YOLO FAMILY

01

SCALED YOLOV4

VISUAL LEARNERS

02

Table of Contents

Introduction

Principles of Model Scaling

Designing Scaled YOLOv4

Ablation Study & Results

Introduction

- In the previous video, we have discussed about YOLOv4's architecture, and how it became a SOTA (state-of-the-art) model for the object detection task offering the best performance in terms of both speed and accuracy.

- But it could not beat Google's EfficientDet in terms of overall accuracy on the COCO dataset. So, the authors of YOLOv4 came back and pushed the YOLOv4 model forward by scaling it's design and thus outperforming the benchmarks of EfficientDet. This resulted in Scaled-YOLOv4 model.

EfficientDet Models - Model Scaling

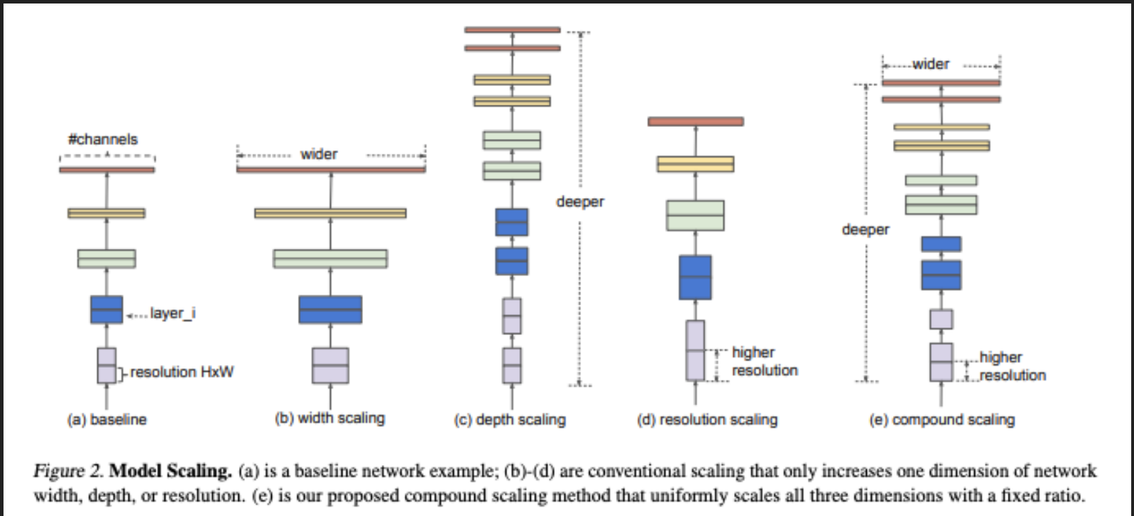


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

- The **EfficientDet family** of models has been the preferred object detection models since it was published by the **Google Research/Brain team.**
- The most novel contribution in the EfficientDet paper was to think strategically about **how to scale object detection models up and down**.
- Object detection models can be scaled by using a **large image input resolution**, **scaling the width of the convolutional network layer**, **scaling the depth of convolutional layers**, and **scaling all of these things together.**

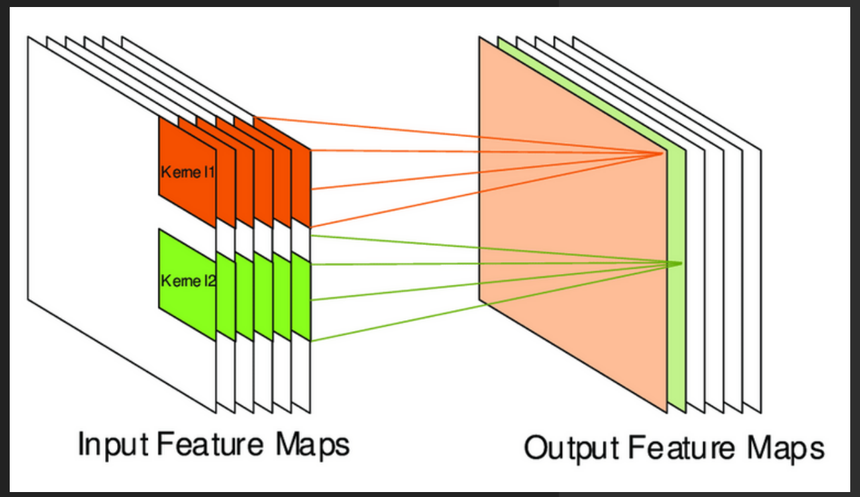
General Principle of Model Scaling

Model	original	size α	depth β	width γ
Res layer	$r = 17whkb^2/16$	$\alpha^2 r$	βr	$\gamma^2 r$
ResX layer	$x = 137whkb^2/128$	$\alpha^2 x$	βx	$\gamma^2 x$
Dark layer	$d = 5whkb^2$	$\alpha^2 d$	βd	$\gamma^2 d$

- Some factors are considered when working on model scaling for object detection task.
- Let the **scaling factors** that can be used to adjust the image size, the number of layers, and the number of channels be ***α***, ***β***, and ***γ***, respectively.
- ResNet, ResNeXt and DarkNet layers are investigated.
- For the k-layer CNNs with b base layer channels, when these scaling factors vary, the corresponding changes on FLOPs are shown as above table.

Model	original	size α	depth β	width γ
Res layer	$r = 17whkb^2/16$	$\alpha^2 r$	βr	$\gamma^2 r$
ResX layer	$x = 137whkb^2/128$	$\alpha^2 x$	βx	$\gamma^2 x$
Dark layer	$d = 5whkb^2$	$\alpha^2 d$	βd	$\gamma^2 d$

06



The scaling size, depth and width cause increase in the computation cost.

They respectively show square, linear, and square increase.

CSP-ized YOLOv4

Before we talk about the architecture design, let us look at some of the advantages of using CSP connections:

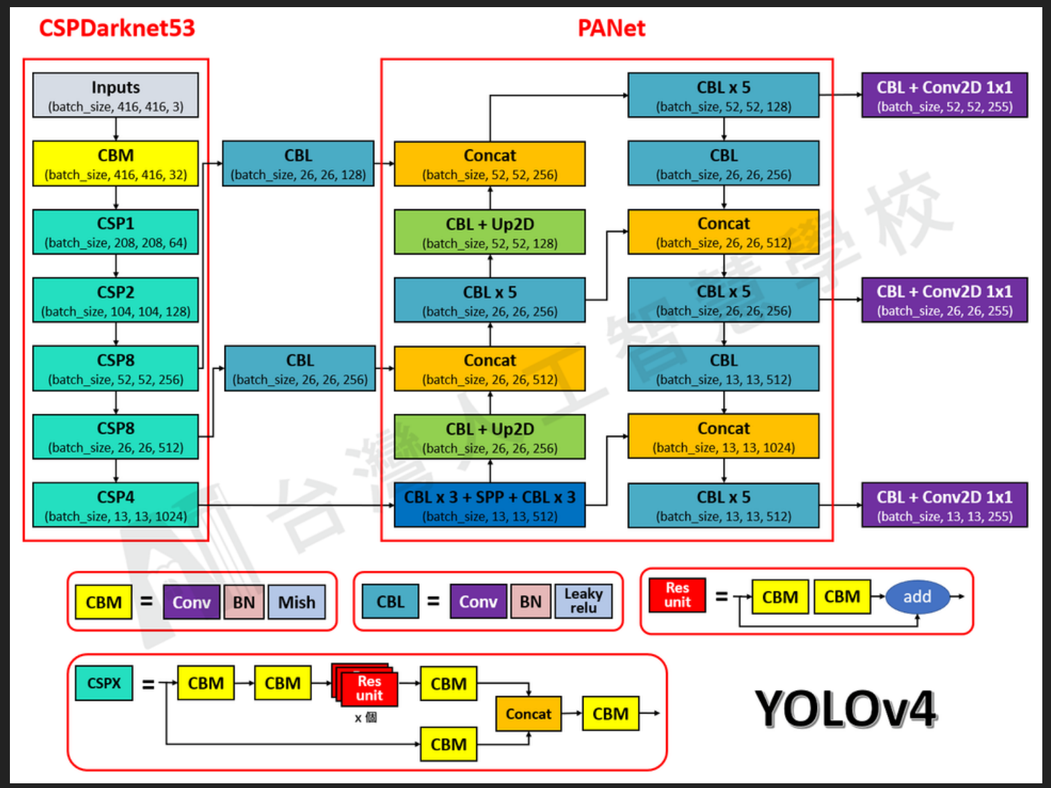
- Half of the signal going through the main path will generate more semantic information with a large receiving field
- The other half that is being bypassed would preserve more spatial information with a small perceiving field.

CSP-ized YOLOv4

08

The original YOLOv4 model only used CSP in the back bone of the network and not in the PANet neck and SPP.

The authors of Scaled YOLOv4 wanted to investigate the effects of introducing the CSP method in other layers of the network to reduce computational cost and improve performance.



"CSP-izing"

09

- We've discussed in the YOLOv4 Architecture Model, about how using Cross Stage Partial strategy reduces the computation costs.
- In the Scaled-YOLOv4 paper the authors often write that they "CSP-ized" a given portion of the network.
- To CSP-ize means to apply the concepts laid out in the Cross-Stage Partial Networks paper, written by WongKinYiu.
- In brief, CSPNet splits the input into two paths. One performs convolutions. One performs no convolution. They are fused at the output.

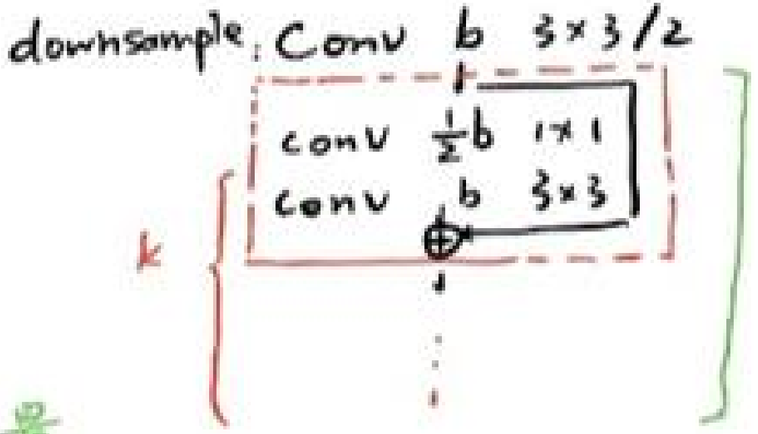
"CSP-izing"

Table 2: FLOPs of different computational layers with/without CSP-ization.

Model	original	to CSP
Res layer	$17whkb^2/16$	$whb^2(3/4+13k/16)$
ResX layer	$137whkb^2/128$	$whb^2(3/4+73k/128)$
Dark layer	$5whkb^2$	$whb^2(3/4+5k/2)$

- When CSPNet is applied to ResNet, ResNeXt, and Darknet and observe the changes in the amount of computations, in the above table.
- CSPNet can effectively reduce the amount of computations (FLOPs) on ResNet, ResNeXt, and Darknet by 23.5%, 46.7%, and 50.0%, respectively.
- *Therefore, CSP-ized models are used as the best model for performing model scaling.*

Darknet 残差层.

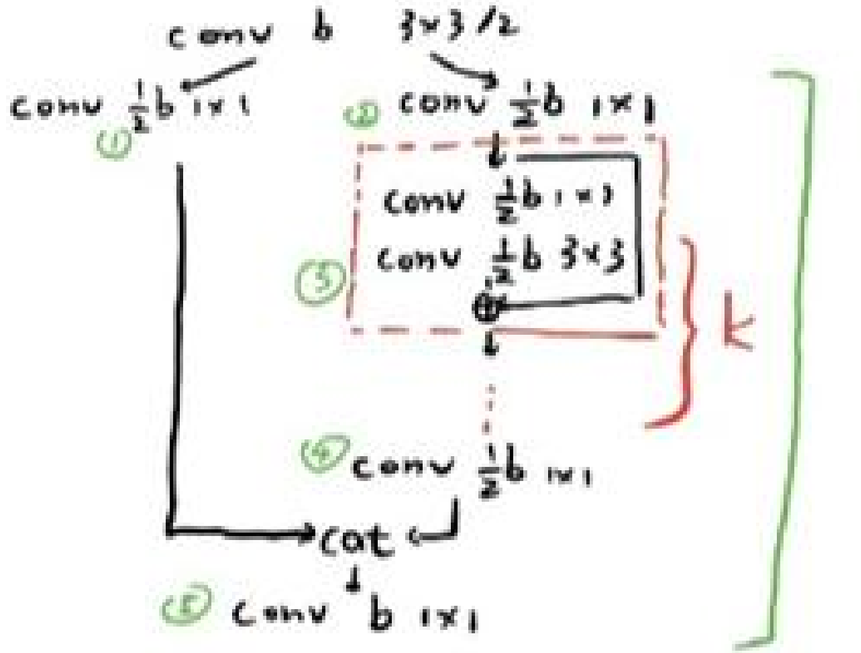


计算量

$k[wh \cdot b \cdot (1\times1\times\frac{1}{2}b) + wh \cdot \frac{1}{2}b(3\times3\times b)]$

$= kwhb^2(\frac{1}{2} + \frac{9}{2}) = 5whkb^2$

Csp-Darknet.



$wh\frac{b}{2}$ ① $(1\times1\times\frac{1}{2}b) + wh\frac{1}{2}b$ ③ $(1\times1\times\frac{1}{2}b)$

$+ [wh\frac{1}{2}b(1\times1\times\frac{1}{2}b) + wh\frac{1}{2}b(3\times3\times\frac{1}{2}b)]k$ ⑤

$+ whb$ ⑥ $(1\times1\times\frac{1}{2}b) + wh2b$ ⑦ $(1\times1\times b)$

$= whb^2[\frac{1}{4} + (\frac{1}{4} + \frac{5}{2}k + \frac{1}{2}) + 2]$

$= whb^2[\frac{9}{4} + (\frac{3}{4} + \frac{5}{2}k)]$

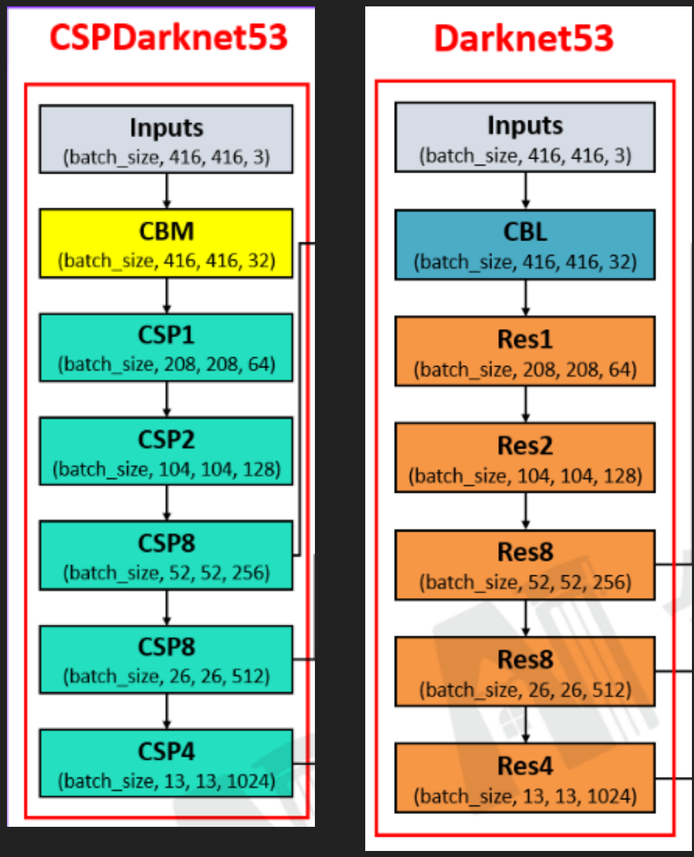
CSP-ized YOLOv4

Now let's talk about the architecture modifications

- For General GPUs, YOLOv4 is redesigned to YOLOv4-CSP
- Backbone: The first CSP stage is converted to original Darknet Residual Layer.
- Neck: The PAN architecture is CSP-ized and Mish activation is used to reduce computation by 40%
- SPP: It was originally inserted in the middle position of the neck, the same idea is borrowed and implemented in CSPPAN too.

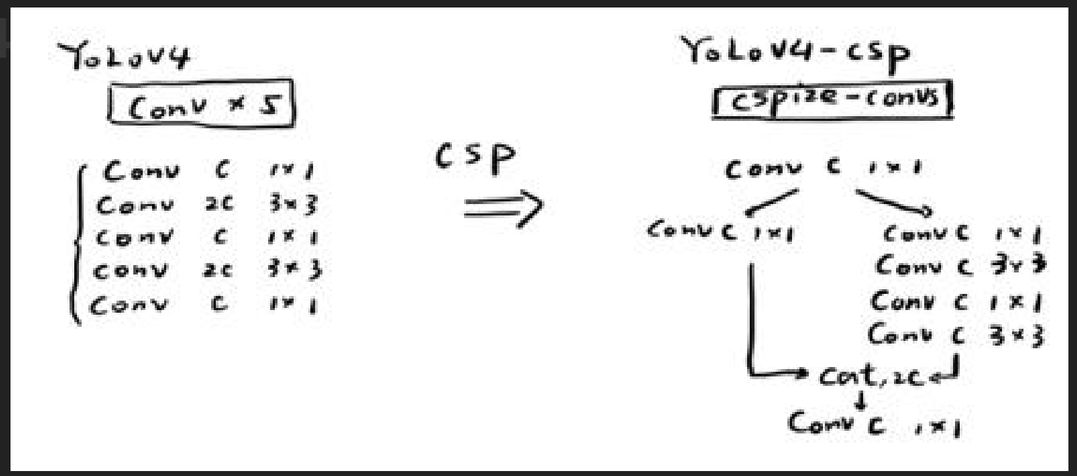
1. CSPizing Backbone

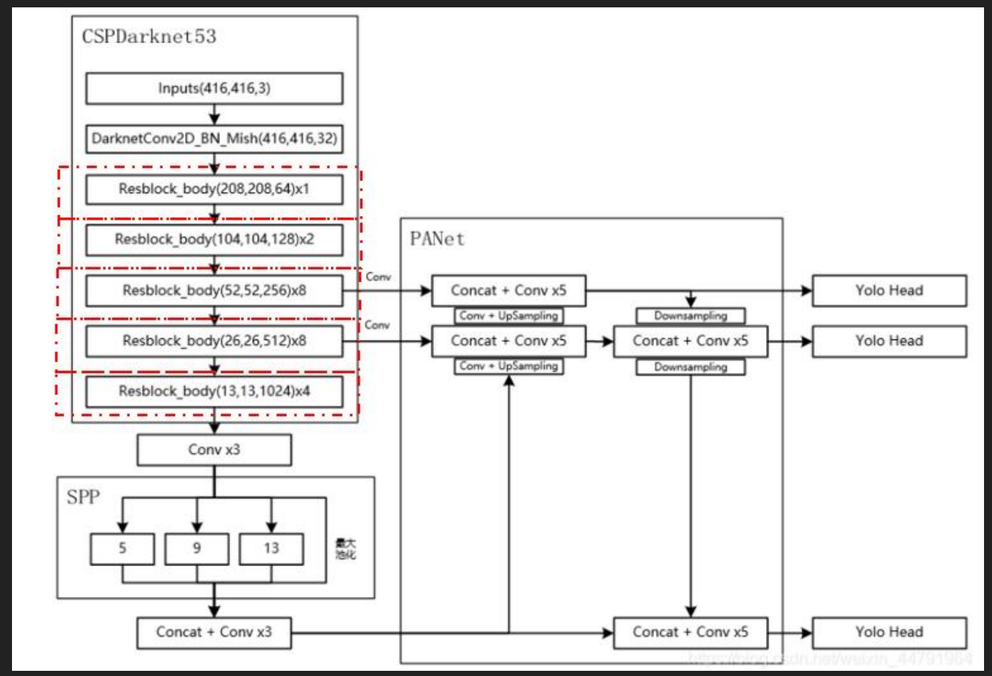
- The amount of computation of each CSPDarknet stage is $whb^2(9/4+3/4+5k/2)$.
- According to the previous section, CSPDarknet stage will have a better computational advantage over Darknet stage only when k>1 is satisfied.
- The number of residual layer owned by each stage in CSPDarknet53 is 1-2-8-8-4 respectively.
- In order to get a better speed/accuracy trade-off, the first CSP stage is converted into original Darknet residual layer.

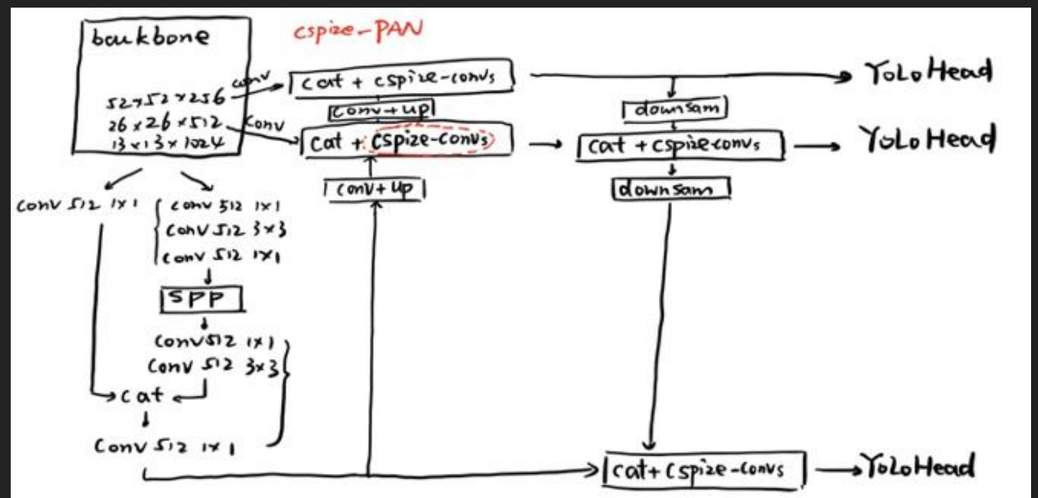


2. CSPizing Neck

- The PAN architecture in YOLOv4 is also CSP-ized

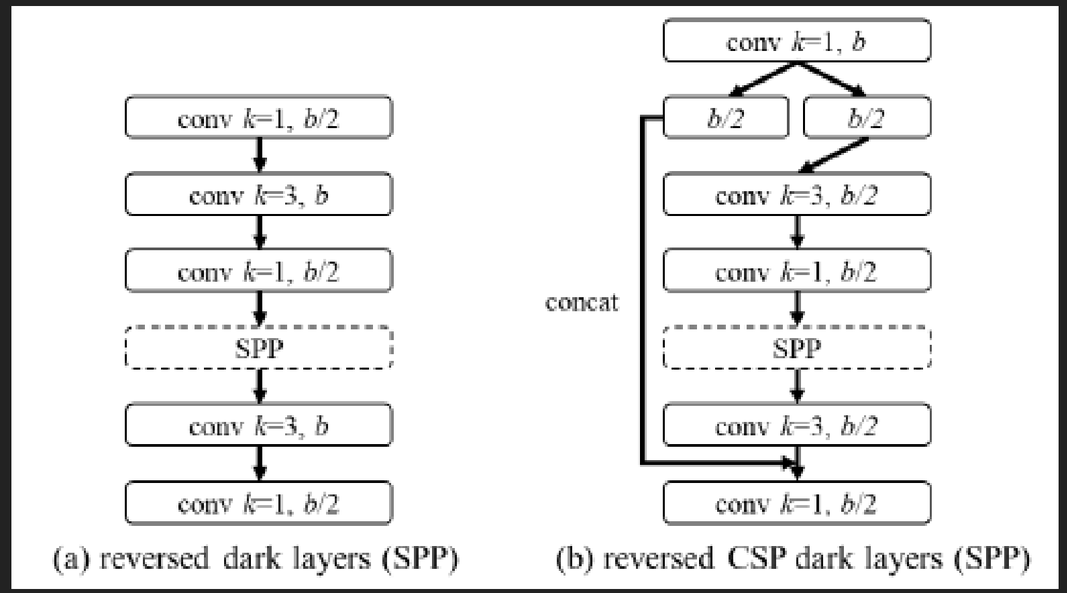


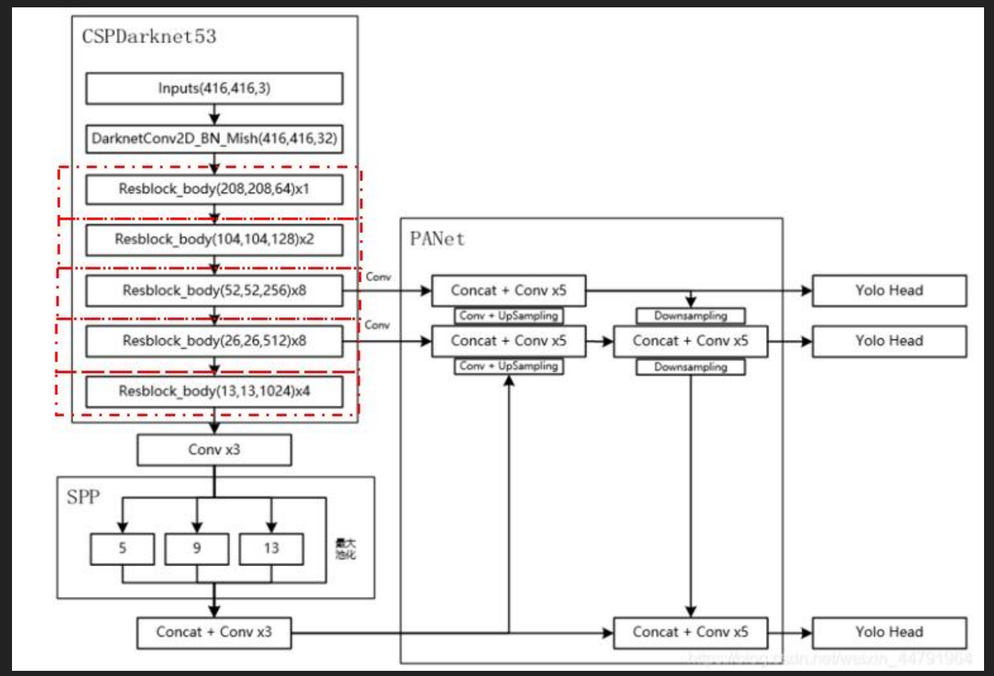


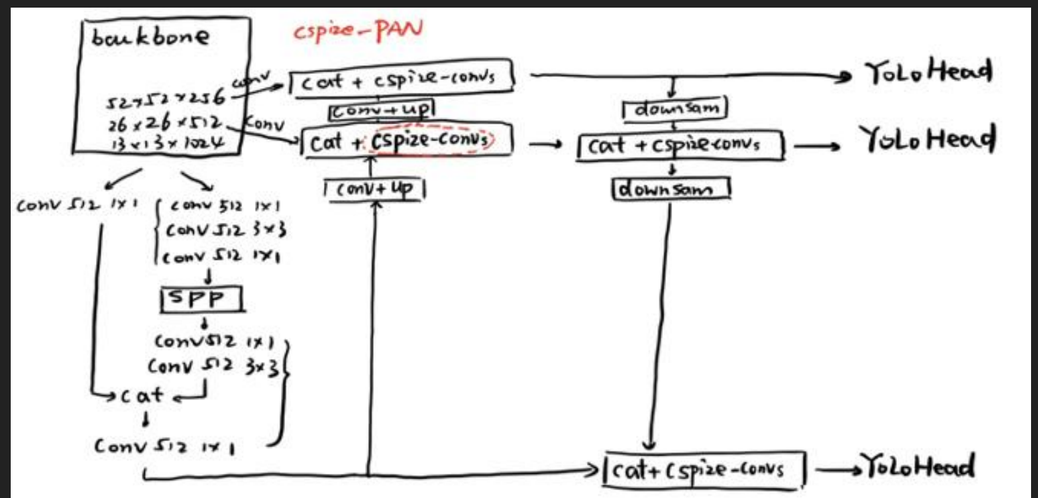


3. CSPizing SPP

- Also, SPP module (SPPNet) is now inserted in the middle position of the first computation list group of the CSPPAN.



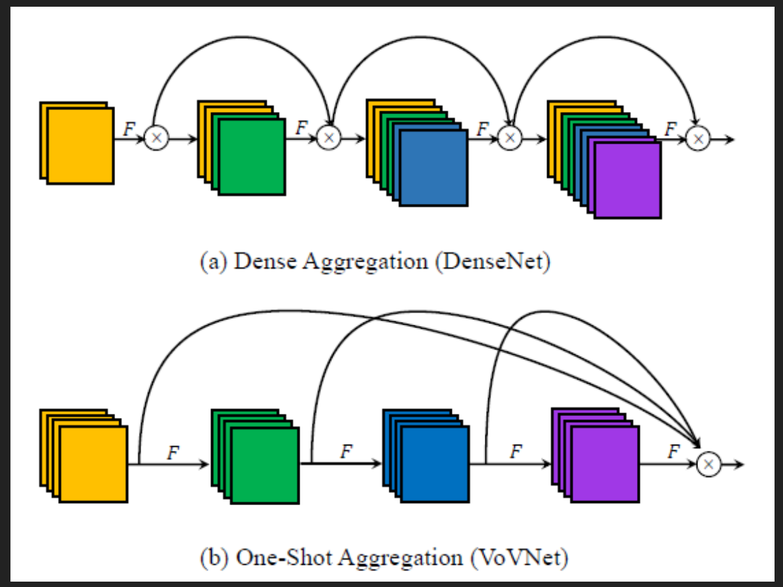




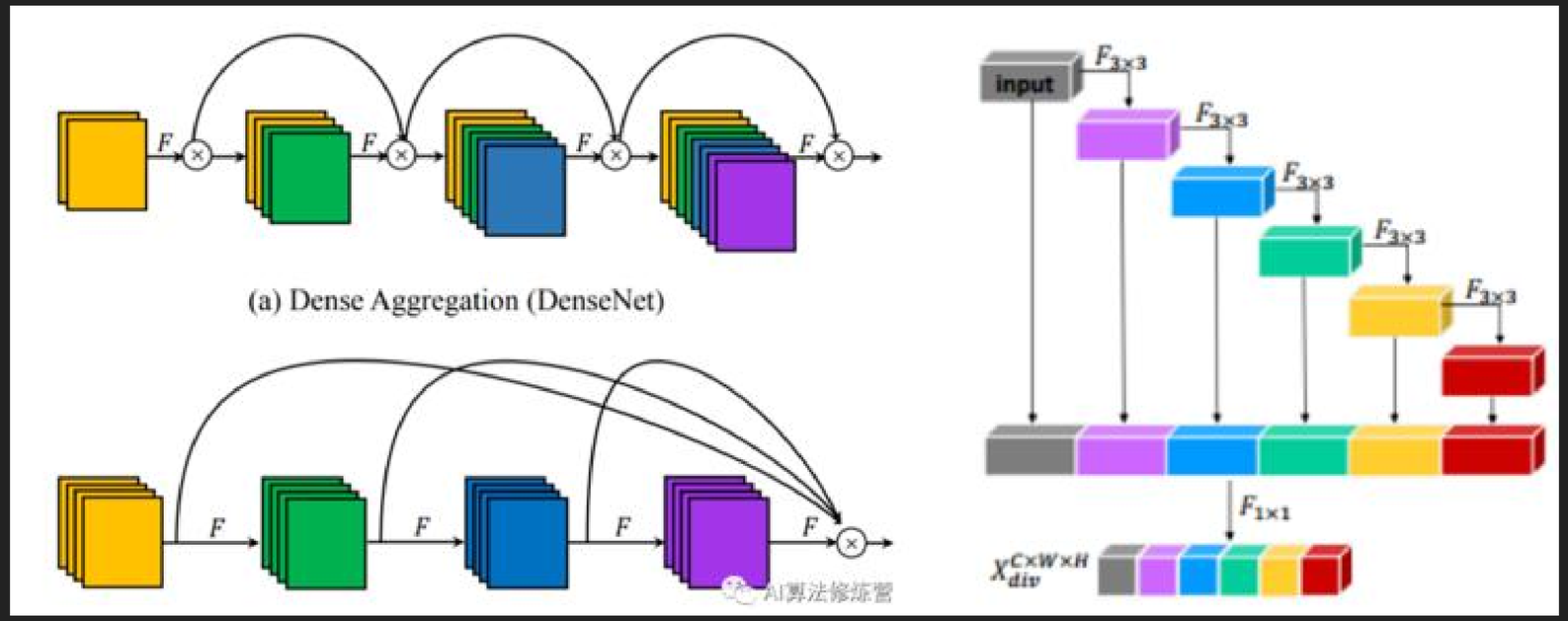
Scaling on different devices - Scaling Tiny Models for Low-End Devices (YOLOv4 tiny)

- Lightweight models are different from large models in that their parameter utilization efficiency must be higher and are used in low end devices.
- Memory bandwidth, memory access cost (MACs), and DRAM traffic should also be considered.
- The network with efficient parameter utilization is analyzed, such as the computation load of DenseNet and OSANet, where *g* means growth rate.

DenseNet Vs OSANet



- In **DenseNet**, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. **Concatenation** is used.
- ***Dense connection makes later intermediate layer produce the features that are better but also similar to the features from former layers.*** *In this case, the final layer is not required to learn to aggregate both features because they are representing* ***redundant information****.*
- ***One-shot aggregation (OSA)*** *module is designed to* ***aggregate its feature in the last layer at once***



- The speed of Densenet is slower, mainly because of the high memory access cost and energy consumption caused by dense connections in DenseNet.

- VoVNet is intended to solve the problem of DenseNet. The performance of the target detection model based on VoVNet exceeds the model based on DenseNet.

- OSA only aggregates all the previous layers at the end. This change will solve the problem of DenseNet, because the number of input channels for each layer is fixed, and the number of output channels can be made consistent with the input to obtain the smallest MAC.

CSP-izing OSANet - Order of computation should be less

Model	FLOPs
Dense layer	$whgbk + whg^2k(k-1)/2$
OSA layer	$whbg + whg^2(k-1)$

OSA improves GPU computation efficiency and has much less MAC (Memory Access Cost) than that with dense block.

The order of computation complexity of DenseNet is O(whgbk), and that of OSANet is O(max(whbg, whkg2)).

The tiny model is designed with the help of OSANet, which has a smaller computation complexity.

CSP-izing OSANet - Minimize/balance size of feature map

layer ID	original	CSP	partial in CB
1	$b \rightarrow g$	$g \rightarrow g$	$g \rightarrow g$
2	$g \rightarrow g$	$g \rightarrow g$	$g \rightarrow g$
...	$g \rightarrow g$	$g \rightarrow g$	$g \rightarrow g$
k	$g \rightarrow g$	$g \rightarrow g$	$g \rightarrow g$
T	$(b + kg)$ $\rightarrow (b + kg)/2$	$kg \rightarrow kg$	$(b + kg)/2$ $\rightarrow (b + kg)/2$

Gradient Truncation between computational block of the **CSPOSANet** is performed to get the best calculation speed..

b channels of the base layer and the **kg channels** generated by computational block, and split them into two paths with equal channel numbers i.e **(b + kg)/2**.

The CSPOSANet is designed to dynamically adjust the channel allocation.

$$MAC = hw(C_{in} + C_{out}) + KC_{in}C_{out}$$

When evaluating the computational cost of low-end devices, power consumption must also be considered.

The biggest factor affecting power consumption is the memory access cost (MAC).

By calculating the geometric inequality, the minimum MAC when Cin = Cout can be derived.

CSP-izing OSANet - Minimize (CIO)

original	CSP	partial in CB
$bg + (k-1)g^2 + (b+kg)^2/2$	$kg^2 + (kg)^2$	$kg^2 + (b+kg)^2/4$

Convolutional Input/Output (CIO) is an indicator that can measure the status of **DRAM IO.**

The CIO of OSA, CSP, and the designed CSPOSANet.

When kg>b/2, the proposed CSPOSANet can obtain the best CIO.

Scaling YOLOv4 Tiny

- YOLOv4-tiny is designed for low-end GPU device.

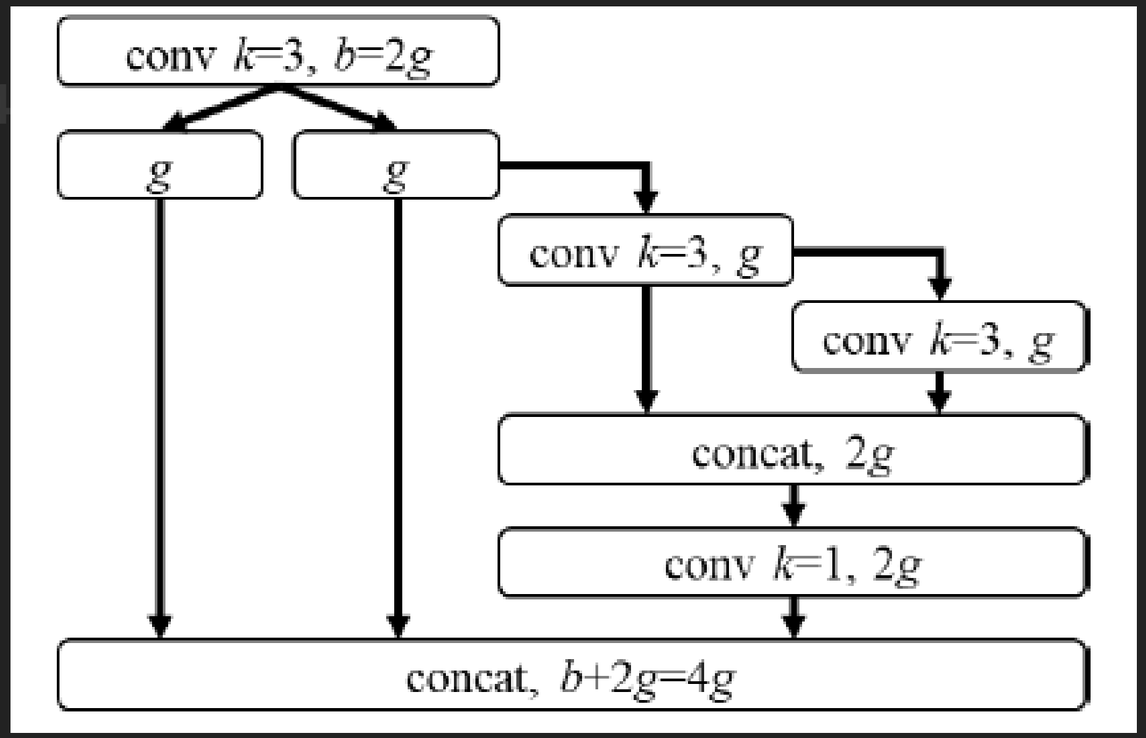
- The CSPOSANet with partial in computational block (PCB) architecture is used to form the backbone of YOLOv4.

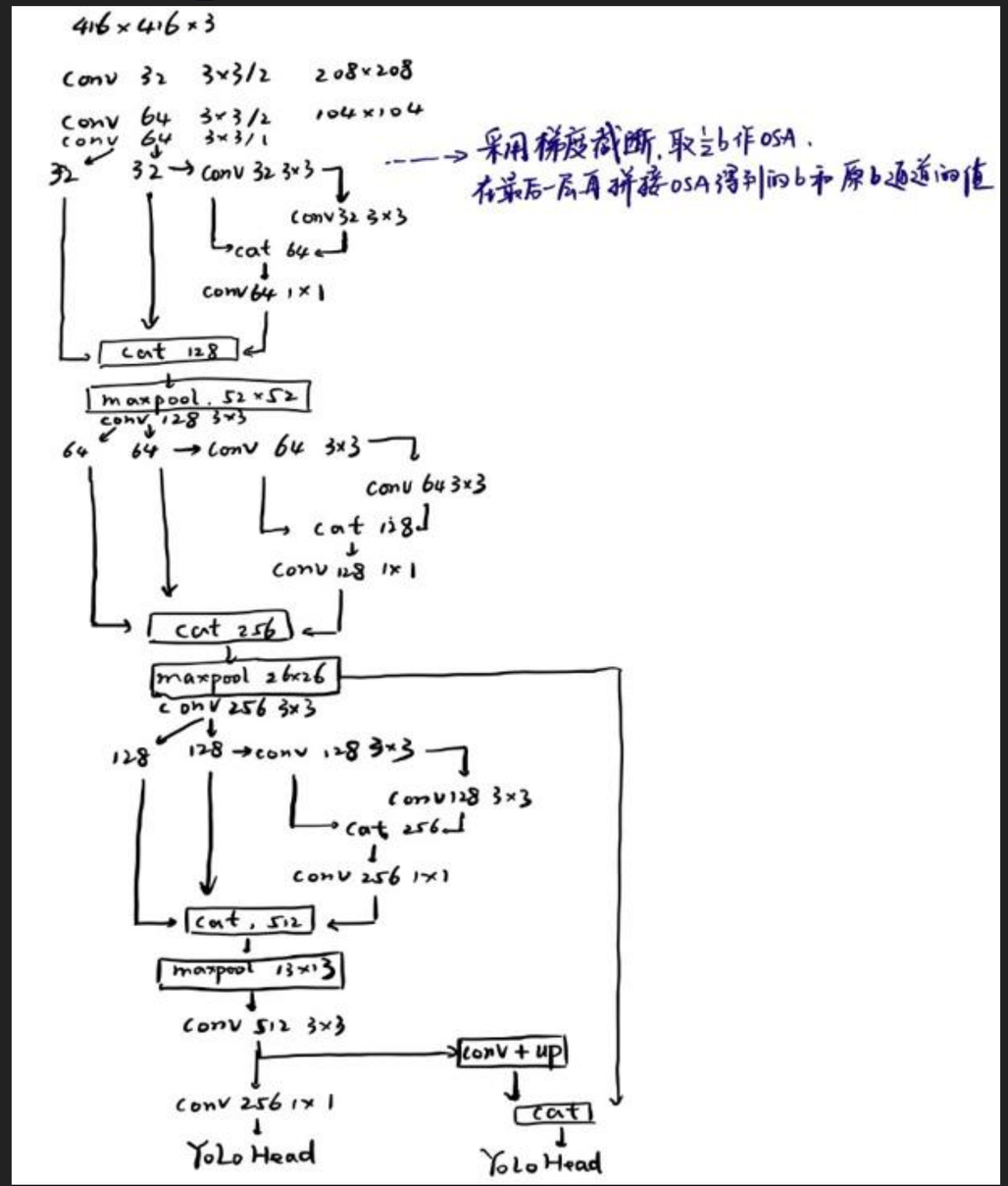
- g=b=2 is set as the growth rate and make it grow to b/2+kg=2b at the end

.

- Through calculation, k=3 is deduced, and its architecture is shown on next slide.

Scaling YOLOv4 Tiny



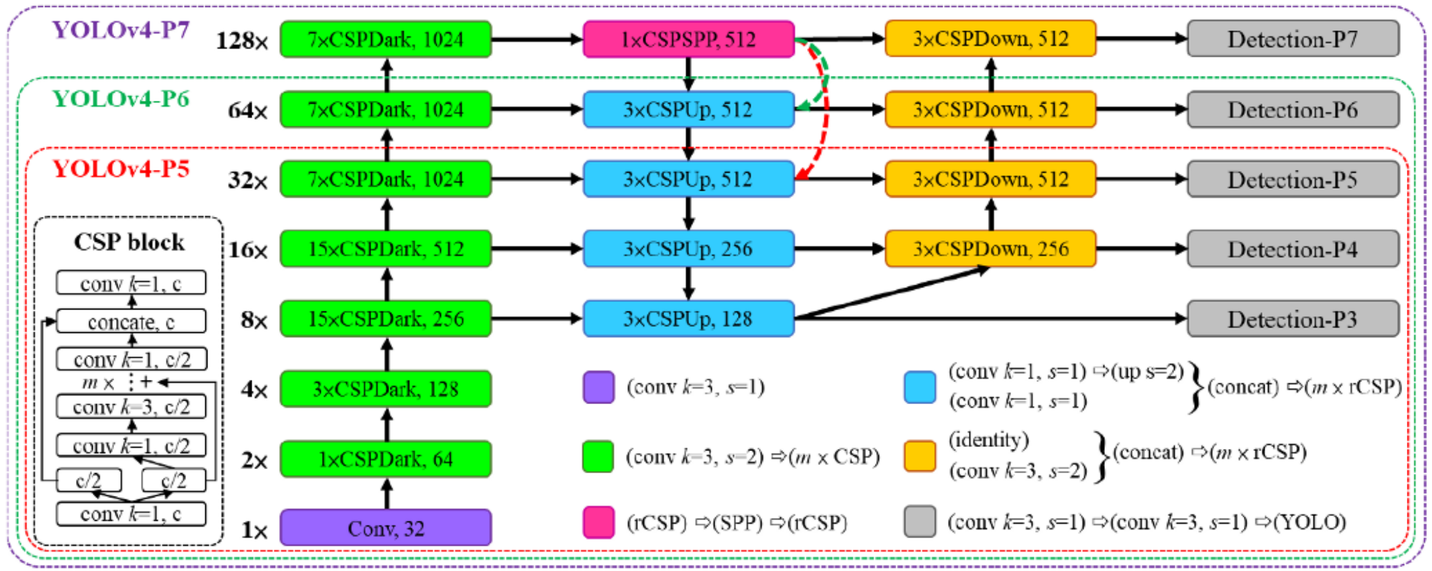


Scaling on different devices - Scaling YOLOv4 large

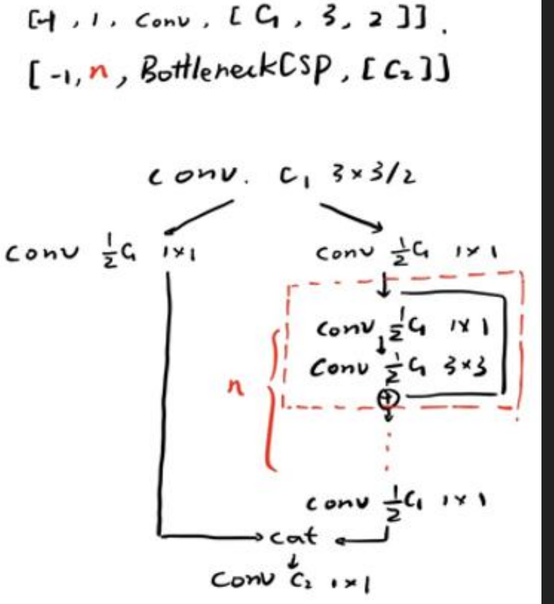
- YOLOv4-large is designed for cloud GPU, the main purpose is to achieve high accuracy for object detection.
- We designed a fully CSP-ized model YOLOv4-P5 and scaling it up to YOLOv4-P6 and YOLOv4-P7.
- YOLOv4-P6 can reach real-time performance at 30 FPS video when the width scaling factor is equal to 1.
- **YOLOv4-P7** can reach real-time performance at **16 FPS** video when the **width scaling factor is equal to 1.25**.

Scaling YOLOv4 Large

In the backbone section, change the original size of [1,2,8,8,4]CSPDarkNet53 to [1,3,15,15,7].




```yaml
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [32, 3, 1]],  # 0
   [-1, 1, Conv, [64, 3, 2]],  # 1-P1/2
   [-1, 1, BottleneckCSP, [64]],
   [-1, 1, Conv, [128, 3, 2]],  # 3-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 5-P3/8
   [-1, 15, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 7-P4/16
   [-1, 15, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]], # 9-P5/32
   [-1, 7, BottleneckCSP, [1024]],  # 10
  ]
```



Ablation Study on CSP-ized Model

- MSCOCO 2017 object detection dataset is used.
- All models are trained from scratched
- Darknet53 (D53) is used as backbone and FPN with SPP (FPNSPP) and PAN with SPP (PANSPP) are chosen as necks to design ablation studies.
- LeakyReLU (Leaky) and Mish activation function are tried.
- *CSP-ized models have greatly reduced the amount of parameters and computations by 32%, and brought improvements in both Batch 8 throughput and AP.*

Ablation Study on CSP-ized Model

- Both CD53s-CFPNSPP-Mish, and CD53s-CPANSPP-Leaky have the same batch 8 throughput with D53-FPNSPP-Leaky, but they respectively have 1% and 1.6% AP improvement with lower computing resources.
- Therefore, CD53s-CPANSPP-Mish is decided to used, as it results in the highest AP in the above table as the backbone of YOLOv4-CSP.

Backbone	Neck	Act.	#Param.	FLOPs	Batch 8 FPS	AP^{val}
D53	FPNSPP	Leaky	63M	142B	208	43.5%
D53	FPNSPP	Mish	63M	142B	196	45.3%
CD53s	**CFPNSPP**	Leaky	43M	97B	222	45.7%
CD53s	**CFPNSPP**	Mish	43M	97B	208	46.3%
D53	PANSPP	Leaky	78M	160B	196	46.5%
D53	PANSPP	Mish	78M	160B	185	46.9%
CD53s	**CPANSPP**	Leaky	53M	109B	208	46.9%
CD53s	**CPANSPP**	Mish	53M	109B	200	47.5%

Ablation Study on YOLOv4-tiny

- The designed PCB technique can make the model more flexible, because such a design can be adjusted according to actual needs.

- The proposed COSA can get a higher AP.

- *Therefore, COSA-2x2x is finally chosen, which received the best speed/accuracy trade-off in the experiment as the YOLOv4-tiny architecture.*

Backbone	Neck	FLOPs	FPS_{TX2}	AP^{val}
tinyCD53s	tinyFPN	7.0B	30	22.2%
COSA-1x3x	tinyFPN	7.6B	38	**22.5%**
COSA-2x2x	tinyFPN	6.9B	42	22.0%
COSA-3x1x	tinyFPN	**6.3B**	**46**	21.2%

Ablation Study on YOLOv4-tiny

- For YOLOv4-large, 300 epochs are firstly executed and then followed by 150 epochs for fine-tuning using stronger data augmentation method.

- *AP is improved with further fine-tuning using stronger data augmentation method.*

Model	scratch	finetune	AP^{val}	AP^{val}_{50}	AP^{val}_{75}
YOLOv4-P5	300	-	50.5%	68.9%	55.2%
YOLOv4-P5	300	150	51.7%	70.3%	56.7%
YOLOv4-P6	300	-	53.4%	71.5%	58.5%
YOLOv4-P6	300	150	54.4%	72.7%	59.5%
YOLOv4-P7	300	-	54.6%	72.4%	59.7%
YOLOv4-P7	300	150	55.3%	73.3%	60.4%

Input Resolution

- The blue curve represents the removal of the P7 stage from the trained P7 model, and the yellow represents the removal of the P7 and P6 stages from the trained P7 model.

- *It can be seen that YOLOv4-P7 has the best AP at high resolution, while YOLOv4-P7\P7 and YOLOv4-P7\P7\P6 have the best AP at medium and low resolutions.*