CS 546 – Advanced Topics in NLP

Dilek Hakkani-Tür

University of Illinois Urbana-Champaign

CONVAI

CDS Siebel School of Computing and Data Science

Topics for Today

Sequence Modeling

- Convolutional Neural Networks
- Recurrent Neural Networks
- Example Applications

Convolutional Neural Networks (CNNs)

CONVAI

- We have discussed models that deal with paired data: input words or utterances and output categories.
- Example: One-hot representations or word embeddings to represent input words
- Sometimes data exhibits rich structure, such as images, natural language.
  - Structure-less networks, i.e., MLPs, can fall short.
- CNNs: a type of NNs well-suited to detecting spatial substructure.

Why CNNs? Example: Image Recognition

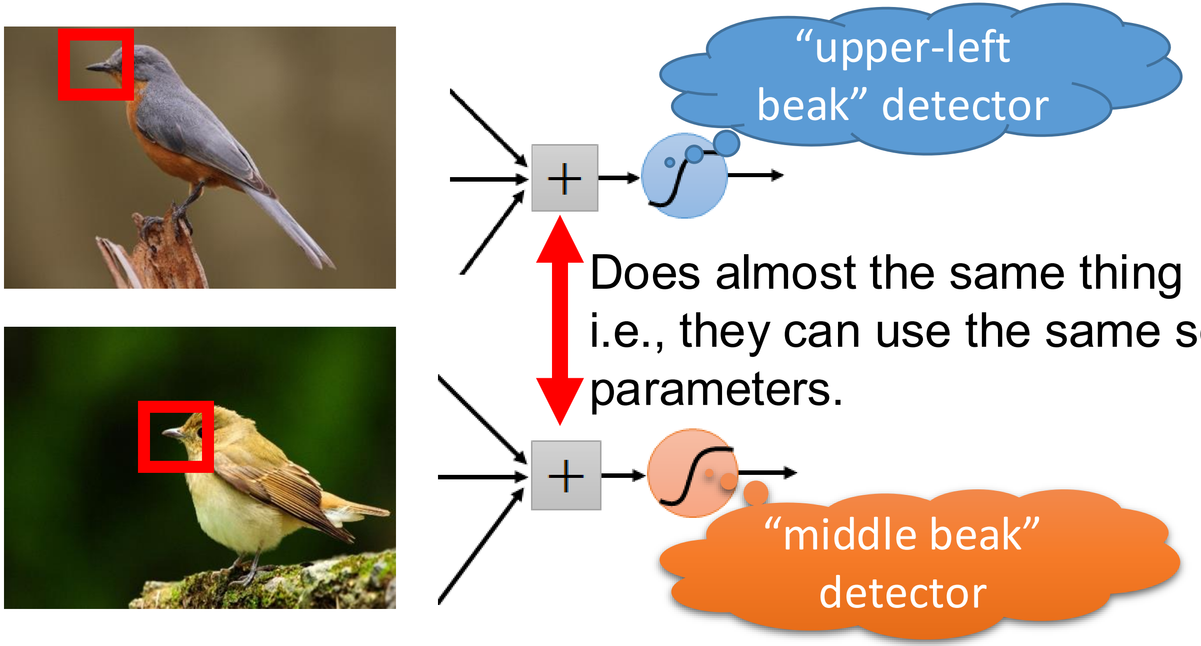
- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with fewer parameters

Why CNNs? Example: Image Recognition

- The same pattern can appear in different regions.



Why CNNs? Example: Text Classification

- Recognizing original nationalities from last names

  O'Neill **O'**Shaughnessy

  Anton**opoulos** Kost**opoulos** Gianni**opoulos**

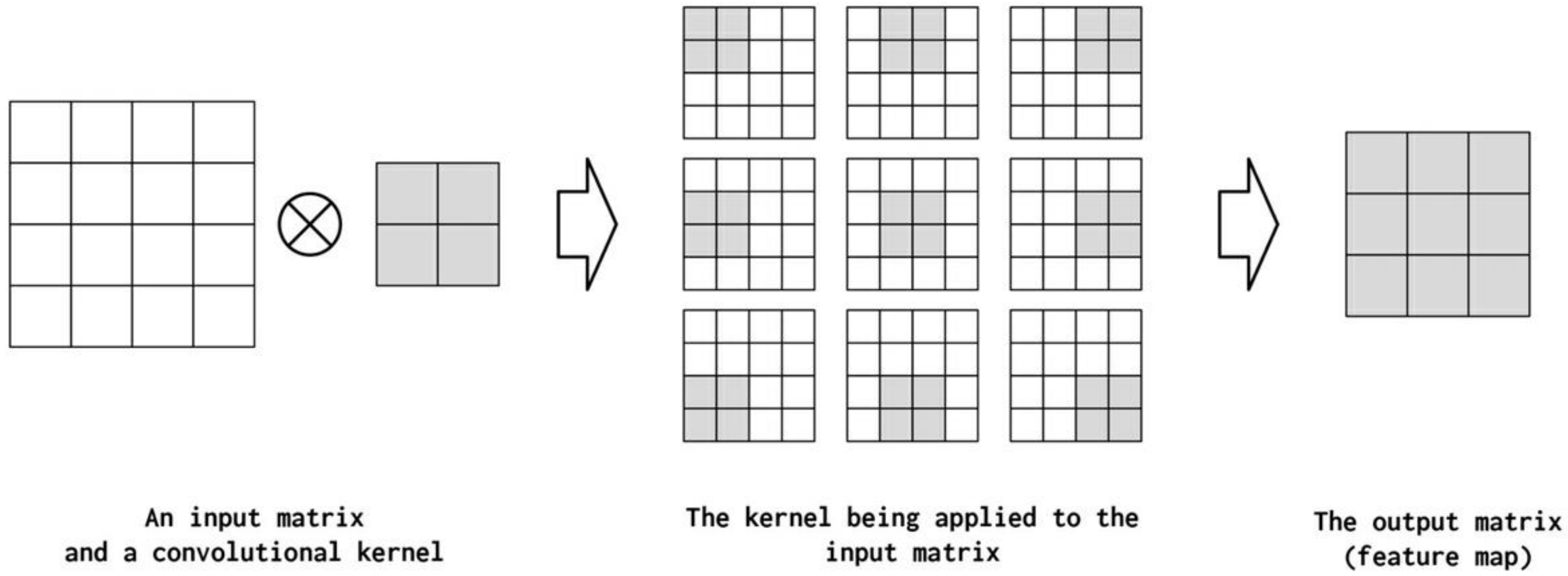
- Sentiment classification

  A **delicious** breakfast was served to us at Pillerago that morning.

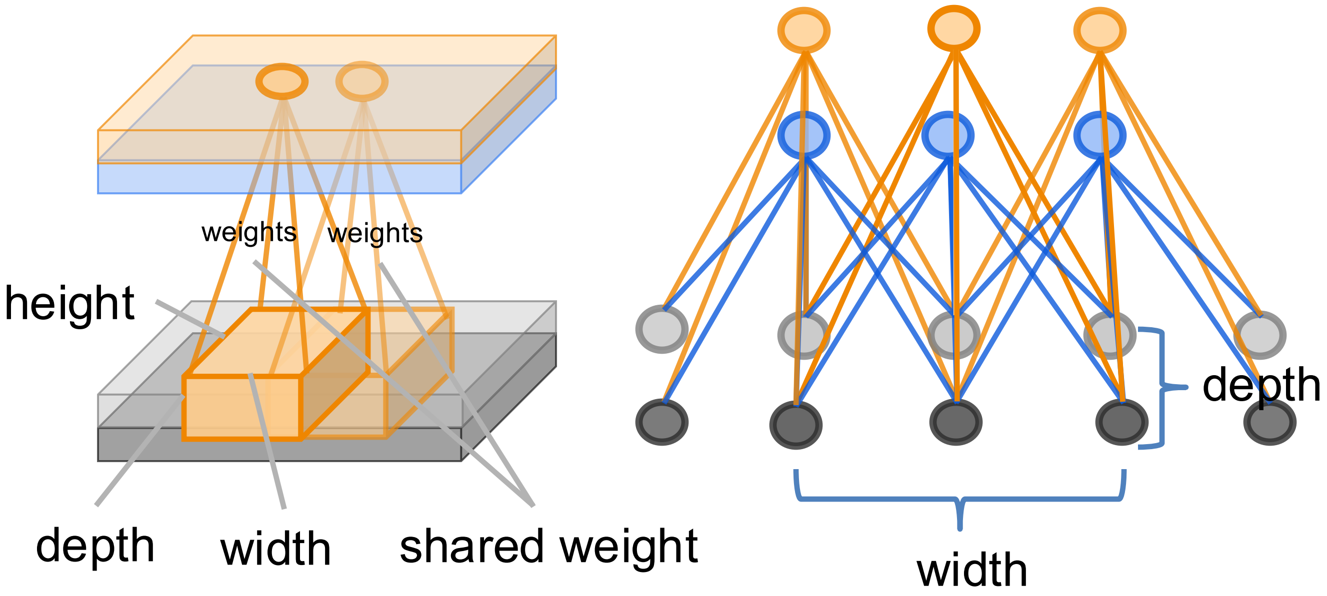
  The brunch at Margoli, especially the scones, were **delicious**.

Where’s Waldo?

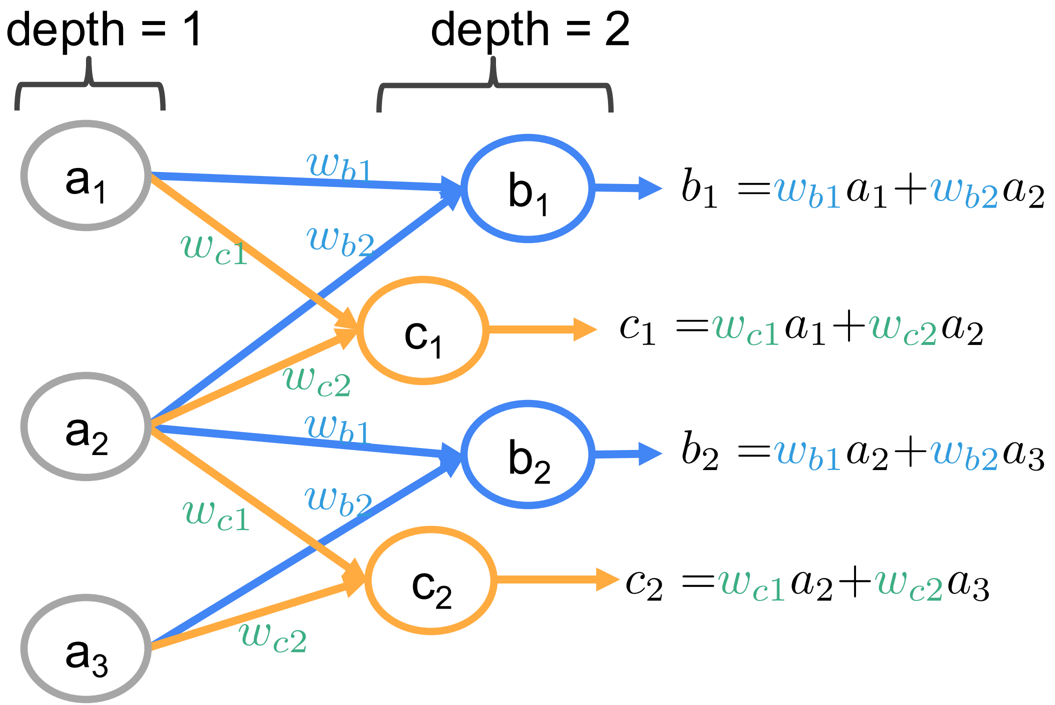
Convolutional Layers



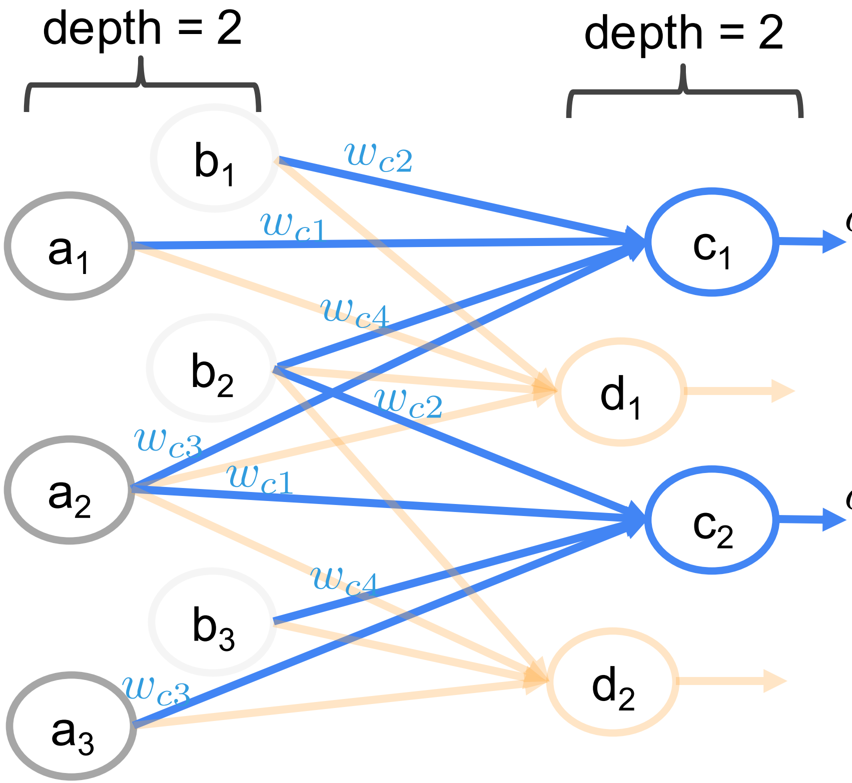
Convolutional Layers (cont.)



Convolutional Layers (cont.)



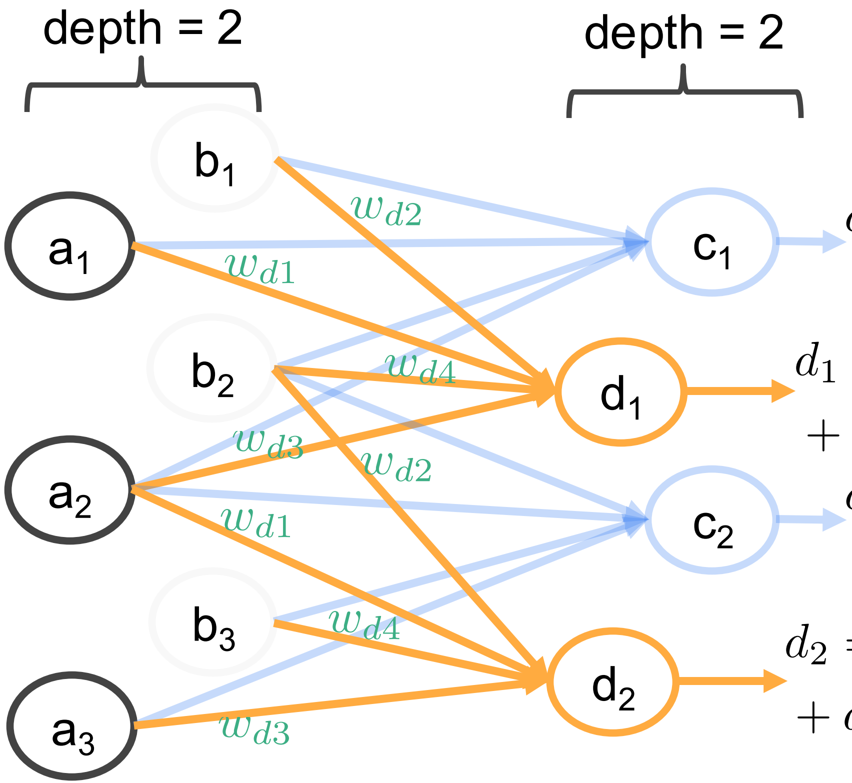
Convolutional Layers (cont.)



$$c_1 = a_1 w_{c1} + b_1 w_{c2} + a_2 w_{c3} + b_2 w_{c4}$$

$$c_2 = a_2 w_{c1} + b_2 w_{c2} + a_3 w_{c3} + b_3 w_{c4}$$

Convolutional Layers (cont.)



$$c_1 = a_1 w_{c1} + b_1 w_{c2} + a_2 w_{c3} + b_2 w_{c4}$$

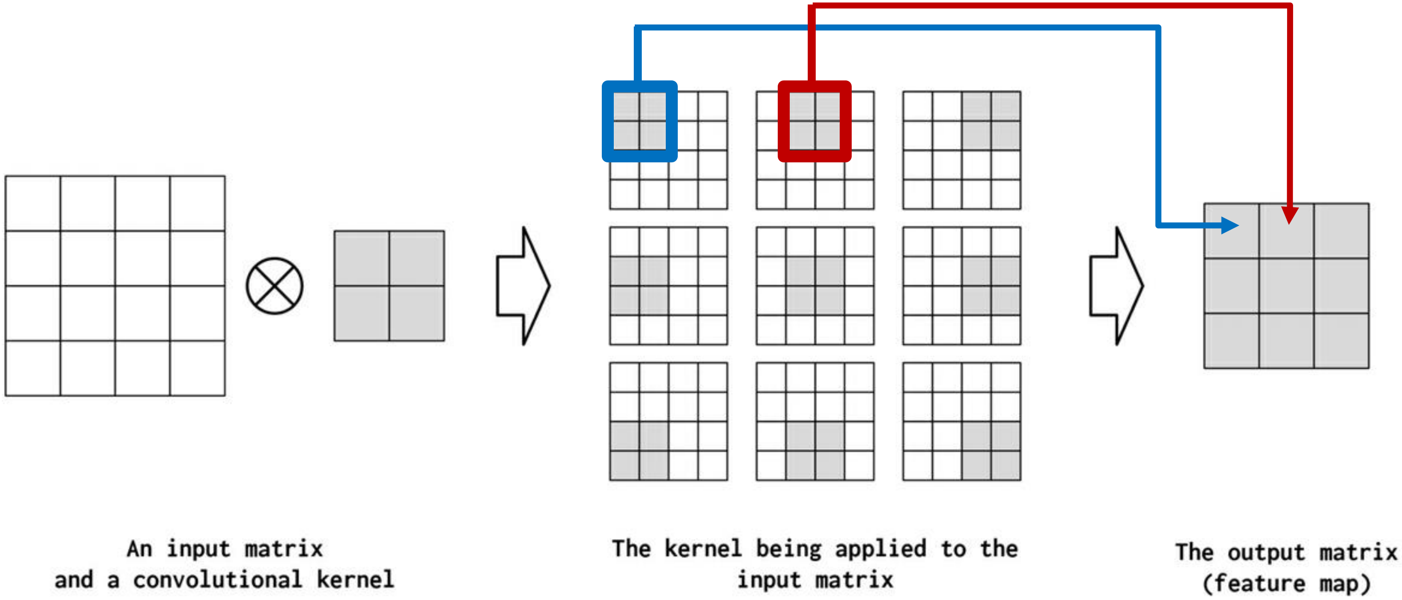
$$d_1 = a_1 w_{d1} + b_1 w_{d2} + a_2 w_{d3} + b_2 w_{d4}$$

$$c_2 = a_2 w_{c1} + b_2 w_{c2} + a_3 w_{c3} + b_3 w_{c4}$$

$$d_2 = a_2 w_{d1} + b_2 w_{d2} + a_3 w_{d3} + b_3 w_{d4}$$

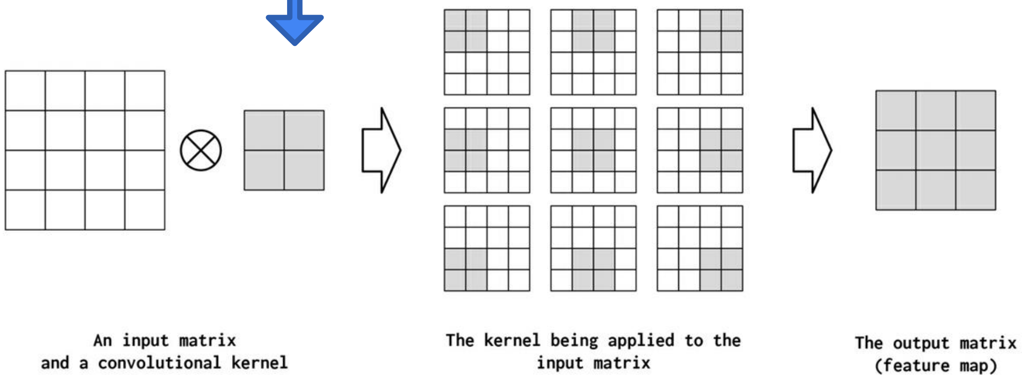
CNNs (cont.)

- CNNs: a type of NNs well-suited to detecting spatial substructure.



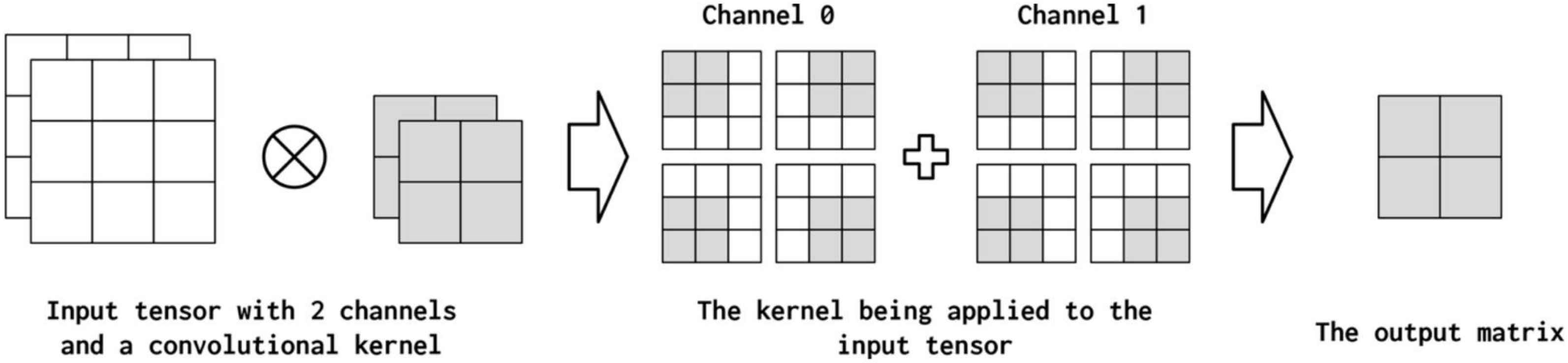
Hyper-parameters of CNNs

- **Dimension of the Convolution Operation**
- 2D convolution



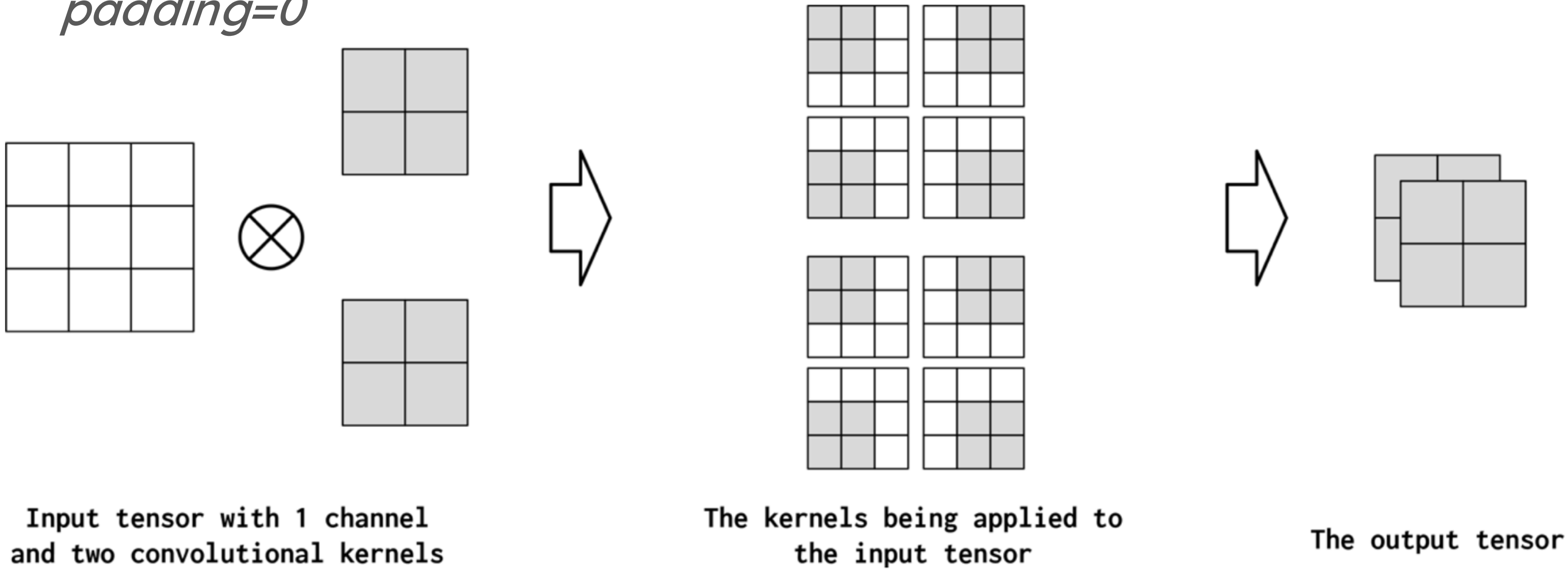
Hyper-parameters of CNNs (cont.)

- **Channels**
- *input_channels=2, output_channels=1, kernel_size=2, stride=1, padding=0*



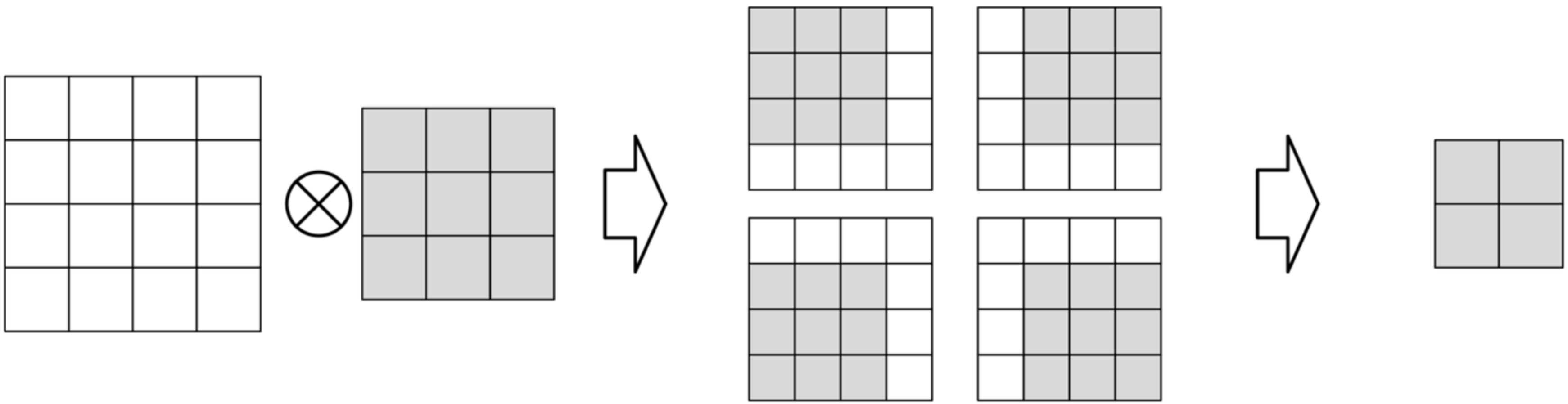
Hyper-parameters of CNNs (cont.)

- **Channels**
- *input_channels=1, output_channels=2, kernel_size=2, stride=1, padding=0*



Hyper-parameters of CNNs (cont.)

- **Kernel Size**



An input matrix
and a convolutional kernel

The kernel being applied to the
input matrix

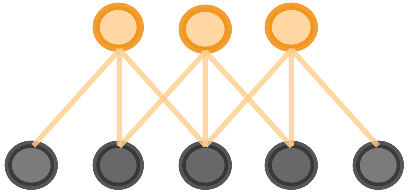
The output matrix

Kernel Size = 3, compare with the example 5 slides before!

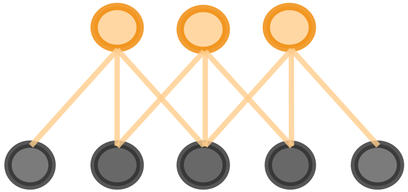
Hyper-parameters of CNNs (cont.)

- **Stride**

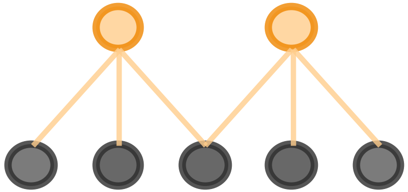
Padding



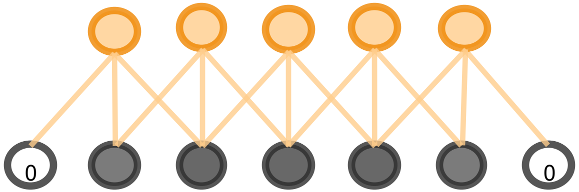
Stride = 1



Padding = 0



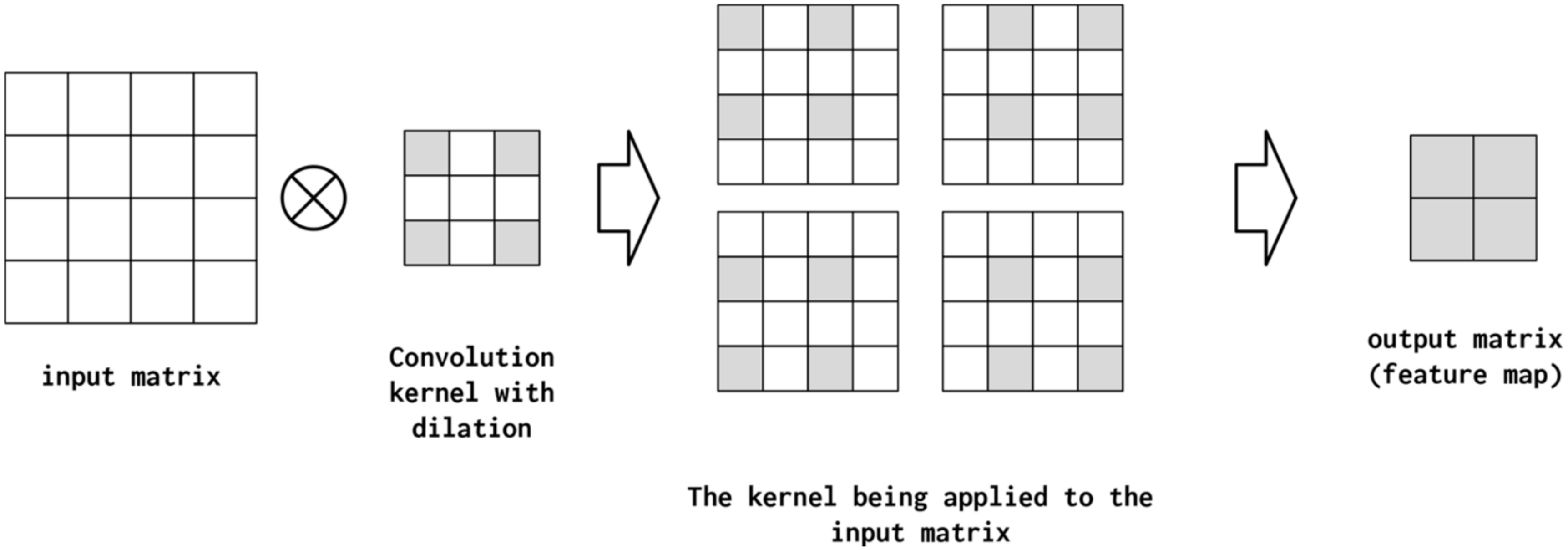
Stride = 2



Padding = 1

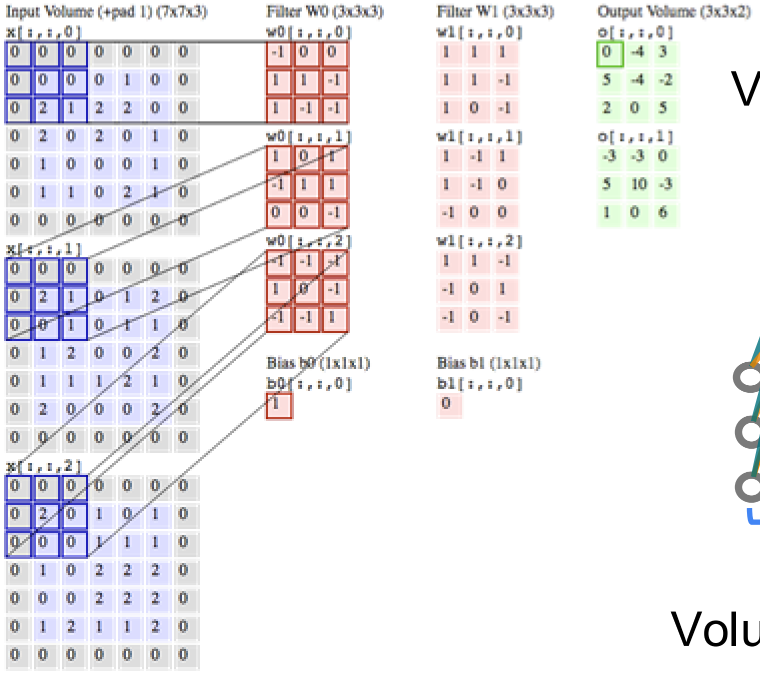
Hyper-parameters of CNNs (cont.)

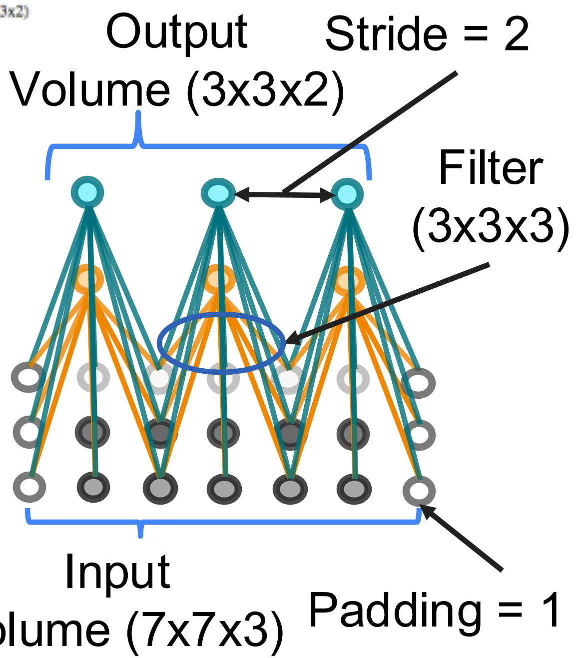
- **Dilation**



Dilation=2

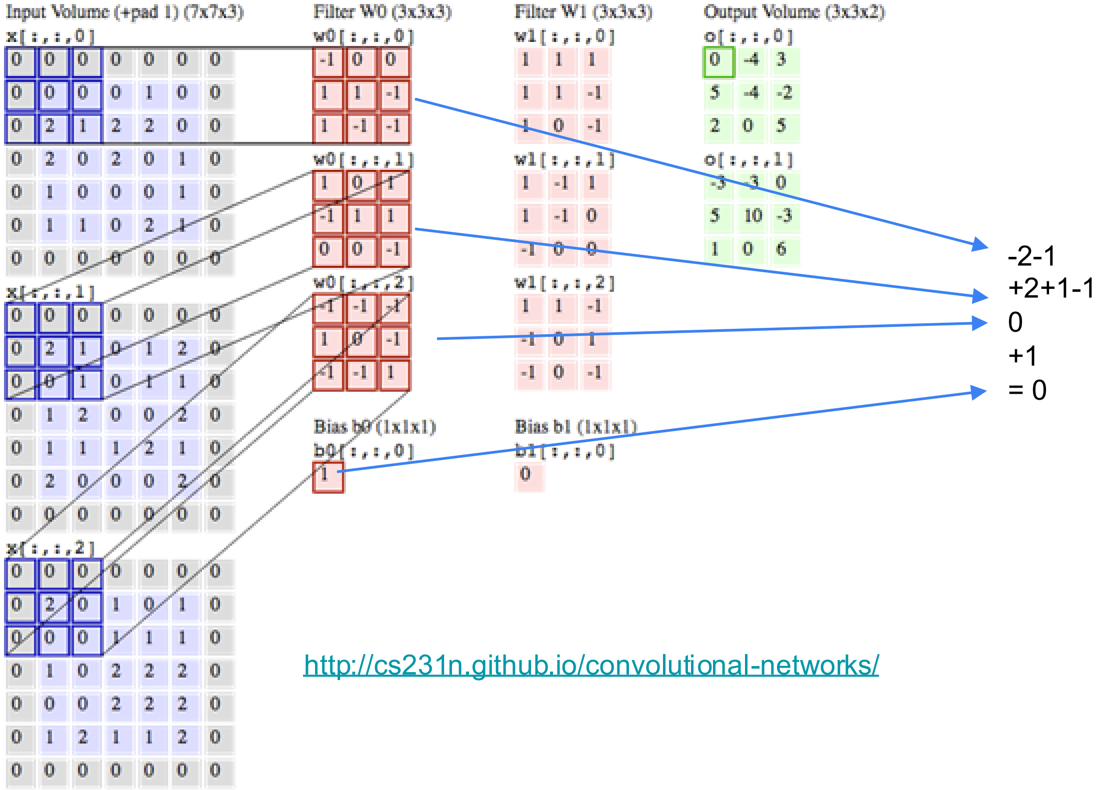
CNNs – Example Computation





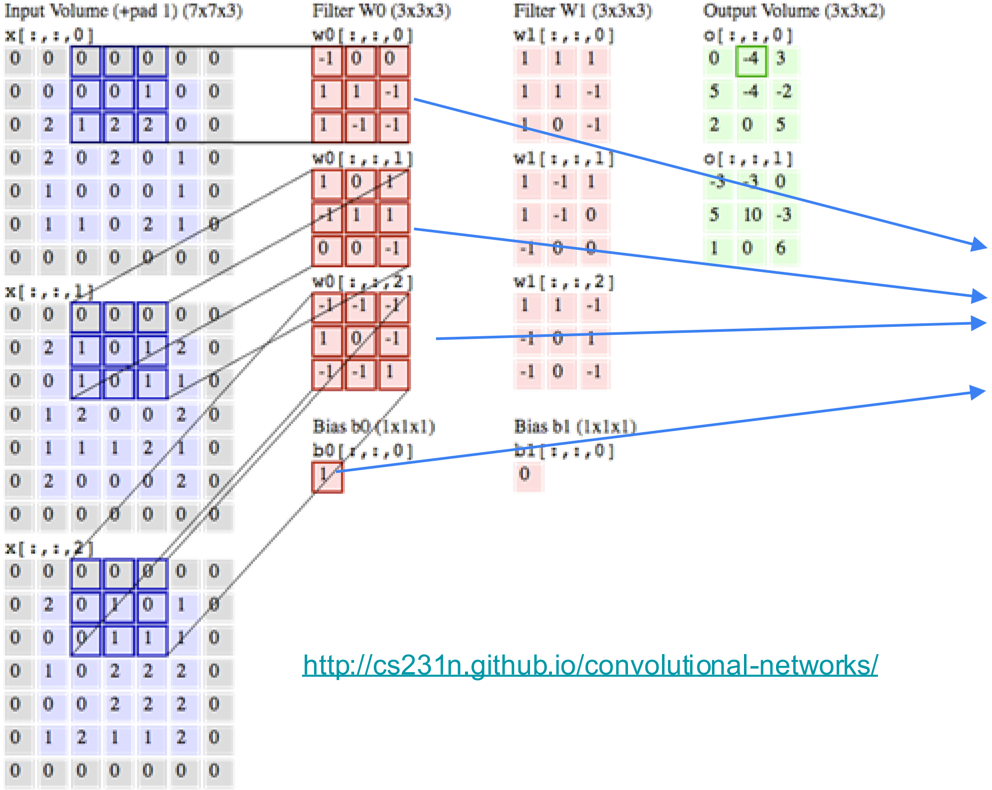
http://cs231n.github.io/convolutional-networks/

CNNs – Example Computation (cont.)



http://cs231n.github.io/convolutional-networks/

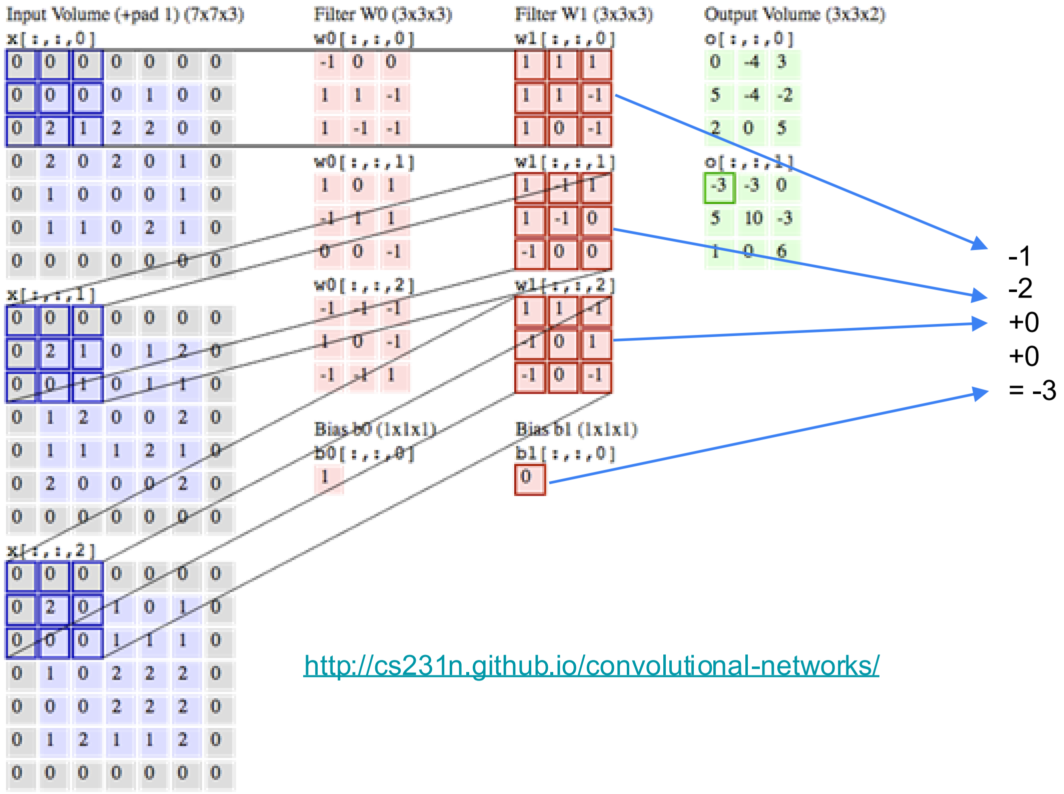
CNNs – Example Computation (cont.)



-1+1-2-2
-1+1-1
-1+1
+1
= -4

http://cs231n.github.io/convolutional-networks/

CNNs – Example Computation (cont.)



http://cs231n.github.io/convolutional-networks/

Relationship with Convolution in Math

- The convolution between two functions, say $f, g : \mathbb{R}^d \to R$ is defined as:

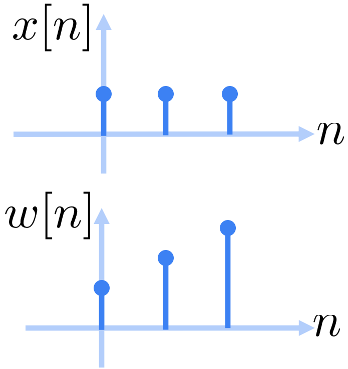
$$[f \circledast g](x) = \int_{\mathbb{R}^d} f(z) g(x - z) dz.$$

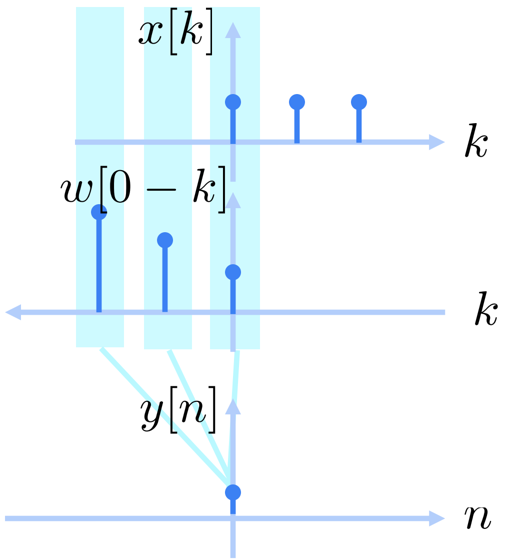
- Whenever we have discrete objects, the integral turns into a sum.

$$[f \circledast g](i) = \sum_a f(a) g(i - a).$$

Relationship with Convolution in Math (cont.)

$$y[n] = \sum_k x[k]w[n-k]$$

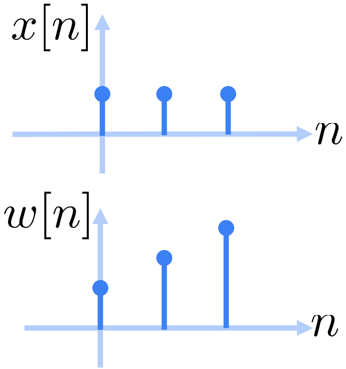


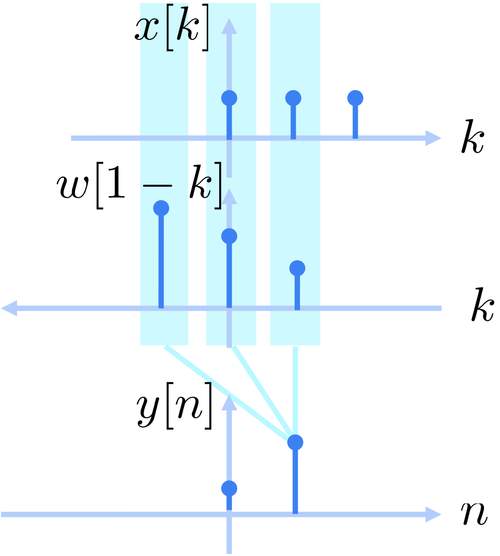


$$y[0] = x[-2]w[2] + x[-1]w[1] + x[0]w[0]$$

Relationship with Convolution in Math (cont.)

$$y[n] = \sum_k x[k]w[n-k]$$

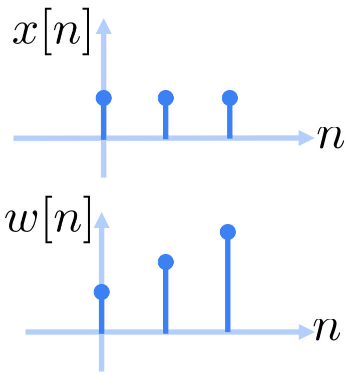


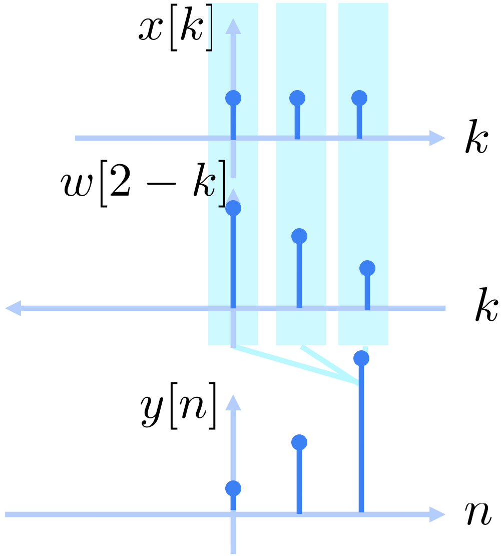


$$y[1] = x[-1]w[2] + x[0]w[1] + x[2]w[0]$$

Relationship with Convolution in Math (cont.)

$$y[n] = \sum_k x[k]w[n-k]$$

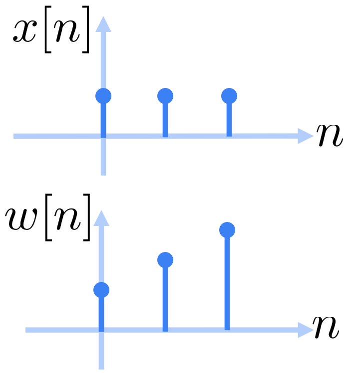


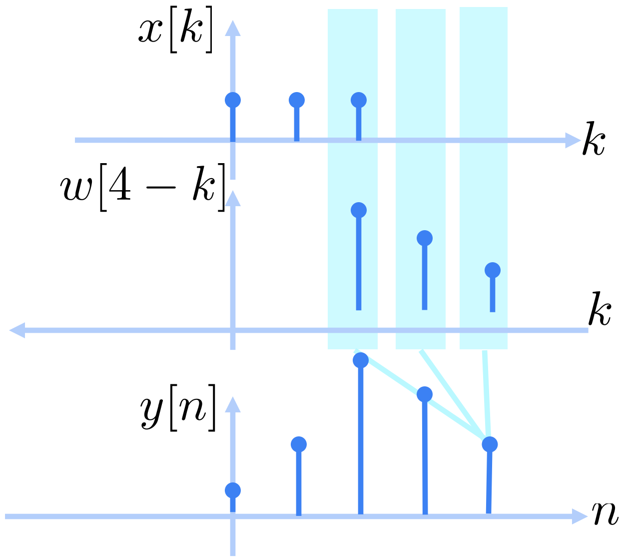


$$y[2] = x[0]w[2] + x[1]w[1] + x[2]w[0]$$

Relationship with Convolution in Math (cont.)

$$y[n] = \sum_k x[k]w[n-k]$$

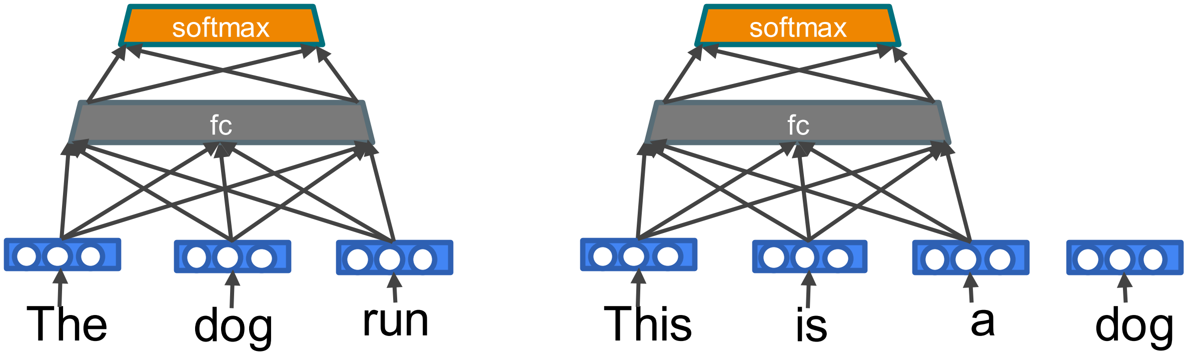




$$y[4] = x[2]w[2] + x[3]w[1] + x[4]w[0]$$

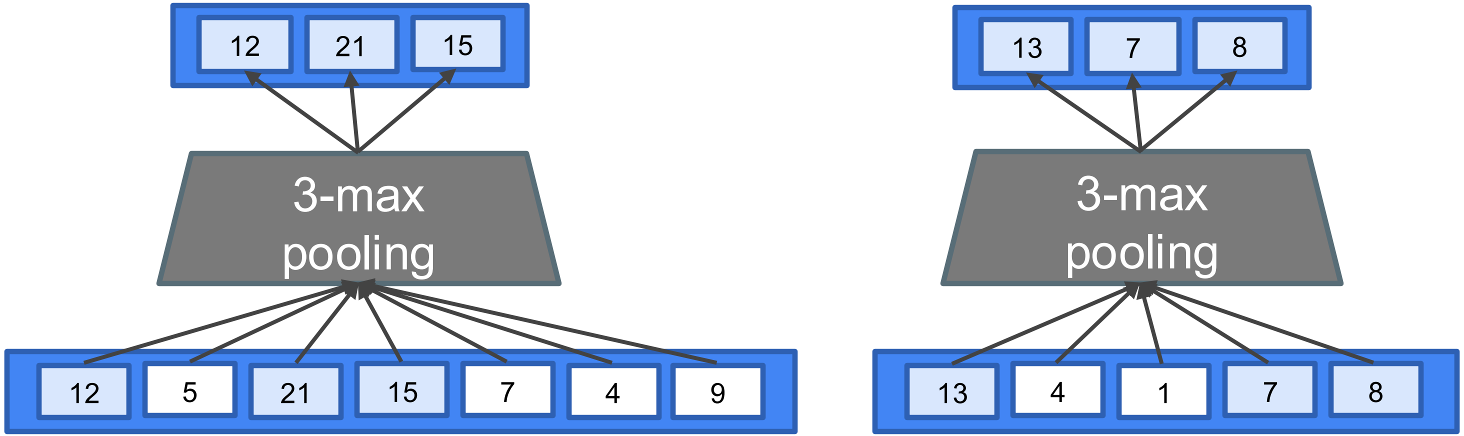
Various Input Sizes

- Fully-connected layer and softmax layer
  - need fixed-size input

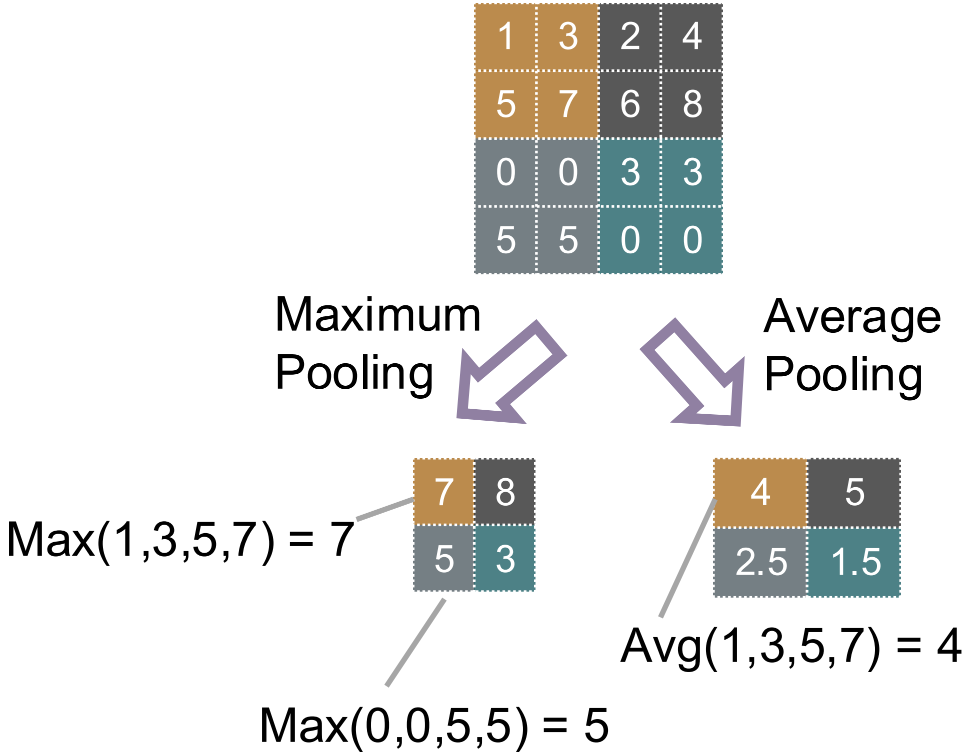


k-max Pooling

- choose the k-max values
- preserve the order of input values
- variable-size input, fixed-size output



Pooling Layer



Topics for Today

Sequence Modeling

- Convolutional Neural Networks
- Recurrent Neural Networks
- Example Applications

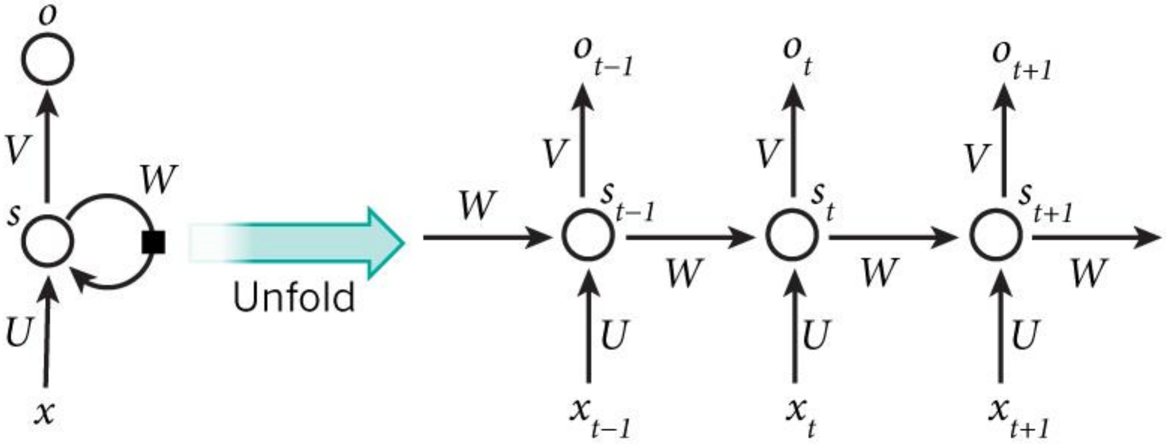
Recurrent Neural Networks (RNNs)

- Idea: condition the neural network on <u>all previous words</u> and <u>tie the weights</u> at each time step
- Assumption: temporal information matters

RNNs – Definition

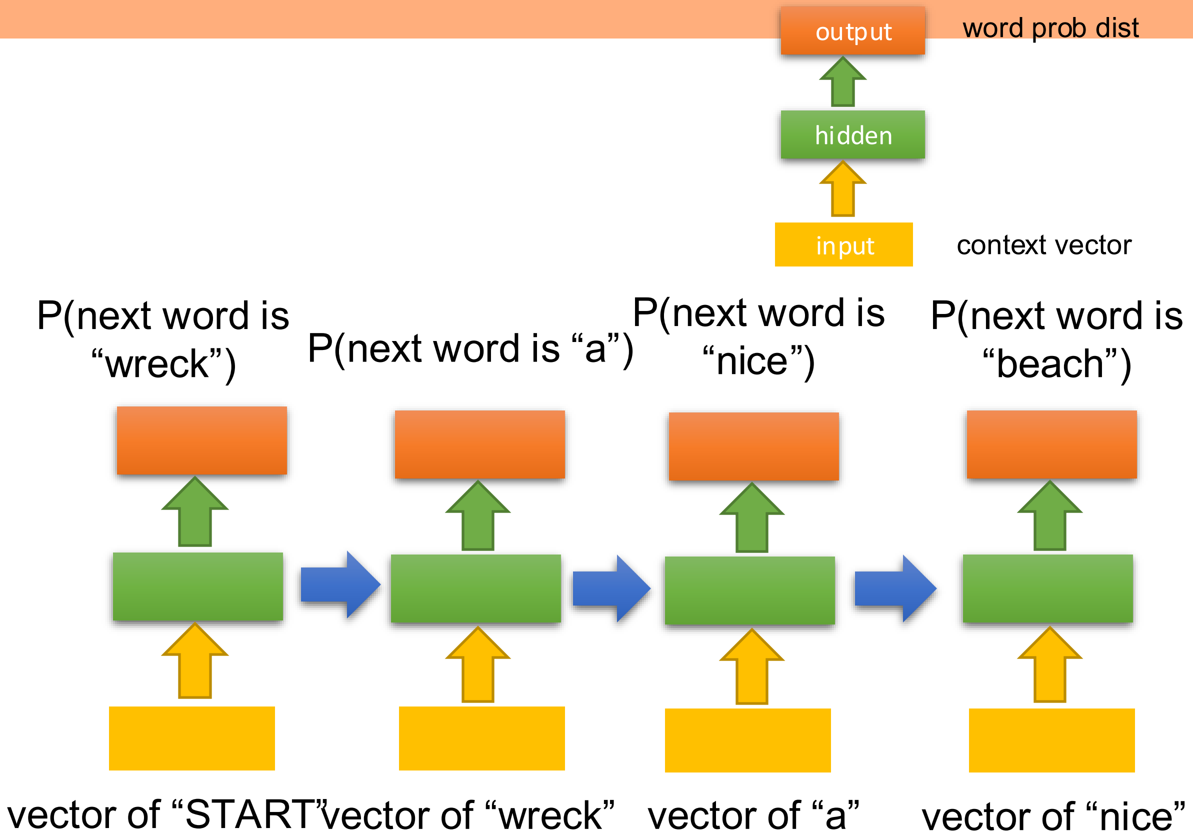
$$s_t = \sigma(W s_{t-1} + U x_t)$$
$$o_t = \text{softmax}(V s_t)$$

$\sigma(\cdot)$: tanh, ReLU



http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN Language Modeling (RNN-LM)



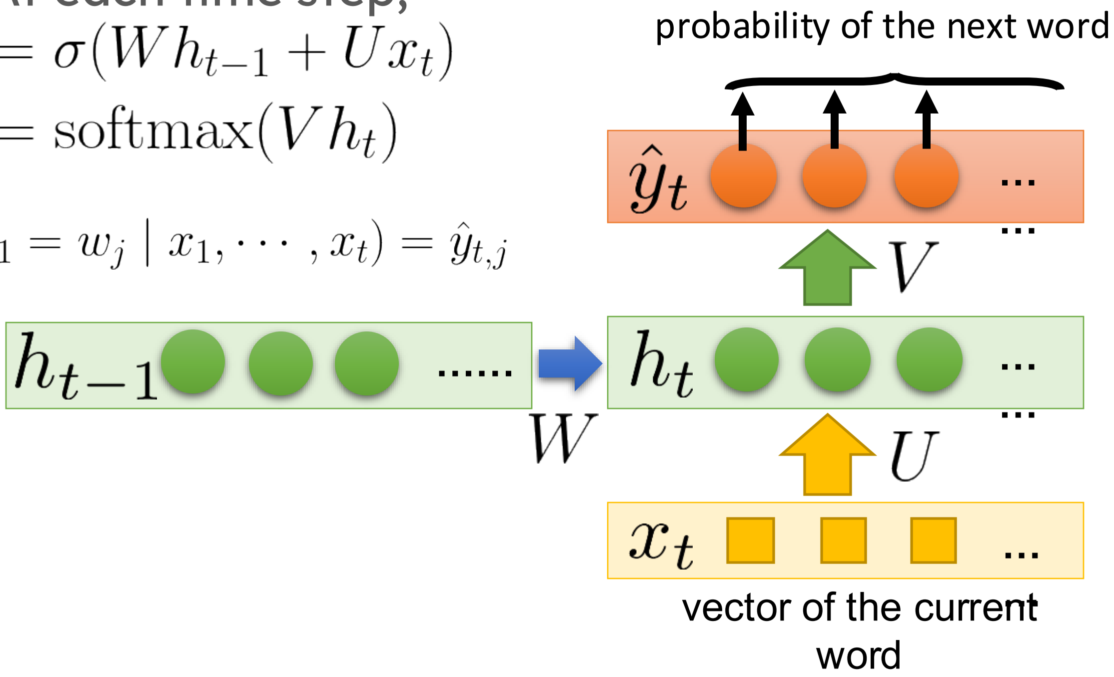
Idea: pass the information from the previous hidden layer to leverage all contexts

RNN-LM Formulation

- At each time step,

$$h_t = \sigma(Wh_{t-1} + Ux_t)$$
$$\hat{y}_t = \text{softmax}(Vh_t)$$

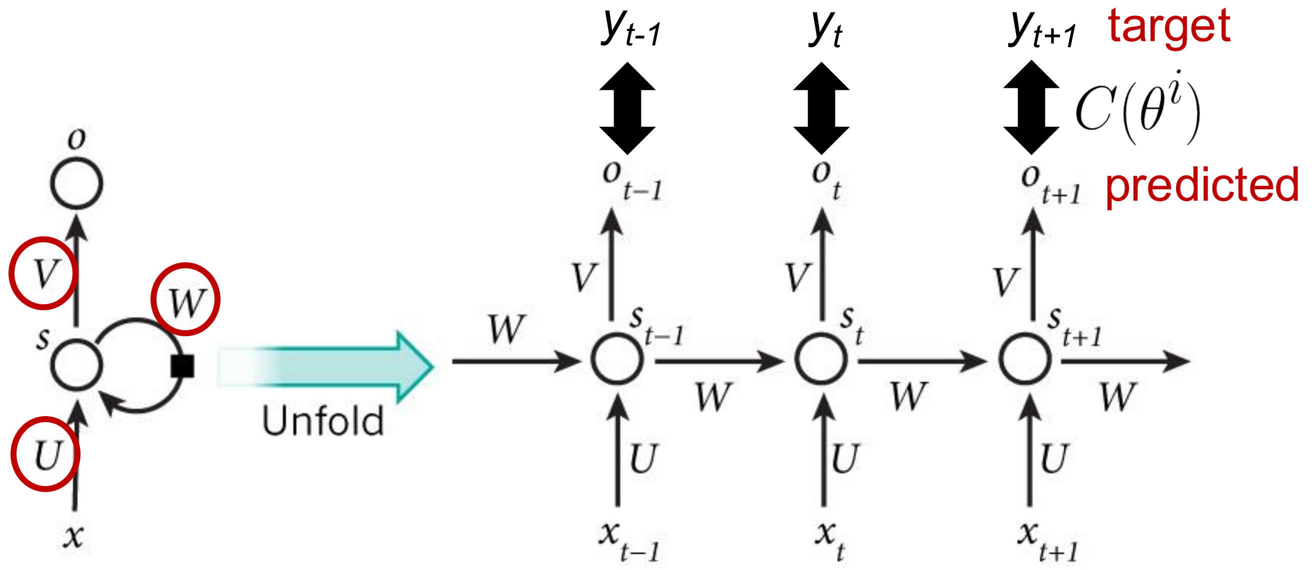
$$P(x_{t+1} = w_j \mid x_1, \cdots, x_t) = \hat{y}_{t,j}$$



RNNs – Model Training

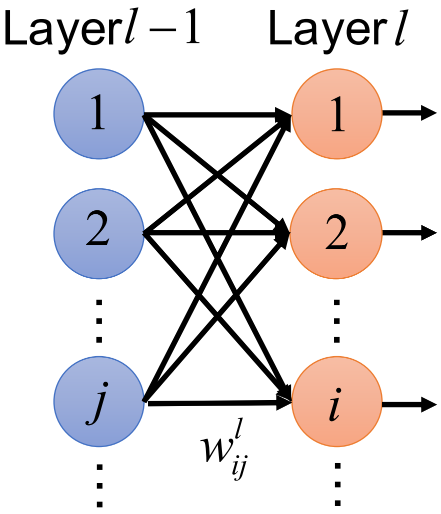
- All model parameters $\theta = \{U, V, W\}$ can be updated by

$$\theta^{i+1} \leftarrow \theta^i - \eta \nabla_\theta C(\theta^i)$$



http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Backpropagation



$$\frac{\partial C(\theta)}{\partial w_{ij}^l} = \boxed{\frac{\partial C(\theta)}{\partial z_i^l}} \boxed{\frac{\partial z_i^l}{\partial w_{ij}^l}}$$

$\boxed{\delta_i^l}$ Error signal

$$\begin{cases} a_j^{l-1} & l > 1 \\ x_j & l = 1 \end{cases}$$

Backward Pass

$$\delta^L = \sigma'(z^L) \odot \nabla C(y)$$
$$\delta^{L-1} = \sigma'(z^{L-1}) \odot (W^L)^T \delta^L$$
$$\vdots$$
$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$
$$\vdots$$

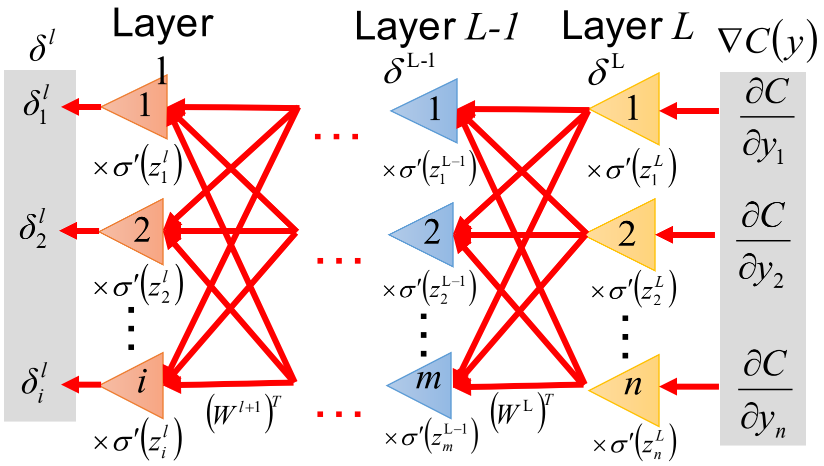
Forward Pass

$$z^1 = W^1 x + b^1$$
$$a^1 = \sigma(z^1)$$
$$\vdots$$
$$z^l = W^l a^{l-1} + b^l$$
$$a^l = \sigma(z^l)$$
$$\vdots$$

Backpropagation (cont.)

$$\frac{\partial C(\theta)}{\partial w_{ij}^l} = \boxed{\frac{\partial C(\theta)}{\partial z_i^l}} \frac{\partial z_i^l}{\partial w_{ij}^l}$$

$\boxed{\delta_i^l}$ Error signal

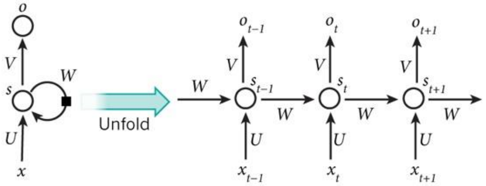


Backward Pass

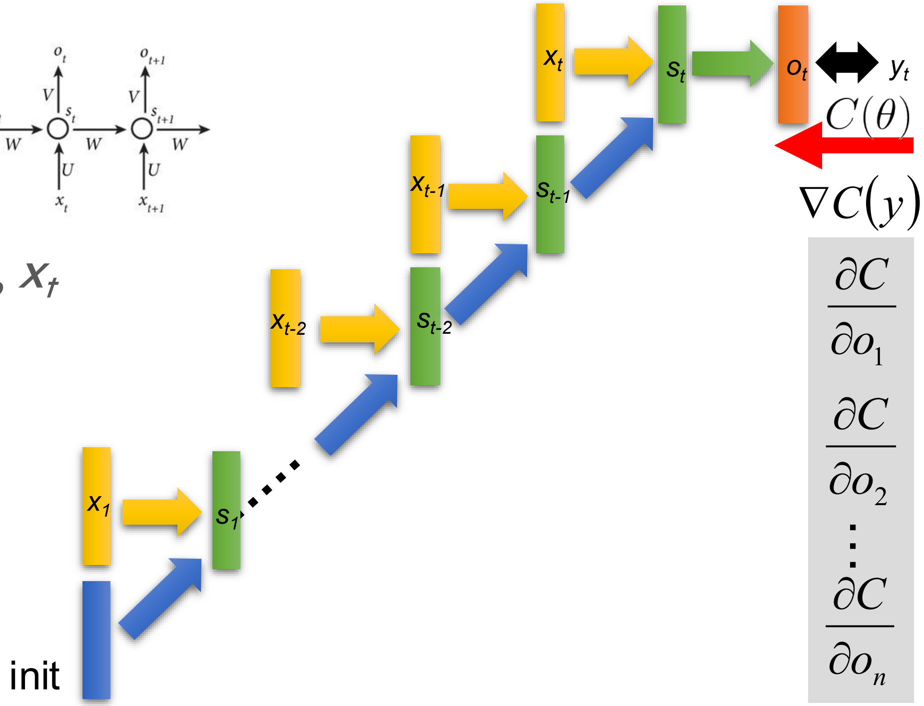
$$\delta^L = \sigma'(z^L) \odot \nabla C(y)$$
$$\delta^{L-1} = \sigma'(z^{L-1}) \odot (W^L)^T \delta^L$$
$$\vdots$$
$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$
$$\vdots$$

Backpropagation Through Time (BPTT)

- <u>Unfold</u>

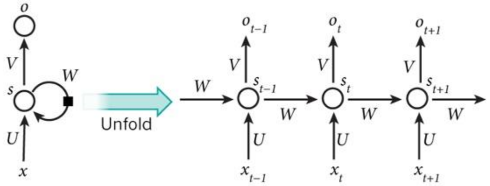


  - Input: init, x_1, x_2, …, x_t
  - Output: o_t
  - Target: y_t

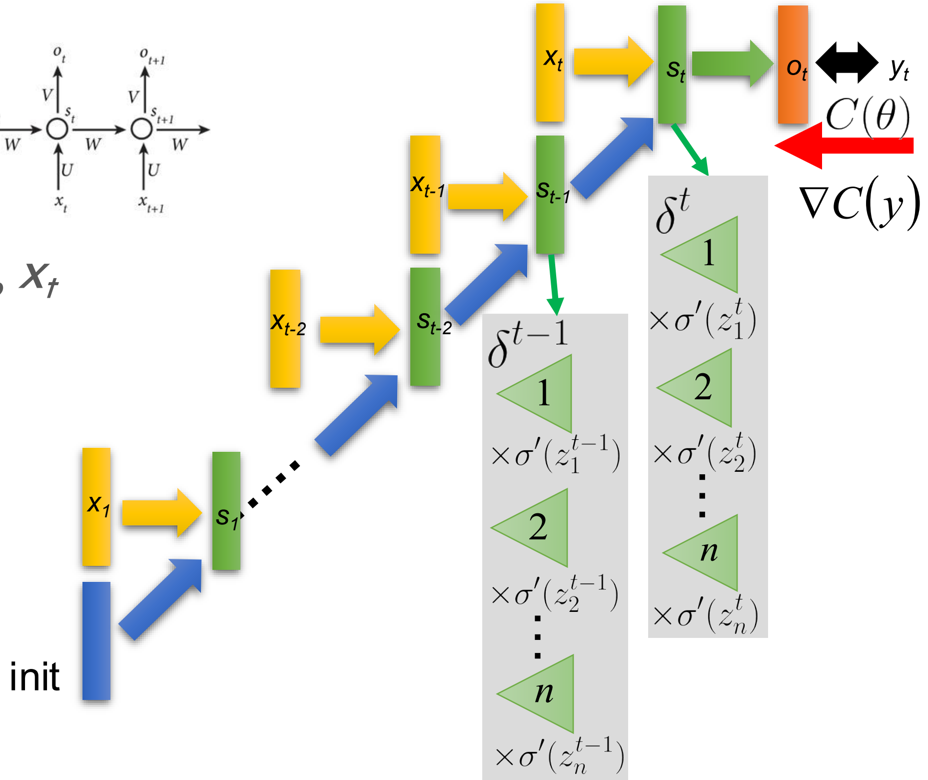


BPTT (cont.)

- Unfold

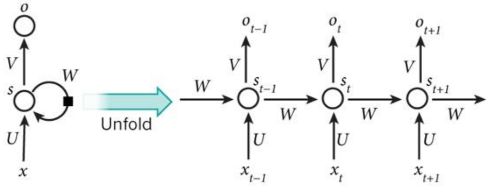


  - Input: init, x_1, x_2, ..., x_t
  - Output: o_t
  - Target: y_t

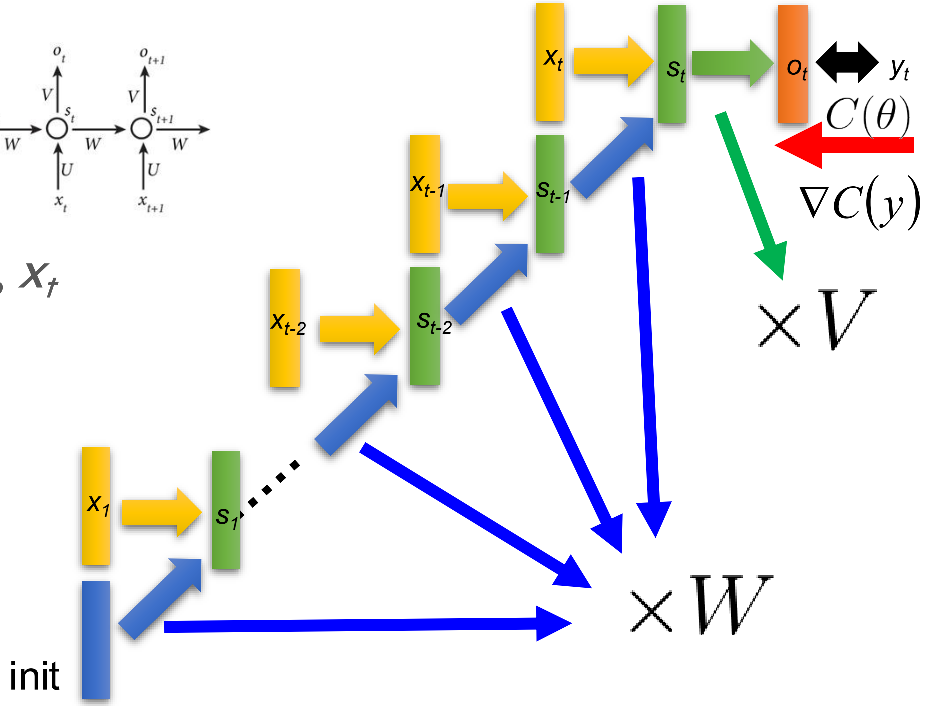


BPTT (cont.)

- <u>Unfold</u>

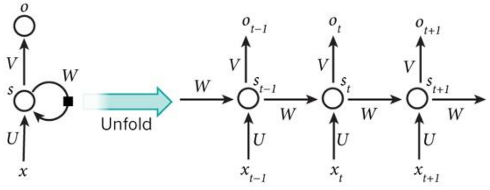


  - Input: init, x_1, x_2, ..., x_t
  - Output: o_t
  - Target: y_t

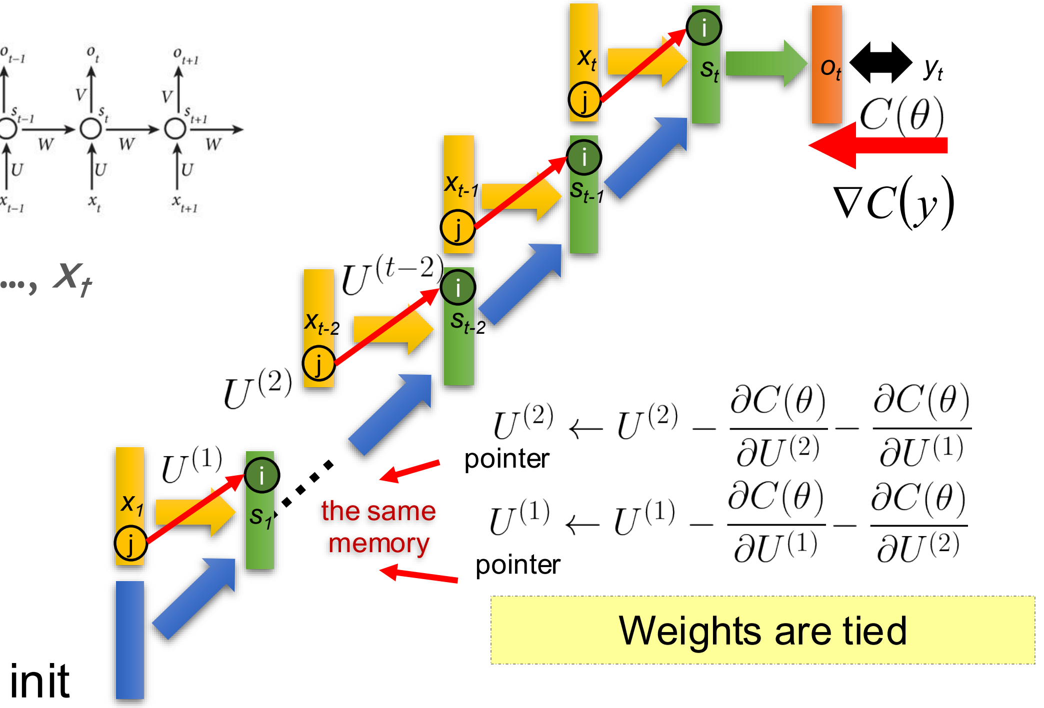


BPTT (cont.)

- Unfold

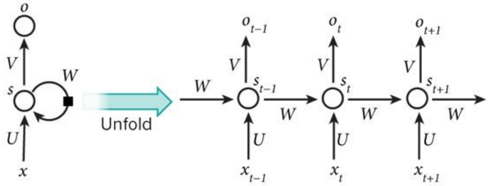


  - Input: init, x_1, x_2, ..., x_t
  - Output: o_t
  - Target: y_t

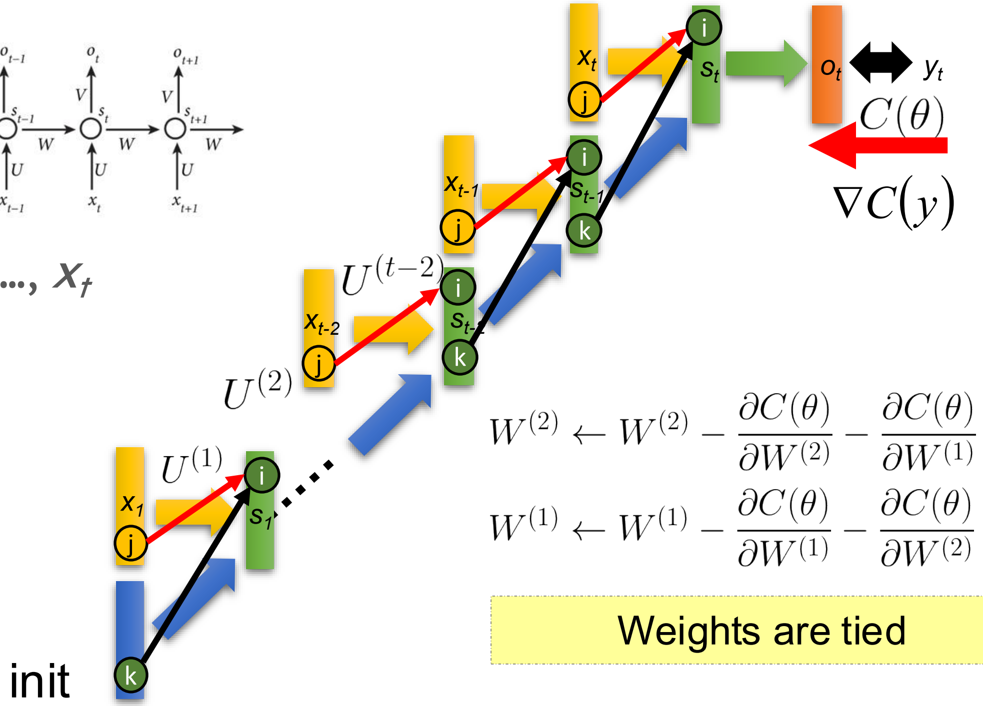


BPTT (cont.)

- <u>Unfold</u>



  - Input: init, x_1, x_2, ..., x_t
  - Output: o_t
  - Target: y_t



$$W^{(2)} \leftarrow W^{(2)} - \frac{\partial C(\theta)}{\partial W^{(2)}} - \frac{\partial C(\theta)}{\partial W^{(1)}}$$

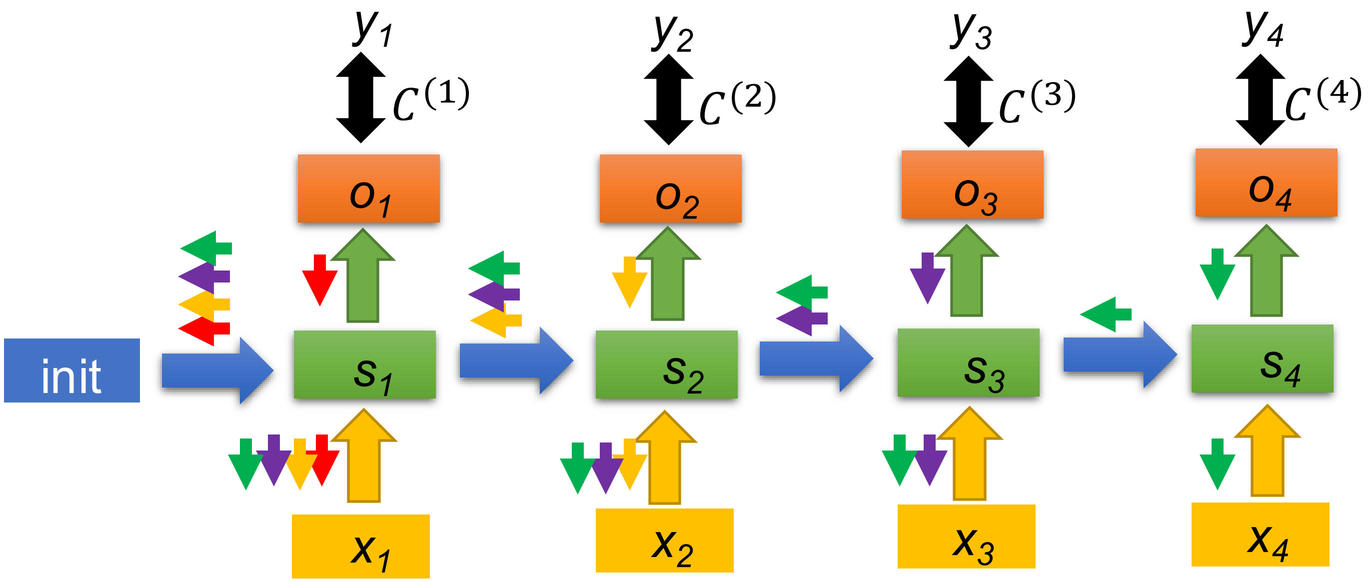
$$W^{(1)} \leftarrow W^{(1)} - \frac{\partial C(\theta)}{\partial W^{(1)}} - \frac{\partial C(\theta)}{\partial W^{(2)}}$$

Weights are tied

BPTT (cont.)

Forward Pass: Compute s_1, s_2, s_3, s_4 ……

Backward Pass: For $C^{(4)}$ For $C^{(3)}$ For $C^{(2)}$ For $C^{(1)}$



Training Issues with RNN

- The gradient is a product of Jacobian matrices, each associated with a step in the forward computation
- Multiply the same matrix at each time step during backprop

$$\delta^l = \sigma'(z^l) \odot \boxed{(W^{l+1})^T} \delta^{l+1}$$

The gradient becomes very small or very large quickly
→ **vanishing or exploding gradient**

Bengio et al., “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. of Neural Networks*, 1994. [link]
Pascanu et al., “On the difficulty of training recurrent neural networks,” in *ICML*, 2013. [link]

Clipping: A Solution to Exploding Gradients

Idea: control the gradient value to avoid exploding

Algorithm 1 Pseudo-code for norm clipping

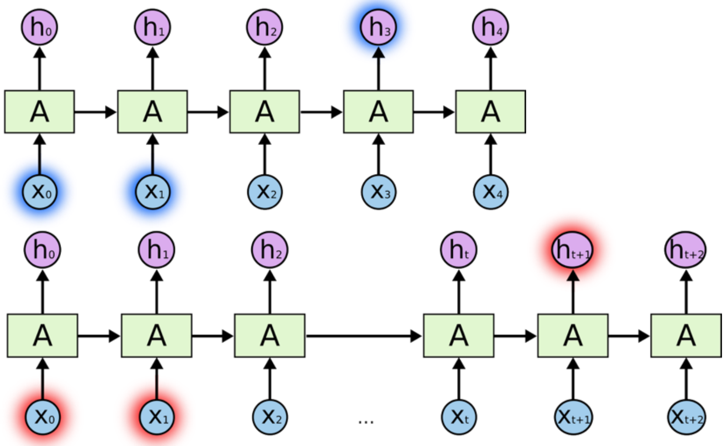
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
    $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Parameter setting: values from half to ten times the average can still yield convergence

Pascanu et al., "On the difficulty of training recurrent neural networks," in *ICML*, 2013. [link]

Gating Mechanisms: Solution to Vanishing Gradients

- RNN models temporal sequence information
  - can handle "long-term dependencies" *in theory*



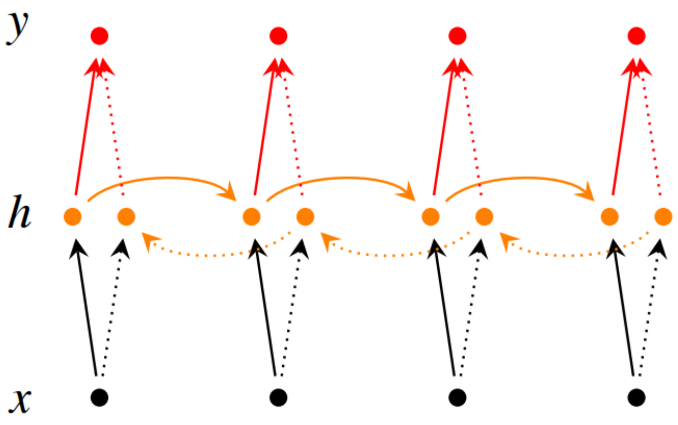
"I grew up in France…
I speak fluent *French*."

Issue: RNN cannot handle such "long-term dependencies" in practice due to vanishing gradient
→ apply the gating mechanism to directly encode the long-distance information

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

On Thursday!

RNN Extensions – Bidirectional RNNs



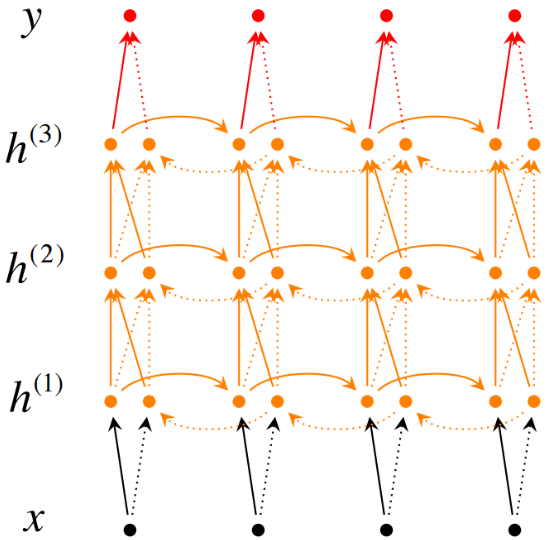
$$\overrightarrow{h}_t = f(\overrightarrow{W}x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\overrightarrow{h}_t;\overleftarrow{h}_t] + c)$$

$h = [\vec{h}; \overleftarrow{h}]$ represents (summarizes) the past and future around a single token

RNN Extensions – Deep Bidirectional RNNs



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

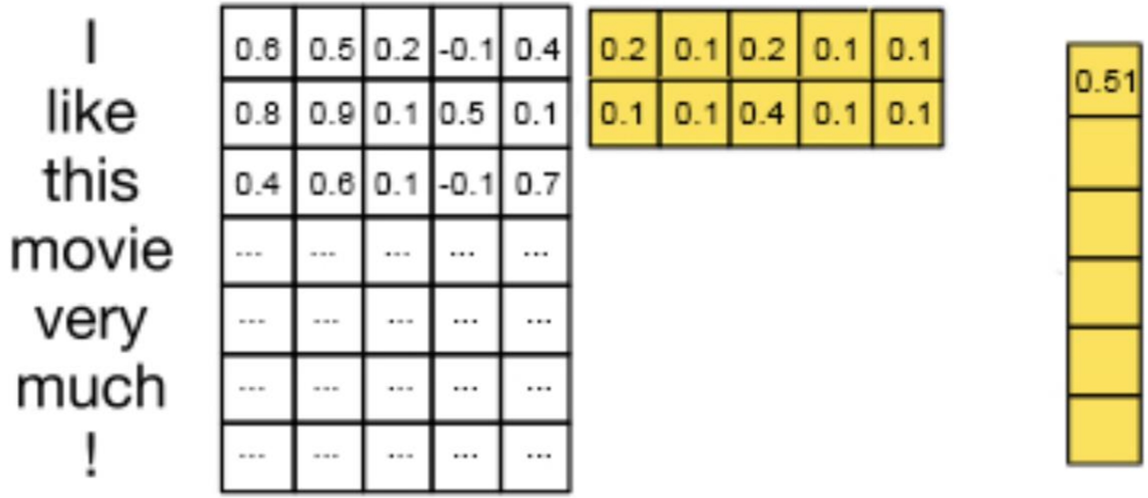
Each memory layer passes an intermediate representation to the next

Topics for Today

Sequence Modeling

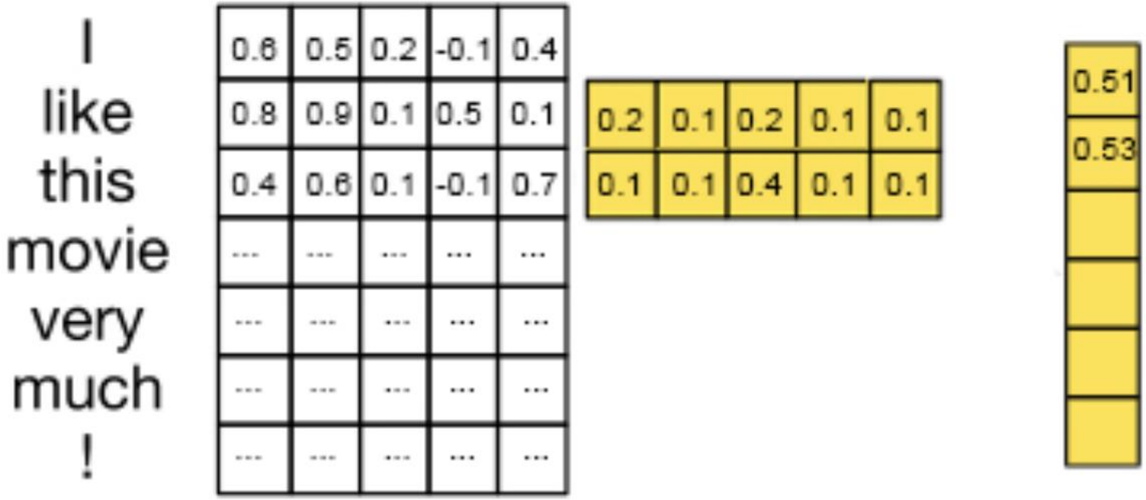
- Convolutional Neural Networks
- Recurrent Neural Networks
- Example Applications

Example: CNNs for Text Classification



- Example depiction from: http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/

Example: CNNs for Text Classification (cont.)

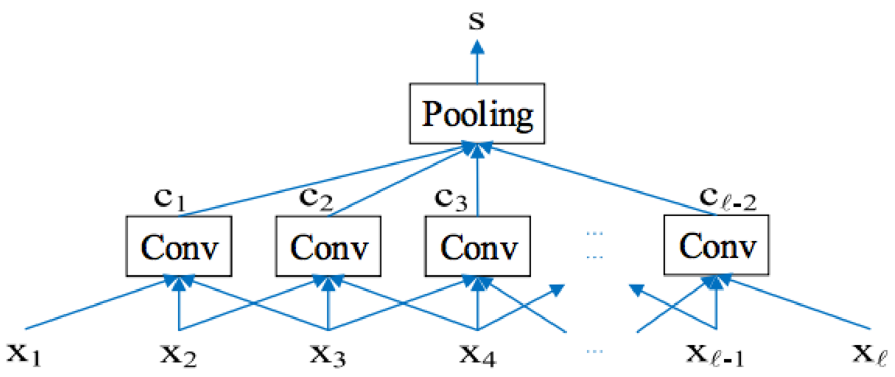


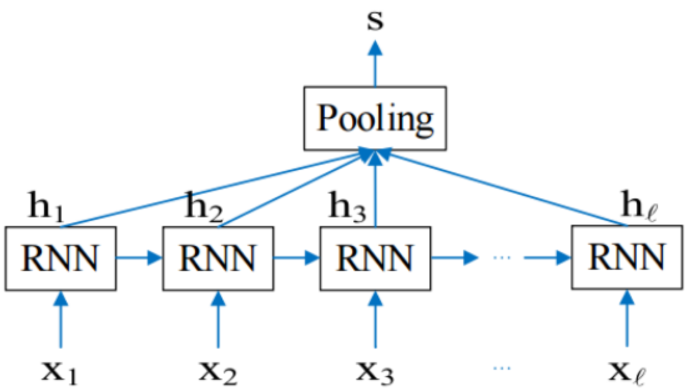
- Example depiction from: http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/

Example: CNNs and RNNs for Text Classification

- ([Lee & Dernoncourt, NAACL, 2016]())

- Dialogue Act Classification

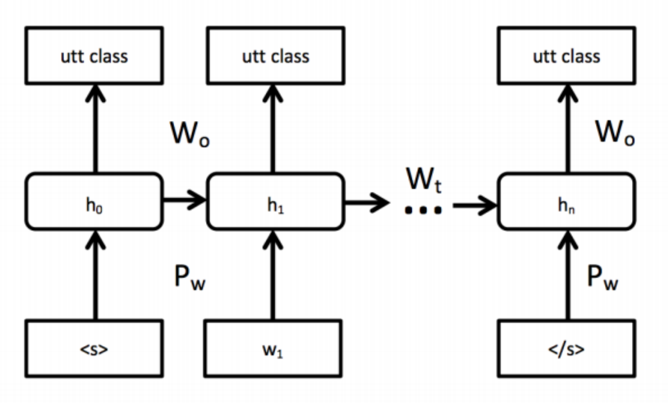


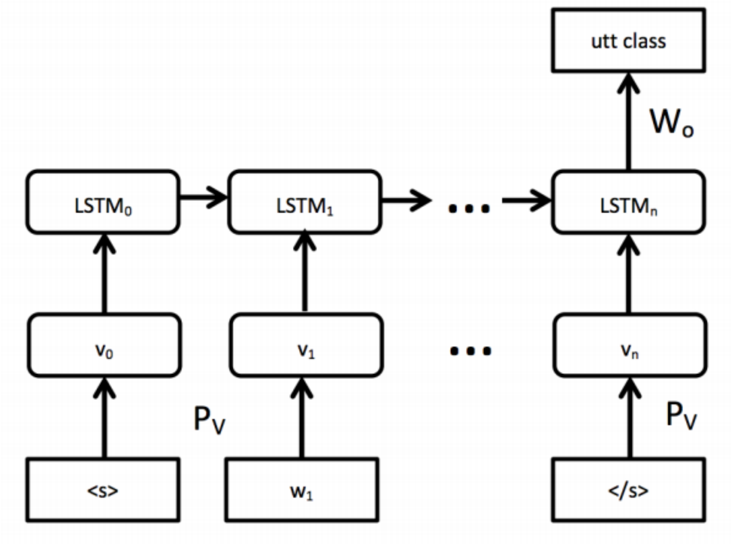


Example: RNNs for Text Classification

- (Ravuri and Stolcke, Interspeech, 2015)

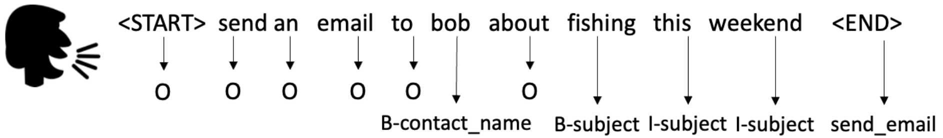
- Addressee Detection

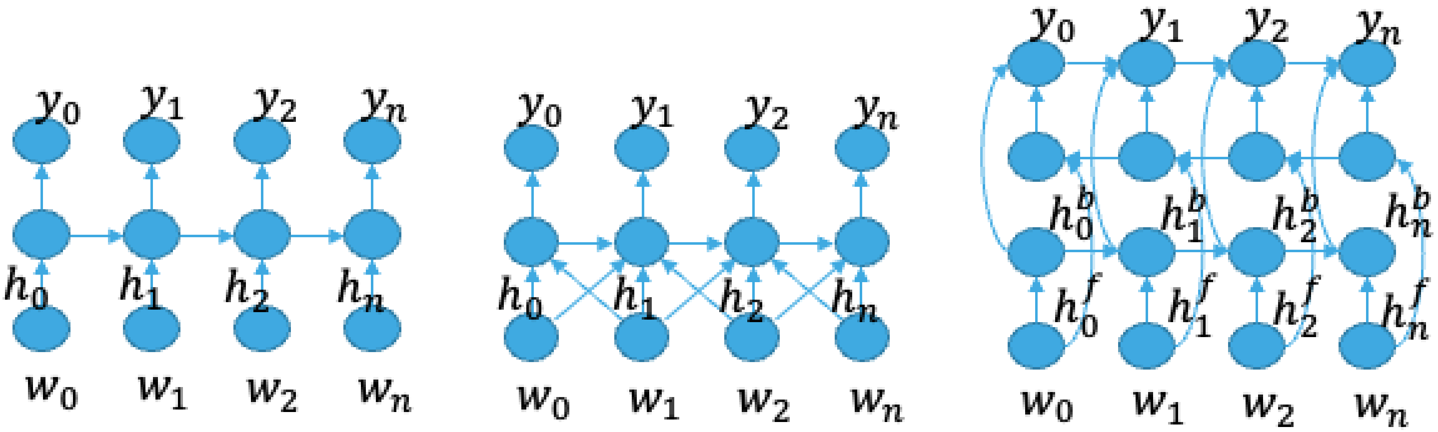




Example: RNNs for Sequence Tagging

- (Mesnil et al., IEEE TASLP, 2015)
- Slot tagging

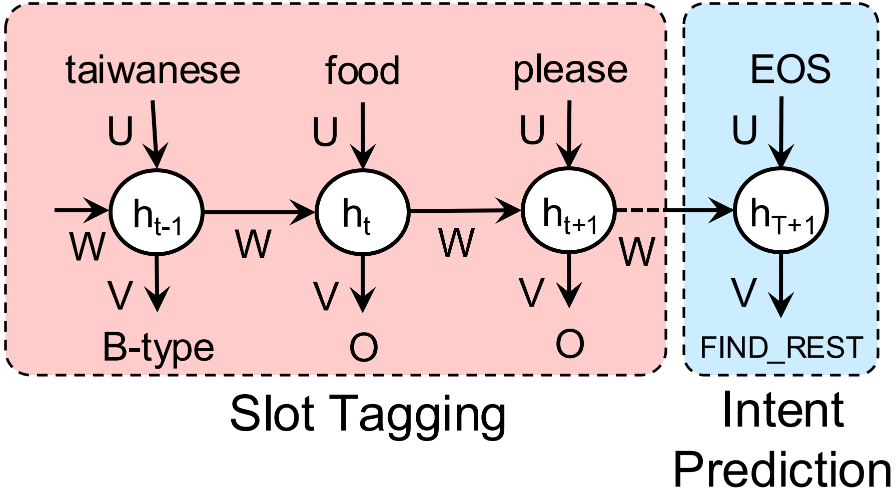




(a) RNN (b) RNN-LA (c) bi-RNN

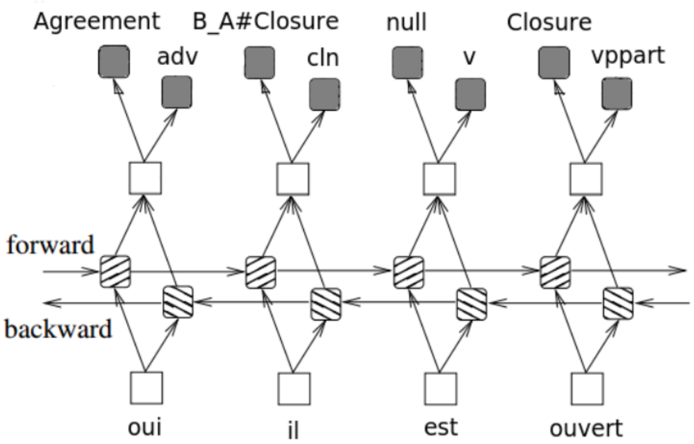
Example: RNNs for Joint Utterance Classification and Sequence Tagging

- (Hakkani-Tür et al., Interspeech, 2016)
- Slot filling (or tagging) and intent prediction in the same output sequence



Example: RNNs for Multi-Task Sequence Tagging

- (Tafforeau et al., Interspeech, 2016)
  - Goal: exploit data from domains/tasks with a lot of data to improve ones with less data
  - Lower layers are shared across domains/tasks
  - Output layer is specific to task



Topics for Thursday

Model Architectures and Contextual Embeddings

- Long-Short Term Memory (LSTM)
- Gated Recurrent Units (GRU)
- Example Sequence Classification Tasks
- Elmo and Contextual Embeddings

Midterm 1

- September 30th
- In class
- True/False and multiple-choice questions from content we discussed in class.

Preparing Final Project Proposals

- Final Project Proposal team sign up deadline: Sept 23rd
- Spreadsheet to sign up project teams: https://docs.google.com/spreadsheets/d/1EJ_5Xby0mRhHFmSRSmxlv6Gws4Qs5T8P5JUiKYKAZcA/edit?usp=sharing
- Reach out to me or TAs soon if you need help with project ideas and teaming.