

YOLOv8 with Shuffle Attention and 4 Detect Heads

ABSTRACT

报告详细介绍了Shuffle Attention机制[1]，以及如何将该机制应用于现有的YOLOv8网络中。同时，YOLOv8模型在对较小的特征提取中存在一些弱势，因此，在YOLOv8现有的3个检测头的基础上增加第4个检测头，用于检测更小的特征。最后，报告在COCO128数据集上进行实验，对模型效果进行验证，并且实现了利用YOLOv8及其一系列模型来执行AimBot任务的接口。

1. RELATED WORK

Grouped Features. 通过分组来学习特征最早是为了将模型分布在更多的计算资源上(GPU)，后来许多网络和模型都指出：在组上进行卷积可以学习到更好的特征表示。

Attention Mechanisms. 注意力机制是当今备受关注且非常重要的机制，有无数的文章研究了各种注意力机制。注意力机制的应用通常可以强化更有效的特征，并且弱化其它无用的特征，从而提升模型对特征提取的能力。

2. SHUFFLE ATTENTION

Shuffle Attention [1]很好的融合了Channel Attention和Spatial Attention两种注意力机制，接下来简述Shuffle Attention的计算过程：

Feature Grouping. 对于给定的特征图 $X \in \mathbb{R}^{C \times H \times W}$ ， C , H , W 分别为通道数、特征图的高和宽。将其分为 G 组，即 $X = [X_1, \dots, X_G]$ ，其中 $X_k \in \mathbb{R}^{C/G \times H \times W}$ 。接下来，对于分好组的子特征，对其进行Shuffle Attention的计算。在计算前，还要将每组再分为两个分支，分别计算Channel Attention和Spatial Attention，即 $X_{k1}, X_{k2} \in \mathbb{R}^{C/2G \times H \times W}$ 。

Channel Attention. 对 X_{k1} 求其对应的Channel Attention。首先通过Global Averaging Pooling (GAP)来得到每个通道的统计量：

$$s = \mathcal{F}_{gp}(X_{k1}) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W X_{k1}(i, j)$$

接下来通过一个门机制来使其可以自适应的调整每个通道的权重：

$$X'_{k1} = \sigma(\mathcal{F}_c(s)) \cdot X_{k1} = \sigma(W_1 s + b_1) \cdot X_{k1}$$

其中 σ 是 sigmoid 激活函数， $W_1, b_1 \in \mathbb{R}^{C/2G \times 1 \times 1}$ 是可学习的参数， W_1, b_1 会广播到各个组中。

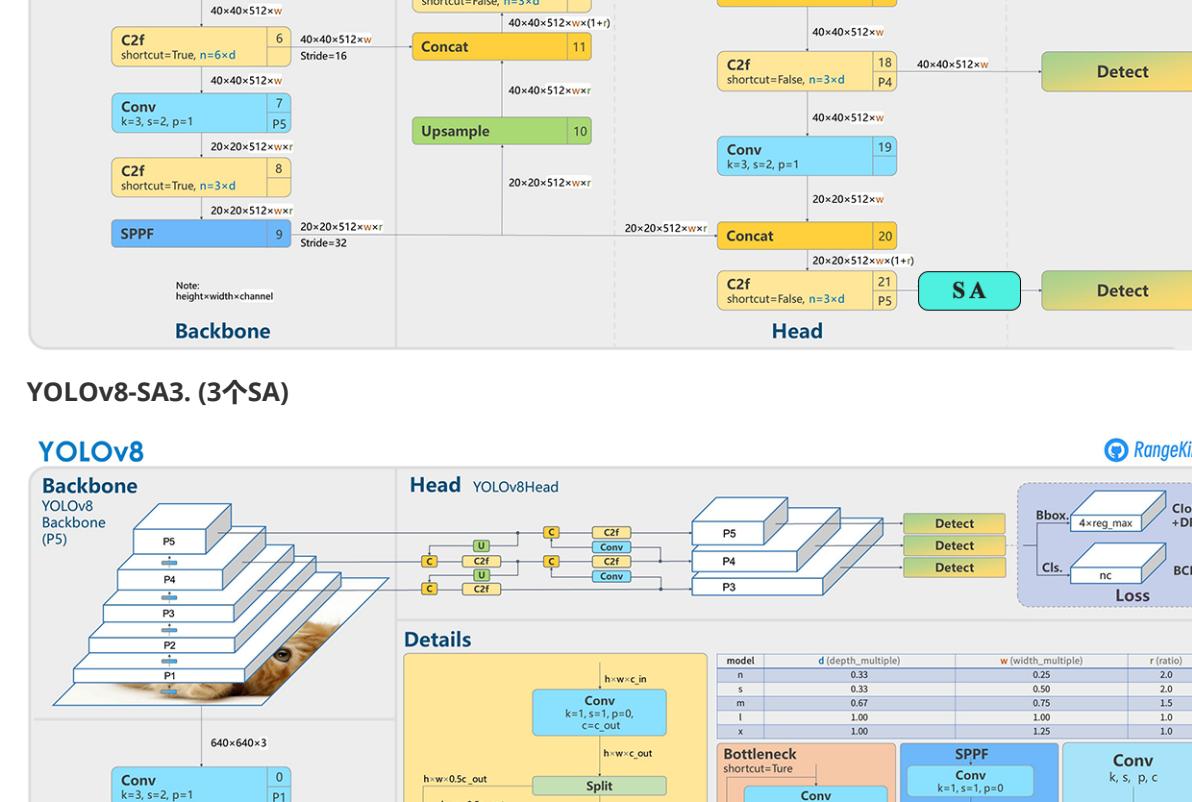
Spatial Attention. 空间注意力关注特征图中应该关心的部分在哪里，首先通过Group Norm(GN)[2]对 X_{k2} 进行空间独立采样，然后用线性函数加强这一样本，最后得到Spatial Attention的表示为：

$$X'_{k2} = \sigma(W_2 \cdot \text{GN}(X_{k2}) + b_2) \cdot X_{k2}$$

其中 σ 是 sigmoid 激活函数， $W_2, b_2 \in \mathbb{R}^{C/2G \times 1 \times 1}$ 是可学习的参数。

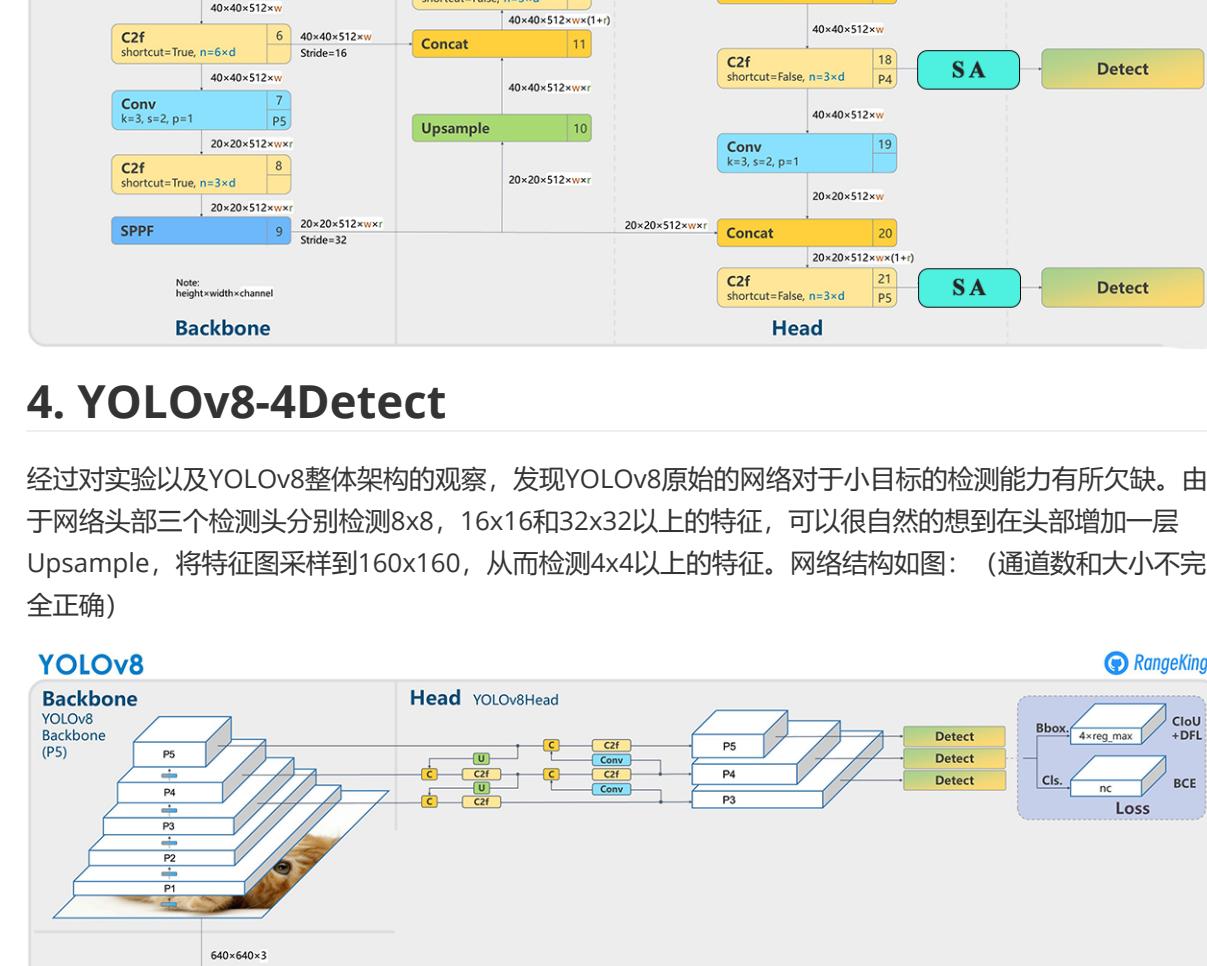
将 X'_{k1} 和 X'_{k2} 进行 Concat 得到该组的 X' ，然后对每个组进行 Channel Shuffle(将每个组的信息混合，从而实现组间共享信息)得到最后的输出。

Shuffle Attention:



Implementation.

参考[3]，对 SA 添加独立的残差连接：



SA模块很容易就可以通过Pytorch实现：

```
def shuffleAttention(x, channel_weight, channel_bias, spatial_weight, spatial_bias, groups):
    b, _, h, w = x.shape
    res = x.clone()

    x_grouped = x.reshape(b * groups, -1, h, w)
    x_0, x_1 = x_grouped.chunk(2, dim=1)

    x_channel = avg_pool(x_0)
    x_channel = channel_weight * x_channel + channel_bias
    x_channel = x_1 * sigmoid(x_channel)

    x_spatial = group_norm(x_1)
    x_spatial = spatial_weight * x_spatial + spatial_bias
    x_spatial = x_1 * sigmoid(x_spatial)

    out = torch.cat([x_channel, x_spatial], dim=1)
    out = out.reshape(b, -1, h, w)

    out = channel_shuffle(out, 2)
    out += res
    out = relu(out)
    return out
```

Shuffle Attention的参数量较小，且输入与输出形状相同，可以很方便的嵌入各种网络结构中。

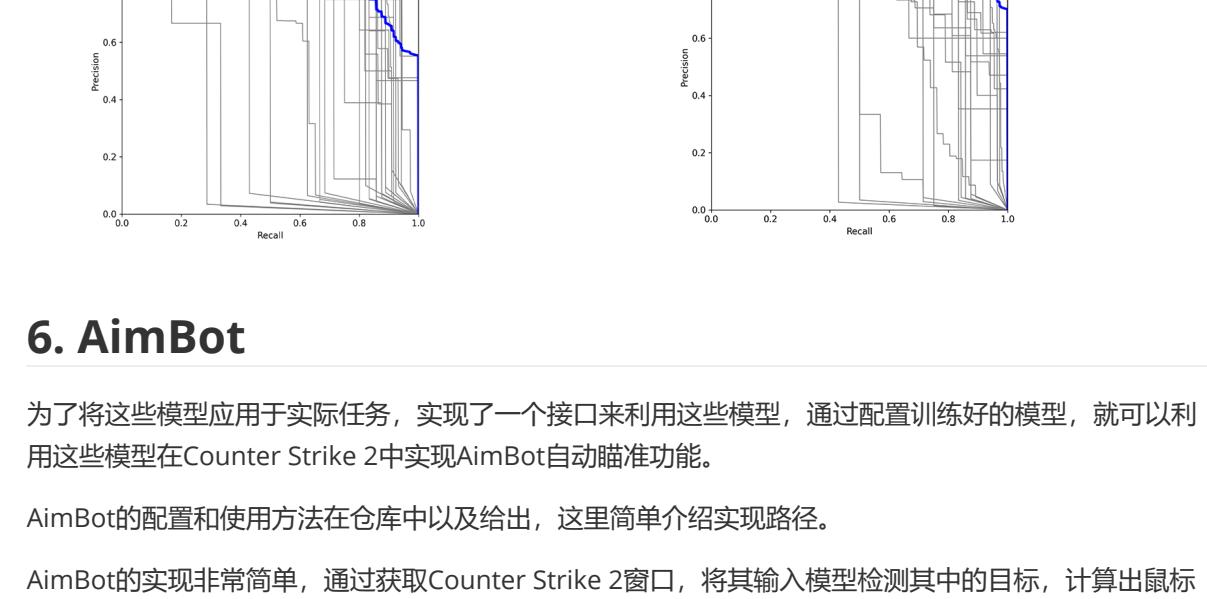
3. YOLOv8-SA

由YOLOv8的网络设计，可以在检测头前增加Shuffle Attention机制，以让模型可以更好的提取到特征图中的特征。

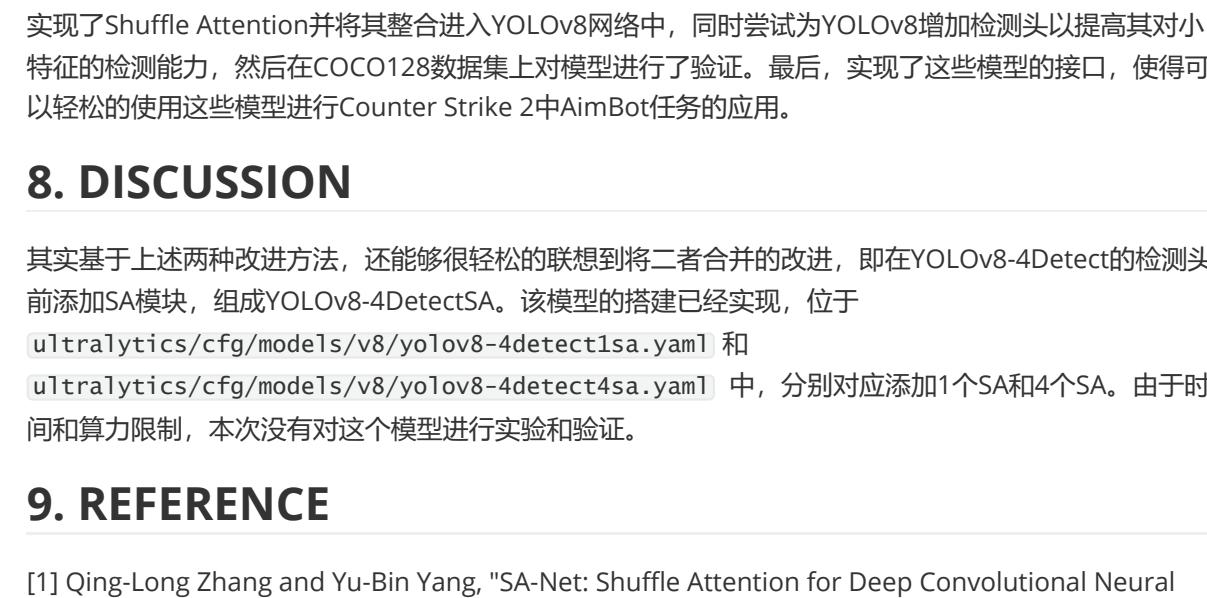
可以选择在部分检测头前增加，也可以只在一个头前增加，经过实验1个检测头前添加SA与3个检测头前添加SA并无明显差异。

以下为YOLOv8-SA结构图：

YOLOv8-SA1. (1个SA)



YOLOv8-SA3. (3个SA)



4. YOLOv8-4Detect

通过对实验以及YOLOv8整体架构的观察，发现YOLOv8原始的网络对于小目标的检测能力有所欠缺。由于网络头部三个检测头分别检测8x8, 16x16和32x32以上的特征，可以很自然的想到在头部增加一层Upsample，将特征图采样到160x160，从而检测4x4以上的特征。网络结构如图：(通道数和大小不完全正确)

通过增加该检测头，增强了模型对于小特征的识别能力，例如下列两张图，在4Detect和原始3检测头模型下，4Detect提取到了更小的特征：

5. EXPERIMENTS

Dataset. 数据集由于算力和时间限制，选择COCO128，其中有128张图片，每张图片中都有众多需要检测的目标，目标种类一共80个，可以很好的测试模型在特征检测和分类上的能力。

各个模型评估结果如下：

Model	Param.	GFLOPs	P	R	mAP50	mAP50-95
YOLOv8n	3.15M	8.7	0.89	0.8	0.844	0.701
YOLOv8n-SA1	3.15M	8.7	0.932	0.833	0.886	0.773
YOLOv8n-SA3	3.15M	8.7	0.928	0.843	0.887	0.785
YOLOv8n-4Detect	3.40M	17.5	0.935	0.877	0.934	0.805

可以看出，YOLOv8n-SA相比原模型，有一定的提升，而YOLOv8n-SA1和YOLOv8n-SA3在表现上并没有明显的差距。YOLOv8n-4Detect相比其他三个模型，在性能表现上都有优势，但是其参数量稍多于其他模型，需要的浮点运算次数增加了一倍。

PR-Curve:

YOLOv8-4 Detect:

6. AimBot

为了将这些模型应用于实际任务，实现了一个接口来利用这些模型，通过配置训练好的模型，就可以利用这些模型在Counter Strike 2中实现AimBot自动瞄准功能。

AimBot的配置和使用方法在仓库中已经给出，这里简单介绍实现路径。

AimBot的实现非常简单，通过获取Counter Strike 2窗口，将其输入模型检测其中的目标，计算出鼠标移动的距离，然后模拟鼠标移动，将准星瞄准到目标上。

7. CONCLUSION

实现了Shuffle Attention并将其整合进YOLOv8网络中，同时尝试为YOLOv8增加检测头以提高其对小特征的检测能力，然后在COCO128数据集上对模型进行了验证。最后，实现了这些模型的接口，使得可以轻松的使用这些模型进行Counter Strike 2中AimBot任务的应用。

8. DISCUSSION

其实基于上述两种改进方法，还能够很轻松的联想到将二者合并的改进，即在YOLOv8-4Detect的检测头前添加SA模块，组成YOLOv8-4Detect-SA。该模型的搭建已经实现，位于

- [ultralytics/cfg/models/v8/yolov8-4detect1sa.yaml](#)
- [ultralytics/cfg/models/v8/yolov8-4detect4sa.yaml](#)

中，分别对应添加1个SA和4个SA。由于时间和算力限制，本次没有对这个模型进行实验和验证。

9. REFERENCE

- [1] Qing-Long Zhang and Yu-Bin Yang, "SA-Net: Shuffle Attention for Deep Convolutional Neural Networks," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 2021, pp. 2235-2239, doi: 10.1109/ICASSP39728.2021.9414568.
- [2] Yuxin Wu and Kaiming He, "Group normalization," in Computer Vision- ECCV 2018- 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII, 2018, pp. 3-19.
- [3] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, 2018, pp. 7132-7141.