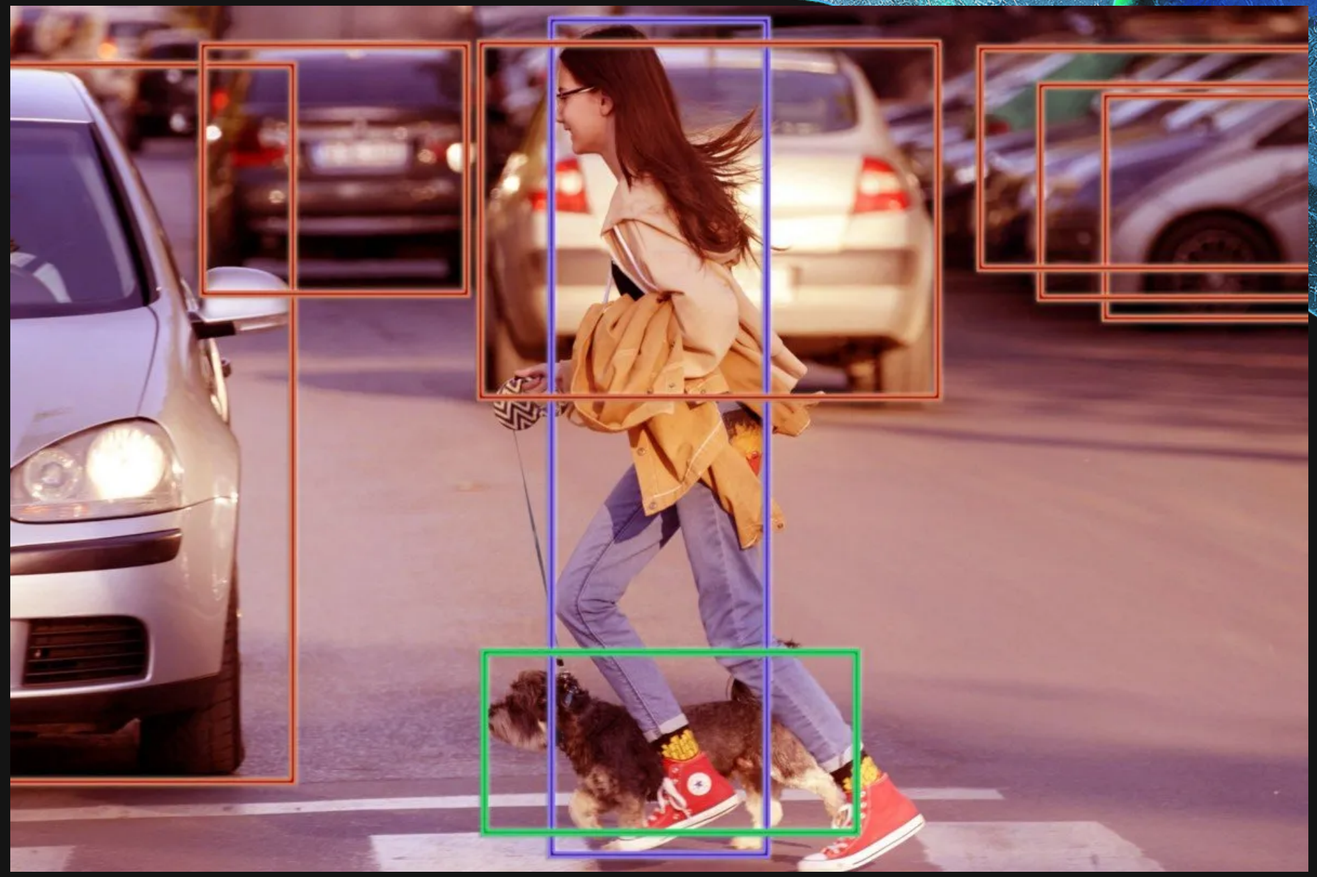
YOLO FAMILY

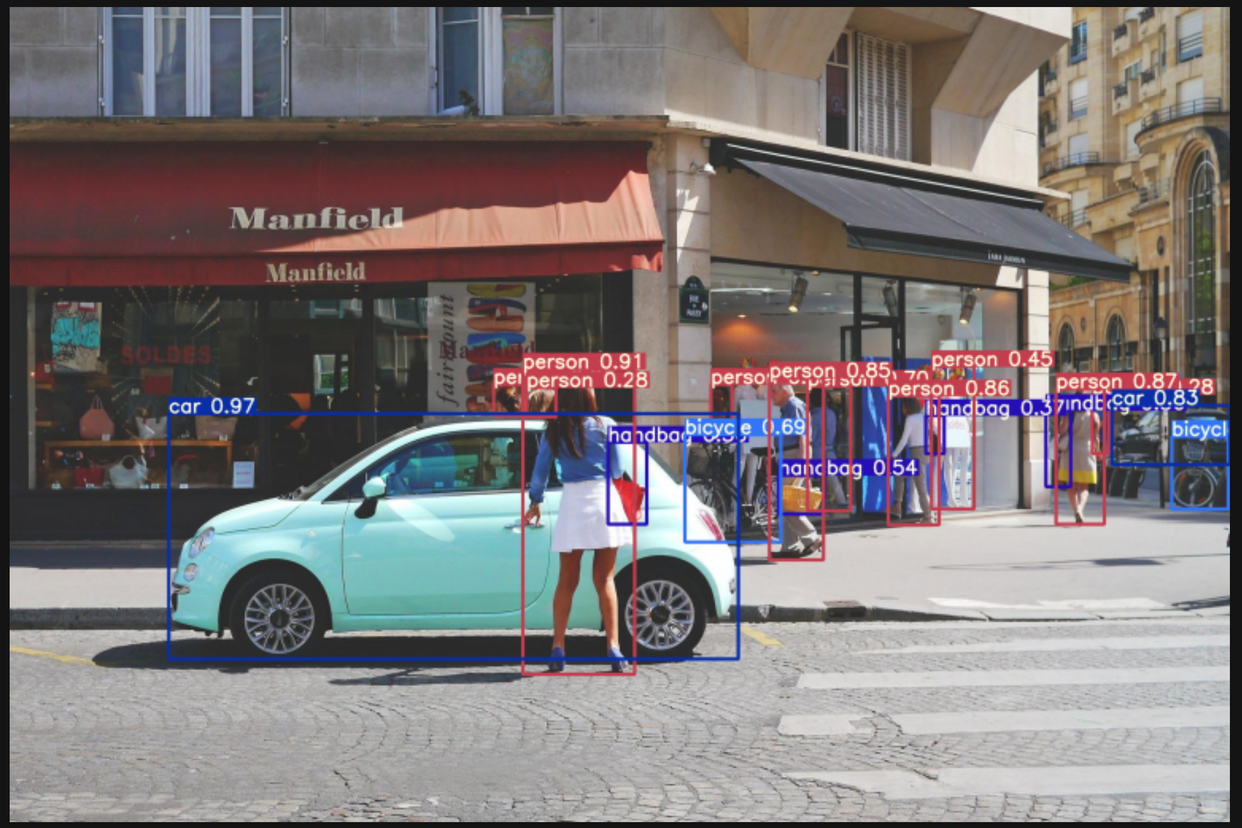
YOLOv7

What it is and why it's important

VISUAL LEARNERS




VISUAL LEARNERS



What is YOLOv7?

YOLOv7 is the fastest and most accurate real-time object detection model for computer vision tasks.

The official YOLOv7 paper named “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors” was released in July 2022

Authors of YOLOv7

- The YOLO v7 model was authored by WongKinYiu and Alexey Bochkovskiy (AlexeyAB).
- AlexeyAB took up the YOLO torch from the original author, Joseph Redmon when Redmon quit the Computer Vision (CV) industry due to ethical concerns.
- WongKinYiu entered the CV research stage with Cross Stage Partial networks, which allowed YOLOv4 and YOLOv5 to build more efficient backbones.
- From there, WongKinYiu steamrolled ahead making a large contribution to the YOLO family of research with Scaled-YOLOv4.
- After that, WongKinYiu released YOLOR, which introduced new methods of tracking implicit knowledge in tandem with explicit knowledge in neural networks.
- Finally the two authors came together and collab to release YOLOv7,

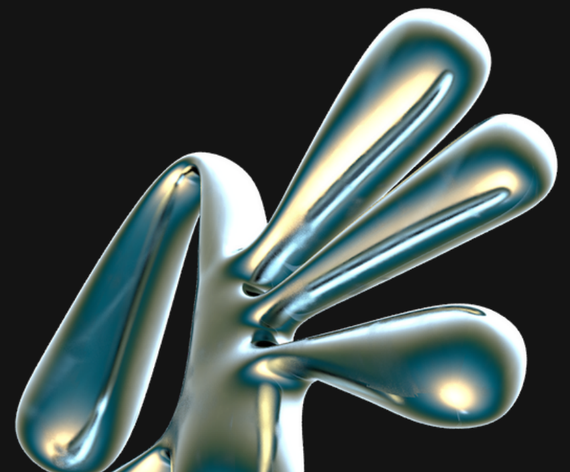
What makes YOLOv7 different?

The YOLOv7 authors sought to set the state of the art in object detection by creating a network architecture that would predict bounding boxes more accurately than its peers at similar inference speeds..

What’s New in YOLOv7?

Architectural Reforms

- E-ELAN (Extended Efficient Layer Aggregation Network)
- Model Scaling for Concatenation-based Models



Trainable BoF (Bag of Freebies)

- Planned re-parameterized convolution
- Coarse for auxiliary and Fine for lead loss

Two Important Concepts

There are many complex things presented in the YOLO v7 paper, so before we delve into it let's first clarify two important concepts.

1. MODEL RE-PARAMETERIZATION

- The idea behind model re-parameterization is that it merges multiple computational modules into one at the inference stage, thus giving us better inference time.
- The model re-parameterization techniques are divided into two main categories: model-level ensemble and module-level ensemble.
- The former is to train multiple identical models with different training data and then average the weights of multiple trained models.
- The latter is to perform a weighted average of the weights of models at different iteration numbers.
- In recent years module-level re-parameterization has gained a lot of traction. Also, some of the re-parameterization techniques are not architecture agnostic; meaning they can be used only with some architectures.
- YOLO v7 introduces a new kind of re-parameterization that take care of previous methods' drawback.

Two Important Concepts

There are many complex things presented in the YOLO v7 paper, so before we delve into it let's first clarify two important concepts.

2. MODEL SCALING

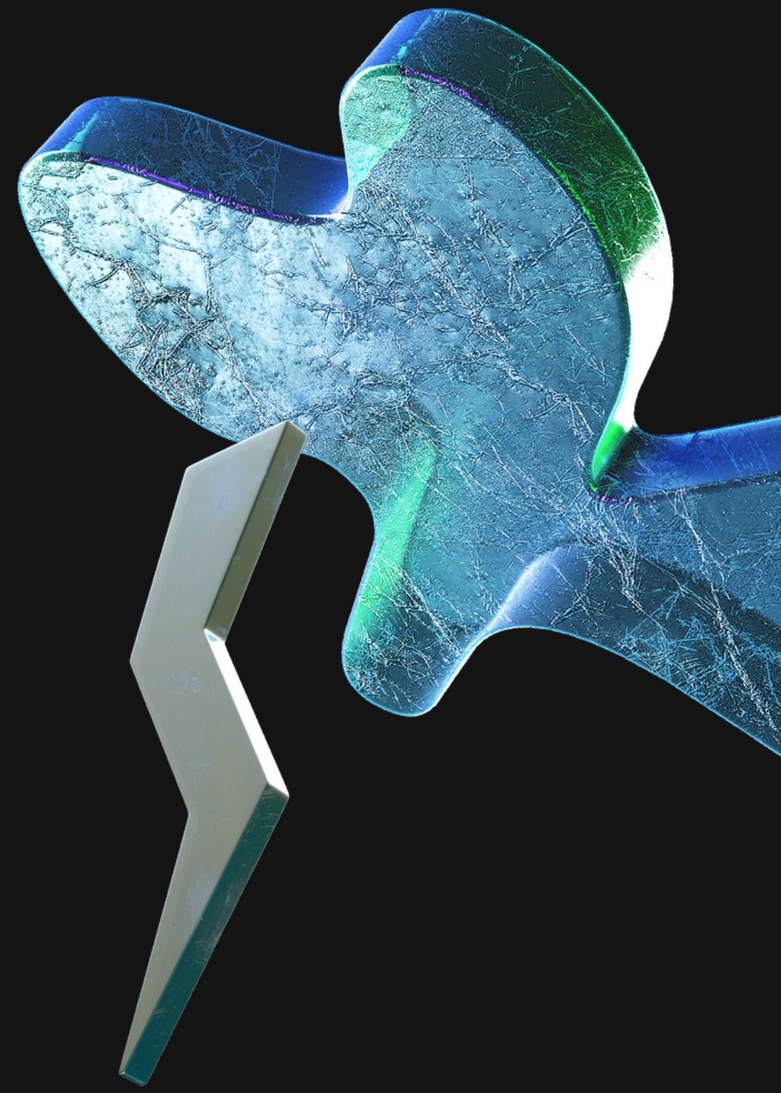
- Model scaling is a way to scale up or down an already designed model and make it fit any computing device.
- This gives the designed architecture great flexibility.
- The former is to train multiple identical models with different training data and then average the weights of multiple trained models.
- There are different types of scaling like resolution scaling (size of the input image), depth scaling (number of layers), width scaling (number of channels), and stage scaling (number of feature pyramids).
- Model scaling basically gives us the trade-off between speed and accuracy..

Model Architecture

1. E-ELAN (Extended Efficient Layer Aggregation Network)

- The E-ELAN is the computational block in the YOLOv7 backbone. It takes inspiration from previous research on network efficiency.

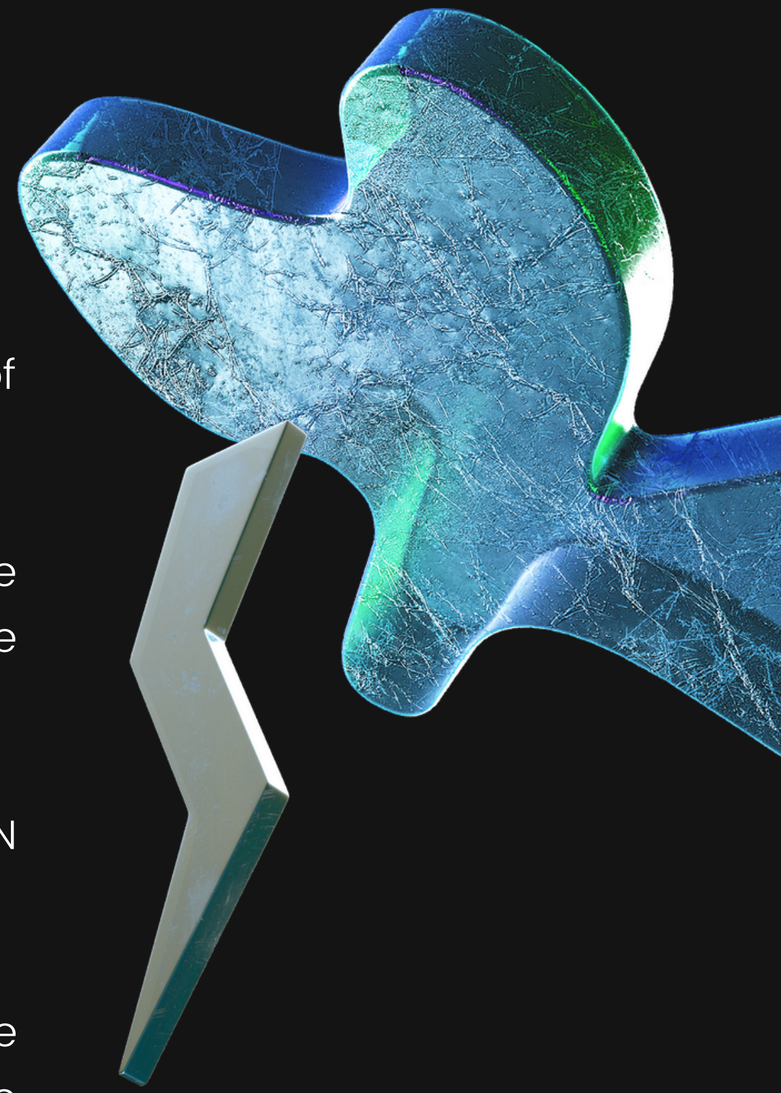
- In simple terms, E-ELAN architecture enables the framework to learn better. It is based on the ELAN computational block. At the time of writing this post, the ELAN paper has not been published.



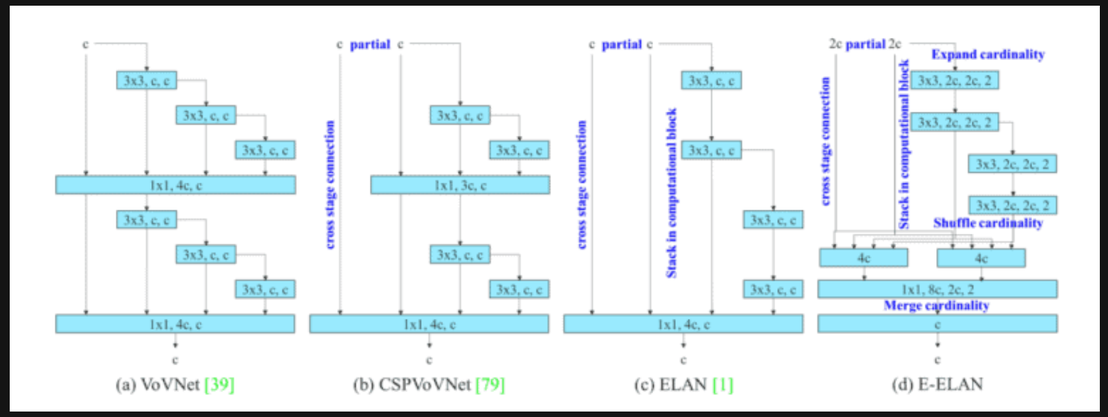
Model Architecture

Evolution of Efficient Architectures

- The design of an efficient architecture primarily focuses on the number of parameters, amount of computations, and computational density of a model.
- The VovNet model goes a bit further and analyses the influence of the input/output channel ratio, the number of branches of the architecture, and the element-wise operation on the network inference speed.
- The subsequent prominent development in architecture search is called ELAN and YOLO v7 extends that and calls it E-ELAN.
- The conclusions drawn from the ELAN paper were that by controlling the shortest longest gradient path, a deeper network can learn and converge effectively.



Model Architecture



How E-ELAN does its job?

- E-ELAN strategy is to use group convolution to expand the channel and cardinality of computational blocks.
- It applies the same group parameter and channel multiplier to all the computational blocks of a computational layer.
- Then, the feature map calculated by each computational block is shuffled into g groups according to the set group parameter g and concatenated together.
- At this time, the number of channels in each group of feature maps will be the same as the number of channels in the original architecture.
- Finally, add g groups of feature maps to perform merge cardinality. In addition to maintaining the original ELAN design architecture, E-ELAN can also guide different groups of computational blocks to learn more diverse features.

Model Architecture

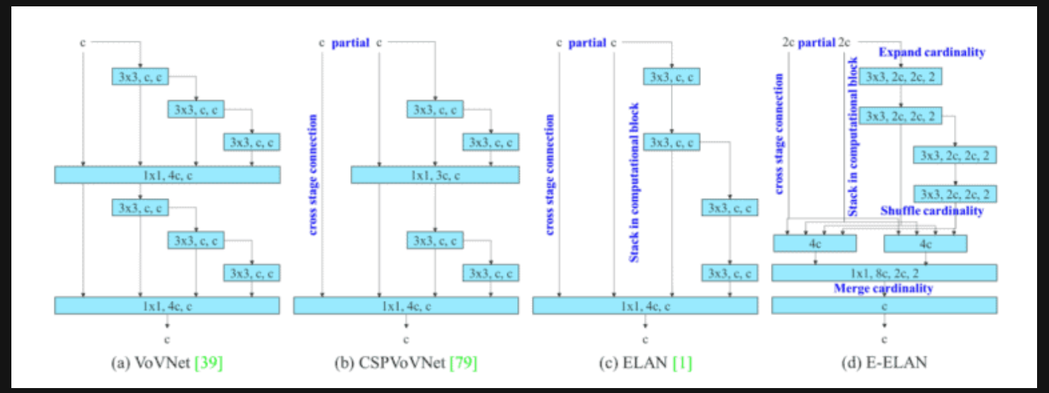
How E-ELAN does its job?

- Regardless of the gradient path length and the stacking number of computational blocks in large-scale ELAN, it has reached a stable state.

- If more computational blocks are stacked unlimitedly, this stable state may be destroyed, and the parameter utilization rate will decrease.

- E-ELAN uses expand, shuffle, and merge cardinality to achieve the ability to continuously enhance the learning ability of the network without destroying the original gradient path.

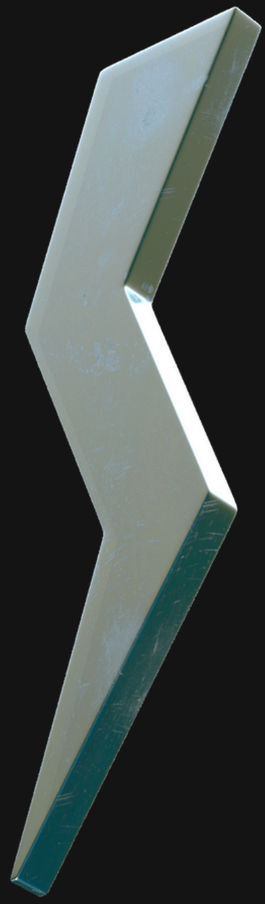
- E-ELAN only changes the architecture in the computational block, while the architecture of the transition layer is entirely unchanged.



Model Architecture

2. Model scaling for concatenation-based models

- The primary purpose of model scaling is to adjust some attributes of the model and generate models of different scales to meet the needs of different inference speeds.
- In Google's famous architecture called EfficeintNet, they scale between width, depth, and resolution.
- But later on, researchers tried to see the effect of vanilla convolution and group convolution on the amount of parameter and computation when performing scale.
- NAS (Network Architecture Search) is a commonly used model scaling method. It is used by researchers to iterate through the parameters to find the best scaling factors.
- However, methods like NAS do parameter-specific scaling. The scaling factors are independent in this case.



Model Architecture

Problems with Efficient Net Scaling Method

- The strategy used by EfficientNet is not suitable for concatenation-based architecture, when scaling up or scaling down is performed on depth, the in-degree of a translation layer which is immediately after a concatenation-based computational block will decrease or increase.

- For instance, scaling-up depth will cause a ratio change between the input channel and output channel of a transition layer, which may lead to a decrease in the hardware usage of the model.

- When the depth factor of a computational block is scaled, the proposed method should also calculate the change of the output channel of that block. Then perform width factor scaling with the same amount of change on the transition layers, and the result is shown in the below figure.

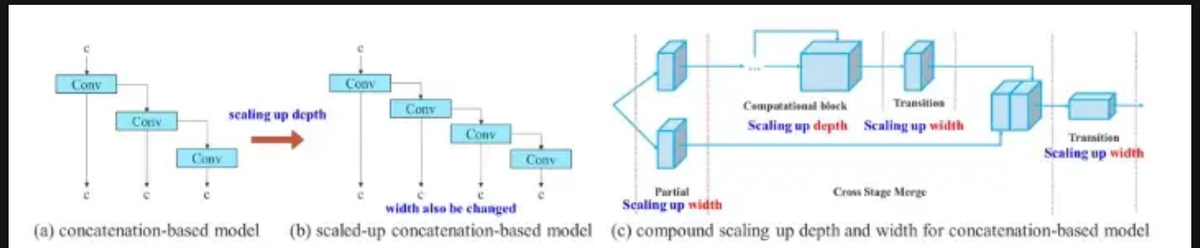


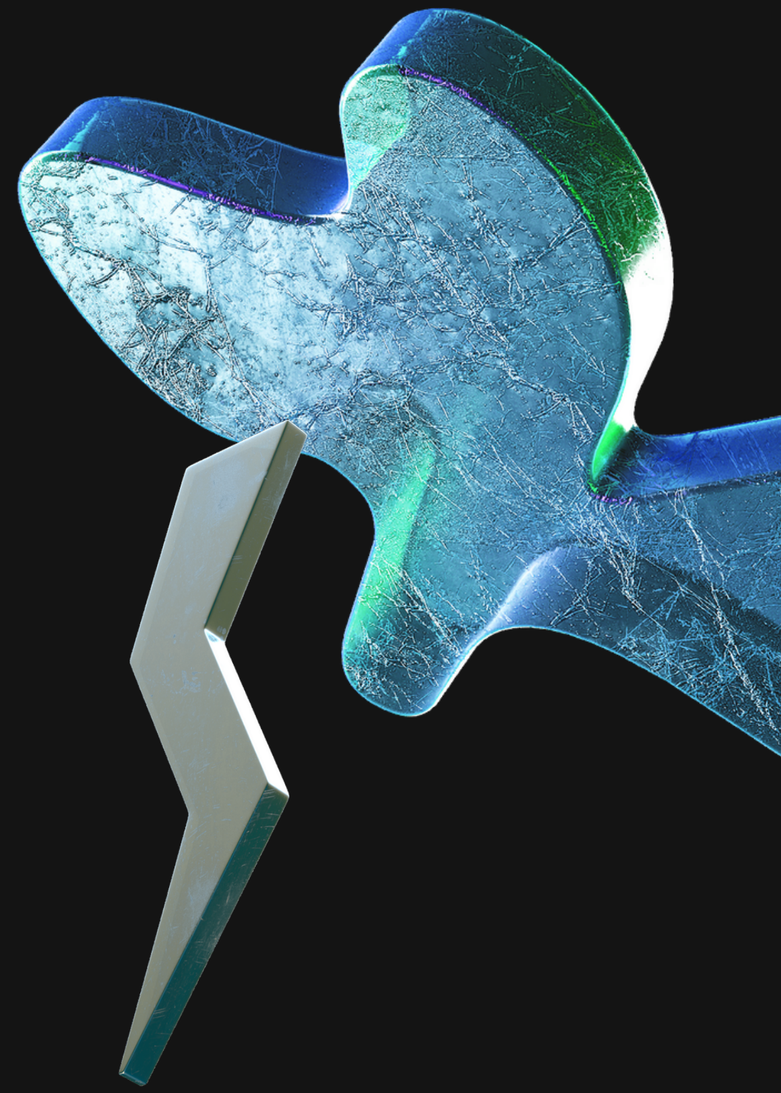
Figure 3: Model scaling for concatenation-based models. From (a) to (b), we observe that when depth scaling is performed on concatenation-based models, the output width of a computational block also increases. This phenomenon will cause the input width of the subsequent transmission layer to increase. Therefore, we propose (c), that is, when performing model scaling on concatenation-based models, only the depth in a computational block needs to be scaled, and the remaining of transmission layer is performed with corresponding width scaling.

Trainable bag-of-freebies

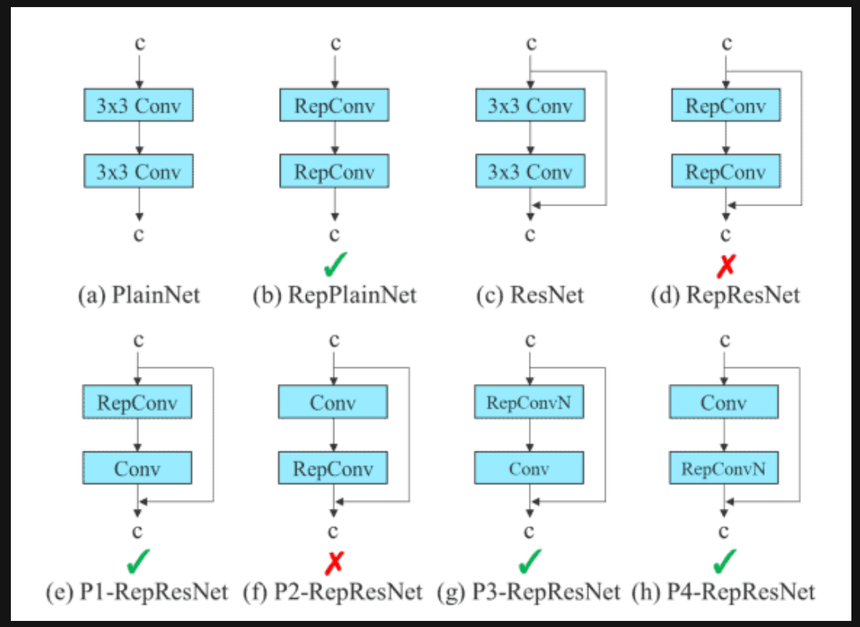
1. Planned Re-parameterized Convolution

- Re-parameterization is a technique used after training to improve the model. It increases the training time but improves the inference results.

- There are two types of re-parametrization used to finalize models, Model level and Module level ensemble.



Model level vs Module level



MODEL LEVEL RE-PARAMETRIZATION

Model level re-parametrization can be done in the following two ways.

- Using different training data but the same settings, train multiple models. Then average their weights to obtain the final model.
- Take the average of the weights of models at different epochs.

MODULE LEVEL RE-PARAMETERIZATION

Recently, Module level re-parameterization has gained a lot of traction in research.

In this method, the model training process is split into multiple modules. The outputs are ensemled to obtain the final model.

The authors in the YOLOv7 paper show the best possible ways to perform module-level ensemble

Re-parameterized Convolution

Rep Conv

- Before we discuss this in detail, let's look at another type of convolutional block called RepConv.

- RepConv is similar to what we see in Resnet but apart from one identity connection it also has one more connection with 1x1filter in between.

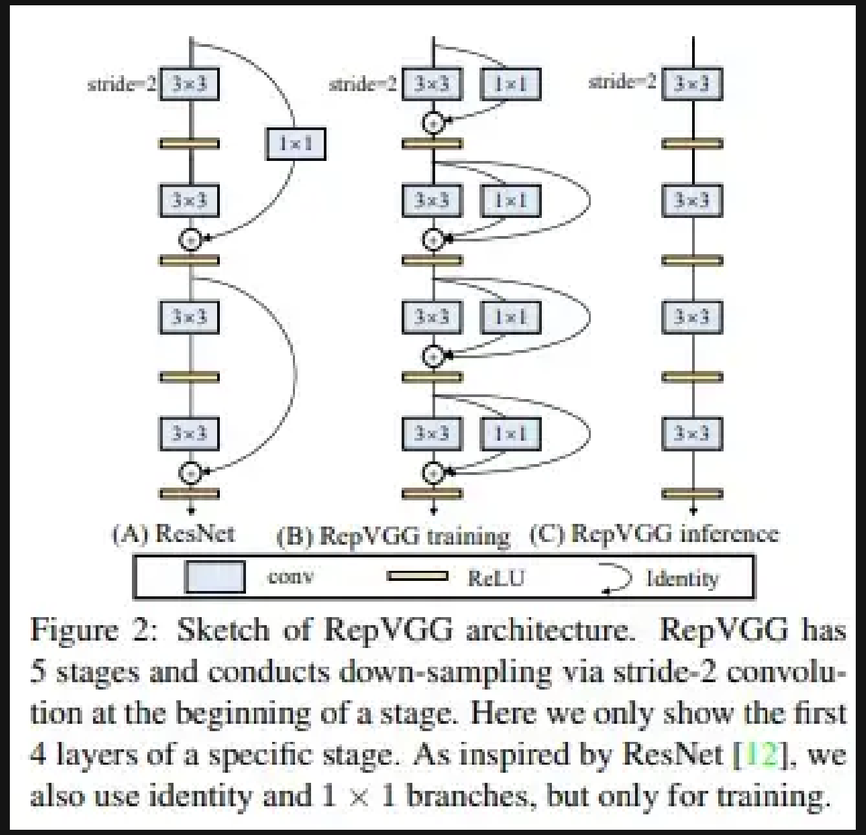
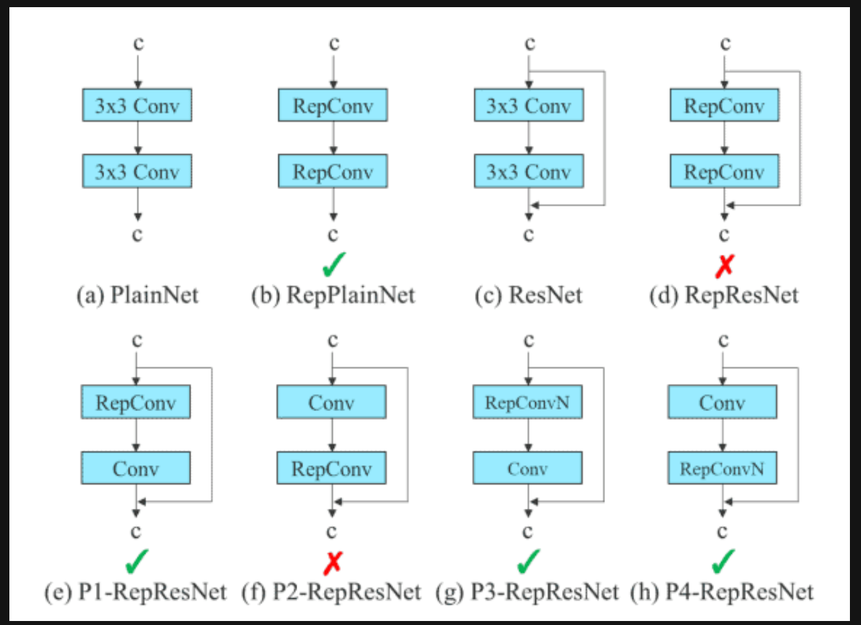


Figure 2: Sketch of RepVGG architecture. RepVGG has 5 stages and conducts down-sampling via stride-2 convolution at the beginning of a stage. Here we only show the first 4 layers of a specific stage. As inspired by ResNet [12], we also use identity and 1×1 branches, but only for training.

Trainable bag-of-freebies

Using RepConv in YOLOv7

- YOLOv7 researchers used gradient flow propagation paths to analyze how re-parameterized convolution should be combined with different networks.
- In the diagram, The 3×3 convolution layer of the E-ELAN computational block is replaced with the RepConv layer.
- Experiments we carried out by switching or replacing the positions of RepConv, 3×3 Conv, and Identity connection.
- The residual bypass arrow shown above is an identity connection. It is nothing but a 1×1 convolutional layer.

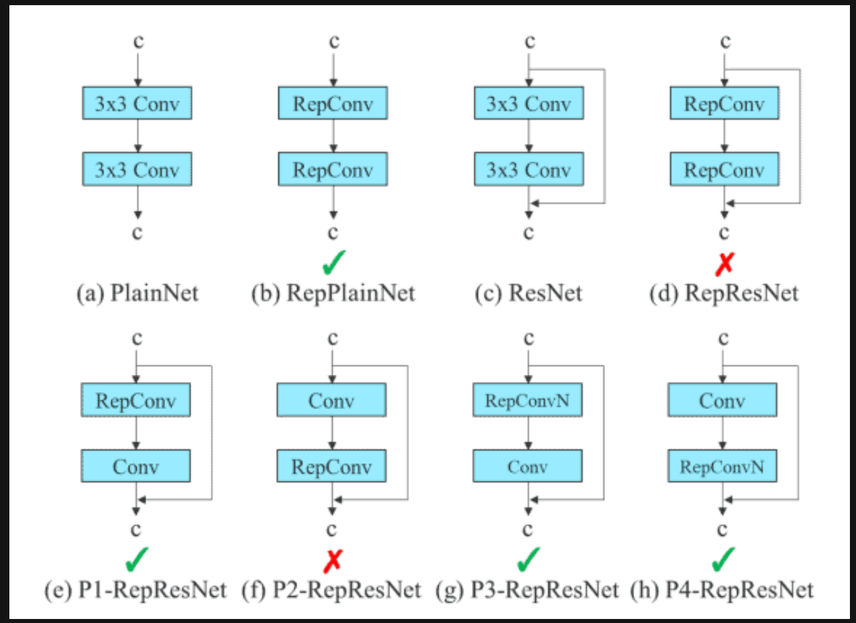


Trainable bag-of-freebies

Using RepConv in YOLOv7

- Including RepConv, YOLOv7 also performs re-parameterization on Conv-BN (Convolution Batch Normalization), OREPA(Online Convolutional Re-parameterization), and YOLO-R to get the best results.

- The diagram on the right shows in what way the conv blocks should be placed. Out of 8 combinations, 4 work out great (marked with a green tick in the below image).

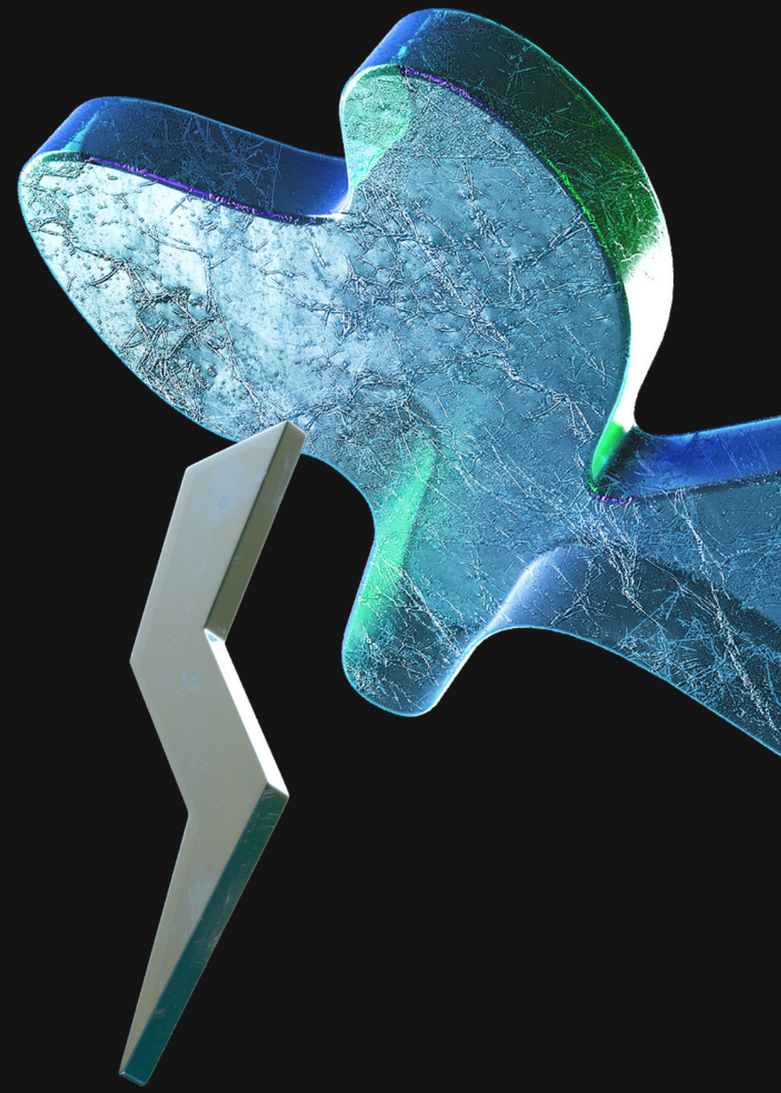


Trainable bag-of-freebies

2. Coarse for auxiliary and fine for lead loss

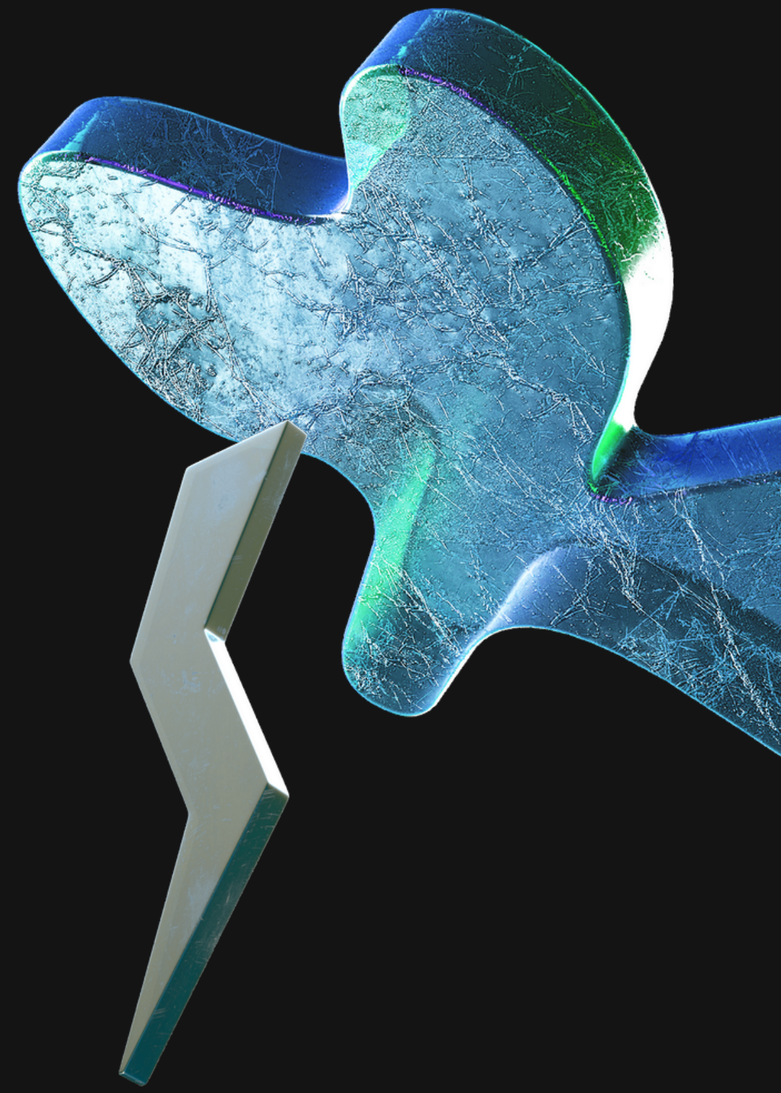
- Deep supervision is a technique that is often used in training deep networks. Its central concept is to add an extra auxiliary head in the middle layers of the network, and the shallow network weights with assistant loss as the guide.

- Even for architectures such as ResNet and DenseNet which usually converge well, deep supervision can still significantly improve the model's performance on many tasks.



Trainable bag-of-freebies

2. Coarse for auxiliary and fine for lead loss

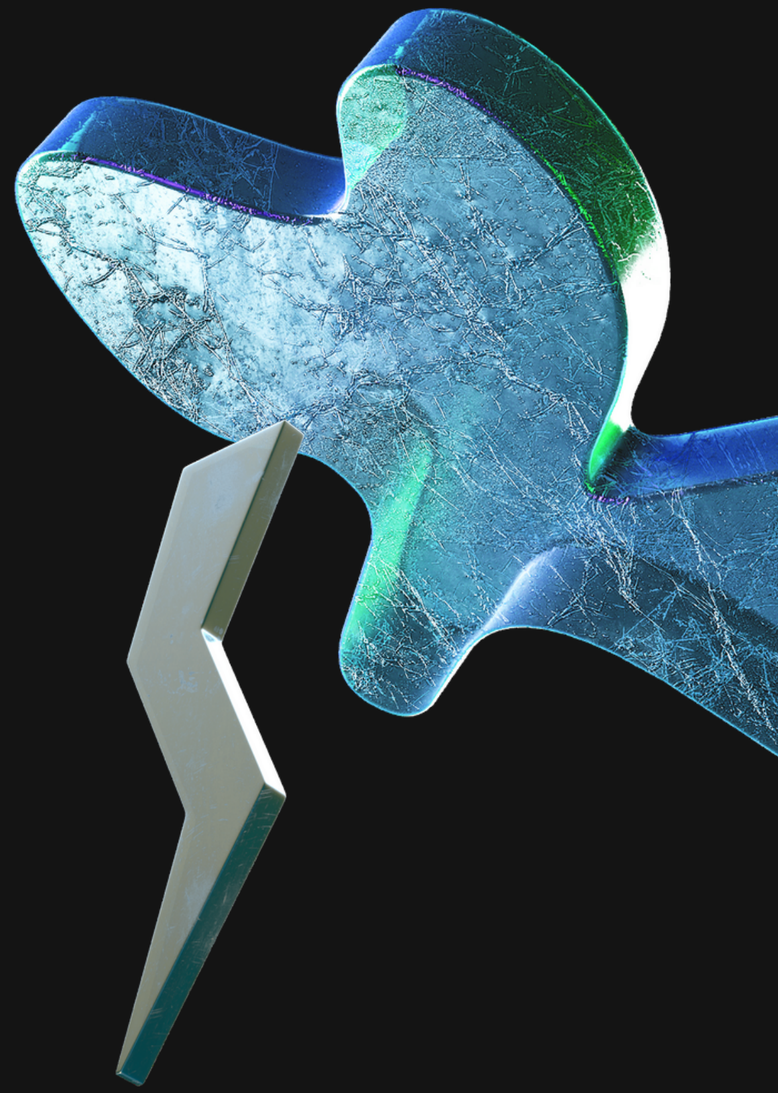


- As you already know by now, YOLO architecture comprises a backbone, a neck, and a head.
- The head contains the predicted outputs. YOLOv7 does not limit itself to a single head.
- It has multiple heads to do whatever it wants. Interesting isn't it?
- In YOLOv7, the head responsible for final output is called the Lead Head.
- And the head used to assist training in the middle layers is called the Auxiliary Head.

Trainable bag-of-freebies

2. Coarse for auxiliary and fine for lead loss

- With the help of an assistant loss, the weights of the auxiliary heads are updated.
- It allows for Deep Supervision and the model learns better.
- These concepts are closely coupled with the Lead Head and the Label Assigner.

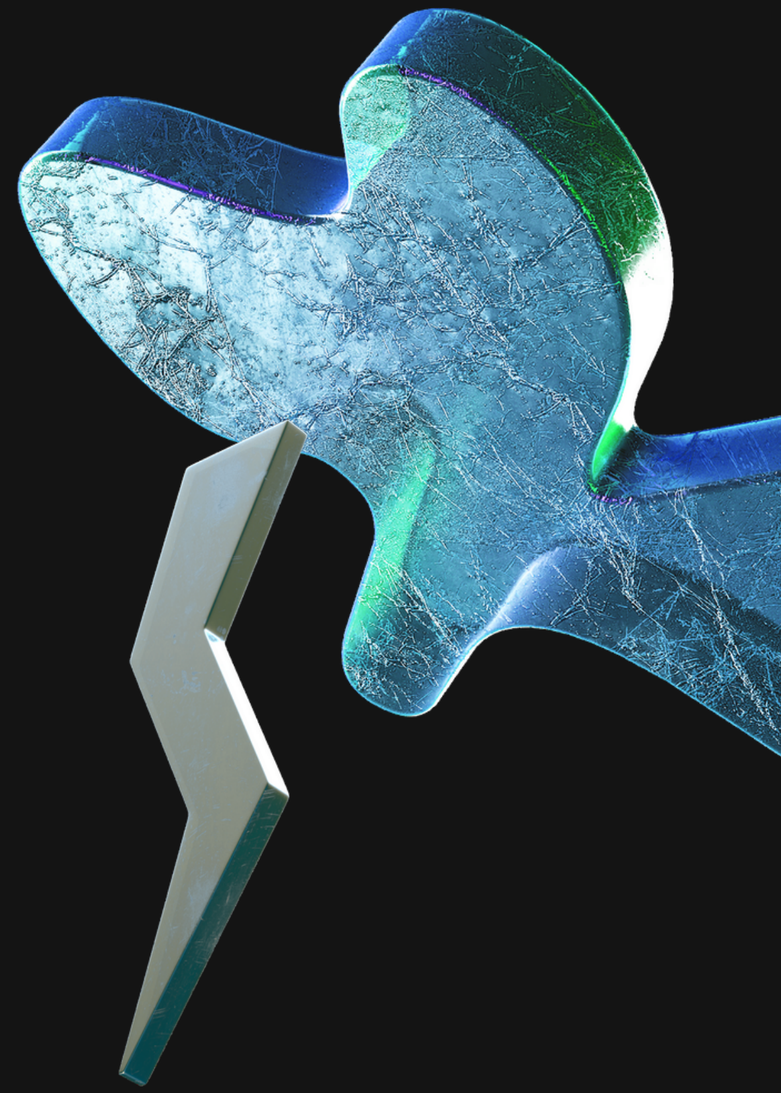


Trainable bag-of-freebies

Soft and Hard labels

- In the past, in the training of the deep network, label assignment usually refers directly to the ground truth and generates hard labels according to the given rules.

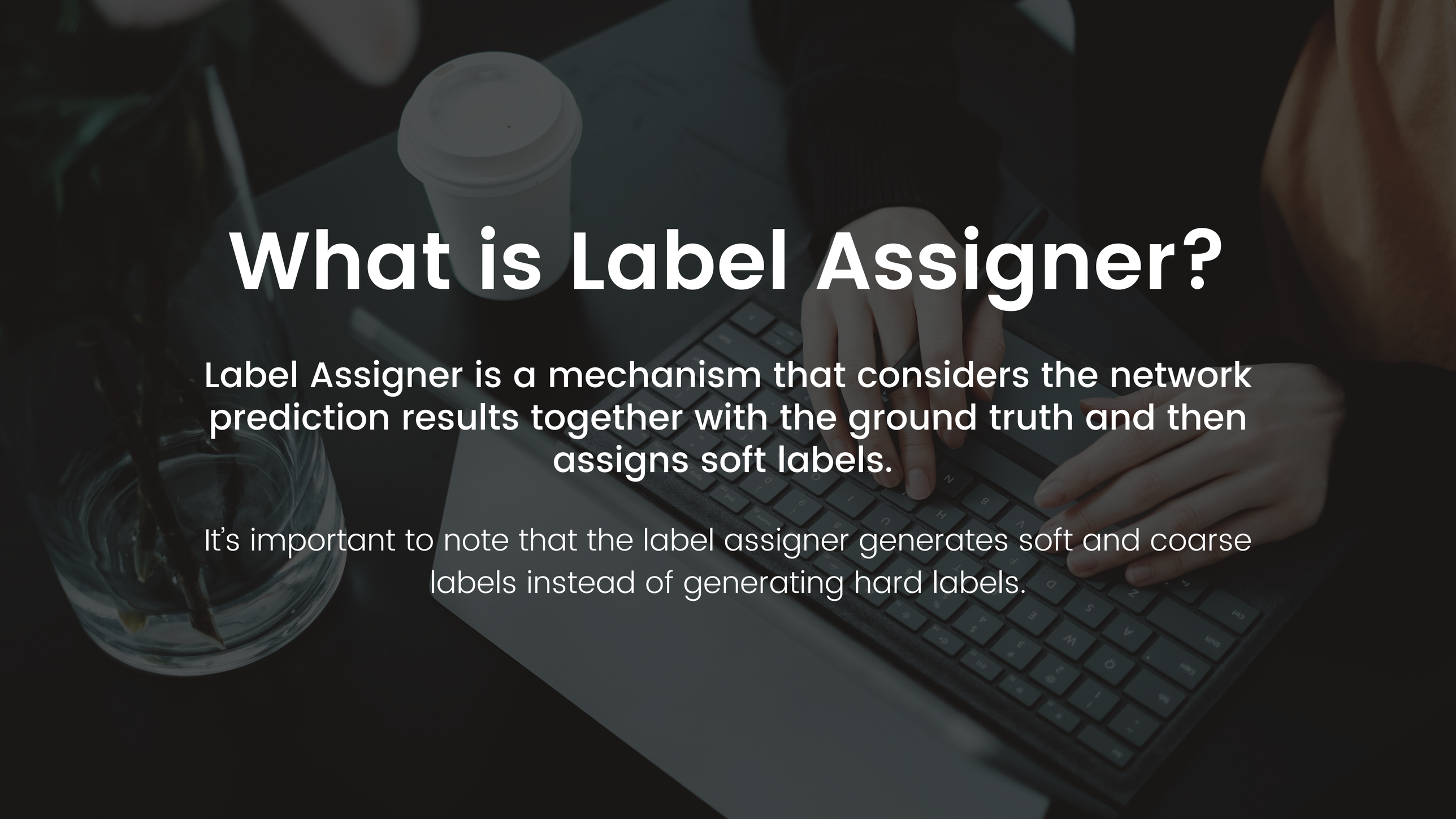
- However, in recent years, if we take object detection as an example, researchers often use the quality and distribution of prediction output by the network and then consider together with the ground truth to use some calculation and optimization methods to generate a reliable soft label.



But here’s a question: “How to assign a soft label to the auxiliary head and lead head ?”.

ANSWER

Using Label Assigner



What is Label Assigner?

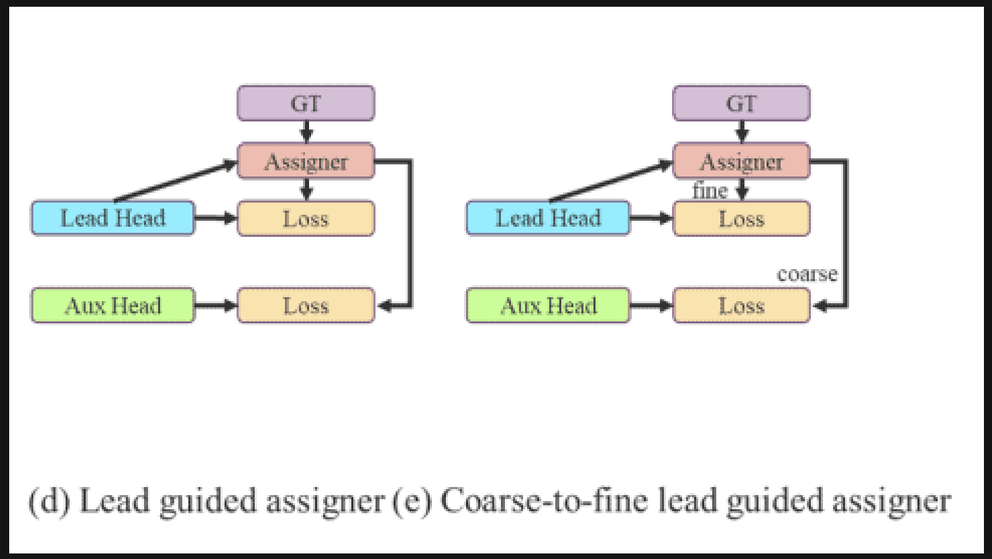
Label Assigner is a mechanism that considers the network prediction results together with the ground truth and then assigns soft labels.

It’s important to note that the label assigner generates soft and coarse labels instead of generating hard labels.

Label Assigner

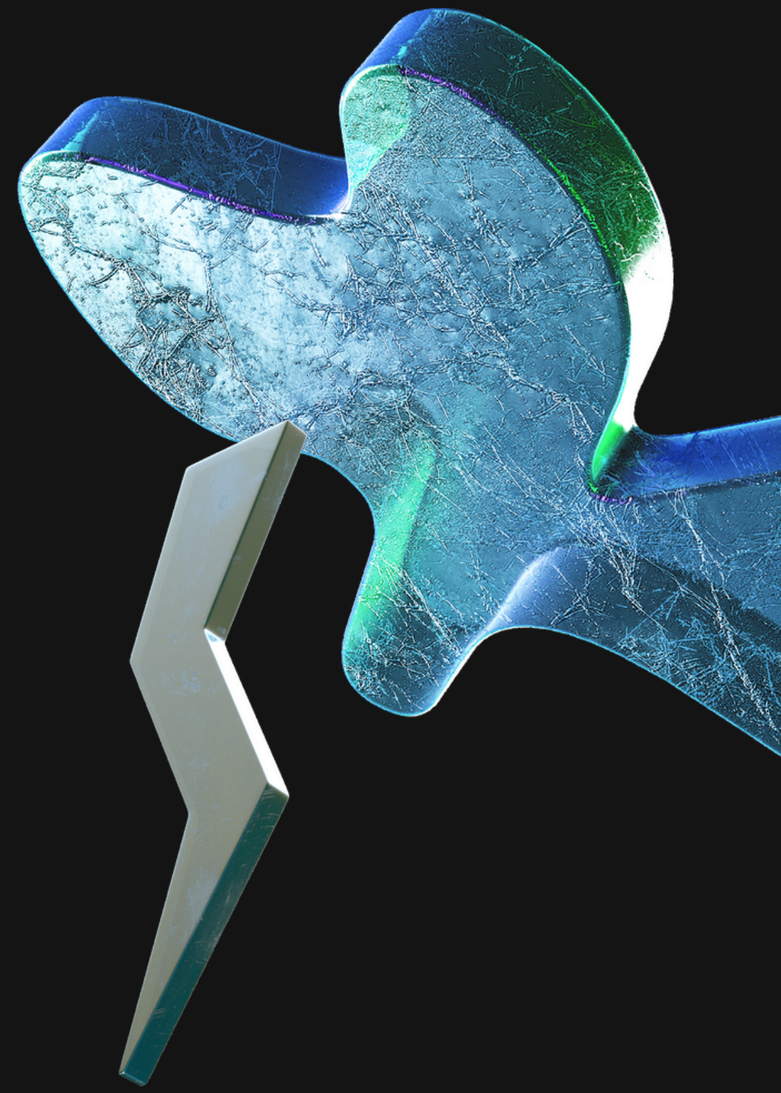
1. Lead Head Guided Label Assigner

2. Coarse-to-Fine Lead Head Guided Label Assigner



(d) Lead guided assigner (e) Coarse-to-fine lead guided assigner

Lead Head Guided Label Assigner



The Lead Head Guided Label Assigner encapsulates the following three concepts.

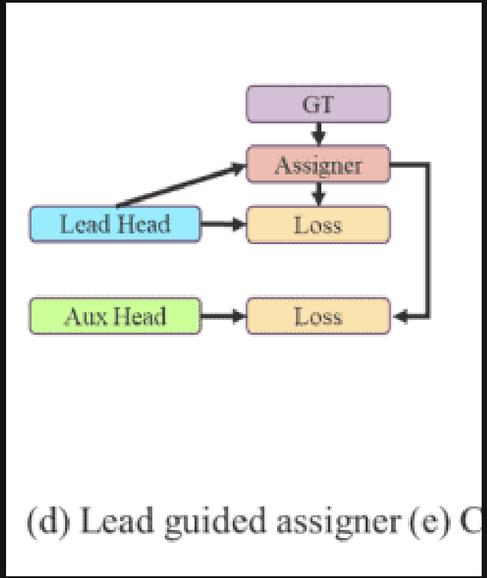
- Lead Head
- Auxiliary Head
- Soft Label Assigner

Lead Head Guided Label Assigner

The Lead Head in the YOLOv7 network predicts the final results. Soft labels are generated based on these final results.

The important part is that the loss is calculated for both the lead head and the auxiliary head based on the same soft labels that are generated.

Ultimately, both heads get trained using the soft labels. This is shown in the right image.



One may ask here, “why soft labels?”. The authors have put it quite well in the paper:

“The reason to do this is that the lead head has a relatively strong learning capability. So the soft label generated from it should be more representative of the distribution and correlation between the source data and the target. By letting the shallower auxiliary head directly learn the information that the lead head has learned, the lead head will be more able to focus on learning residual information that has not yet been learned.”

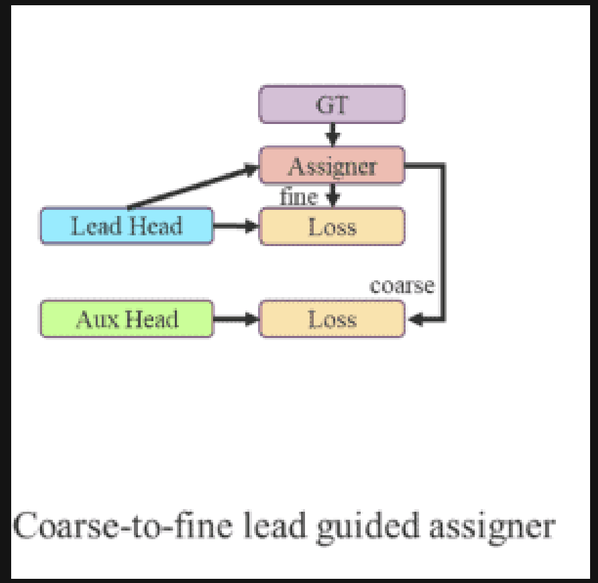
Coarse-to-Fine Lead Head Guided Label Assigner

Now, coming to the **coarse-to-fine** labels as shown in the right image. Actually, two different sets of soft labels are generated.

- A fine label to train the lead head
- A set of coarse labels to train the auxiliary head.

The fine labels are the same as the directly generated soft labels.

However, to generate the coarse labels, more grids are treated as positive targets. This is done by relaxing the constraints of the positive sample assignment process.



Coarse-to-fine lead guided assigner

The below figure shows the object detector architecture “without” and “with” deep supervision.

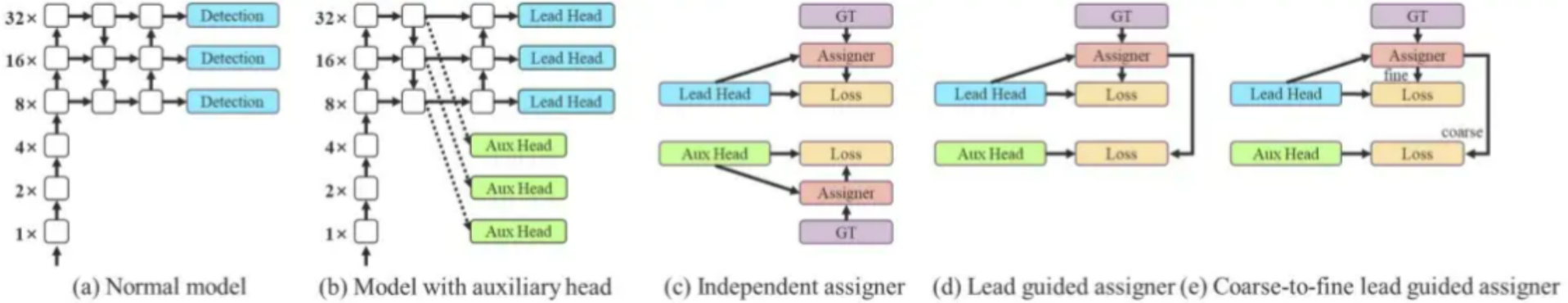


Figure 5: Coarse for auxiliary and fine for lead head label assigner. Compare with normal model (a), the schema in (b) has auxiliary head. Different from the usual independent label assigner (c), we propose (d) lead head guided label assigner and (e) coarse-to-fine lead head guided label assigner. The proposed label assigner is optimized by lead head prediction and the ground truth to get the labels of training lead head and auxiliary head at the same time. The detailed coarse-to-fine implementation method and constraint design details will be elaborated in Apendix.

The differences between the basic YOLOv7 versions

- YOLOv7 is the basic model that is optimized for ordinary GPU computing.
- YOLOv7-tiny is a basic model optimized for edge GPU.
- YOLOv7-W6 is a basic model optimized for cloud GPU computing.
- Other variations include YOLOv7-X, YOLOv7-E6, and YOLOv7-D6, which were obtained by applying the proposed compound scaling method

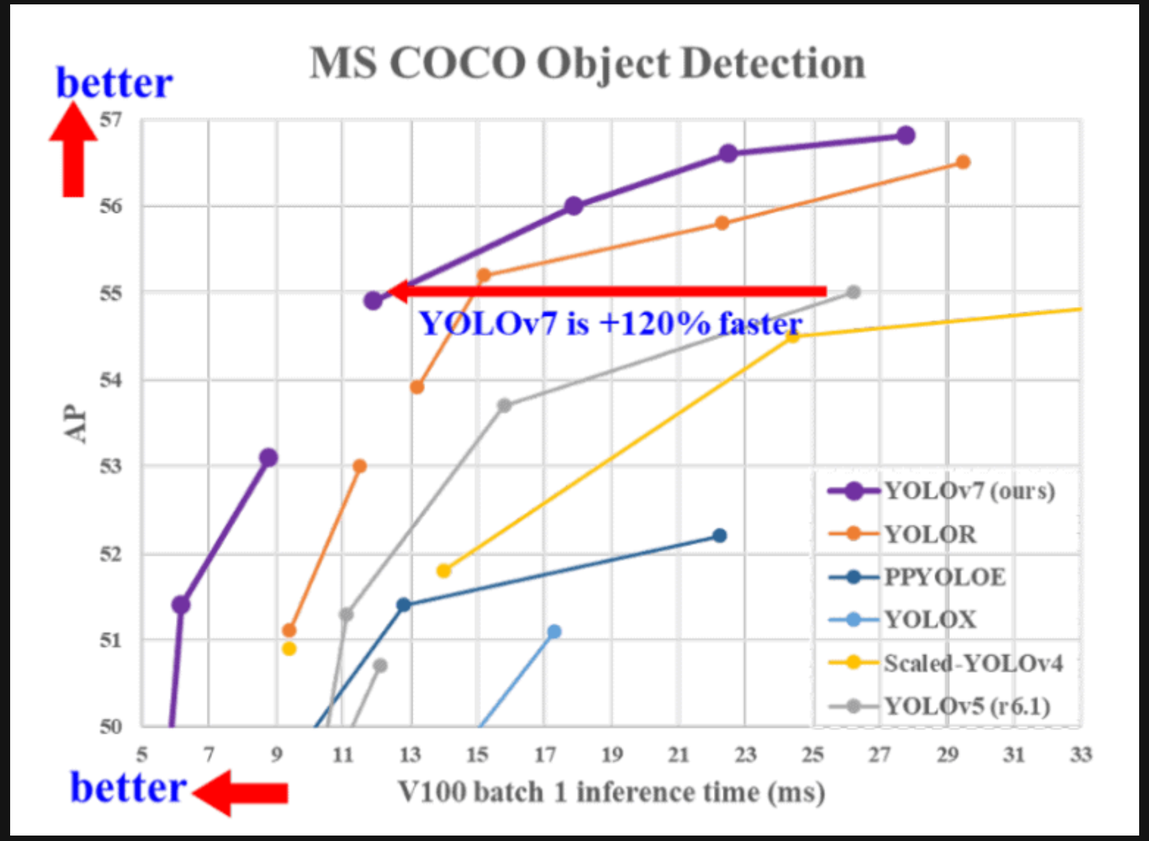
Results

It is already established that the YOLOv7 has the highest FPS and mAP in the range of 5 FPS to 160 FPS.

Model	Parameters (million)	FPS	AP test (%)
YOLO7-Tiny	6.2	286	38.7
YOLOv7	36.9	161	51.4
YOLOv7-X	71.3	114	53.1
YOLOv7-W6	70.04	84	54.9
YOLOv7-E6	**97.2**	**56**	**56.0**
YOLOv7-D6	154.7	44	56.6
YOLOv7-E6E	151.7	36	56.8

YOLOv7 Experiments and Results

All the YOLOv7 models surpass the previous object detectors in speed and accuracy in the range of 5 FPS to 160 FPS.

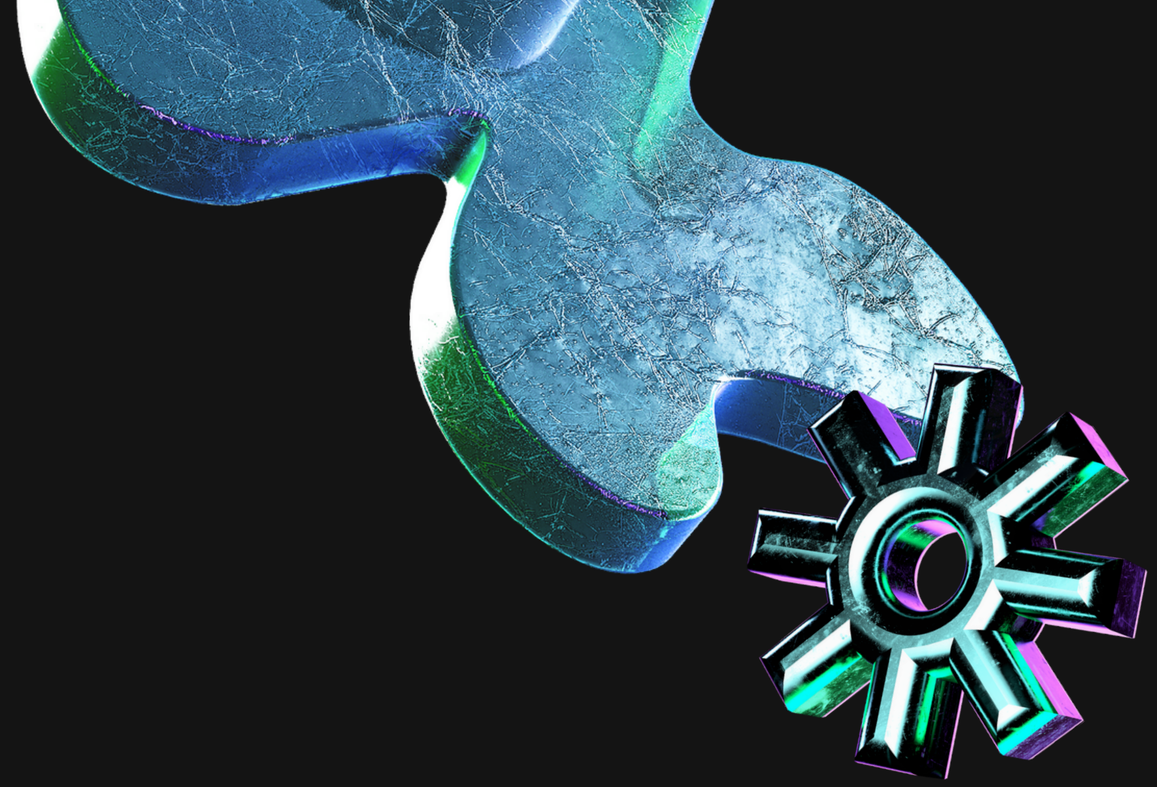


mAP Comparison: YOLOv7 vs Others

The following table shows the comparison of YOLOv7 models with other baseline object detectors.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

VISUAL LEARNERS



Thank you

You have Completed the lesson! We hope you learned something new.