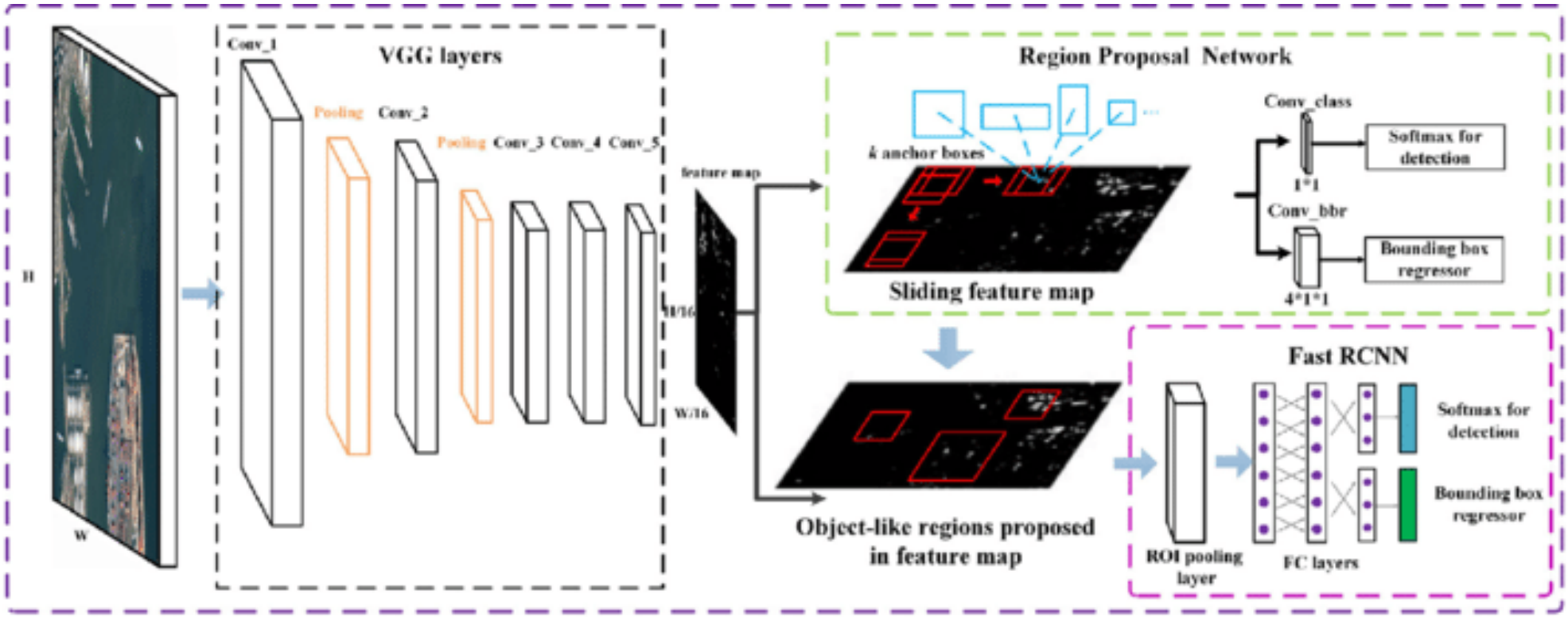
Focal loss for dense object detection

Lin, Tsung-Yi, et al. "Focal loss for dense object detection." *IEEE transactions on pattern analysis and machine intelligence*(2018).

One-stage vs Two-stage Detectors

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Two-stage Detectors

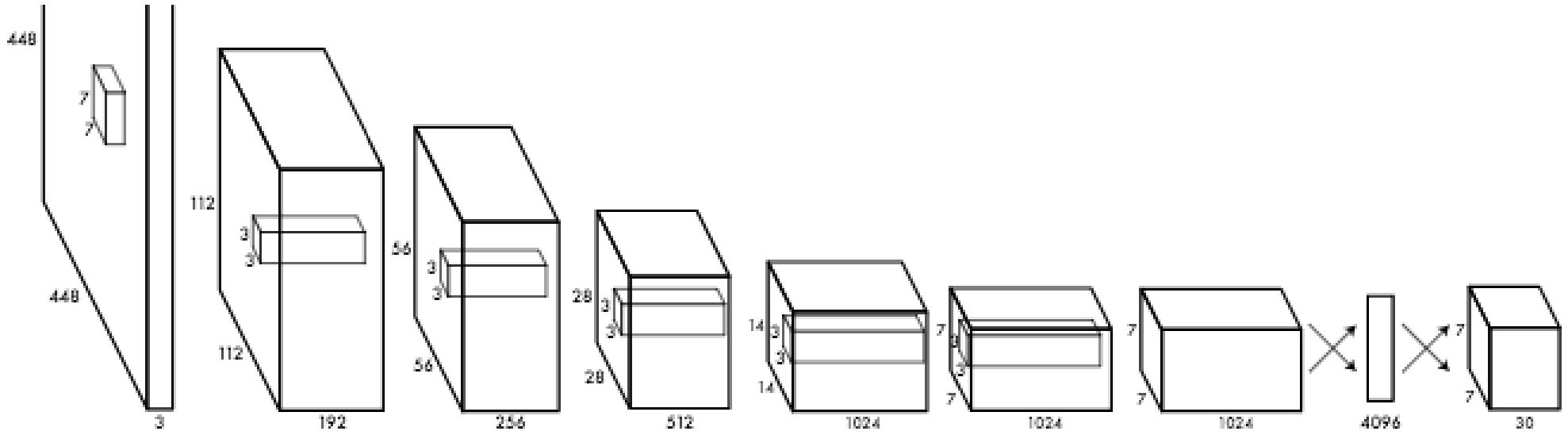


Two-stage Detectors

- 복잡하다
- 느리다

- 하지만 정확하다

One-stage Detectors



One-stage Detectors

- 빠르다
- 가볍다
- 구현하기도 쉽다

- 하지만 정확성이 떨어진다

One-stage Detectors

- number of Hard Positives (맞추기 어려운 물체들) 가 매우 적음
- number of Easy Negatives (맞추기 쉬운 배경) 가 매우 많음

-> 트레이닝을 방해

많은 Easy Negative들이 학습 과정을 압도한다. (overwhelming)

Cross Entropy

$$-plog(\hat{p}) - (1-p)log(1-\hat{p})$$

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

$$p_{\mathrm{t}} = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases}$$

$$\mathrm{CE}(p, y) = \mathrm{CE}(p_{\mathrm{t}}) = -\log(p_{\mathrm{t}})$$

ure 1. One notable property of this loss, which can be easily seen in its plot, is that even examples that are easily classified ($p_t \gg .5$) incur a loss with non-trivial magnitude. When summed over a large number of easy examples, these small loss values can overwhelm the rare class.

Balanced Cross Entropy

weighting factor $\alpha \in [0, 1]$ for class 1 and $1-\alpha$ for class -1
α may be set by inverse class frequency or treated as a hyperparameter

$$\mathrm{CE}(p_\mathrm{t}) = -\alpha_\mathrm{t} \log(p_\mathrm{t})$$

$$-\alpha p log(\hat{p}) - (1-\alpha)(1-p) log(1-\hat{p})$$

This loss is a simple extension to CE that we consider as an experimental baseline for our proposed focal loss.

Focal Loss

we propose to reshape the loss function to down-weight easy examples
and thus focus training on hard negatives

$$\mathrm{FL}(p_\mathrm{t}) = -(1-p_\mathrm{t})^{\gamma}\log(p_\mathrm{t})$$

감마는 해보니깐 2가 제일로 좋더라! (we found $\gamma = 2$ to work best in our experiments).

$$-(1-\hat{p})^2 p log(\hat{p}) - (\hat{p})^2(1-p)log(1-\hat{p})$$

배경 (easy to predict) or 강아지 (hard to predict)

$$-plog(\hat{p}) - (1-p)log(1-\hat{p})$$

– 강아지인 확률 * log(강아지로 예측한 확률) – 배경인 확률 * log(배경으로 예측한 확률)

실제	예측	예측 확률		
강아지	배경	0.89	**0.958607315**	- 1 * log(0.11) – 0 * log(0.89)
배경	배경	0.89	**0.050609993**	- 0 * log(0.11) – 1 * log(0.89)

배경 (easy to predict) or 강아지 (hard to predict)

$$-(1-\hat{p})^2 p log(\hat{p}) - (\hat{p})^2 (1-p) log(1-\hat{p})$$

– modulating factor * 강아지인 확률 * log(강아지로 예측한 확률) – modulating factor * 배경인 확률 * log(배경으로 예측한 확률)

실제	예측	예측 확률		
강아지	배경	0.89	**0.759312854**	- 0.89^2 * 1 * log(0.11) – 0.11^2 * 0 * log(0.89)
배경	배경	0.89	**0.000612381**	- 0.89^2 * 0 * log(0.11) – 0.11^2 * 1 * log(0.89)

배경 (easy to predict) or 강아지 (hard to predict)

$$-plog(\hat{p}) - (1-p)log(1-\hat{p})$$
$$-(1-\hat{p})^2 plog(\hat{p}) - (\hat{p})^2(1-p)log(1-\hat{p})$$

실제	예측	예측 확률		실제	예측	예측 확률	
강아지	배경	0.89		배경	배경	0.89	
0.958607315		- 1 * log(0.11) – 0 * log(0.89)		**0.050609993**		- 0 * log(0.11) – 1 * log(0.89)	

실제	예측	예측 확률		실제	예측	예측 확률	
강아지	배경	0.89		배경	배경	0.89	
0.759312854		- 0.89^2 * 1 * log(0.11) – 0.11^2 * 0 * log(0.89)		**0.000612381**		- 0.89^2 * 0 * log(0.11) – 0.11^2 * 1 * log(0.89)	

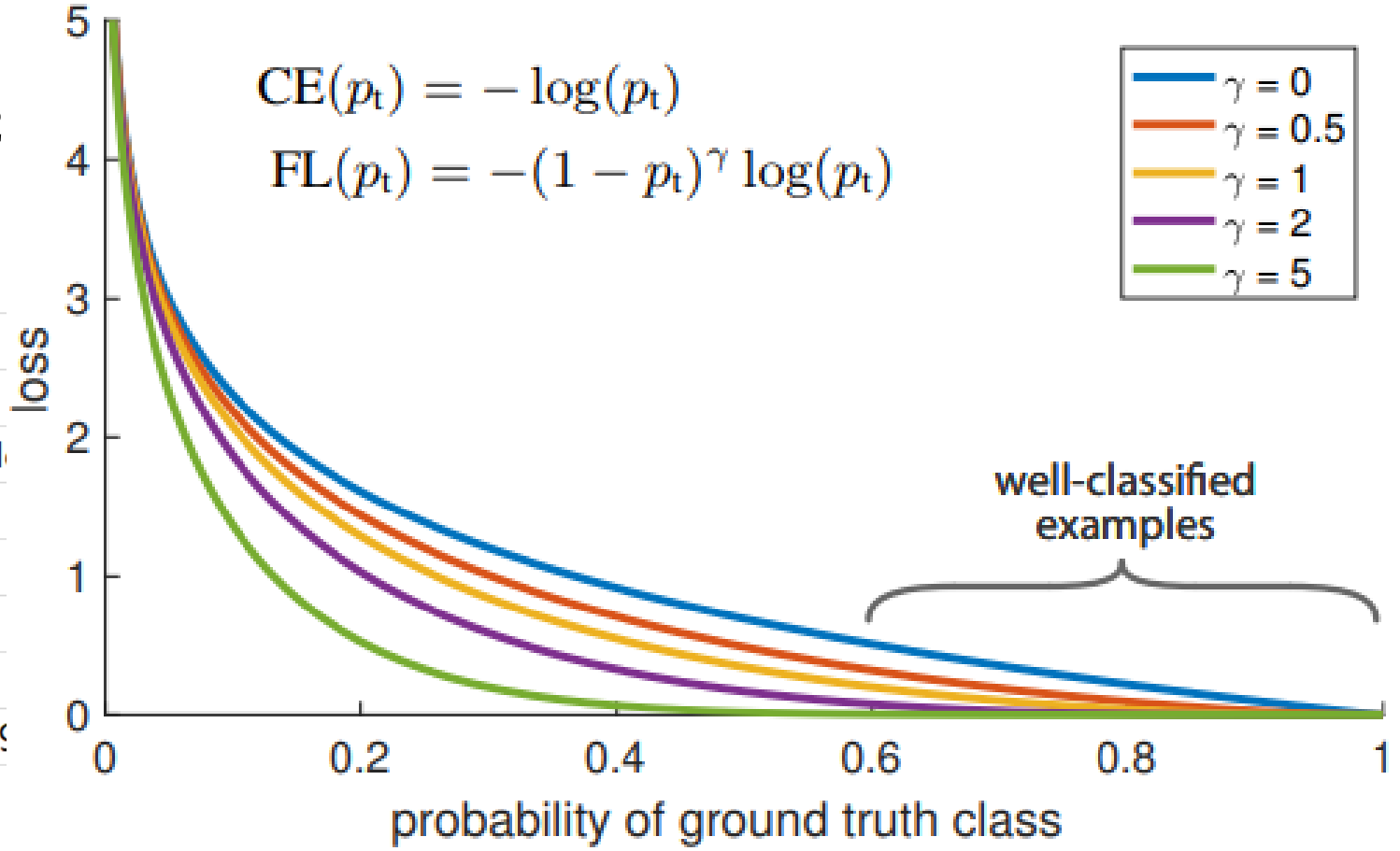
배경 (easy to predict) or 강아지 (hard to predict)

$-plog(p\hat{})$

$-(1-p\hat{})^2$

실제	예측	예측
강아지	배경	
0.958607315		- 1 * l

실제	예측	예측
강아지	배경	
0.759312854		- 0.8



· 1 * log(0.89)

)g(0.11) – 0.11^2 * 1 * log(0.89)

이미 **예측을 잘하는 쉬운 문제**(배경)의 loss는 **down-weight**
예측을 잘 못하고 있는 어려운 문제(강아지)의 loss는 **그대로**

결과물 (+ Balanced Cross Entropy)

$$\mathrm{FL}(p_\mathrm{t}) = -\alpha_\mathrm{t}(1-p_\mathrm{t})^\gamma \log(p_\mathrm{t})$$

Balanced Cross Entropy 추가 하니깐 더 잘되더라!

We adopt this form in our experiments as it yields slightly improved accuracy over the non-α-balanced form. Finally,

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

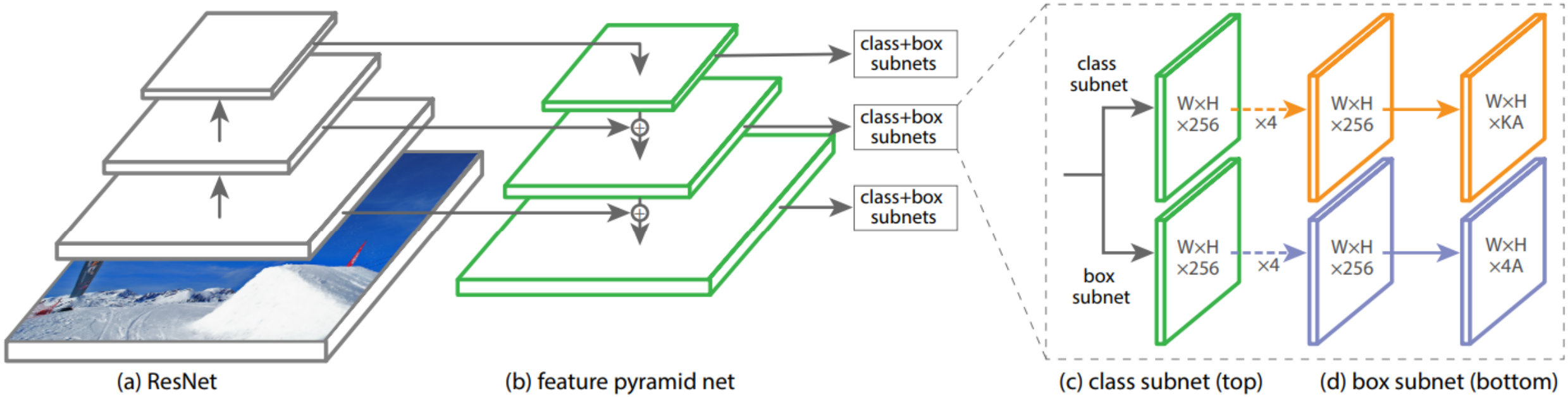
(a) **Varying α for CE loss** ($\gamma = 0$)

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)

RetinaNet

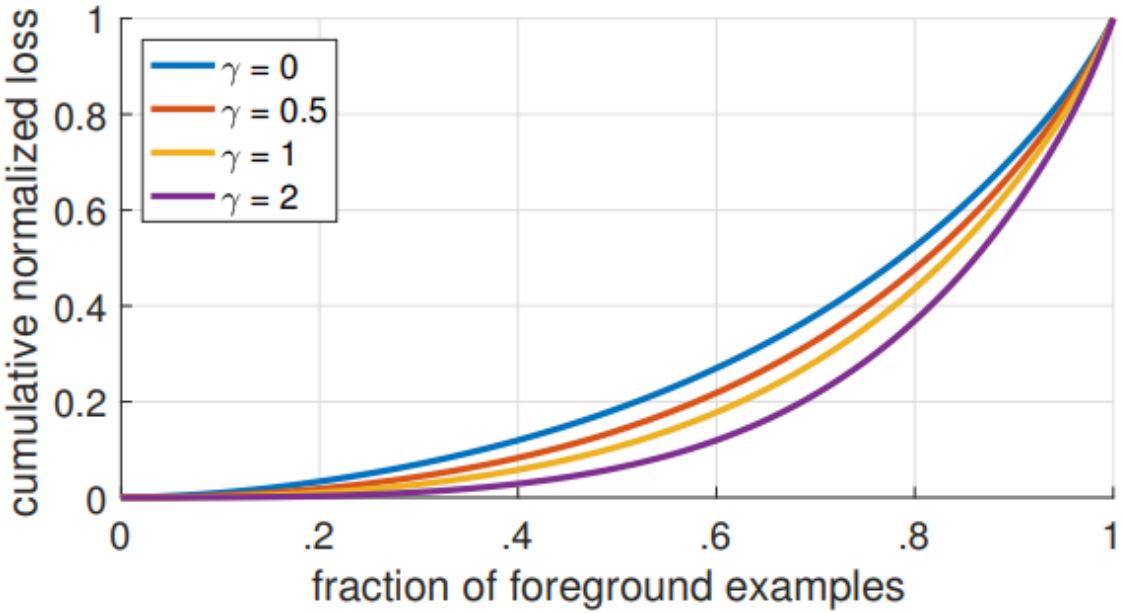
RetinaNet

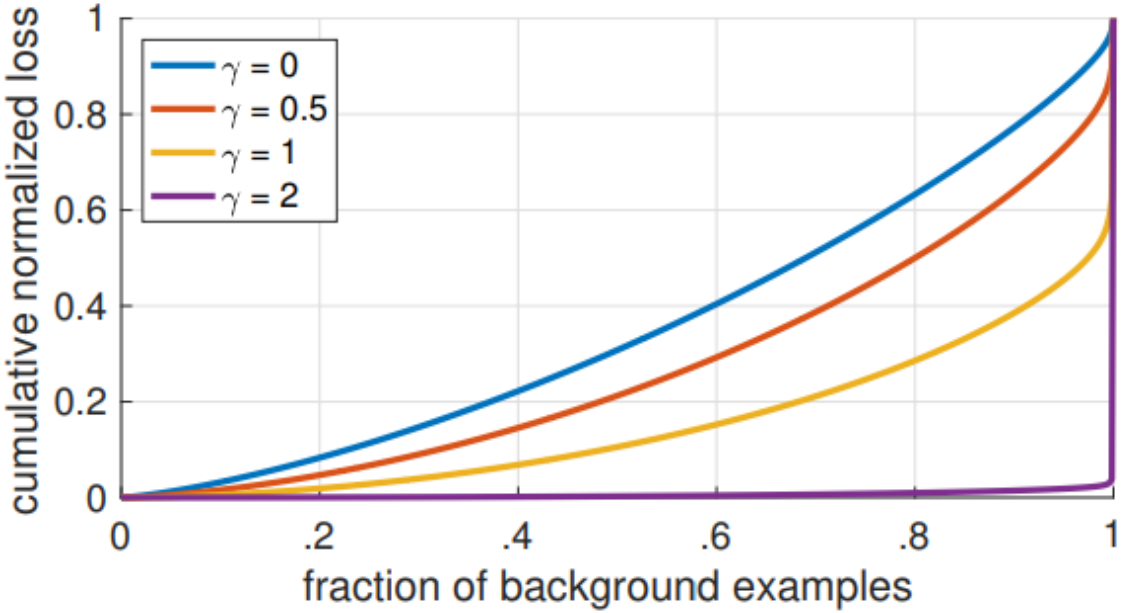


(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

Experiments

Analysis of the Focal Loss





Focal Loss
vs OHEM(online hard example mining)

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)

About RetinaNet

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

About RetinaNet

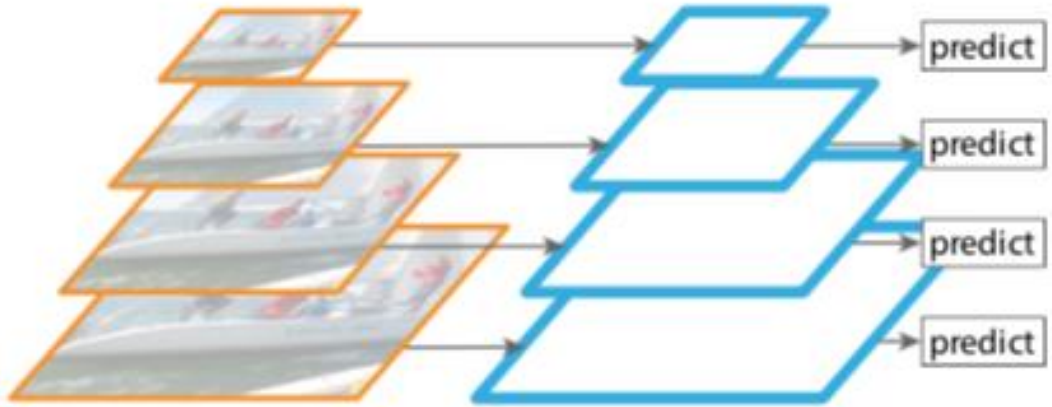
#sc	#ar	AP	AP_{50}	AP_{75}
1	1	30.3	49.0	31.8
2	1	31.9	50.0	34.0
3	1	31.8	49.4	33.7
1	3	32.4	52.3	33.9
2	3	**34.2**	**53.1**	**36.5**
3	3	34.0	52.5	**36.5**
4	3	33.8	52.1	36.2

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

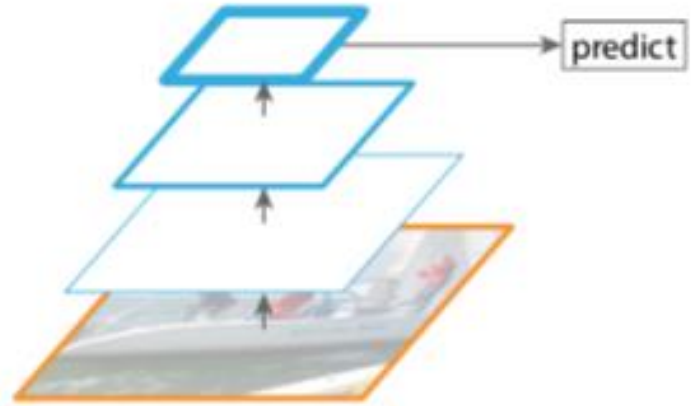
depth	scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	time
50	400	30.5	47.8	32.7	11.2	33.8	46.1	64
50	500	32.5	50.9	34.8	13.9	35.8	46.7	72
50	600	34.3	53.2	36.9	16.2	37.4	47.4	98
50	700	35.1	54.2	37.7	18.0	39.3	46.4	121
50	800	35.7	55.0	38.5	18.9	38.9	46.3	153
101	400	31.9	49.5	34.1	11.6	35.8	48.5	81
101	500	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`)

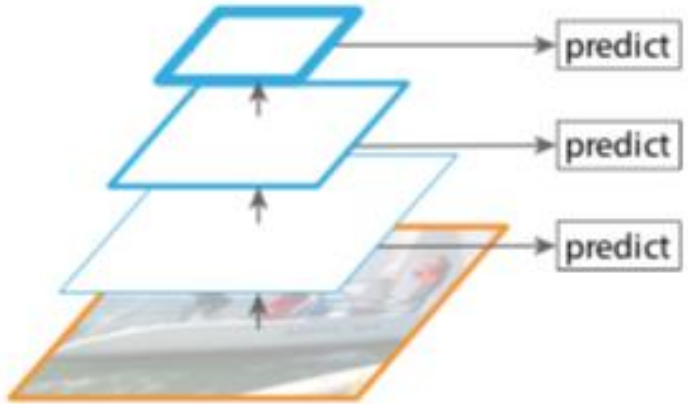
Appendix.



(a) Featurized image pyramid



(b) Single feature map



(c) Pyramidal feature hierarchy

(d) Feature Pyramid Network