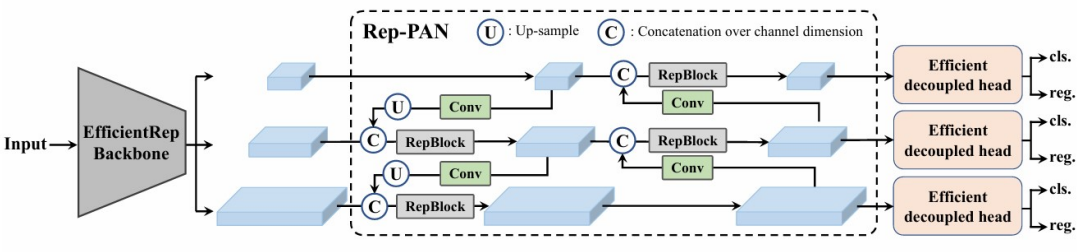
# YOLO v6

## 1. Network Design




### 1.1 Background

(1) Multi branch networks → better classification performance, but often with the reduction of the parallelism and results in an increase of inference latency.

(2) Plain single-path networks like **VGG** → high parallelism, less memory footprint, leading to higher inference efficiency.

Lately a new network **RepVGG** achieves a better speed accuracy trade-off.

The network design of YOLO v6 is inspired by the above works.

### What is VGG?

使用 **多个小卷积核（3×3）**来代替大卷积核（如 5×5 或 7×7），整个网络几乎只使用 3×3 卷积和 2×2 最大池化

### What is RepVGG?

RepVGG, a structural **re-parameterization** method is proposed to decouple the training-time multi-branch topology with an inference-time plain architecture to achieve a better speed accuracy trade-off.

基本架构：将 20 多层 3x3 卷积堆起来，分成 5 个 stage，每个 stage 的第一层是 stride=2 的降采样，每个卷积层都用 ReLU 作为激活函数。

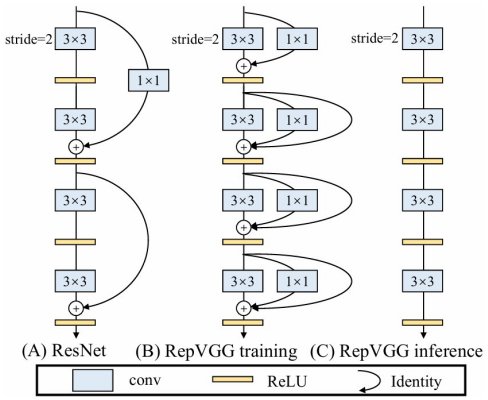



Figure 2: Sketch of RepVGG architecture. RepVGG has 5 stages and conducts down-sampling via stride-2 convolution at the beginning of a stage. Here we only show the first 4 layers of a specific stage. As inspired by ResNet [12], we also use identity and $1 \times 1$ branches, but only for training.

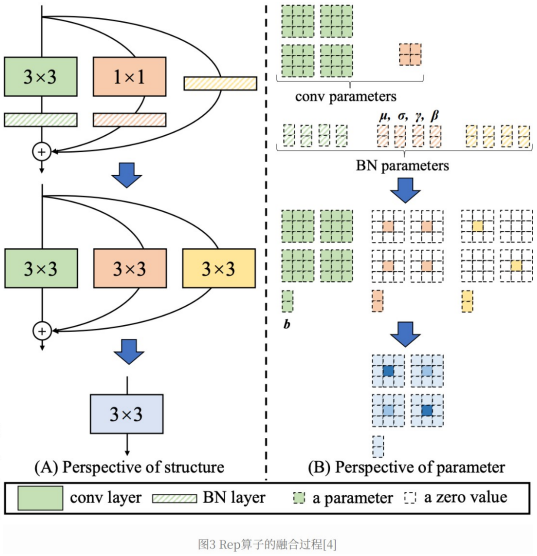


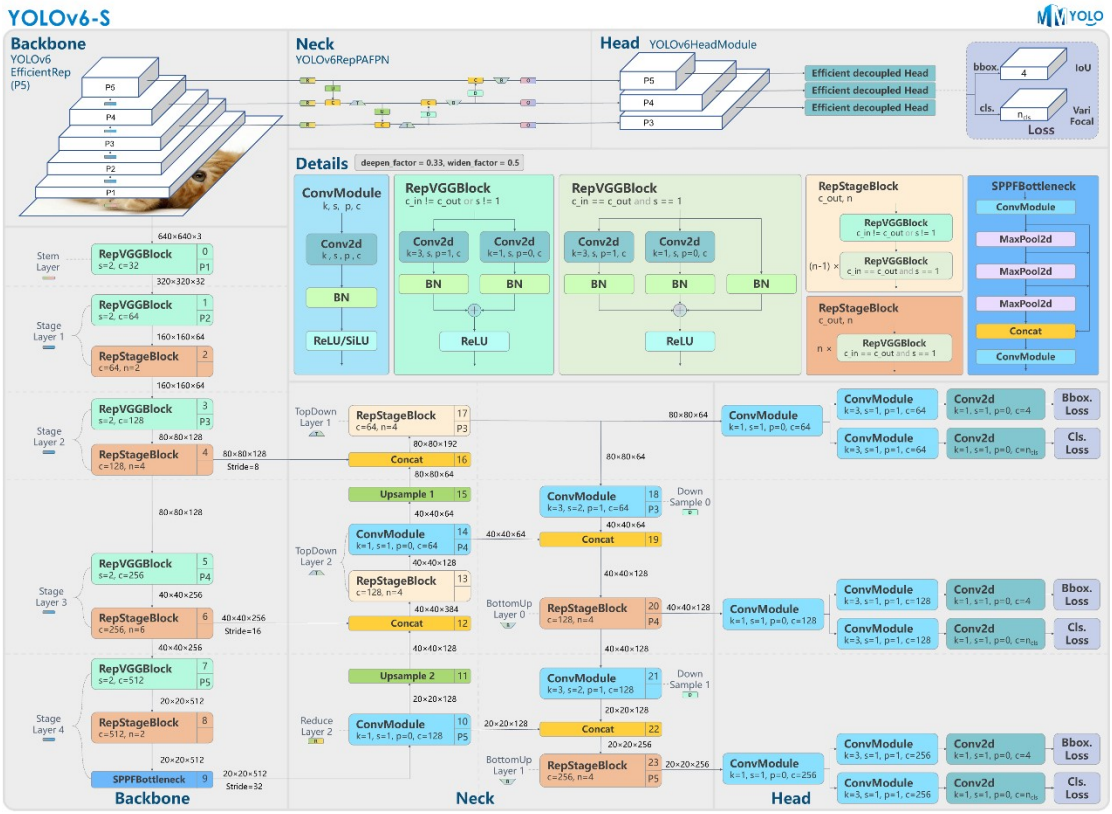

图3 Rep算子的融合过程[4]

**RepVGG** 优点：

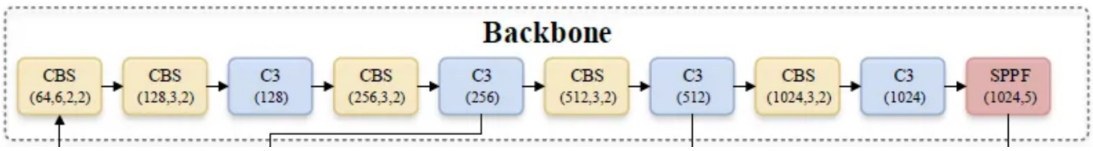
1. 3x3 卷积非常快

2. 单路架构非常快，因为并行度高；省内存；灵活性更好，容易改变各层的宽度（如剪枝）。
3. RepVGG 主体部分只有**一种算子**：3x3 卷积接 ReLU。在设计专用芯片时，给定芯片尺寸或造价，我们可以集成海量的 **3x3 卷积-ReLU 计算单元**来达到很高的效率。
4. 结构重参数化，同时利用**多分支模型训练时的优势**（性能高）和**单路模型推理时的好处**（速度快、省内存）

### 1.2 EfficientRep Backbone




YOLO v5




YOLO v6

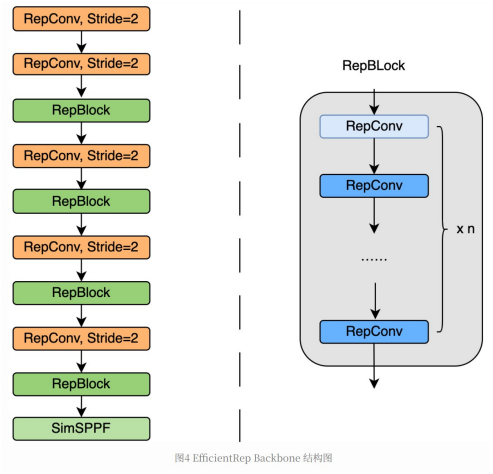



图4 EfficientRep Backbone 结构图

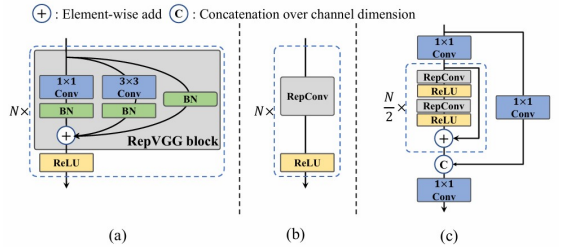



Figure 3: (a) RepBlock is composed of a stack of RepVGG blocks with ReLU activations at training. (b) During inference time, RepVGG block is converted to RepConv. (c) CSPStackRep Block comprises three $1\times1$ convolutional layers and a stack of sub-blocks of double RepConvs following the ReLU activations with a residual connection.

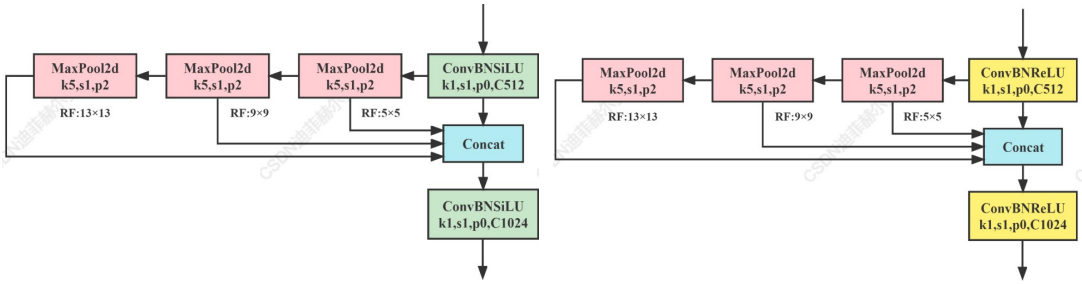
An efficient re-parameterizable backbone denoted as **EfficientRep.**
For small models, the main component of backbone is **RepBlock.**
When training, **RepBlock = RepVGG Blocks + Relu**
When inferring, **RepBlock = RepConv (3x3 convolutional layers) + Relu**
For medium/large models, the main component of backbone is **CSPStackRep Block**.
To achieve a better trade-off between the computation burden and accuracy.

**为什么训练的时候使用多分支?**
能够增加模型的表征能力，提高模型的准确度

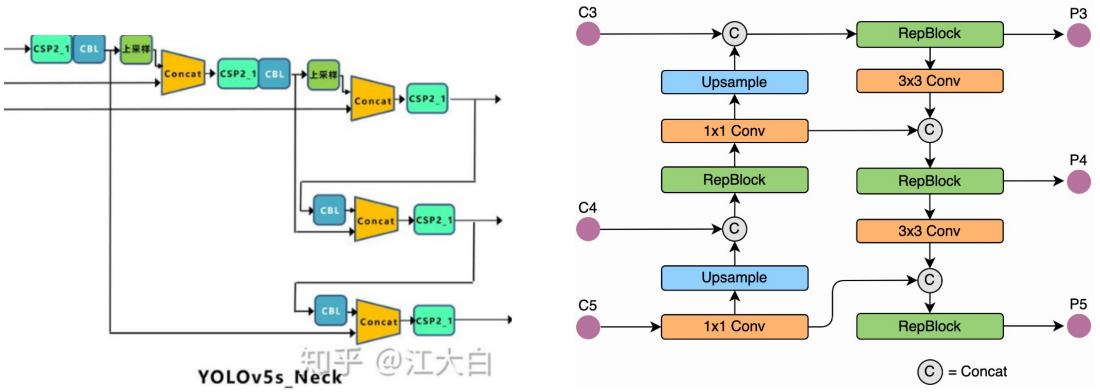
| Identity branch | $1\times1$ branch | Accuracy | Inference speed w/o re-param |
|---|---|---|---|
| | | 72.39 | 1810 |
| ✓ | | 74.79 | 1569 |
| | ✓ | 73.15 | 1230 |
| ✓ | ✓ | **75.14** | 1061 |

**为什么推理的时候使用单分支?**
使用单分支更快，多分支的每个分支计算量可能不同，但是需要将每个分支的结果相加，导致并行度下降，降低速度，另外多分支调用的算子更多，更耗时
更省内存
更加灵活

**SPPF** VS **SimSPPF**?
激活函数由 SiLU 变成 **ReLU**，计算速度更快



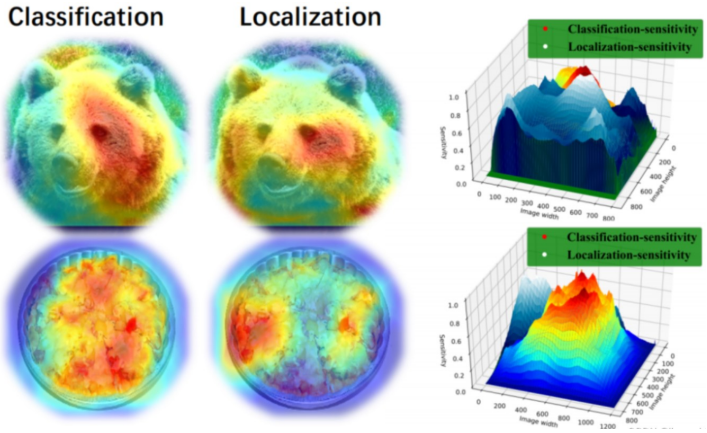
### 1.3 Rep-PAN Neck



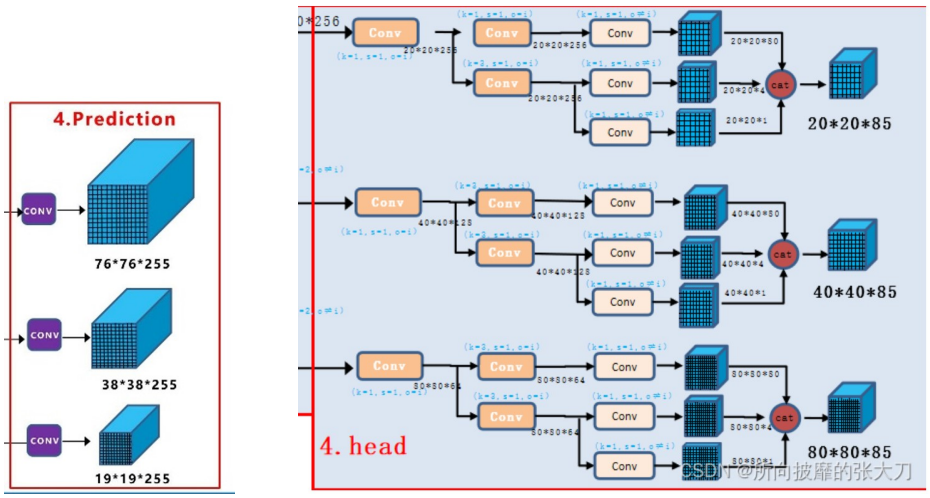

基于硬件感知神经网络设计思想，在 YOLO v5 基础上改动，同样采用 **RepStageBlock** 或 **BepC3StageBlock** 对原本的 CSPLayer 进行了替换。目的是在硬件上达到高效推理的同时，保持较好的多尺度特征融合能力，达到更好的精度与速度的平衡。

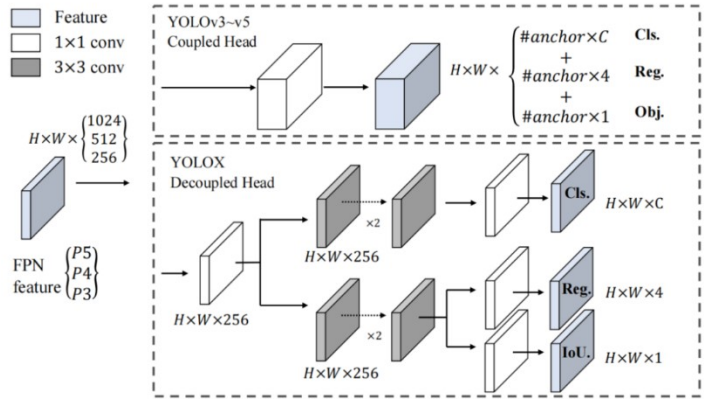
需要注意的是，Neck 中 Down Sample 部分仍然使用了 stride=2 的 **3×3 ConvModule**，而不是像 Backbone 一样替换为 RepVGGBlock。

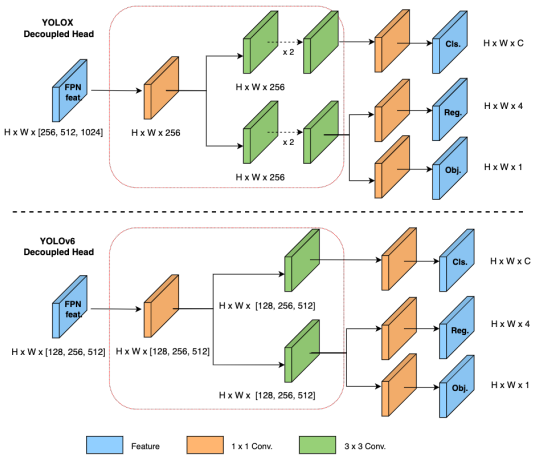
### 1.4 Efficient Decoupled Head




分类任务更关注所提取的特征与已有类别哪一类最为相近，而定位任务更加关注与 GT Box 的位置坐标从而进行边界框参数修正。因此如果采取用同一个特征图（耦合头）进行分类和定位，效果会不好。fc-head 更适合分类任务，conv-head 更适合定位任务。




目标检测任务中，需要预测类别、置信度、位置。传统的预测头（如 YOLOv5）将这三项任务通过一个共享的卷积分支来完成，这种方式称为 **Coupled Head（耦合头）**。

FCOS 和 YOLOX 中则将分类和回归分支解耦，并且在每个分支中引入额外的两个 3×3 卷积层以提高性能。这种方式叫 **Decoupled Head（解耦头）**，将这三项任务分开处理，每个任务有独立的卷积分支。

YOLO v6 采用混合通道策略来建立一个**更有效的解耦头**。

- 将解耦头中最后的两个 3*3 卷积层改成一层；

- 将这个 3*3 卷积层的输出通道改为与输入通道一致。

这些修改进一步**降低了计算成本，以实现更低的推理延迟**。（与 YOLO X 相比）

In YOLOv6, we adopt a **hybrid-channel** strategy to build a more efficient decoupled head. Specifically, we reduce the number of the middle 3x3 convolutional layers to only one. The width of the head is jointly scaled by the width multiplier for the backbone and the neck. These modifications further reduce computation costs to achieve a lower inference latency.

**Output channel = 85**

85 = 4（bbox 回归值: top, bottom, left, right） + 1（目标置信度） + 80（分类置信度）

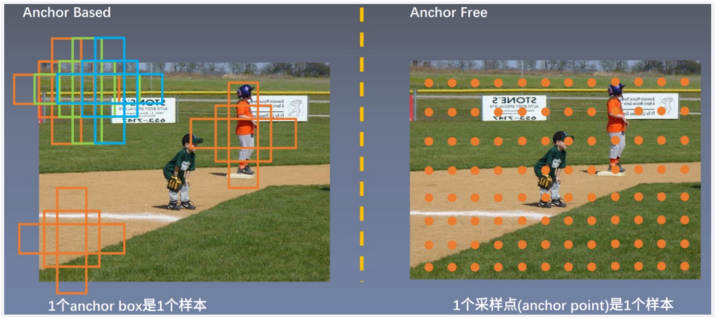
### 1.5 Anchor Free - Anchor Point Based

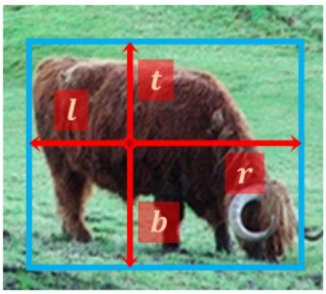
YOLO v2-5 都是 **Anchor Based**，缺点：

1. 需要大量超参数设计（尺度、比例、数量），且泛化能力差
2. 正负样本不平衡
3. 冗余计算和内存开销大

YOLO v6 uses **anchor free detector**: **anchor point based** (from **FCOS**).

不是预测 anchor box 的偏移量，而是直接预测 bbox 的坐标 (x1,y1,x2,y2)。实际上是预测该 anchor point 到 GT bbox 四条边的距离 top, bottom, left, right，然后使用解码器解码到坐标。




**FCOS** first introduced **anchor point based.**

**Q: anchor point 和 yolo v1 的 grid cell 有什么区别?**

YOLOv1 predicts bounding boxes at points near the center of objects. Only the points near the center are used since they are considered to be able to produce higher quality detection. Compared to YOLOv1, FCOS takes advantages of all points in a ground truth bounding box to predict the bounding boxes and the low-quality detected bounding boxes are suppressed by the proposed “center-ness” branch. As a result, FCOS is able to provide comparable recall with anchor-based detectors as shown in our experiments”.

这两种点可以说是相同的，但在正样本分配和预测内容有些区别。

1. YOLO v1 的一个 grid cell 负责预测中心点落在该 grid cell 里的 GT；而所有在 GT bbox 里的 anchor point 都可能负责预测该 GT。
2. YOLO v1 里，一个 grid cell 预测四个值（坐标），而 FCOS 预测五个值（left right top down，还有一个 centerness 中心度）

**Q. l t b r 有没有做特殊处理?**
没有，预测的就是像素值。

**2. Label Assignment**
**Task alignment learning 任务对齐学习**
When training,
0 <= epoch < 4，使用 **ATSS Assigner** 进行 warm-up（静态匹配）
epoch >= 4，切换为 **Task Aligned Assigner**（动态匹配）

**ATSS Assigner（Adaptive Training Sample Selection）**
**① 构造 anchor 区域。**将 anchor point 转化为大小为 5*stride 的 虚拟 anchor 框
**② 计算每个 GT 与所有 anchor 的中心距离**
对于每个 GT，在每个特征层（假设如 P3、P4、P5，有 3 层）上，选出距离 GT 中心最近的 Top-K 个 anchor；构成**初筛样本**（一共是 3 * k 个）。
**③ 计算初筛样本的 IoU 分布**
对每 GT, 计算其**初筛样本**(3 * k 个)的 IoU 的均值 mean 与标准差 std, 将 mean + std 作为该 GT 的正样本的自适应 IoU 阈值。
**④ 筛选正样本**
满足以下两个条件的 anchor 为正样本：IoU > 阈值 且 anchor 的中心点（即 anchor point）落在 GT 框内部
**⑤ 生成标签**
为每个正样本分配对应的 GT 类别和边界框；其余 anchor 被视为负样本或忽略。

**Q：如果两个 GT 都选中同一个 anchor 作为正样本?**
看 anchor 和哪个 GT 的 IoU 更高，就选哪个

**Q：每个 GT 框会在所有特征层上寻找最合适的 anchor，如何做到的?**
第③步是基于所有特征层上的初筛样本的计算来生成阈值的。如果 GT 和某一层上的 anchor 的 IoU 明显高于其他层, 那么该层的 anchor 会被优先选中。如果多个层的 IoU 都差不多，那么都有可能被选中，保证了对边界尺寸目标的公平性。

**TaskAlignedAssigner** 借鉴了 **TOOD**（Task-aligned One-stage Object Detection）中的思想，通过一个综合评分**（alignment metric）**，同时考虑分类分数和回归质量（IOU），来衡量每个预测框与真实框的匹配程度。

**Step1：对于每个 anchor point 和每个 GT 框，计算对齐分数（Align Metric）**
例如, 输入图像为 640×640, 有 3 个 GT。生成的三个特征图尺寸分别为：80×80 40×40 20×20，一共是 6400 + 1600 + 400 = 8400 个 anchor point。
对于每一个 GT，对所有的 anchor point 生成的预测框，基于 **GT 类别对应的分类分数**与**预测框与 GT 的 IoU** 的加权得到一个对齐分数 **Align Metric**
最终得到一个对齐分数矩阵，尺寸为（8400，3）。

**Step2：筛选候选正样本**
**① 空间筛选**

只考虑那些**落在 GT 框中心区域** 的 anchor point。**中心区域**是以 GT 框中心为中心，扩展一个固定半径（如 2.5 * stride）形成的矩形。目的是排除远离目标的点，减少计算量。

**② 分数筛选**

对每个 GT 框，从其候选点中选出对齐分数最高的 Top-K 个点（如 K=13）。这些点被认为是该 GT 框的“最可能正样本”。

**③ 合并筛选结果**

这些点构成了候选正样本。生成一个布尔类型的张量 fg_mask，形状为 (batch_size, num_anchor points)，标记哪些 anchor points 被选为**正样本，**从而参与损失函数的计算。

**Step3：处理多 GT 冲突**

如果一个 anchor point 被多个 GT 框选中，则分配给 IOU 最大的那个 GT 框

**Step4：生成训练标签**

为每个正样本分配：类别标签，边界框坐标，分类得分，对应的 GT 索引，用于后续损失函数计算。

**Q：为什么 YOLO v6 在训练初期要用 ATSS 进行 warm-up?**

在训练初期，模型参数尚未收敛，分类分数和预测框与 GT 的 IoU 都不可靠。如果直接使用 TaskAlignedAssigner（需要预测分数参与匹配），可能会导致错误分配。所以先用 ATSS 这种**不依赖预测分数的静态策略**，帮助模型建立初步的分类和定位能力。

**3. Loss function**

参与 Loss 计算的共有两个值：**cls_ loss** 和 **bbox_loss**

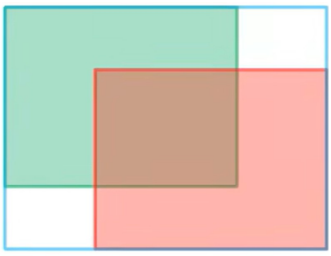
**Bbox_loss：**l/m/s 使用的是 **GIoU Loss**, t/n 用的是 **SIoU Loss**

**GIoU Loss**

- IOU 问题：当ground truth(GT)和prediction这两个bbox不相交时，loss为0
- GIOU 引入最小封闭矩形$A_c$

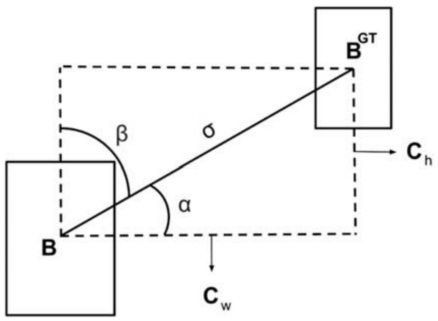
$$GIoU = IoU - \frac{A_c - U}{A_c}$$

$$L_{GIoU} = 1 - GIoU = 1 + \frac{A_c - U}{A_c} - IoU$$



**SIoU Loss** 主要由四个度量方面组成：IoU 成本 / 距离成本 / 形状成本 / 角度成本

如下图所示，**角度成本** 就是指图中预测框 $B$ 向 $B^{GT}$ 的回归过程中，尽可能去使得优化过程中的不确定性因素减少，比如现将图中的角度 $\alpha$ 或者 $\beta$ 变为 0，再去沿着 $x$ 轴或者 $y$ 轴去回归边界。



**cls_ loss 使用的是 VarifocalLoss**

Focal Loss 修改了传统的交叉熵损失，以解决正负样本或难易样本之间的类不平衡问题。VariFocal Loss 是 Focal Loss 的改进版，它通过考虑不同重要程度的正样本和负样本，平衡了来自两个样本的学习信号。

$$\mathrm{VFL}(p,q)=\begin{cases}-q(q\log(p)+(1-q)\log(1-p)) & q>0\\ -\alpha p^{\gamma}\log(1-p) & q=0,\end{cases} \tag{2}$$

where $p$ is the predicted IACS and $q$ is the target score. For a foreground point, $q$ for its ground-truth class is set as the IoU between the generated bounding box and its ground truth (gt_IoU) and 0 otherwise, whereas for a background point, the target $q$ for all classes is 0. See Figure 1.

q 是一个和 IOU 有关的软标签。对于挑选出的**正样本**（也就是和 GT 相交的样本）的分类标签分数不再是 0、1 两个极端，而是与 IOU 有关的标签 q，这样的**好处**在于：**比如 IOU 很大，标签就是 0.9，IOU 很小，标签就是 0.1。**

**没有 Obj_loss？**

由于额外的置信度预测头可能与 Aligned Head 有所冲突，经实验验证在不同大小的模型上也都有掉点，所以最后选择弃用 **Obj** 分支。

**4. Other Tricks**

**4.1 More training epochs** --400 epochs

**4.2 Gray border of images**

Gray border 指的是在图像四周 padding 了一圈灰色像素边框，这些灰边并不包含有用的图像信息，但它们可以帮助模型更好地检测图像边缘附近的目标。在训练或推理时，图像经过 letterbox 处理后，灰边可能被自动添加。

**有什么问题?**

不添加灰边，模型在图像边缘的检测性能下降；添加灰边虽然提升了边缘检测能力，但会显著降低推理速度

YOLOv6 采用了两种策略：

1. 训练时，最后几个 epoch 关闭 Mosaic 增强
2. 推理时，将带有灰色边框的图像直接调整为目标图像大小

这两种策略结合后，YOLOv6 能在 **不降低推理速度的情况下保持甚至提升检测性能**

**4.3 Self-distillation**

The teacher network is the student itself in FP32-precision

**教师模型**：是学生模型的预训练版本（通常是 FP32 精度），不需要额外构建大型模型；

**学生模型**：当前正在训练的模型（可能是量化模型或轻量模型）；

**蒸馏目标**：

分类任务：使用 KL 散度最小化教师和学生的类别预测分布差异

回归任务：借助 DFL（Distribution Focal Loss）对边界框位置进行蒸馏

**Reference**

*2209.02976
2101.03697
2107.08430

YOLO 系列梳理（九）初尝新鲜出炉的 YOLOv6 - CV 技术指南（公众号） - 博客园
YOLOv5、YOLOX、YOLOv6 的分析与比较 yolox 和 yolov5 哪个好-CSDN 博客
YOLOv6 原理和实现全解析 — MMYOLO 0.6.0 文档
【YOLO 系列】YOLOv6 论文超详细解读（翻译 + 学习笔记）-CSDN 博客

(39 封私信 / 25 条消息) RepVGG：极简架构，SOTA 性能，让 VGG 式模型再次伟大（CVPR-2021） - 知乎
【目标检测】YOLO 系列 Anchor 标签分配、边框回归（坐标预测）方式、LOSS 计算方式 yolov1 每个 grid 多少个 anchor-CSDN 博客
目标检测：一文读懂 FCOS (CVPR 2019) fcos 目标检测-CSDN 博客
美团提出的 YOLOV6 之损失计算 yolov6 损失为 0-CSDN 博客
【YOLO 改进】Efficient Decoupled Head 解耦合头的添加 解耦头-CSDN 博客