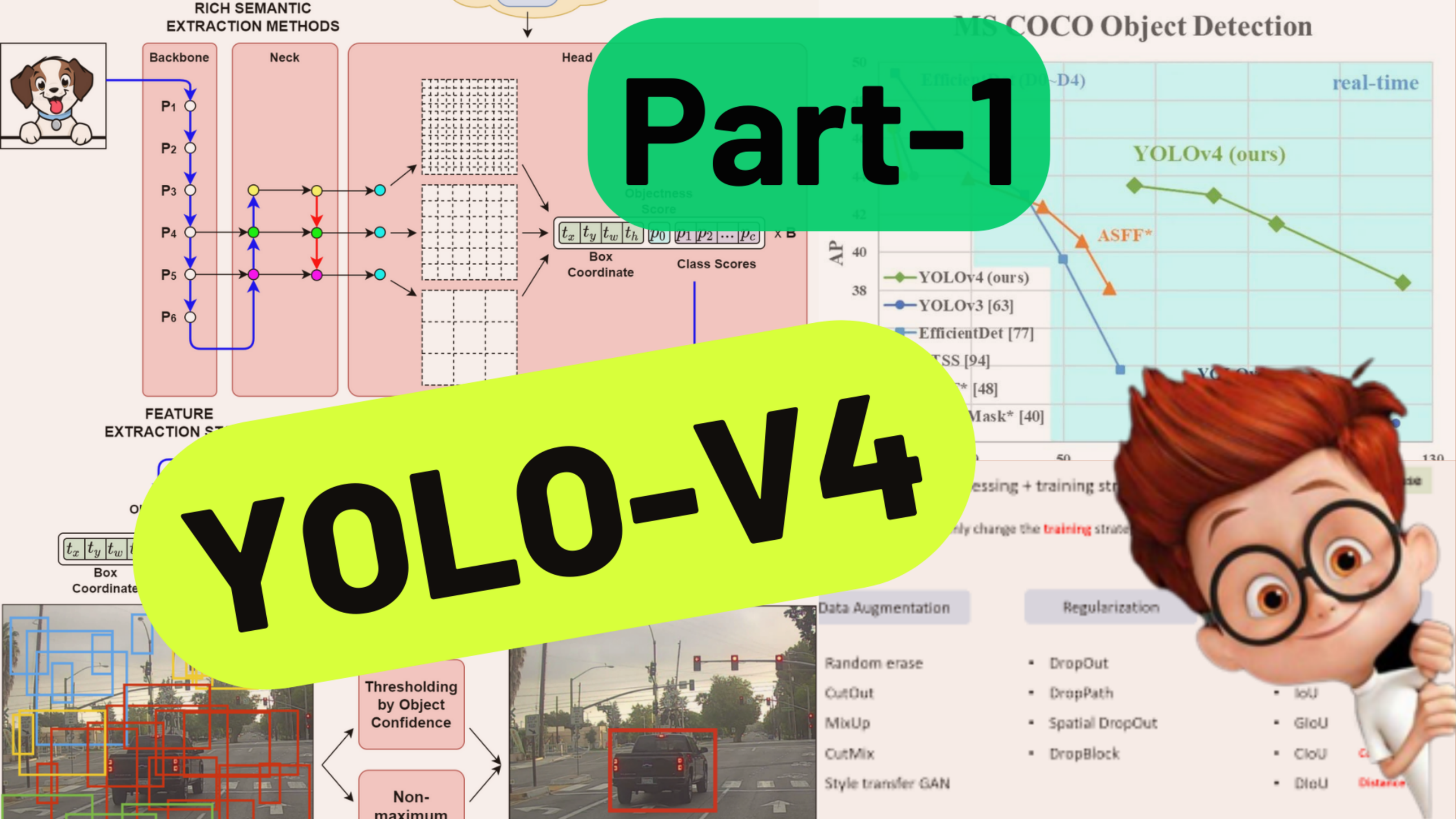
Part-1

YOLO-V4



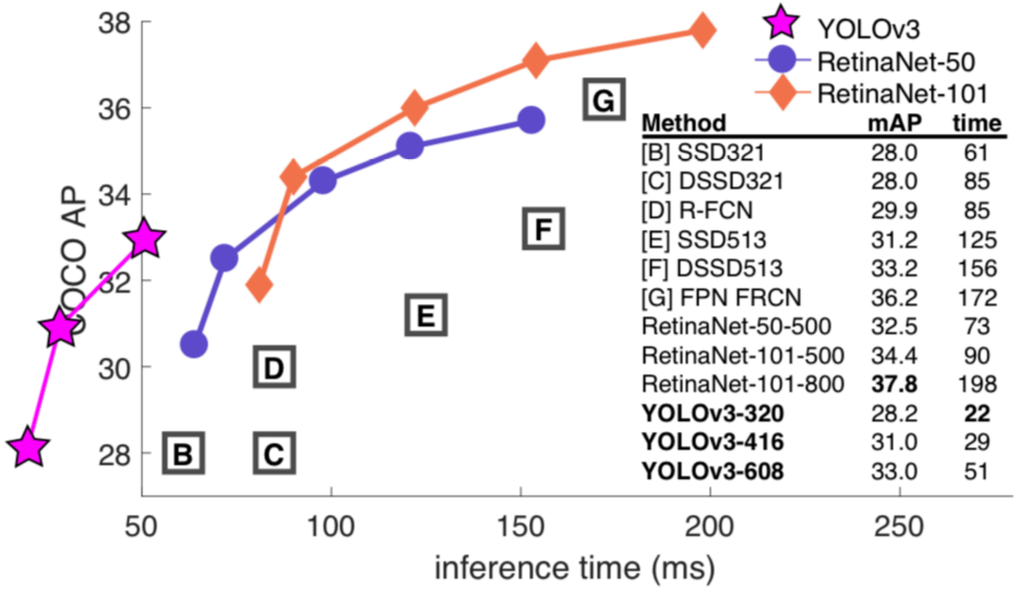
YOLOv3: An Incremental Improvement

Joseph Redmon Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

1. Introduction



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

Joe Redmon
@pjreddie

I stopped doing CV research because I saw the impact my work was having. I loved the work but the military applications and privacy concerns eventually became impossible to ignore.twitter.com/RogerGrosse/st…

Alexey

AlexeyAB

Follow

Alexey Bochkovskiy (Aleksei Bochkovskii)

4.4k followers · **4** following

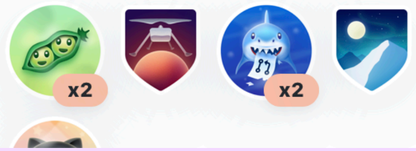
Ex Intel

Germany, Munich

https://scholar.google.com/citations?user=ljmswJ0AAAAJ

@alexeyab84

Achievements



Overview | Repositories 123 | Projects | Packages 52 | Stars 699

darknet Public

Forked from pjreddie/darknet

YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)

C · 20.3k · 7.8k

WongKinYiu/yolov7 Public

Implementation of paper - YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

Jupyter Notebook · 9.1k · 2.8k

isl-org/DPT Public

Dense Prediction Transformers

Python · 1.4k · 202

object_threadsafe Public

We make any object thread-safe and std::shared_mutex 10 times faster to achieve the speed of lock-free algorithms on >85% reads

C++ · 434 · 114

isl-org/MiDaS Public

Code for robust monocular depth estimation described in "Ranftl et. al., Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer, TPAMI 2022"

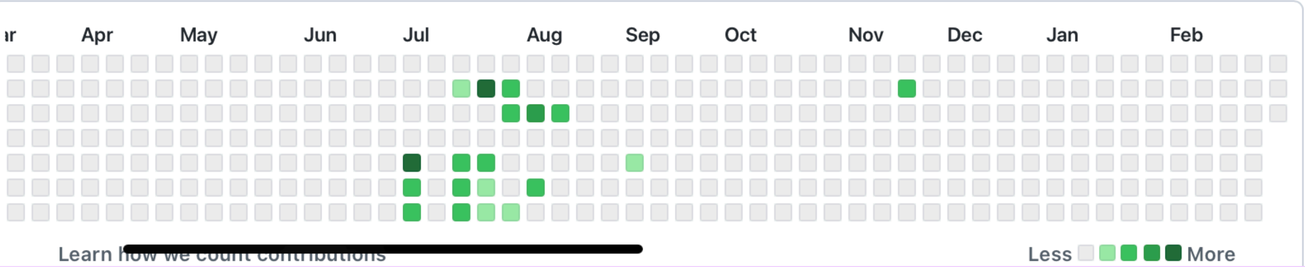
Python · 2.6k · 408

Yolo_mark Public

GUI for marking bounded boxes of objects in images for training neural network Yolo v3 and v2

C++ · 1.7k · 670

46 contributions in the last year



YOLOv4: Optimal Speed and Accuracy of Object Detection

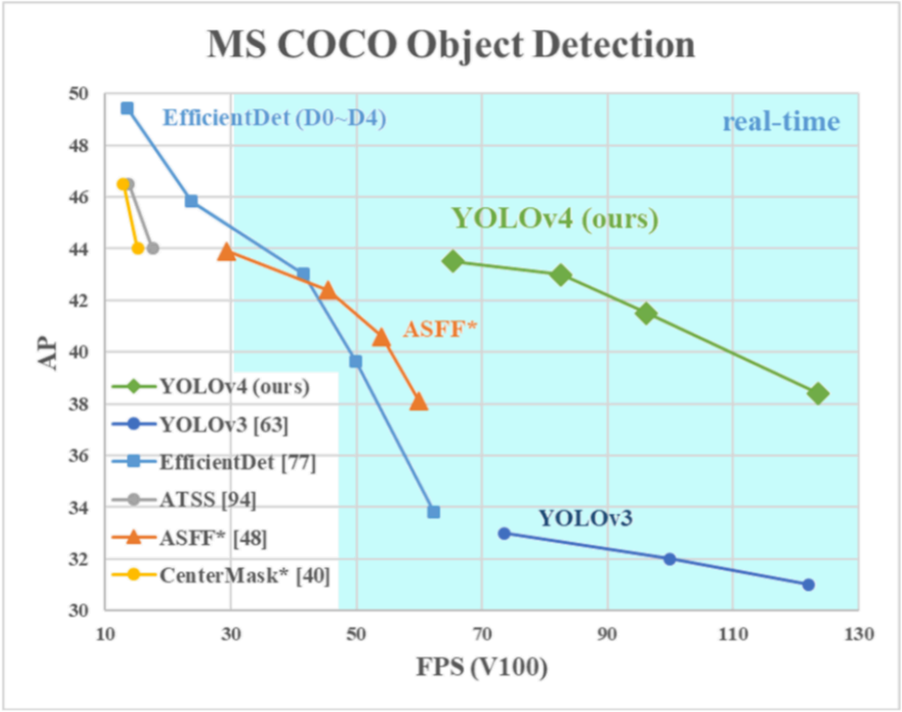
Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

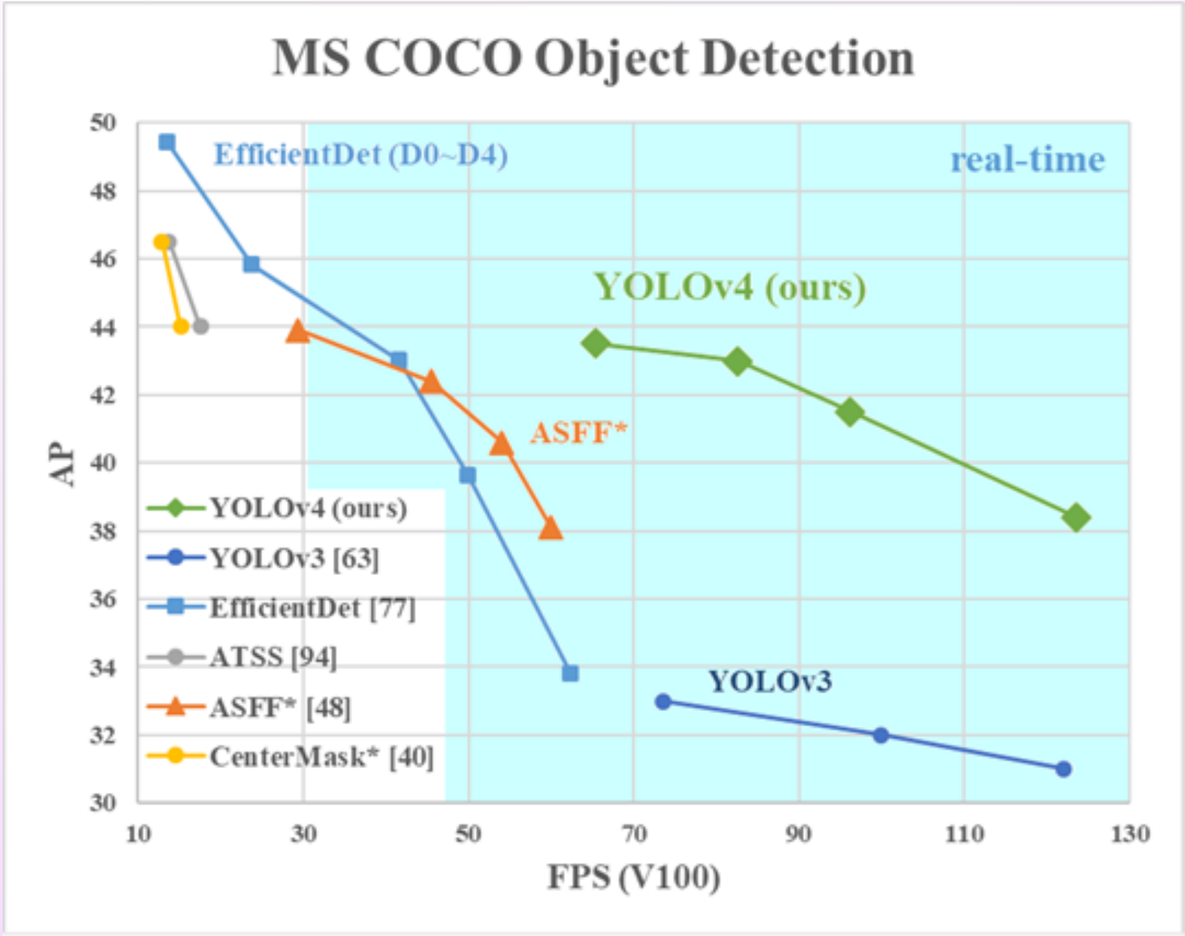
Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

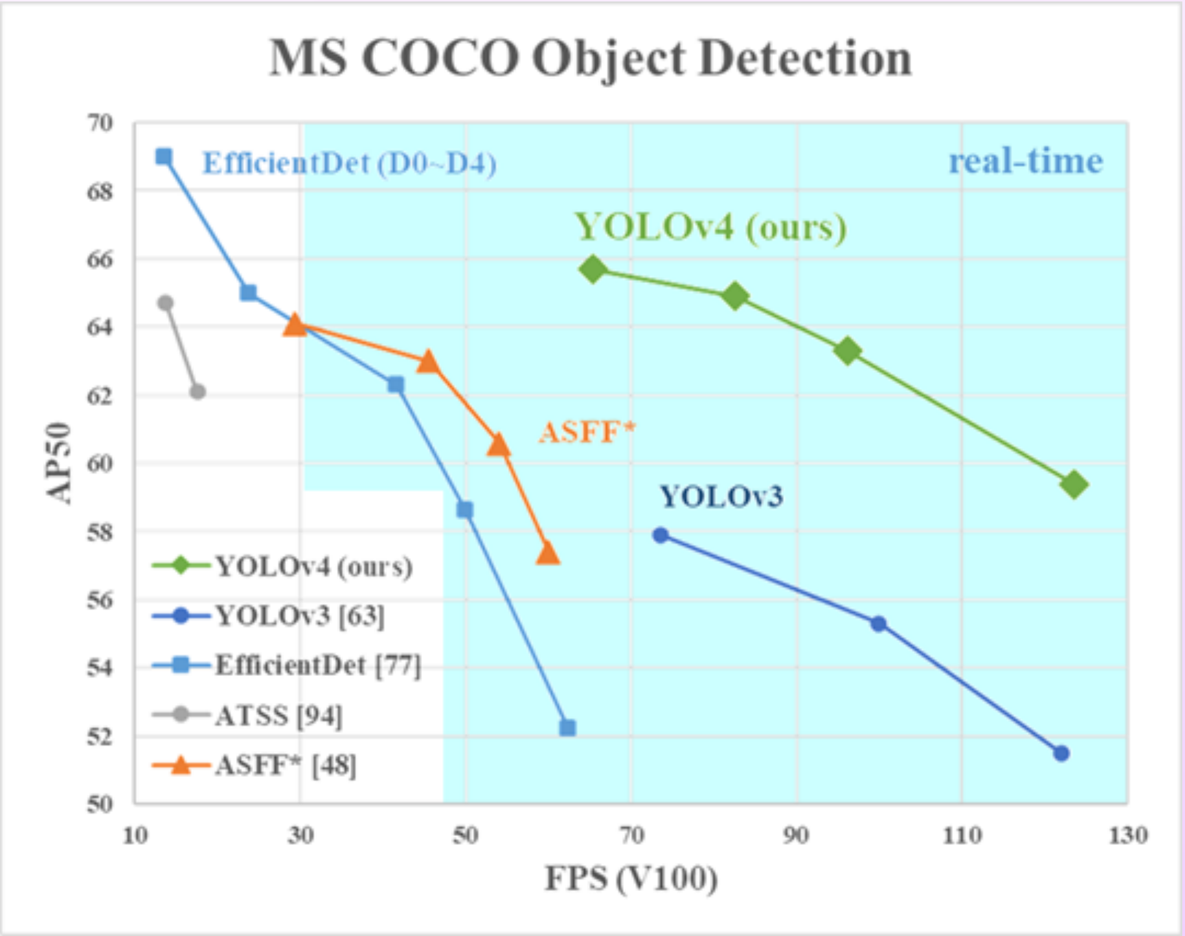
Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation,

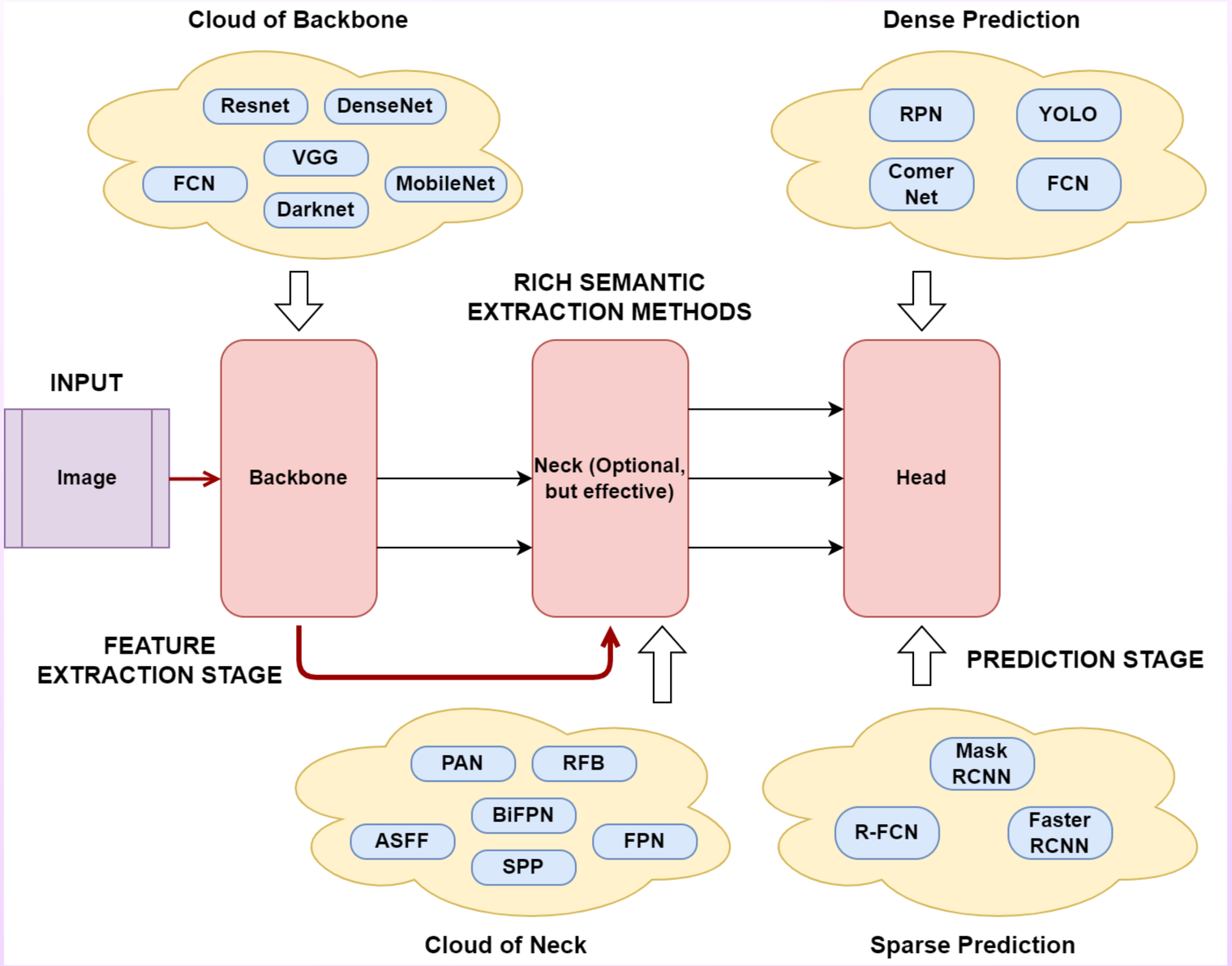


RESULTS

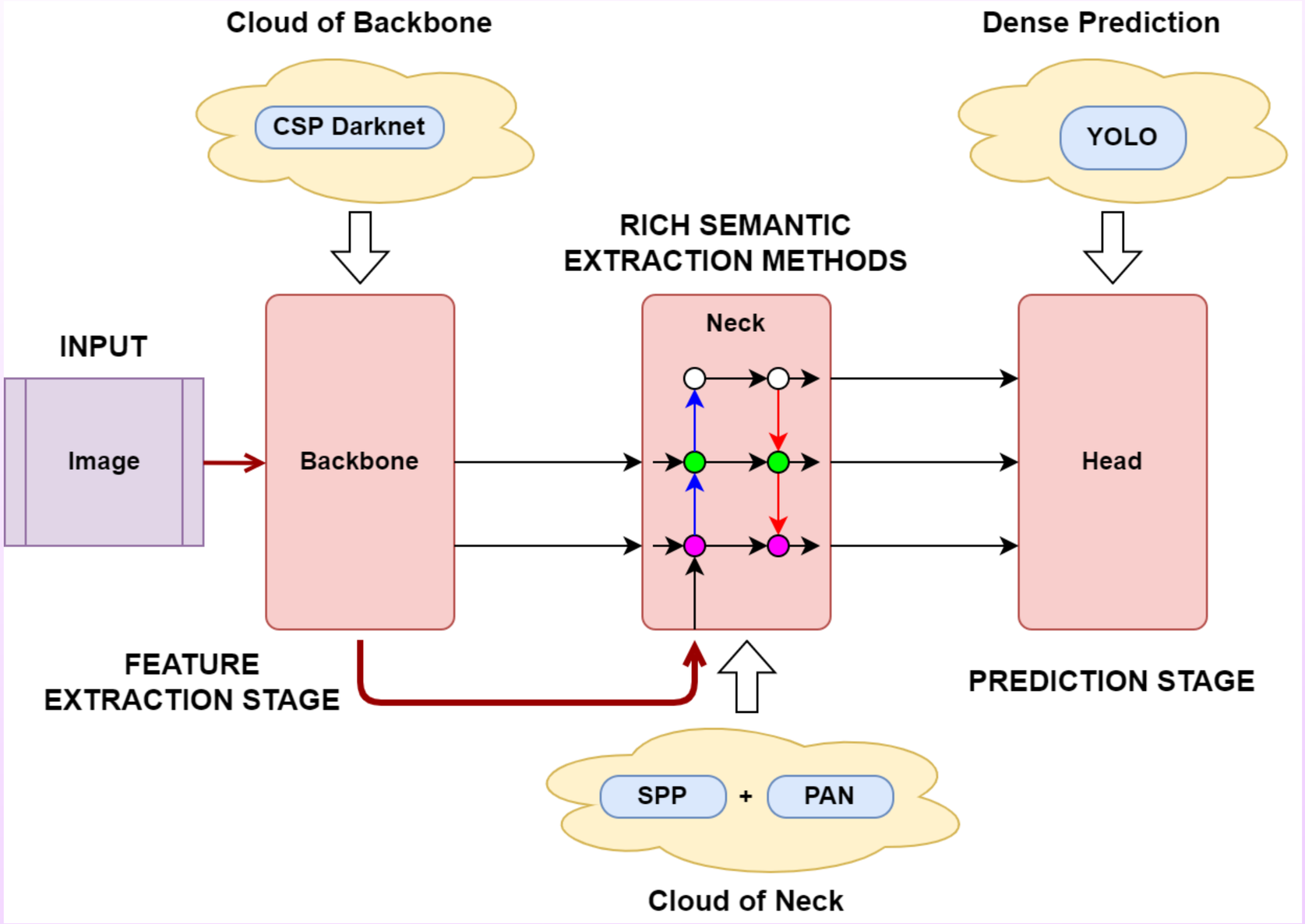

MS COCO Object Detection
AP
FPS (V100)
EfficientDet (D0~D4)
real-time
YOLOv4 (ours)
ASFF*
YOLOv3
YOLOv4 (ours)
YOLOv3 [63]
EfficientDet [77]
ATSS [94]
ASFF* [48]
CenterMask* [40]


MS COCO Object Detection
AP50
FPS (V100)
EfficientDet (D0~D4)
real-time
YOLOv4 (ours)
ASFF*
YOLOv3
YOLOv4 (ours)
YOLOv3 [63]
EfficientDet [77]
ATSS [94]
ASFF* [48]

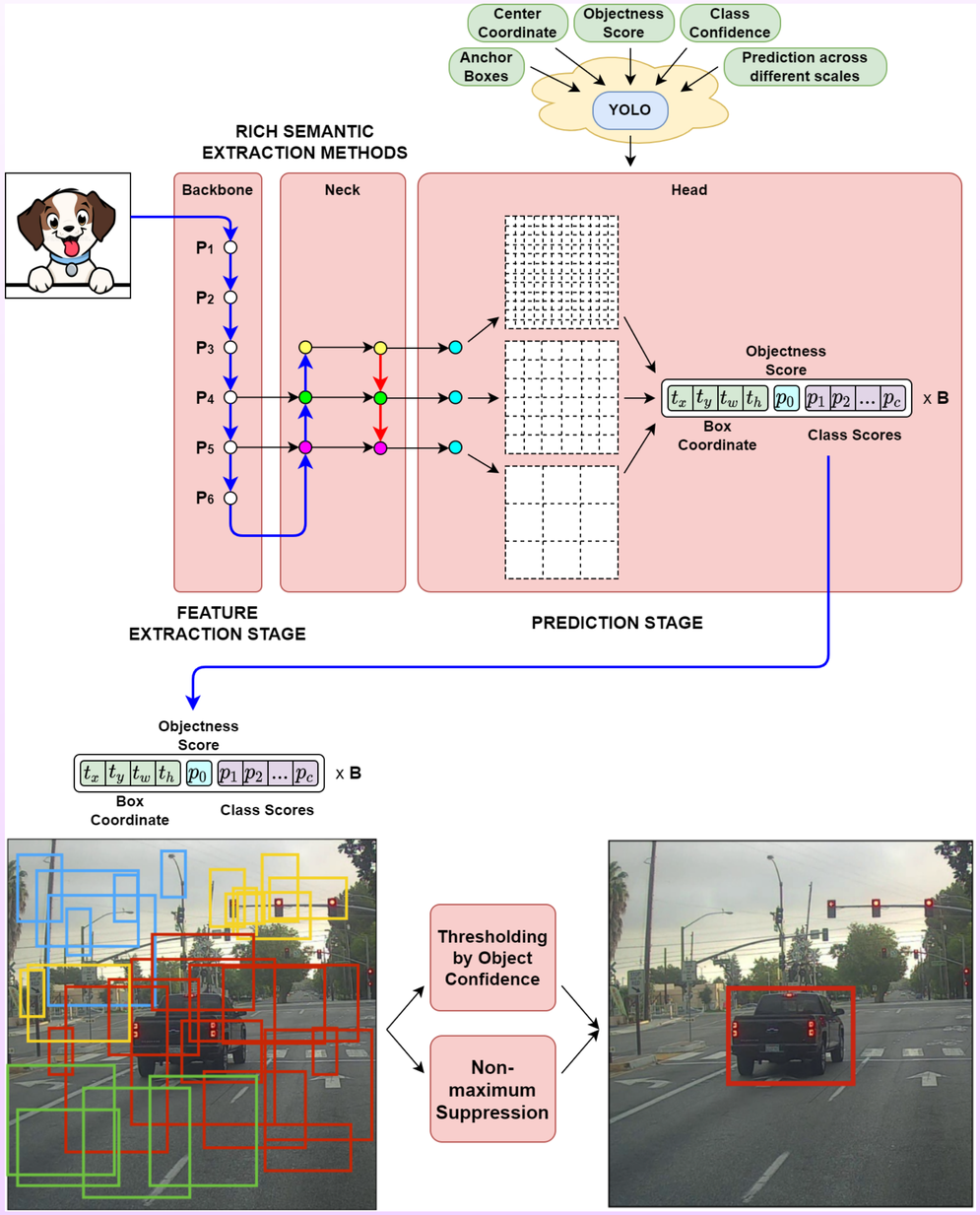
Components


Cloud of Backbone
Resnet
DenseNet
VGG
FCN
MobileNet
Darknet
Dense Prediction
RPN
YOLO
Comer Net
FCN
RICH SEMANTIC EXTRACTION METHODS
INPUT
Image
Backbone
Neck (Optional, but effective)
Head
FEATURE EXTRACTION STAGE
PREDICTION STAGE
PAN
RFB
BiFPN
ASFF
FPN
SPP
Mask RCNN
R-FCN
Faster RCNN
Cloud of Neck
Sparse Prediction

YoloV4


Cloud of Backbone
Dense Prediction
CSP Darknet
YOLO
RICH SEMANTIC
EXTRACTION METHODS
INPUT
Neck
Image
Backbone
Head
FEATURE
EXTRACTION STAGE
PREDICTION STAGE
SPP
+
PAN
Cloud of Neck

YoloV4


Center Coordinate
Objectness Score
Class Confidence
Anchor Boxes
Prediction across different scales
YOLO
RICH SEMANTIC EXTRACTION METHODS
Backbone
Neck
Head
P1
P2
P3
P4
P5
P6
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... p_c
x B
Box Coordinate
Class Scores
FEATURE EXTRACTION STAGE
PREDICTION STAGE
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... p_c
x B
Box Coordinate
Class Scores
Thresholding by Object Confidence
Non-maximum Suppression

Performance Optimizations

- **Bag of Freebies**
- **Bag of Specials**

Performance Optimizations

- **Bag of Freebies**
  - **Different opt methods during training and inference**
  - **Doesn’t increase the inference time**
  - **Increases the training time**
- **Bag of Specials**
  - **Different modules that drastically improve accuracy**
  - **Increase the inference time by a small amount**

BoF & BoS

	Backbone	Detector
Bag of Freebies (BoF)	• CutMix • Mosaic data augmentation • DropBlock • Class label smoothing	• CIoU-loss • Cross mini-Batch Normalization • DropBlock • Mosaic data augmentation • Self-Adversarial Training • Multiple anchors for a single ground truth • Cosine annealing scheduler • Optimal hyperparameters • Random training shapes
Bag of Specials (BoS)	• Mish activation • Cross-stage partial connections (CSP) • Multi-input weighted residual connections (MiWRC)	• Mish activation • SPP-block • SAM-block • PAN path-aggregation block • DIoU-NMS

Techniques Evaluated

- **Input**: Image, Patches, Image Pyramid
- **Backbones**: VGG16 [68], ResNet-50 [26], SpineNet [12], EfficientNet-B0/B7 [75], CSPResNeXt50 [81], CSPDarknet53 [81]
- **Neck**:
  - **Additional blocks**: SPP [25], ASPP [5], RFB [47], SAM [85]
  - **Path-aggregation blocks**: FPN [44], PAN [49], NAS-FPN [17], Fully-connected FPN, BiFPN [77], ASFF [48], SFAM [98]
- **Heads:**:
  - **Dense Prediction (one-stage)**:
    - RPN [64], SSD [50], YOLO [61], RetinaNet [45] (anchor based)
    - CornerNet [37], CenterNet [13], MatrixNet [60], FCOS [78] (anchor free)
  - **Sparse Prediction (two-stage)**:
    - Faster R-CNN [64], R-FCN [9], Mask R-CNN [23] (anchor based)
    - RepPoints [87] (anchor free)

- **Activations**: ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish
- **Bounding box regression loss**: MSE, IoU, GIoU, CIoU, DIoU
- **Data augmentation**: CutOut, MixUp, CutMix
- **Regularization method**: DropOut, DropPath [36], Spatial DropOut [79], or DropBlock
- **Normalization of the network activations by their mean and variance**: Batch Normalization (BN) [32], Cross-GPU Batch Normalization (CGBN or SyncBN) [93], Filter Response Normalization (FRN) [70], or Cross-Iteration Batch Normalization (CBN) [89]
- **Skip-connections**: Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP)


Thank You!