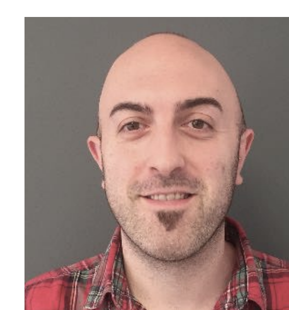
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

ConvNets for NLP

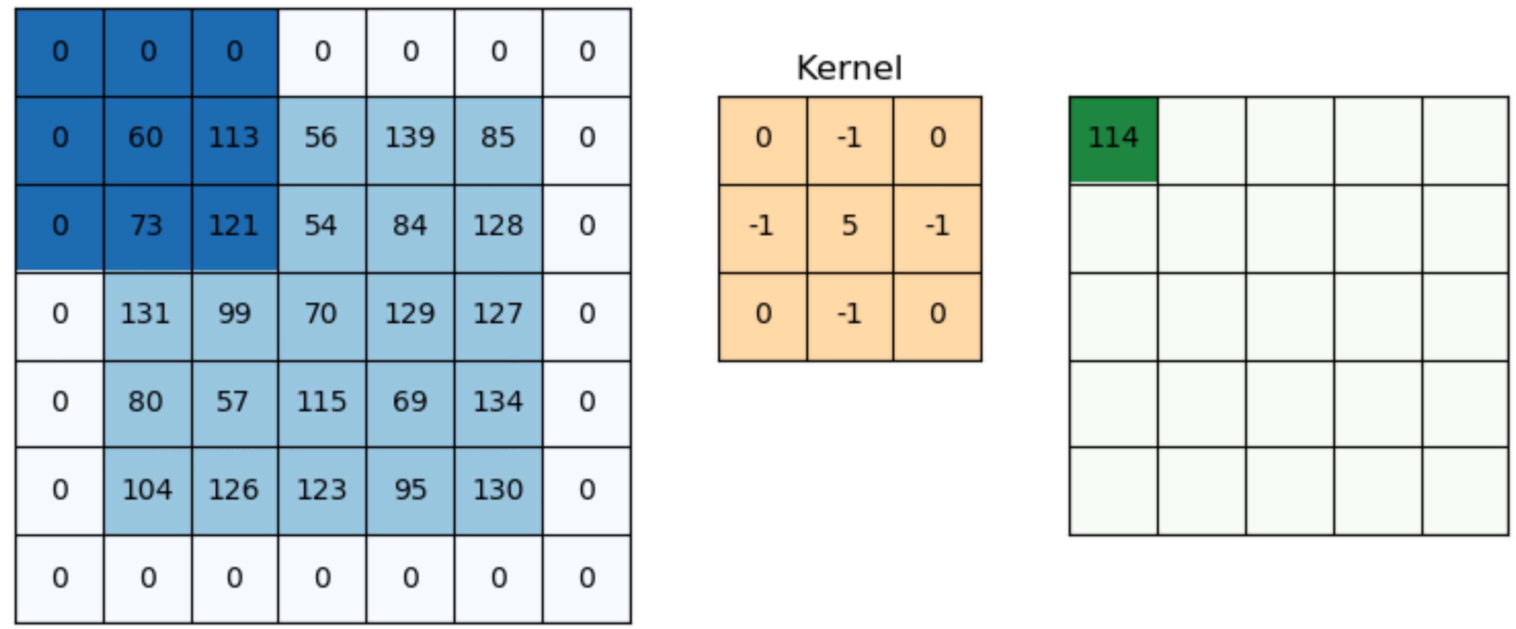
Noé Casas



noe.casas@upc.edu

Theory

Discrete Convolutions. Intuition



- How it works:

    1. Slide a small filter matrix (**kernel**) over the **input** matrix.

    2. At each position, compute the product of kernel and input values, and add them together.

    3. The output matrix is the concatenation of the application of the filter over the input matrix.

- The “trainable” weights are the filters (kernels).

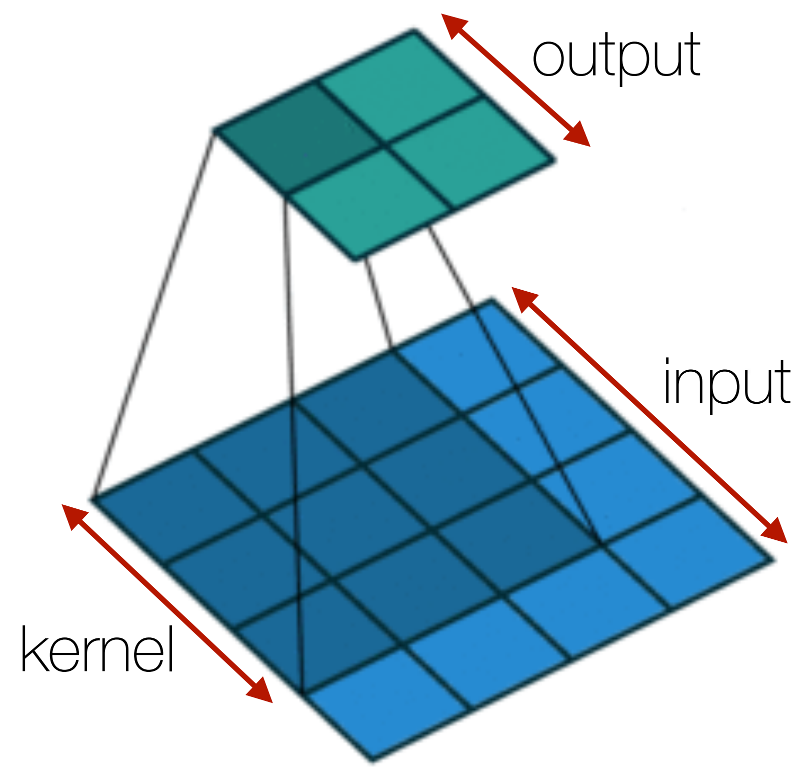
Discrete Convolutions. Purpose

- Used to detect local features.
- Invariant to feature position.
- Stacked convolutional layers detect hierarchical features.

Discrete Convolutions. Domains

- Images (a matrix of pixels - 2D convolution)
- Text (a sequence of tokens - 1D convolution)
- Video (a sequence of images - 3D convolution)

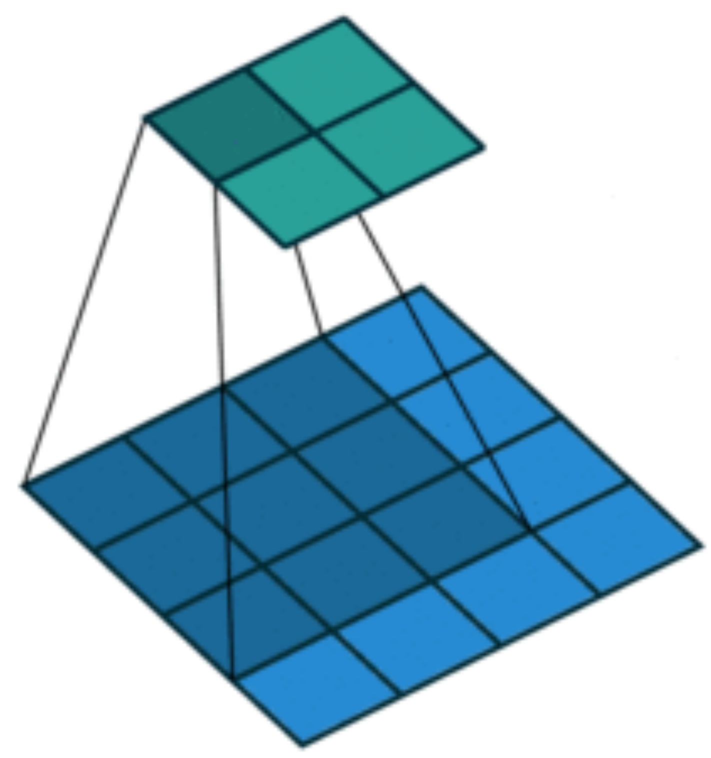
(Notation in images)



https://github.com/vdumoulin/conv_arithmetic

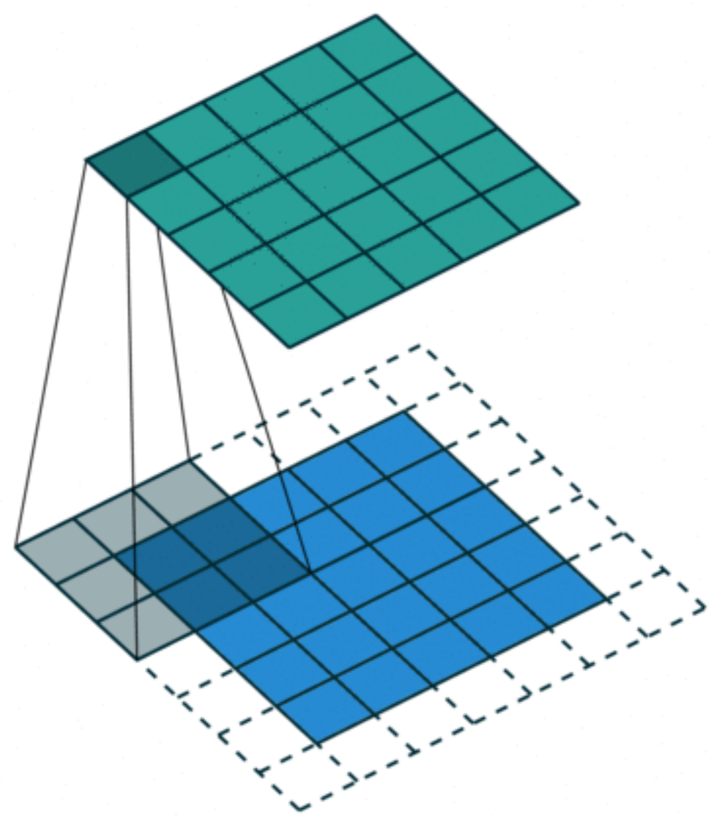
Discrete 2D Convolutions. Hyperparams (1/4)

- Input size: 4 x 4
- Kernel size: 3
- Output size: 2 x 2

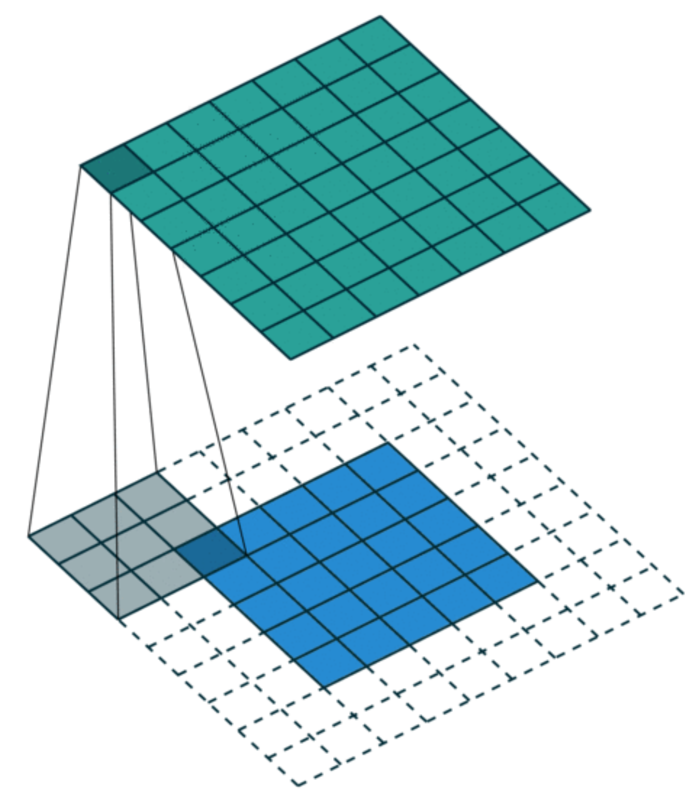


https://github.com/vdumoulin/conv_arithmetic

Discrete 2D Convolutions. Hyperparams (2/4)



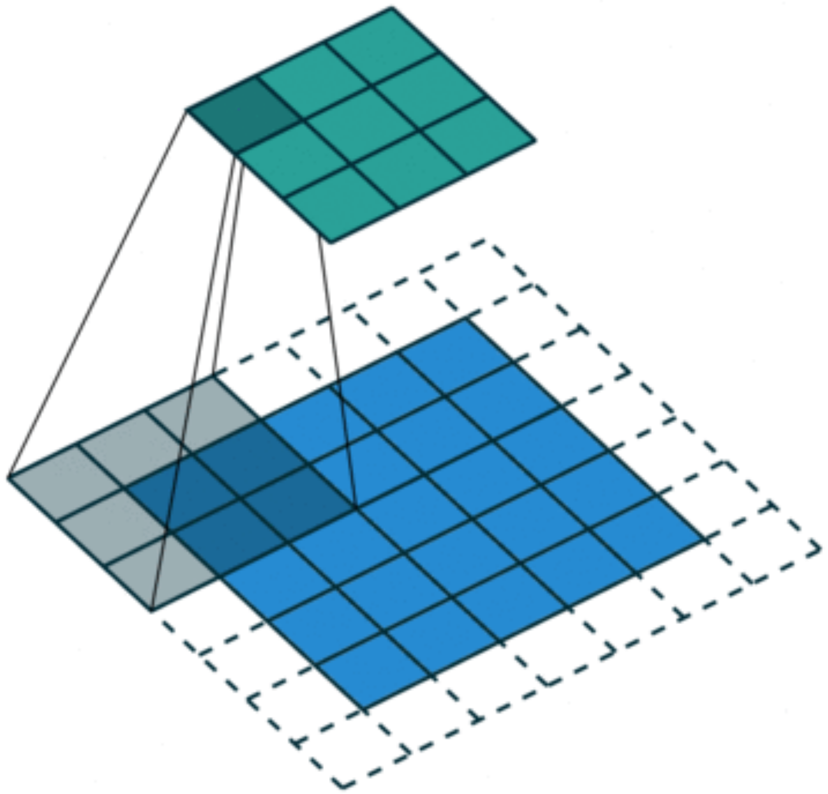
- Input size: 5 x 5
- Kernel size: 3
- Padding: 1
- Output size: 5 x 5



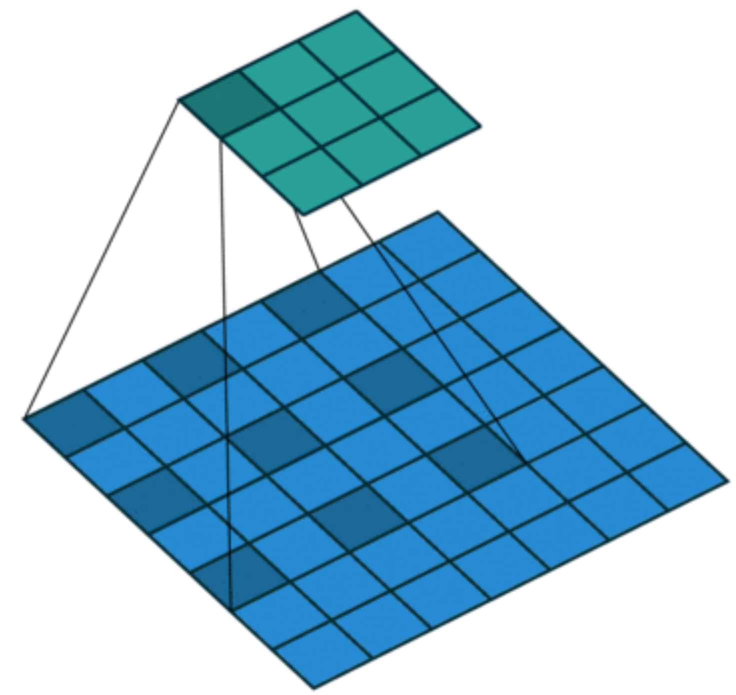
- Input size: 5 x 5
- Kernel size: 3
- Padding: 2
- Output size: 7 x 7

https://github.com/vdumoulin/conv_arithmetic

Discrete 2D Convolutions. Hyperparams (3/4)



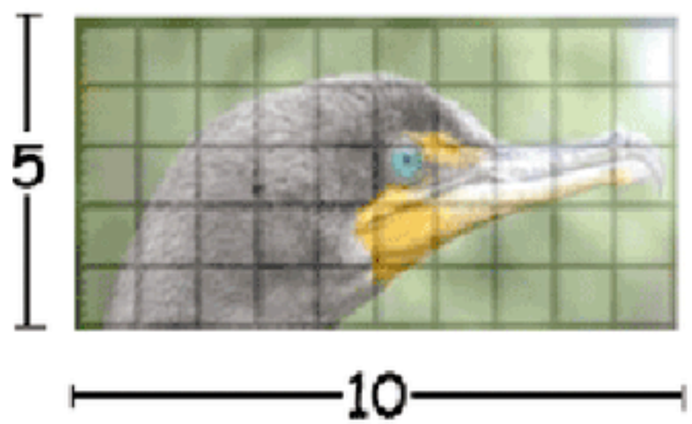
- Input size: 5 x 5
- Kernel size: 3
- Padding: 1
- Stride: 2
- Output size: 3 x 3

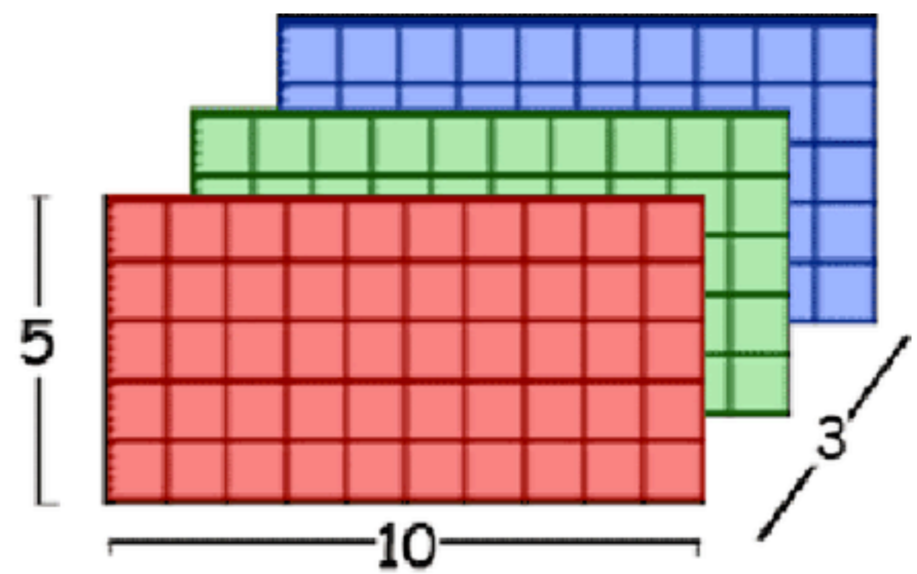


- Input size: 7 x 7
- Kernel size: 3
- Padding: 0
- Stride: 1
- Dilation: 2
- Output size: 3 x 3

https://github.com/vdumoulin/conv_arithmetic

(Multichannel inputs)



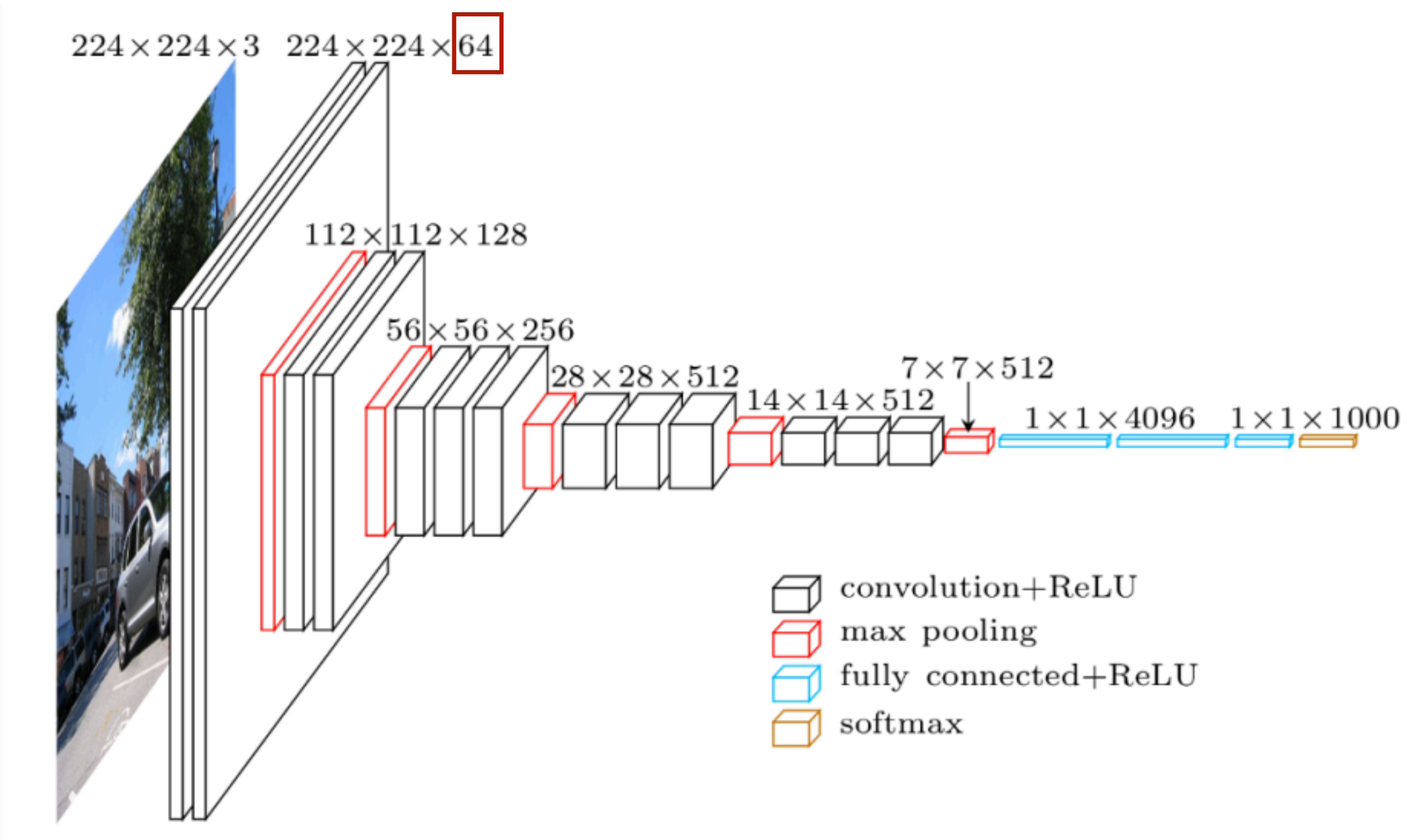


- Input size: 5 x 10 x 3
- Kernel size: 3
- Padding: 1
- Output size: 5 x 10 x 1

https://www.engineersgarage.com/contributions/content-based-image-retrieval-cbir-using-matlab/

Discrete 2D Convolutions. Hyperparams (4/4)

- Number of filters (= number of output channels)



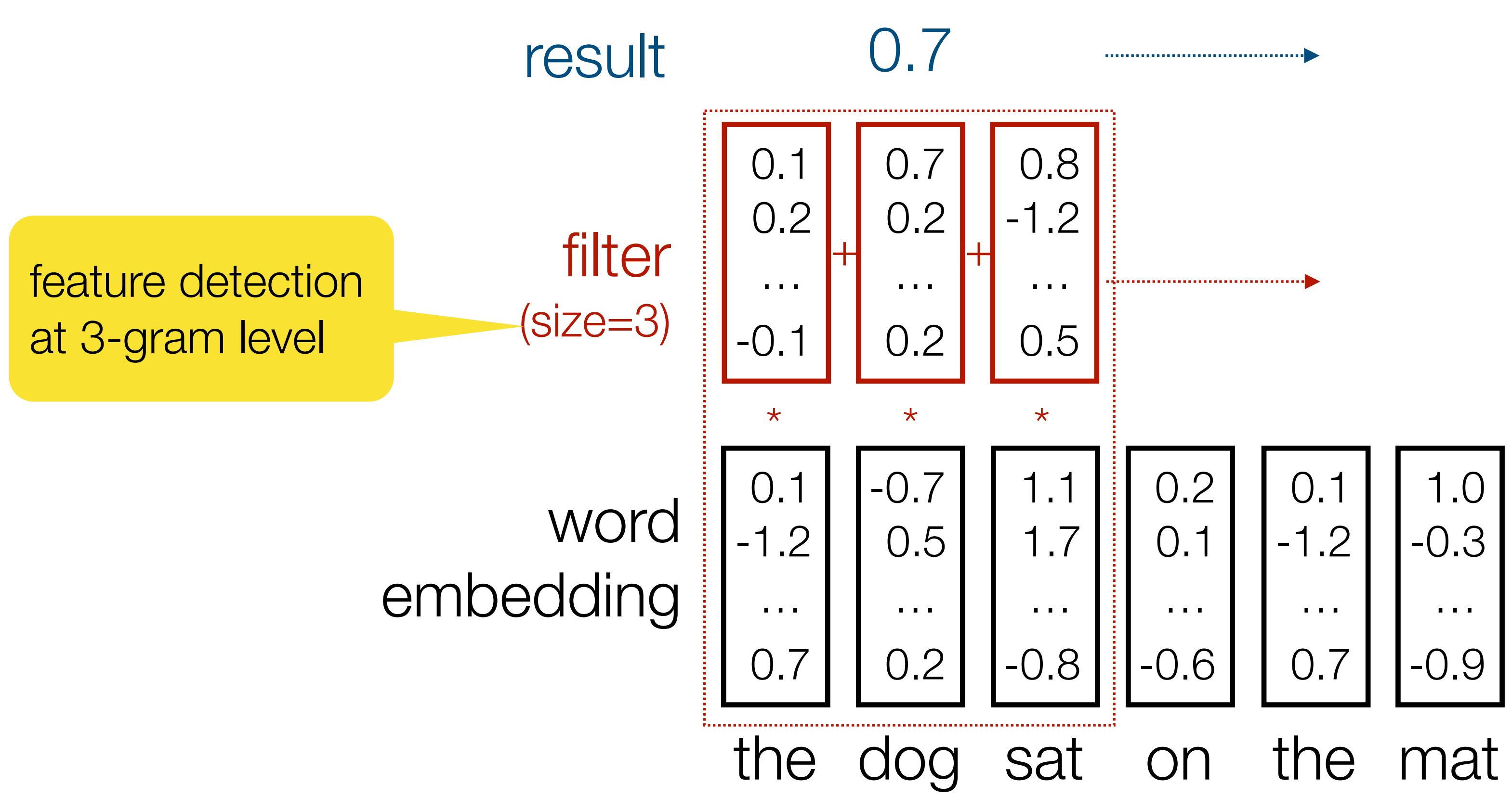
https://www.jeremyjordan.me/convnet-architectures/

Discrete 1D Convolutions

Same as in 2D, but with 1D input and kernel:

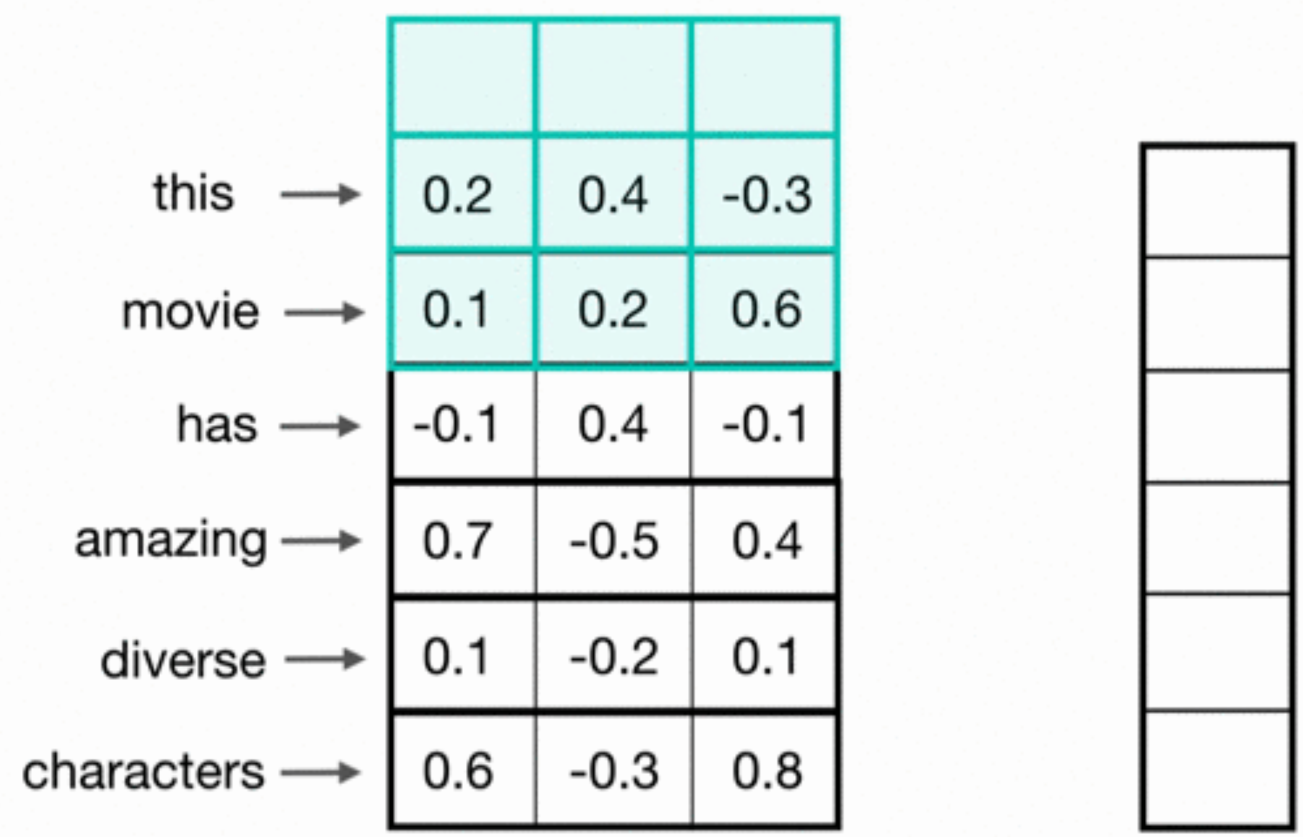
$$(I * k)[n] = \sum_{-M}^{M} I[n-m]k[m]$$

Applied over a continuous representation, e.g.:



Example

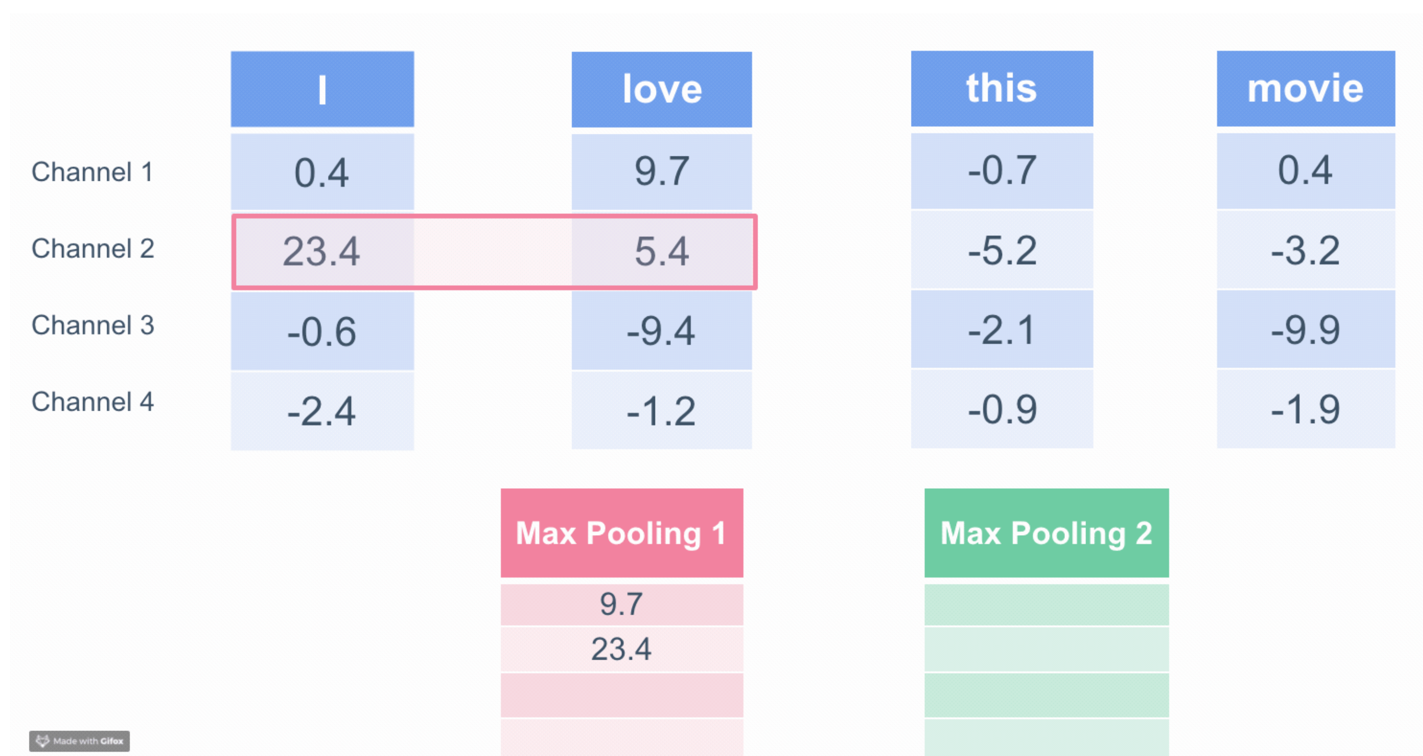
- Sequence length: 6
- Embedding dimensionality: 3
- Kernel size: 3
- Padding: 1
- Output length: 6



https://cezannec.github.io/CNN_Text_Classification/

Sliding Window operations: Max pooling over time

- Sliding window over input along time dimension.
- Output is maximum value in window.
- Usually has stride of the same size of the window size.
- Used after convolution.
- Purpose: reduce complexity while capturing most important activation from previous layer.



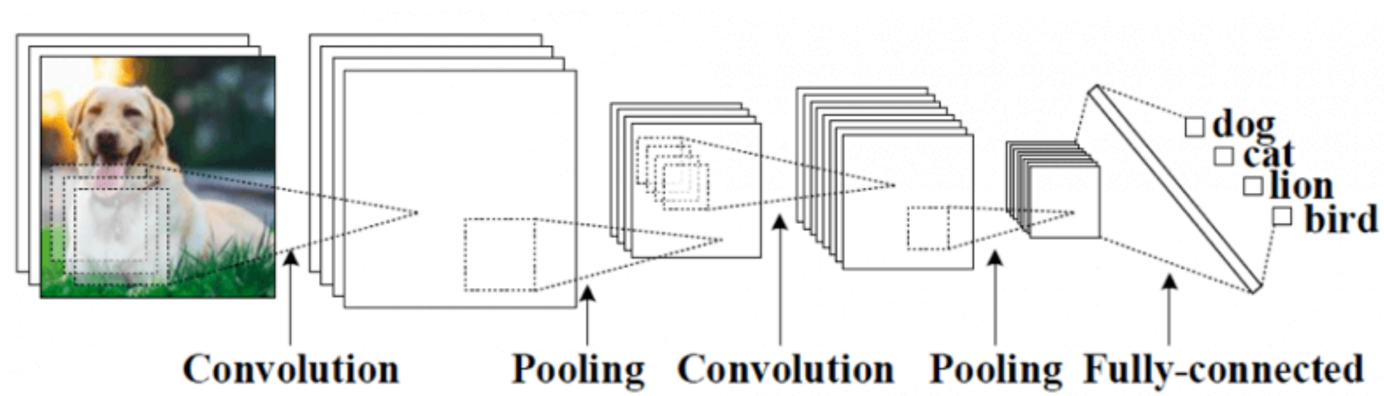
https://www.wandb.com/classes/intro/class-8-notes

Sliding Window operations: Average pooling over time

- Sliding window over input along time dimension.
- Output is average value in window.
- Usually has stride of the same size of the window size.
- Used after convolution.
- Purpose: reduce complexity while retaining info.

1D Convnets Architectural Fit

- 2D convnets are normally used as a group with pooling and ReLU, over fixed-sized inputs:

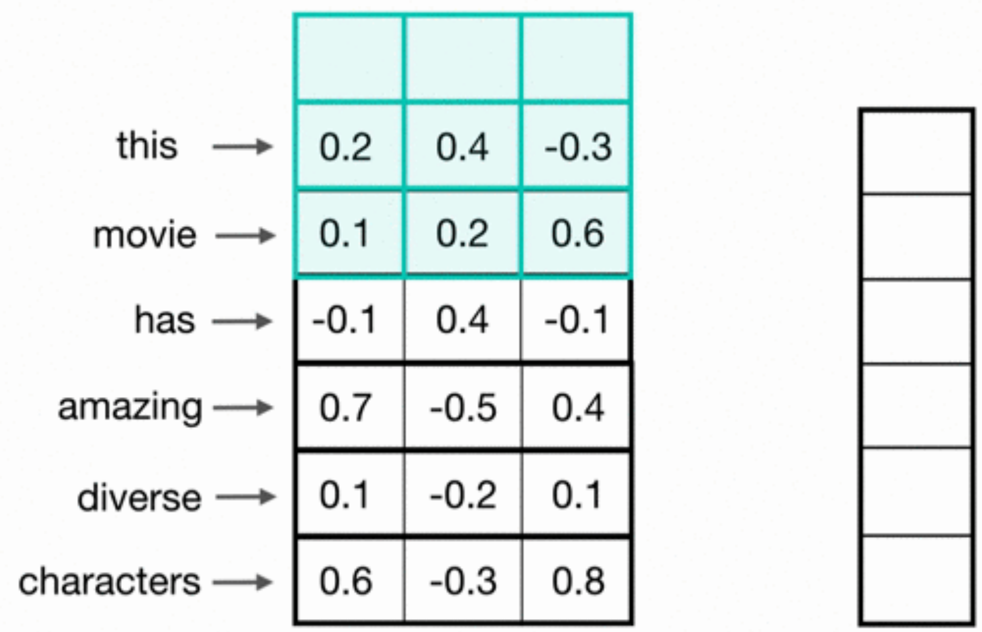


- In NLP, convnet inputs are variable-length sequences

- Depending on the task, a "collapsing" operation (e.g. max) may be needed to obtain a fixed-size representation.

https://www.ayasdi.com/awards/artificial-intelligence-100/

Collapsing pooling approaches



- Max pooling: computes the maximum values per channel in the input sequence.

- Average pooling: computes the average values per channel in the input sequence.

- k-max pooling: computes the subsequence of *k* maximum values in the input sequence. Keeps order of appearance.

Conv. with Kernel size = 1

- Position-wise linear transformation.
- Increases or decreases channel dimensionality (depending on number of filters / num. output channels).
- In 2D convolutions they are known as 1x1 convolutions or “network-in-network”.

Depthwise Separable Conv.

- Operation divided in two steps:

  1. Per-channel normal convolution
     → output has same number of channels as input

  2. Position-wise convolution (kernel width=1)

- Less parameters and less computational cost.

Dynamic convolutions

- Normal convolutions have fixed (trainable) kernels.
- Idea: compute kernels dynamically in the neural network.

Batch processing

- Most neural networks are fed mini-batches of data.
- In-batch padding is needed (apart from the CNN padding).

Comparison with RNNs

- Speed:
  - Convnets are computed in parallel.
  - RNNs must be computed sequentially.
- Dependency range / receptive field:
  - RNNs tend to capture too much of previous vector (bad if output is taken from the last position). Attention mitigates this.
  - Convnets only handle dependencies within the filter size. Dilation and stacked convolutions mitigate this.

Exercises

E1: vanilla convolution, no padding

tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3

t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture11-convnets.pdf

E2: padding = 1

Ø	**0.0**	**0.0**	**0.0**	**0.0**
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	**0.0**	**0.0**	**0.0**	**0.0**

Ø,t,d	−0.6
t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3
g,o,Ø	−0.5

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture11-convnets.pdf

E3: channels=3, padding = 1

∅	**0.0**	**0.0**	**0.0**	**0.0**
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
∅	**0.0**	**0.0**	**0.0**	**0.0**

∅,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,∅	−0.5	−0.9	0.1

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

E4: padding, max pooling

∅	**0.0**	**0.0**	**0.0**	**0.0**
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
∅	**0.0**	**0.0**	**0.0**	**0.0**

∅,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,∅	−0.5	−0.9	0.1

max p	0.3	1.6	1.4

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

E5: padding, avg. pooling

∅	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
∅	0.0	0.0	0.0	0.0

∅,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,∅	−0.5	−0.9	0.1

ave p	−0.87	0.26	0.53

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture11-convnets.pdf

E6: stride=2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
d,r,t	−0.5	−0.1	0.8
t,k,g	−0.2	0.1	1.2
g,o,Ø	−0.5	−0.9	0.1

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

E7: max.pooling over time, stride=2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
Ø	−Inf	−Inf	−Inf

Ø,t,d,r	−0.6	1.6	1.4
d,r,t,k	−0.5	0.3	0.8
t,k,g,o	0.3	0.6	1.2
g,o,Ø,Ø	−0.5	−0.9	0.1

http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture11-convnets.pdf

E8: *k*-max.pooling, *k*=2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

2-max p	−0.2	1.6	1.4
	0.3	0.6	1.2

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

E9: dilation=2

Ø	**0.0**	**0.0**	**0.0**	**0.0**
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	**0.0**	**0.0**	**0.0**	**0.0**

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

1,3,5	0.3	0.0
2,4,6		
3,5,7		

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

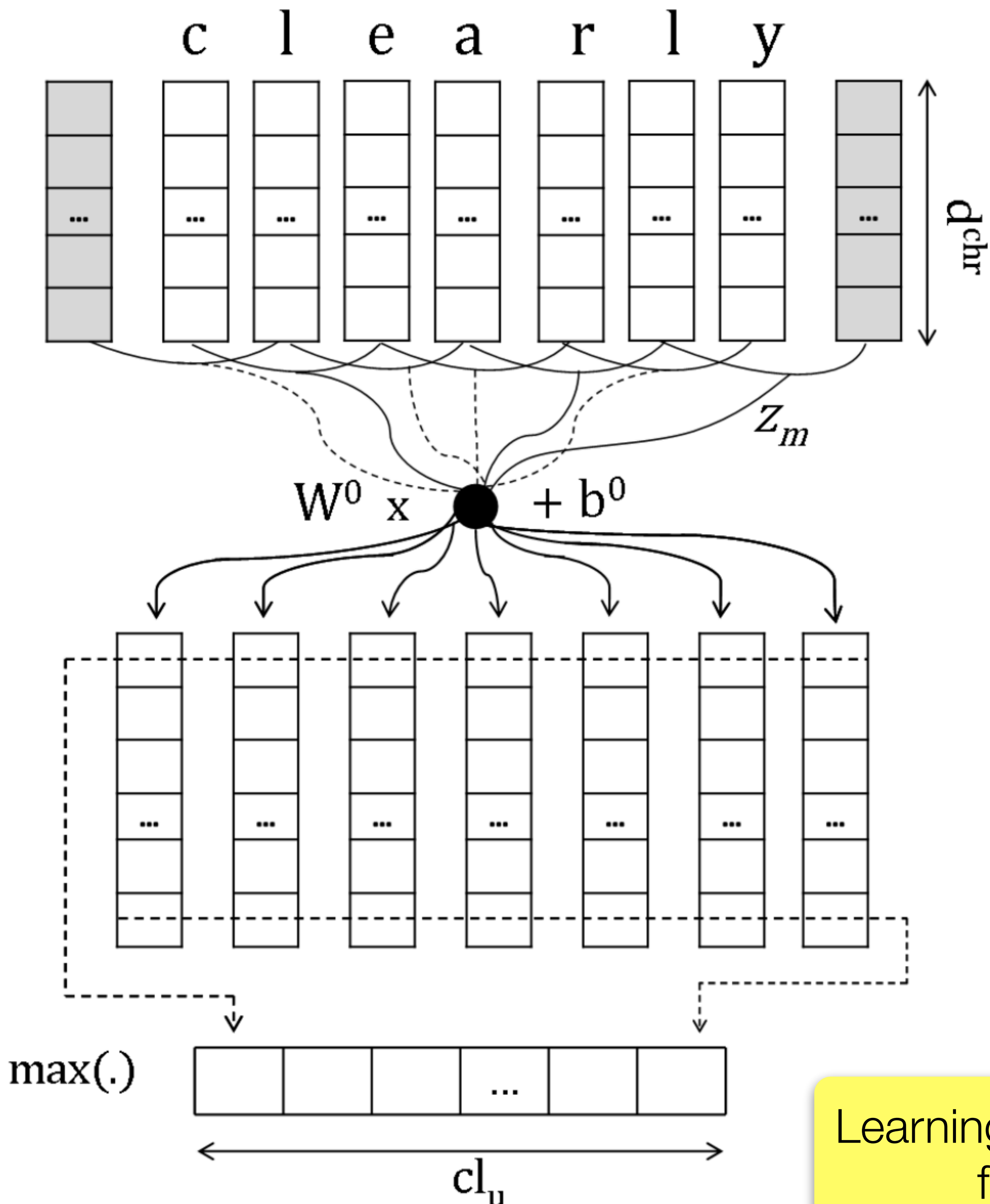
1	−1	2	−1
1	0	−1	3
0	2	2	1

2	3	1
1	−1	−1
3	1	0

1	3	1
1	−1	−1
3	1	−1

Application Examples

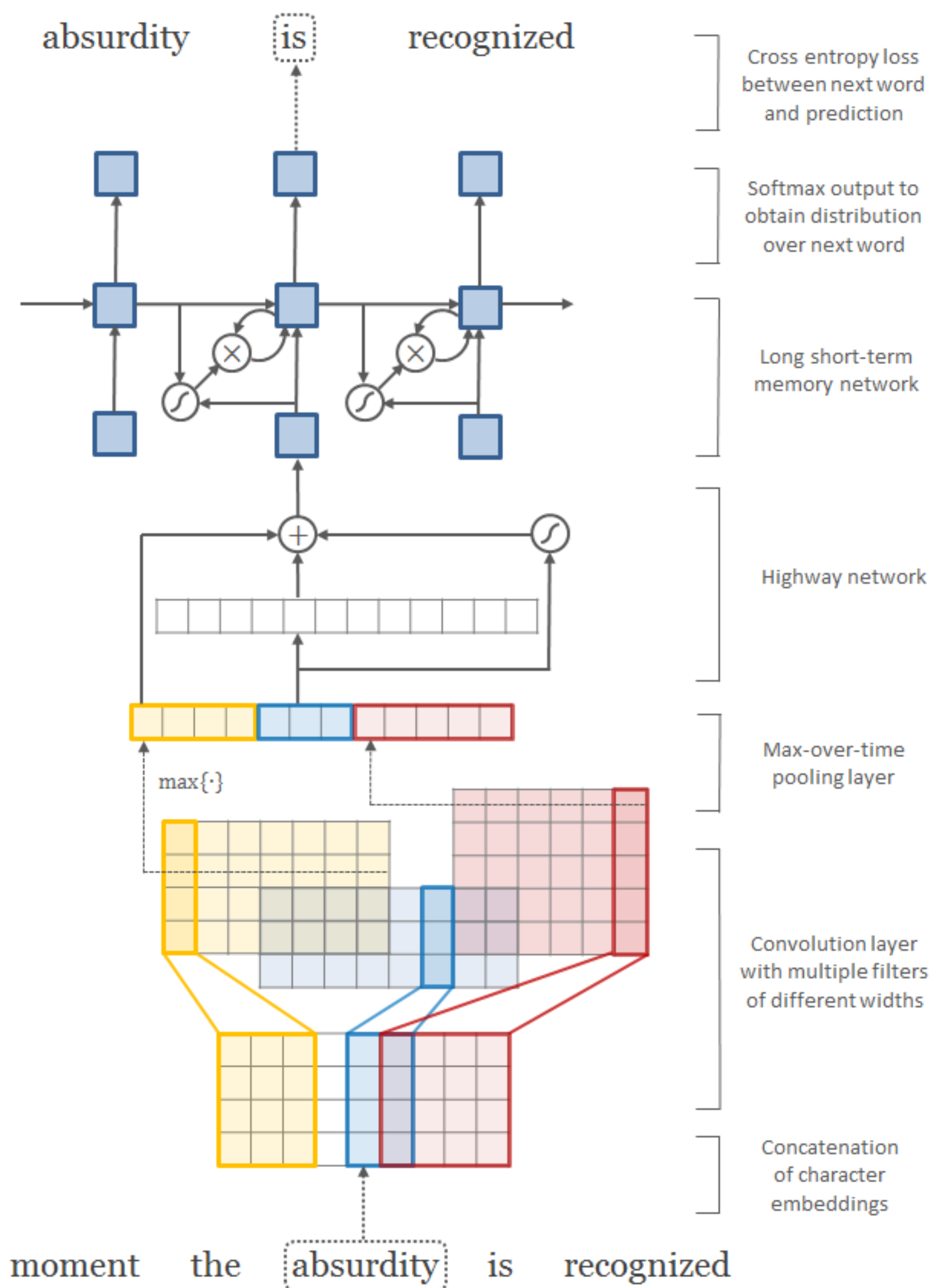
Word Representation Learning (1/4)



Learning Character-level Representations for Part-of-Speech Tagging

Dos Santos and Zadrozny, 2015

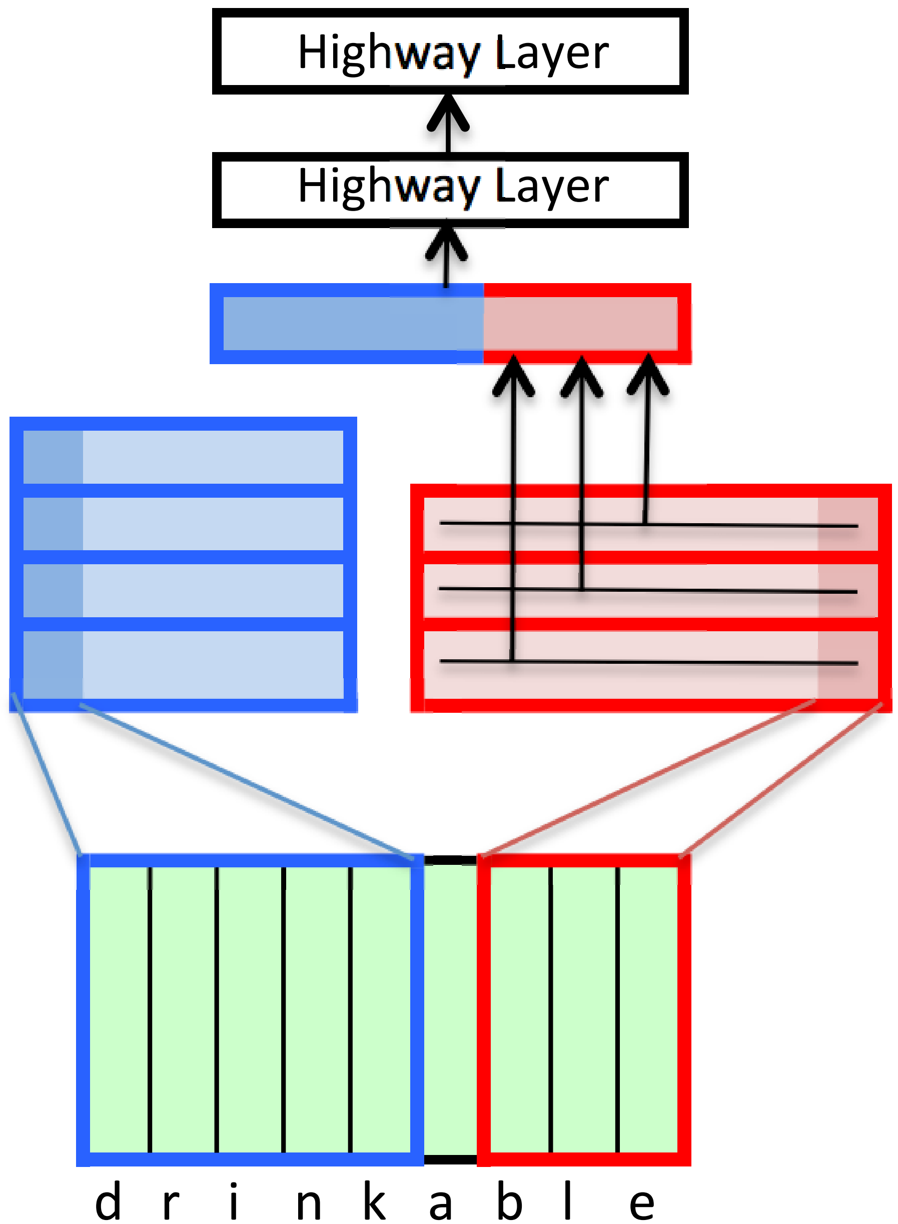
Word Representation Learning (2/4)



Character-Aware Neural Language Models

Kim et al., 2015

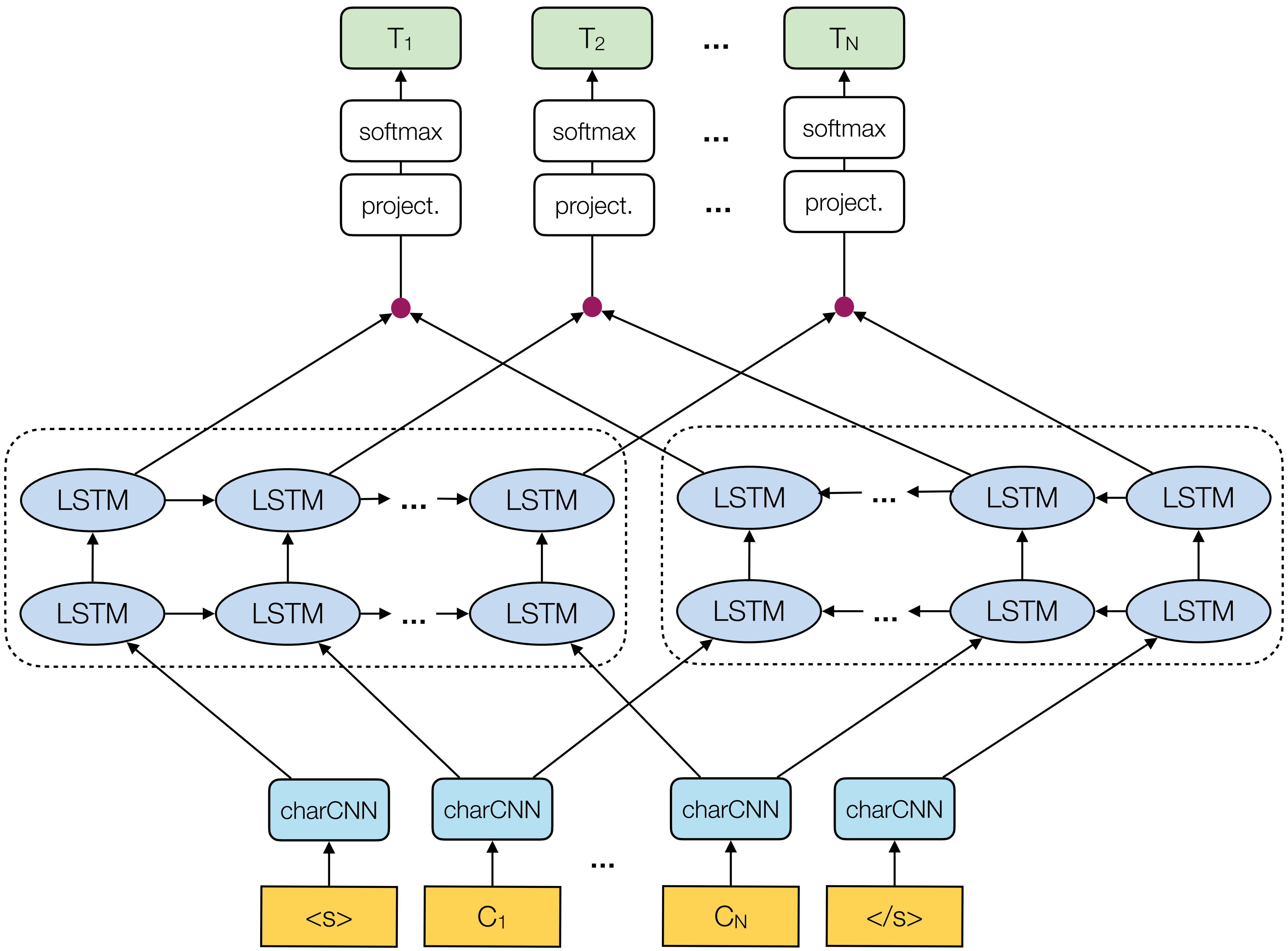
Word Representation Learning (3/4)



Character-based Neural Machine Translation

Costa-jussà and Fonollosa, 2016

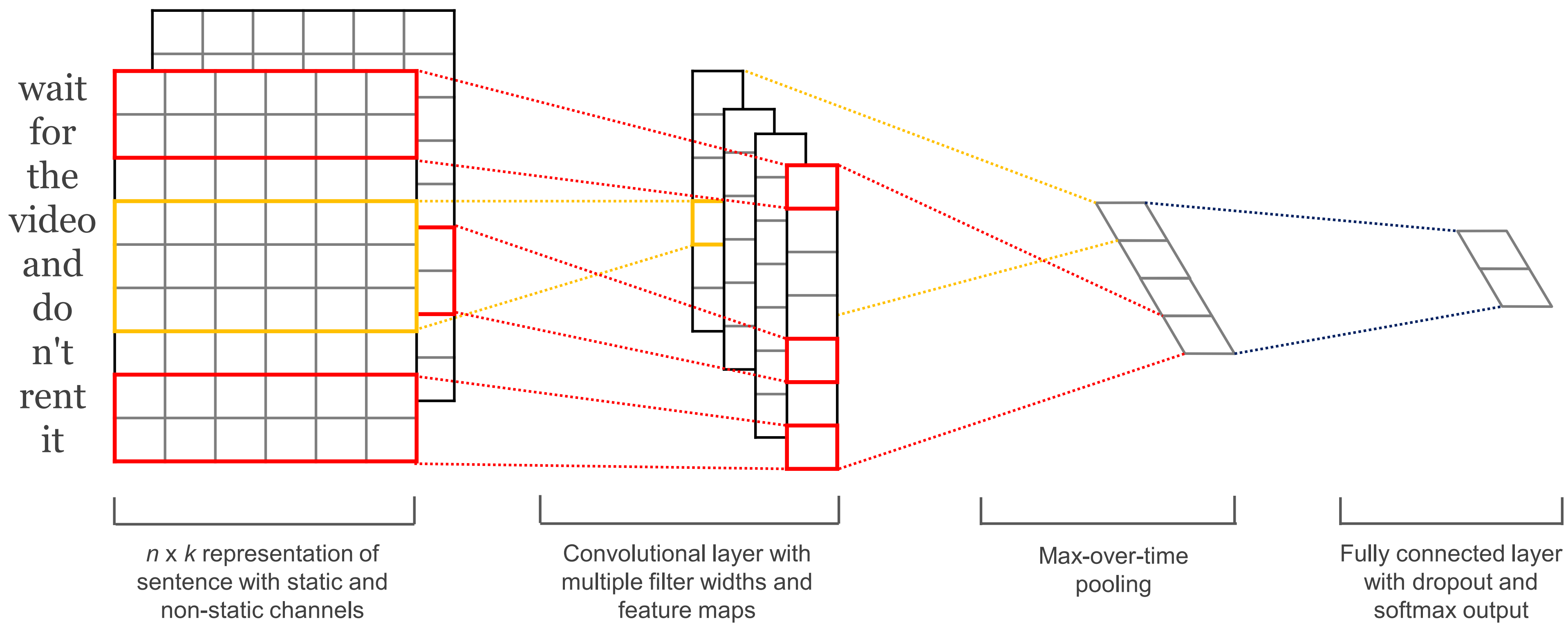
Word Representation Learning (4/4)



Deep contextualized word representations

Peters et al., 2018

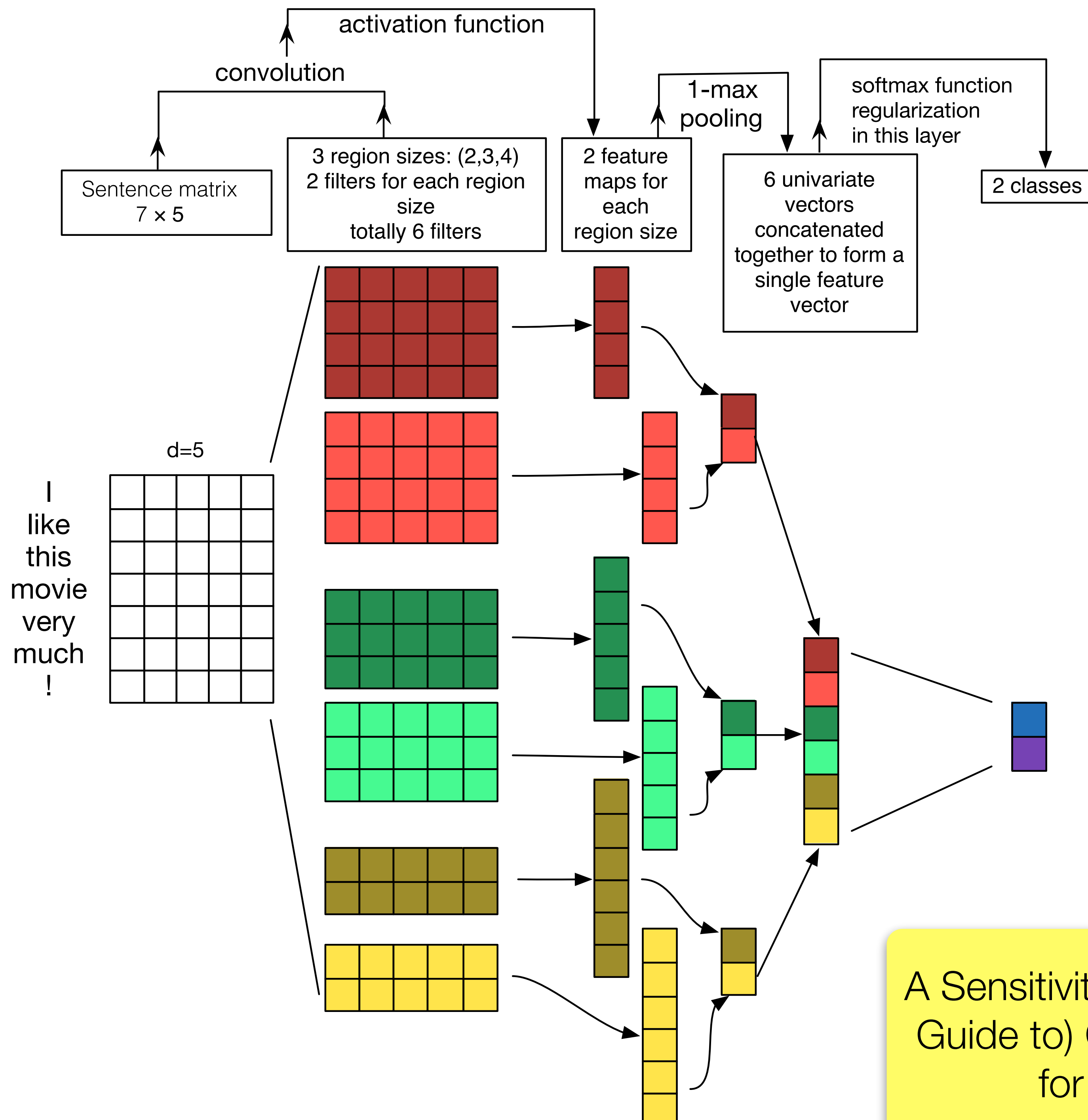
Classification (1/2)



Convolutional Neural Networks for Sentence Classification

Kim, 2014

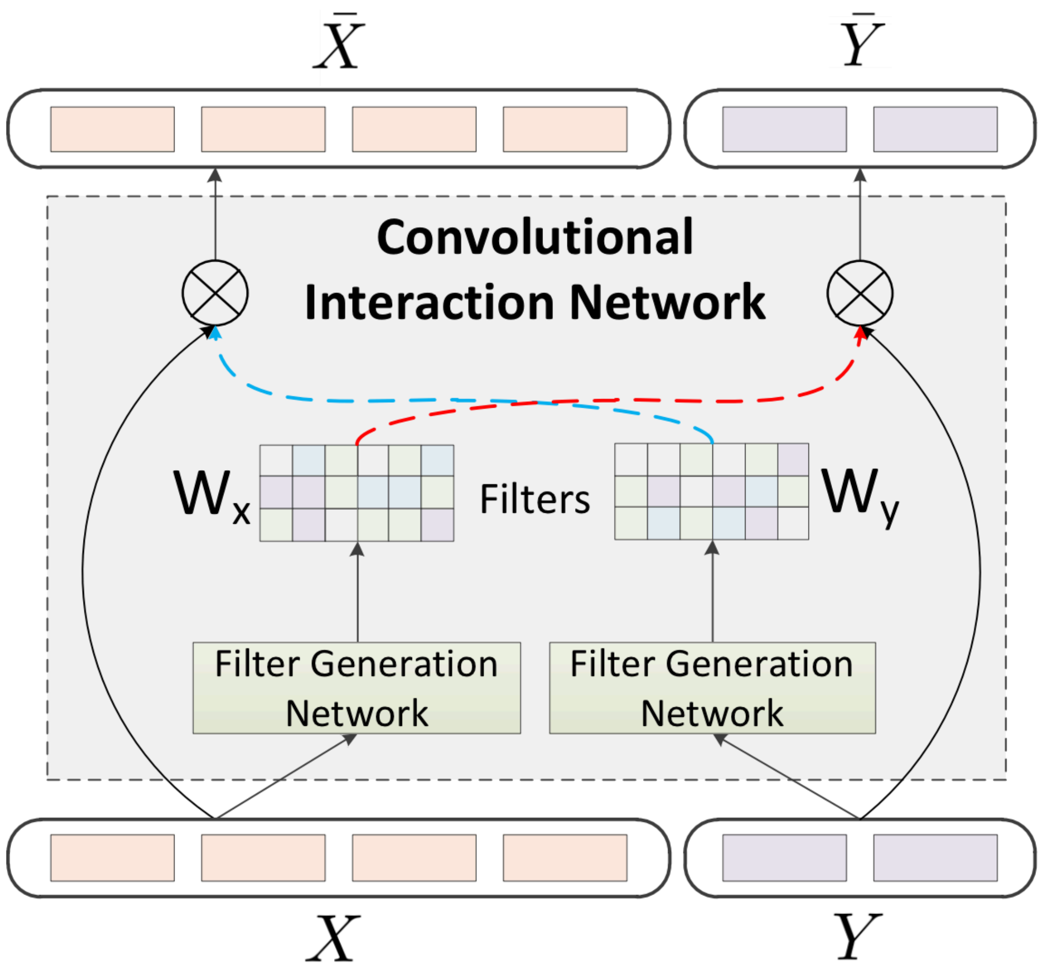
Classification (2/2)

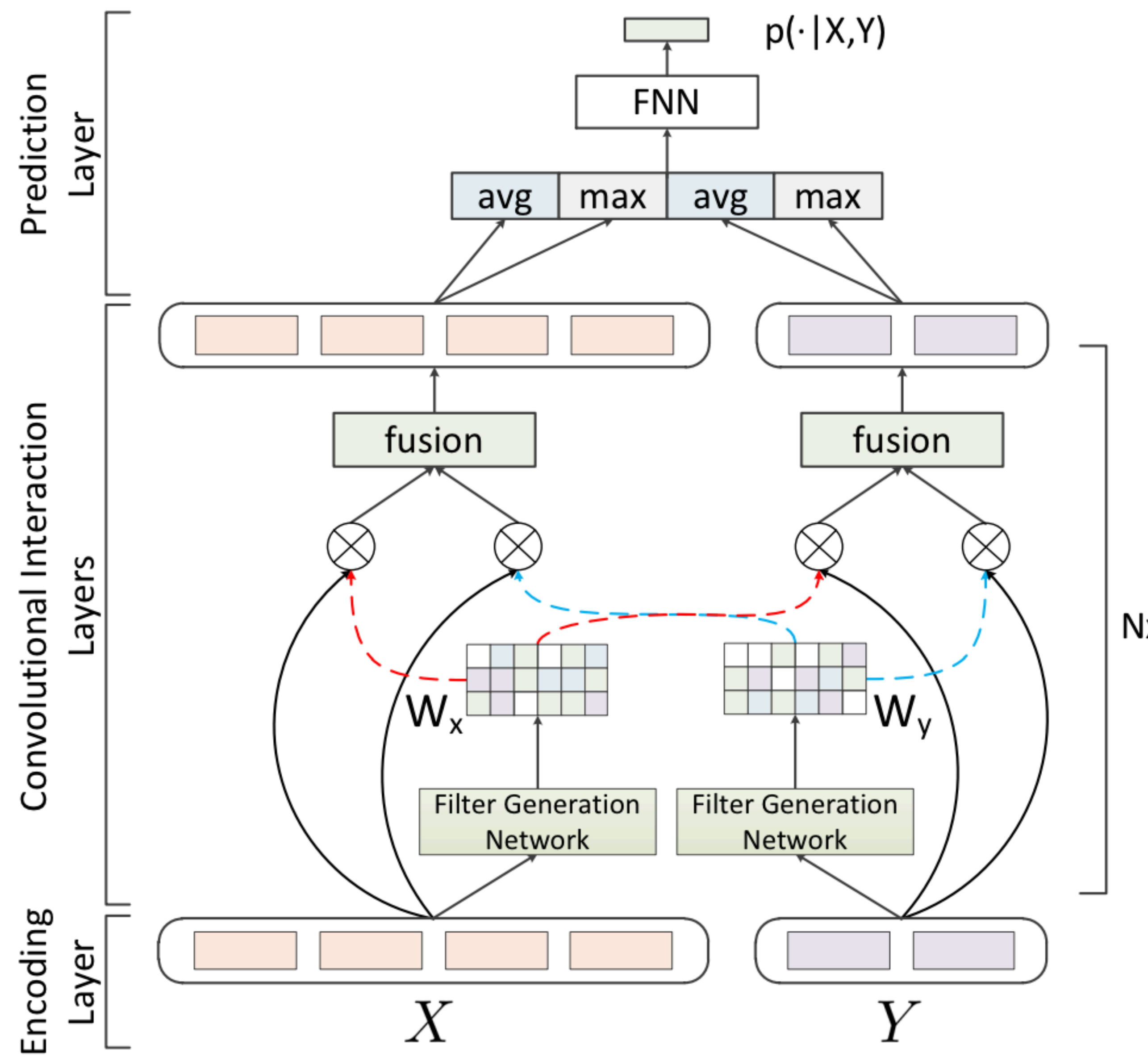


A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

Zhang and Wallace, 2015

Natural Language Inference

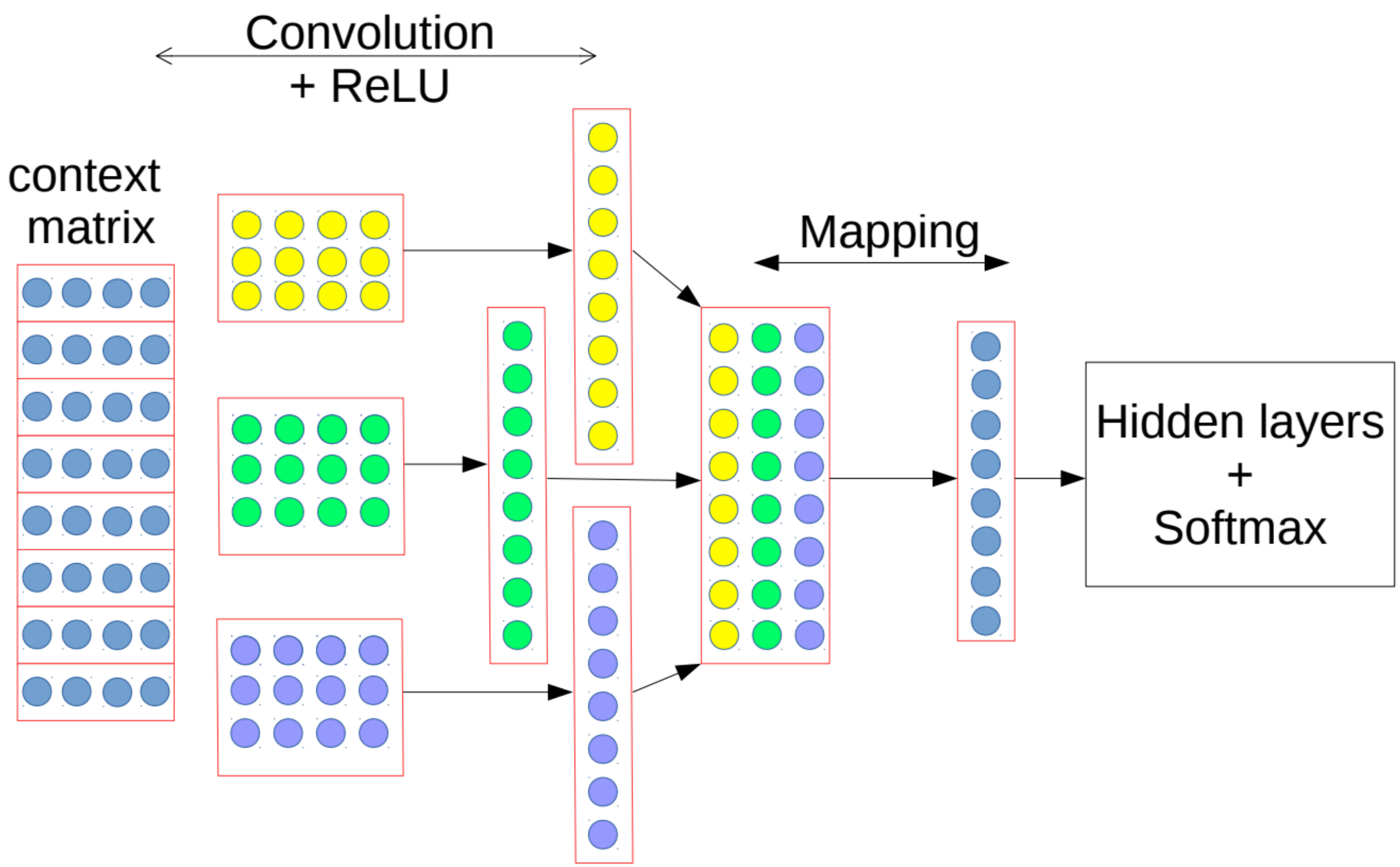




Convolutional Interaction Network for Natural Language Inference

Gong et al., 2018

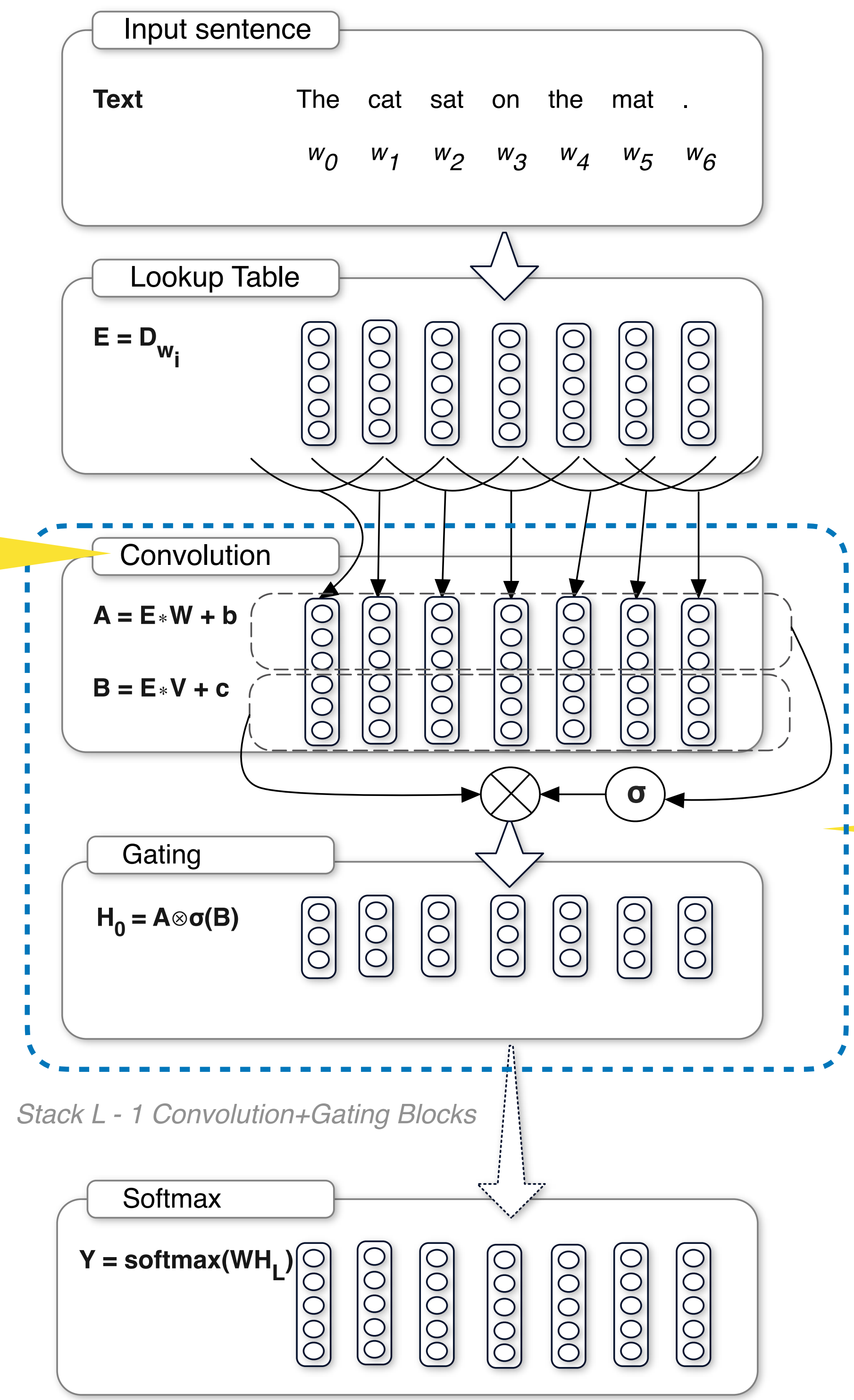
Language modeling (1/2)



Convolutional Neural Network Language Models

Pham et al., 2016

Language modeling (2/2)



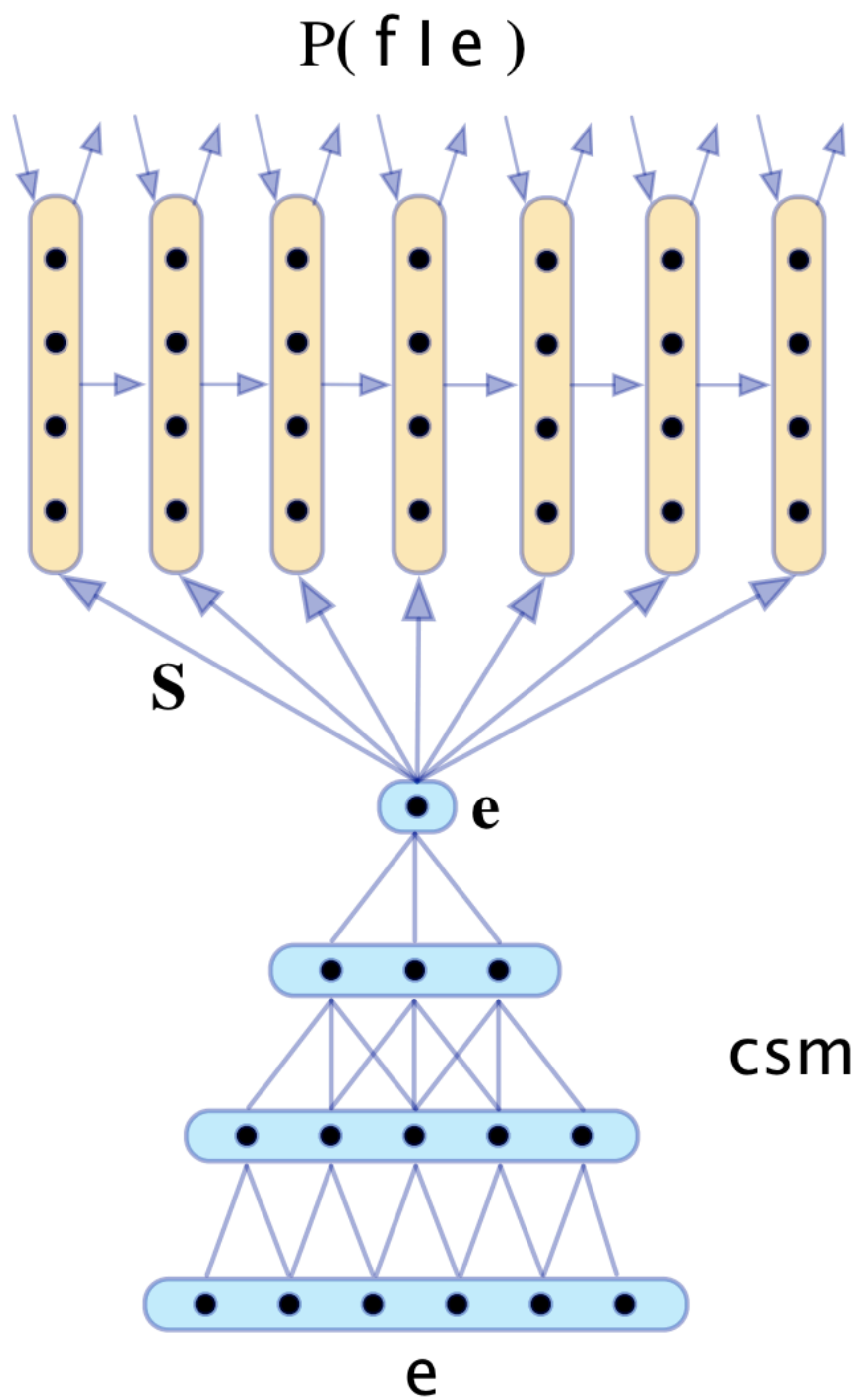
Use of padding to ensure causal dependencies

Gated Linear Unit (GLU)

Language modeling with gated convolutional networks

Dauphin et al., 2016

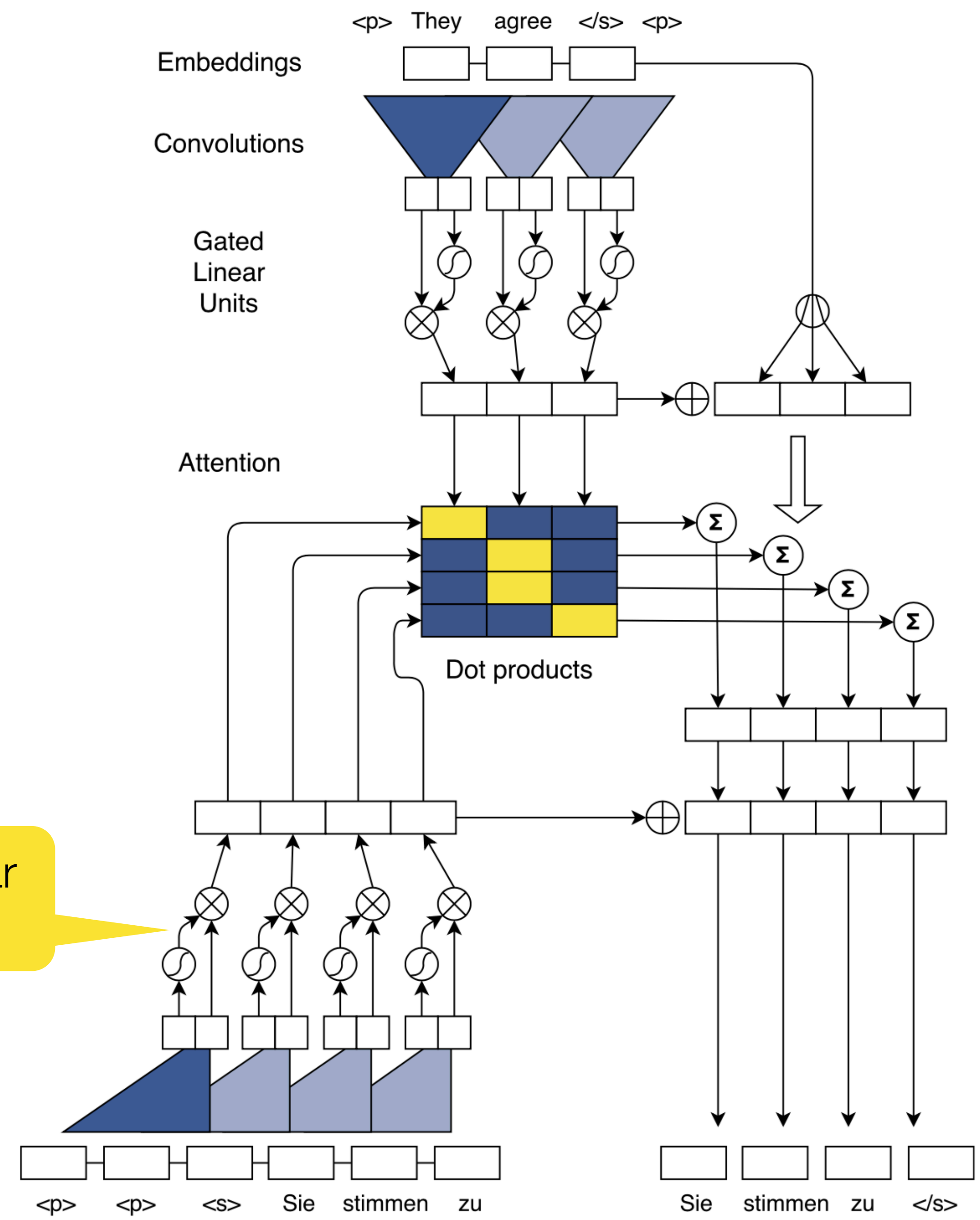
Machine Translation (1/4)



Recurrent Continuous Translation Models

Kalchbrenner and Blunsomet, 2013

Machine Translation (3/4)

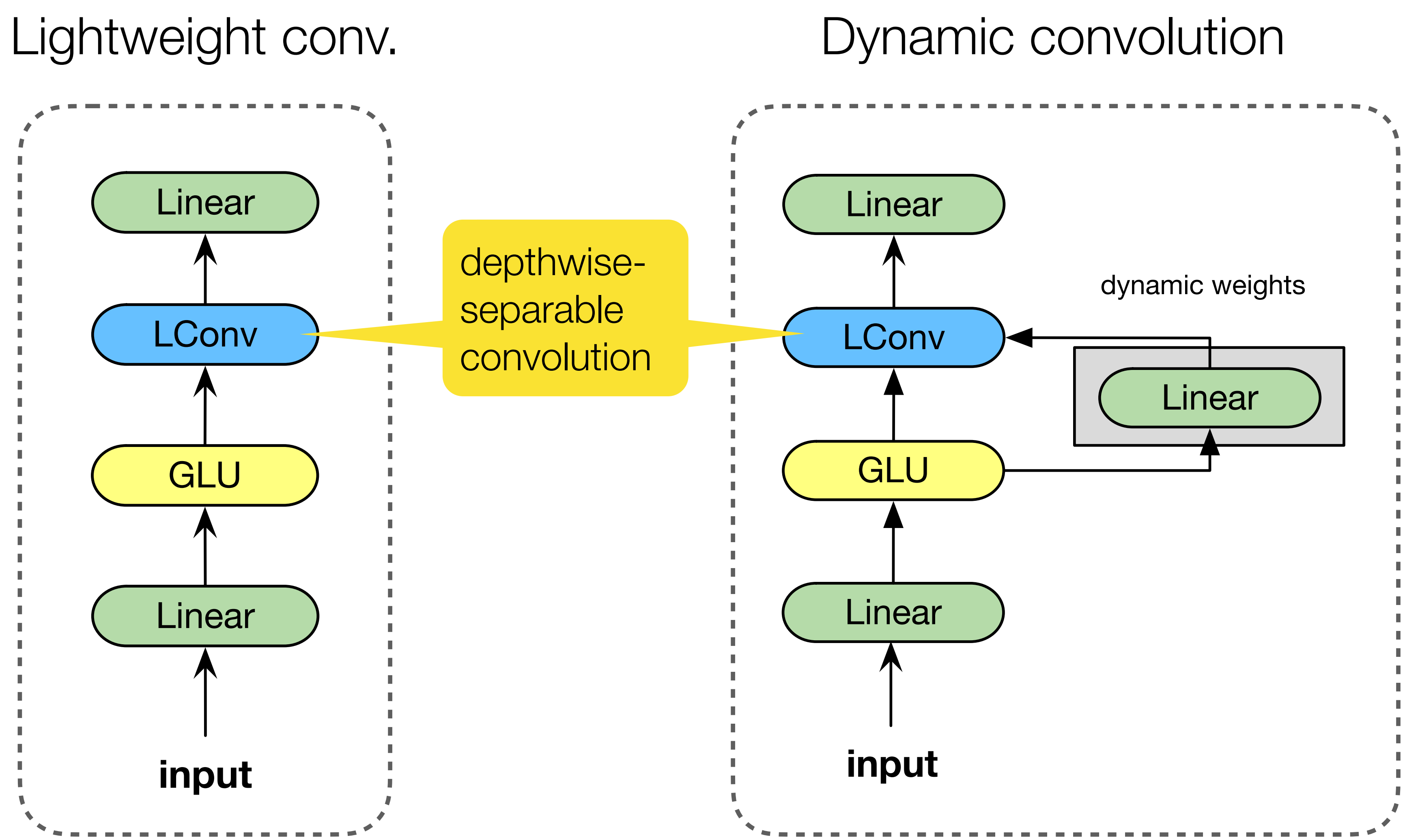


Gated Linear Unit (GLU)

Convolutional Sequence to Sequence Learning

Gehring et al., 2017

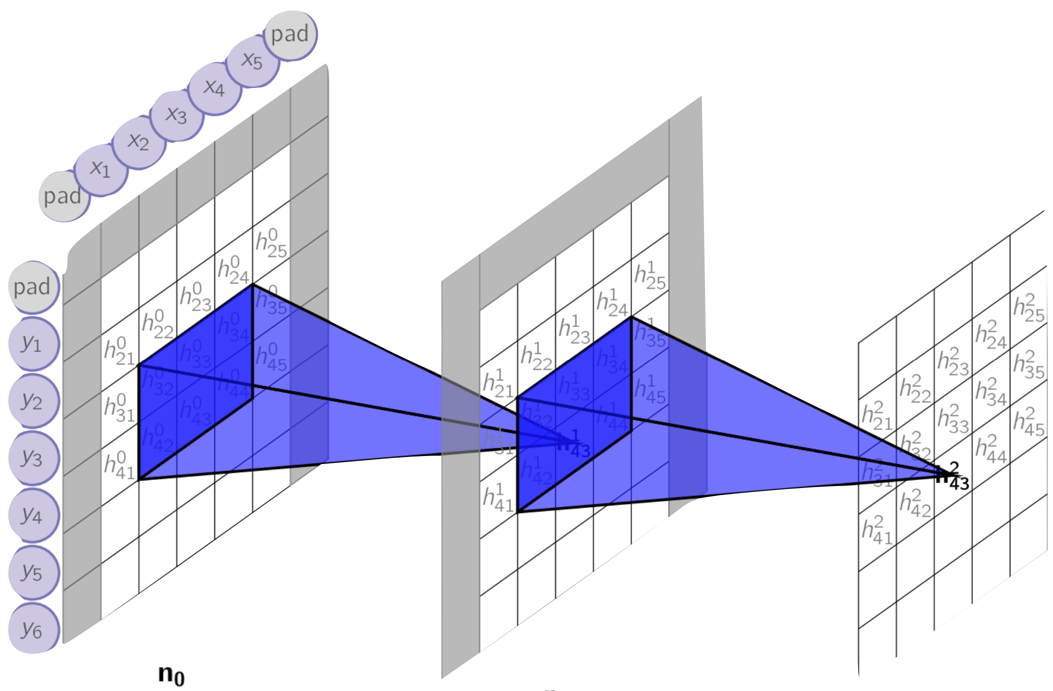
Machine Translation (3/4)

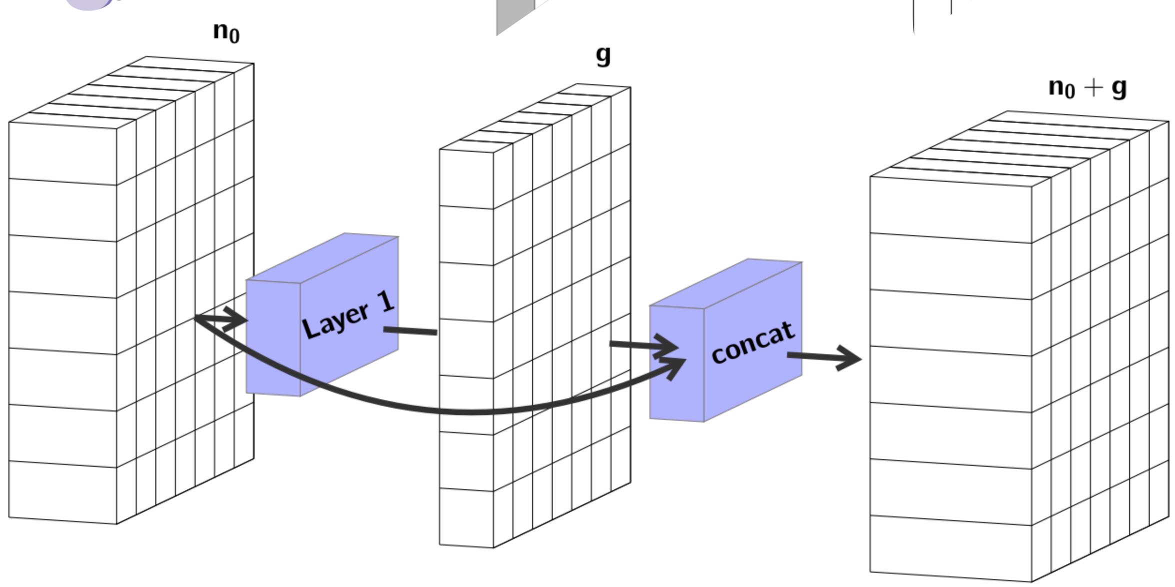


Pay Less Attention with Lightweight and Dynamic Convolutions

Wu et al., 2019

Machine Translation (4/4)





Layer = BN + ReLU + Conv(1) + BN + ReLU + Conv(k)
Growing the input feature maps by $\mathbf{g} = \mathbf{32}$

https://elbayadm.github.io/assets/talks/attn2d/nve_seminar_attn2d.pdf

- 2D convolutions: source x target
- Causality: with masked filters in the target direction.
- Context: grown with stacked convolutions.
- Padding: throughout the network to maintain source/target resolution.
- Each layer grows its input channels by g.

Pervasive Attention - 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction

Elbayad et al., 2018

Summary

- Convolutions are local feature detectors.
- Convolutions are invariant to position.
- Stacked convolutions detect hierarchical features.
- Convolutions are fast.
- Convolutions have been applied to multiple NLP tasks.

References

- (book) *Neural Network Methods for Natural Language Processing Synthesis Lectures on Human Language Technologies*. Yoav Goldberg.

- (online course) Stanford's CS224n: Natural Language Processing with Deep Learning
  http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture11-convnets.pdf