Deep Learning for NLP
Convolutional Neural Networks

Nils Reimers

TECHNISCHE UNIVERSITÄT DARMSTADT

Course-Website: www.deeplearning4nlp.com

UKP

Recommended Readings

TECHNISCHE UNIVERSITÄT DARMSTADT

- https://www.youtube.com/watch?v=EevTPpQvxiU
- Kim, 2014, *Convolutional Neural Networks for Sentence Classification*
- http://cs231n.github.io/convolutional-networks/

UKP

Convolutional Neural Network

- Universal architecture achieving state-of-the-art performance

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**	–	–

Kim, 2014. Performance on Sentence Classification Tasks

Some notes on Conv. Neural Networks

- Convolutional Neural Networks are dominating computer vision
- For NLP, they became popular in ~2014
- Understanding the notation is quite difficult in computer vision
  - Images are typically 3 dimensional (width x height x color)
  - In NLP, we mainly deal with 1 dimensional data (our sentence / document)
  - Notation for 1 dimensional data is much simpler

Convolutional Neural Networks solve 2 crucial Challenges

TECHNISCHE UNIVERSITÄT DARMSTADT

- **First Challenge:**
  - In a lot of cases, we have variable sized input data, e.g. length of sentence / length of document.
  - Our network / our hidden layers are of fixed sizes
  - Solution 1: Window Approach (Collobert et al.), but we don't capture information outside of the window
  - Solution 2: Recursive & Recurrent Neural Networks

- **Second Challenge:**
  - Increasing the window in the window approach allows us to capture more context information, but increases dramatically the number of parameters
  - Often the position in the context window is of minor importance:
    - Jim $[\text{sells}]_{\text{Pred}}$ his car for $[\$5{,}000]_{???}$
    - Jim $[\text{sells}]_{\text{Pred}}$ his car, which he inherited from his dad, for $[\$5{,}000]_{???}$

Single Layer CNN – Single Filter

- We compute a single filter for a window size of n (here n=3):

Word Vectors: $w_i \in \mathbb{R}^2$
Weight Matrix: $W \in \mathbb{R}^{1\times 6}$
Bias: $b \in \mathbb{R}$

$$\text{output} = \tanh\left(W \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} + b \right)$$

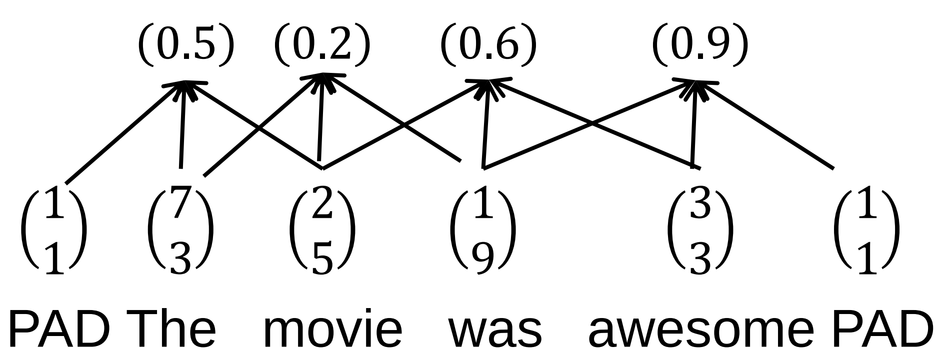
(0.8)

$\binom{1}{1}$ $\binom{7}{3}$ $\binom{2}{5}$ $\binom{1}{9}$ $\binom{3}{3}$ $\binom{1}{1}$

PAD The movie was awesome PAD

Single Layer CNN – Single Filter

- Compute the output for all windows of size n (in our case n=3)
- For each window use the same weight and bias values (shared weights)
- This gives us the same number of digits as the length of the sentence



Single Layer CNN – Pooling Layer

TECHNISCHE UNIVERSITÄT DARMSTADT

- New building block: Pooling
- Idea: Capture the most important activation

- Let $o_1, o_2, ... \in \mathbb{R}$ denote the output values for our filter

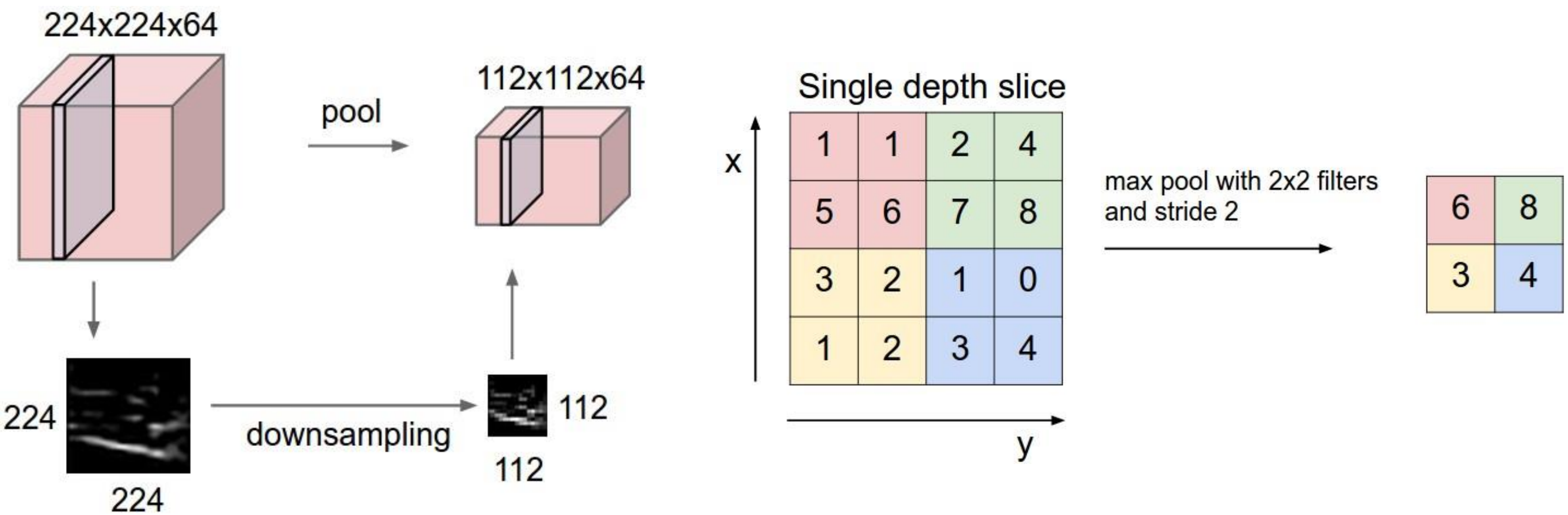
- Compute a max-over-time pooling layer:

$$c = \max_i(o_i) \in \mathbb{R}$$

- Because of max-over-time pooling, length of input sequence is irrelevant.
- We could use the output *c* and forward it to a softmax classifier and derive a sentiment class for the sentence
- Max-pooling most common in NLP. In Computer Vision, min-pooling and mean-pooling also common.

Excursion: Max Pooling in Computer Vision

- In computer vision, pooling is often applied over fixed windows (e.g. 2x2)
- Be careful, don't confuse max pooling and max-over-time pooling



Example:
- 224x224 pixel gray scale images, 64 images / batch
- 2x2 max-pooling reduces each image it to 112 x 112 dimensions

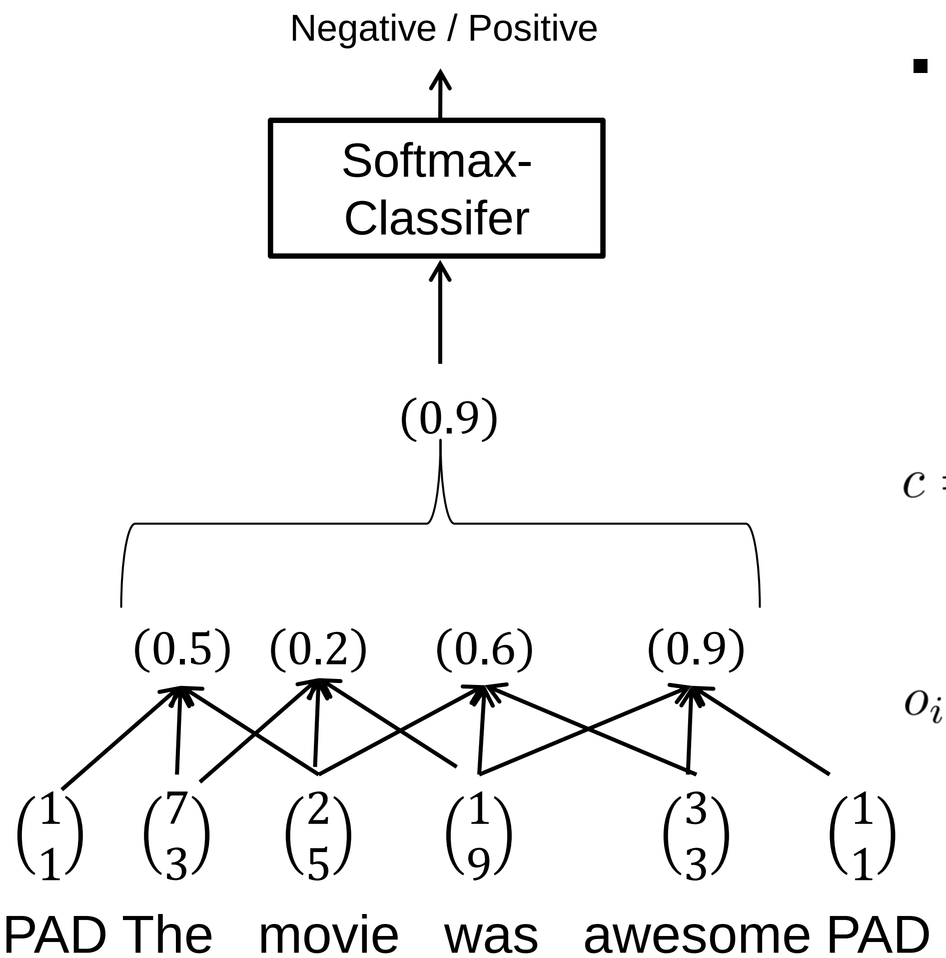
Img-Source: http://cs231n.github.io/convolutional-networks/

Max Pooling vs. Max-over-Time

- Max pool with 2x2 filters on variable sized input generates variable sized output
- Max-over-time generates fixed-sized output
- In NLP, mostly max-over-time is used
  (as presented by Collobert et al., NLP almost from scratch, section 3.2.2)

- A lot of literature on max pooling originated from computer vision
  - Be careful with different terminology and hyper parameters for those pooling layers
- Most libraries (Lasagne, Keras) are optimized for computer vision and only support window sized max pooling
  - Using the existent layers for 1-dimensional max-over-time can be a bottleneck
  - Implementation of max-over-time quite easy in Theano: *T.max(matrix, axis=1)*

Single Layer CNN + Classification

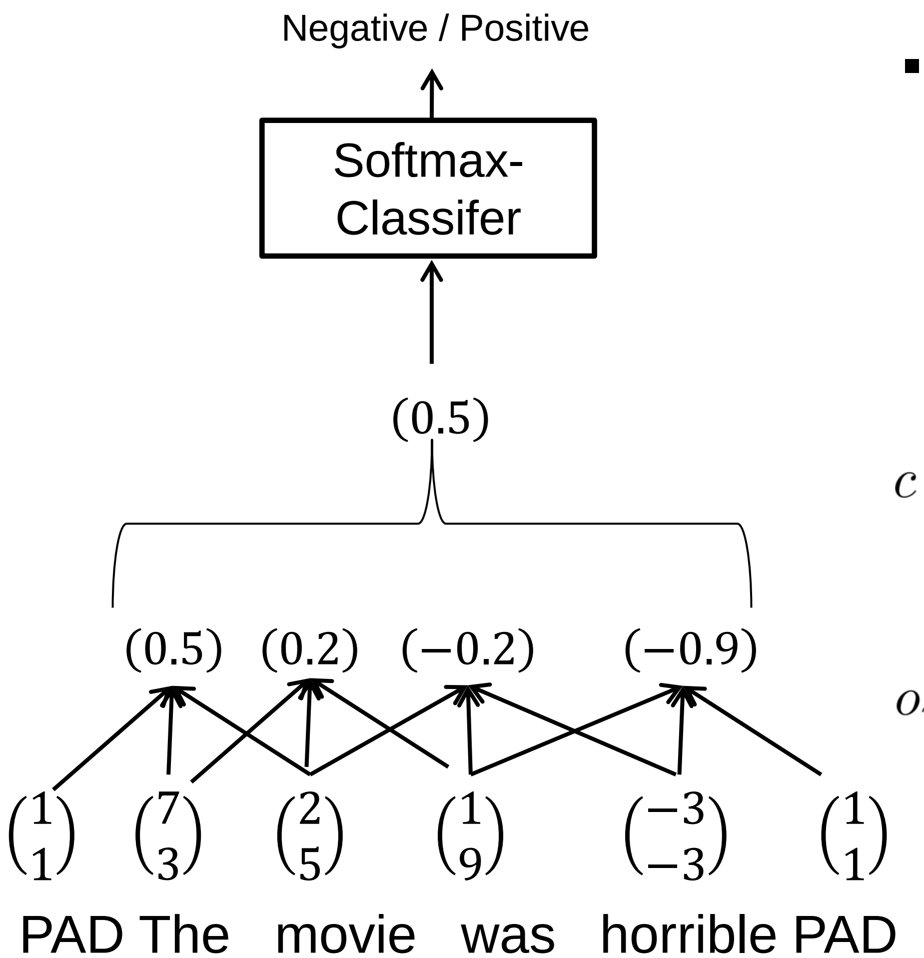


- You can train this like any other Neural Network

$$c = \max_i(o_i)$$

$$o_i = \tanh\left(W \begin{pmatrix} w_{i-1} \\ w_i \\ w_{i+1} \end{pmatrix} + b\right)$$

1 Dimension – not enough information

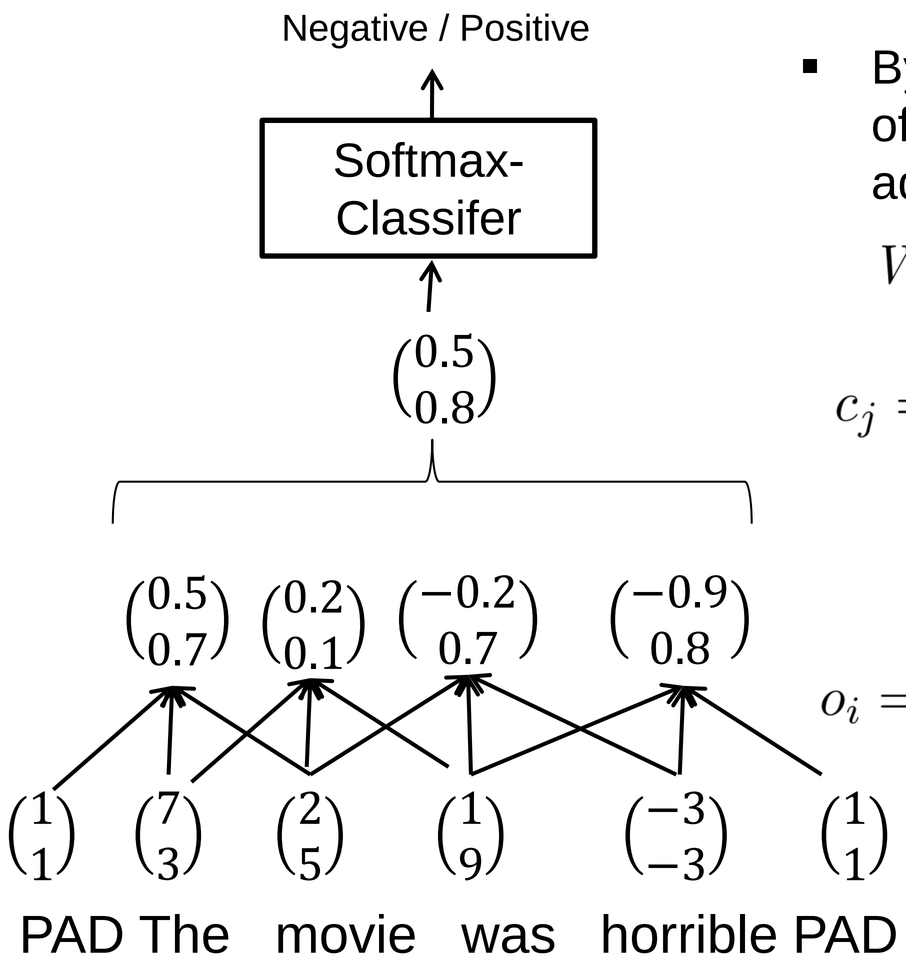


- With only a single filter our possibilities are limited

$$c = \max_i(o_i)$$

$$o_i = \tanh\left(W\begin{pmatrix} w_{i-1} \\ w_i \\ w_{i+1} \end{pmatrix} + b\right)$$

Single Layer CNN – Multiple Filters



- By changing the dimensionality of the weight matrix, we can add further filters:

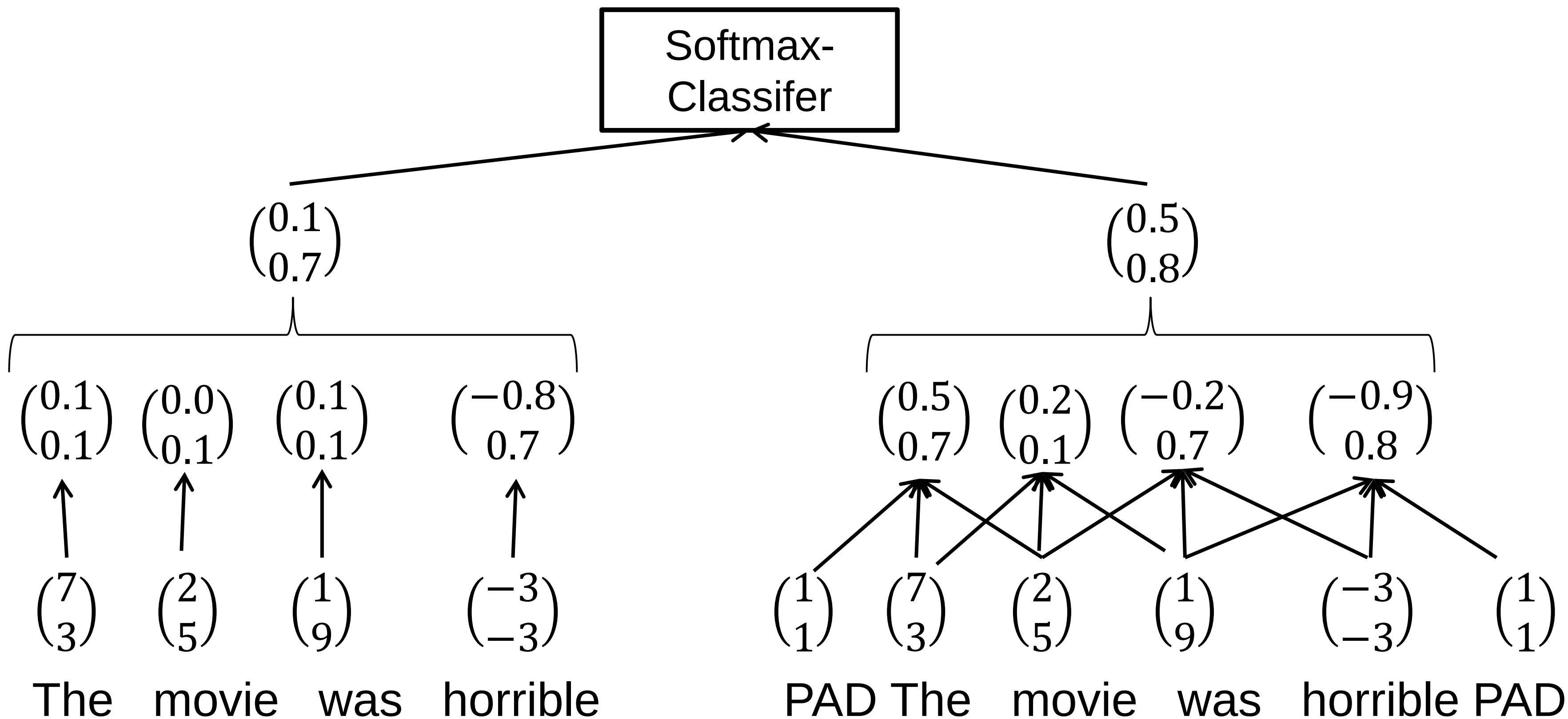
$$W \in \mathbb{R}^{k \times 6}$$

$$c_j = \max_i(o_{i,j}) \text{ for } 0 < j < k$$

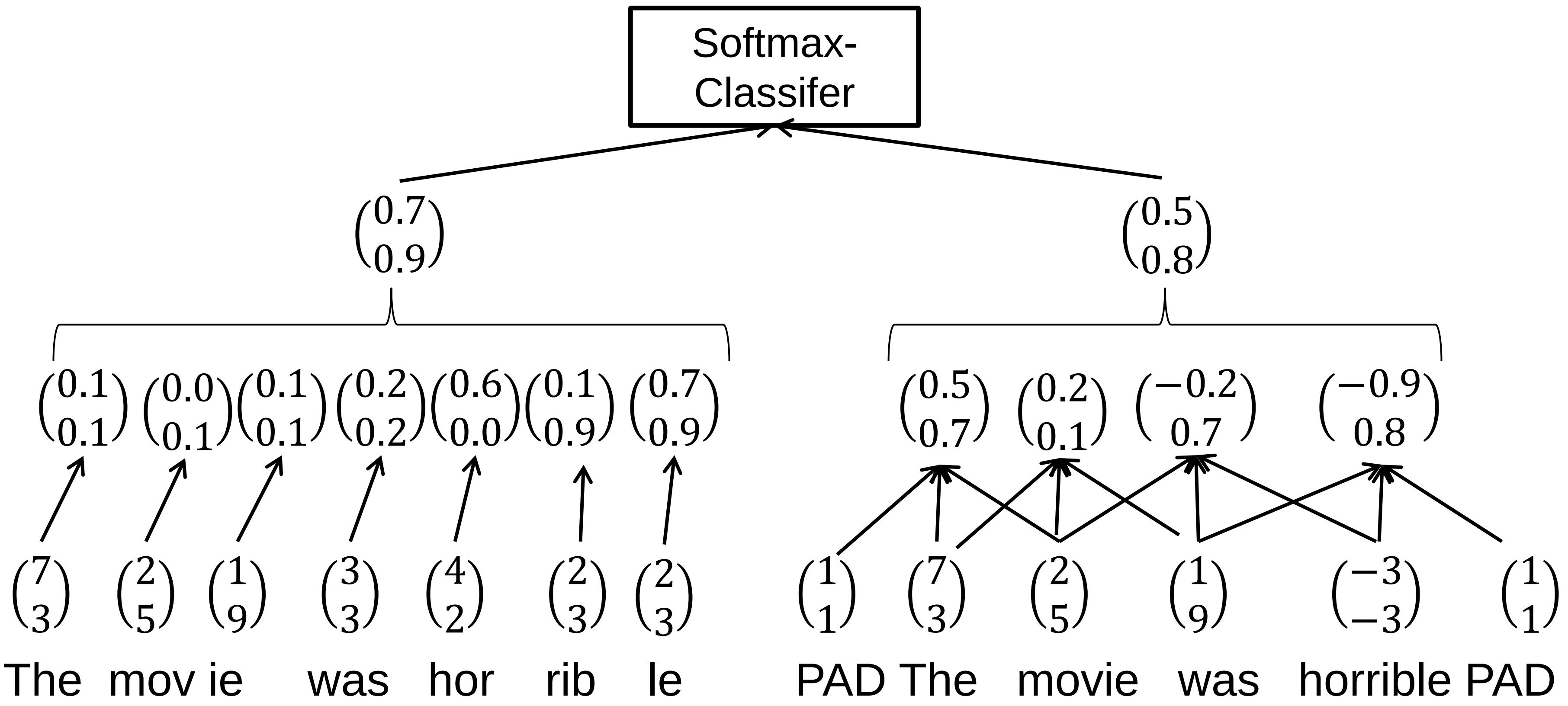
$$o_i = \tanh\left(W \begin{pmatrix} w_{i-1} \\ w_i \\ w_{i+1} \end{pmatrix} + b\right) \in \mathbb{R}^k$$

Going further with Filters

- We can create convolutional layers working on unigrams, bigrams, trigrams etc. and combine their output



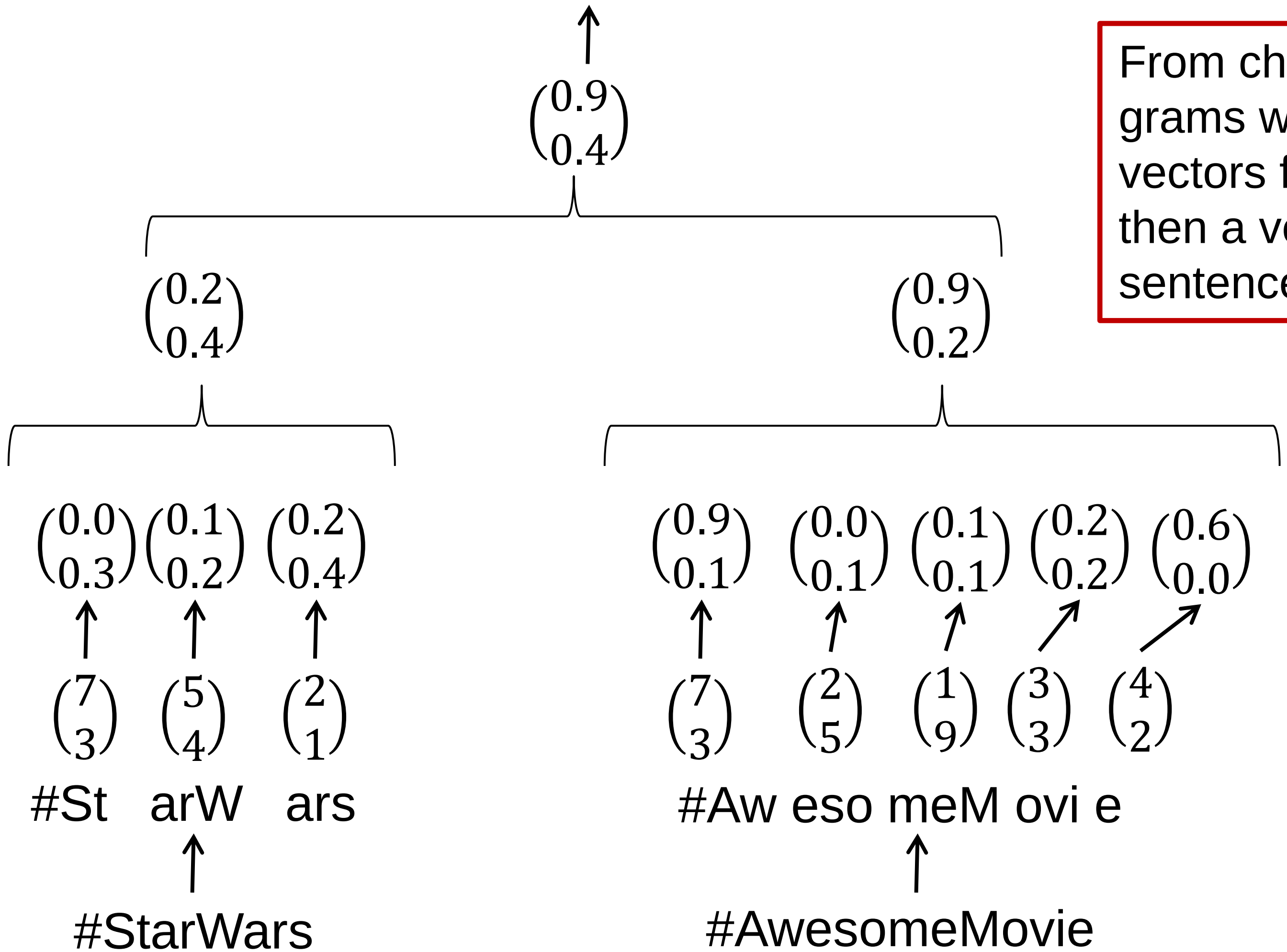
Or work on a different granularity



Conv. Layer on character trigrams

Conv. Layer on word trigrams

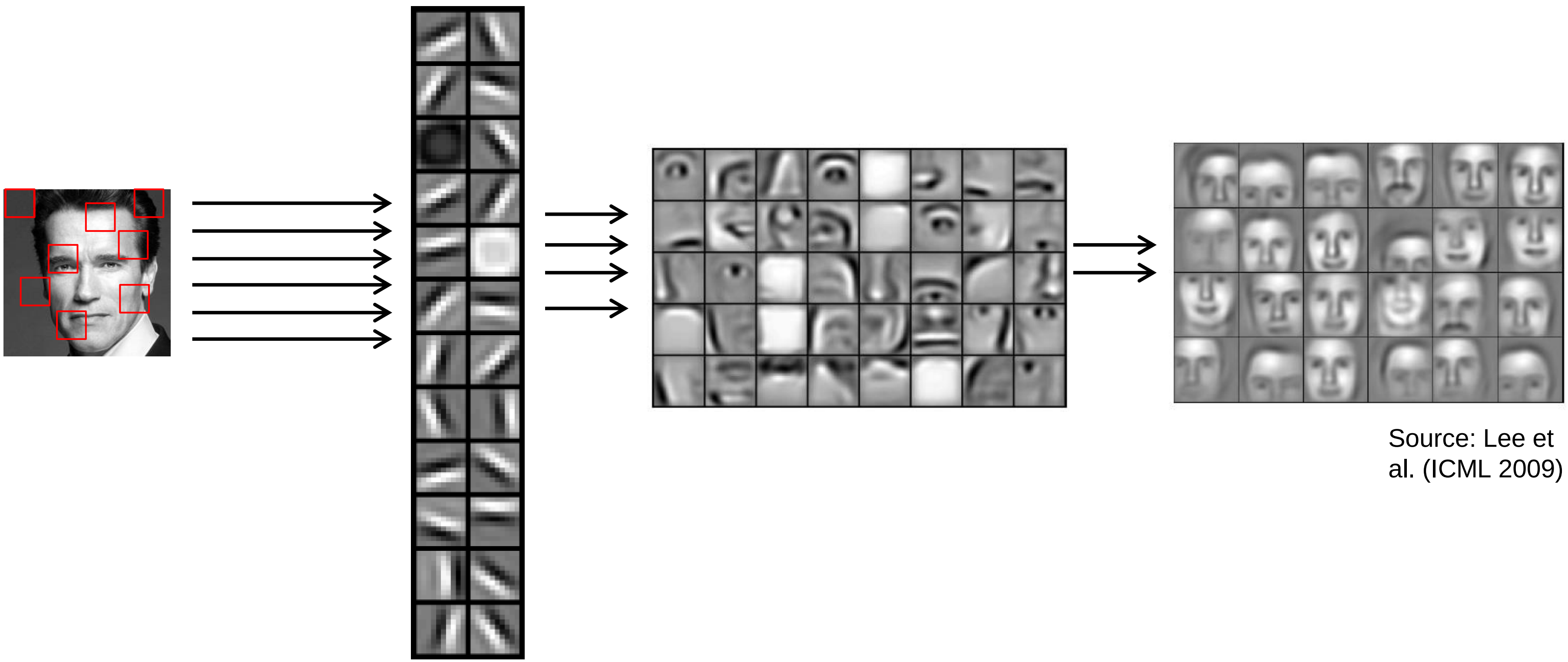
Stacking Convolutional Layers



From character n-grams we could derive vectors for words and then a vector for the sentence

Stacked Convolutional Layers

- Computer vision uses stacked convolutional layers to derive from simple representations high level representations



Source: Lee et al. (ICML 2009)

Input
e.g. 40 000 dim

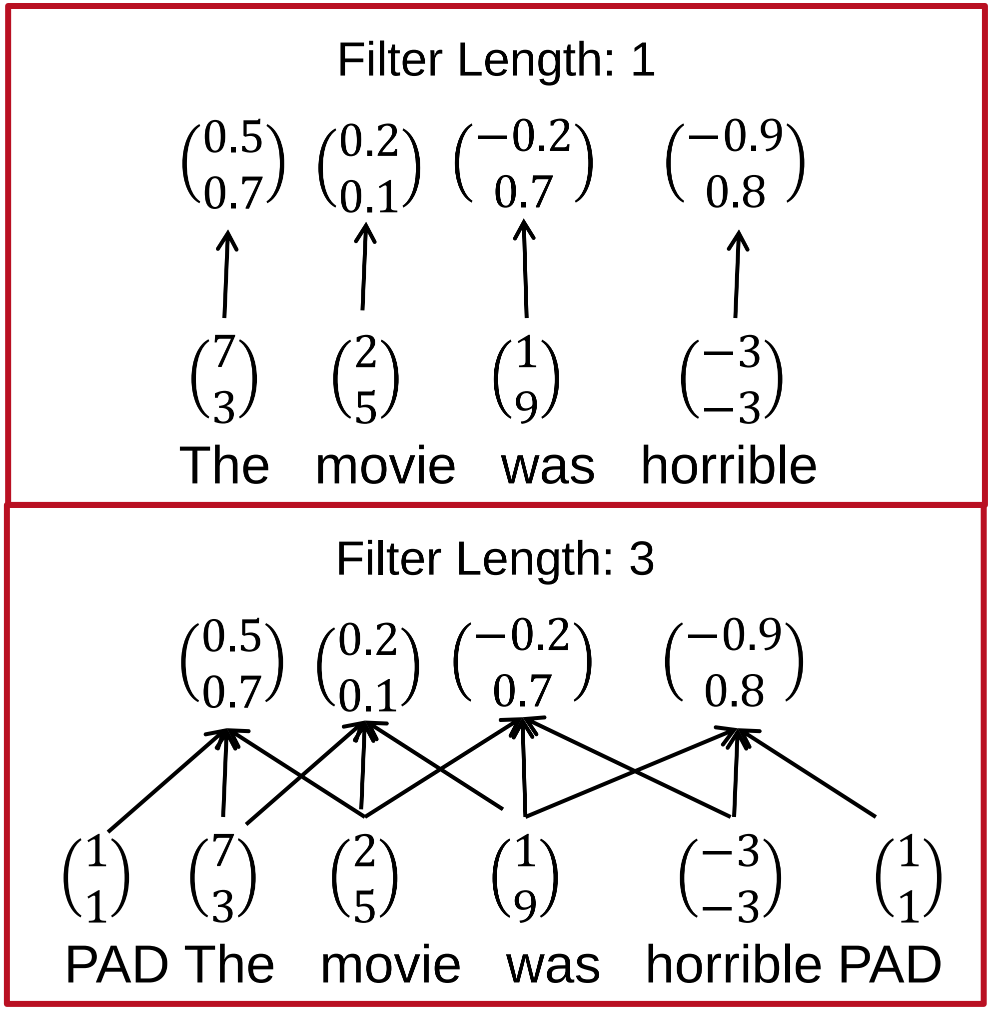
Layer 1
9 600 dim

Layer 2
3800 dim

Layer 3
900 dim

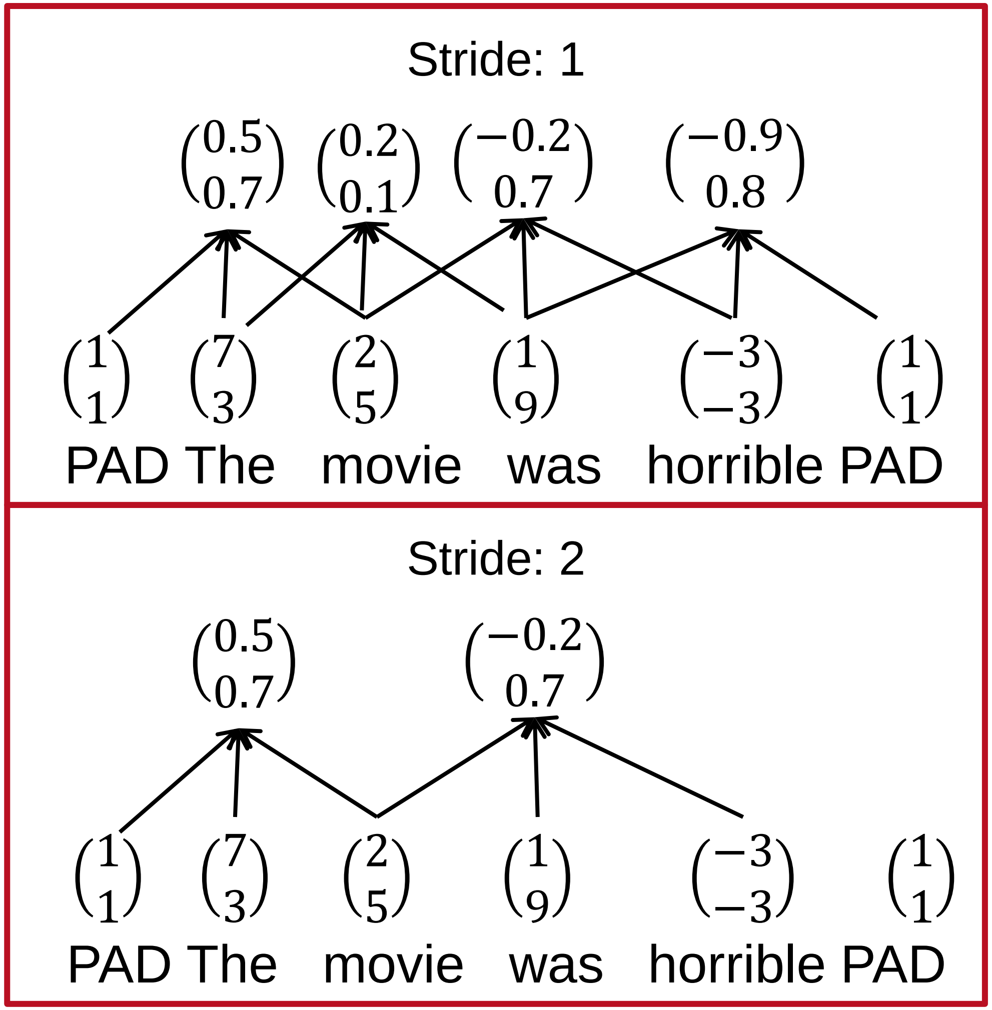
Terminology: Filter Length

- The filter length is the extension of each filter
- Mainly inspired by Computer Vision where we work on spatial close pixels
- In NLP we are more flexible:
  - Use a context window of size *n*
  - Use dependency links / syntax tree



Terminology: Stride

- The *stride* specifies the steps size we move across a sentence
- In NLP: Typically stride=1
- In computer vision: Other values can be used



How to choose the embeddings?

TECHNISCHE UNIVERSITÄT DARMSTADT

- When we use words, how should we initialize the embeddings?

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

- Rand: Random initialization
- Static: word2vec, no updates during training
- Non-static: word2vec with updates
- Multichannel: next slide

Source: Kim, 2014. Performance on Sentence Classification Tasks

Multi-Channel Idea

- We start with 2 copies for the word vectors, both initialized with word2vec/GloVe etc.
- Only one version of them is updated, the other is static
- We apply the same filters to both channels before we apply the pooling

- The one channel can learn task specific embeddings
  - E.g. for sentiment, *good* and *bad* should be far away in vector space

- So far mixed results
- Different Idea: Use differently pre-trained word embeddings
  - E.g. based on local context, on dependency trees, on relations from knowledge bases etc.

- Reference: Kim, 2014. *Performance on Sentence Classification Tasks*

Hints on the Implementation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Numpy and Theano cannot work with variable sized rows
- How to model a dataset like this?
  [[This is my first sentence .]
  [This is my second , longer sentence .]
  [Super short]]
- 2 Approaches:
  - Ignore minibatches, just input 1 sentence at a time for training / testing
    - Bad for performance
  - Pad the sentences with 0 to make them of the same length
    - Be careful with the padding, that the max-pooling does not choose your padded values
    - Be careful with the runtime, that a single super long sentence does not create too much padding for all other sentences.
    - Great to run on GPU (convolutions can be computed easily in parallel)

Hints on the Implementation II

- Most implementation for convolutional layers are targeted for computer vision
- They introduce a lot more hyper parameters.
- Keras.Convolution1D:
  - nb_filter: dimensionality of the output
  - filter_length: The extension (spatial or temporal) of each filter.
  - border_mode: How are vales at the border handled. ‘valid' or 'full'.
  - subsample_length: The stride value for the filter
- Keras.MaxPooling1D:
  - pool_length: factor by which to downscale. 2 will halve the input.
  - stride: Stride value.
  - ignore_border: boolean
- **Note**: Existent Convolution1D and MaxPooling1D not suitable for max-over-time implementation