ppt.cc/fNKNzx

Link of slides

Focal Loss for Dense Object Detection (RetinaNet)

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar
(FAIR)

Tutorial Presentation -- PyTorch Taipei
Sept. 20th 2018

王威翔 (Wei-Hisang Wang)

[**Website**](), [**CV**]()

- Education:
  - BS (2016) & MS (2018) in Mechanical Engineering, National Taiwan University (NTU)
- Community:
  - Vice-coordinator, PyTorch Taipei (Feb.-Aug., 2018)
  - Teaching Assistant, Google AI Boot Camp (July, 2018)
- **Job Hunting** recently
  - DL / ML Engineer
  - CV / IoT / Data Analysis
  - 跪求內推

One Rule Today:

Ask Questions ANYTIME

PyTorch Taipei 是個
論文~~報告~~讀書會
➜研討

What's OLD in this paper?

1. **Basic Knowledge of Object Detection**
   - One-stage algo.: [YOLO] / [SSD] / FPN
   - Two-stage algo.: R-CNN / fast R-CNN / faster R-CNN
   - Pros & Cons of these algo.
2. **Cross Entropy**
3. **Feature Pyramid Network (FPN)**
   - Paper
   - The speech given by 郭瑞申 last week. **[Link to be announced]**
4. **Concept of Anchor Box** in RPN (or YOLO/SSD)

What's NEW in this paper?

1. **Focal Loss**
   - Modified from cross entropy (CE)
2. **RetinaNet**
   - Modified from FPN
   - Use anchor box when classification

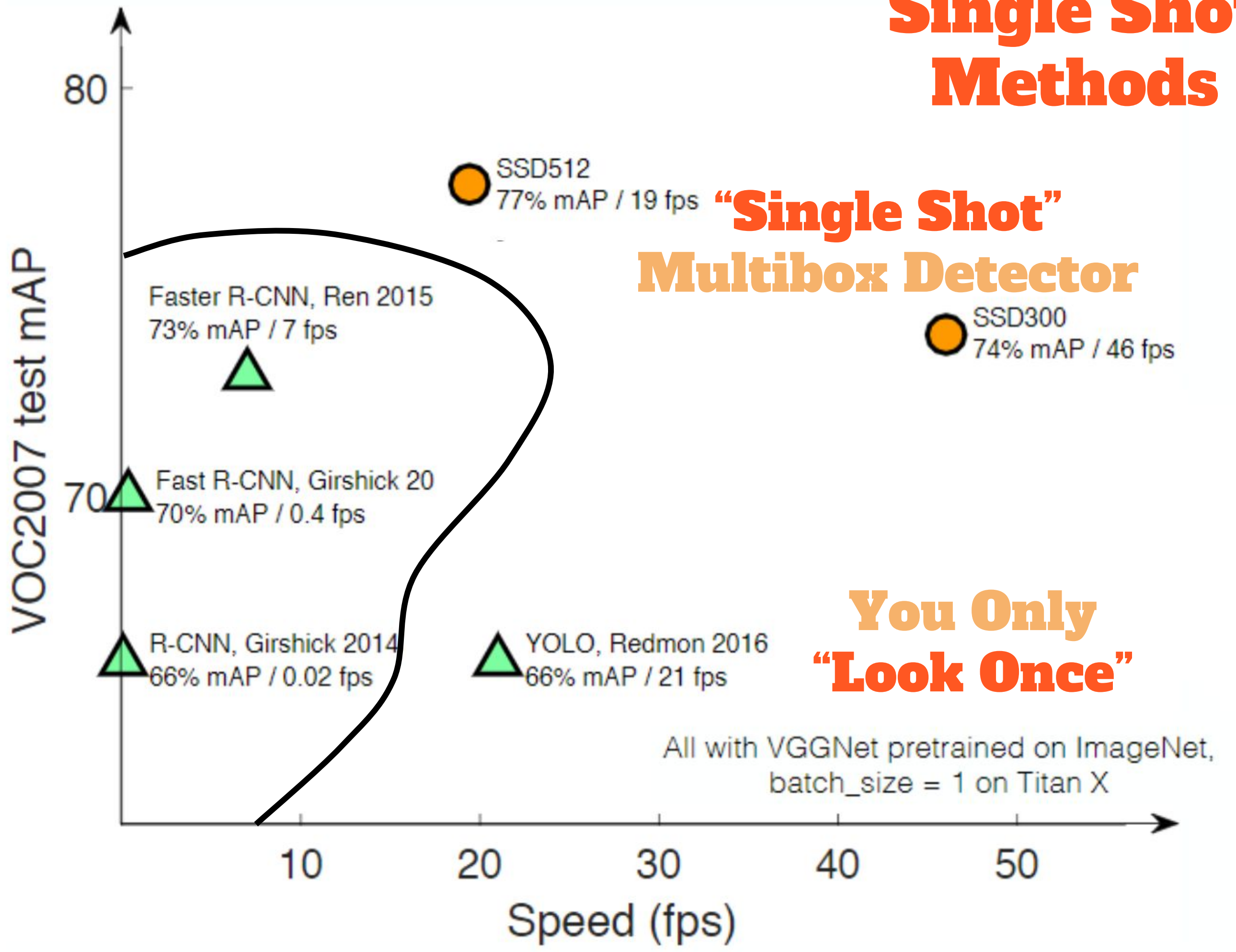
Section 1.

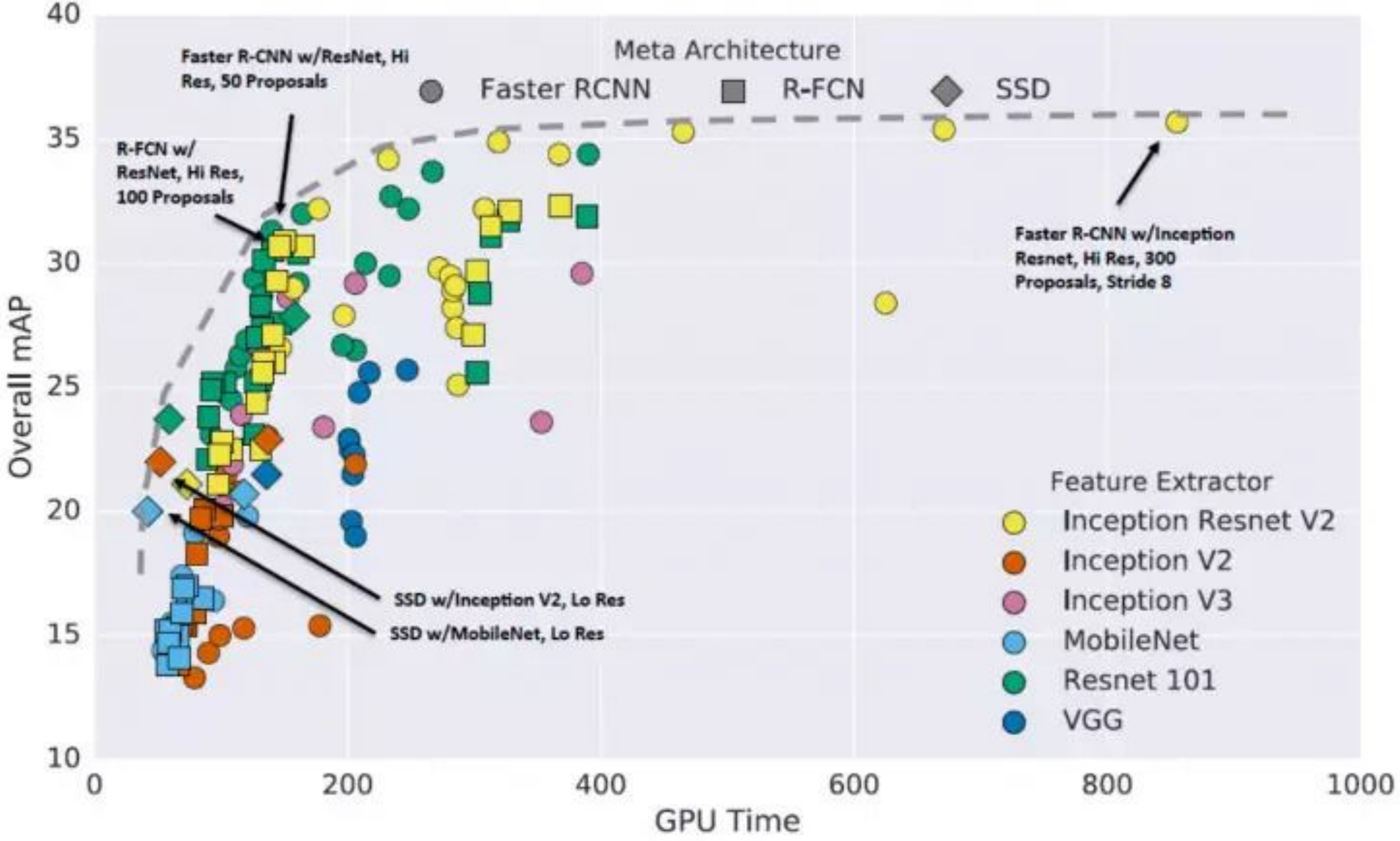
Background

Single Shot Methods

Two-Stage Methods

Box proposal
+
postclassify




Meta Architecture
Faster RCNN
R-FCN
SSD
Faster R-CNN w/ResNet, Hi Res, 50 Proposals
R-FCN w/ ResNet, Hi Res, 100 Proposals
Faster R-CNN w/Inception Resnet, Hi Res, 300 Proposals, Stride 8
SSD w/Inception V2, Lo Res
SSD w/MobileNet, Lo Res
Feature Extractor
Inception Resnet V2
Inception V2
Inception V3
MobileNet
Resnet 101
VGG
Overall mAP
GPU Time
40
35
30
25
20
15
10
0
200
400
600
800
1000

Section 2.

Loss Function

Cross Entropy

1. Information Content
   - Higher the probability of the event x is, it has less information

$$I(x) = -log(p(x))$$

2. Entropy
   - Measurement (expectation) of information
   - Average rate at which information is produced by a stochastic source of data

$$\mathrm{H}(X) = \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log \mathrm{P}(x_i)$$

https://blog.csdn.net/rtygbwwwerr/article/details/50778098
https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

小弟數學不好，歡迎大家補充

Cross Entropy

3. Cross Entropy
   - Measurement between two probability distributions **p** and **q** over the same underlying set of events.

$$H(p,q) = -\sum_{x} p(x)\,\log q(x)$$

   - In the fields of machine learning, CE is often taken as loss function that describing the difference between true data and predicted data.
   - For binary classification,

$$\mathrm{CE}(p,y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise} \end{cases}$$

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases}$$

Cross Entropy

4. α-balanced Cross Entropy
    - Problem of **Class Imbalance**
    - For binary classification, weighting factor α ∈ [0, 1]
    - α can be determined by cross-validation

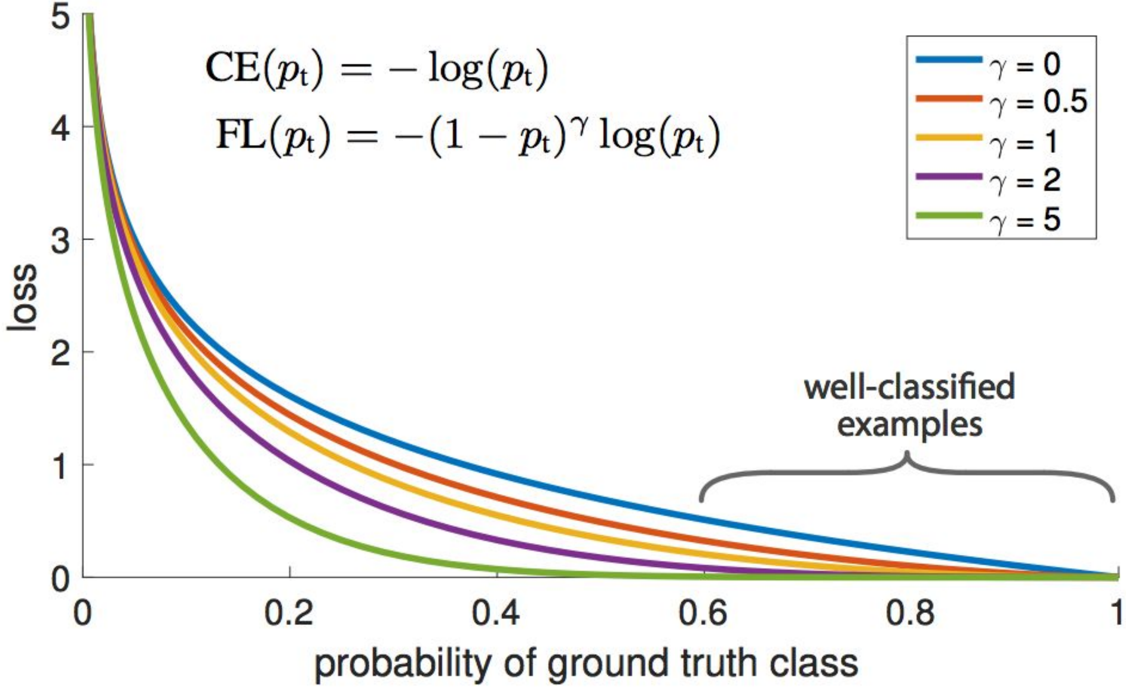
$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t)$$

$$\alpha_t = \begin{cases} \alpha & \text{if } y = 1 \\ 1 - \alpha & \text{otherwise} \end{cases}$$

Class Imbalance

- Both one- & two- stage detectors face the problem of class imbalance
- Classic detectors usually first propose thousands of candidate location, but few actually contain objects.
  - easy examples give less learning signal ➔ **Inefficient training**
  - Too many negatives ➔ **degenerated model**
- Traditional Solution:
  - Robust Estimation
    - Reduce the loss from examples with large errors (hard examples).
    - e.g. α-balanced cross entropy
  - [SSD] hard negative mining
    - Use part of the hard negative examples only (fixed ratio between pos/neg)
- Defect of the solution:
  - Loss from **easy examples** still need to be down-weighted

Focal Loss



(1) Small p (p<0.5)

for γ=2, FL=CE/4 (at most)

➔ loss from misclassified examples unaffected

(2) Large p (p>0.5)

for γ=2, p=0.9,
FL=CE/100

for γ=2, p=0.968,
FL=CE/1000

➔ loss from well-classified examples reduced

α-balanced Focal Loss

- Combine the merits from α-balanced and focal loss.
- Pros:
  - Increasing accuracy
  - Numerical stablity

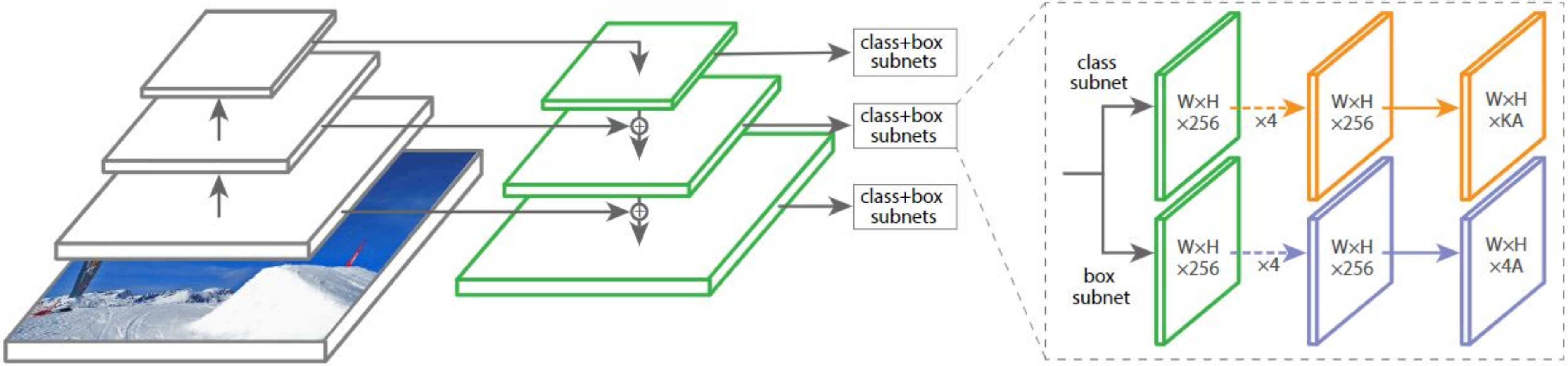
$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

Precise form of loss is not crucial

Section 3.

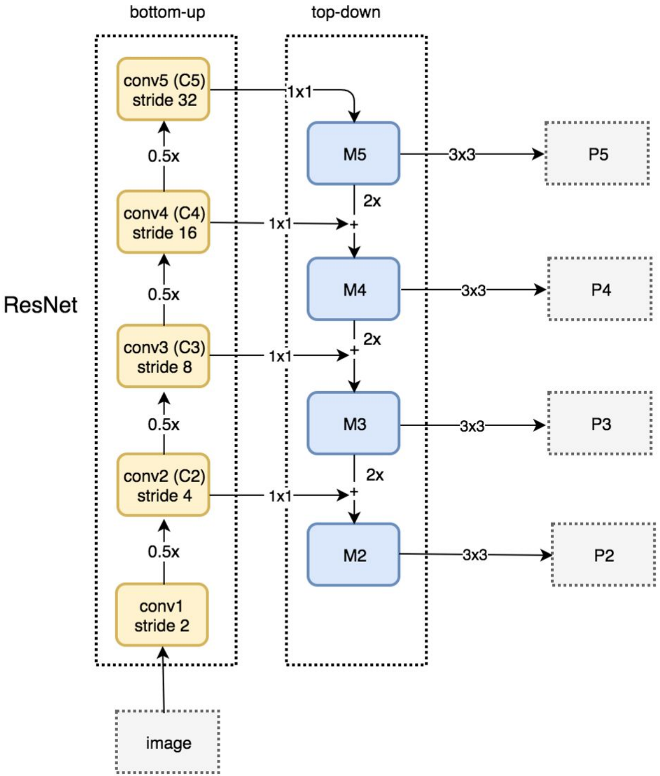
Architecture

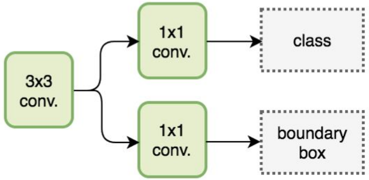
Overall Architecture



(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

FPN-ResNet

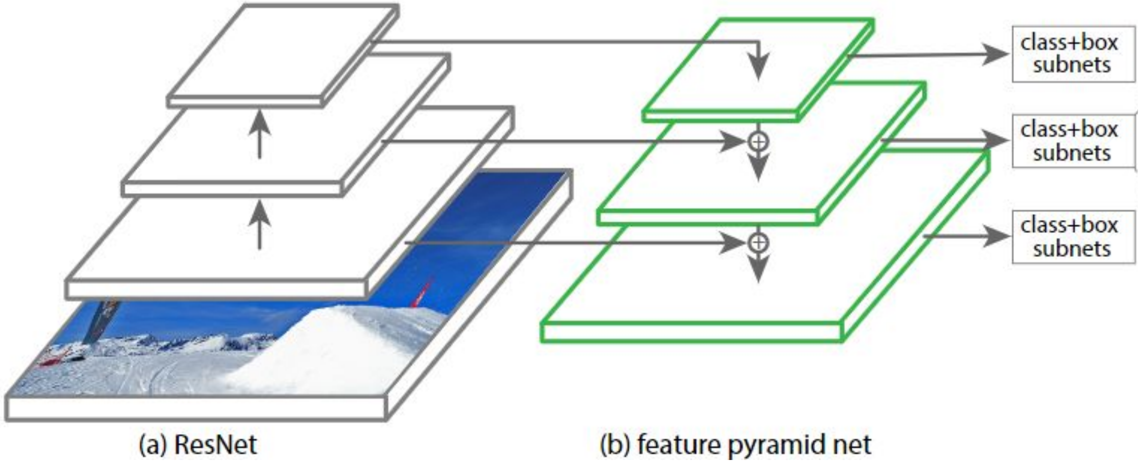




predictor head

RetinaNet

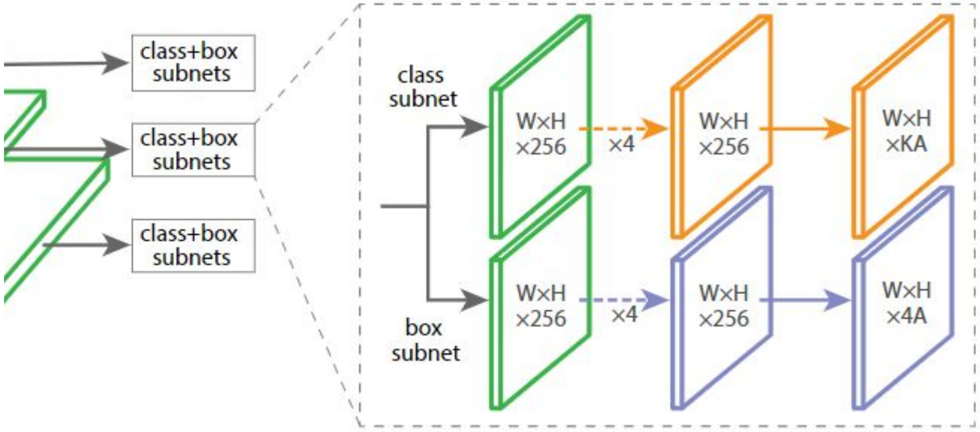
- Use P3~P5 layer of FPN-ResNet
- Deprecate C2 to improve speed
- Add two layer
  - P6: 3x3conv-s2 from C5
  - P7: 3x3conv-s2 & ReLU from P6



→ Actually 5 layer is used

Subnet

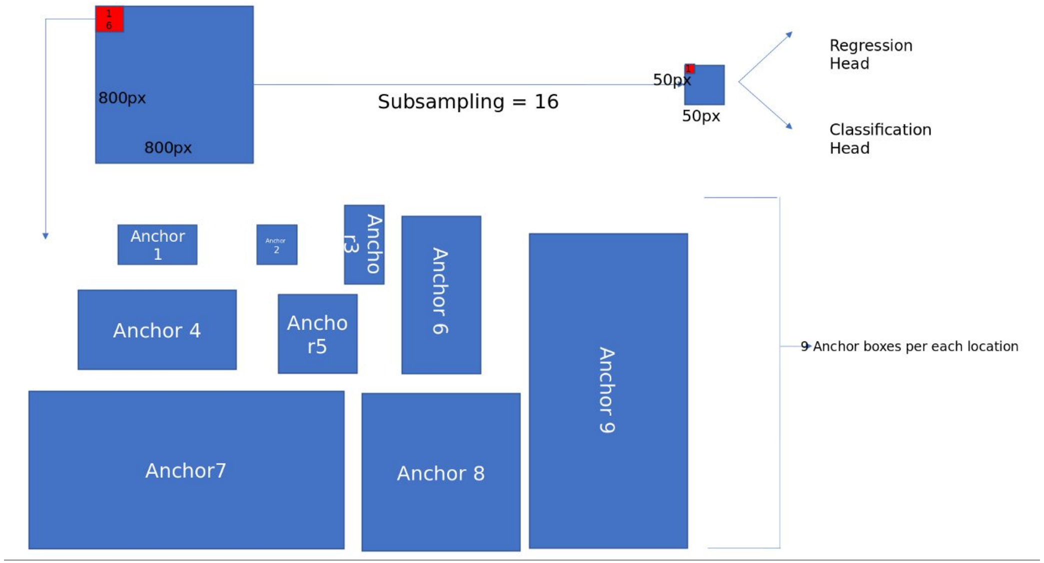
- Two sub-networks
  - bounding box regression
    - stage layer ➔ four 3x3 conv layer ➔ ReLU ➔
      ➔ 3x3 conv layer with **K(class)A(anchor) kernel** ➔ Sigmoid
  - object classification
    - stage layer ➔ four 3x3 conv layer ➔ ReLU ➔
      ➔ 3x3 conv layer with **4A kernel** ➔ Sigmoid



(c) class subnet (top)　　(d) box subnet (bottom)

Anchor

- aspect ratios {1:2; 1:1, 2:1}
- anchor sizes {1 , 2^(⅓) , 2^(⅔)}
- Total 9 anchors at each location
- Anchor assigned to object if IOU>0.5



https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d

Inference

- Decode 1000 top prediction boxes at each FPN level
- Thresholding confidence of 0.05
- Non-maximum suppression with threshold 0.5

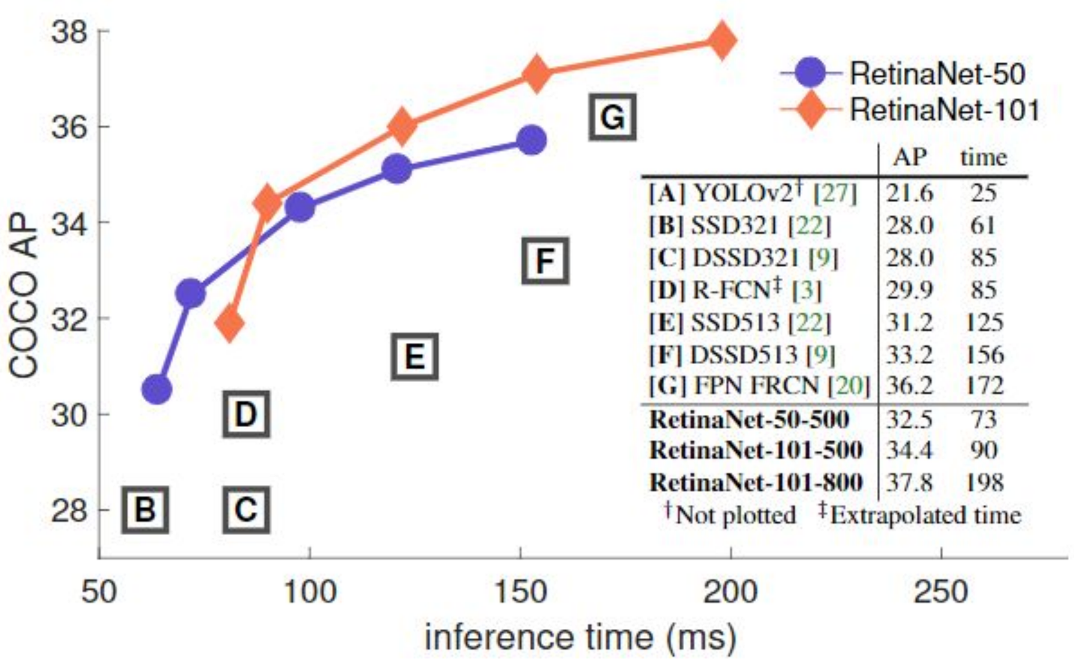
Initialization

- FPN-ResNet pre-trained on ImageNet1k
- variable init:
  - last layer of classification subnet: b = -log((1-pi)/pi), pi=0.1
    ➜ all anchors labeled as foreground at start of training
  - rest: b=0, Gaussian weight with sigma = 0.01

Optimization

- SGD (batch = 16)
- Learning rate:
  - 0~60k iter.: 0.01
  - 60~80k iter.: 0.001
  - 80~90k iter.: 0.0001
- Weight decay 0.0001, momentum=0.9
- Loss = focal loss + smoothL1 loss of box regression

Comparison between models



	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

Comparison between different hyper-para.

α	AP	AP_{50}	AP_{75}
.10	0.0	0.0	0.0
.25	10.8	16.0	11.7
.50	30.2	46.7	32.8
.75	31.1	49.4	33.0
.90	30.8	49.7	32.3
.99	28.7	47.4	29.9
.999	25.1	41.7	26.1

(a) **Varying α for CE loss** ($\gamma = 0$)

γ	α	AP	AP_{50}	AP_{75}
0	.75	31.1	49.4	33.0
0.1	.75	31.4	49.9	33.1
0.2	.75	31.9	50.7	33.4
0.5	.50	32.9	51.7	35.2
1.0	.25	33.7	52.0	36.2
2.0	.25	**34.0**	**52.5**	**36.5**
5.0	.25	32.2	49.6	34.8

(b) **Varying γ for FL** (w. optimal α)

#sc	#ar	AP	AP_{50}	AP_{75}
1	1	30.3	49.0	31.8
2	1	31.9	50.0	34.0
3	1	31.8	49.4	33.7
1	3	32.4	52.3	33.9
2	3	**34.2**	**53.1**	**36.5**
3	3	34.0	52.5	**36.5**
4	3	33.8	52.1	36.2

(c) **Varying anchor scales and aspects**

method	batch size	nms thr	AP	AP_{50}	AP_{75}
OHEM	128	.7	31.1	47.2	33.2
OHEM	256	.7	31.8	48.8	33.9
OHEM	512	.7	30.6	47.0	32.6
OHEM	128	.5	32.8	50.3	35.1
OHEM	256	.5	31.0	47.4	33.0
OHEM	512	.5	27.6	42.0	29.2
OHEM 1:3	128	.5	31.1	47.2	33.2
OHEM 1:3	256	.5	28.3	42.4	30.3
OHEM 1:3	512	.5	24.0	35.5	25.8
FL	n/a	n/a	**36.0**	**54.9**	**38.7**

(d) **FL *vs.* OHEM** baselines (with ResNet-101-FPN)

depth	scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	time
50	400	30.5	47.8	32.7	11.2	33.8	46.1	64
50	500	32.5	50.9	34.8	13.9	35.8	46.7	72
50	600	34.3	53.2	36.9	16.2	37.4	47.4	98
50	700	35.1	54.2	37.7	18.0	39.3	46.4	121
50	800	35.7	55.0	38.5	18.9	38.9	46.3	153
101	400	31.9	49.5	34.1	11.6	35.8	48.5	81
101	500	34.4	53.1	36.8	14.7	38.5	49.1	90
101	600	36.0	55.2	38.7	17.4	39.6	49.7	122
101	700	37.1	56.6	39.8	19.1	40.6	49.4	154
101	800	37.8	57.5	40.8	20.2	41.1	49.2	198

(e) **Accuracy/speed trade-off** RetinaNet (on `test-dev`)