

# 92586 Computational Linguistics

## Lesson 16. Convolutions in Text

Alberto Barrón-Cedeño

Alma Mater Studiorum-Università di Bologna  
a.barron@unibo.it @albarron

27/04/2022



### Quick Keras Reminder

## Table of Contents

Quick Keras Reminder

Prologue to CNN and RNN

CNN

Quick Reminders on CNNs

CNNs for NLP

Chapter 7 of Lane et al. (2019)

## Keras

### **Sequential()**

- ▶ Python class
- ▶ Neural network abstraction
- ▶ Grants access to the basic Keras API

### **Sequential.compile()**

- ▶ Builds the underlying weights
- ▶ Builds the interconnected relationships

### **Sequential.fit()**

- ▶ Computes the training errors (loss)
- ▶ Applies backpropagation (weight adjustment)

### **Some “cooking” hyperparameters**

epochs number of iterations over the data  
batch\_size number of instances before adjusting  
optimizer function

## Prologue to CNN and RNN

## Prologue

- ▶ We have learned to build embedding spaces for words and texts
- ▶ We are considering the neighborhood of the words (~the bag)
- ▶ We are not considering *actual* connections yet
- ▶ The downstream application is usually classification or regression

We will start heading towards text generation

Words have relations and influence each other

Word order

$s_1$  = The dog chased the cat.

$s_2$  = The cat chased the dog.

$$\text{sim}(\text{tfidf}(s_1), \text{tfidf}(s_2)) = 1$$

$$\text{sim}(\text{wv}(s_1), \text{wv}(s_2)) = 1$$

$$\text{sim}(\text{dv}(s_1), \text{dv}(s_2)) = 1$$

But  $s_1$  and  $s_2$  are not the same!

**Word proximity**

$s$  = His mother, besides her son's willingness to amend  
the issue, decided to punish him

mother... decided | son... him

(Lane et al., 2019, p. 220)

Words have relations and influence each other

**Spatial relation**

Consider the position of words  
(~written)

→ fixed-width window

**convolutional neural networks**

**Temporal relation**

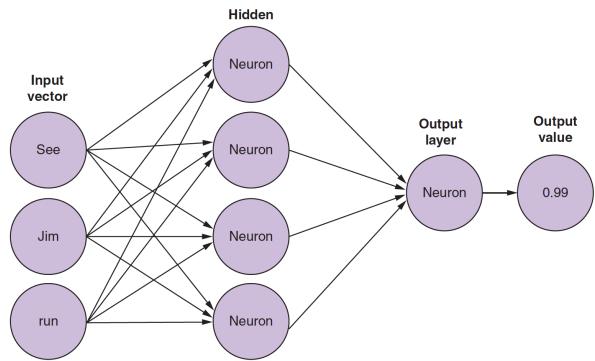
Consider words as time series  
(~spoken)

→ ongoing (unk) amount of time

**recurrent neural networks**

(Lane et al., 2019, p. 220)

## Multiple Input Words



- ▶ Three tokens are passed at a time
- ▶ Two input alternatives
  - ▶ one-hot vector
  - ▶ pre-trained word vector

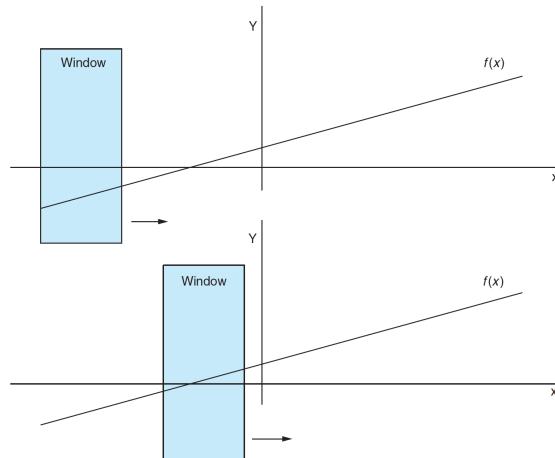
See Jim run  $\neq$  run See Jim (!)

(Lane et al., 2019, p. 221)

## CNN

## Convolutional Neural Networks

Sliding —or convolving<sup>1</sup>— a window over the sample

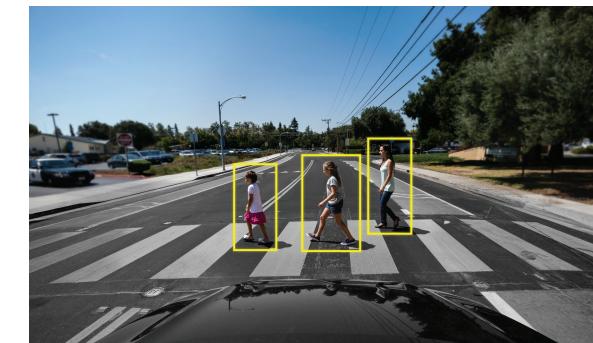


<sup>1</sup>To roll or wind together (Webster's)  
(Lane et al., 2019, p. 222)

## Convolutional Neural Networks

Back to the roots: image recognition

- ▶ Input: pixels of an image
- ▶ Output: the image contains  $x$



# Convolutional Neural Networks

When the input is an image

- B&W: [0,1] (with a smooth binariser)
- Grayscale: [0, 255]
- Colour: R: [0, 255] G: [0, 255] B: [0, 255]



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
1	111	109	111	106	107	118	120	110	104	117	120	118	119	121	114	119	118	120	118	114	119	109	102	122	114	117		
2	120	119	120	116	117	120	115	117	117	120	118	120	119	121	114	119	118	120	118	114	119	109	102	122	114	117		
3	109	102	114	117	108	116	108	111	117	111	118	117	119	117	118	119	117	118	116	117	118	114	117	119	118	117		
4	108	109	109	108	109	112	115	111	117	117	117	117	116	116	112	116	118	116	114	110	41	81	110	102	119	116	117	
5	102	112	109	98	106	93	111	112	115	111	117	117	117	117	116	112	116	116	114	110	41	81	110	102	119	116	117	
6	111	111	109	105	108	110	103	103	106	111	115	118	114	108	103	113	114	115	115	48	87	105	105	114	113	115		
7	111	111	109	105	108	110	103	103	106	111	115	118	114	108	103	113	114	115	115	48	87	105	105	114	113	115		
8	109	109	109	108	109	110	103	103	106	111	115	118	114	108	103	113	114	115	115	48	87	105	105	114	113	115		
9	109	109	109	108	109	110	103	103	106	111	115	118	114	108	103	113	114	115	115	48	87	105	105	114	113	115		
10	99	111	102	86	95	109	104	103	106	106	108	116	114	112	113	111	113	111	113	111	43	89	104	104	109	114	115	
11	109	109	109	108	109	110	103	103	106	111	115	118	114	108	103	113	114	115	115	48	87	105	105	114	113	115		
12	101	101	101	101	101	107	107	108	110	107	103	111	111	110	109	106	112	111	111	109	27	82	108	107	111	112	111	
13	100	102	102	102	102	95	92	99	99	97	103	107	106	106	102	101	106	104	106	102	23	75	105	102	109	108	98	
14	100	102	102	102	102	95	92	99	99	97	103	107	106	106	102	101	106	104	106	102	23	75	105	102	109	108	98	
15	102	102	102	102	102	95	92	99	99	97	103	107	106	106	102	101	106	104	106	102	23	75	105	102	109	108	98	
16	71	82	87	85	78	69	104	104	99	108	106	106	105	106	103	103	106	107	103	21	72	105	106	109	109	109		
17	69	82	87	85	78	69	104	104	99	108	106	106	105	106	103	103	106	107	103	21	72	105	106	109	109	109		
18	69	82	87	85	78	69	104	104	99	108	106	106	105	106	103	103	106	107	103	21	72	105	106	109	109	109		
19	68	82	87	85	78	69	104	104	99	108	106	106	105	106	103	103	106	107	103	21	72	105	106	109	109	109		
20	84	98	87	94	101	100	101	101	101	101	101	101	100	98	98	100	96	97	97	12	71	100	97	93	105	91	101	
21	77	80	88	92	94	97	94	95	93	92	92	92	91	94	96	95	98	89	12	79	105	91	100	89	98	104		
22	77	80	88	92	94	97	94	95	93	92	92	92	91	94	96	95	98	89	12	79	105	91	100	89	98	104		
23	60	64	82	92	96	90	87	98	94	94	94	94	92	92	94	95	91	91	91	0	76	99	92	81	77	65	58	
24	60	64	82	92	96	90	87	98	94	94	94	94	92	92	94	95	91	91	91	0	76	99	92	81	77	65	58	
25	87	88	88	83	85	91	91	89	90	91	92	93	91	91	90	89	90	89	91	73	8	66	92	83	84	92	91	91
26	81	86	86	81	89	93	93	89	93	93	93	93	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	

(Lane et al., 2019, p. 223)

## Filter

- $n \times m$  surfaces
- Typically  $n = m = 3$  (often  $n \neq m$ )
- Includes a set of weights (fix for the whole image)
- Includes an activation function: usually ReLU

$$z = \max(\sum(x * w), 0)$$

# Convolutional Neural Networks

When the input is an image

## An image is just a bunch of numbers

- Appropriate as input for a NN
- But one single pixel has no real meaning

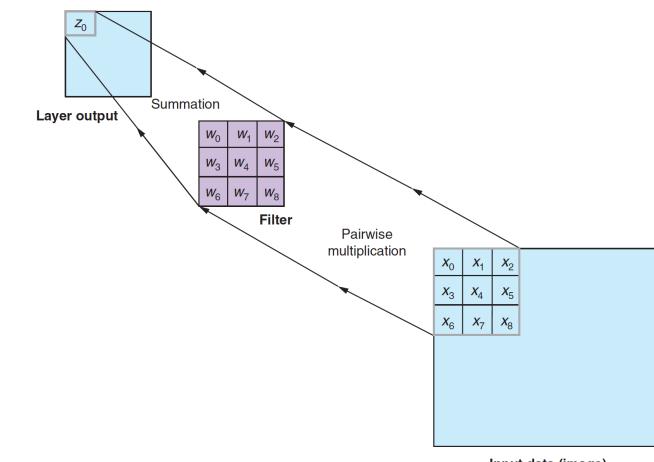
## → Sliding over fragments of the image

The convolution defines a set of filters (aka kernels) to do just that

- Take “snapshots” of different areas of the image
- Process them, one at a time

# Convolutional Neural Networks

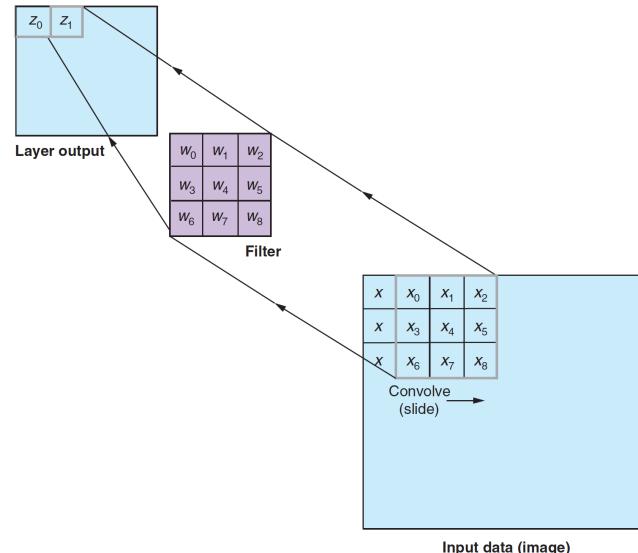
## Convolutional step



(Lane et al., 2019, p. 225)

## Convolutional Neural Networks

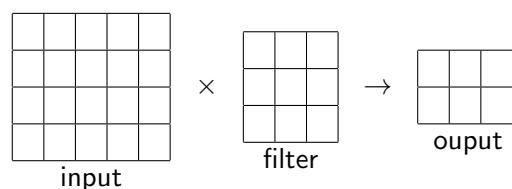
### Convolution



(Lane et al., 2019, p. 226)

## Convolutional Neural Networks

### Padding



### We are producing smaller images

"I don't care": Keras' argument `padding='valid'`  
The edges of the image are undersampled  
"I do care": Keras' `padding argument padding='same'`

In NLP we care

## Convolutional Neural Networks

### Producing multiple images

- ▶  $k$  filters exist which carry out different operations
- ▶ Every filter will produce a new image, combination of source and filter

## Convolutional Neural Networks

### Pipeline

**Input:** an image, text

**Output:** a class, a real number

- ▶ Produce  $k$  new images through  $k$  filters
- ▶ Wire the filtered images to a feed-forward network
- ▶ Proceed as usual

### We can add multiple convolution layers

A full path of learning layers and abstractions

- ▶ Edges
- ▶ Shapes
- ▶ Colours
- ▶ Concepts

### What is learned

- ▶ Good filters
- ▶ "Standard" weights

## Convolutional Neural Networks

Keras premier

```
from keras.models import Sequential
from keras.layers import Conv1D

model = Sequential()

model.add(Conv1D(filters=16,
                 kernel_size=3,
                 padding='same',
                 activation='relu',
                 strides=1,
                 input_shape=(100, 300))
)
```

### Quick Reminders on CNNs

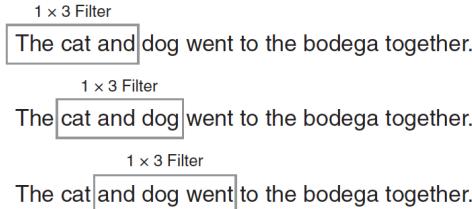
#### Quick Reminders on CNNs

- ▶ Sliding —or convolving— a window over the sample
- ▶ Filters (kernels; matrices) slide over fragments of the image
- ▶ “Snapshots” of different areas of the image are taken and processed
- ▶ Multiple filters produce multiple images
- ▶ Multiple convolution layers can be added
- ▶ At the end, we can plug a “standard” fully-connected NN

### CNNs for NLP

## Back to Text

- ▶ In images both vertical and horizontal relationships are relevant
- ▶ In text only horizontal ones do<sup>2</sup>
- ▶ **We need “1D” filters**

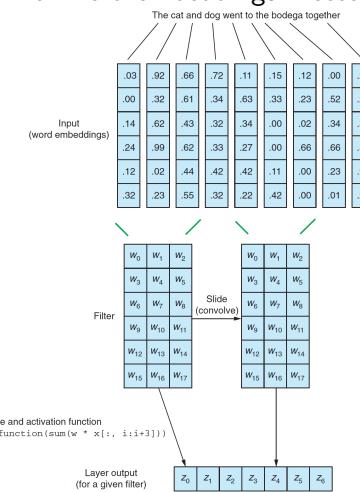


<sup>2</sup>|l2r or r2l; for some languages it's the vertical direction that matters (e.g., Japanese)

(Lane et al., 2019, p. 229)

## But we do have 2D “filters”

Words are represented with word embeddings: vectors



(Lane et al., 2019, p. 229)

## The convolution is (practically) the same as for images

- ▶ We now *convolve* in one dimension (not two)
- ▶ The computation order is irrelevant, but the outputs have to be fed in the same order
- ▶ The filters' weights are fixed for a full sample (parallel computation)
- ▶ Their output becomes the features for the classifier

✍ Let us see

## Padding

- ▶ (In general) in image processing the inputs are of fixed size, regardless of the image (same source!)
- ▶ Texts are not fixed length (regardless of their source)
- ▶ Instances longer than maxlen will be truncated
- ▶ Instances shorter than maxlen will be **padded**

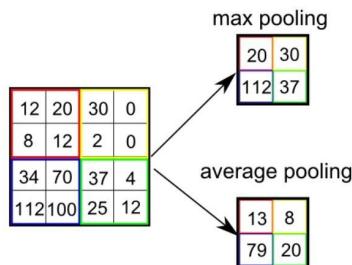
$x_0, x_1, x_2, x_3, \dots, x_{398}, x_{399}, x_{400}, x_{401}$

$x_0, x_1, x_2, x_3, \dots, x_{397}$  PAD PAD

✍ Let us see

## Pooling

- ▶ For each filter one new version of the instance is produced (250 in the example)
- ▶ Pooling evenly divides the output of each filter into subsections
- ▶ It selects (or computes) a representative value for each subsection



- ▶ Average is more intuitive: retaining most of the info
- ▶ Max is better: the NN keeps the most prominent feature

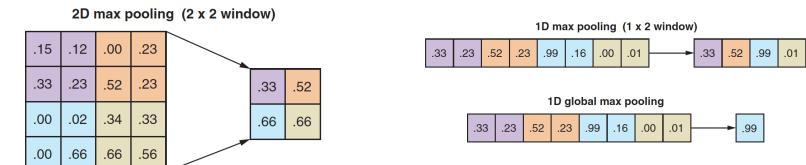
✍ Let us see

Image borrowed from [www.quora.com/What-is-max-pooling-in-convolutional-neural-networks](http://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks)

## Pooling

Pooling is “the CNN path to dimensionality reduction [...] by learning higher-order representations of the source data” (Lane et al., 2019, p. 236)

- ▶ The filters job is finding patterns → relationships between words and their neighbours
- ▶ Pooling in text: a 1D window (e.g.,  $1 \times 2$  or  $1 \times 3$ )



(Lane et al., 2019, p. 237)

## Pooling

Max vs Average Pooling

## Recap

- ▶ Each filter will produce a  $1 \times 398$
- ▶ For each of the 250 filter outputs, we take the single maximum value for each 1D vector
- ▶ Output: one  $1 \times 250$  vector

**This is a crude semantic representation of the text**

## Dropout: Preventing Overfitting

On each training pass **turn off** a percentage of the input of a layer; it will become 0

- ▶ Chosen randomly on each pass
- ▶ It will not rely heavily on any feature
- ▶ It will generalise better
- ▶ Dropout is applied during training only



⌚ Let us see

Photogram of "The Platform" (2019)

## Closing Remarks

- ▶ Your input is a series of max 400 words; 300 elements each
- ▶ Nothing prevents you from stacking other embeddings (think of RGB)
- ▶ The output of the convolution layer is tied to the task (in this case, sentiment analysis)
- ▶ A CNN is more efficient, thanks to the pooling process and the filters
- ▶ You can add many convolution layers

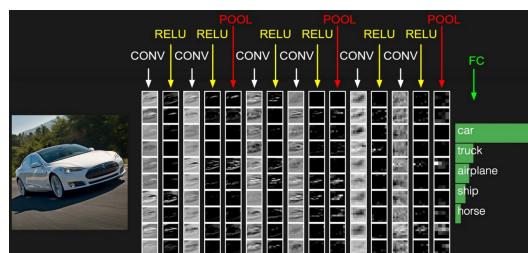


Image borrowed from <https://blog.mapillary.com>

## Workhorse Loss Functions

Out of the 13+ available loss functions:

binary\_crossentropy: the output neuron is either on or off

categorical\_crossentropy: the output is one out of many classes

⌚ Let us see

## Next time

- ▶ Recurrent Neural Networks

## References

Lane, H., C. Howard, and H. Hapke  
2019. *Natural Language Processing in Action*. Shelter Island,  
NY: Manning Publication Co.