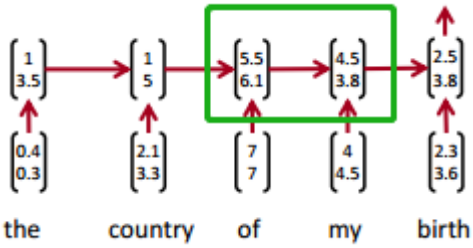
Lecture11 ConvNets for NLP

---

from RNNs to CNNs

RNN在没有上下文的情况下不能捕获短语；不能很好的并行化；速度很慢
最后一个向量表示被它最近的单词代表的意思支配

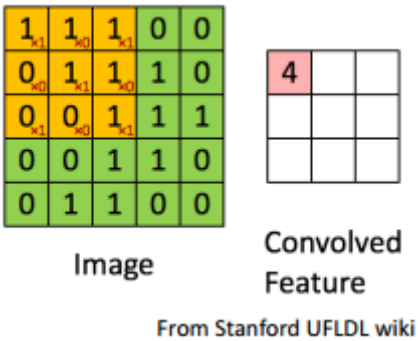


Convolutional Neural Nets
取一定长度的每个子序列并计算它的表示
例如： “tentative deal reached to keep government open” 计算的向量为tentative deal reached, deal reached to, reached to keep, to keep government, keep government open
然后对他们分组

什么是卷积？

$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$$

一维离散卷积：
经典使用于从图片中提取特征

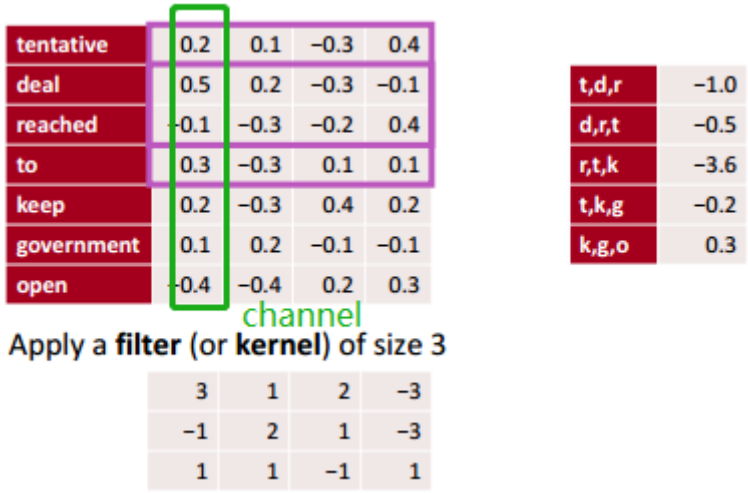


From Stanford UFLDL wiki

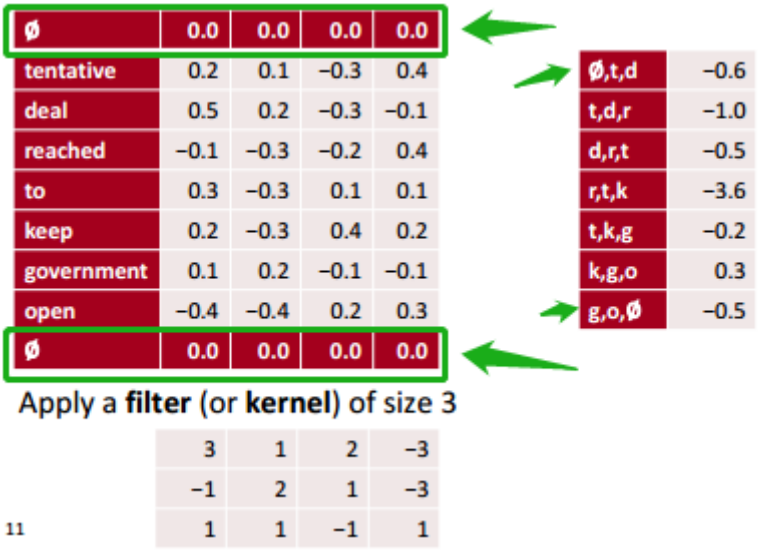
二维离散卷积例子：
每个补丁（可看作是向量）内的红字与黑字相乘后相加得到右侧卷积特征中粉格中的数字

文本的卷积神经网络

- 有一个输入，对于句中的每一个单词也有一个密集的单词向量，每一个竖着的维度成为信道（channel）

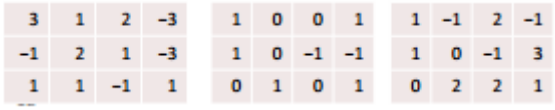


- size为3的filter，会做三个步骤和时间，三个字。得到了如图这个补丁
- 将这个补丁与前三行（t,d,r）点乘，得到乘积为-1.0
- 将这个过滤器向下滑动并在此执行这些单元的点积，得到图中右边部分
- 这样就将一个句子减小到一个向量。但可以看出句子有所缩小，最开始有七个单词，却得到五个位置。解决这一问题，可以在两端增加零填充，然后对此进行卷积，得到一个与输入长度相同的向量



11

- 在列方向上。输入有四个信道而输出只有一个信道。因此使用的三个不同但大小相同的filter，



运行文本中的每一个信道得到一列输出，最终得到三列输出

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
max p	0.3	1.6	1.4

- 针对这些filter，可以通过反向传播来学习它们，但希望这些filter能以某种方式专注于不同的事物（是否是礼貌用语、是否在讨论某种内容等），将得到不同功能的输出，即从文本中获得不同的潜在特征。
- 总结一个CNN的输出，适用于一维最简单的方法被称为是 **max pooling over time** 最大池化随时间的推移。
  找出每列输出信道的最大值

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
max p	0.3	1.6	1.4

  例如前两列，不是很礼貌的文字，但是真的是关于事物
- 如果文本中出现相关的词汇，最大池的输出将具有高值
- 也可以使用 **average pooling**（平均合并），只需要取这些数字的平均值。某些目的下，average pooling更好，因为它考虑了所有值

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

average pooling

ave p	−0.87	0.26	0.53

  很多时候max pooling更好，因为很多自然语言的信号很稀疏，max pooling能更好的捕获到。
- min pooling来捕获很少活跃的词

In PyThon

```
batch_size = 16
word_embed_size = 4
seq_len = 7
input = torch.randn(batch_size,
word_embed_size, seq_len)
conv1 = Conv1d(in_channels=word_embed_size,
out_channels=3,
kernel_size=3) # can add: padding=1
hidden1 = conv1(input)
hidden2 = torch.max(hidden1, dim=2) #
max pool
```

- 每次滑动是一个步幅（one stride）；
  使用两个步幅，依然能包含其中的一个单词，就可以做一半计算，得到原输出一半的行数，使representation更为紧凑。还有其他方法可以减少从句子中删除表示的内容
- **local pooling**
  取每两行进行最大池化得到下面的

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

  就得到一个 local max pooling的两个步幅。
- **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

  不是按大到小的顺序排列，而是按它们在这些列中的顺序排列

- 另一种压缩数据的方法——扩张的卷积
  跳过一些行，例如选取1/3/5行点乘新的filter，获得一个新的输出

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

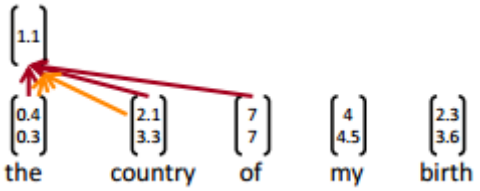
1,3,5	0.3	0.0
2,4,6		
3,5,7		

2	3	1
1	−1	−1
3	1	0

1	3	1
1	−1	−1
3	1	−1

Single Layer CNN for Sentence Classification

- 单句的文本分类
  Yoon Kim (2014): Convolutional Neural
  Networks for Sentence Classification. EMNLP 2014.
  https://arxiv.org/pdf/1408.5882.pdf
  Code: https://arxiv.org/pdf/1408.5882.pdf
  - A simple use of one convolutional layer and **pooling**
  - 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)
  - Note, filter is a vector!
  - Filter could be of size 2, 3, or 4:
- 从长度为k的单词向量开始，通过将所有这些单词向量连接在一起来构成句子， 这一系列的单词是该句子向量的子部分。（详细见图）
- 使用不同大小的卷积

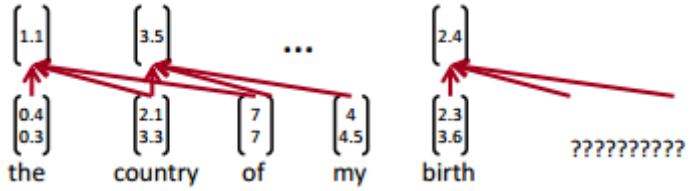


- 计算一个信道的特征

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

得到：

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$$



在进行 max-pooling

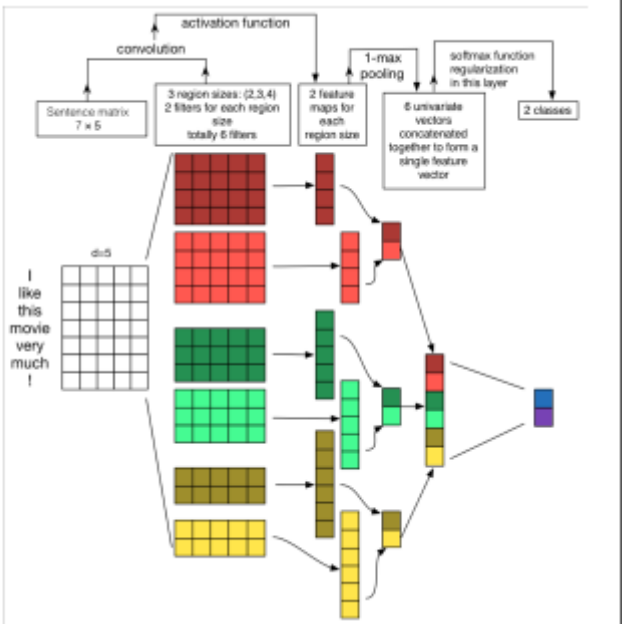
- 使用预先训练过的单词向量（将单词向量copy两组），每个单词都有两个word vector，因此有两个通道。将一组冻结，另一组在训练中调整。将这两组进行 max pooling操作。
- 最大池化之后，为每一个通道输出一个数字，最终结果通过softmax，给出分类结果

From:

Zhang and Wallace (2015) A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

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

(follow on paper, not famous, but a nice picture)



- 正则化。利用dropout来缩放权重矩阵
- softmax的权重矩阵。限制了L2范式不超过一个超参数s。
- 一些超参数设置：

```
  Find hyperparameters based on dev set
• Nonlinearity: ReLU
• Window filter sizes h = 3, 4, 5
• Each filter size has 100 feature maps
• Dropout p = 0.5
    • Kim (2014) reports 2 - 4% accuracy
improvement from dropout
• L2 constraint s for rows of softmax, s =
3
• Mini batch size for SGD training: 50
• Word vectors: pre-trained with
word2vec, k = 300
```

- 实验结果

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**	–	–

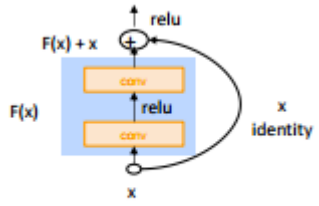
  - 一些对比模型是在dropout出现之前的结果，因此对比试验有些小问题
  - 是很强大的文本分类器

模型对比

RNN：序列标记和分类、结合注意力机制等。但是比CNN慢

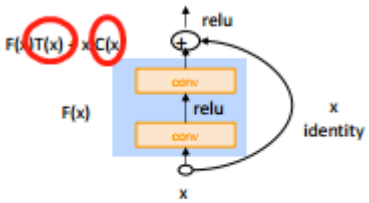
垂直的门控单元

- 残差块，用于残差网络。对于每个区块，允许一个只跳到下一层。在进行相加前要进行填充，是它们保持相同的大小



Residual block
(He et al. ECCV 2016)

- 更加复杂



Highway block
(Srivistava et al. NeurIPS 2015)

Batch Normalization批量标准化

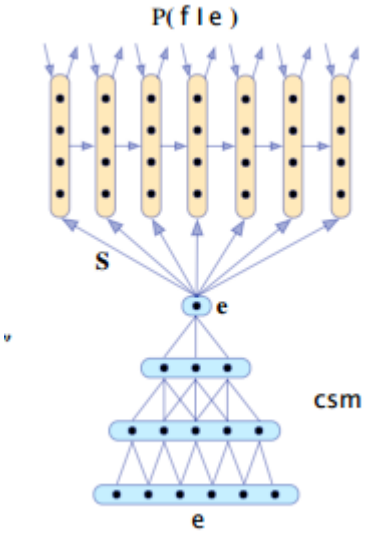
- Ioffe and Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.
- Z变换：将批量的卷积输出按比例变换为零均值和单位方差
  通过一层神经网络，获取这些mini batch的输出，然后进行Z变换，之后再经过一层神经网络，依次对输出做Z变换。
  没有那么多的波动，可以使模型更可靠的训练

1x1 Convolutions

- 内核大小为1的卷积kernel，充当信道上的一个小型嵌入式全连接网络。
- 为每行的数据做位置特定的全连接网络，可以从多个信道映射到较少的信道
- 涉及很多参数，而一个卷积涉及很少的参数，因为只在一个单词的水平上做。

CNN应用：翻译

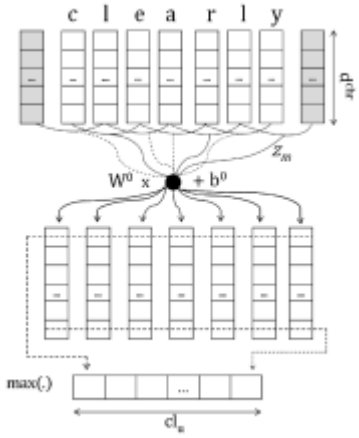
- 神经机器翻译
  Kalchbrenner and Blunsom (2013)
  “Recurrent Continuous Translation Models”



- 编码器使用CNN，如图，有一堆逐渐输入缩小的卷积神经网络得到句子表示，然后使用一个sequence model作为解码器。

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

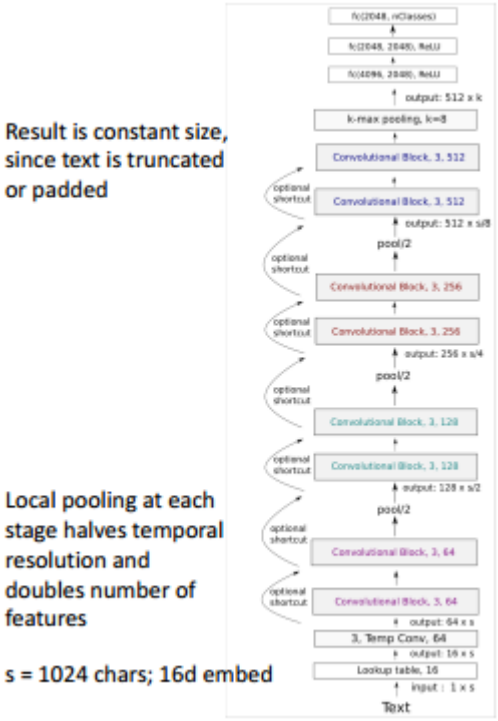
Dos Santos and Zadrozny (2014)
对字符使用CNN生成一个单词嵌入



Very Deep Convolutional Networks for Text Classification

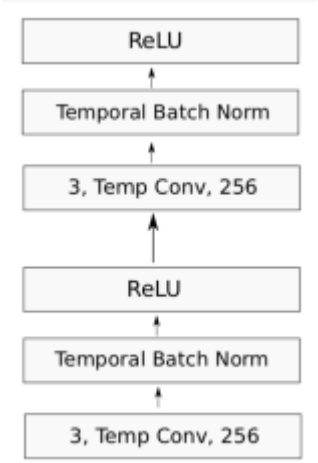
Conneau, Schwenk, Lecun, Barrault.
EACL 2017
建立在字符上，一个真正的深层神经网络

VD-CNN architecture



character embedding16维

卷积块



图中有一个大小为三的卷积块，根据在序列中的位置，对一些通道进行卷积，然后通过一个batch norm然后通过ReLu非线性。重复这三件事情

实验:
使用大文本分类数据集

Corpus:	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Method	n-TFIDF	n-TFIDF	n-TFIDF	ngrams	Conv	Conv+RNN	Conv	Conv
Author	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Xiao]	[Zhang]	[Zhang]
Error	7.64	2.81	1.31	4.36	37.95*	28.26	40.43*	4.93*
[Yang]	-	-	-	-	-	24.2	36.4	-

Table 4: Best published results from previous work. Zhang et al. (2015) best results use a Thesaurus data augmentation technique (marked with an *). Yang et al. (2016)'s hierarchical methods is particularly

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

Table 5: Testing error of our models on the 8 data sets. No data preprocessing or augmentation is used.

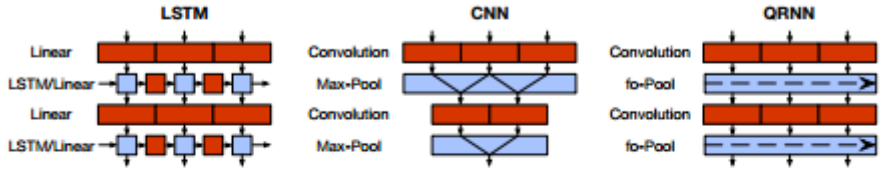
- 表中使用的是错误率，因此数字越低越好
- 深层网络的性能更好；残差层的结果也很好；

Q-RNN

RNNs是深度NLP的一个非常标准的构建块但是考虑到他的缺点，可以取RNNs和CNNs最好的和可并行的部分——**Q-RNN**

- Quasi-Recurrent Neural Networks by James Bradbury, Stephen Merity, Caiming Xiong & Richard Socher. ICLR 2017

- Tries to combine the best of both model families



- Convolutions for parallelism across time:

$$\mathbf{z}_t = \tanh(\mathbf{W}_z^1 \mathbf{x}_{t-1} + \mathbf{W}_z^2 \mathbf{x}_t)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f^1 \mathbf{x}_{t-1} + \mathbf{W}_f^2 \mathbf{x}_t)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o^1 \mathbf{x}_{t-1} + \mathbf{W}_o^2 \mathbf{x}_t).$$

→

$$\mathbf{Z} = \tanh(\mathbf{W}_z * \mathbf{X})$$
$$\mathbf{F} = \sigma(\mathbf{W}_f * \mathbf{X})$$
$$\mathbf{O} = \sigma(\mathbf{W}_o * \mathbf{X}),$$

Convolutions compute candidate, forget & output gates

- Element-wise gated pseudo-recurrence for parallelism across channels is **done in pooling layer**: $\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t,$

- 在time-1与进入卷积神经网络的max pooling层的time之间建立关系，计算候选、遗忘门、输出门
- 通常比LSTMs更好更快；更好的可解释