Text Classification and Convolutional Neural Networks

COSC 7336: Advanced Natural Language Processing
Fall 2017

UNIVERSITY of HOUSTON

Some content on these slides was borrowed from J&M

UNIVERSIDAD NACIONAL DE COLOMBIA

Today’s lecture

★ Text Classification: task definition

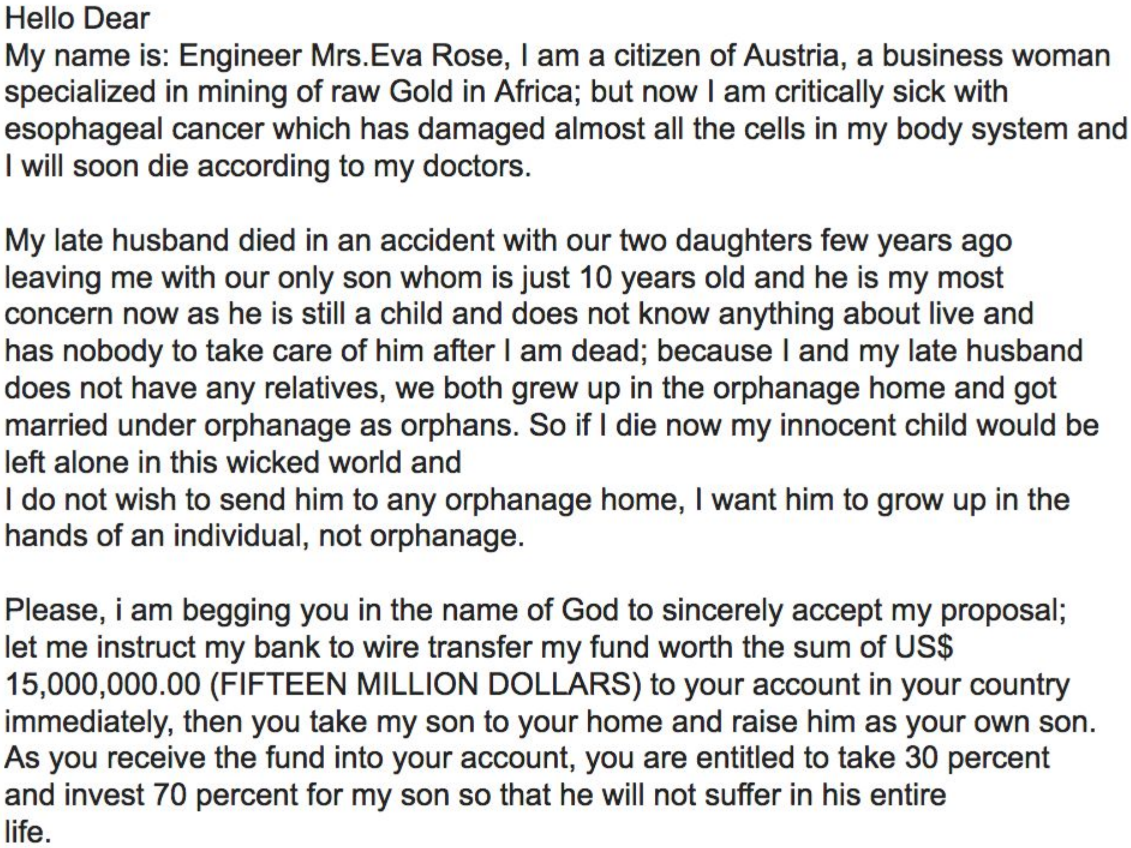
★ Classical approaches to Text Classification

★ Convolutional Neural Networks (CNN)

★ Recent work using CNNs for Text Classification problems

★ Demo: CNN for text

★ Practical



Hello Dear

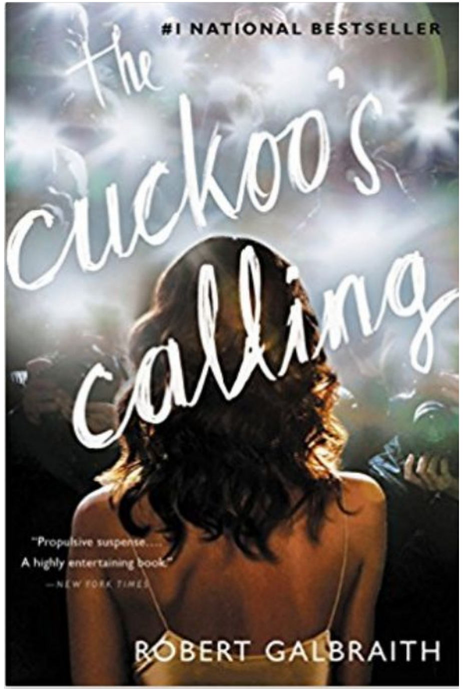
My name is: Engineer Mrs.Eva Rose, I am a citizen of Austria, a business woman specialized in mining of raw Gold in Africa; but now I am critically sick with esophageal cancer which has damaged almost all the cells in my body system and I will soon die according to my doctors.

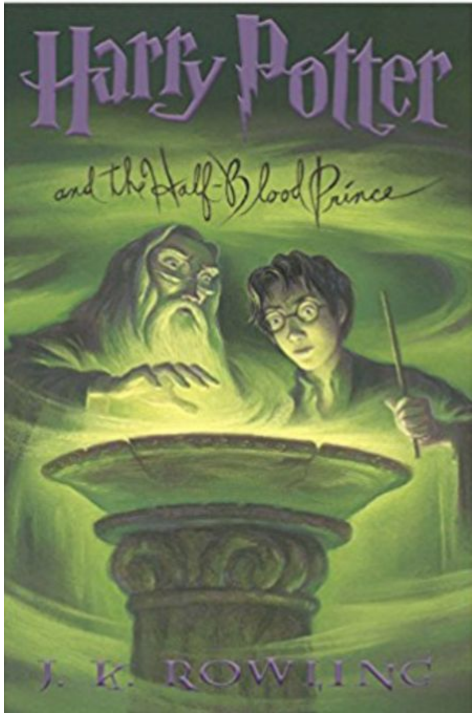
My late husband died in an accident with our two daughters few years ago leaving me with our only son whom is just 10 years old and he is my most concern now as he is still a child and does not know anything about live and has nobody to take care of him after I am dead; because I and my late husband does not have any relatives, we both grew up in the orphanage home and got married under orphanage as orphans. So if I die now my innocent child would be left alone in this wicked world and

I do not wish to send him to any orphanage home, I want him to grow up in the hands of an individual, not orphanage.

Please, i am begging you in the name of God to sincerely accept my proposal; let me instruct my bank to wire transfer my fund worth the sum of US$ 15,000,000.00 (FIFTEEN MILLION DOLLARS) to your account in your country immediately, then you take my son to your home and raise him as your own son. As you receive the fund into your account, you are entitled to take 30 percent and invest 70 percent for my son so that he will not suffer in his entire life.

What do these books have in common?





Other tasks that can be solved as TC

- Sentiment classification
- Native language identification
- Profiling

Formal definition of the TC task

- Input:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- Output: a predicted class $c \in C$

Methods for TC tasks

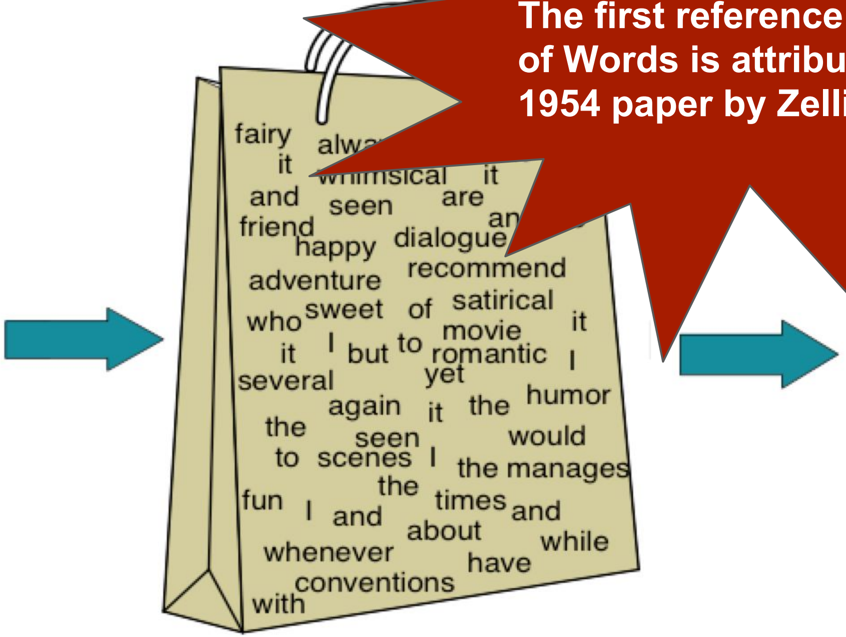
- Rule based approaches
- Machine Learning algorithms
  - Naive Bayes
  - Support Vector Machines
  - Logistic Regression
  - And now deep learning approaches

Naive Bayes for Text Classification

- Simple approach
- Based on the bag-of-words representation

Bag of words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



The first reference to Bag of Words is attributed to a 1954 paper by Zellig Harris

	3
[illegible]	3
een	
et	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Naive Bayes

Probabilistic classifier $\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d)$ (eq. 1)

According to Bayes rule: $P(x|y) = \dfrac{P(y|x)P(x)}{P(y)}$ (eq. 2)

Replacing eq. 2 into eq. 1: $\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) = \underset{c \in C}{\operatorname{argmax}}\, \dfrac{P(d|c)P(c)}{P(d)}$

Dropping the denominator: $\boxed{\underset{c \in C}{\operatorname{argmax}}\, P(d|c)P(c)}$

Naive Bayes

A document d is represented as a set of features $f_1, f_2, \ldots, f_n$

$$\hat{c} = \operatorname*{argmax}_{c\in C} \overbrace{P(f_1, f_2, ..., f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

How many parameters do we need to learn in this model?

Naive Bayes Assumptions

1. Position doesn't matter
2. Naive Bayes assumption: probabilities $P(f_i \mid c)$ are independent given the class c and thus we can multiply them:

$$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

This leads us to:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

Naive Bayes in Practice

We consider word positions:

$$\begin{aligned}\text{positions} &\leftarrow \text{all word positions in test document}\\ c_{NB} &= \underset{c\in C}{\operatorname{argmax}}\, P(c) \prod_{i\in positions} P(w_i|c)\end{aligned}$$

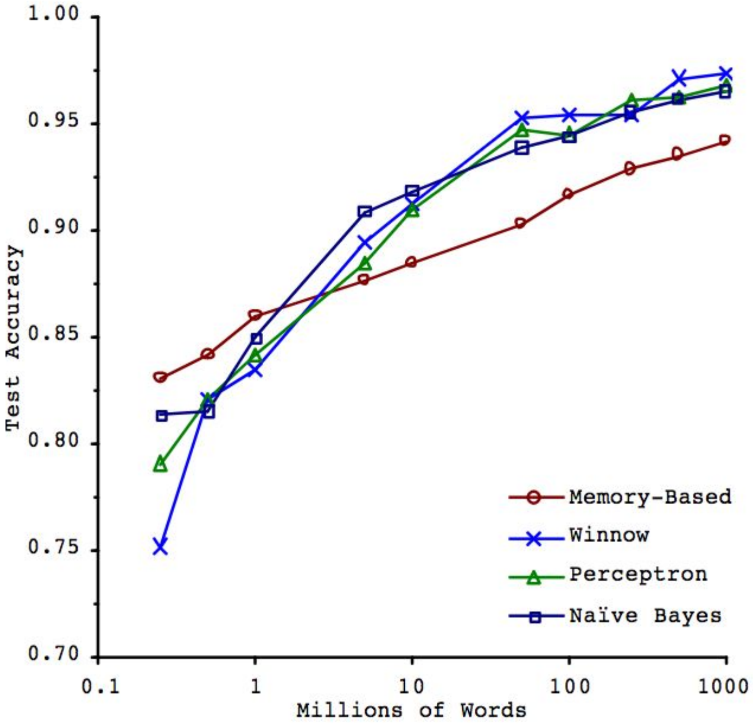
We also do everything in log space:

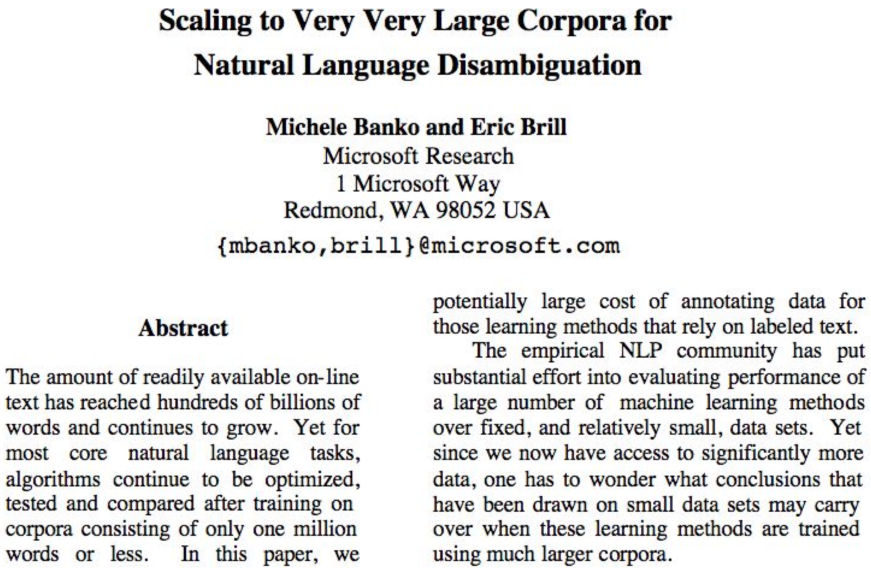
$$c_{NB} = \underset{c\in C}{\operatorname{argmax}}\, \log P(c) + \sum_{i\in positions} \log P(w_i|c)$$

Naive Bayes: Training

How do we compute $P(c)$ and $P(f_i|c)$?

Is Naive Bayes a good option for TC?





Scaling to Very Very Large Corpora for Natural Language Disambiguation

Michele Banko and Eric Brill
Microsoft Research
1 Microsoft Way
Redmond, WA 98052 USA
{mbanko,brill}@microsoft.com

Abstract

The amount of readily available on-line text has reached hundreds of billions of words and continues to grow. Yet for most core natural language tasks, algorithms continue to be optimized, tested and compared after training on corpora consisting of only one million words or less. In this paper, we potentially large cost of annotating data for those learning methods that rely on labeled text.

The empirical NLP community has put substantial effort into evaluating performance of a large number of machine learning methods over fixed, and relatively small, data sets. Yet since we now have access to significantly more data, one has to wonder what conclusions that have been drawn on small data sets may carry over when these learning methods are trained using much larger corpora.

Evaluation in TC

Confusion table

	Gold Standard	
	True	False
True	TP = true positives	FP = False positives
False	FN = false negatives	TN = True negatives

$$\text{Accuracy} = \frac{TP + TN}{(TP + TN + FN + FP)}$$

Evaluation in TC: Issues with Accuracy?

Suppose we want to learn to classify each message in a web forum as “extremely negative”. We have a collected gold standard data:

- ★ 990 instances are labeled as negative
- ★ 10 instances are labeled as positive
- ★ Test data has 100 instances (99- and 1+)
- ★ A dumb classifier can get 99% accuracy by always predicting “negative” !

More Sensible Metrics: Precision, Recall and F-measure

P= TP/(TP+FP)

R=TP/(TP+FN)

	Gold Standard	
	True	False
True	TP = true positives	FP = False positives
False	FN = false negatives	TN = True negatives

F-measure = $$\frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

What about Multi-class problems?

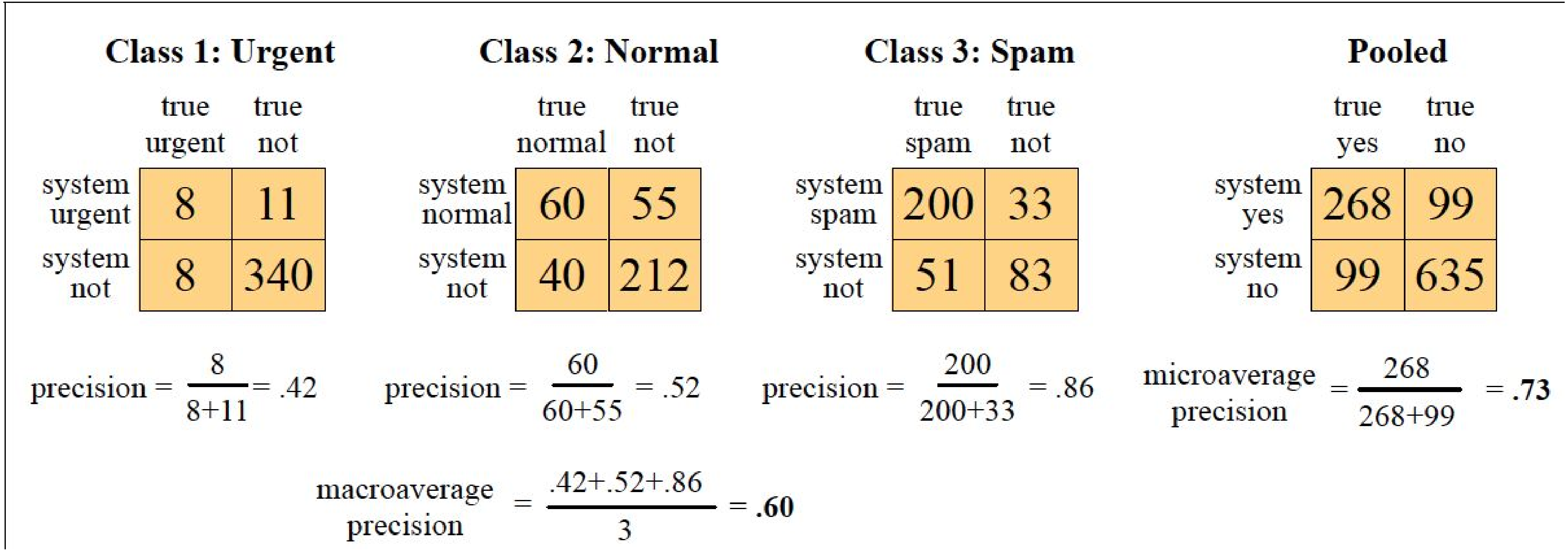
- Multi-class: c > 2
- P, R, and F-measure are defined for a single class
- We assume classes are mutually exclusive
- We use per class evaluation metrics

P = $\dfrac{c_{ii}}{\sum_j c_{ji}}$	R = $\dfrac{c_{ii}}{\sum_j c_{ij}}$

Micro vs Macro Average

- Macro average: measure performance per class and then average
- Micro average: collect predictions for all classes then compute TP, FP, FN, and TN
- Weighted average: compute performance per label and then average where each label score is weighted by its support

Example



Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

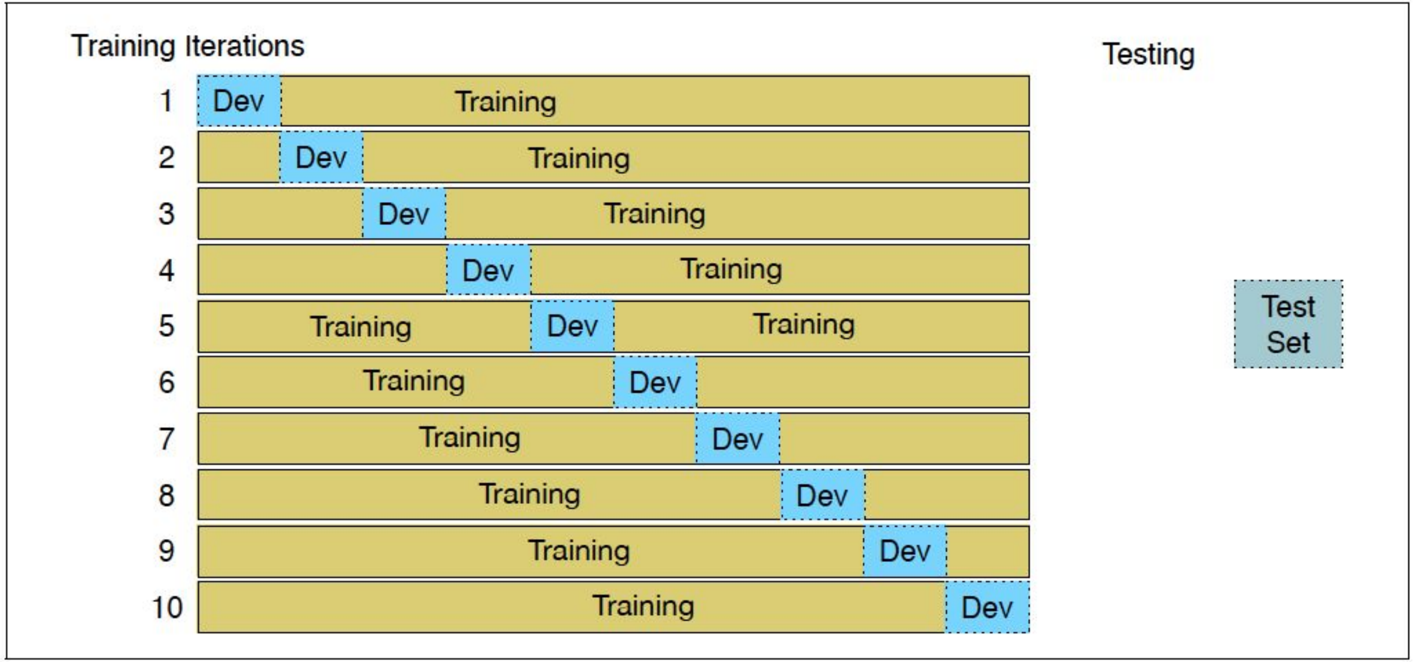
Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\begin{array}{c}\text{microaverage}\\\text{precision}\end{array} = \frac{268}{268+99} = \mathbf{.73}$$

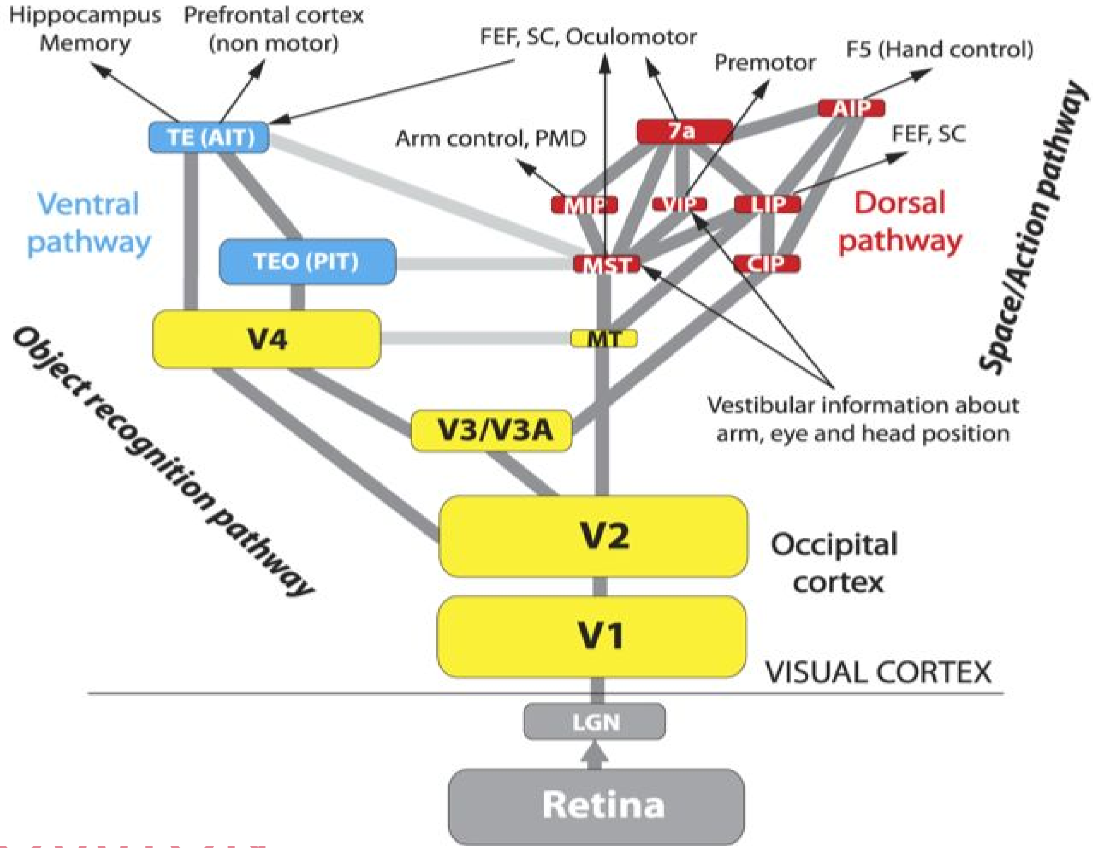
$$\begin{array}{c}\text{macroaverage}\\\text{precision}\end{array} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

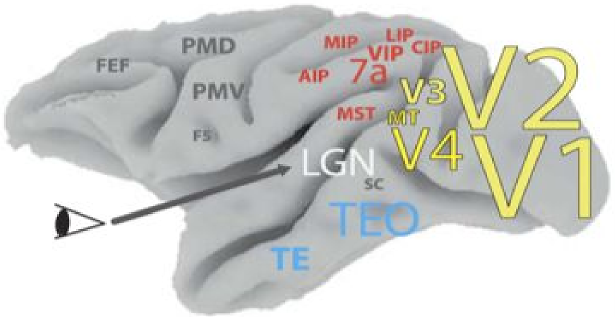
Train/Test Data Separation



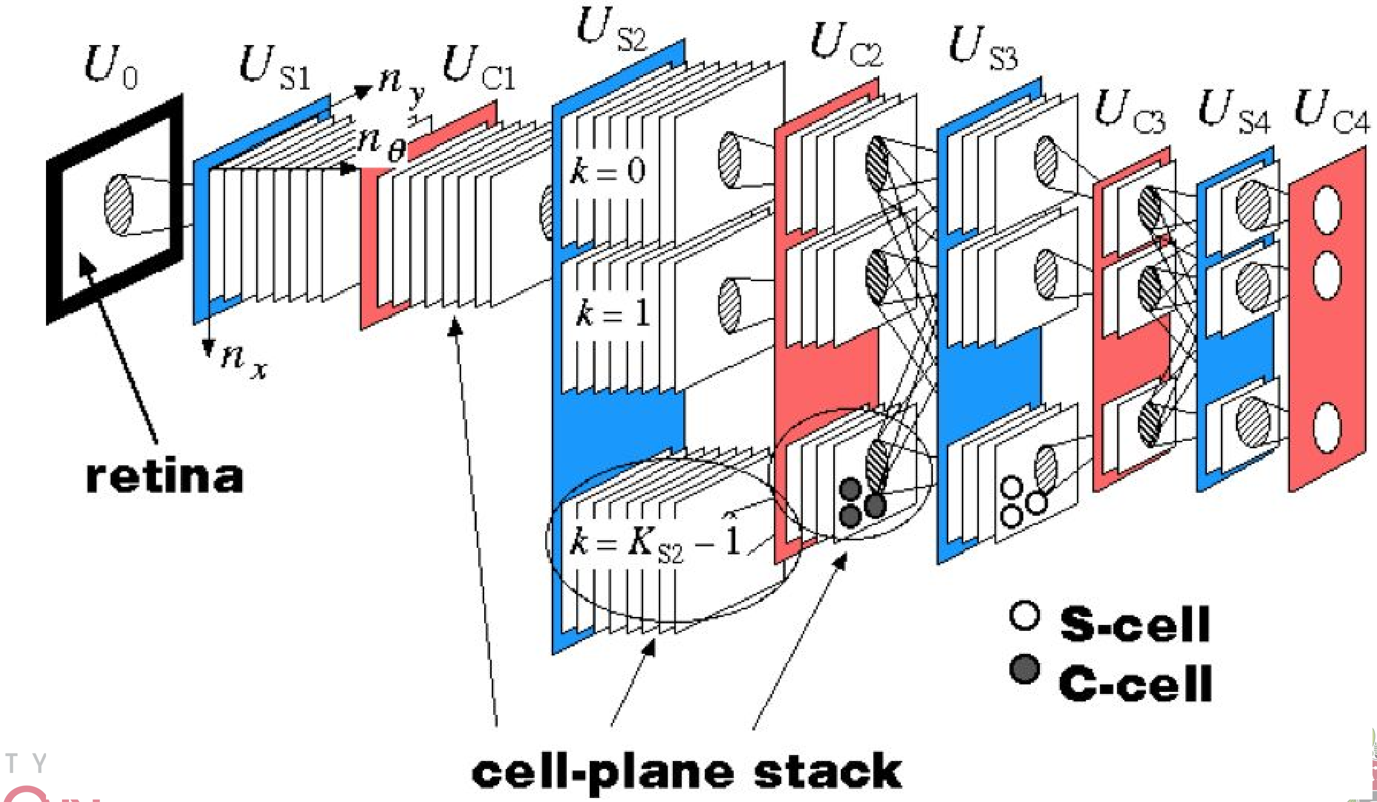
Convolutional Neural Networks

Visual Cortex

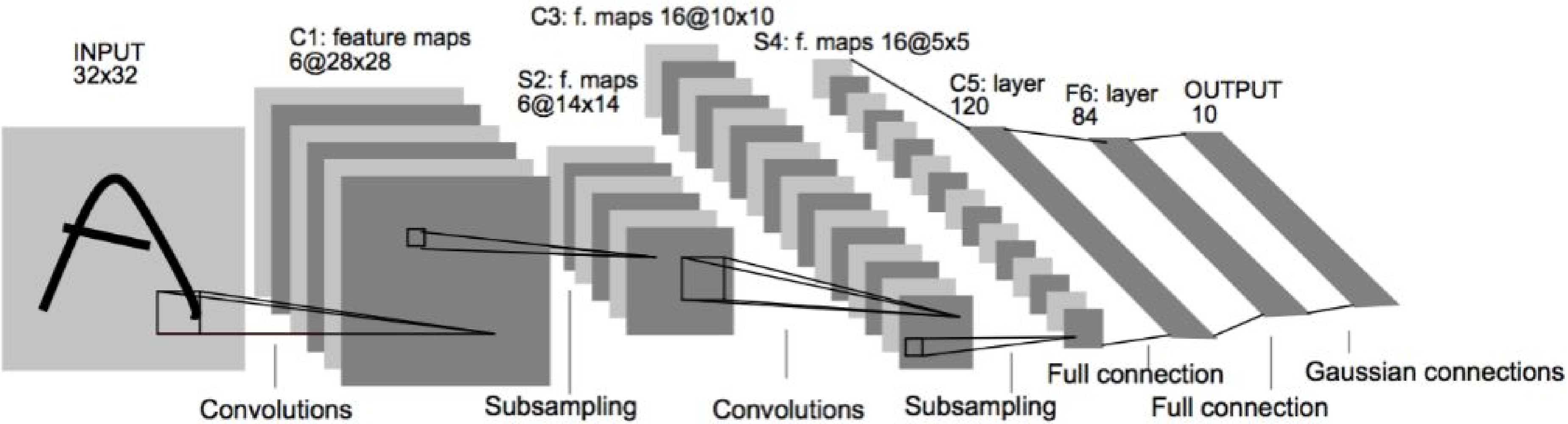




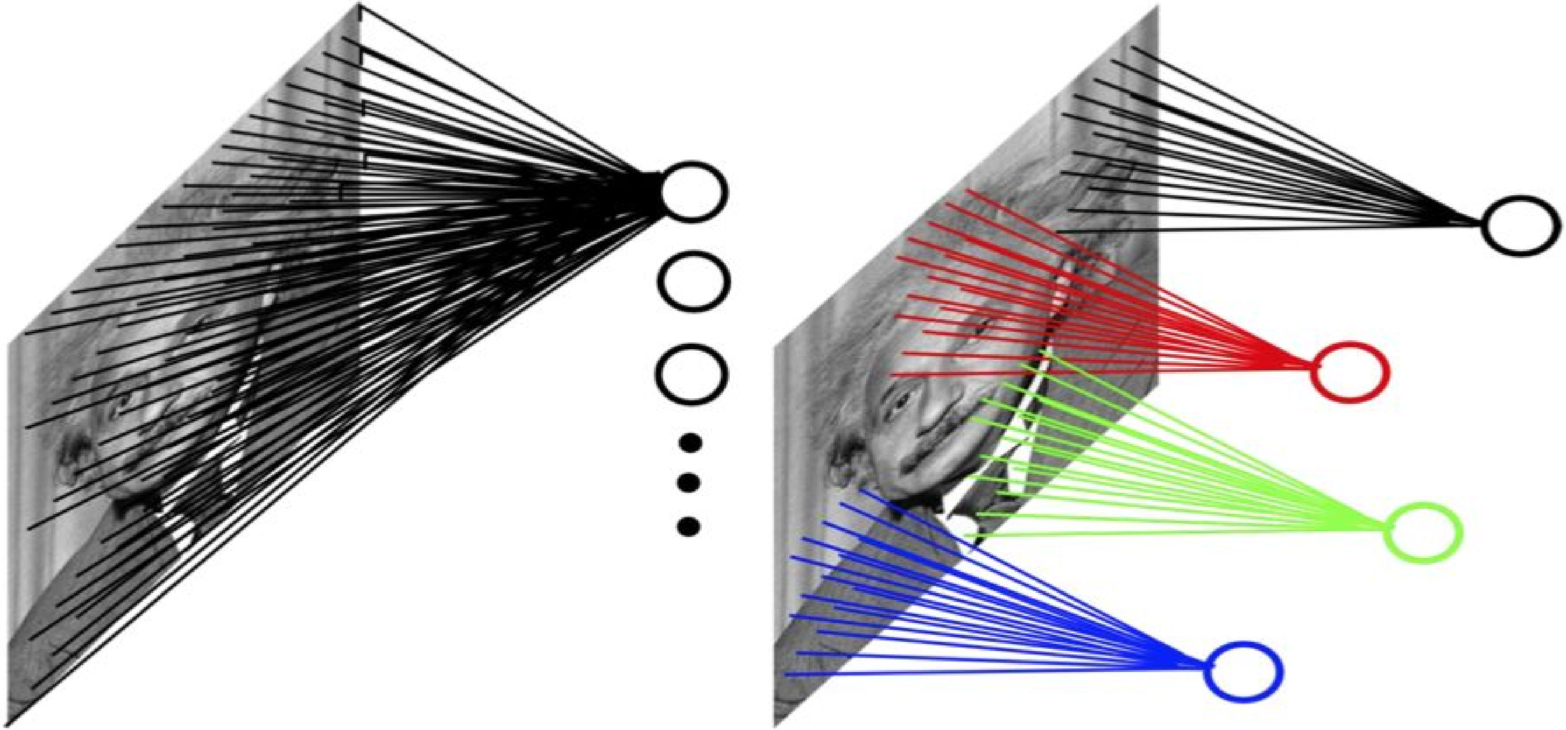
Neocognitron (Fukushima, 1980)



LeNet (LeCun, 1998)

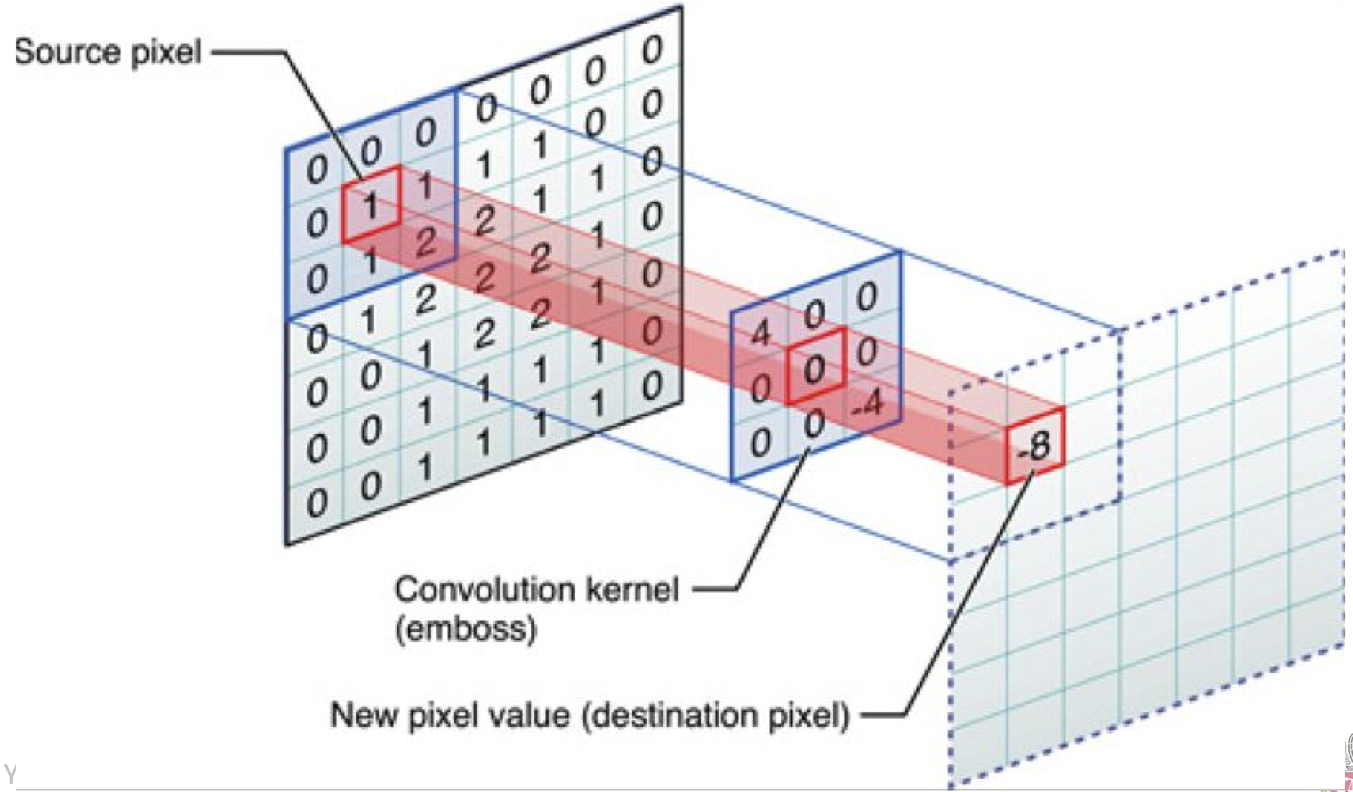


Convolution



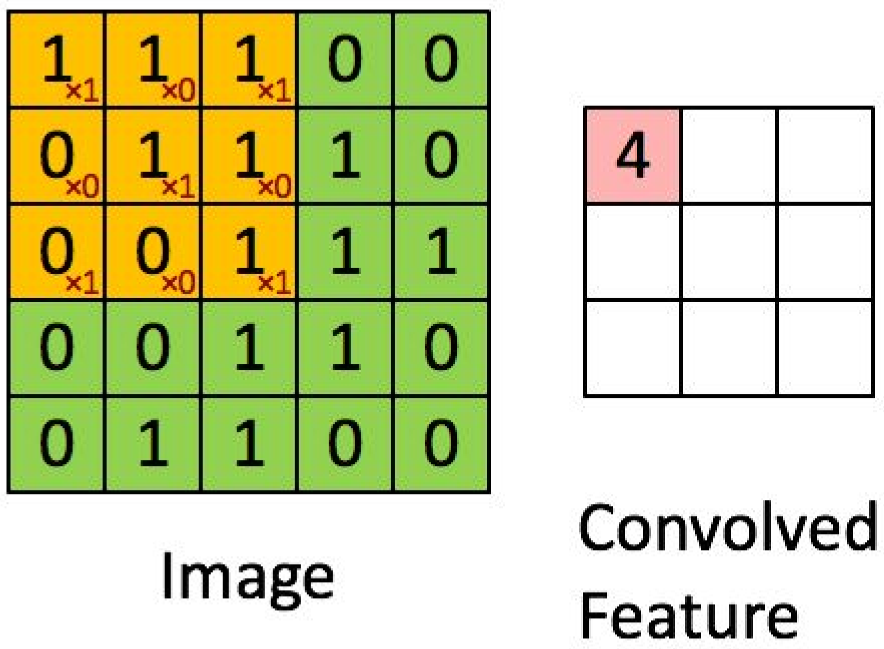
(source: ICML2013 Deep Learning Tutorial, Yan LeCun et al.)

Convolution


Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)
-8

(Source:Feature extraction using convolution, Stanford Deep Learning Wiki)

Convolution



$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

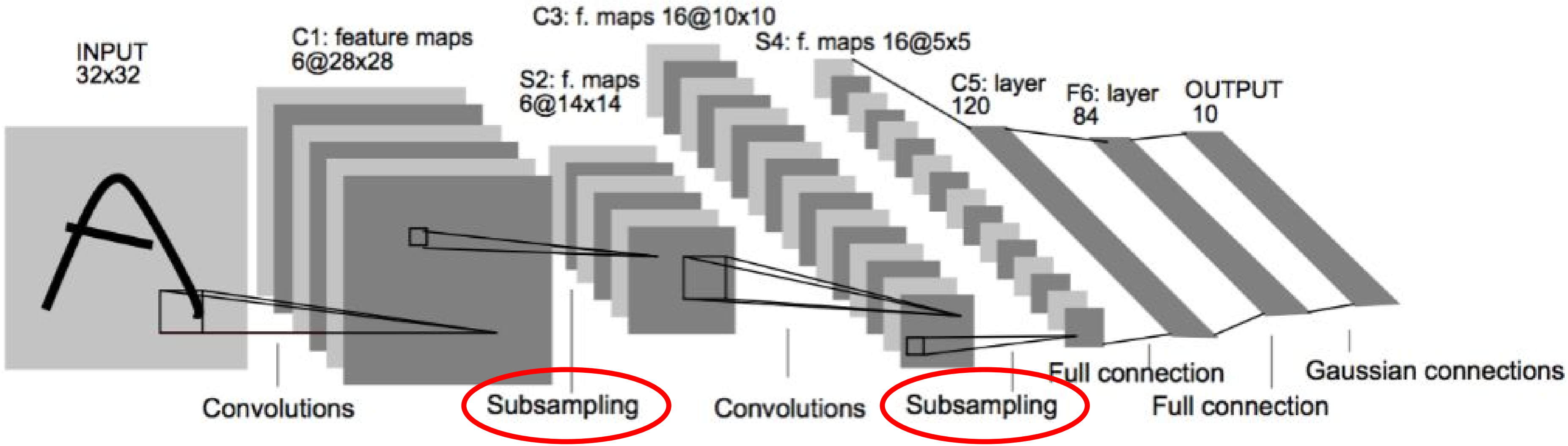
Image

4		

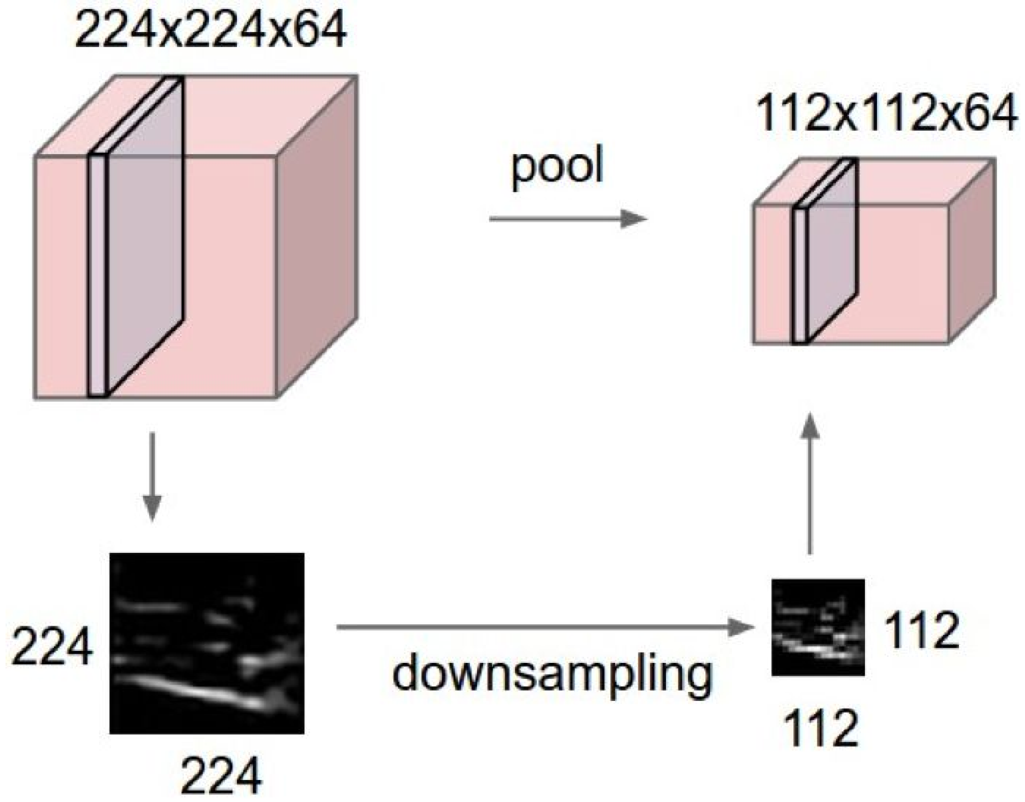
Convolved Feature

(Source:Feature extraction using convolution, Stanford Deep Learning Wiki)

Pooling or Subsampling

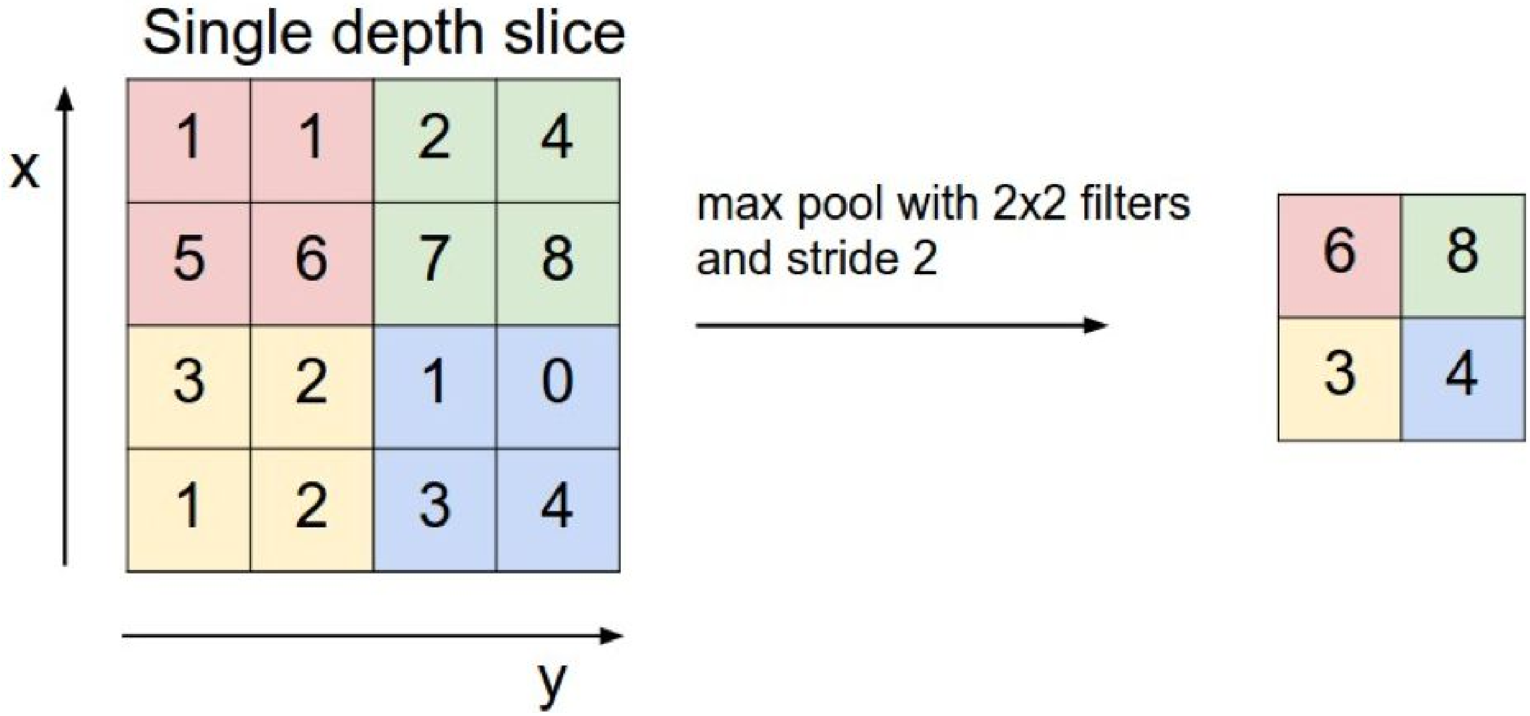


Pooling



(source: Karpathy, CS231n Convolutional Neural Networks for Visual Recognition)

Pooling



Single depth slice

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

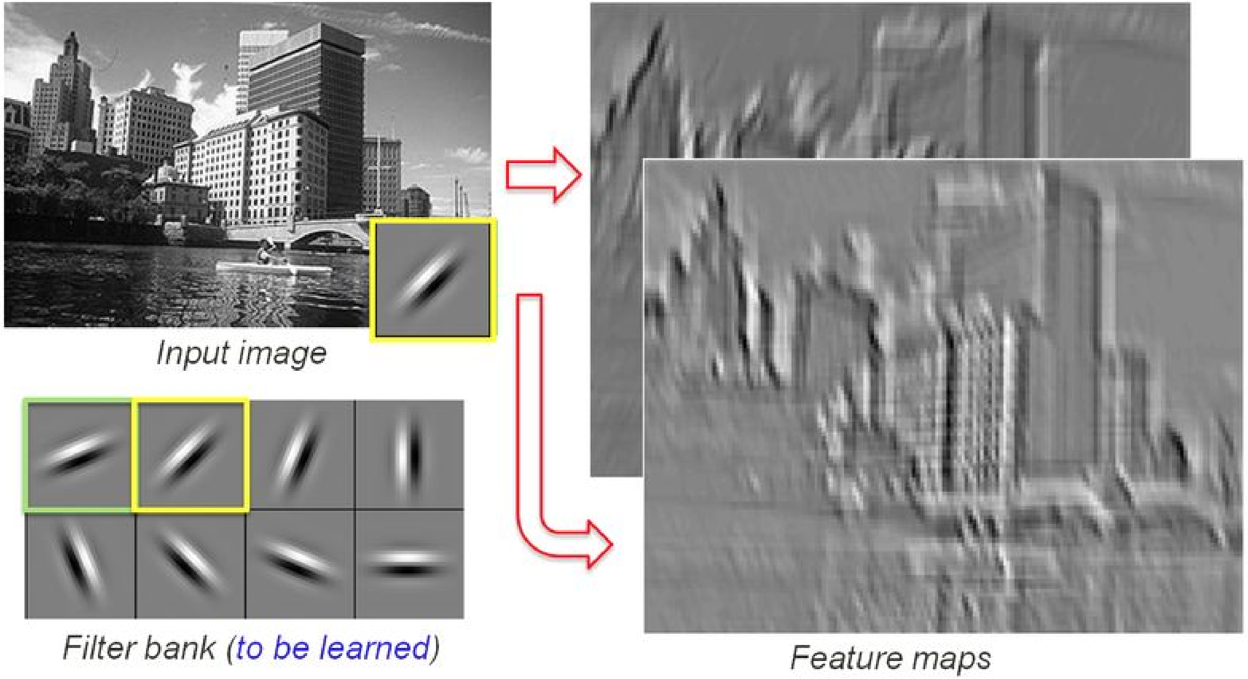
max pool with 2x2 filters and stride 2

6	8
3	4

(source: Karpathy, CS231n Convolutional Neural Networks for Visual Recognition)

Properties

- ★ Local invariance
- ★ Compositionality



Adapted from: http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

CNNs for NLP

★ Same as images, text exhibits some local invariance properties that can be modeled by CNNs

★ CNNs are not as popular as recurrent neural networks (to be discussed next class) for text analysis, but there are many cases where they work pretty well.

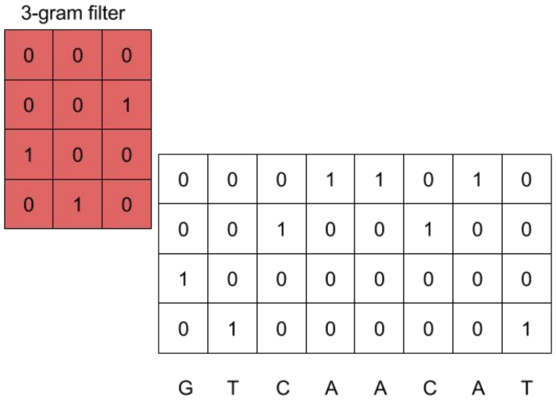
★ Big advantage: CNNs can be trained efficiently since they take full advantage of parallelism.

A character-level CNN

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

Example from Sebastián Sierra
http://lin99.github.io/NLPTM-2016/4.Docs/cnn%20for%20text.pdf

A character-level CNN

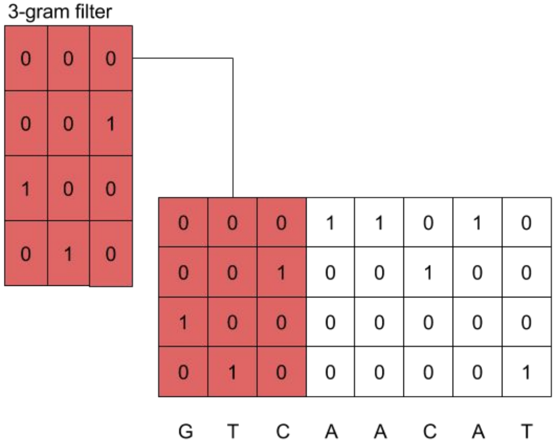


3-gram filter

0	0	0
0	0	1
1	0	0
0	1	0

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

A character-level CNN

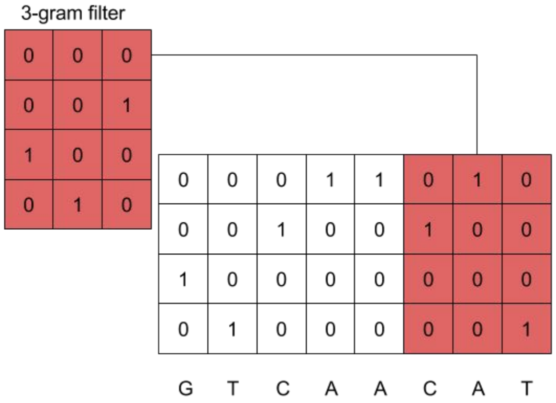


3-gram filter

0	0	0
0	0	1
1	0	0
0	1	0

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

A character-level CNN

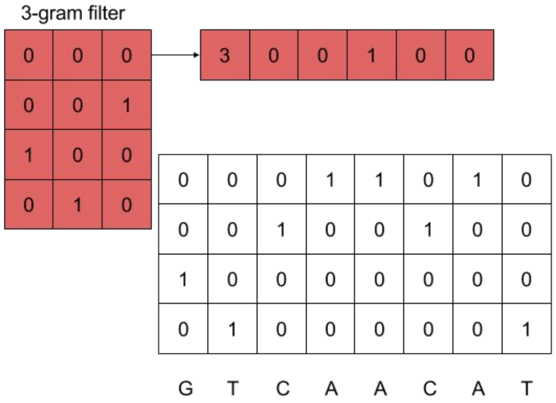


3-gram filter

0	0	0
0	0	1
1	0	0
0	1	0

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

A character-level CNN



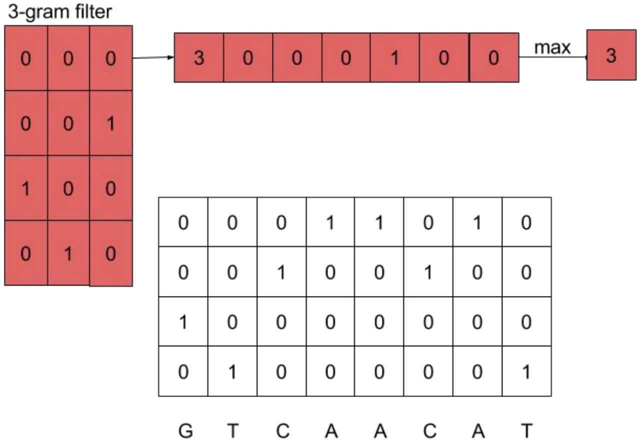
3-gram filter

0	0	0
0	0	1
1	0	0
0	1	0

3	0	0	1	0	0

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

A character-level CNN



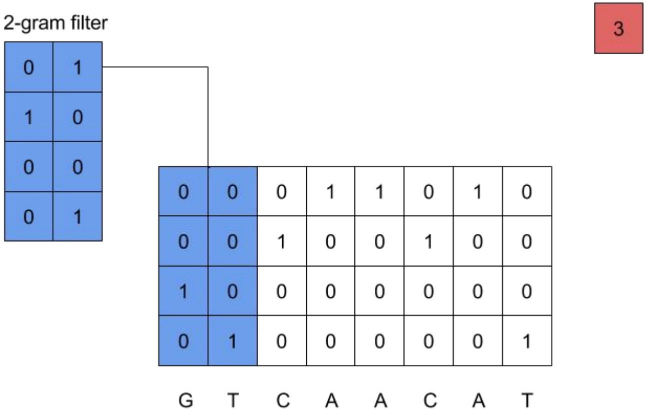
3-gram filter

0	0	0
0	0	1
1	0	0
0	1	0

3	0	0	0	1	0	0

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

A character-level CNN

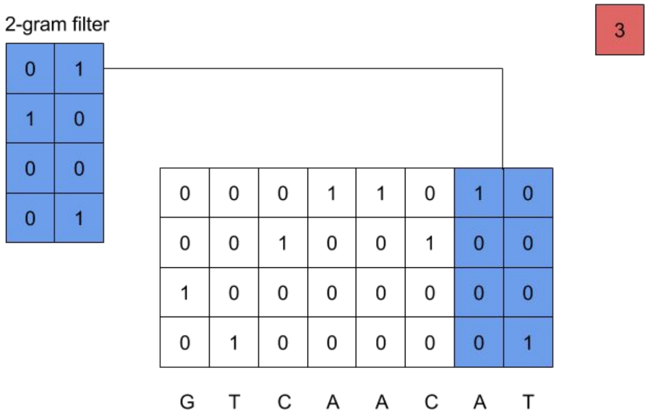


2-gram filter

0	1
1	0
0	0
0	1

G	T	C	A	A	C	A	T
0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1

A character-level CNN



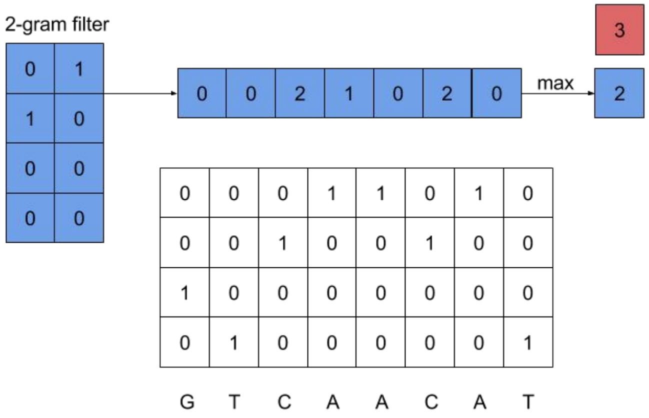
2-gram filter

0	1
1	0
0	0
0	1

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

3

A character-level CNN



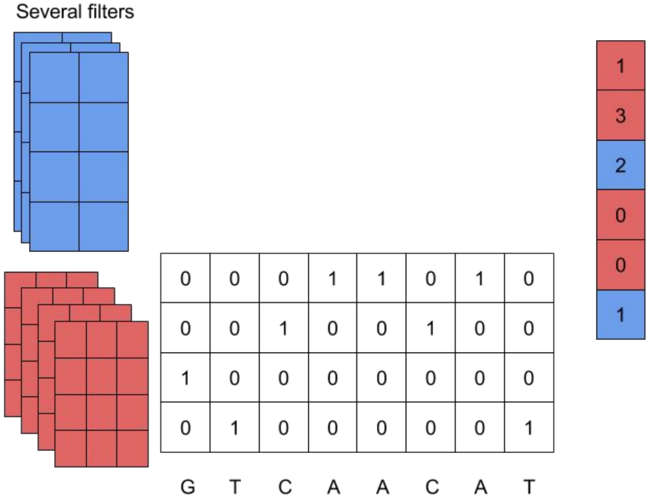
0	1
1	0
0	0
0	0

0	0	2	1	0	2	0

2

0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

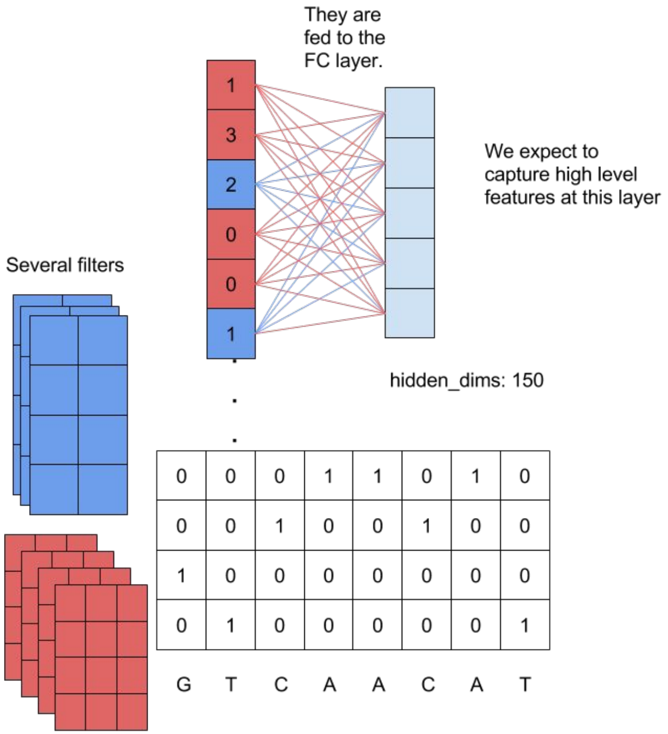
A character-level CNN



0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

1
3
2
0
0
1

A character-level CNN



They are fed to the FC layer.

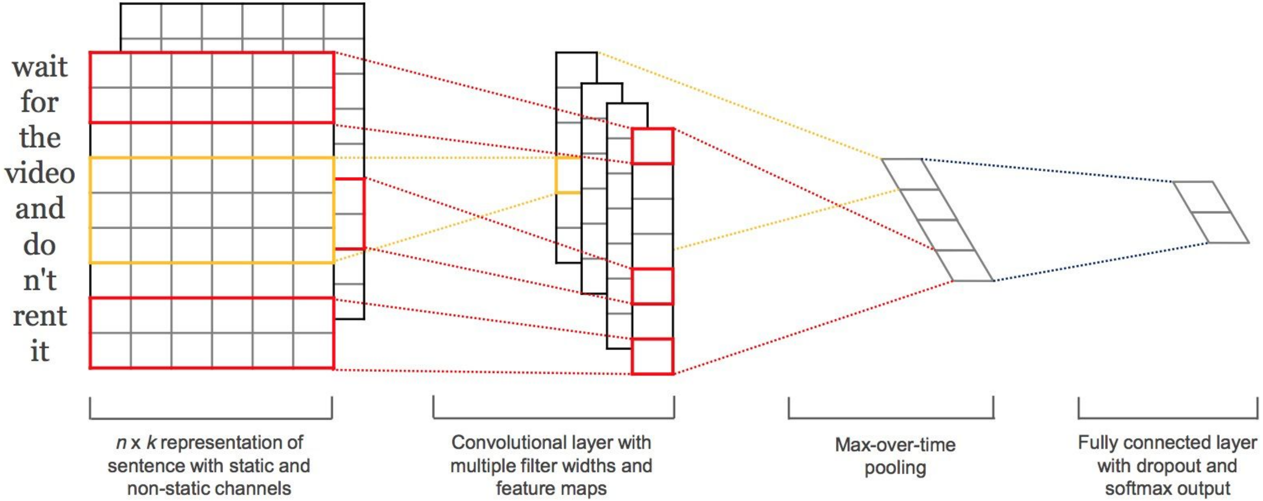
We expect to capture high level features at this layer

Several filters

hidden_dims: 150

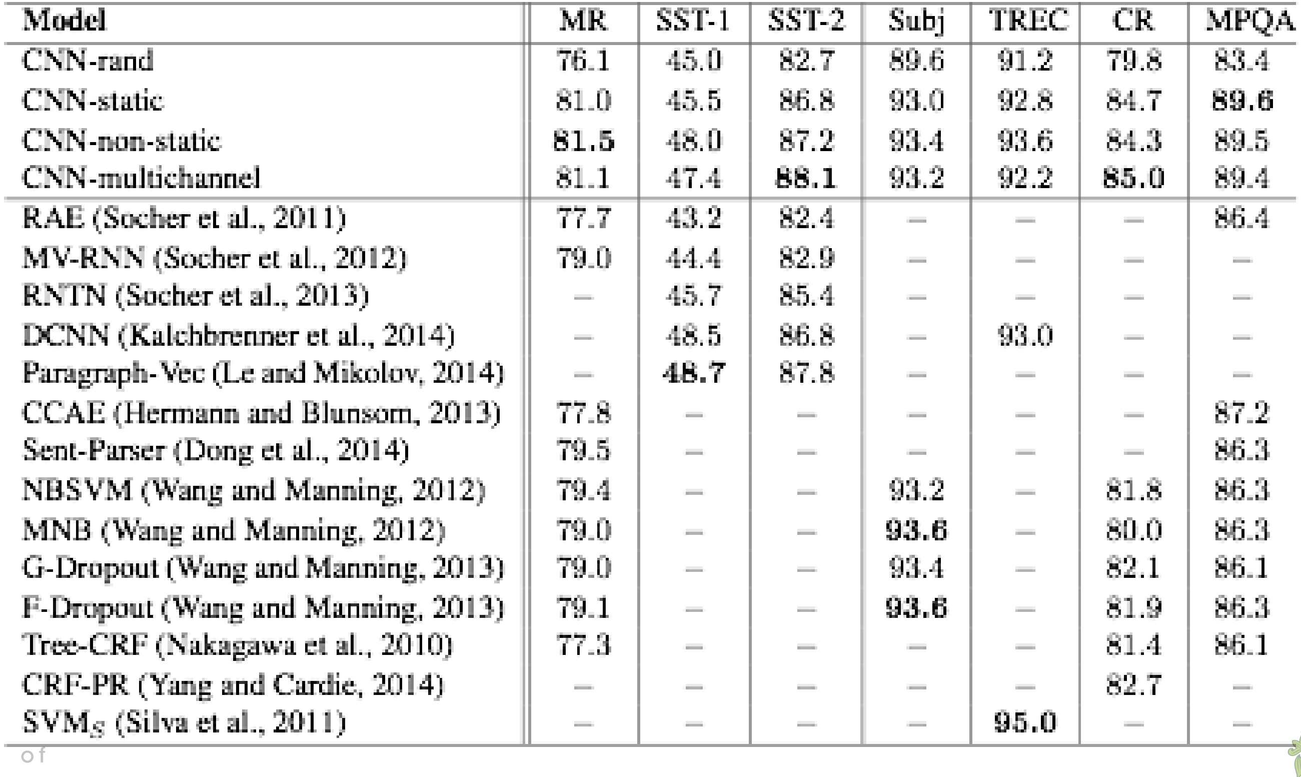
0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
G	T	C	A	A	C	A	T

Convolutional neural networks for sentence classification


wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

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

Convolutional neural networks for sentence classification



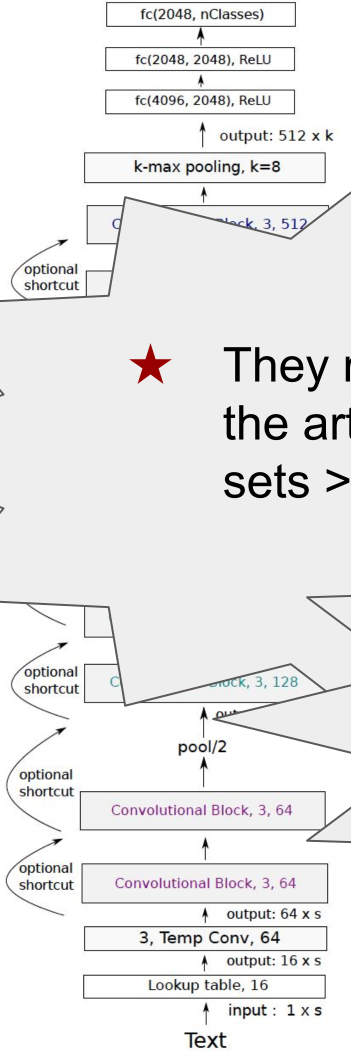
Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (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**	–	–

Recen CNNs: Text Classification

Alexis Conn
Facebook AI
acc

Ab

The dominant app
tasks are recurrent r
ticular LSTMs, and
networks. Howev
are rather shallow
deep convolutional
pushed the state of



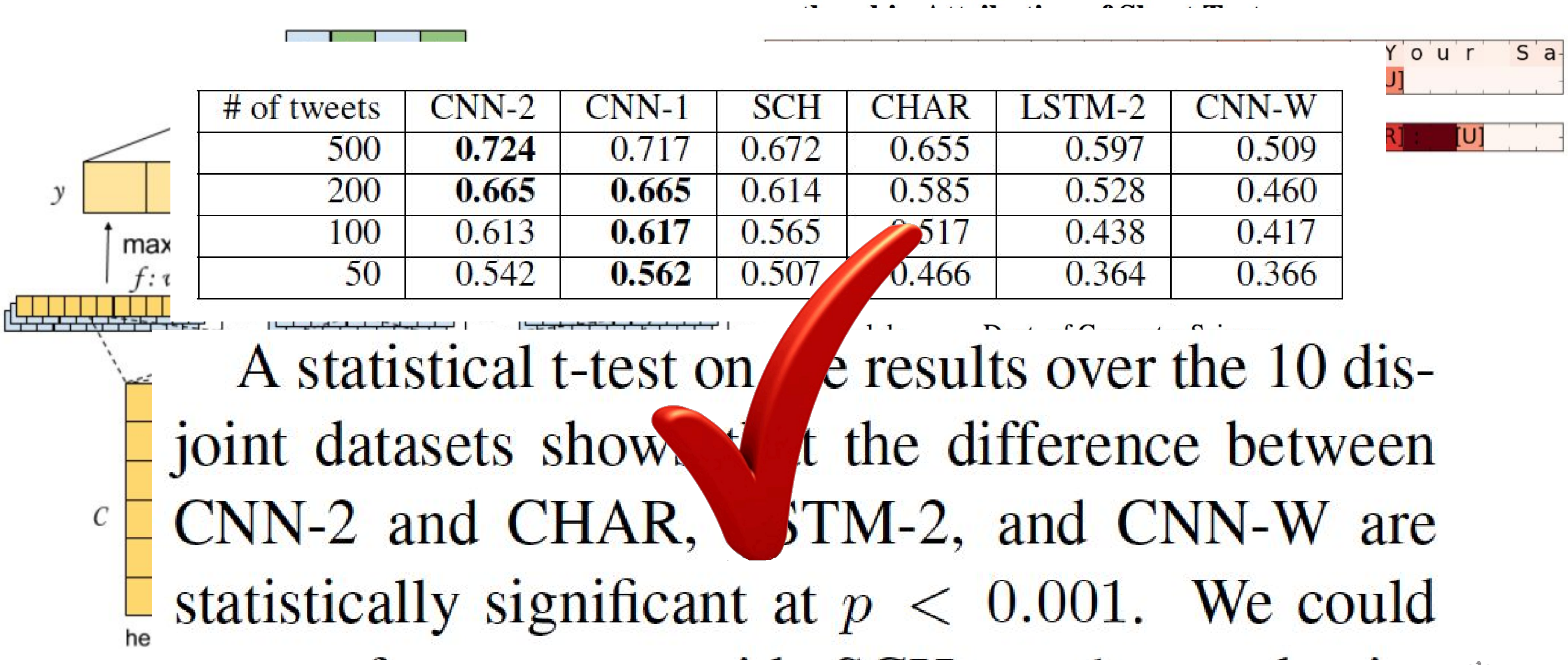
vas
niti

Depth:	9	17	29	49
conv block 512	2	4	4	6
conv block 256	2	4	4	10
conv block 128	2	4	10	16
conv block 64	2	4	10	16
[illegible]	[illegible]	1	1	1
[illegible]	[illegible]	[illegible]	[illegible]	7.8

- ★ They reach state of the art on large data sets > 630k

- ★ No statistical tests for significance
- ★ They couldn't outperform a hierarchical method adapted for multiple sentences.

Recent work using CNNs: Authorship Attribution



# of tweets	CNN-2	CNN-1	SCH	CHAR	LSTM-2	CNN-W
500	**0.724**	0.717	0.672	0.655	0.597	0.509
200	**0.665**	**0.665**	0.614	0.585	0.528	0.460
100	0.613	**0.617**	0.565	[illegible]517	0.438	0.417
50	0.542	**0.562**	0.507	0.466	0.364	0.366

A statistical t-test on e results over the 10 disjoint datasets show t the difference between CNN-2 and CHAR, STM-2, and CNN-W are statistically significant at $p < 0.001$. We could

CNNs for Sentence Classification Demo