

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



Table of Contents

- 1 Quick Keras Reminder
- 2 Prologue to CNN and RNN
- 3 CNN
- 4 Quick Reminders on CNNs
- 5 CNNs for NLP

Chapter 7 of Lane et al. (2019)

Quick Keras Reminder

Keras

Sequential()

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

Keras

Sequential()

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

Sequential.compile()

- Builds the underlying weights
- Builds the interconnected relationships

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)

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

Prologue

- We have learned to build embedding spaces for words and texts
- We are considering the neighborhood of the words (~the bag)

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

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

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.

Words have relations and influence each other

Word order

s_1 = The dog chased the cat.

s_2 = The cat chased the dog.

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

Words have relations and influence each other

Word order

s_1 = The dog chased the cat.

s_2 = The cat chased the dog.

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

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

Words have relations and influence each other

Word order

s_1 = The dog chased the cat.

s_2 = The cat chased the dog.

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

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

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

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!

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

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)

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

Words have relations and influence each other

Spatial relation

Consider the position of words
(~written)

Temporal relation

Consider words as time series
(~spoken)

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

Words have relations and influence each other

Spatial relation

Consider the position of words
(~written)

Temporal relation

Consider words as time series
(~spoken)

→ fixed-width window

Words have relations and influence each other

Spatial relation

Consider the position of words
(~written)

Temporal relation

Consider words as time series
(~spoken)

→ fixed-width window

convolutional neural networks

(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

Words have relations and influence each other

Spatial relation

Consider the position of words
(~written)

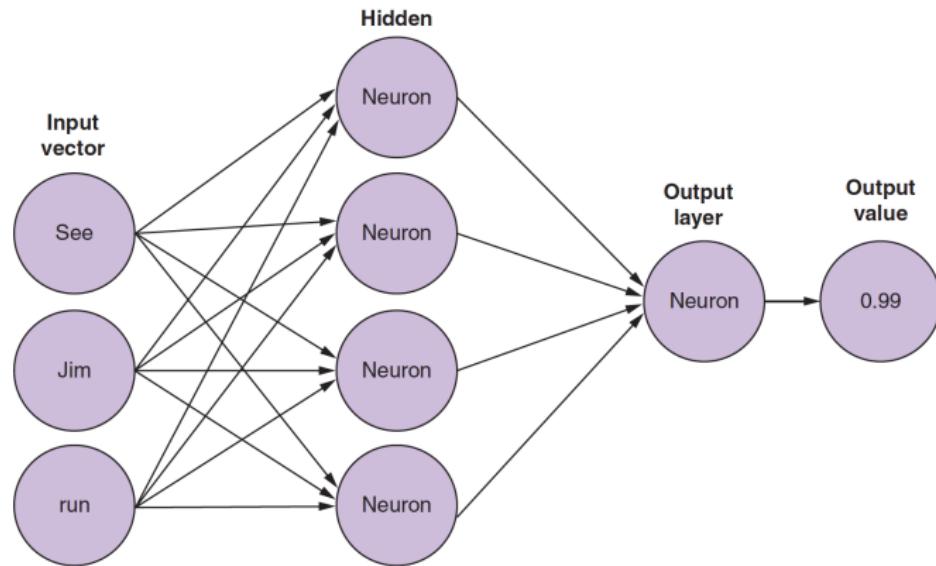
→ fixed-width window
convolutional neural networks

Temporal relation

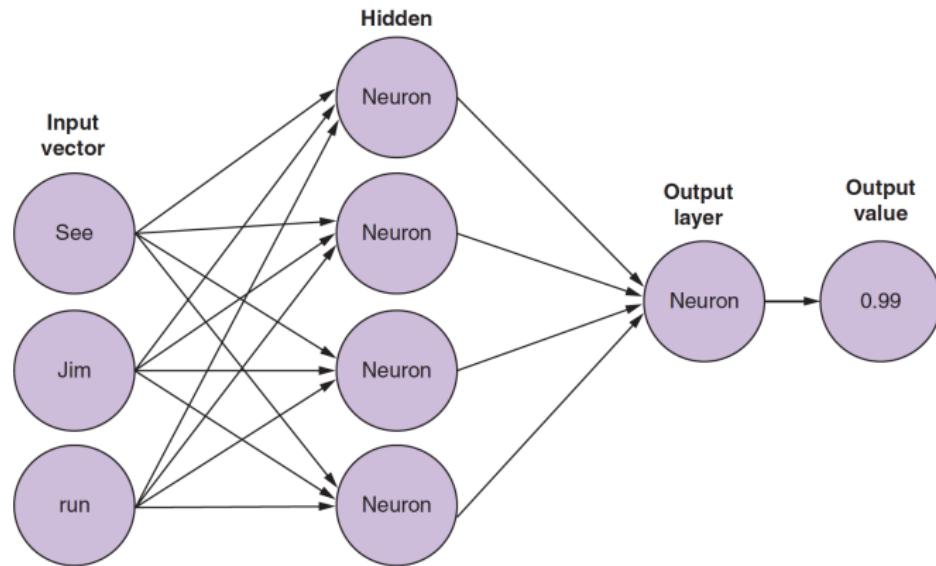
Consider words as time series
(~spoken)

→ ongoing (unk) amount of time
recurrent neural networks

Multiple Input Words

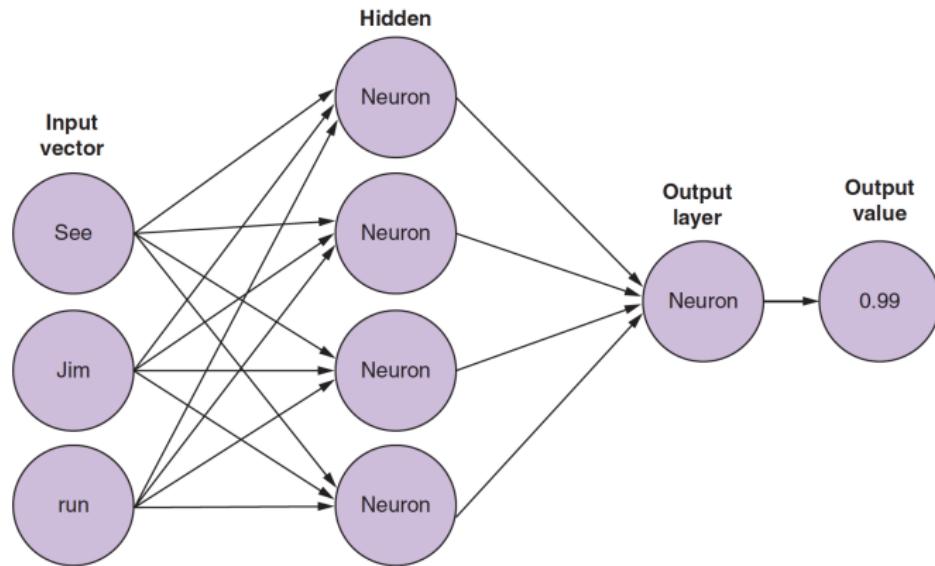


Multiple Input Words



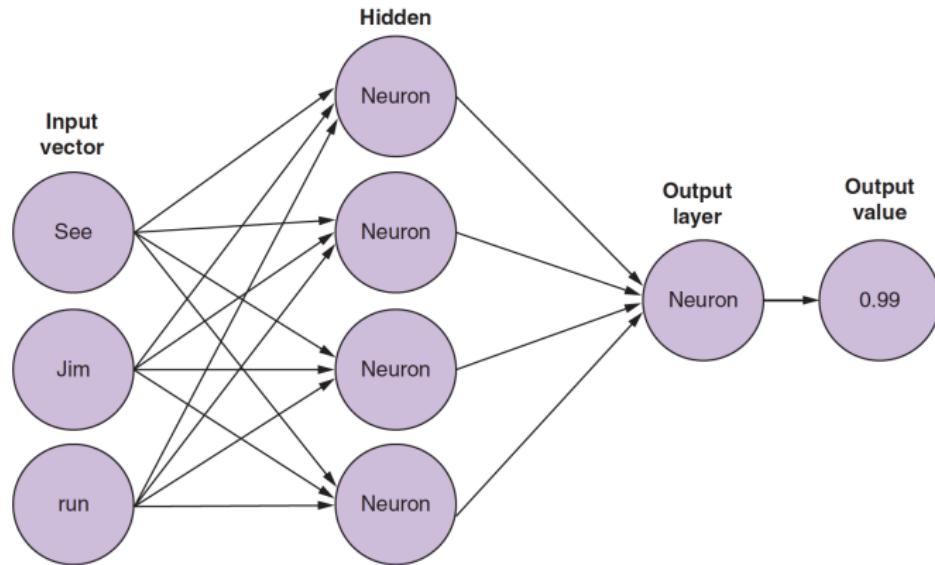
- Three tokens are passed at a time

Multiple Input Words



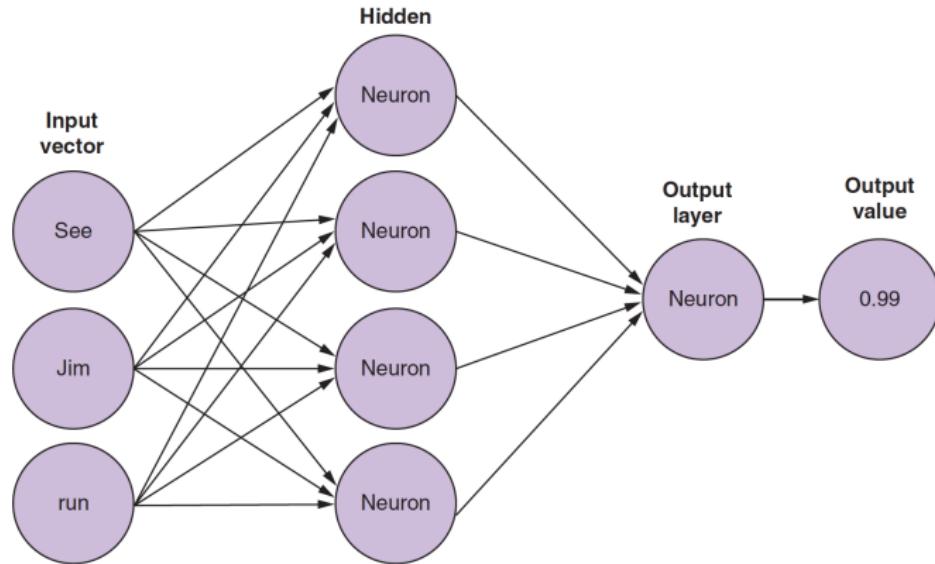
- Three tokens are passed at a time
- Two input alternatives
 - ▶ one-hot vector

Multiple Input Words



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

Multiple Input Words



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

See Jim run ≠ run See Jim (!)

CNN

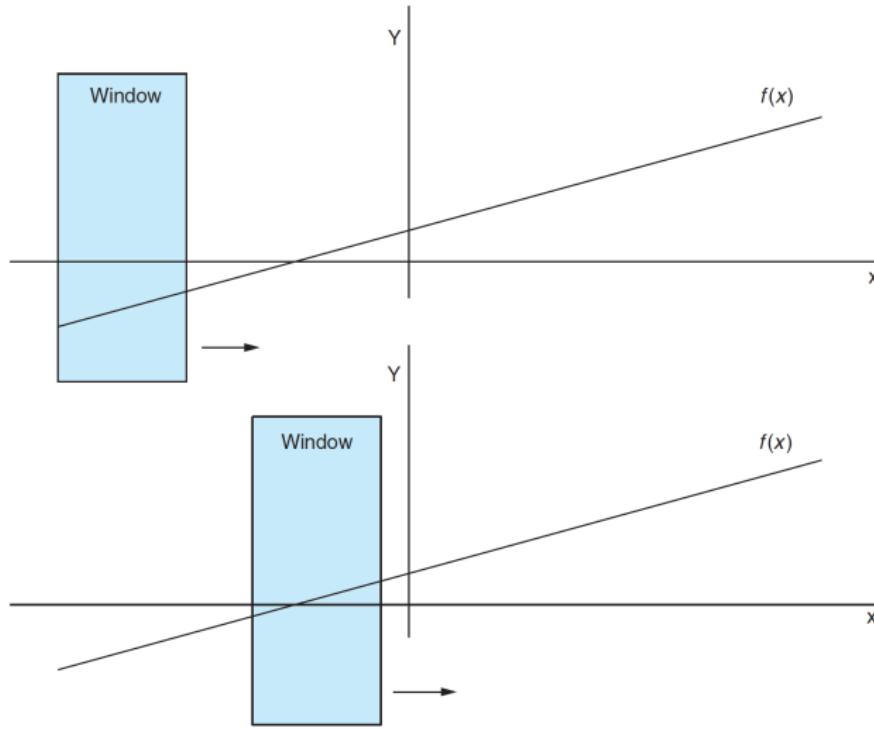
Convolutional Neural Networks

Sliding —or convolving¹— a window over the sample

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

Convolutional Neural Networks

Sliding —or convolving¹— a window over the sample



¹To 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

[https:](https://blogs.nvidia.com/wp-content/uploads/2019/04/ADAS-IMG_0052.jpg)

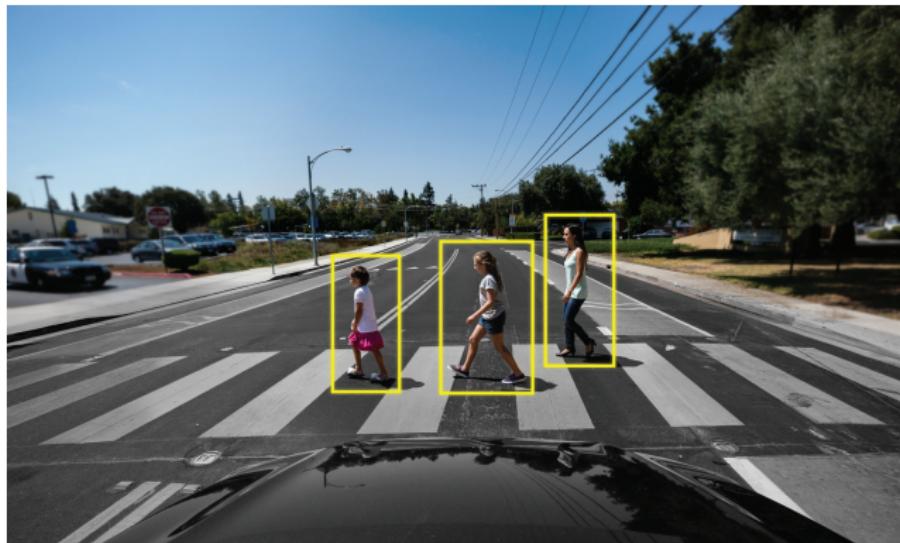
//blogs.nvidia.com/wp-content/uploads/2019/04/ADAS-IMG_0052.jpg



Convolutional Neural Networks

Back to the roots: image recognition

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



[https:](https://blogs.nvidia.com/wp-content/uploads/2019/04/ADAS-IMG_0052.jpg)

//blogs.nvidia.com/wp-content/uploads/2019/04/ADAS-IMG_0052.jpg

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]

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]



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

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 | | | | | | |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|----|----|
| 0 | 120 | 119 | 118 | 108 | 103 | 111 | 113 | 115 | 111 | 117 | 120 | 120 | 120 | 119 | 121 | 114 | 118 | 120 | 120 | 120 | 121 | 121 | 121 | 115 | 100 | 120 | 118 | 117 | | | | | | |
| 1 | 111 | 109 | 111 | 106 | 107 | 119 | 120 | 110 | 106 | 117 | 120 | 119 | 119 | 121 | 114 | 118 | 119 | 118 | 119 | 119 | 118 | 114 | 119 | 109 | 102 | 122 | 114 | 117 | | | | | | |
| 2 | 109 | 104 | 114 | 117 | 108 | 116 | 108 | 111 | 117 | 118 | 118 | 119 | 119 | 119 | 121 | 114 | 118 | 119 | 118 | 119 | 119 | 118 | 114 | 119 | 110 | 103 | 121 | 112 | 119 | | | | | |
| 3 | 109 | 102 | 114 | 117 | 108 | 116 | 108 | 111 | 117 | 118 | 118 | 119 | 119 | 119 | 121 | 114 | 118 | 119 | 118 | 119 | 119 | 118 | 114 | 119 | 110 | 103 | 120 | 112 | 119 | | | | | |
| 4 | 108 | 110 | 100 | 116 | 111 | 95 | 104 | 110 | 114 | 117 | 117 | 117 | 117 | 117 | 117 | 117 | 106 | 107 | 108 | 94 | 90 | 42 | 54 | 70 | 62 | 86 | 70 | 96 | 118 | | | | | |
| 5 | 103 | 112 | 109 | 98 | 100 | 92 | 111 | 112 | 115 | 113 | 117 | 117 | 117 | 117 | 117 | 116 | 116 | 112 | 118 | 116 | 114 | 110 | 41 | 81 | 81 | 102 | 119 | 114 | 119 | 117 | | | | |
| 6 | 103 | 112 | 109 | 98 | 100 | 92 | 111 | 112 | 115 | 113 | 117 | 117 | 117 | 117 | 117 | 117 | 116 | 116 | 112 | 118 | 116 | 114 | 110 | 41 | 81 | 81 | 102 | 119 | 114 | 119 | 117 | | | |
| 7 | 111 | 111 | 109 | 105 | 103 | 110 | 103 | 103 | 106 | 111 | 115 | 115 | 114 | 114 | 114 | 114 | 115 | 115 | 115 | 115 | 115 | 115 | 115 | 105 | 105 | 114 | 111 | 116 | 115 | | | | | |
| 8 | 112 | 105 | 108 | 94 | 89 | 102 | 95 | 106 | 111 | 113 | 111 | 108 | 106 | 109 | 109 | 114 | 113 | 113 | 115 | 115 | 115 | 115 | 115 | 103 | 104 | 108 | 111 | 112 | 114 | 114 | | | | |
| 9 | 112 | 105 | 108 | 86 | 95 | 109 | 104 | 103 | 106 | 104 | 108 | 109 | 114 | 114 | 113 | 108 | 112 | 113 | 114 | 114 | 114 | 113 | 109 | 111 | 113 | 114 | 114 | 113 | | | | | | |
| 10 | 110 | 106 | 103 | 93 | 93 | 102 | 108 | 109 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | 111 | | | | | |
| 11 | 110 | 106 | 93 | 96 | 108 | 107 | 110 | 109 | 111 | 112 | 108 | 107 | 111 | 112 | 111 | 111 | 107 | 111 | 111 | 111 | 110 | 38 | 78 | 78 | 109 | 108 | 108 | 111 | 114 | 103 | | | | |
| 12 | 101 | 93 | 76 | 101 | 103 | 107 | 107 | 108 | 110 | 107 | 103 | 111 | 111 | 110 | 109 | 106 | 112 | 111 | 111 | 109 | 37 | 82 | 82 | 108 | 107 | 111 | 113 | 111 | 100 | | | | | |
| 13 | 98 | 92 | 99 | 115 | 108 | 109 | 111 | 106 | 100 | 98 | 106 | 108 | 109 | 110 | 107 | 106 | 109 | 106 | 109 | 107 | 37 | 78 | 78 | 106 | 103 | 106 | 108 | 103 | 98 | | | | | |
| 14 | 98 | 73 | 97 | 103 | 92 | 95 | 95 | 93 | 97 | 97 | 97 | 97 | 97 | 97 | 97 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 97 | 97 | | | | |
| 15 | 84 | 89 | 92 | 87 | 75 | 82 | 89 | 95 | 95 | 98 | 100 | 107 | 107 | 107 | 108 | 106 | 104 | 104 | 108 | 107 | 109 | 105 | 29 | 74 | 74 | 107 | 107 | 110 | 106 | 99 | 109 | | | |
| 16 | 71 | 82 | 87 | 85 | 78 | 89 | 106 | 104 | 99 | 106 | 106 | 105 | 106 | 105 | 104 | 103 | 106 | 106 | 107 | 103 | 21 | 72 | 72 | 106 | 107 | 109 | 100 | 102 | 109 | | | | | |
| 17 | 67 | 87 | 64 | 68 | 84 | 89 | 98 | 96 | 99 | 104 | 104 | 104 | 104 | 104 | 105 | 103 | 101 | 105 | 103 | 105 | 102 | 23 | 76 | 76 | 103 | 103 | 106 | 98 | 108 | 101 | | | | |
| 18 | 64 | 71 | 77 | 72 | 74 | 82 | 82 | 91 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | | | | |
| 19 | 60 | 71 | 77 | 77 | 80 | 88 | 82 | 91 | 93 | 96 | 96 | 101 | 100 | 101 | 100 | 98 | 101 | 101 | 104 | 92 | 13 | 64 | 93 | 89 | 81 | 89 | 81 | 81 | 79 | | | | | |
| 20 | 84 | 98 | 87 | 94 | 101 | 108 | 101 | 101 | 103 | 101 | 101 | 101 | 101 | 103 | 98 | 98 | 100 | 96 | 97 | 87 | 12 | 71 | 100 | 97 | 93 | 105 | 91 | 101 | 101 | | | | | |
| 21 | 92 | 80 | 88 | 59 | 99 | 100 | 98 | 100 | 100 | 97 | 97 | 97 | 97 | 96 | 92 | 91 | 92 | 91 | 96 | 87 | 13 | 79 | 105 | 95 | 98 | 96 | 96 | 99 | 100 | | | | | |
| 22 | 81 | 87 | 83 | 84 | 89 | 89 | 91 | 87 | 90 | 92 | 93 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | 94 | | | | |
| 23 | 81 | 87 | 83 | 84 | 89 | 89 | 91 | 87 | 90 | 92 | 93 | 95 | 94 | 94 | 94 | 94 | 95 | 95 | 94 | 94 | 93 | 84 | 7 | 74 | 89 | 86 | 90 | 70 | 81 | 73 | | | | |
| 24 | 60 | 66 | 82 | 92 | 90 | 90 | 87 | 90 | 94 | 94 | 94 | 94 | 93 | 94 | 94 | 94 | 95 | 95 | 95 | 91 | 77 | 4 | 73 | 91 | 92 | 81 | 77 | 65 | 58 | | | | | |
| 25 | 87 | 81 | 83 | 86 | 87 | 84 | 90 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 91 | 92 | 92 | 92 | 92 | 92 | 77 | 4 | 73 | 91 | 92 | 81 | 95 | 96 | 95 | | | | |
| 26 | 87 | 88 | 88 | 83 | 85 | 93 | 91 | 89 | 90 | 93 | 91 | 91 | 91 | 91 | 91 | 90 | 92 | 90 | 89 | 89 | 89 | 84 | 83 | 79 | 80 | 0 | 60 | 89 | 77 | 90 | 91 | 91 | 91 | |
| 27 | 81 | 86 | 88 | 91 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 84 | 83 | 79 | 80 | 0 | 60 | 89 | 77 | 90 | 91 | 91 | 91 |

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

Convolutional Neural Networks

When the input is an image

An image is just a bunch of numbers

Convolutional Neural Networks

When the input is an image

An image is just a bunch of numbers

- Appropriate as input for a NN

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

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

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

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

Strides and filters

Stride

- The distance “traveled” when sliding
- Yet another parameter
- Never bigger than the size of the filter → overlapping areas

Convolutional Neural Networks

Strides and filters

Stride

- The distance “traveled” when sliding
- Yet another parameter
- Never bigger than the size of the filter → overlapping areas

Sounds familiar?

Convolutional Neural Networks

Strides and filters

Stride

- The distance “traveled” when sliding
- Yet another parameter
- Never bigger than the size of the filter → overlapping areas

Sounds familiar? *n-grams!*

Convolutional Neural Networks

Strides and filters

Stride

- The distance “traveled” when sliding
- Yet another parameter
- Never bigger than the size of the filter → overlapping areas

Sounds familiar? *n-grams!*

Filter

- $n \times m$ surfaces

Convolutional Neural Networks

Strides and filters

Stride

- The distance “traveled” when sliding
- Yet another parameter
- Never bigger than the size of the filter → overlapping areas

Sounds familiar? *n-grams!*

Filter

- $n \times m$ surfaces
- Typically $n = m = 3$ (often $n \neq m$)

Convolutional Neural Networks

Strides and filters

Stride

- The distance “traveled” when sliding
- Yet another parameter
- Never bigger than the size of the filter → overlapping areas

Sounds familiar? *n-grams!*

Filter

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

Convolutional Neural Networks

Strides and filters

Stride

- The distance “traveled” when sliding
- Yet another parameter
- Never bigger than the size of the filter → overlapping areas

Sounds familiar? *n-grams!*

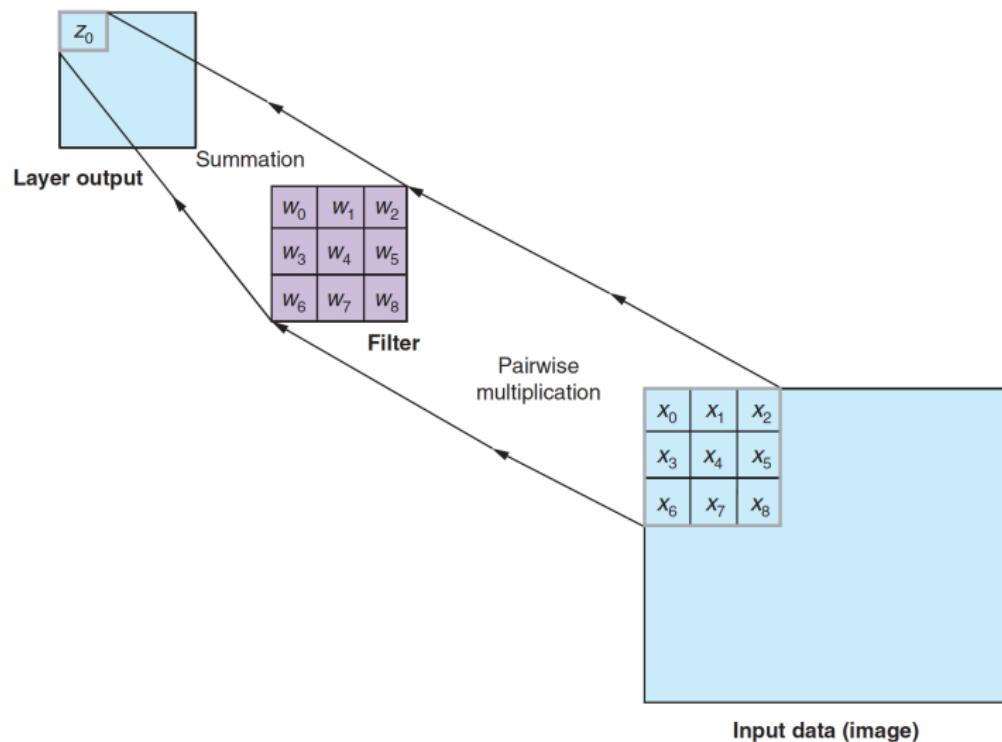
Filter

- $n \times m$ 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

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

Convolutional Neural Networks

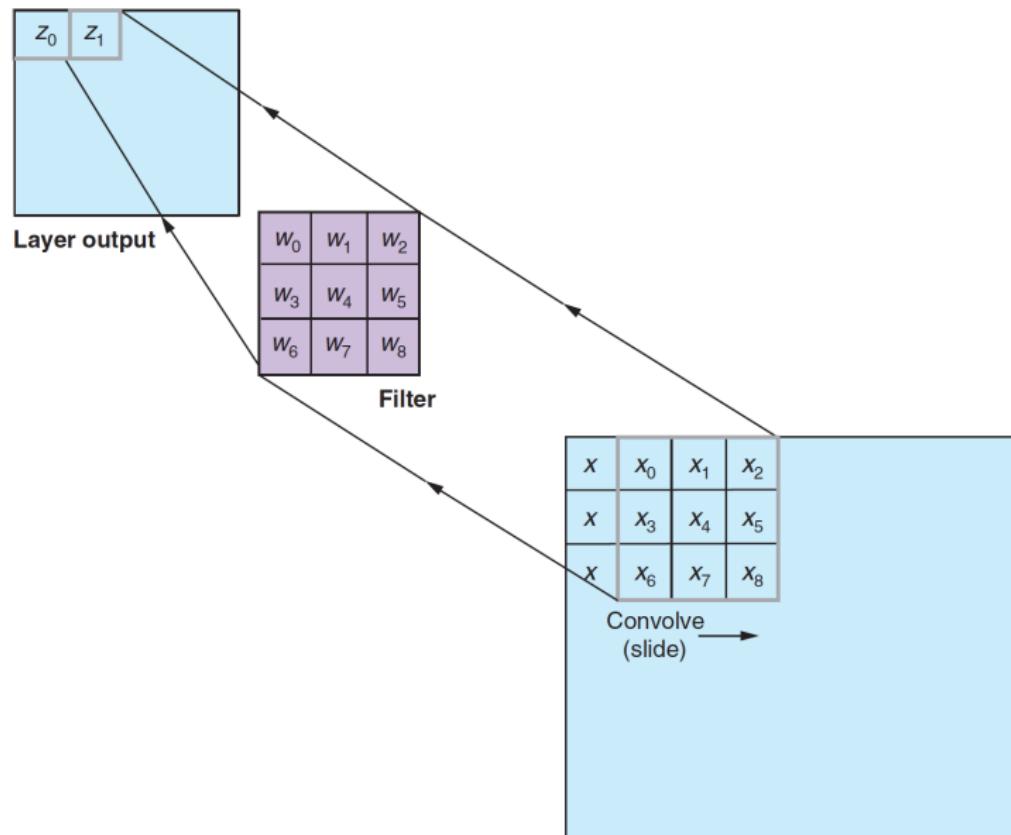
Convolutional step



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

Convolutional Neural Networks

Convolution



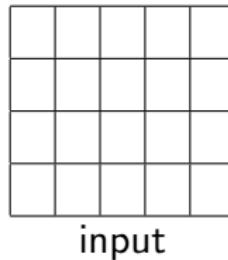
Convolutional Neural Networks

Producing multiple images

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

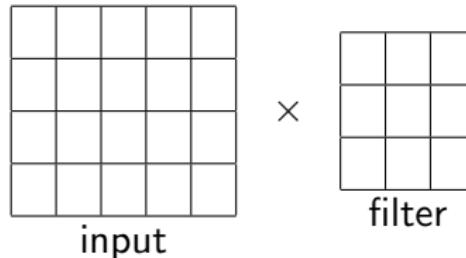
Convolutional Neural Networks

Padding



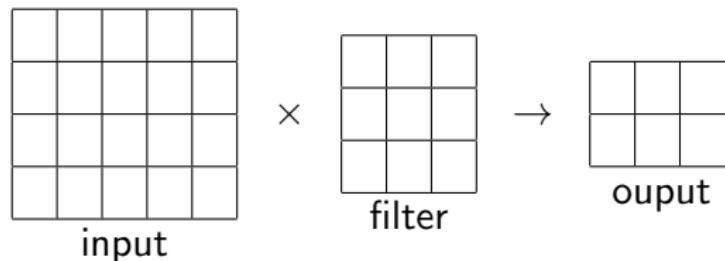
Convolutional Neural Networks

Padding



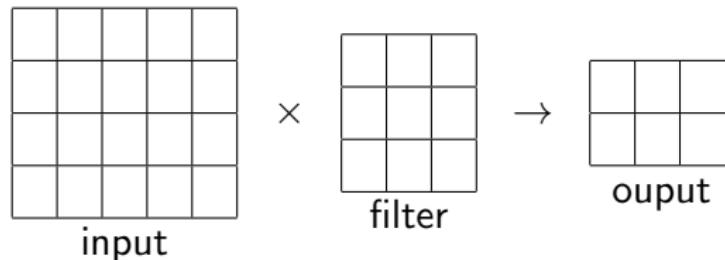
Convolutional Neural Networks

Padding



Convolutional Neural Networks

Padding

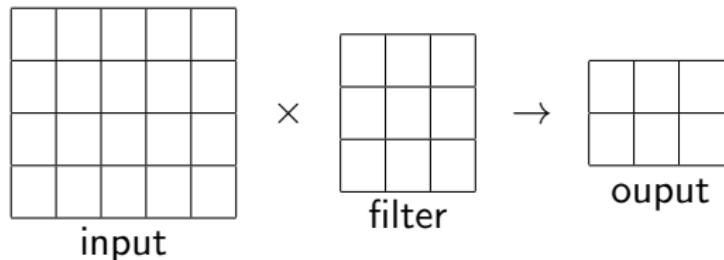


We are producing smaller images

"I don't care": Keras' argument padding='valid'

Convolutional Neural Networks

Padding



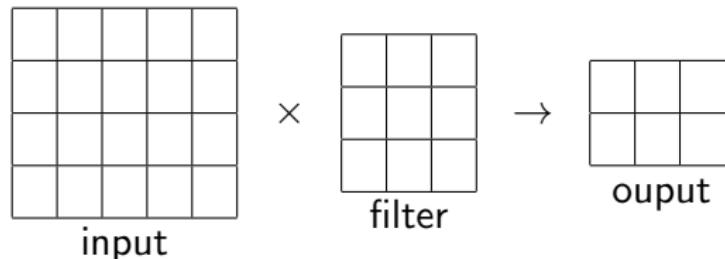
We are producing smaller images

"I don't care": Keras' argument padding='valid'

The edges of the image are undersampled

Convolutional Neural Networks

Padding



We are producing smaller images

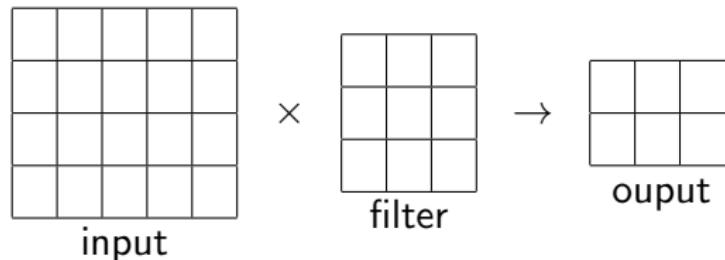
“I don't care”: Keras' argument padding='valid'

The edges of the image are undersampled

“I do care”: Keras' padding argument padding='same'

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

Pipeline

Input: an image, text

Output: a class, a real number

- Produce k new images through k filters

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

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

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

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

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

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

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**

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

Quick Reminders on CNNs

- Sliding —or convolving— a window over the sample
- Filters (kernels; matrices) slide over fragments of the image

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

Quick Reminders on CNNs

- Sliding —or convolving— a window over the sample
- Filters (kernels; matrices) slide over fragments of the image
- “Sna^pshots” of different areas of the image are taken and processed
- Multiple filters produce multiple images

Quick Reminders on CNNs

- Sliding —or convolving— a window over the sample
- Filters (kernels; matrices) slide over fragments of the image
- “Sna^pshots” of different areas of the image are taken and processed
- Multiple filters produce multiple images
- Multiple convolution layers can be added

Quick Reminders on CNNs

- Sliding —or convolving— a window over the sample
- Filters (kernels; matrices) slide over fragments of the image
- “Sna^pshots” 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

²|2r or r2l; for some languages it's the vertical direction that matters (e.g., Japanese)
(Lane et al., 2019, p. 229)

Back to Text

- In images both vertical and horizontal relationships are relevant
- In text only horizontal ones do²

²|l2r or r2l; for some languages it's the vertical direction that matters (e.g., Japanese)
(Lane et al., 2019, p. 229)

Back to Text

- In images both vertical and horizontal relationships are relevant
- In text only horizontal ones do²
- **We need “1D” filters**

1 × 3 Filter

The cat and dog went to the bodega together.

1 × 3 Filter

The cat and dog went to the bodega together.

1 × 3 Filter

The cat and dog went to the bodega together.

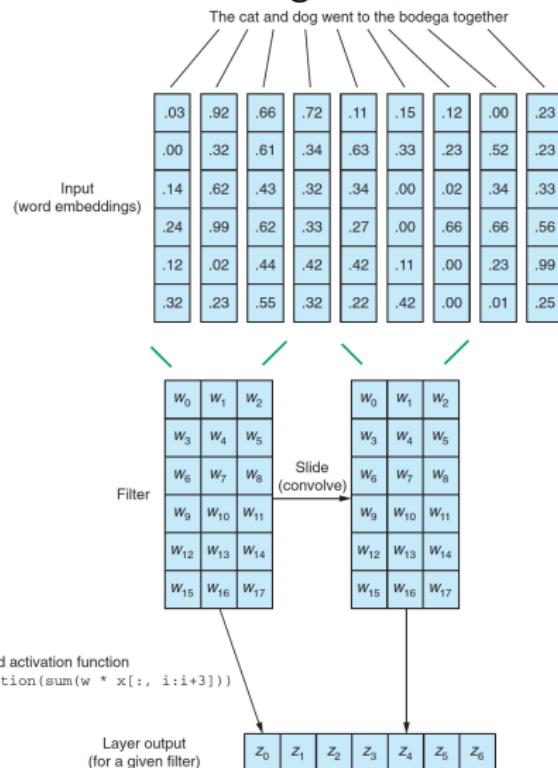
²|l2r or r2l; for some languages it's the vertical direction that matters (e.g., Japanese)
(Lane et al., 2019, p. 229)

But we do have 2D “filters”

Words are represented with word embeddings: vectors

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 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 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)

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

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

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!)

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)

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

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**

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**

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

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

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

- 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

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

Pooling

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

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

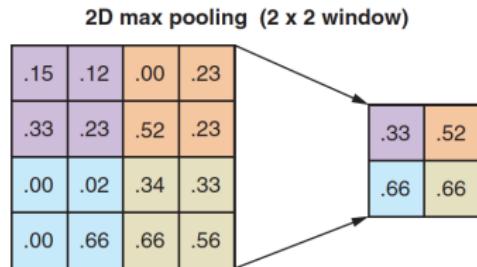
Pooling is “the CNN path to dimensionality reduction [...] by learning higher-order representations of the source data” (Lane et al., 2019, p. 236)

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

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., 1×2 or 1×3)

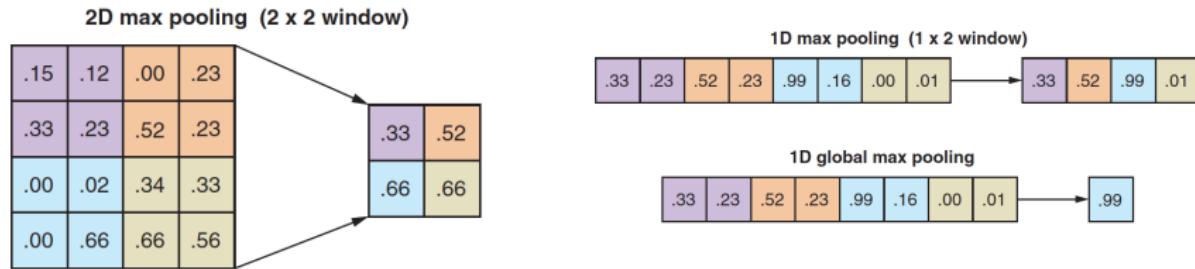


(Lane et al., 2019, p. 237)

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., 1×2 or 1×3)



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

Pooling

Max vs Average Pooling

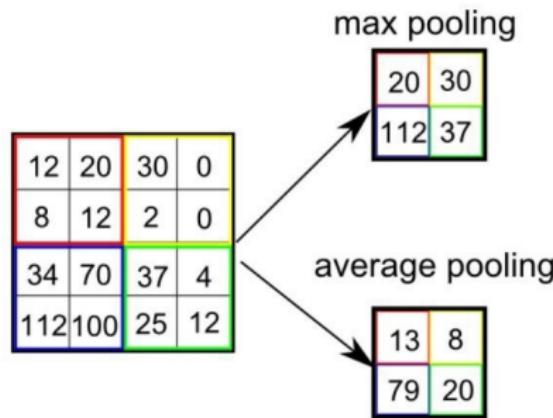
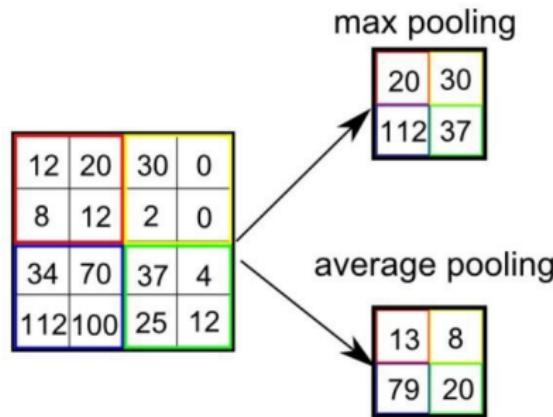


Image borrowed from

www.quora.com/What-is-max-pooling-in-convolutional-neural-networks

Pooling

Max vs Average Pooling



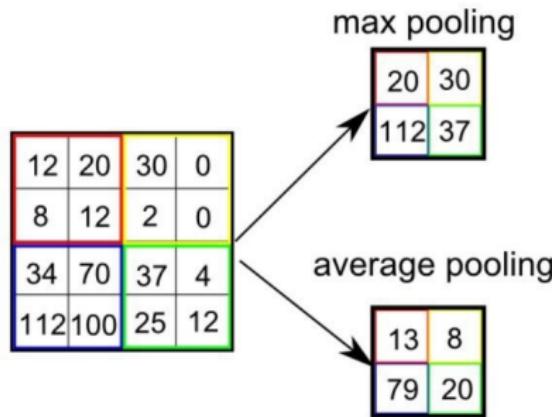
- Average is more intuitive: retaining most of the info

Image borrowed from

www.quora.com/What-is-max-pooling-in-convolutional-neural-networks ↗ ↘ ↙

Pooling

Max vs Average Pooling



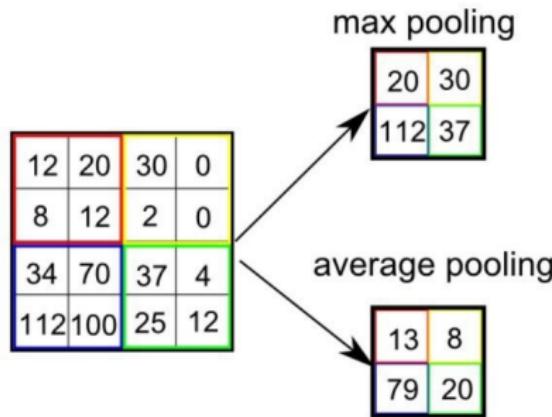
- Average is more intuitive: retaining most of the info
- Max is better: the NN keeps the most prominent feature

Image borrowed from

www.quora.com/What-is-max-pooling-in-convolutional-neural-networks ↗ ↘ ↙

Pooling

Max vs Average Pooling



- 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

Recap

- Each filter will produce a 1×398

Recap

- Each filter will produce a 1×398
- For each of the 250 filter outputs , we take the single maximum value for each 1D vector

Recap

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

Recap

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

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

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: 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

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



Photogram of "The Platform" (2019)

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)

Workhorse Loss Functions

Out of the 13+ available loss functions:

`binary_crossentropy`: the output neuron is either on or off

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

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

Closing Remarks

- Your input is a series of max 400 words; 300 elements each
- Nothing prevents you from stacking other embeddings (think of RGB)

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

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)

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

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

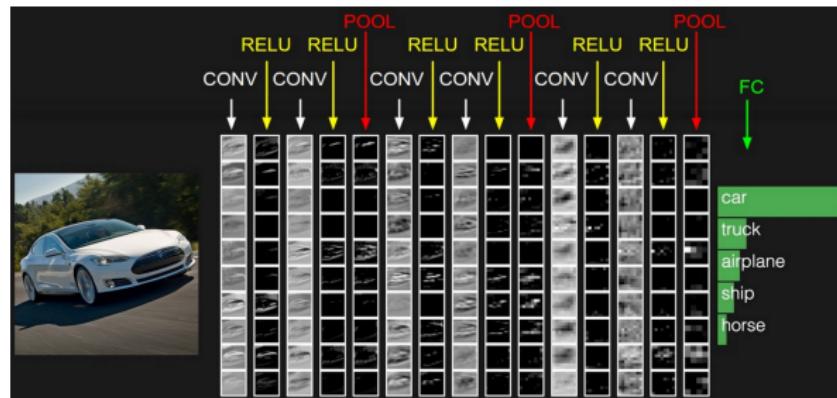


Image borrowed from <https://blog.mapillary.com>

Next time

- Recurrent Neural Networks

References

Lane, H., C. Howard, and H. Hapkem

2019. *Natural Language Processing in Action*. Shelter Island, NY:
Manning Publication Co.