Lecture 11.
Convolution Neural Networks for NLP

CS224n Natural Language Processing with Deep Learning

UOS STAT NLP Study
Changdae Oh

2021. 01. 16

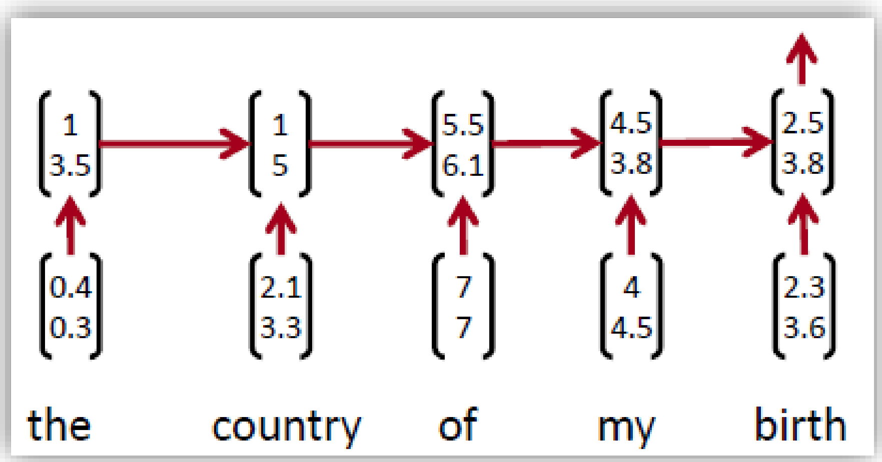
Contents

Contents

From RNN to CNN

Recurrent Neural Networks

- Cannot capture phrases without prefix context
- Often capture too much of last words in final vector



RNN consider the final step' s hidden state as a representation about input sequence !

Bottleneck,,,

Convolution Neural Networks

- What if we compute vectors for every possible word subsequence of a certain length?
- And calculate a representation for it
- Another solution to bottleneck problem !

For NLP

1) Can get N-gram features
2) Capture patterns locally
3) Fastet than RNN

Contents

Introduction to CNN

What is a convolution ?

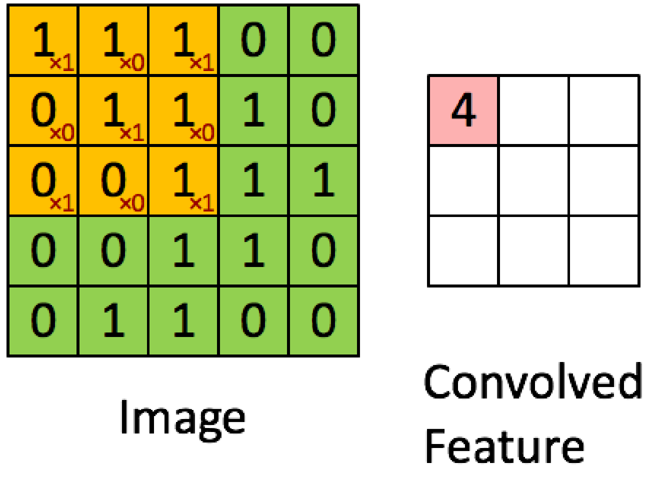
Convolution is classically used
to **extract features** from images

- Sliding window function applied to a matrix
- The sliding window is called a filter (also kernel)
- Use n*n filter, multiply its values element-wise with the original matrix, then sum them up

< Various terms used on CNN >

- Channel
- Filter
- Kernel
- Stride
- Padding
- Pooling
- Feature Map
- Activation Map

For image



http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

For text

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

Introduction to CNN

1D convolution for text

(zero) padding

∅	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
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,∅	−0.5

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

3	1	2	−3
−1	2	1	−3
1	1	−1	1

- Adjust the size of output sequence
- Preserve dimensions of data flowing across the layer

Multiple filters

∅	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

- Increase output channels
- Capture various features
- The filters to be specialized for different domain

pooling

∅,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
ave p	−0.87	0.26	0.53

- Summarize the output of convolution
- Capture sparse signal
- Subsampling(down-sampling)
- Give some invariance to fluctuation of inputs

Introduction to CNN

Other notions

increase stride

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
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

Compress data

local max pooling

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

Compress representation

Focus on
more local characteristics

k-max pooling

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

Compress representation

Keep more information
than 1-max pooling

dilated convolution

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

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

2	3	1
1	−1	−1
3	1	0

1	3	1
1	−1	−1
3	1	−1

Compress data

Use a relatively small number of parameters to have a larger receptive field

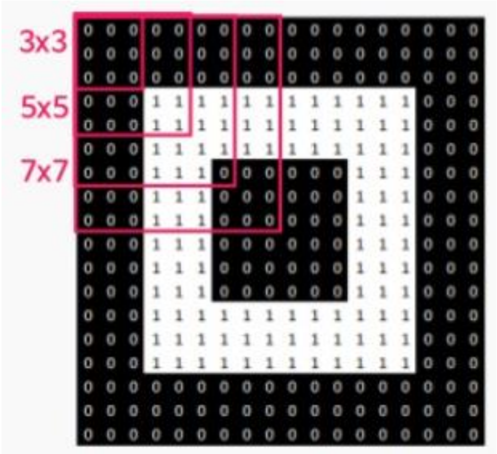
Introduction to CNN

Tools to see a wider range at once

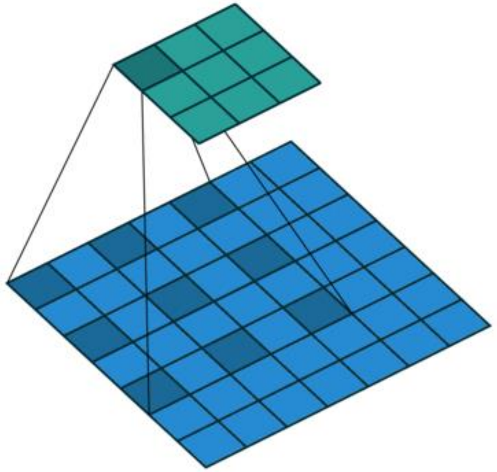
1) Use bigger filters

2) Dilated convolution

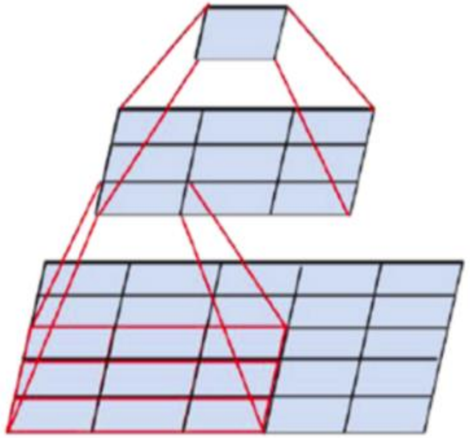
3) Make CNNs deeper



•https://www.slideshare.net/modulabs/2-cnn-rnn



https://zzsza.github.io/data/2018/02/23/introduction-convolution/



https://zzsza.github.io/data/2018/05/25/cs231n-cnn-architectures/

Contents

Simple CNN for Sentence Classification

Yoon Kim (2014) https://arxiv.org/pdf/1408.5882.pdf

- Goal : Sentence Classification
- A simple use of one convolutional layer and pooling

Notation

- Word vectors: $\mathbf{x}_i \in \mathbb{R}^k$
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus x_2 \oplus \cdots \oplus \mathbf{x}_n$ (vectors concatenated)
- Concatenation of words in range: $\mathbf{x}_{i:i+j}$ (symmetric more common)
- Convolutional filter: $\mathbf{w} \in \mathbb{R}^{hk}$ (over window of *h* words)
- All possible windows of length *h*: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$

Feature extraction

$$c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$$

Note,

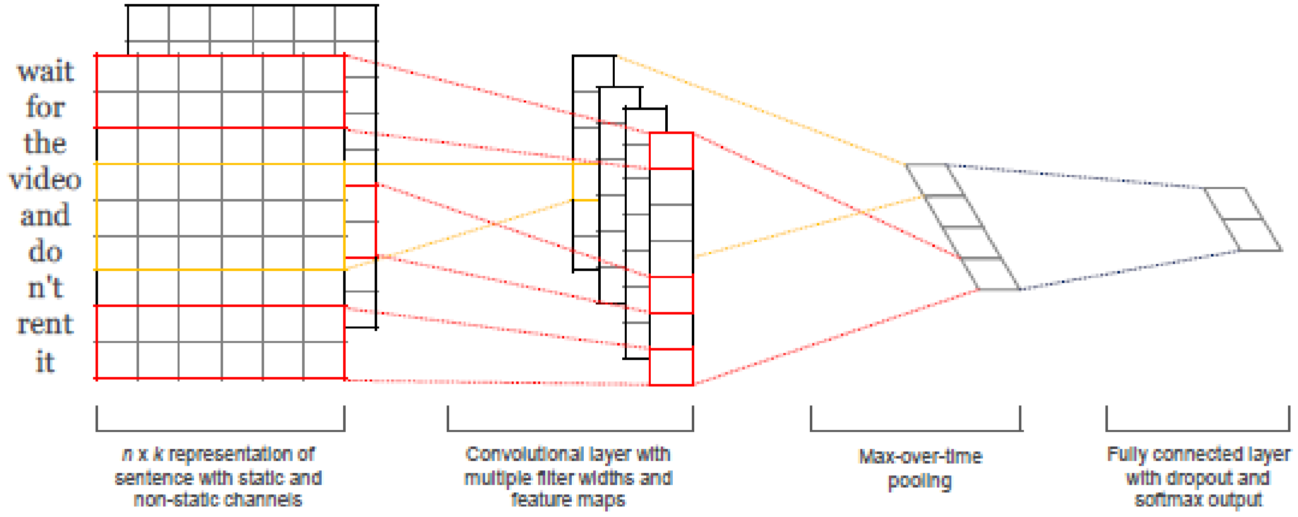
filter is a vector (all inputs and filters are flatten)

Simple CNN for Sentence Classification

Max-over-time pooling

$$\hat{c} = \max\{\mathbf{c}\}$$

Take only one maximum value as the feature from each feature maps



Use multiple filters(100) with various widths

$$y = softmax\left(W^{(S)}z + b\right)$$

- Different window sizes h(3, 4, 5) -> look at various n-grams
- Can represent lots of domain features

Multi-channel input idea

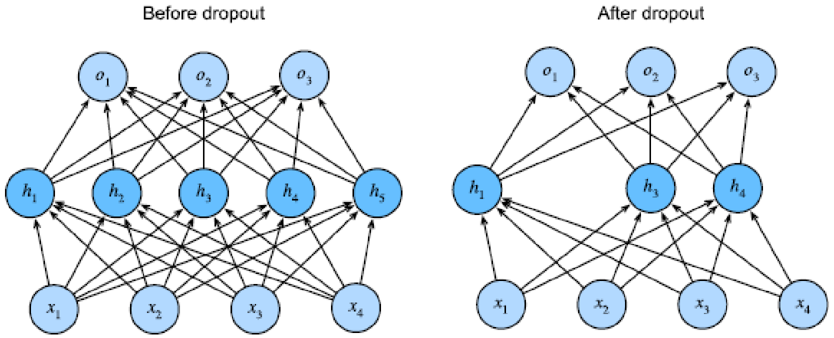
- Having **two channels of word vectors**
  -- one that is kept static throughout training and one that is fine-tuned via backpropagation
- Both channel sets are added to feature map before max-pooling

Simple CNN for Sentence Classification

Regularization

Dropout

$$y = softmax\left(W^{(S)}(r \circ z) + b\right)$$



https://d2l.ai/

- Masking vector **r** of Bernoulli r.v. with prob. **p** of being 1
- Delete feature during training -> prevents co-adaptation / overfitting
- At testing, scale final vector by keep probability **p**

$$\hat{W}^{(S)} = pW^{(S)}$$

Max-norm regularization

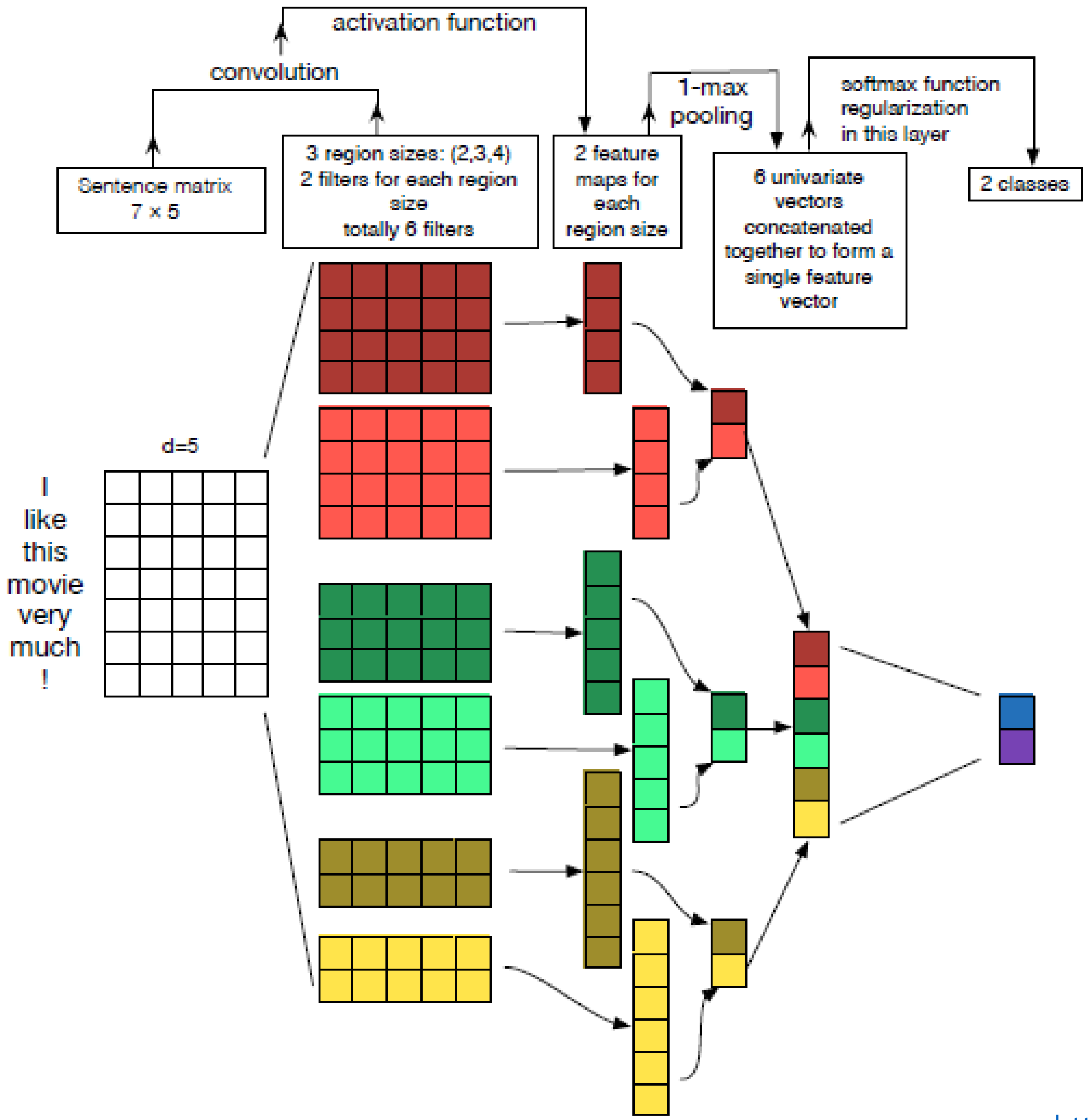
$$\| w \|_2 \leq s$$

$$w \leftarrow w \frac{s}{\| w \|_2}$$

- Constrain l2 norms of weights vectors of each class to fixed threshold value **s**
- prevents overfitting

Simple CNN for Sentence Classification

Model Architecture



https://arxiv.org/pdf/1510.03820.pdf,

Simple CNN for Sentence Classification

Conclusion

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**	–	–

the model is very simple,
but it' s quite powerful !

Contents

Toolkit & ideas for NLP task

Model comparison

Good baseline method !

Bag of Vectors

The average of word embeddings in sentence can be the vector of that sentence
Text classification can be performed by just average of word vectors.

Window Model

Good for single word classification for problems that do not need wide context (POS, NER)

Advanced Model

CNNs

- Good for representing sentence meaning -> sentence classification
- Easy to parallelize on GPUs.
- Efficient and versatile

RNNs

- Cognitively plausible, good for sequence tagging / classification
- Great for language models
- Can be amazing with attention mechanisms
- Much slower than CNNs

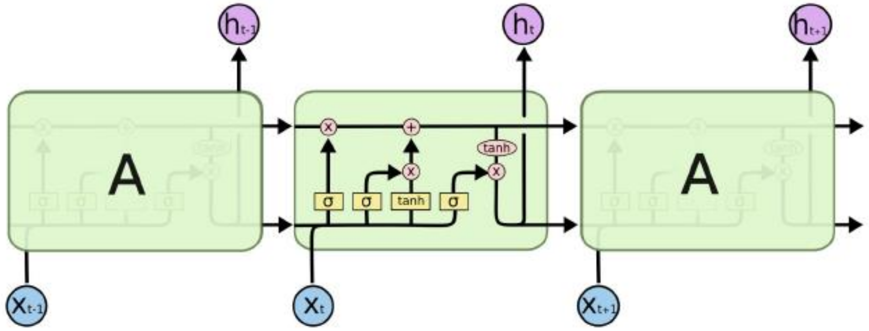
Toolkit & ideas for NLP task

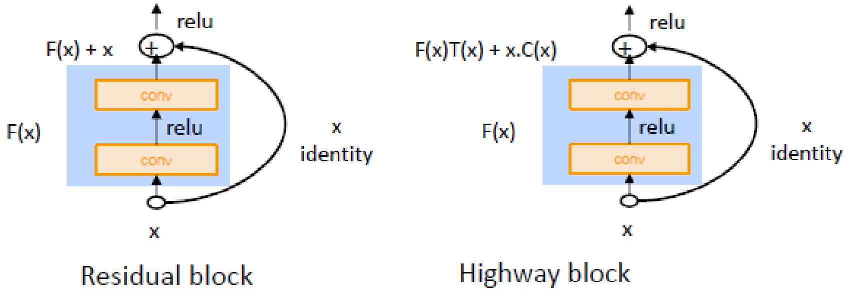
Gated units used vertically

- The gating in LSTMs / GRUs is a general idea
- Can also gate vertically (in CNNs, deeps FNN)

Gating = Skipping

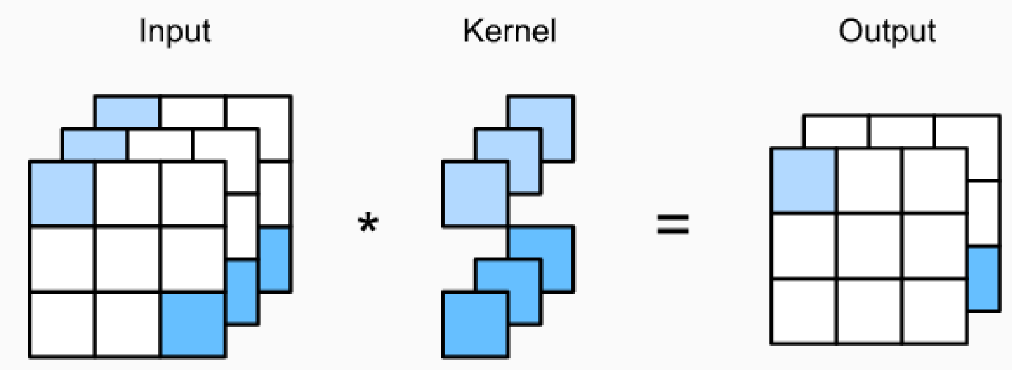
- Add shortcut connection
- Can stack many layers without gradient vanishing





1 x 1 convolutions

- Convolution with kernels _ size : 1
- Add additional neural network layers with very few additional parameters
- Can be used to map from many channels to fewer channels (preserves spatial dimensions, reduces depth)



http://d2l.ai/chapter_convolutional-neural-networks/channels.html

Toolkit & ideas for NLP task

Batch Normalization

- It is widely used in CNNs and Deep FNNs
- In CNNs, transform the convolution output of a batch

  by scaling the activations to have zero mean and unit variance (Z−transform)

advantage

- Learn faster
- Less sensitive to initialization
- Prevent overfitting

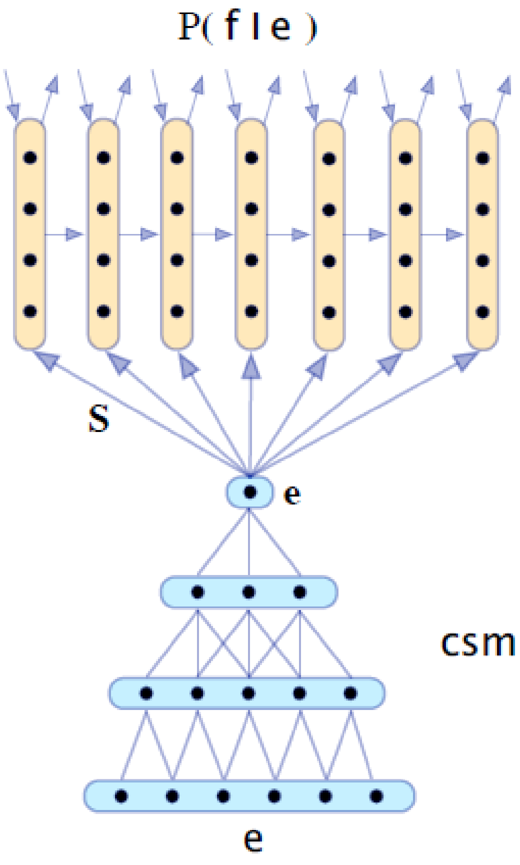
disadvantage

- Increase complexity
- Computing resource

1. $\boldsymbol{\mu}_B = \frac{1}{m_B}\sum_{i=1}^{m_B}\mathbf{x}^{(i)}$
2. $\boldsymbol{\sigma}_B{}^2 = \frac{1}{m_B}\sum_{i=1}^{m_B}\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_B\right)^2$
3. $\hat{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \boldsymbol{\mu}_B}{\sqrt{\boldsymbol{\sigma}_B{}^2 + \varepsilon}}$
4. $\mathbf{z}^{(i)} = \boldsymbol{\gamma} \otimes \hat{\mathbf{x}}^{(i)} + \boldsymbol{\beta}$

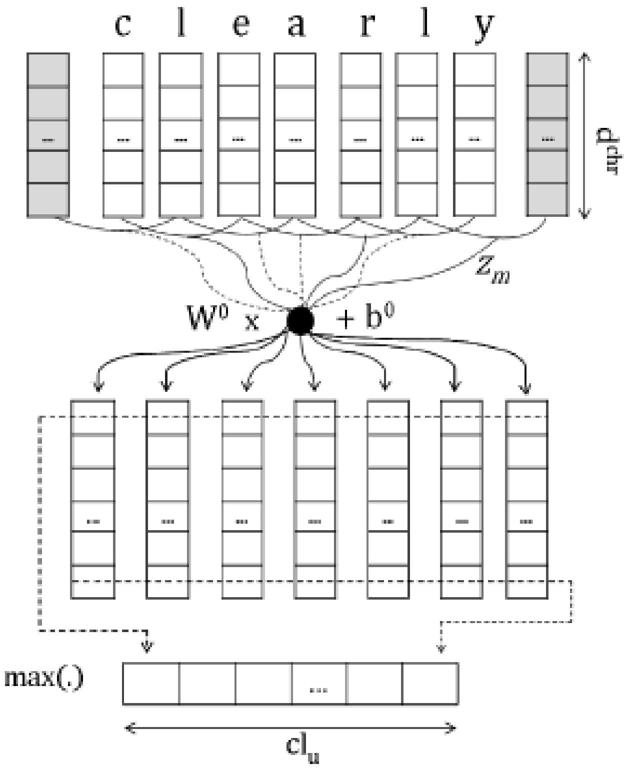
Toolkit & ideas for NLP task

Encoder(CNN)-Decoder(RNN) networks



- Shrink the input data continuously by stacking up convolutional layers
- Take the final pulled vector as a sentence representation

Character-level representations



NEXT CHAPTER

Contents

Deep CNN for Sentence Classification

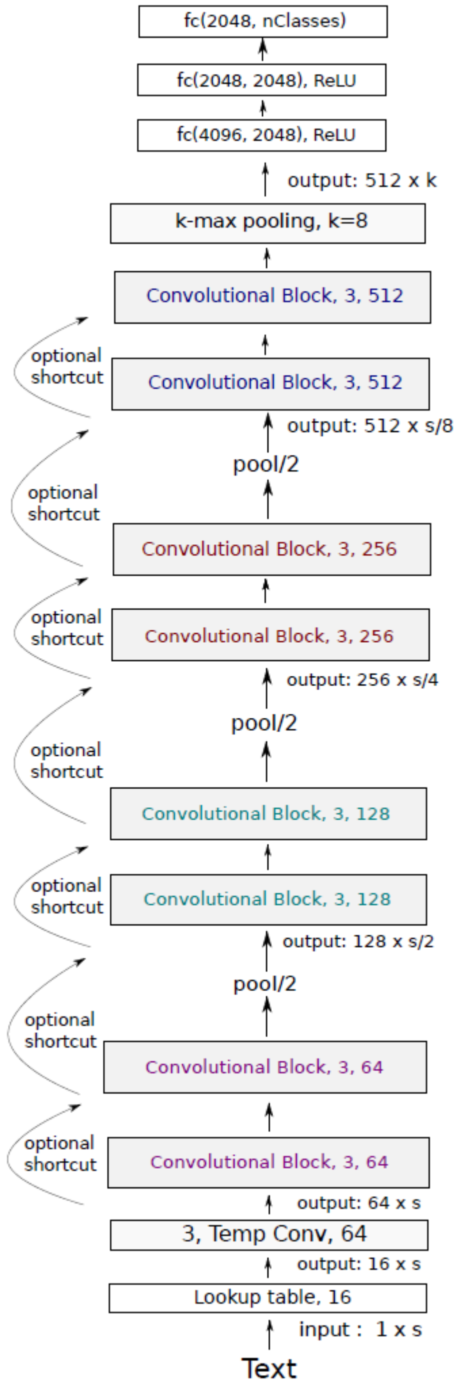
Conneau, Schwenk, Lecun, Barrault. EACL 2017 https://arxiv.org/abs/1606.01781

VeryDeep-CNN

- Starting point : sequence models have been very dominant in NLP
  but all the models are basically not very deep

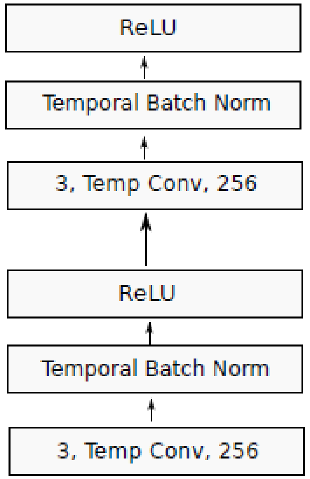
Note

- Build a vision-like system for NLP
- Works from the character-level
- Result is constant size, since text is truncated or padded
- Local max pooling



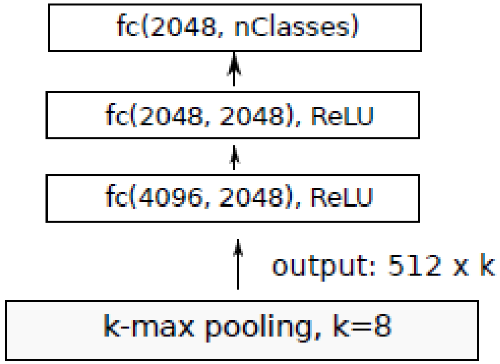
Deep CNN for Sentence Classification

Convolutional block



- Two conv. Layers, each one followed by a temporal BN and an ReLU activation
- Padding for preservation of temporal resolution
- Use small size filters (3) so that networks can be deep and get more expressive in a few parameters

Classification tasks



- Temporal resolution of the output is first **down-sampled** to a fixed dimension **using k-max pooling** (extracts the k most important features indep. position)
- the 512 x k resulting features are transformed into a single vector (Flatten) which is the input to a three FC layer with ReLU
- The number of output neurons in there depends on clf task

Deep CNN for Sentence Classification

Conclusion

Depth	Pooling	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
9	Convolution	10.17	4.22	1.64	5.01	37.63	28.10	38.52	4.94
9	KMaxPooling	9.83	3.58	1.56	5.27	38.04	28.24	39.19	5.69
9	MaxPooling	9.17	3.70	1.35	4.88	36.73	27.60	37.95	4.70
17	Convolution	9.29	3.94	1.42	4.96	36.10	27.35	37.50	4.53
17	KMaxPooling	9.39	3.51	1.61	5.05	37.41	28.25	38.81	5.43
17	MaxPooling	8.88	3.54	1.40	4.50	36.07	27.51	37.39	4.41
29	Convolution	9.36	3.61	1.36	4.35	**35.28**	27.17	37.58	**4.28**
29	KMaxPooling	**8.67**	**3.18**	1.41	4.63	37.00	27.16	38.39	4.94
29	MaxPooling	8.73	3.36	**1.29**	**4.28**	35.74	**26.57**	**37.00**	4.31

- ✓ deeper networks are better
- ✓ However, if it is too deep, its performance will be degraded
- ✓ Shrinkage method – "MaxPooling" is easier

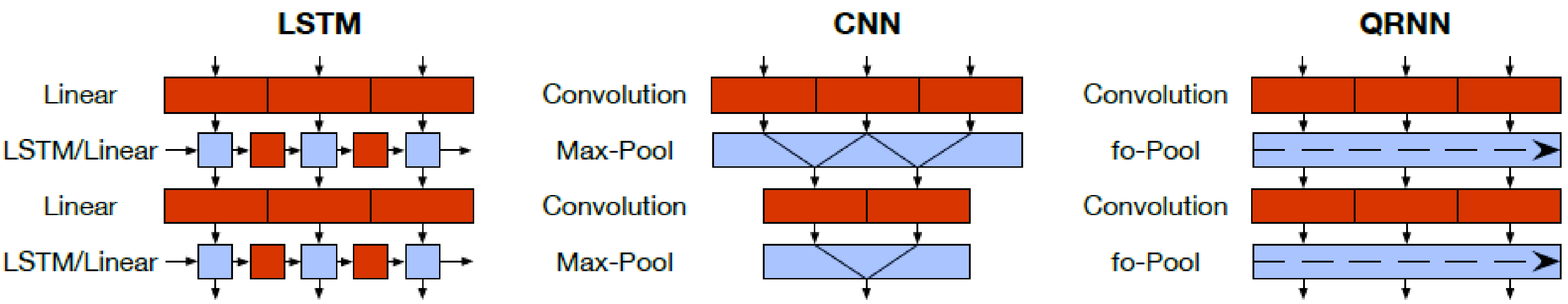
Contents

Quasi-recurrent Neural Networks

https://arxiv.org/abs/1611.01576

QRNNs : have the advantages of both CNN and RNN

- Parallel computation across both timestep & minibatch dimensions
- Output depend on the overall order of elements in the sequence



LSTM notation

$$
\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_{xf}^T \cdot \mathbf{x}_t + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_{xi}^T \cdot \mathbf{x}_t + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_{xo}^T \cdot \mathbf{x}_t + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right) \\
\mathbf{g}_t &= \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_g\right) \\
\mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \otimes \tanh(\mathbf{c}_t)
\end{aligned}
$$

QRNN notation

$$
\begin{aligned}
z_t &= tanh(W_z^1 x_{t-k+1} + W_z^2 x_{t-k+2} + \cdots + W_z^k x_t) \\
f_t &= \sigma(W_f^1 x_{t-k+1} + W_f^2 x_{t-k+2} + \cdots + W_f^k x_t) \\
o_t &= \sigma(W_o^1 x_{t-k+1} + W_o^2 x_{t-k+2} + \cdots + W_o^k x_t)
\end{aligned}
$$

$$
\Rightarrow \quad
\begin{aligned}
\mathbf{Z} &= \tanh(\mathbf{W}_z * \mathbf{X}) \\
\mathbf{F} &= \sigma(\mathbf{W}_f * \mathbf{X}) \\
\mathbf{O} &= \sigma(\mathbf{W}_o * \mathbf{X})
\end{aligned}
$$

(*) Denotes a masked convolution along the timestep dimension

"Dynamic average pooling"

$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t$$

function controlled by gates
that can **mix states across timesteps**,
but which acts independently
on each channel of the state vector

Parallelism across channels

Parallelism across timesteps

Reference

Kim, Yoon. "Convolutional neural networks for sentence classification." *arXiv preprint arXiv:1408.5882* (2014).

Zhang, Ye, and Byron Wallace. "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification." *arXiv preprint arXiv:1510.03820* (2015).

Conneau, Alexis, et al. "Very deep convolutional networks for text classification." *arXiv preprint arXiv:1606.01781* (2016).

Bradbury, J., et al. "Quasi-recurrent neural networks. arXiv 2016." *arXiv preprint arXiv:1611.01576*.

Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." *Advances in neural information processing systems* 28 (2015): 649-657.

https://zzsza.github.io/data/2018/05/25/cs231n-cnn-architectures/

https://www.slideshare.net/modulabs/2-cnn-rnn

http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

https://happyzipsa.tistory.com/10?category=748723

http://taewan.kim/post/cnn/

https://wikidocs.net/103496