Basic Knowledge About Computer Vision— Based On Chapter13 of *Dive into Deep Learning*

Speaker: Qiqi Gong

Personal Introduction

- Name: Gong Qiqi
- Hometown: Xuzhou, Jiangsu Province
- Education Experience:
  - 2017-   Beijing Jiaotong University, Weihai Campus   Communication Engineering

CONTENTS

01 Image Augmentation

Image Augmentation

- Produce similar but different traing examples
- Expand traning dataset
- Imporve the capability for generalization (reduce the dependence on a certain property, e.g. brightness, color and etc.)

Image Augmentation—some operations

- Flipping and cropping
  - pytorch syntax (Flipping):
    - torchvision.transforms.RandomHorizontalFlip(*p=0.5*) # randomly L-R flip
    - torchvision.transforms.RandomVerticalFlip(*p=0.5*) # randomly U-D flip

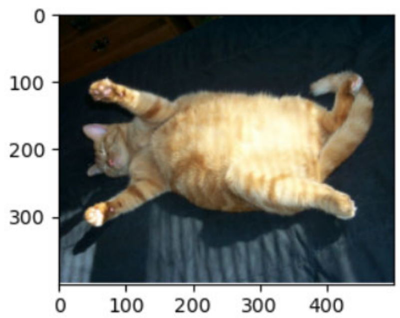


Image Augmentation—some operations

- Flipping and cropping
  - pytorch syntax (Cropping):
    - torchvision.transforms.RandomResizedCrop(size, scale=(0.08, 1), ratio=(0.75, 1.3333333333333333))
      - **size** : output size
      - **scale**: random scale of origin input from *low-boud* to *up-bound*
      - **ratio** : *w/h* of cropping region

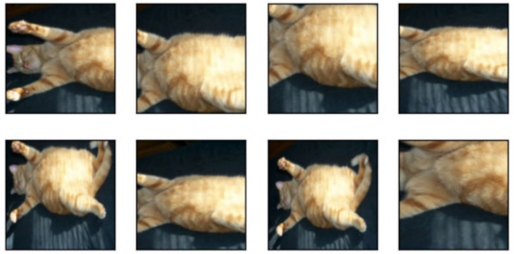
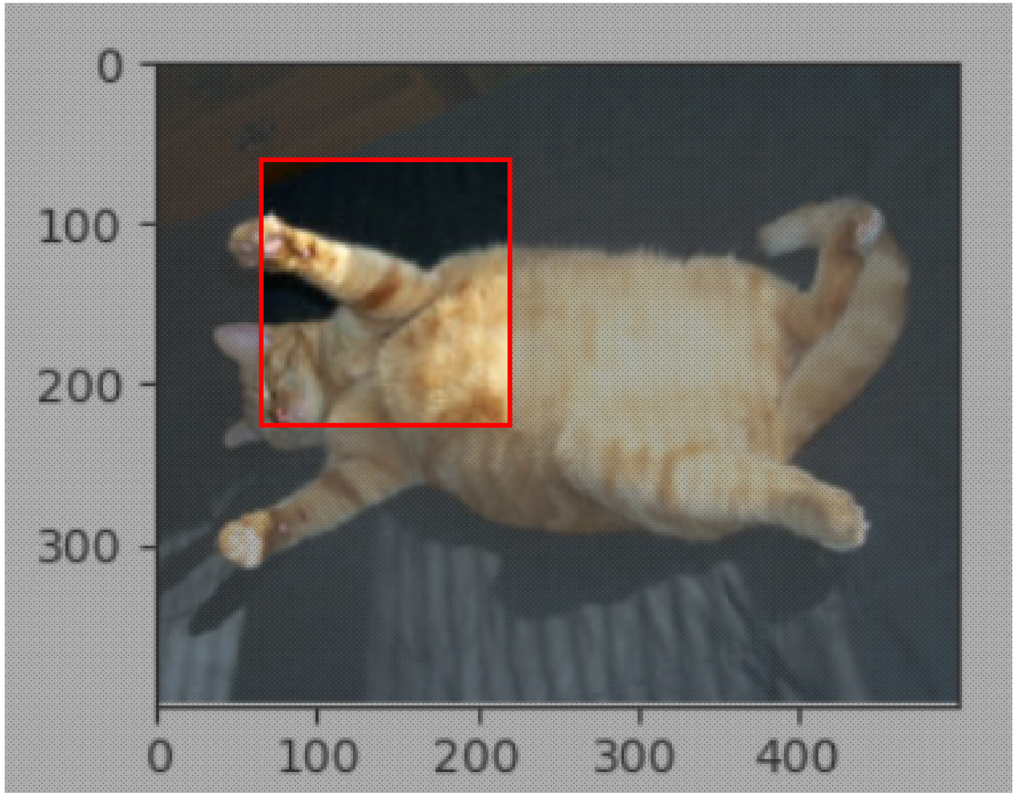


Image Augmentation—some operations



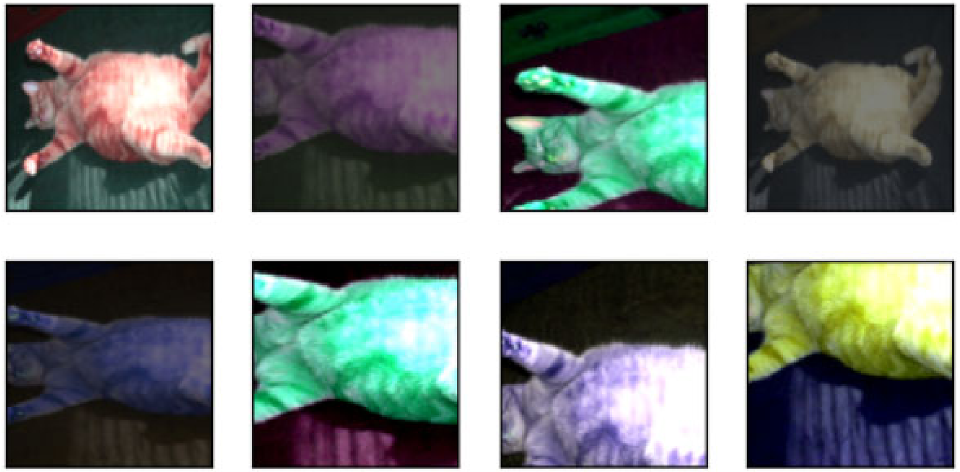
- Assume the area of cropping window is ***s***, then ***s= S*scale***
- *w=sqrt(sr) , h=sqrt(s/r)*

Image Augmentation—some operations

- Color change
  - torchvision.transforms.ColorJitter(brightness=0, contrast=0, saturation=0, hue=0)
    - parameters vary between [max(0 , 1 - ParaVal), 1 + ParaVal]
    - 亮度、对比度、饱和度、色调

Image Augmentation—some operations

- Composing change
  - torchvision.transforms.Compose([*transform list*])
    - compose kinds of transforms



02 Fine tuning

Fine-tuning - A kind of *transfer learning* tech.

- Pre-train a neural network model, i.e., the source model, on a source dataset;
- Create a new neural network model, i.e., the target model;
- Randomly initialize parameters for output layer;
- Train on target dataset and fine-tune para. for each layer

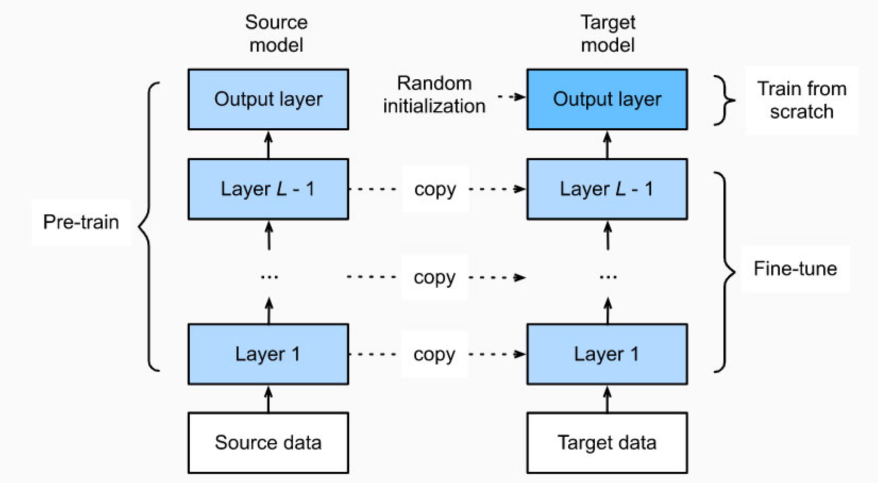
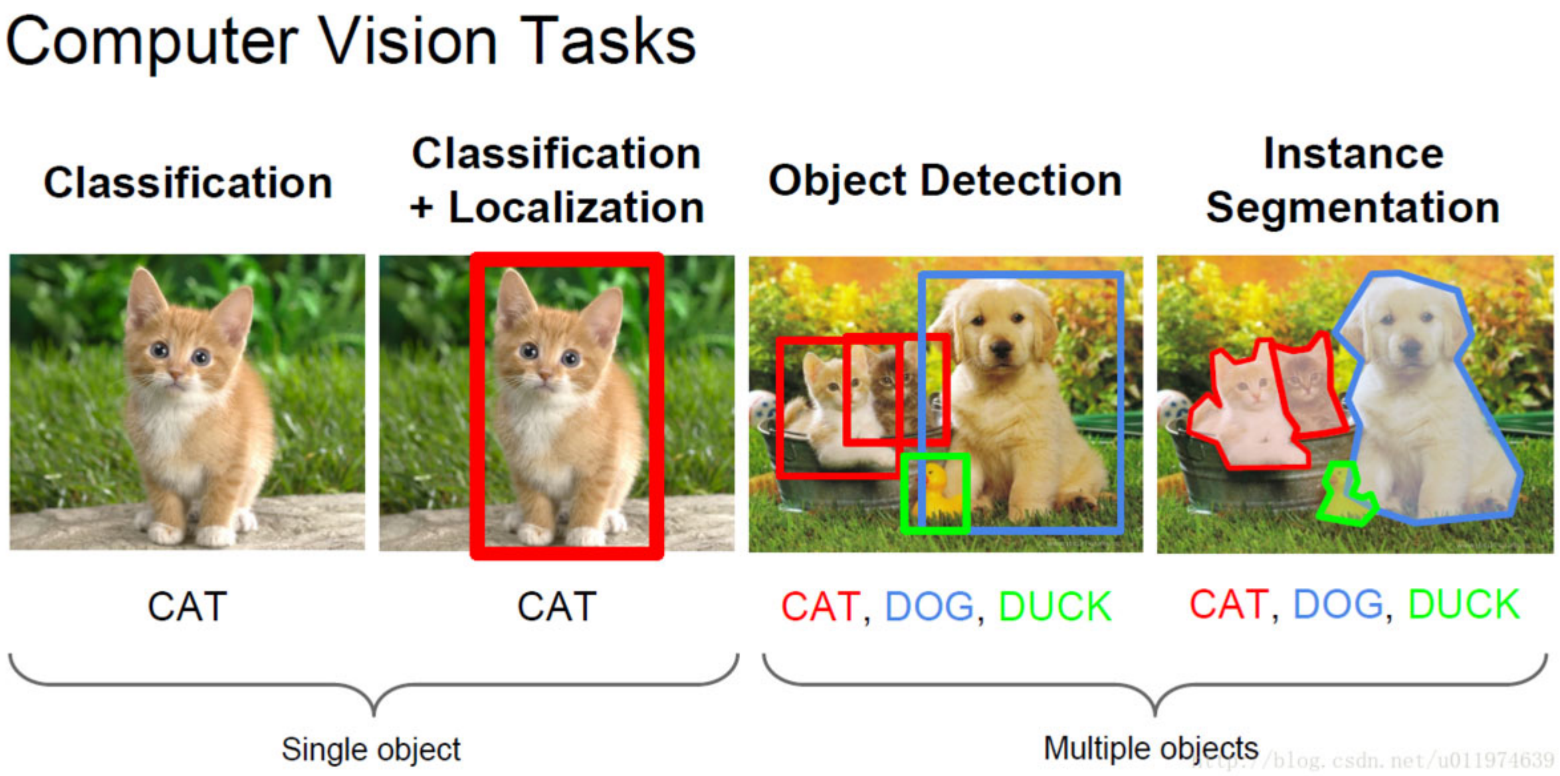


Fig. 13.2.1 Fine tuning.

03 Object Detection

Object Detection - What is Object Detection?

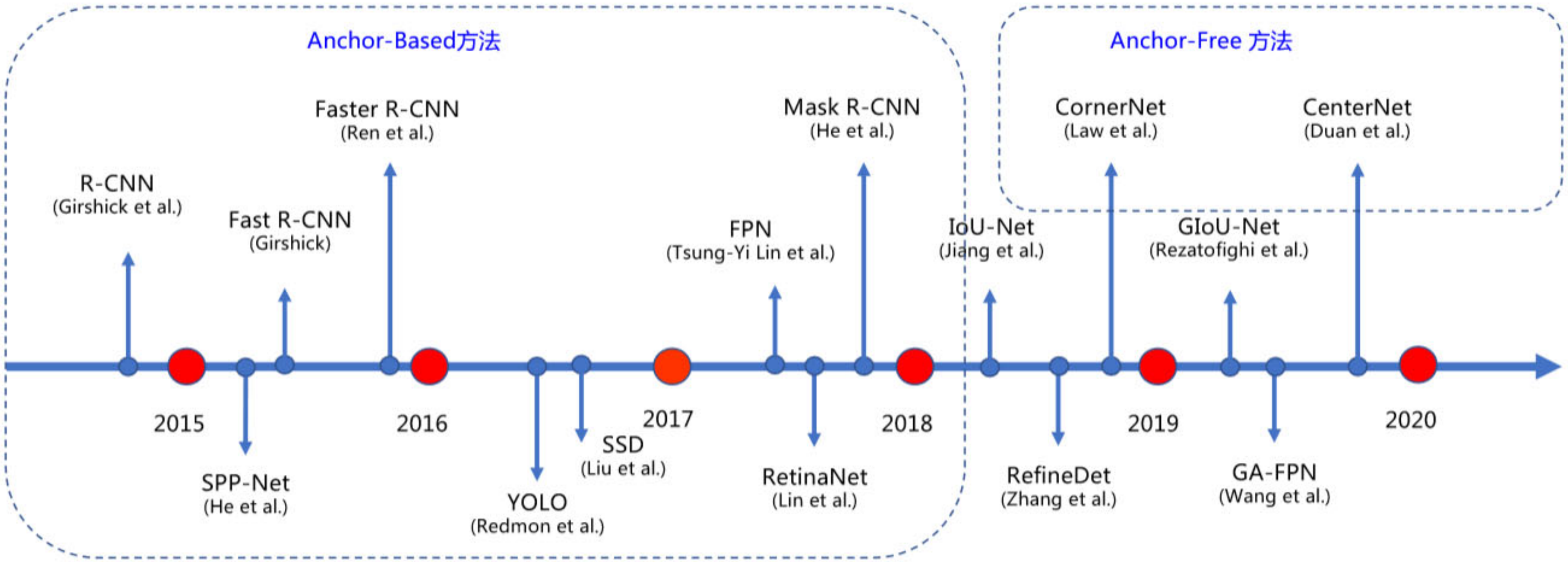


Object Detection - Application

Object Detection - Challenge

- Environment
  - Illumination
  - Fuzzy
- Crowded（密集）
- Occluded（遮挡）
- Overlapped（重叠）
- Multi-scale
  - Extremely small
  - Large
- Rare Samples (e.g. fire detection) Few shot
- Actual Use(device size, power, computation)

Object Detection - History



- Anchor-Based方法
  - 两阶段　R-CNN系列
  - 单阶段　YOLO系列、SSD、RetinaNet
- Anchor-Free方法

Object Detection - Basic Knowledge

- Bounding Box
  - Describe the position of an object
  - Predicting box, anchor box, RoI, ground-truth...
  - Describe with LU coordinate and RL coordinate

- Anchor Box
  - Multiple bounding boxes with different sizes and aspect ratios(宽高比) while centering on each pixels — produced by design
  - Called **prior box** (先验框) in SSD (Single Shot Multiple Detection, 单发多框检测)

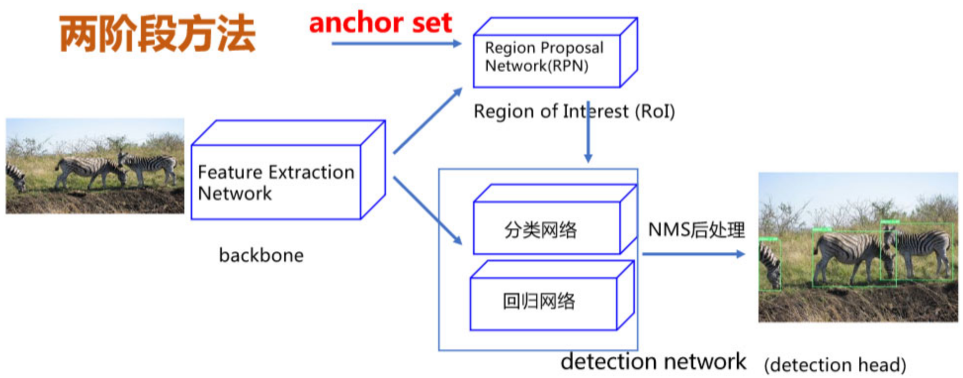
Object Detection - Basic Knowledge

- IoU - Intersection over Union
    - Assess the quality of a predicting box, the greater, the better
- RoI - Region of Interest
- Region Proposal
- RPN

◆ 两阶段方法：

✓ 先使用anchor回归候选目标框，划分前景和背景

✓ 使用候选目标框进一步回归和分类，输出最终目标框和对应的类别



Object Detection - Basic Knowledge

- NMS - Non-Maximum Suppression, 非极大值抑制
  - “去同存异”
  - Remove similar boxes and remain different boxes

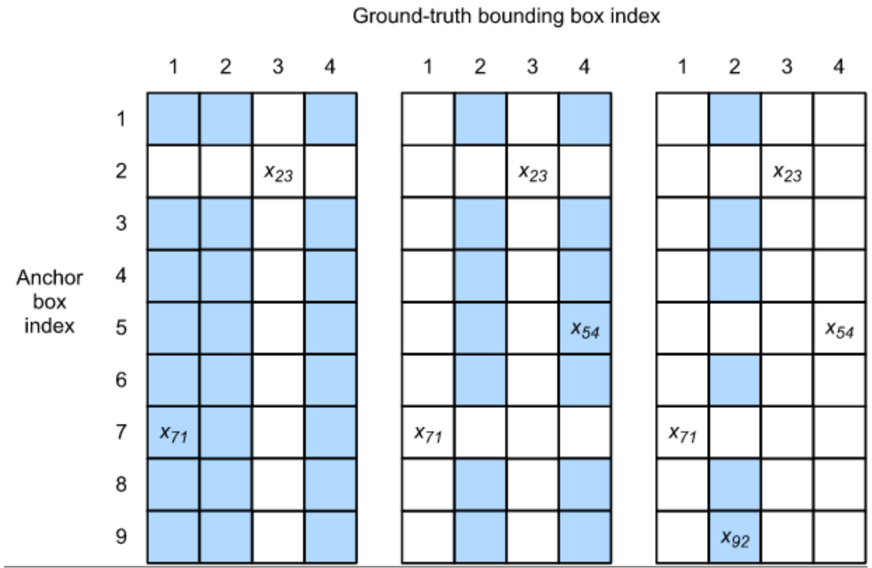
Object Detection - Basic Knowledge

- Synthesis
  - 1. Generate anchor boxes
    - Assume the size of input image is $h*w$, areas of anchor boxes are ($s_1, s_2, s_3, \ldots s_n$) and $s_i \in (0,1]$, aspect ratios are ($r_1, r_2, r_3, \ldots r_m$) and $r_i > 0$
    - Thus, the width and height of the anchor box are $ws\sqrt{r}$ and $hs/\sqrt{r}$, respectively
    - This will make computation complexed ($hwmn$ *anchor boxes*) although ground-truch bounding boxes will be included. So, we only focuse on a combination containing s1 or r1 sizes and aspect ratios. I.e. $(s_1, r_1), (s_1, r_2), \ldots, (s_1, r_m), (s_2, r_1), (s_3, r_1), \ldots, (s_n, r_1)$.

$w\sqrt{sr}$ $h\sqrt{s/r}$

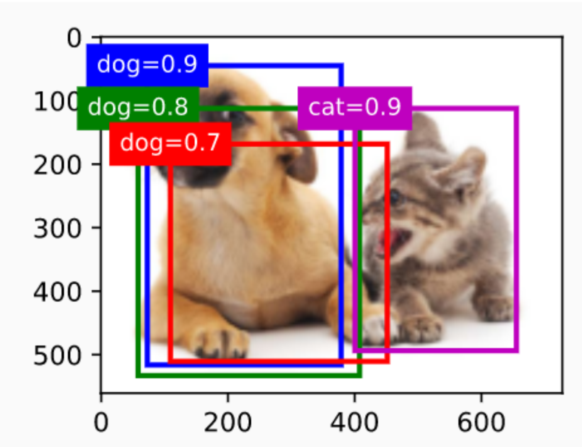
Object Detection - Basic Knowledge

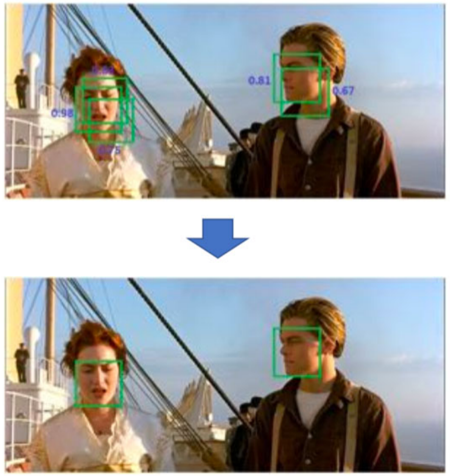
- Synthesis
  - 2. Labeling traing set anchor boxes
    - Labeling two labels for each a-box: i) category; ii) Relative offset of GT box to a-box (offset)



Object Detection - Basic Knowledge

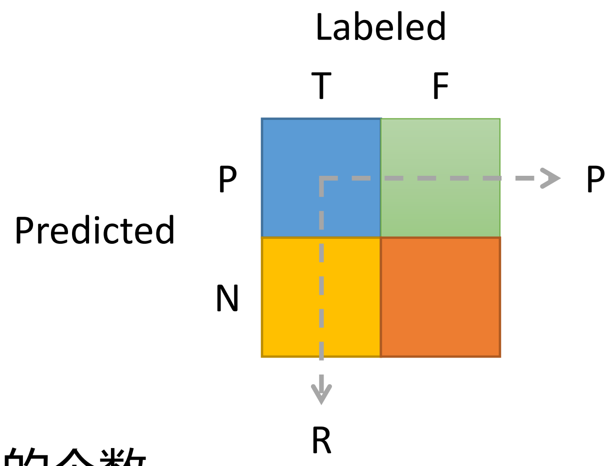
- Synthesis
  - 3. Output predicting boxes (NMS)
    - Many anchor boxes could be produced
    - Remove similar ones





```
def nms(dets, thresh):
    # boxes 位置
    x1 = dets[:, 0]
    y1 = dets[:, 1]
    x2 = dets[:, 2]
    y2 = dets[:, 3]
    # boxes scores
    scores = dets[:, 4]

    areas = (x2 - x1 + 1) * (y2 - y1 + 1) # 各 box 的面积
    order = scores.argsort()[::-1] # boxes 的按照 score 排序

    keep = [] # 记录保留下的 boxes
    while order.size > 0:
        i = order[0] # score 最大的 box 对应的 index
        keep.append(i) # 将本轮 score 最大的 box 的 index 保留

        # 计算剩余 boxes 与当前 box 的重叠程度 IoU
        xx1 = np.maximum(x1[i], x1[order[1:]])
        yy1 = np.maximum(y1[i], y1[order[1:]])
        xx2 = np.minimum(x2[i], x2[order[1:]])
        yy2 = np.minimum(y2[i], y2[order[1:]])

        w = np.maximum(0.0, xx2 - xx1 + 1)
        h = np.maximum(0.0, yy2 - yy1 + 1)
        inter = w * h
        # IoU
        ovr = inter / (areas[i] + areas[order[1:]] - inter)

        # 保留 IoU 小于设定阈值的 boxes
        inds = np.where(ovr <= thresh)[0]
        order = order[inds + 1]
    return keep
```

Object Detection - Basic Knowledge



- mAP - mean Average Precision
  - TP: No. of detecting boxes when IoU >= Thresh
  - FP: No. of detecting boxes when IoU < Thresh
  - TN: No. of non-detected gound-truth objects
  - Precision(P) = TP/(TP+FP) = 正确检测的个数/所有检测到的个数
  - Recall (R) = TP/(TP+TN) = 正确检测的个数/实际有的(人工标注)的个数
  - E.g.
    - There should be 10 boxes for **a certain category,** 8 are predicted among which 6 are right
    - P = 6 / 8 and R = 6 / 10

Object Detection - Basic Knowledge

- mAP - mean Average Precision
  - P-R Curve: P-y axis, R-x axis
    - Sort the predicting boxes **for a certain category** according to their confidence from High to Low
    - Calculate Precision and Recall accumulatively

Object Detection - Basic Knowledge

- mAP - Example

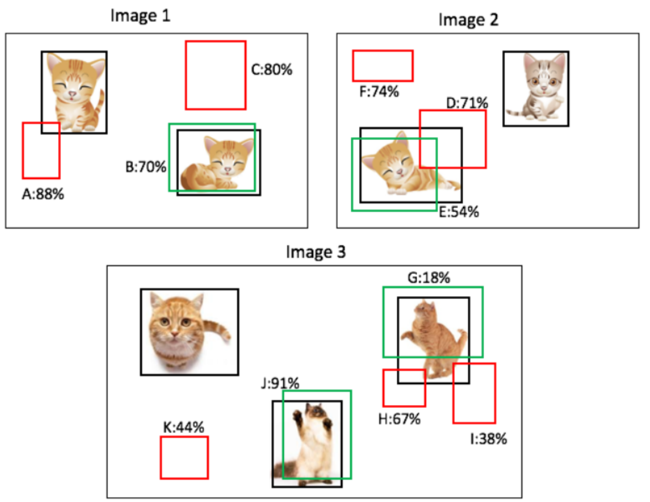
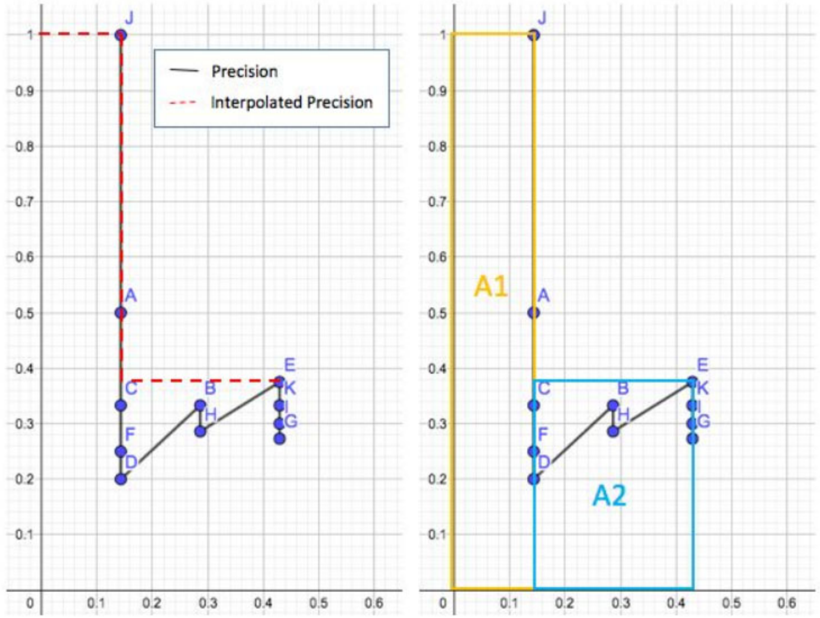


image	Detections	confidences	TP	FP	累积 TP	累积 FP	Precision	Recall
Image3	J	91%	1	0	1	0	1.000	0.143
Image1	A	88%	0	1	1	1	0.500	0.143
Image1	C	80%	0	1	1	2	0.333	0.143
Image2	F	74%	0	1	1	3	0.250	0.143
Image2	D	71%	0	1	1	4	0.200	0.143
Image1	B	70%	1	0	2	4	0.333	0.286
Image3	H	67%	0	1	2	5	0.286	0.286
Image2	E	54%	1	0	3	5	0.375	0.429
Image3	K	44%	0	1	3	6	0.333	0.429
Image3	I	38%	0	1	3	7	0.300	0.429
Image3	G	18%	0	1	3	8	0.273	0.429

Object Detection - Basic Knowledge

- mAP - Example

image	Detections	confidences	TP	FP	累积 TP	累积 FP	Precision	Recall
Image3	J	91%	1	0	1	0	1.000	0.143
Image1	A	88%	0	1	1	1	0.500	0.143
Image1	C	80%	0	1	1	2	0.333	0.143
Image2	F	74%	0	1	1	3	0.250	0.143
Image2	D	71%	0	1	1	4	0.200	0.143
Image1	B	70%	1	0	2	4	0.333	0.286
Image3	H	67%	0	1	2	5	0.286	0.286
Image2	E	54%	1	0	3	5	0.375	0.429
Image3	K	44%	0	1	3	6	0.333	0.429
Image3	I	38%	0	1	3	7	0.300	0.429
Image3	G	18%	0	1	3	8	0.273	0.429



- AP: areas under the PR curve **for a certain category**
- mAP: AP **for all categories**

AP = A1+A2

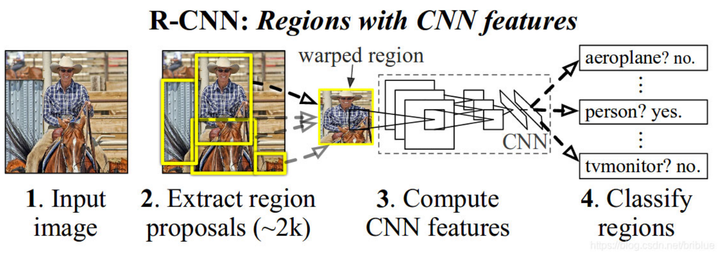
A1 = (0.143 - 0) × 1 = 0.143

A2 = (0.429 - 0.143) × 0.375 = 0.107

AP = 0.143 + 0.107 = 0.250 = 25%

Object Detection - R-CNNs

- R-CNN (Region-based CNN)
  - Extract about 2k region proposals for an input image;
  - Each region will be wrapped into the size that the CNN (a pre-trained network) requires, outputing the features extracted from the proposed regions;
  - Use theses features to train several SVM classifiers, each SVM is used to determine whether an example blongs to a certain category(Y or N);
  - Train a linear regression model for ground-truth bounding box prediction with region features.

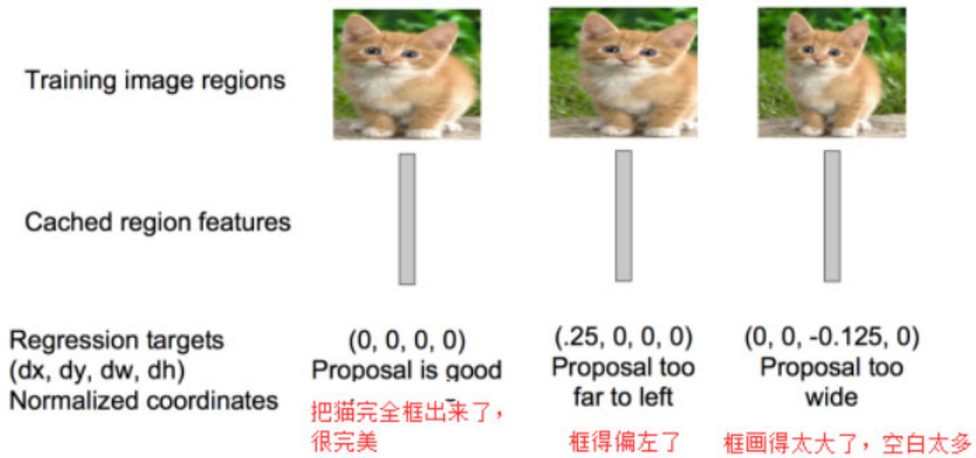


Object Detection - R-CNNs

- R-CNN - Selective Search
  - Divide the image into small regions
  - Merge similar regions (until form the image)
    - Color
    - Pattern (纹理)
    - Small regions first
  - Output all the regions once existed (region proposals)
- R-CNN - Extract features
  - Adjust the size of region proposals, inputting into CNN

Object Detection - R-CNNs

- R-CNN - Classification
  - Train several SVM classifiers to determine whether the object belongs to a certain category
- R-CNN - Adjust b-box position **bounding box regression**

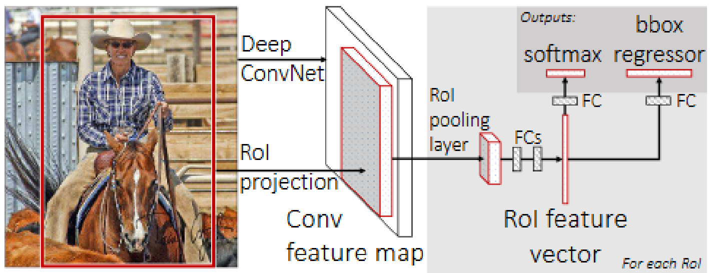


Object Detection - R-CNNs

- R-CNN - Drawbacks
  - **Training process is multi-stage** : adjust ConvNet when putting region proposals, training SVMs, b-box aggression
  - High temporal and spatral cost
  - Low quality of extrated regions
  - Slow : 47'' per image with VGG on GPU

Object Detection - R-CNNs

- Fast R-CNN
  - Take the entire image as the input into a convolutional network;
  - Project multiple regions of interest(RoI) on the feature map;
  - Introduce *RoI pooling layer** to produce fixed-size feature maps and then converted into feature vectors with full-connected layers(FCs)
  - One RoI pooling outputs two vectors: one for predict category and the other for predict b-box



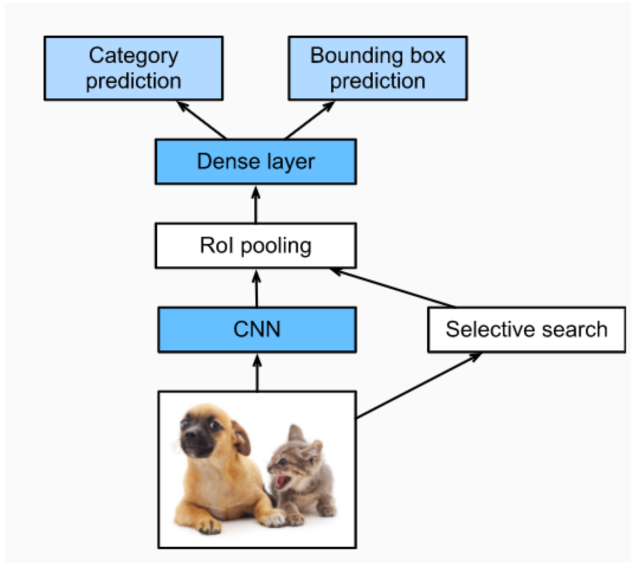
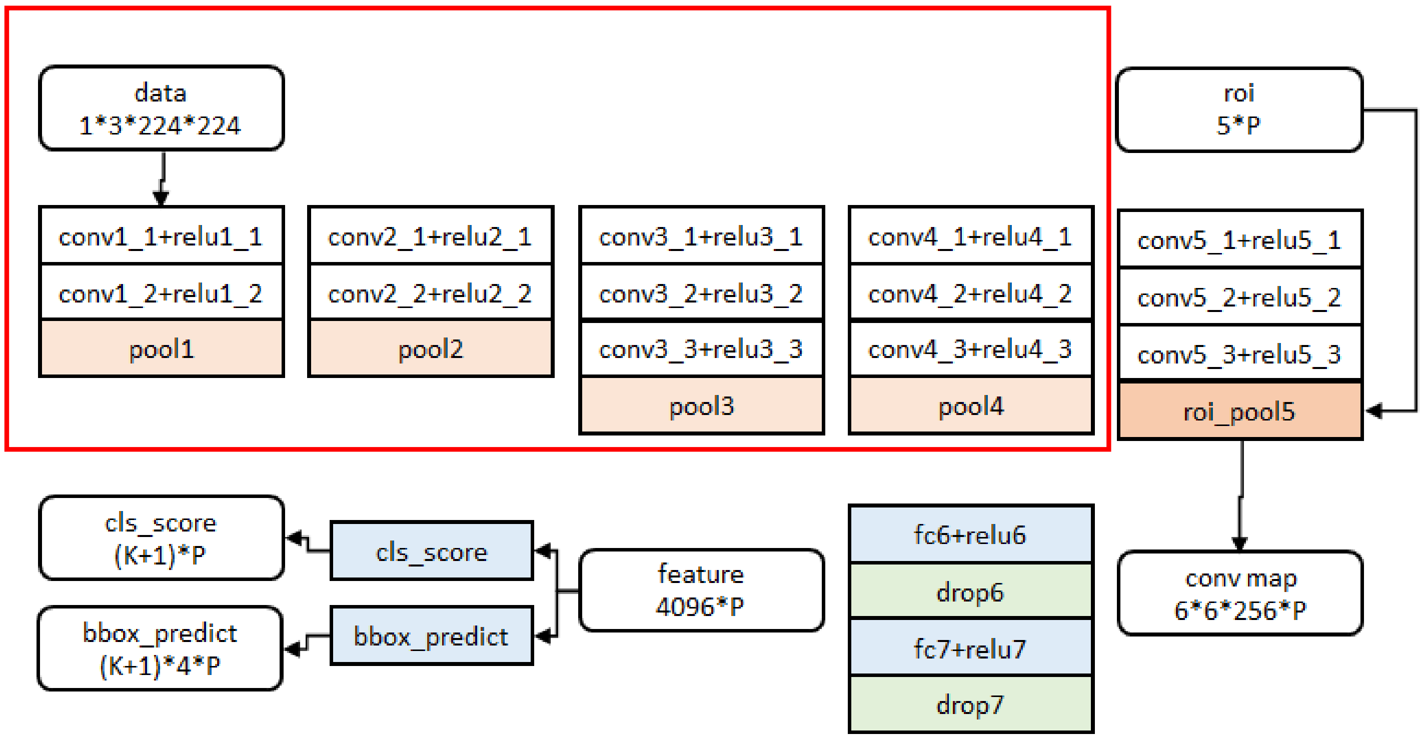


Fig. 13.8.2 Fast R-CNN model.

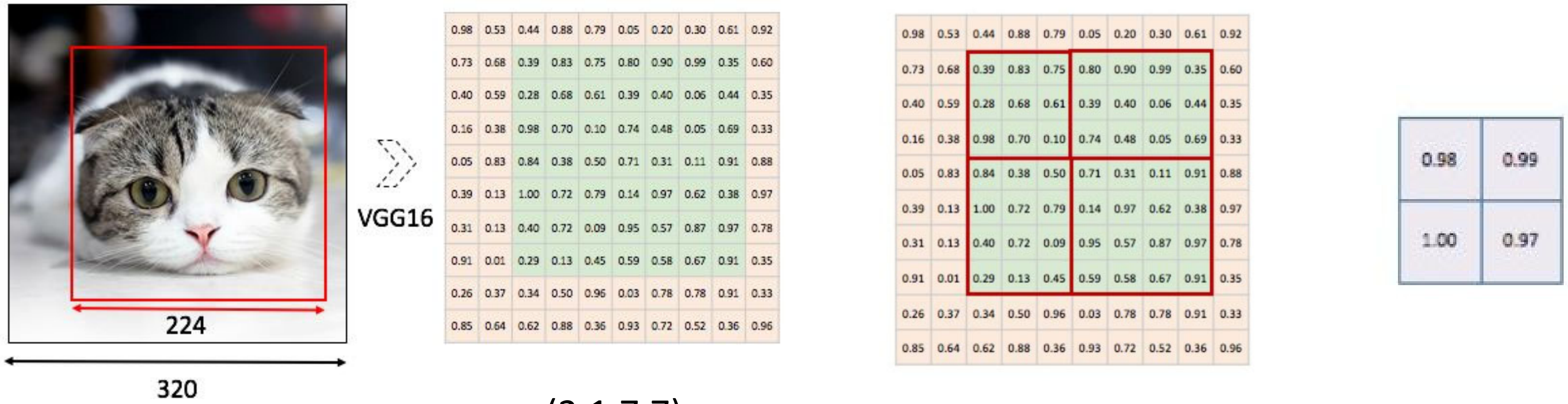
Object Detection - R-CNNs

- Fast R-CNN - Stage1



Object Detection - R-CNNs

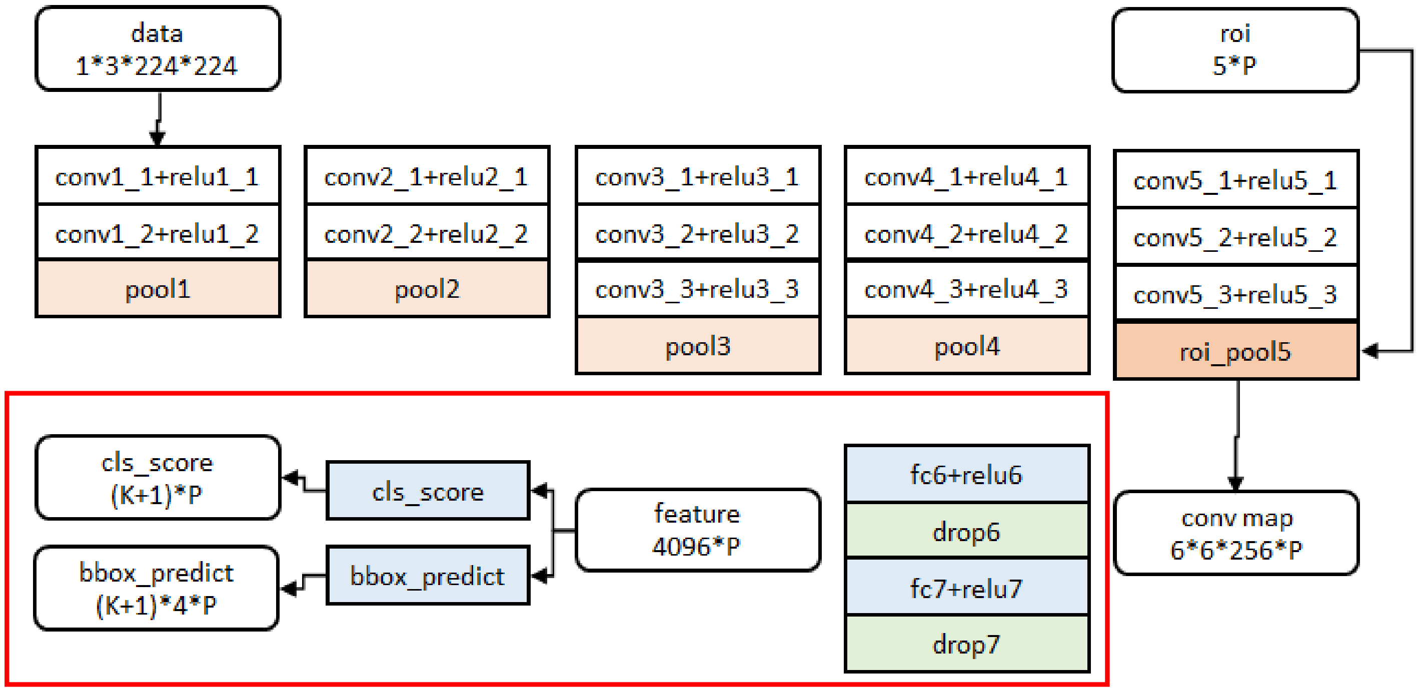
- Fast R-CNN - Stage2 :RoI pooling
  - Assume the size of RoI is *h*w*
  - Set hyper-parameters for the size of output of RoI pooling layer as *H*W*
  - RoI is divided into several sub-windows whose sizes are *(h/H)*(w/W)*
  - *e.g.*



(2,1,7,7)

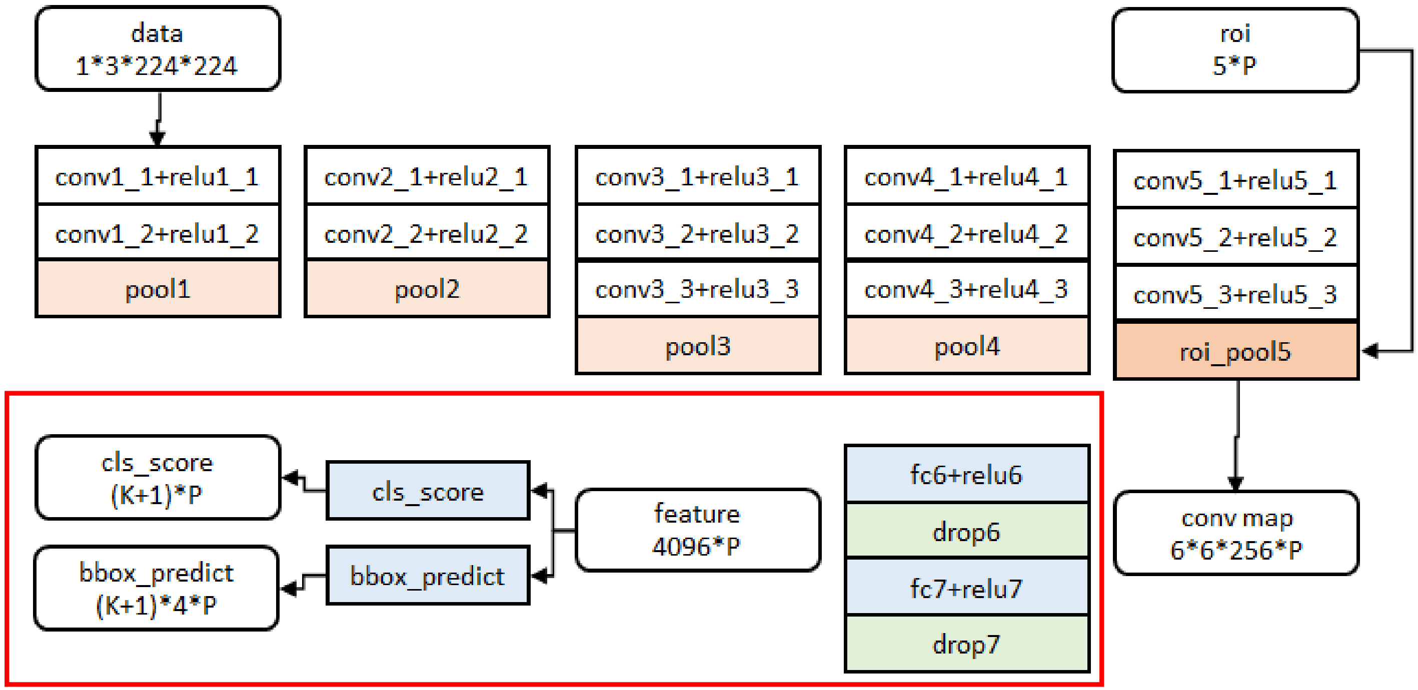
Object Detection - R-CNNs

- Fast R-CNN - Stage3 & 4



Object Detection - R-CNNs

- Fast R-CNN - Stage3 & 4



Object Detection - R-CNNs

- Fast R-CNN - Ads. and Disads.

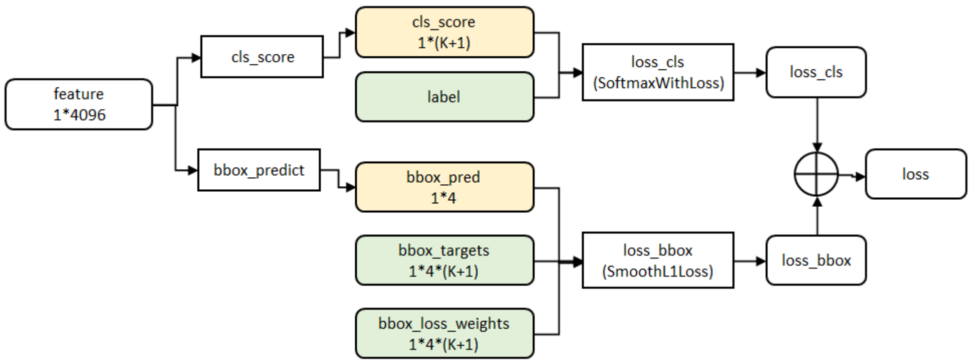
Ads

- Instead input region proposals, Fast R-CNN use the whole image as the input
- Construct **multi-task loss**
- Higher quality (mAP) compared with R-CNN
- Faster than R-CNN

	Training Time	Testing Time
R-CNN	84 H	47 sec
Fast R-CNN	9.5 H	0.32 sec

DisAds

- Still use Selective Search to produce region proposals



Object Detection - R-CNNs

- Faster R-CNN
  - Replace “Selective search” part with a RPN (Region Proposal Network)

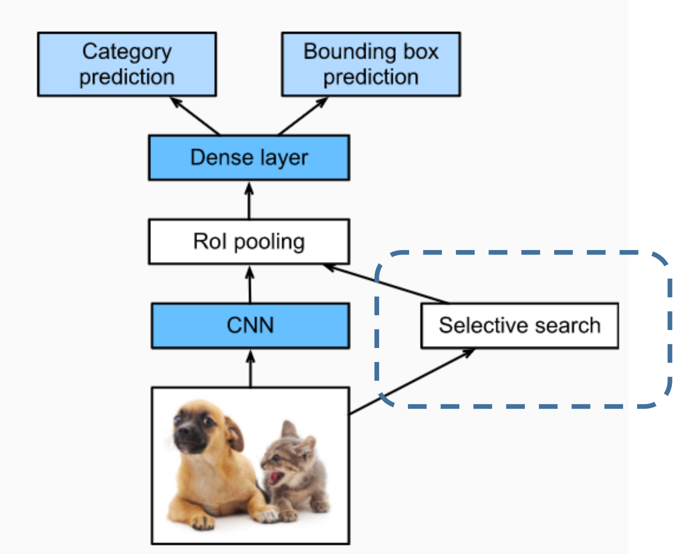


Fig. 13.8.2 Fast R-CNN model.

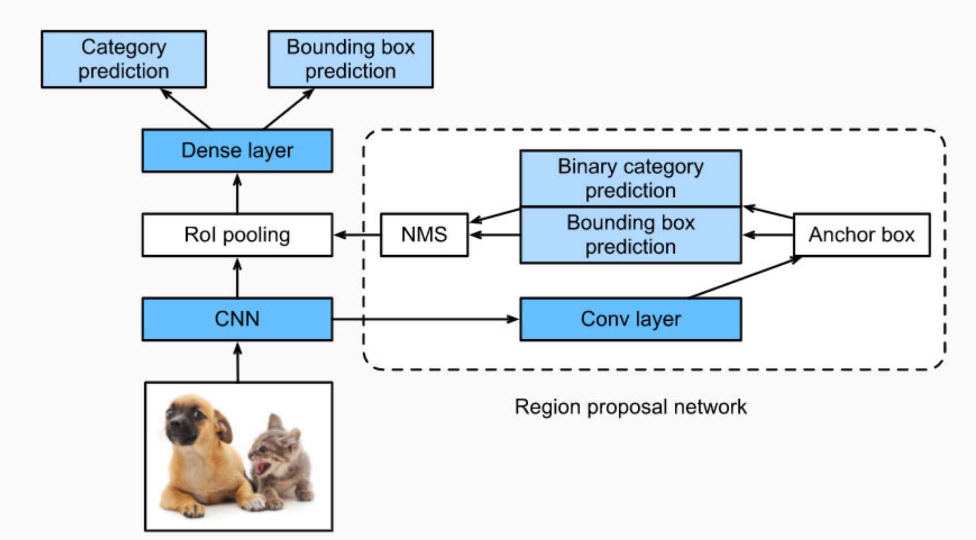


Fig. 13.8.4 Faster R-CNN model.

Reference

- *Dive into Deep Learning,* Chapter 13
- RCNN 论文阅读记录, *https://zhuanlan.zhihu.com/p/42643788*
- Fast RCNN 论文阅读记录, *https://zhuanlan.zhihu.com/p/43037119*
- 【目标检测】Fast RCNN算法详解, *https://blog.csdn.net/shenxiaolu1984/article/details/51036677*