Course 4 : Convolutional Neural Network

Content :

Week 1 : Foundations of CNNs

Week 2 : Deep convolutional models : case studies

Week 3 : Object detection

Week 4 : Special applications : Face recognition &
Neural style transfer

Coursera
Deep Learning Specialization

Vertical detector

1	0	-1
1	0	-1
1	0	-1

Horizontal

1	1	1
0	0	0
-1	-1	-1

Sobel filter

→ Taking care of middle row

1	0	-1
2	0	-2
1	0	-1

Scharr filter

3	0	-3
10	0	-10
3	0	-3

Padding

imagine size: $n \times n$; Convolution size: $f \times f$

Padding size: p

Output size after Convolution:

Without Padding: $(n-f+1) * (n-f+1)$

Padding: $(n+2p-f+1) * (n+2p-f+1)$

Convention: $p = (f-1)/2$

valid Convolution: no Padding

Same Convolution: output size = input size

f usually odd

Strided Convolution

stride : s

matrix $n*n$, $f*f$ filter, padding p, stride s

output size after convolution:

$$\lfloor (n+2-f)/s+1 \rfloor * \lfloor (n+2p-f)/s+1 \rfloor$$

1 layer of a Convolutional Network

filter $f^{[l]}$, Padding $p^{[l]}$, stride $s^{[l]}$

filters $n_c^{[l]}$

filter shape $f^{[l]} * f^{[l]} * n_c^{[l-1]}$

input shape $n_H^{[l-1]} * n_w^{[l-1]} * n_c^{[l-1]}$

output shape $n_H^{[l]} * n_w^{[l]} * n_c^{[l]}$

.. height $n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$

.. weight n_w

activation a[l] $n_H^{[l]} * n_w^{[l]} * n_c^{[l]}$

.. A[l] $m * n_H^{[l]} * n_w^{[l]} * n_c^{[l]}$

weights $f^{[l]} * f^{[l]} * n_c^{[l-1]} * n_c^{[l]}$

bias $(1, 1, 1, n_c^{[l]})$

CNN

$m \rightarrow$ CONV 1 $\rightarrow$ Pool 1 $\rightarrow$ Conv2 $\rightarrow$ Pool 2 $\rightarrow m \rightarrow$ FC3 $\rightarrow$ FC4

layer 1 layer 2

	Act. shape	Act. Size	# Par.
Input	(32, 32, 3)	3072	0
CONV1 (f=5, S=1)	(28, 28, 8)	6272	(5*5*3+1)*8
Pool 1	(14, 14, 8)	1568	0
Conv2 (f=5, S=1)	(10, 10, 16)	1600	(5*5*8+1)*16
Pool2	(5, 5, 16)	400	0
FC3	(120, 1)	120	400*120+120
FC4	(84, 1)	84	120*84+84
Softmax	(10, 1)	10	84*10+10

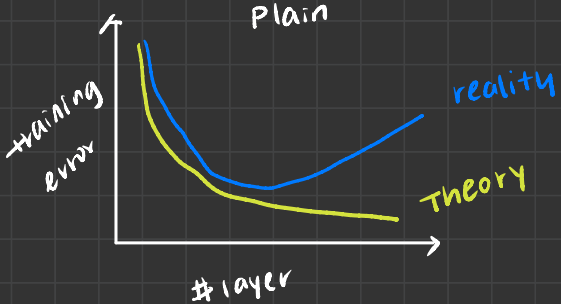
classic network } LeNet - 5
AlexNet
VGG

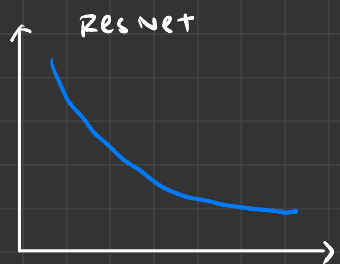
Lecture for details

ResNet

Inception

Plain





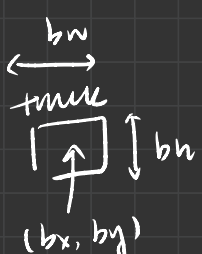
Performance ↑ as network goes deeper

Detection algorithm

Image classification

1 object vs. multi object

Classification with localization



1. Pedestrian, 2 car, 3 moto, 4. background

$y = (P_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3)^T$

Bounding Box Predictions (YOLO)

Intersection over union (IoU)

= Size of inter. area / Size of union area

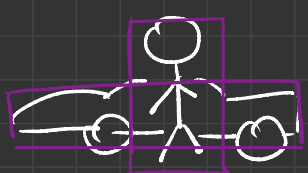
if IoU $\geq$ 0.5, Correct.

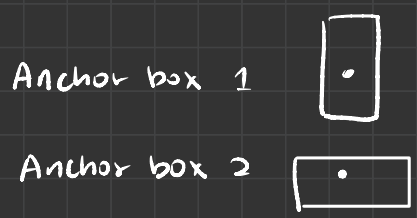
Non max Suppression:

Each output prediction is $(P_c \ b_x \ b_y \ b_h \ b_w)^T$

Discard all boxes with $P_c \leq 0.6$

Anchor Boxes





$y = (P_c \ b_x \ b_y \ b_h \ b_w \ C_1 \ C_2 \ C_3 \ P_c \ b_x \ b_y \ b_h \ b_w \ C_1 \ C_2 \ C_3)^T$

$(1 \ b_x \ b_y \ b_h \ b_w \ 1 \ 0 \ 0 \ 1 \ b_x \ b_y \ b_h \ b_w \ 0 \ 1 \ 0)^T$

Car only? $(0 \ ? \ ? \ ? \ ? \ ? \ ? \ ? \ 1 \ b_x \ b_y \ b_h \ b_w \ 0 \ 1 \ 0)^T$

R-CNN Propose regions. Classify proposed regions one at a time. Run "segmentation algorithm"

↓

slow

Fast R-CNN Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.

↓

Clustering step slow

Faster R-CNN Use convolutional network to propose regions

↓

still a bit slower than YOLO.

Triplet loss

↓

dis. b/t Anchor imagine & a positive or negative image

different person ↑

↓

same person