Focal Loss for Dense Object Detection

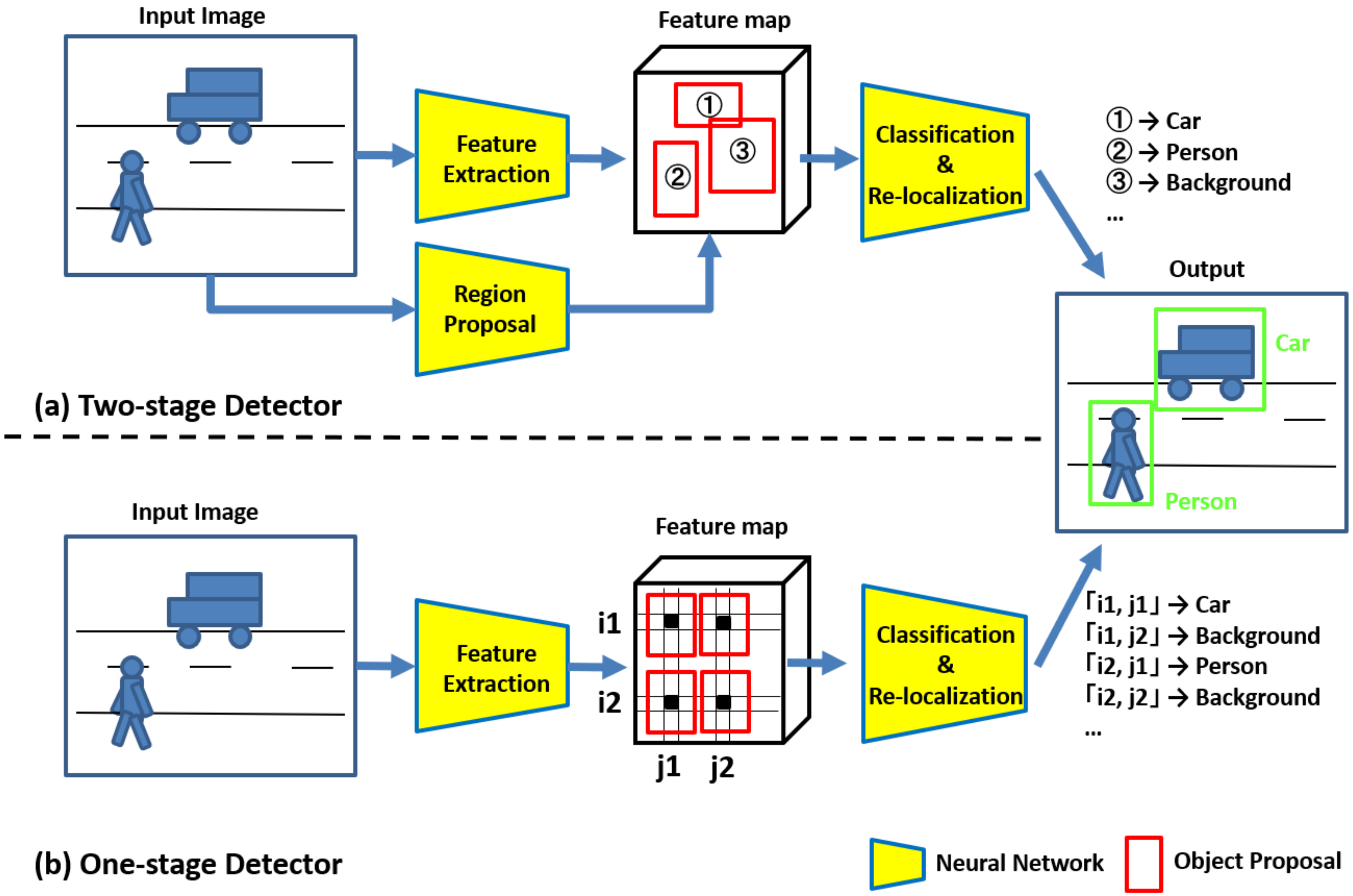
Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr Dollar
Facebook AI Research (FAIR)

Mohamed Ahmed Abdel-Rahman Mohamed
2000694@eng.asu.edu.eg

Introduction

- Object detection is one of the most widely studied topics in the computer vision community.
- Currently, deep learning-based object detection can be majorly classified into two groups:
  A. Two-stage detectors, such as Region-based CNN (R-CNN)
  B. One-stage detectors, such as the YOLO family of detectors and SSD
- Two-stage detectors detect objects in two successive steps. Firstly, they use a region proposal generator to generate a set of object-like proposals and extract features from each proposal, which are then fed to a classifier that predicts the category of the proposed regions.
- One-stage detectors directly make categorical predictions of pre-defined proposals generated from each location of the feature maps regardless of whether it contains an object or a background region.
- Commonly, two-stage detectors achieve relatively better detection performance, whereas one-stage detectors are significantly faster and have greater applicability to real-time object detection.

Two-Stage Detector vs. One-Stage Detector



(a) Two-stage Detector

(b) One-stage Detector

Class Imbalance

- One-stage detectors have the potential to be faster and simpler but have trailed the accuracy of two-stage detectors because of extreme class imbalance encountered during training.
- One-stage detection methods, evaluate almost 10^4 to 10^5 candidate locations per image but only a few locations contain objects (i.e. Foreground) and rest are just background objects, and this leads to the class imbalance problem.
- This imbalance causes two problems:

  A. Training is inefficient as most locations are easy negatives (meaning that they can be easily classified by the detector as background) that contribute no useful learning.

  B. Since easy negatives (detections with high probabilities) account for a large portion of inputs. Although they result in small loss values individually but collectively, they can overwhelm the loss & computed gradients and can lead to degenerated models.
- FAIR has released a paper in 2018, in which they introduced the concept of Focal loss to handle this class imbalance problem with their one stage detector called RetinaNet.

Focal Loss

- In simple words, Focal Loss (FL) is an improved version of Cross-Entropy Loss (CE) that tries to handle the class imbalance problem by assigning more weights to hard or easily misclassified examples (i.e. background with noisy texture or partial object or the object of our interest) and to down-weight easy examples (i.e. Background objects).
- Focal Loss reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples.
- In general, *Cross Entropy is used to formularize the classification loss of deep learning-based object detectors*

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

$$p_{\mathrm{t}} = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases} \qquad (2)$$

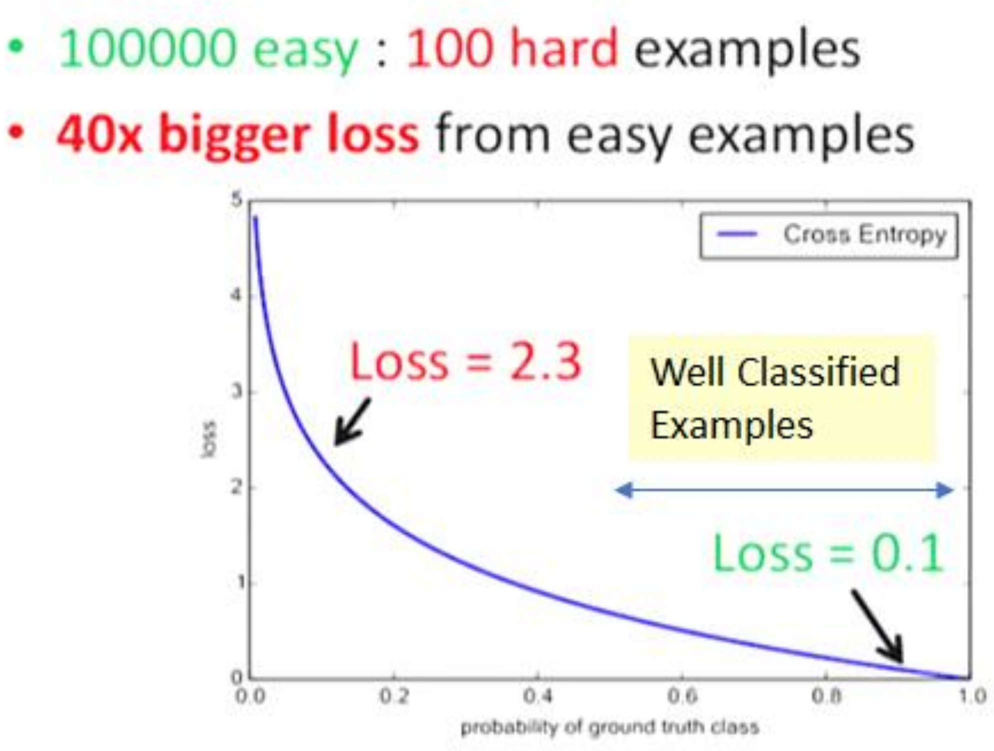
$$\mathrm{CE}(p, y) = \mathrm{CE}(p_{\mathrm{t}}) = -\log(p_{\mathrm{t}})$$

$y \in \{\pm 1\}$ specifies the ground-truth class

$p \in [0, 1]$ is the model's estimated probability

Focal Loss

- If we have 100000 easy examples (0.1 each) and 100 hard examples (2.3 each). When we need to sum over to estimate the CE loss.
- The loss from easy examples = 100000×0.1 = 10000
- The loss from hard examples = 100×2.3 = 230
- 10000 / 230 = 43. It is about 40× bigger loss from easy examples.
- Thus, CE loss is not a good choice when there is extreme class imbalance.



- 100000 easy : 100 hard examples
- **40x bigger loss** from easy examples

Focal Loss

- A common method for addressing class imbalance is to introduce a weighting factor α [0; 1] for class 1 and 1- α for class -1 (α-Balanced CE).

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

- α may be set by inverse class frequency or treated as a hyperparameter to set by cross validation.
- While α balances the importance of positive/negative examples, it does not differentiate between easy/hard examples.
- And that's where Focal loss (extension to cross-entropy) comes to rescue.

Focal Loss

- The loss function is reshaped to down-weight easy examples and thus focus training on hard negatives. A modulating factor (1-*pt*)^ *γ* is added to the cross entropy loss where *γ* is tested from [0,5] in the experiment.

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

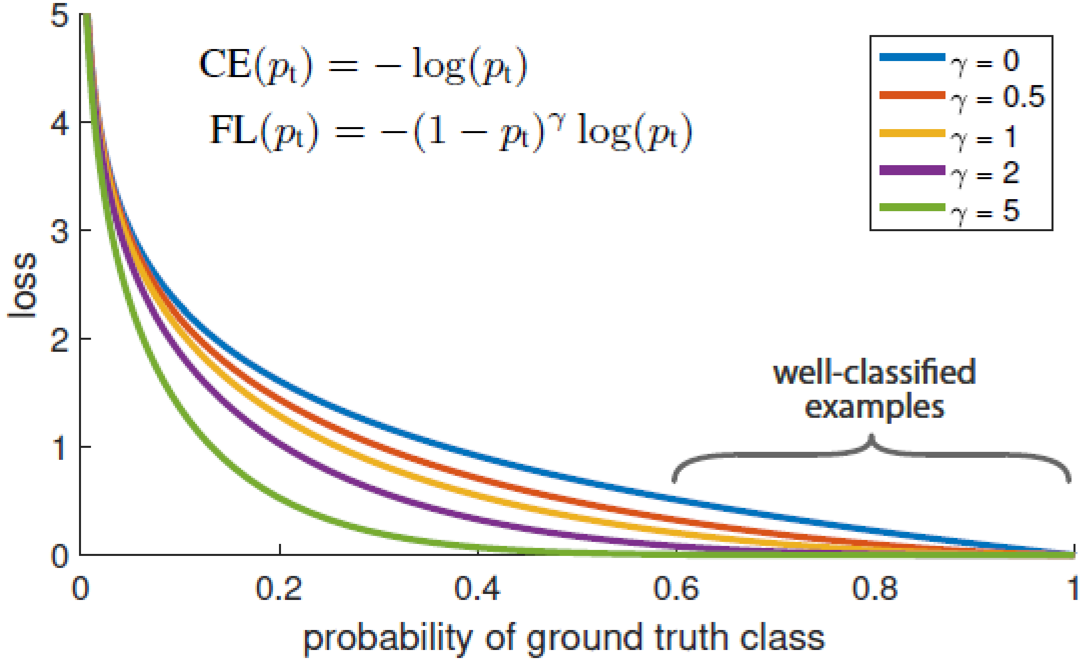
- There are two properties of the FL:

A. When an example is misclassified and pt is small, the modulating factor is near 1 and the loss is unaffected. As *pt* →1, the factor goes to 0 and the loss for well-classified examples is down-weighted.

B. The focusing parameter *γ* smoothly adjusts the rate at which easy examples are down-weighted. When *γ* = 0, FL is equivalent to CE. When *γ* is increased, the effect of the modulating factor is likewise increased. (*γ*=2 works best in experiment.)

Focal Loss

- Intuitively, the modulating factor reduces the loss contribution from easy examples and extends the range in which an example receives low loss.
- For instance, with γ = 2, an example classified with pt = 0.9 would have 100 lower loss compared with CE and with pt = 0.968 it would have 1000 lower loss.
- This in turn increases the importance of correcting misclassified examples.
- The loss is scaled down by at most 4× for pt ≤ 0.5 and γ = 2.

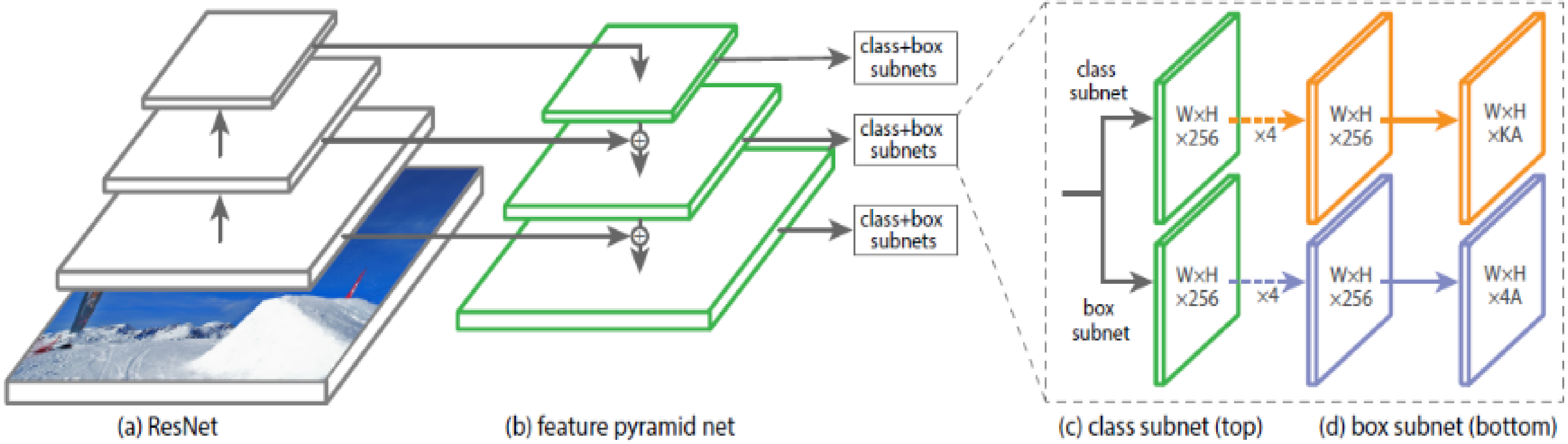


Focal Loss

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

- The above form is used in experiment in practice where *α* is added into the equation, which yields slightly improved accuracy over the one without *α*. And using sigmoid activation function for computing *p* resulting in greater numerical stability.
- *γ*: Focus more on hard examples.
- *α*: Offset class imbalance of number of examples.
- A prior π is set for the value of *p* at the start of training, so that the model's estimated *p* for examples of the rare class is low, e.g. 0.01, in order to improve the training stability in the case of heavy class imbalance.
- It is found that training RetinaNet uses standard CE loss WITHOUT using prior π for initialization leads to network divergence during training and eventually failed.
- And results are insensitive to the exact value of π. And π = 0.01 is used for all experiments.

RetinaNet



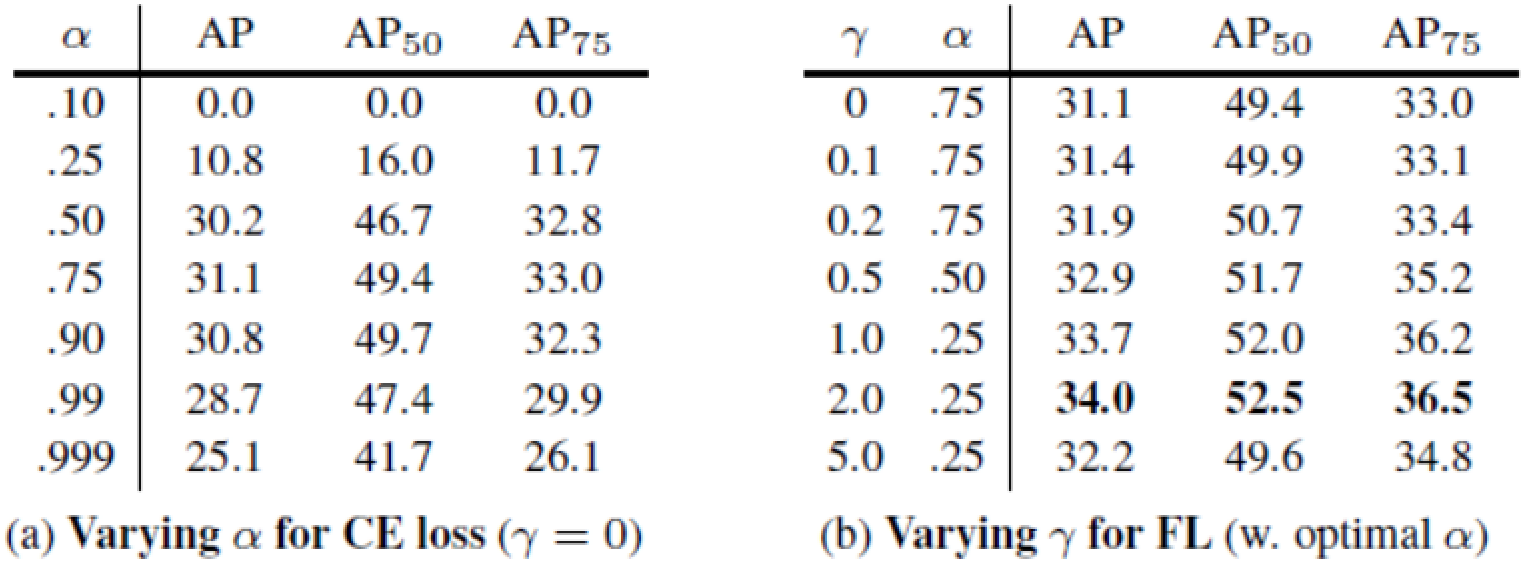
RetinaNet Detector Architecture

RetinaNet

- The one-stage RetinaNet network architecture uses a Feature Pyramid Network (FPN) backbone on top of a feedforward ResNet architecture to generate a rich, multi-scale convolutional feature pyramid.
- To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes and one for regressing from anchor boxes to ground-truth object boxes.
- The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN while running at faster speeds.

α-Balanced CE vs. Focal Loss

- COCO dataset is used. COCO trainval35k split is used for training. And minival (5k) split is used for validation.



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

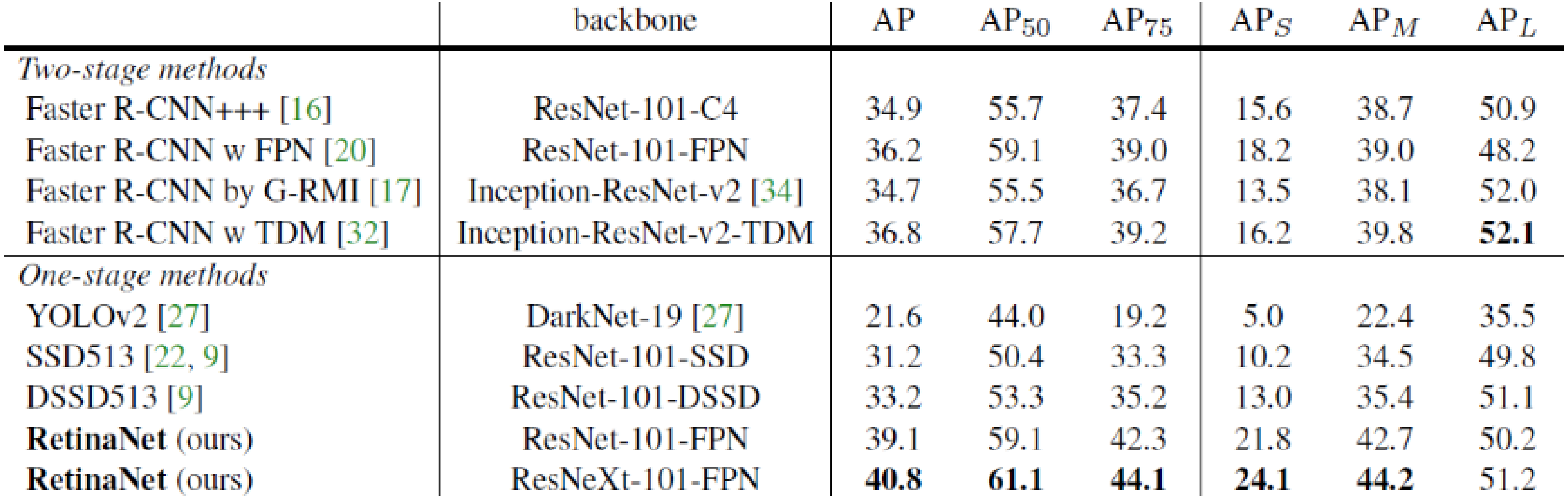
(a) **Varying** α **for CE loss** ($\gamma = 0$)

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)

α for CE loss (Left), γ for FL (Right)

State-of-the-art Accuracy



	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

Object detection single-model results (bounding box AP), vs. state-of-the-art on COCO test-dev

End of Presentation