Mask Region-Based Convolutional Neural Networks for the Detection and Measurement of Copper Chalcogenide Nanoparticles in Transmission Electron Microscope Images

Team 6

Authors: Colin Dingley, Alex Davis

Course: AI 570 - Deep Learning

Summer September, 2024

Project Background Information:

The goal of this project is to develop a Mask R-CNN model for analyzing TEM images of nanoparticles. The original images were obtained from Dr. Kate Plass at Franklin and Marshall College. The aim is to automate the detection and measurement of Chalcogenide Nanoparticles.

This code was generated after consulting the work of bnsreenu and their github repository: https://github.com/bnsreenu/python_for_microscopists/blob/master/330_Detectron2_Instance_3D_EM_Platelet.ipynb

Installation and Setup

Install Libraries and Mount Google Drive

All image and annotation files are stored on google drive. The following mounts the users google drive to CoLab to allow for access as though the files were local files.

```
import sys, os, distutils.core

# Install Detectron2
!git clone 'https://github.com/facebookresearch/detectron2'
dist = distutils.core.run_setup("./detectron2/setup.py")
!python -m pip install {' '.join([f"'{x}'" for x in dist.install_requires])}
sys.path.insert(0, os.path.abspath('./detectron2'))

# Install Tensorboard
%pip install tensorboard
%load_ext tensorboard

# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Setting Current Working Directory to Project Folder
prog_folder = "/content/drive/MyDrive/Colab_Notebooks/AI570_Project"
os.chdir(prog_folder)
print(os.getcwd())
```

```
Cloning into 'detectron2'...
remote: Enumerating objects: 15743, done.
remote: Total 15743 (delta 0), reused 0 (delta 0), pack-reused 15743
Receiving objects: 100% (15743/15743), 6.32 MiB | 10.47 MiB/s, done.
Resolving deltas: 100% (11492/11492), done.
Ignoring dataclasses: markers 'python_version < "3.7"' don't match your environment
Requirement already satisfied: Pillow>=7.1 in /usr/local/lib/python3.10/dist-packages (9.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: pycocotools>=2.0.2 in /usr/local/lib/python3.10/dist-packages (2.0.8)
Requirement already satisfied: termcolor>=1.1 in /usr/local/lib/python3.10/dist-packages (2.4.0)
Collecting yacs>=0.1.8
  Downloading yacs-0.1.8-py3-none-any.whl.metadata (639 bytes)
Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (0.9.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (2.2.1)
```

```
Requirement already satisfied: tqdm>4.29.0 in /usr/local/lib/python3.10/dist-packages (4.66.4)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.10/dist-packages (2.15.2)
Collecting fvcore<0.1.6,>=0.1.5
  Downloading fvcore-0.1.5.post20221221.tar.gz (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.2/50.2 kB 3.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting iopath<0.1.10,>=0.1.7
  Downloading iopath-0.1.9-py3-none-any.whl.metadata (370 bytes)
Collecting omegaconf<2.4,>=2.1
  Downloading omegaconf-2.3.0-py3-none-any.whl.metadata (3.9 kB)
Collecting hydra-core>=1.1
  Downloading hydra_core-1.3.2-py3-none-any.whl.metadata (5.5 kB)
Collecting black
  Downloading black-24.4.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.1/77.1 kB 5.0 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (24.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.26.4)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from yacs>=0.1.8) (6.0.1)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (1.4.0)
Requirement already satisfied: grpcio>=1.48.2 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (1.64.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorb
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (3.6)
Requirement already satisfied: protobuf!=4.24.0,>=3.19.6 in /usr/local/lib/python3.10/dist-packages (from tensorboar
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (2.
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (71.
Requirement already satisfied: six>1.9 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (1.16.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard) (3.0.3)
Collecting portalocker (from iopath<0.1.10,>=0.1.7)
  Downloading portalocker-2.10.1-py3-none-any.whl.metadata (8.5 kB)
Collecting antlr4-python3-runtime==4.9.* (from omegaconf<2.4,>=2.1)
  Downloading antlr4-python3-runtime-4.9.3.tar.gz (117 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.0/117.0 kB 11.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from black) (8.1.7)
```

```
Collecting mypy-extensions>=0.4.3 (from black)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
```

Importing libraries and functions

```
import torch, detectron2
import numpy as np
import os, json, cv2, random
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
import torchvision
import pandas as pd
from detectron2.utils.logger import setup_logger
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor, DefaultTrainer
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer, ColorMode
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.data import build_detection_train_loader, build_detection_test_loader
from detectron2.data.datasets import register_coco_instances
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.utils.events import TensorboardXWriter, EventStorage
from torch.utils.tensorboard import SummaryWriter

writer = SummaryWriter("./output/logs")
setup_logger()
```

```
<Logger detectron2 (DEBUG)>
```

Detectron2 Evaluation

Detectron2 was developed by facebook and was trained on a broad range of images. It should do well on detecting, segmenting

Detectron2 was developed by facebook and was trained on a broad range of images. It should do well on detecting, segmenting and identifying regular images. To demonstrate this, we will run a pre-trained model on the following image of a street. The image was selected based on a google image search for non-copytighted images of a street with people. We will use the pretrained model to evalueate its ability to analyse everyday images.

```
test_image = cv2.imread("./data/images/street_image.jpg")
scaled_image = cv2.resize(test_image, (0,0), fy=0.4, fx=0.4)
cv2_imshow(scaled_image)
```


```
def visualize_predictions(image_path, predictor, scale=0.7):
    """
    Visualize predictions on a random sample of images from the dataset.

    Args:
        image_path (str): Path to the image file.
        predictor: The Detectron2 predictor object.
        scale(float): Scale factor for the visualizer. Default is 0.7.
    """

    # Read and scale image
    image = cv2.imread(image_path)
    scaled_image = cv2.resize(image, (0,0), fy=scale, fx=scale)
```

```
    # Make predictions on scaled image
    predictions = predictor(scaled_image)

    # Draw the predictions on the image.
    labels = Visualizer(scaled_image[:, :, ::-1],
                        MetadataCatalog.get(cfg.DATASETS.TRAIN[0]))
                        #instance_mode=ColorMode.IMAGE_BW)
    labeled_image = labels.draw_instance_predictions(predictions["instances"].to("cpu"))

    # Combine original and labeled images side by side
    combined_image = np.hstack((scaled_image, labeled_image.get_image()[:, :, ::-1]))

    # Add titles
    title_height = 50
    title_image = np.zeros((title_height, combined_image.shape[1], 3), dtype=np.uint8)
    cv2.putText(title_image, "Original",
                (scaled_image.shape[1] // 2 - 40, title_height //2 + 10),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
    cv2.putText(title_image, "Labeled",
                (scaled_image.shape[1] + scaled_image.shape[1] // 2 - 40, title_height //2 + 10),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)

    combined_image_with_title = np.vstack((title_image, combined_image))

    # Display the combined image with titles
    cv2_imshow(combined_image_with_title)
```

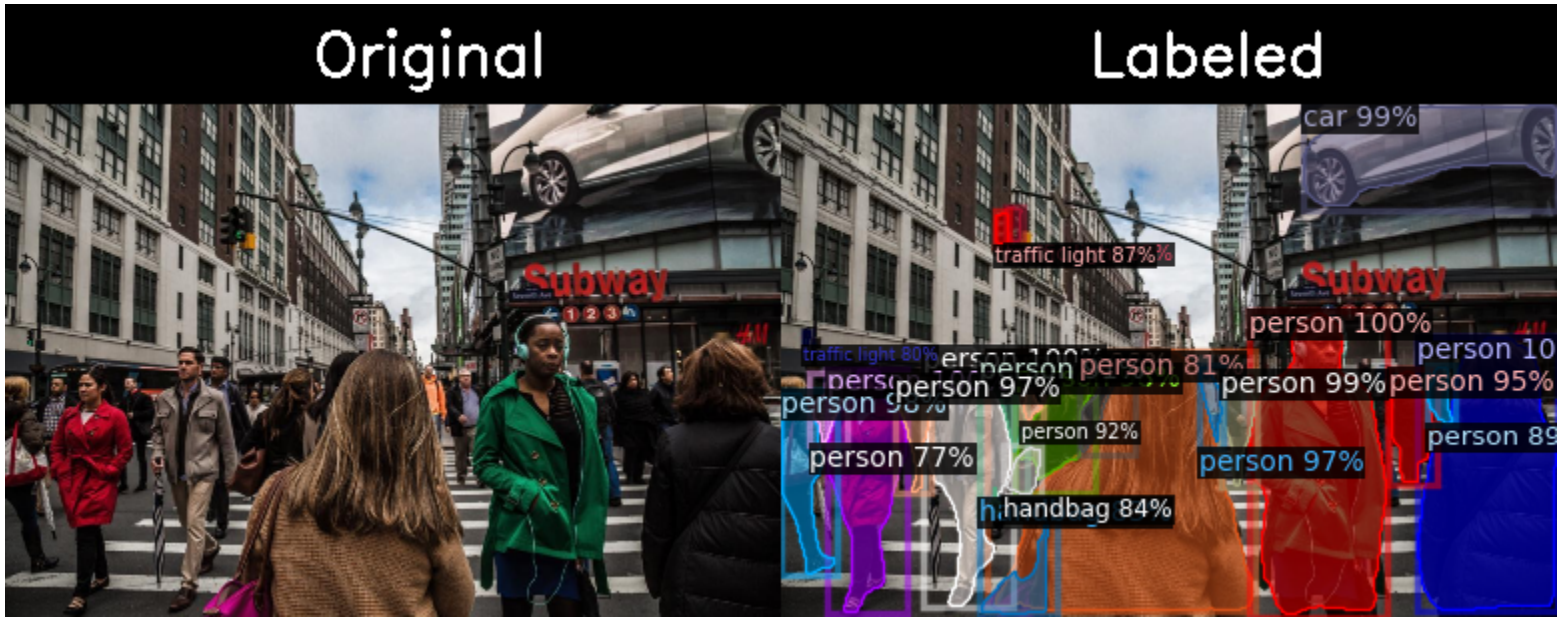
We will be using facebooks detectron2's "model zoo" model. Details of this model can be found here: # https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.6  # set threshold for this model
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
predictor = DefaultPredictor(cfg)
```

```
outputs = predictor(scaled_image)
```

```
[07/30 14:43:21 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from https://dl.fbaipublicfiles
model_final_f10217.pkl: 178MB [00:01, 90.3MB/s]
/usr/local/lib/python3.10/dist-packages/torch/functional.py:512: UserWarning: torch.meshgrid: in an upcoming release
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
```

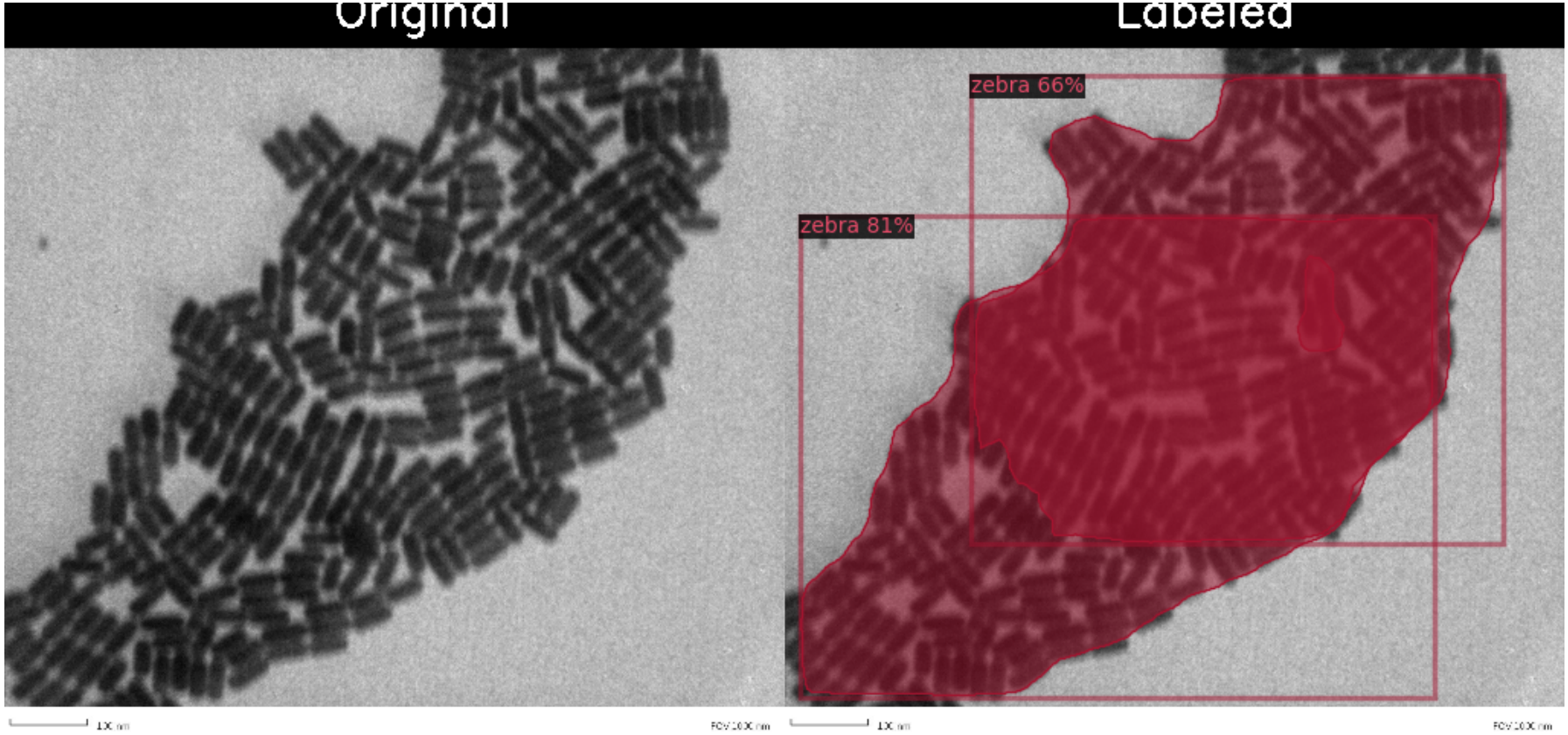
```
image_path = "./data/images/street_image.jpg"
visualize_predictions(image_path, predictor, scale=0.4)
```



Default Model on TEM Images

The model did very well on identifying everyday objects, which the model was trained on. However, the model was not trained on TEM images. Let's see how well it performs on one of our images:

```
image_path = "./data/images/all/AE-002_000004.jpg"
visualize_predictions(image_path, predictor, scale=0.4)
```



Since zebras cannot fit into a TEM the labels are clearly incorrect. We will therefore need to refit the last layer of the model using our own labeled images in order to take advantage of transfer learning.

Training on our labeled TEM images

```
def clear_dataset():
  """
```

```
    """
    Checks for the existance of the dataset and removes it if it exists.
    This is needed for rerunning the notebook, or if the dataset is altered.
    """

    if "my_dataset_train" in MetadataCatalog.list():
        MetadataCatalog.remove("my_dataset_train")
        DatasetCatalog.remove("my_dataset_train")

    if "my_dataset_val" in MetadataCatalog.list():
        MetadataCatalog.remove("my_dataset_val")
        DatasetCatalog.remove("my_dataset_val")

    if "my_dataset_test" in MetadataCatalog.list():
        MetadataCatalog.remove("my_dataset_test")
        DatasetCatalog.remove("my_dataset_test")
```

Loading and Regestering COCO Datasets

```
clear_dataset()
# Register datasets
register_coco_instances("my_dataset_train", {}, "./data/annotations/train_annotations.json", "./data/images/all/")
register_coco_instances("my_dataset_val", {}, "./data/annotations/val_annotations.json", "./data/images/all/")
register_coco_instances("my_dataset_test", {}, "./data/annotations/test_annotations.json", "./data/images/all/")

# Retrieve metadata and dictionaries for the training, testing and validation
# datasets from the DatasetCatelog
train_metadata = MetadataCatalog.get("my_dataset_train")
train_dataset_dicts = DatasetCatalog.get("my_dataset_train")
val_metadata = MetadataCatalog.get("my_dataset_val")
val_dataset_dicts = DatasetCatalog.get("my_dataset_val")
test_metadata = MetadataCatalog.get("my_dataset_test")
test_dataset_dicts = DatasetCatalog.get("my_dataset_test")
```

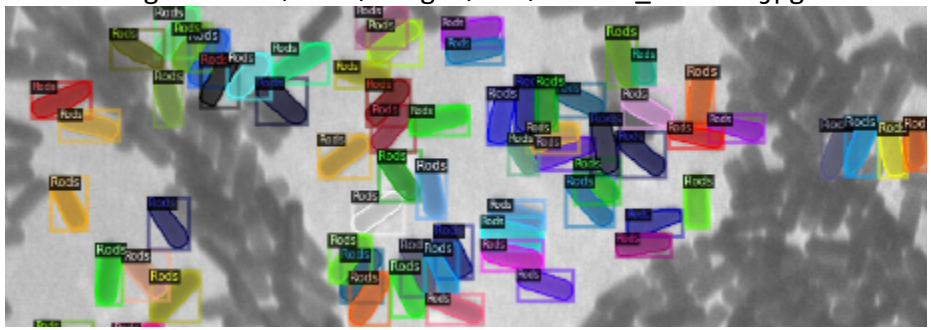
```
WARNING [07/30 14:44:03 d2.data.datasets.coco]:
Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

[07/30 14:44:03 d2.data.datasets.coco]: Loaded 74 images in COCO format from ./data/annotations/train_annotations.js
```

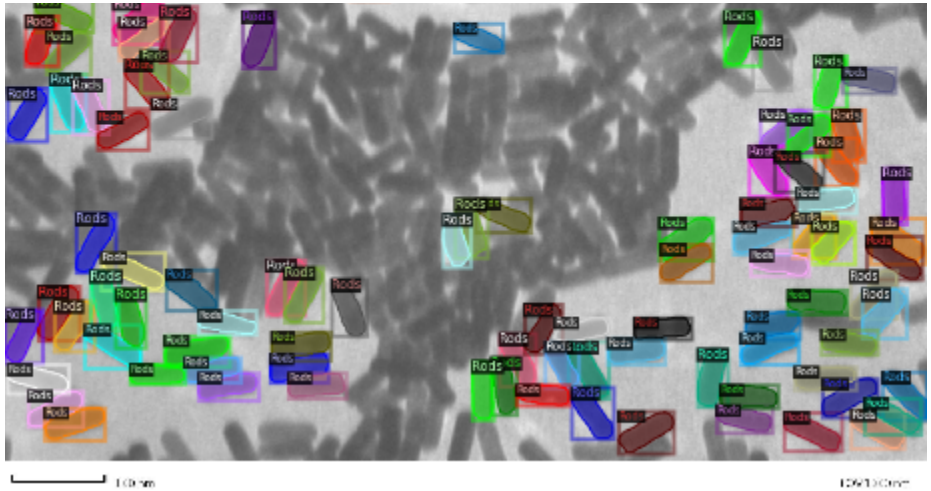
```
[07/30 14:44:03 d2.data.datasets.coco]: Loaded 74 images in COCO format from ./data/annotations/train_annotations.js
WARNING [07/30 14:44:04 d2.data.datasets.coco]:
Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

[07/30 14:44:04 d2.data.datasets.coco]: Loaded 21 images in COCO format from ./data/annotations/val_annotations.json
WARNING [07/30 14:44:04 d2.data.datasets.coco]:
Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

[07/30 14:44:04 d2.data.datasets.coco]: Loaded 11 images in COCO format from ./data/annotations/test_annotations.jso
```

Visualize Some Random Samples

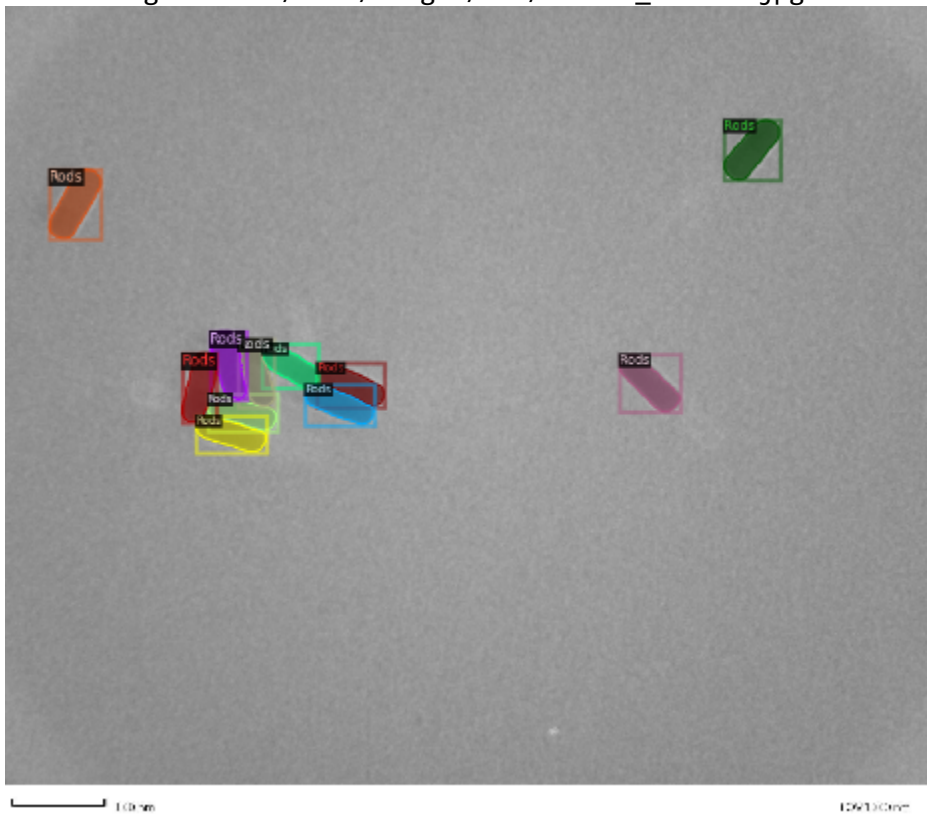
```
# Visualize some random samples
for d in random.sample(train_dataset_dicts, 2):
    print(f"Processing file: {d['file_name']}")
    image = cv2.imread(d["file_name"])
    if image is None:
        print(f"Error loading image {d['file_name']}")
        continue

    visualizer = Visualizer(image[:, :, ::-1], metadata=train_metadata, scale=0.6)
    vis = visualizer.draw_dataset_dict(d)

    # Display the image with annotations
    labeled_image = vis.get_image()[:, :, ::-1]
    scaled_labeled_image = cv2.resize(labeled_image, (0,0), fy=0.6, fx=0.6)
    cv2_imshow(scaled_labeled_image)
```

```
Processing file: ./data/images/all/CD-072_000005.jpg
```





```
Processing file: ./data/images/all/AS-005_000006.jpg
```



Train

Setting and configuring the model

```
class TrainerWithTensorboard(DefaultTrainer):
    @classmethod
    def build_writers(cls, cfg):
        return [TensorboardXWriter(cfg.OUTPUT_DIR)]

cfg = get_cfg()
cfg.OUTPUT_DIR = "./models/Detectron2_Models"
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_test",)
cfg.DATASETS.VAL = ("my_dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # Let training
cfg.SOLVER.IMS_PER_BATCH = 4  # Adjust this parameter (Batch Size)
cfg.SOLVER.BASE_LR = 0.0025  # Adjust this parameter (Learning Rate) initial=0.00025
cfg.SOLVER.MAX_ITER = 500    # 1000 iterations does well for this dataset
cfg.SOLVER.STEPS = []         # do not decay learning rate
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512   # Default is 512
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2  # We have 2 classes ["Tubes", "Spheres"].

os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg) #Create an instance of of DefaultTrainer with the given congiguration
trainer.resume_or_load(resume=False) #Load a pretrained model if available (resume training) or start training from scra
```

```
[07/30 14:44:56 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(
```

```
    (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(
```

Train the model

```
trainer.train()
```

```
[07/30 14:45:05 d2.engine.train_loop]: Starting training from iteration 0
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompatibl
  self.pid = os.fork()
[07/30 14:45:31 d2.utils.events]:  eta: 0:08:50  iter: 19  total_loss: 2.979  loss_cls: 1.049  loss_box_reg: 0.5983
[07/30 14:45:54 d2.utils.events]:  eta: 0:08:28  iter: 39  total_loss: 2.158  loss_cls: 0.6302  loss_box_reg: 0.647
[07/30 14:46:17 d2.utils.events]:  eta: 0:08:08  iter: 59  total_loss: 1.832  loss_cls: 0.5043  loss_box_reg: 0.6613
[07/30 14:46:40 d2.utils.events]:  eta: 0:07:55  iter: 79  total_loss: 1.518  loss_cls: 0.3865  loss_box_reg: 0.6077
[07/30 14:47:04 d2.utils.events]:  eta: 0:07:35  iter: 99  total_loss: 1.3  loss_cls: 0.3201  loss_box_reg: 0.5213
[07/30 14:47:28 d2.utils.events]:  eta: 0:07:14  iter: 119  total_loss: 1.078  loss_cls: 0.2729  loss_box_reg: 0.395
[07/30 14:47:51 d2.utils.events]:  eta: 0:06:50  iter: 139  total_loss: 1.043  loss_cls: 0.275  loss_box_reg: 0.3804
[07/30 14:48:15 d2.utils.events]:  eta: 0:06:28  iter: 159  total_loss: 0.8852  loss_cls: 0.2419  loss_box_reg: 0.30
[07/30 14:48:38 d2.utils.events]:  eta: 0:06:05  iter: 179  total_loss: 0.8003  loss_cls: 0.1957  loss_box_reg: 0.28
[07/30 14:49:02 d2.utils.events]:  eta: 0:05:47  iter: 199  total_loss: 0.8074  loss_cls: 0.1947  loss_box_reg: 0.29
[07/30 14:49:26 d2.utils.events]:  eta: 0:05:24  iter: 219  total_loss: 0.7845  loss_cls: 0.1799  loss_box_reg: 0.28
[07/30 14:49:50 d2.utils.events]:  eta: 0:05:01  iter: 239  total_loss: 0.7438  loss_cls: 0.1856  loss_box_reg: 0.26
[07/30 14:50:14 d2.utils.events]:  eta: 0:04:38  iter: 259  total_loss: 0.7317  loss_cls: 0.1753  loss_box_reg: 0.25
[07/30 14:50:37 d2.utils.events]:  eta: 0:04:15  iter: 279  total_loss: 0.7169  loss_cls: 0.1726  loss_box_reg: 0.25
[07/30 14:51:02 d2.utils.events]:  eta: 0:03:52  iter: 299  total_loss: 0.6586  loss_cls: 0.1443  loss_box_reg: 0.23
[07/30 14:51:26 d2.utils.events]:  eta: 0:03:29  iter: 319  total_loss: 0.7388  loss_cls: 0.1522  loss_box_reg: 0.26
[07/30 14:51:50 d2.utils.events]:  eta: 0:03:06  iter: 339  total_loss: 0.6568  loss_cls: 0.1634  loss_box_reg: 0.23
[07/30 14:52:13 d2.utils.events]:  eta: 0:02:43  iter: 359  total_loss: 0.6692  loss_cls: 0.1609  loss_box_reg: 0.23
[07/30 14:52:38 d2.utils.events]:  eta: 0:02:20  iter: 379  total_loss: 0.6814  loss_cls: 0.1417  loss_box_reg: 0.23
[07/30 14:53:03 d2.utils.events]:  eta: 0:01:56  iter: 399  total_loss: 0.6528  loss_cls: 0.13  loss_box_reg: 0.2398
[07/30 14:53:29 d2.utils.events]:  eta: 0:01:33  iter: 419  total_loss: 0.6574  loss_cls: 0.1397  loss_box_reg: 0.22
```

```
[07/30 14:53:54 d2.utils.events]:  eta: 0:01:10  iter: 439  total_loss: 0.646  loss_cls: 0.1369  loss_box_reg: 0.229
[07/30 14:54:17 d2.utils.events]:  eta: 0:00:46  iter: 459  total_loss: 0.6063  loss_cls: 0.1235  loss_box_reg: 0.21
[07/30 14:54:42 d2.utils.events]:  eta: 0:00:23  iter: 479  total_loss: 0.65  loss_cls: 0.1444  loss_box_reg: 0.2269
[07/30 14:55:13 d2.utils.events]:  eta: 0:00:00  iter: 499  total_loss: 0.6354  loss_cls: 0.1275  loss_box_reg: 0.22
[07/30 14:55:13 d2.engine.hooks]: Overall training speed: 498 iterations in 0:09:54 (1.1932 s / it)
[07/30 14:55:13 d2.engine.hooks]: Total training time: 0:10:02 (0:00:08 on hooks)
WARNING [07/30 14:55:13 d2.data.datasets.coco]:
Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

[07/30 14:55:13 d2.data.datasets.coco]: Loaded 11 images in COCO format from ./data/annotations/test_annotations.jso
[07/30 14:55:13 d2.data.build]: Distribution of instances among all 2 categories:
|  category  | #instances   |  category  | #instances   |
|:----------:|:-------------|:----------:|:-------------|
|    Rods    | 518          |  Spheres   | 0            |
|            |              |            |              |
|   total    | 518          |            |              |
[07/30 14:55:13 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in inference: [ResizeShortestEdge(short_
[07/30 14:55:13 d2.data.common]: Serializing the dataset using: <class 'detectron2.data.common._TorchSerializedList'
[07/30 14:55:13 d2.data.common]: Serializing 11 elements to byte tensors and concatenating them all ...
[07/30 14:55:13 d2.data.common]: Serialized dataset takes 0.28 MiB
WARNING [07/30 14:55:13 d2.engine.defaults]: No evaluator found. Use `DefaultTrainer.test(evaluators=)`, or implemen
```

```
# Close the TensorBoard writer
writer.close()
```

Save the Configuration

```
import yaml
import datetime

# Create a configuration file with timestamp
timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
model_path = "./models/Detectron2_Models/"
yaml_file_name = f"config_{timestamp}.yaml"
config_yaml_path = os.path.join(model_path, yaml_file_name)
with open(config_yaml_path, 'w') as file:
    yaml.dump(cfg, file)
```

```
    yaml.dump(cfg, file)

model_weight_filename = f'model_weights_{timestamp}.pth'
model_weight_path = os.path.join(model_path, model_weight_filename)
torch.save(trainer.model.state_dict(), model_weight_path)
```

Evalation of trained model

```
def load_model(config_file_path, model_weights_path, score_thresh=0.6):
    """
    Load the Detectron2 model with the given configuration and weights.

    Args:
        config_file_path (str): Path to the configuration file.
        model_weights_path (str): Path to the model weights file.
        score_thresh (float): Score threshold for the model. Default is 0.6.

    Returns:
        DefaultPredictor: The predictor with the loaded model.
    """

    cfg = get_cfg()
    cfg.merge_from_file(config_file_path)
    cfg.MODEL.WEIGHTS = model_weights_path
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = score_thresh
    predictor = DefaultPredictor(cfg)

    return predictor
```

Uncomment the below code to load a previously trained model.

```
#config_file_path = "./models/Detectron2_Models/config.yaml"
#model_weights_path = "./models/Detectron2_Models/model_final.pth"
#load_model(config_file_path, model_weights_path, score_thresh=0.6)
```

TensorBoard

```
%load_ext tensorboard
%tensorboard --logdir ./output/logs #./models/Detectron2_Models
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

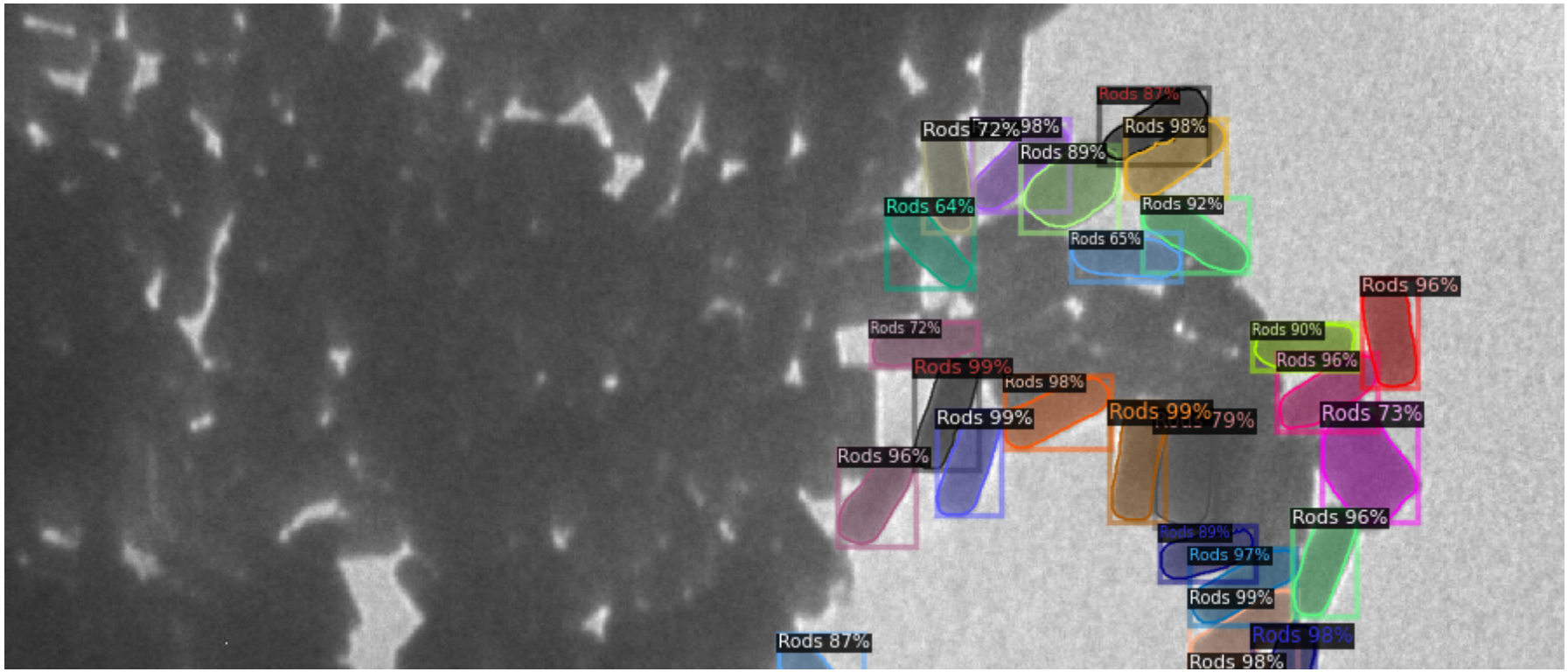
```
# For evaluation
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader

# Evaluate the model
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
predictor = DefaultPredictor(cfg)

evaluator = COCOEvaluator("my_dataset_val", output_dir="./output")
val_loader = build_detection_test_loader(cfg, "my_dataset_val")
print(inference_on_dataset(predictor.model, val_loader, evaluator))
```

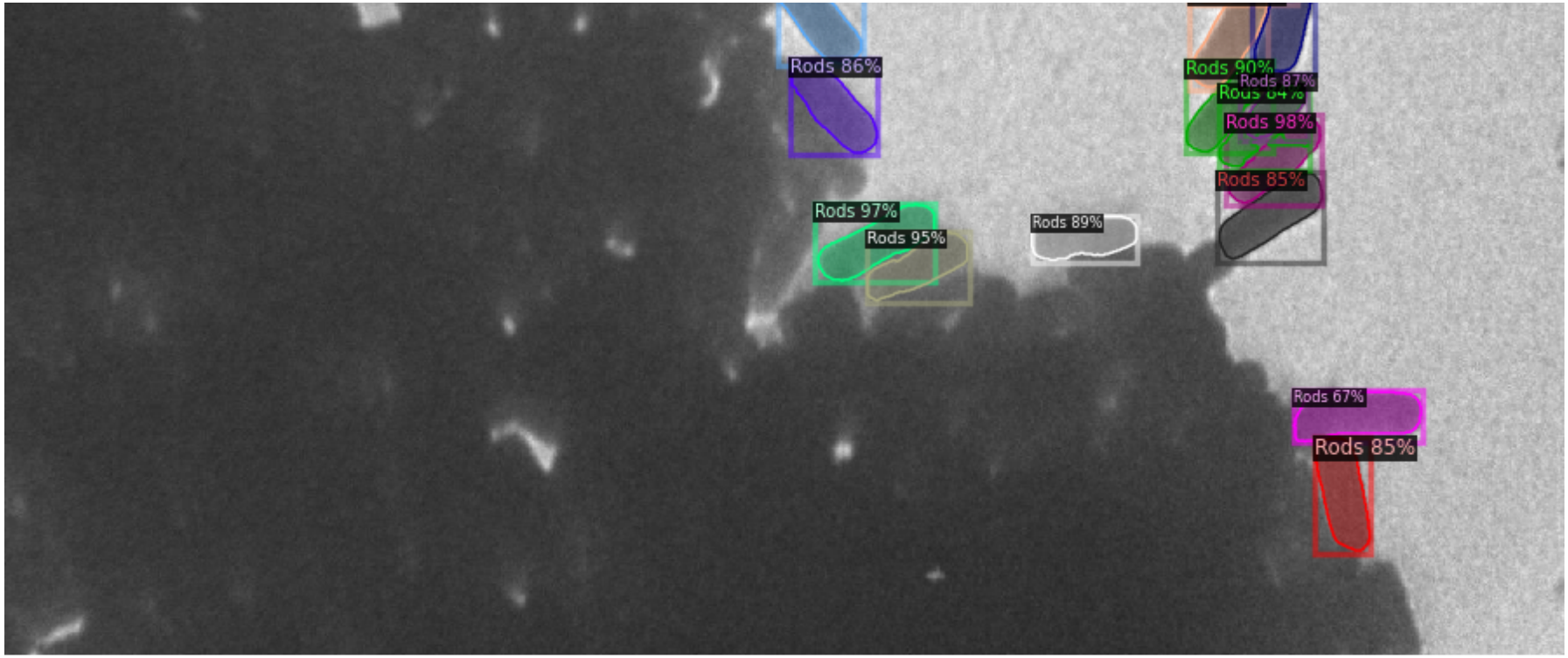
```
[07/30 04:46:04 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from ./models/Detectron2_Models
[07/30 04:46:04 d2.evaluation.coco_evaluation]: Fast COCO eval is not built. Falling back to official COCO eval.
WARNING [07/30 04:46:04 d2.data.datasets.coco]:
Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

[07/30 04:46:04 d2.data.datasets.coco]: Loaded 21 images in COCO format from ./data/annotations/val_annotations.json
[07/30 04:46:04 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in inference: [ResizeShortestEdge(short_
[07/30 04:46:04 d2.data.common]: Serializing the dataset using: <class 'detectron2.data.common._TorchSerializedList'
[07/30 04:46:04 d2.data.common]: Serializing 21 elements to byte tensors and concatenating them all ...
[07/30 04:46:04 d2.data.common]: Serialized dataset takes 0.91 MiB
[07/30 04:46:04 d2.evaluation.evaluator]: Start inference on 21 batches
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompatibl
```

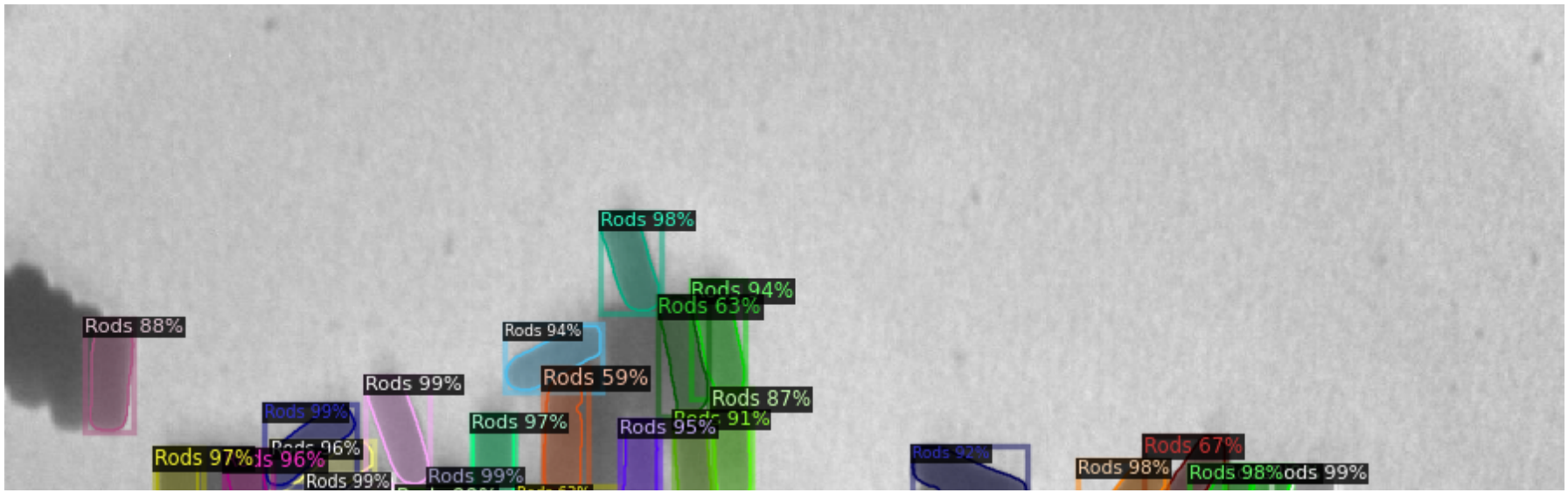
```
  self.pid = os.fork()
[07/30 04:46:09 d2.evaluation.evaluator]: Inference done 11/21. Dataloading: 0.0017 s/iter. Inference: 0.1326 s/iter
[07/30 04:46:15 d2.evaluation.evaluator]: Inference done 19/21. Dataloading: 0.0042 s/iter. Inference: 0.1714 s/iter
[07/30 04:46:16 d2.evaluation.evaluator]: Total inference time: 0:00:08.844365 (0.552773 s / iter per device, on 1 d
[07/30 04:46:16 d2.evaluation.evaluator]: Total inference pure compute time: 0:00:02 (0.168142 s / iter per device,
[07/30 04:46:16 d2.evaluation.coco_evaluation]: Preparing results for COCO format ...
[07/30 04:46:16 d2.evaluation.coco_evaluation]: Saving results to ./output/coco_instances_results.json
[07/30 04:46:16 d2.evaluation.coco_evaluation]: Evaluating predictions with official COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=2.63s).
Accumulating evaluation results...
DONE (t=0.02s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.562
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.716
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.694
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.483
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.567
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.010
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.092
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.607
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.507
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.624
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
[07/30 04:46:18 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl  |
|:------:|:------:|:------:|:------:|:------:|:-----:|
| 56.171 | 71.577 | 69.361 | 48.273 | 56.687 |  nan  |
[07/30 04:46:18 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[07/30 04:46:18 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category   | AP     |
|:-----------|:-------|:-----------|:-------|
| Rods       | 49.264 | Spheres    | 63.079 |
Loading and preparing results...
DONE (t=0.01s)
creating index...
index created!
```

```
Index created!
Running per image evaluation...
Evaluate annotation type *segm*
DONE (t=2.62s).
Accumulating evaluation results...
```

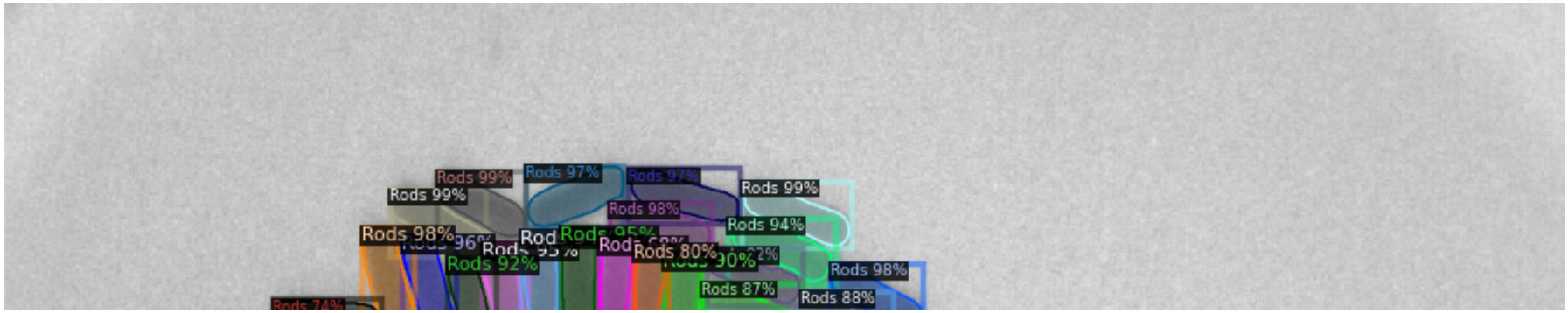
```
for d in random.sample(val_dataset_dicts, 4):    #select number of images for display
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1],
                   metadata=val_metadata,
                   scale=0.7,
                   instance_mode=ColorMode.IMAGE_BW   # remove the colors of unsegmented pixels. This option is only ava
    )
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(out.get_image()[:, :, ::-1])
    output_filepath = os.path.join("./output/predictions/images/", os.path.basename(d["file_name"]))
    cv2.imwrite(output_filepath, out.get_image()[:, :, ::-1])
```

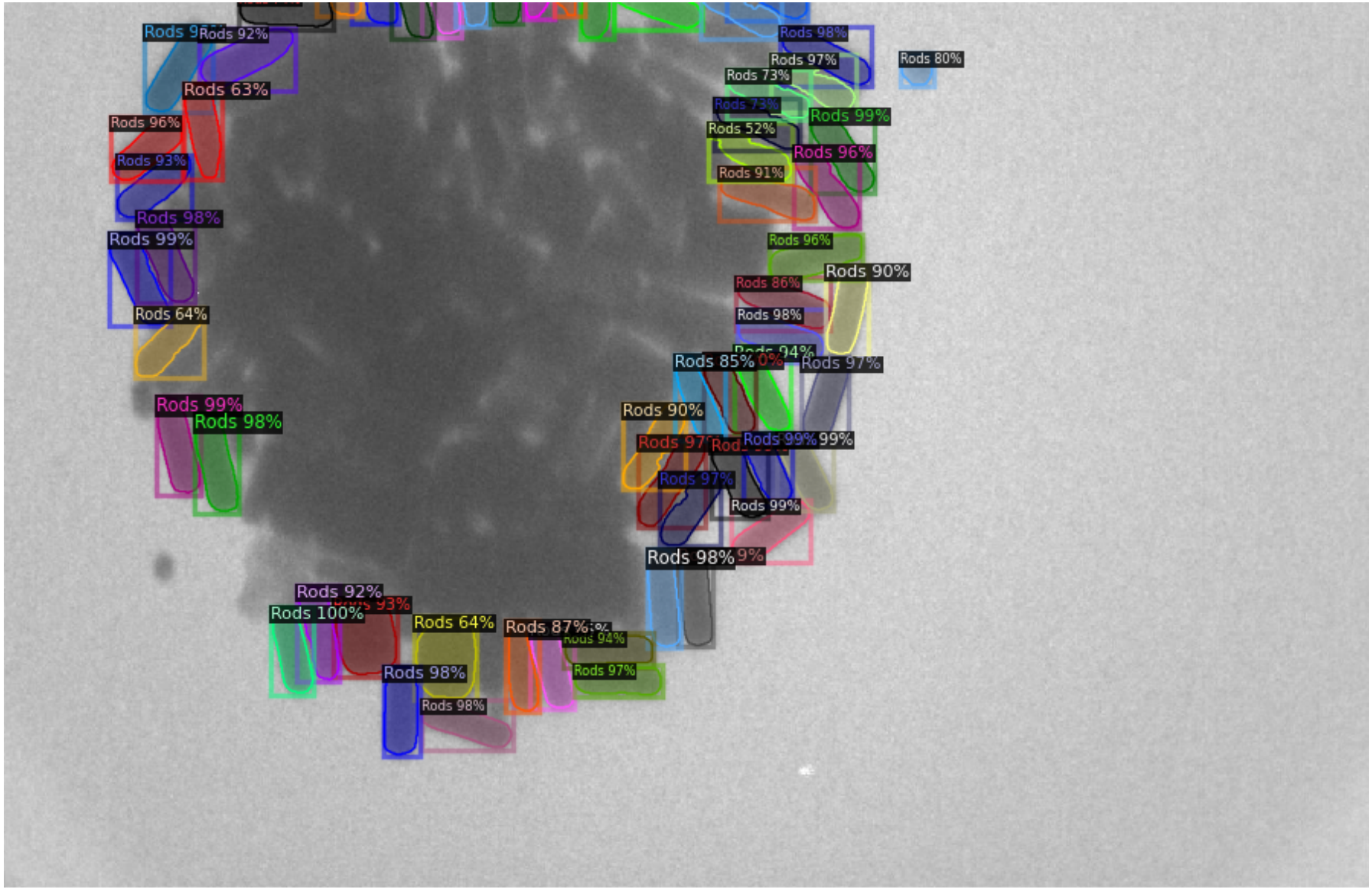



Rods 86%
Rods 90%
Rods 87%
Rods 98%
Rods 85%
Rods 97%
Rods 95%
Rods 89%
Rods 67%
Rods 85%

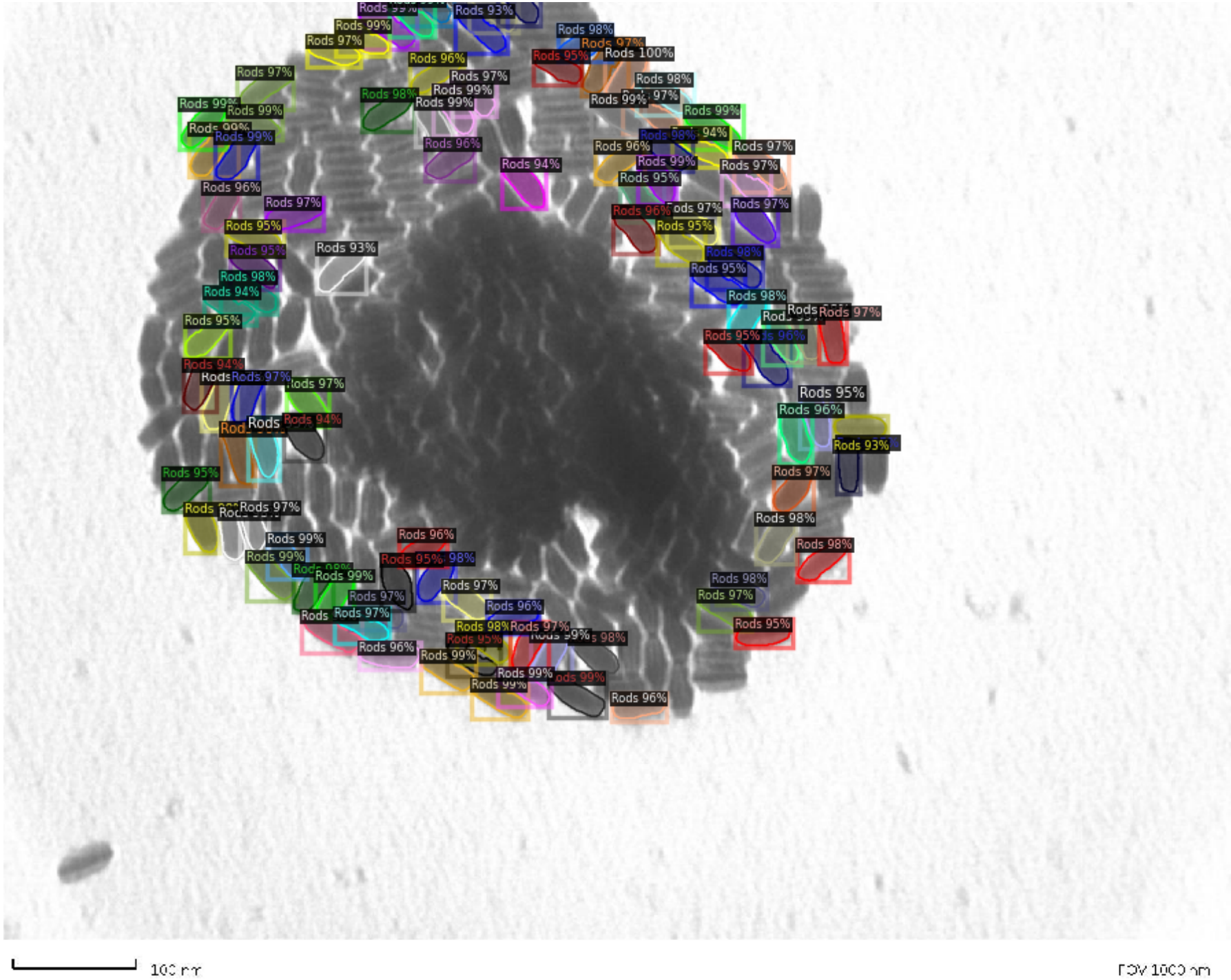

Rods 98%
Rods 94%
Rods 63%
Rods 88%
Rods 94%
Rods 99%
Rods 59%
Rods 87%
Rods 97%
Rods 95%
Rods 92%
Rods 67%
Rods 98%
Rods 98%
Rods 99%


Rods 99%
Rods 80%
Rods 99%
Rods 86%
Rods 82%
Rods 83%
Rods 94%
Rods 93%
Rods 71%
Rods 53%
Rods 94%
Rods 97%
Rods 86%
Rods 92%
Rods 98%
Rods 96%
Rods 88%
Rods 95%
Rods 97%
Rods 97%
Rods 87%
Rods 99%
Rods 99%
Rods 94%
Rods 66%
Rods 82%
Rods 98%
Rods 52%
Rods 99%
Rods 97%
Rods 97%
Rods 51%
Rods 77%
Rods 96%
Rods 87%
Rods 63%


Rods 99%
Rods 97%
Rods 97%
Rods 99%
Rods 99%
Rods 98%
Rods 98%
Rods 94%
Rods 92%
Rods 80%
Rods 98%
Rods 87%
Rods 88%


Rods 92%
Rods 63%
Rods 96%
Rods 93%
Rods 98%
Rods 99%
Rods 64%
Rods 99%
Rods 98%
Rods 92%
Rods 100%
Rods 64%
Rods 98%
Rods 98%
Rods 87%
Rods 97%
Rods 98%
Rods 73%
Rods 52%
Rods 91%
Rods 96%
Rods 86%
Rods 98%
Rods 90%
Rods 97%
Rods 99%
Rods 97%
Rods 99%
Rods 98%
Rods 80%
Rods 73%
Rods 99%
Rods 96%
Rods 90%
Rods 85%
Rods 90%
Rods 97%
Rods 99%
Rods 99%


Rods 97%
Rods 98%


Rods 93%
Rods 99%
Rods 97%
Rods 98%
Rods 100%
Rods 96%
Rods 95%
Rods 99%
Rods 94%

Double-click (or enter) to edit

```
import pandas as pd

def run_inference_and_save_results(cfg, image_dir, output_csv):
    predictor = DefaultPredictor(cfg)
    results = []

    for filename in os.listdir(image_dir):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            file_path = os.path.join(image_dir, filename)
            image = cv2.imread(file_path)
            outputs = predictor(image)

            pred_classes = outputs["instances"].pred_classes.cpu().numpy()
            pred_masks = outputs["instances"].pred_masks.cpu().numpy()
            unique, counts = np.unique(pred_classes, return_counts=True)
            class_counts = dict(zip(unique, counts))

            num_rods = class_counts.get(0, 0)  # Assuming 'Rods' is class 0
            num_spheres = class_counts.get(1, 0)  # Assuming 'Spheres' is class 1

            rod_area = []
            sphere_area = []

            for i, cls in enumerate(pred_classes):
                area = np.sum(pred_masks[i])
                if cls == 0:  # Rods
                    rod_area.append(area)
                elif cls == 1:  # Spheres
                    sphere_area.append(area)

            results.append({
```

```
                "filename": filename,
                "number_of_rods": num_rods,
                "number_of_spheres": num_spheres,
                "rod_area": rod_area,
                "sphere_area": sphere_area
            })

    df = pd.DataFrame(results)
    df.to_csv(output_csv, index=False)
    print(f"Results saved to {output_csv}")
```

Save Data to csv file

```
# Run inference and save results
image_dir = "./data/images/evaluate/"
output_csv = "./output/results/output.csv"
run_inference_and_save_results(cfg, image_dir, output_csv)
```

```
[07/30 04:46:29 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from ./models/Detectron2_Models
Results saved to ./output/results/output.csv
```