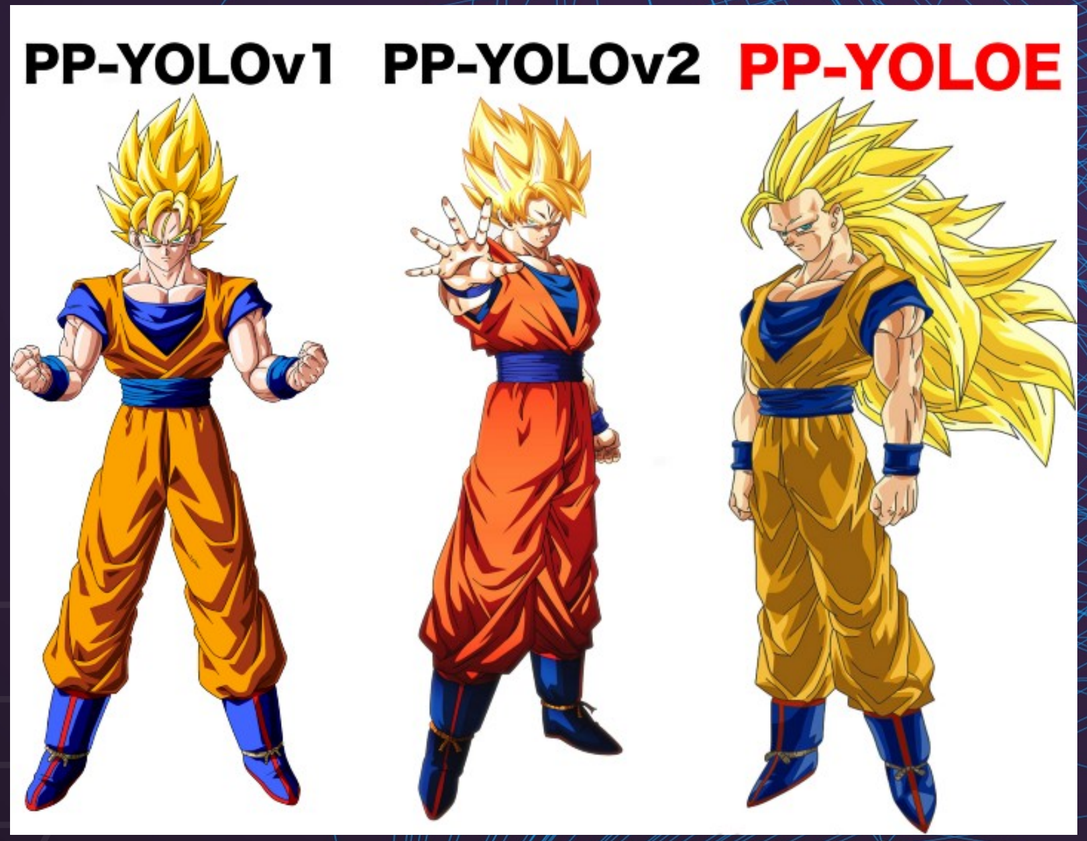
YOLO FAMILY



PP-YOLO V2

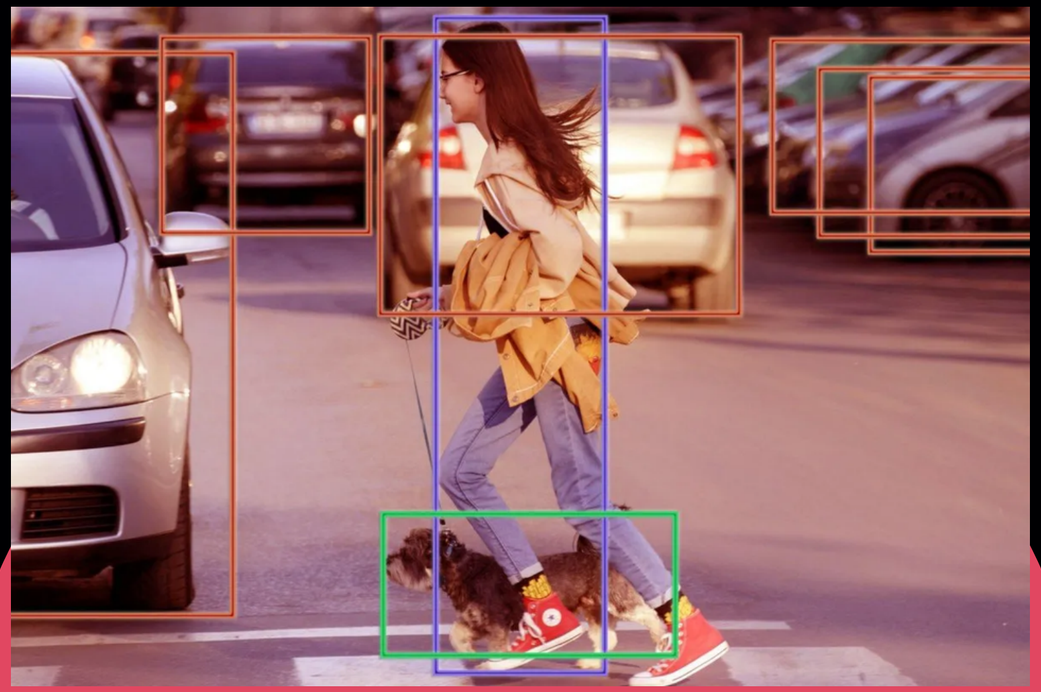
PP-YOLOE

PART TWO

VISUAL LEARNERS

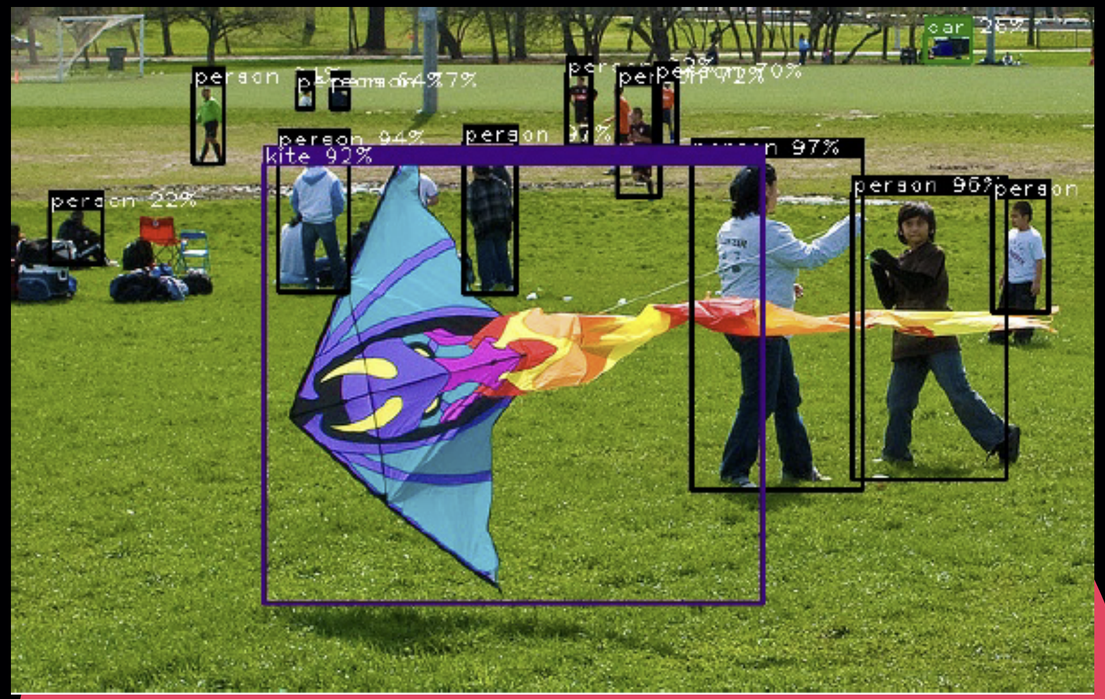

VISUAL LEARNERS

RECAP



*So last time we covered that Baidu, started out creating their own flavour of the YOLO Objector Detector that is based on the already popular **YOLOv3** for which they branded it as **PP-YOLO**.*

RECAP



__PP-YOLO__ took the __YOLOv3__ network and made iterative improvements in training and architecture by adding:

Resnet backbone, DropBlock regularization, IoU loss, Grid Sensitivity, Matrix NMS, CoordConv, SPP layers,** and **a better ImageNet pretraining checkpoint.

PP-YOLO V2

In April 2021, Baidu released their second iteration of PP-YOLO, called PP-YOLOv2... surprise surprise.

PP-YOLO V2

okay, the competition is getting heated now.

According to their paper called PP-YOLOv2 — A practical Object Detector, they have mentioned that they had surpassed existing object detectors like YOLOv4-CSP and YOLOv5-l with the same amount of parameters.

PP-YOLO

YOLOV2 VS PP-YOLOV2

The object detector PP-YOLOv2 improves upon YOLOv2 in several ways:

01

PANet

The FPN includes a Path Aggregation Network to create bottom-up pathways.

02

Backbone

A ResNet50-vd backbone with deformable convolution

03

Head

A lightweight IoU aware head. the authors tune YOLO's loss function to make it more aware of the overlap between bounding boxes

04

Mish

They added the mish activation function to the neck of their object detector.
(but not to the backbone, preferring to use the ImageNet pretrained backbone).

05

Input size

Next, they increase image input size, drawing uniformly across different input sizes. The input volume has increased.

06

Soft Label

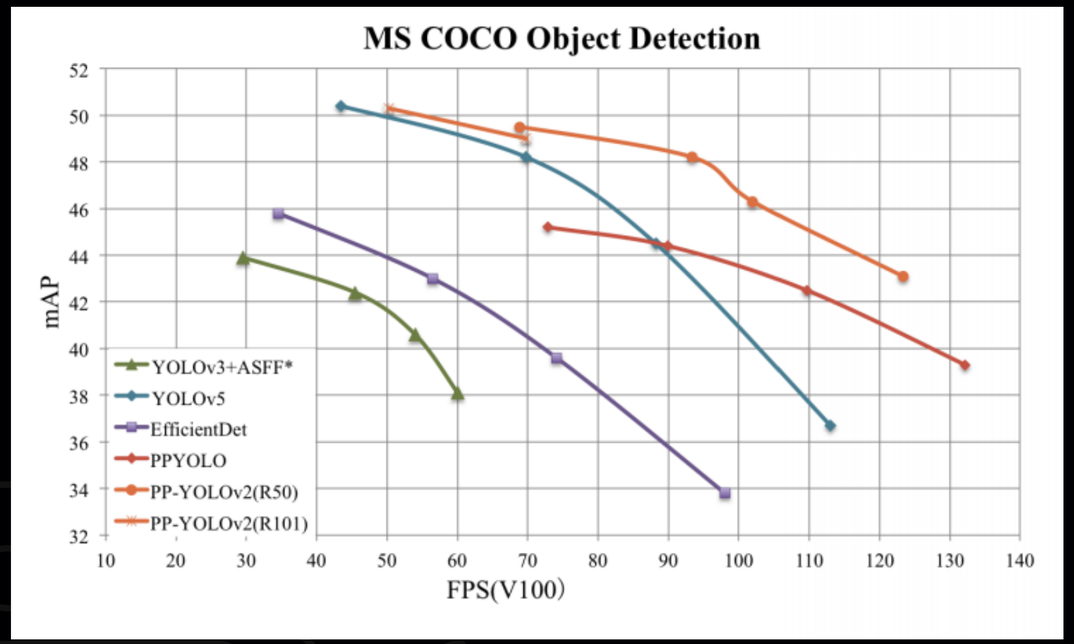
A soft label format is used to calculate an IoU-aware branch.

SIGNIFICANCE OF PP-YOLOV2

The main significance of **PP-YOLOv2** is that it primarily aims to combine multiple already-used techniques that rarely increase the number of model parameters and FLOPs, to improve the detector's accuracy as much as possible while keeping **the speed nearly the same**.

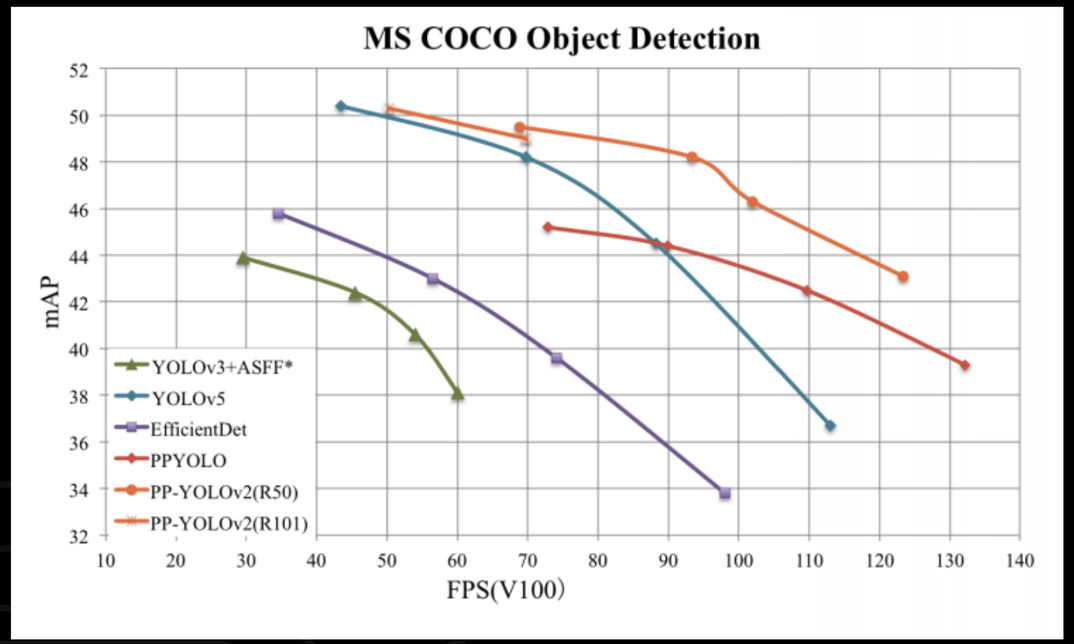
The term **FLOPS** describes how many floating point operations a computing device can carry out in a single second.

RESULTS



- Using these techniques, the authors compare their newly forged **PP-YOLOv2** against **other similar object detection models** in deployment conditions.
- For the evaluation they evaluate on the well-known Microsoft **Common Objects in Context (COCO) dataset**.
- **Inference time** is measured as the time it takes to infer against a single frame on a **Tesla V100 GPU**. Inference speeds are measured both with and without **TensorRT acceleration**.

RESULTS



In **PP-YOLOv2**, we increase the PP-performance of YOLOs from **45.9% mAP** to **49.5% mAP** by combining several efficient enhancements.

PP-YOLOv2 operates at **68.9FPS** with a **640×640 input size**.

The training batch size has increased from **64 to 192**, and the **Darknet53 backbone** of **YOLO v3** has been replaced with a **ResNet backbone (as a mini-batch size of 24 on 8 GPUs)**.

ISSUES WITH PP-YOLOV2

Even though it has overcome a lot of problems faced in the other detection models, there are some minor issues with **PP-YOLOv2**:

- Because each **grid** can only **detect one object,** performance suffers when there are **clusters of little objects**. These clusters of many tiny objects fall under many grids making the decision more difficult.
- Because **the ratio of the bounding box** is entirely learned from data, the main error of YOLO is from **localization**, and PP-YOLOv2 makes errors on the **uncommon ratio of the bounding box**.
- The **localization** and **the uncommon ratio** of the bounding box are major disadvantages to this model.

ENTER PP-YOLOE

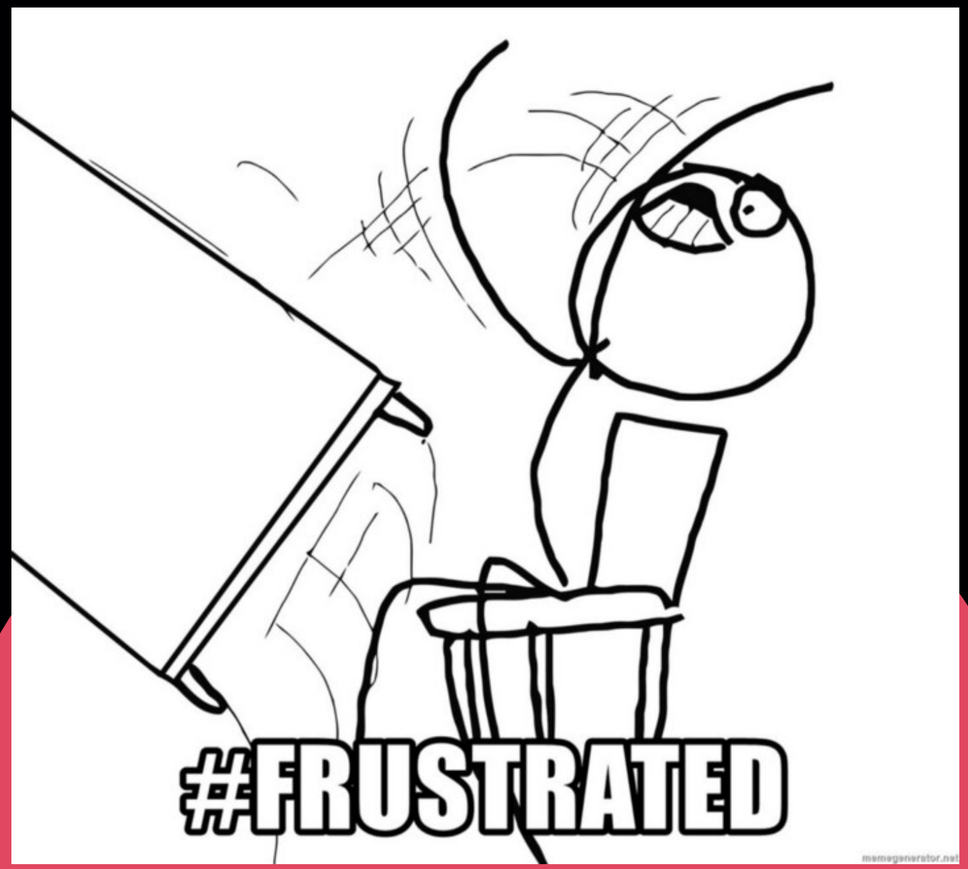
This brings us to PP-YOLOE.

Now, I don't know whether it was the pandemic or a change of management.

But why did they call it YOLOE?

Could they not have called it something like PP-YOLOv3.

ENTER PP-YOLOE

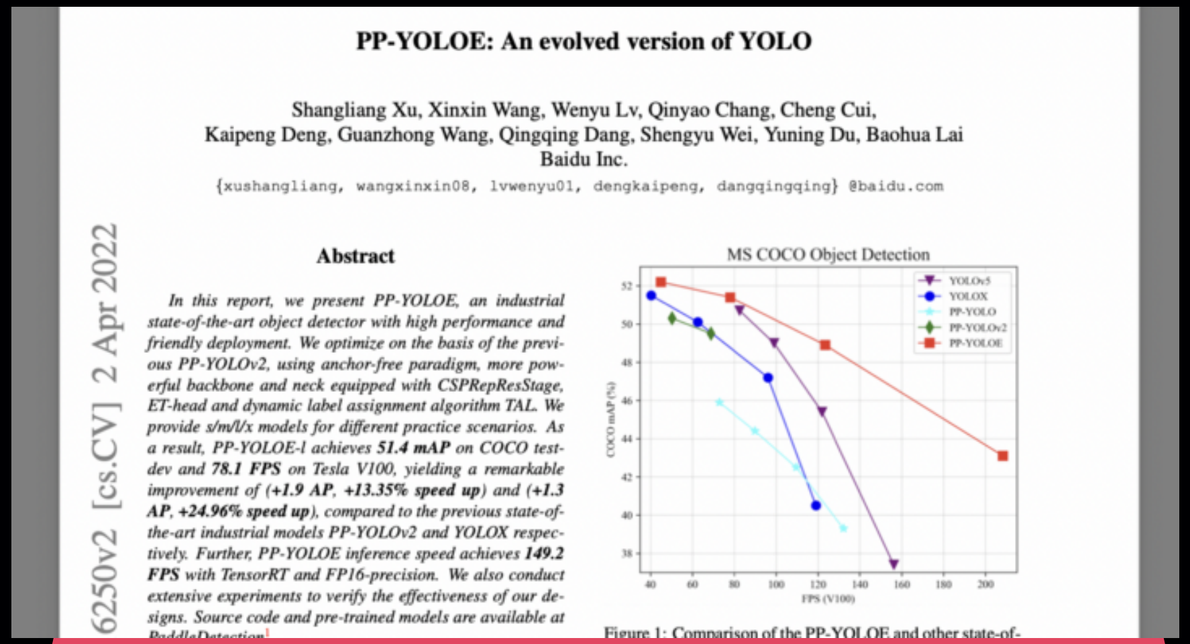


Was it because of all the newer models being called YOLOX, YOLOR, YOLOP at that time?

Well we don't have an answer for that.

The only reasonable explanation is that maybe PP-YOLOv3 would have been confused with YOLOv3, in terms of search engine optimization. I don’t know honestly.

PP-YOLOE



PP-YOLOE: An evolved version of YOLO

Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui,
Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, Baohua Lai
Baidu Inc.
{xushangliang, wangxinxin08, lvwenyu01, dengkaipeng, dangqingqing} @baidu.com

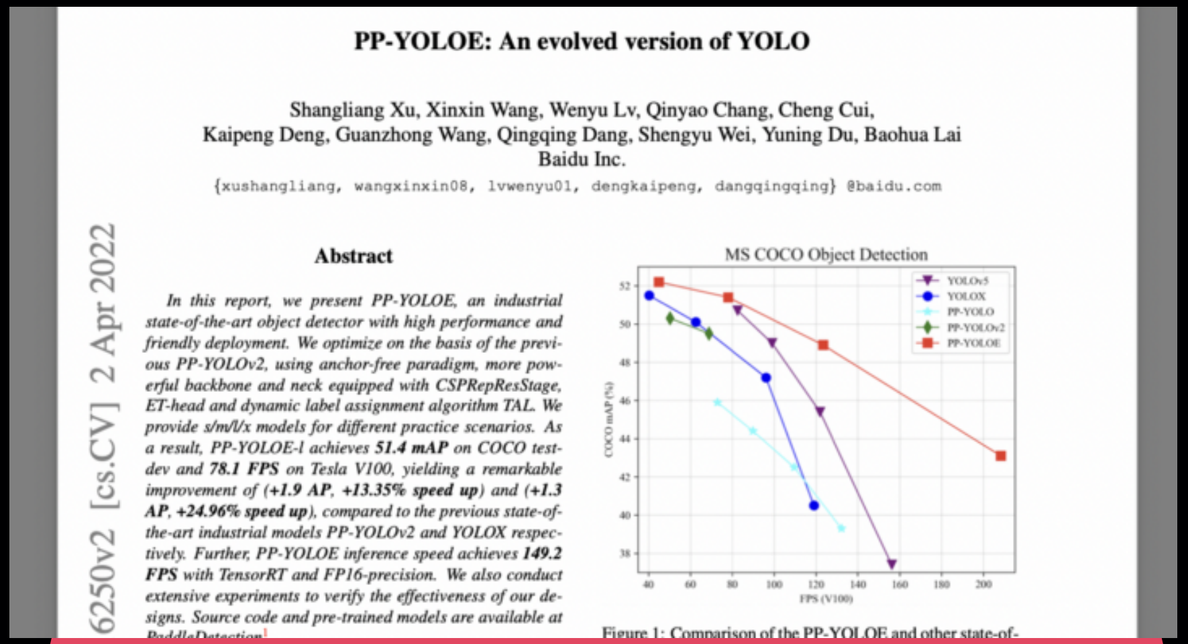
Abstract

*In this report, we present PP-YOLOE, an industrial state-of-the-art object detector with high performance and friendly deployment. We optimize on the basis of the previous PP-YOLOv2, using anchor-free paradigm, more powerful backbone and neck equipped with CSPRepResStage, ET-head and dynamic label assignment algorithm TAL. We provide s/m/l/x models for different practice scenarios. As a result, PP-YOLOE-l achieves **51.4 mAP** on COCO test-dev and **78.1 FPS** on Tesla V100, yielding a remarkable improvement of (**+1.9 AP**, **+13.35% speed up**) and (**+1.3 AP**, **+24.96% speed up**), compared to the previous state-of-the-art industrial models PP-YOLOv2 and YOLOX respectively. Further, PP-YOLOE inference speed achieves **149.2 FPS** with TensorRT and FP16-precision. We also conduct extensive experiments to verify the effectiveness of our designs. Source code and pre-trained models are available at PaddleDetection.*

Figure 1: Comparison of the PP-YOLOE and other state-of-

Anyways on the **2nd of April 2022**, Baidu released their subsequent paper **PP-YOLOE — An Evolved version of YOLO**, which is said to be an industrial state-of-the-art object detector with high performance and friendly deployment.

PP-YOLOE



PP-YOLOE: An evolved version of YOLO

Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui,
Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, Baohua Lai
Baidu Inc.
{xushangliang, wangxinxin08, lvwenyu01, dengkaipeng, dangqingqing} @baidu.com

Abstract

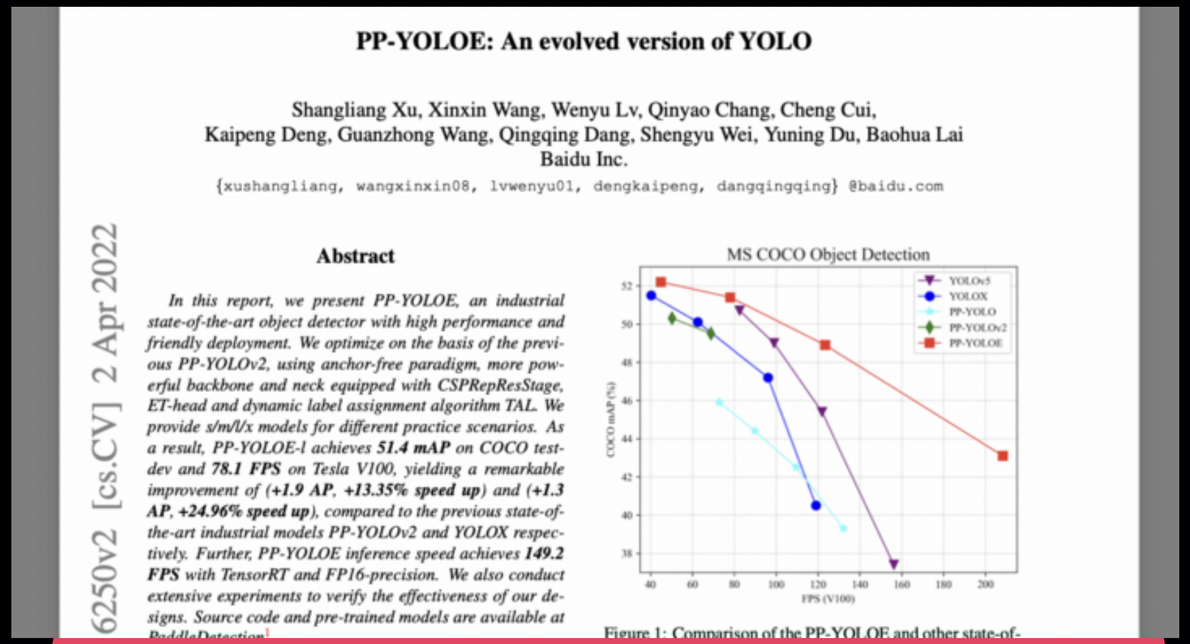
*In this report, we present PP-YOLOE, an industrial state-of-the-art object detector with high performance and friendly deployment. We optimize on the basis of the previous PP-YOLOv2, using anchor-free paradigm, more powerful backbone and neck equipped with CSPRepResStage, ET-head and dynamic label assignment algorithm TAL. We provide s/m/l/x models for different practice scenarios. As a result, PP-YOLOE-l achieves **51.4 mAP** on COCO test-dev and **78.1 FPS** on Tesla V100, yielding a remarkable improvement of (**+1.9 AP**, **+13.35% speed up**) and (**+1.3 AP**, **+24.96% speed up**), compared to the previous state-of-the-art industrial models PP-YOLOv2 and YOLOX respectively. Further, PP-YOLOE inference speed achieves **149.2 FPS** with TensorRT and FP16-precision. We also conduct extensive experiments to verify the effectiveness of our designs. Source code and pre-trained models are available at PaddleDetection.*

Figure 1: Comparison of the PP-YOLOE and other state-of-

*They state that they were inspired by **YOLOX** for its **anchor-free method** equipped with dynamic label assignment to improve the performance of the detector, which they claim has significantly outperformed **YOLOv5** in terms of precision.*

Knowing this, they further optimised their previous work in **PP-YOLOV2**.

PP-YOLOE



PP-YOLOE: An evolved version of YOLO

Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui,
Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, Baohua Lai
Baidu Inc.
{xushangliang, wangxinxin08, lvwenyu01, dengkaipeng, dangqingqing} @baidu.com

Abstract

*In this report, we present PP-YOLOE, an industrial state-of-the-art object detector with high performance and friendly deployment. We optimize on the basis of the previous PP-YOLOv2, using anchor-free paradigm, more powerful backbone and neck equipped with CSPRepResStage, ET-head and dynamic label assignment algorithm TAL. We provide s/m/l/x models for different practice scenarios. As a result, PP-YOLOE-l achieves **51.4 mAP** on COCO test-dev and **78.1 FPS** on Tesla V100, yielding a remarkable improvement of (**+1.9 AP**, **+13.35% speed up**) and (**+1.3 AP**, **+24.96% speed up**), compared to the previous state-of-the-art industrial models PP-YOLOv2 and YOLOX respectively. Further, PP-YOLOE inference speed achieves **149.2 FPS** with TensorRT and FP16-precision. We also conduct extensive experiments to verify the effectiveness of our designs. Source code and pre-trained models are available at PaddleDetection.*

Figure 1: Comparison of the PP-YOLOE and other state-of-

*Similar to **YOLOv3**, **PP-YOLOv2** only assigns one **anchor box** for each **ground truth object**.*

This mechanism however requires a number of **extra hyperparameters** and relies heavily on **hand-crafted design** that may not generalize well when trained on other datasets.

IMPROVEMENTS

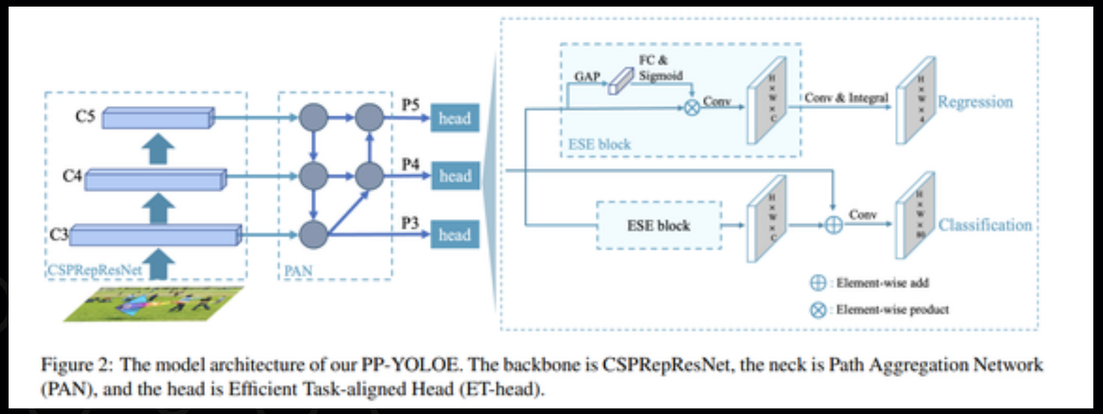


Figure 2: The model architecture of our PP-YOLOE. The backbone is CSPRepResNet, the neck is Path Aggregation Network (PAN), and the head is Efficient Task-aligned Head (ET-head).

To address this issue, the Baidu researchers introduce an **anchor-free method** to **PP-YOLOv2** that tiles **one anchor point on each pixel** and sets **upper and lower bounds** for detection heads to assign ground truths to a corresponding feature map.

The center of a bounding box can then be calculated to select the **closest pixel as positive samples**.

A **4D vector** is also predicted for **regression**, with the modifications resulting in slight **model speedups** and **precision drops**.

PP-YOLO

IMPROVEMENTS

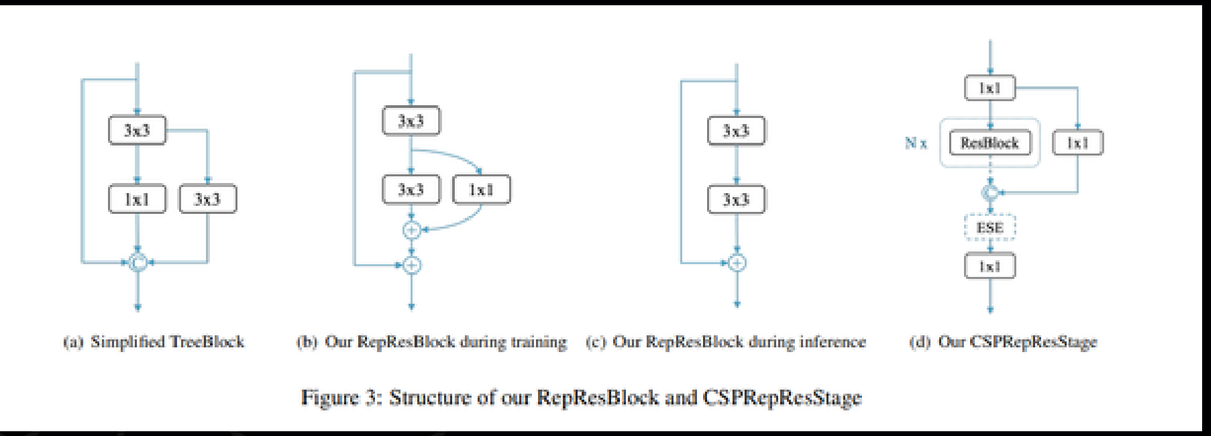


Figure 3: Structure of our RepResBlock and CSPRepResStage

The team gains **backbone** and **neck** improvements by using a novel **RepResBlock** to build a **CSPRepResNet backbone** with one stem composed of **three convolution layers** and **four subsequent stages** stacked by **RepResBlock**.

In each stage, **cross-stage partial connections** are used to **reduce parameters** and **computational burden**. Following **PP-YOLOv2**, the team also builds a **neck** with **RepResBlock** and **CSPRepResStage**.

IMPROVEMENTS

Model	mAP(%)	Parameters(M)	GFLOPs	Latency(ms)	FPS
PP-YOLOv2 baseline model	49.1	54.58	115.77	14.5	68.9
+Anchor-free	48.8 (−0.3)	54.27	114.78	14.3	69.8
+CSPRepResNet	49.5 (+0.7)	47.42	101.87	11.7	85.5
+TAL	50.4 (+0.9)	48.32	104.75	11.9	84.0
+ET-head	**50.9** (+0.5)	52.20	110.07	12.8	78.1

Table 2: Ablation study of PP-YOLOE-l on COCO *val*. We use 640×640 resolution as input with FP32-precision, and test on Tesla V100 without post-processing.

The researchers also use **task alignment learning (TAL/TOOD, proposed by Feng et al. in 2021)** to replace **label assignment**, which achieves a **0.9 percent AP improvement** and pushes **overall AP** to **50.4 percent**.

IMPROVEMENTS

Model	mAP(%)	Parameters(M)	GFLOPs	Latency(ms)	FPS
PP-YOLOv2 baseline model	49.1	54.58	115.77	14.5	68.9
+Anchor-free	48.8 (−0.3)	54.27	114.78	14.3	69.8
+CSPRepResNet	49.5 (+0.7)	47.42	101.87	11.7	85.5
+TAL	50.4 (+0.9)	48.32	104.75	11.9	84.0
+ET-head	**50.9** (+0.5)	52.20	110.07	12.8	78.1

Table 2: Ablation study of PP-YOLOE-l on COCO *val*. We use 640×640 resolution as input with FP32-precision, and test on Tesla V100 without post-processing.

Finally, to solve the task conflict between **classification and localization** in **object detection**, the researchers use **effective squeeze and extraction (ESE)** to replace the **layer attention** in conventional **task-aligned one-stage object detection (TOOD)**, simplify the **alignment of classification branches** to **shortcuts**, and **replace the alignment of regression branches** with a **distribution focal loss (DFL)** layer.

Additional modifications to the resulting **Efficient Task-aligned Head (ET-head)** further improve performance.

RESULTS

Method	Backbone	Size	FPS (v100)		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
			w/o TRT	with TRT						
YOLOv3 + ASFF* [17]	Darknet-53	320	60	-	38.1%	57.4%	42.1%	16.1%	41.6%	53.6%
YOLOv3 + ASFF* [17]	Darknet-53	416	54	-	40.6%	60.6%	45.1%	20.3%	44.2%	54.1%
YOLOv4 [2]	CSPDarknet-53	416	96	-	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4 [2]	CSPDarknet-53	512	83	-	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4-CSP [26]	Modified CSPDarknet-53	512	97	-	46.2%	64.8%	50.2%	24.6%	50.4%	61.9%
YOLOv4-CSP [26]	Modified CSPDarknet-53	640	73	-	47.5%	66.2%	51.7%	28.2%	51.2%	59.8%
EfficientDet-D0 [24]	Efficient-B0	512	98.0	-	33.8%	52.2%	35.8%	12.0%	38.3%	51.2%
EfficientDet-D1 [24]	Efficient-B1	640	74.1	-	39.6%	58.6%	42.3%	17.9%	44.3%	56.0%
EfficientDet-D2 [24]	Efficient-B2	768	56.5	-	43.0%	62.3%	46.2%	22.5%	47.0%	58.4%
EfficientDet-D2 [24]	Efficient-B3	896	34.5	-	45.8%	65.0%	49.3%	26.6%	49.4%	59.8%
PP-YOLO [18]	ResNet50-vd-dcn	320	132.2^{+}	242.2^{+}	39.3%	59.3%	42.7%	16.7%	41.4%	57.8%
PP-YOLO [18]	ResNet50-vd-dcn	416	109.6^{+}	215.4^{+}	42.5%	62.8%	46.5%	21.2%	45.2%	58.2%
PP-YOLO [18]	ResNet50-vd-dcn	512	89.9^{+}	188.4^{+}	44.4%	64.6%	48.8%	24.4%	47.1%	58.2%
PP-YOLO [18]	ResNet50-vd-dcn	608	72.9^{+}	155.6^{+}	45.9%	65.2%	49.9%	26.3%	47.8%	57.2%
PP-YOLOv2 [13]	ResNet50-vd-dcn	320	123.3	152.9	43.1%	61.7%	46.5%	19.7%	46.3%	61.8%
PP-YOLOv2 [13]	ResNet50-vd-dcn	416	102^{+}	145.1^{+}	46.3%	65.1%	50.3%	23.9%	50.2%	62.2%
PP-YOLOv2 [13]	ResNet50-vd-dcn	512	93.4^{+}	141.2^{+}	48.2%	67.1%	52.7%	27.7%	52.1%	62.1%
PP-YOLOv2 [13]	ResNet50-vd-dcn	640	68.9^{+}	106.5^{+}	49.5%	68.2%	54.4%	30.7%	52.9%	61.2%
PP-YOLOv2 [13]	ResNet101-vd-dcn	640	50.3^{+}	87.0^{+}	50.3%	69.0%	55.3%	31.6%	53.9%	62.4%
YOLOv5-s [14]	Modified CSP v6	640	156.2^{+}	454.5^{*}	37.4%	56.8%	-	-	-	-
YOLOv5-m [14]	Modified CSP v6	640	121.9^{+}	263.1^{*}	45.4%	64.1%	-	-	-	-
YOLOv5-l [14]	Modified CSP v6	640	99.0^{+}	172.4^{*}	49.0%	67.3%	-	-	-	-
YOLOv5-x [14]	Modified CSP v6	640	82.6^{+}	117.6^{*}	50.7%	68.9%	-	-	-	-
YOLOX-s [6]	Modified CSP v5	640	119.0^{*} | 102.0^{+}	246.9^{*}	40.5%	-	-	-	-	-
YOLOX-m [6]	Modified CSP v5	640	96.1^{*} | 81.3^{+}	177.3^{*}	47.2%	-	-	-	-	-
YOLOX-l [6]	Modified CSP v5	640	62.5^{*} | 68.9^{+}	120.1^{*}	50.1%	-	-	-	-	-
YOLOX-x [6]	Modified CSP v5	640	40.3^{*} | 57.8^{+}	87.4^{*}	51.5%	-	-	-	-	-
PP-YOLOE-s	CSPRepResNet	640	208.3	333.3	**43.1%**	**60.5%**	**46.6%**	**23.2%**	**46.4%**	**56.9%**
PP-YOLOE-m	CSPRepResNet	640	123.4	208.3	**48.9%**	**66.5%**	**53.0%**	**28.6%**	**52.9%**	**63.8%**
PP-YOLOE-l	CSPRepResNet	640	78.1	149.2	**51.4%**	**68.9%**	**55.6%**	**31.4%**	**55.3%**	**66.1%**
PP-YOLOE-x	CSPRepResNet	640	45.0	95.2	**52.2%**	**69.9%**	**56.5%**	**33.3%**	**56.3%**	**66.4%**

Table 4: Comparison of the speed and accuracy of different object detectors on COCO 2017 *test-dev*. Results marked by "+" are updated results from the corresponding official release. Results marked by "*" are tested in our environment using official codebase and model. The input size of YOLOv5 is not exactly square of 640×640 in validation and speed test, so we skip it in the table. The default precision of speed is FP32 for *w/o trt* and FP16 for *with trt*. Moreover, we provide both FP32 and FP16 for YOLOX *w/o trt* scene, the FP32 speed on the left side of split line and FP16 speed on the right.

The researchers performed evaluation experiments on the **MS COCO-2017 training set**, comparing the proposed **PP-YOLOE** with state-of-the-art object detectors such as **YOLOX**, **YOLOv5** and **EfficientDet**.

RESULTS

Method	Backbone	Size	FPS (v100)		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
			w/o TRT	with TRT						
YOLOv3 + ASFF* [17]	Darknet-53	320	60	-	38.1%	57.4%	42.1%	16.1%	41.6%	53.6%
YOLOv3 + ASFF* [17]	Darknet-53	416	54	-	40.6%	60.6%	45.1%	20.3%	44.2%	54.1%
YOLOv4 [2]	CSPDarknet-53	416	96	-	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4 [2]	CSPDarknet-53	512	83	-	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4-CSP [26]	Modified CSPDarknet-53	512	97	-	46.2%	64.8%	50.2%	24.6%	50.4%	61.9%
YOLOv4-CSP [26]	Modified CSPDarknet-53	640	73	-	47.5%	66.2%	51.7%	28.2%	51.2%	59.8%
EfficientDet-D0 [24]	Efficient-B0	512	98.0	-	33.8%	52.2%	35.8%	12.0%	38.3%	51.2%
EfficientDet-D1 [24]	Efficient-B1	640	74.1	-	39.6%	58.6%	42.3%	17.9%	44.3%	56.0%
EfficientDet-D2 [24]	Efficient-B2	768	56.5	-	43.0%	62.3%	46.2%	22.5%	47.0%	58.4%
EfficientDet-D2 [24]	Efficient-B3	896	34.5	-	45.8%	65.0%	49.3%	26.6%	49.4%	59.8%
PP-YOLO [18]	ResNet50-vd-dcn	320	132.2^{+}	242.2^{+}	39.3%	59.3%	42.7%	16.7%	41.4%	57.8%
PP-YOLO [18]	ResNet50-vd-dcn	416	109.6^{+}	215.4^{+}	42.5%	62.8%	46.5%	21.2%	45.2%	58.2%
PP-YOLO [18]	ResNet50-vd-dcn	512	89.9^{+}	188.4^{+}	44.4%	64.6%	48.8%	24.4%	47.1%	58.2%
PP-YOLO [18]	ResNet50-vd-dcn	608	72.9^{+}	155.6^{+}	45.9%	65.2%	49.9%	26.3%	47.8%	57.2%
PP-YOLOv2 [13]	ResNet50-vd-dcn	320	123.3	152.9	43.1%	61.7%	46.5%	19.7%	46.3%	61.8%
PP-YOLOv2 [13]	ResNet50-vd-dcn	416	102^{+}	145.1^{+}	46.3%	65.1%	50.3%	23.9%	50.2%	62.2%
PP-YOLOv2 [13]	ResNet50-vd-dcn	512	93.4^{+}	141.2^{+}	48.2%	67.1%	52.7%	27.7%	52.1%	62.1%
PP-YOLOv2 [13]	ResNet50-vd-dcn	640	68.9^{+}	106.5^{+}	49.5%	68.2%	54.4%	30.7%	52.9%	61.2%
PP-YOLOv2 [13]	ResNet101-vd-dcn	640	50.3^{+}	87.0^{+}	50.3%	69.0%	55.3%	31.6%	53.9%	62.4%
YOLOv5-s [14]	Modified CSP v6	640	156.2^{+}	454.5^{*}	37.4%	56.8%	-	-	-	-
YOLOv5-m [14]	Modified CSP v6	640	121.9^{+}	263.1^{*}	45.4%	64.1%	-	-	-	-
YOLOv5-l [14]	Modified CSP v6	640	99.0^{+}	172.4^{*}	49.0%	67.3%	-	-	-	-
YOLOv5-x [14]	Modified CSP v6	640	82.6^{+}	117.6^{*}	50.7%	68.9%	-	-	-	-
YOLOX-s [6]	Modified CSP v5	640	119.0^{*} | 102.0^{+}	246.9^{*}	40.5%	-	-	-	-	-
YOLOX-m [6]	Modified CSP v5	640	96.1^{*} | 81.3^{+}	177.3^{*}	47.2%	-	-	-	-	-
YOLOX-l [6]	Modified CSP v5	640	62.5^{*} | 68.9^{+}	120.1^{*}	50.1%	-	-	-	-	-
YOLOX-x [6]	Modified CSP v5	640	40.3^{*} | 57.8^{+}	87.4^{*}	51.5%	-	-	-	-	-
PP-YOLOE-s	CSPRepResNet	640	208.3	333.3	**43.1%**	**60.5%**	**46.6%**	**23.2%**	**46.4%**	**56.9%**
PP-YOLOE-m	CSPRepResNet	640	123.4	208.3	**48.9%**	**66.5%**	**53.0%**	**28.6%**	**52.9%**	**63.8%**
PP-YOLOE-l	CSPRepResNet	640	78.1	149.2	**51.4%**	**68.9%**	**55.6%**	**31.4%**	**55.3%**	**66.1%**
PP-YOLOE-x	CSPRepResNet	640	45.0	95.2	**52.2%**	**69.9%**	**56.5%**	**33.3%**	**56.3%**	**66.4%**

Table 4: Comparison of the speed and accuracy of different object detectors on COCO 2017 *test-dev*. Results marked by "+" are updated results from the corresponding official release. Results marked by "*" are tested in our environment using official codebase and model. The input size of YOLOv5 is not exactly square of 640 × 640 in validation and speed test, so we skip it in the table. The default precision of speed is FP32 for *w/o trt* and FP16 for *with trt*. Moreover, we provide both FP32 and FP16 for YOLOX *w/o trt* scene, the FP32 speed on the left side of split line and FP16 speed on the right.

In the tests, the **PP-YOLOE-l** variant scored **51.4 percent AP** with **640 x 640 resolution** at a speed of **78.1 FPS**, an AP improvement of **1.9 percent** and a **13.35 percent speedup** compared to **PP-YOLOv2**, and a 1.**3 percent AP improvement** and **24.96 percent speedup** compared to **YOLOX**.

The proposed model's **inference speed** reached **149.2 FPS** with **TensorRT and FP16-precision**.

RESULTS

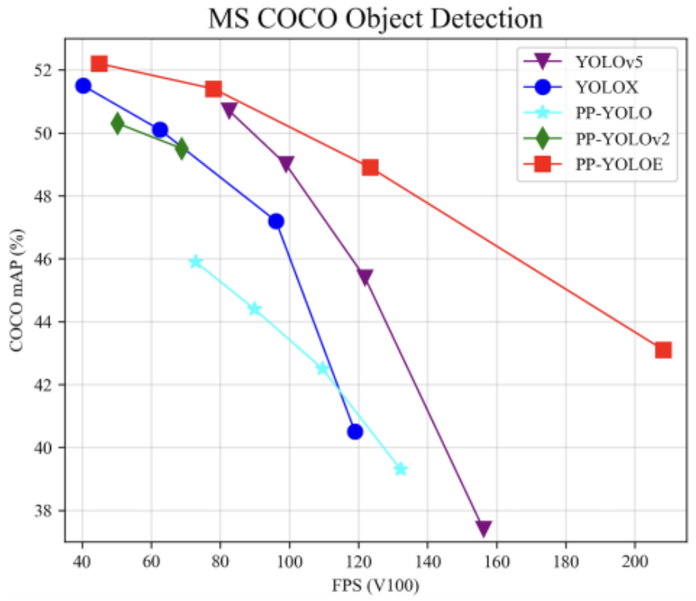


Figure 1: Comparison of the PP-YOLOE and other state-of-the-art models. PP-YOLOE-l achieves 51.4 mAP on COCO *test-dev* and 78.1 FPS on Tesla V100, obtains 1.9 AP and 9.2 FPS improvement compared with PP-YOLOv2[13].

Overall, **PP-YOLOE** is shown to be a high-performance object detector. The model series can also smoothly transition to deployment thanks to support from the **PaddlePaddle deep learning framework**.

The team hopes their design updates and encouraging results can inspire developers and researchers working in object detection.

THANK YOU

VISUAL LEARNERS