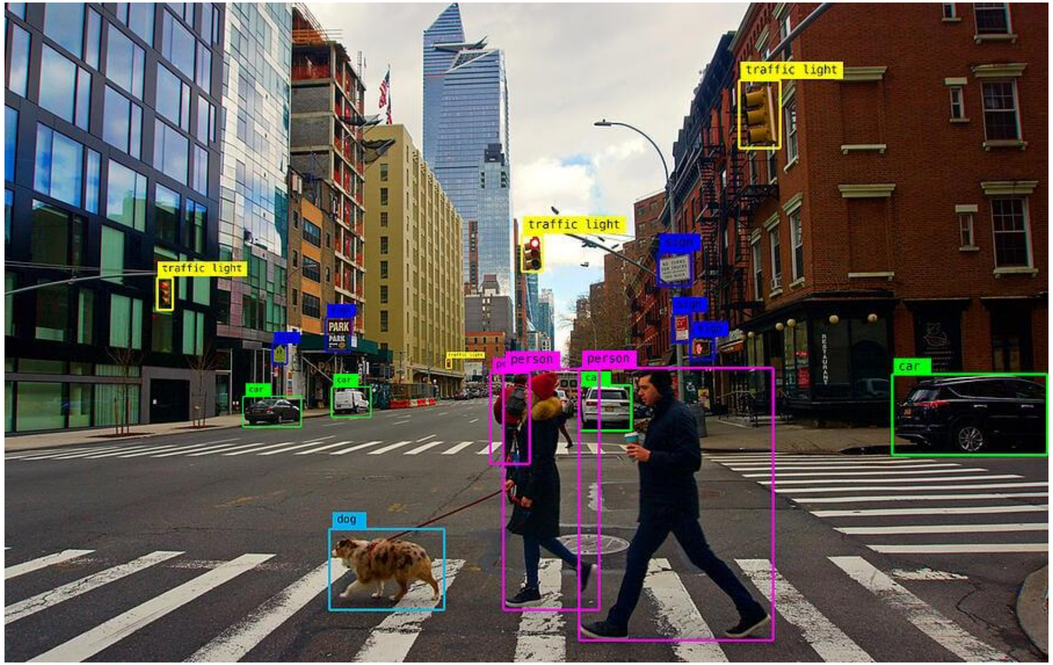
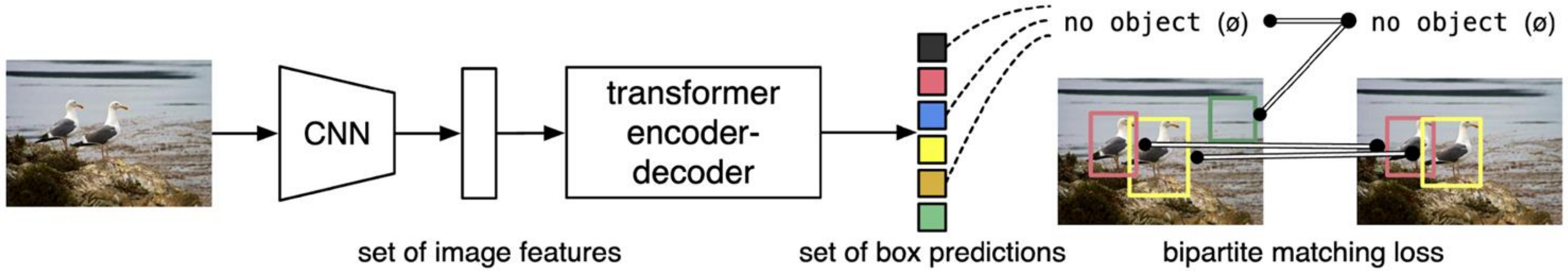
AI VIETNAM
All-in-One Course

Object Detection



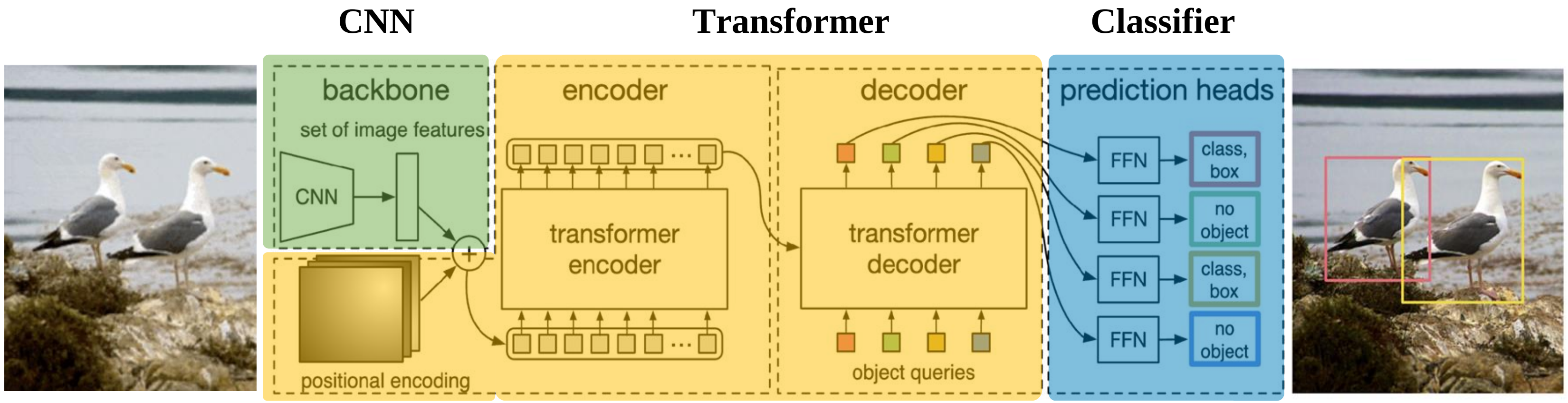
TA Hùng An

DERT: End-to-End Object Detection with Transformers



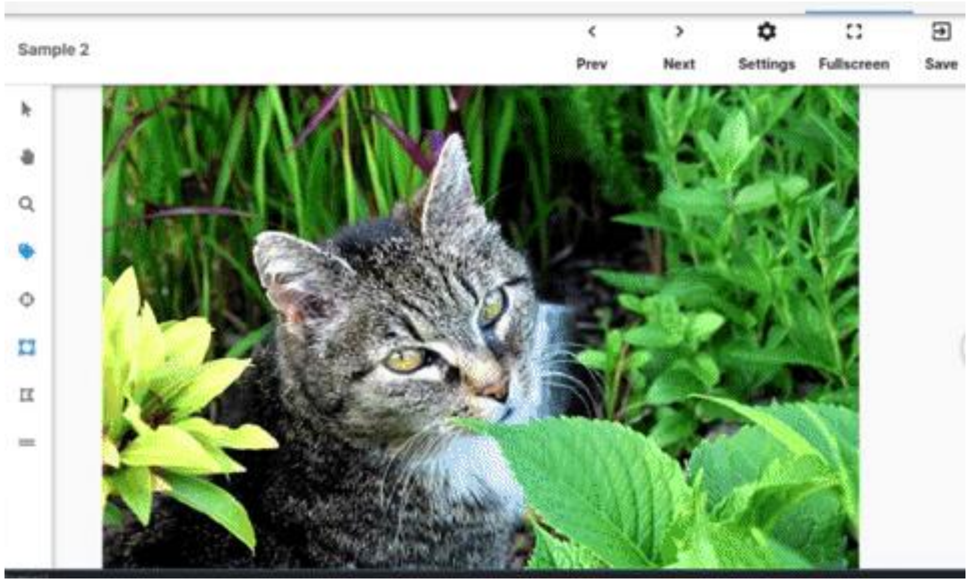
- **simplicity**: simplify the object detection pipeline by removing the need for intermediate steps like proposal generation and NMS
- **end-to-end**: predict the final set of objects (and their bounding boxes) in a single step
- **fully differentiable**: can be entirely trained in a single pass of gradient descent, which usually comes with significant speed and simplicity

DERT: End-to-End Object Detection with Transformers



Label Tools

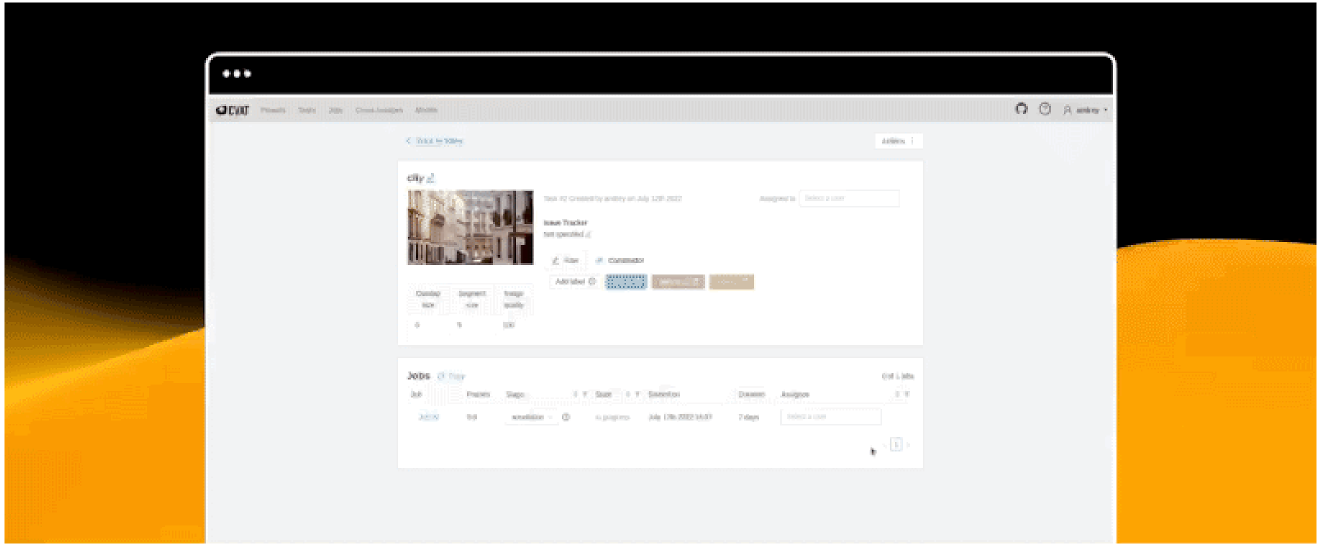
1- Universal Data Tool



Universal Data Tool là một web/desktop app cho phép chỉnh sửa và đánh nhãn image, text, audio, document và cho phép xem và chỉnh sửa lại ngay trên giao diện.

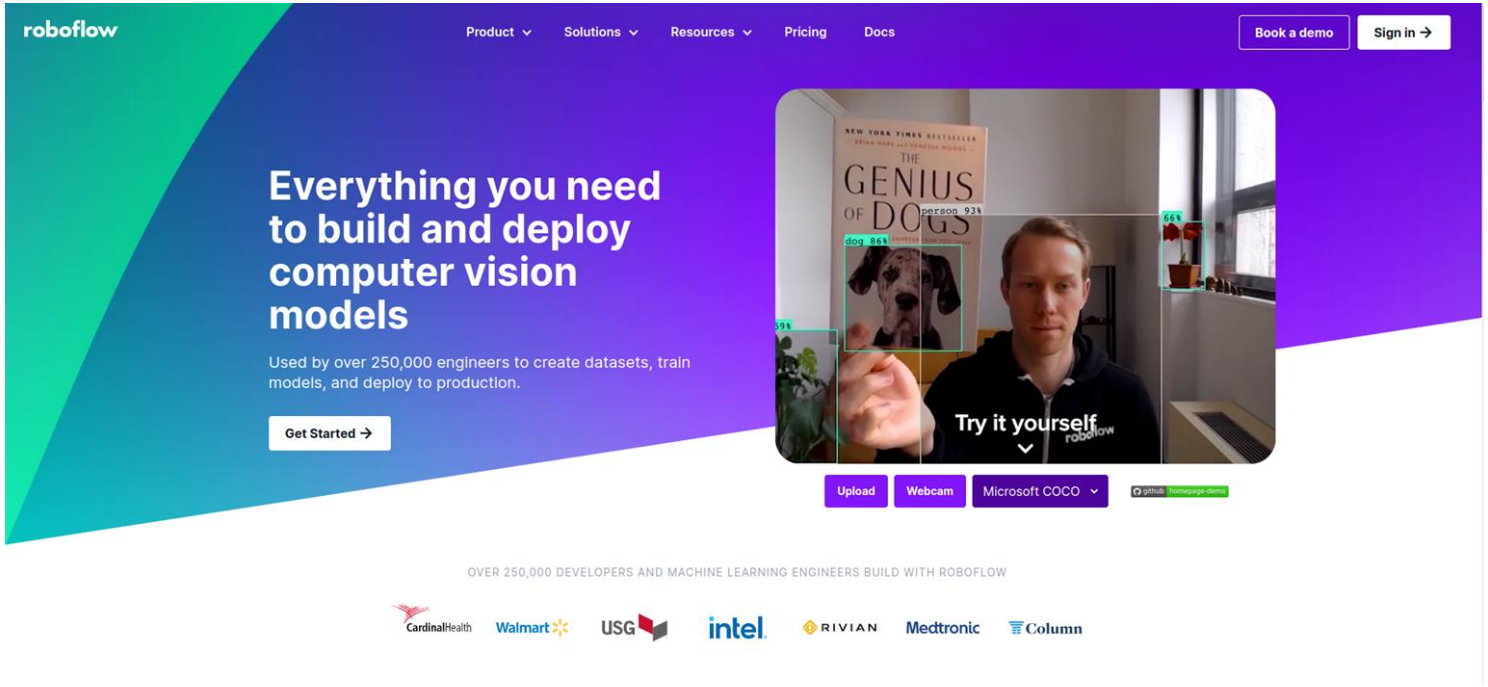
Label Tools

2- Computer Vision Annotation Tool (CVAT)



Label Tools

3- Label Studio



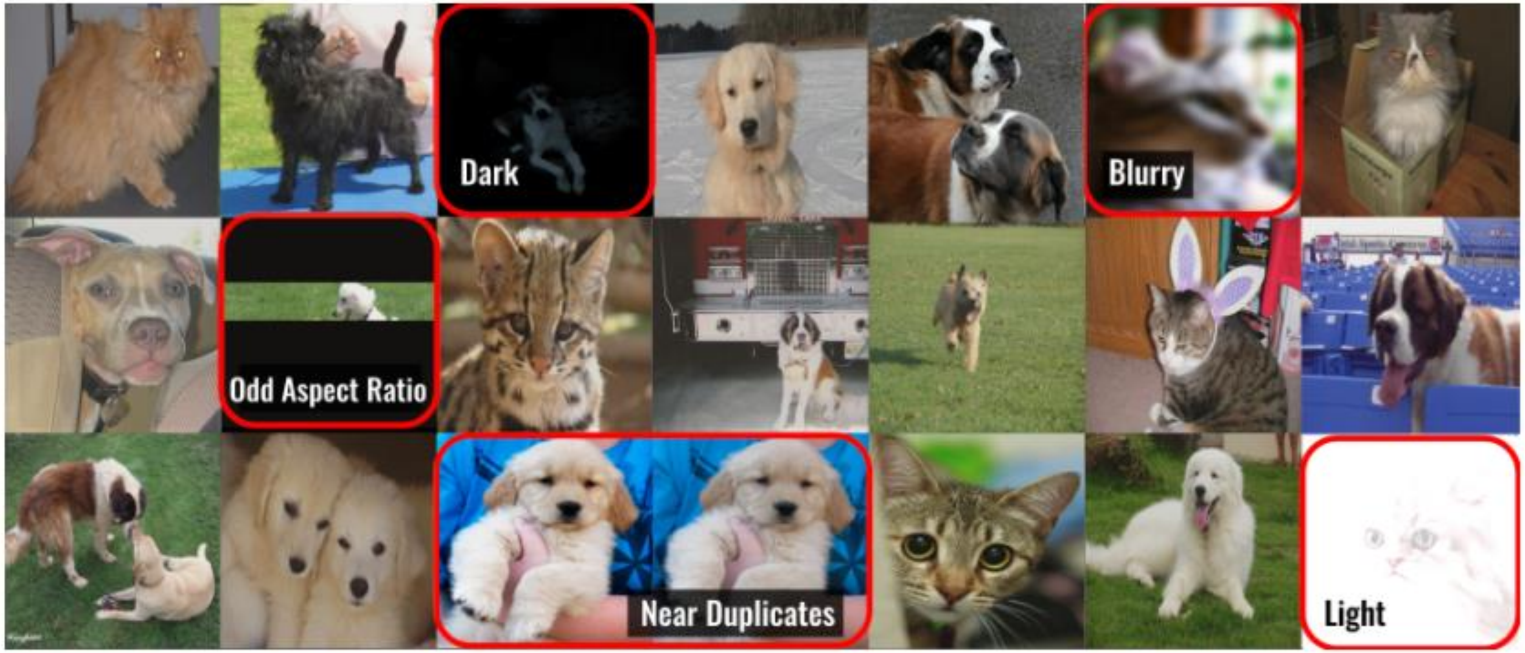
Everything you need to build and deploy computer vision models

Used by over 250,000 engineers to create datasets, train models, and deploy to production.

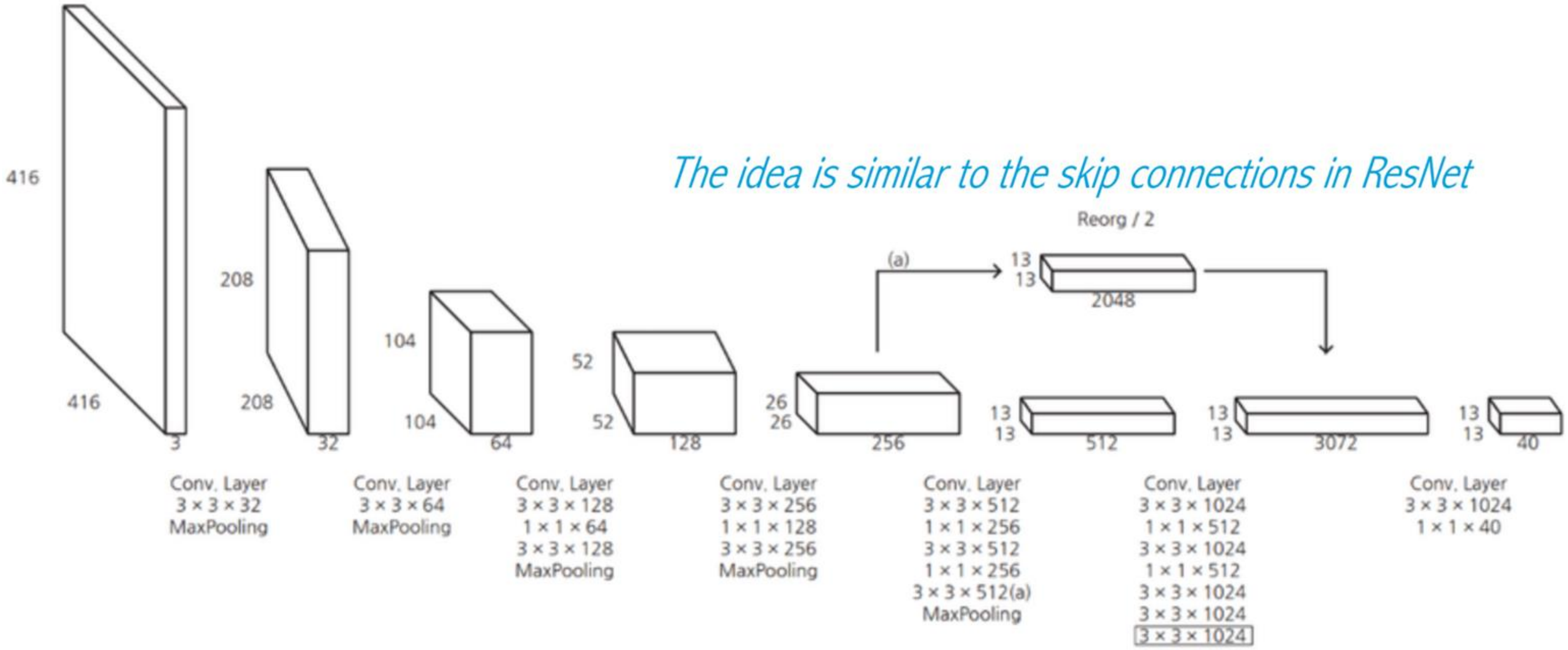
OVER 250,000 DEVELOPERS AND MACHINE LEARNING ENGINEERS BUILD WITH ROBOFLOW

Label Tools

4- CleanVision

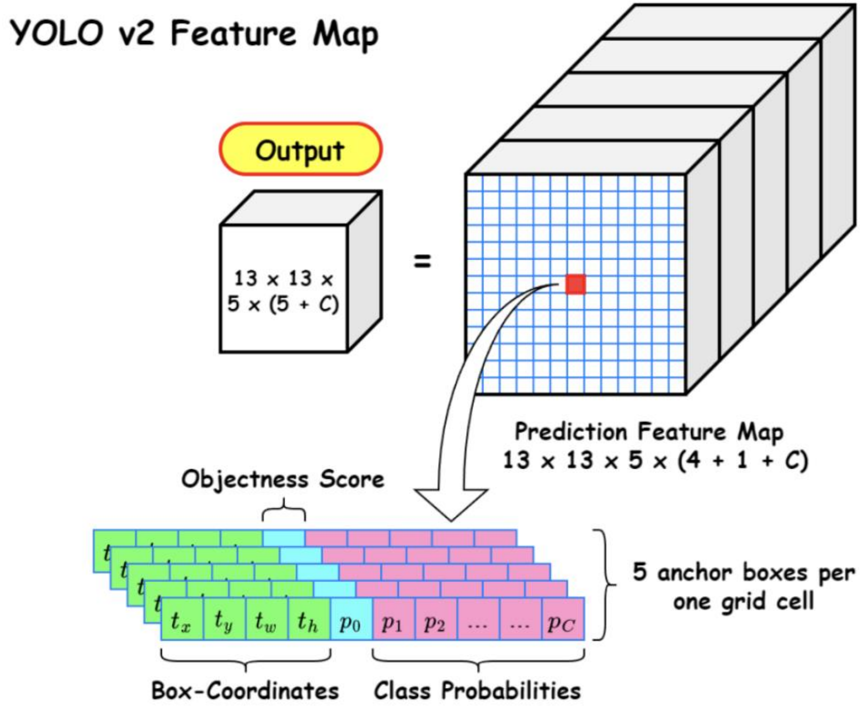


YOLOv2

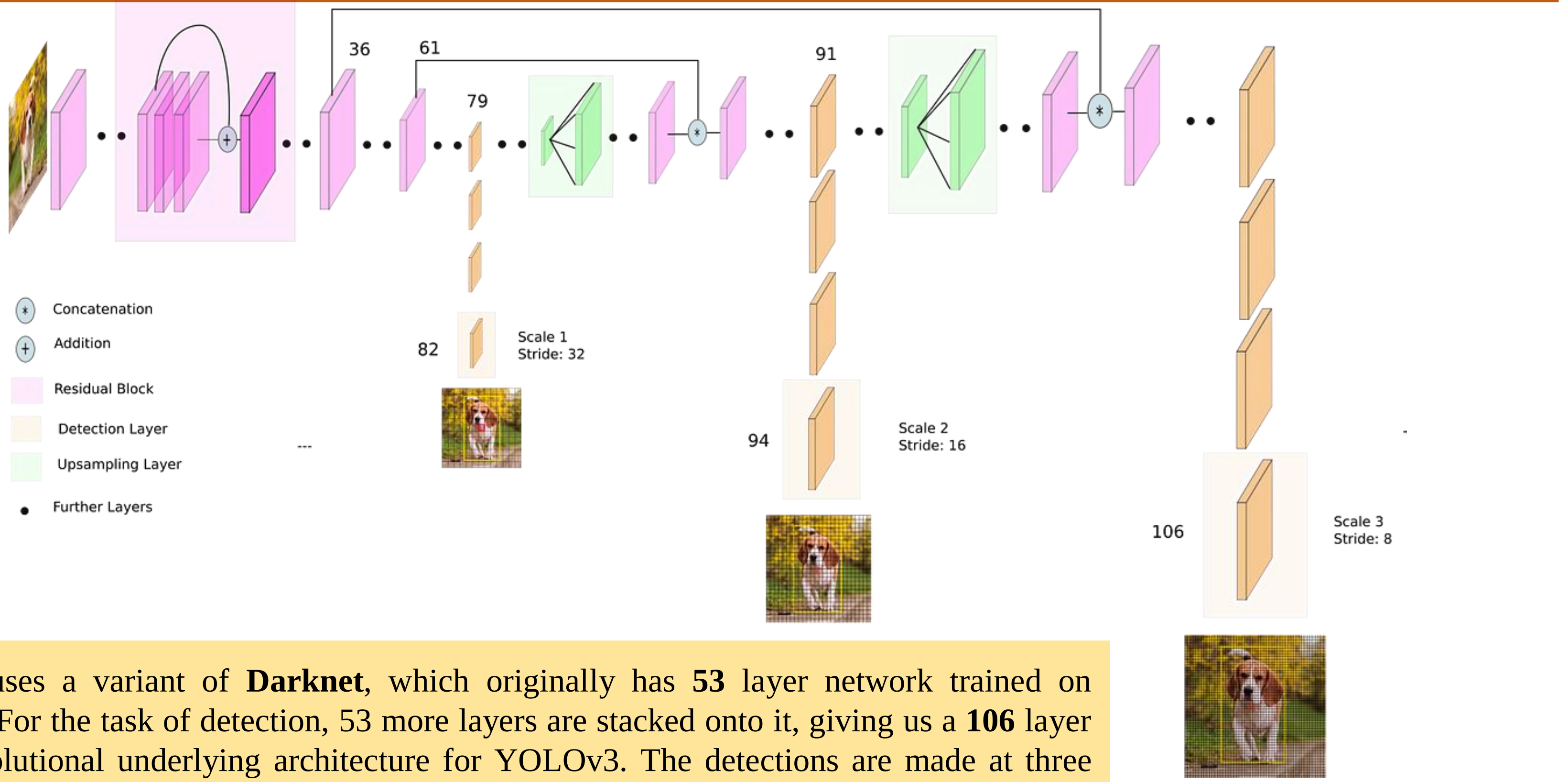


The idea is similar to the skip connections in ResNet
Reorg / 2
(a)
416
416
3
208
208
32
104
104
64
52
52
128
26
26
256
13
13
512
13
13
2048
13
13
3072
13
13
40
Conv. Layer
3 × 3 × 32
MaxPooling
Conv. Layer
3 × 3 × 64
MaxPooling
Conv. Layer
3 × 3 × 128
1 × 1 × 64
3 × 3 × 128
MaxPooling
Conv. Layer
3 × 3 × 256
1 × 1 × 128
3 × 3 × 256
MaxPooling
Conv. Layer
3 × 3 × 512
1 × 1 × 256
3 × 3 × 512
1 × 1 × 256
3 × 3 × 512(a)
MaxPooling
Conv. Layer
3 × 3 × 1024
1 × 1 × 512
3 × 3 × 1024
1 × 1 × 512
3 × 3 × 1024
3 × 3 × 1024
3 × 3 × 1024
Conv. Layer
3 × 3 × 1024
1 × 1 × 40

YOLOv2

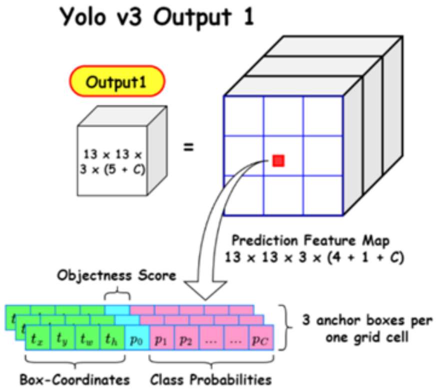

YOLO v2 Feature Map
Output
13 × 13 × 5 × (5 + C)
=
Prediction Feature Map
13 × 13 × 5 × (4 + 1 + C)
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 … … p_C
5 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

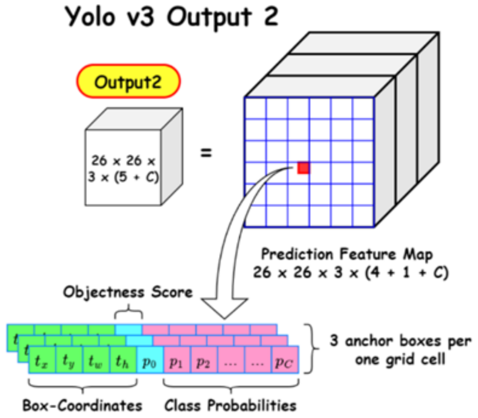
YOLOv3

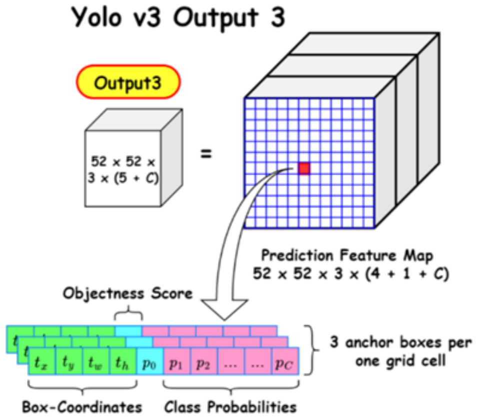


YOLOv3 uses a variant of **Darknet**, which originally has **53** layer network trained on ImageNet. For the task of detection, 53 more layers are stacked onto it, giving us a **106** layer fully convolutional underlying architecture for YOLOv3. The detections are made at three layers **82nd**, **94th** and **106th** layer.

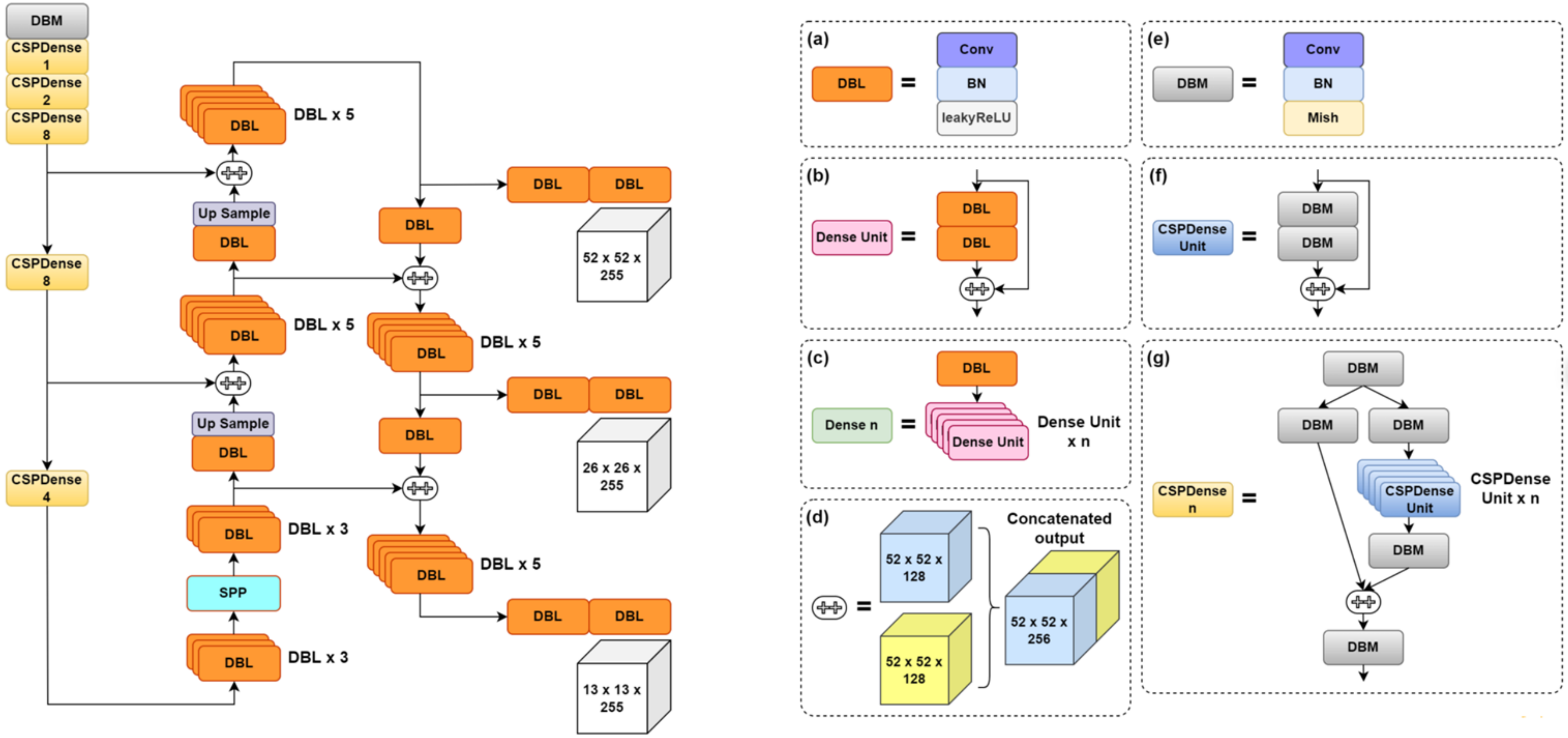
YOLOv3


Yolo v3 Output 1
Output1
13 x 13 x 3 x (5 + C)
=
Prediction Feature Map
13 x 13 x 3 x (4 + 1 + C)
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

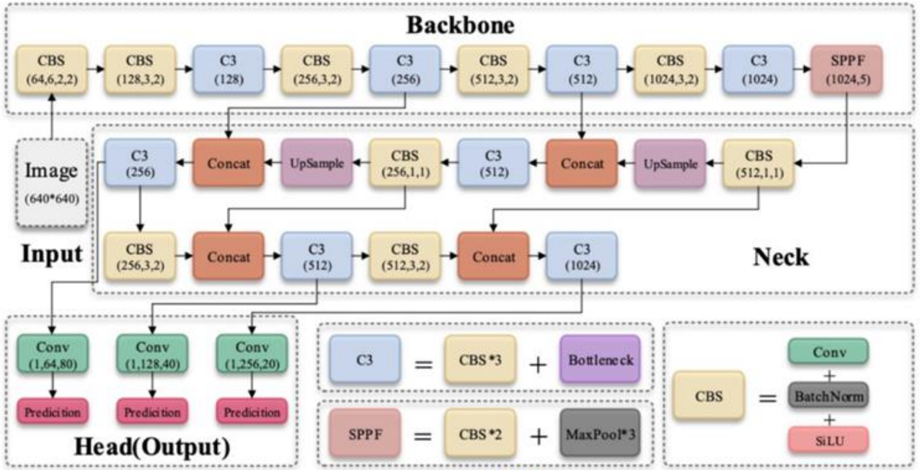

Yolo v3 Output 2
Output2
26 x 26 x 3 x (5 + C)
=
Prediction Feature Map
26 x 26 x 3 x (4 + 1 + C)
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities


Yolo v3 Output 3
Output3
52 x 52 x 3 x (5 + C)
=
Prediction Feature Map
52 x 52 x 3 x (4 + 1 + C)
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

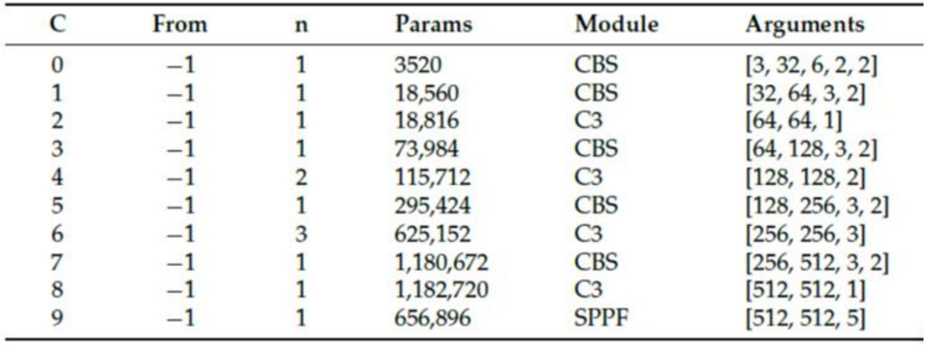
YOLOv4


DBM
CSPDense 1
CSPDense 2
CSPDense 8
CSPDense 8
CSPDense 4
DBL x 5
DBL
Up Sample
DBL
DBL x 5
Up Sample
DBL
DBL x 3
SPP
DBL x 3
DBL x 5
DBL x 5
DBL DBL
52 x 52 x 255
26 x 26 x 255
13 x 13 x 255
(a) DBL = Conv BN leakyReLU
(b) Dense Unit = DBL DBL
(c) Dense n = DBL Dense Unit x n
(d) Concatenated output
52 x 52 x 128
52 x 52 x 128
52 x 52 x 256
(e) DBM = Conv BN Mish
(f) CSPDense Unit = DBM DBM
(g) CSPDense n = DBM DBM DBM CSPDense Unit x n DBM DBM

YOLOv5

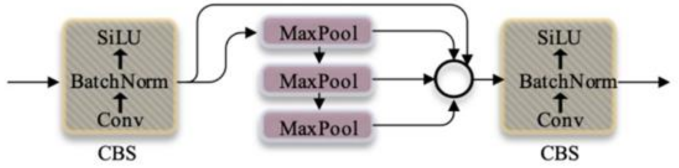


The backbone is CSPDarknet53.

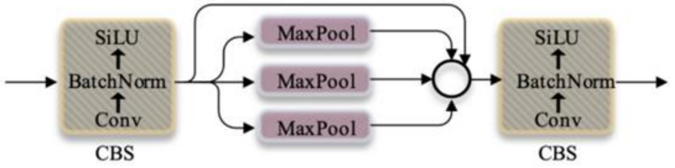


C	From	n	Params	Module	Arguments
0	−1	1	3520	CBS	[3, 32, 6, 2, 2]
1	−1	1	18,560	CBS	[32, 64, 3, 2]
2	−1	1	18,816	C3	[64, 64, 1]
3	−1	1	73,984	CBS	[64, 128, 3, 2]
4	−1	2	115,712	C3	[128, 128, 2]
5	−1	1	295,424	CBS	[128, 256, 3, 2]
6	−1	3	625,152	C3	[256, 256, 3]
7	−1	1	1,180,672	CBS	[256, 512, 3, 2]
8	−1	1	1,182,720	C3	[512, 512, 1]
9	−1	1	656,896	SPPF	[512, 512, 5]

Parameter of backbone in YOLOv5 network structure.

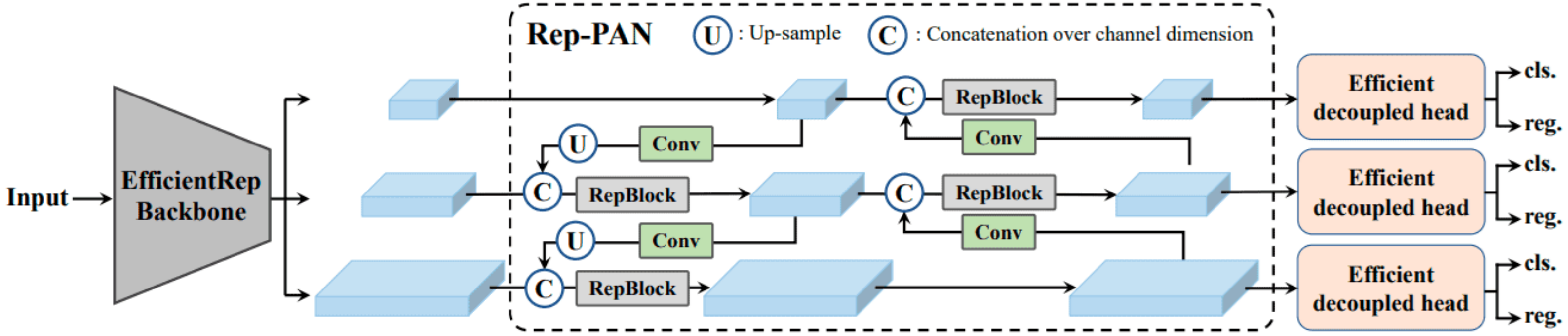


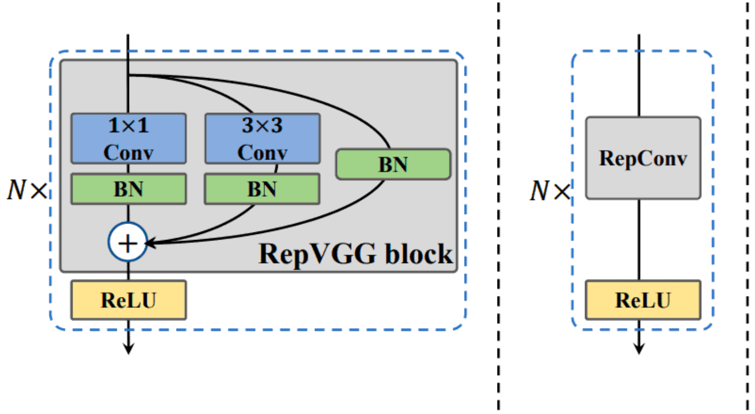
(a) SPPF



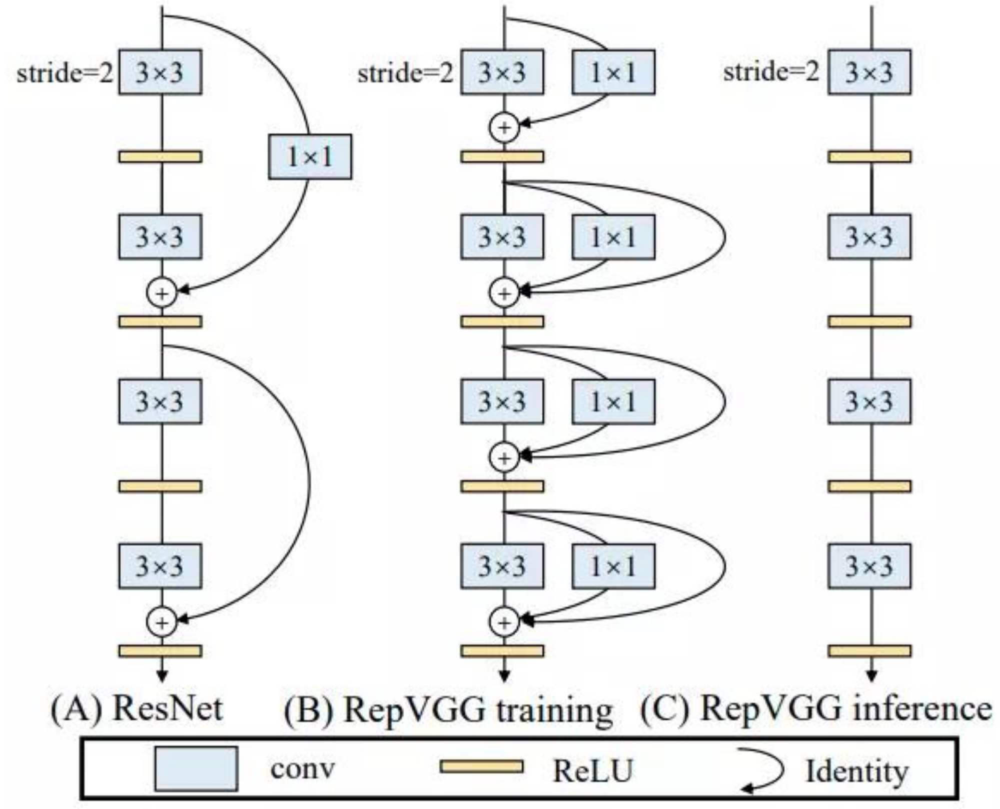
(b) SPP

YOLOv6

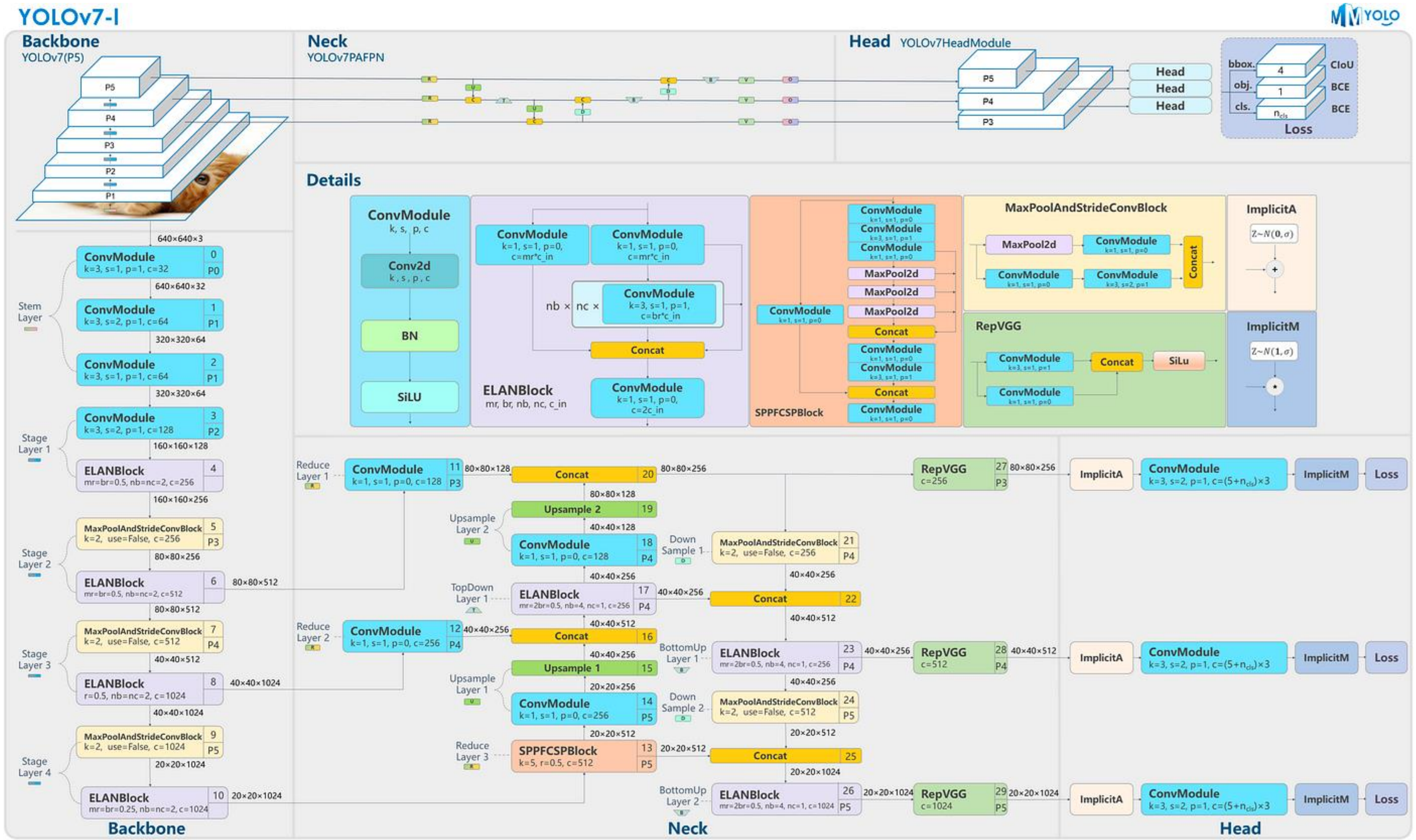

Rep-PAN
U : Up-sample
C : Concatenation over channel dimension
Input
EfficientRep Backbone
Conv
RepBlock
U
C
Efficient decoupled head
cls.
reg.


1×1 Conv
3×3 Conv
BN
BN
BN
N×
RepVGG block
ReLU
RepConv
N×
ReLU

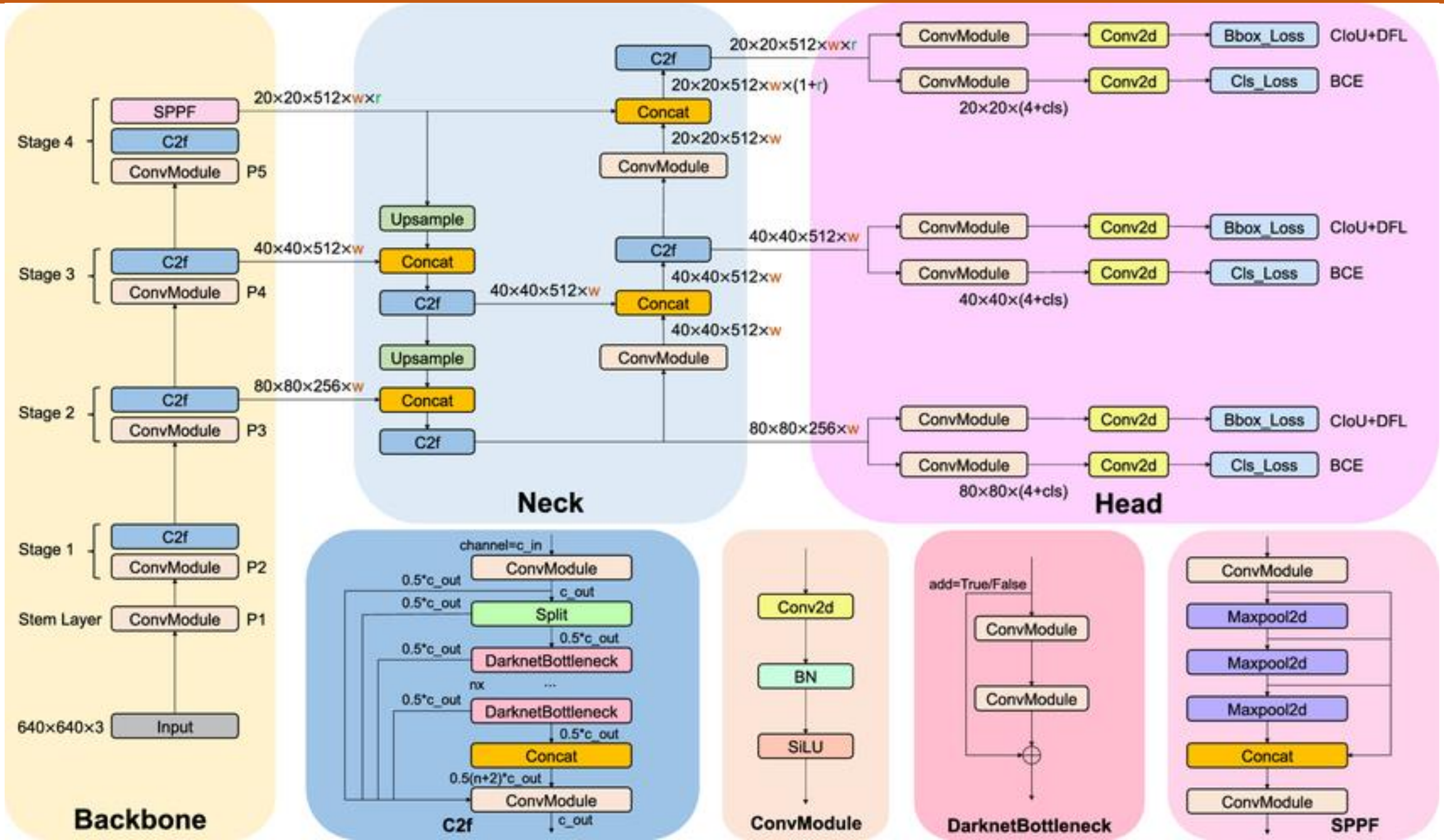
YOLOv6


stride=2
3×3
1×1
(A) ResNet
(B) RepVGG training
(C) RepVGG inference
conv
ReLU
Identity

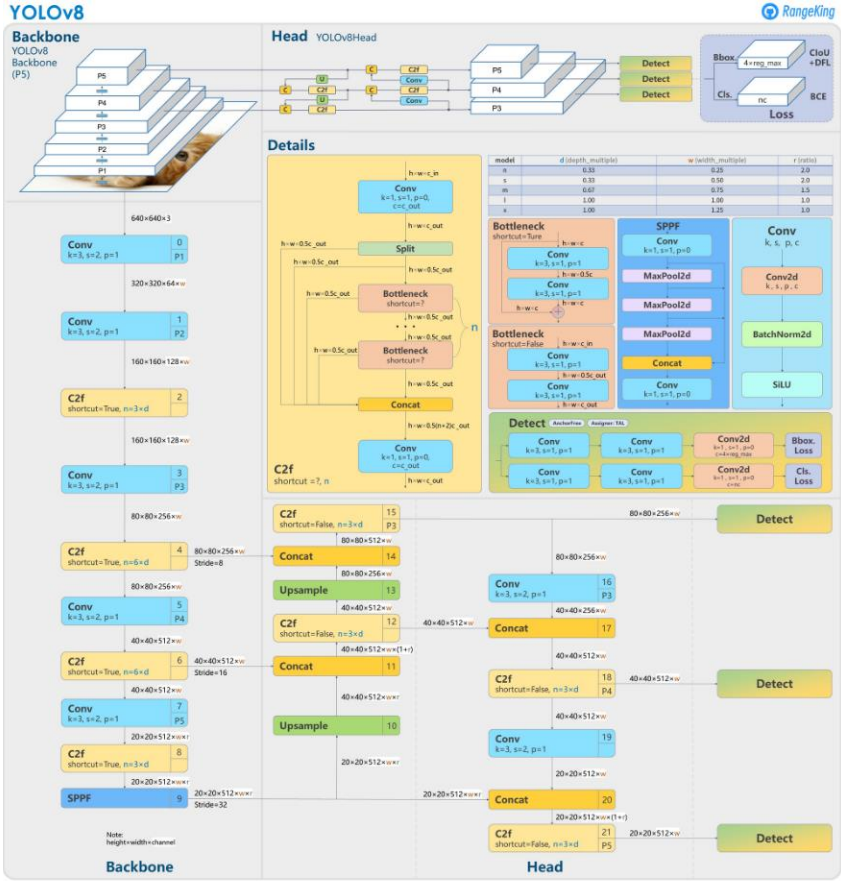
YOLOv7


YOLOv7-l
MMYOLO
Backbone
YOLOv7(P5)
P5
P4
P3
P2
P1
Neck
YOLOv7PAFPN
Head
YOLOv7HeadModule
Head
Head
Head
bbox.
obj.
cls.
4
1
n_{cls}
CIoU
BCE
BCE
Loss
Details
ConvModule
k, s, p, c
Conv2d
k, s, p, c
BN
SiLU
ConvModule k=1, s=1, p=0, c=mr*c_in
ConvModule k=1, s=1, p=0, c=mr*c_in
nb × nc × ConvModule k=3, s=1, p=1, c=br*c_in
Concat
ELANBlock
mr, br, nb, nc, c_in
ConvModule k=1, s=1, p=0, c=2c_in
ConvModule k=1, s=1, p=0
ConvModule k=3, s=1, p=1
ConvModule k=1, s=1, p=0
MaxPool2d
MaxPool2d
MaxPool2d
Concat
ConvModule k=1, s=1, p=0
ConvModule k=3, s=1, p=1
Concat
ConvModule k=1, s=1, p=0
ConvModule k=1, s=1, p=0
SPPFCSPBlock
MaxPoolAndStrideConvBlock
MaxPool2d
ConvModule k=1, s=1, p=0
ConvModule k=1, s=1, p=0
ConvModule k=3, s=2, p=1
Concat
RepVGG
ConvModule k=3, s=1, p=1
ConvModule k=1, s=1, p=0
Concat
SiLu
ImplicitA
$Z \sim N(0, \sigma)$
+
ImplicitM
$Z \sim N(1, \sigma)$
·
640×640×3
Stem Layer
ConvModule k=3, s=1, p=1, c=32 0 P0
640×640×32
ConvModule k=3, s=2, p=1, c=64 1 P1
320×320×64
ConvModule k=3, s=1, p=1, c=64 2 P1
320×320×64
Stage Layer 1
ConvModule k=3, s=2, p=1, c=128 3 P2
160×160×128
ELANBlock mr=br=0.5, nb=nc=2, c=256 4
160×160×256
Stage Layer 2
MaxPoolAndStrideConvBlock k=2, use=False, c=256 5 P3
80×80×256
ELANBlock mr=br=0.5, nb=nc=2, c=512 6
80×80×512
80×80×512
Stage Layer 3
MaxPoolAndStrideConvBlock k=2, use=False, c=512 7 P4
40×40×512
ELANBlock r=0.5, nb=nc=2, c=1024 8
40×40×1024
40×40×1024
Stage Layer 4
MaxPoolAndStrideConvBlock k=2, use=False, c=1024 9 P5
20×20×1024
ELANBlock mr=br=0.25, nb=nc=2, c=1024 10
20×20×1024
Backbone
Reduce Layer 1
ConvModule k=1, s=1, p=0, c=128 11 P3
80×80×128
Concat 20
80×80×256
80×80×128
Upsample Layer 2
Upsample 2 19
40×40×128
ConvModule k=1, s=1, p=0, c=128 18 P4
40×40×256
TopDown Layer 1
ELANBlock mr=2br=0.5, nb=4, nc=1, c=256 17 P4
40×40×512
Reduce Layer 2
ConvModule k=1, s=1, p=0, c=256 12 P4
40×40×256
Concat 16
40×40×256
Upsample Layer 1
Upsample 1 15
20×20×256
ConvModule k=1, s=1, p=0, c=256 14 P5
20×20×512
Reduce Layer 3
SPPFCSPBlock k=5, r=0.5, c=512 13 P5
20×20×512
Down Sample 1
MaxPoolAndStrideConvBlock k=2, use=False, c=256 21 P4
40×40×256
40×40×256
Concat 22
40×40×512
BottomUp Layer 1
ELANBlock mr=2br=0.5, nb=4, nc=1, c=256 23 P4
40×40×256
Down Sample 2
MaxPoolAndStrideConvBlock k=2, use=False, c=512 24 P5
20×20×512
20×20×512
Concat 25
20×20×1024
BottomUp Layer 2
ELANBlock mr=2br=0.5, nb=4, nc=1, c=1024 26 P5
20×20×1024
Neck
RepVGG c=256 27 P3
80×80×256
RepVGG c=512 28 P4
40×40×512
RepVGG c=1024 29 P5
20×20×1024
ImplicitA
ConvModule k=3, s=2, p=1, c=(5+n_{cls})×3
ImplicitM
Loss
Head

YOLOv8


Backbone
Stage 4
SPPF
C2f
ConvModule
P5
Stage 3
C2f
ConvModule
P4
Stage 2
C2f
ConvModule
P3
Stage 1
C2f
ConvModule
P2
Stem Layer
ConvModule
P1
640×640×3
Input
20×20×512×w×r
40×40×512×w
80×80×256×w
Neck
Upsample
Concat
C2f
Upsample
Concat
C2f
40×40×512×w
C2f
20×20×512×w×(1+r)
Concat
20×20×512×w
ConvModule
C2f
40×40×512×w
Concat
40×40×512×w
ConvModule
Head
20×20×512×w×r
40×40×512×w
80×80×256×w
ConvModule
Conv2d
Bbox_Loss
CIoU+DFL
Cls_Loss
BCE
20×20×(4+cls)
40×40×(4+cls)
80×80×(4+cls)
C2f
channel=c_in
ConvModule
c_out
0.5*c_out
Split
DarknetBottleneck
nx
...
Concat
0.5(n+2)*c_out
ConvModule
Conv2d
BN
SiLU
DarknetBottleneck
add=True/False
SPPF
Maxpool2d

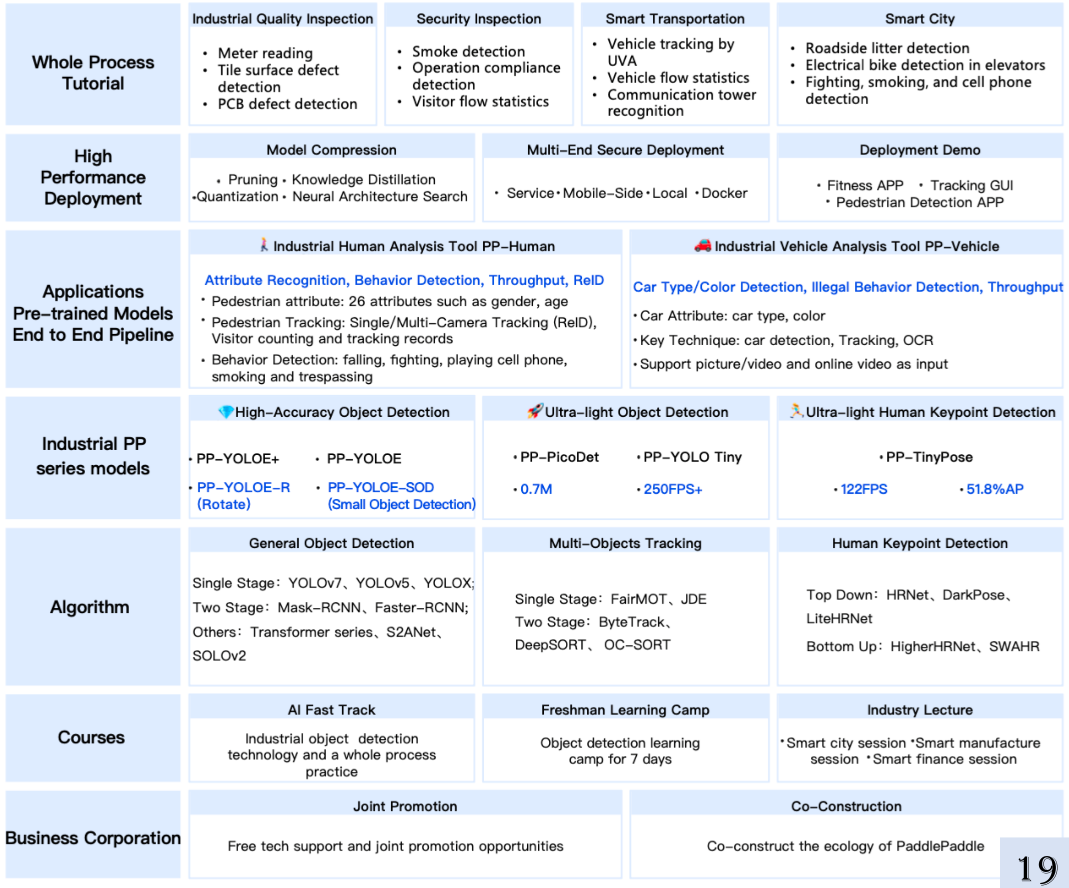
YOLOv8



model	d (depth_multiple)	w (width_multiple)	r (ratio)
n	0.33	0.25	2.0
s	0.33	0.50	2.0
m	0.67	0.75	1.5
l	1.00	1.00	1.0
x	1.00	1.25	1.0

PaddleDectection

- PaddleDetection is an end-to-end object detection development kit based on PaddlePaddle.
- Providing over **30 model algorithm** and **over 300 pre-trained models**.
- Task: object detection, instance segmentation, keypoint detection, multi-object tracking.
- Offer high- performance & light-weight industrial SOTA models on servers and mobile devices, champion solution and cutting-edge algorithm.



Whole Process Tutorial

Industrial Quality Inspection
- Meter reading
- Tile surface defect detection
- PCB defect detection

Security Inspection
- Smoke detection
- Operation compliance detection
- Visitor flow statistics

Smart Transportation
- Vehicle tracking by UVA
- Vehicle flow statistics
- Communication tower recognition

Smart City
- Roadside litter detection
- Electrical bike detection in elevators
- Fighting, smoking, and cell phone detection

High Performance Deployment

Model Compression
- Pruning • Knowledge Distillation
- Quantization • Neural Architecture Search

Multi-End Secure Deployment
- Service • Mobile-Side • Local • Docker

Deployment Demo
- Fitness APP • Tracking GUI
- Pedestrian Detection APP

Applications Pre-trained Models End to End Pipeline

Industrial Human Analysis Tool PP-Human

Attribute Recognition, Behavior Detection, Throughput, ReID
- Pedestrian attribute: 26 attributes such as gender, age
- Pedestrian Tracking: Single/Multi-Camera Tracking (ReID), Visitor counting and tracking records
- Behavior Detection: falling, fighting, playing cell phone, smoking and trespassing

Industrial Vehicle Analysis Tool PP-Vehicle

Car Type/Color Detection, Illegal Behavior Detection, Throughput
- Car Attribute: car type, color
- Key Technique: car detection, Tracking, OCR
- Support picture/video and online video as input

Industrial PP series models

High-Accuracy Object Detection
- PP-YOLOE+
- PP-YOLOE
- PP-YOLOE-R (Rotate)
- PP-YOLOE-SOD (Small Object Detection)

Ultra-light Object Detection
- PP-PicoDet
- PP-YOLO Tiny
- 0.7M
- 250FPS+

Ultra-light Human Keypoint Detection
- PP-TinyPose
- 122FPS
- 51.8%AP

Algorithm

General Object Detection

Single Stage: YOLOv7、YOLOv5、YOLOX;
Two Stage: Mask-RCNN、Faster-RCNN;
Others: Transformer series、S2ANet、SOLOv2

Multi-Objects Tracking

Single Stage: FairMOT、JDE
Two Stage: ByteTrack、DeepSORT、OC-SORT

Human Keypoint Detection

Top Down: HRNet、DarkPose、LiteHRNet
Bottom Up: HigherHRNet、SWAHR

Courses

AI Fast Track

Industrial object detection technology and a whole process practice

Freshman Learning Camp

Object detection learning camp for 7 days

Industry Lecture
- Smart city session • Smart manufacture session • Smart finance session

Business Corporation

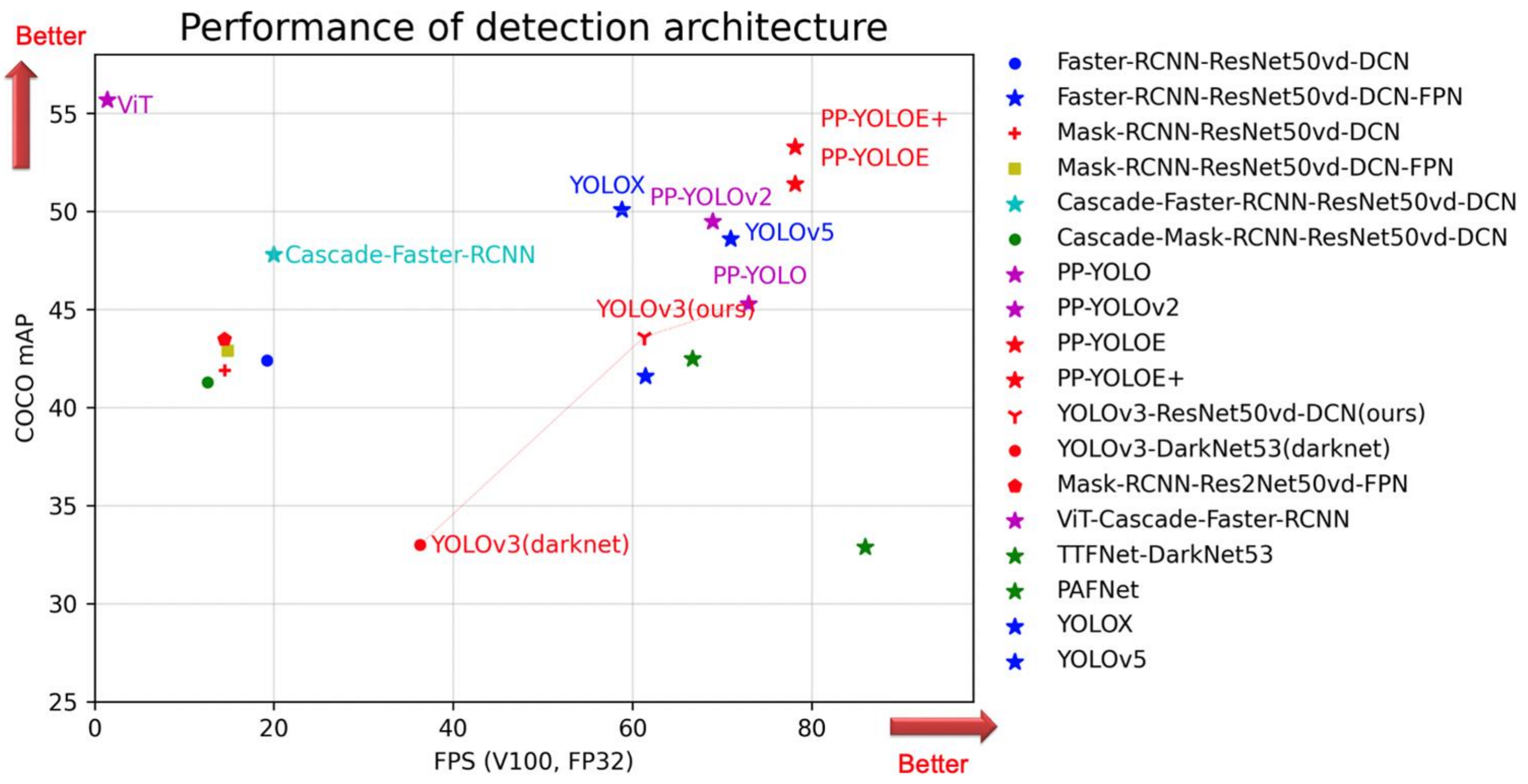
Joint Promotion

Free tech support and joint promotion opportunities

Co-Construction

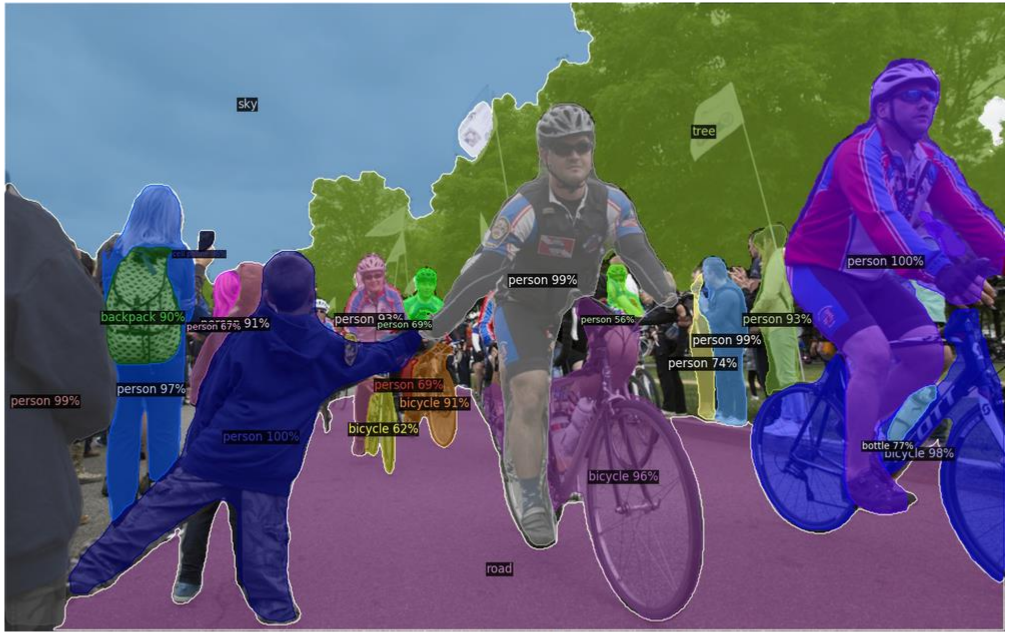
Co-construct the ecology of PaddlePaddle

PaddleDectection



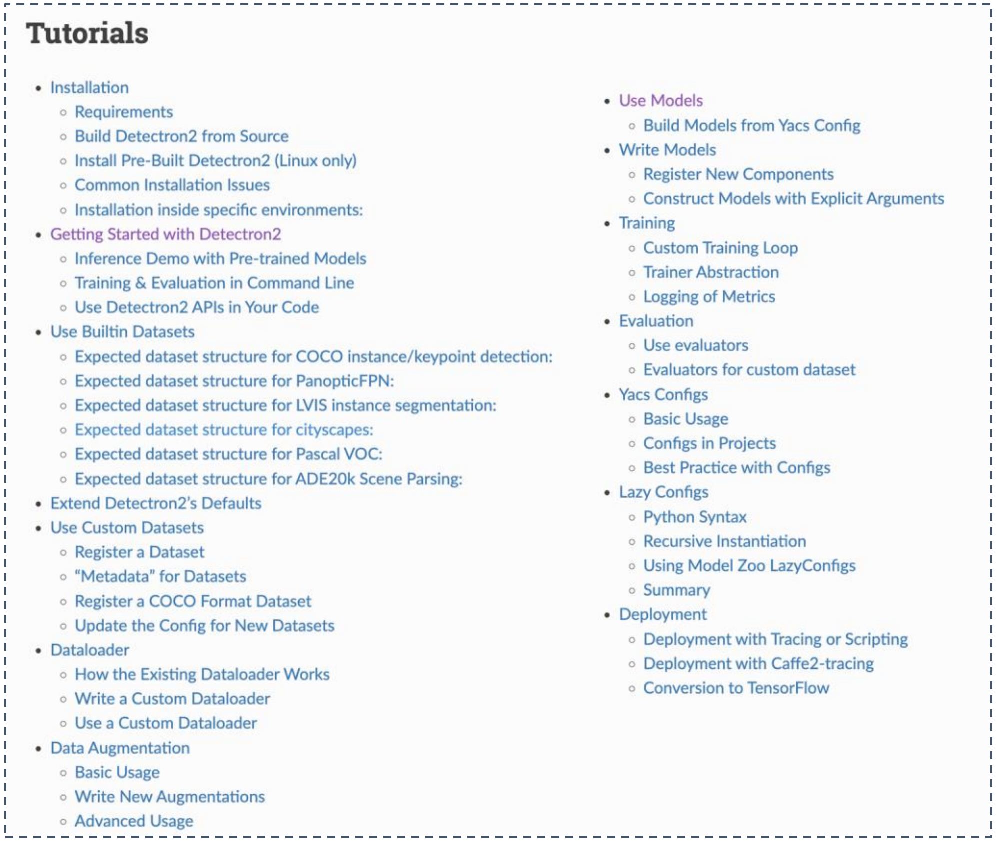
Detectron2

- Detectron2 is Facebook AI Research next generation library that provides SoTA detection and segmentation algorithms.
- It is the successor of Detectron and maskrcnn-benchmark.
- Support a number of computer vision research projects and production applications in Facebook.



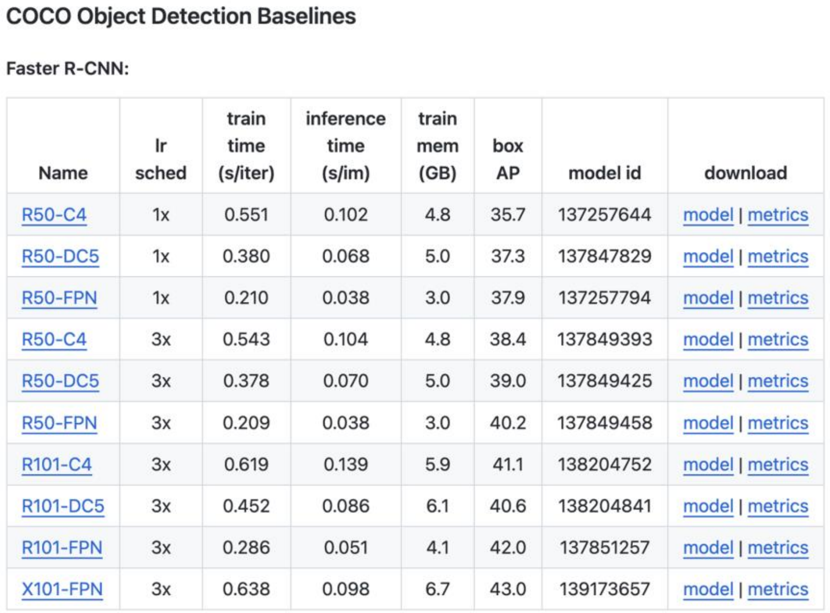
AI VIETNAM
All-in-One Course

Detectron2



Tutorials

- Installation
  - Requirements
  - Build Detectron2 from Source
  - Install Pre-Built Detectron2 (Linux only)
  - Common Installation Issues
  - Installation inside specific environments:
- Getting Started with Detectron2
  - Inference Demo with Pre-trained Models
  - Training & Evaluation in Command Line
  - Use Detectron2 APIs in Your Code
- Use Builtin Datasets
  - Expected dataset structure for COCO instance/keypoint detection:
  - Expected dataset structure for PanopticFPN:
  - Expected dataset structure for LVIS instance segmentation:
  - Expected dataset structure for cityscapes:
  - Expected dataset structure for Pascal VOC:
  - Expected dataset structure for ADE20k Scene Parsing:
- Extend Detectron2's Defaults
- Use Custom Datasets
  - Register a Dataset
  - "Metadata" for Datasets
  - Register a COCO Format Dataset
  - Update the Config for New Datasets
- Dataloader
  - How the Existing Dataloader Works
  - Write a Custom Dataloader
  - Use a Custom Dataloader
- Data Augmentation
  - Basic Usage
  - Write New Augmentations
  - Advanced Usage
- Use Models
  - Build Models from Yacs Config
- Write Models
  - Register New Components
  - Construct Models with Explicit Arguments
- Training
  - Custom Training Loop
  - Trainer Abstraction
  - Logging of Metrics
- Evaluation
  - Use evaluators
  - Evaluators for custom dataset
- Yacs Configs
  - Basic Usage
  - Configs in Projects
  - Best Practice with Configs
- Lazy Configs
  - Python Syntax
  - Recursive Instantiation
  - Using Model Zoo LazyConfigs
  - Summary
- Deployment
  - Deployment with Tracing or Scripting
  - Deployment with Caffe2-tracing
  - Conversion to TensorFlow

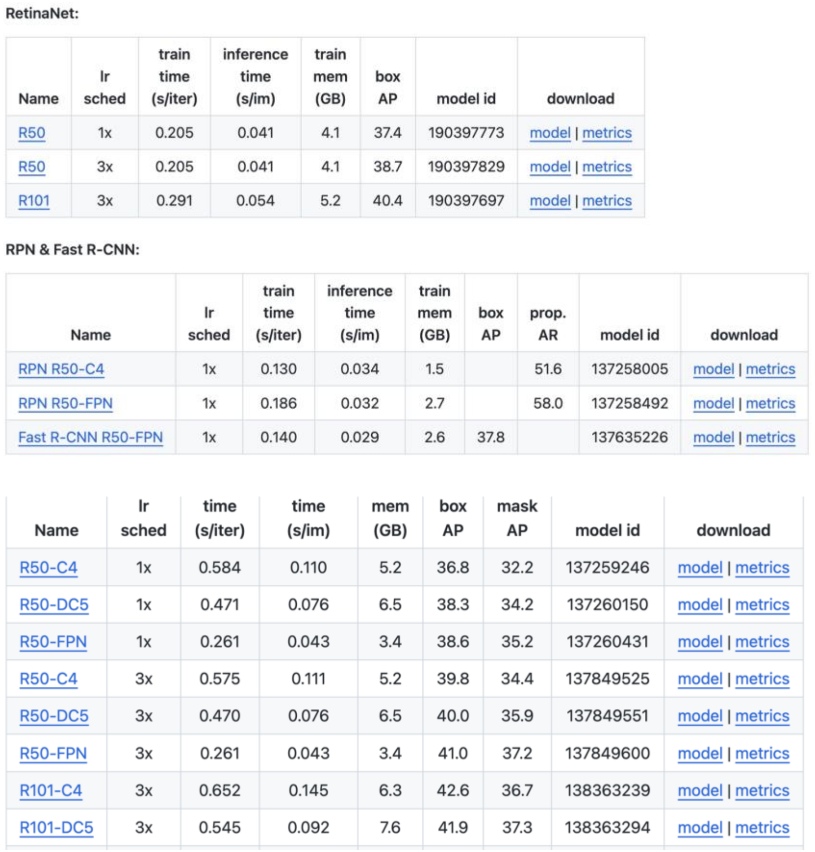


COCO Object Detection Baselines

Faster R-CNN:

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download
R50-C4	1x	0.551	0.102	4.8	35.7	137257644	model | metrics
R50-DC5	1x	0.380	0.068	5.0	37.3	137847829	model | metrics
R50-FPN	1x	0.210	0.038	3.0	37.9	137257794	model | metrics
R50-C4	3x	0.543	0.104	4.8	38.4	137849393	model | metrics
R50-DC5	3x	0.378	0.070	5.0	39.0	137849425	model | metrics
R50-FPN	3x	0.209	0.038	3.0	40.2	137849458	model | metrics
R101-C4	3x	0.619	0.139	5.9	41.1	138204752	model | metrics
R101-DC5	3x	0.452	0.086	6.1	40.6	138204841	model | metrics
R101-FPN	3x	0.286	0.051	4.1	42.0	137851257	model | metrics
X101-FPN	3x	0.638	0.098	6.7	43.0	139173657	model | metrics

Detectron2



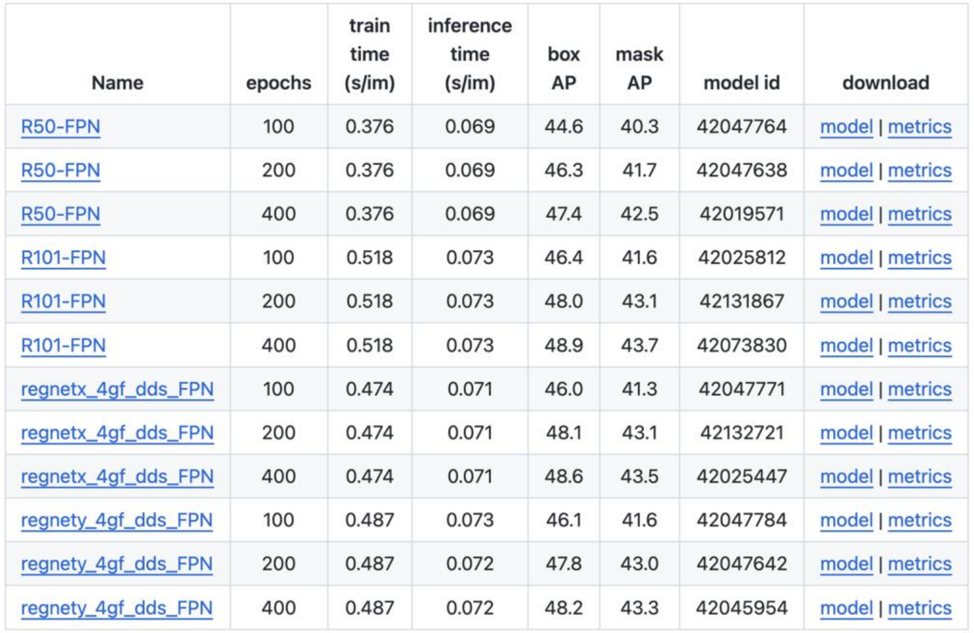
RetinaNet:

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download
R50	1x	0.205	0.041	4.1	37.4	190397773	model | metrics
R50	3x	0.205	0.041	4.1	38.7	190397829	model | metrics
R101	3x	0.291	0.054	5.2	40.4	190397697	model | metrics

RPN & Fast R-CNN:

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	prop. AR	model id	download
RPN R50-C4	1x	0.130	0.034	1.5		51.6	137258005	model | metrics
RPN R50-FPN	1x	0.186	0.032	2.7		58.0	137258492	model | metrics
Fast R-CNN R50-FPN	1x	0.140	0.029	2.6	37.8		137635226	model | metrics

Name	lr sched	time (s/iter)	time (s/im)	mem (GB)	box AP	mask AP	model id	download
R50-C4	1x	0.584	0.110	5.2	36.8	32.2	137259246	model | metrics
R50-DC5	1x	0.471	0.076	6.5	38.3	34.2	137260150	model | metrics
R50-FPN	1x	0.261	0.043	3.4	38.6	35.2	137260431	model | metrics
R50-C4	3x	0.575	0.111	5.2	39.8	34.4	137849525	model | metrics
R50-DC5	3x	0.470	0.076	6.5	40.0	35.9	137849551	model | metrics
R50-FPN	3x	0.261	0.043	3.4	41.0	37.2	137849600	model | metrics
R101-C4	3x	0.652	0.145	6.3	42.6	36.7	138363239	model | metrics
R101-DC5	3x	0.545	0.092	7.6	41.9	37.3	138363294	model | metrics



Name	epochs	train time (s/im)	inference time (s/im)	box AP	mask AP	model id	download
R50-FPN	100	0.376	0.069	44.6	40.3	42047764	model | metrics
R50-FPN	200	0.376	0.069	46.3	41.7	42047638	model | metrics
R50-FPN	400	0.376	0.069	47.4	42.5	42019571	model | metrics
R101-FPN	100	0.518	0.073	46.4	41.6	42025812	model | metrics
R101-FPN	200	0.518	0.073	48.0	43.1	42131867	model | metrics
R101-FPN	400	0.518	0.073	48.9	43.7	42073830	model | metrics
regnetx_4gf_dds_FPN	100	0.474	0.071	46.0	41.3	42047771	model | metrics
regnetx_4gf_dds_FPN	200	0.474	0.071	48.1	43.1	42132721	model | metrics
regnetx_4gf_dds_FPN	400	0.474	0.071	48.6	43.5	42025447	model | metrics
regnety_4gf_dds_FPN	100	0.487	0.073	46.1	41.6	42047784	model | metrics
regnety_4gf_dds_FPN	200	0.487	0.072	47.8	43.0	42047642	model | metrics
regnety_4gf_dds_FPN	400	0.487	0.072	48.2	43.3	42045954	model | metrics