Natural Language and Speech Processing

Lecture 9: Convolutional Models for Text

Tanel Alumäe

Addendum to last lecture/lab

- Is an array like [1.0 2.5 4.7 2.2 5.5] 5-dimensional vector or a 1-dimensional vector with 5 elements?

- Dimensionality of a vector refers to the space of which the vector is a member, in this case R^n
- In other words, vector is a **rank-1 tensor** (or N-dimensional tensor/vector)
- Similarly, a **matrix is rank 2 tensor** (or **NxM dimensional tensor**)

Contents

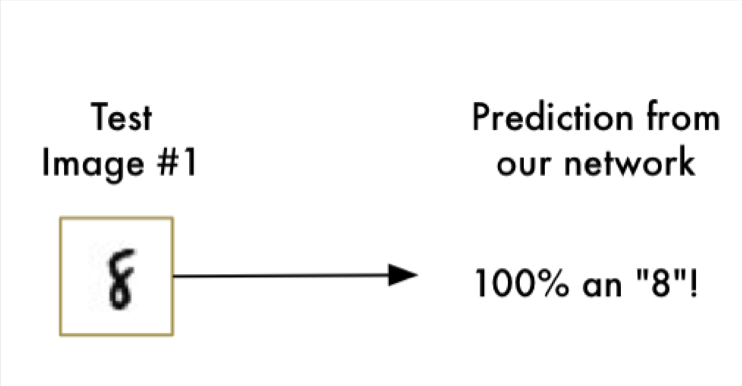
- What are convolutional layers in DNNs?
- Pooling
- Hierarchical convolutions
- Why and how to use convolutional neural networks (CNNs) for text classification?

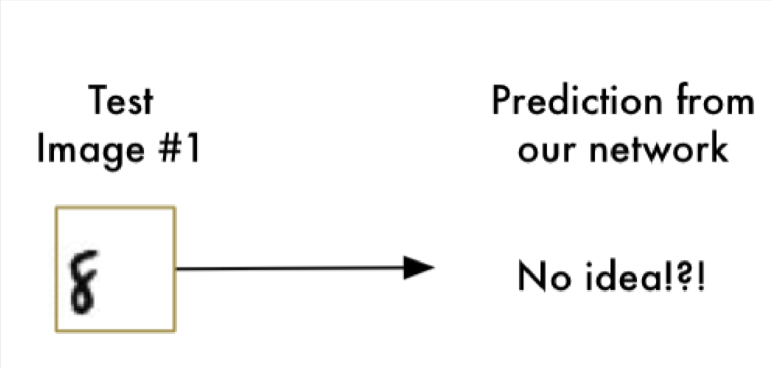
Recognizing objects in images

- Recognizing objects in images is a very active research field
- Most modern systems use convolutional networks
- Available in many end-user systems
  - e.g. search for „cat” or „table” in Google Photos

Why convolutions?

- Example: digit recognition from images
- Works relatively well when the digit is in the centre of the image
- But fails if it's not in the centre



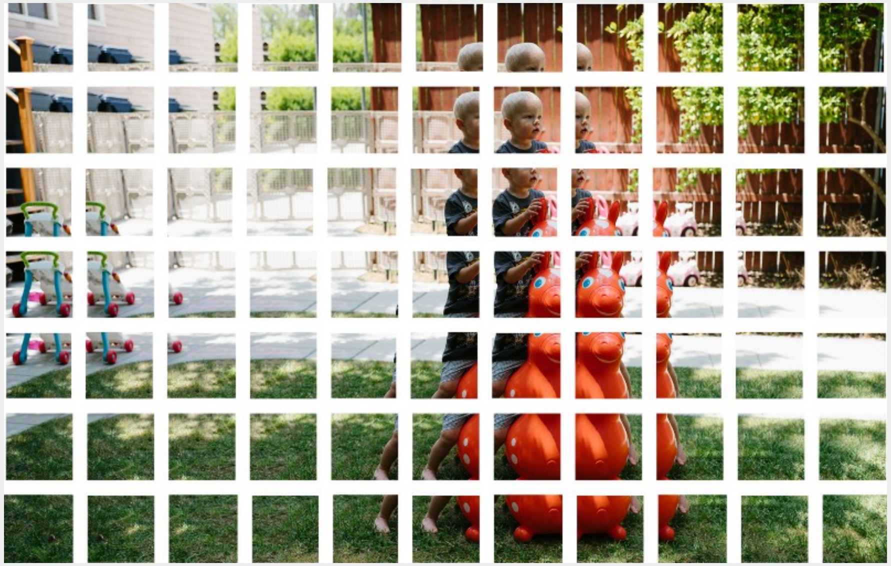


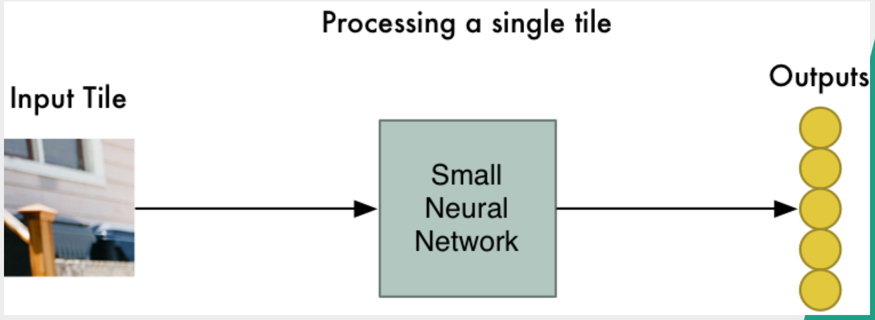
Why convolution?

- Humans recognize instantly that there is a child on the picture
- We recognize the idea of a child no matter what surface the child is on
- We need to give our neural network understanding of translation invariance—an “8” is an “8” no matter where in the picture it shows up

Idea behind convolution

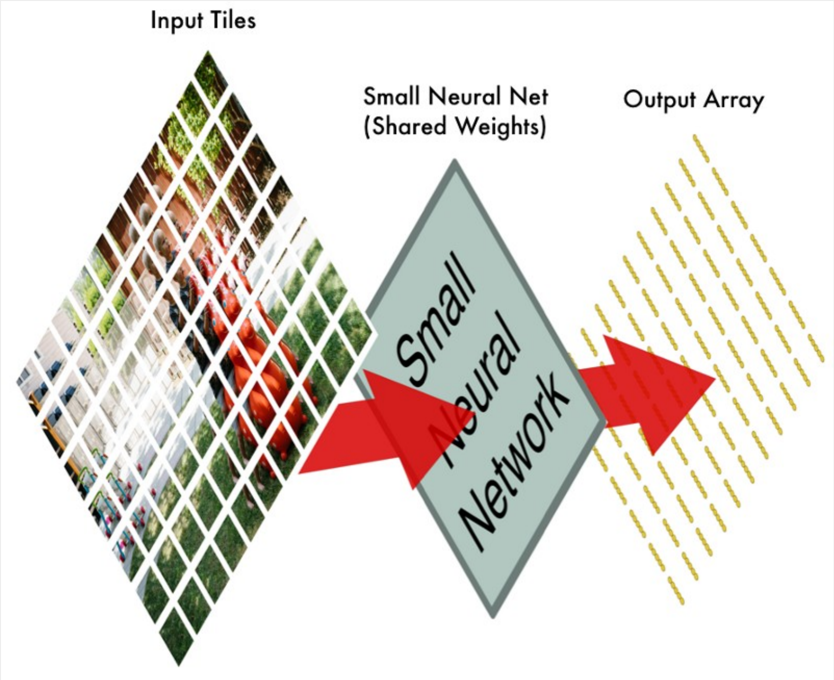
- Break the image into overlapping image tiles
- Feed each image into a small neural network
- The outputs of the small neural network can be trated as high-level features
  - e.g. is there a child's face in the tile?
- We'll keep the same neural network weights for every single tile
  - i.e., the weights are **shared**





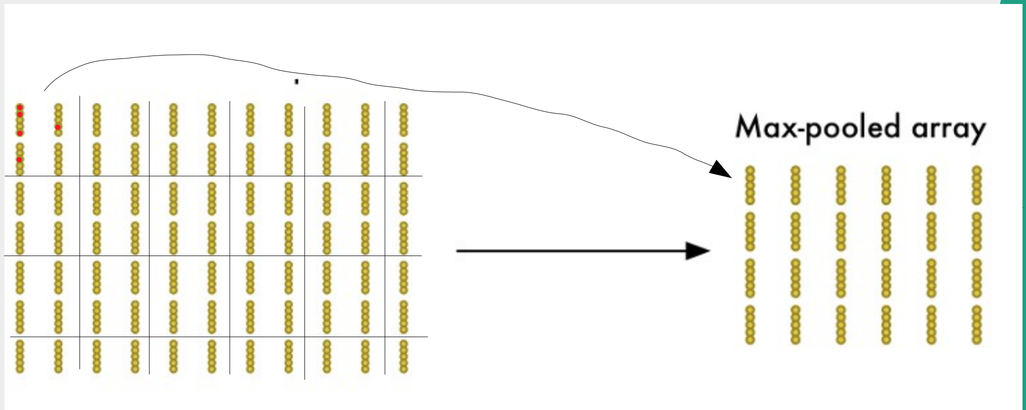
Idea behind convolution, cont.

- Save the outputs of the small neural network into a new array
- We've started with a large image and we ended with a slightly smaller array that records which sections of our original image were the most „interesting”



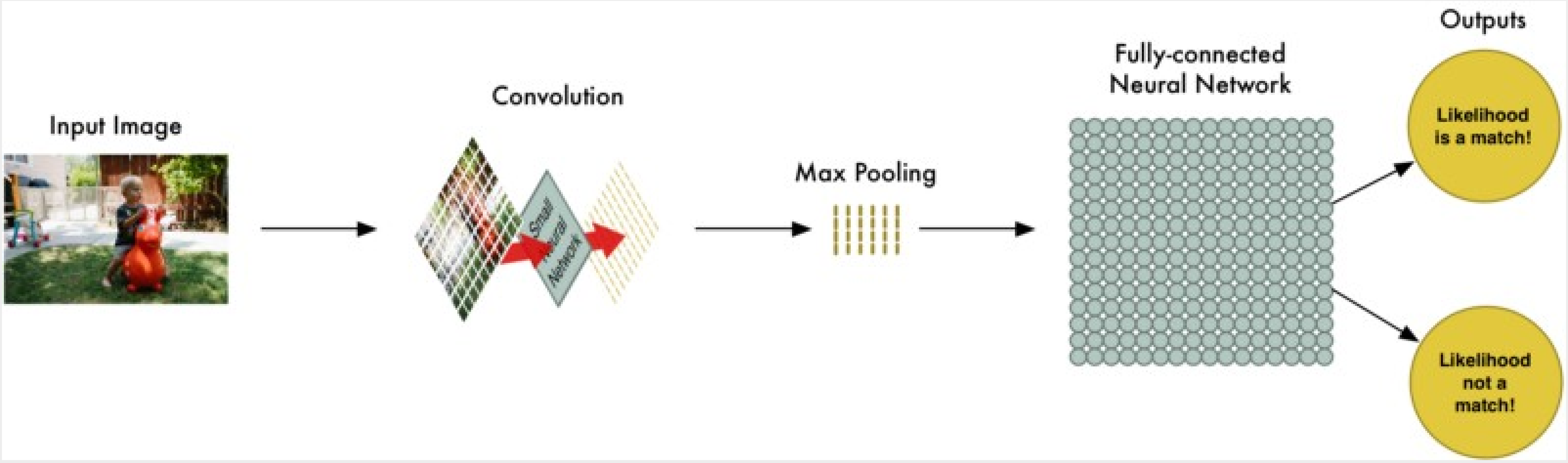
Downsampling

- The result from 1st convolution is still pretty big
- To reduce the size of the array, we downsample it using an algorithm called max pooling
- We'll just look at each 2x2 square of the array and keep the biggest number
- Or average pooling



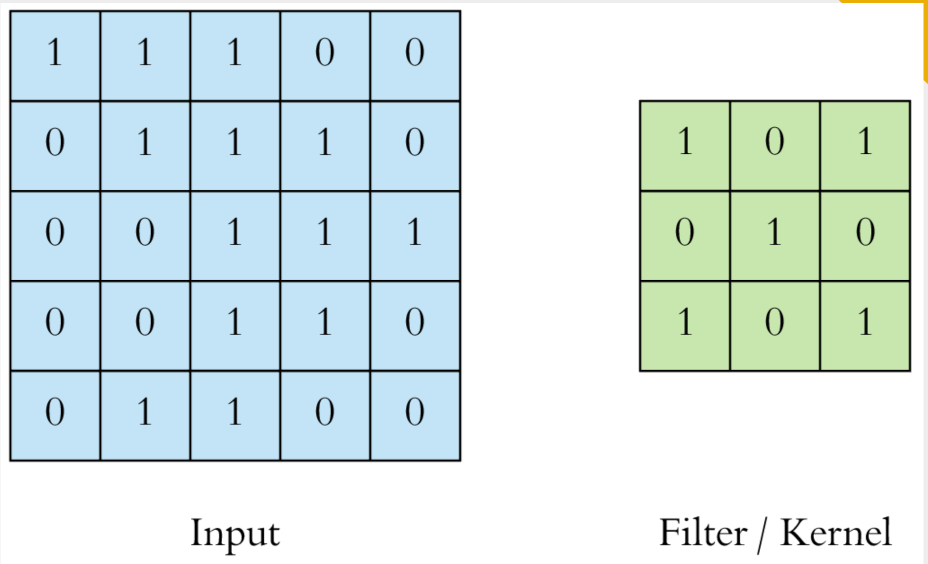
Finalizing the convolution

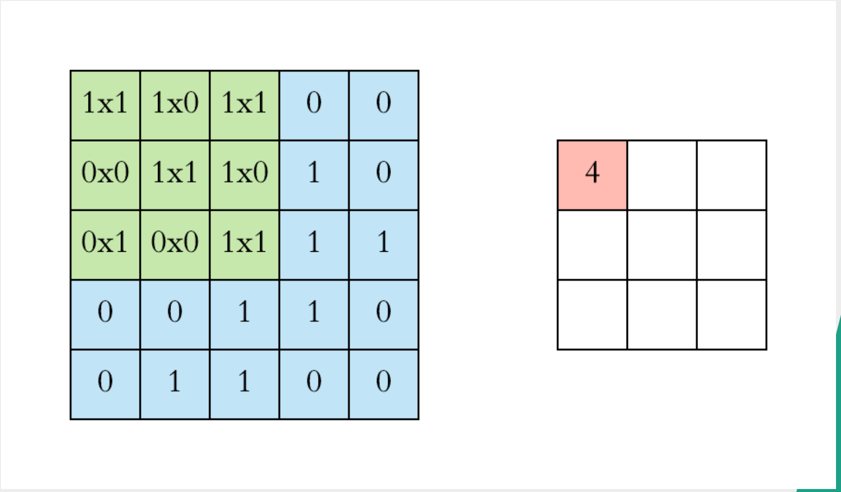
- So far, we've reduced a giant image down into a fairly small array
- That array is just a bunch of numbers, so we can use that small array as input into another neural network



Convolutional layer, again

- Convolution is a mathematical operation to merge two sets of information – input and filter (kernel)
- We perform the convolution operation by sliding this filter over the input
- At every location, we do element-wise matrix multiplication and sum the result
- The green area where the convolution operation takes place is called the **receptive field**





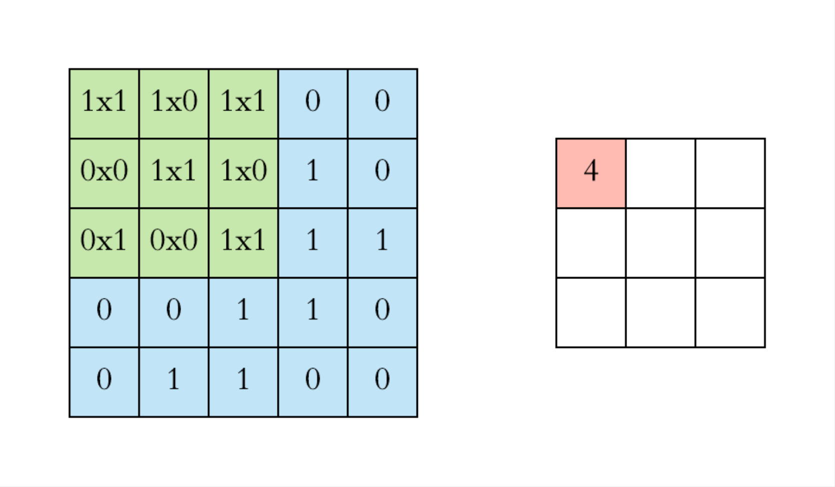
What can convolutions do?

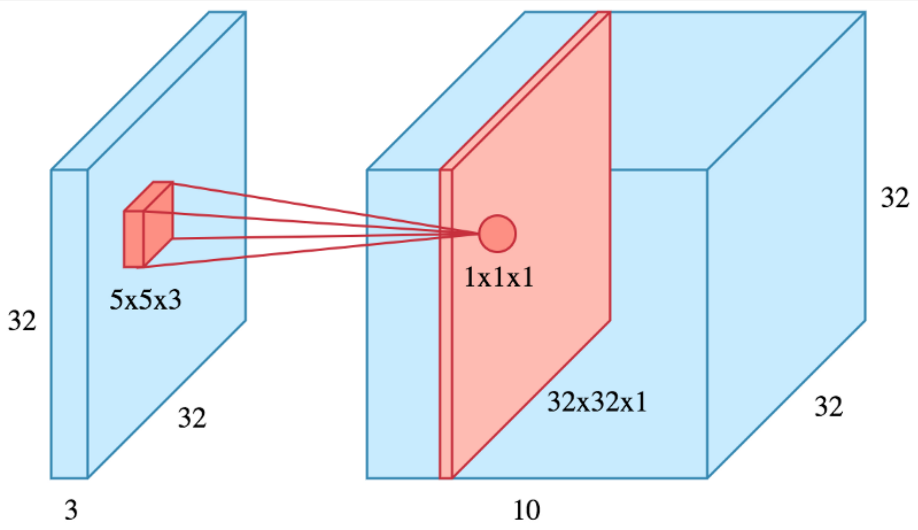
- Hand-designed convolutions can be used for:
  - Contour detection
  - Edge detection
  - Sharpening
  - Blurring

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Filters, cont.

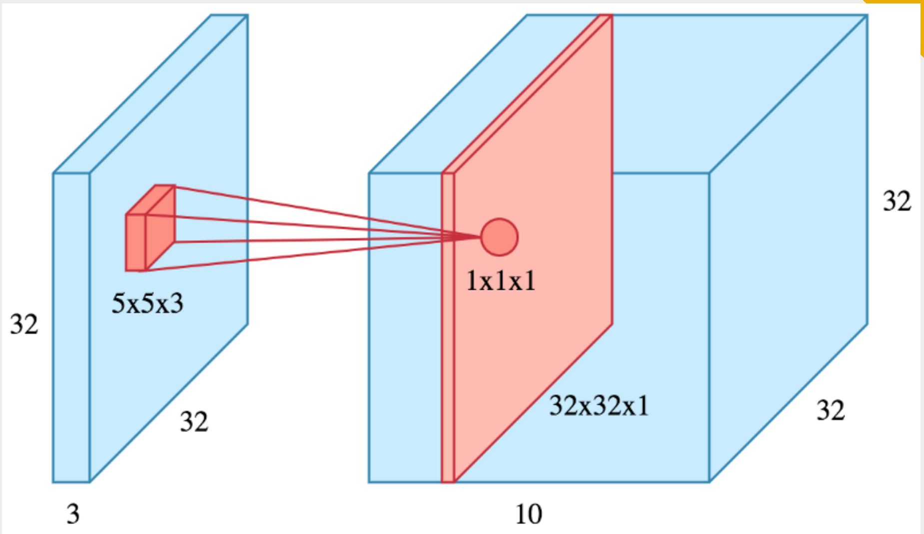
- The example shows convolution in 2D, using 3x3 filter
- In reality these convolutions are performed in 3D
  - Image is represented as a 3D matrix with dimensions of height, width and depth, where depth corresponds to color channels (RGB)
- A convolution filter covers the entire depth of its input so it needs to be 3D as well
- We perform multiple convolutions on an input, each using a different filter and resulting in a distinct feature map
- On the right: 10 filters

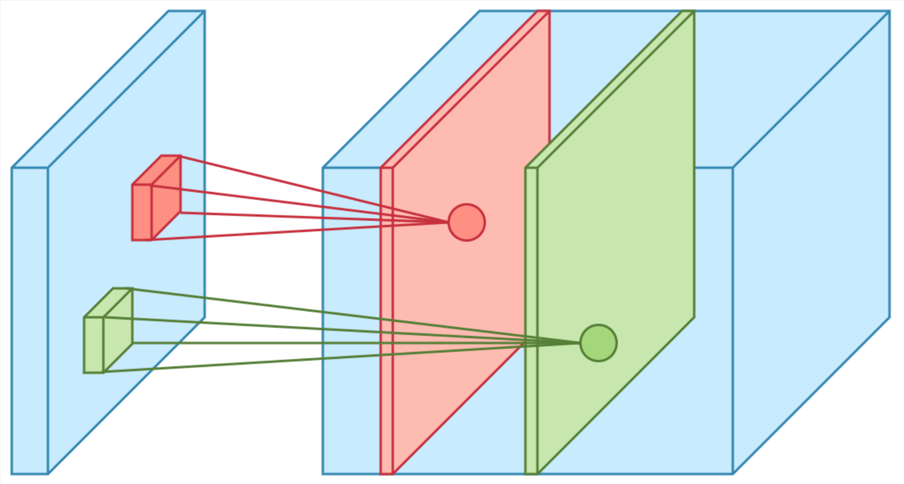




Input and output dimensions of convolution

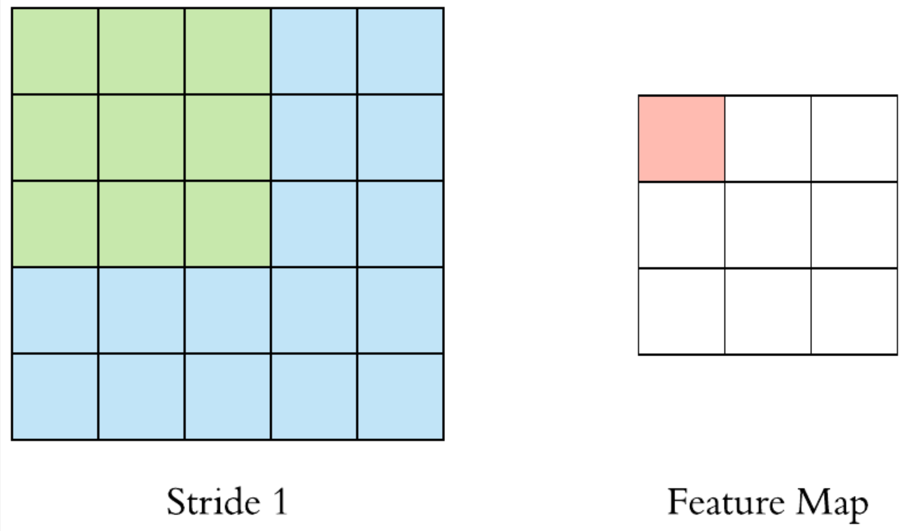
- Input: RGB image of 32x32 pixels, 3 channels
- Convolution: 5x5, 10 filters
- Output: 32x32x10 „volume”
- Output of the convolution is passed through a non-linearity (e.g. ReLU)

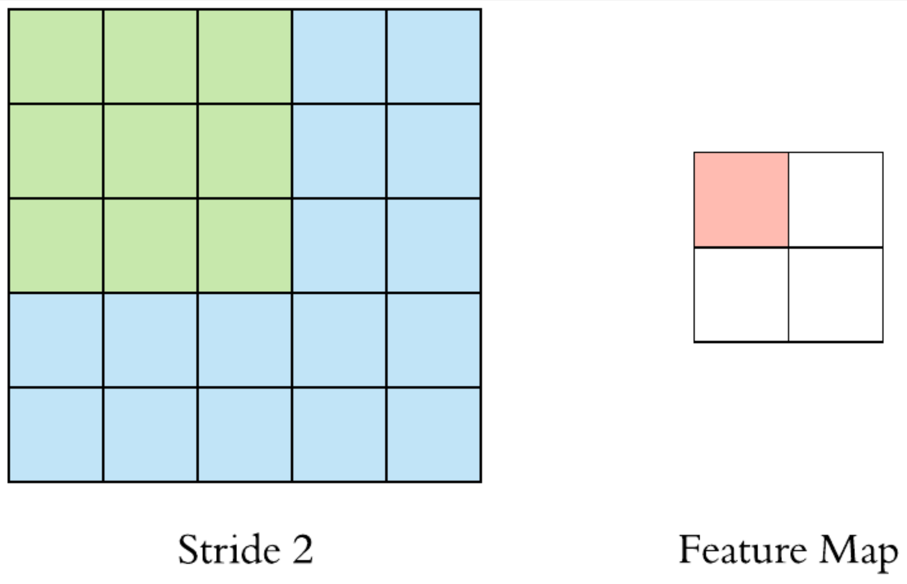




Stride

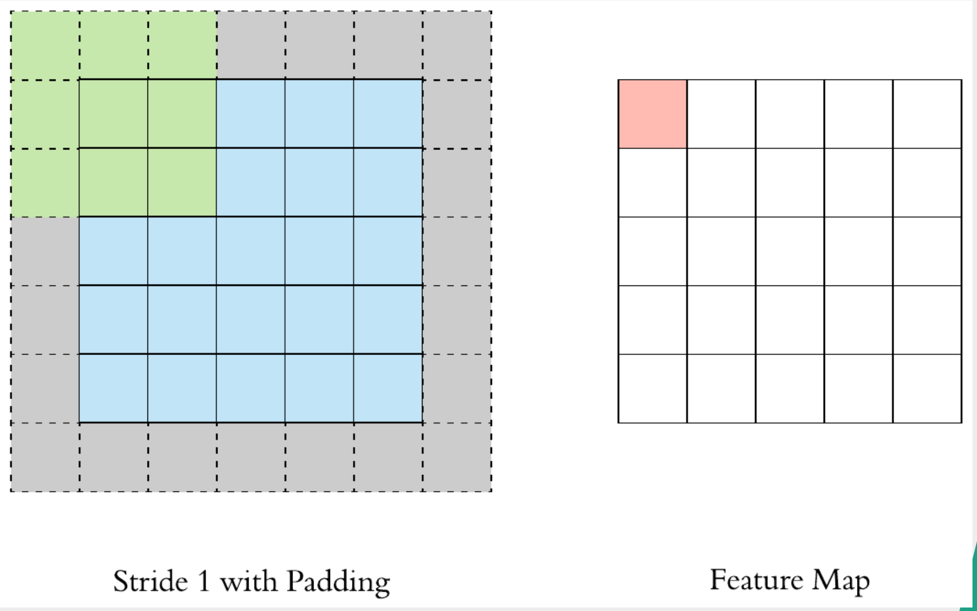
- **Stride** specifies how much we move the convolution filter at each step
- By default the value is 1
- We can have bigger strides if we want less overlap between the receptive fields.





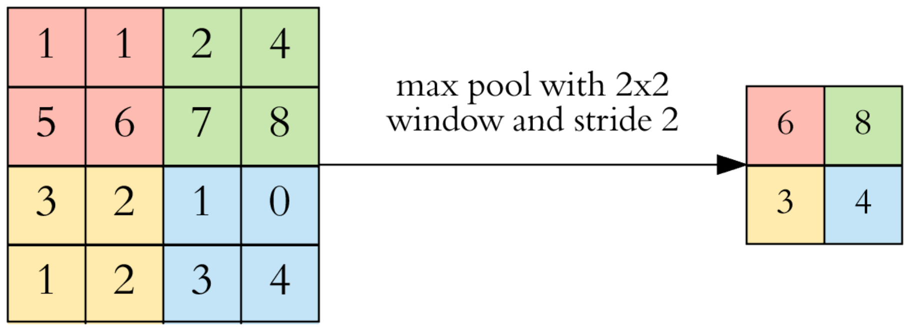
Padding

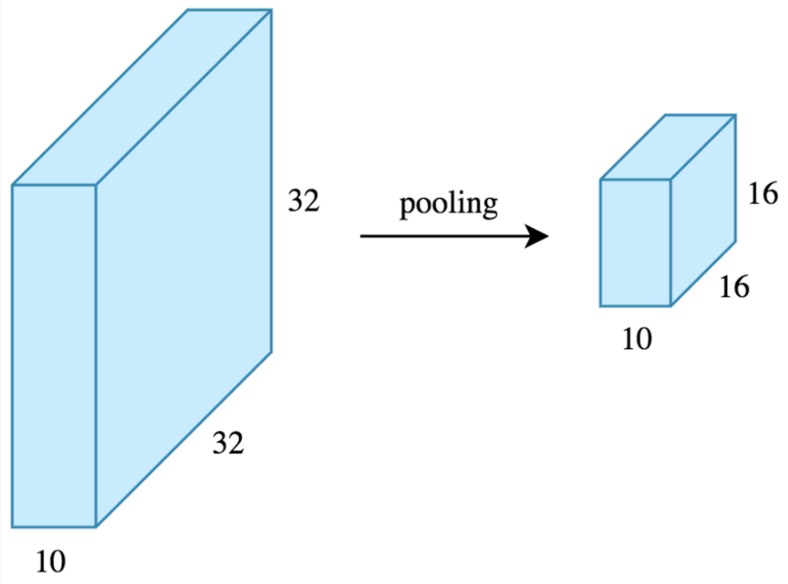
- The size of the output is smaller than the input, because the convolution filter needs to be contained in the input
- If we want to maintain the same dimensionality, we can use **padding** to surround the input with **zeros**
- Padding is commonly used in CNN to preserve the size of the feature maps
  - otherwise they would shrink at each layer



Pooling

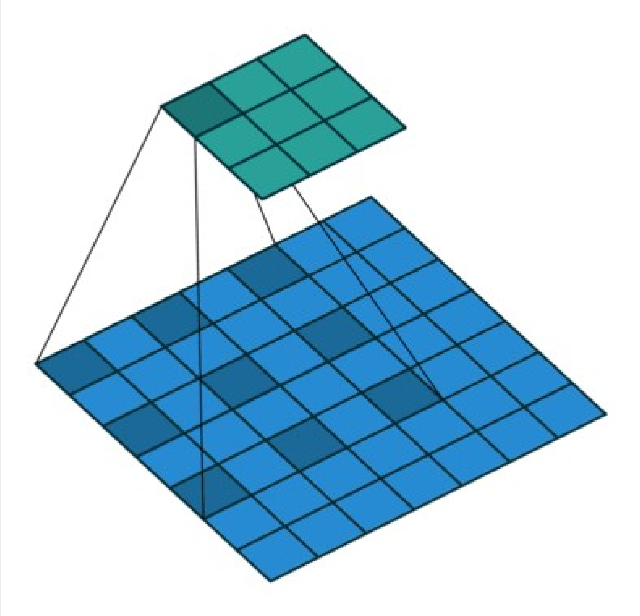
- After a convolution operation we often perform pooling to reduce the dimensionality
- This enables us to reduce the number of parameters, which both shortens the training time and combats overfitting
- Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact
- The most common type of pooling is max pooling which just takes the max value in the pooling window

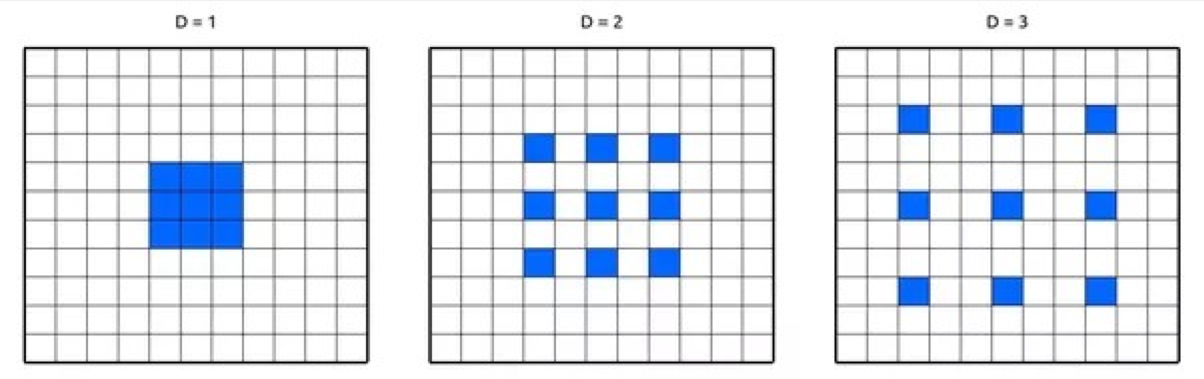




Dilated convolutions

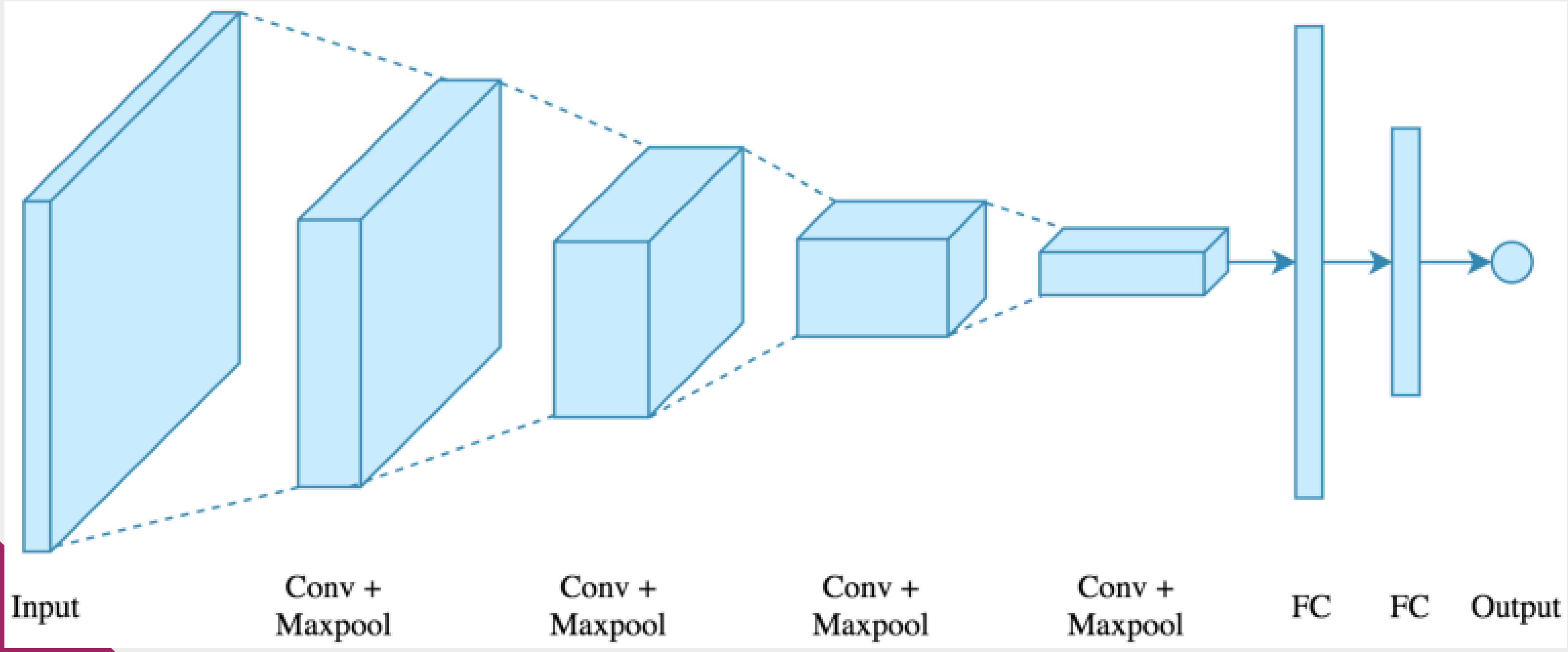
- Dilated convolutions have gaps in the kernels





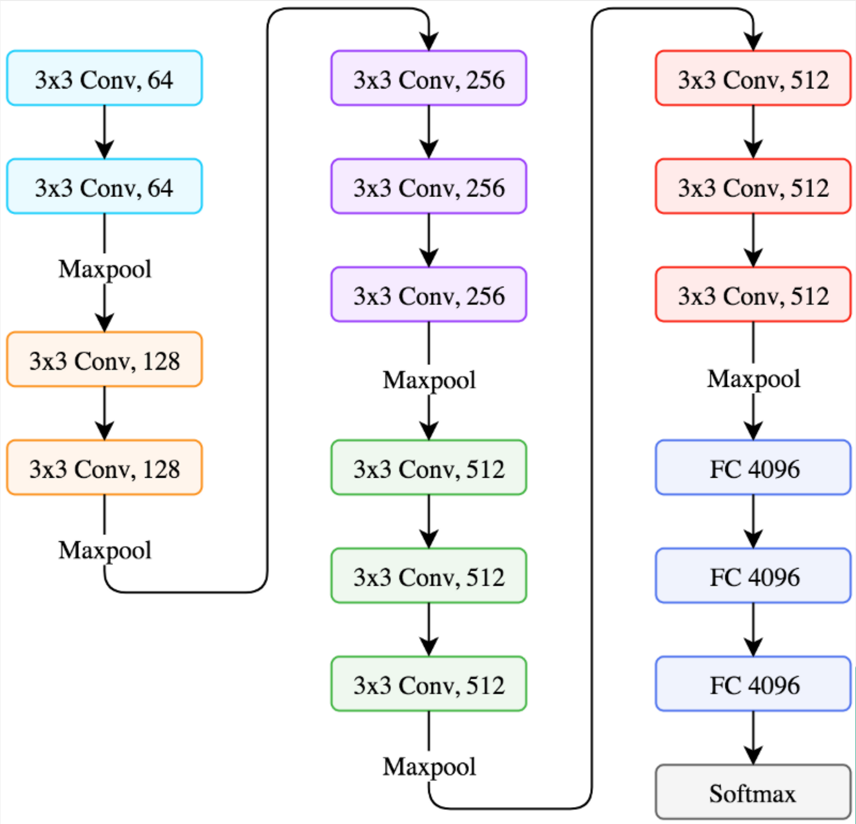
Many convolutional layers

- The output of a convolution layer can be also viewd as an image
  - Instead of 3 channels, the number of channels now equals the number of filters
- That means we can apply another convolution to it
- Typically, many conv+maxpool layers are appended to each other
  - The number of filters is typically increased as maxpooling reduces the width and height
- Followed by a dense (fully connected) layer and a softmax

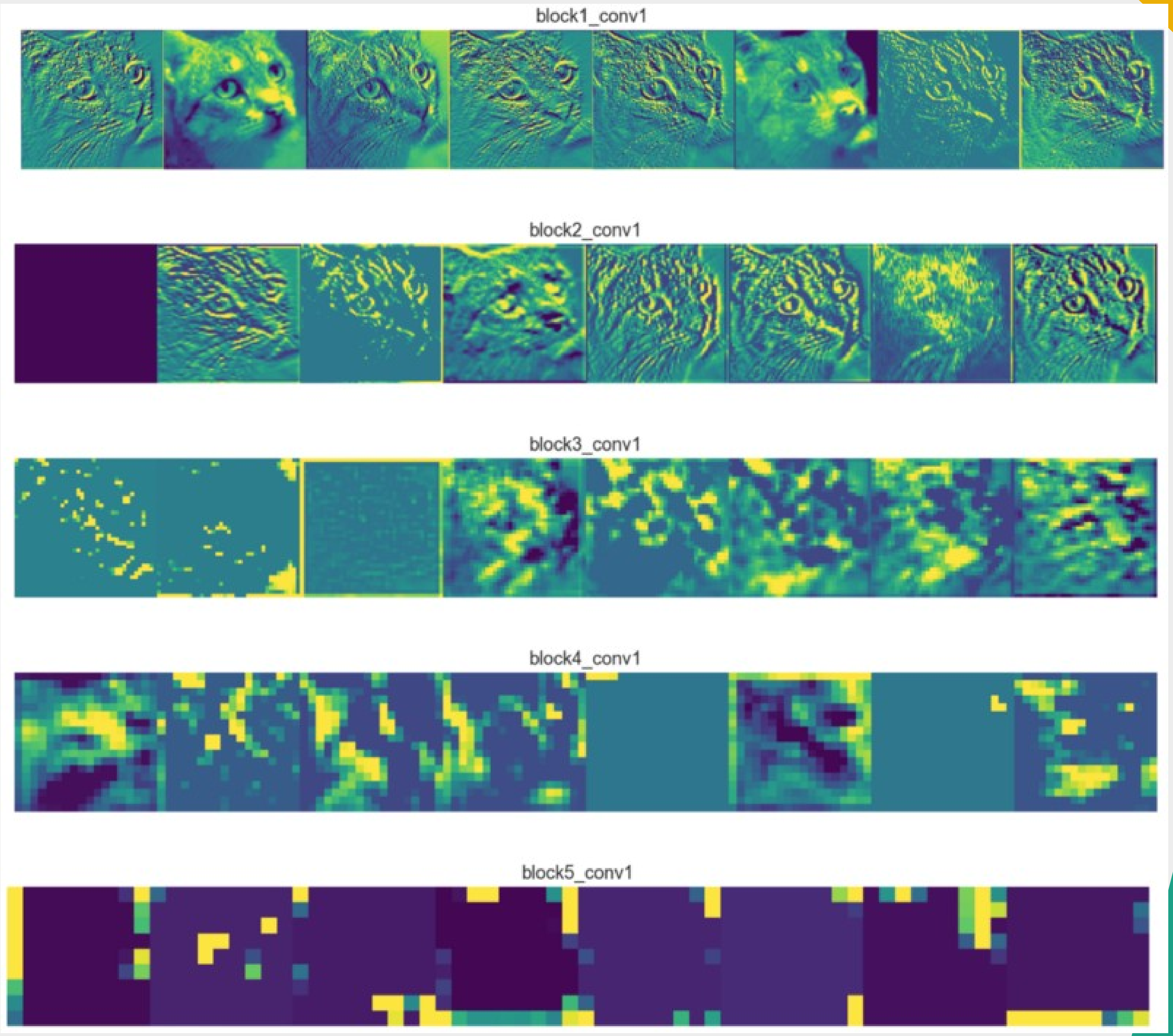


Convoutional neural network

- A CNN model can be thought as a combination of two components: feature extraction part and the classification part
  - The convolution + pooling layers perform feature extraction: detect features such as two eyes, long ears, four legs, a short tail
  - The fully connected layers then act as a classifier on top of these features, and assign a probability for the input image being a dog.

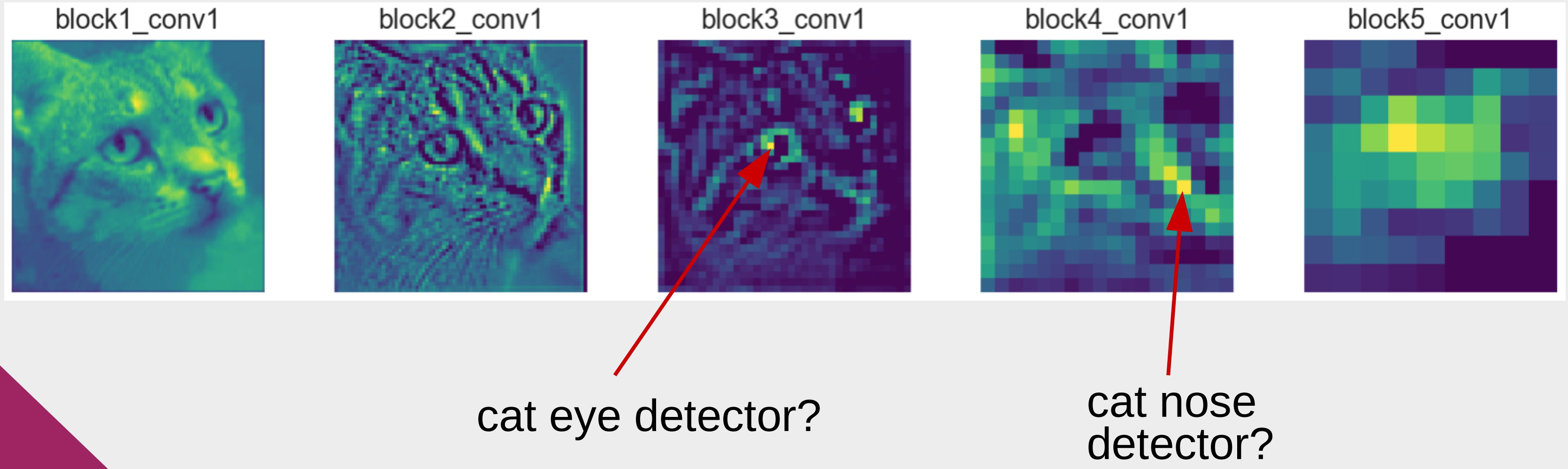


But what do convolutional layers really do?



Interpretation of convolutions

- The images below show **only single (hand-picked) channel** of each convolutional layer
  - In reality, there are several of such „feature maps”, and more as we go deeper
- First layers usually act as edge detectors
- As we go deeper into the network, the feature maps look less like the original image and more like an abstract representation of it
- Deeper feature maps encode high level concepts like “cat nose” or “dog ear”
- Deeper feature maps are also more granular (lower resolution) but there are **more of them** (because we usually increase the number of filters as a reduce the resolution)

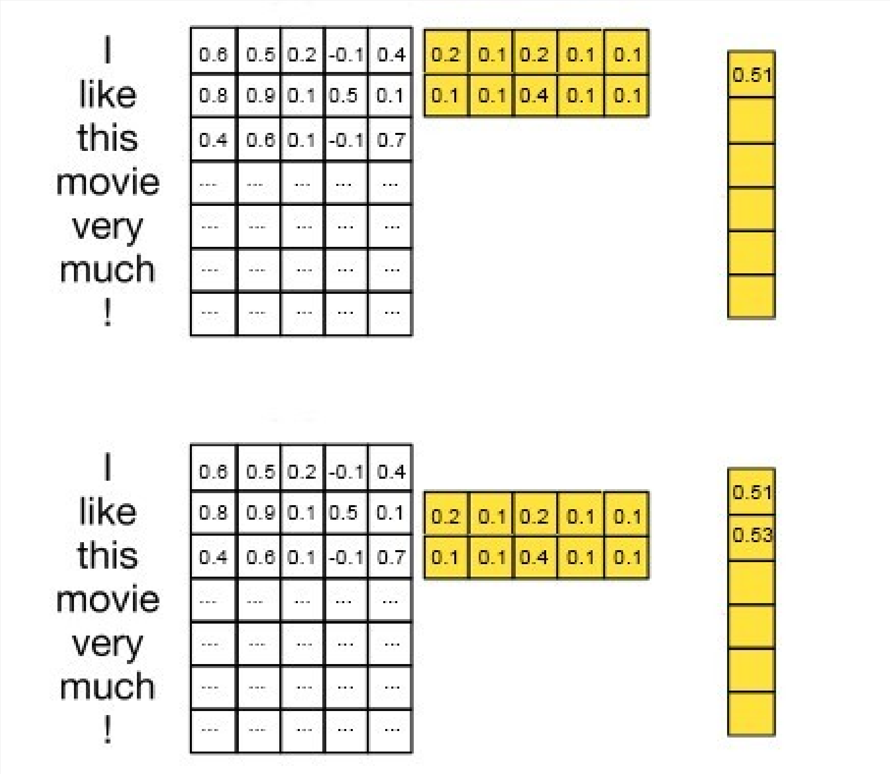


Demo

- Try: http://scs.ryerson.ca/~aharley/vis/conv/flat.html
-

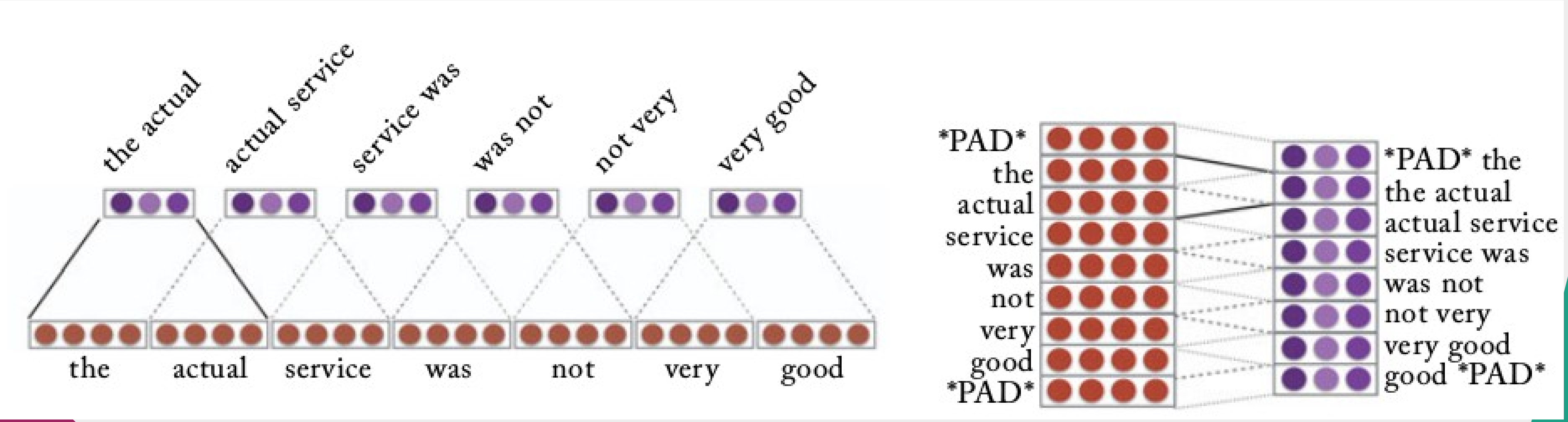
But what does this have to do with NLP?

- If we replace words in a sentence with word embeddings, the result can be viewed as 2D image
  - height=1, width=N, channels=embedding dim
- We can apply convolutions to the 2D text representation



Convolution applied to text

- Convolutions applied to word embeddings extract features of word n-grams
- Each convolution operation only „sees” a limited word window
- E.g., a convolution with width=2, and #kernels=3, applied to 4-dimensional embeddings

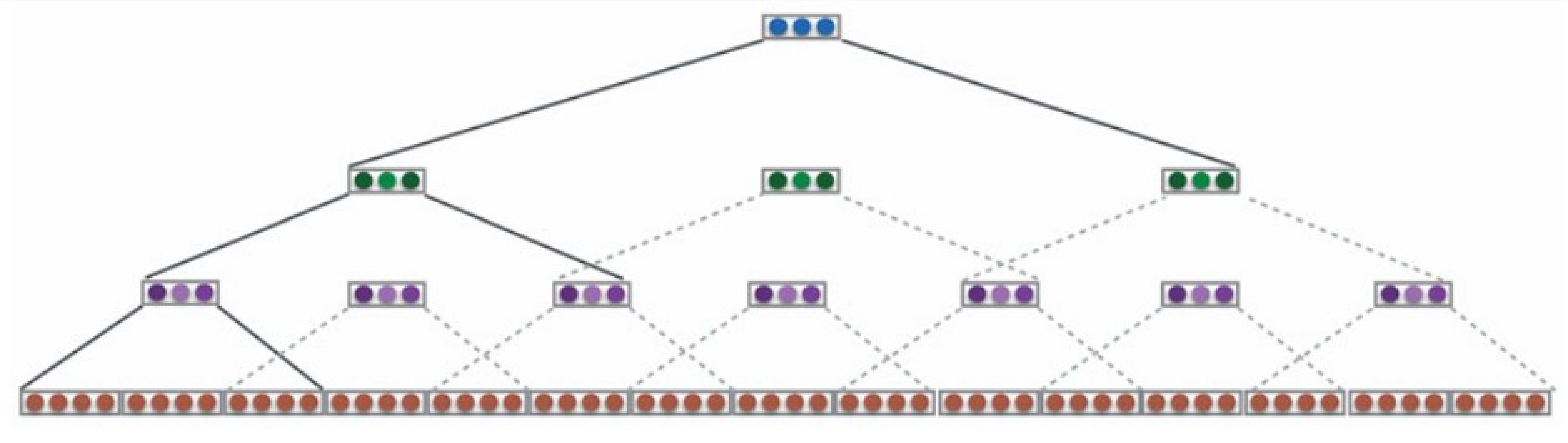


Why should this make sense?

- Remember: the individual dimensions of word embeddings **encode certain aspects of a word**, e.g.
  - Is it a plural noun?
  - Is it an adjective with positive sentiment (e.g. dimension 99)
  - Is it a word marking negation, such as *not*, *none* (e.g., dimension 67)
- A convolutional filter applied to word embeddings can extract features from word combinations that are important for the task
  - E.g., a filter of width 3 with all zeros, except with 1 at [0, 67] and [2, 99] activates (produces large output) for expressions such as „***not*** *very* ***good***", „***not*** *really* ***enjoyable***"

Hierarchical convolutions

- Like with images, convolutions can be applied hierarchically to text
- Deeper convolutional layers use features extracted by lower layers
- Deeper layers see a wider window of words
- Below: three convolutional layers, each with width=3 and stride=2
- Result: the 3rd convolution „sees” 15 words



Visualizing hierarchical feature detectors for text

- Understanding what the learned high-level features have learnt is difficult, especially for text
- But we can go through our data and select the text segments that cause the highest activation for a certain high-level feature
- Example, for a sentiment classification task (for a filter with a receptive field of 7 words):

			POSITIVE			
lovely	comedic	moments	and	several	fine	performances
good	script	,	good	dialogue	,	funny
sustains	throughout	is	daring	,	inventive	and
well	written	,	nicely	acted	and	beautifully
remarkably	solid	and	subtly	satirical	tour	de
			NEGATIVE			
,	nonexistent	plot	and	pretentious	visual	style
it	fails	the	most	basic	test	as
so	stupid	,	so	ill	conceived	,
,	too	dull	and	pretentious	to	be
hood	rats	butt	their	ugly	heads	in

Dealing with varying text lengths

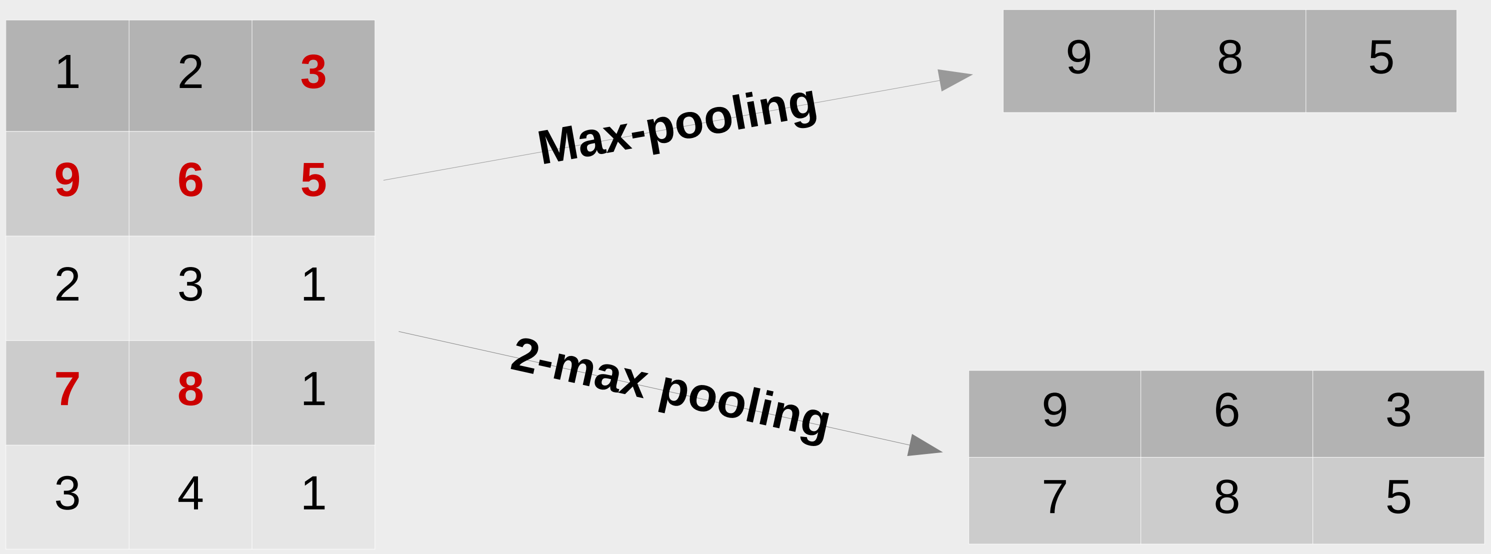
- Texts (sentences, documents) to be classified have different lengths
- How to reduce the features of all documents to the same length?
- For texts that are **shorter** than the receptive field: pad with zero vectors
- For texts that are **longer** than the receptive field: crop the document (e.g., delete the ending), or use **pooling**
  - Max-pooling: we take the maximum value of each high-level feature over the document
  - Average pooling: just average the feature values over the document

Dynamic pooling

- Max-pooling doesn’t retain positional information
- Sometimes positional information can be important
- Solution: dynamic pooling
- E.g., for document topic classification:
  - Apply one max-pooing pooling for the title
  - Another max-pooing for the 1st paragraph
  - Another for the rest of the document

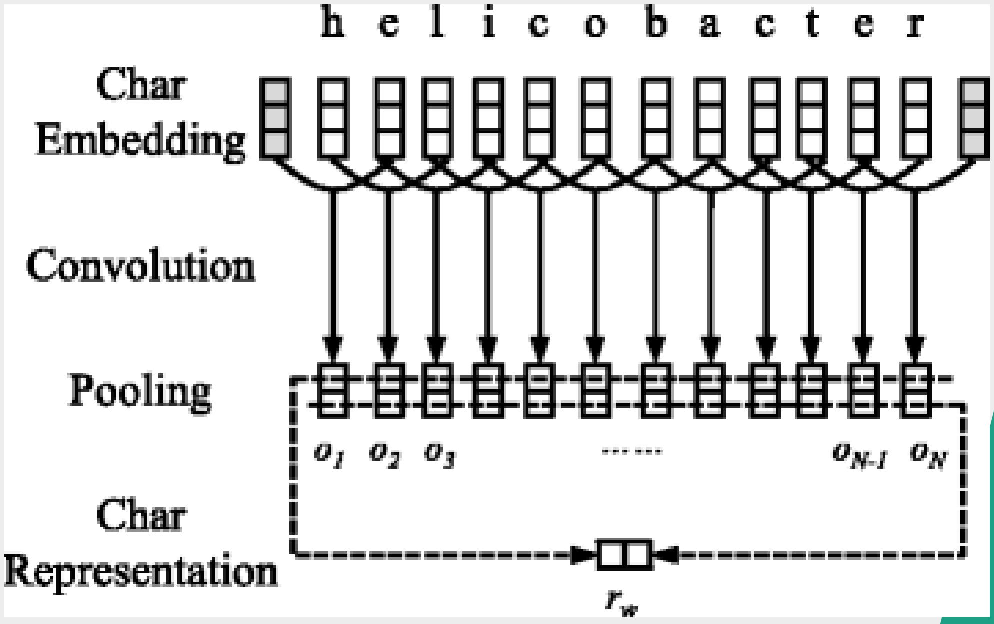
k-max pooling

- Another variation of global pooling (useful for reducing vector sequences of varying length to a single dimension)
- **k-max pooling**: retain top k values of each dimension, and preserve their ordering
- Preserves the order of the features but is insensitive to their specific positions
- Can give information about how many times a feature is highly activated



Not only words

- We can also use convolutions for character sequences
- The resulting features can be used for example for isolated word classification
- Or, use the learned representation as an additional knowledge source (in addition to word embeddings) in text classification
  - Can be useful for highly inflective languages (like Estonian)



- Questions?