An Introduction to Modern Object Detection

Gang Yu

yugang@megvii.com

Face++ 旷视

Visual Recognition

A fundamental task in computer vision

- Classification
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Key point Detection
- VQA

…

Face++ 旷视

Category-level Recognition

Category-level Recognition

Instance-level Recognition

Face++ 旷视

Representation

- Bounding-box
  - Face Detection, Human Detection, Vehicle Detection, Text Detection, general Object Detection
- Point
  - Semantic segmentation (Instance Segmentation)
- Keypoint
  - Face landmark
  - Human Keypoint

Face++ 旷视

Outline

- Detection
- Conclusion

Face++ 旷视

Outline

- **Detection**
- Conclusion

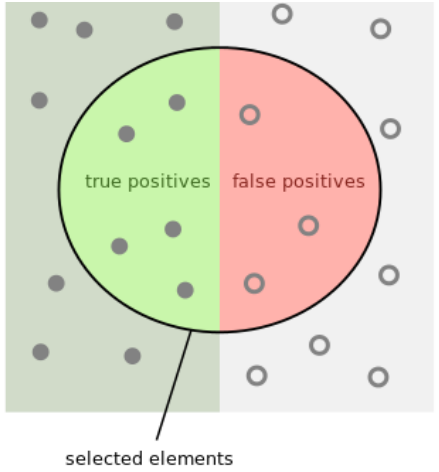
Face++ 旷视

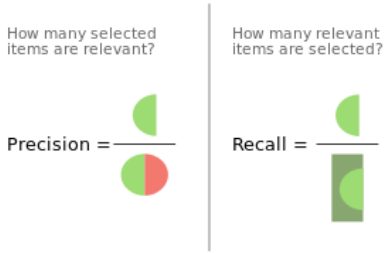
Detection - Evaluation Criteria

Average Precision (AP) and mAP

Precision and recall are single-value metrics based on the whole list of documents returned by the system. For systems that return a ranked sequence of documents, it is desirable to also consider the order in which the returned documents are presented. By computing a precision and recall at every position in the ranked sequence of documents, one can plot a precision-recall curve, plotting precision $p(r)$ as a function of recall r. Average precision computes the average value of $p(r)$ o the interval from $r = 0$ to $r = 1$:[9]

$$\text{AveP} = \int_0^1 p(r)dr$$





Face++ 旷视

Figures are from wikipedia

Detection - Evaluation Criteria

mmAP

```
Average Precision (AP):
  AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small            % AP for small objects: area < 32^2
  AP^medium           % AP for medium objects: 32^2 < area < 96^2
  AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1            % AR given 1 detection per image
  AR^max=10           % AR given 10 detections per image
  AR^max=100          % AR given 100 detections per image
AR Across Scales:
  AR^small            % AR for small objects: area < 32^2
  AR^medium           % AR for medium objects: 32^2 < area < 96^2
  AR^large            % AR for large objects: area > 96^2
```

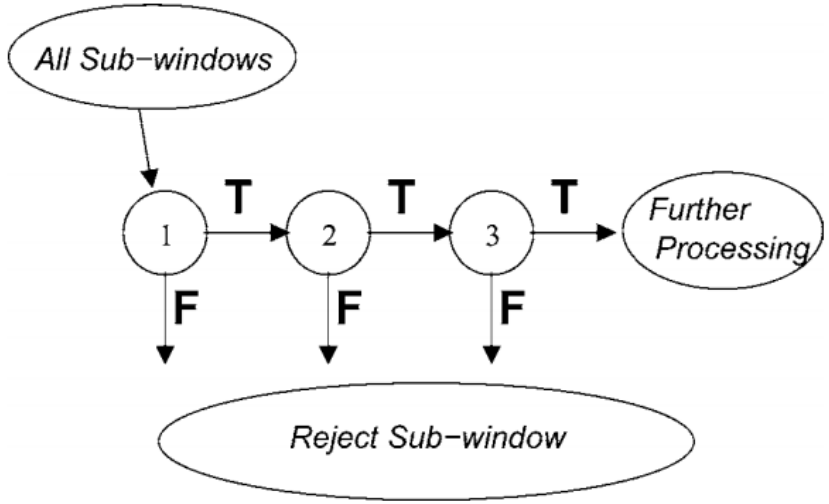
1. Unless otherwise specified, *AP and AR are averaged over multiple Intersection over Union (IoU) values*. Specifically we use 10 IoU thresholds of .50:.05:.95. This is a break from tradition, where AP is computed at a single IoU of .50 (which corresponds to our metric $AP^{IoU=.50}$). Averaging over IoUs rewards detectors with better localization.
2. AP is averaged over all categories. Traditionally, this is called "mean average precision" (mAP). We make no distinction between AP and mAP (and likewise AR and mAR) and assume the difference is clear from context.
3. AP (averaged across all 10 IoU thresholds and all 80 categories) will determine the challenge winner. This should be considered the single most important metric when considering performance on COCO.
4. In COCO, there are more small objects than large objects. Specifically: approximately 41% of objects are small (area < 32^2), 34% are medium (32^2 < area < 96^2), and 24% are large (area > 96^2). Area is measured as the number of pixels in the segmentation mask.
5. AR is the maximum recall given a fixed number of detections per image, averaged over categories and IoUs. AR is related to the metric of the same name used in proposal evaluation but is computed on a per-category basis.
6. All metrics are computed allowing for at most 100 top-scoring detections per image (across all categories).
7. The evaluation metrics for detection with bounding boxes and segmentation masks are identical in all respects except for the IoU computation (which is performed over boxes or masks, respectively).

Face++ 旷视

Figures are from http://cocodataset.org

How to perform a detection?

- Sliding window: enumerate all the windows (up to millions of windows)
  - VJ detector: cascade chain
- Fully Convolutional network
  - shared computation



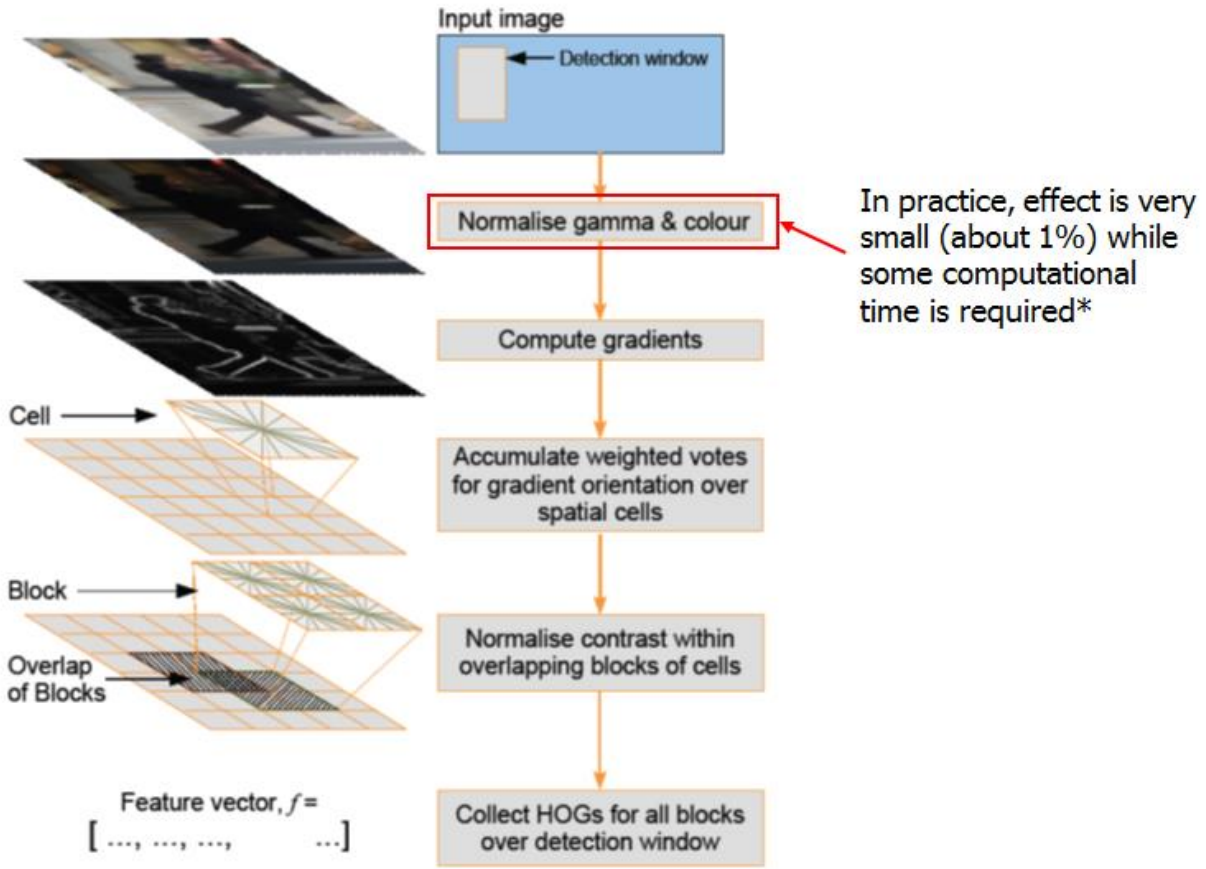
Face++ 旷视

Robust Real-time Object Detection; Viola, Jones; IJCV 2001
http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

General Detection Before Deep Learning

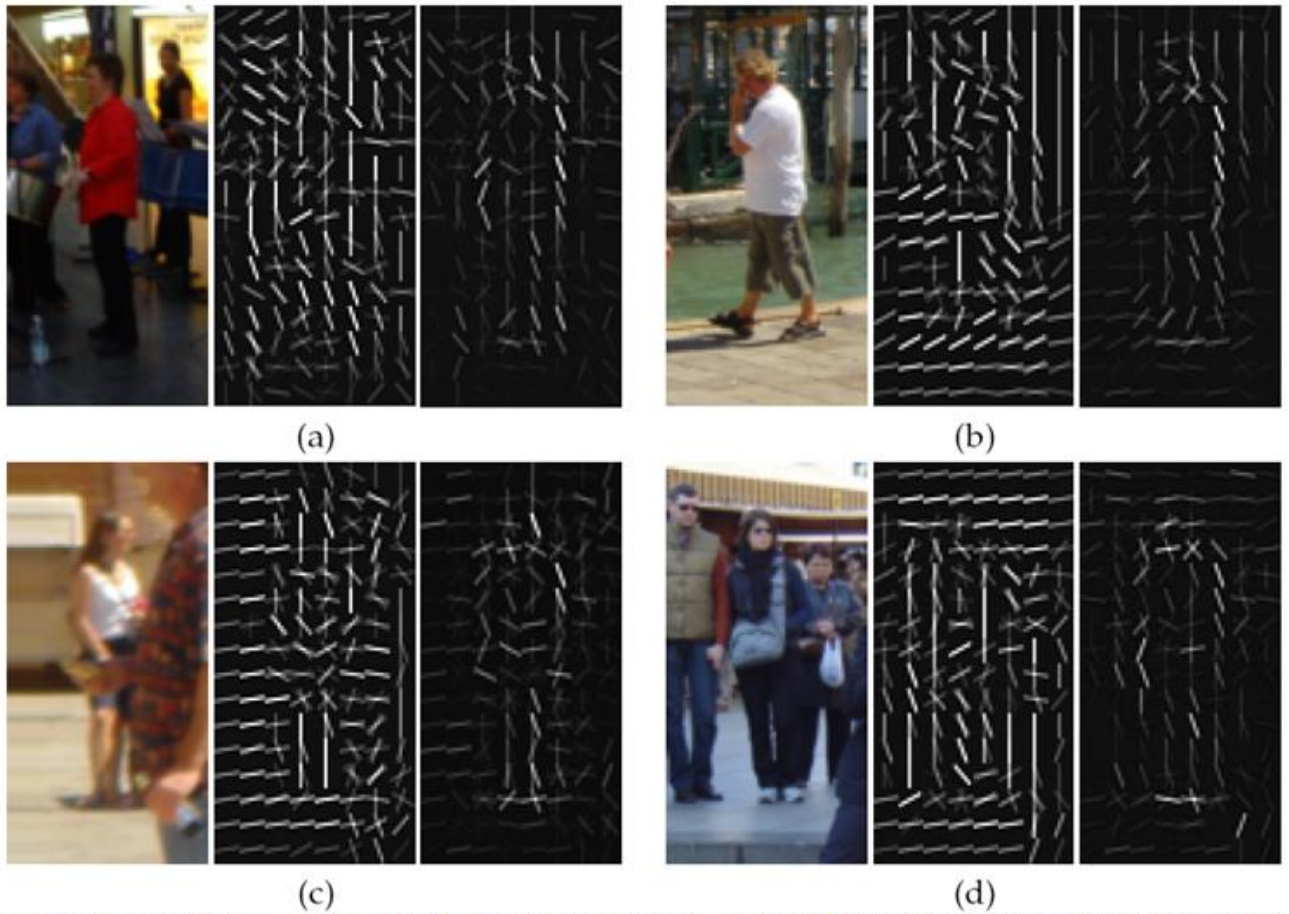
- Feature + classifier
- Feature
  - Haar Feature
  - HOG (Histogram of Gradient)
  - LBP (Local Binary Pattern)
  - ACF (Aggregated Channel Feature)
  - …
- Classifier
  - SVM
  - Bootsing
  - Random Forest

Traditional Hand-crafted Feature: HoG



*Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, SanDiego, USA, June 2005. Vol. II, pp. 886-893.

Traditional Hand-crafted Feature: HoG



In each triplet: (1) the input image, (2) the corresponding R-HOG feature vector (only the dominant orientation of each cell is shown), (3) the dominant orientations selected by the SVM (obtained by multiplying the feature vector by the corresponding weights from the linear SVM).

General Detection Before Deep Learning

Traditional Methods

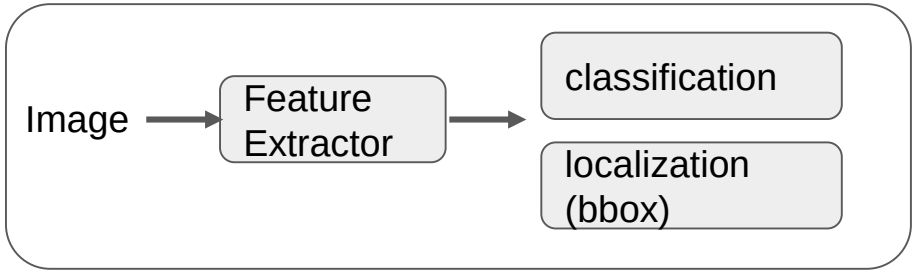
- Pros
  - Efficient to compute (e.g., HAAR, ACF) on CPU
  - Easy to debug, analyze the bad cases
  - reasonable performance on limited training data
- Cons
  - Limited performance on large dataset
  - Hard to be accelerated by GPU

Deep Learning for Object Detection

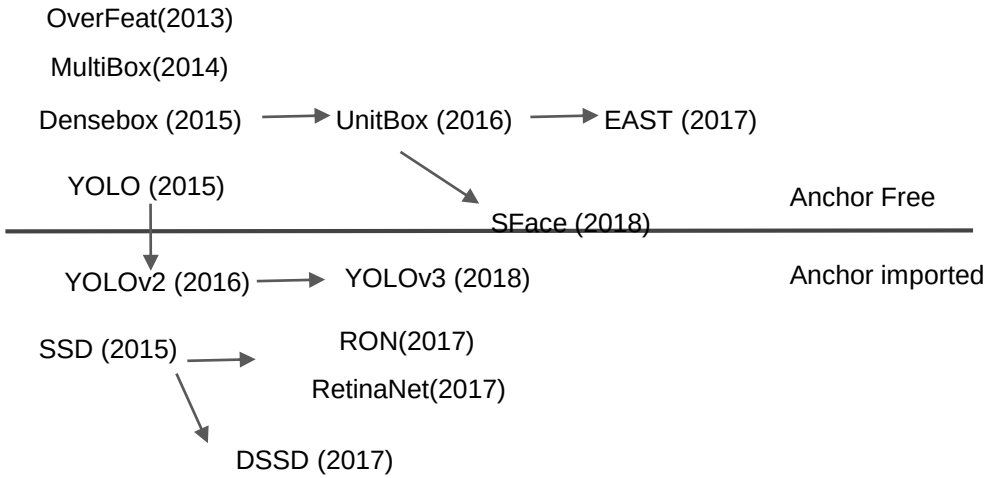
Based on the whether following the “proposal and refine”

- One Stage
  - Example: Densebox, YOLO (YOLO v2), SSD, Retina Net
  - Keyword: Anchor, Divide and conquer, loss sampling
- Two Stage
  - Example: RCNN (Fast RCNN, Faster RCNN), RFCN, FPN, MaskRCNN
  - Keyword: speed, performance

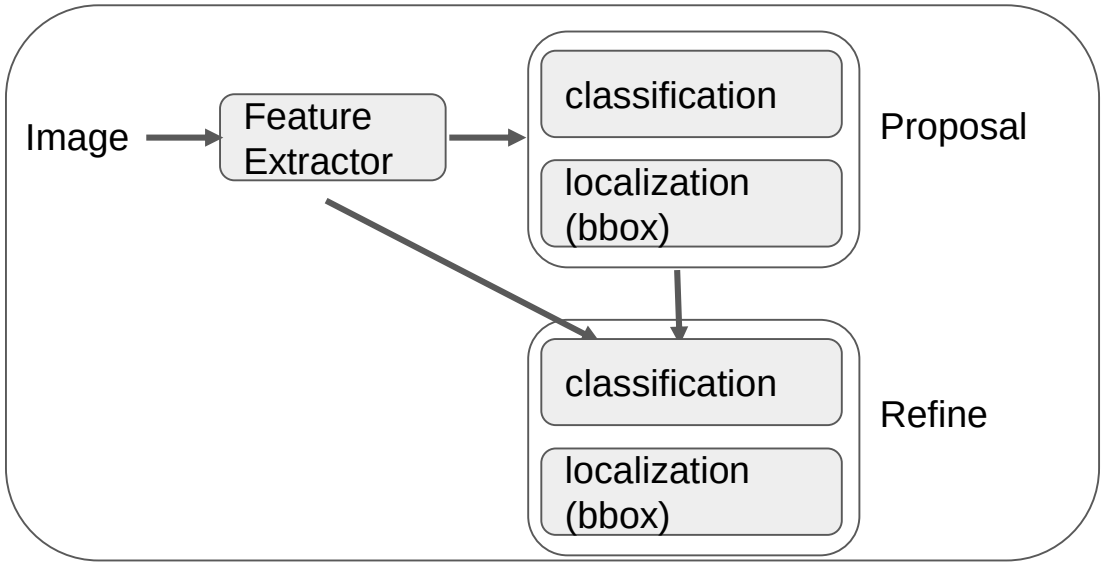
A bit of History

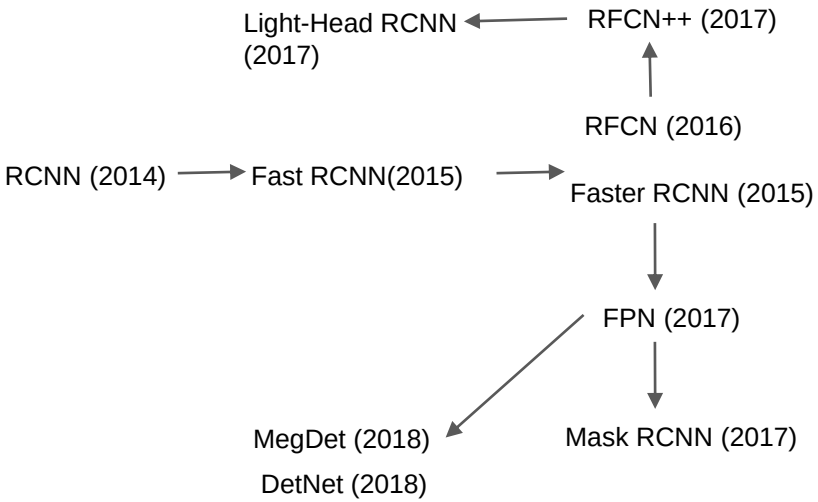


One stage detector



two stages detector





Face++ 旷视

One Stage Detector: Densebox

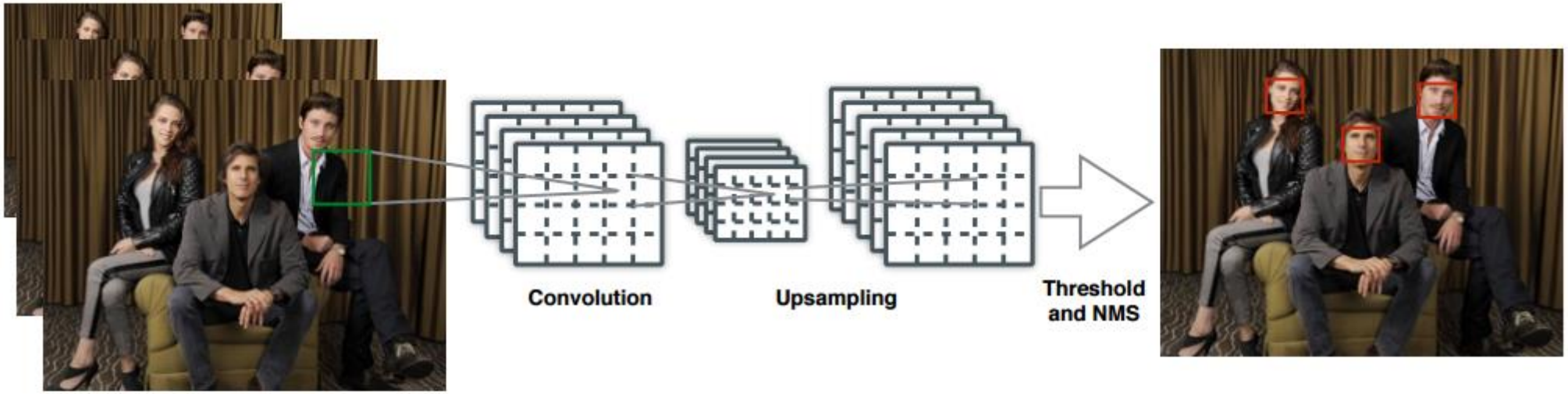


Figure 1: **The DenseBox Detection Pipeline.** 1) Image pyramid is fed to the network. 2) After several layers of convolution and pooling, upsampling feature map back and apply convolution layers to get final output. 3) Convert output feature map to bounding boxes , and apply non-maximum suppression to all bounding boxes over the threshold.

DenseBox: Unifying Landmark Localization with End to End Object Detection, Huang etc, 2015

https://arxiv.org/abs/1509.04874

One Stage Detector: Densebox

- No Anchor: GT Assignment
  - A sub-circle in the GT is labeled as positive
    - fail when two GT highly overlaps
    - the size of the sub-circle matters
    - more attention (loss) will be placed to large faces
- Loss sampling
  - All pos/negative positions will be used to compute the cls loss

One Stage Detector: Densebox

Problems

- L2 loss is not robust to scale variation (UnitBox)
  - learnt features are not robust
- GT assignment issue (SSD)
  - Fail to handle the crowd case
- relatively large localization error (Two stages detector)
- more false positive (FP) (Two stages detector)
  - does not obviously kill the fp

One Stage Detector: Densebox -> UnitBox

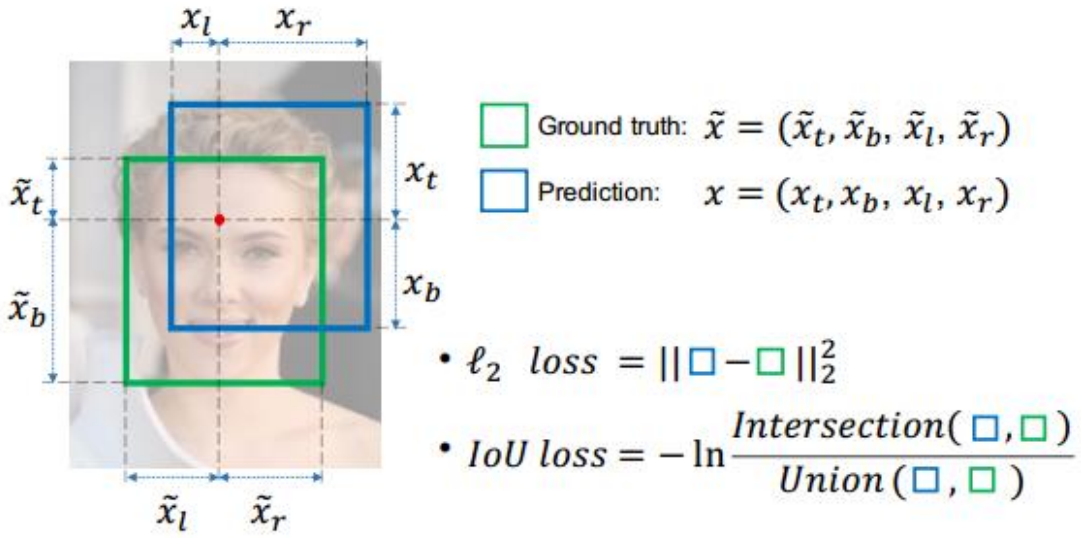


Figure 1: Illustration of IoU loss and ℓ_2 loss for pixel-wise bounding box prediction.

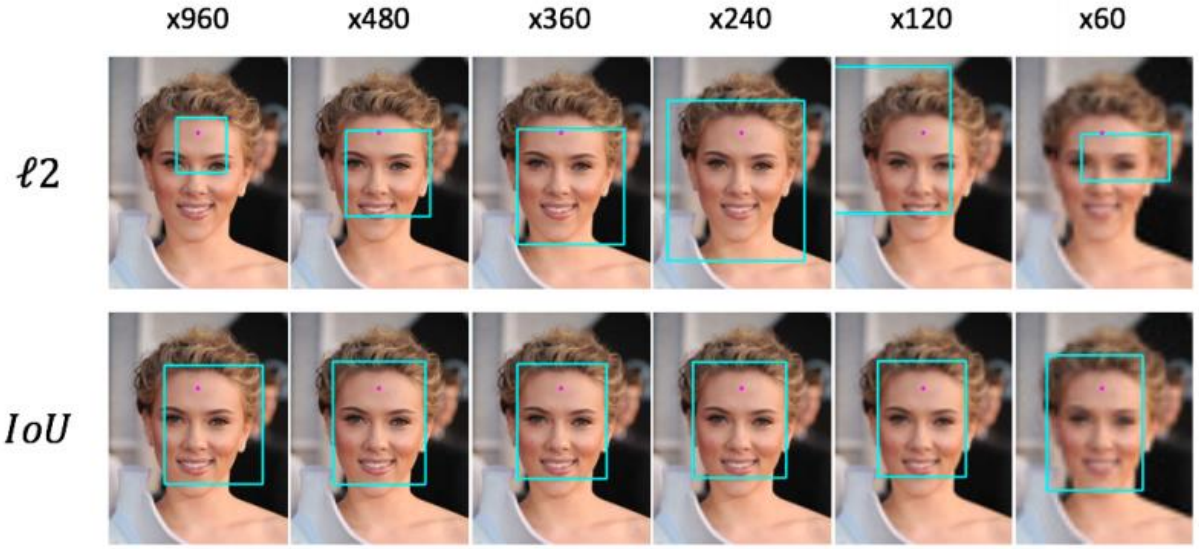
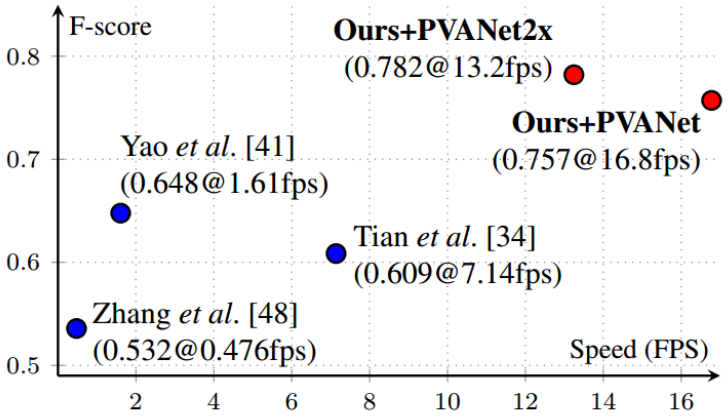


Figure 5: Compared to ℓ_2 loss, the IoU loss is much more robust to scale variations for bounding box prediction.

Face++ 旷视

UnitBox: An Advanced Object Detection Network, Yu etc, 2016
http://cn.arxiv.org/pdf/1608.01471.pdf

One Stage Detector: Densebox -> UnitBox->EAST



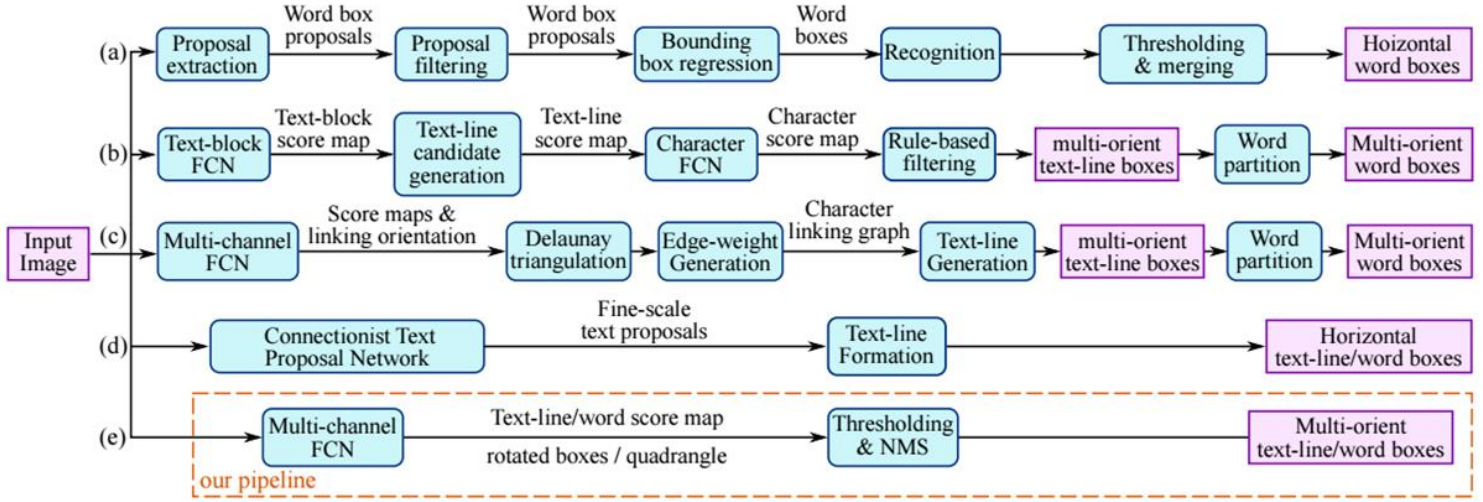


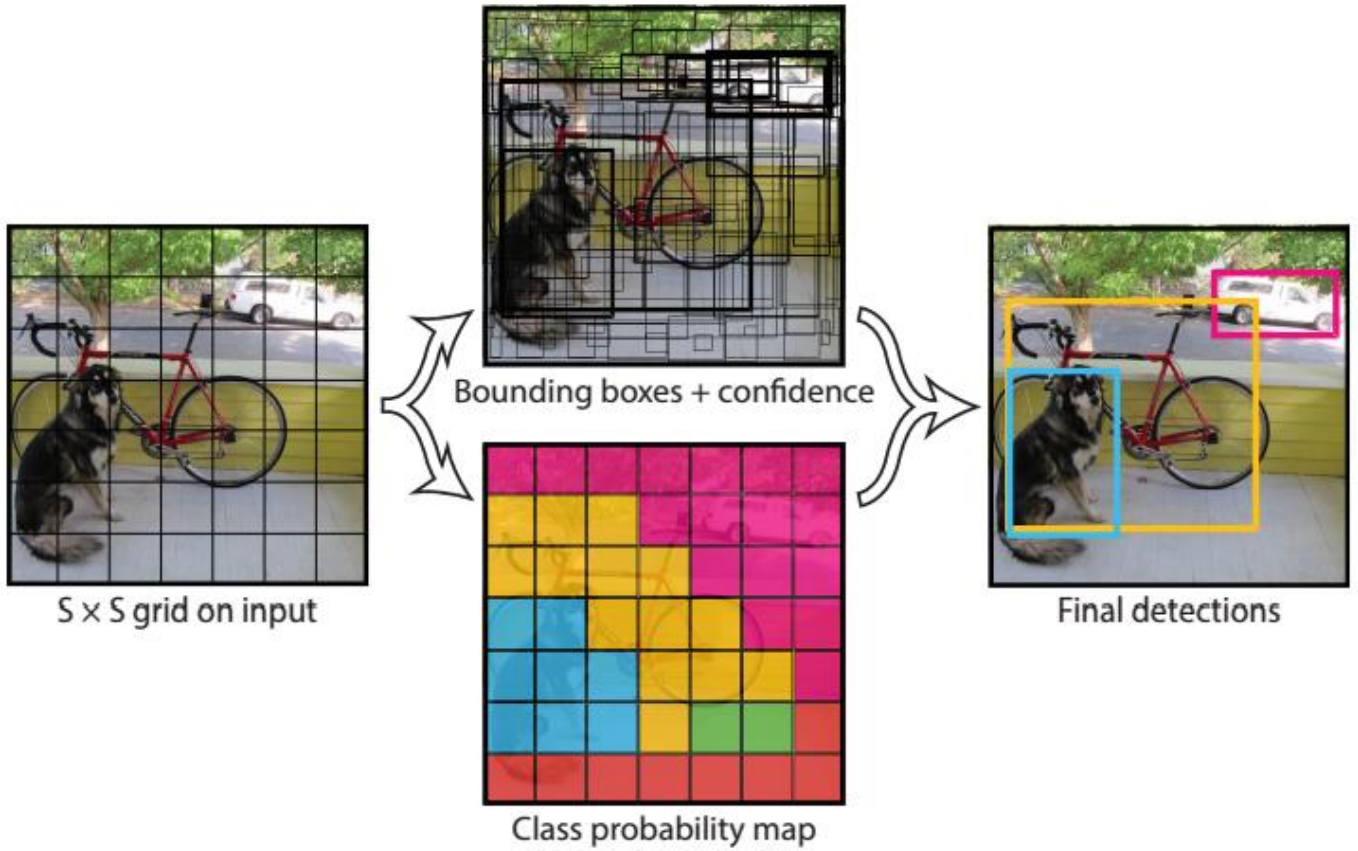
Figure 2. Comparison of pipelines of several recent works on scene text detection: (a) Horizontal word detection and recognition pipeline proposed by Jaderberg *et al.* [12]; (b) Multi-orient text detection pipeline proposed by Zhang *et al.* [48]; (c) Multi-orient text detection pipeline proposed by Yao *et al.* [41]; (d) Horizontal text detection using CTPN, proposed by Tian *et al.* [34]; (e) Our pipeline, which eliminates most intermediate steps, consists of only two stages and is much simpler than previous solutions.

Face++ 旷视

EAST: An Efficient and Accurate Scene Text Detector, Zhou etc, CVPR 2017

https://arxiv.org/abs/1704.03155

One Stage Detector: YOLO



You Only Look Once: Unified, Real-Time Object Detection, Redmon etc, CVPR 2016

https://arxiv.org/abs/1506.02640

One Stage Detector: YOLO

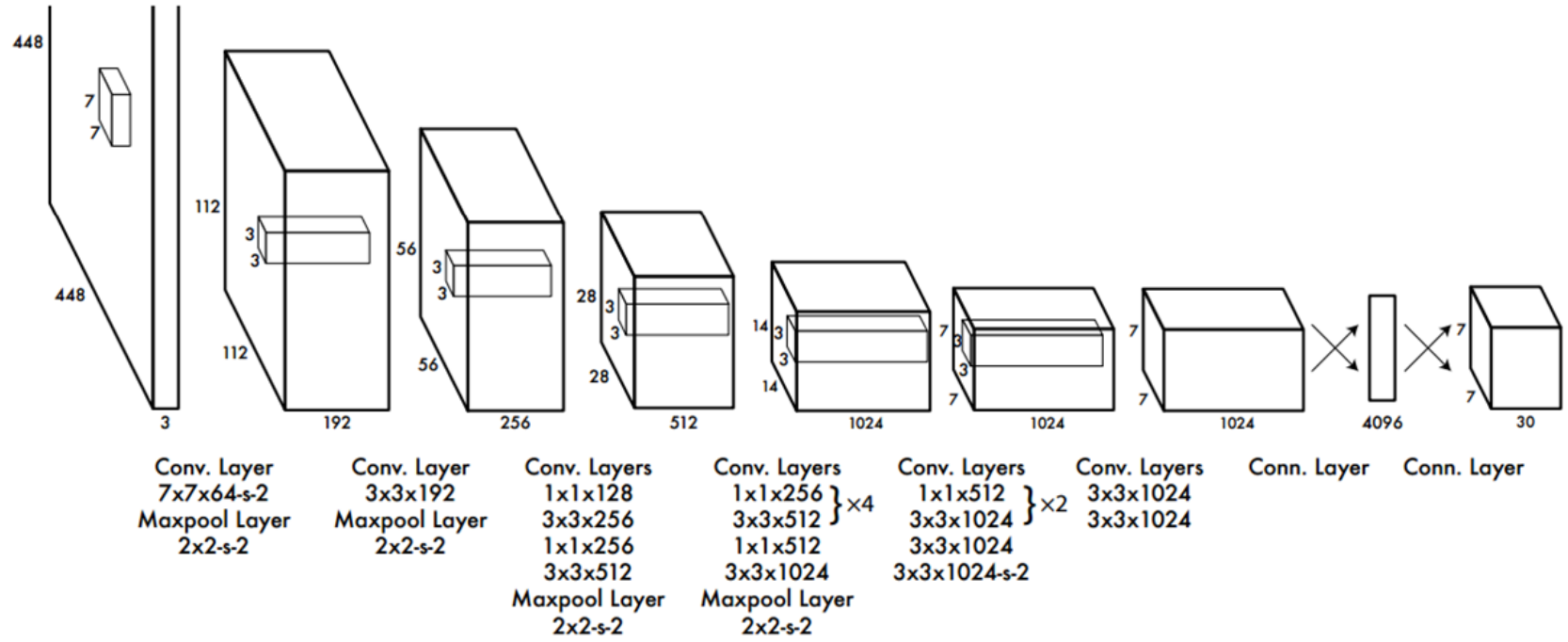


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Face++ 旷视

You Only Look Once: Unified, Real-Time Object Detection, Redmon etc, CVPR 2016

https://arxiv.org/abs/1506.02640

One Stage Detector: YOLO

- No Anchor
  - GT assignment is based on the cells (7x7)
- Loss sampling
  - all pos/neg predictions are evaluated (but more sparse than densebox)

You Only Look Once: Unified, Real-Time Object Detection, Redmon etc, CVPR 2016

https://arxiv.org/abs/1506.02640

One Stage Detector: YOLO

Discussion

- fc reshape (4096-> 7x7x30)
  - more context
  - but not fully convolutional
- One cell can output up to two boxes in one category
  - fail to work on the crowd case
- Fast speed
  - small imagenet base model
  - small input size (448x448)

You Only Look Once: Unified, Real-Time Object Detection, Redmon etc, CVPR 2016

https://arxiv.org/abs/1506.02640

One Stage Detector: YOLO

Experiments on general detection

Method	VOC 2007 test	VOC 2012 test	COCO	time
YOLO	57.9/NA	52.7/63.4	NA	fps: 45/155

You Only Look Once: Unified, Real-Time Object Detection, Redmon etc, CVPR 2016

https://arxiv.org/abs/1506.02640

One Stage Detector: YOLO -> YOLOv2

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

Face++ 旷视

YOLO9000: Better, Faster, Stronger Redmon etc, CVPR 2016

https://arxiv.org/abs/1612.08242

One Stage Detector: YOLO -> YOLOv2

Experiments:

Method	VOC 2007 test	VOC 2012 test	COCO	time
YOLO	52.7/63.4	57.9/NA	NA	fps: 45/155
YOLOv2	78.6	73.4	21.6	fps: 40

Face++ 旷视

YOLO9000: Better, Faster, Stronger Redmon etc, CVPR 2016

https://arxiv.org/abs/1612.08242

One Stage Detector: YOLO -> YOLOv2

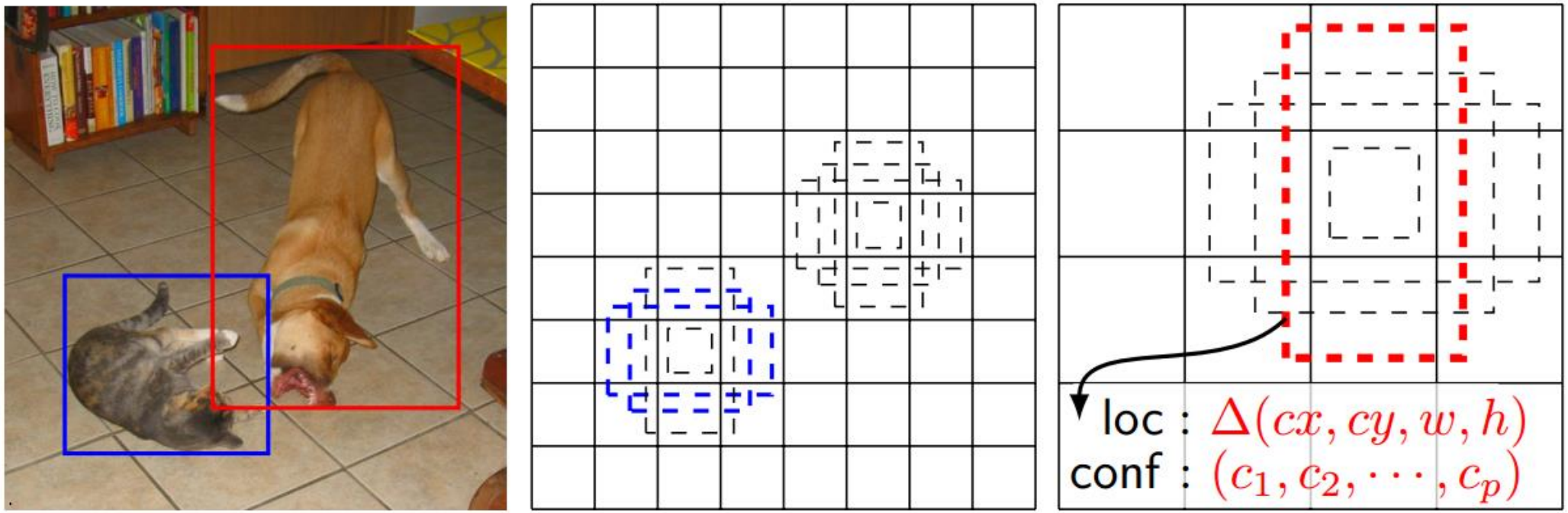
Video demo: https://pjreddie.com/darknet/yolo/

Face++ 旷视

YOLO9000: Better, Faster, Stronger Redmon etc, CVPR 2016

https://arxiv.org/abs/1612.08242

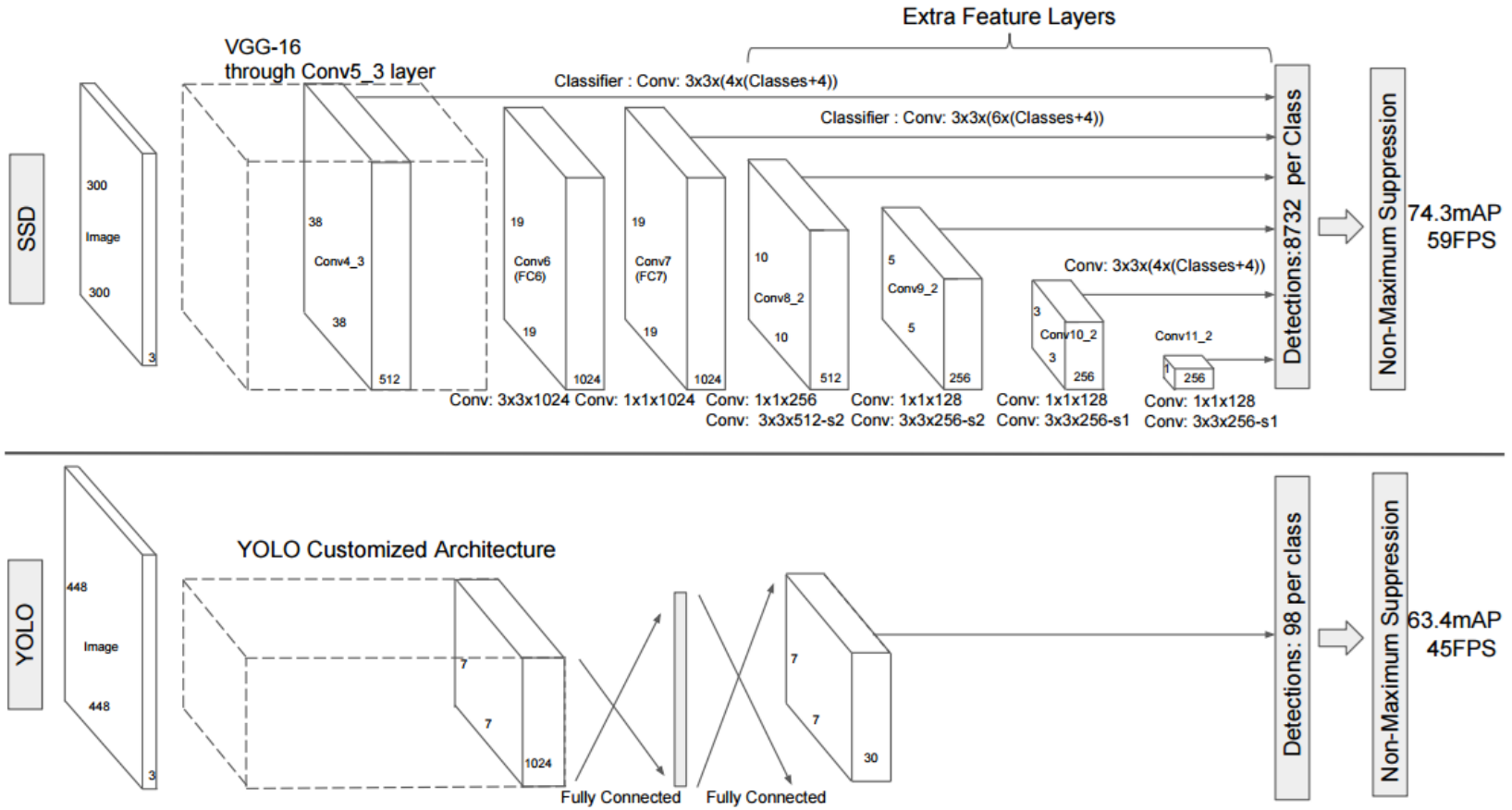
One Stage Detector: SSD



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

SSD: Single Shot MultiBox Detector, Liu etc
https://arxiv.org/pdf/1512.02325.pdf

One Stage Detector: SSD



SSD: Single Shot MultiBox Detector, Liu etc, ECCV 2016
https://arxiv.org/pdf/1512.02325.pdf

One Stage Detector: SSD

- Anchor
  - GT-anchor assignment
    - GT is predicted by one best matched (IOU) anchor or matched with an anchor with IOU > 0.5
      - better recall
    - dense or sparse anchor?
- Divide and Conquer
  - Different layers handle the objects with different scales
    - Assume small objects can be predicted in earlier layers (not very strong semantics)
- Loss sampling
  - OHEM: negative positions are sampled (not balanced pos/neg ratio)
  - negative:pos is at most 3:1

Face++ 旷视

SSD: Single Shot MultiBox Detector, Liu etc, ECCV 2016
https://arxiv.org/pdf/1512.02325.pdf

One Stage Detector: SSD

Discussion:

- Assume small objects can be predicted in earlier layers (not very strong semantics) (DSSD, RON, RetinaNet)
- strong data augmentation
- VGG model (Replace by resnet in DSSD)
  - cannot be easily adapted to other models
  - a lot of hacks
- A long tail (Large computation)

SSD: Single Shot MultiBox Detector, Liu etc, ECCV 2016
https://arxiv.org/pdf/1512.02325.pdf

One Stage Detector: SSD

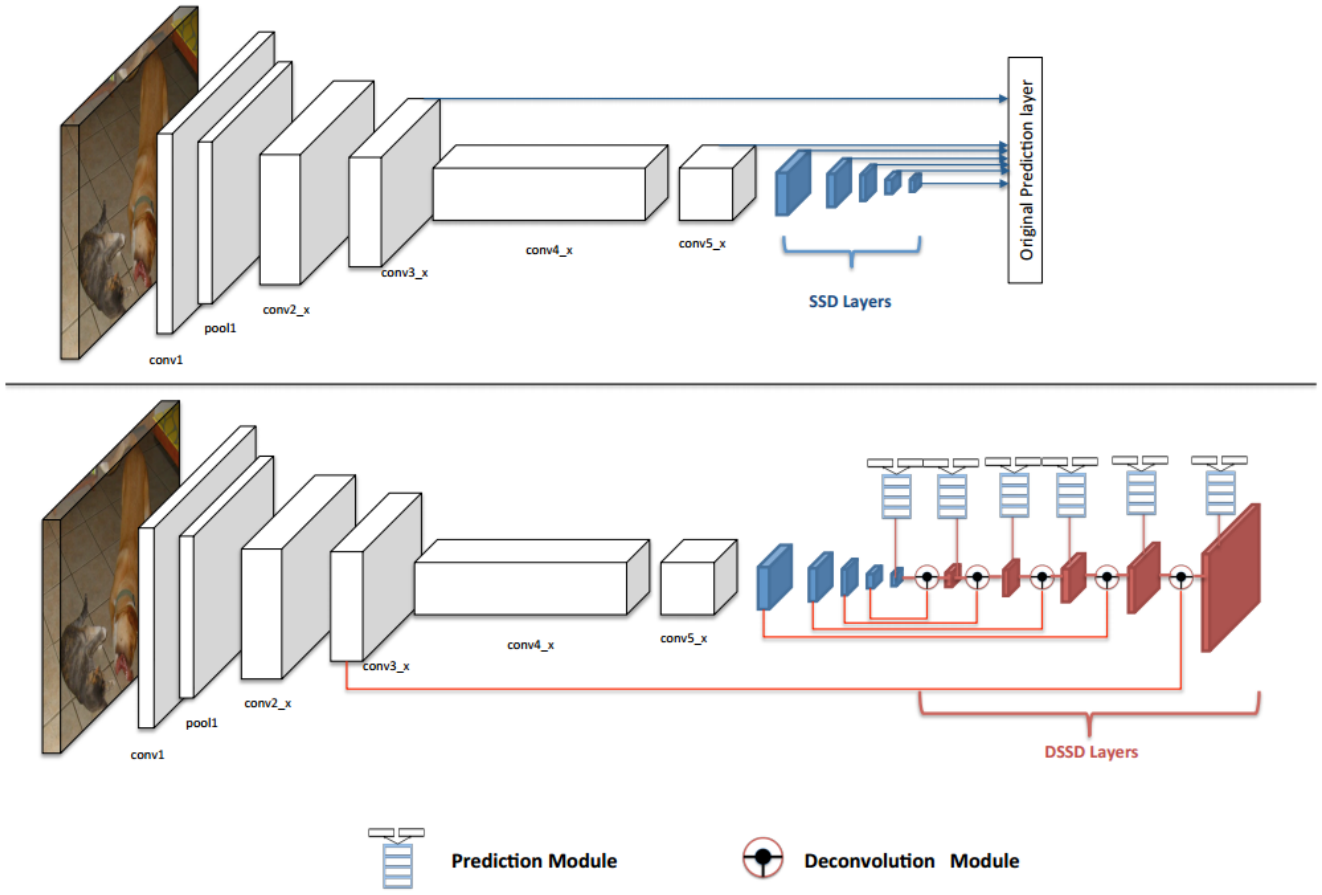
Experiments

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19

Face++ 旷视

SSD: Single Shot MultiBox Detector, Liu etc, ECCV 2016
https://arxiv.org/pdf/1512.02325.pdf

One Stage Detector: SSD -> DSSD



DSSD : Deconvolutional Single Shot Detector, Fu etc 2017, https://arxiv.org/abs/1701.06659

One Stage Detector: DSSD

Experiments

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5

Face++ 旷视

DSSD : Deconvolutional Single Shot Detector, Fu etc 2017,
https://arxiv.org/abs/1701.06659

One Stage Detector: SSD -> RON

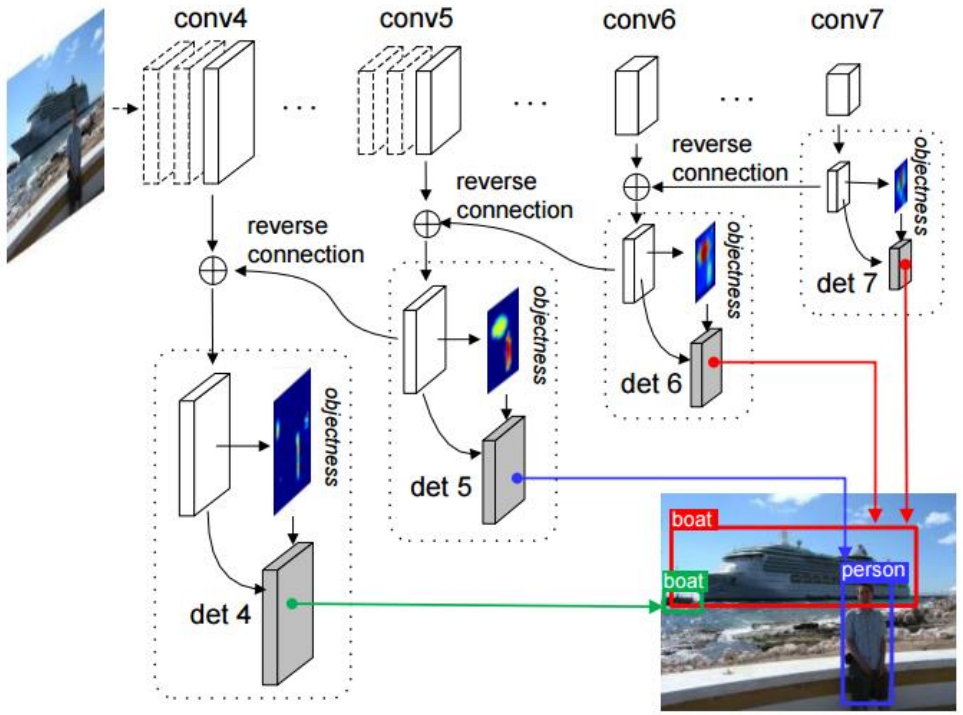


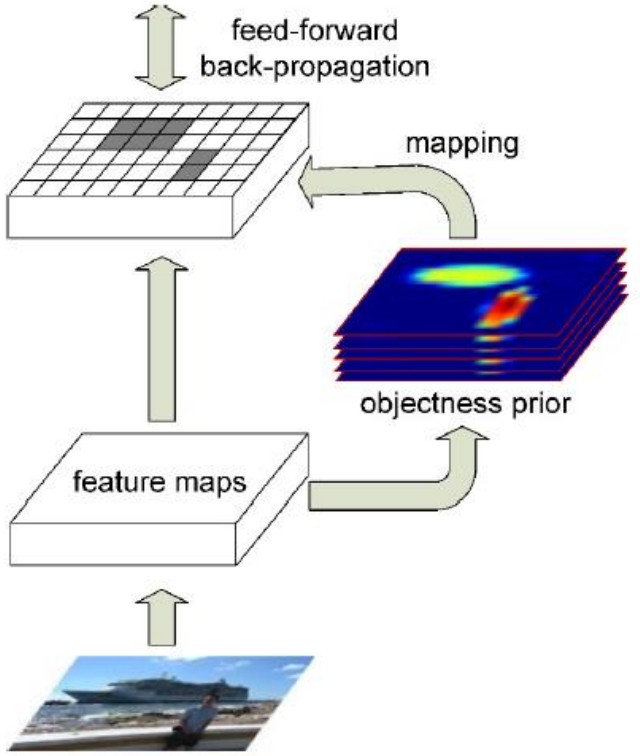
Figure 2. RON object detection overview. Given an input image, the network firstly computes features of the backbone network. Then at each detection scale: (a) adds reverse connection; (b) generates objectness prior; (c) detects object on its corresponding CNN scales and locations. Finally, all detection results are fused and selected with non-maximum suppression.

Face++ 旷视

RON: Reverse Connection with Objectness Prior Networks for Object Detection, Kong etc, CVPR 2017
https://arxiv.org/pdf/1707.01691.pdf

One Stage Detector: RON

- Anchor
- Divide and conquer
  - Reverse Connect (similar to FPN)
- Loss Sampling
  - Objectness prior
    - pos/neg unbalanced issue
    - split to 1) binary cls 2) multi-class cls



RON: Reverse Connection with Objectness Prior Networks for Object Detection, Kong etc, CVPR 2017
https://arxiv.org/pdf/1707.01691.pdf

One Stage Detector: RON

Experiments

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15

Face++ 旷视

RON: Reverse Connection with Objectness Prior Networks for Object Detection, Kong etc, CVPR 2017
https://arxiv.org/pdf/1707.01691.pdf

One Stage Detector: SSD -> RetinaNet

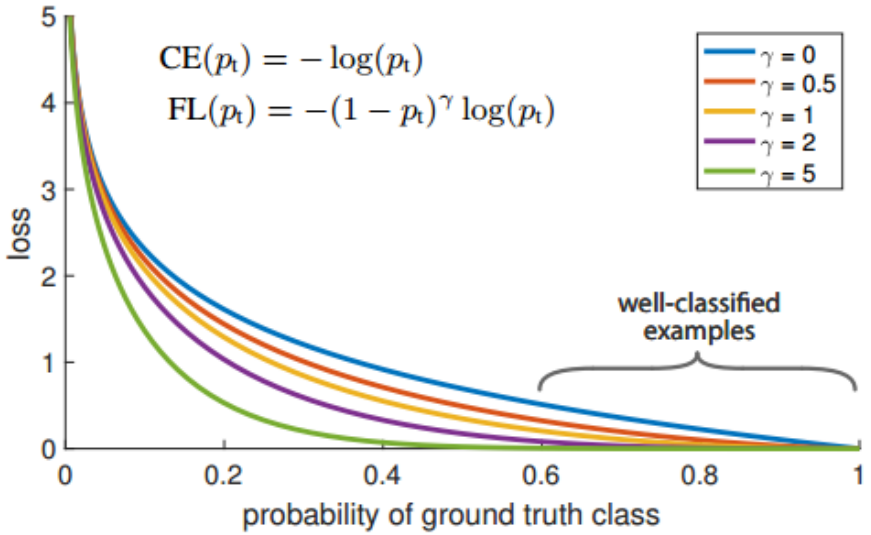
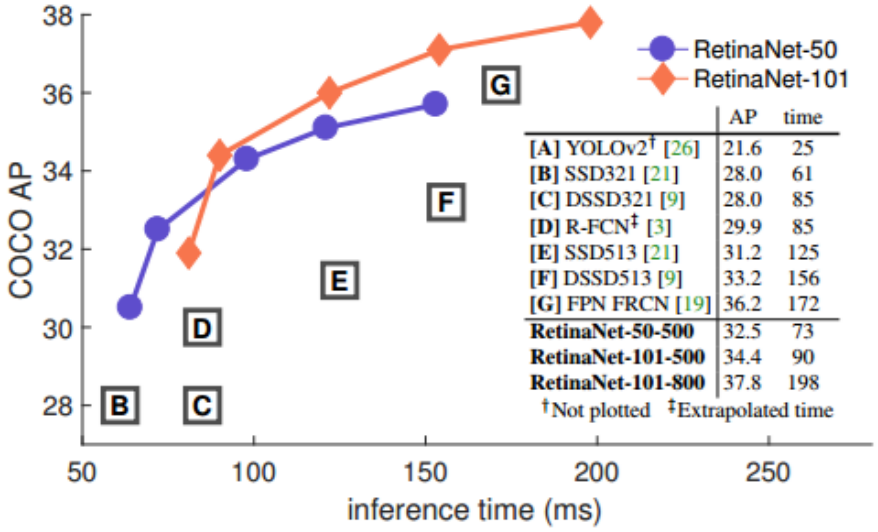


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.



	AP	time
[A] YOLOv2† [26]	21.6	25
[B] SSD321 [21]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [21]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [19]	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

Figure 2. Speed (ms) versus accuracy (AP) on COCO `test-dev`. Enabled by the focal loss, our simple one-stage *RetinaNet* detector outperforms all previous one-stage and two-stage detectors, including the best reported Faster R-CNN [27] system from [19]. We show variants of RetinaNet with ResNet-50-FPN (blue circles) and ResNet-101-FPN (orange diamonds) at five scales (400-800 pixels). Ignoring the low-accuracy regime (AP<25), RetinaNet forms an upper envelope of all current detectors, and a variant trained for longer (not shown) achieves 39.1 AP. Details are given in §5.

Face++ 旷视

Focal Loss for Dense Object Detection, Lin etc, ICCV 2017
https://arxiv.org/pdf/1708.02002.pdf

One Stage Detector: SSD -> RetinaNet

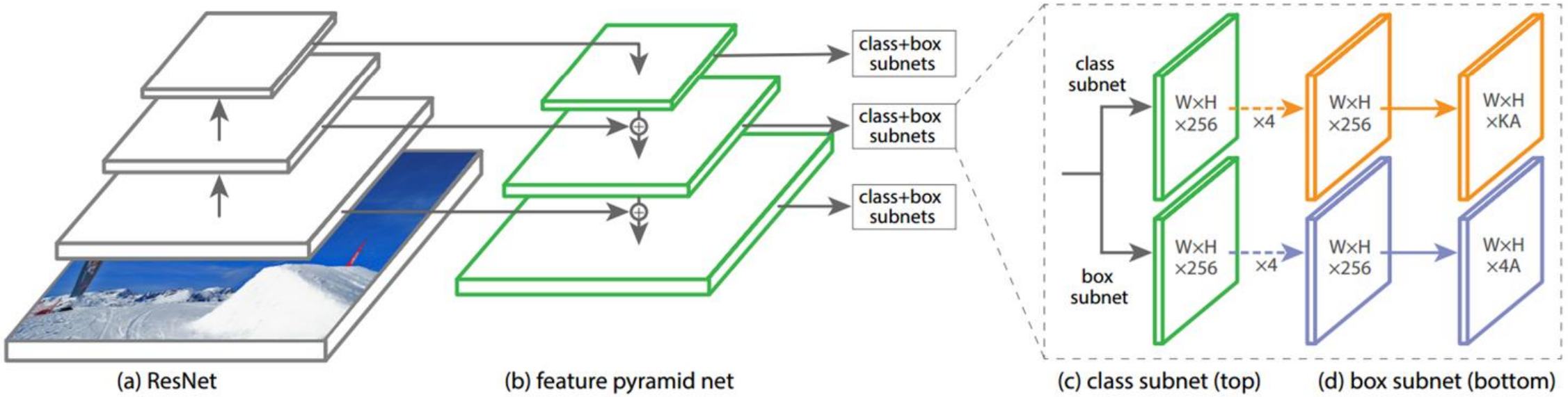


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [19] backbone on top of a feedforward ResNet architecture [15] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). 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 [19] while running at faster speeds.

Face++ 旷视

Focal Loss for Dense Object Detection, Lin etc, ICCV 2017
https://arxiv.org/pdf/1708.02002.pdf

One Stage Detector: RetinaNet

- Anchor
- Divide and Conquer
  - FPN
- Loss Sampling
  - Focal loss
    - pos/neg unbalanced issue
    - new setting (e.g., more anchor)

Face++ 旷视

Focal Loss for Dense Object Detection, Lin etc, ICCV 2017
https://arxiv.org/pdf/1708.02002.pdf

One Stage Detector: RetinaNet

Experiments

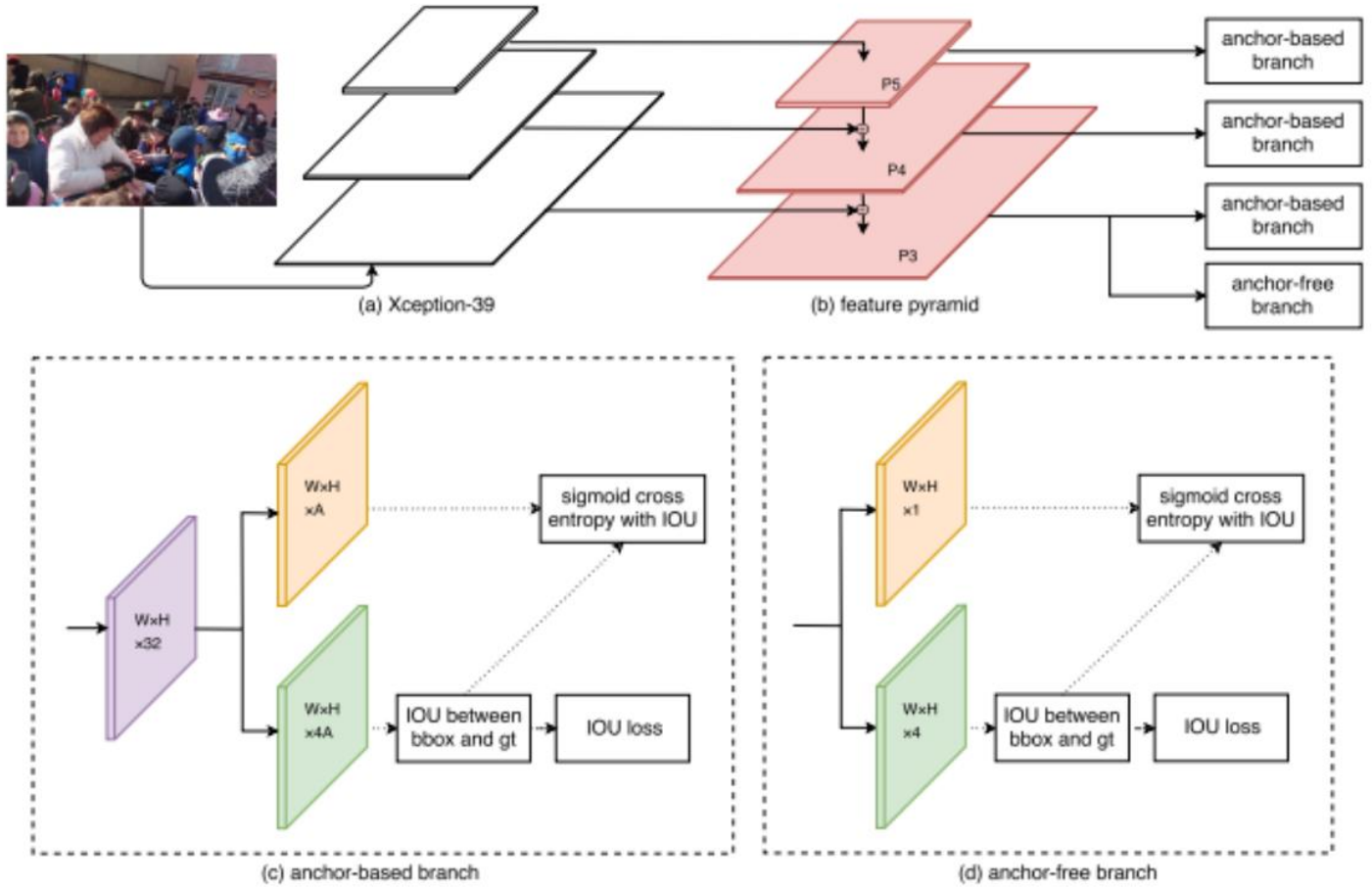
Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5

Face++ 旷视

Focal Loss for Dense Object Detection, Lin etc, ICCV 2017
https://arxiv.org/pdf/1708.02002.pdf

One Stage Detector: SFace

- Integrate Anchor-free and Anchor-based idea to address the scale issue in face detection



SFace: An Efficient Network for Face Detection in Large Scale Variations
Jianfeng Wang, Ye Yuan, Boxun Li, Gang Yu, Sun Jian
https://arxiv.org/pdf/1804.06559.pdf

Face++ 旷视

One Stage Detector: SFace

- Standard face sizes:
  - Anchor based solution
  - Good performance
- Too small/Large faces:
  - Anchor-free based solution
  - Flexible, Fast speed for inference

Face++ 旷视

SFace: An Efficient Network for Face Detection in Large Scale Variations
Jianfeng Wang, Ye Yuan, Boxun Li, Gang Yu, Sun Jian
https://arxiv.org/pdf/1804.06559.pdf

One Stage Detector: SFace

BaseNet	P3-P5 layer	re-score	Anchor-based Branch	Anchor-free Branch	AP (easy)	AP (medium)	AP (hard)
RetinaNet					**92.6**	**91.2**	65.0
RetinaNet(multi-scale)					90.7	90.3	75.2
RetinaNet	✓				43.8	64.9	74.7
UnitBox					70.6	76.0	67.8
SFace	✓		✓		43.5	64.4	73.7
SFace	✓			✓	71.6	78.1	73.7
SFace	✓		✓	✓	71.6	78.1	73.8
SFace	✓	✓	✓		39.5	62.4	72.9
SFace	✓	✓		✓	90.0	88.8	78.8
SFace	✓	✓	✓	✓	89.8	88.7	**80.7**

Table 3. The ablation study of SFace on the WIDER FACE validation dataset.

Min size	1080	1200	1500	2160
Time	12.46ms	14.30ms	21.53ms	41.13ms
AP (WIDER FACE hard)	76.7	78.4	80.7	78.8

Face++ 旷视

SFace: An Efficient Network for Face Detection in Large Scale Variations
Jianfeng Wang, Ye Yuan, Boxun Li, Gang Yu, Sun Jian
https://arxiv.org/pdf/1804.06559.pdf

One Stage Detector: Summary

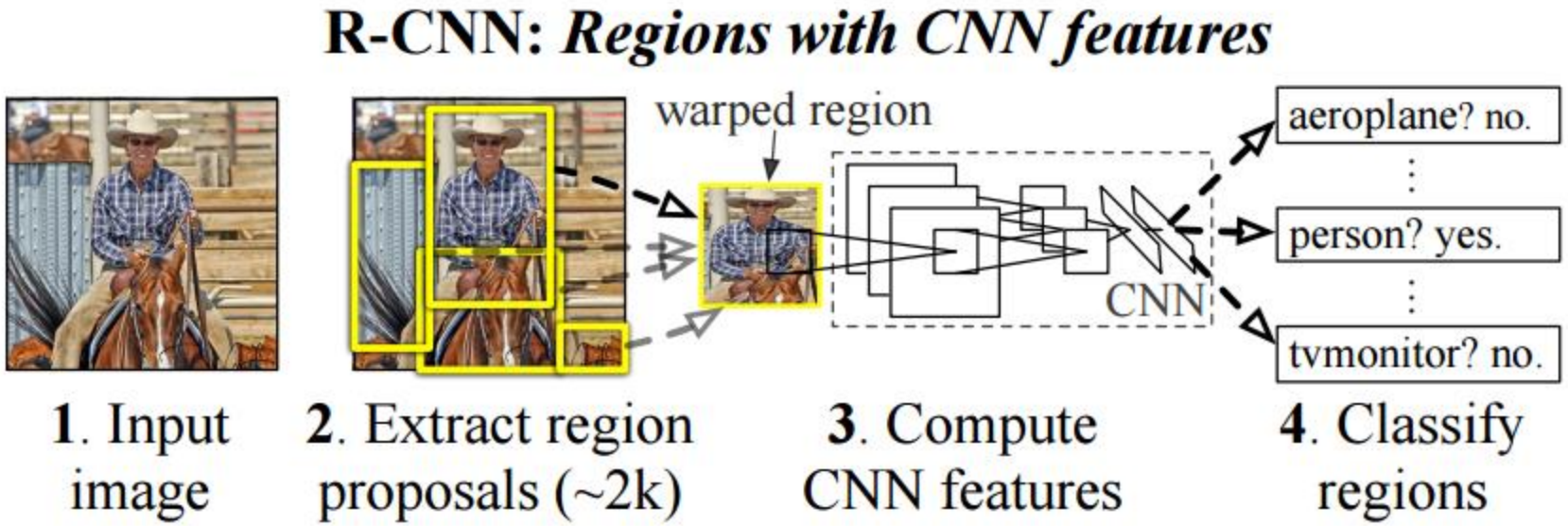
- Anchor
  - No anchor: YOLO, densebox/unitbox/east
  - Anchor: YOLOv2, SSD, DSSD, RON, RetinaNet
- Divide and conquer
  - SSD, DSSD, RON, RetinaNet
- loss sample
  - all sample: densebox
  - OHEM: SSD
  - focal loss: RetinaNet

One Stage Detector: Discussion

Anchor (YOLO v2, SSD, RetinaNet) or Without Anchor (Densebox, YOLO)

- Model Complexity
  - Difference on the extremely small model (< 30M flops on 224x224 input)
- Sampling
- Application
  - No Anchor: Face
  - With Anchor: Human, General Detection
- Problem for one stage detector
  - Unbalanced pos/neg data
  - Pool localization precision

Two Stages Detector: RCNN



Face++ 旷视

Rich feature hierarchies for accurate object detection and semantic segmentation, Girshirk etc, CVPR 2014
https://arxiv.org/pdf/1311.2524.pdf

Two Stages Detector: RCNN

Discussion

- Extremely slow speed
  - selective search proposal (CPU)/warp
- not end-to-end optimized
- Good for small objects

Face++ 旷视

Rich feature hierarchies for accurate object detection and semantic segmentation, Girshirk etc, CVPR 2014
https://arxiv.org/pdf/1311.2524.pdf

Two Stages Detector: RCNN

Experiments

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5
RCNN	66	NA	NA	47s

Face++ 旷视

Rich feature hierarchies for accurate object detection and semantic segmentation, Girshirk etc, CVPR 2014
https://arxiv.org/pdf/1311.2524.pdf

Two Stages Detector: RCNN -> Fast RCNN

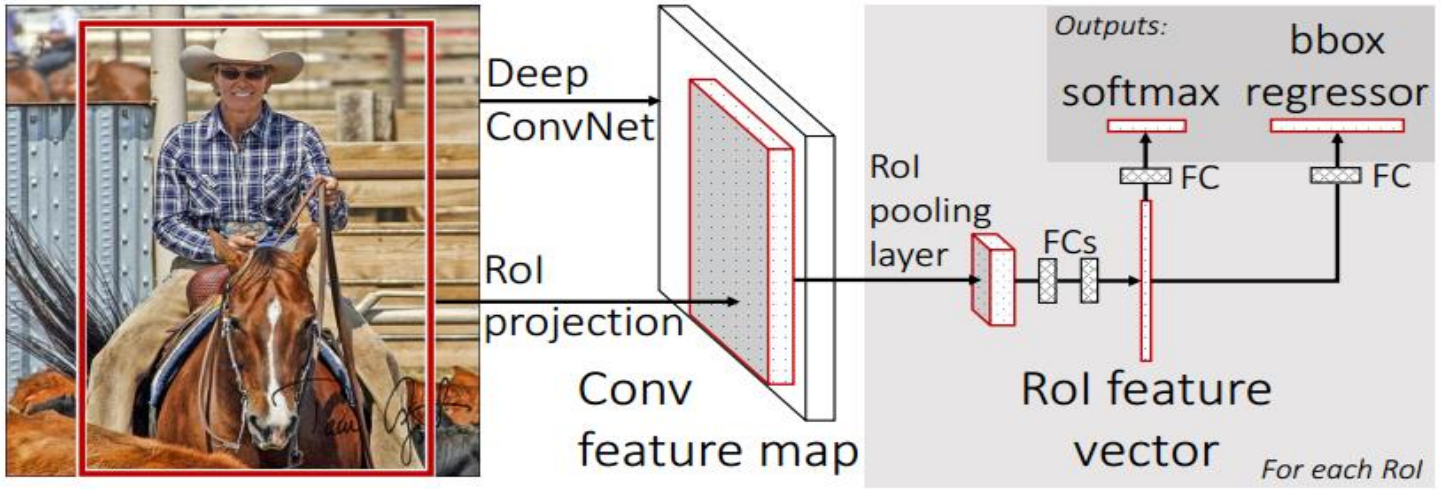


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Face++ 旷视

Fast R-CNN, Girshick etc, ICCV 2015
https://arxiv.org/pdf/1504.08083.pdf

Two Stages Detector: Fast RCNN

Discussion

- slow speed
  - selective search proposal (CPU)
- not end-to-end optimized
- ROI pooling
  - alignment issue
  - sampling
  - aspect ratio changes

Face++ 旷视

Fast R-CNN, Girshick etc, ICCV 2015
https://arxiv.org/pdf/1504.08083.pdf

Two Stages Detector: Fast RCNN

Experiments

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5
RCNN	66	NA	NA	47s
Fast RCNN	77.0	82.3 (wth coco data)	NA	0.5s

Face++ 旷视

Fast R-CNN, Girshick etc, ICCV 2015
https://arxiv.org/pdf/1504.08083.pdf

Two Stages Detector:
RCNN -> Fast RCNN -> FasterRCNN

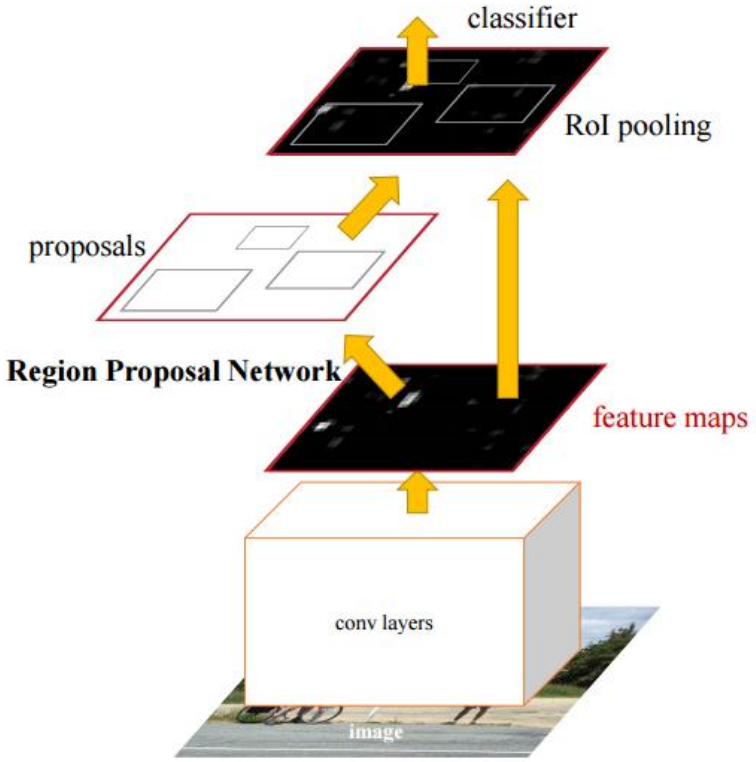


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Face++ 旷视

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Ren etc, CVPR 2016
https://arxiv.org/pdf/1506.01497.pdf

Two Stages Detector: Faster RCNN

Discussion

- speed
  - selective search proposal (CPU) -> RPN
- alternative optimization/end-to-end optimization
- Recall issue due to two stages detector

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Ren etc, CVPR 2016
https://arxiv.org/pdf/1506.01497.pdf

Two Stages Detector: Faster RCNN

Experiments

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5
RCNN	66	NA	NA	47s
Fast RCNN	77.0	82.3 (wth coco data)	NA	0.5s
Faster RCNN	73.2	70.4	NA	5

Face++ 旷视

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Ren etc, CVPR 2016
https://arxiv.org/pdf/1506.01497.pdf

Two Stages Detector:
RCNN -> Fast RCNN -> FasterRCNN -> RFCN

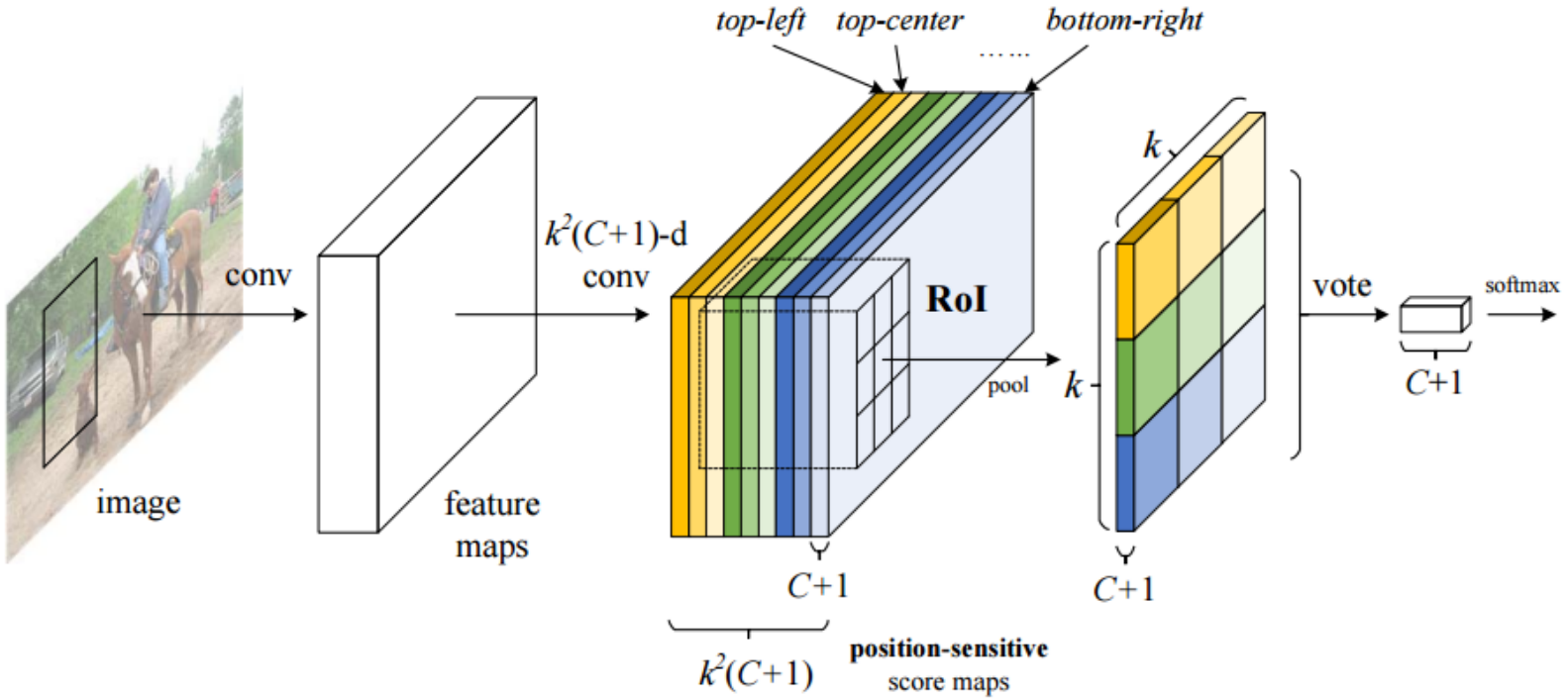


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

Face++ 旷视

R-FCN: Object Detection via Region-based Fully Convolutional Networks, Dai etc, NIPS 2016, https://arxiv.org/pdf/1605.06409.pdf

Two Stages Detector: RFCN

Discussion

- Share convolution
  - fasterRCNN: shared Res1-4 (RPN), not shared Res5 (RCNN)
  - RFCN: shared Res1-5 (both RPN and RCNN)
- PSPooling
  - a large number of channels:(7x7xC)xWxH
  - Problems in ROIPooling also exist
- Fully connected vs Convolution
  - fc: global context
  - conv: can be shared but the context is relative small
  - trade-off: large kernel

Face++ 旷视

R-FCN: Object Detection via Region-based Fully Convolutional Networks, Dai etc, NIPS 2016, https://arxiv.org/pdf/1605.06409.pdf

Two Stages Detector: RFCN

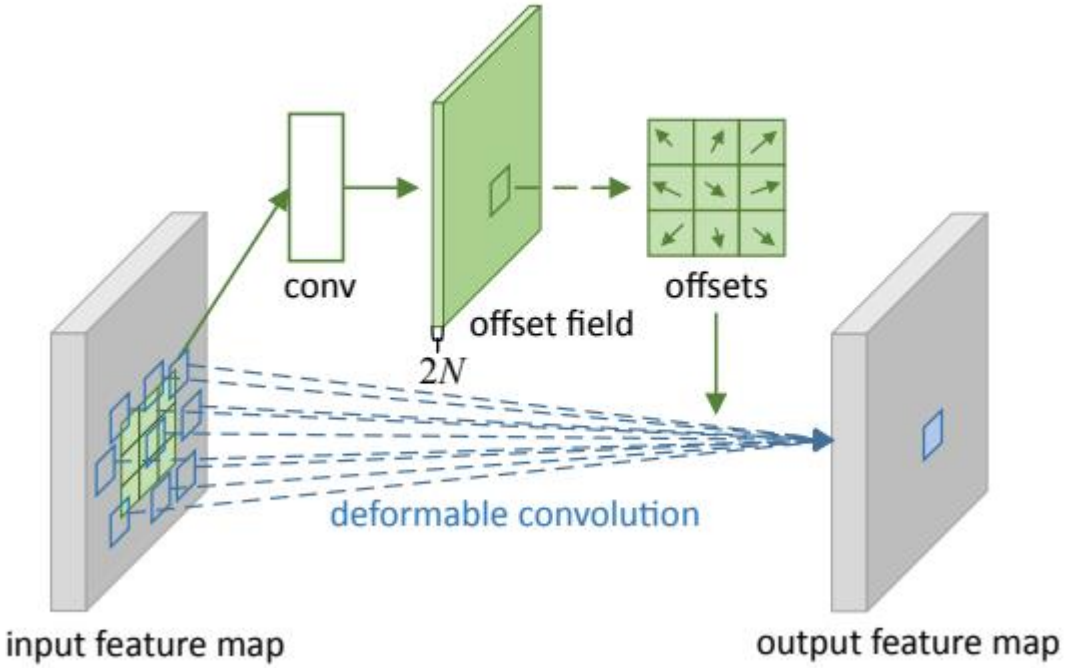
Experiments

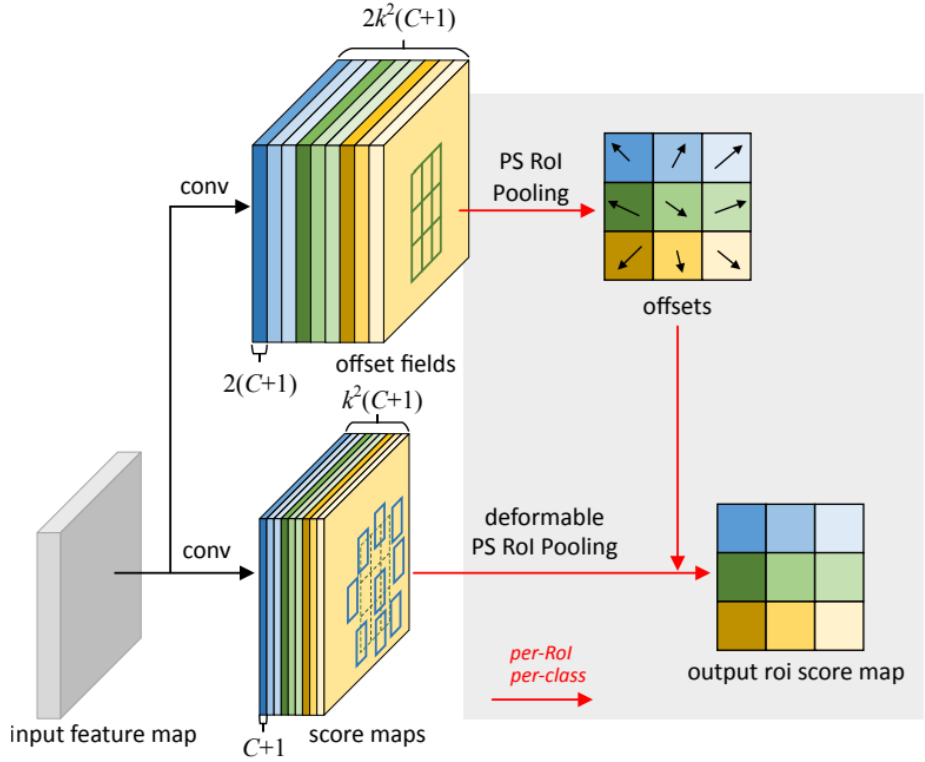
Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5
RCNN	66	NA	NA	47s
Fast RCNN	77.0	82.3 (wth coco data)	NA	0.5s
Faster RCNN	73.2	70.4	NA	200ms
RFCN	79.5	77.6	29.9	170ms

Face++ 旷视

R-FCN: Object Detection via Region-based Fully Convolutional Networks, Dai etc, NIPS 2016, https://arxiv.org/pdf/1605.06409.pdf

Two Stages Detector:
RFCN -> Deformable Convolutional Networks





Face++ 旷视

Deformable Convolutional Networks, Dai etc, ICCV 2017
https://arxiv.org/abs/1703.06211

Two Stages Detector:
RFCN -> Deformable Convolutional Networks

Figure 6: Each image triplet shows the sampling locations ($9^3 = 729$ red points in each image) in three levels of 3×3 deformable filters (see Figure 5 as a reference) for three activation units (green points) on the background (left), a small object (middle), and a large object (right), respectively.

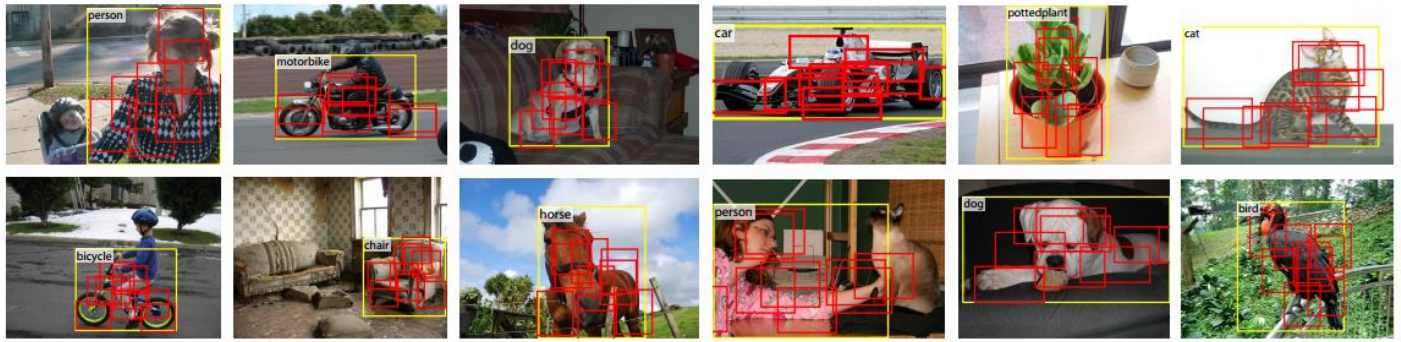


Figure 7: Illustration of offset parts in deformable (positive sensitive) RoI pooling in R-FCN [7] and 3×3 bins (red) for an input RoI (yellow). Note how the parts are offset to cover the non-rigid objects.

Face++ 旷视

Deformable Convolutional Networks, Dai etc, ICCV 2017
https://arxiv.org/abs/1703.06211

Two Stages Detector:
RFCN -> Deformable Convolutional Networks

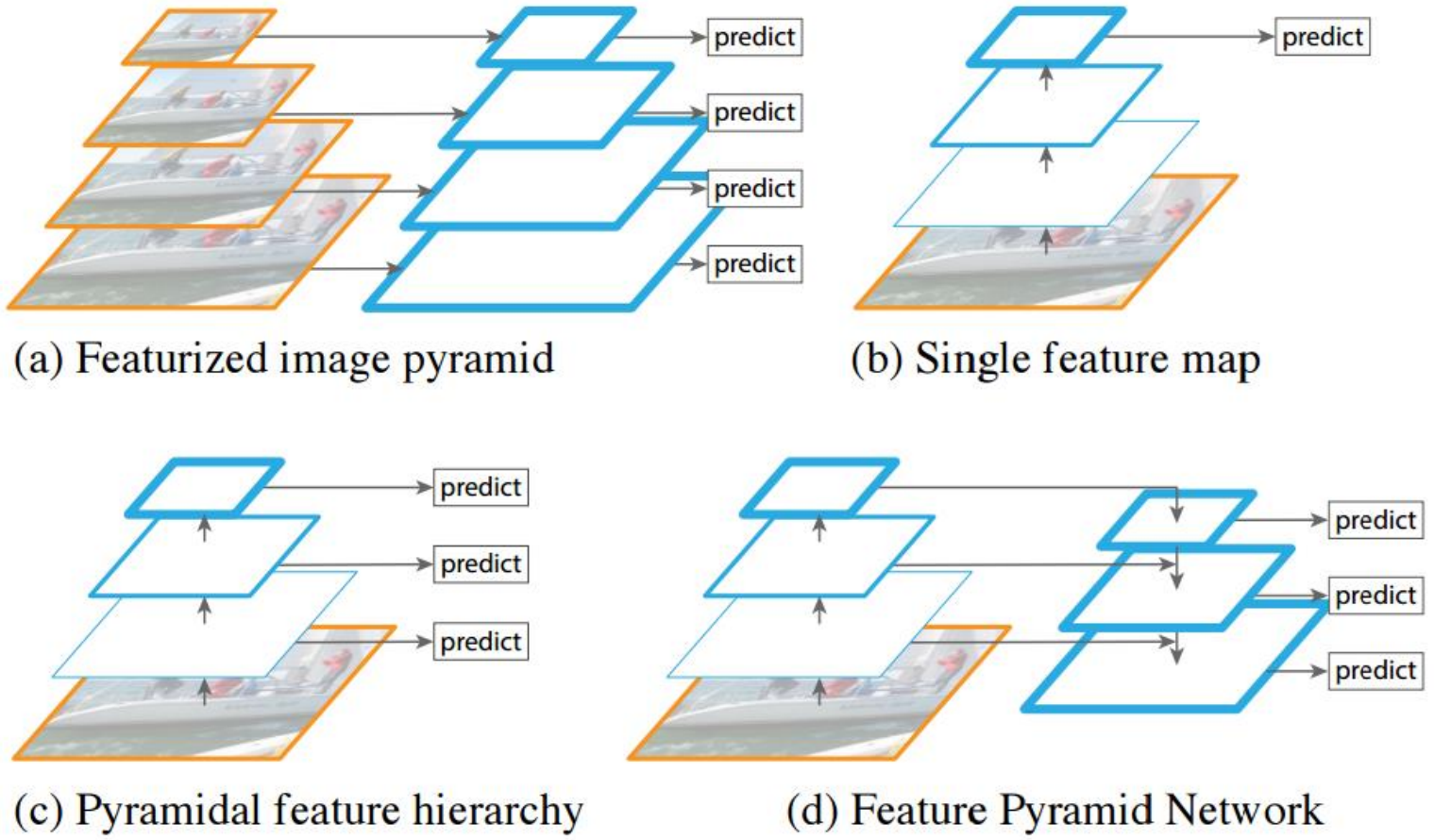
Discussion

- Deformable pool is similar to ROIAlign (in Mask RCNN)
- Deformable conv
  - flexible to learn the non-rigid objects

Face++ 旷视

Deformable Convolutional Networks, Dai etc, ICCV 2017
https://arxiv.org/abs/1703.06211

Two Stages Detector:
RCNN -> Fast RCNN -> FasterRCNN -> FPN



Face++ 旷视

Feature Pyramid Networks for Object Detection, Lin etc, CVPR 2017
https://arxiv.org/pdf/1612.03144.pdf

Two Stages Detector: FPN

Discussion

- FasterRCNN reproduced (setting)
- Deeply supervised (better feature)

Feature Pyramid Networks for Object Detection, Lin etc, CVPR 2017
https://arxiv.org/pdf/1612.03144.pdf

Two Stages Detector: FPN

Experiments

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5
RCNN	66	NA	NA	47s
Fast RCNN	77.0	82.3 (wth coco data)	NA	0.5s
Faster RCNN	73.2	70.4	NA	200ms
RFCN	79.5	77.6	29.9	170ms
FPN	NA	NA	36.2	6

Face++ 旷视

Feature Pyramid Networks for Object Detection, Lin etc, CVPR 2017
https://arxiv.org/pdf/1612.03144.pdf

Two Stages Detector:
RCNN -> Fast RCNN -> FasterRCNN -> FPN -> MaskRCNN

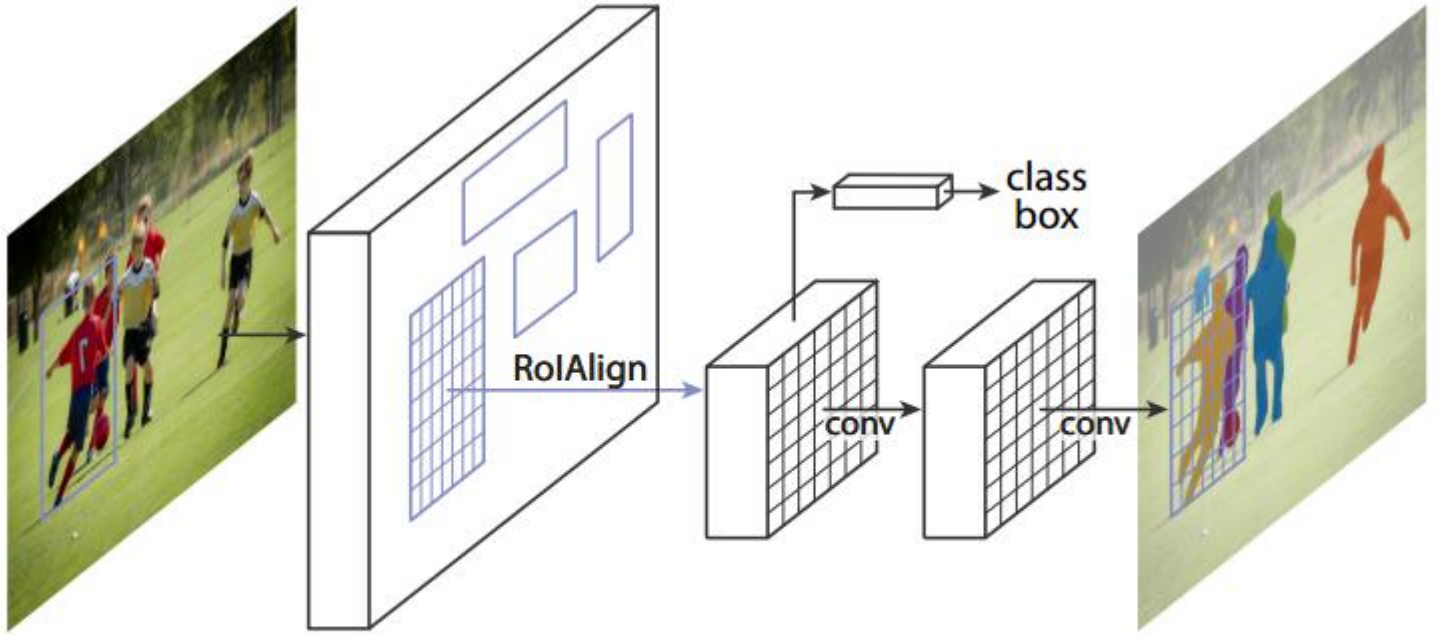


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Face++ 旷视

Mask R-CNN, He etc, ICCV 2017
https://arxiv.org/pdf/1703.06870.pdf

Two Stages Detector:
RCNN -> Fast RCNN -> FasterRCNN -> FPN -> MaskRCNN

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

Face++ 旷视

Mask R-CNN, He etc, ICCV 2017
https://arxiv.org/pdf/1703.06870.pdf

Two Stages Detector: Mask RCNN

Discussion

- Alignment issue in ROIPooling -> ROIAlign
- Multi-task learning: detection & mask

Mask R-CNN, He etc, ICCV 2017
https://arxiv.org/pdf/1703.06870.pdf

Two Stages Detector: Mask RCNN

Experiments

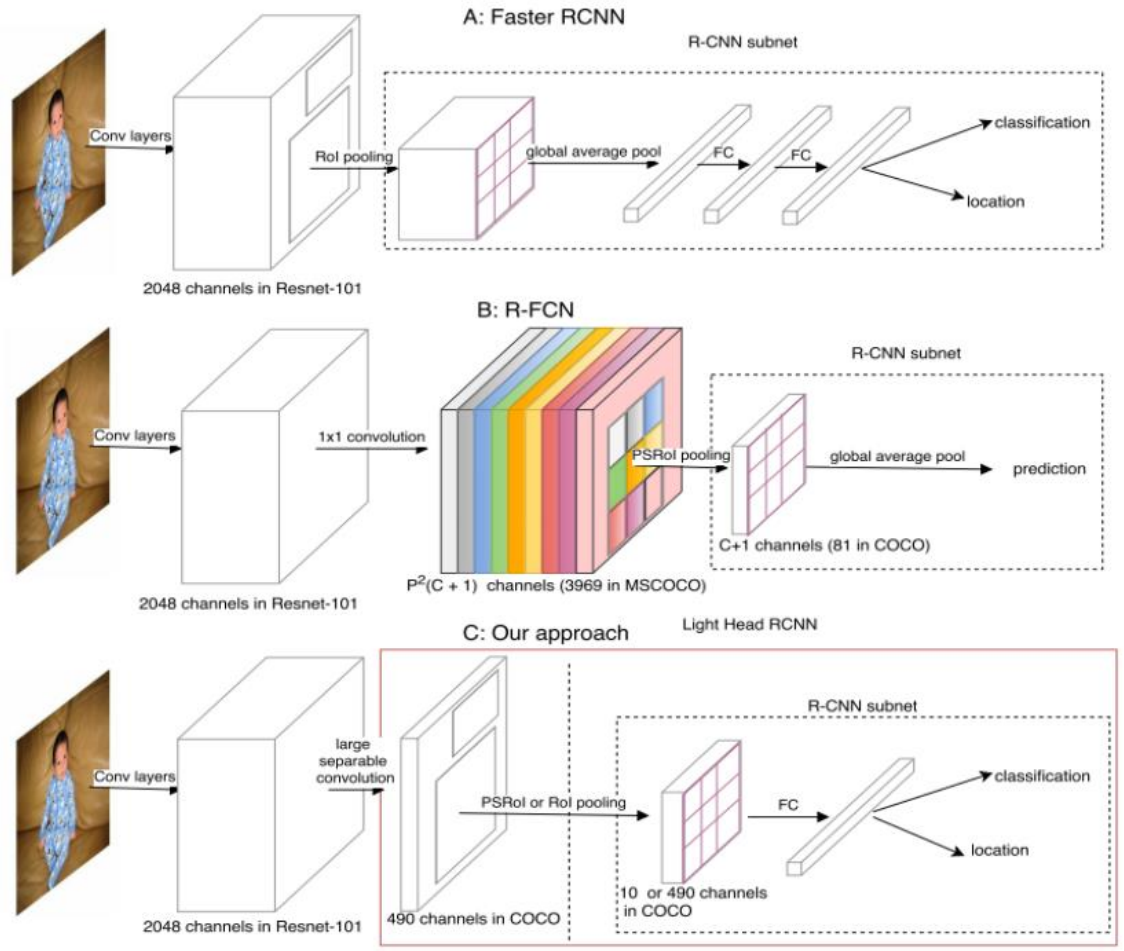
Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5
RCNN	66	NA	NA	47s
Fast RCNN	77.0	82.3 (wth coco data)	NA	0.5s
Faster RCNN	73.2	70.4	NA	200ms
RFCN	79.5	77.6	29.9	170ms
FPN	NA	NA	36.2	6
Mask RCNN	NA	NA	38.2	2.5

Face++ 旷视

Mask R-CNN, He etc, ICCV 2017
https://arxiv.org/pdf/1703.06870.pdf

Two Stages Detector: Light Head R-CNN

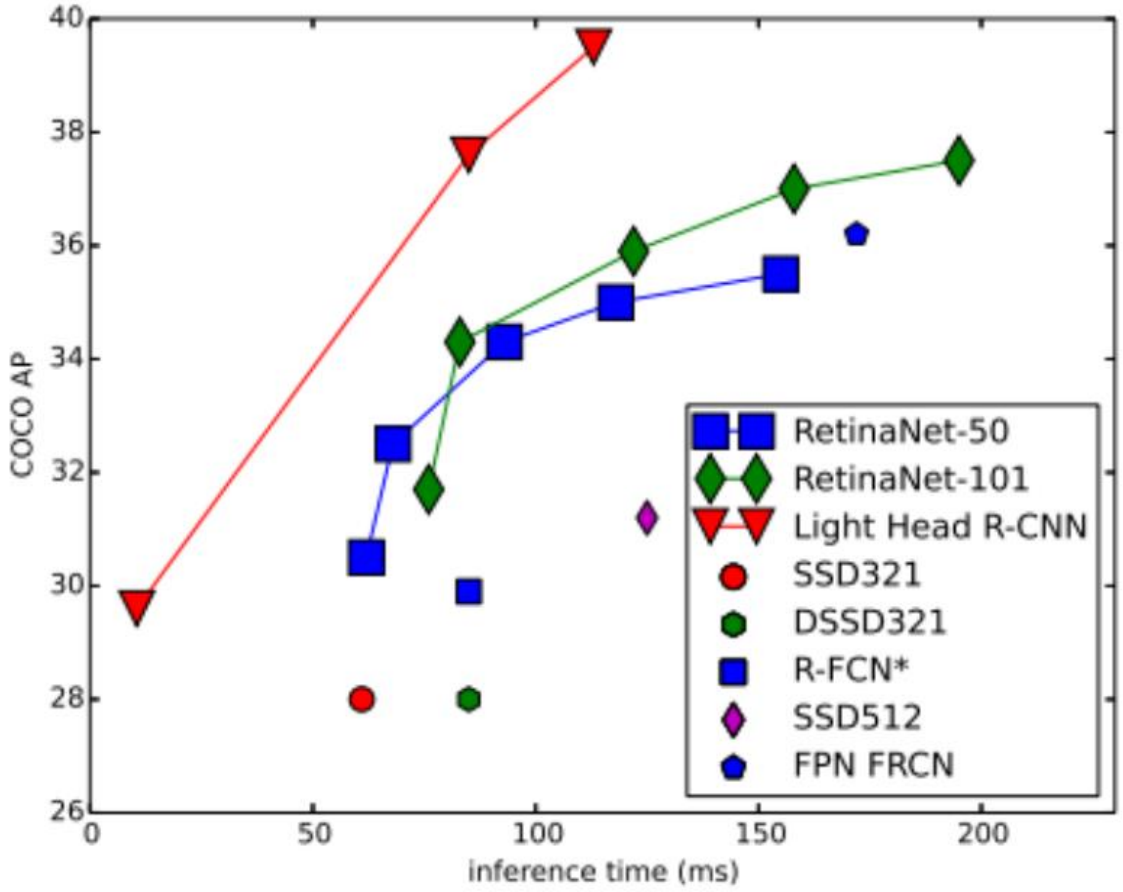
- Improve Inference speed in detection algorithms



Light-Head R-CNN: In Defense of Two-Stage Object Detector, Li etc, https://arxiv.org/pdf/1711.07264.pdf

Two Stages Detector: Light Head R-CNN

- Improve Inference speed in detection algorithms



Light-Head R-CNN: In Defense of Two-Stage Object Detector, Li etc, https://arxiv.org/pdf/1711.07264.pdf

Face++ 旷视

Two Stages Detector: MegDet

- Batchsize issue in general object detection
- Problems in small batch size
  - Long training time
  - Inaccurate BN statistics
  - Inbalacned positive and negative ratios

Face++ 旷视

MegDet: A Large Mini-Batch Object Detector, Peng etc, CVPR2018
https://arxiv.org/pdf/1711.07240.pdf

Two Stages Detector: MegDet

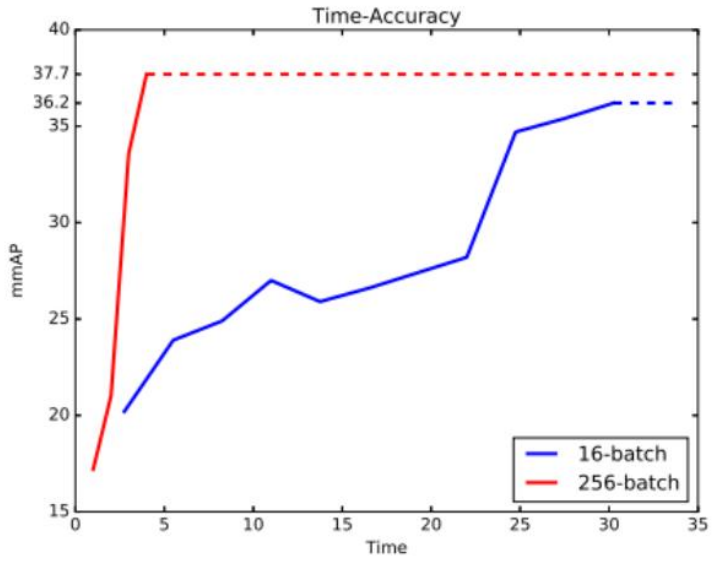
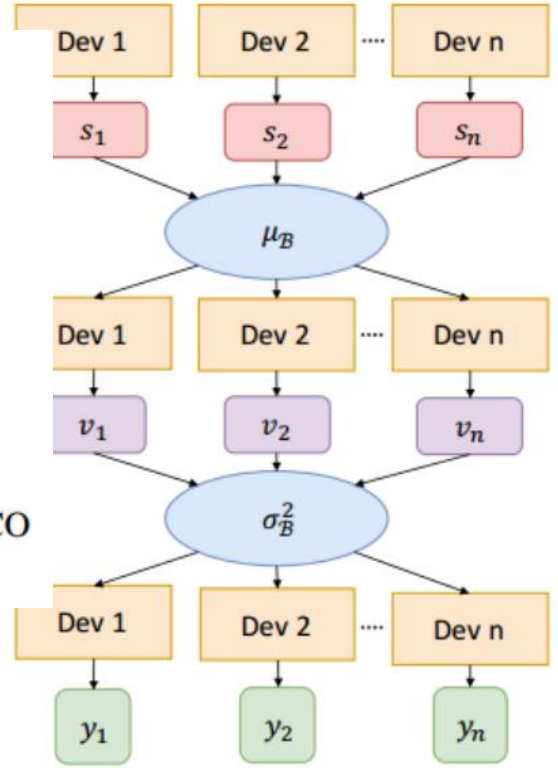


Figure 1: Validation accuracy of the same FPN object detector trained on COCO dataset, with mini-batch size 16 (on 8 GPUs) and mini-batch size 256 (on 128 GPUs). The large mini-batch detector is more accurate and its training is nearly an order-of-magnitude faster.

name	mmAP	mmAR
DANet	45.7	62.7
Trimps-Soushen+QINIU	48.0	65.4
bharat_umd	48.1	64.8
FAIR Mask R-CNN [14]	50.3	66.1
MSRA	50.4	69.0
UCenter	51.0	67.9
MegDet (Ensemble)	**52.5**	**69.0**

4: Result of (enhanced) MegDet on test-dev of COCO et.



MegDet: A Large Mini-Batch Object Detector, Peng etc, CVPR2018
https://arxiv.org/pdf/1711.07240.pdf

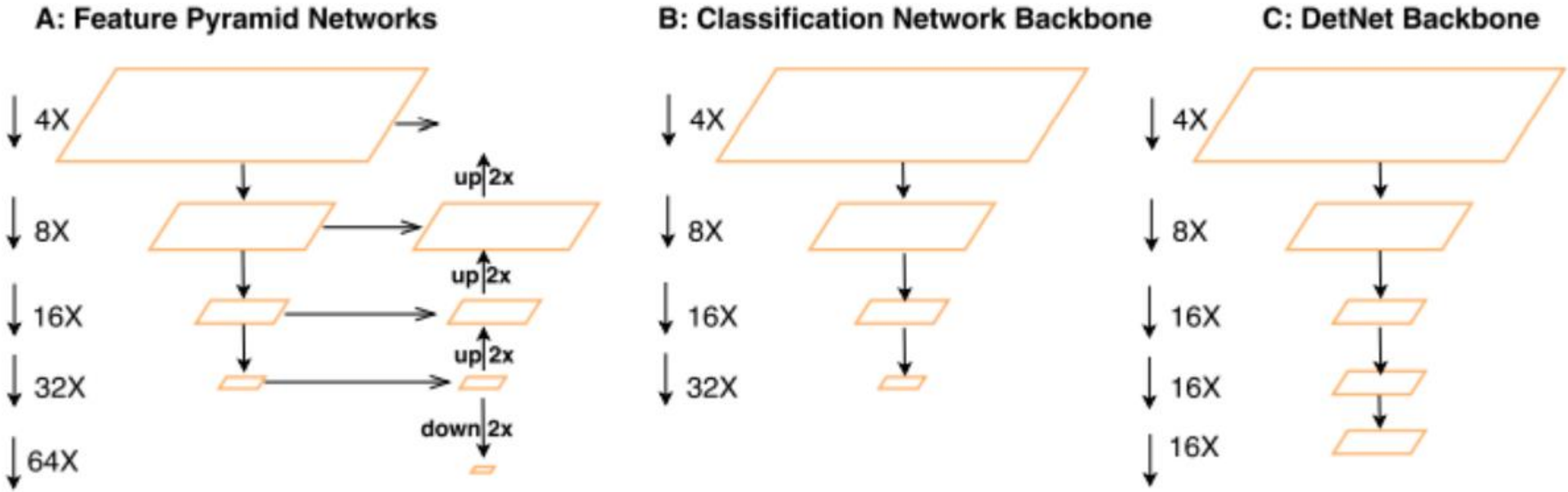
Two Stages Detector: DetNet

- Pretrain the backbone network for Detection
- Problems with the ImageNet pretrain model
  - Target for the classification problem, not localization friendly
  - Gap between the backbone and detection network
    - Not initialization for P6 (and P7)
- Train the Backbone by maintaining the spatial resolution (localization) and receptive field (classification)

Face++ 旷视

DetNet: A Backbone network for Object Detection, Li etc
https://arxiv.org/abs/1804.06215

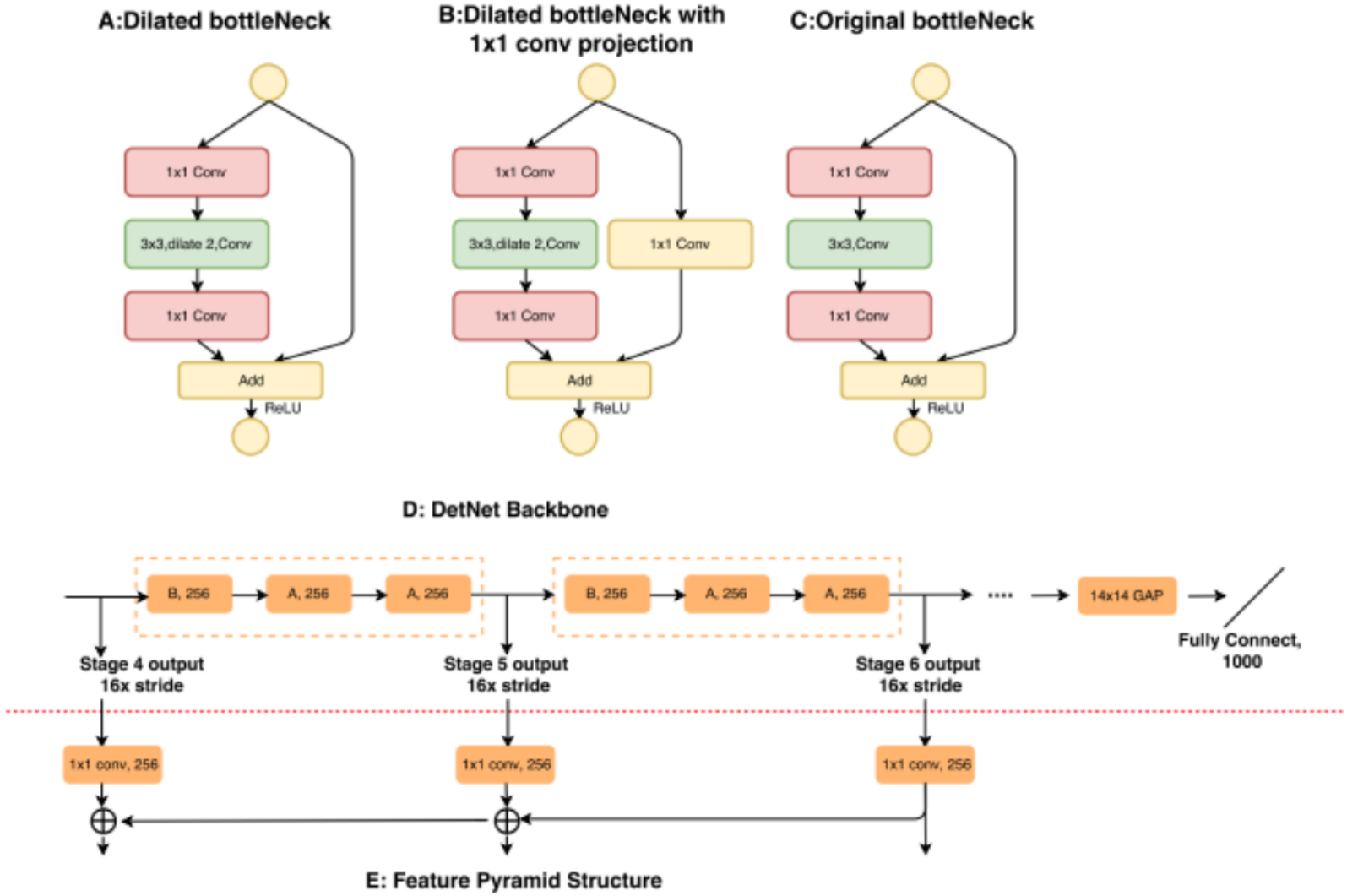
Two Stages Detector: DetNet



Face++ 旷视

DetNet: A Backbone network for Object Detection, Li etc
https://arxiv.org/abs/1804.06215

Two Stages Detector: DetNet



DetNet: A Backbone network for Object Detection, Li etc
https://arxiv.org/abs/1804.06215

Two Stages Detector: DetNet

Models	scales	mAP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{85}
ResNet-50	over all scales	37.9	60.0	55.1	47.2	33.1	22.1
	small	22.9	40.1	35.5	28.0	17.5	10.4
	middle	40.6	63.9	59.0	51.2	35.7	23.3
	large	49.2	72.2	68.2	60.8	46.6	34.5
DetNet-59	over all scales	40.2	61.7	57.0	49.6	36.2	25. 8
	small	23.9	41.8	36.8	29.8	17.7	10.5
	middle	43.2	65.8	61.2	53.6	39.9	27.3
	large	52.0	73.1	69.5	63	51.4	40.0

Models	Backbone	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
SSD513 [3]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3,37]	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
Faster R-CNN +++ [11]	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN G-RMI [2] [38]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
RetinaNet [4]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
FPN [33]	ResNet-101	37.3	59.6	40.3	19.8	40.2	48.8
FPN	**DetNet-59**	**40.3**	62.1	43.8	23.6	42.6	50.0

Table 7. Comparison of object detection results between our approach and state-of-the-art on MSCOCO test-dev datasets. Based on our simple and effective backbone DetNet-59, our model outperforms all previous state-of-the-art. It is worth nothing that DetNet-59 yields better results with much lower FLOPs.

Face++ 旷视

DetNet: A Backbone network for Object Detection, Li etc
https://arxiv.org/abs/1804.06215

Two Stages Detector: Summary

- Speed
  - RCNN -> Fast RCNN -> Faster RCNN -> RFCN -> Light Head R-CNN
- performance
  - Divide and conquer
    - FPN
  - Deformable Pool/ROIAlign
  - Deformable Conv
  - Multi-task learning
  - Multi-GPU BN

Two Stages Detector: Discussion

FasterRCNN vs RFCN

One stage vs two Stage

Open Problem in Detection

- FP
- NMS (detection in crowd)
  - CrowdHuman Dataset: https://sshao0516.github.io/CrowdHuman/
- GT assignment issue
- Detection in video
  - detect & track in a network

Outline

- Detection
- **Conclusion**

Face++ 旷视

Conclusion

- Detection
  - One stage: Densebox, YOLO, SSD, RetinaNet
  - Two Stage: RCNN, Fast RCNN, FasterRCNN, RFCN, FPN, Mask RCNN

Introduction to Face++ Detection Team

- Category-level Recognition
  - Detection
    - Face Detection:
      - FAN: https://arxiv.org/pdf/1711.07246.pdf
      - Sface: https://arxiv.org/pdf/1804.06559.pdf
    - Human Detection:
      - Repulsion loss: https://arxiv.org/abs/1711.07752
      - CrowdHuman: https://arxiv.org/pdf/1805.00123.pdf
    - General Object Detection:
      - Light Head: https://arxiv.org/pdf/1711.07264.pdf https://github.com/zengarden/light_head_rcnn
      - MegDet: https://arxiv.org/pdf/1711.07240.pdf
      - DetNet: https://arxiv.org/pdf/1804.06215.pdf
  - Segmentation
    - Large Kernel Matters: https://arxiv.org/pdf/1703.02719.pdf
    - DFN: https://arxiv.org/pdf/1804.09337.pdf
  - Skeleton:
    - CPN: https://arxiv.org/pdf/1711.07319.pdf
    - https://github.com/chenyilun95/tf-cpn

Face++ 旷视

Thanks

Face++ 旷视