In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [16]:
```
!pip install pandas
!pip install pycocotools
!pip install opencv-python
!pip install albumentations
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.
2.2)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packa
ges (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/d
ist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packag
es (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-pack
ages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages
(from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: pycocotools in /usr/local/lib/python3.11/dist-package
s (2.0.8)
Requirement already satisfied: matplotlib>=2.1.0 in /usr/local/lib/python3.11/dist-p
ackages (from pycocotools) (3.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (fro
m pycocotools) (2.0.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pa
ckages (from matplotlib>=2.1.0->pycocotools) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packag
es (from matplotlib>=2.1.0->pycocotools) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-p
ackages (from matplotlib>=2.1.0->pycocotools) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-p
ackages (from matplotlib>=2.1.0->pycocotools) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pac
kages (from matplotlib>=2.1.0->pycocotools) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages
(from matplotlib>=2.1.0->pycocotools) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-pa
ckages (from matplotlib>=2.1.0->pycocotools) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dis
t-packages (from matplotlib>=2.1.0->pycocotools) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages
(from python-dateutil>=2.7->matplotlib>=2.1.0->pycocotools) (1.17.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packa
ges (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-packa
ges (from opencv-python) (2.0.2)
Requirement already satisfied: albumentations in /usr/local/lib/python3.11/dist-pack
ages (2.0.6)
Requirement already satisfied: numpy>=1.24.4 in /usr/local/lib/python3.11/dist-packa
ges (from albumentations) (2.0.2)
Requirement already satisfied: scipy>=1.10.0 in /usr/local/lib/python3.11/dist-packa
ges (from albumentations) (1.15.2)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.11/dist-packages (fr
om albumentations) (6.0.2)
Requirement already satisfied: pydantic>=2.9.2 in /usr/local/lib/python3.11/dist-pac
kages (from albumentations) (2.11.4)
Requirement already satisfied: albucore==0.0.24 in /usr/local/lib/python3.11/dist-pa
ckages (from albumentations) (0.0.24)
Requirement already satisfied: opencv-python-headless>=4.9.0.80 in /usr/local/lib/py
thon3.11/dist-packages (from albumentations) (4.11.0.86)
Requirement already satisfied: stringzilla>=3.10.4 in /usr/local/lib/python3.11/dist
-packages (from albucore==0.0.24->albumentations) (3.12.5)
```

```
Requirement already satisfied: simsimd>=5.9.2 in /usr/local/lib/python3.11/dist-pack
ages (from albucore==0.0.24->albumentations) (6.2.1)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/d
ist-packages (from pydantic>=2.9.2->albumentations) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/di
st-packages (from pydantic>=2.9.2->albumentations) (2.33.2)
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.1
1/dist-packages (from pydantic>=2.9.2->albumentations) (4.13.2)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.1
1/dist-packages (from pydantic>=2.9.2->albumentations) (0.4.0)
```

In [2]:
```python
import zipfile

zip_path = "/content/drive/MyDrive/P2-Dhaka_Dataset_COCO/P2_Dhaka_Dataset.v29i.coco
extract_path = "/content/drive/MyDrive/P2-Dhaka_Dataset_COCO/"

try:
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_path)
    print("Extraction successful!")
except zipfile.BadZipFile:
    print("Bad ZIP file: corrupted or incomplete.")
```

```
Extraction successful!
```

In [19]:
```python
import torchvision
from torchvision.models.detection import retinanet_resnet50_fpn_v2

model = retinanet_resnet50_fpn_v2(weights=None,weights_backbone="DEFAULT", num_clas
model.load_state_dict(torch.load('retinanet_weights.pth'))
model.train()
model.to('cuda')
```

Out[19]:
```
RetinaNet(
  (backbone): BackboneWithFPN(
    (body): IntermediateLayerGetter(
      (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bi
as=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_s
tats=True)
      (relu): ReLU(inplace=True)
      (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mo
de=False)
      (layer1): Sequential(
        (0): Bottleneck(
          (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
          (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (relu): ReLU(inplace=True)
          (downsample): Sequential(
            (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          )
        )
        (1): Bottleneck(
          (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
          (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (relu): ReLU(inplace=True)
        )
        (2): Bottleneck(
          (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runni
ng_stats=True)
          (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (relu): ReLU(inplace=True)
        )
      )
```

```
      (layer2): Sequential(
        (0): Bottleneck(
          (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (relu): ReLU(inplace=True)
          (downsample): Sequential(
            (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
            (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          )
        )
        (1): Bottleneck(
          (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (relu): ReLU(inplace=True)
        )
        (2): Bottleneck(
          (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (relu): ReLU(inplace=True)
        )
        (3): Bottleneck(
          (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
```

```
          (relu): ReLU(inplace=True)
        )
      )
      (layer3): Sequential(
        (0): Bottleneck(
          (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
          (downsample): Sequential(
            (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
            (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          )
        )
        (1): Bottleneck(
          (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
        )
        (2): Bottleneck(
          (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
        )
        (3): Bottleneck(
          (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
```

```
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
        )
        (4): Bottleneck(
          (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
        )
        (5): Bottleneck(
          (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
        )
      )
      (layer4): Sequential(
        (0): Bottleneck(
          (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_run
```

```
ning_stats=True)
          (relu): ReLU(inplace=True)
          (downsample): Sequential(
            (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
            (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          )
        )
        (1): Bottleneck(
          (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
        )
        (2): Bottleneck(
          (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
          (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
          (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=Fals
e)
          (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
          (relu): ReLU(inplace=True)
        )
      )
    )
    (fpn): FeaturePyramidNetwork(
      (inner_blocks): ModuleList(
        (0): Conv2dNormActivation(
          (0): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1))
        )
        (2): Conv2dNormActivation(
          (0): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1))
        )
      )
      (layer_blocks): ModuleList(
        (0-2): 3 x Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        )
```

```
      )
      (extra_blocks): LastLevelP6P7(
        (p6): Conv2d(2048, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        (p7): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      )
    )
  )
  (anchor_generator): AnchorGenerator()
  (head): RetinaNetHead(
    (classification_head): RetinaNetClassificationHead(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
        (2): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
        (3): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
      )
      (cls_logits): Conv2d(256, 81, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    )
    (regression_head): RetinaNetRegressionHead(
      (conv): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
        (2): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
```

```
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
        (3): Conv2dNormActivation(
          (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
          (1): GroupNorm(32, 256, eps=1e-05, affine=True)
          (2): ReLU(inplace=True)
        )
      )
      (bbox_reg): Conv2d(256, 36, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
    )
  )
  (transform): GeneralizedRCNNTransform(
      Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
      Resize(min_size=(800,), max_size=1333, mode='bilinear')
  )
)
```

In [20]:

```python
import torch
from torchvision.datasets import CocoDetection
from torchvision import transforms as T
import os

class CocoDetectionRetinaNet(CocoDetection):
    def __init__(self, root, annFile, transform=None):
        super(CocoDetectionRetinaNet, self).__init__(root, annFile)
        self.transform = transform

    def __getitem__(self, idx):
        img, ann = super().__getitem__(idx)

        boxes = []
        labels = []

        for obj in ann:
            if 'iscrowd' in obj and obj['iscrowd']:
                continue

            bbox = obj['bbox']
            x1 = bbox[0]
            y1 = bbox[1]
            x2 = bbox[0] + bbox[2]
            y2 = bbox[1] + bbox[3]

            boxes.append([x1, y1, x2, y2])
            labels.append(obj['category_id'])

        if len(boxes) == 0:
            boxes = torch.zeros((0, 4), dtype=torch.float32)
            labels = torch.zeros((0,), dtype=torch.int64)
        else:
            boxes = torch.tensor(boxes, dtype=torch.float32)
            labels = torch.tensor(labels, dtype=torch.int64)
```

```
        target = {
            "boxes": boxes,
            "labels": labels
        }

        if self.transform is not None:
            img = self.transform(img)

        return img, target
```

In [4]:
```
!pip install pycocotools
```

```
Requirement already satisfied: pycocotools in e:\anaconda\envs\cpe313_cenv_backup\li
b\site-packages (2.0.8)
Requirement already satisfied: matplotlib>=2.1.0 in c:\users\ery\appdata\roaming\pyt
hon\python310\site-packages (from pycocotools) (3.5.0)
Requirement already satisfied: numpy in c:\users\ery\appdata\roaming\python\python31
0\site-packages (from pycocotools) (1.24.4)
Requirement already satisfied: cycler>=0.10 in e:\anaconda\envs\cpe313_cenv_backup\l
ib\site-packages (from matplotlib>=2.1.0->pycocotools) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in e:\anaconda\envs\cpe313_cenv_bac
kup\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (4.55.3)
Requirement already satisfied: kiwisolver>=1.0.1 in e:\anaconda\envs\cpe313_cenv_bac
kup\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\ery\appdata\roaming\pytho
n\python310\site-packages (from matplotlib>=2.1.0->pycocotools) (24.2)
Requirement already satisfied: pillow>=6.2.0 in e:\anaconda\envs\cpe313_cenv_backup
\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (11.1.0)
Requirement already satisfied: pyparsing>=2.2.1 in e:\anaconda\envs\cpe313_cenv_back
up\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in e:\anaconda\envs\cpe313_cenv_
backup\lib\site-packages (from matplotlib>=2.1.0->pycocotools) (2.9.0.post0)
Requirement already satisfied: setuptools-scm>=4 in c:\users\ery\appdata\roaming\pyt
hon\python310\site-packages (from matplotlib>=2.1.0->pycocotools) (8.2.0)
Requirement already satisfied: six>=1.5 in c:\users\ery\appdata\roaming\python\pytho
n310\site-packages (from python-dateutil>=2.7->matplotlib>=2.1.0->pycocotools) (1.1
7.0)
Requirement already satisfied: setuptools>=61 in c:\users\ery\appdata\roaming\python
\python310\site-packages (from setuptools-scm>=4->matplotlib>=2.1.0->pycocotools) (7
8.1.0)
Requirement already satisfied: tomli>=1 in e:\anaconda\envs\cpe313_cenv_backup\lib\s
ite-packages (from setuptools-scm>=4->matplotlib>=2.1.0->pycocotools) (2.0.1)
```

In [21]:
```
from torchvision.datasets import CocoDetection
from torchvision.transforms import ToTensor

train_ds = CocoDetectionRetinaNet(root='P2_Dhaka_COCO/train',
                          annFile='P2_Dhaka_COCO/annotations/train_annotations.coco.
                          transform=ToTensor())

test_ds = CocoDetectionRetinaNet(root='P2_Dhaka_COCO/test',
                         annFile='P2_Dhaka_COCO/annotations/test_annotations.coco.js
                         transform=ToTensor())

val_ds = CocoDetectionRetinaNet(root='P2_Dhaka_COCO/valid',
```

```
                        annFile='P2_Dhaka_COCO/annotations/val_annotations.coco.json
                        transform=ToTensor())
```

```
loading annotations into memory...
Done (t=0.34s)
creating index...
index created!
loading annotations into memory...
Done (t=0.03s)
creating index...
index created!
loading annotations into memory...
Done (t=0.03s)
creating index...
index created!
```

In [22]:
```
from torch.utils.data import DataLoader

train_dl = DataLoader(train_ds,
                      batch_size=5,
                      shuffle=True,
                      collate_fn=lambda batch: tuple(zip(*batch)))

test_dl = DataLoader(test_ds,
                     batch_size=5,
                     shuffle=False,
                     collate_fn=lambda batch: tuple(zip(*batch)))

val_dl = DataLoader(val_ds,
                    batch_size=8,
                    shuffle=False,
                    collate_fn=lambda batch: tuple(zip(*batch)))
```

In [23]:
```
from tqdm import tqdm
```

In [24]:
```
from torchvision.models.detection import retinanet_resnet50_fpn
import torch

optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
num_epochs = 10
curr_loss = 32.6018
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0

    for images, targets in tqdm(train_dl, desc=f"Epoch {epoch+1}/{num_epochs}", lea
        images = [img.to('cuda') for img in images]
        targets = [{k: v.to('cuda') for k, v in t.items()} for t in targets]

        if any(t['boxes'].numel() == 0 for t in targets):
            continue

        loss_dict = model(images, targets)
        losses = sum(loss for loss in loss_dict.values())

        optimizer.zero_grad()
```

```
        losses.backward()
        optimizer.step()

        running_loss += losses.item()
    for images, targets in tqdm(test_dl, desc=f"Epoch {epoch+1}/{num_epochs}", leav
        images = [img.to('cuda') for img in images]
        targets = [{k: v.to('cuda') for k, v in t.items()} for t in targets]

        if any(t['boxes'].numel() == 0 for t in targets):
            continue

        loss_dict = model(images, targets)
        losses = sum(loss for loss in loss_dict.values())

        optimizer.zero_grad()
        losses.backward()
        optimizer.step()

        running_loss += losses.item()
    print(f"Epoch {epoch+1}, Loss: {running_loss:.4f}")
    if running_loss < curr_loss:
        print(f"Loss decreased from {curr_loss} to {running_loss}")
        curr_loss = running_loss
        torch.save(model.state_dict(), "retinanet_weights.pth")
```

```
Epoch 1/10: 100%|██████████| 896/896 [17:04<00:00,  1.14s/it]
Epoch 1/10: 100%|██████████| 61/61 [01:06<00:00,  1.09s/it]
Epoch 1, Loss: 36.0527
Epoch 2/10: 100%|██████████| 896/896 [14:19<00:00,  1.04it/s]
Epoch 2/10: 100%|██████████| 61/61 [01:05<00:00,  1.07s/it]
Epoch 2, Loss: 35.5273
Epoch 3/10: 100%|██████████| 896/896 [13:49<00:00,  1.08it/s]
Epoch 3/10: 100%|██████████| 61/61 [00:58<00:00,  1.04it/s]
Epoch 3, Loss: 37.0433
Epoch 4/10: 100%|██████████| 896/896 [13:02<00:00,  1.15it/s]
Epoch 4/10: 100%|██████████| 61/61 [00:59<00:00,  1.02it/s]
Epoch 4, Loss: 32.5598
Loss decreased from 32.6018 to 32.55977524537593
Epoch 5/10: 100%|██████████| 896/896 [18:03<00:00,  1.21s/it]
Epoch 5/10: 100%|██████████| 61/61 [00:53<00:00,  1.13it/s]
Epoch 5, Loss: 32.3230
Loss decreased from 32.55977524537593 to 32.32299549691379
Epoch 6/10: 100%|██████████| 896/896 [20:01<00:00,  1.34s/it]
Epoch 6/10: 100%|██████████| 61/61 [00:53<00:00,  1.15it/s]
Epoch 6, Loss: 32.3955
Epoch 7/10: 100%|██████████| 896/896 [12:49<00:00,  1.16it/s]
Epoch 7/10: 100%|██████████| 61/61 [00:58<00:00,  1.04it/s]
Epoch 7, Loss: 33.5681
Epoch 8/10: 100%|██████████| 896/896 [13:49<00:00,  1.08it/s]
Epoch 8/10: 100%|██████████| 61/61 [01:10<00:00,  1.16s/it]
Epoch 8, Loss: 32.1155
Loss decreased from 32.32299549691379 to 32.11548292078078
Epoch 9/10: 100%|██████████| 896/896 [20:17<00:00,  1.36s/it]
Epoch 9/10: 100%|██████████| 61/61 [00:55<00:00,  1.10it/s]
```

```
Epoch 9, Loss: 32.3645
Epoch 10/10: 100%|██████████| 896/896 [17:19<00:00,  1.16s/it]
Epoch 10/10: 100%|██████████| 61/61 [01:07<00:00,  1.11s/it]
Epoch 10, Loss: 32.3073
```

In [26]:
```python
model.eval()
with torch.no_grad():
    for images, targets in val_dl:
        images = list(img.to('cuda') for img in images)
        outputs = model(images)
```

In [35]:
```python
import torch
from torchmetrics.detection.mean_ap import MeanAveragePrecision

# Init metric
metric = MeanAveragePrecision(iou_type="bbox", iou_thresholds=[0.5])  # mAP@0.5

model.eval()
metric.reset()

with torch.no_grad():
    for images, targets in val_dl:
        images = [img.to('cuda') for img in images]
        targets = [{k: v.to('cuda') for k, v in t.items()} for t in targets]

        outputs = model(images)

        # Format prediction and targets for torchmetrics
        preds = []
        for out in outputs:
            preds.append({
                'boxes': out['boxes'].cpu(),
                'scores': out['scores'].cpu(),
                'labels': out['labels'].cpu(),
            })

        gts = []
        for t in targets:
            gts.append({
                'boxes': t['boxes'].cpu(),
                'labels': t['labels'].cpu(),
            })

        metric.update(preds, gts)
results = metric.compute()
print("mAP@0.5:", results["map_50"].item())
print("mAP@0.5-0.95:", results["map"].item())
```

```
mAP@0.5: 0.8128464818000793
mAP@0.5-0.95: 0.8128464818000793
```

In [34]:
```python
total_params = sum(p.numel() for p in model.parameters())

num_layers = len(list(model.modules()))
```

```
print(f"Total parameters: {total_params}")
print(f"Number of layers: {num_layers}")
```

```
Total parameters: 36497845
Number of layers: 210
```

In [ ]: