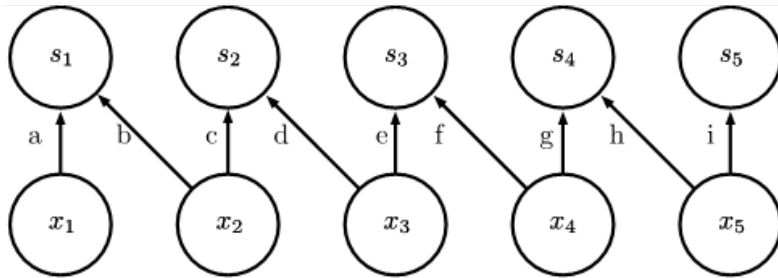
- Take max of certain neighbourhood
- Often combined, followed by downsampling



# Max pooling transform invariance



# Types of connectivity




Local connection:  
like convolution,  
but no sharing

$$\begin{bmatrix} a & b & & \\ & c & d & \\ & & e & f & \end{bmatrix}$$

$$\begin{bmatrix} a & b & & \\ & a & b & \\ & & a & b & \end{bmatrix}$$

$$\begin{bmatrix} a & b & c & d & \dots \\ h & i & j & k & \dots \\ o & p & q & r & \dots \end{bmatrix}$$

# Convolution calculation illustrated



A hand-drawn step plot representing the input sequence. The plot starts at a low level, steps up at the first two positions (1, 1), steps up again at the third position (2), and reaches its maximum level at the fourth position (4). It remains at this maximum level for the next three positions (5, 5, 5), then steps down at the eighth position (3), and returns to the initial low level for the final two positions (0, 0).

$$\begin{array}{r} [1 \ 1 \ 2 \ 4 \ 5 \ 5 \ 5 \ 3 \ 0 \ 0] \\ * [-1 \ 1] \end{array}$$

# Choosing architecture family

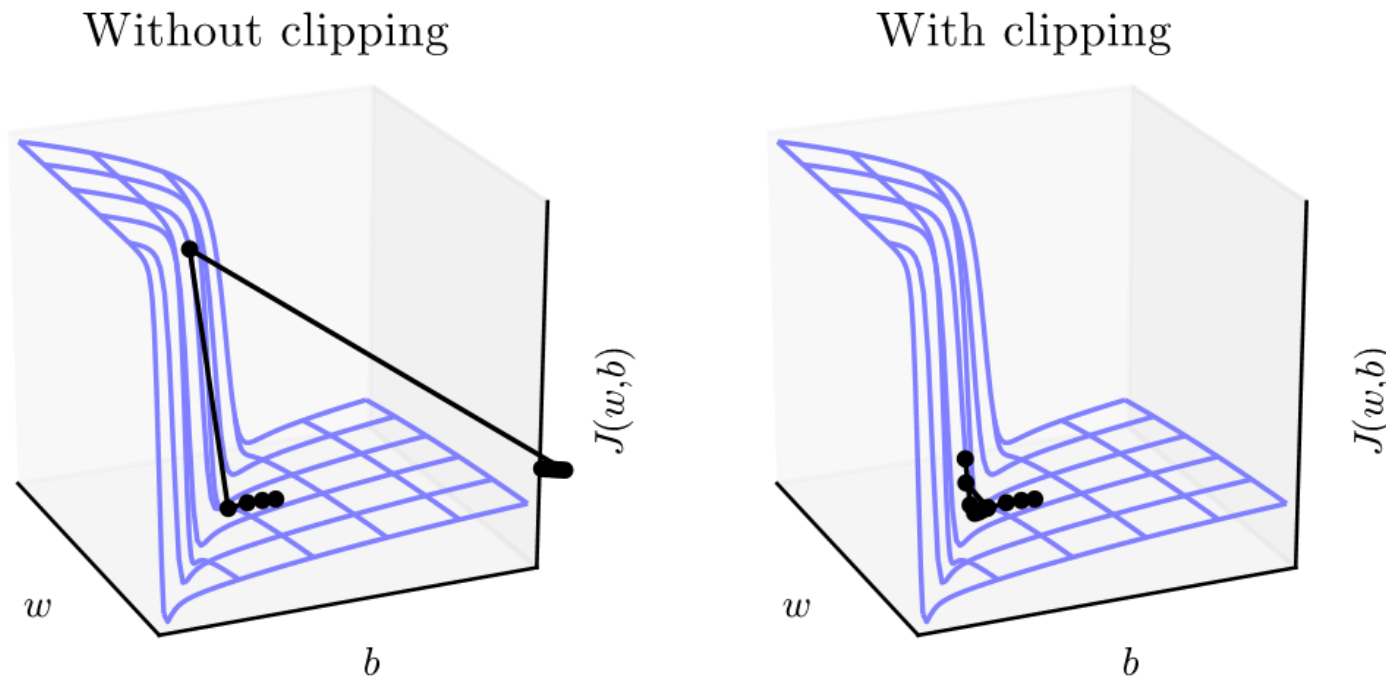
- No structure → fully connected
- Spatial structure → convolutional
- Sequential structure → recurrent

# Optimization Algorithm

- Lots of variants address choice of learning rate
- See [Visualization of Algorithms](#)
- AdaDelta and RMSprop often work well

# Gradient Clipping

- Add learning rate time gradient to update parameters
- Believe direction of gradient, but not its magnitude





# Development strategy

- Identify needs: High accuracy or low accuracy?
- Choose metric
  - Accuracy (% of examples correct), Coverage (% examples processed)
  - Precision  $TP/(TP+FP)$ , Recall  $TP/(TP+FN)$
  - Amount of error in case of regression
- Build end-to-end system
  - Start from baseline, e.g. initialize with pre-trained network
- Refine driven by data

# Software for Deep Learning

# Current Frameworks

- [Tensorflow / Keras](#)
- [PyTorch](#)
- DL4J
- Caffe (superseded by Caffe2, which is merged into PyTorch)
- [And many more](#)
- Most have CPU-only mode but much faster on NVIDIA GPU

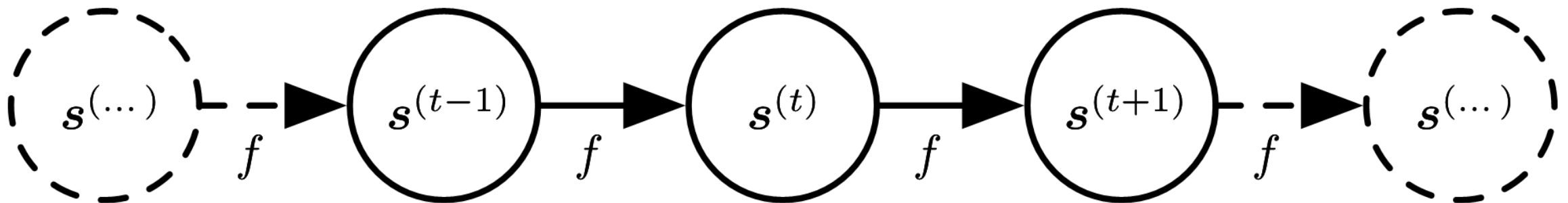
# Recap: Choosing architecture family

- No structure → fully connected
- Spatial structure → convolutional
  - Adjacency or order of inputs has meaning
- Sequential structure → recurrent

# Sequence Modeling with Recurrent Nets

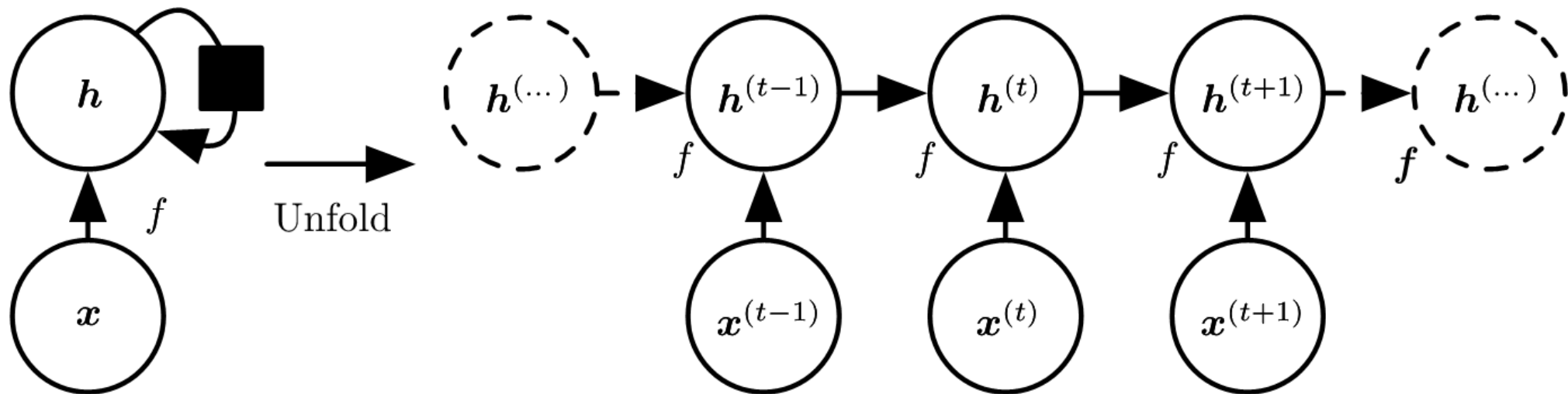
# Classical Dynamical Systems

- Recurrent network models a dynamical system that is updated in discrete steps over time
- Function  $f$  takes input from time  $t$  to output at time  $t+1$
- Rules persist across time



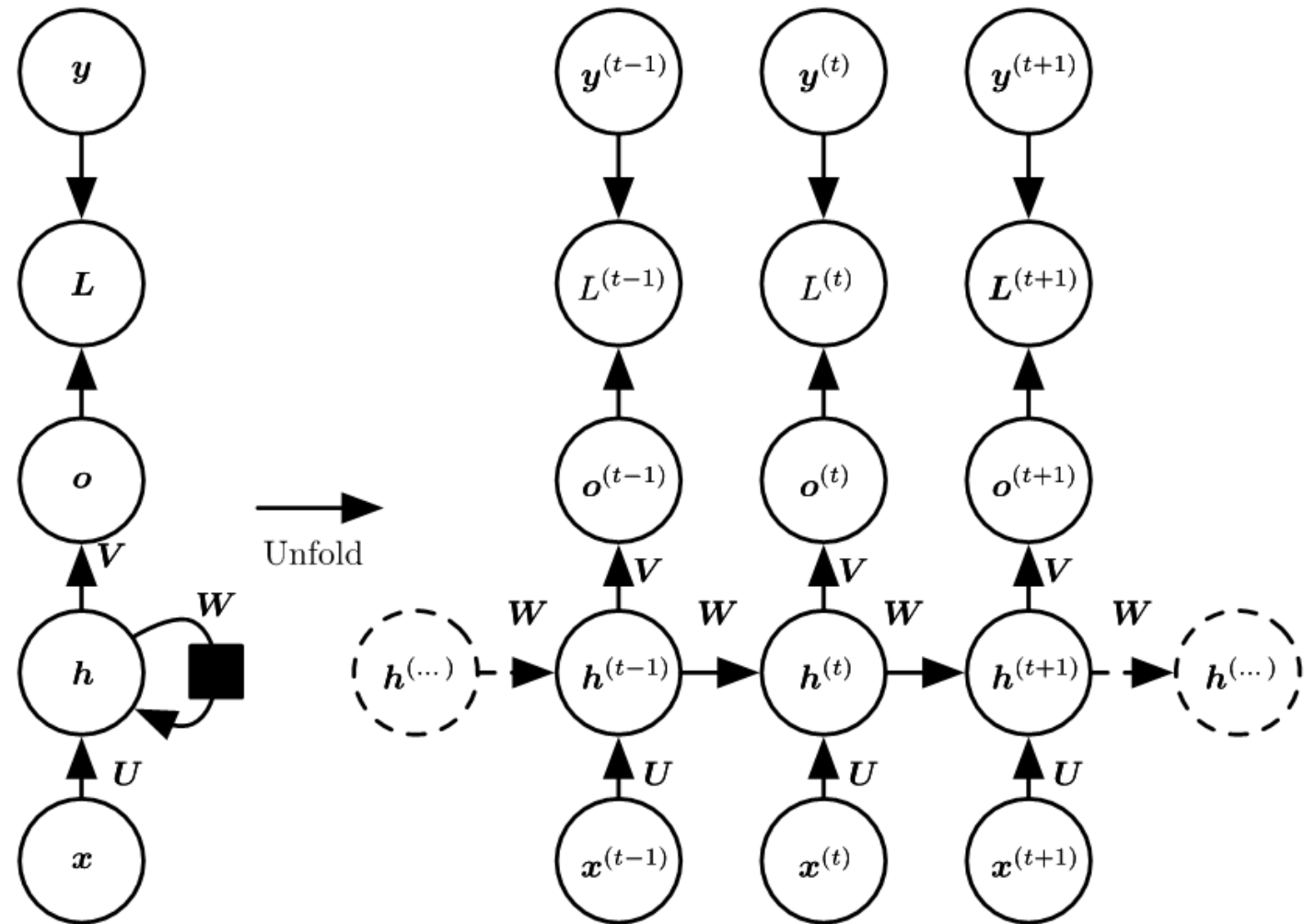
# Unfolding Computation Graphs

- Recurrent graph can be unfolded, where hidden state  $h$  is influencing itself
- Backprop through time is just backprop on unfolded graph



# Recurrent Hidden Units

- Can have more than one layer

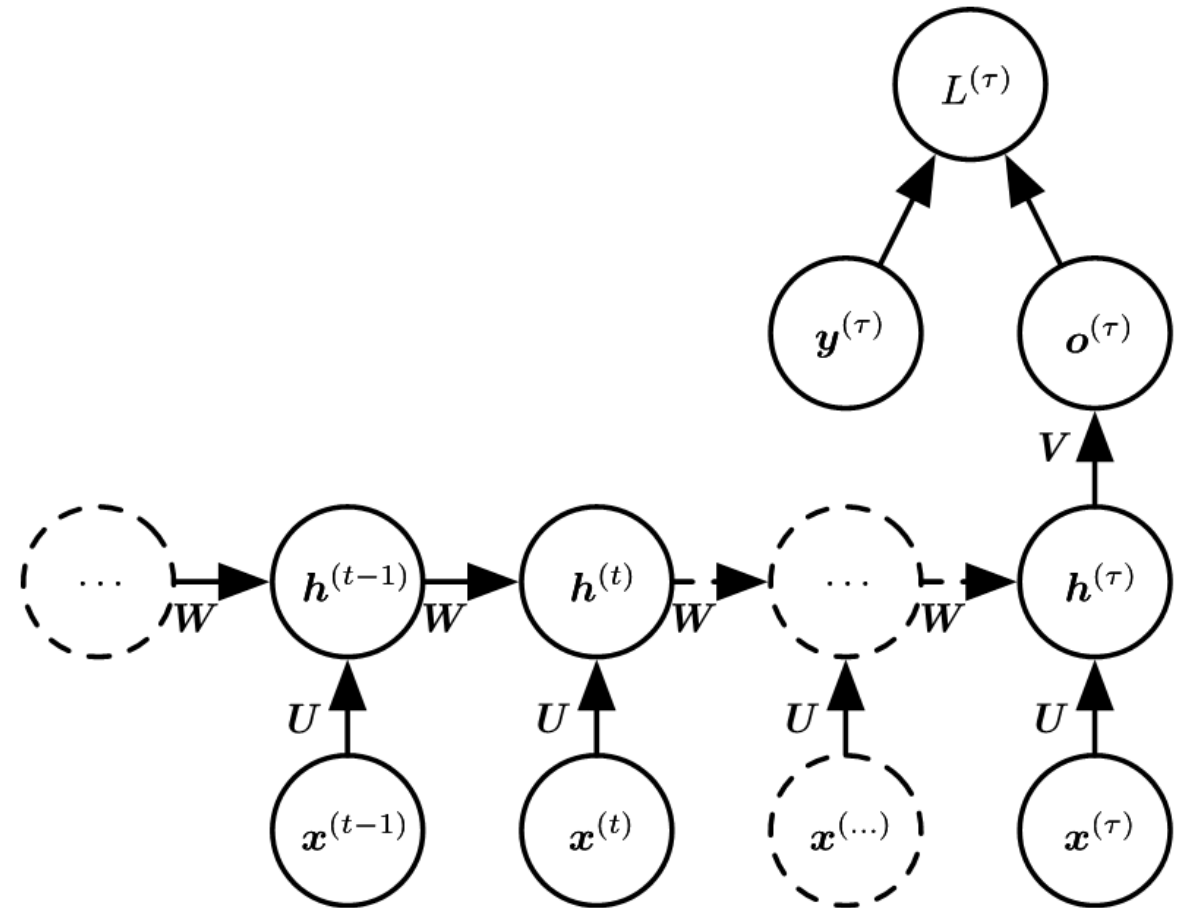




# Sequence Input, Single Output

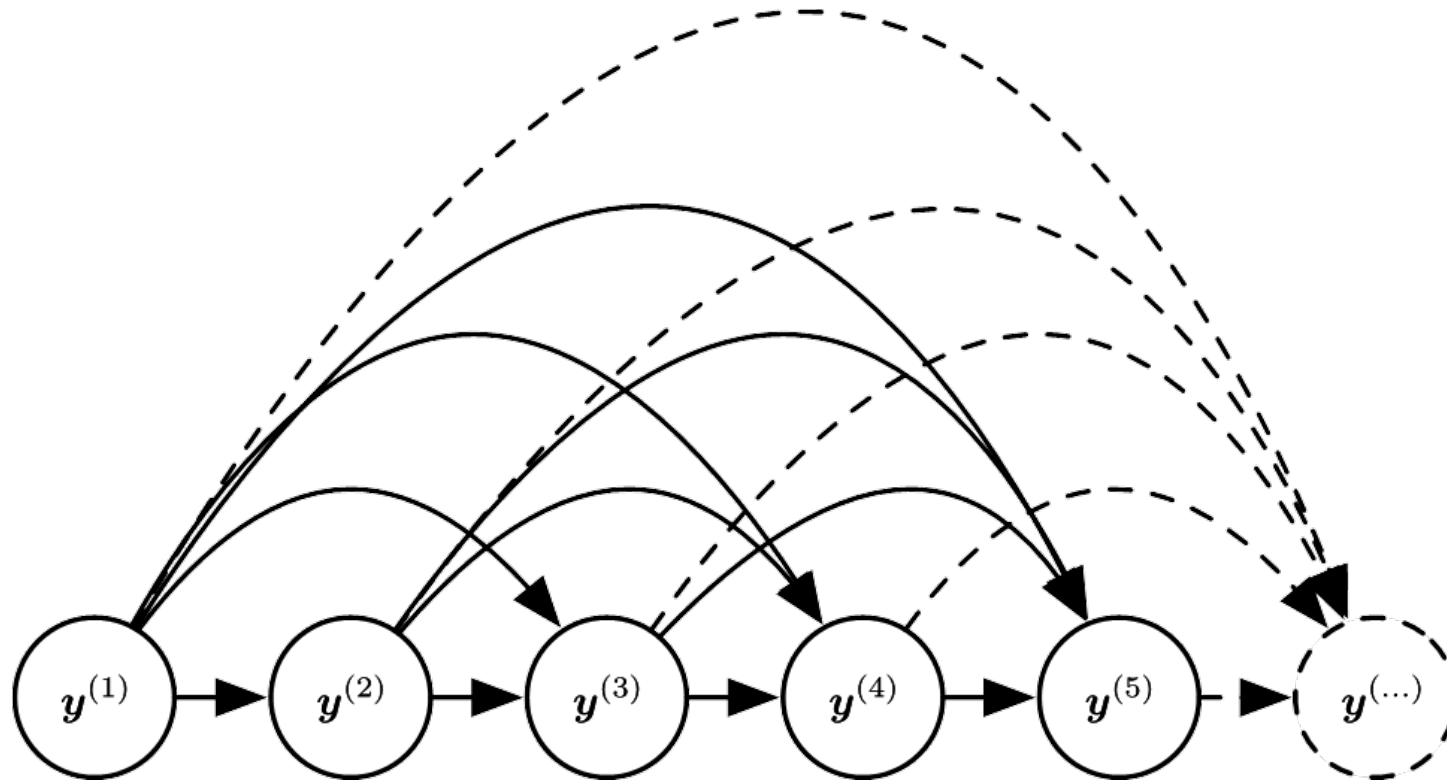
## Example

Sentiment analysis of text



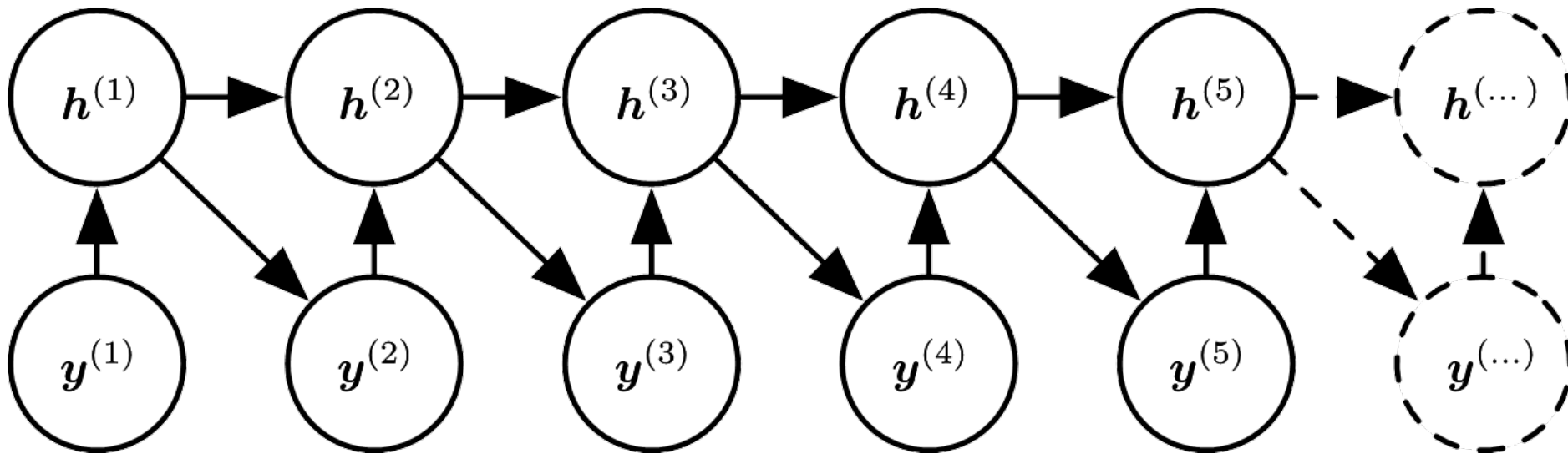
# Fully Connected Graphical Model

- Too many dependencies among variables, if each has its own set of parameters



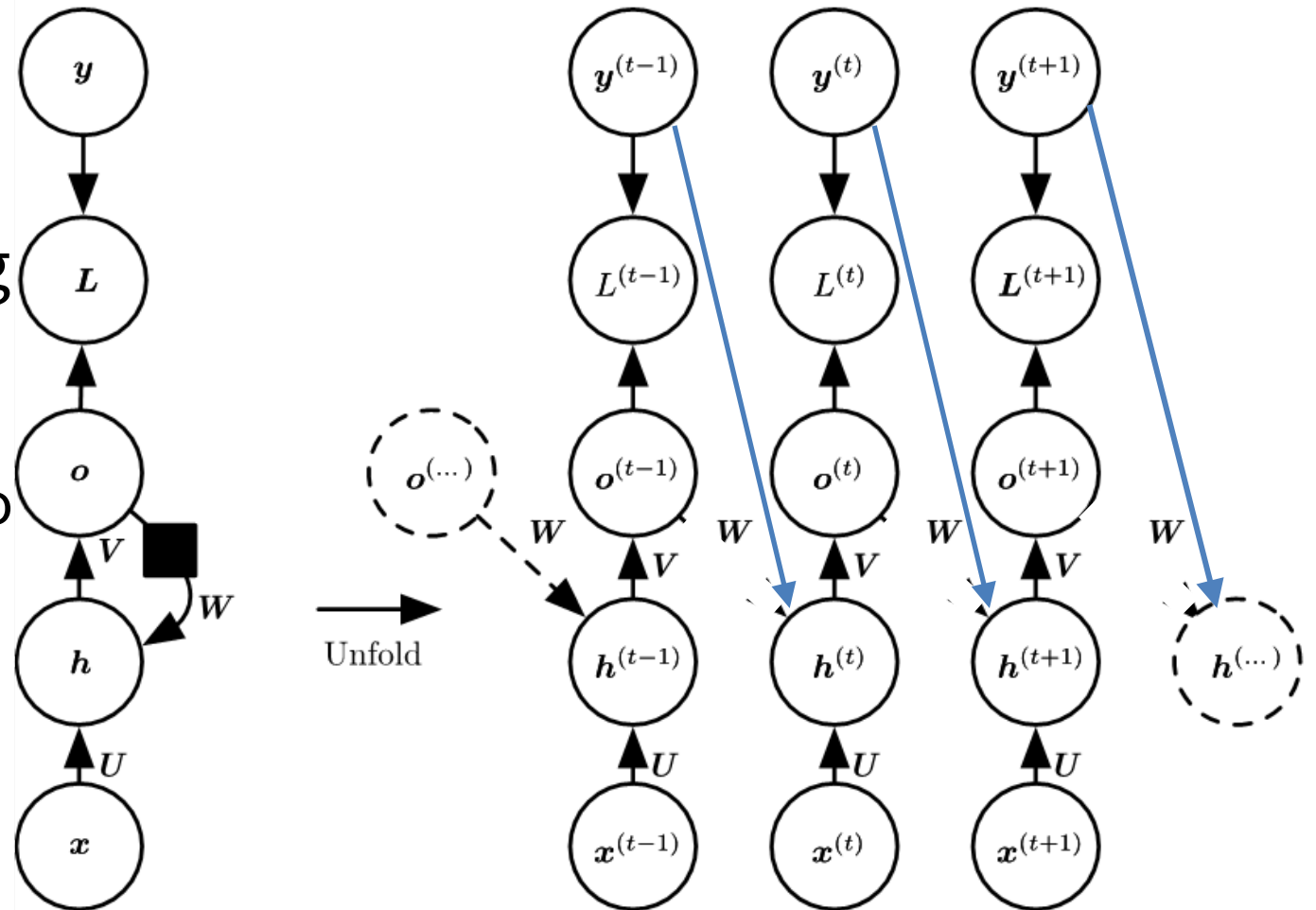
# RNN Graphical Model

- Organize variables according to time with single update rule
- Finite set of relationships may extend to infinite sequences
- $h$  acts as “memory state” summarizing relevant history



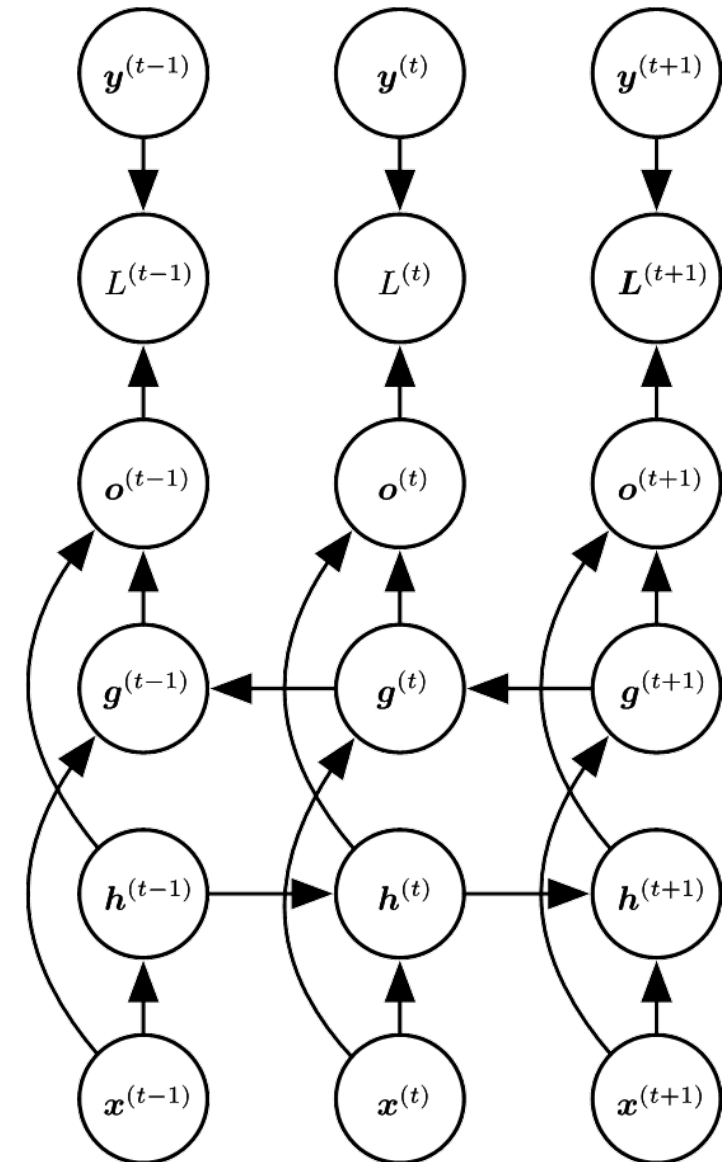
# Recurrence only through output

- Avoid backprop through time
- Mitigation: Teacher forcing
  - Use actual or expected output from the training dataset at current time  $y(t)$  as input  $o(t)$  to the next time step, rather than generated output
  - Backprop stops when it reaches  $y(t-1)$  via  $o(t-1)$



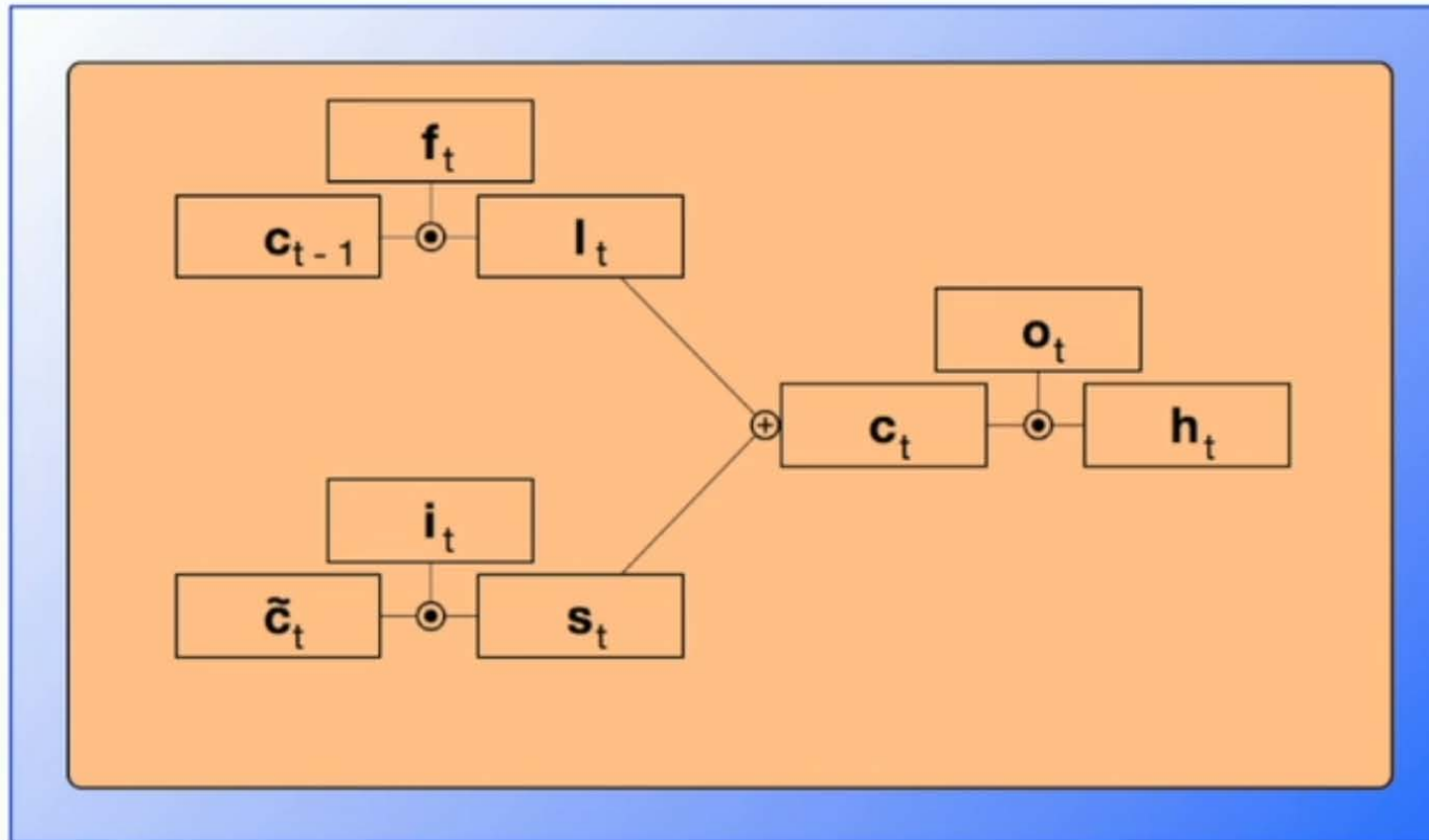
# Bidirectional RNN

- Later information may be used to reassess previous observations



# LSTMs

- Use addition over time instead of multiplication



Forget Gate  $f_t = \dots$   
Remember Gate  $i_t = \dots$   
Output Gate  $o_t = \dots$   
Cell Input  $\tilde{c}_t = \dots$   
Long-term Memory  $l_t = \dots$   
Short-term Memory  $s_t = \dots$   
Cell Memory  $c_t = \dots$   
Output  $h_t = \dots$

# Further Architectures

- [Transformers](#)
- [Deep Reinforcement Learning](#)

# Visualization for DL

- **Tensorboard: Visualizing Learning**
- How to use t-SNE efficiently

## **Model visualization**

- **LSTM-Vis:** <http://lstm.seas.harvard.edu/client/index.html>
- Building blocks of interpretability



# Sources

- I. Goodfellow, Y. Bengio, A. Courville “Deep Learning” MIT Press 2016 [[link](#)]
- Apala Guha’s CMPT 733 slides