TEAM YOLOGANg, ICDEC'24 submission for Vehicle Detection in Adverse Weather Condtitions.

Dataset Format

The dataset is organized into training and validation sets with corresponding images and labels:

```
dataset/
 images/
    train/
    val/
 labels/
     train/
     val/
```

Ensure your dataset follows this structure for consistency.

Novelty of the Method

After testing models from YOLOv8s, v8n, v10s, v10m, YOLOv10m performed best. Optuna was used for hyperparameter tuning. Initial SDG optimizer results were poor compared to Adam optimizer. Best scores: 0.543 mAP@0.5 IoU and 0.273 mAP@0.5:0.95 IoU. Augmentations led to overfitting; only 'randomaug', 'mosaic', and 'fliplr' were used. Added Squeeze and Excitation (SE) blocks to each C2f Convolutional block, slightly improving performance (0.267 to 0.273 mAP@0.5:0.95). With Optuna's hyperparameters, the pretrained yolov10m.pt model achieved mentioned metrics in 5-10 epochs, taking 10-12 minutes on a Colab T4 GPU. Model size: 31.9 MB. Optuna's persistent memory storage ensures future fine-tuning.

Observations

Overlaying single-vehicle images on backgrounds reduced model usability due to insufficient image variety. Using CycleGAN for style-transfer (day to night, night to day) also led to overfitting.

Changes to Architecture

1. Created `se_block.py` in `ultralytics/nn/modules/`, implementing SEBlock class.
2. Modified C2f class in `ultralytics/nn/modules/block.py` to incorporate SEBlock:

```
class C2f(nn.Module):
    def __init__(self, c1, c2, n=1, shortcut=False, g=1, e=0.5):
        super().__init__()
        self.c = int(c2 * e)  # hidden channels
        self.cv1 = Conv(c1, 2 * self.c, 1, 1)
        self.cv2 = Conv((2 + n) * self.c, c2, 1)  # optional act=FReLU(c2)
        self.m = nn.ModuleList(Bottleneck(self.c, self.c, shortcut, g, k=((3, 3), (3
        self.se = SEBlock(c2)  # Add SE block here

    def forward(self, x):
        y = list(self.cv1(x).chunk(2, 1))
        y.extend(m(y[-1]) for m in self.m)
        return self.se(self.cv2(torch.cat(y, 1)))  # Apply SE block to the output

    def forward_split(self, x):
        y = list(self.cv1(x).split((self.c, self.c), 1))
        y.extend(m(y[-1]) for m in self.m)
        return self.se(self.cv2(torch.cat(y, 1)))  # Apply SE block to the output
```

3. Reinstalled modified Ultralytics package using `pip install -e .` in the package directory.

Validation

Use the validation scripts provided by Ultralytics via CLI or Python code.

Scores

F1 Confidence (all classes): 0.49 @ 0.086 Confidence.
mAP@0.5 IoU: 0.54365
mAP@0.5:0.95 IoU: 0.27373
Precision: 0.53035
Recall: 0.49303
Other results can be checked in either train or val folder of the github