AIML428

- Two text representations
    - TF-IDF
    - Word embeddings
- What are the differences?
- How do you use word embedding in KNN, NB, or SVM?

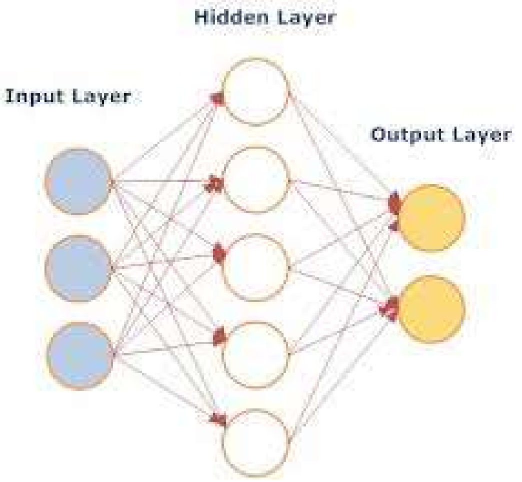
Typical text classification

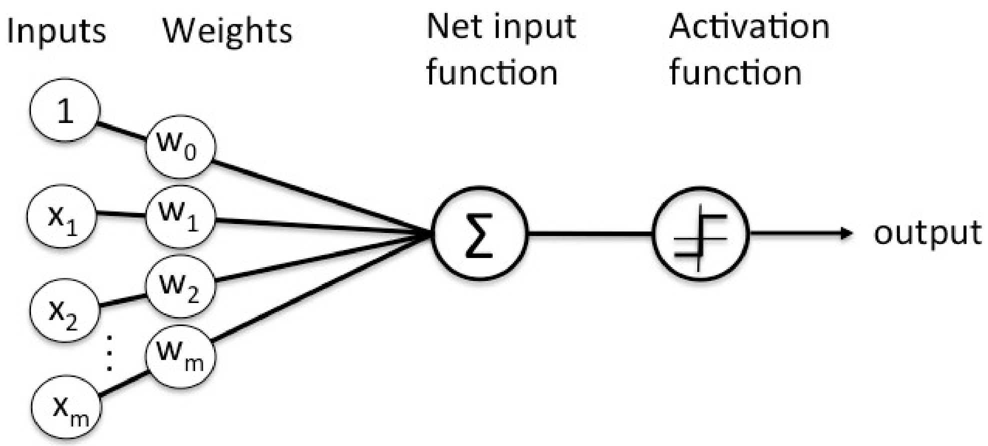
- Text representation: Word embedding

- Classification algorithm: CNN
    - CNN for image processing
    - CNN for text classification
        - A toy example

Quick Recap: Neural Networks

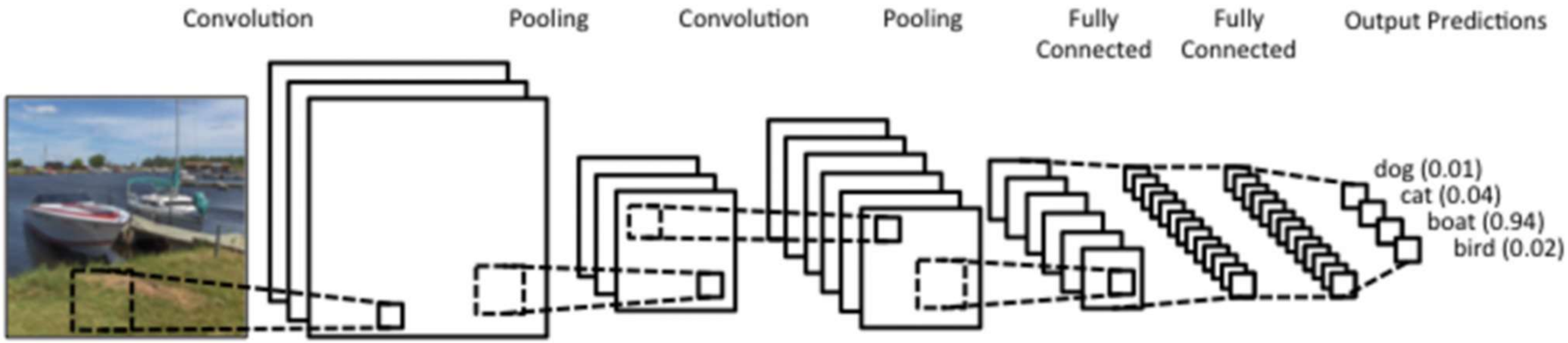
- Made up of neurons that have learnable weights and biases, *W, b*
- Each neuron receives some inputs:
    - performs a dot product *W.X + b*
    - apply an activation function *f(WX+b)* for output e.g. softmax applied on the last layer





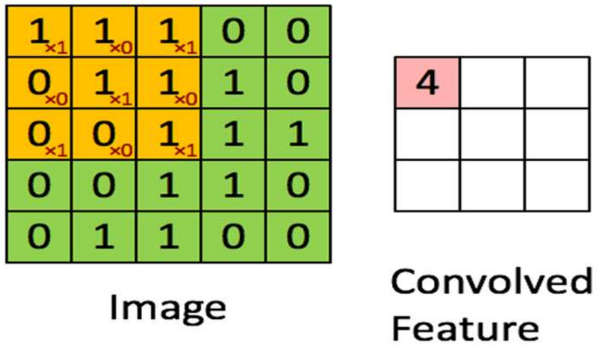
CNN architecture

- There are 3 main types of layers to build CNN
  - Convolutional Layer
  - Pooling Layer
  - Fully Connected Layer



What is a convolutional layer?

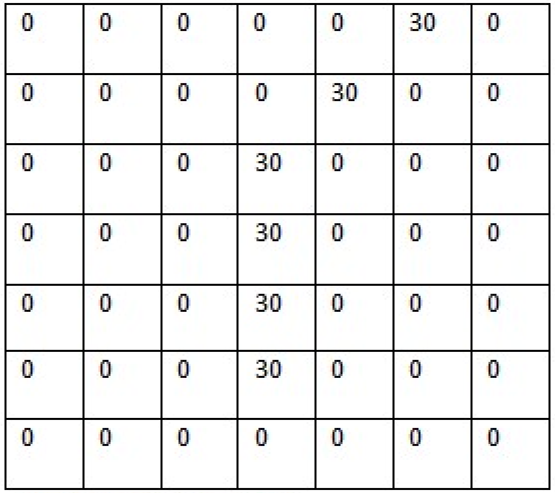
- Sliding window function applied to a matrix
  - Matrix on the left represents a black and white image
  - There is a 3x3 filter/kernel, with weights/values
- We multiple its values with the original matrix and take the sum
- A full convolution is done for each element by sliding the filter over the whole matrix
- Visualisation at https://stats.stackexchange.com/questions/296679/what-does-kernel-size-mean



1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

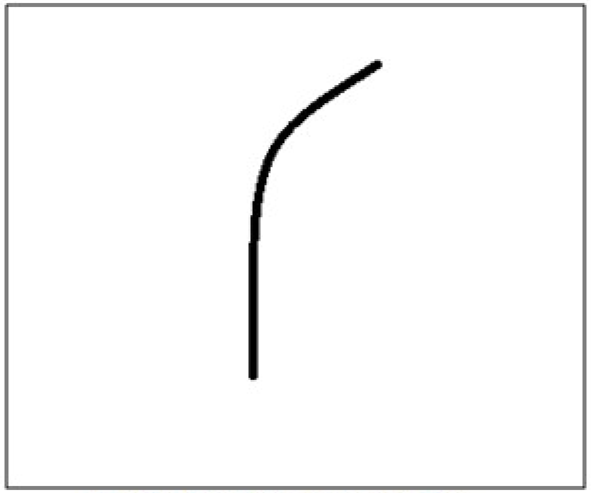
4		

A filter can be a curve detector

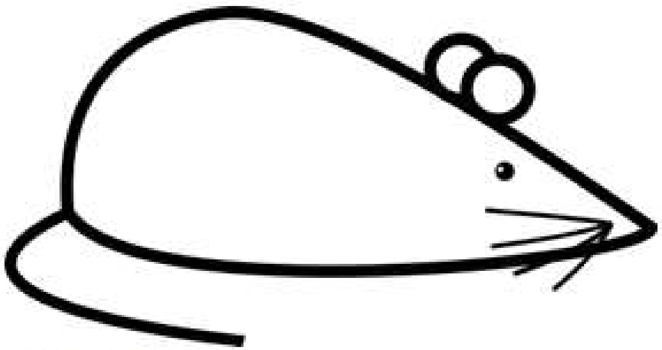


0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

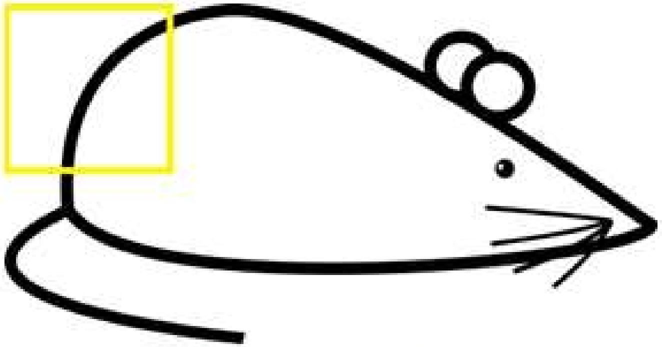
Pixel representation of filter



Visualization of a curve detector filter



Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

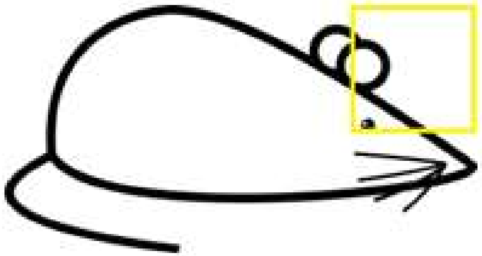
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Move to a different area



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

$*$

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

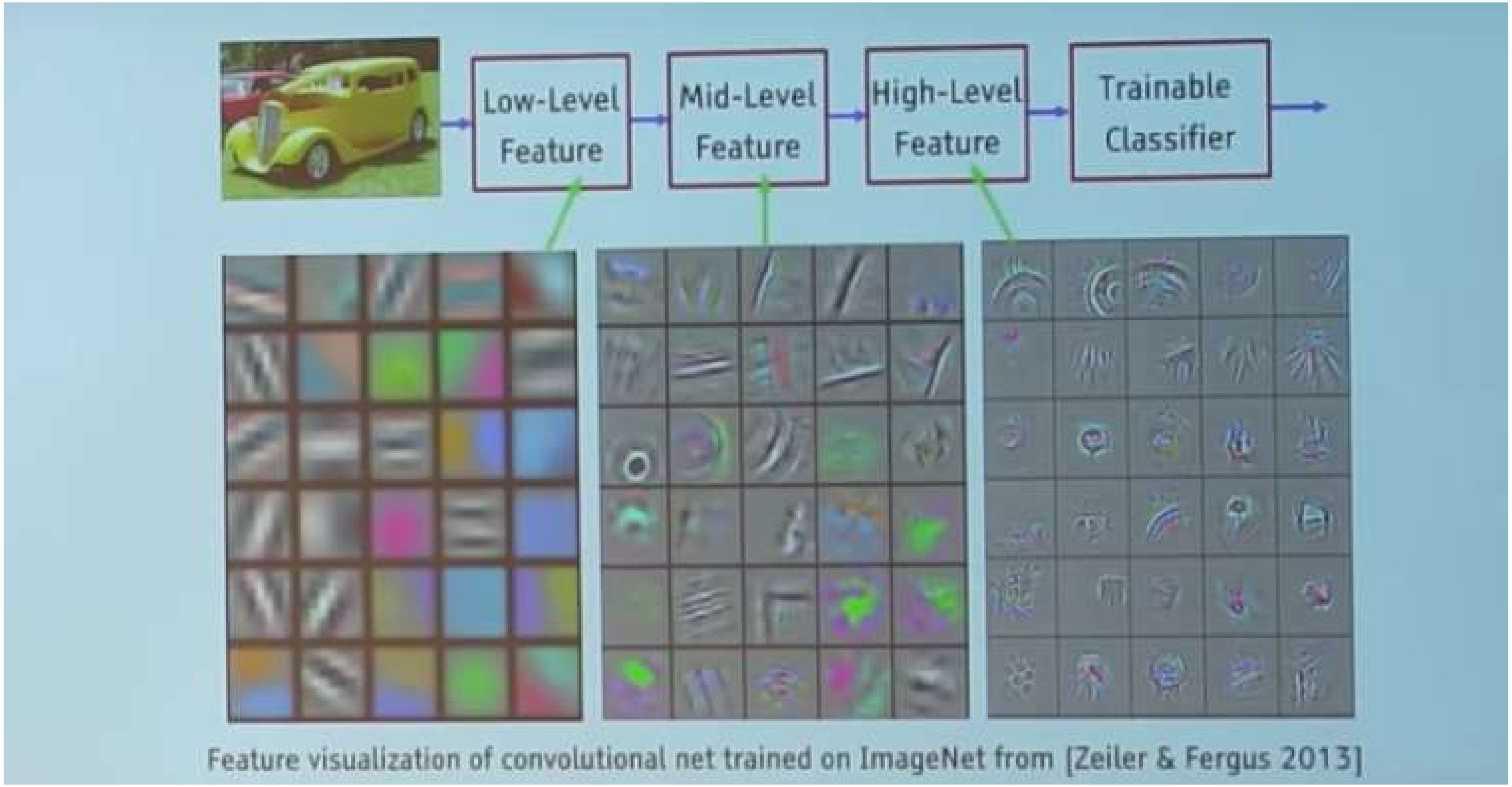
Pixel representation of filter

Multiplication and Summation = 0

Image Processing Detection

1. A CNN may learn to detect edges from raw pixels in the first layer
2. Use the edges to detect simple shapes in the second layer
3. Use these shapes to deter higher level features such as facial shapes
4. The last layer is a classifier that uses these high-level features

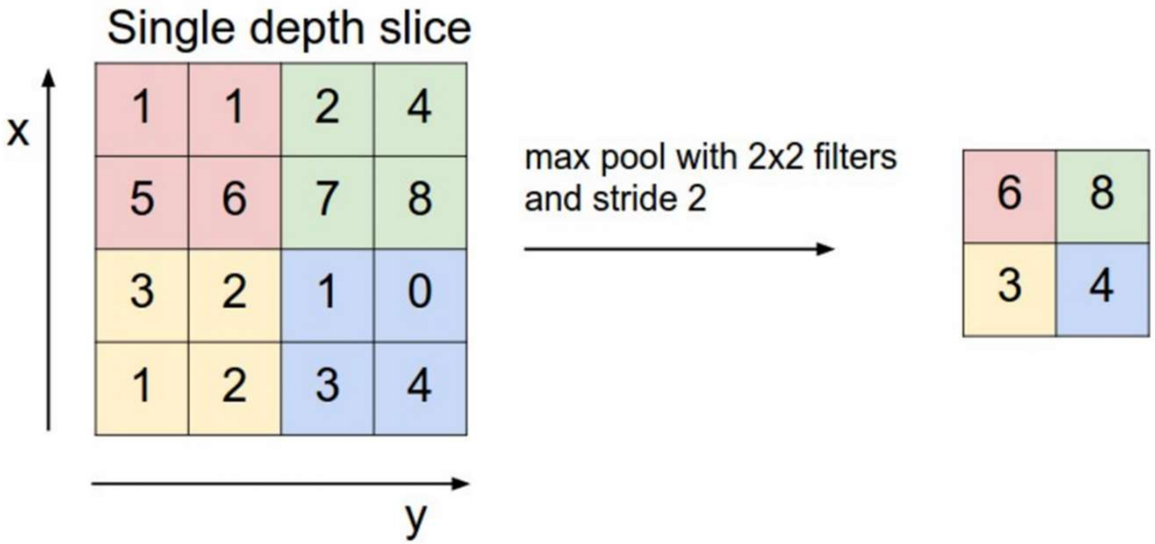
Feature visualisation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Pooling Layer

- Typically applied after the convolutional layer(s), the pooling layers subsample their input. Often, a max operation is applied
- Provides a fixed size output matrix
- You can use variable size sentences, variable size filters but always get the same output dimensions to feed into a classifier



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Understanding CNN for text classificatio

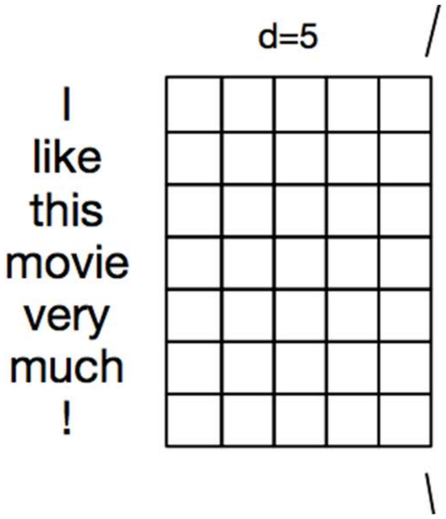
- http://www.joshuakim.io/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/

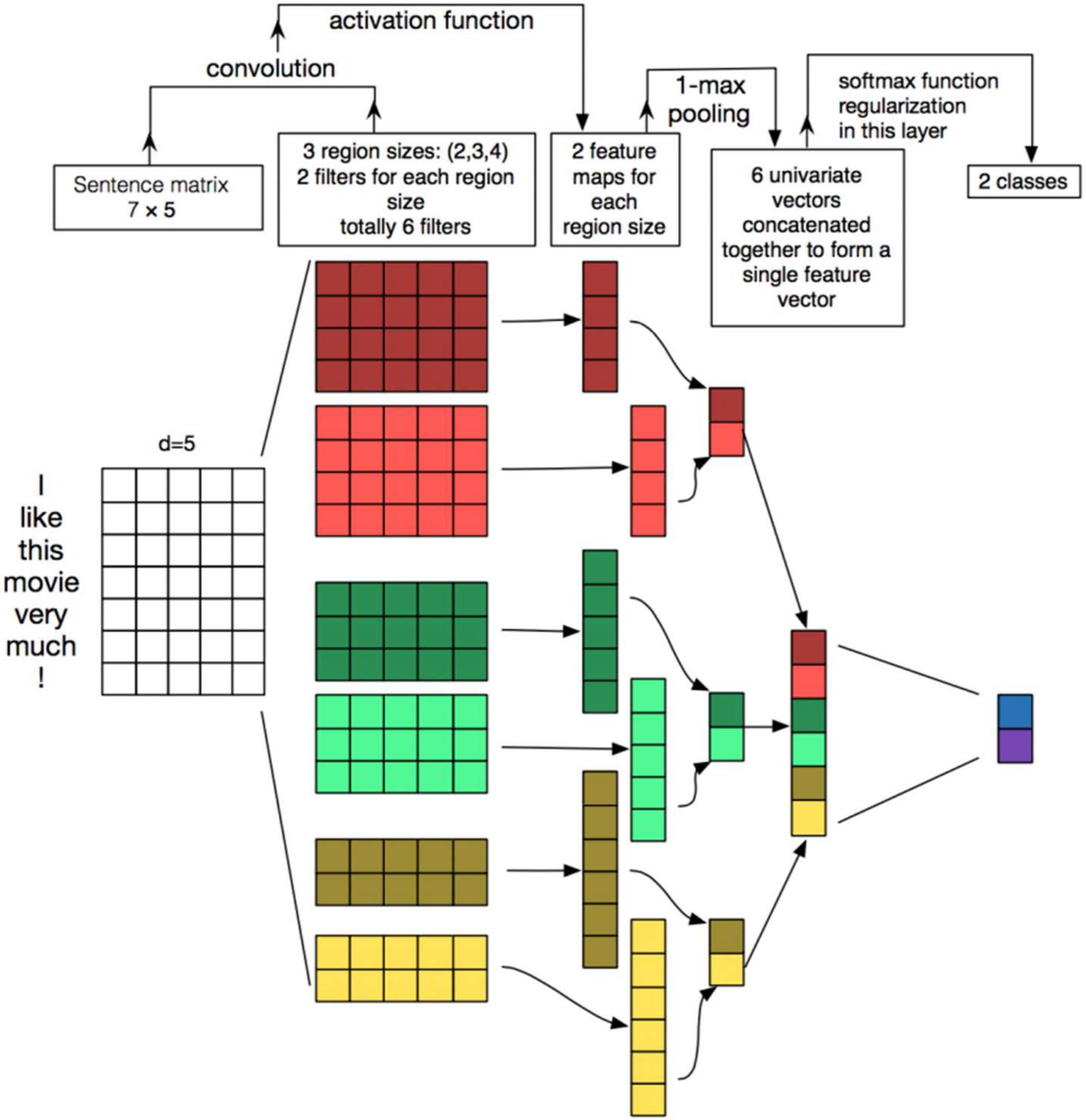
- Original paper: Zhang Y, Wallace B. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:151003820. 2015; PMID: 463165

- A toy example: a one-layer CNN on a 7-word sentence, with word embeddings of dimension 5

Natural Language Processing (NLP)

- The input to most NLP tasks are sentences or documents represented as a matrix.
- Each row of the matrix corresponds to one token. I.e. a word, a character or a group of words
- This can be done with some sort of transformation word2vec/GloVe, one hot vectors that index the word into a vocabulary





activation function

convolution

Sentence matrix
7 × 5

3 region sizes: (2,3,4)
2 filters for each region size
totally 6 filters

2 feature maps for each region size

1-max pooling

6 univariate vectors concatenated together to form a single feature vector

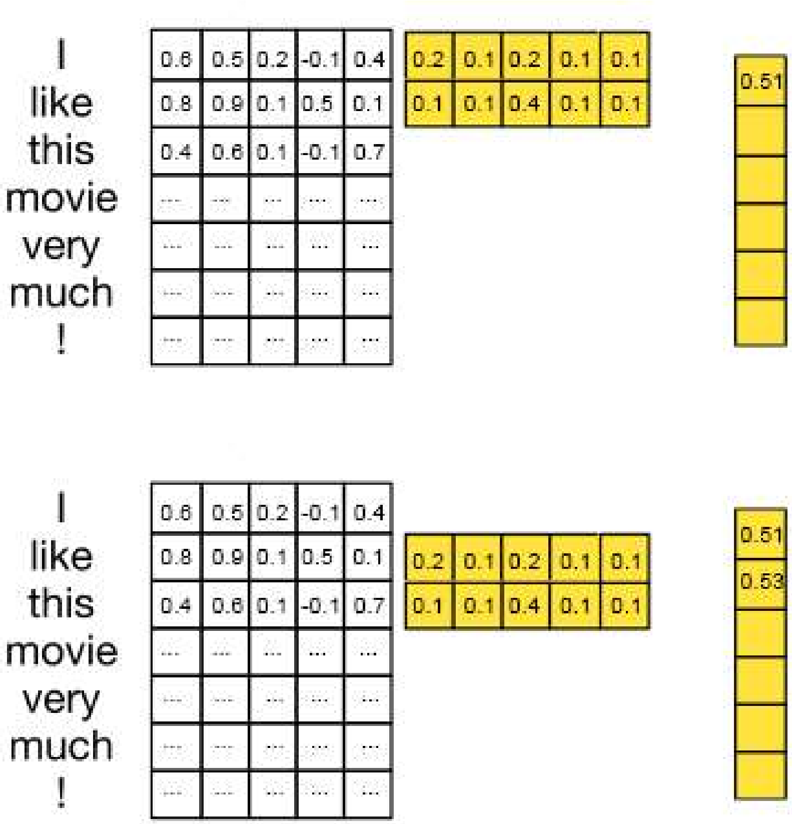
softmax function
regularization
in this layer

2 classes

d=5

I
like
this
movie
very
much
!

Convolution/ Filter



I	0.6	0.5	0.2	-0.1	0.4
like	0.8	0.9	0.1	0.5	0.1
this	0.4	0.6	0.1	-0.1	0.7
movie	...	...	...	...	...
very	...	...	...	...	...
much	...	...	...	...	...
!	...	...	...	...	...

0.2	0.1	0.2	0.1	0.1
0.1	0.1	0.4	0.1	0.1

0.51

I	0.6	0.5	0.2	-0.1	0.4
like	0.8	0.9	0.1	0.5	0.1
this	0.4	0.6	0.1	-0.1	0.7
movie	...	...	...	...	...
very	...	...	...	...	...
much	...	...	...	...	...
!	...	...	...	...	...

0.2	0.1	0.2	0.1	0.1
0.1	0.1	0.4	0.1	0.1

0.51
0.53

- it performs an element-wise product for all its 2 x 5 elements, and then sum them up and obtain one number (0.6 x 0.2 + 0.5 x 0.1 + … + 0.1 x 0.1 = 0.51).

Learning process

- the two-word filter, represented by the 2 x 5 yellow matrix *w*
- The 2-word region filter window moves down each row, do a element-wise dot product and then sum to get one value, 6 values produce *o* (1D, in yellow)
- To obtain the feature map, *c*, we add a bias term (a scalar, i.e., shape 1×1) and apply an activation function (e.g. ReLU)
- What to learn
  - The *w* matrices that produced *o*
  - The bias term that is added to *o* to produce *c*
  - Word vectors (optional, use validation performance to decide)

Discussion

- What is each filter looking for?
- What if we have multiple documents
  - Different words
  - Different length
- What if the word is not in the list