OCR实战

第二课 单字符识别

第二课 单字符识别

- 单字符识别
  - 模版匹配
  - 特征提取 + 传统分类器
  - CNN模型
- 过切分的路径选择
  - VITERBI + BEAM SEARCH
- 语言模型
  - N-GRAM
  - LSTM
  - 2D-PCFG

单字符识别

- 模版匹配

$$D(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N}[S^{ij}(m,n)-T(m,n)]^2$$

欧式距离相似性函数

$$\sum_{m=1}^{M}\sum_{n=1}^{N}[S^{ij}(m,n)]^2-2\sum_{m=1}^{M}\sum_{n=1}^{N}S^{ij}(m,n)\cdot T(m,n)+\sum_{m=1}^{M}\sum_{n=1}^{N}[T(m,n)]^2$$

展开后相似性函数

$$R(i,j)=\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}S^{ij}(m,n)\cdot T(m,n)}{\sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N}[S^{ij}(m,n)]^2}\sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N}[T(m,n)]^2}}$$

归一化相似性函数

- method=CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

- method=CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

- method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

- method=CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

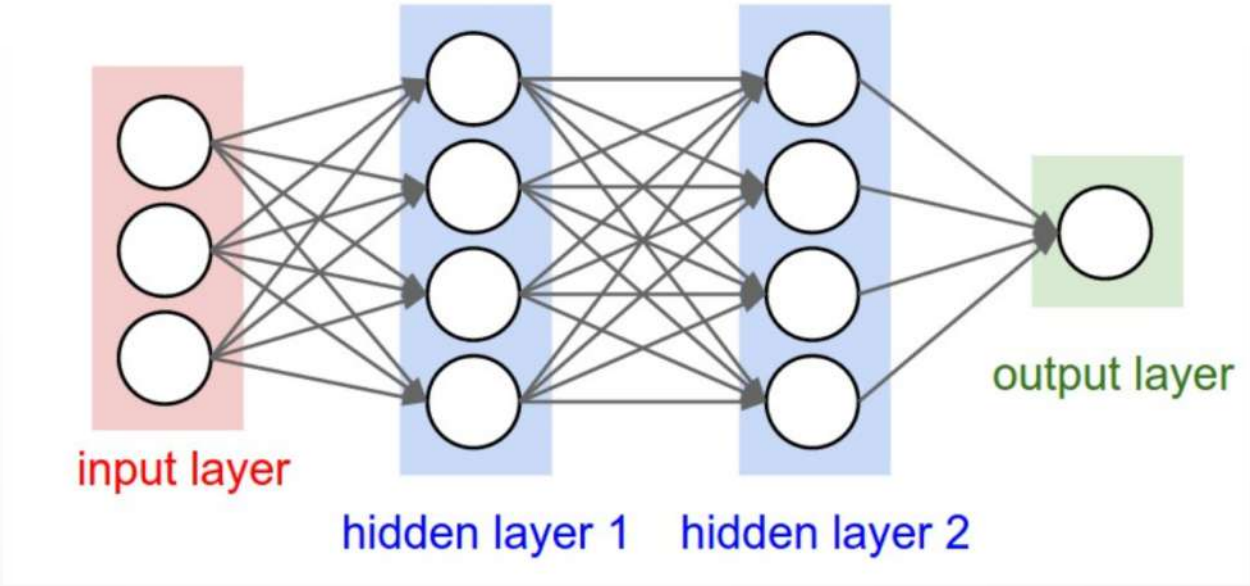
单字符识别

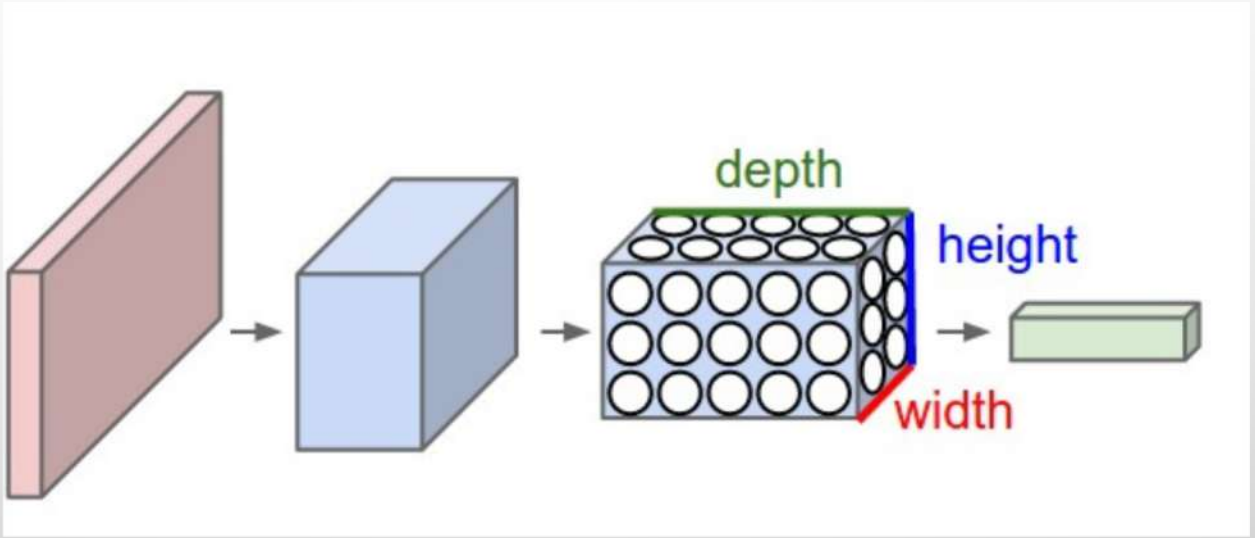
- 特征提取 + 传统分类器
  - 位图
  - 垂直/水平投影
  - 占空比 / 孔洞 / 穿刺
  - 方向 / 弧度
  - …

  +
  - SVM
  - MLP

单字符识别

- CNN模型

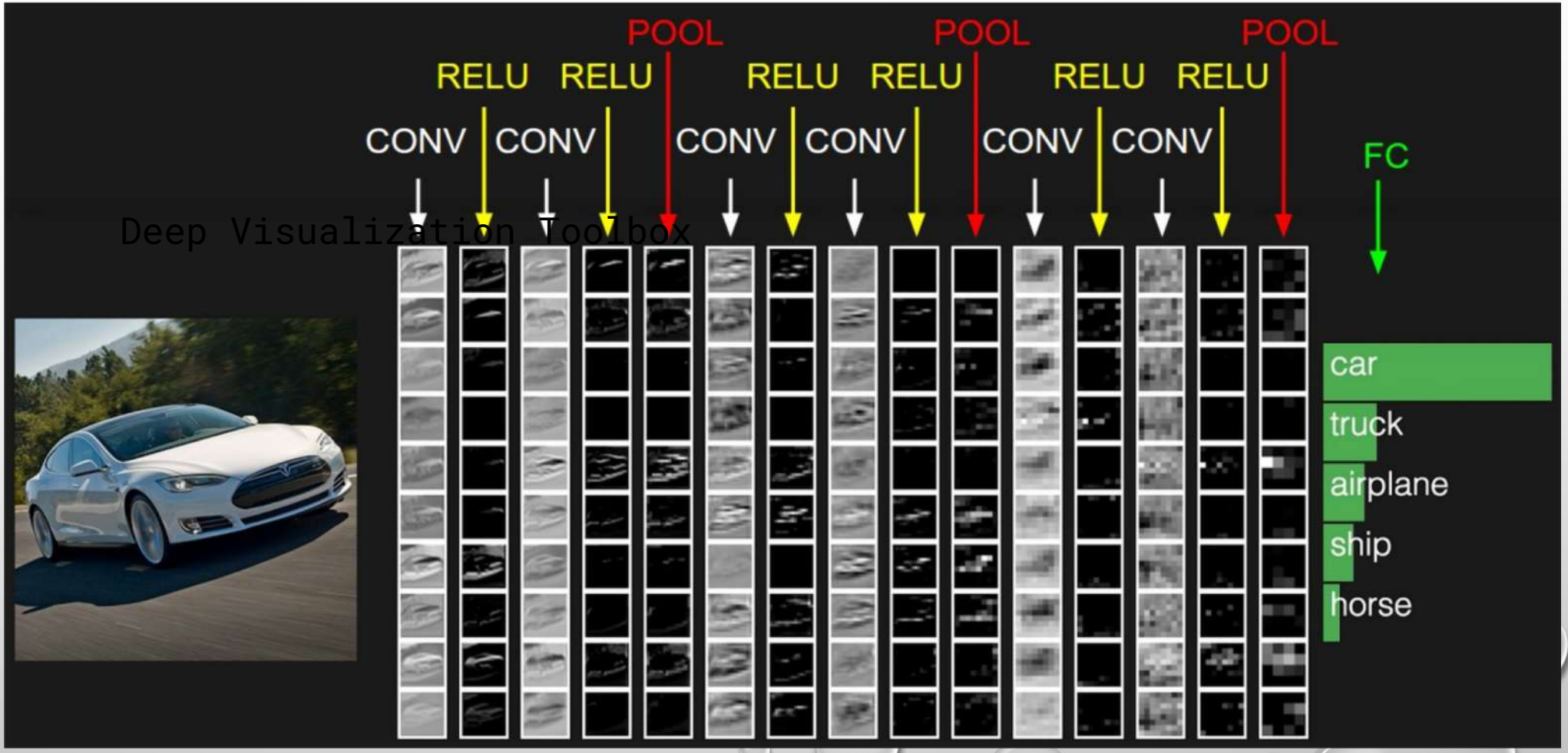




http://cs231n.stanford.edu

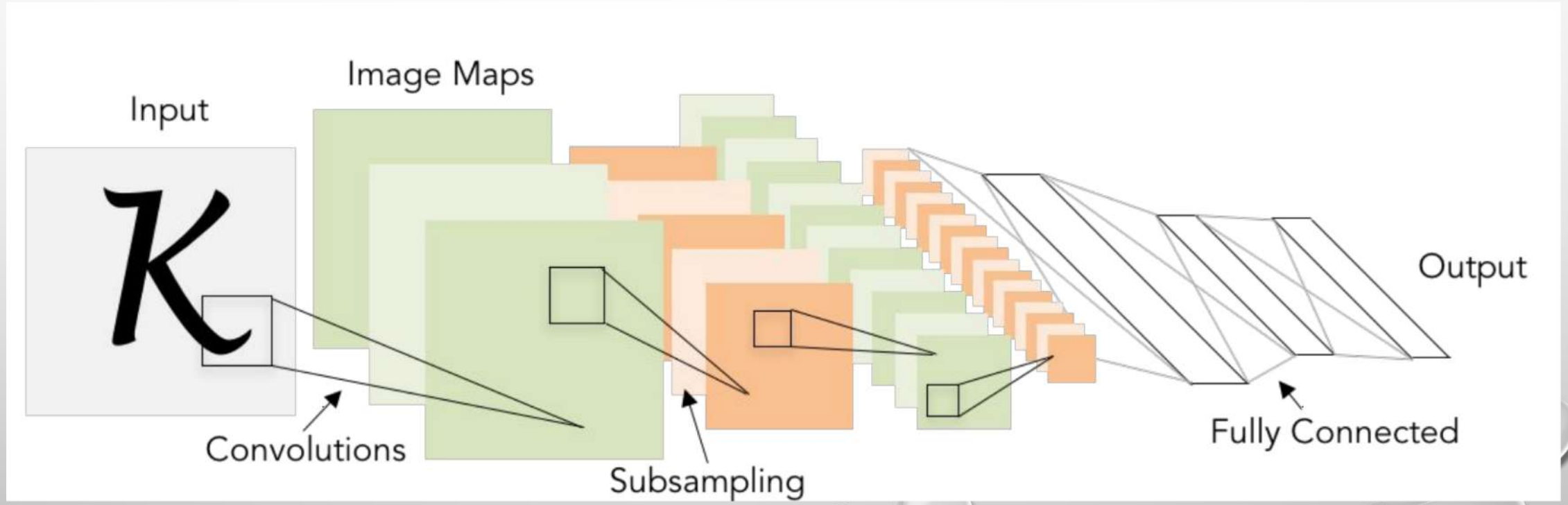
单字符识别

- CNN模型

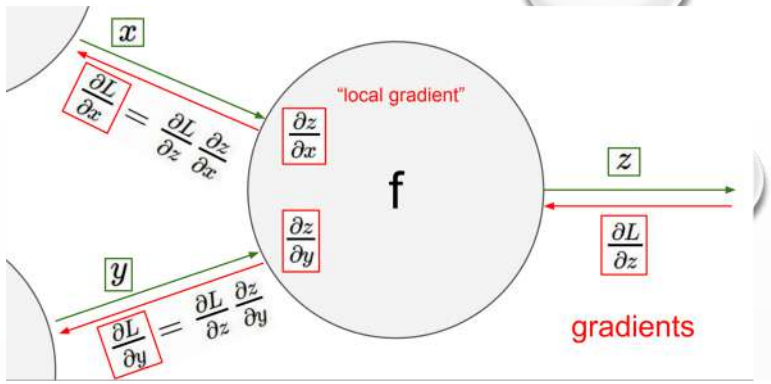

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
Deep Visualization Toolbox
car
truck
airplane
ship
horse

单字符识别

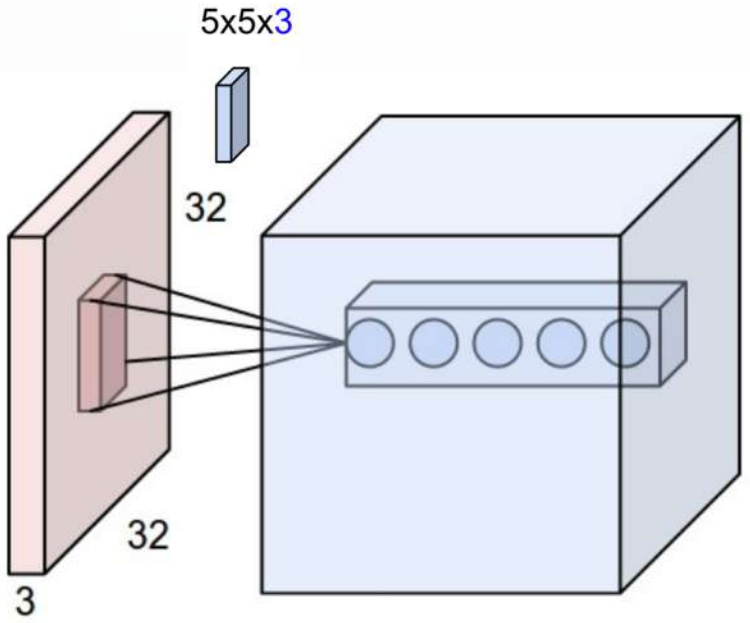
- CNN模型

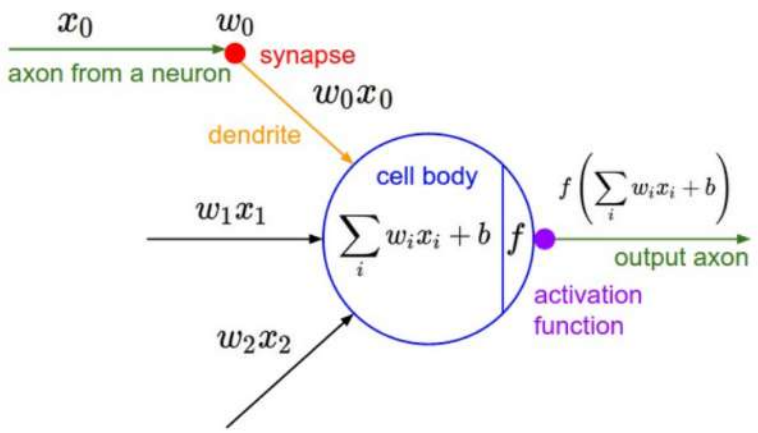

Input
Image Maps
Output
Convolutions
Subsampling
Fully Connected

单字符识别



- CNN模型

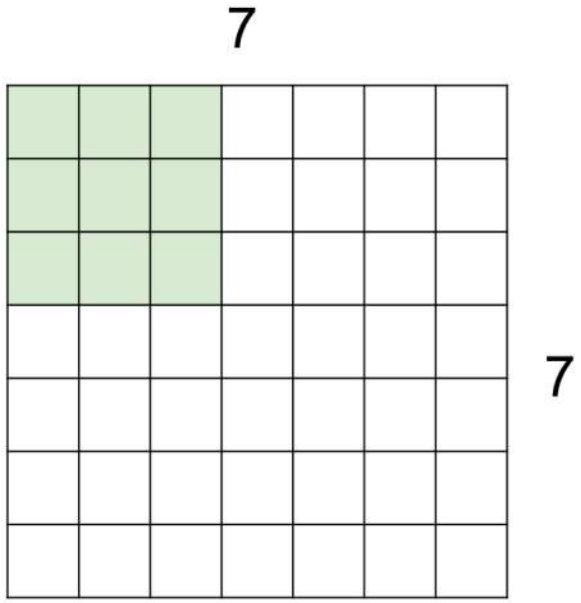




Left: An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input - see discussion of depth columns in text below. **Right:** The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

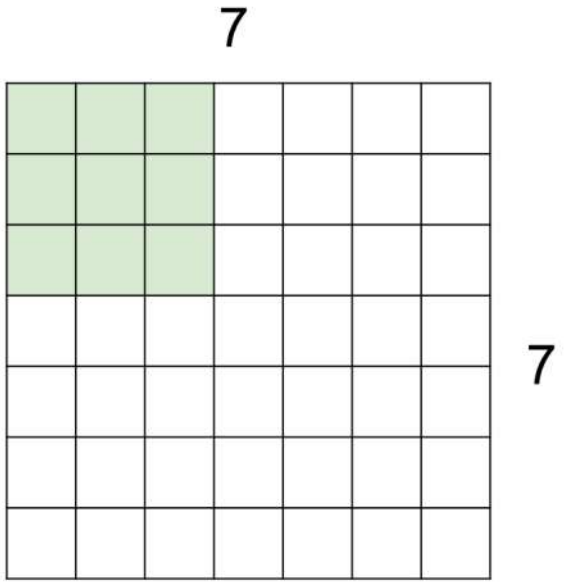
单字符识别

- CNN模型

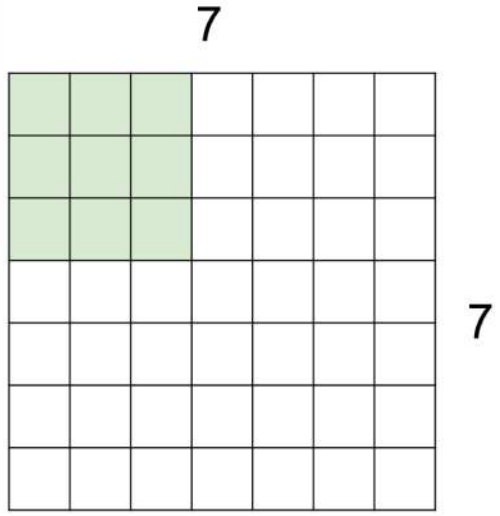


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

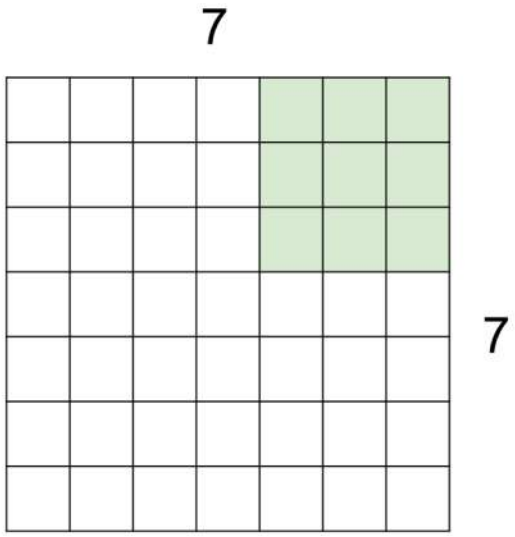


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

单字符识别

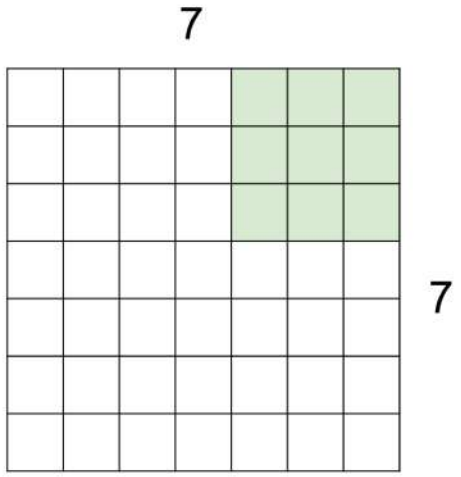
- CNN模型

A closer look at spatial dimensions:

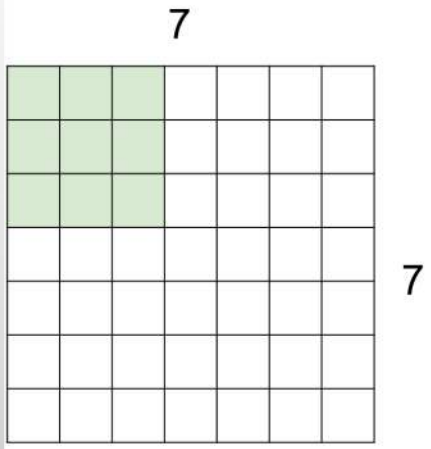


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

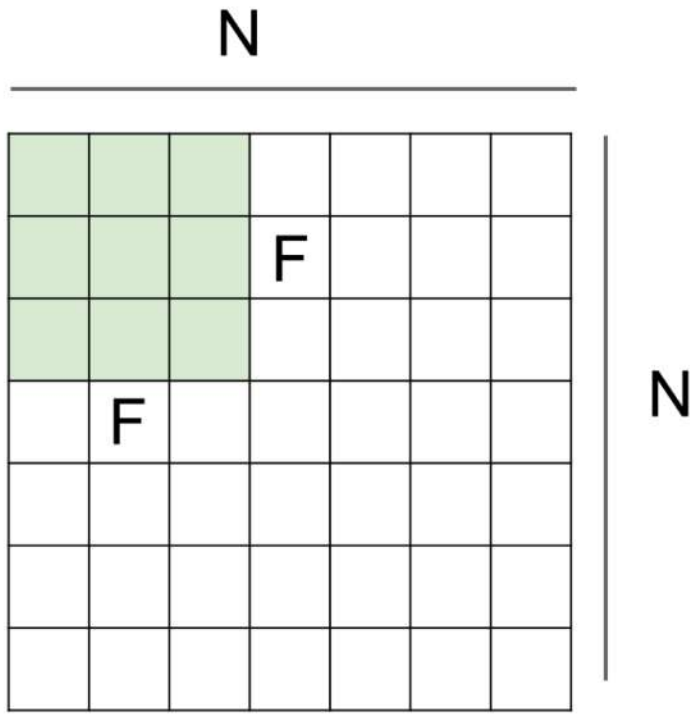


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

单字符识别

- CNN模型



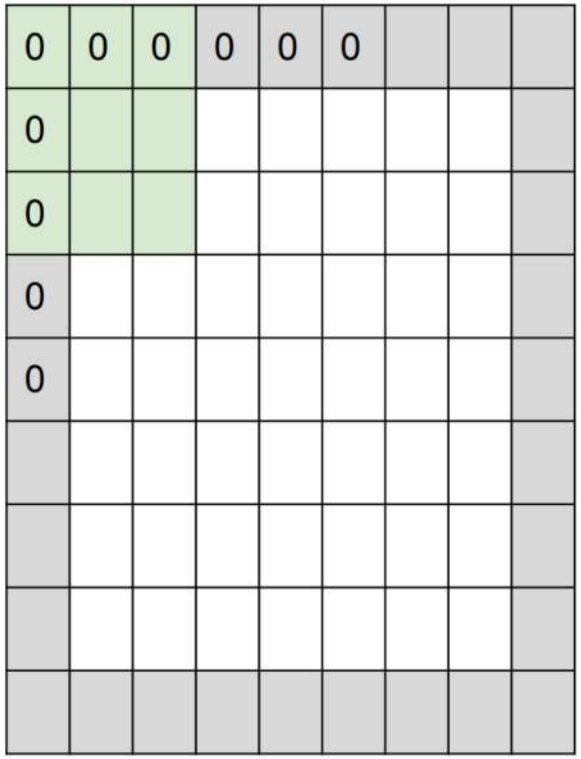
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

单字符识别

- CNN模型

In practice: Common to zero pad the border



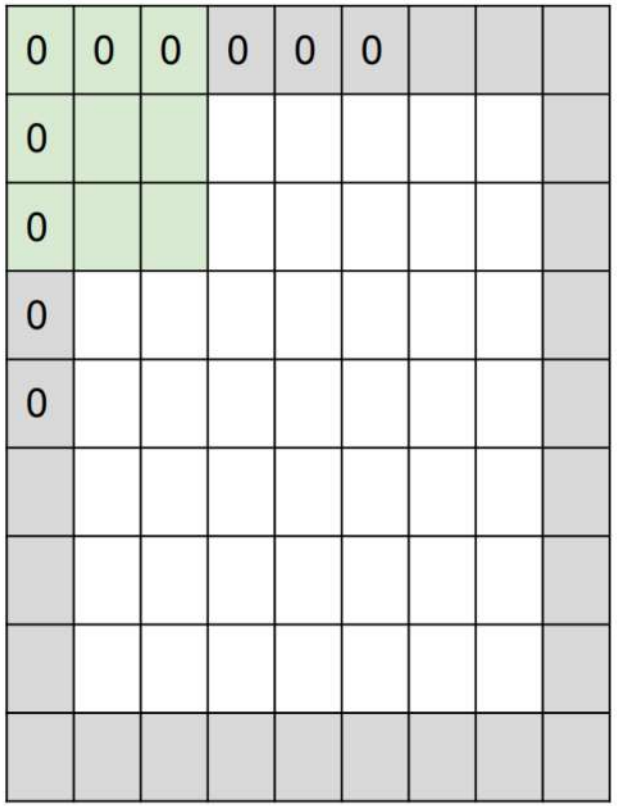
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

单字符识别

- CNN模型

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

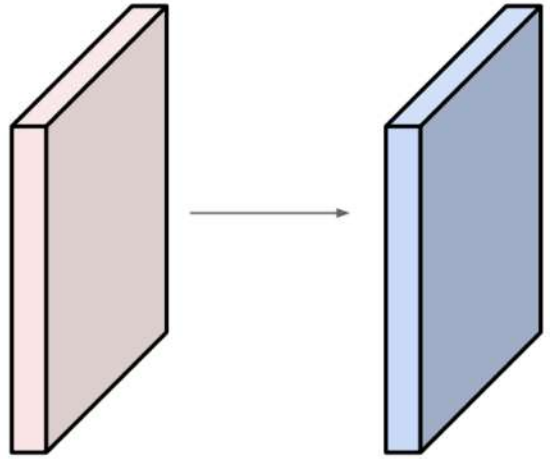
单字符识别

- CNN模型

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?



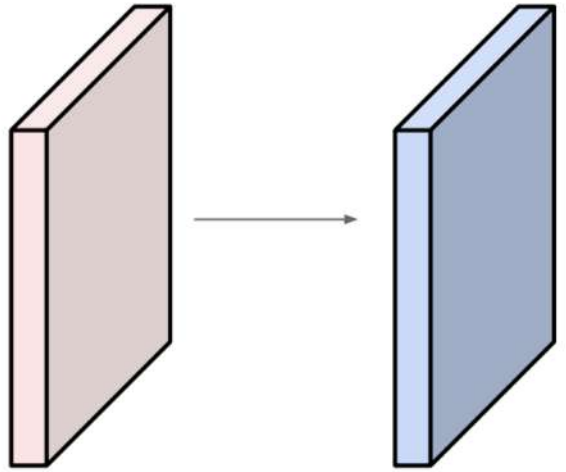
单字符识别

- CNN模型

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10



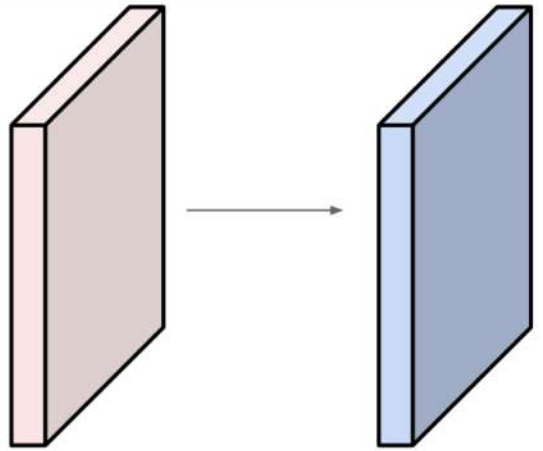
单字符识别

- CNN模型

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

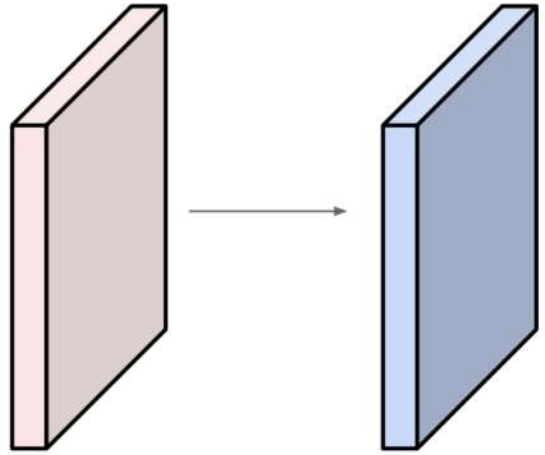
Number of parameters in this layer?



单字符识别

- CNN模型

Examples time:

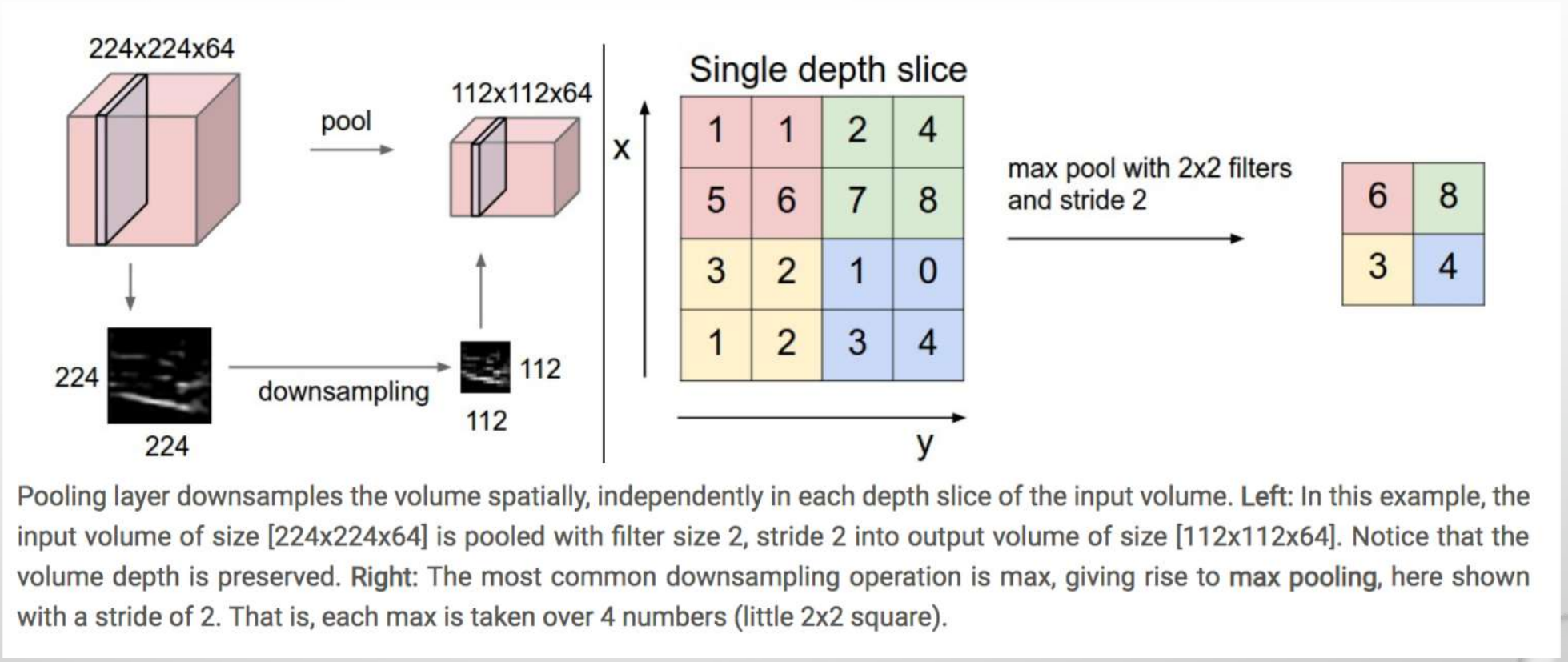


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

单字符识别

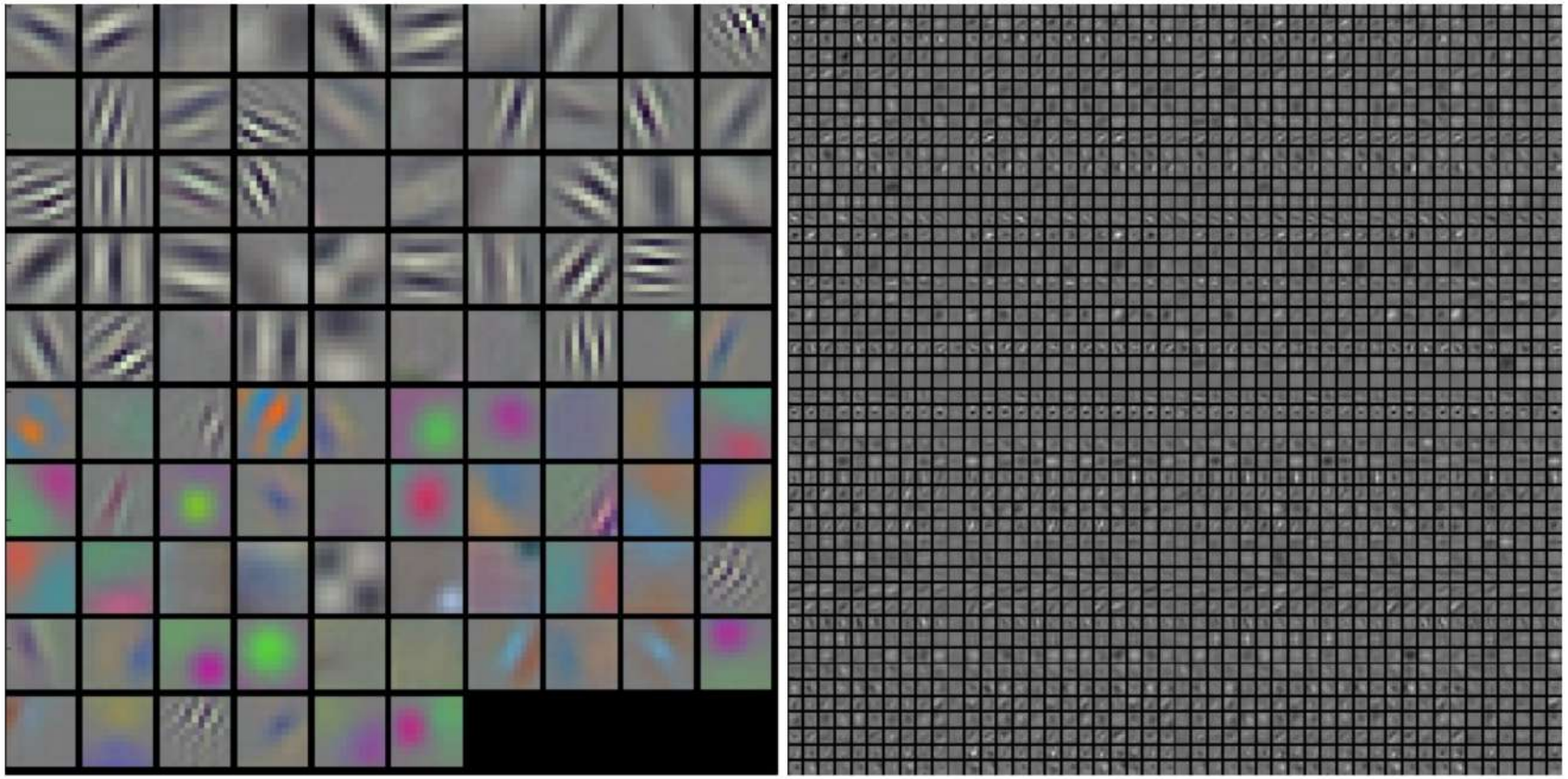
- CNN模型



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

单字符识别

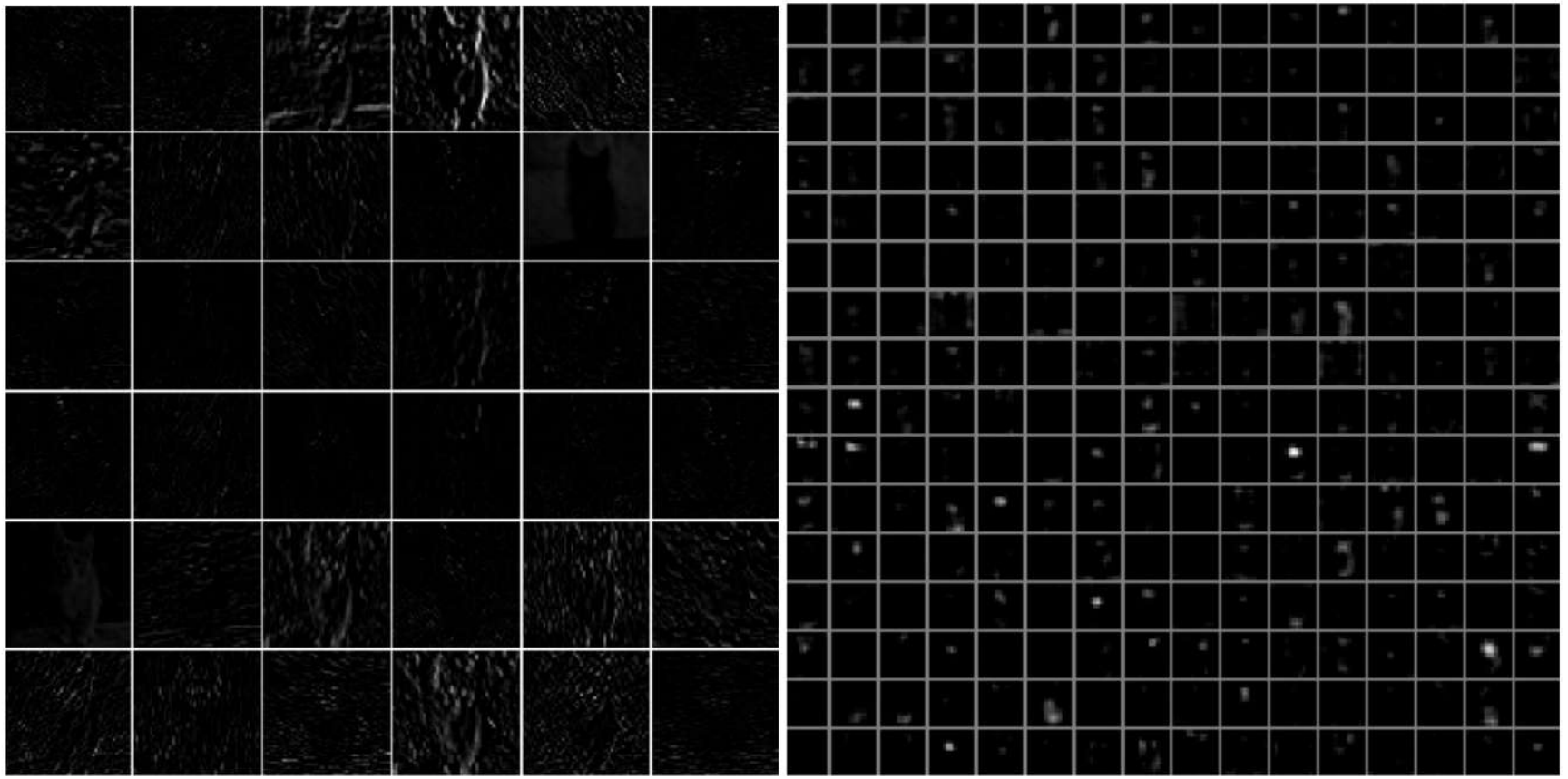
- CNN模型



Typical-looking filters on the first CONV layer (left), and the 2nd CONV layer (right) of a trained AlexNet. Notice that the first-layer weights are very nice and smooth, indicating nicely converged network. The color/grayscale features are clustered because the AlexNet contains two separate streams of processing, and an apparent consequence of this architecture is that one stream develops high-frequency grayscale features and the other low-frequency color features. The 2nd CONV layer weights are not as interpretable, but it is apparent that they are still smooth, well-formed, and absent of noisy patterns.

单字符识别

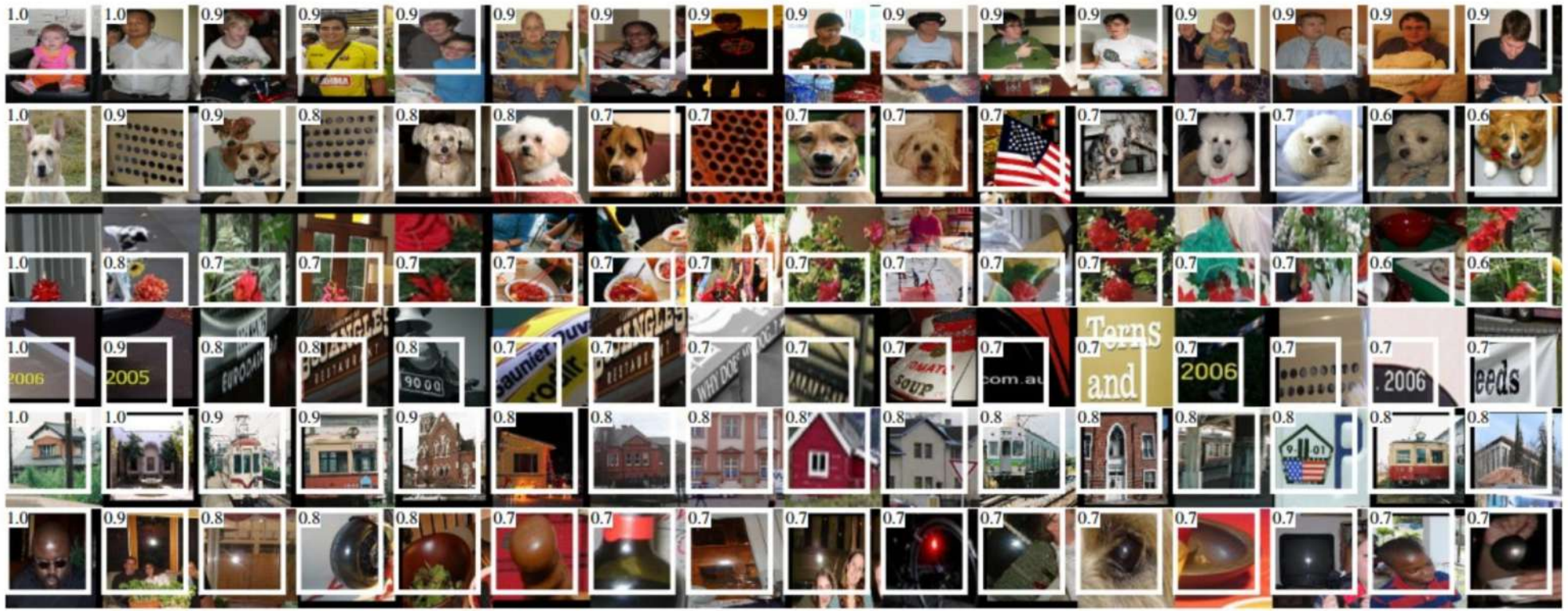
- CNN模型



Typical-looking activations on the first CONV layer (left), and the 5th CONV layer (right) of a trained AlexNet looking at a picture of a cat. Every box shows an activation map corresponding to some filter. Notice that the activations are sparse (most values are zero, in this visualization shown in black) and mostly local.

单字符识别

- CNN模型



Maximally activating images for some POOL5 (5th pool layer) neurons of an AlexNet. The activation values and the receptive field of the particular neuron are shown in white. (In particular, note that the POOL5 neurons are a function of a relatively large portion of the input image!) It can be seen that some neurons are responsive to upper bodies, text, or specular highlights.

单字符识别

- CNN模型
  - DEEP VISUALIZATION TOOLBOX
    - HTTPS://WWW.YOUTUBE.COM/WATCH?V=AGKFIQ4IGAM&T=7S
  - CONVNETJS
    - HTTP://CS.STANFORD.EDU/PEOPLE/KARPATHY/CONVNETJS/

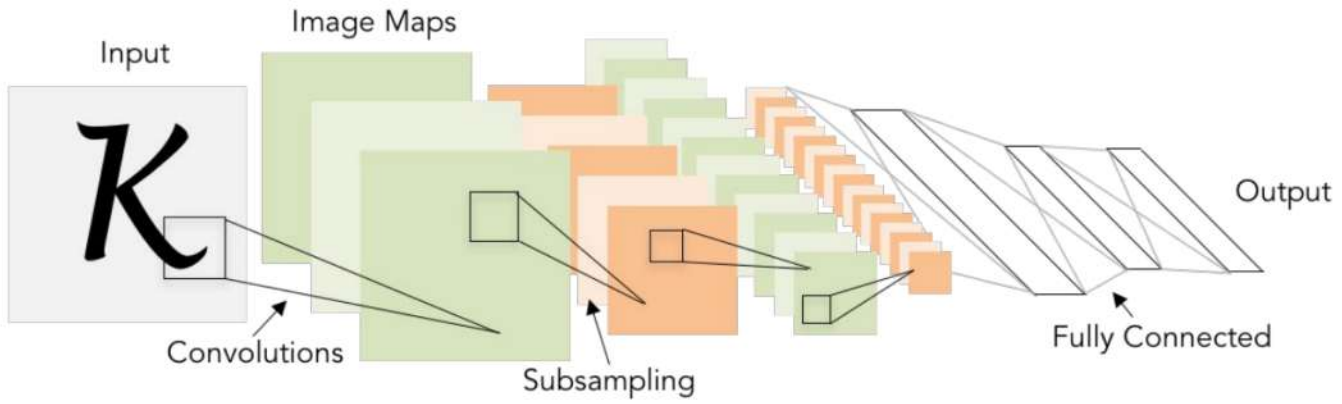
单字符识别

Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

- CNN模型

Gradient-based learning applied to document recognition

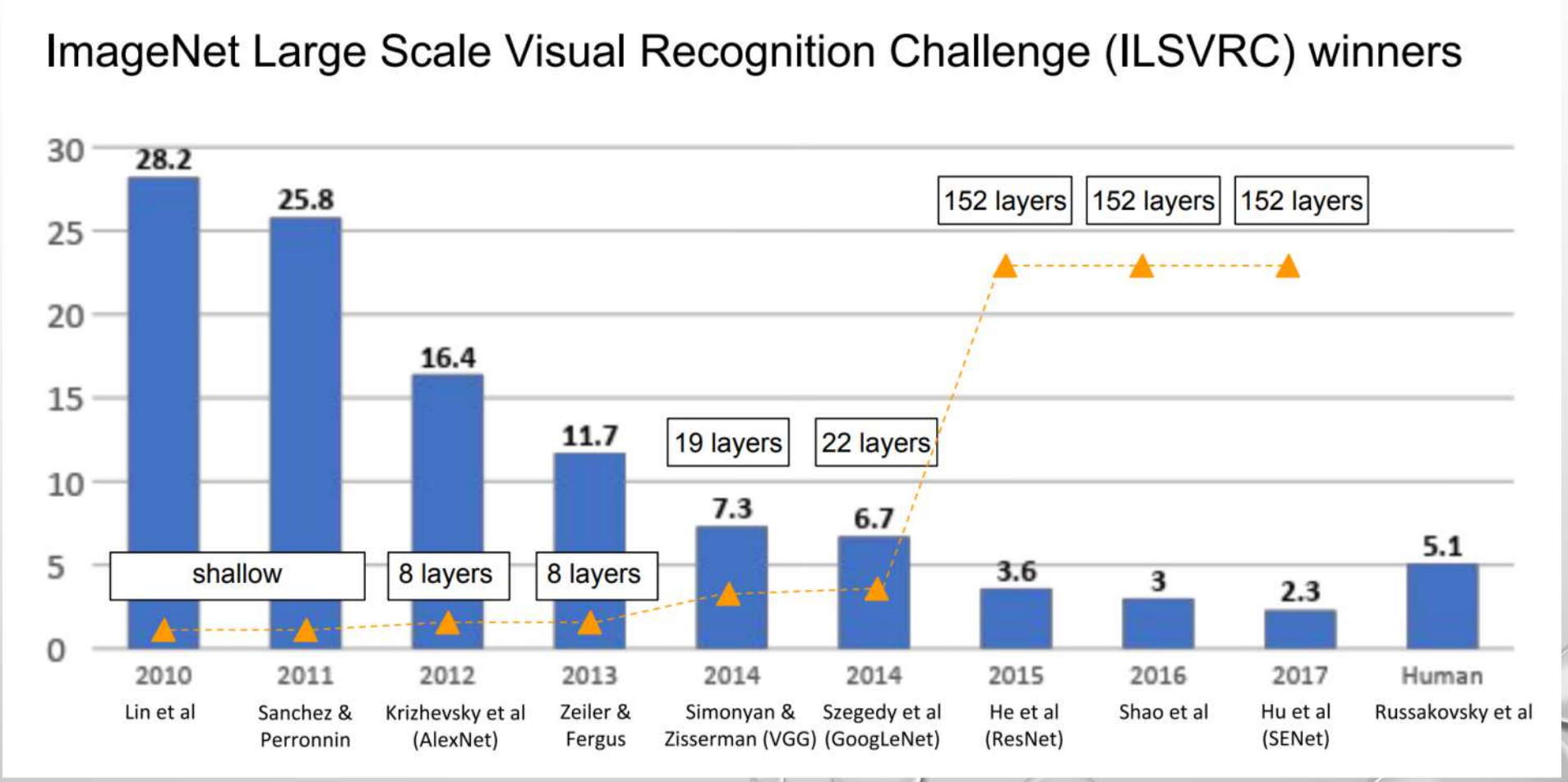
[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

单字符识别

- CNN模型


ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al

单字符识别

- CNN模型

Case Study: VGGNet

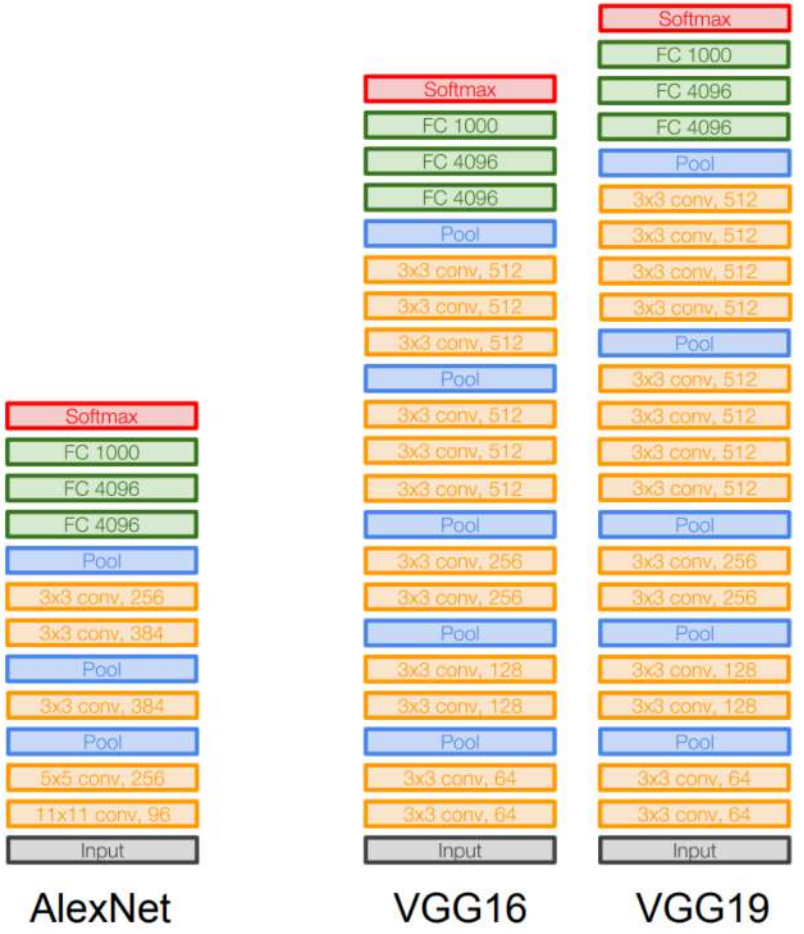
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



单字符识别

- CNN模型

Case Study: VGGNet

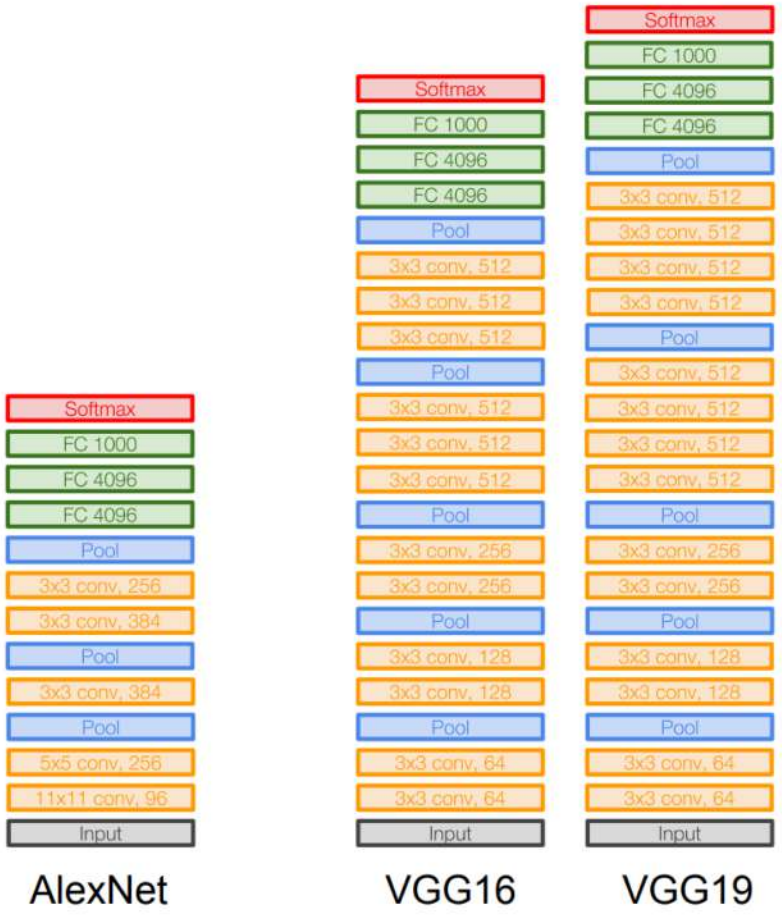
[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

But deeper, more non-linearities

And fewer parameters: $3 * (3^2C^2)$ vs. 7^2C^2 for C channels per layer



单字符识别

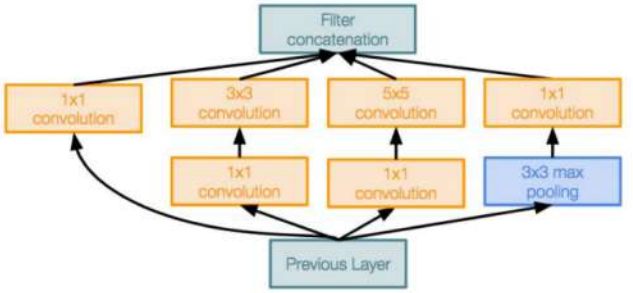
- CNN模型

Case Study: GoogLeNet

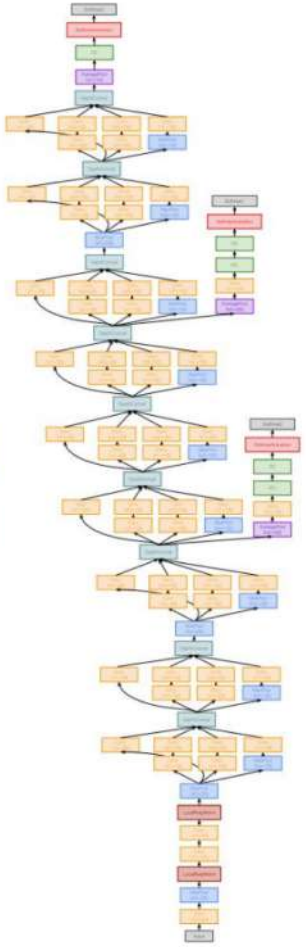
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC’14 classification winner
  (6.7% top 5 error)



Inception module

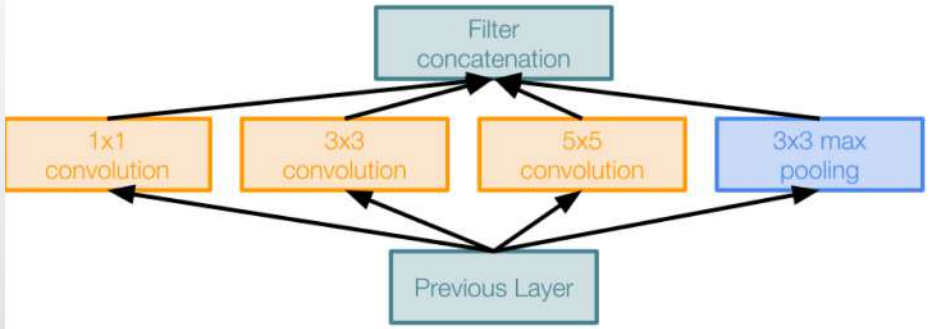


单字符识别

- CNN模型

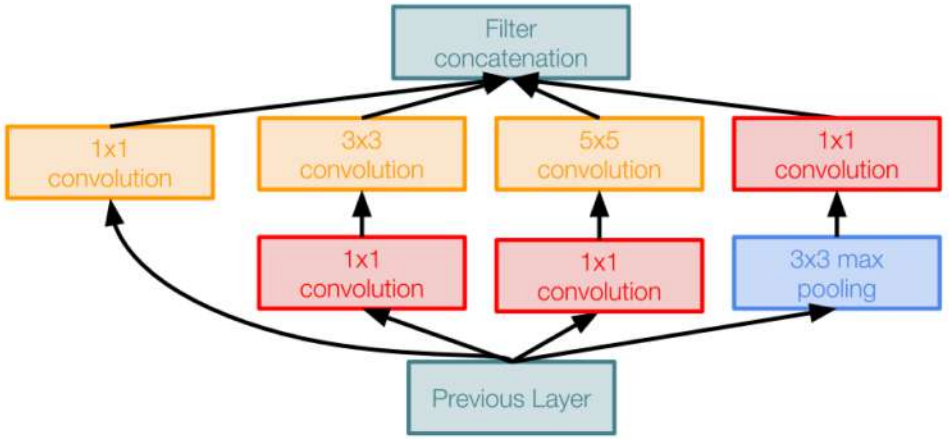
Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

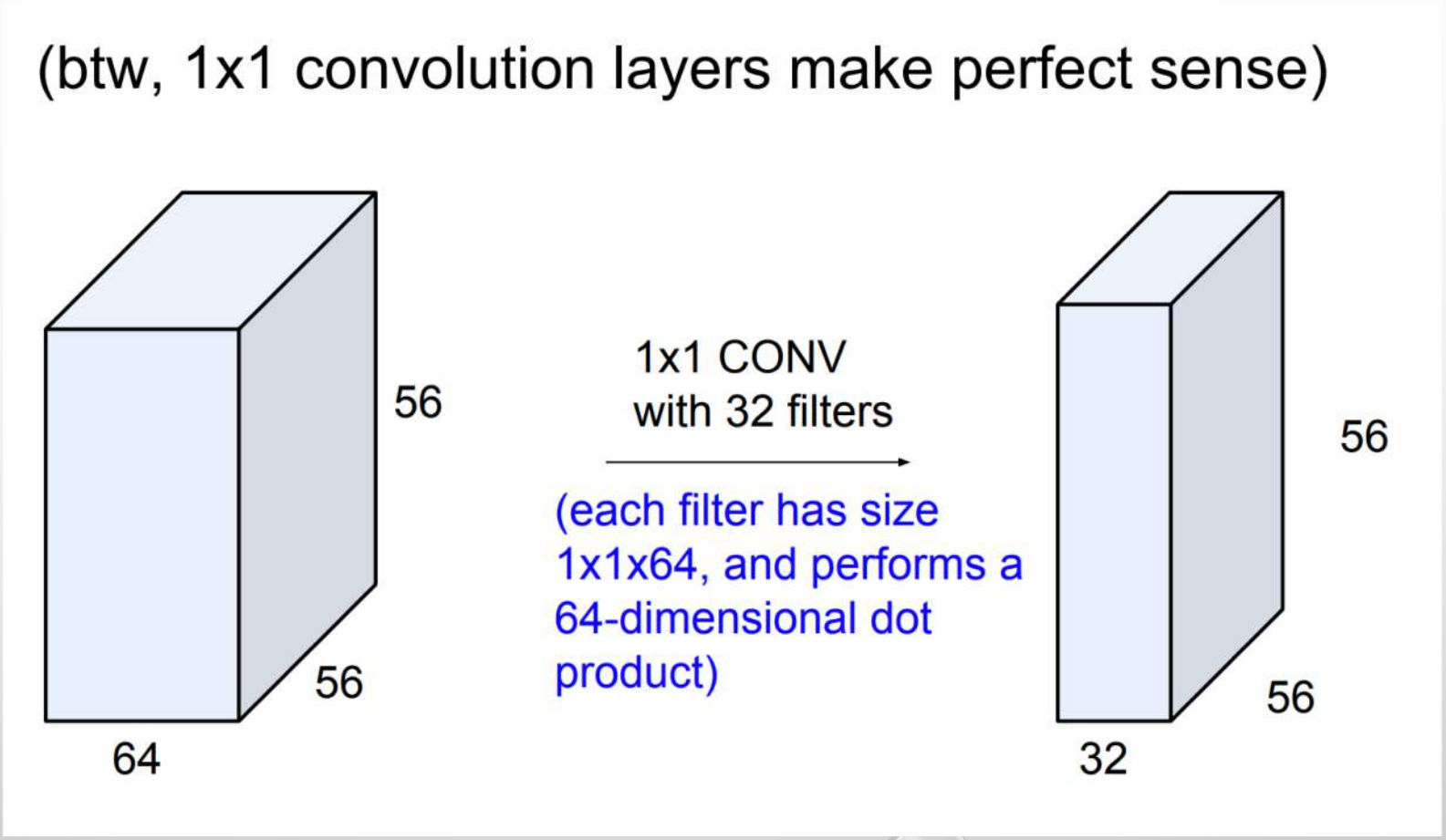
1x1 conv "bottleneck" layers



Inception module with dimension reduction

单字符识别

- CNN模型

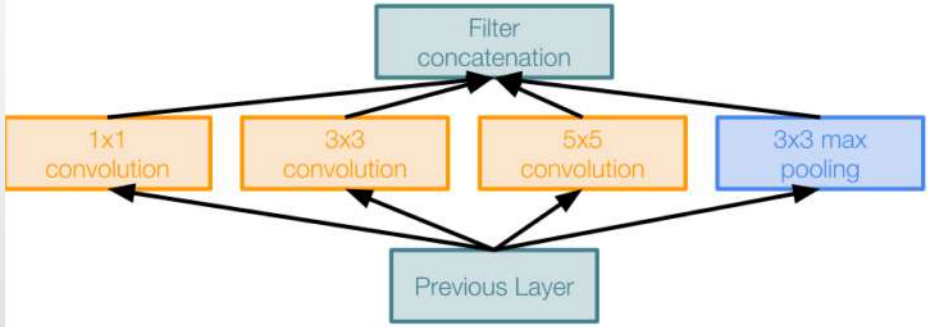

(btw, 1x1 convolution layers make perfect sense)
56
56
64
1x1 CONV
with 32 filters
(each filter has size
1x1x64, and performs a
64-dimensional dot
product)
56
56
32

单字符识别

- CNN模型

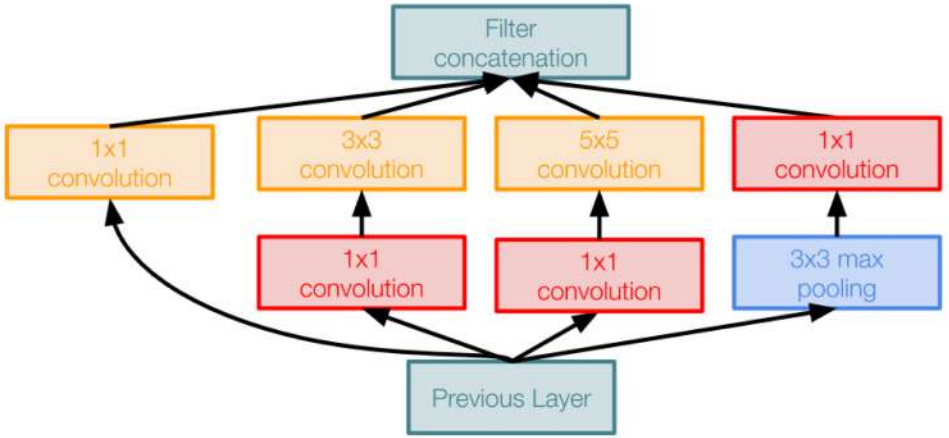
Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

1x1 conv "bottleneck" layers



Inception module with dimension reduction

单字符识别

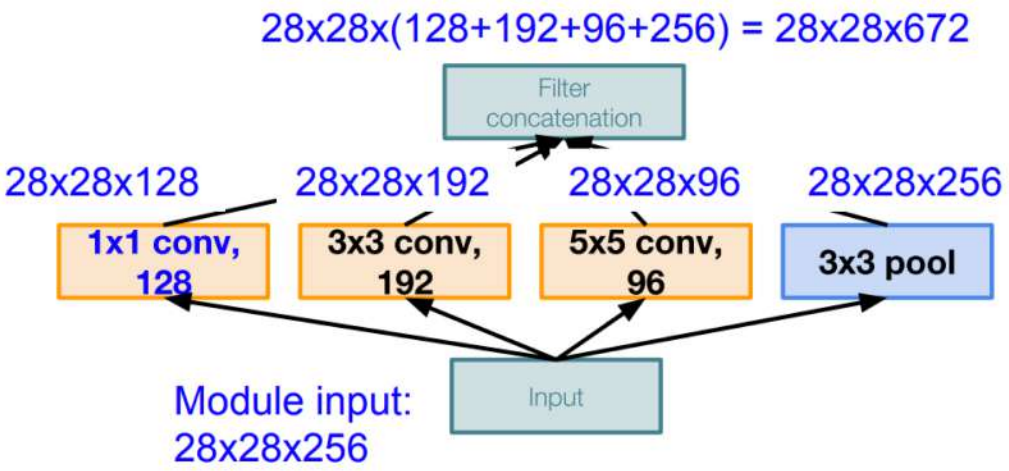
- CNN模型

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3:What is output size after filter concatenation?



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Very expensive compute

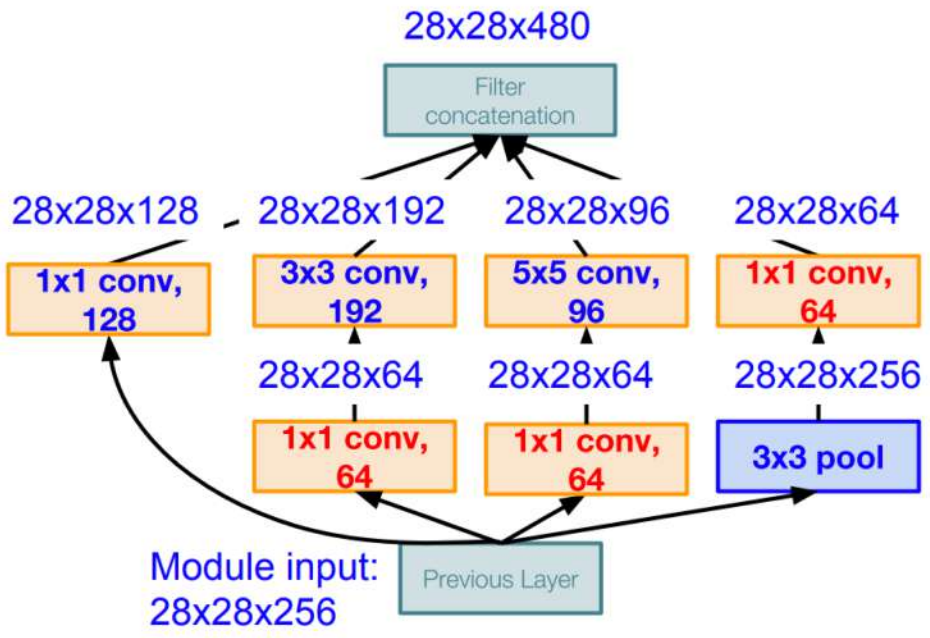
Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

单字符识别

- CNN模型

Case Study: GoogLeNet

[Szegedy et al., 2014]



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

单字符识别

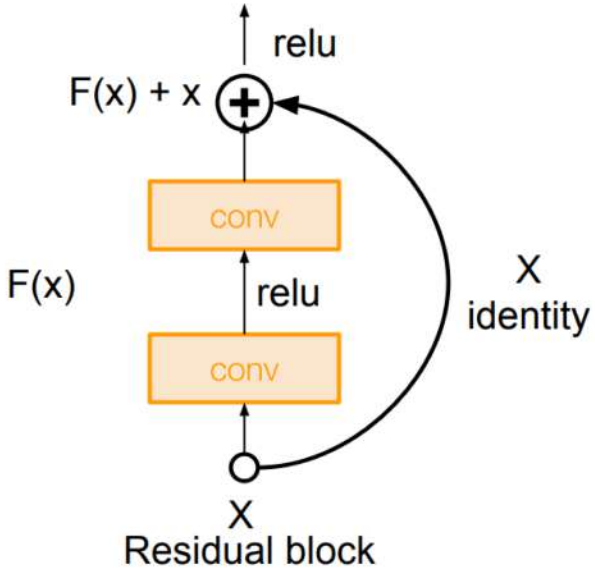
- CNN模型

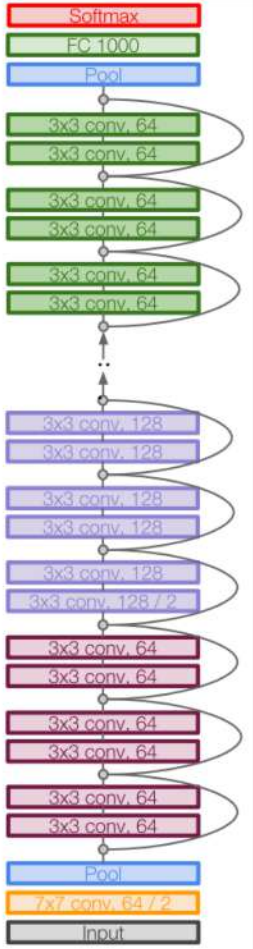
Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

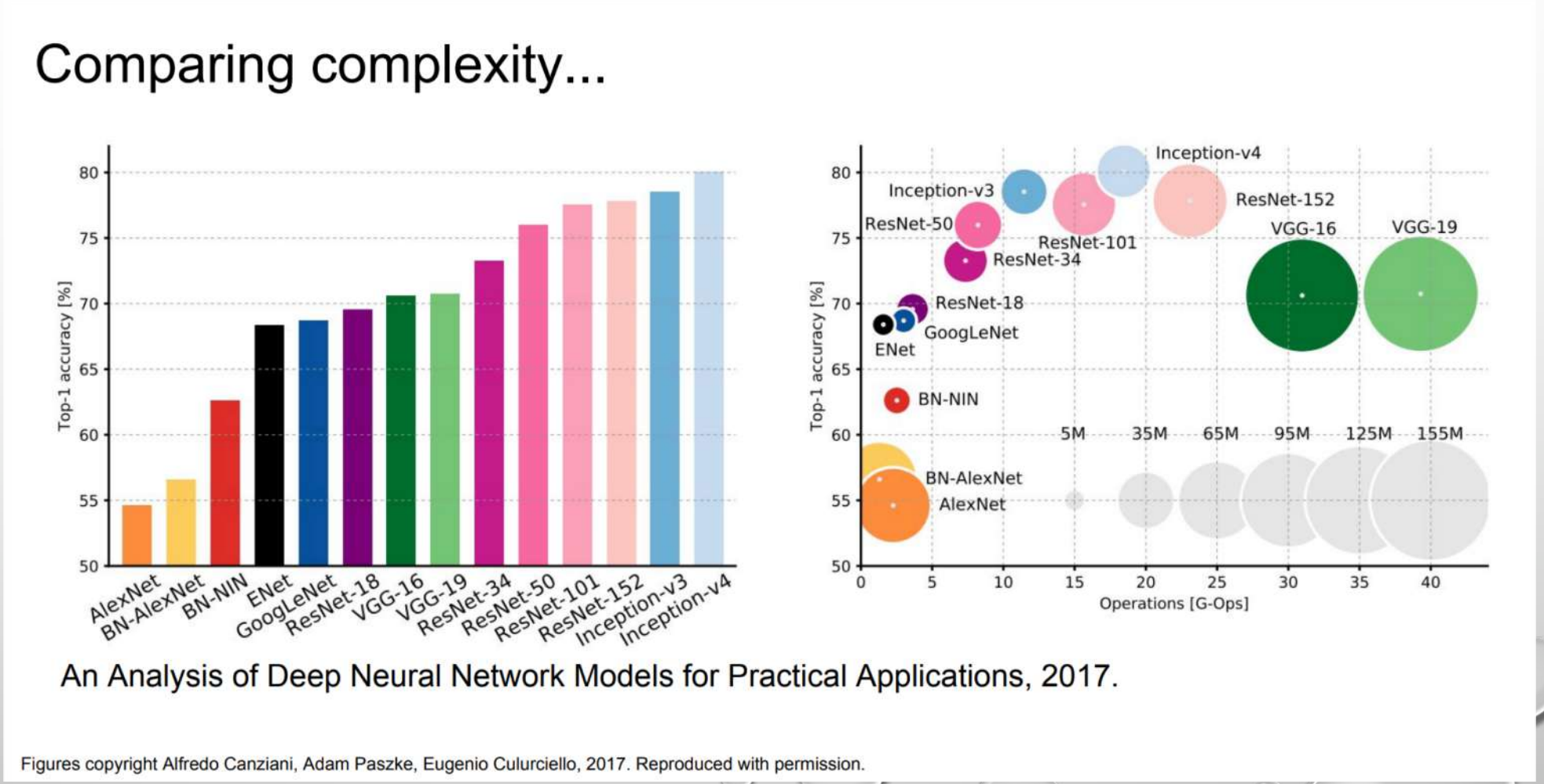
- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!





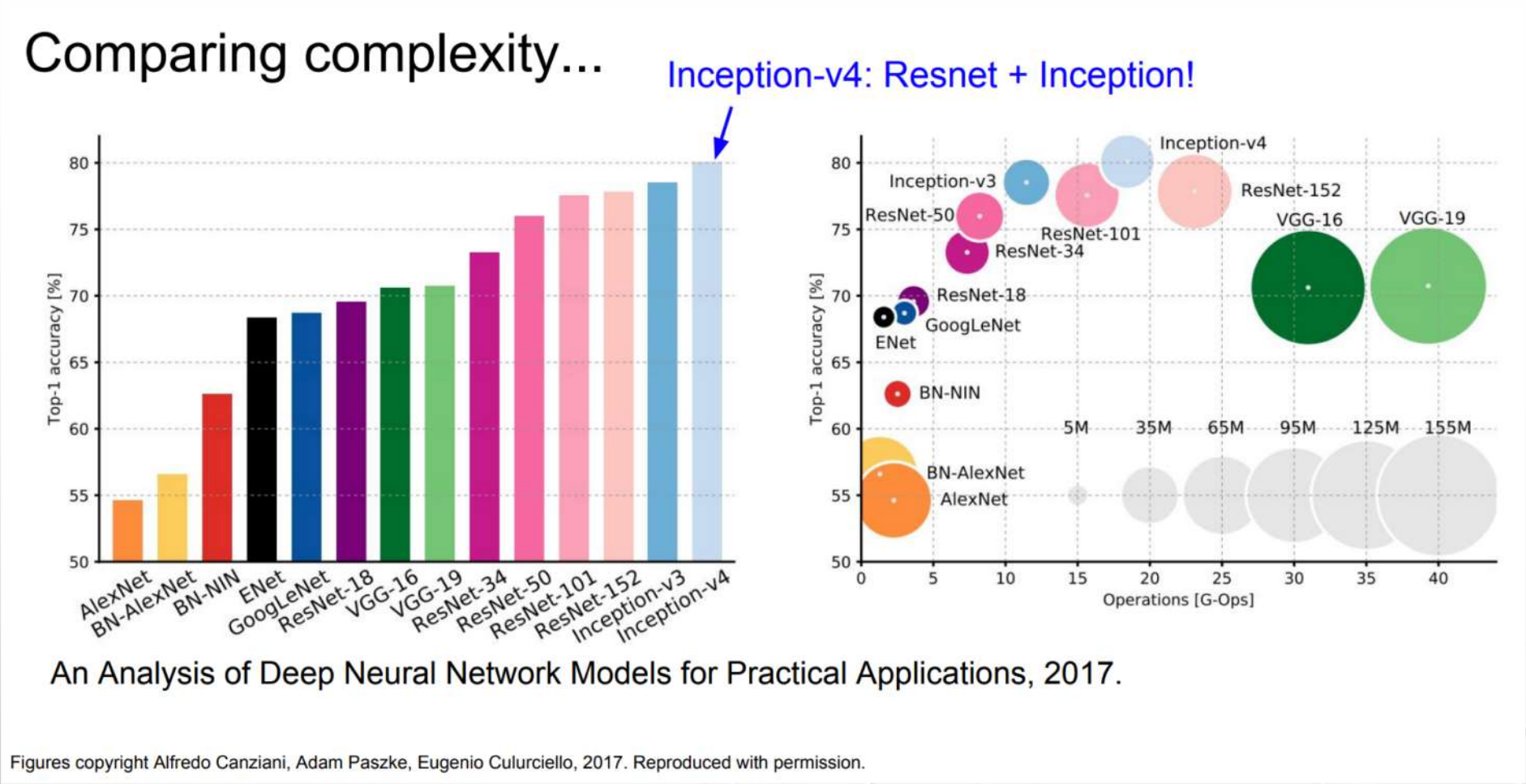
单字符识别

- CNN模型



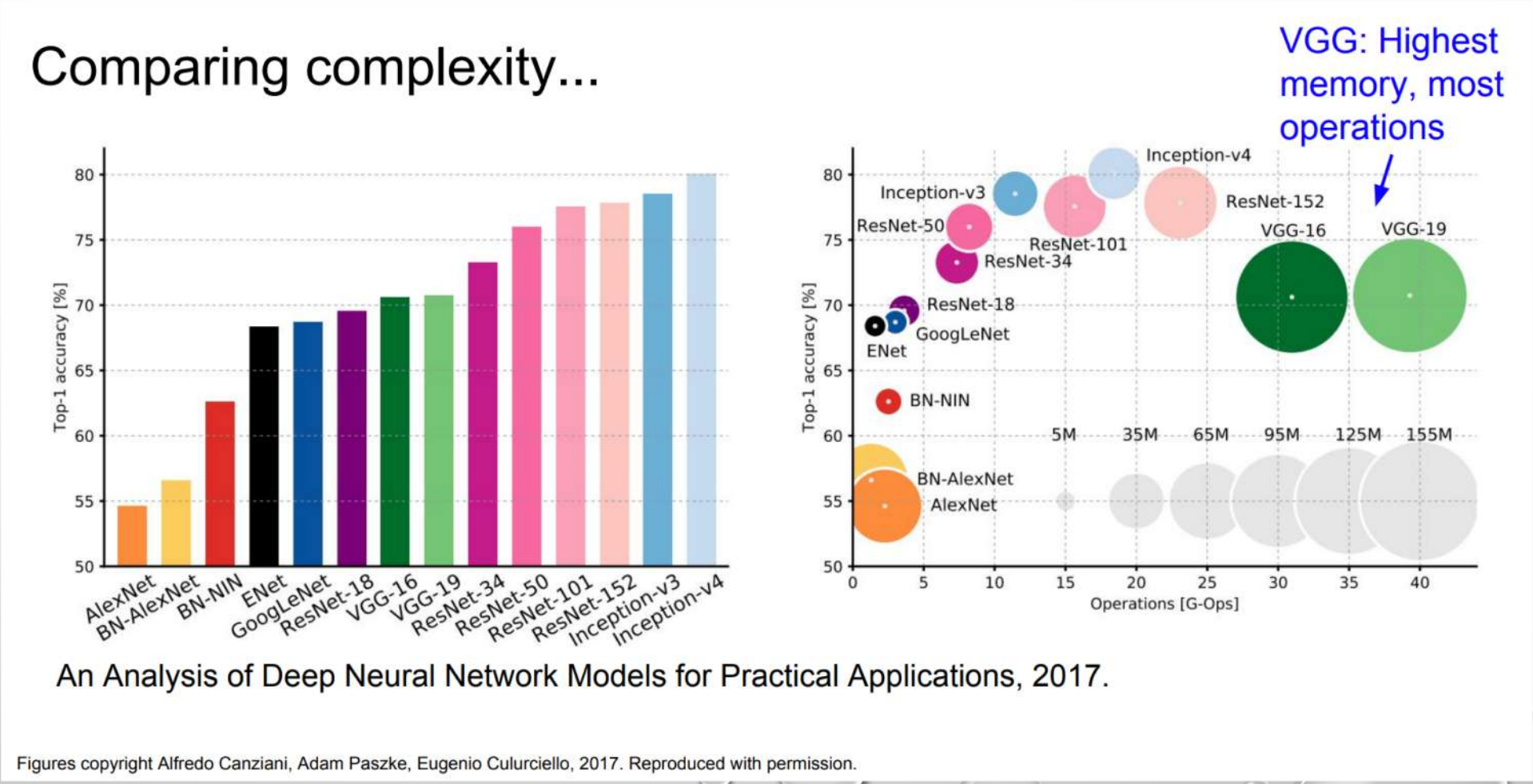
单字符识别

- CNN模型



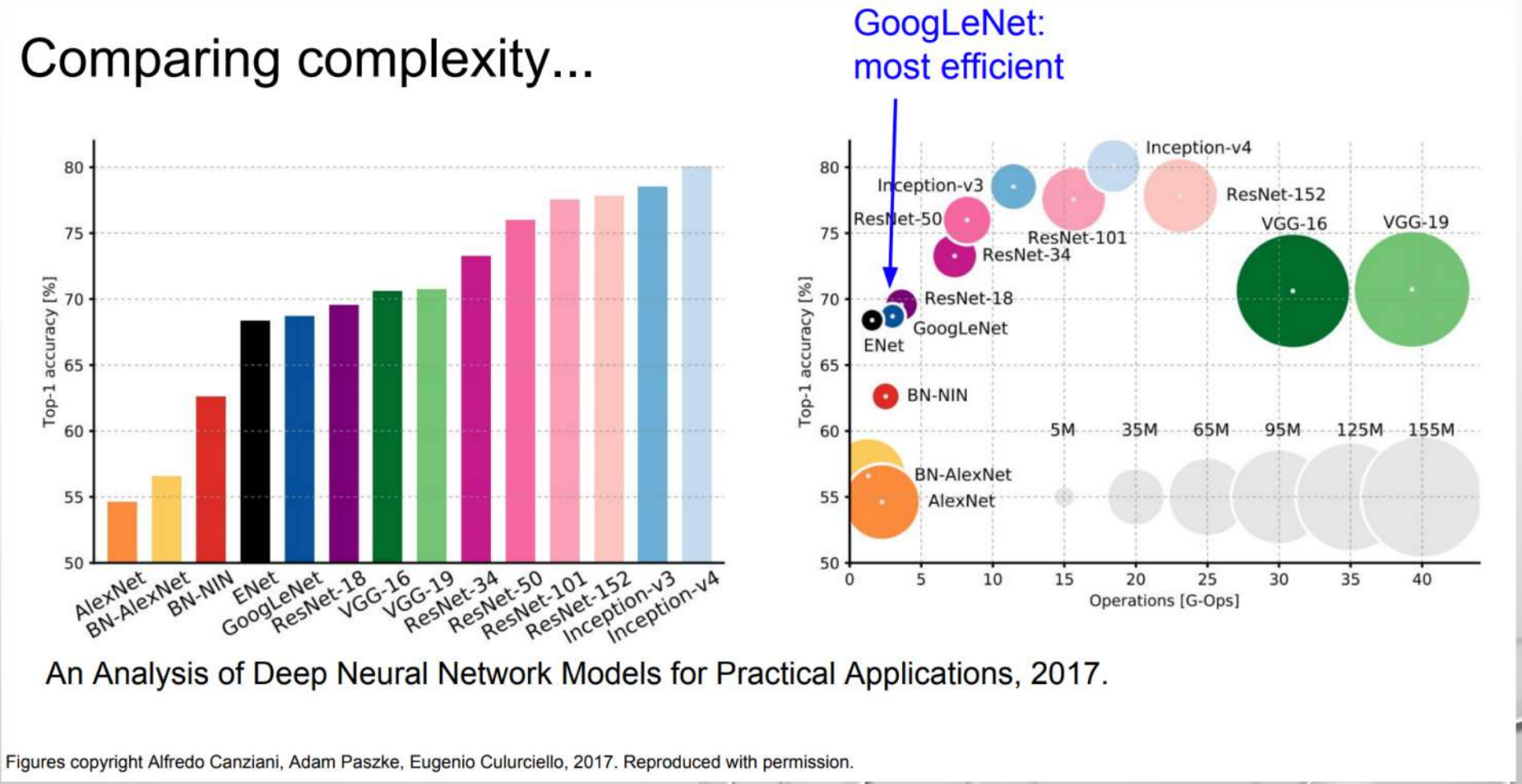
单字符识别

- CNN模型



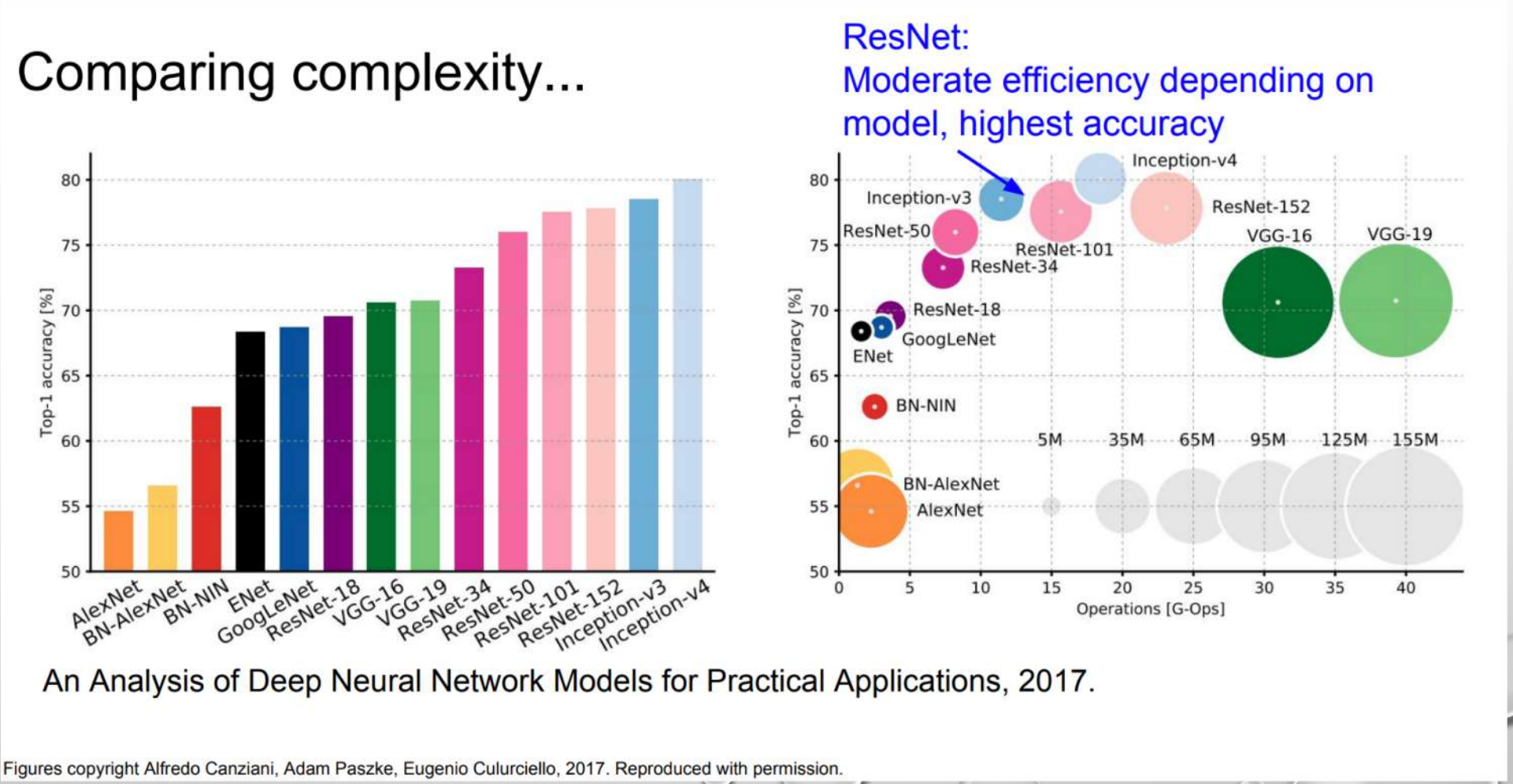
单字符识别

- CNN模型



单字符识别

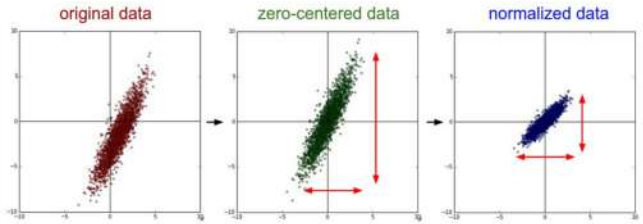
- CNN模型



单字符识别

- CNN模型
  - 输入图片归一化
  - 数据增强
  - PADDING
  - 灰度图还是二值图？
  - 英文单词可以用CNN学？

    宽高比变化范围大怎么办？



休息一下

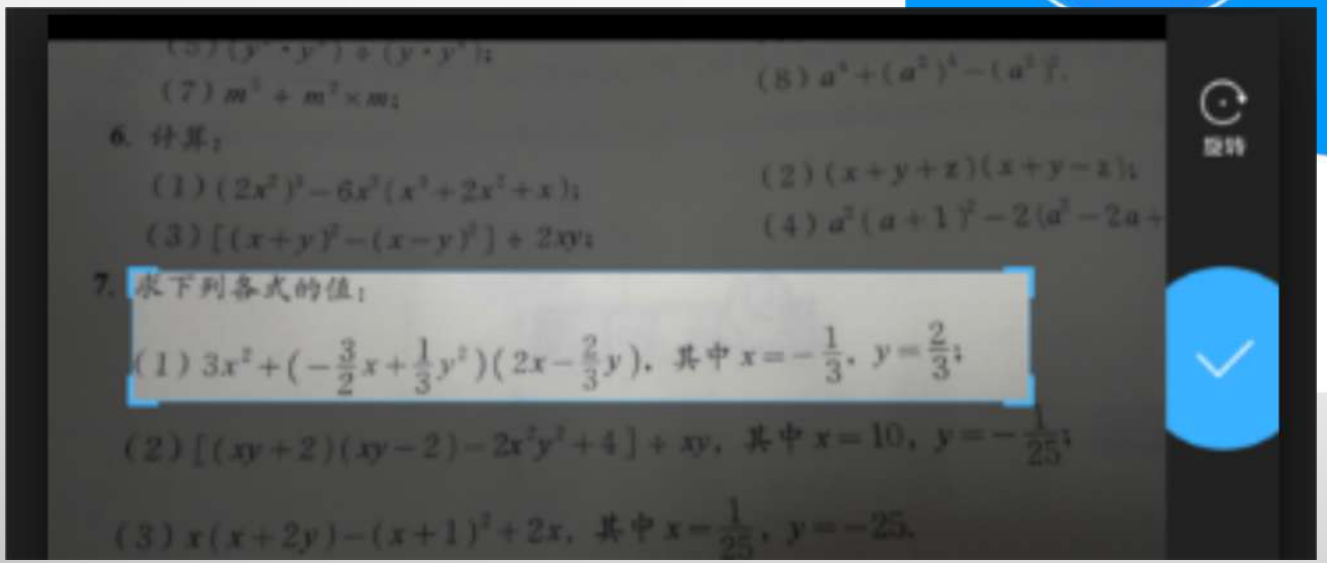
- DEEP DREAM
  - HTTPS://WWW.YOUTUBE.COM/WATCH?V=BSSMBPMPEYQ&T=683S

第二课 单字符识别

- 单字符识别
  - 模版匹配
  - 特征提取 + 传统分类器
  - CNN模型
- 过切分的路径选择
  - VITERBI + BEAM SEARCH
- 语言模型
  - N-GRAM
  - LSTM
  - 2D-PCFG

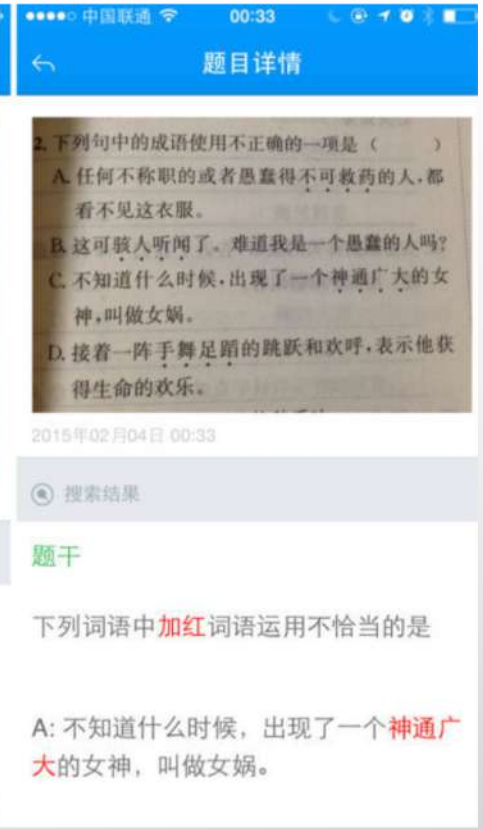
实例一

- 通用文本识别

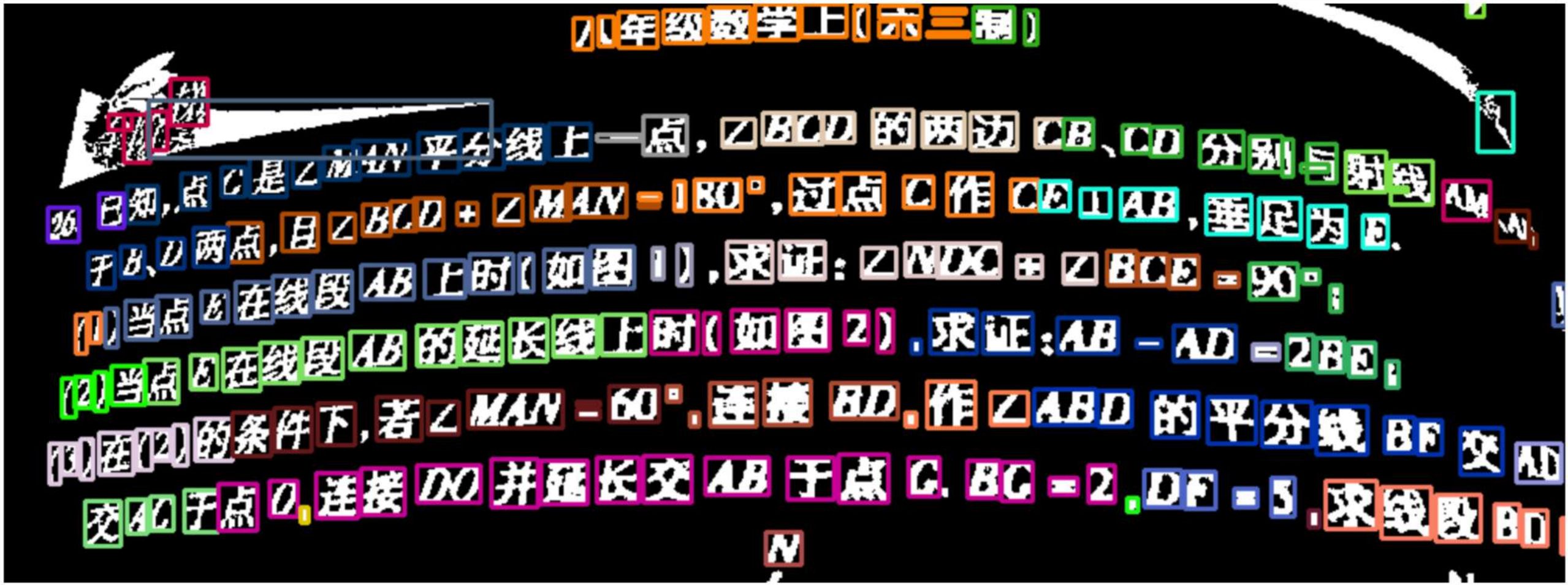

旋转


中国联通 00:33
拍照搜题


中国联通 00:36
题目详情
Wang Lin is a good 1 . She is twenty-one years old. She drives a car in a 2 . She 3 from Sunday to Friday. Her home isn't 4 the factory. She gets up 5 at six. She goes to work at 6:30. 6 7:30 she must get there. She has 7 in the factory. She has lunch there, too. She 8 the factory at 5:00 in the afternoon. She cooks and then does housework in the evening. She likes 9 very much. She 10 to bed at about 10:30 pm.
2015年02月04日 00:36
搜索结果
完形填空。
Wang Lin is a good 1. She is twenty-one years old. She drives a car in a 2. She 3 from Sunday to Friday. Her home is n't 4 the factory. She gets up 5 at six

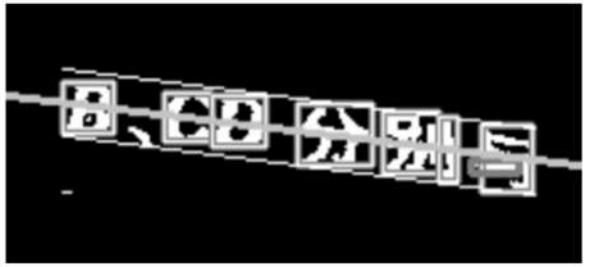

中国联通 00:33
题目详情
2. 下列句中的成语使用不正确的一项是（ ）
A. 任何不称职的或者愚蠢得不可救药的人，都看不见这衣服。
B. 这可骇人听闻了。难道我是一个愚蠢的人吗？
C. 不知道什么时候，出现了一个神通广大的女神，叫做女娲。
D. 接着一阵手舞足蹈的跳跃和欢呼，表示他获得生命的欢乐。
2015年02月04日 00:33
搜索结果
题干
下列词语中加红词语运用不恰当的是
A: 不知道什么时候，出现了一个神通广大的女神，叫做女娲。

(Sub) Text Line Extraction



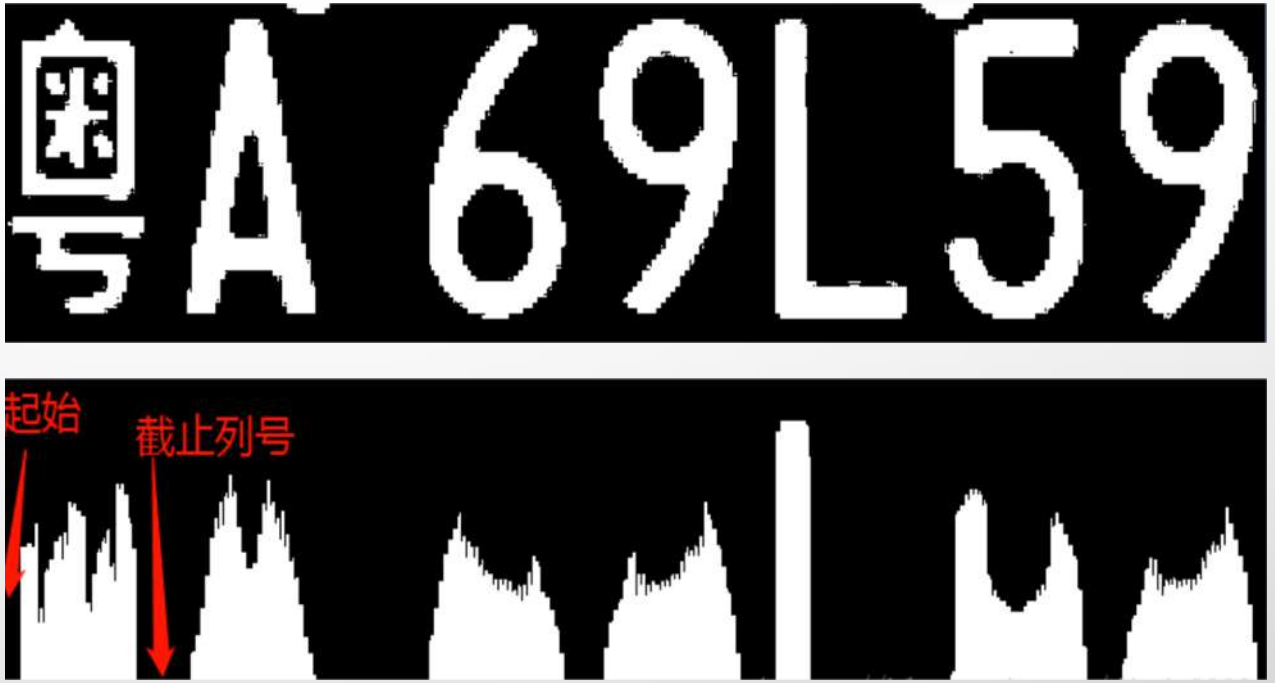
Character Segmentation

基于行片段拟合边界，再次对连通域进行划分



Character Segmentation

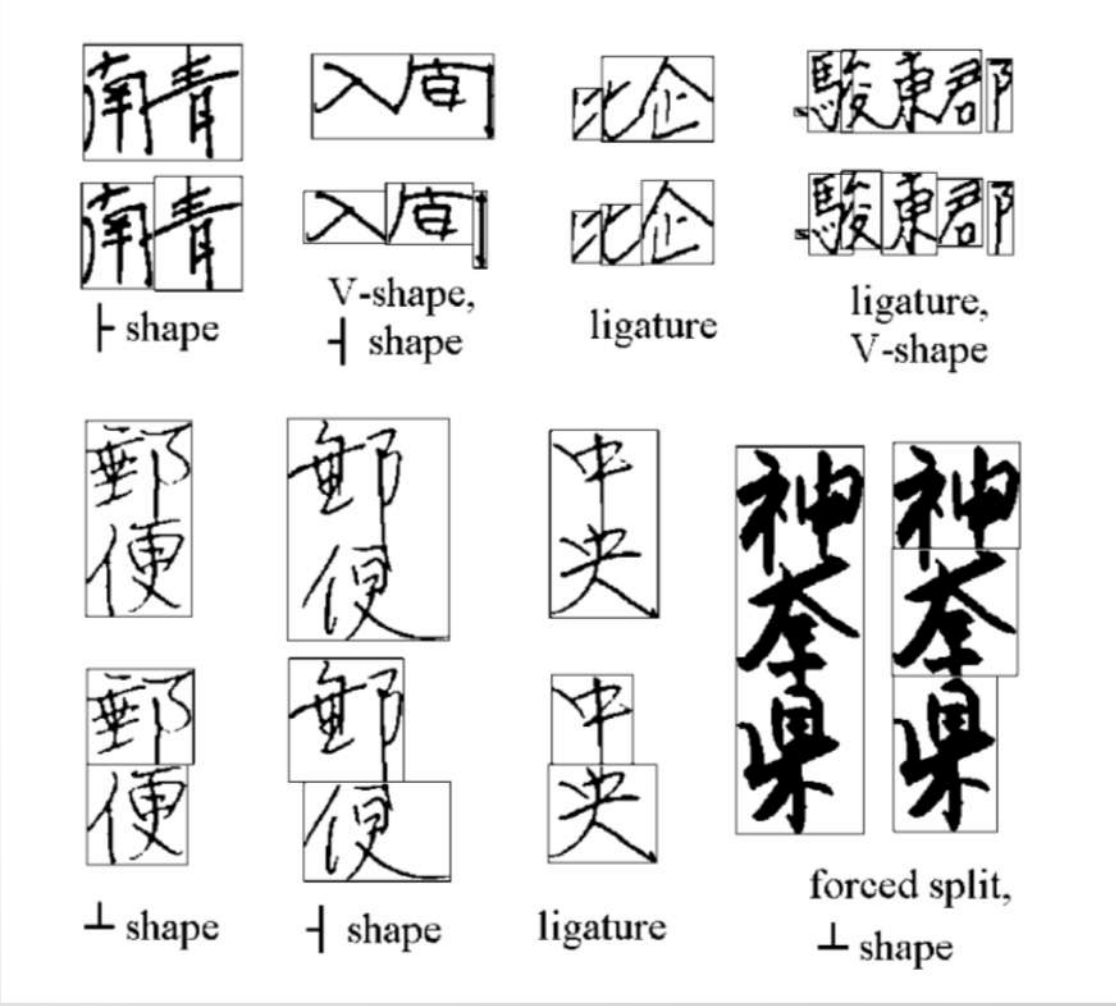
计算分割点置信度，在粗分割后，递归地以置信度最大点进行分割



投影法

Character Segmentation

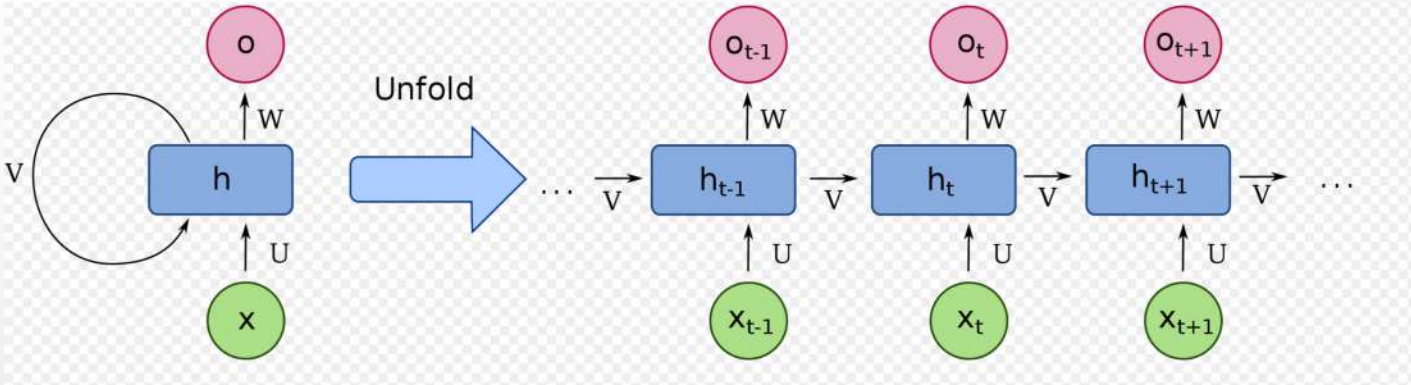
计算分割点置信度，在粗分割后，递归地以置信度最大点进行分割



C.-L. Liu, M. Koga, and H. Fujisawa, Lexicon-Driven Segmentation and Recognition of Handwritten Character Strings for Japanese Address Reading, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.24, 1425-1437, 2002

Character Segmentation

计算分割点置信度，在粗分割后，递归地以置信度最大点进行分割



遗留问题

- 字符分割

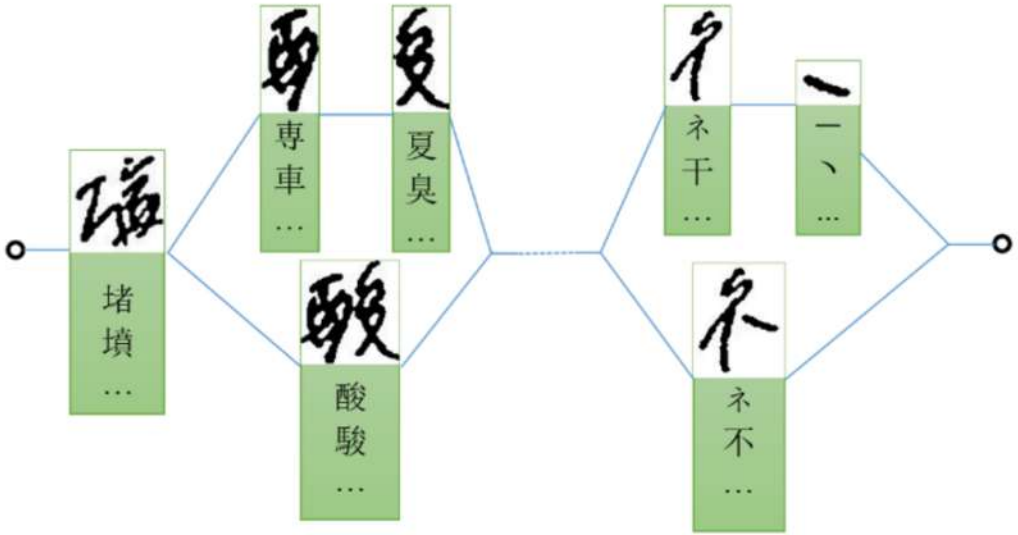
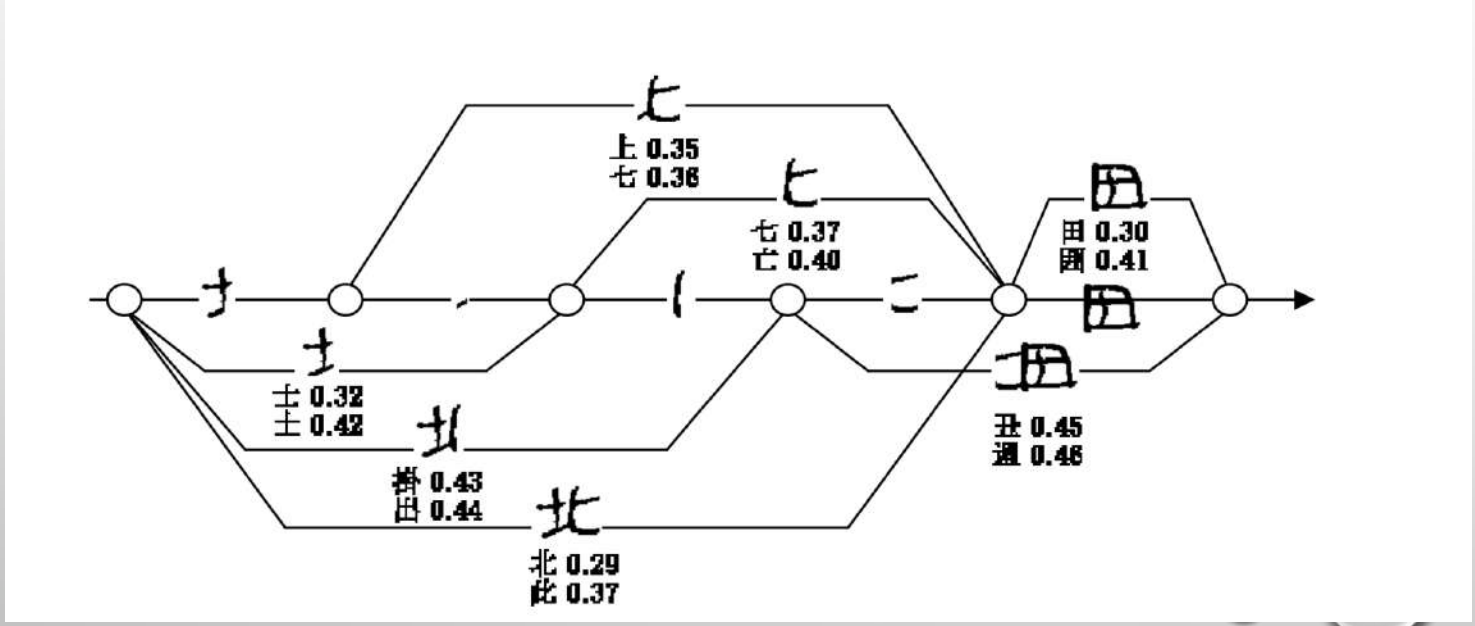


Fig. 11: Candidate lattice diagram.

路径选择

- VITERBI算法 + BEAM SEARCH

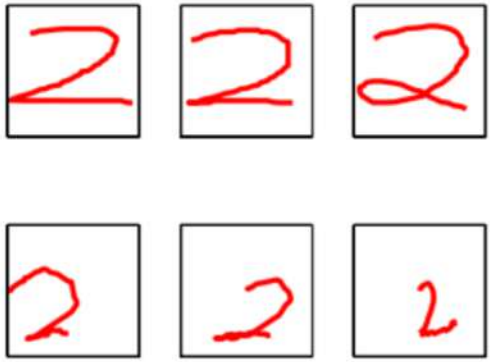
  多步骤每步多选择模型的最优选择问题，其在每一步的所有选择都保存了前续所有步骤到当前步骤当前选择的最小总代价（或者最大价值）以及当前代价的情况下前继步骤的选择。
  依次计算完所有步骤后，通过回溯的方法找到最优选择路径。



路径选择

- HMM + VITERBI算法

Top row: examples of on-line handwritten digits. Bottom row: synthetic digits sampled generatively from a left-to-right hidden Markov model that has been trained on a data set of 45 handwritten digits.



The most natural unit of handwriting is a letter. A letter is represented by a 7-state left-to-right HMM. The HMM model is illustrated in Figure 2-3.

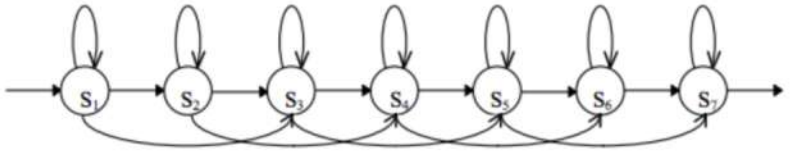


Figure 2-3: A 7-state HMM for a letter.

The left-to-right type of HMM, a special class of HMMs, have an additional property that the state index is non-decreasing as the time increases, *i.e.*

$$a_{ij} = P(Q_n = S_j | Q_{n-1} = S_i) = 0, \quad i > j. \qquad (2.31)$$

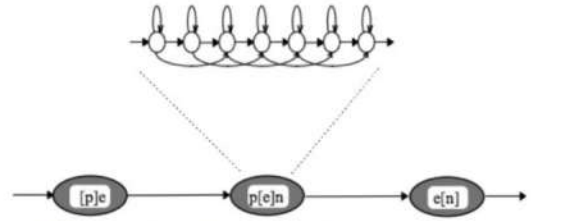


Figure 2-5: Illustration of an HMM modeling of the word *"pen"*, consists of three 7-state HMMs for the trigraph *" [p]e", "p[e]n"*, and *"e[n] "*.

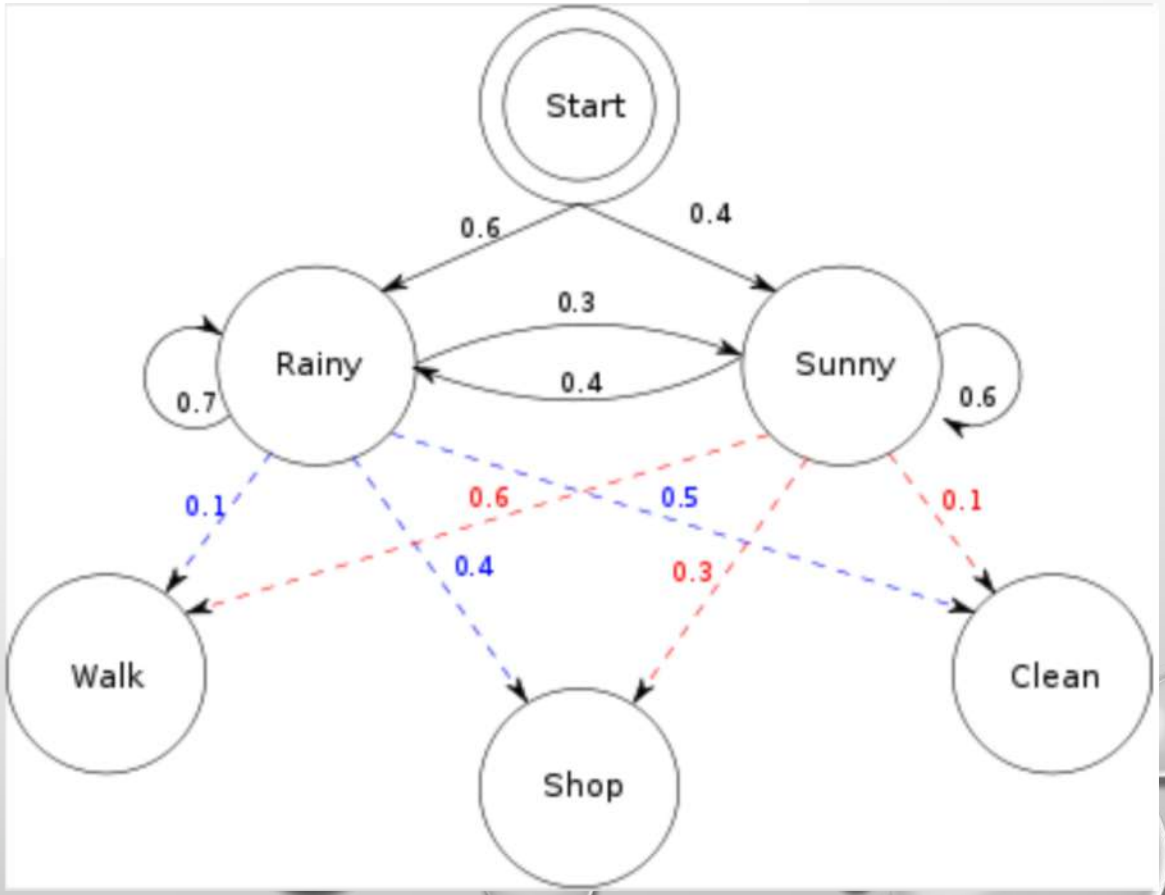
"PRML", P.615, 13.2. Hidden Markov Models

HMM 模型

隐马尔可夫模型由初始状态概率向量 π、状态转移概率矩阵 A 和观测概率矩阵 B 决定．π 和 A 决定状态序列，B 决定观测序列．因此，隐马尔可夫模型 λ 可以用三元符号表示，即

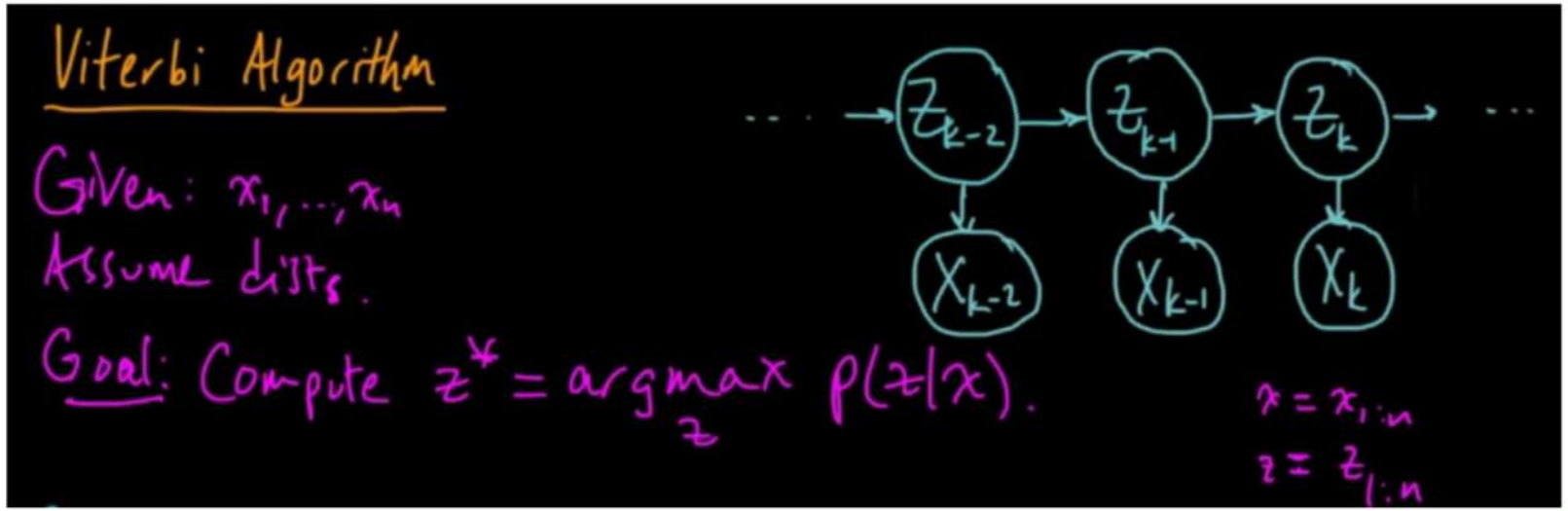
$$\lambda = (A, B, \pi)$$

A, B, π 称为隐马尔可夫模型的三要素．



HMM + VITERBI

Viterbi Algorithm



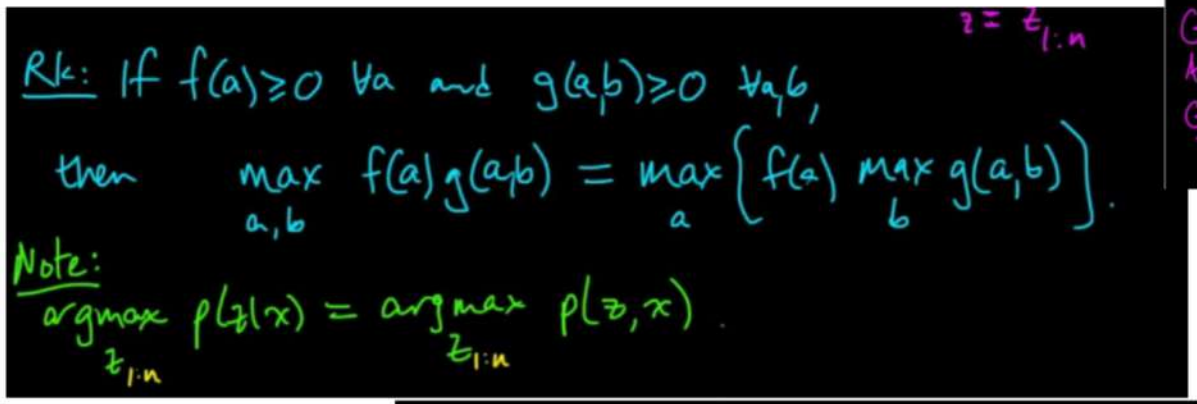
Rk: If $f(a) \geq 0 \ \forall a$ and $g(a,b) \geq 0 \ \forall a,b$,

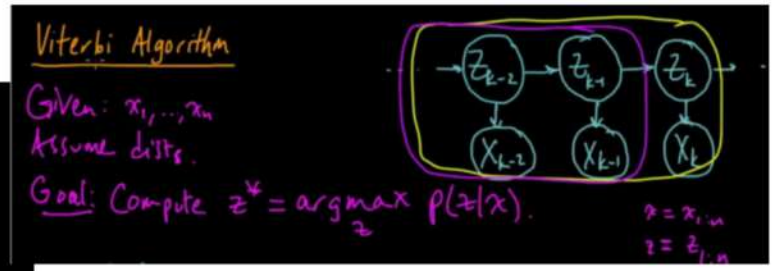
then $\max_{a,b} f(a)g(a,b) = \max_a \left\{ f(a) \max_b g(a,b) \right\}$.

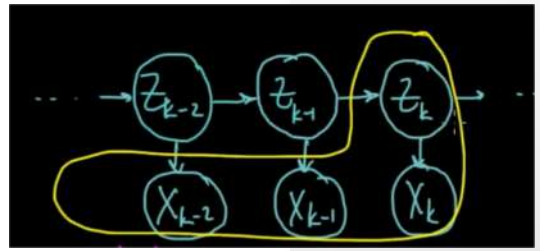
https://www.youtube.com/watch?v=RwwfUICZLsA&nohtml5=False

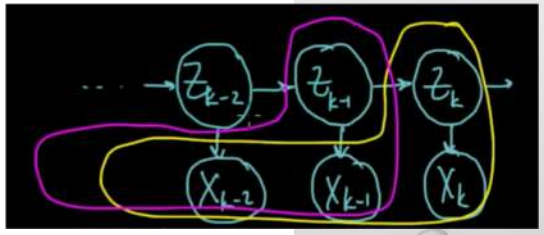
HMM + VITERBI

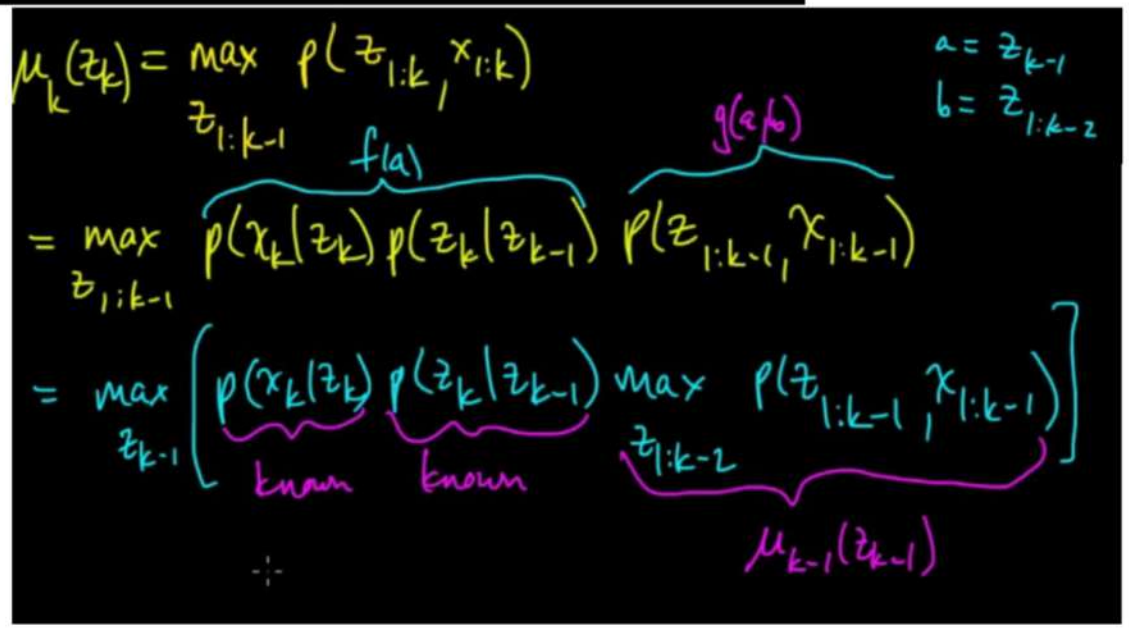
-


RK: If f(a) ≥ 0 ∀a and g(a,b) ≥ 0 ∀a,b,
then max f(a) g(a,b) = max [f(a) max g(a,b)].
Note:
argmax p(z|x) = argmax p(z,x)


Viterbi Algorithm
Given: x1, ..., xn
Assume d.i.t.s.
Goal: Compute z* = argmax p(z|x).






μk(zk) = max p(z1:k, x1:k)
z1:k-1
f(a)
g(a,b)
= max p(xk|zk) p(zk|zk-1) p(z1:k-1, x1:k-1)
z1:k-1
= max [p(xk|zk) p(zk|zk-1) max p(z1:k-1, x1:k-1)]
zk-1 z1:k-2
known known
μk-1(zk-1)
a = zk-1
b = z1:k-2

HMM + VITERBI

$$= \max_{z_{k-1}} \left[p(x_k|z_k)\, p(z_k|z_{k-1}) \max_{z_{1:k-2}} p(z_{1:k-1}, x_{1:k-1}) \right]$$

known known

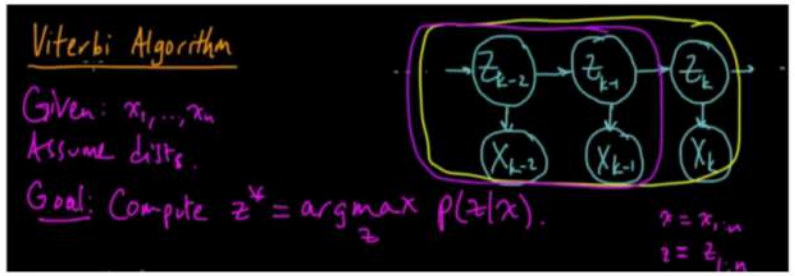
$$\mu_{k-1}(z_{k-1})$$

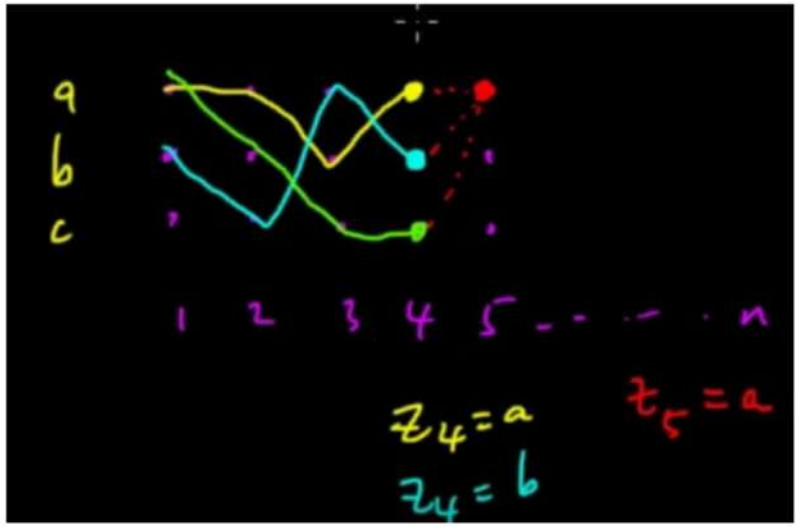
$$\mu_k(z_k) = \max_{z_{k-1}} p(x_k|z_k)\, p(z_k|z_{k-1})\, \mu_{k-1}(z_{k-1}) \quad \text{for } k = 2, \ldots, n$$

$$\mu_1(z_1) = p(z_1, x_1) = p(z_1)\, p(x_1|z_1)$$

$$\mu_n(z_n) = \max_{z_{1:n-1}} p(x_{1:n}, z_{1:n})$$

$$\max_{z_n} \mu_n(z_n) = \max_{z_{1:n}} p(x_{1:n}, z_{1:n})$$





第二课 单字符识别

- 单字符识别
  - 模版匹配
  - 特征提取 + 传统分类器
  - CNN模型
- 过切分的路径选择
  - VITERBI + BEAM SEARCH
- 语言模型
  - N-GRAM
  - LSTM
  - 2D-PCFG

- N-GRAM

当 n=1, 一个一元模型（unigram model)即为：

$$P(w_1, w_2, \cdots, w_m) = \prod_{i=1}^{m} P(w_i)$$

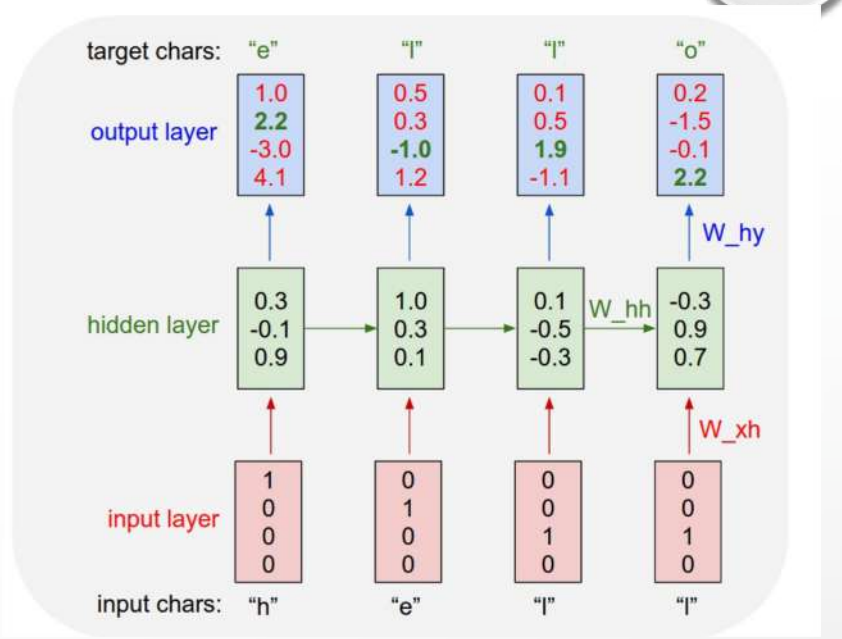
当 n=2, 一个二元模型（bigram model)即为：

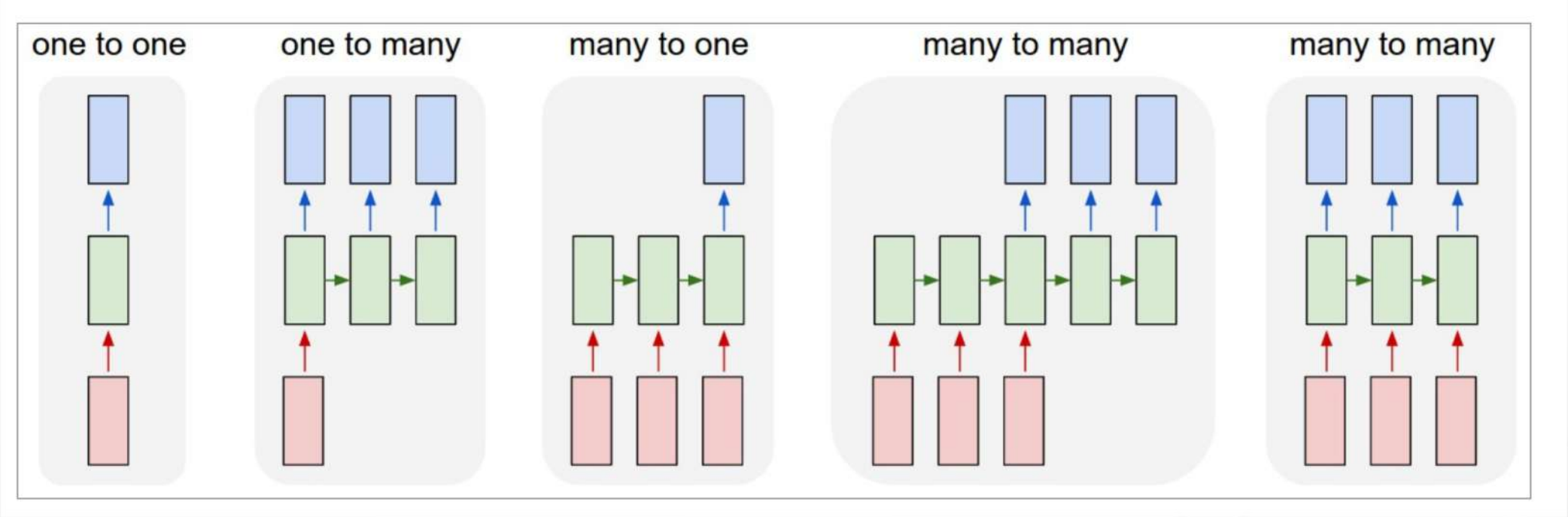
$$P(w_1, w_2, \cdots, w_m) = \prod_{i=1}^{m} P(w_i|w_{i-1})$$

当 n=3, 一个三元模型（trigram model)即为

$$P(w_1, w_2, \cdots, w_m) = \prod_{i=1}^{m} P(w_i|w_{i-2}w_{i-1})$$

- LSTM


target chars: "e" "l" "l" "o"
output layer
1.0 2.2 -3.0 4.1
0.5 0.3 -1.0 1.2
0.1 0.5 1.9 -1.1
0.2 -1.5 -0.1 2.2
W_hy
hidden layer
0.3 -0.1 0.9
1.0 0.3 0.1
0.1 -0.5 -0.3
W_hh
-0.3 0.9 0.7
W_xh
input layer
1 0 0 0
0 1 0 0
0 0 1 0
0 0 1 0
input chars: "h" "e" "l" "l"


one to one
one to many
many to one
many to many
many to many

- CFG (CONTEXT FREE GRAMMAR)

上下文无关语法 (Context-Free Grammars) ：

一个上下文无关语法是一个四元组2：

$$G = (N, \Sigma, R, S)$$

其中N是一组非终结符(non-terminal symbols)

Σ是一组终结符(terminal symbols)

R是一系列的变换规则，即是：

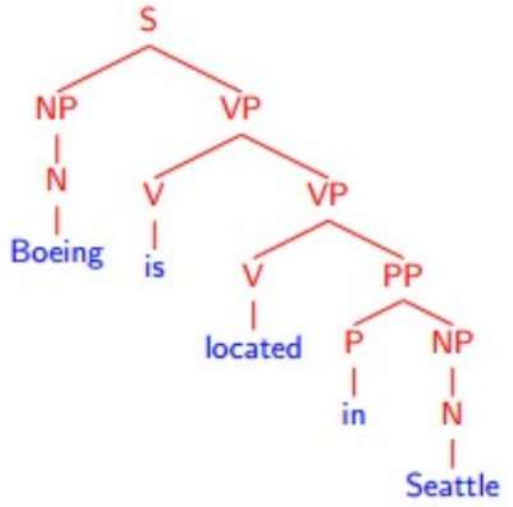
$$X \to Y_1 Y_2 \ldots Y_n \quad \text{for } n \geq 0,\ X \in N,\ Y_i \in (N \cup \Sigma)$$

S是一个特殊的开始符号

INPUT:

Boeing is located in Seattle.

OUTPUT:

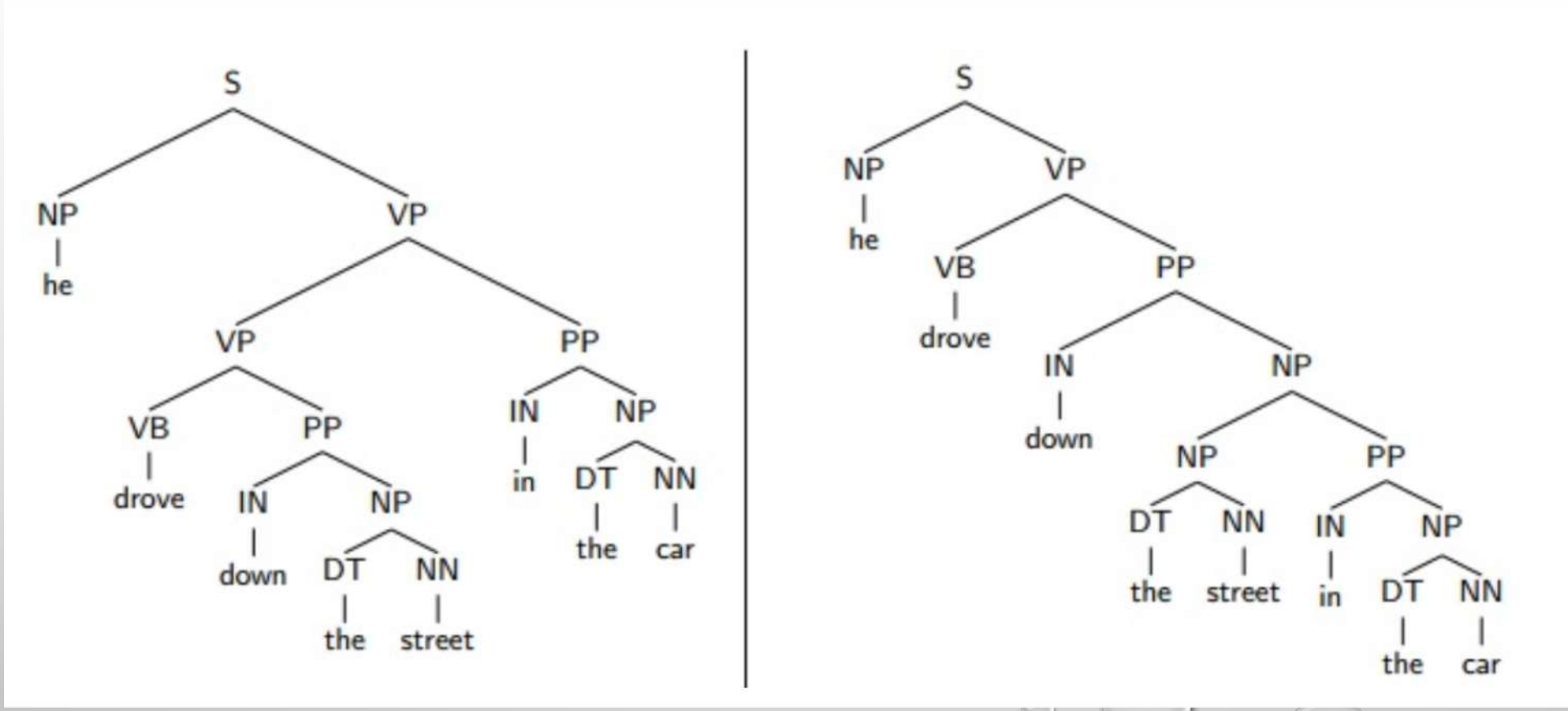


Noun Phrases (NP) Verb Phrases (VP) Sentences (S)

(N = noun, V = verb, DT = determiner)

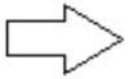
CFG (context free grammar)

Ambiguity



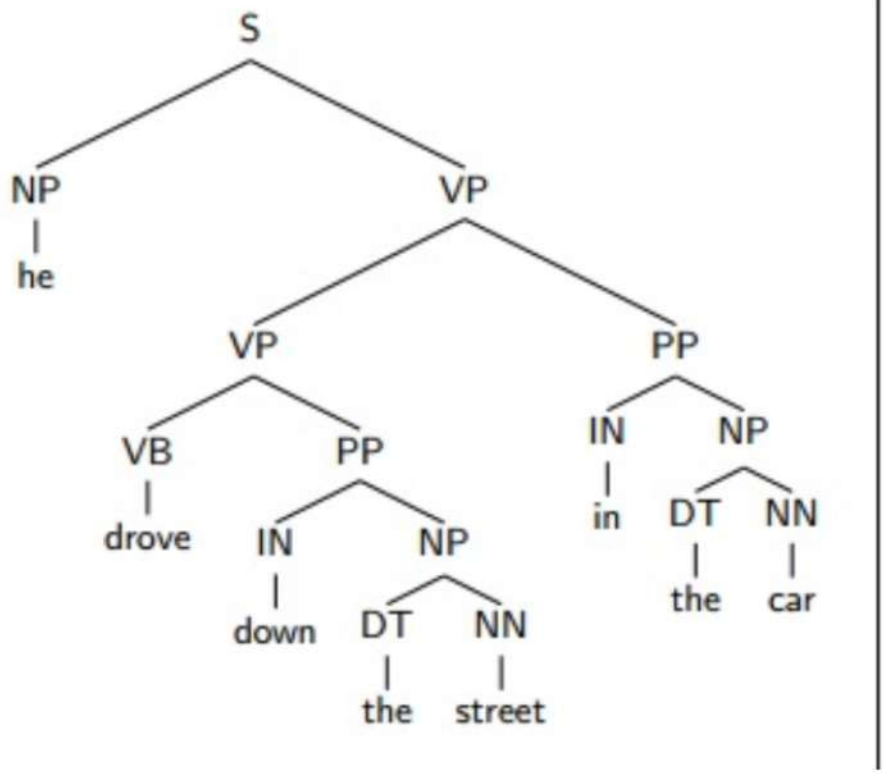
PCFG (probabilistic context free grammar)

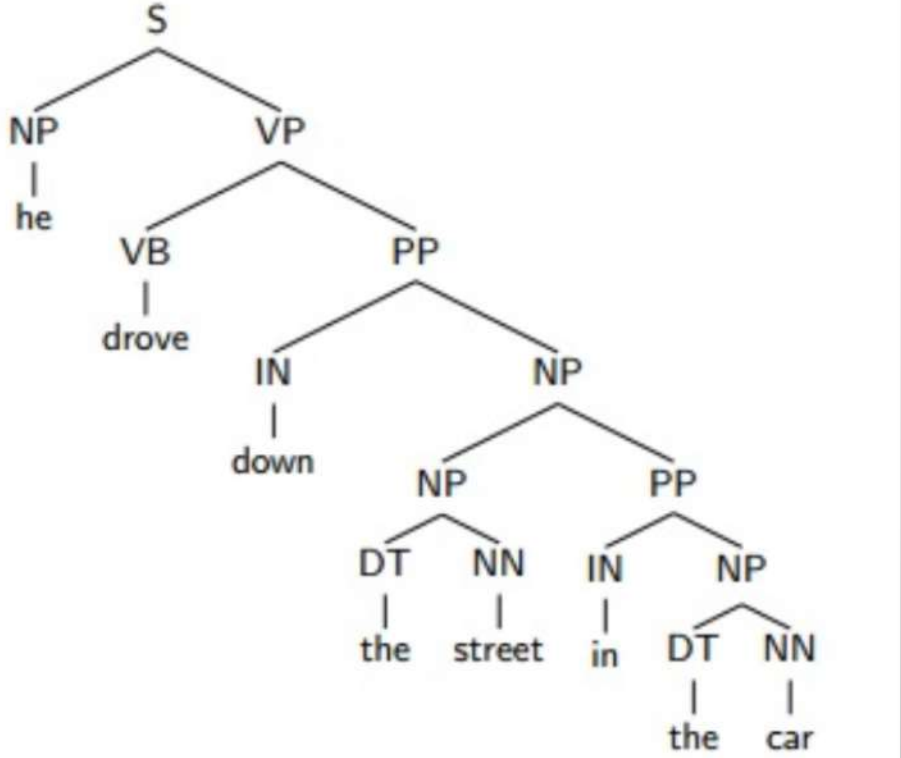
S	⇒	NP	VP	
VP	⇒	Vi		
VP	⇒	Vt	NP	
VP	⇒	VP	PP	
NP	⇒	DT	NN	
NP	⇒	NP	PP	
PP	⇒	P	NP	



S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	P	NP	1.0

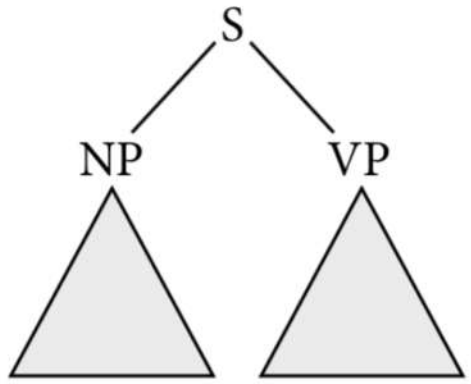
Ambiguity





- PCFG

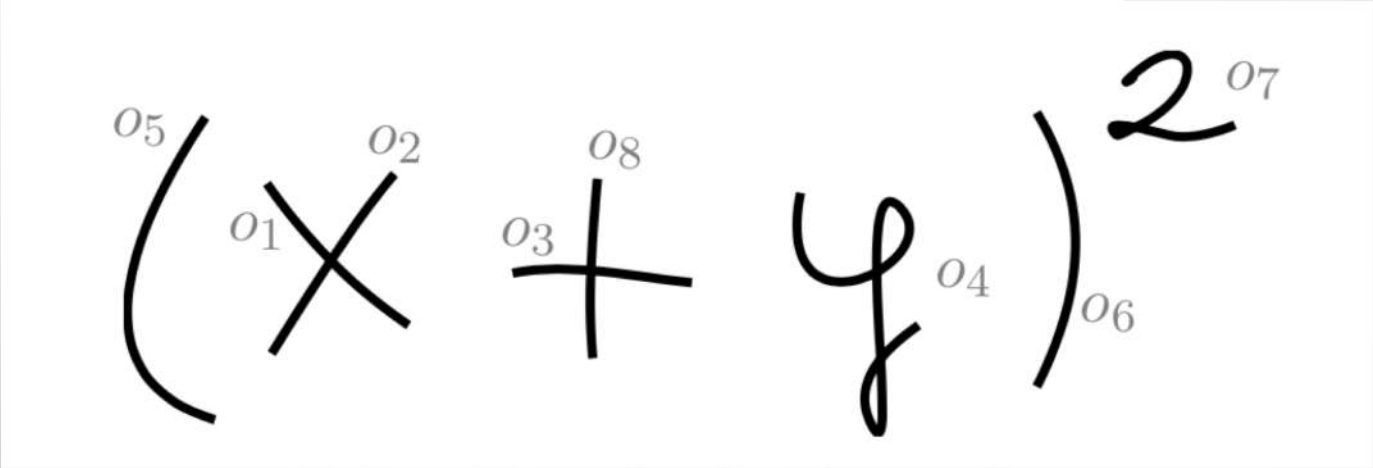
- Context-free grammar: One rule can only "see" parent and its children, not anything above or below.
- PCFG: Assumes statistical independence of all rewrite events.



NP → PRP?
NP → Det N?

$$a_0 = x^2 + \frac{\sqrt{\pi}}{3}$$

- 2D-PCFG



$$\|f\|_\alpha^2 = \frac{\Gamma(n+\alpha)}{2^n \pi^n \Gamma(\alpha)} \int_D (-\rho)^\alpha |f|^2 \, \mathrm{d}V$$

$$k^*(w') \geqq k^*(w_0) - \varepsilon\} \geqq \frac{3\pi^n \delta_1^{2n-1}}{2\sqrt{n}\, 4^{2n-1}}$$

$$\sum_{i=0}^{k-1} (\alpha_i - \beta_i)\bar{\delta}_i = (\beta_k - \alpha_k)\bar{\delta}_k. \qquad \prod_{p \in P} C(p),$$

$$\sigma(T)(\subset \sigma(A)) \qquad (\sigma = i_{k-1}, 2 \leqslant$$

$$\int e^{x^2} x^3 \, dx \qquad \int e^{x^2} x^3 \, dx \qquad \int e^{x^2} x^3 \, dx$$

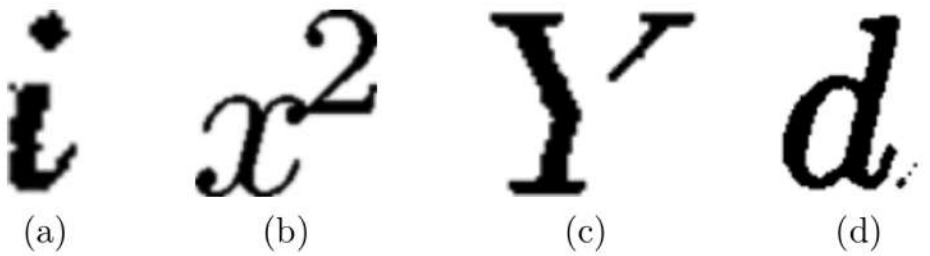
$$\sum_{n=1}^{N} (-1)^n \sin(nx) \qquad \sum_{n=1}^{N} (-1)^n \sin(nx)$$

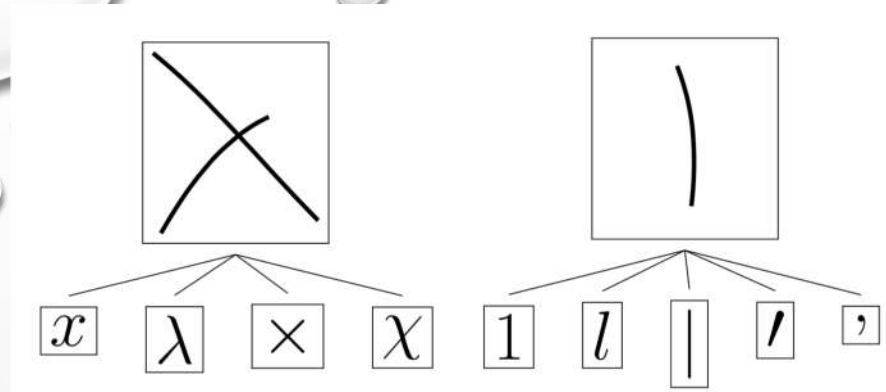
$$\frac{\sin(x)}{\cos(x)} = \tan(x)$$

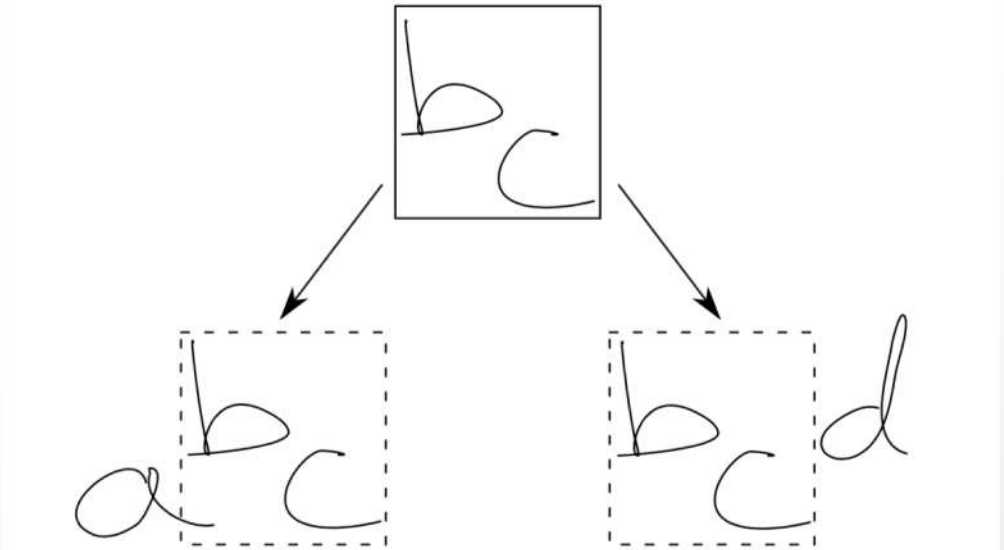
$$\sqrt{9+-j} \qquad -\left[2^{\omega\infty}\right]$$

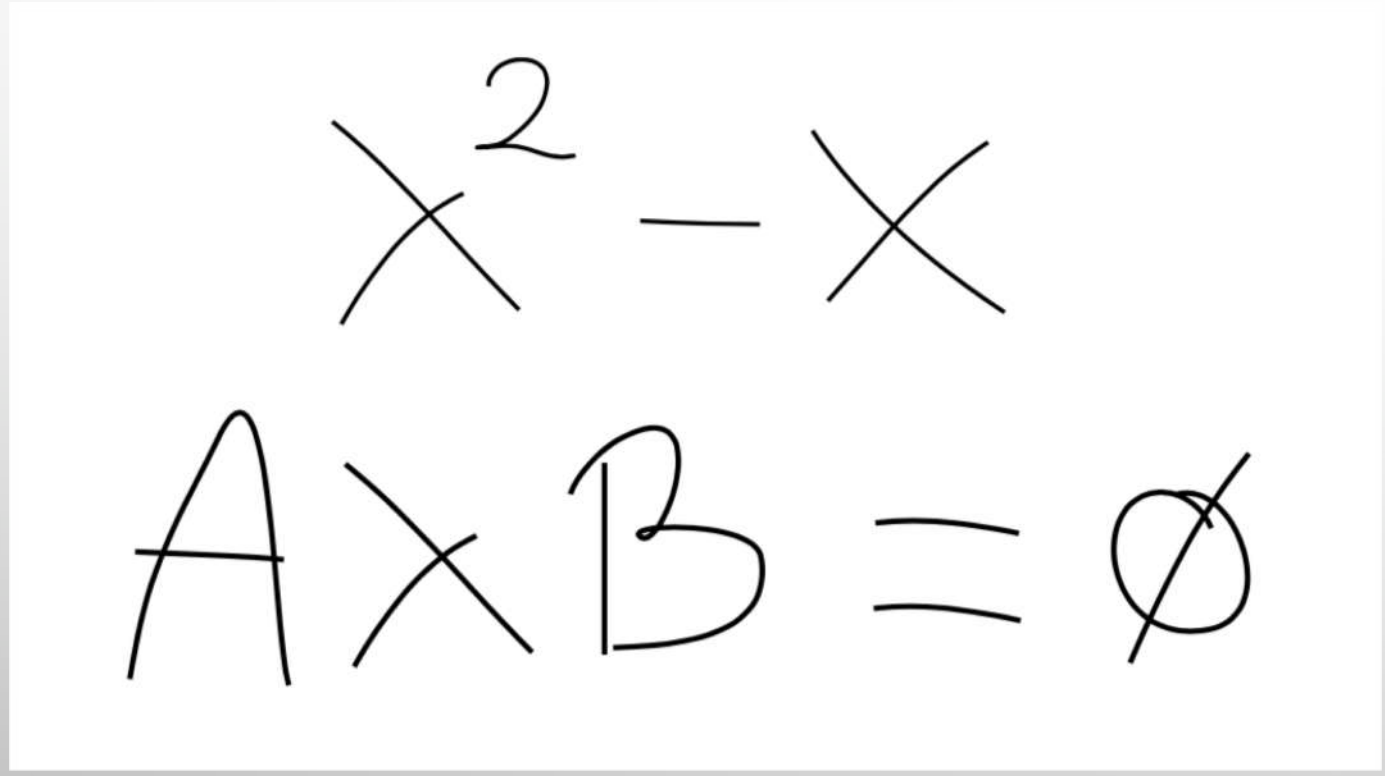
$$\sqrt{\frac{22^3}{\sqrt{\frac{32^2}{\sqrt{42}}}}}$$

(a) $\pm \frac{2}{5}$ (b) $|-|< X$









Mathematical Expression Recognition based on Probabilistic Grammars, Francisco A´lvaro Mun˜oz , 2015

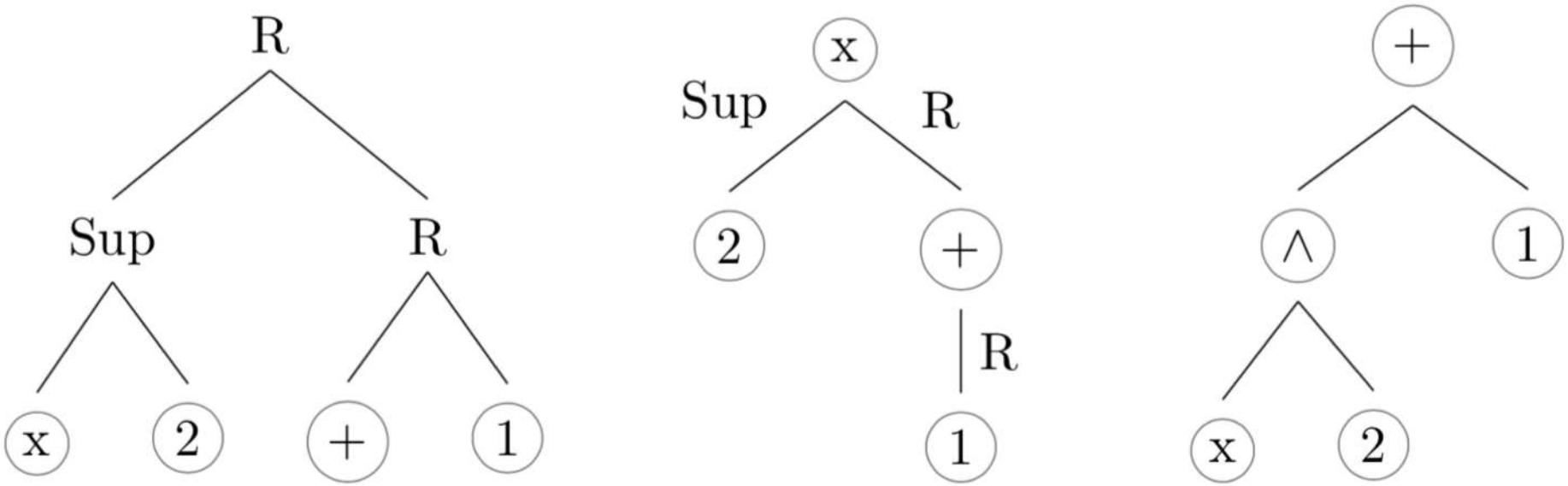


Figure 1.8: Example of tree representation of the expression $\boxed{x^2 + 1}$. Left to right: relational tree, symbol layout tree, and operator tree.

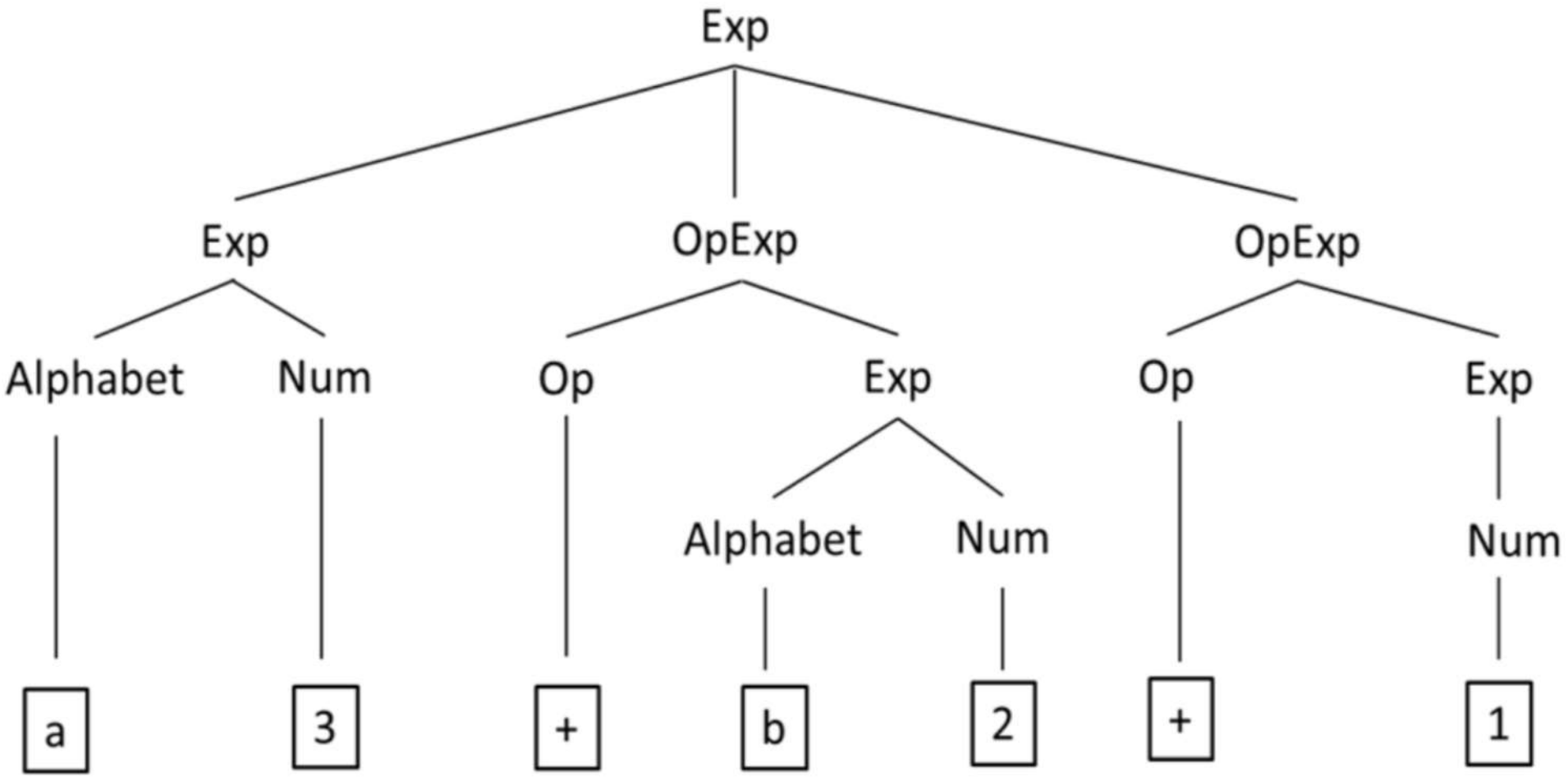
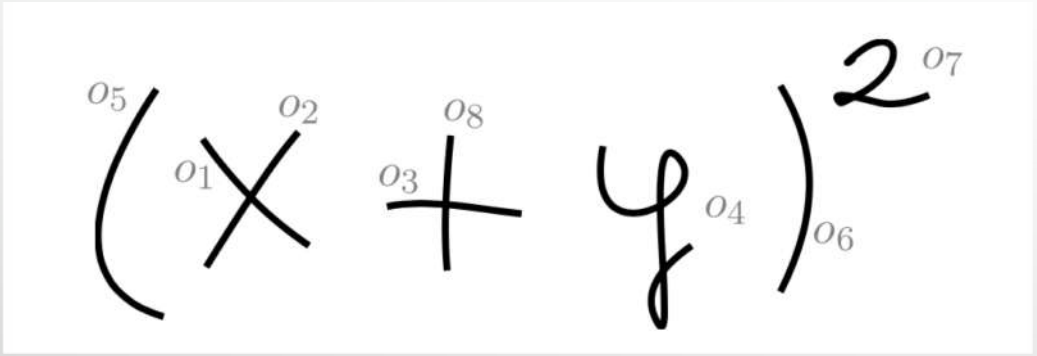
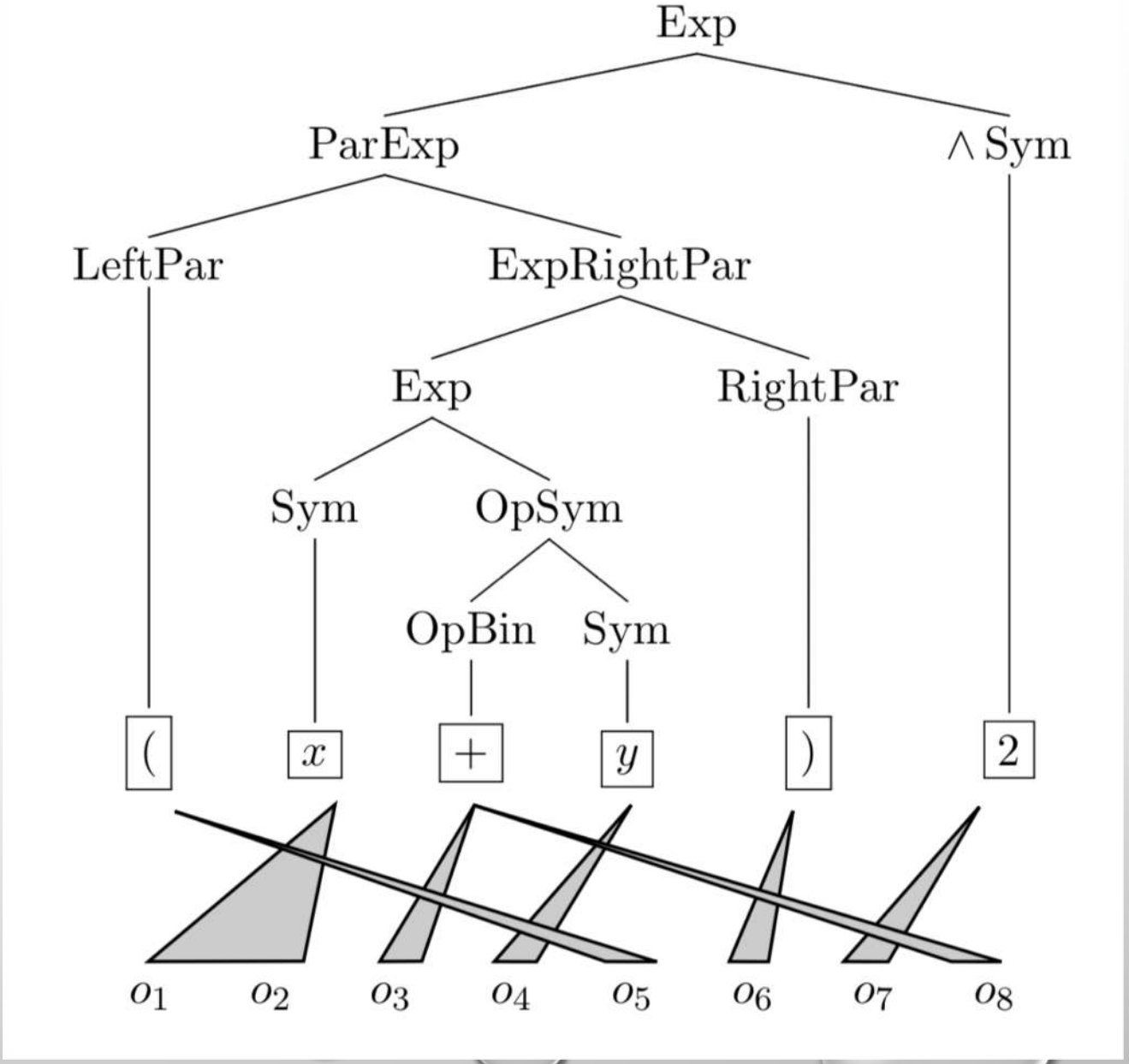


Figure 2.4 Parsing tree for $a^3 + b^2 + 1$

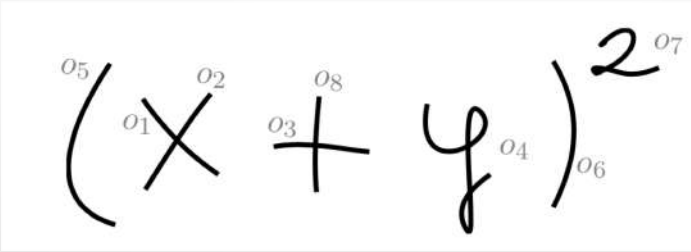
- 2D-PCFG

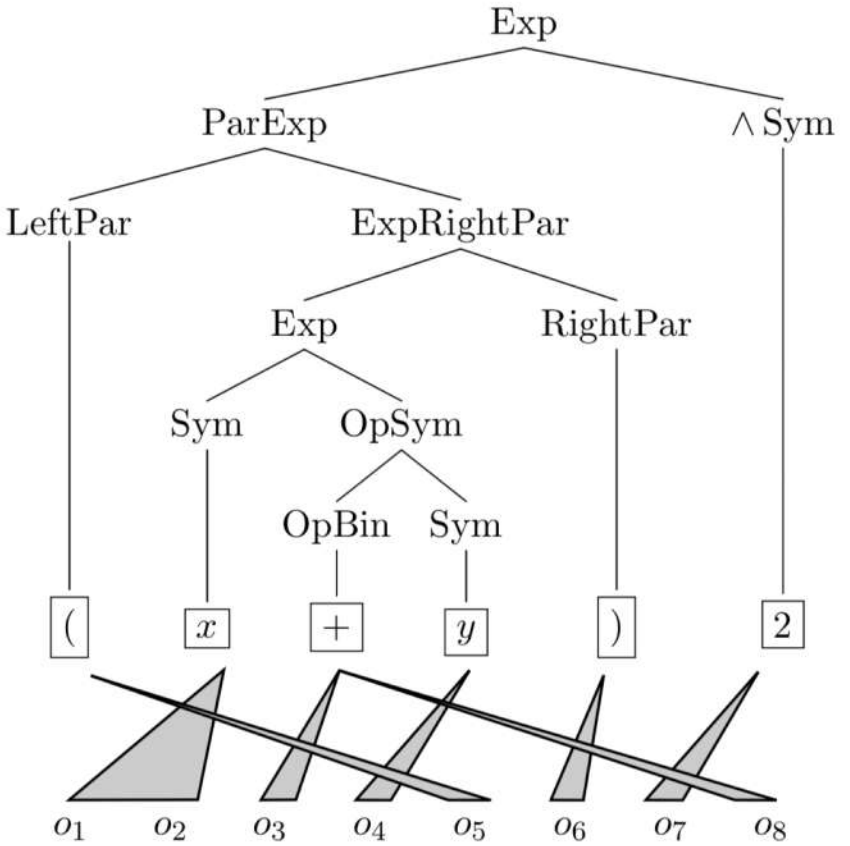

o_5
o_2
o_8
o_7
o_1
o_3
o_4
o_6


Exp
ParExp
∧ Sym
LeftPar
ExpRightPar
Exp
RightPar
Sym
OpSym
OpBin
Sym
(
x
+
y
)
2
o_1
o_2
o_3
o_4
o_5
o_6
o_7
o_8

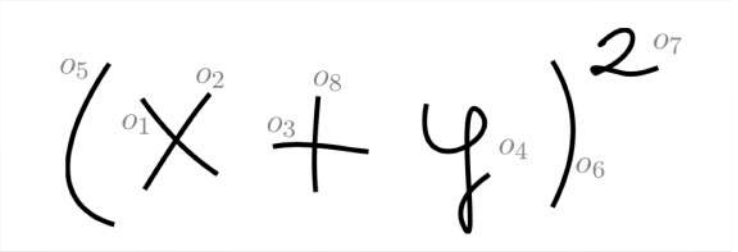
Definition 2.4.1. A *Context-Free Grammar* (CFG) G is a 4-tuple $(\mathcal{N}, \Sigma, S, \mathcal{P})$ where $\mathcal{N}$ is a finite set of nonterminal symbols, Σ is a finite set of terminal symbols ($\mathcal{N} \cap \Sigma = \emptyset$), $S \in \mathcal{N}$ is the start symbol of the grammar, and $\mathcal{P}$ is a finite set of rules: $A \to \alpha$, $A \in N$, $\alpha \in (\mathcal{N} \cup \Sigma)^+$.

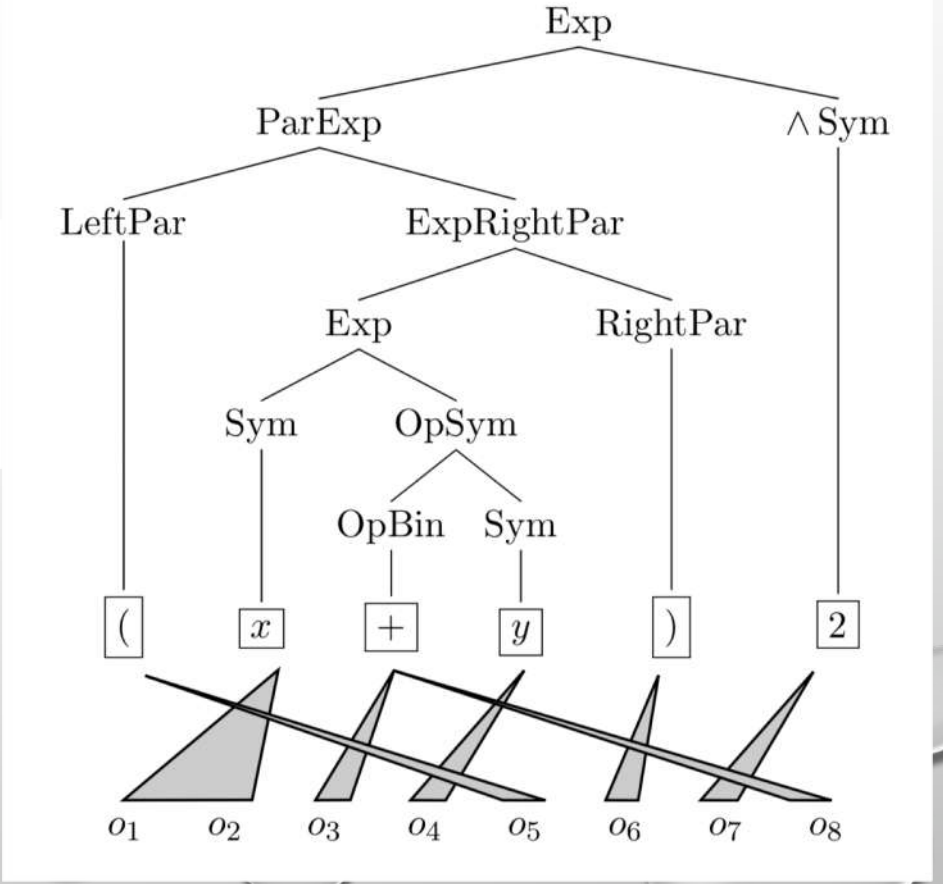
A CFG in Chomsky Normal Form (CNF) is a CFG in which the rules are of the form $A \to BC$ or $A \to a$ (where $A, B, C \in \mathcal{N}$ and $a \in \Sigma$).



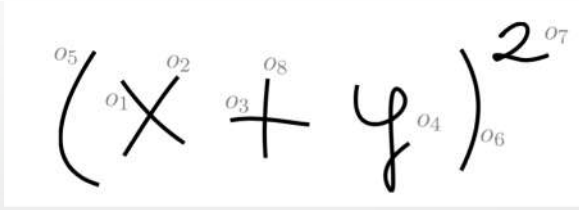


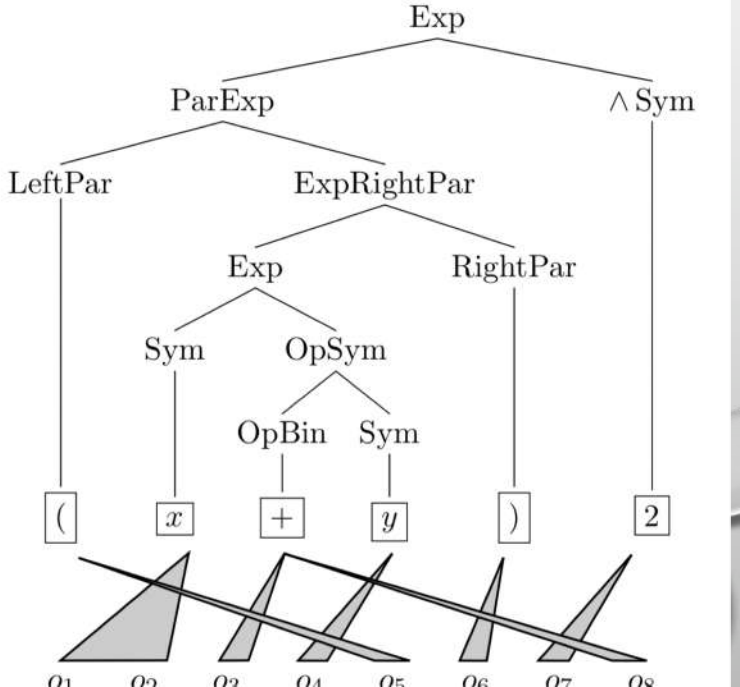
Definition 2.4.2. A *Probabilistic Context-Free Grammar* (PCFG) $\mathcal{G}$ is defined as a pair (G, p), where G is a CFG and $p : P \rightarrow]0, 1]$ is a probability function of rule application such that $\forall A \in \mathcal{N} : \sum_{i=1}^{n_A} p(A \rightarrow \alpha_i) = 1$, where n_A is the number of rules associated with nonterminal symbol A.





Definition 2.4.3. A *Two-Dimensional Probabilistic Context-Free Grammar* (2D-PCFG) is a generalization of a PCFG, where terminal and nonterminal symbols describe two-dimensional regions. This grammar in Chomsky Normal Form results in two types of rules: terminal rules and binary rules. First, the terminal rules $A \to a$ represent the mathematical symbols which are ultimately the terminal symbols of 2D-PCFG. Second, the binary rules $A \xrightarrow{r} BC$ have an additional parameter r that represents a given spatial relationship, and its interpretation is that regions B and C must be spatially arranged according to the spatial relationship r.





```
#Non-terminals
Exp
Sym
ExpOp
OpUn
ROpUn
OBExp
OpBin
OverExp
Over
OverSym
LeftPar
RightPar
RPExp
SSExp
BigOpExp
BigOp
Sqrt
Func
2Let
Let
SupSym
Num
Point
Dec
```

```
#Initial symbol of the grammar
START
Exp
Sym
```

```
#Terminal productions
PTERM
#S    ->    [t1, t2, ... tn]
Sym       symbols.term
Let       letters.term
Num       numbers.term
Point     point.term
Over      over.term
BigOp     bigop.term
OpUn      opun.term
ROpUn     ropun.term
OpBin     opbin.term
OverSym   oversym.term
SupSym    supsyms.term
LeftPar   leftpar.term
RightPar  rightpar.term
Sqrt      sqrt.term
```

```
#Binary productions
PBIN
#pr   t    S     ->    A         B         TeX-out
1.0   Sup  Exp         ExpOp     SupSym    "{$1}^{$2}"
1.0   Sup  Exp         Sym       SupSym    "{$1}^{$2}"
1.0   Sub  Exp         BigOp     Exp       "$1_{$2}"
1.0   Sub  Exp         BigOp     Sym       "$1_{$2}"
1.0   H    Exp         Exp       Exp       "$1 $2"
1.0   H    Exp         Exp       Sym       "$1 $2"
1.0   H    Exp         Sym       Exp       "$1 $2"
1.0   H    Exp         Sym       Sym       "$1 $2"
1.0   H    Exp         OpUn      Exp       "$1 $2"
1.0   H    Exp         OpUn      Sym       "$1 $2"
1.0   H    Exp         Exp       ROpUn     "$1 $2"
1.0   H    Exp         Sym       ROpUn     "$1 $2"
1.0   H    Exp         Exp       OBExp     "$1 $2"
1.0   H    Exp         Sym       OBExp     "$1 $2"
1.0   H    OBExp       OpBin     Exp       "$1 $2"
1.0   H    OBExp       OpBin     Sym       "$1 $2"
1.0   Sup  Exp         ExpOp     Exp       "{$1}^{$2}"
1.0   Sup  Exp         ExpOp     Sym       "{$1}^{$2}"
1.0   Sup  Exp         Sym       Exp       "{$1}^{$2}"
1.0   Sup  Exp         Sym       Sym       "{$1}^{$2}"
1.0   Sub  Exp         ExpOp     Exp       "{$1}_{$2}"
1.0   Sub  Exp         Sym       Exp       "{$1}_{$2}"
1.0   Sub  Exp         ExpOp     Sym       "{$1}_{$2}"
1.0   Sub  Exp         Sym       Sym       "{$1}_{$2}"
1.0   V    Exp         Exp       OverExp   "\frac{$1}{$2}"
1.0   V    Exp         Sym       OverExp   "\frac{$1}{$2}"
1.0   V    OverExp     Over      Exp       "$2"
1.0   V    OverExp     Over      Sym       "$2"
1.0   Vs   Exp         OverSym   Exp       "$1{$2}"
1.0   Vs   ExpOp       OverSym   Exp       "$1{$2}"
1.0   Vs   Exp         OverSym   Sym       "$1{$2}"
1.0   Vs   ExpOp       OverSym   Sym       "$1{$2}"
1.0   Vs   Exp         OverSym   Exp       "$1{$2}"
1.0   Vs   Exp         OverSym   Sym       "$1{$2}"
```

$$\text{Fraction} \xrightarrow{below} \text{Expression}\ \ \text{OverExp}$$
$$\text{OverExp} \xrightarrow{below} \text{HorzLine}\ \ \text{Expression}$$

Demo: http://cat.prhlt.upv.es/mer/

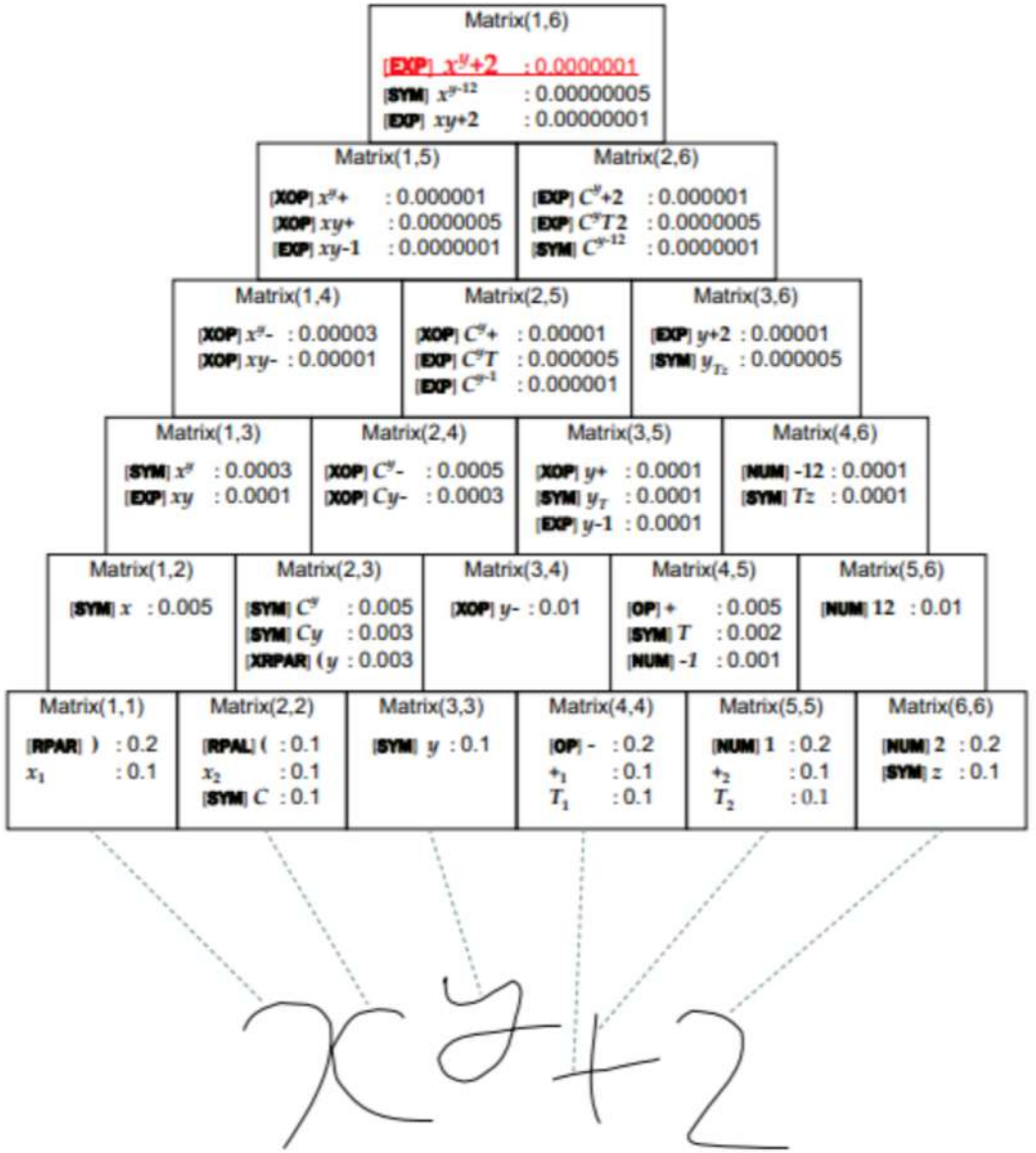
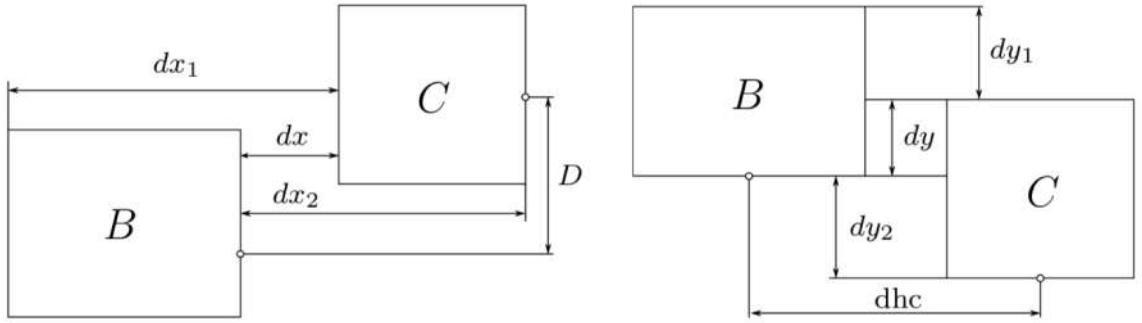
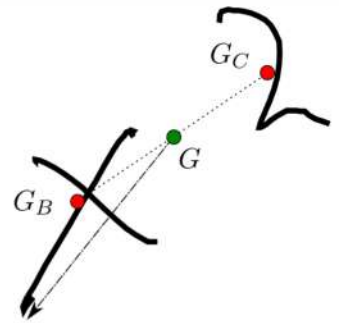


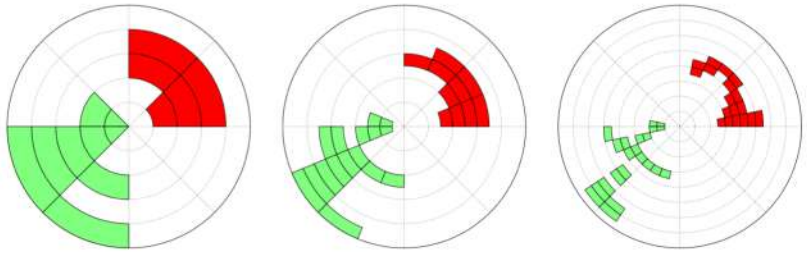
Figure 2.5 Example of a search for most likely expression candidate using the CYK algorithm



$$\mathrm{h}(B, C) = [H, D, \mathrm{dhc}, dx, dx_1, dx_2, dy, dy_1, dy_2]$$

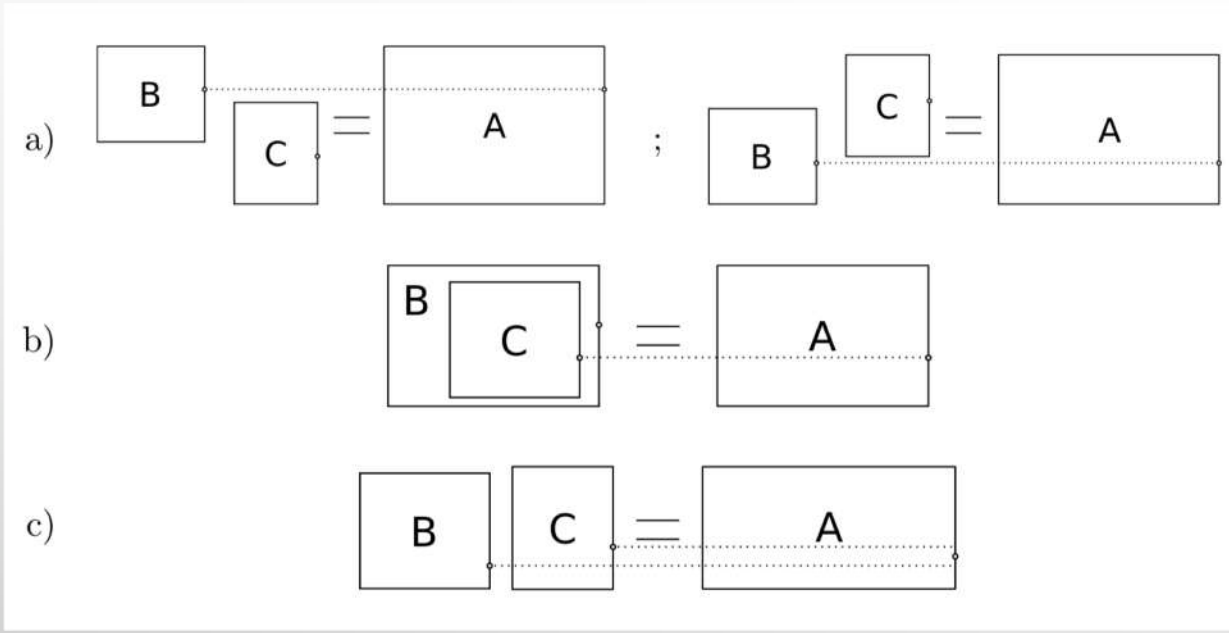


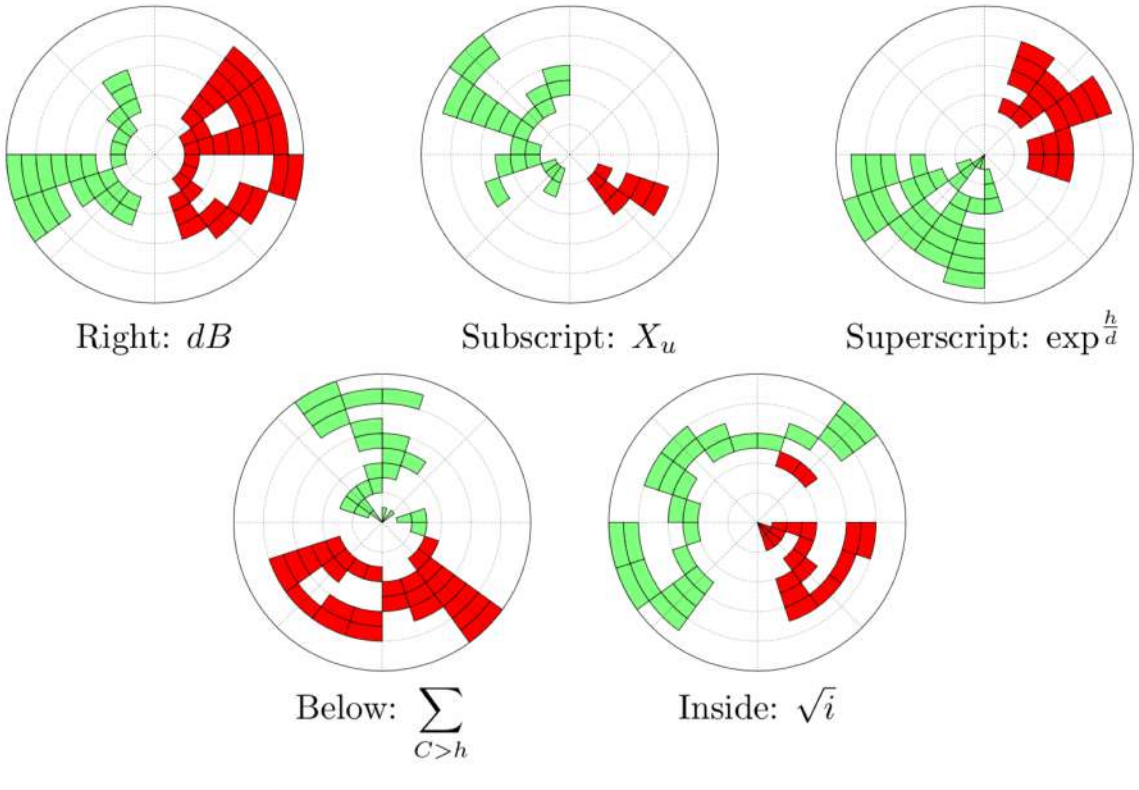
Symbol pair (centers for x, 2 and midpoint shown).



$5 \times 8 = 40$ bins $\quad 10 \times 16 = 160$ bins $\quad 15 \times 32 = 480$ bins

Figure 4.4: Varying distance (n) $\times$ angle (m) resolution in a polar histogram layout descriptor. Values shown using green (-1), red (+1), and white (0).


a)
B
C
=
A
;
B
C
=
A
b)
B
C
=
A
c)
B
C
=
A


Right: dB
Subscript: X_u
Superscript: $\exp^{\frac{h}{d}}$
Below: $\sum_{C>h}$
Inside: $\sqrt{i}$

下周预告

第三课 字符序列识别

知识点1： 概览RNN以及LSTM + CTC、CRNN、RARE

知识点2： 概览注意力机制Attention，以及DRAM / DRAW、Transformer

实战项目： 实战CNN和RNN的综合体CRNN

THANKS