

Introduction to Object Detection

Part 1

Gang Yu

Megvii Researcher

yugang@megvii.com

Visual Recognition

A fundamental task in computer vision

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

...



Category-level Recognition



Category-level Recognition



Instance-level Recognition

Representation

- Bounding-box
 - Face Detection, Human Detection, Vehicle Detection, Text Detection, general Object Detection
- Point
 - Semantic segmentation (will be discussed in next week)
- Keypoint
 - Face landmark
 - Human Keypoint

Outline

- Detection
- Human Keypoint
- Conclusion

Outline

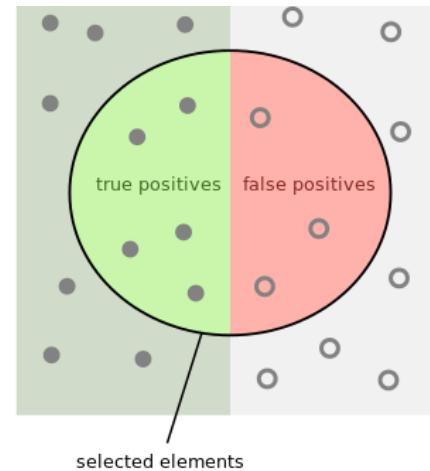
- **Detection**
- Human Keypoint
- Conclusion

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 $p(r)$ over the interval from $r = 0$ to $r = 1$.^[9]

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



How many selected items are relevant?
How many relevant items are selected?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$
$$\text{Recall} = \frac{\text{true positives}}{\text{relevant items}}$$

Detection - Evaluation Criteria

mmAP

```
Average Precision (AP):
    AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
    APIoU=.50        % AP at IoU=.50 (PASCAL VOC metric)
    APIoU=.75        % AP at IoU=.75 (strict metric)

    AP Across Scales:
        APsmall          % AP for small objects: area < 322
        APmedium         % AP for medium objects: 322 < area < 962
        APlarge           % AP for large objects: area > 962

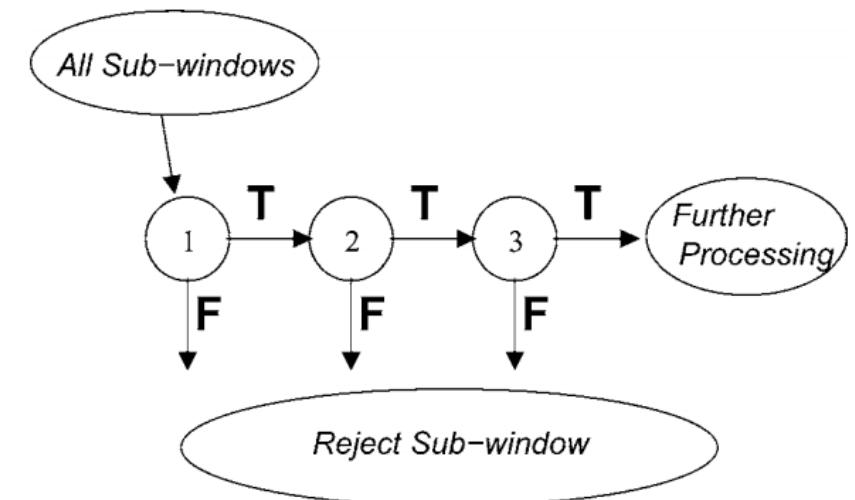
    Average Recall (AR):
        ARmax=1          % AR given 1 detection per image
        ARmax=10         % AR given 10 detections per image
        ARmax=100        % AR given 100 detections per image

    AR Across Scales:
        ARsmall          % AR for small objects: area < 322
        ARmedium         % AR for medium objects: 322 < area < 962
        ARlarge           % AR for large objects: area > 962
```

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).

How to perform a detection?

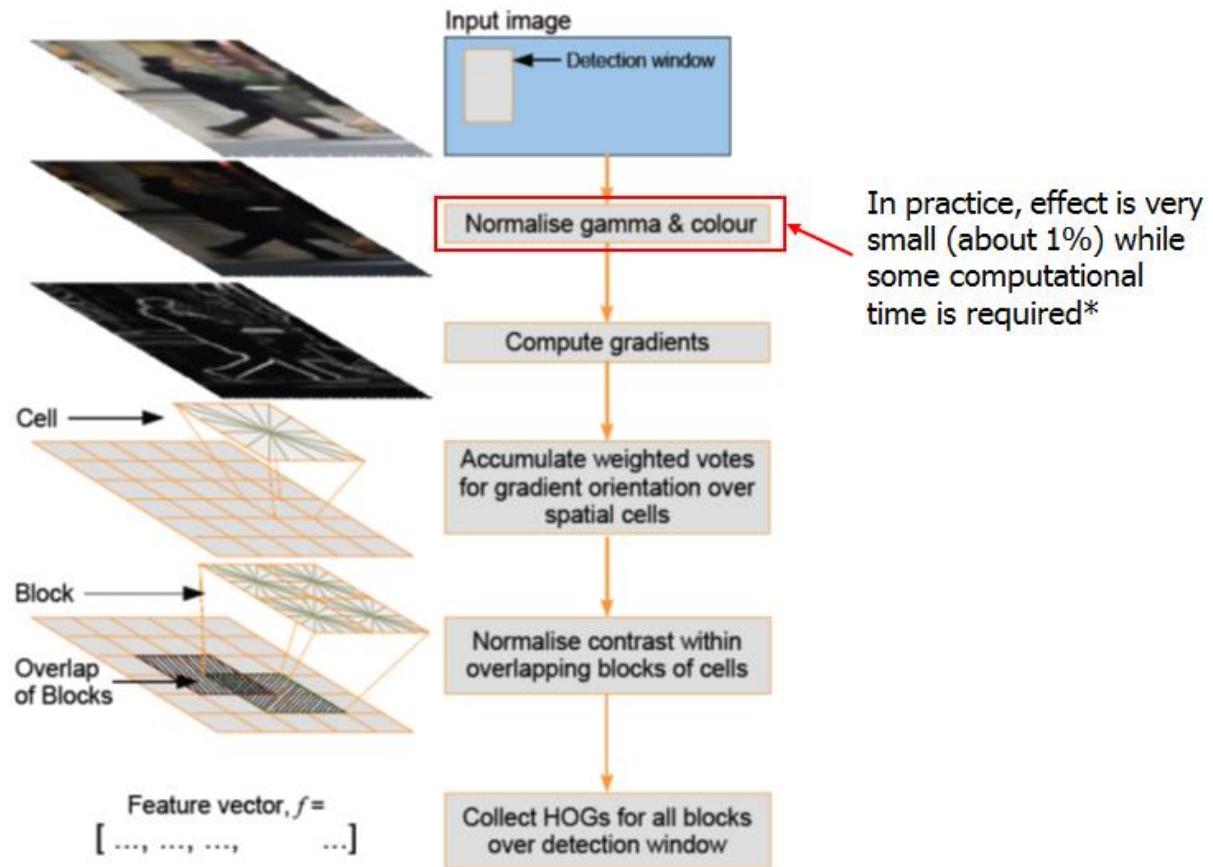
- Sliding window: enumerate all the windows (up to millions of windows)
 - VJ detector: cascade chain
- Fully Convolutional network
 - shared computation



General Detection Before Deep Learning

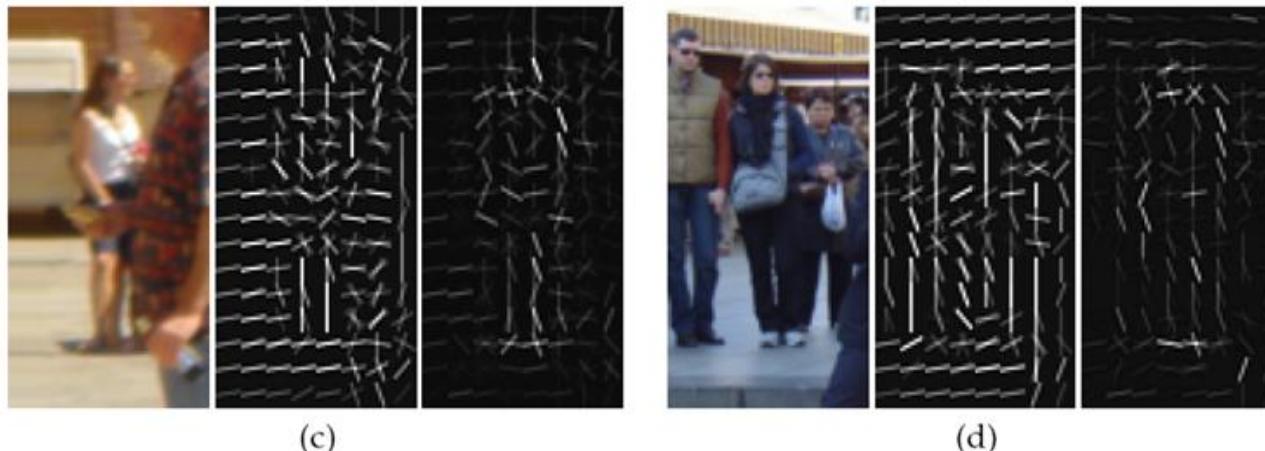
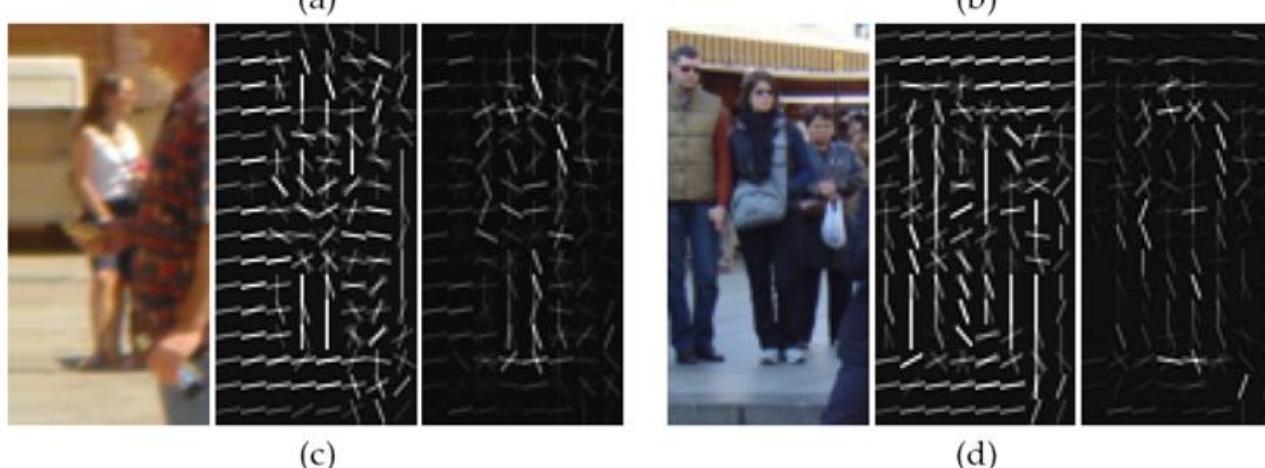
- Feature + classifier
- Feature
 - Haar Feature
 - HOG (Histogram of Gradient)
 - LBP (Local Binary Pattern)
 - ACF (Aggregated Channel Feature)
 - ...
- Classifier
 - SVM
 - Bootsing
 - Random Forest

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),
the dominant orientations selected by the SVM (obtained by multiplying the feature vector by the corresponding weights from the linear SVM). (3)

General Detection Before Deep Learning

Traditional Methods

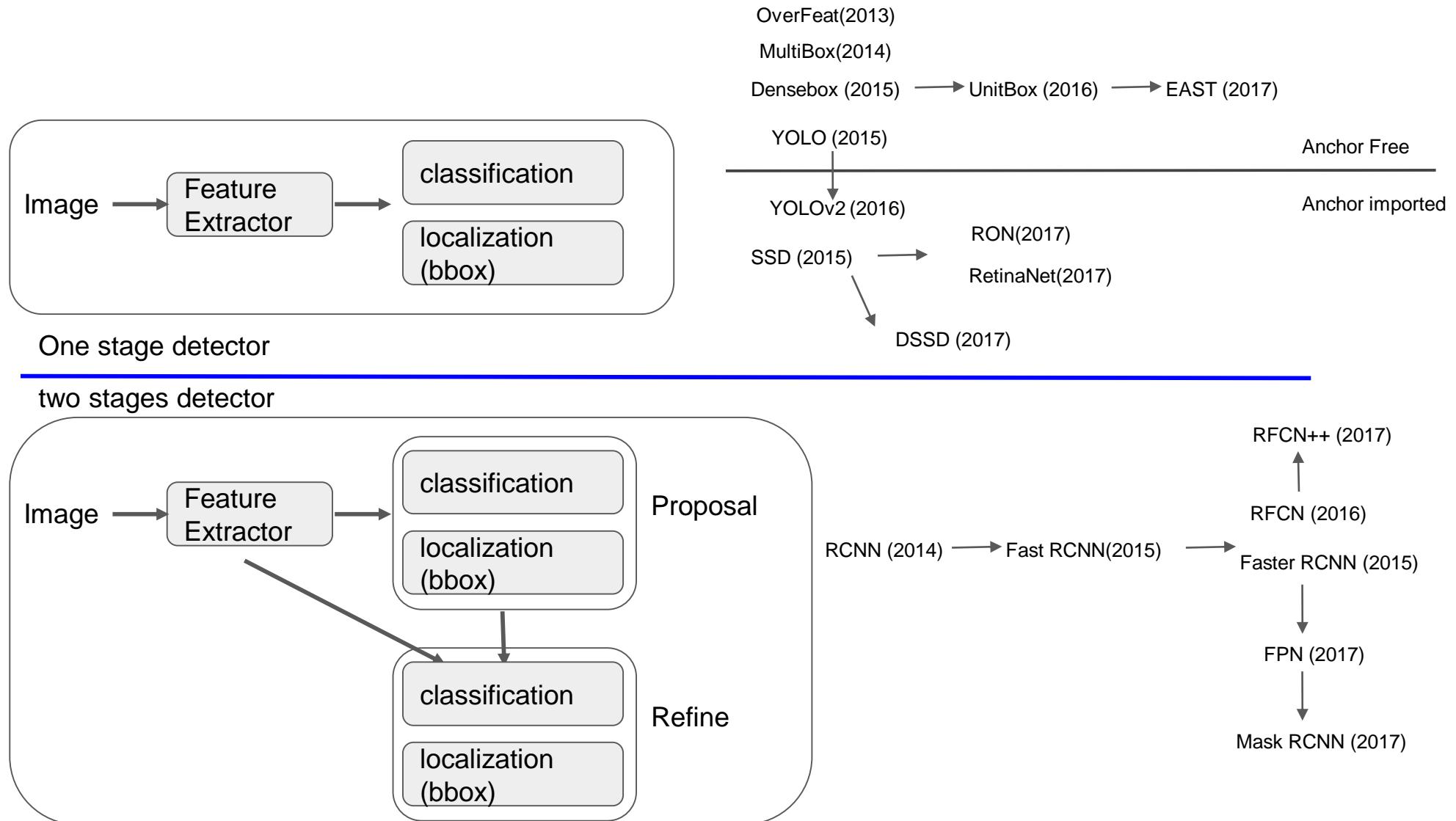
- Pros
 - Efficient to compute (e.g., HAAR, ACF) on CPU
 - Easy to debug, analyze the bad cases
 - reasonable performance on limited training data
- Cons
 - Limited performance on large dataset
 - Hard to be accelerated by GPU

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: Densebox

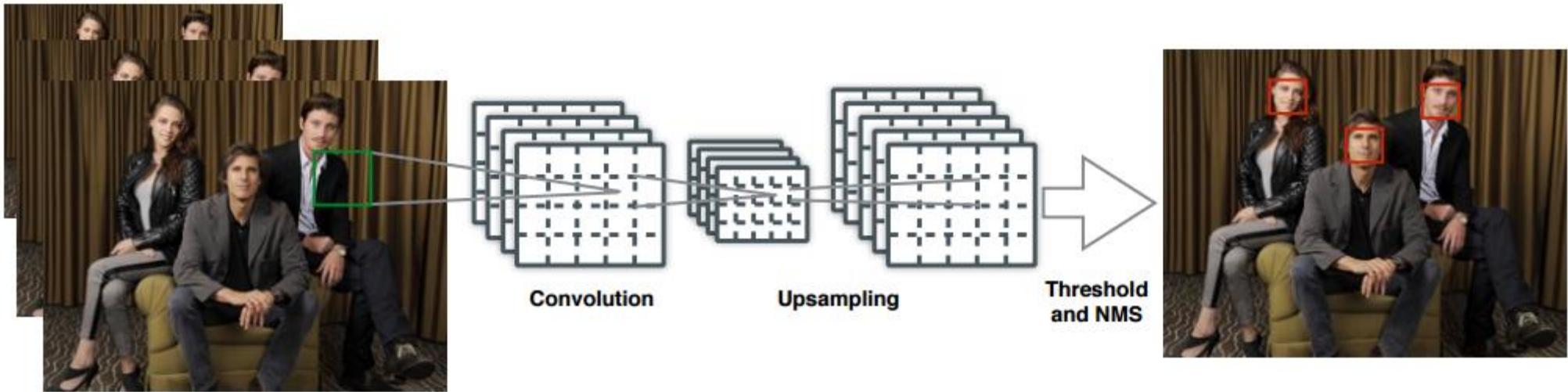


Figure 1: **The DenseBox Detection Pipeline.** 1) Image pyramid is fed to the network. 2) After several layers of convolution and pooling, upsampling feature map back and apply convolution layers to get final output. 3) Convert output feature map to bounding boxes , and apply non-maximum suppression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

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

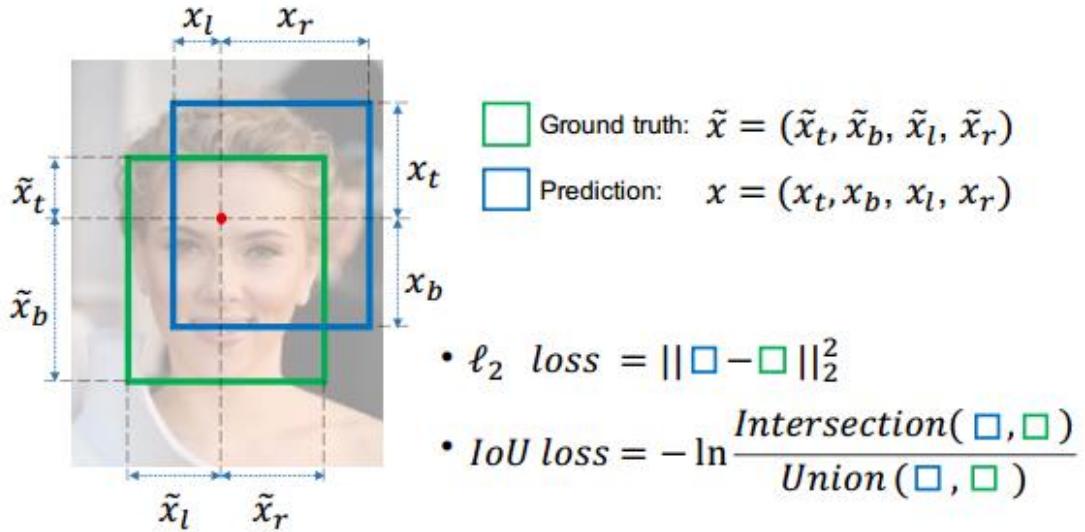


Figure 1: Illustration of IoU loss and ℓ_2 loss for pixel-wise bounding box prediction.

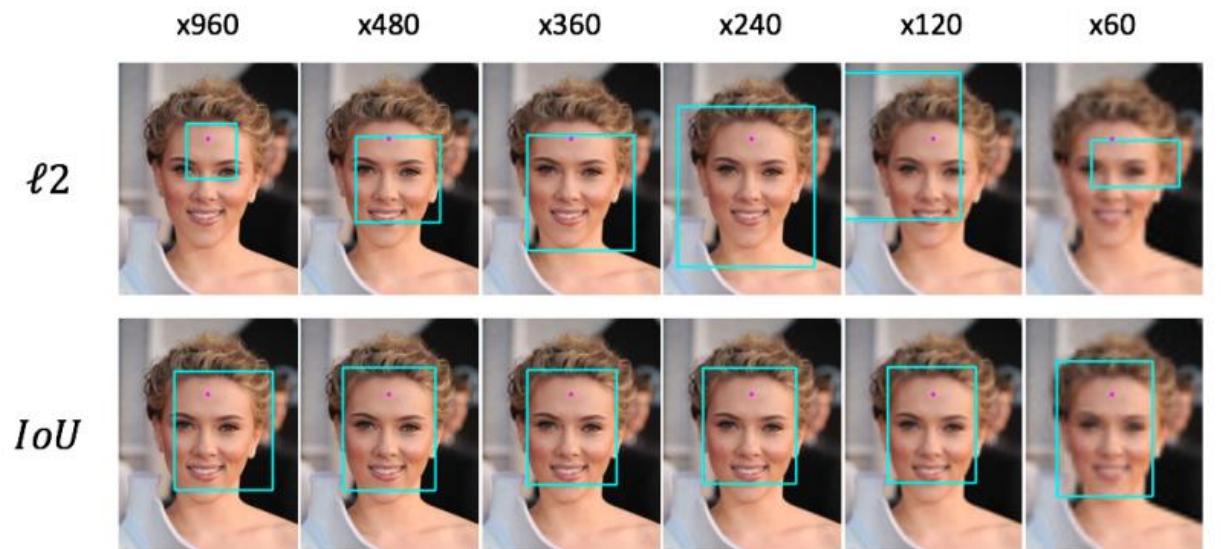


Figure 5: Compared to ℓ_2 loss, the IoU loss is much more robust to scale variations for bounding box prediction.

One Stage Detector: Densebox -> UnitBox->EAST

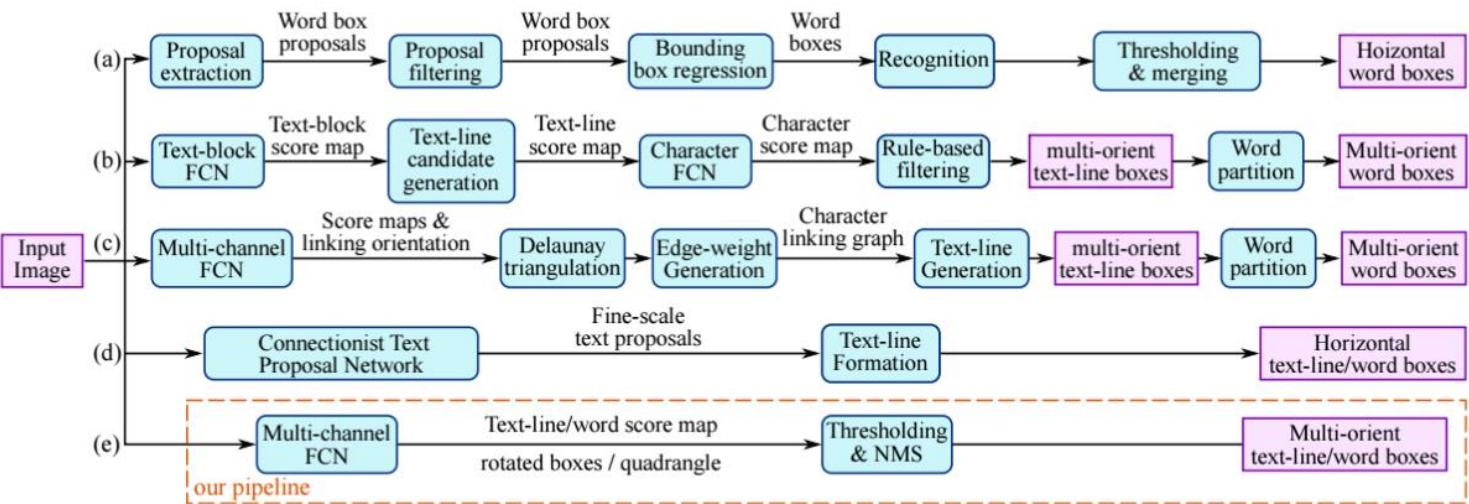
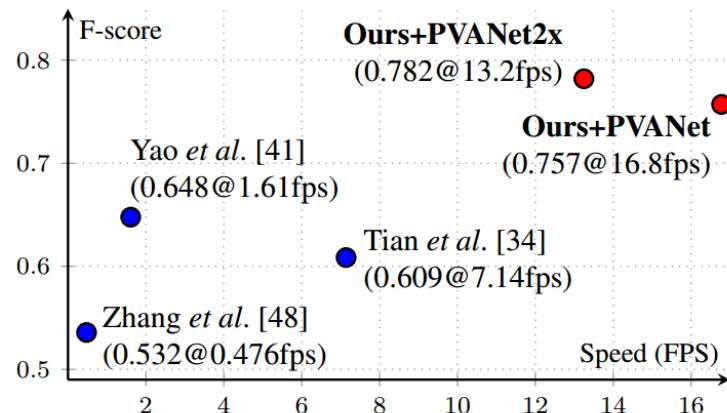
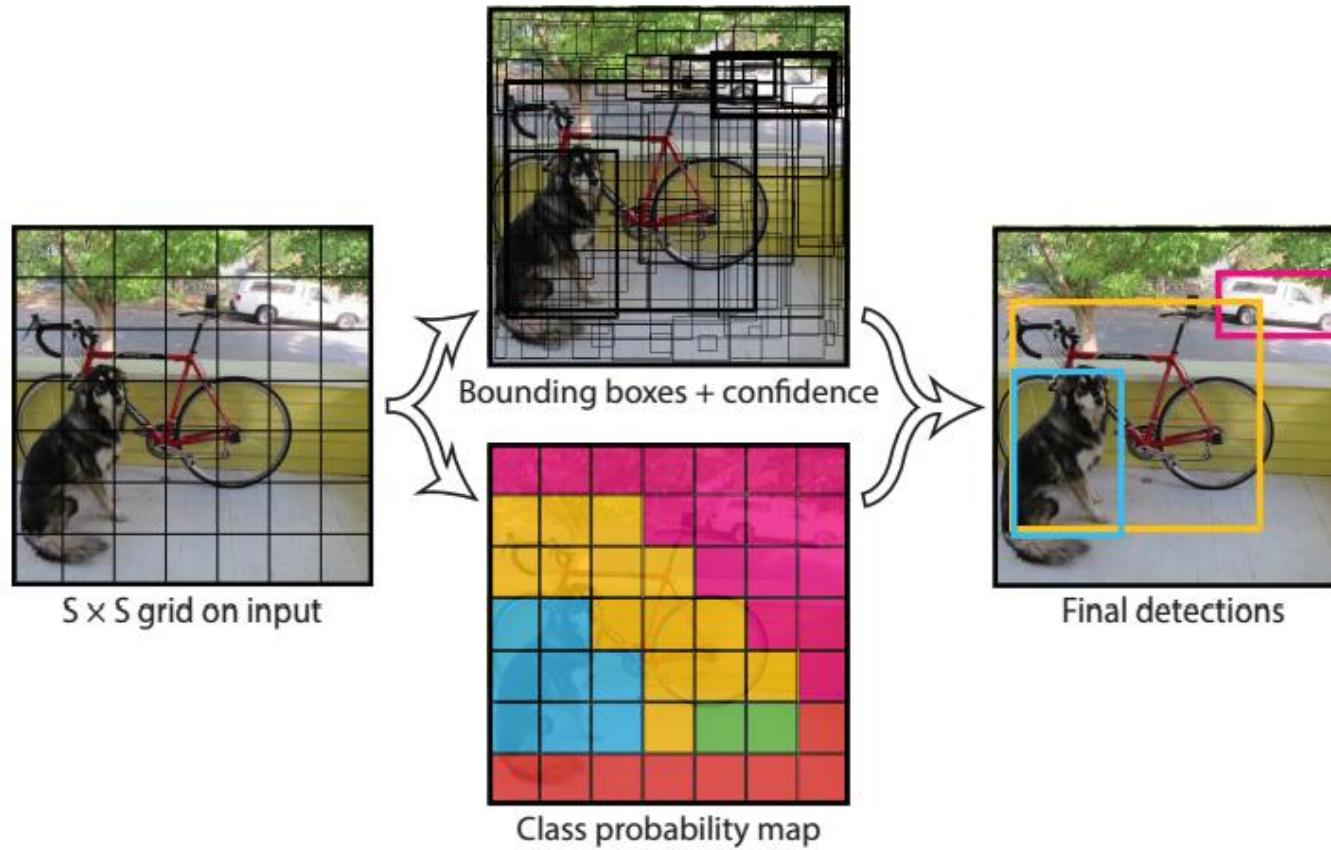


Figure 2. Comparison of pipelines of several recent works on scene text detection: (a) Horizontal word detection and recognition pipeline proposed by Jaderberg *et al.* [12]; (b) Multi-orient text detection pipeline proposed by Zhang *et al.* [48]; (c) Multi-orient text detection pipeline proposed by 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.

One Stage Detector: YOLO



One Stage Detector: YOLO

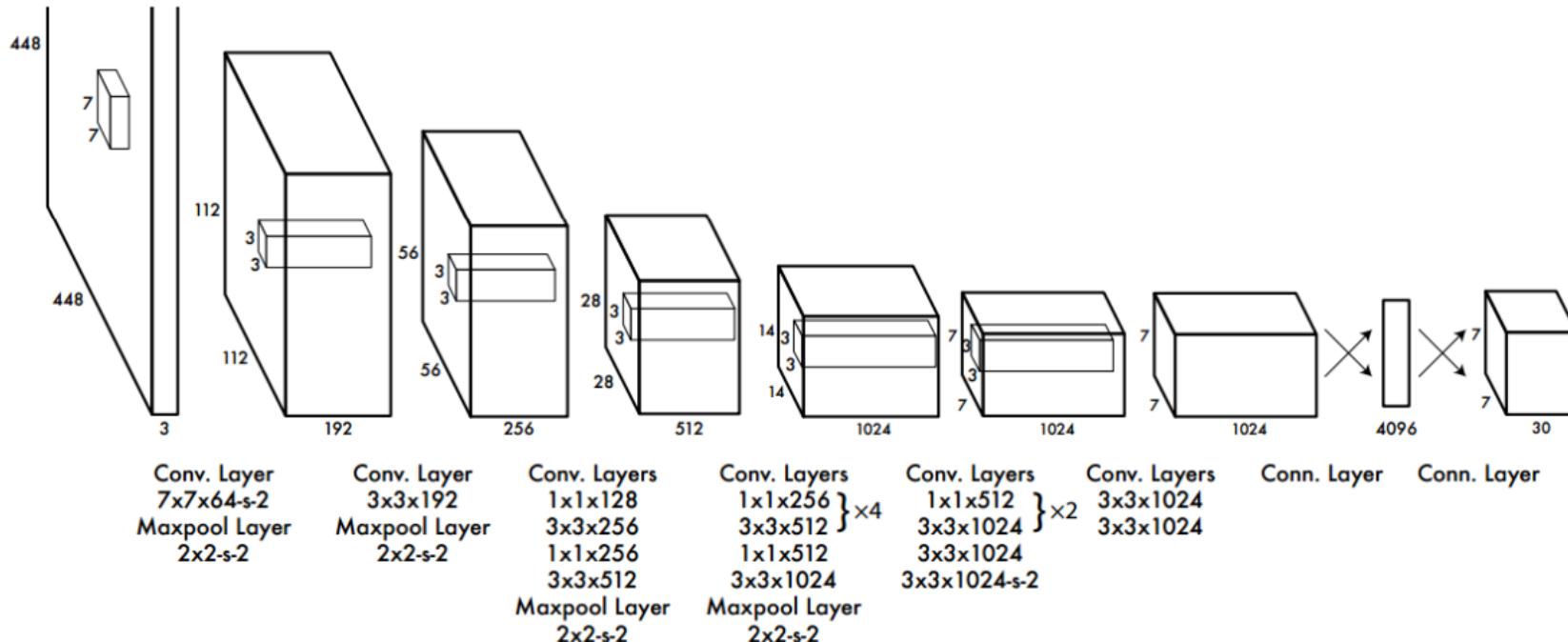


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

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)

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)

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

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%.

One Stage Detector: YOLO -> YOLOv2

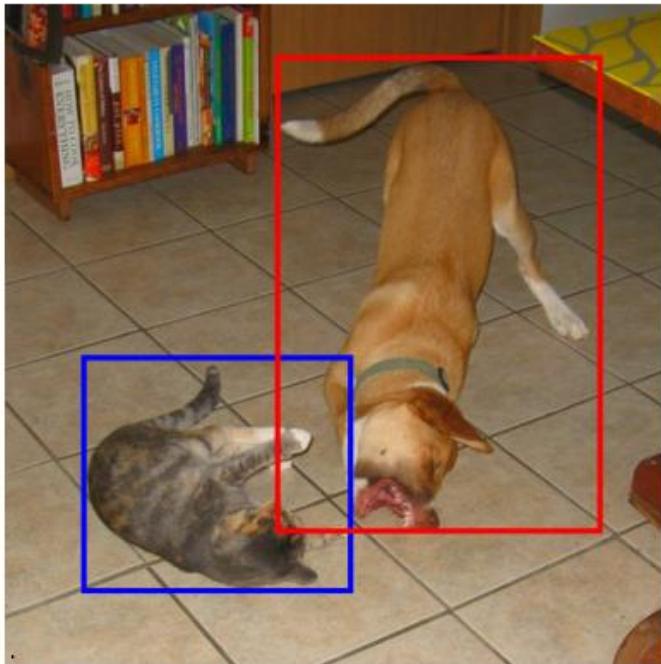
Experiments:

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

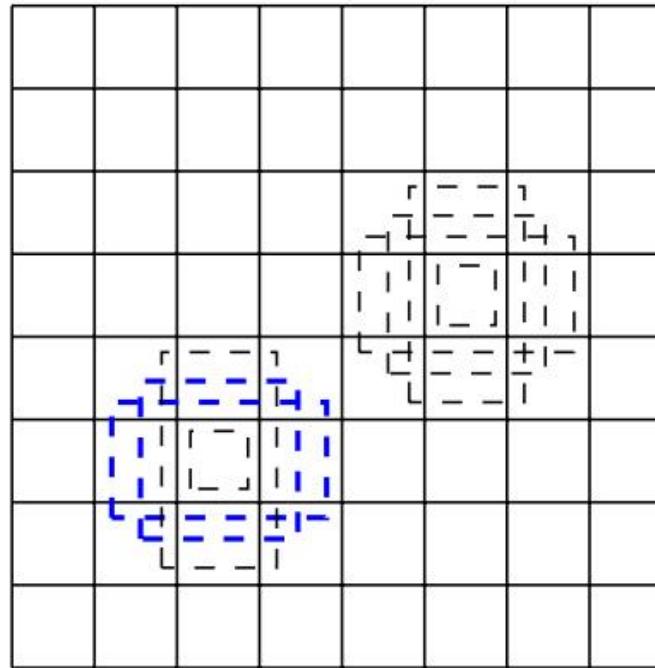
One Stage Detector: YOLO -> YOLOv2

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

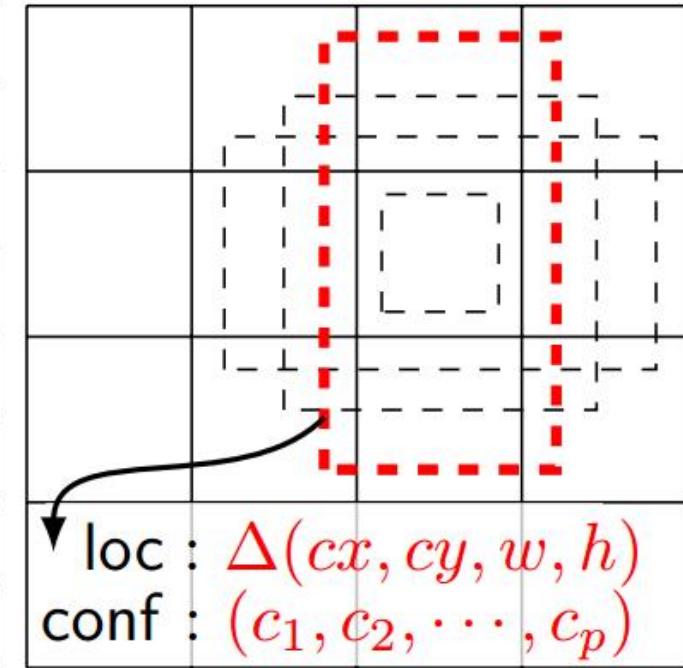
One Stage Detector: SSD



(a) Image with GT boxes



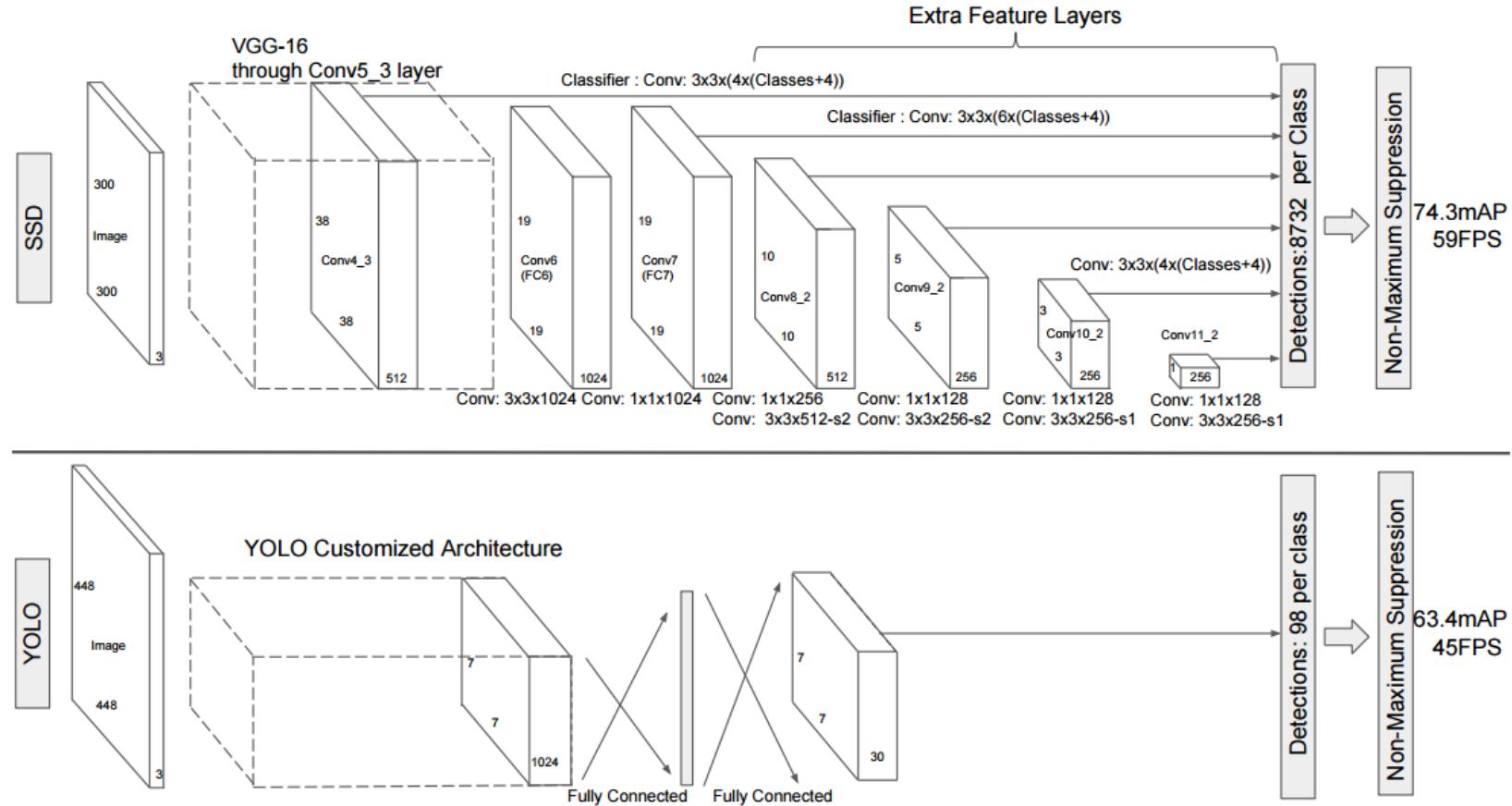
(b) 8×8 feature map



loc : $\Delta(cx, cy, w, h)$
conf : (c_1, c_2, \dots, c_p)

(c) 4×4 feature map

One Stage Detector: SSD



One Stage Detector: SSD

- Anchor
 - GT-anchor assignment
 - GT is predicted by one best matched (IOU) anchor or matched with an anchor with $\text{IOU} > 0.5$
 - 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

One Stage Detector: SSD

Discussion:

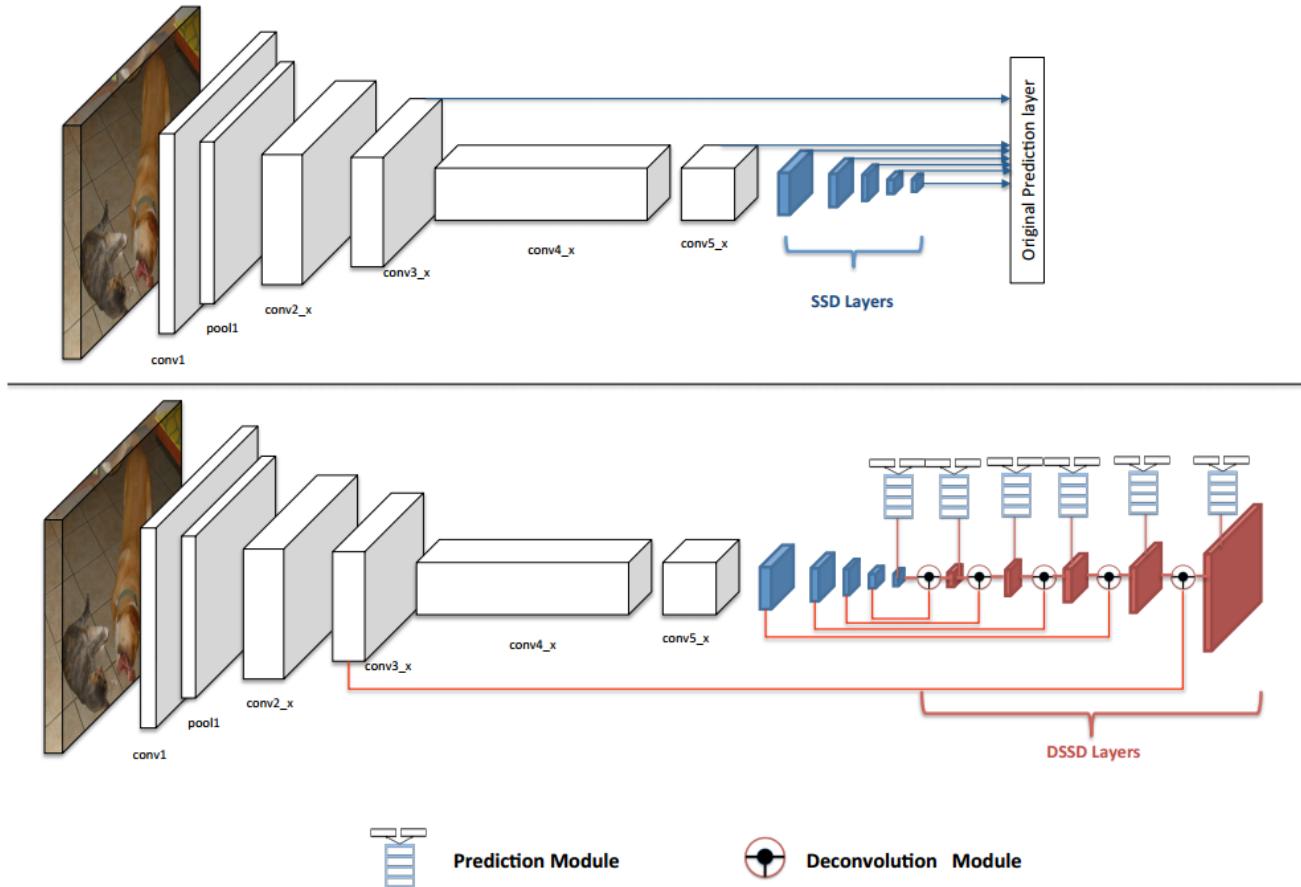
- Assume small objects can be predicted in earlier layers (not very strong semantics) (**DSSD, RON, RetinaNet**)
- strong data augmentation
- VGG model (**Replace by resnet in DSSD**)
 - cannot be easily adapted to other models
 - a lot of hacks
- A long tail (Large computation)

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

One Stage Detector: SSD -> DSSD



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

One Stage Detector: SSD -> RON

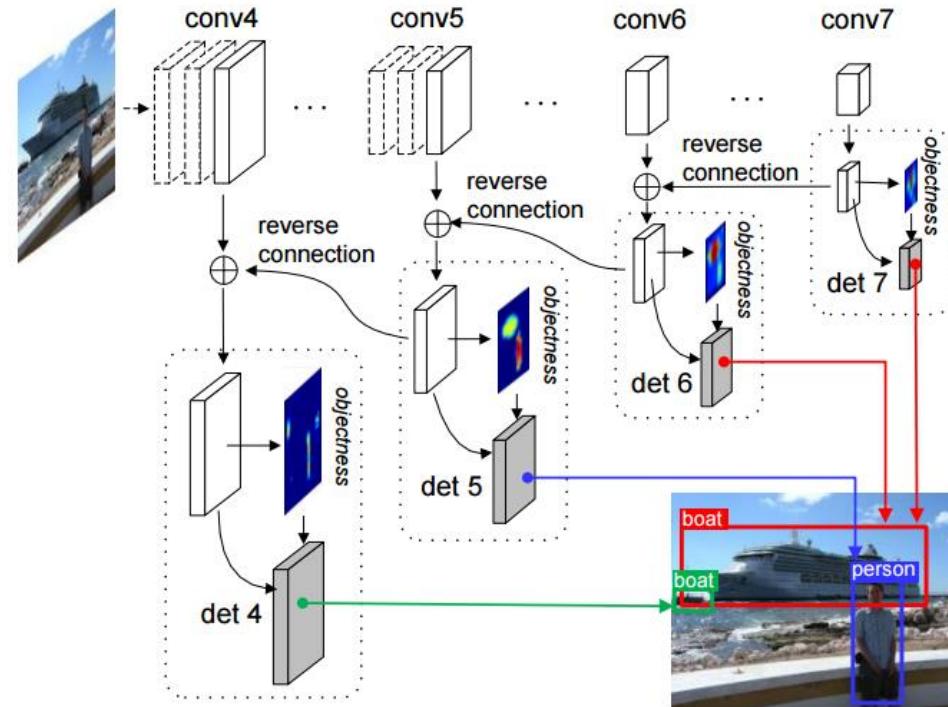
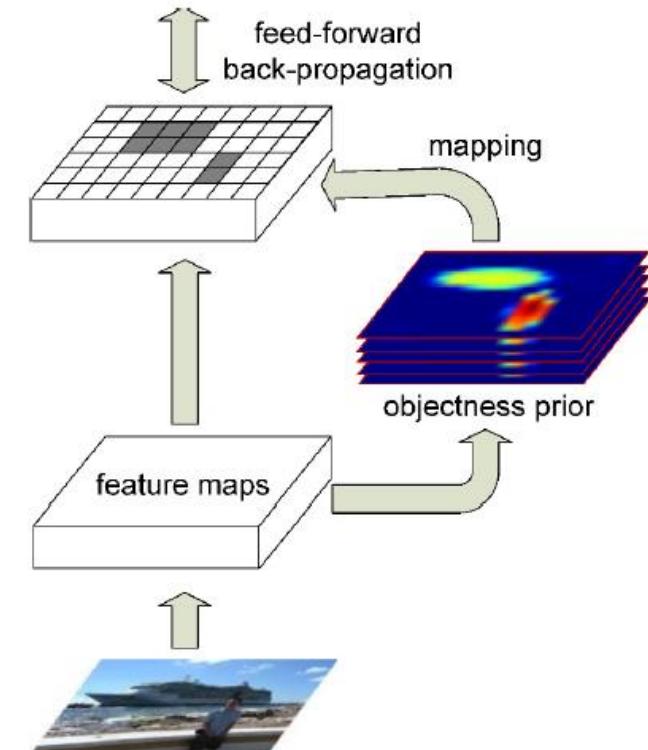


Figure 2. RON object detection overview. Given an input image, the network firstly computes features of the backbone network. Then at each detection scale: (a) adds reverse connection; (b) generates objectness prior; (c) detects object on its corresponding CNN scales and locations. Finally, all detection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

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



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

One Stage Detector: SSD -> RetinaNet

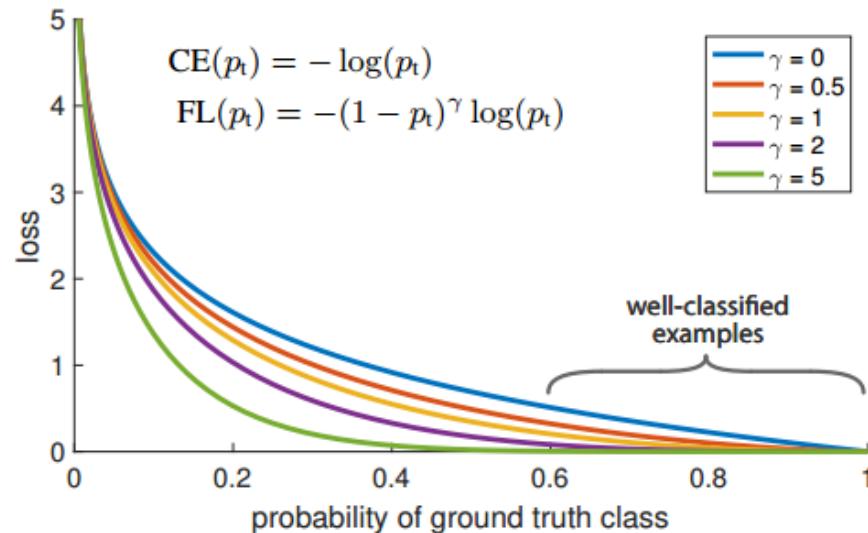


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^\gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will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

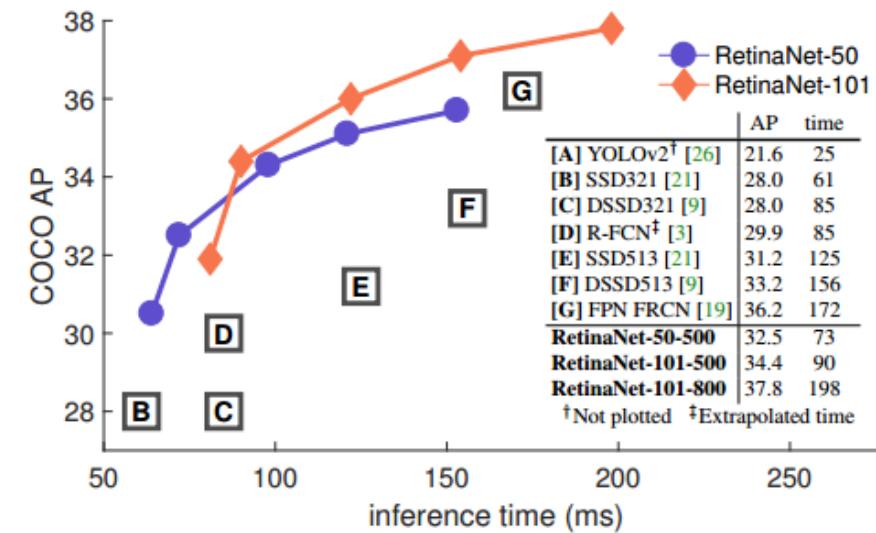


Figure 2. Speed (ms) versus accuracy (AP) on COCO test-dev. Enabled by the focal loss, our simple one-stage *RetinaNet* detector outperforms all previous one-stage and two-stage detectors, including the best reported Faster R-CNN [27] system from [19]. We show variants of RetinaNet with ResNet-50-FPN (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.

One Stage Detector: SSD -> RetinaNet

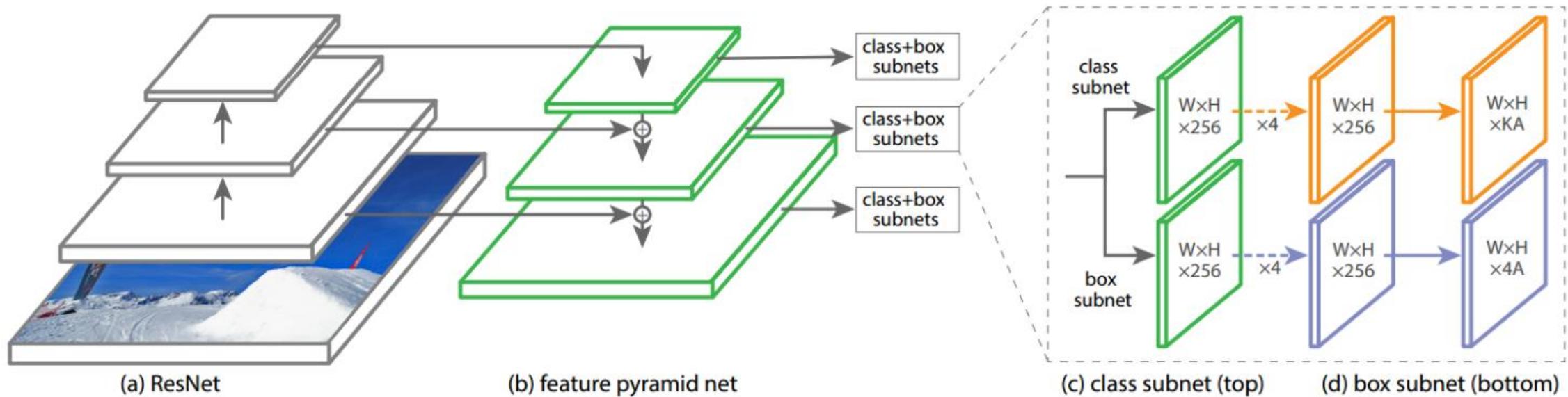


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [19] backbone on top of a feedforward ResNet architecture [15] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying 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.

One Stage Detector: RetinaNet

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

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

One Stage Detector: Summary

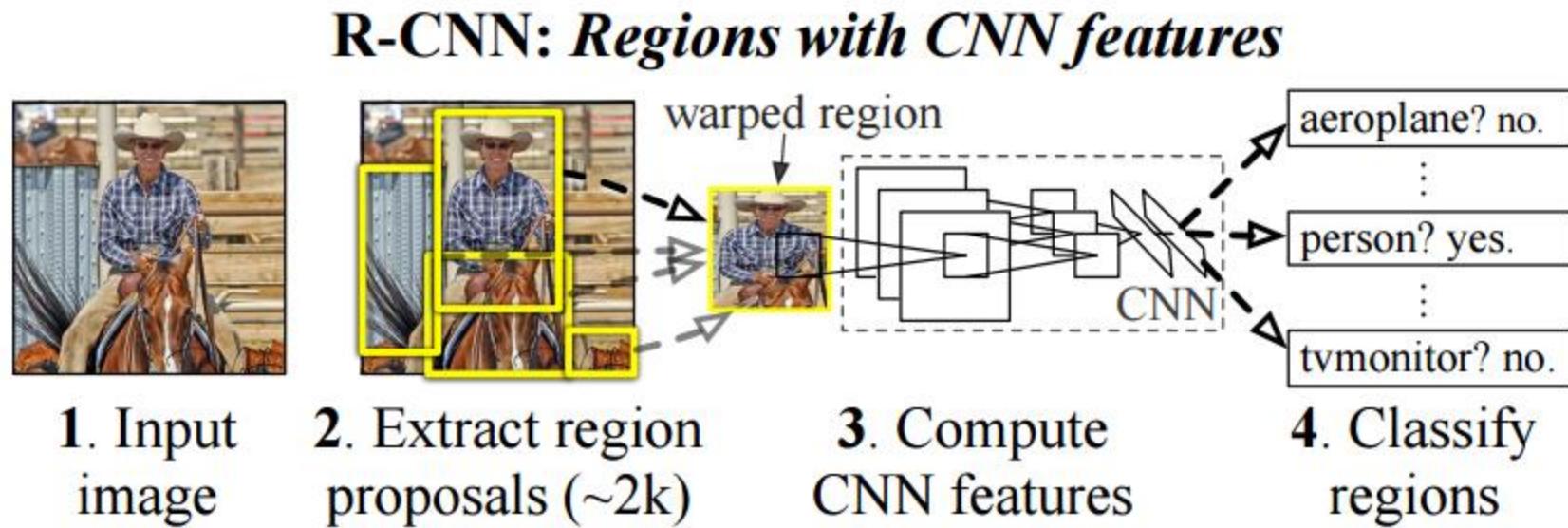
- Anchor
 - No anchor: YOLO, densebox/unitbox/east
 - Anchor: YOLOv2, SSD, DSSD, RON, RetinaNet
- Divide and conquer
 - SSD, DSSD, RON, RetinaNet
- loss sample
 - all sample: densebox
 - OHEM: SSD
 - focal loss: RetinaNet

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



Two Stages Detector: RCNN

Discussion

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

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

Two Stages Detector: RCNN -> Fast RCNN

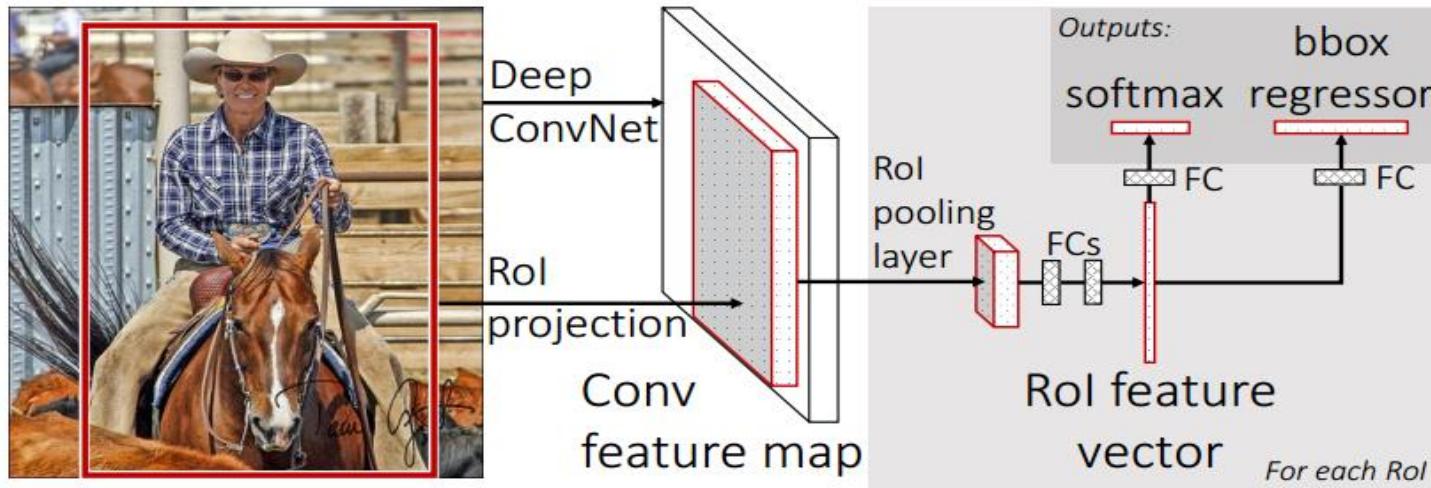


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each ROI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per ROI: softmax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

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

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

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

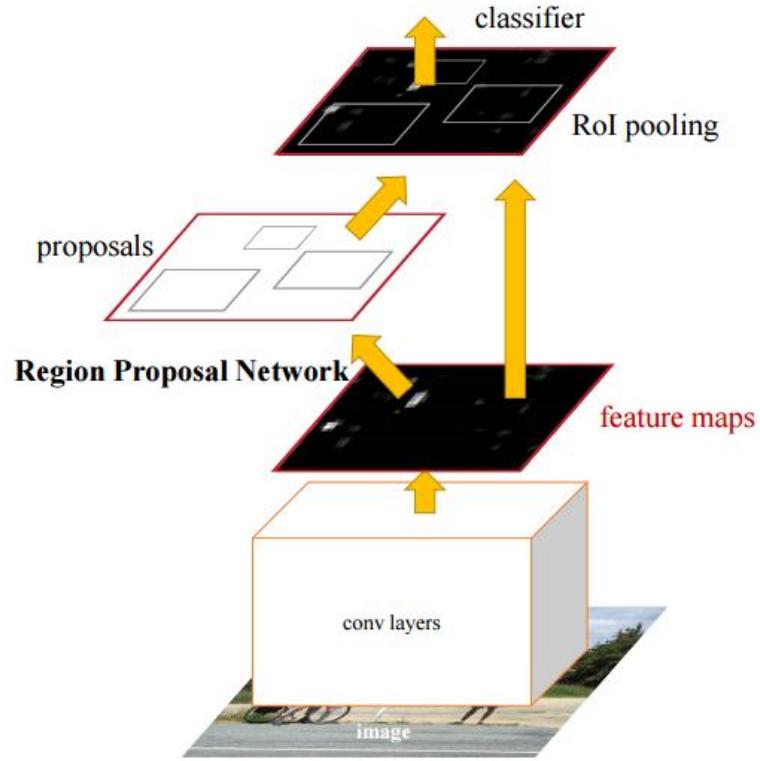


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the ‘attention’ of this unified network.

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

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

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

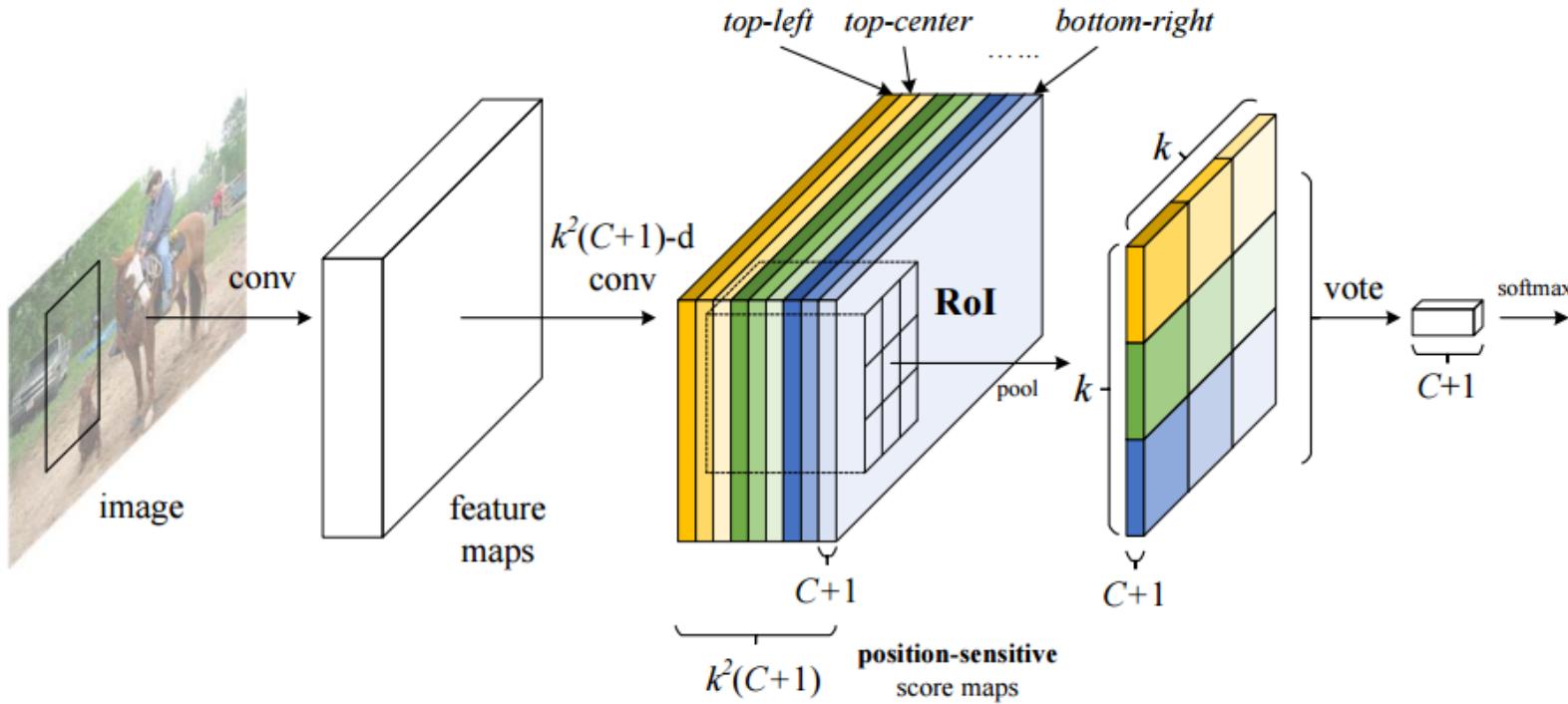


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an ROI, pooling is only performed on one of the k^2 maps (marked by different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

Two Stages Detector: RFCN

Discussion

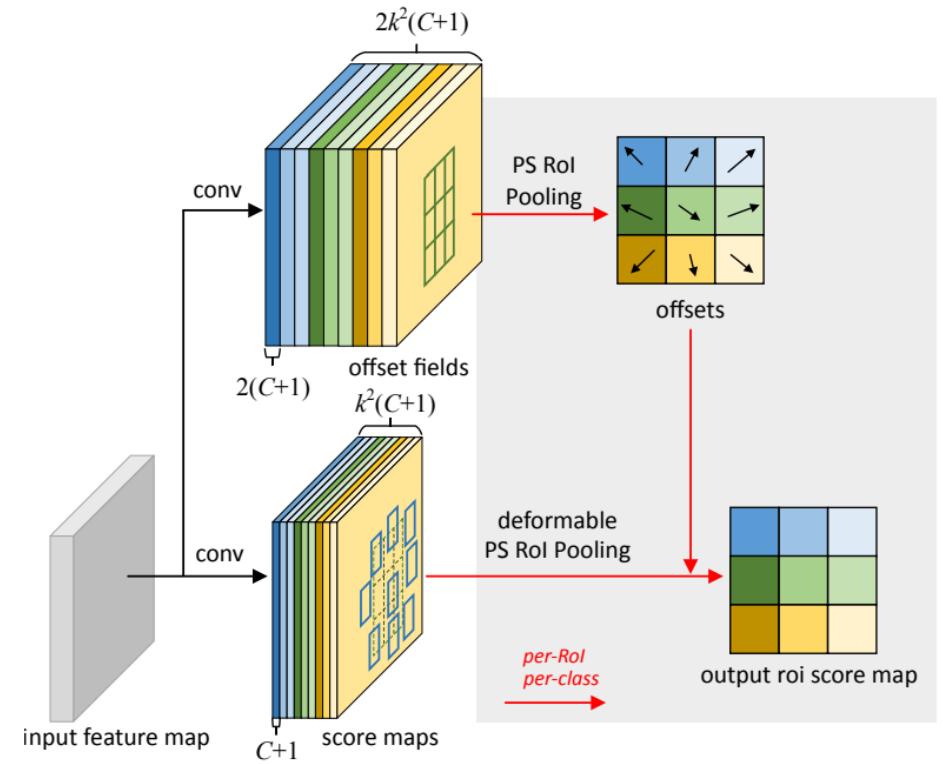
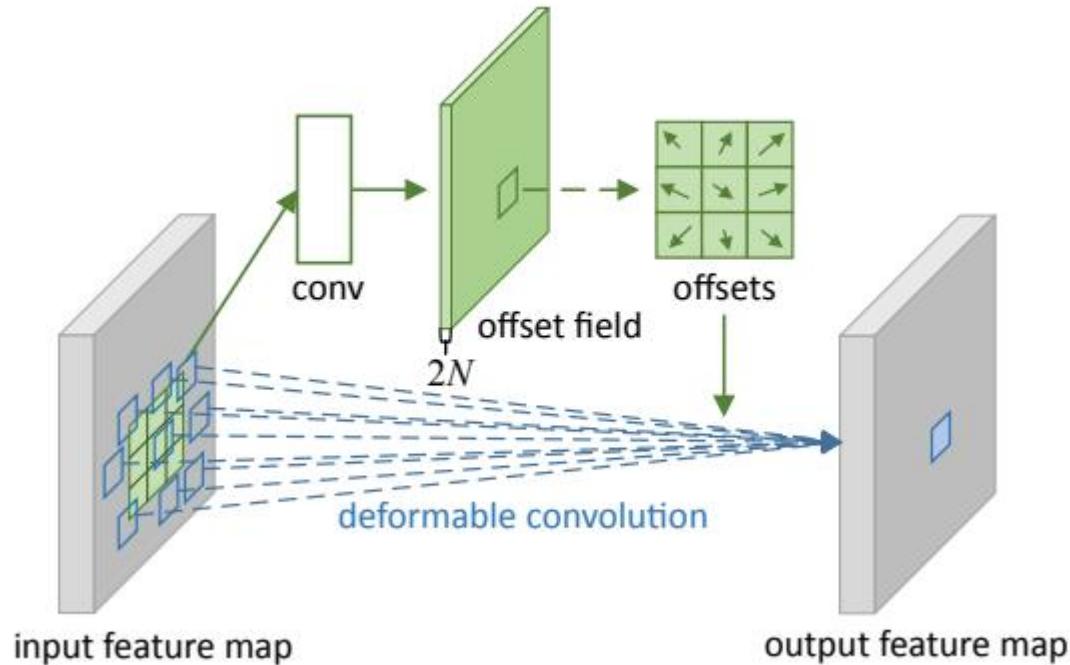
- Share convolution
 - fasterRCNN: shared Res1-4 (RPN), not shared Res5 (RCNN)
 - RFCN: shared Res1-5 (both RPN and RCNN)
- PSPooling
 - a large number of channels: $(7 \times 7 \times C) \times W \times H$
 - Problems in ROI Pooling also exist
- Fully connected vs Convolution
 - fc: global context
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

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

Two Stages Detector: RFCN -> Deformable Convolutional Networks



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.

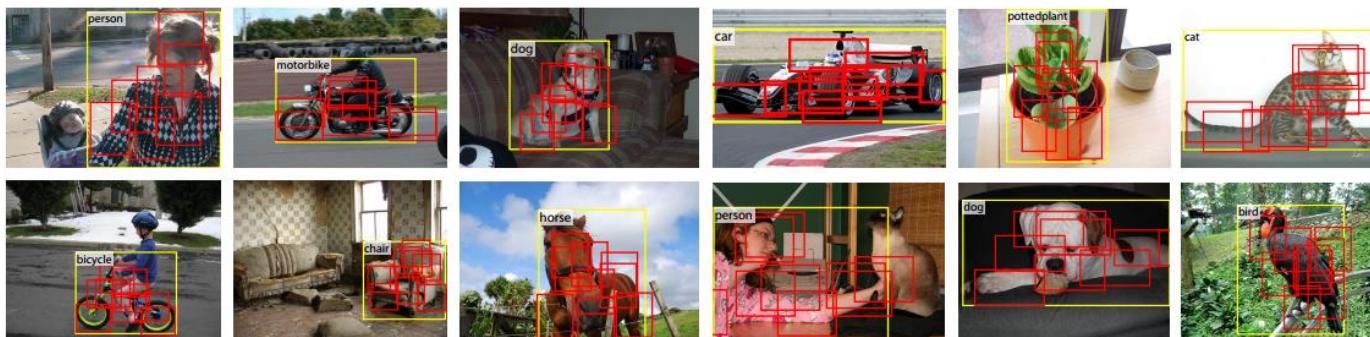


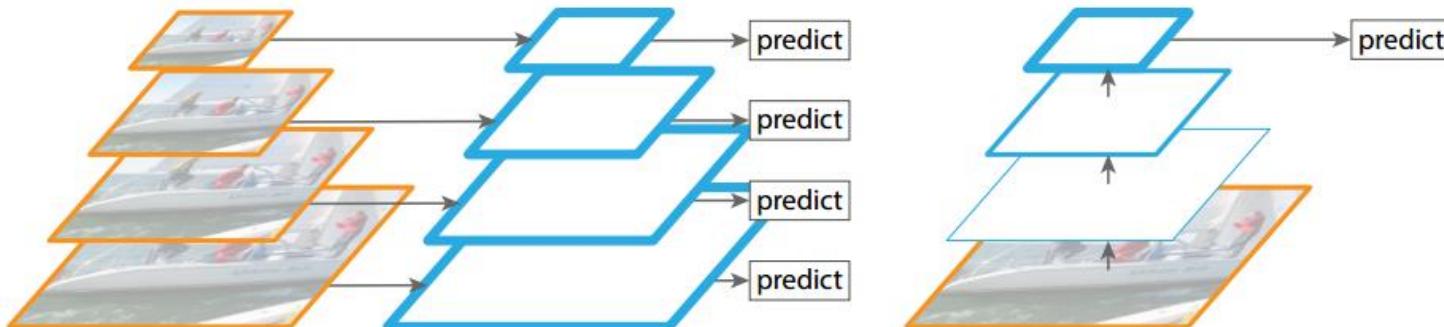
Figure 7: Illustration of offset parts in deformable (positive sensitive) ROI pooling in R-FCN [7] and 3×3 bins (red) for an input ROI (yellow). Note how the parts are offset to cover the non-rigid objects.

Two Stages Detector: RFCN -> Deformable Convolutional Networks

Discussion

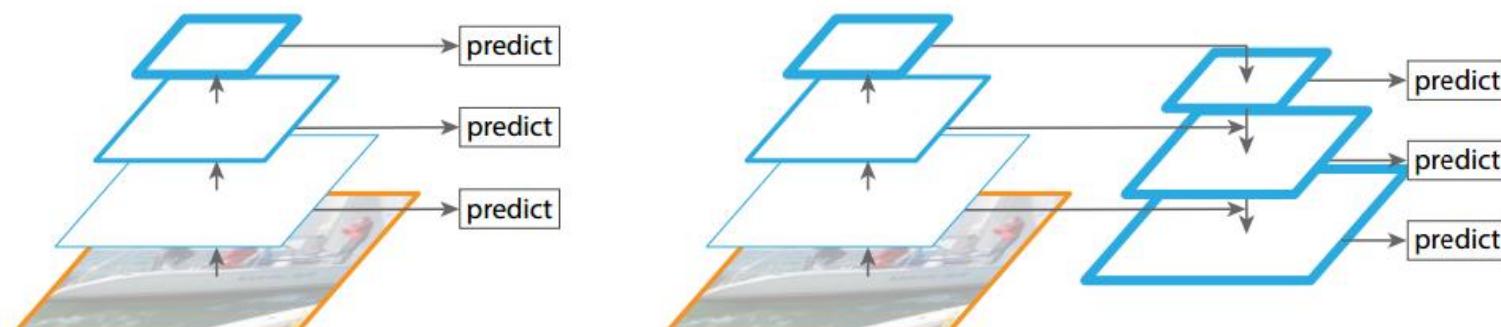
- Deformable pool is similar to ROIAlign (in Mask RCNN)
- Deformable conv
 - flexible to learn the non-rigid objects

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



(a) Featurized image pyramid

(b) Single feature map



(c) Pyramidal feature hierarchy

(d) Feature Pyramid Network

Two Stages Detector: FPN

Discussion

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

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

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

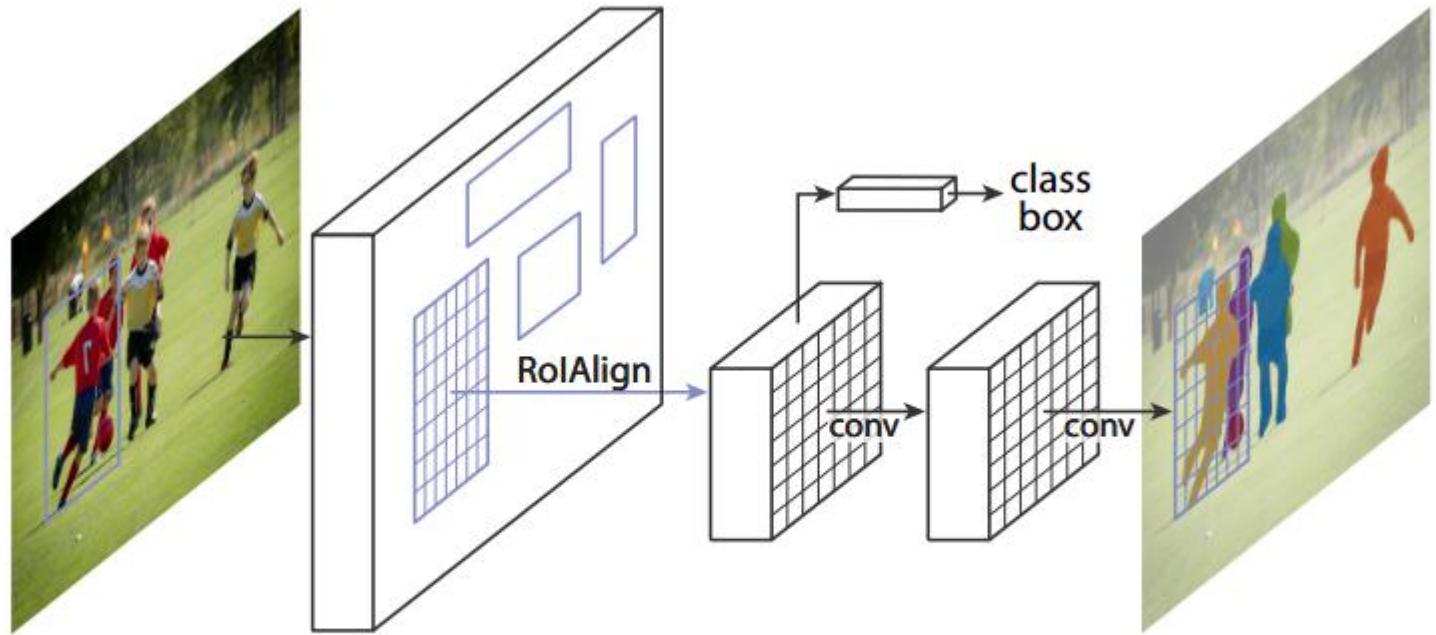


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Two Stages Detector:

RCNN -> Fast RCNN -> FasterRCNN -> FPN ->
MaskRCNN

	align?	bilinear?	agg.	AP	AP ₅₀	AP ₇₅
<i>RoIPool</i> [12]			max	26.9	48.8	26.4
<i>RoIWarp</i> [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
<i>RoIAlign</i>	✓	✓	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 AP₇₅ by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

Two Stages Detector: Mask RCNN

Discussion

- Alignment issue in ROI Pooling -> ROI Align
- Multi-task learning: detection & mask

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

Two Stages Detector: Summary

- Speed
 - RCNN -> Fast RCNN -> Faster RCNN -> RFCN
- performance
 - Divide and conquer
 - FPN
 - Deformable Pool/ROIAlign
 - Deformable Conv
 - Multi-task learning

Two Stages Detector: Discussion

FasterRCNN vs RFCN

One stage vs two Stage

MegDetection

Introduction & Demo Video

Open Problem in Detection

- FP
- NMS (detection in crowd)
- GT assignment issue
- Detection in video
 - detect & track in a network

Outline

- Detection
- **Human Keypoint**
- Conclusion

Human Keypoint Task

- Single Person Skeleton
 - Cropped RGB image -> 2d key points / 3d key points
 - Keyword: [inter-middle loss](#), [large receptive field](#), [context](#)
- Multiple-Person Skeleton
 - RGB image -> human localization & human Keypoint for each person

Single Person Skeleton: CPM

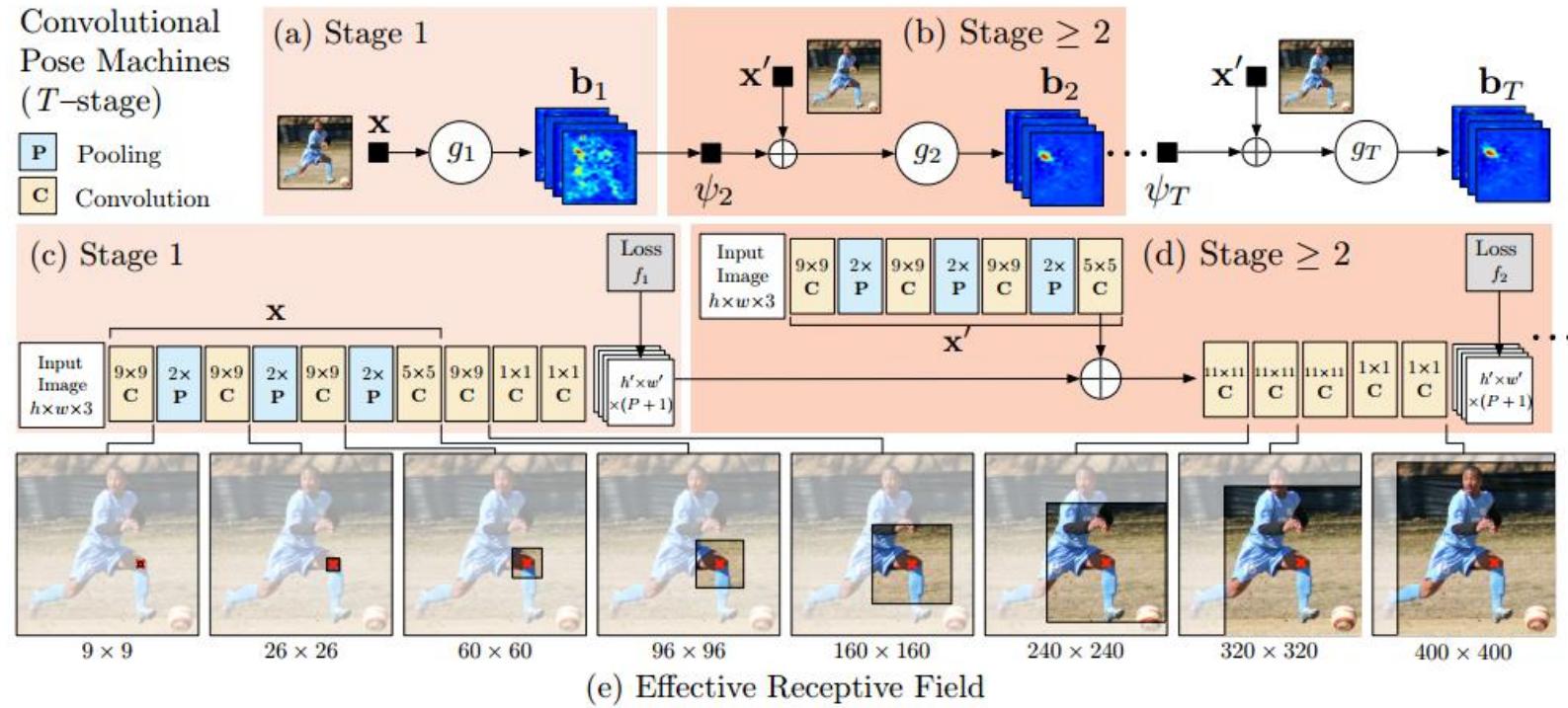
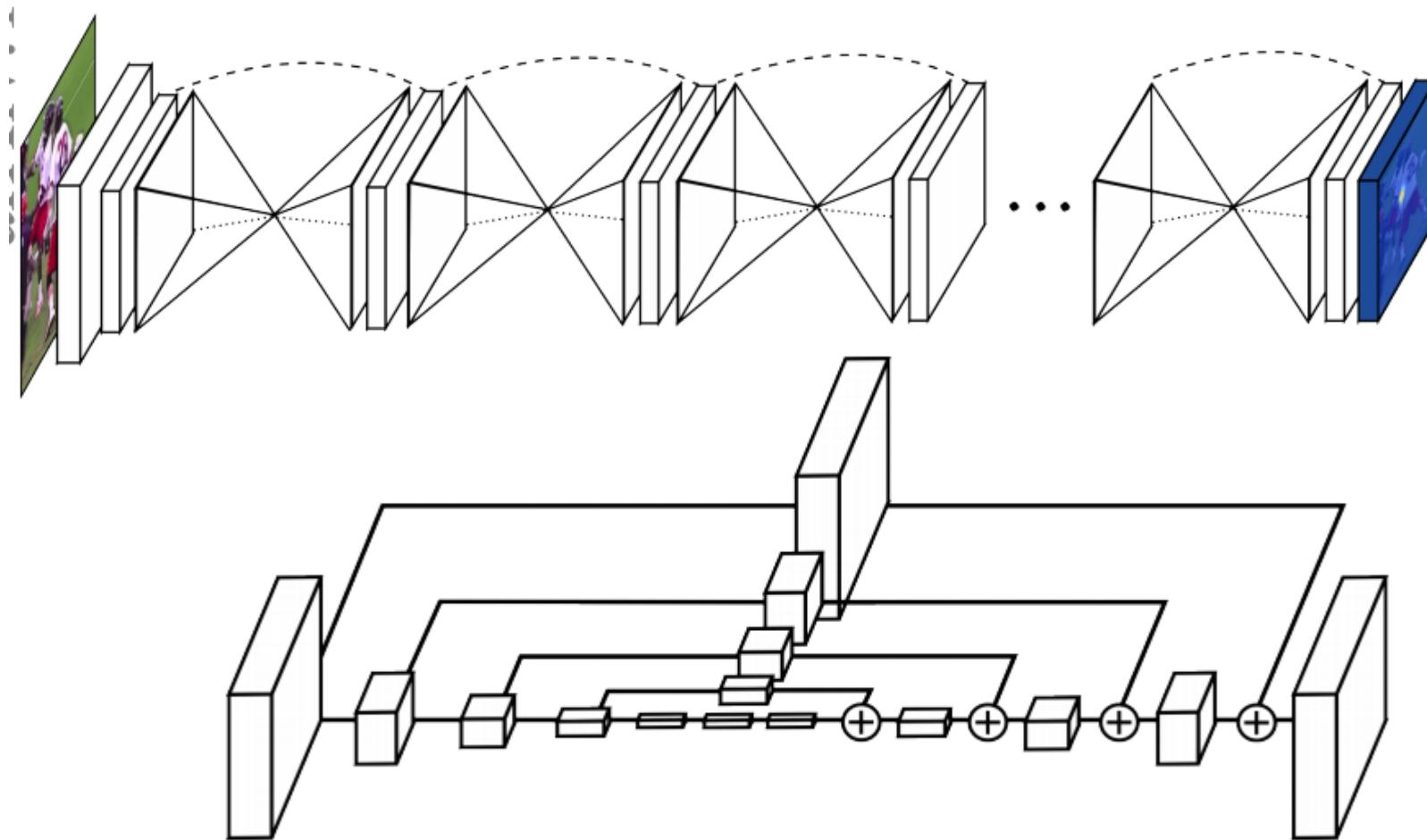


Figure 2: Architecture and receptive fields of CPMs. We show a convolutional architecture and receptive fields across layers for a CPM with any T stages. The pose machine [29] is shown in insets (a) and (b), and the corresponding convolutional networks are shown in insets (c) and (d). Insets (a) and (c) show the architecture that operates only on image evidence in the first stage. Insets (b) and (d) shows the architecture for subsequent stages, which operate both on image evidence as well as belief maps from preceding stages. The architectures in (b) and (d) are repeated for all subsequent stages (2 to T). The network is locally supervised after each stage using an intermediate loss layer that prevents vanishing gradients during training. Below in inset (e) we show the effective receptive field on an image (centered at left knee) of the architecture, where the large receptive field enables the model to capture long-range spatial dependencies such as those between head and knees. (Best viewed in color.)

Single Person Skeleton: Hourglass



Multiple-Person Skeleton

- Top Down
 - Detect -> Single person skeleton
- Bottom Up
 - Deep/Deeper Cut
 - OpenPose
 - Associative Embedding

Multiple-Person Skeleton: OpenPose

CPM + PAF



Figure 1. **Top:** Multi-person pose estimation. Body parts belonging to the same person are linked. **Bottom left:** Part Affinity Fields (PAFs) corresponding to the limb connecting right elbow and right wrist. The color encodes orientation. **Bottom right:** A zoomed in view of the predicted PAFs. At each pixel in the field, a 2D vector encodes the position and orientation of the limbs.

Multiple-Person Skeleton: OpenPose

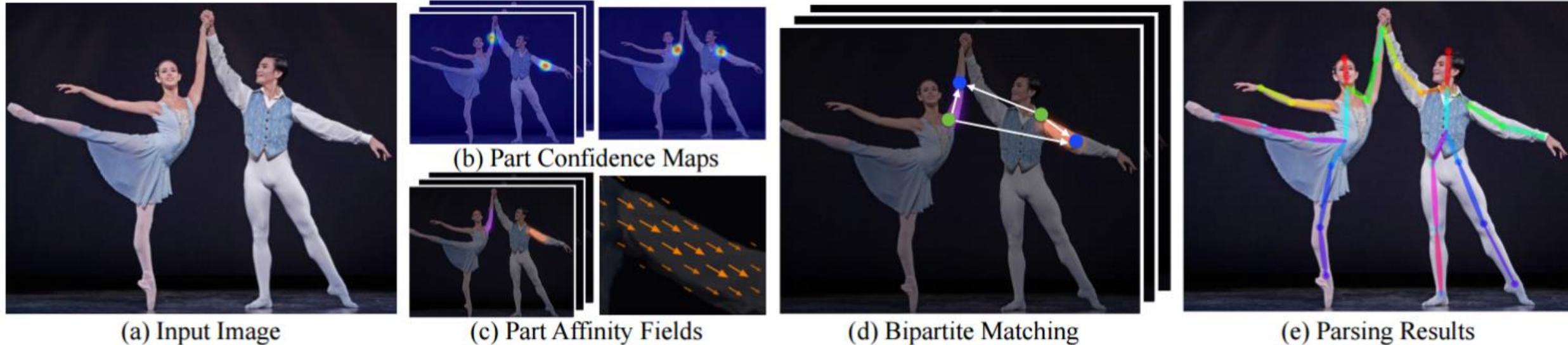


Figure 2. Overall pipeline. Our method takes the entire image as the input for a two-branch CNN to jointly predict confidence maps for body part detection, shown in (b), and part affinity fields for parts association, shown in (c). The parsing step performs a set of bipartite matchings to associate body parts candidates (d). We finally assemble them into full body poses for all people in the image (e).

<https://github.com/CMU-Perceptual-Computing-Lab/openpose>

Multiple-Person Skeleton: Associative Embedding

Hourglass + AE



Figure 1. Both multi-person pose estimation and instance segmentation are examples of computer vision tasks that require detection of visual elements (joints of the body or pixels belonging to a semantic class) and grouping of these elements (as poses or individual object instances).

Multiple-Person Skeleton: Associative Embedding

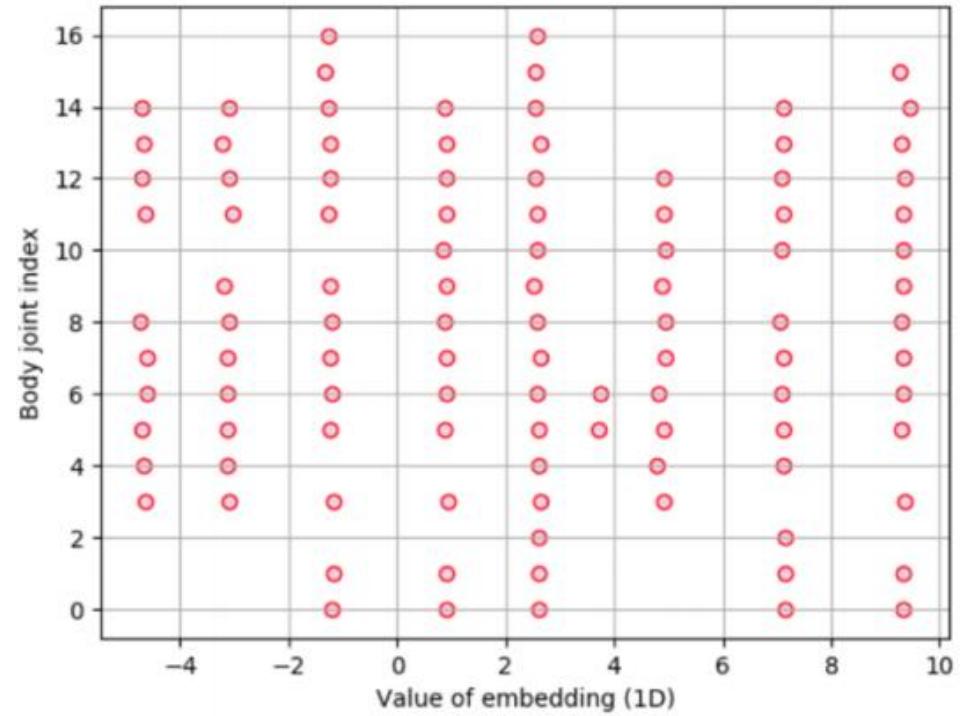


Figure 4. Tags produced by our network on a held-out validation image from the MS-COCO training set. The tag values are already well separated and decoding the groups is straightforward.

Multiple-Person Skeleton: Discussion

- **Top Down:**
 - Depends on the detector
 - Fail in the crowd case
 - Fail with partial observation
 - can detect the small-scale human
 - More computation
 - Better localization when the input-size of single person skeleton is large
- **Bottom up:**
 - Fast computational speed
 - good at localizing the human with partial observation
 - Hard to assemble human

Challenges in Skeleton

- combine top-down approaches with bottom-up approaches
- perform pose track
- handle the crowd case

MegSkeleton

Introduction and demo Video

Outline

- Detection
- Human Keypoint
- Conclusion

Conclusion

- Detection
 - One stage: Densebox, YOLO, SSD, RetinaNet
 - Two Stage: RCNN, Fast RCNN, FasterRCNN, RFCN, FPN, Mask RCNN
- Skeleton
 - Single Person Skeleton: CPM, Hourglass
 - Multi-person Skeleton
 - Top Down
 - Bottom up: Openpose, Associative Embedding

Introduction to Object Detection

Part 2

Outline

- Modern Object detectors
 - One Stage detector vs Two-stage detector
- Challenges
 - Backbone
 - Head
 - Scale
 - Batch Size
 - Crowd
- Conclusion

What is object detection?



Face

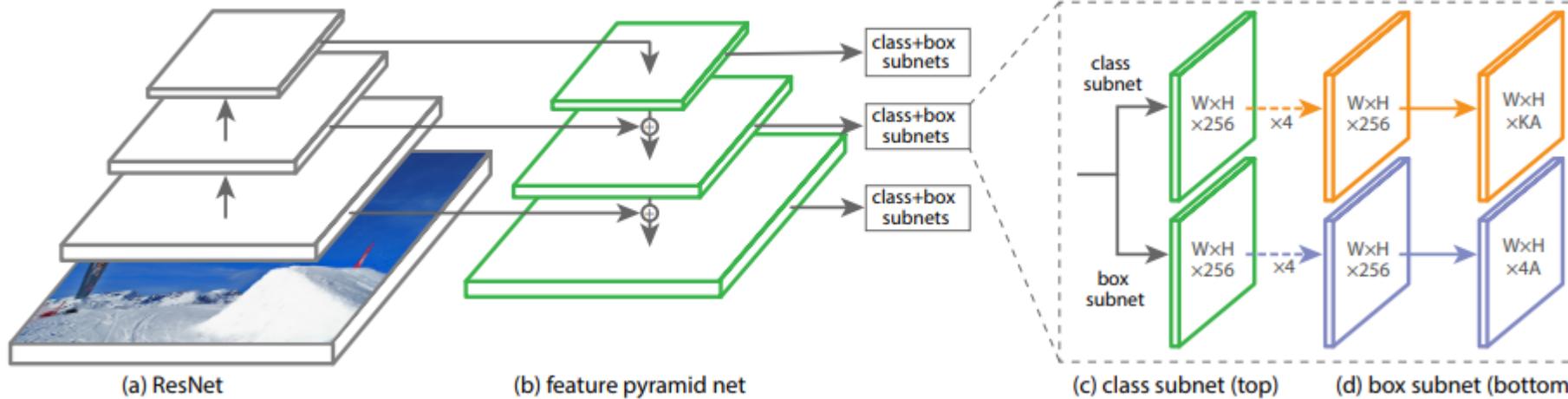
Modern Object detectors



- Modern object detectors
 - RetinaNet
 - f1-f7 for backbone, f3-f7 with 4 convs for head
 - FPN with ROIAlign
 - f1-f6 for backbone, two fc's for head
 - Recall vs localization
 - One stage detector: Recall is high but compromising the localization ability
 - Two stage detector: Strong localization ability

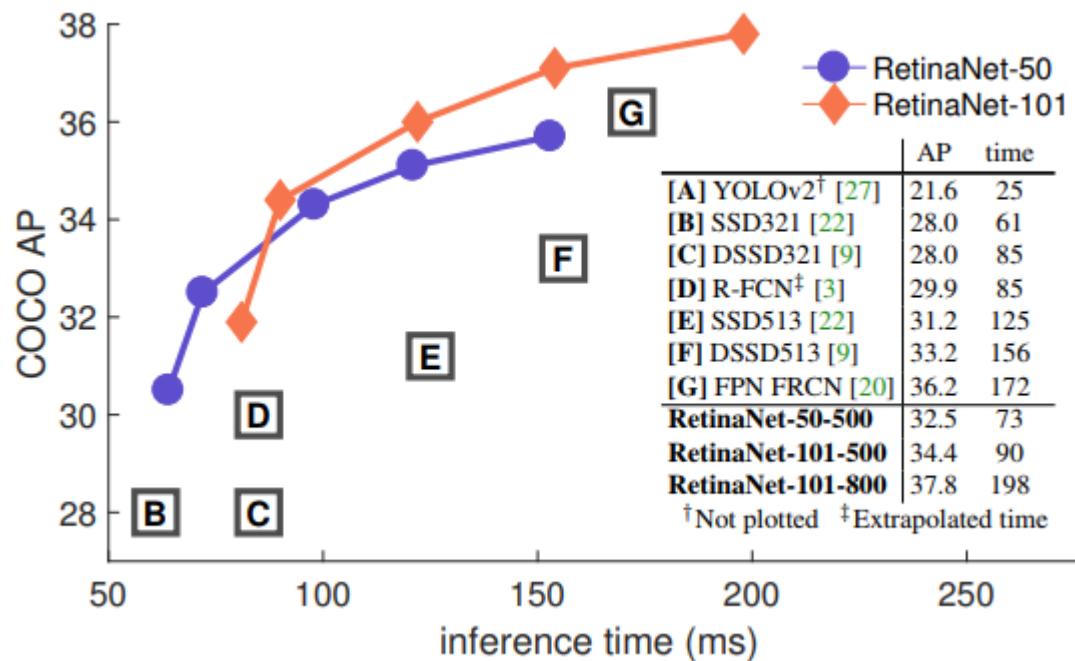
One Stage detector: RetinaNet

- FPN Structure
- Focal loss



One Stage detector: RetinaNet

- FPN Structure
- Focal loss



Two-Stage detector: FPN/Mask R-CNN

- FPN Structure
- ROIAlign

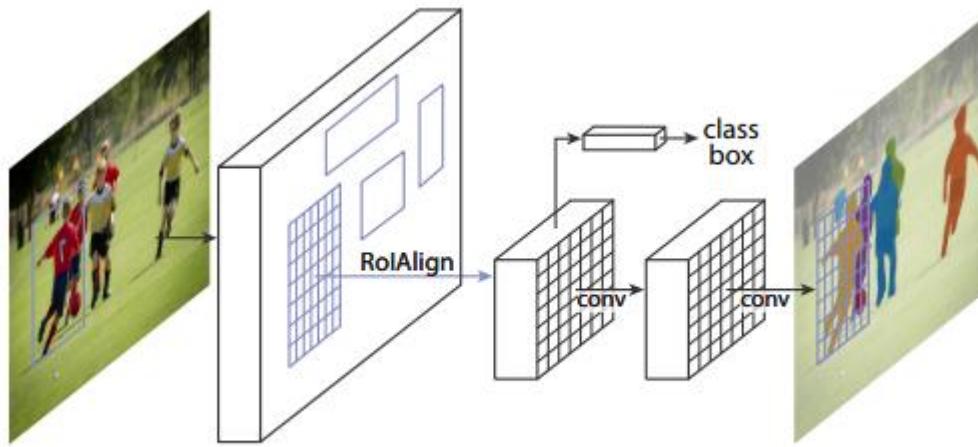


Figure 1. The **Mask R-CNN** framework for instance segmentation.

	backbone	AP ^{bb}	AP ^{bb} ₅₀	AP ^{bb} ₇₅	AP ^{bb} _S	AP ^{bb} _M	AP ^{bb} _L
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	39.8	62.3	43.4	22.1	43.2	51.2

What is next for object detection?

- The pipeline seems to be mature
- There still exists a large gap between existing state-of-arts and product requirements
- The devil is in the detail

Challenges Overview

- Backbone
- Head
- Scale
- Batch Size
- Crowd

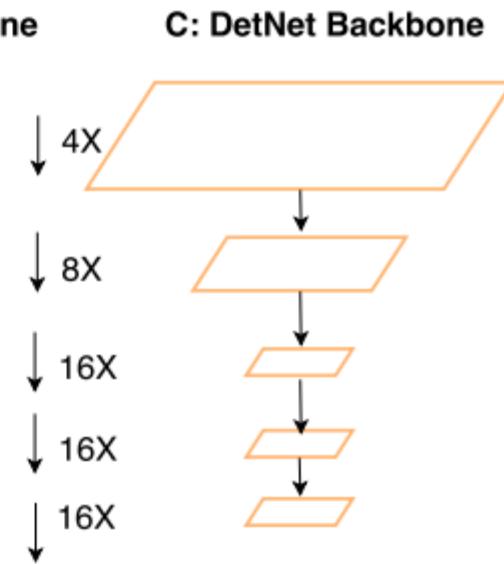
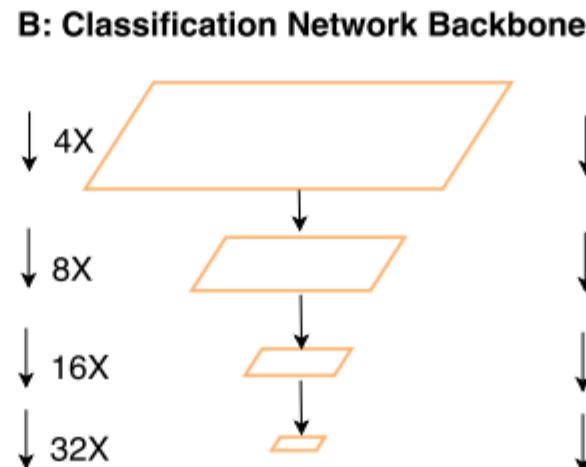
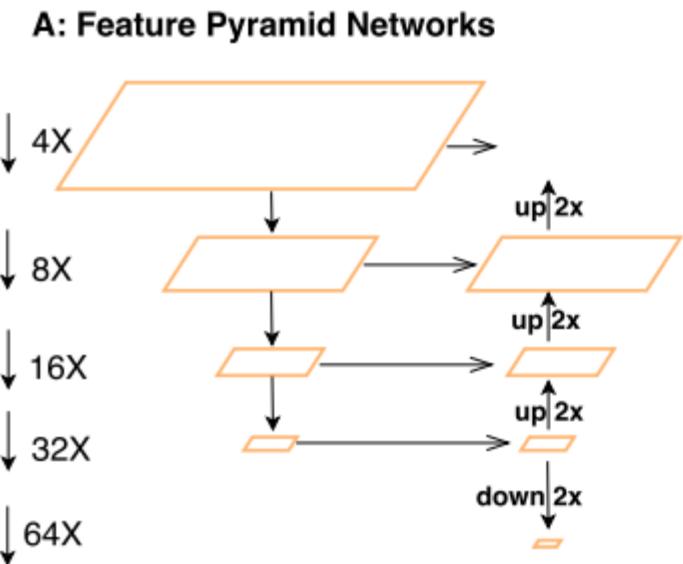


Challenges - Backbone

- Backbone network is designed for classification task but not for localization task
 - Receptive Field vs Spatial resolution
 - Only f1-f5 is pretrained but randomly initializing f6 and f7 (if applicable)

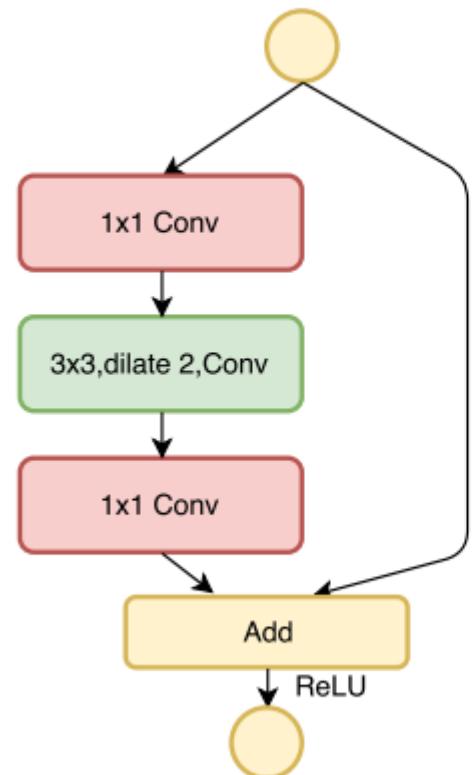
Backbone - DetNet

- DetNet: A Backbone network for Object Detection, Li etc, 2018, <https://arxiv.org/pdf/1804.06215.pdf>

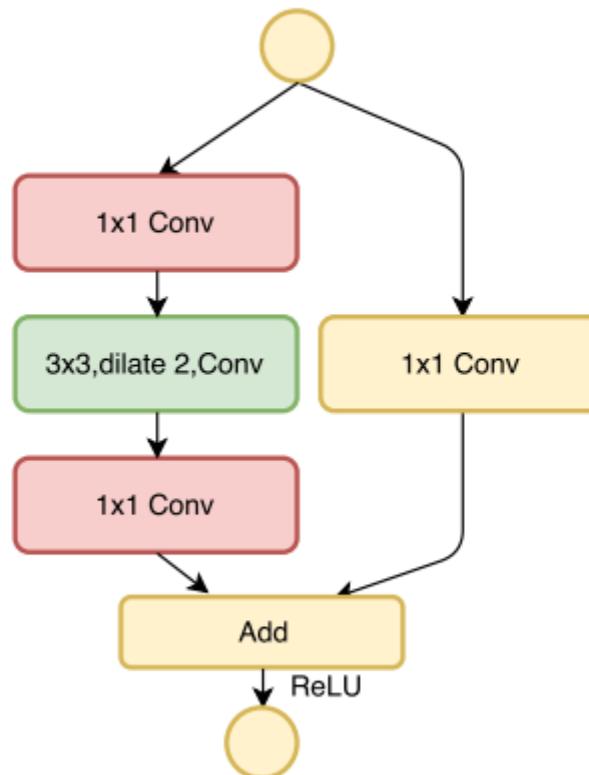


Backbone - DetNet

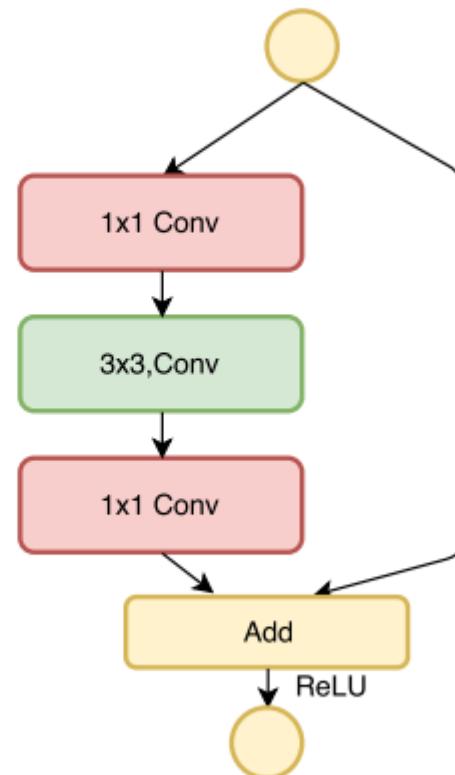
A:Dilated bottleNeck



B:Dilated bottleNeck with 1x1 conv projection

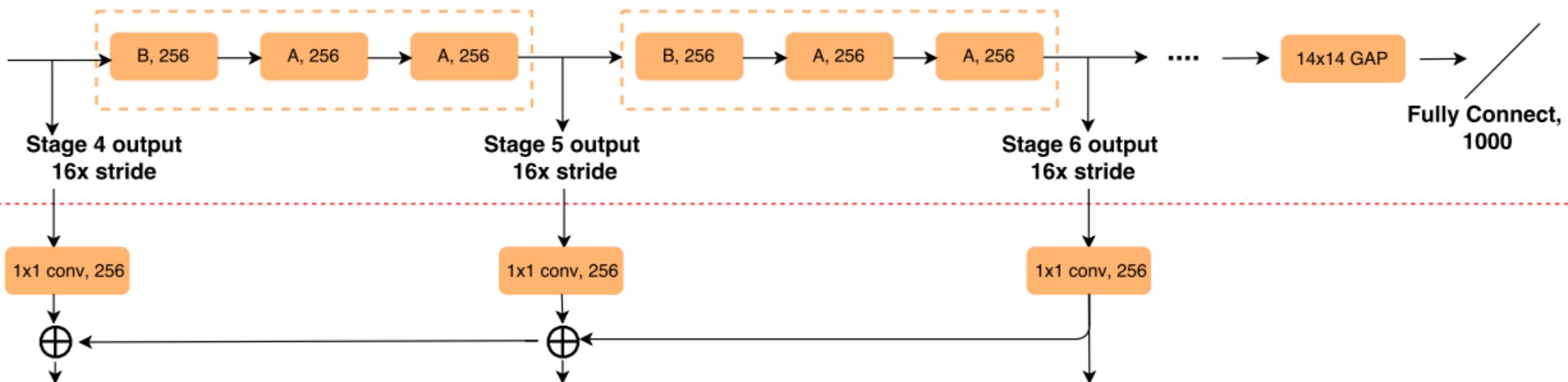


C:Original bottleNeck



Backbone - DetNet

D: DetNet Backbone



E: Feature Pyramid Structure

Backbone - DetNet

bacbone	Classification		FPN on COCO minival						FPN on COCO test-dev					
	Err	FLOPs	mAP	AP ₅₀	AP ₇₅	AP _s	AP _m	AP _l	mAP	AP ₅₀	AP ₇₅	AP _s	AP _m	AP _l
D-59	23.5	4.8G	40.2	61.7	43.7	23.9	43.2	52.0	40.3	62.1	43.8	23.6	42.6	50.0
R-62	23.4	4.7G	38.8	60.6	42.4	22.6	41.6	51.6	39.0	61.0	42.3	21.9	41.2	49.7
R-50	24.1	3.8G	37.9	60.0	41.2	22.9	40.6	49.2	38.4	60.4	41.6	22.5	40.7	47.9
D-101	23.0	7.9G	41.9	62.8	45.7	25.4	45.2	55.1	42.2	63.2	45.8	24.5	44.8	53.1
R-101	22.9	7.8G	39.9	62.0	43.7	24.1	43.4	52.0	40.3	62.5	44.0	23.3	43.1	50.6

Table 1. Comparison of ‘D’ DetNet and ‘R’ ResNet. We report both results on ImageNet classification (Top1 Error) and FPN COCO detection. Results validate that DetNet is more suitable for object detection. Keeping same model size, DetNet consistently outperform ResNet.

Backbone - DetNet

Models	scales	mAP	AP ₅₀	AP ₆₀	AP ₇₀	AP ₈₀	AP ₈₅
<i>ResNet-50</i>	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
<i>DetNet-59</i>	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	scales	mAR	AR ₅₀	AR ₆₀	AR ₇₀	AR ₈₀	AR ₈₅
<i>ResNet-50</i>	over all scales	52.8	80.5	74.7	64.3	46.8	34.2
	small	35.5	60.0	53.8	43.3	28.7	18.7
	middle	56.0	84.9	79.2	68.7	50.5	36.2
	large	67.0	95.0	90.9	80.3	63.1	50.2
<i>DetNet-59</i>	over all scales	56.1	83.1	77.8	67.6	51.0	38.9
	small	39.2	66.4	59.4	47.3	29.5	19.6
	middle	59.5	87.4	82.5	72.6	55.6	41.2
	large	70.1	95.4	91.8	82.9	69.1	56.3

Backbone - DetNet

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]		33.2	53.3	35.2	13.0	35.4	51.1
Faster R-CNN +++ [11]		34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN G-RMI ² [38]		34.7	55.5	36.7	13.5	38.1	52.0
RetinaNet [4]		39.1	59.1	42.3	21.8	42.7	50.2
FPN [33]		37.3	59.6	40.3	19.8	40.2	48.8
FPN		40.3	62.1	43.8	23.6	42.6	50.0

Models	Backbone	mAP	AP ₅₀	AP ₇₅	AP _s	AP _m	AP _l
MNC [39]	ResNet-101	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [40] + OHEM [41]		29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [40] +OHEM		33.6	54.5	-	-	-	-
Mask R-CNN [33]		35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN		37.1	60.0	39.6	18.6	39.0	51.3

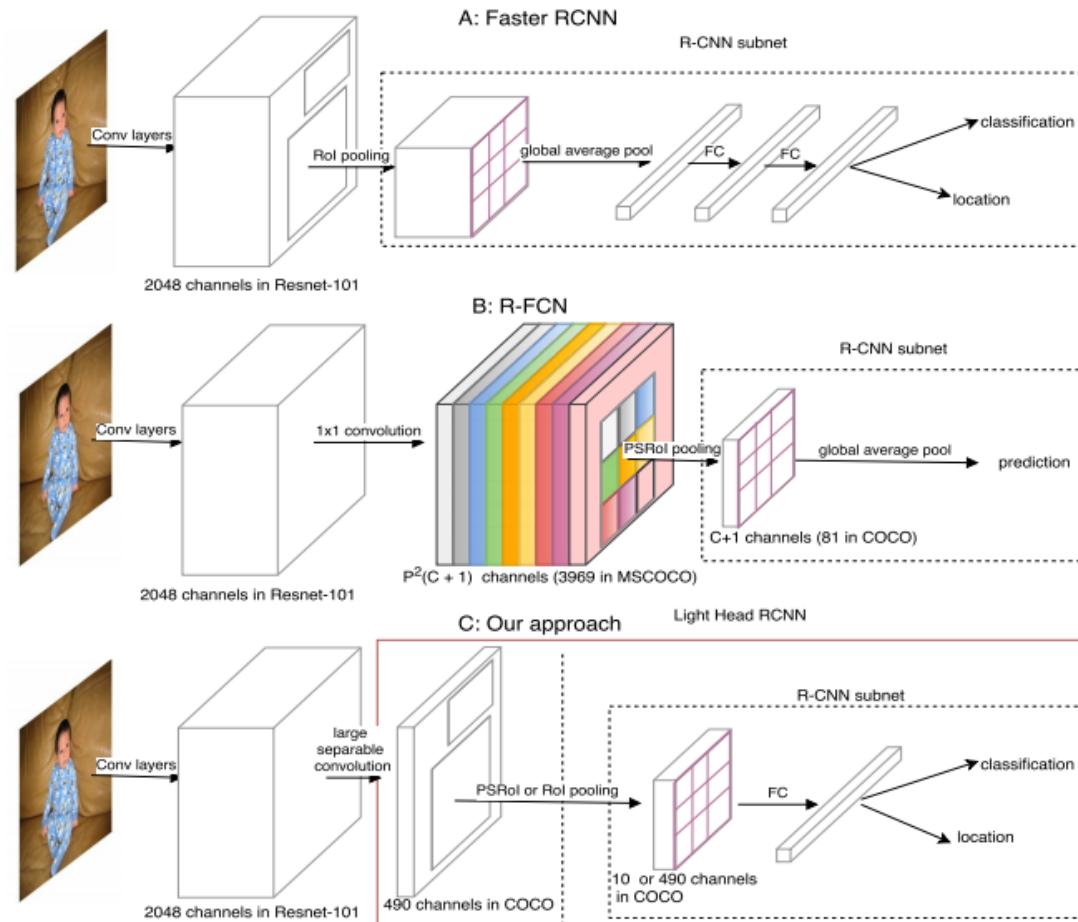
Challenges - Head

- Speed is significantly improved for the two-stage detector
 - RCNN -> Fast RCNN -> Faster RCNN -> RFCN
- How to obtain efficient speed as one stage detector like YOLO, SSD?
 - Small Backbone
 - Light Head

Head – Light head RCNN

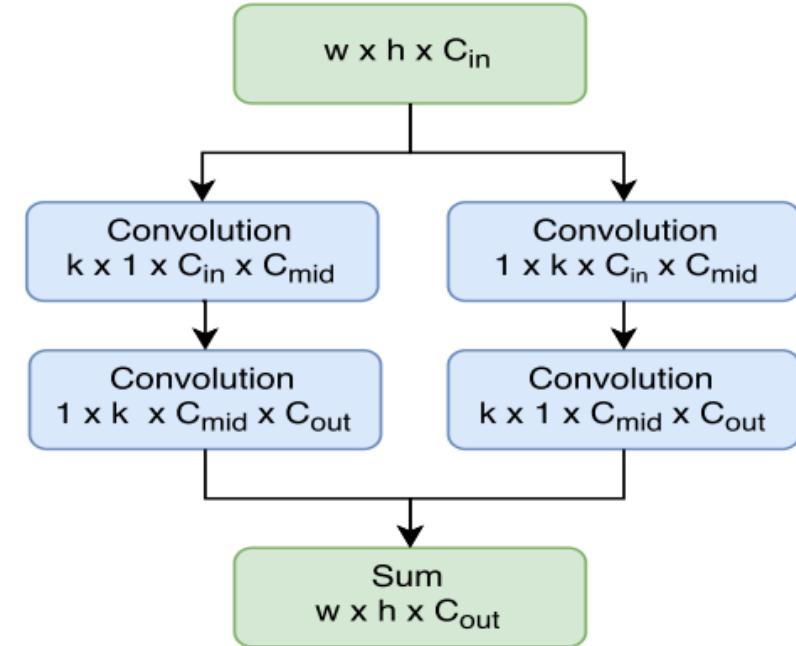
- Light-Head R-CNN: In Defense of Two-Stage Object Detector, 2017,
<https://arxiv.org/pdf/1711.07264.pdf>

Code: https://github.com/zengarden/light_head_rcnn



Head – Light head RCNN

- Backbone
 - L: Resnet101
 - S: Xception145
- Thin Feature map
 - L: $C_{\text{mid}} = 256$
 - S: $C_{\text{mid}} = 64$
 - $C_{\text{out}} = 10 * 7 * 7$
- R-CNN subnet
 - A fc layer is connected to the PS ROI pool/Align

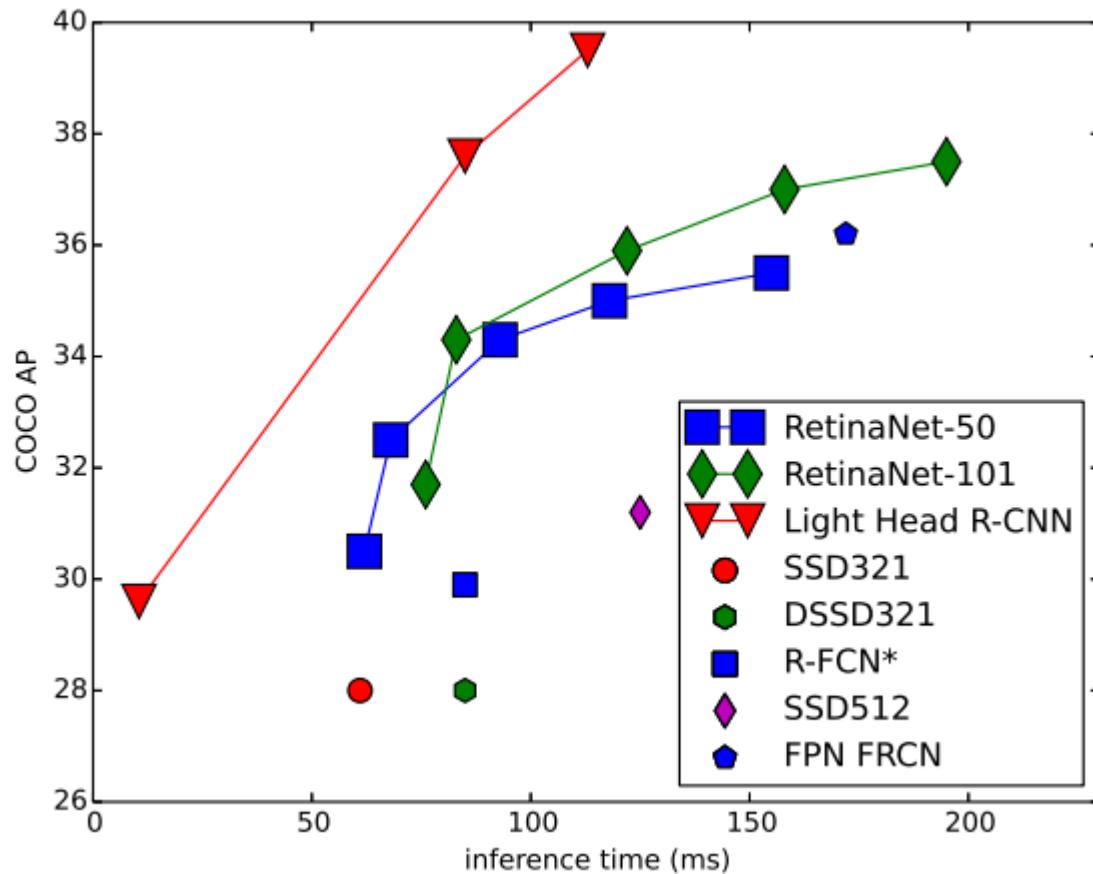


Head – Light head RCNN

Models	mAP@[0.5:0.95]
baseline Faster R-CNN	35.6
B2 (R-FCN)	35.2
+Large Kernel	35.9
+Large Kernel and Light-Head R-CNN	37.7

method	test size shorter edge/max size	feature pyramid	align	mAP@[0.5:0.95]	AP _s	AP _m	AP _l
R-FCN [17]	600/1000			32.1	12.8	34.9	46.1
Faster R-CNN (2fc)	600/1000			30.3	9.9	32.2	47.4
Deformable [3]	600/1000		✓	34.5	14.0	37.7	50.3
G-RMI [13]	600/1000			35.6	-	-	-
FPN [19]	800/1200	✓		36.2	18.2	39.0	48.2
Mask R-CNN [7]	800/1200	✓	✓	38.2	20.1	41.1	50.2
RetinaNet [20]	800/1200	✓		37.8	20.2	41.1	49.2
RetinaNet ms-train [20]	800/1200	✓		39.1	21.8	42.7	50.2
Light head R-CNN	800/1200		✓	39.5	21.8	43.0	50.7
Light head R-CNN ms-train	800/1200		✓	40.8	22.7	44.3	52.8
Light head R-CNN	800/1200	✓	✓	41.5	25.2	45.3	53.1

Head – Light head RCNN



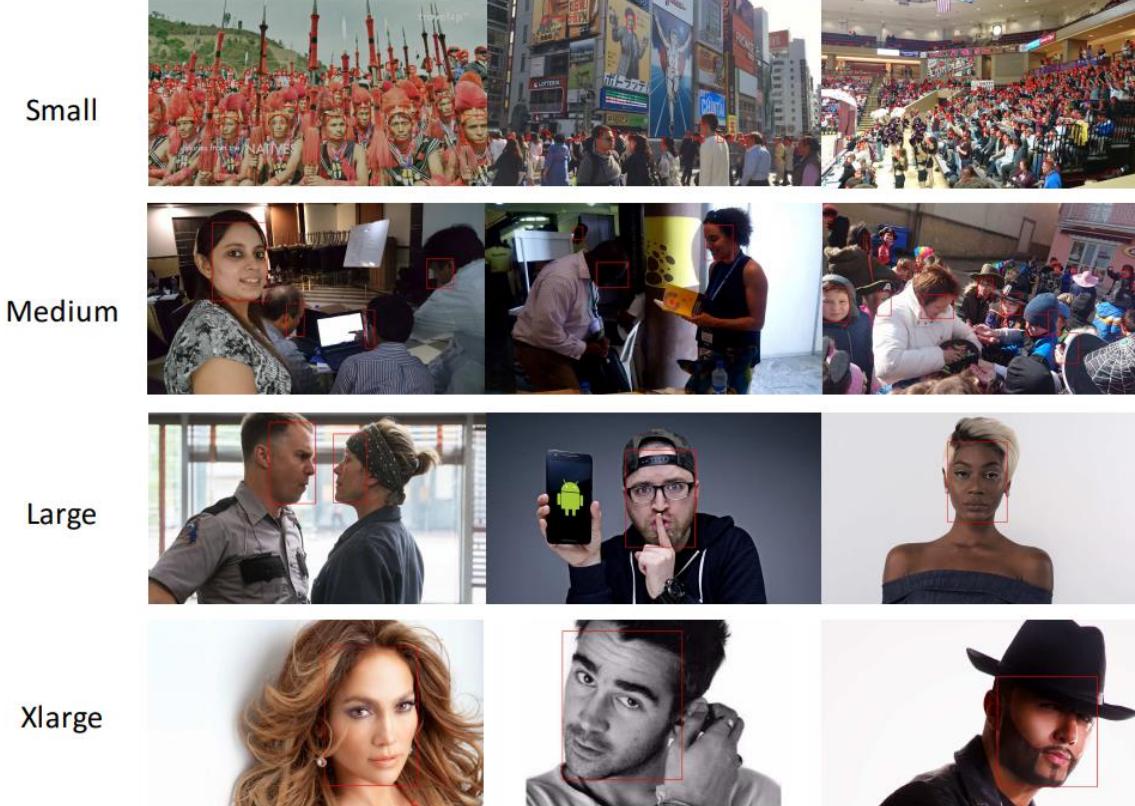
Model	backbone	test size short/max edge	speed (fps)	mAP @[0.5:0.95]
YOLO V2	Darknet	448/448	40	21.6
SSD	VGG	300/300	58	25.1
SSD	Resnet101	300/300	16	28.0
SSD	Resnet101	500/500	8	31.2
DSSD [4]	Resnet101	300/300	8	28.0
DSSD	Resnet101	500/500	6	33.2
R-FCN	Resnet101	600/1000	11	29.9
DeNet	Resnet34	512/512	83	29.4
Light Head R-CNN	xception*	800/1200	95	31.5
Light Head R-CNN	xception*	700/1100	102	30.7

Head – Light head RCNN

- Mobile Version
 - A technique report will be released soon
 - The first two-stage detector on mobile devices

Challenges - Scale

- Scale variations is extremely large for object detection



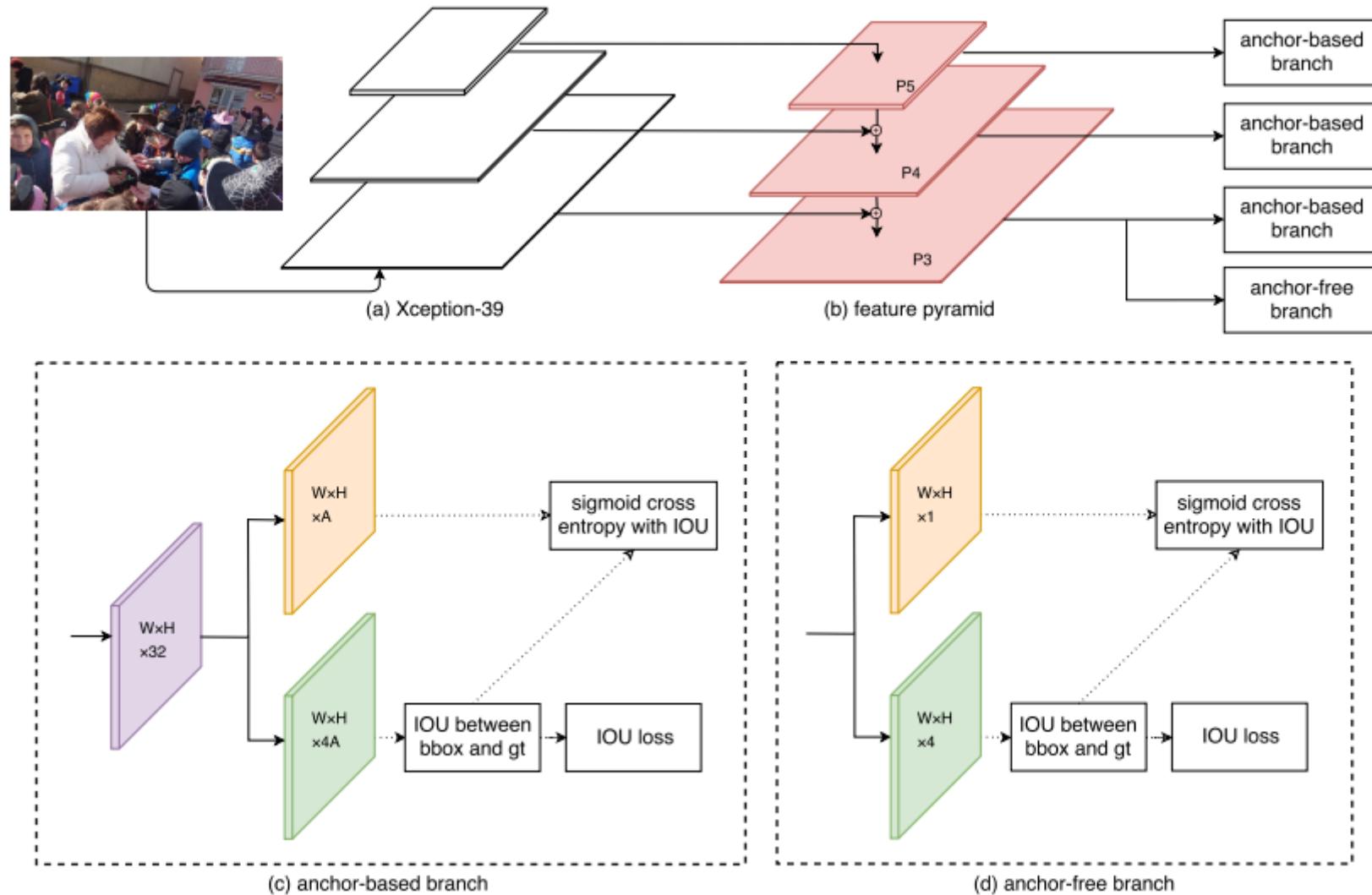
Challenges - Scale

- Scale variations is extremely large for object detection
- Previous works
 - Divide and Conquer: SSD, DSSD, RON, FPN, ...
 - Limited Scale variation
 - Scale Normalization for Image Pyramids, Singh etc, CVPR2018
 - Slow inference speed
- How to address extremely large scale variation without compromising inference speed?

Scale - SFace

- SFace: An Efficient Network for Face Detection in Large Scale Variations, 2018, <http://cn.arxiv.org/pdf/1804.06559.pdf>
 - Anchor-based:
 - Good localization for the scales which are covered by anchors
 - Difficult to address all the scale ranges of faces
 - Anchor-free:
 - Able to cover various face scales
 - Not good for the localization ability

Scale - SFace



Scale - 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

BaseNet	P3-P5 layer	re-score	Anchor-based Branch	Anchor-free Branch	AP(< 32)	AP(32 – 256)	AP(256 – 512)	AP(> 1024)	AP(all)
RetinaNet					1.22	54.62	74.51	47.74	53.34
RetinaNet	✓				29.23	49.72	0.00	0.00	32.73
UnitBox					3.27	61.09	81	50.97	63.82
SFace	✓		✓		26.86	46.51	0.00	0.00	30.72
SFace	✓			✓	3.41	57.35	77.80	53.82	61.49
SFace	✓		✓	✓	3.51	57.38	77.81	53.80	61.60
SFace	✓	✓	✓		23.62	48.03	0.00	0.00	31.50
SFace	✓	✓		✓	4.07	62.93	81.38	60.48	64.30
SFace	✓	✓	✓	✓	6.70	63.09	81.35	60.48	65.39

Scale - SFace

Method	AP(easy)	AP (medium)	AP(hard)
ACF [31]	69.5	58.8	29.0
Faceness [32]	71.6	60.4	31.5
LDCF+ [33]	79.7	77.2	56.4
MT-CNN [23]	85.1	82.0	60.7
CMS-RCNN [34]	90.2	87.4	64.3
ScaleFaces [35]	86.7	86.6	76.4
SFace	89.1	87.9	80.4

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

Scale - SFace

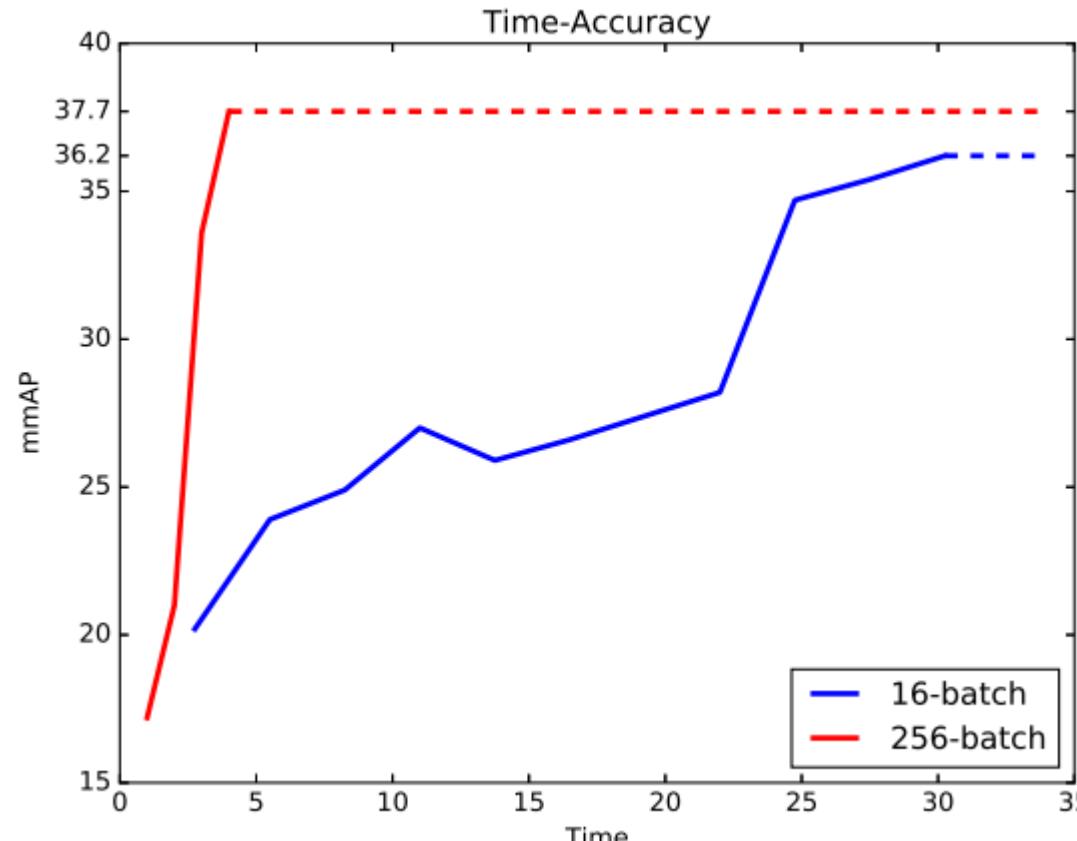
- Summary:
 - Integrate anchor-based and anchor-free for the scale issue
 - A new benchmark for face detection with large scale variations: 4K Face

Challenges - Batchsize

- Small mini-batchsize for general object detection
 - 2 for R-CNN, Faster RCNN
 - 16 for RetinaNet, Mask RCNN
- Problem with small mini-batchsize
 - Long training time
 - Insufficient BN statistics
 - Inbalanced pos/neg ratio

Batchsize – MegDet

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



Batchsize – MegDet

- Techniques
 - Learning rate warmup
 - Cross-GPU Batch Normalization

name	mmAP	mmAR
DANet	45.7	62.7
Trimps-Soushen+QINU	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

Challenges - Crowd

- NMS is a post-processing step to eliminate multiple responses on one object instance
 - Reasonable for mild crowdness like COCO and VOC
 - Will Fail in the case when the objects are in a crowd



Figure 1. Illustrative examples from different human dataset benchmarks. The images inside the green, yellow, blue boxes are from the COCO [17], Caltech [6], and CityPersons [31] datasets, respectively. The images from the second row inside the red box are from our CrowdHuman benchmark with full body, visible body, and head bounding box annotations for each person.

Challenges - Crowd

- A few works have been devoted to this topic
 - Softnms, Bodla etc, ICCV 2017, <http://www.cs.umd.edu/~bharat/snms.pdf>
 - Relation Networks, Hu etc, CVPR 2018,
<https://arxiv.org/pdf/1711.11575.pdf>
- Lacking a good benchmark for evaluation in the literature

Crowd - CrowdHuman

- CrowdHuman: A Benchmark for Detecting Human in a Crowd, 2018,
<https://arxiv.org/pdf/1805.00123.pdf>,
<http://www.crowdhuman.org/>
 - A benchmark with Head, Visible Human, Full body bounding-box
 - Generalization ability for other head/pedestrian datasets
 - **Crowdness**

Crowd - CrowdHuman

	Caltech	KITTI	CityPersons	COCOPersons	CrowdHuman
# images	42,782	3,712	2,975	64,115	15,000
# persons	13,674	2,322	19,238	257,252	339,565
# ignore regions	50,363	45	6,768	5,206	99,227
# person/image	0.32	0.63	6.47	4.01	22.64
# unique persons	1,273	< 2,322	19,238	257,252	339,565

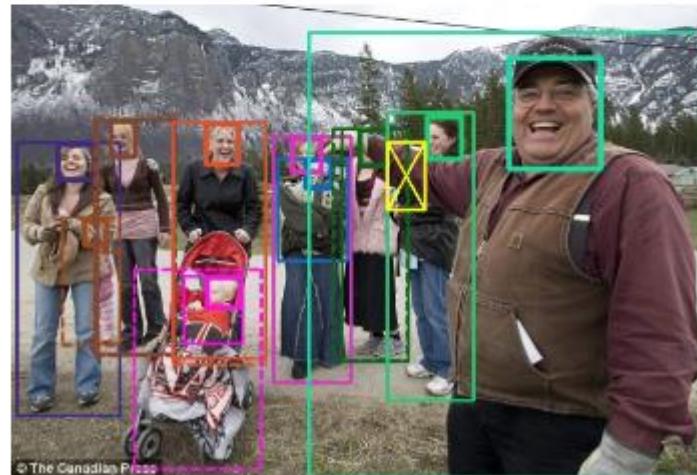
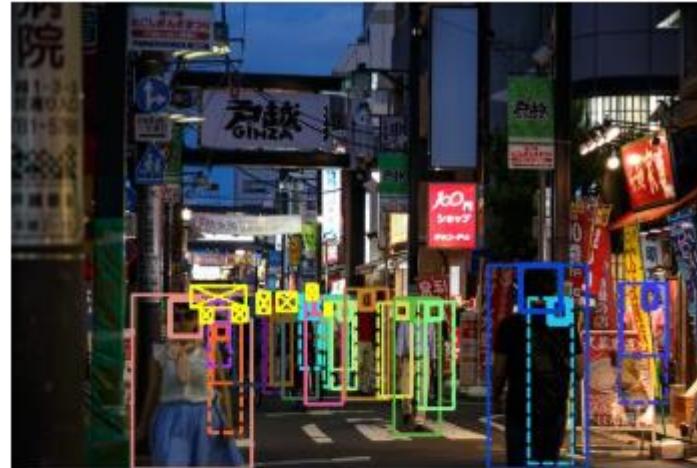
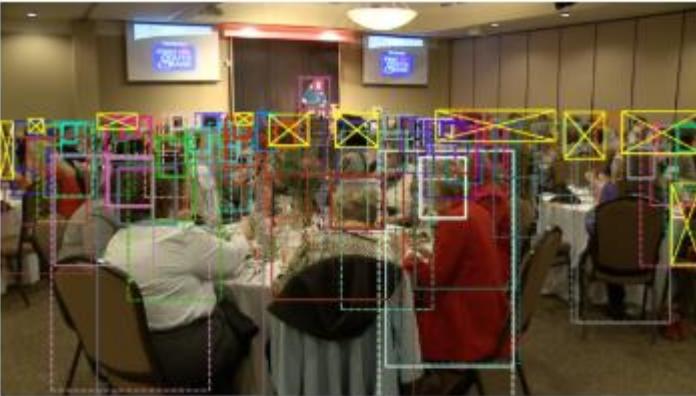
pair/img	Cal	City	COCO	CrowdHuman
iou>0.3	0.06	0.96	0.13	9.02
iou>0.4	0.03	0.58	0.05	4.89
iou>0.5	0.02	0.32	0.02	2.40
iou>0.6	0.01	0.17	0.01	1.01
iou>0.7	0.00	0.08	0.00	0.33
iou>0.8	0.00	0.02	0.00	0.07
iou>0.9	0.00	0.00	0.00	0.01

pair/img	Cal	City	COCO	CrowdHuman
iou>0.1	0.02	0.30	0.02	8.70
iou>0.2	0.00	0.11	0.00	2.09
iou>0.3	0.00	0.04	0.00	0.51
iou>0.4	0.00	0.01	0.00	0.12
iou>0.5	0.00	0.00	0.00	0.03

Table 5. Comparison of high-order overlaps among three human instances.

Table 4. Comparison of pair-wise overlap between two human instances.

Crowd-CrowdHuman



Crowd-CrowdHuman

- Generalization
 - Head
 - Pedestrian
 - COCO

Table 11. Experimental results on CityPerson

Train-set	Recall	AP	mMR
Caltech	87.21	65.87	45.52
CityPersons	97.97	94.35	14.81
CrowdHuman	98.73	98.10	21.18
Crowd⇒City	97.78	95.58	10.67
CityPersons [31]	-	-	14.8
Repulsion [26]	-	-	13.2

Table 10. Experimental results on Caltech dataset.

Train-set	Recall	AP	mMR
Caltech	99.76	89.95	10.08
CityPersons	99.05	85.81	14.69
CrowdHuman	99.88	90.58	8.81
Crowd⇒Calt	99.88	95.69	3.46
CityPersons⇒Calt [31]	-	-	5.1
Repulsion [26]	-	-	4.0
[18]	-	-	5.5

Table 12. Experimental results on Brainwash.

Train-set	Recall	AP	mMR
Brainwash	98.52	95.74	19.77
Crowd⇒Brain	98.66	96.15	17.24
[23]	-	78.0	-

Table 9. Experimental results on COCOPersons.

Train-set	Recall	AP	mMR
COCOPersons	95.57	83.83	41.89
Crowd⇒COCO	95.87	85.02	39.79

Conclusion

- The task of object detection is still far from solved
- Details are important to further improve the performance
 - Backbone
 - Head
 - Scale
 - Batchsize
 - Crowd
- The improvement of object detection will be a significantly boost for the computer vision industry

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>
 - BiSeNet: <https://arxiv.org/pdf/1808.00897.pdf>
- Skeleton:
 - CPN: <https://arxiv.org/pdf/1711.07319.pdf>

Introduction to Face++ Detection Team

- COCO2017 Detection/Skeleton Winner (ICCV 2017)
- ActivityNet AVA Winner (CVPR 2018)
- WAD Instance Segmentation Winner (CVPR 2018)
- WiderFace Challenge Winner (ECCV 2018)
- COCO 2018 Detection/Panoptic Seg/Skeleton Winner (ECCV 2018)
- Mapillary Panoptic Seg Winner (ECCV 2018)

广告部分

- Looking for Researchers/RSDEs/Interns
- 做“实用”的视觉技术

Email: yugang@megvii.com

Thanks