Experimental Results

Parameters:

P: Precision

R: Recall

Wsz: Weight size (MB)

Ablation Study:

model	RegNety	MSAM	WIoU	Map.5	Map.95	P	R	Wsz
Baseline				**89.92**	**62.81**	**89.01**	**85.37**	**5.22**
RegNety	✓			90.43	62.12	86.69	87.76	9.01
MSAM		✓		91.47	61.38	92.52	85.33	5.4
WIoU			✓	91.12	60.91	89.67	86.1	5.22
RegNety+MSAM+WIoU	✓	✓	✓	**93.4**	**66.66**	**91.66**	**88.75**	**9.07**

Comparative Experiments

model	Map.5	Map.95	P	R	Wsz
yolov11n	**89.92**	**62.81**	**89.01**	**85.37**	**5.22**
yolov10n	81.53	52.99	76.25	76.96	5.48
yolov8n	83.69	53.74	83.03	76.16	5.36
ssd	59.83	27.1	31.11	86.62	17.86
retinanet	51.55	21.4	57.96	25.03	76.07
rtdetr	81.79	53	81.75	72.63	38.53
ours	**93.4**	**66.66**	**91.66**	**88.75**	**9.07**

Summary of Improvements:

1. **RegNetY backbone**

   Core logic:

   1. RegNetY is a variant in the RegNet family that introduces the Squeeze-and-Excitation (SE) module.
   2. The SE module enhances representational capacity by adaptively recalibrating channel-wise feature responses.
   3. RegNetY uses quantized linear parameterization to design the network.
   4. Network width and depth are controlled via parameters such as w_a, w_0, and w_m.
   5. Group convolutions and bottleneck structures improve computational efficiency.
   6. It serves as an efficient backbone for object detectors such as YOLO.

2. **WIoU loss**

   Core logic:

   1. Initialize the WIoU_Scale class with IoU values.
   2. Automatically call the _update method to maintain a running mean (iou_mean).

    3. Choose different loss formulations based on the monotonous flag.
    4. Apply composite scaling based on the gamma and delta parameters.
    5. Return the scaled loss value.
3. **MSAM attention mechanism**
   Core logic:
    1. Built as an improvement over CBAM: replace the original channel attention with multi-scale convolutions, giving the channel attention multi-scale capability.
    2. The first half is replaced with an **MSCAAttention** module, which extracts features using multiple convolutions at different scales.
    3. The latter half still uses CBAM’s **spatialAttention** module.
    4. Element-wise multiply the outputs of (2) and (3) to obtain the final output.

Modification locations

1. ultralytics/nn/modules/block.py: import the various improved modules.
2. ultralytics/nn/modules/__init__.py: import the improved module classes and functions and add them to the package namespace.
3. ultralytics/nn/tasks.py: add functions in the parsing module to register the improved modules.
4. YAML files: add the corresponding modules and insert attention before the detect head.
5. ultralytics/utils/loss.py: add the loss function definition.
6. ultralytics/utils/loss.py: apply the loss function.