

Natural Language Understanding, Generation, and Machine Translation

Lecture 27: Convolutional Neural Networks

Laura Perez-Beltrachini

(Slides by Mirella Lapata)

School of Informatics
University of Edinburgh
`mlap@inf.ed.ac.uk`

Outline

- 1 Introduction
- 2 Convolutional Neural Networks
 - Application to NLP
 - Filters, hyperparameters
 - Sentence Classification
- 3 Discussion

What is Convolution?

Convolution is an important operation in signal and image processing; it operates on two signals (1D) or two images (2D). Think of one as the **input** signal and the other, the kernel as a **filter** on the input producing an **output**.

Definition

$$(f * g)(i) = \sum_{j=1}^m g(j) \cdot f(i - j + m/2)$$

- f is the input vector and g is the kernel
- f has length n and g has length m
- **We are sliding the kernel over input vector!**

1D Convolution Example

Suppose we have 1D input vector denoted by f :

$$f = \begin{bmatrix} 10 & 50 & 60 & 10 & 20 & 40 & 30 \end{bmatrix}$$

and our kernel is $g = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}$

Let's call the output vector h . What is the value of $h(3)$?

10	50	60	10	20	40	30
	1/3	1/3	1/3			

$$\frac{1}{3}50 + \frac{1}{3}60 + \frac{1}{3}10 = \frac{50}{3} + \frac{60}{3} + \frac{10}{3} = 40 = h(3)$$

1D Convolution Example

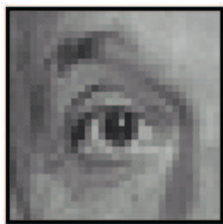
What is the kernel doing?

- The kernel is computing a windowed average of the input vector
- The kernel replaces each entry with the average of that entry and its left and right neighbor.
- We can compute the other values of h as well.

$$h = \begin{array}{|c|c|c|c|c|c|c|} \hline 20 & 40 & 40 & 30 & 20 & 30 & 23.333 \\ \hline \end{array}$$

2D Convolution Example

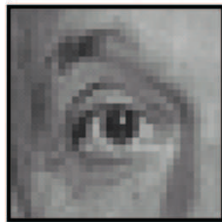
- Our images and kernels are now 2D functions (aka matrices).
- We slide the kernel over each pixel of the image, multiply the corresponding entries of the input and kernel, and add them up.



Original



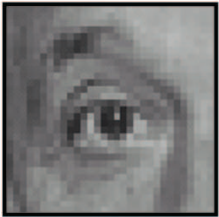
0	0	0
0	1	0
0	0	0




Identical image

2D Convolution Example

- Our images and kernels are now 2D functions (aka matrices).
- We slide the kernel over each pixel of the image, multiply the corresponding entries of the input and kernel, and add them up.


$$*$$
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

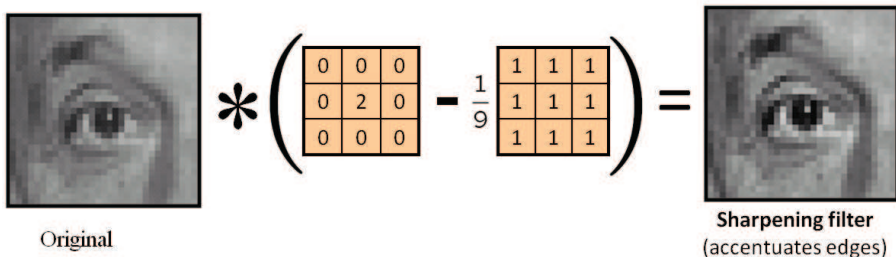
$$=$$


Original

Blur (with a mean filter)

2D Convolution Example

- Our images and kernels are now 2D functions (aka matrices).
- We slide the kernel over each pixel of the image, multiply the corresponding entries of the input and kernel, and add them up.



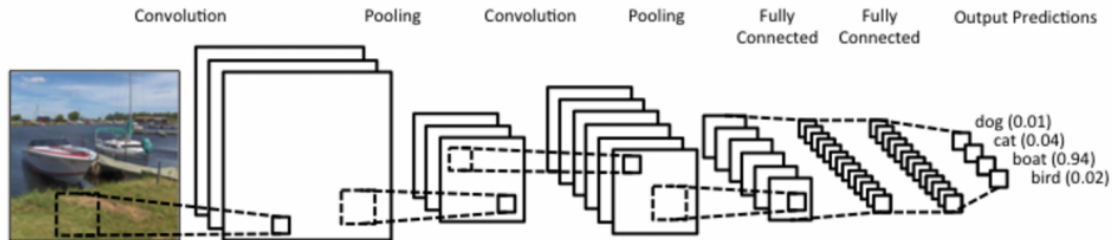
The diagram shows the convolution of an original image with a sharpening filter. The original image is a grayscale photo of an eye. It is convolved with a kernel represented as the difference between two 3x3 matrices. The first matrix has a center value of 2, and the second matrix has all values of 1. The result is a sharpened image where edges are more pronounced.

$$\text{Original} * \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) = \text{Sharpening filter (accentuates edges)}$$

What are Convolutional Neural Networks?

- **Feedforward NNs** each input neuron is connected to each output neuron in next layer.
- CNNs use convolutions over the input layer to compute the output.
- This results in **local connections**: each region of the input is connected to a neuron in the output
- Each layer applies **different filters** (hundreds or thousands) and combines their results
- A CNN automatically **learns the values of its filters** based on the task you want to perform.

What are Convolutional Neural Networks?



So What this have to do with NLU?

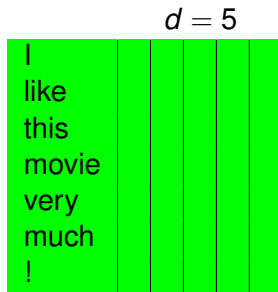
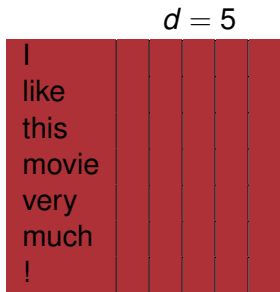
- We begin with a tokenized sentence and convert it into a **matrix**.
- **Rows** are d -dimensional word vectors for each token
- Let s denote sentence length, then matrix is $s \times d^2$
- Sentence looks like an image now, we can apply convolutions.

$d = 5$

I					
like					
this					
movie					
very					
much					
!					

What are Filters for Sentences?

- In vision filters slide over local patches of an image.
- In NLP filters slide over **full rows** of the matrix (words).
- The **width** of the filter is same as d width of input matrix.
- The **height** h or **region size** of the filter is number of adjacent rows.
- Sliding windows over 2-5 words at a time is typical.



What are Filters for Sentences?

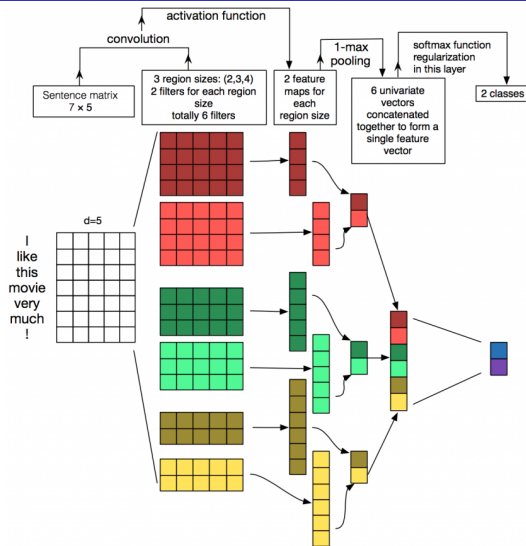
- A filter is **parameterized** by weight vector $\mathbf{w} \in \mathbb{R}^{h \times d}$
- Let $\mathbf{A} \in \mathbb{R}^{s \times d}$ denote a sentence matrix.
- Let $A[i : j]$ denote sub-matrix of \mathbf{A} from row i to row j .
- Obtain **output sequence** of convolution operator by repeatedly applying filter on submatrices of \mathbf{A} .
- Include a bias term b and an **activation function** f to each o_i inducing **feature map** $\mathbf{c} \in \mathbb{R}^{s-h+1}$ for the filter

$$o_i = \mathbf{w} \cdot \mathbf{A}[i : i + h - 1]$$

$i = 1 \dots s - h + 1$ and \cdot is dot product between sub-matrix and filter (sum over element-wise multiplications)

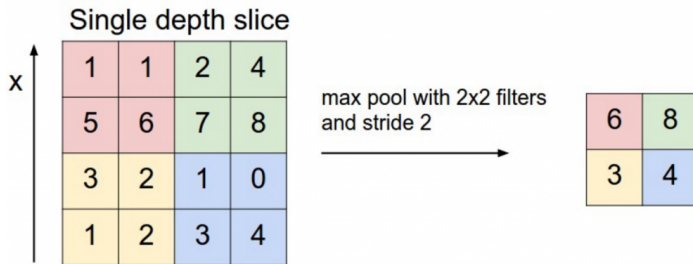
$$c_i = f(o_i + b)$$

Illustration of CNN



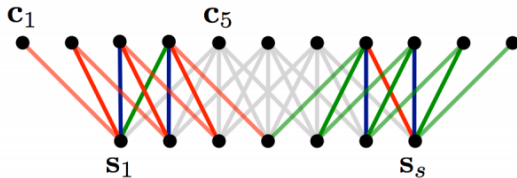
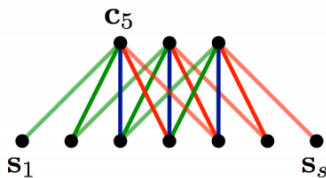
Pooling

- The **dimensionality** of feature map **c** will vary with sentence length.
- A **pooling** function is applied to each map to reduce dimensionality and number of parameters
- Most common pooling operator is **1-max pooling** function
- Captures feature with the highest value for each feature map



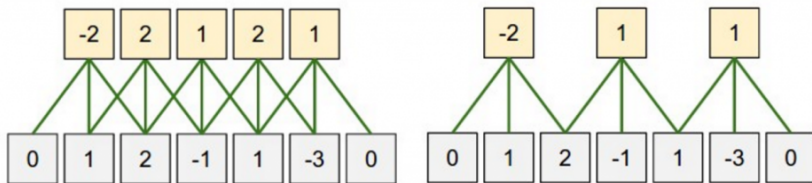
CNN Hyperparameters

- Applying a 3×3 filter at the center of the matrix works fine.
- But how would you apply the filter to the first element of a matrix that doesn't have any neighboring elements to the top and left?
- **zero-padding**: all elements that fall outside of the matrix are zero.
- **wide** convolution vs **narrow** convolution.



CNN Hyperparameters

- **stride size**: how much do you want to shift your filter at each step
- If stride size is 1, consecutive applications of the filter overlap
- A larger stride size leads to fewer applications of the filter and a smaller output size

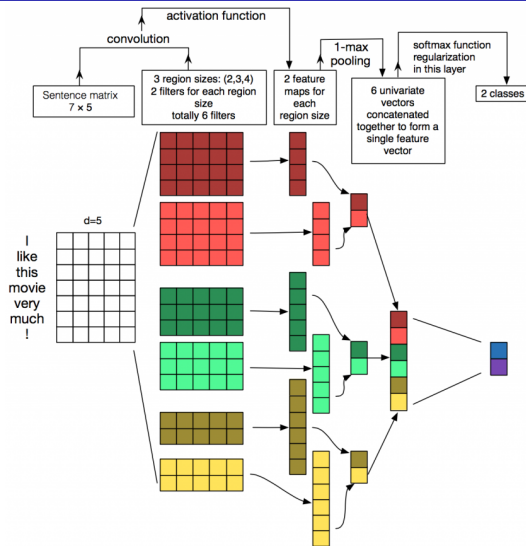


Source: <http://cs231n.github.io/convolutional-networks/>

Final Model

- Each sentence represented by feature vector and fed through a softmax function to generate final classification.
- You may apply **dropout** or **l_2 norm constraint**.
- Training objective: minimize cross-entropy loss.
- Parameters: weight vector(s) of the filter(s), bias term in activation function, and weight vector in softmax.
- Word vectors can be fixed or inferred.

Illustration of CNN



Evaluation (Kim, EMNLP-2014)

Data	c	l	N	$ V $	$ V_{pre} $	<i>Test</i>
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

MR: movie reviews, one sentence per review; SST: Stanford sentiment treebank; Subj: sentence subjective or objective; TREC: question classification; CR: customer reviews, positive/negative sentences; MPQA: opinion polarity detection for sentences.

Model Comparisons (Kim, EMNLP-2014)

- **CNN-rand**: baseline model where all words are randomly initialized and then modified during training.
- **CNN-static**: model with pre-trained `word2vec` vectors; unknown words are initialized randomly.
- **CNN-non-static**: pretrained vectors are fine-tuned for each task.
- **CNN-multichannel**: two sets of word vectors (one static, one non-static), each is a channel; each filter is applied to both channels (form of fine-tuning).
- **Hyperparameters**: filter windows (h) of 3, 4, 5 with 100 feature maps each.

Results (Kim, EMNLP-2014)

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	89.6
CNN-non-static	81.5	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	—	—	—	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	—	—	—	—
RNTN (Socher et al., 2013)	—	45.7	85.4	—	—	—	—
DCNN (Kalchbrenner et al., 2014)	—	48.5	86.8	—	93.0	—	—
Paragraph-Vec (Le and Mikolov, 2014)	—	48.7	87.8	—	—	—	—
CCAE (Hermann and Blunsom, 2013)	77.8	—	—	—	—	—	87.2
Sent-Parser (Dong et al., 2014)	79.5	—	—	—	—	—	86.3
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	—	—	93.6	—	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	—	—	93.4	—	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	—	—	93.6	—	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	—	—	—	—	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	—	—	—	—	—	82.7	—
SVM _S (Silva et al., 2011)	—	—	—	—	95.0	—	—

Discussion

- CNNs often used for classification tasks (e.g., sentiment analysis)
- Simple architectures perform well across the board.
- Features are **learned** through filters and pooling operations.
- A window-size- k kernel extracts **local** features from k -grams
- Max pooling reduces dimensions, forcing the network to discriminate important features.