The flower dataset was downloaded from the link

Out of the images, 42 daisy images and 28 dandelion images were chosen for training

7 daisy and 8 dandelion images were chosen for testing, due to time constraint and limited computational ability

Labelme was used to annotate the dataset and the annotation for each file was exported as a json file

In [1]:
```
#importing libraries
from detectron2.engine import DefaultTrainer
from detectron2.config import get_cfg
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()


import numpy as np
import os, json, cv2, random


from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.structures import BoxMode
import os
import numpy as np
import json
import random
import matplotlib.pyplot as plt
%matplotlib inline

from detectron2.structures import BoxMode
from detectron2.data import DatasetCatalog, MetadataCatalog
import torch
#assert torch.__version__.startswith("1.8")
import torchvision
import cv2
from detectron2.engine import DefaultTrainer
```

```
C:\Users\shash\anaconda3\envs\detect\lib\site-packages\tqdm\auto.py:22: TqdmWarning:
IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.rea
dthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```
##define classes
classes = ['daisy', 'dandelion']
```

In [3]:
```
##function for uploading a dataset to detectron2##
def get_data_dicts(directory, classes):
    dataset_dicts = []
    for filename in [file for file in os.listdir(directory) if file.endswith('.json')]
        json_file = os.path.join(directory, filename)
```

```
        with open(json_file) as f:
            img_anns = json.load(f)

        record = {}

        filename = os.path.join(directory, img_anns["imagePath"])

        record["file_name"] = filename
        record["height"] = 512
        record["width"] = 512

        annos = img_anns["shapes"]
        objs = []
        for anno in annos:
            px = [a[0] for a in anno['points']] # x coord
            py = [a[1] for a in anno['points']] # y-coord
            poly = [(x, y) for x, y in zip(px, py)] # poly for segmentation
            poly = [p for x in poly for p in x]

            obj = {
                "bbox": [np.min(px), np.min(py), np.max(px), np.max(py)],
                "bbox_mode": BoxMode.XYXY_ABS,
                "segmentation": [poly],
                "category_id": classes.index(anno['label']),
                "iscrowd": 0
            }
            objs.append(obj)
        record["annotations"] = objs
        dataset_dicts.append(record)
    return dataset_dicts
```

In [4]:
```
#uploading data
for d in ["train", "test"]:
    DatasetCatalog.register(
        "outlierz" + d,
        lambda d=d: get_data_dicts(d, classes)
    )
    MetadataCatalog.get("outlierz" + d).set(thing_classes=classes)
```

In [6]:
```
#saving metadata
microcontroller_metadata = MetadataCatalog.get("outlierztrain")
```

In [7]:
```
cfg = get_cfg()


cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_5
cfg.DATASETS.TRAIN = ("outlierztrain",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 0
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_
# Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025
# pick a good LR
cfg.SOLVER.MAX_ITER = 300
# 300 iterations seems good enough for this toy dataset; you will need to train longer
cfg.SOLVER.STEPS = []
# do not decay learning rate
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
```

```
# faster, and good enough for this toy dataset (default: 512)
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2
# only has one class (ballon). (see https://detectron2.readthedocs.io/tutorials/datase
# NOTE: this config means the number of classes, but a few popular unofficial tutorial

os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

```
[10/09 12:23:01 d2.engine.defaults]: Model:
GeneralizedRCNN(
  (backbone): FPN(
    (fpn_lateral2): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
 1))
    (fpn_lateral3): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
 1))
    (fpn_lateral4): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output4): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
 1))
    (fpn_lateral5): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output5): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
 1))
    (top_block): LastLevelMaxPool()
    (bottom_up): ResNet(
      (stem): BasicStem(
        (conv1): Conv2d(
          3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False
          (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
        )
      )
      (res2): Sequential(
        (0): BottleneckBlock(
          (shortcut): Conv2d(
            64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv1): Conv2d(
            64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv2): Conv2d(
            64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv3): Conv2d(
            64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
        )
        (1): BottleneckBlock(
          (conv1): Conv2d(
            256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv2): Conv2d(
            64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv3): Conv2d(
            64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
        )
        (2): BottleneckBlock(
          (conv1): Conv2d(
            256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
```

```
      )
      (conv2): Conv2d(
        64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
      )
      (conv3): Conv2d(
        64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
      )
    )
  )
  (res3): Sequential(
    (0): BottleneckBlock(
      (shortcut): Conv2d(
        256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False
        (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
      )
      (conv1): Conv2d(
        256, 128, kernel_size=(1, 1), stride=(2, 2), bias=False
        (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
      )
      (conv2): Conv2d(
        128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
      )
      (conv3): Conv2d(
        128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
      )
    )
    (1): BottleneckBlock(
      (conv1): Conv2d(
        512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
      )
      (conv2): Conv2d(
        128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
      )
      (conv3): Conv2d(
        128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
      )
    )
    (2): BottleneckBlock(
      (conv1): Conv2d(
        512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
      )
      (conv2): Conv2d(
        128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
      )
      (conv3): Conv2d(
        128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
      )
    )
    (3): BottleneckBlock(
      (conv1): Conv2d(
```

```
      512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
    )
    (conv2): Conv2d(
      128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
    )
    (conv3): Conv2d(
      128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
    )
  )
)
(res4): Sequential(
  (0): BottleneckBlock(
    (shortcut): Conv2d(
      512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False
      (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
    )
    (conv1): Conv2d(
      512, 256, kernel_size=(1, 1), stride=(2, 2), bias=False
      (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
    )
    (conv2): Conv2d(
      256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
    )
    (conv3): Conv2d(
      256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
    )
  )
  (1): BottleneckBlock(
    (conv1): Conv2d(
      1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
    )
    (conv2): Conv2d(
      256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
    )
    (conv3): Conv2d(
      256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
    )
  )
  (2): BottleneckBlock(
    (conv1): Conv2d(
      1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
    )
    (conv2): Conv2d(
      256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
    )
    (conv3): Conv2d(
      256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
      (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
    )
  )
```

```
    (3): BottleneckBlock(
      (conv1): Conv2d(
        1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
      )
      (conv2): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
      )
      (conv3): Conv2d(
        256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
      )
    )
    (4): BottleneckBlock(
      (conv1): Conv2d(
        1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
      )
      (conv2): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
      )
      (conv3): Conv2d(
        256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
      )
    )
    (5): BottleneckBlock(
      (conv1): Conv2d(
        1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
      )
      (conv2): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
      )
      (conv3): Conv2d(
        256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
      )
    )
  )
  (res5): Sequential(
    (0): BottleneckBlock(
      (shortcut): Conv2d(
        1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False
        (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
      )
      (conv1): Conv2d(
        1024, 512, kernel_size=(1, 1), stride=(2, 2), bias=False
        (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
      )
      (conv2): Conv2d(
        512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
      )
      (conv3): Conv2d(
        512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
        (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
```

```
          )
        )
        (1): BottleneckBlock(
          (conv1): Conv2d(
            2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv2): Conv2d(
            512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv3): Conv2d(
            512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
          )
        )
        (2): BottleneckBlock(
          (conv1): Conv2d(
            2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv2): Conv2d(
            512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv3): Conv2d(
            512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
          )
        )
      )
    )
  )
  (proposal_generator): RPN(
    (rpn_head): StandardRPNHead(
      (conv): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (objectness_logits): Conv2d(256, 3, kernel_size=(1, 1), stride=(1, 1))
      (anchor_deltas): Conv2d(256, 12, kernel_size=(1, 1), stride=(1, 1))
    )
    (anchor_generator): DefaultAnchorGenerator(
      (cell_anchors): BufferList()
    )
  )
  (roi_heads): StandardROIHeads(
    (box_pooler): ROIPooler(
      (level_poolers): ModuleList(
        (0): ROIAlign(output_size=(7, 7), spatial_scale=0.25, sampling_ratio=0, align
ed=True)
        (1): ROIAlign(output_size=(7, 7), spatial_scale=0.125, sampling_ratio=0, alig
ned=True)
        (2): ROIAlign(output_size=(7, 7), spatial_scale=0.0625, sampling_ratio=0, ali
gned=True)
        (3): ROIAlign(output_size=(7, 7), spatial_scale=0.03125, sampling_ratio=0, al
igned=True)
      )
    )
    (box_head): FastRCNNConvFCHead(
```

```
      (flatten): Flatten(start_dim=1, end_dim=-1)
      (fc1): Linear(in_features=12544, out_features=1024, bias=True)
      (fc_relu1): ReLU()
      (fc2): Linear(in_features=1024, out_features=1024, bias=True)
      (fc_relu2): ReLU()
    )
    (box_predictor): FastRCNNOutputLayers(
      (cls_score): Linear(in_features=1024, out_features=3, bias=True)
      (bbox_pred): Linear(in_features=1024, out_features=8, bias=True)
    )
    (mask_pooler): ROIPooler(
      (level_poolers): ModuleList(
        (0): ROIAlign(output_size=(14, 14), spatial_scale=0.25, sampling_ratio=0, ali
gned=True)
        (1): ROIAlign(output_size=(14, 14), spatial_scale=0.125, sampling_ratio=0, al
igned=True)
        (2): ROIAlign(output_size=(14, 14), spatial_scale=0.0625, sampling_ratio=0, a
ligned=True)
        (3): ROIAlign(output_size=(14, 14), spatial_scale=0.03125, sampling_ratio=0,
 aligned=True)
      )
    )
    (mask_head): MaskRCNNConvUpsampleHead(
      (mask_fcn1): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (mask_fcn2): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (mask_fcn3): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (mask_fcn4): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (deconv): ConvTranspose2d(256, 256, kernel_size=(2, 2), stride=(2, 2))
      (deconv_relu): ReLU()
      (predictor): Conv2d(256, 2, kernel_size=(1, 1), stride=(1, 1))
    )
  )
)
[10/09 12:23:01 d2.data.build]: Removed 0 images with no usable annotations. 69 image
s left.
[10/09 12:23:01 d2.data.build]: Distribution of instances among all 2 categories:
|  category  | #instances   |  category  | #instances   |
|:----------:|:-------------|:----------:|:-------------|
|   daisy    | 52           | dandelion  | 29           |
|            |              |            |              |
|   total    | 81           |            |              |
[10/09 12:23:01 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in traini
ng: [ResizeShortestEdge(short_edge_length=(640, 672, 704, 736, 768, 800), max_size=13
33, sample_style='choice'), RandomFlip()]
[10/09 12:23:01 d2.data.build]: Using training sampler TrainingSampler
[10/09 12:23:01 d2.data.common]: Serializing 69 elements to byte tensors and concaten
ating them all ...
[10/09 12:23:01 d2.data.common]: Serialized dataset takes 0.04 MiB
```

```
Skip loading parameter 'roi_heads.box_predictor.cls_score.weight' to the model due to
incompatible shapes: (81, 1024) in the checkpoint but (3, 1024) in the model! You mig
ht want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.cls_score.bias' to the model due to i
ncompatible shapes: (81,) in the checkpoint but (3,) in the model! You might want to
double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.weight' to the model due to
incompatible shapes: (320, 1024) in the checkpoint but (8, 1024) in the model! You mi
ght want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.bias' to the model due to i
ncompatible shapes: (320,) in the checkpoint but (8,) in the model! You might want to
double check if this is expected.
Skip loading parameter 'roi_heads.mask_head.predictor.weight' to the model due to inc
ompatible shapes: (80, 256, 1, 1) in the checkpoint but (2, 256, 1, 1) in the model!
You might want to double check if this is expected.
Skip loading parameter 'roi_heads.mask_head.predictor.bias' to the model due to incom
patible shapes: (80,) in the checkpoint but (2,) in the model! You might want to doub
le check if this is expected.
Some model parameters or buffers are not found in the checkpoint:
roi_heads.box_predictor.bbox_pred.{bias, weight}
roi_heads.box_predictor.cls_score.{bias, weight}
roi_heads.mask_head.predictor.{bias, weight}
[10/09 12:23:01 d2.engine.train_loop]: Starting training from iteration 0
```

```
C:\Users\shash\anaconda3\envs\detect\lib\site-packages\torch\functional.py:568: UserW
arning: torch.meshgrid: in an upcoming release, it will be required to pass the index
ing argument. (Triggered internally at  C:\actions-runner\_work\pytorch\pytorch\build
er\windows\pytorch\aten\src\ATen\native\TensorShape.cpp:2228.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
```

```
[10/09 12:23:19 d2.utils.events]:  eta: 0:02:26  iter: 19  total_loss: 2.203  loss_cl
s: 1.016  loss_box_reg: 0.4728  loss_mask: 0.6943  loss_rpn_cls: 0.002908  loss_rpn_l
oc: 0.007563  time: 0.5924  data_time: 0.0491  lr: 1.6068e-05  max_mem: 1570M
[10/09 12:23:30 d2.utils.events]:  eta: 0:02:16  iter: 39  total_loss: 1.903  loss_cl
s: 0.7097  loss_box_reg: 0.4722  loss_mask: 0.6723  loss_rpn_cls: 0.003308  loss_rpn_
loc: 0.006597  time: 0.5775  data_time: 0.0496  lr: 3.2718e-05  max_mem: 1657M
[10/09 12:23:41 d2.utils.events]:  eta: 0:02:05  iter: 59  total_loss: 1.53  loss_cl
s: 0.4615  loss_box_reg: 0.4609  loss_mask: 0.6329  loss_rpn_cls: 0.005835  loss_rpn_
loc: 0.005716  time: 0.5588  data_time: 0.0382  lr: 4.9367e-05  max_mem: 1657M
[10/09 12:23:52 d2.utils.events]:  eta: 0:01:56  iter: 79  total_loss: 1.483  loss_cl
s: 0.3617  loss_box_reg: 0.5195  loss_mask: 0.5873  loss_rpn_cls: 0.003574  loss_rpn_
loc: 0.007116  time: 0.5649  data_time: 0.0462  lr: 6.6017e-05  max_mem: 1657M
[10/09 12:24:03 d2.utils.events]:  eta: 0:01:45  iter: 99  total_loss: 1.294  loss_cl
s: 0.3132  loss_box_reg: 0.498  loss_mask: 0.5155  loss_rpn_cls: 0.0007185  loss_rpn_
loc: 0.005508  time: 0.5585  data_time: 0.0418  lr: 8.2668e-05  max_mem: 1657M
[10/09 12:24:15 d2.utils.events]:  eta: 0:01:34  iter: 119  total_loss: 1.216  loss_c
ls: 0.2771  loss_box_reg: 0.4954  loss_mask: 0.4422  loss_rpn_cls: 0.003772  loss_rpn
_loc: 0.006196  time: 0.5628  data_time: 0.0499  lr: 9.9318e-05  max_mem: 1657M
[10/09 12:24:27 d2.utils.events]:  eta: 0:01:26  iter: 139  total_loss: 1.055  loss_c
ls: 0.2243  loss_box_reg: 0.43  loss_mask: 0.3976  loss_rpn_cls: 0.0006666  loss_rpn_
loc: 0.005985  time: 0.5700  data_time: 0.0666  lr: 0.00011597  max_mem: 1657M
[10/09 12:24:39 d2.utils.events]:  eta: 0:01:17  iter: 159  total_loss: 1.13  loss_cl
s: 0.2383  loss_box_reg: 0.4984  loss_mask: 0.3325  loss_rpn_cls: 0.0006258  loss_rpn
_loc: 0.005737  time: 0.5749  data_time: 0.0774  lr: 0.00013262  max_mem: 1657M
[10/09 12:24:54 d2.utils.events]:  eta: 0:01:07  iter: 179  total_loss: 1.006  loss_c
ls: 0.179  loss_box_reg: 0.4991  loss_mask: 0.3058  loss_rpn_cls: 0.0009264  loss_rpn
_loc: 0.006522  time: 0.5925  data_time: 0.1224  lr: 0.00014927  max_mem: 1657M
[10/09 12:25:10 d2.utils.events]:  eta: 0:00:58  iter: 199  total_loss: 0.9302  loss_
cls: 0.1742  loss_box_reg: 0.497  loss_mask: 0.263  loss_rpn_cls: 0.0005255  loss_rpn
_loc: 0.006551  time: 0.6140  data_time: 0.1411  lr: 0.00016592  max_mem: 1657M
[10/09 12:25:26 d2.utils.events]:  eta: 0:00:48  iter: 219  total_loss: 0.8432  loss_
cls: 0.1191  loss_box_reg: 0.4761  loss_mask: 0.2366  loss_rpn_cls: 0.0003727  loss_r
pn_loc: 0.006912  time: 0.6298  data_time: 0.1385  lr: 0.00018257  max_mem: 1657M
[10/09 12:25:42 d2.utils.events]:  eta: 0:00:37  iter: 239  total_loss: 0.7571  loss_
cls: 0.1308  loss_box_reg: 0.3765  loss_mask: 0.2066  loss_rpn_cls: 0.0007384  loss_r
pn_loc: 0.008643  time: 0.6459  data_time: 0.1455  lr: 0.00019922  max_mem: 1657M
[10/09 12:25:54 d2.utils.events]:  eta: 0:00:25  iter: 259  total_loss: 0.7593  loss_
cls: 0.1127  loss_box_reg: 0.3621  loss_mask: 0.1912  loss_rpn_cls: 0.0005529  loss_r
pn_loc: 0.008648  time: 0.6431  data_time: 0.0889  lr: 0.00021587  max_mem: 1657M
[10/09 12:26:03 d2.utils.events]:  eta: 0:00:12  iter: 279  total_loss: 0.5024  loss_
cls: 0.0802  loss_box_reg: 0.2557  loss_mask: 0.1607  loss_rpn_cls: 3.492e-05  loss_r
pn_loc: 0.006331  time: 0.6270  data_time: 0.0263  lr: 0.00023252  max_mem: 1657M
[10/09 12:26:12 d2.utils.events]:  eta: 0:00:00  iter: 299  total_loss: 0.5447  loss_
cls: 0.08526  loss_box_reg: 0.2551  loss_mask: 0.1501  loss_rpn_cls: 8.702e-05  loss_
rpn_loc: 0.006882  time: 0.6141  data_time: 0.0273  lr: 0.00024917  max_mem: 1657M
[10/09 12:26:12 d2.engine.hooks]: Overall training speed: 298 iterations in 0:03:03
 (0.6141 s / it)
[10/09 12:26:12 d2.engine.hooks]: Total training time: 0:03:03 (0:00:00 on hooks)
```

In [8]:

```
#testing the data
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
cfg.DATASETS.TEST = ("outlierstest", )
predictor = DefaultPredictor(cfg)
```

```
[10/09 12:26:19 d2.checkpoint.c2_model_loading]: Following weights matched with mode
l:
| Names in Model                                    | Names in Checkpoint
| Shapes                                            |
|:--------------------------------------------------|:---------------------------------
---------------------------------------------------------------------|:---------------
---------------------------------|
| backbone.bottom_up.res2.0.conv1.*                 | backbone.bottom_up.res2.0.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (64,) (64,) (6
4,) (64,) (64,64,1,1)             |
| backbone.bottom_up.res2.0.conv2.*                 | backbone.bottom_up.res2.0.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (64,) (64,) (6
4,) (64,) (64,64,3,3)             |
| backbone.bottom_up.res2.0.conv3.*                 | backbone.bottom_up.res2.0.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,64,1,1)        |
| backbone.bottom_up.res2.0.shortcut.*              | backbone.bottom_up.res2.0.shortcu
t.{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight} | (256,) (256,)
 (256,) (256,) (256,64,1,1)        |
| backbone.bottom_up.res2.1.conv1.*                 | backbone.bottom_up.res2.1.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (64,) (64,) (6
4,) (64,) (64,256,1,1)            |
| backbone.bottom_up.res2.1.conv2.*                 | backbone.bottom_up.res2.1.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (64,) (64,) (6
4,) (64,) (64,64,3,3)             |
| backbone.bottom_up.res2.1.conv3.*                 | backbone.bottom_up.res2.1.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,64,1,1)        |
| backbone.bottom_up.res2.2.conv1.*                 | backbone.bottom_up.res2.2.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (64,) (64,) (6
4,) (64,) (64,256,1,1)            |
| backbone.bottom_up.res2.2.conv2.*                 | backbone.bottom_up.res2.2.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (64,) (64,) (6
4,) (64,) (64,64,3,3)             |
| backbone.bottom_up.res2.2.conv3.*                 | backbone.bottom_up.res2.2.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,64,1,1)        |
| backbone.bottom_up.res3.0.conv1.*                 | backbone.bottom_up.res3.0.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
 (128,) (128,) (128,256,1,1)       |
| backbone.bottom_up.res3.0.conv2.*                 | backbone.bottom_up.res3.0.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
 (128,) (128,) (128,128,3,3)       |
| backbone.bottom_up.res3.0.conv3.*                 | backbone.bottom_up.res3.0.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,128,1,1)       |
| backbone.bottom_up.res3.0.shortcut.*              | backbone.bottom_up.res3.0.shortcu
t.{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight} | (512,) (512,)
 (512,) (512,) (512,256,1,1)       |
| backbone.bottom_up.res3.1.conv1.*                 | backbone.bottom_up.res3.1.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
 (128,) (128,) (128,512,1,1)       |
| backbone.bottom_up.res3.1.conv2.*                 | backbone.bottom_up.res3.1.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
 (128,) (128,) (128,128,3,3)       |
| backbone.bottom_up.res3.1.conv3.*                 | backbone.bottom_up.res3.1.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,128,1,1)       |
| backbone.bottom_up.res3.2.conv1.*                 | backbone.bottom_up.res3.2.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
```

```
 (128,) (128,) (128,512,1,1)       |
| backbone.bottom_up.res3.2.conv2.*               | backbone.bottom_up.res3.2.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
 (128,) (128,) (128,128,3,3)       |
| backbone.bottom_up.res3.2.conv3.*               | backbone.bottom_up.res3.2.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,128,1,1)       |
| backbone.bottom_up.res3.3.conv1.*               | backbone.bottom_up.res3.3.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
 (128,) (128,) (128,512,1,1)       |
| backbone.bottom_up.res3.3.conv2.*               | backbone.bottom_up.res3.3.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (128,) (128,)
 (128,) (128,) (128,128,3,3)       |
| backbone.bottom_up.res3.3.conv3.*               | backbone.bottom_up.res3.3.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,128,1,1)       |
| backbone.bottom_up.res4.0.conv1.*               | backbone.bottom_up.res4.0.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,512,1,1)       |
| backbone.bottom_up.res4.0.conv2.*               | backbone.bottom_up.res4.0.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,256,3,3)       |
| backbone.bottom_up.res4.0.conv3.*               | backbone.bottom_up.res4.0.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (1024,) (102
4,) (1024,) (1024,) (1024,256,1,1)  |
| backbone.bottom_up.res4.0.shortcut.*            | backbone.bottom_up.res4.0.shortcu
t.{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight} | (1024,) (1024,)
(1024,) (1024,) (1024,512,1,1)  |
| backbone.bottom_up.res4.1.conv1.*               | backbone.bottom_up.res4.1.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,1024,1,1)      |
| backbone.bottom_up.res4.1.conv2.*               | backbone.bottom_up.res4.1.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,256,3,3)       |
| backbone.bottom_up.res4.1.conv3.*               | backbone.bottom_up.res4.1.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (1024,) (102
4,) (1024,) (1024,) (1024,256,1,1)  |
| backbone.bottom_up.res4.2.conv1.*               | backbone.bottom_up.res4.2.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,1024,1,1)      |
| backbone.bottom_up.res4.2.conv2.*               | backbone.bottom_up.res4.2.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,256,3,3)       |
| backbone.bottom_up.res4.2.conv3.*               | backbone.bottom_up.res4.2.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (1024,) (102
4,) (1024,) (1024,) (1024,256,1,1)  |
| backbone.bottom_up.res4.3.conv1.*               | backbone.bottom_up.res4.3.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,1024,1,1)      |
| backbone.bottom_up.res4.3.conv2.*               | backbone.bottom_up.res4.3.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,256,3,3)       |
| backbone.bottom_up.res4.3.conv3.*               | backbone.bottom_up.res4.3.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (1024,) (102
4,) (1024,) (1024,) (1024,256,1,1)  |
| backbone.bottom_up.res4.4.conv1.*               | backbone.bottom_up.res4.4.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,1024,1,1)      |
| backbone.bottom_up.res4.4.conv2.*               | backbone.bottom_up.res4.4.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
```

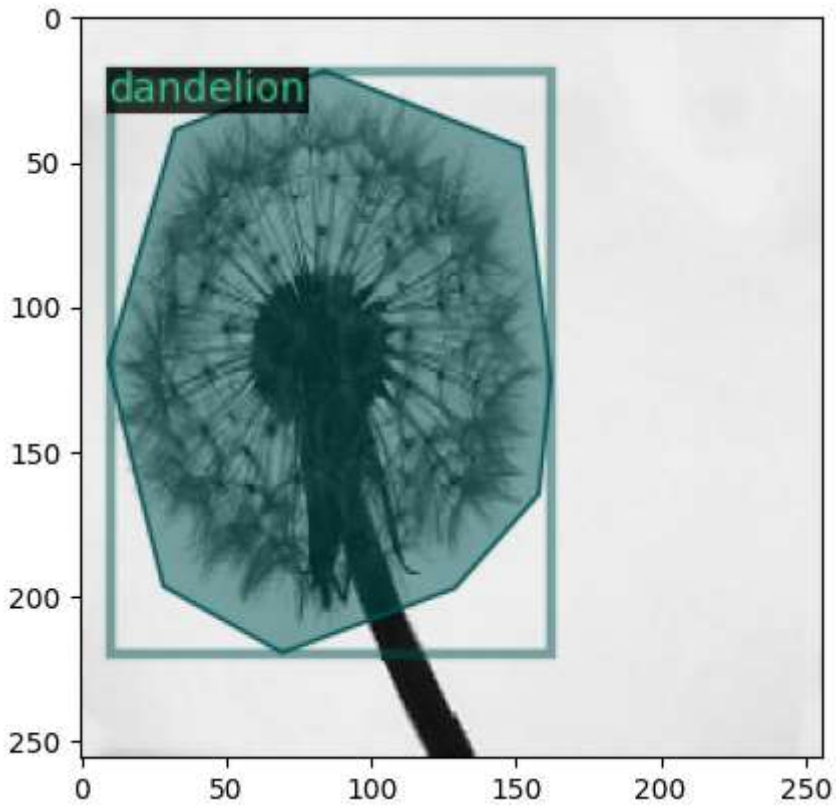
```
 (256,) (256,) (256,256,3,3)       |
| backbone.bottom_up.res4.4.conv3.*              | backbone.bottom_up.res4.4.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (1024,) (102
4,) (1024,) (1024,) (1024,256,1,1)  |
| backbone.bottom_up.res4.5.conv1.*              | backbone.bottom_up.res4.5.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,1024,1,1)      |
| backbone.bottom_up.res4.5.conv2.*              | backbone.bottom_up.res4.5.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (256,) (256,)
 (256,) (256,) (256,256,3,3)       |
| backbone.bottom_up.res4.5.conv3.*              | backbone.bottom_up.res4.5.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (1024,) (102
4,) (1024,) (1024,) (1024,256,1,1)  |
| backbone.bottom_up.res5.0.conv1.*              | backbone.bottom_up.res5.0.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,1024,1,1)      |
| backbone.bottom_up.res5.0.conv2.*              | backbone.bottom_up.res5.0.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,512,3,3)       |
| backbone.bottom_up.res5.0.conv3.*              | backbone.bottom_up.res5.0.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (2048,) (204
8,) (2048,) (2048,) (2048,512,1,1)  |
| backbone.bottom_up.res5.0.shortcut.*           | backbone.bottom_up.res5.0.shortcu
t.{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight} | (2048,) (2048,)
(2048,) (2048,) (2048,1024,1,1) |
| backbone.bottom_up.res5.1.conv1.*              | backbone.bottom_up.res5.1.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,2048,1,1)      |
| backbone.bottom_up.res5.1.conv2.*              | backbone.bottom_up.res5.1.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,512,3,3)       |
| backbone.bottom_up.res5.1.conv3.*              | backbone.bottom_up.res5.1.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (2048,) (204
8,) (2048,) (2048,) (2048,512,1,1)  |
| backbone.bottom_up.res5.2.conv1.*              | backbone.bottom_up.res5.2.conv1.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,2048,1,1)      |
| backbone.bottom_up.res5.2.conv2.*              | backbone.bottom_up.res5.2.conv2.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (512,) (512,)
 (512,) (512,) (512,512,3,3)       |
| backbone.bottom_up.res5.2.conv3.*              | backbone.bottom_up.res5.2.conv3.
{norm.bias,norm.running_mean,norm.running_var,norm.weight,weight}    | (2048,) (204
8,) (2048,) (2048,) (2048,512,1,1)  |
| backbone.bottom_up.stem.conv1.*                | backbone.bottom_up.stem.conv1.{no
rm.bias,norm.running_mean,norm.running_var,norm.weight,weight}      | (64,) (64,) (6
4,) (64,) (64,3,7,7)             |
| backbone.fpn_lateral2.*                        | backbone.fpn_lateral2.{bias,weigh
t}                                                                   | (256,) (256,25
6,1,1)                           |
| backbone.fpn_lateral3.*                        | backbone.fpn_lateral3.{bias,weigh
t}                                                                   | (256,) (256,51
2,1,1)                           |
| backbone.fpn_lateral4.*                        | backbone.fpn_lateral4.{bias,weigh
t}                                                                   | (256,) (256,102
4,1,1)                          |
| backbone.fpn_lateral5.*                        | backbone.fpn_lateral5.{bias,weigh
t}                                                                   | (256,) (256,204
8,1,1)                          |
| backbone.fpn_output2.*                         | backbone.fpn_output2.{bias,weigh
t}                                                                    | (256,) (256,25
```

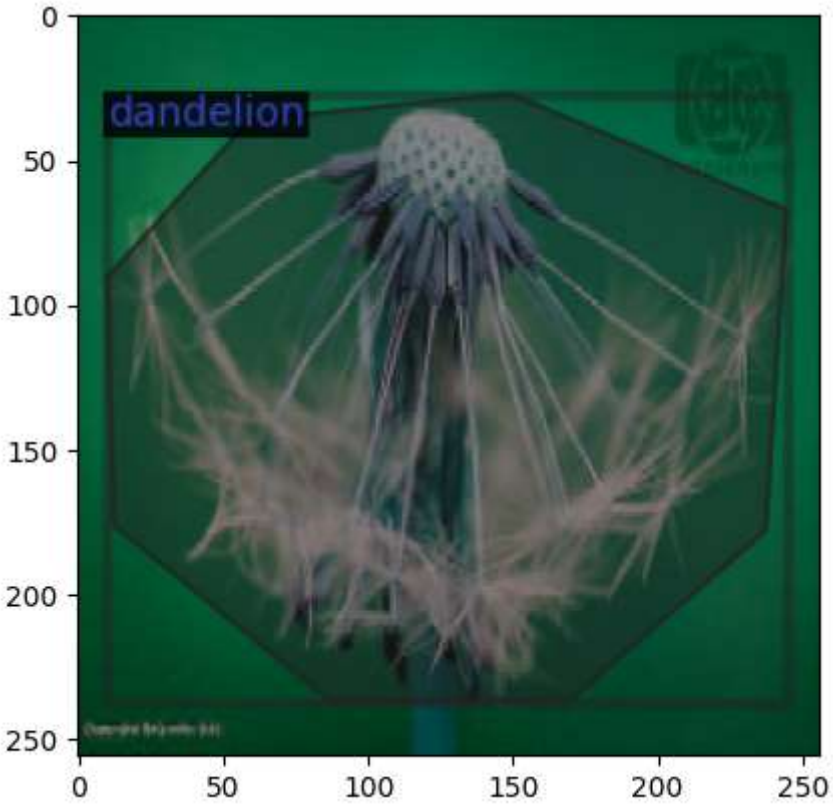
```
6,3,3)                                |
| backbone.fpn_output3.*                          | backbone.fpn_output3.{bias,weigh
t}                                                                | (256,) (256,25
6,3,3)                                |
| backbone.fpn_output4.*                          | backbone.fpn_output4.{bias,weigh
t}                                                                | (256,) (256,25
6,3,3)                                |
| backbone.fpn_output5.*                          | backbone.fpn_output5.{bias,weigh
t}                                                                | (256,) (256,25
6,3,3)                                |
| proposal_generator.rpn_head.anchor_deltas.*     | proposal_generator.rpn_head.ancho
r_deltas.{bias,weight}                                            | (12,) (12,256,
1,1)                                  |
| proposal_generator.rpn_head.conv.*              | proposal_generator.rpn_head.conv.
{bias,weight}                                                     | (256,) (256,25
6,3,3)                                |
| proposal_generator.rpn_head.objectness_logits.* | proposal_generator.rpn_head.objec
tness_logits.{bias,weight}                                        | (3,) (3,256,1,
1)                                    |
| roi_heads.box_head.fc1.*                        | roi_heads.box_head.fc1.{bias,weig
ht}                                                               | (1024,) (1024,1
2544)                               |
| roi_heads.box_head.fc2.*                        | roi_heads.box_head.fc2.{bias,weig
ht}                                                               | (1024,) (1024,1
024)                                  |
| roi_heads.box_predictor.bbox_pred.*             | roi_heads.box_predictor.bbox_pre
d.{bias,weight}                                                   | (8,) (8,1024)
|
| roi_heads.box_predictor.cls_score.*             | roi_heads.box_predictor.cls_scor
e.{bias,weight}                                                   | (3,) (3,1024)
|
| roi_heads.mask_head.deconv.*                    | roi_heads.mask_head.deconv.{bias,
weight}                                                           | (256,) (256,25
6,2,2)                                |
| roi_heads.mask_head.mask_fcn1.*                 | roi_heads.mask_head.mask_fcn1.{bi
as,weight}                                                        | (256,) (256,25
6,3,3)                                |
| roi_heads.mask_head.mask_fcn2.*                 | roi_heads.mask_head.mask_fcn2.{bi
as,weight}                                                        | (256,) (256,25
6,3,3)                                |
| roi_heads.mask_head.mask_fcn3.*                 | roi_heads.mask_head.mask_fcn3.{bi
as,weight}                                                        | (256,) (256,25
6,3,3)                                |
| roi_heads.mask_head.mask_fcn4.*                 | roi_heads.mask_head.mask_fcn4.{bi
as,weight}                                                        | (256,) (256,25
6,3,3)                                |
| roi_heads.mask_head.predictor.*                 | roi_heads.mask_head.predictor.{bi
as,weight}                                                        | (2,) (2,256,1,
1)                                    |
```

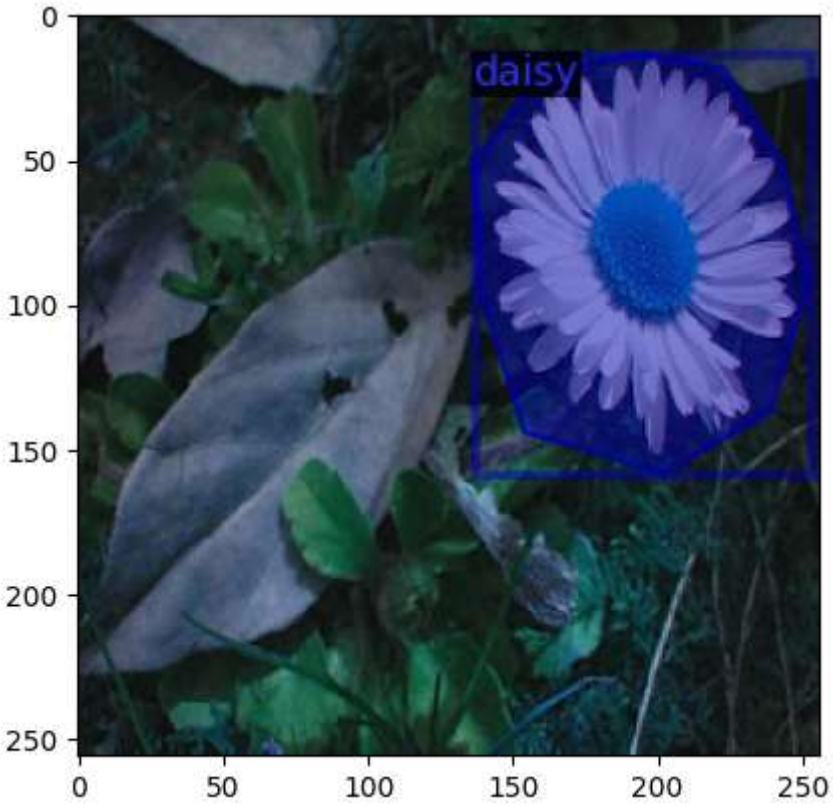
In [10]:
```
##for test data
test_dataset_dicts = get_data_dicts('test', classes)
```

In [11]:
```
##showing examples of the code on testset##
from matplotlib import pyplot as plt
import cv2

my_dataset_train_metadata = microcontroller_metadata
dataset_dicts = test_dataset_dicts
```

```
import random

from detectron2.utils.visualizer import Visualizer

for d in random.sample(dataset_dicts, 3):
    img = cv2.imread(d["file_name"])
    visualizer = Visualizer(img[:, :, ::-1], metadata=my_dataset_train_metadata, scale
    vis = visualizer.draw_dataset_dict(d)
    vis=vis.get_image()[:, :, ::-1]
    #print(vis)
    plt.imshow(vis)
    plt.show()
```




dandelion
0
50
100
150
200
250


daisy
0
50
100
150
200
250