YOLO FAMILY

Baidu 百度

PP-YOLO

PART ONE

VISUAL LEARNERS

WHAT IS PP?

PP is short for Paddle Paddle.

No, not that type of Paddle you are thinking of.

Paddle stands for wait for it:

*(**PA**rallel **D**istributed **D**eep **LE**arning)*

WHAT IS PP?

If PaddlePaddle is new to you, then we are in the same boat.

It is a deep learning framework developed by Baidu.

(The Chinese version of Google).

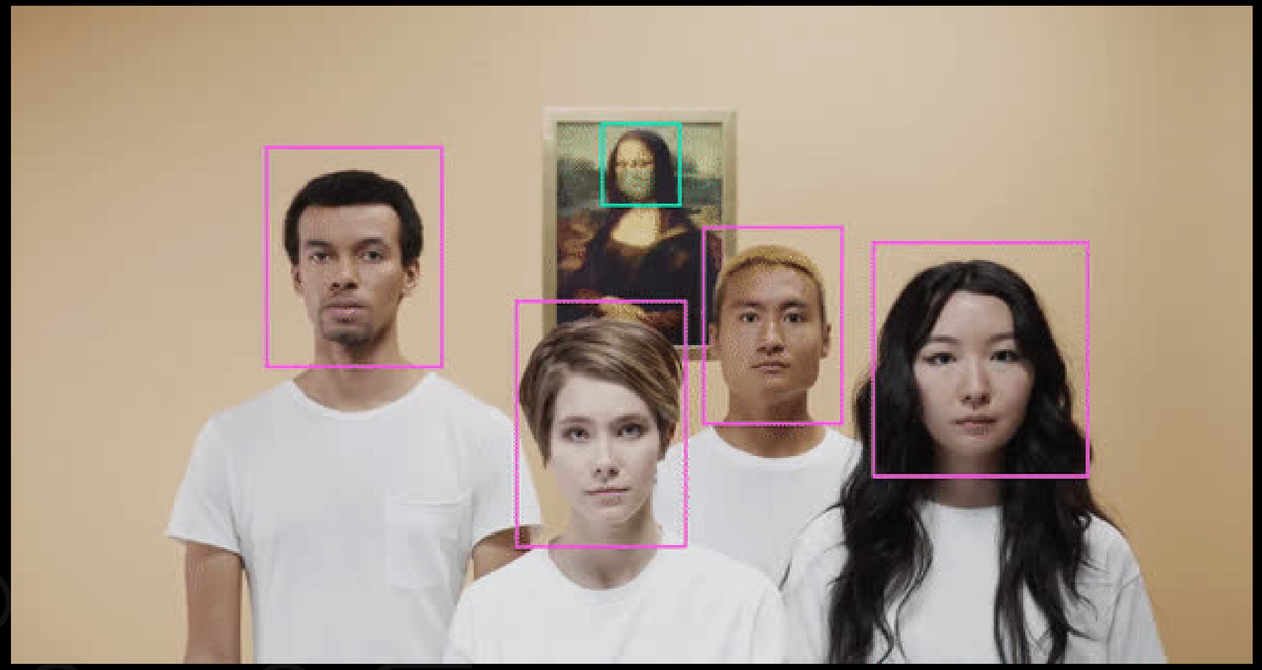
It includes the building blocks required to develop learning models.

PADDLEPADDLE DISTRIBUTIONS PROVIDED ON THEIR WEBSITE.

WHAT IS PP?

Primarily written in Python, PaddlePaddle seems akin to PyTorch and TensorFlow.

A deep dive into the PaddlePaddle framework is intriguing, but beyond the scope of this presentation.



PP-YOLO

Baidu publishes PP-YOLO and pushes the state of the art in object detection research by building on top of YOLOv3, the PaddlePaddle deep learning framework, and cutting edge computer vision research.

PP-YOLO evaluation metrics show improved performance over **YOLOv4**, the incumbent state of the art object detection model. Yet, the Baidu authors write:

MYSTERIOUS INTRODUCTION

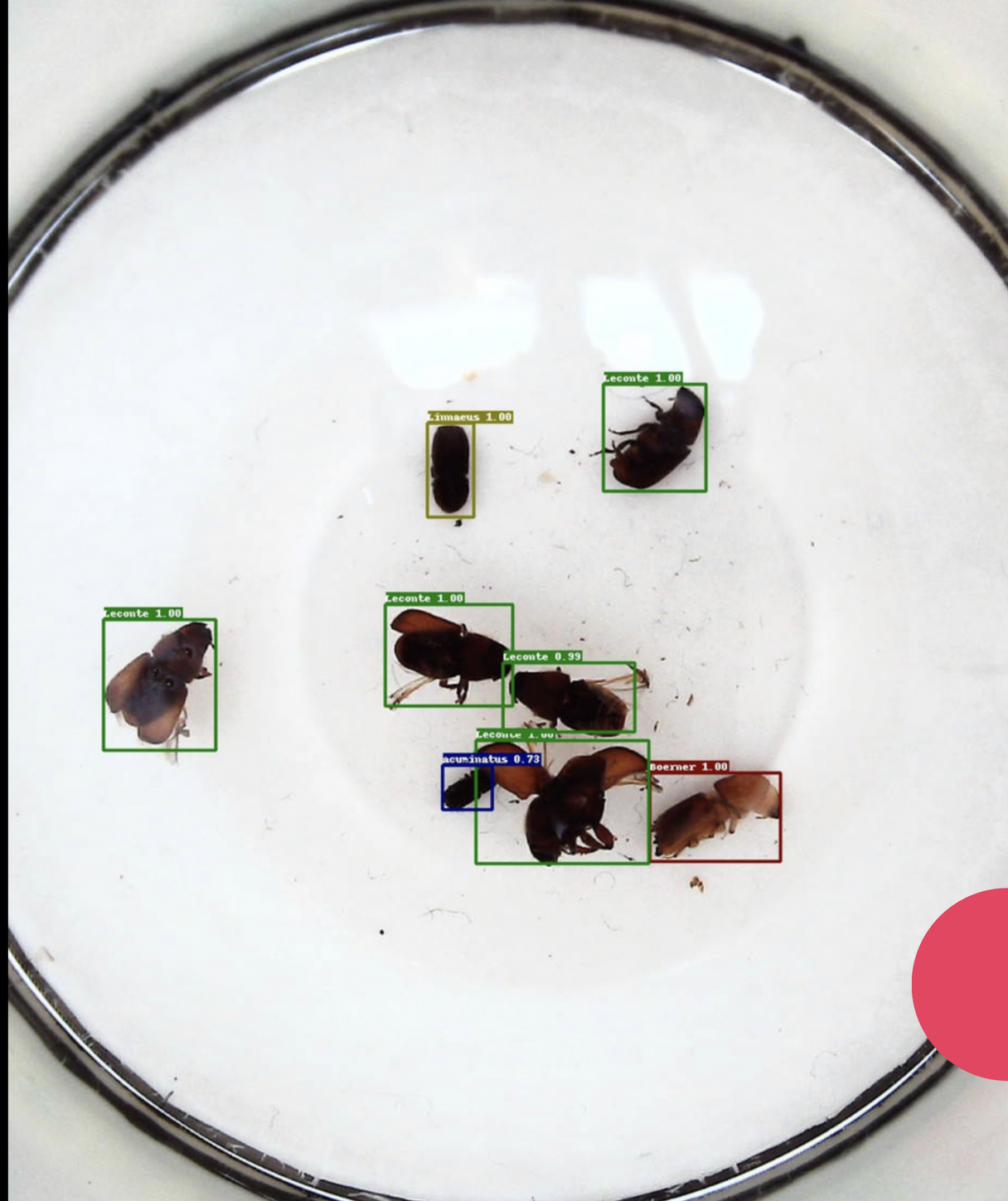
"This paper is not intended to introduce a novel object detecotor.

It is more like a recipe, which tell you how to build a better detector step by step."

PP-YOLO CONTRIBUTIONS

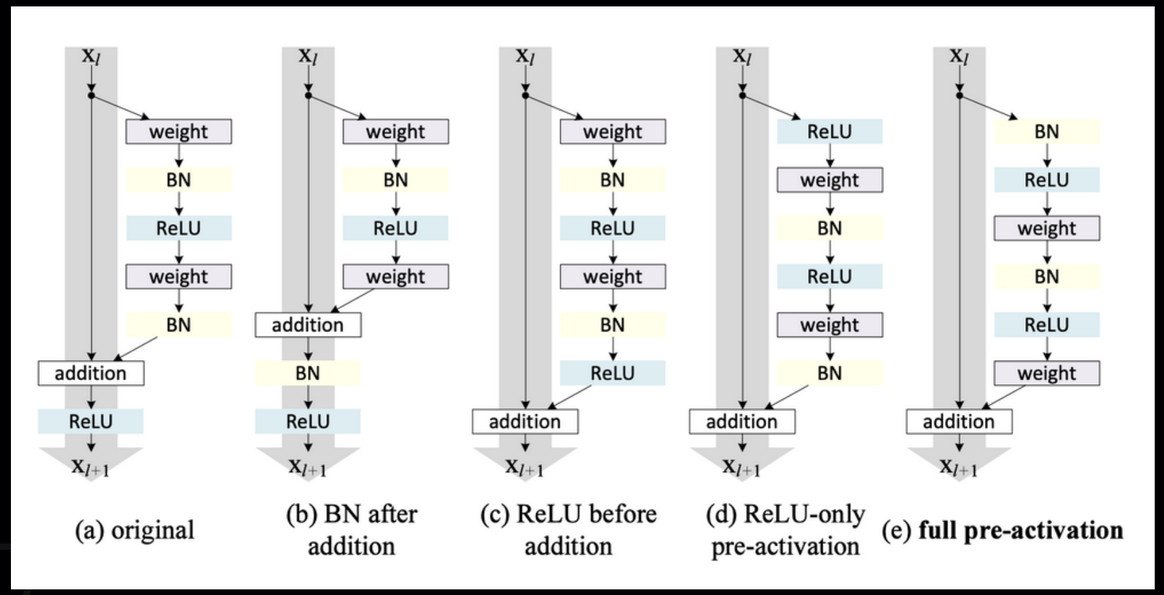
The PP-YOLO paper reads much like the YOLOv4 paper in that it is a compilation of techniques that are known to work in computer vision.

The novel contribution is to prove that the ensemble of these technologies improves performance, and to provide an ablation study of how much each step helps the model along the way.



PP YOLO

REPLACE BACKBONE



The first PP YOLO technique is to replace the **YOLOv3 Darknet53 backbone** with the **Resnet50-vd-dcn** ConvNet backbone

Resnet is a more popular backbone, more frameworks are optimized for its execution, and it has **fewer parameters** than Darknet53.

Seeing a mAP (mean average precision) improvement by swapping this backbone is a huge win for PP YOLO.

EMA OF MODEL PARAMETERS

PP YOLO tracks the **Exponential Moving Average** of network parameters to maintain a shadow of the models weights for prediction time. This has been shown to improve inference accuracy.

Exponential moving average algorithms are quite common in the **financial analytics** domain. Using a similar methodology, the authors improved the performance of the model by **smoothing out the weights** during the training phase.

This process is also very **memory efficient** since at every update step only the value of the smoothing factor needs to be kept in memory. This value was set to **0.9998** in the paper.

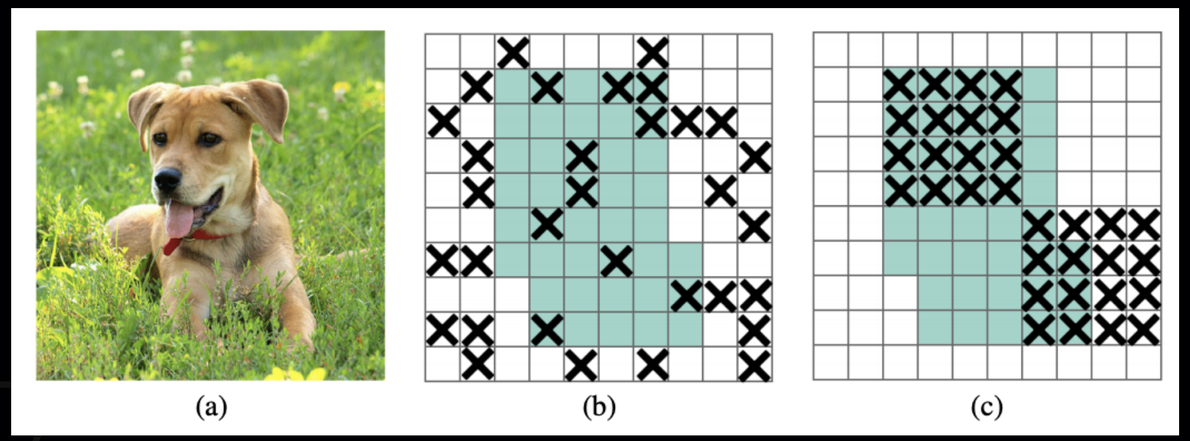
LARGER BATCH SIZE

In order to improve the model performance, the authors of the paper experimented with a batch size of 192.

Training the neural network with a larger batch size leads to a boost in the accuracy of the model by stabilizing the training process.

They bumped up the batch size up from 64 to 192. Of course, this is hard to implement if you have GPU memory constraints.

DROPBLOCK REGULARIZATION

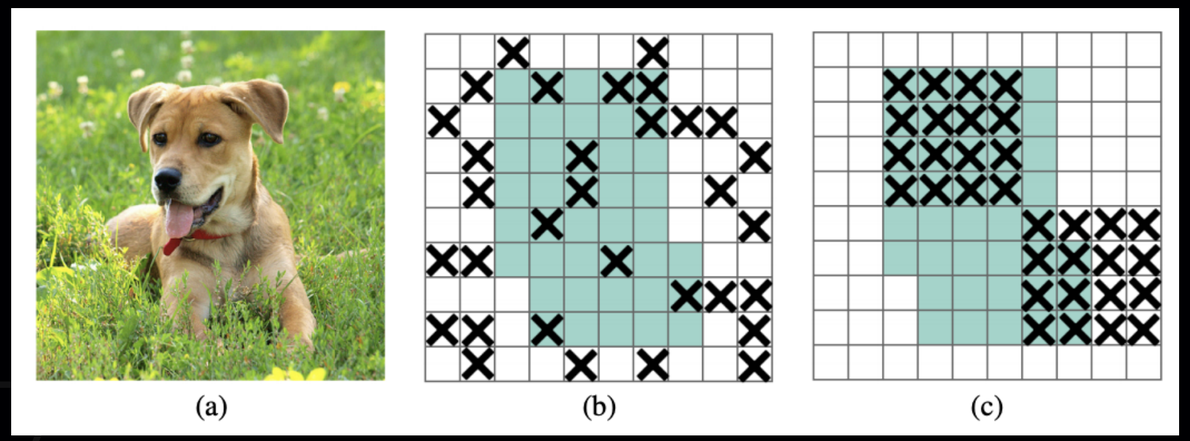


One way to improve the accuracy of the model and ensure that the model does not overfit the training data is to use regularization techniques like adding noise or using a dropout layer.

This process can be made even more robust by dropping neurons in a continuous region as opposed to doing it independently. This ensures that the model can generalize well on unseen data.

DropBlock randomly removes a block of the training features at a given step in the network to teach the model to not rely on key features for detection..

DROPBLOCK REGULARIZATION



PP YOLO implements DropBlock regularization in the **FPN neck**.

In the past, this has usually occurred in the backbone.

IOU LOSS

Neural networks have a loss function to make the network learn and adapt. In the original YOLOv3 paper only the **L1 loss** was used and adapted for estimating the bounding box coordinates.

The **YOLO loss function** does not translate well to the **mAP metric**, which uses the Intersection over Union heavily in its calculation. Therefore, it is useful to edit the training loss function with this end prediction in mind. This edit was also present in YOLOv4.

The authors from the **BAIDU research team** experimented with both types of losses in the training process. The authors of the paper decided to stick with the basic version of the **IoU loss** since it gives good results in estimating the **mAP metric**.

OTHER CONTRIBUTIONS

Grid Sensitivity

The old YOLO models do not do a good job of making predictions right around the boundaries of anchor box regions.

It is useful to define box coordinates slightly differently to avoid this problem. This technique is also present in YOLOv4.

Matrix NMS

Non-Maximum Suppression is a technique to remove over proposals of candidate objects for classification.

Matrix NMS is a technique to sort through these candidate predictions in parallel, speeding up the calculation.

CoordConv

CoordConv was motivated by the problems ConvNets were having with simply mapping (x,y) coordinates to a one-hot pixel space.

The CoordConv solution gives the convolution network access to its own input coordinates.

BETTER PRETRAINED BACKBONE

Using pre-trained models and transferring the knowledge learned from one domain to another domain is a common trick to improve the speed of training the model and the effectiveness of the model.

The PP YOLO authors distilled down a larger ResNet model to serve as the backbone. A better pretrained model shows to improve downstream transfer learning as well.

The authors used a better set of pre-trained weights for the backbone architecture which allowed them to improve the final detection accuracy even more.

	Methods	mAP(%)	Parameters	GFLOPs	infer time	FPS
A	Darknet53 YOLOv3	38.9	59.13 M	65.52	17.2 ms	58.2
B	ResNet50-vd-dcn YOLOv3	39.1	43.89 M	44.71	12.6 ms	79.2
C	B + LB + EMA + DropBlock	41.4	43.89 M	44.71	12.6 ms	79.2
D	C + IoU Loss	41.9	43.89 M	44.71	12.6 ms	79.2
E	D + Iou Aware	42.5	43.90 M	44.71	13.3 ms	74.9
F	E + Grid Sensitive	42.8	43.90 M	44.71	13.4 ms	74.8
G	F + Matrix NMS	43.5	43.90 M	44.71	13.4 ms	74.8
H	G + CoordConv	44.0	43.93 M	44.76	13.5ms	74.1
I	H + SPP	44.3	44.93 M	45.12	13.7 ms	72.9
J	I + Better ImageNet Pretrain	44.6	44.93 M	45.12	13.7 ms	72.9

Table 1. The ablation study of tricks on the MS-COCO minival split.

PP YOLO

RESULTS

Method	Backbone	Size	FPS (V100)		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
			w/o TRT	with TRT						
RetinaNet [22]	ResNet-50	640	37	-	37.0%	-	-	-	-	-
RetinaNet [22]	ResNet-101	640	29.4	-	37.9%	-	-	-	-	-
RetinaNet [22]	ResNet-50	1024	19.6	-	40.1%	-	-	-	-	-
RetinaNet [22]	ResNet-101	1024	15.4	-	41.1%	-	-	-	-	-
EfficientDet-D0 [35]	Efficient-B0	512	98.0^{+}	-	33.8%	52.2%	35.8%	12.0%	38.3%	51.2%
EfficientDet-D1 [35]	Efficient-B1	640	74.1^{+}	-	39.6%	58.6%	42.3%	17.9%	44.3%	56.0%
EfficientDet-D2 [35]	Efficient-B2	768	56.5^{+}	-	43.0%	62.3%	46.2%	22.5%	47.0%	58.4%
EfficientDet-D2 [35]	Efficient-B3	896	34.5^{+}	-	45.8%	65.0%	49.3%	26.6%	49.4%	59.8%
RFBNet[3]	HarDNet68	512	41.5	-	33.9%	54.3%	36.2%	14.7%	36.6%	50.5%
RFBNet[3]	HarDNet85	512	37.1	-	36.8%	57.1%	39.5%	16.9%	40.5%	52.9%
YOLOv3 + ASFF* [26]	Darknet-53	320	60	-	38.1%	57.4%	42.1%	16.1%	41.6%	53.6%
YOLOv3 + ASFF* [26]	Darknet-53	416	54	-	40.6%	60.6%	45.1%	20.3%	44.2%	54.1%
YOLOv3 + ASFF* [26]	Darknet-53	608	45.5	-	42.4%	63.0%	47.4%	25.5%	45.7%	52.3%
YOLOv3 + ASFF* [26]	Darknet-53	800	29.4	-	43.9%	64.1%	49.2%	27.0%	46.6%	53.4%
YOLOv4 [1]	CSPDarknet-53	416	96	164.0^{*}	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4 [1]	CSPDarknet-53	512	83	138.4^{*}	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4 [1]	CSPDarknet-53	608	62	105.5^{*}	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
PP-YOLO	ResNet50-vd-dcn	320	132.2	242.2	39.3%	59.3%	42.7%	16.7%	41.4%	57.8%
PP-YOLO	ResNet50-vd-dcn	416	109.6	215.4	42.5%	62.8%	46.5%	21.2%	45.2%	58.2%
PP-YOLO	ResNet50-vd-dcn	512	89.9	188.4	44.4%	64.6%	48.8%	24.4%	47.1%	58.2%
PP-YOLO	ResNet50-vd-dcn	608	72.9	155.6	45.2%	65.2%	49.9%	26.3%	47.8%	57.2%

The PP-YOLO took the YOLOv3 model from 38.9 to 44.6 mAP on the COCO object detection task and increased inference FPS from 58 to 73.

The model also improves the mAP on COCO from 43.5% to 45.2% at a speed faster than YOLOv4.

These metrics are shown in the paper to beat the currently published results for YOLOv4 and EfficientDet.

IS PP-YOLO STATE OF THE ART?

In fairness, the authors note this may be the wrong question to be asking. The authors' intent appears to not simply "introduce a new novel detector," rather to show the process of carefully tuning an object detector to maximize performance.

PP-YOLO does not introduce a new way of designing object detection models, however, it provides a valuable case study of which tricks known from a wider field of deep learning work well in the context of one-stage object detectors.

THANK YOU

PLEASE TUNE IN FOR THE SECOND PART