```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
import sys, os, distutils.core
# Note: This is a faster way to install detectron2 in Colab, but it does not include all functionalities (e.g. compiled operators).
# See https://detectron2.readthedocs.io/tutorials/install.html for full installation instructions
!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'))
```

```
  Downloading black-23.12.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 21.0 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (23.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
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) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
```

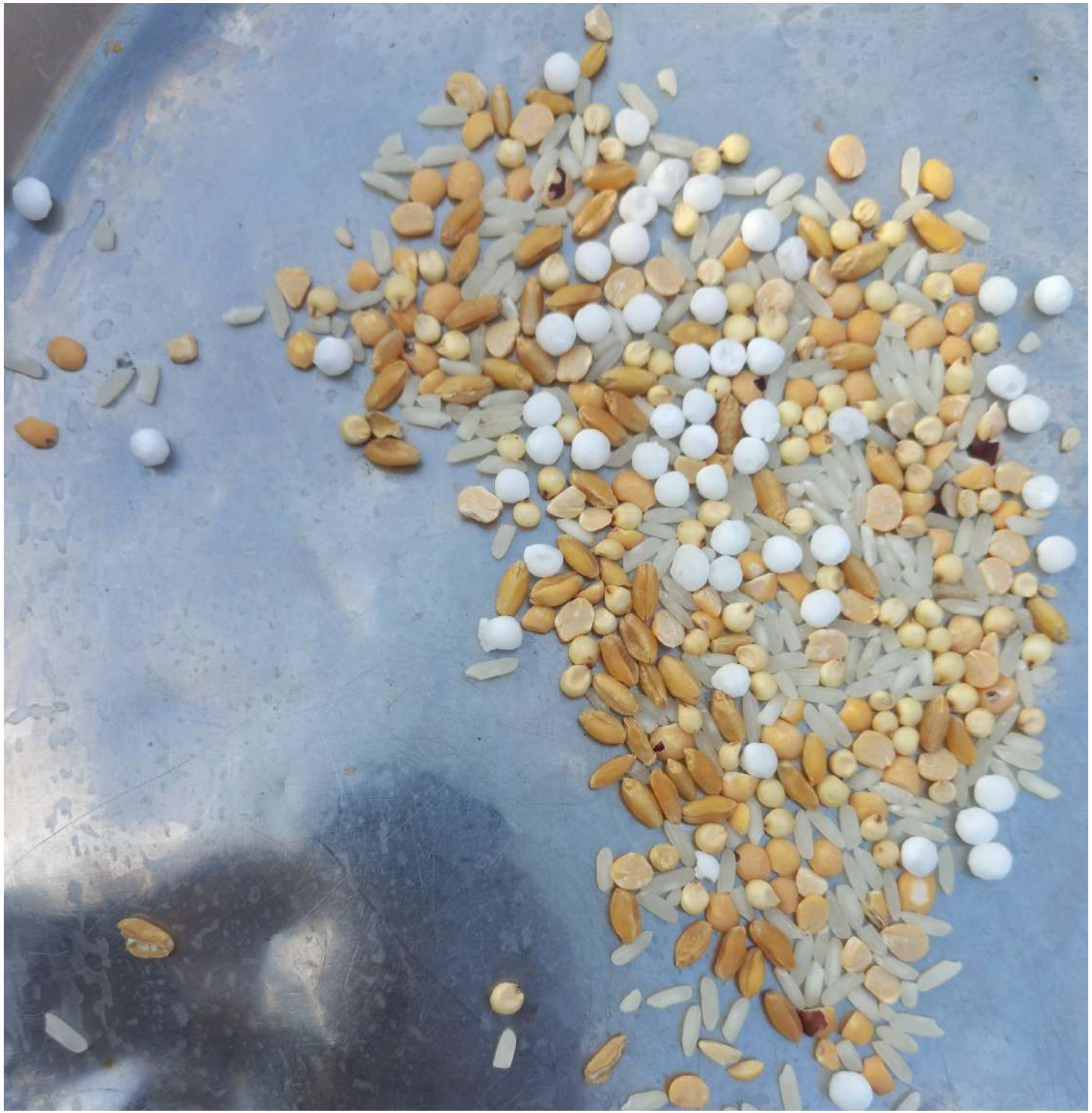
```
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->ten
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard) (2.0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard) (2023
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oaut
Building wheels for collected packages: fvcore, antlr4-python3-runtime
  Building wheel for fvcore (setup.py) ... done
  Created wheel for fvcore: filename=fvcore-0.1.5.post20221221-py3-none-any.whl size=61400 sha256=7285d9663b781f1a860629af009c322e3163b1a9
  Stored in directory: /root/.cache/pip/wheels/01/c0/af/77c1cf53a1be9e42a52b48e5af2169d40ec2e89f7362489dd0
  Building wheel for antlr4-python3-runtime (setup.py) ... done
  Created wheel for antlr4-python3-runtime: filename=antlr4_python3_runtime-4.9.3-py3-none-any.whl size=144554 sha256=5fa95887f4c202698155
  Stored in directory: /root/.cache/pip/wheels/12/93/dd/1f6a127edc45659556564c5730f6d4e300888f4bca2d4c5a88
Successfully built fvcore antlr4-python3-runtime
Installing collected packages: antlr4-python3-runtime, yacs, portalocker, pathspec, omegaconf, mypy-extensions, iopath, hydra-core, black,
Successfully installed antlr4-python3-runtime-4.9.3 black-23.12.1 fvcore-0.1.5.post20221221 hydra-core-1.3.2 iopath-0.1.9 mypy-extensions-
```

```
import torch, detectron2
!nvcc --version
TORCH_VERSION = ".".join(torch.__version__.split(".")[:2])
CUDA_VERSION = torch.__version__.split("+")[-1]
print("torch: ", TORCH_VERSION, "; cuda: ", CUDA_VERSION)
print("detectron2:", detectron2.__version__)
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Aug_15_22:02:13_PDT_2023
Cuda compilation tools, release 12.2, V12.2.140
Build cuda_12.2.r12.2/compiler.33191640_0
torch:  2.1 ; cuda:  cu121
detectron2: 0.6
```

```
# Some basic setup:
# Setup detectron2 logger
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()
# import some common libraries
import numpy as np
import os, json, cv2, random
from google.colab.patches import cv2_imshow
# import some common detectron2 utilities
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
```

```
im = cv2.imread("/content/drive/MyDrive/Train 1/20240120_090416.jpg")
cv2_imshow(im)
```



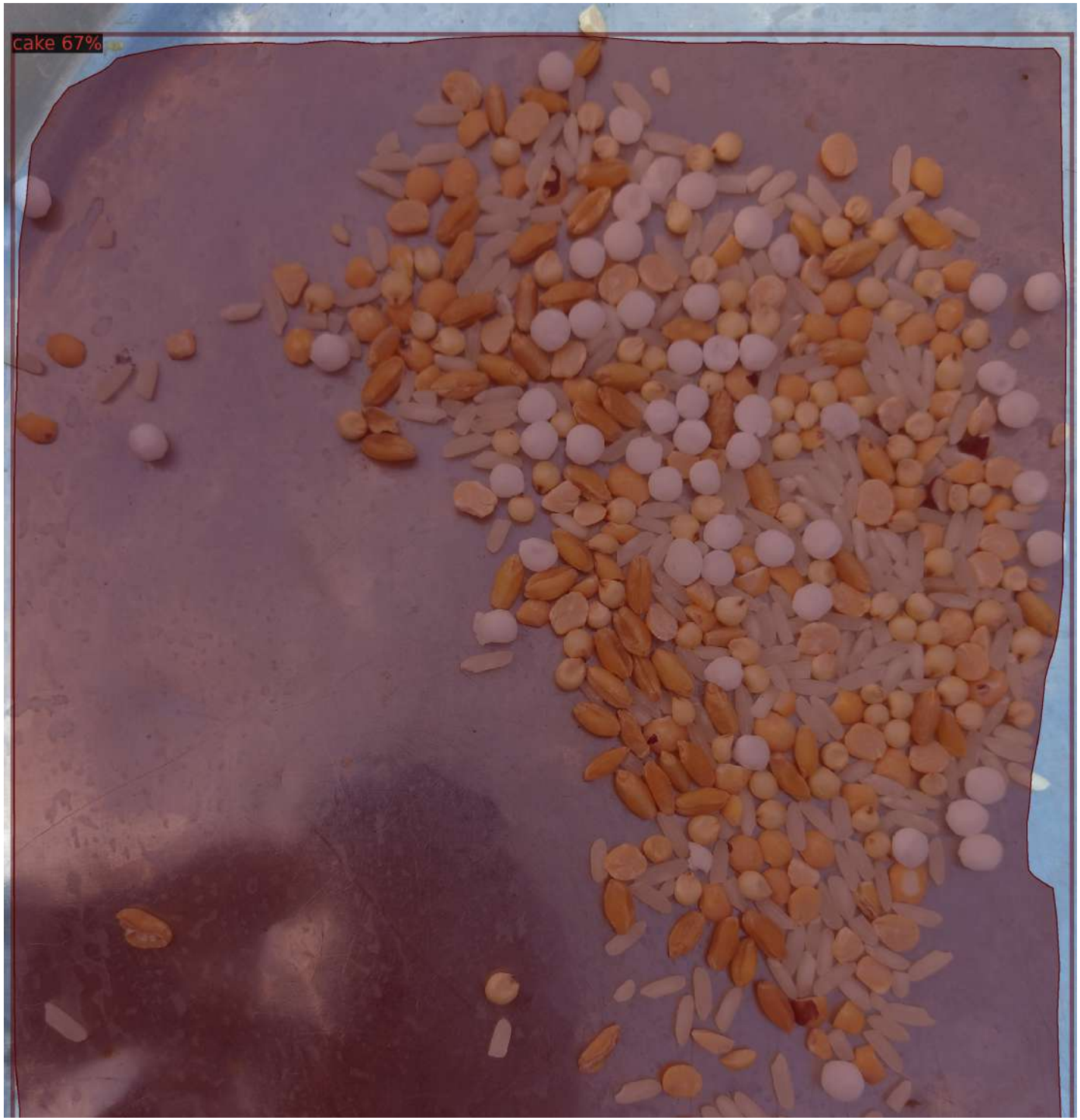
```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5 # set threshold for this model
# Find a model from detectron2's model zoo. https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
predictor = DefaultPredictor(cfg)
outputs = predictor(im)
```

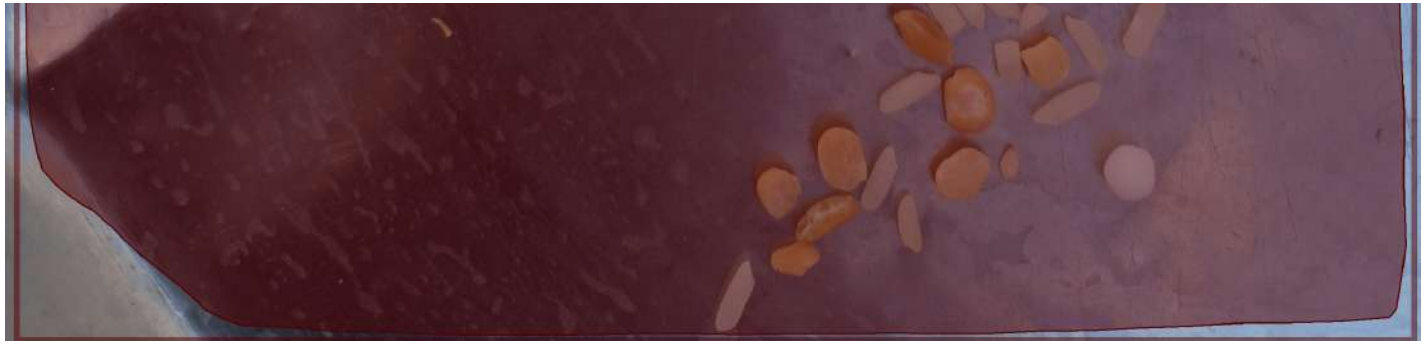
```
[01/21 07:48:27 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from https://dl.fbaipublicfiles.com/detectron2/COCO-Ins
model_final_f10217.pkl: 178MB [00:00, 209MB/s]
/usr/local/lib/python3.10/dist-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming release, it will be required to
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
```

```
# look at the outputs - tensors and bounding boxes.
print(outputs["instances"].pred_classes)
print(outputs["instances"].pred_boxes)
```

```
tensor([55], device='cuda:0')
Boxes(tensor([[  21.5279,   77.5454, 2649.1726, 3398.6694]], device='cuda:0'))
```

```
# We can use `Visualizer` to draw the predictions on the image.
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=0.8)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```


cake 67%



```
from detectron2.data.datasets import register_coco_instances
register_coco_instances("dataset_train", {}, "/content/drive/MyDrive/Train 1/labels_my-project-name_2024-01-20-01-51-10.json", "/content/drive/M
register_coco_instances("dataset_val", {}, "/content/drive/MyDrive/Test/labels_my-project-name_2024-01-20-10-00-25-1.json", "/content/drive/MyDr
```

```
train_metadata = MetadataCatalog.get("dataset_train")
train_dataset_dicts = DatasetCatalog.get("dataset_train")
```

```
[01/21 07:51:04 d2.data.datasets.coco]: Loaded 5 images in COCO format from /content/drive/MyDrive/Train 1/labels_my-project-name_2024-01-20
```

```
val_metadata = MetadataCatalog.get("dataset_val")
val_dataset_dicts = DatasetCatalog.get("dataset_val")
```

```
[01/21 07:51:08 d2.data.datasets.coco]: Loaded 3 images in COCO format from /content/drive/MyDrive/Test/labels_my-project-name_2024-01-20-10
```

```
from detectron2.engine import DefaultTrainer

cfg = get_cfg()
cfg.OUTPUT_DIR = "/content/drive/MyDrive/Train 1"
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("dataset_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # Let training initialize from model z
cfg.SOLVER.IMS_PER_BATCH = 2  # This is the real "batch size" commonly known to deep learning people
cfg.SOLVER.BASE_LR = 0.00025  # pick a good LR
cfg.SOLVER.MAX_ITER = 1000    # 1000 iterations seems good enough for this dataset
cfg.SOLVER.STEPS = []        # do not decay learning rate
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 256   # Default is 512, using 256 for this dataset.
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 5 # We have 5 classes.
# NOTE: this config means the number of classes, without the background. Do not use num_classes+1 here.

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 scratch if no pretrained moc
```

```
      (deconv): ConvTranspose2d(256, 256, kernel_size=(2, 2), stride=(2, 2))
      (deconv_relu): ReLU()
      (predictor): Conv2d(256, 5, kernel_size=(1, 1), stride=(1, 1))
    )
  )
)
[01/21 07:51:27 d2.data.datasets.coco]: Loaded 5 images in COCO format from /content/drive/MyDrive/Train 1/labels_my-project-name_2024-01-
[01/21 07:51:27 d2.data.build]: Removed 0 images with no usable annotations. 5 images left.
[01/21 07:51:27 d2.data.build]: Distribution of instances among all 5 categories:
|  category  | #instances   |  category  | #instances   |  category  | #instances   |
|:----------:|:-------------|:----------:|:-------------|:----------:|:-------------|
|   Shabu    | 25           |   wheat    | 25           |   jowar    | 24           |
|    Dal     | 25           |    Rise    | 25           |            |              |
|   total    | 124          |            |              |            |              |
[01/21 07:51:27 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in training: [ResizeShortestEdge(short_edge_length=(640, 672,
[01/21 07:51:27 d2.data.build]: Using training sampler TrainingSampler
[01/21 07:51:27 d2.data.common]: Serializing the dataset using: <class 'detectron2.data.common._TorchSerializedList'>
[01/21 07:51:27 d2.data.common]: Serializing 5 elements to byte tensors and concatenating them all ...
[01/21 07:51:27 d2.data.common]: Serialized dataset takes 0.03 MiB
[01/21 07:51:27 d2.data.build]: Making batched data loader with batch_size=2
[01/21 07:51:27 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from https://dl.fbaipublicfiles.com/detectron2/COCO-I
WARNING:fvcore.common.checkpoint:Skip loading parameter 'roi_heads.box_predictor.cls_score.weight' to the model due to incompatible shapes
WARNING:fvcore.common.checkpoint:Skip loading parameter 'roi_heads.box_predictor.cls_score.bias' to the model due to incompatible shapes:
WARNING:fvcore.common.checkpoint:Skip loading parameter 'roi_heads.box_predictor.bbox_pred.weight' to the model due to incompatible shapes
WARNING:fvcore.common.checkpoint:Skip loading parameter 'roi_heads.box_predictor.bbox_pred.bias' to the model due to incompatible shapes:
WARNING:fvcore.common.checkpoint:Skip loading parameter 'roi_heads.mask_head.predictor.weight' to the model due to incompatible shapes: (8
WARNING:fvcore.common.checkpoint:Skip loading parameter 'roi_heads.mask_head.predictor.bias' to the model due to incompatible shapes: (80,
WARNING:fvcore.common.checkpoint: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}
```

```
trainer.train() #Start the training process
```

```
[01/21 07:51:42 d2.engine.train_loop]: Starting training from iteration 0
[01/21 07:52:03 d2.utils.events]:  eta: 0:12:05  iter: 19  total_loss: 4.861  loss_cls: 2.004  loss_box_reg: 0.1805  loss_mask: 0.6973  loss
[01/21 07:52:21 d2.utils.events]:  eta: 0:11:32  iter: 39  total_loss: 3.213  loss_cls: 1.776  loss_box_reg: 0.221  loss_mask: 0.6924  loss_
[01/21 07:52:36 d2.utils.events]:  eta: 0:11:18  iter: 59  total_loss: 2.779  loss_cls: 1.538  loss_box_reg: 0.2212  loss_mask: 0.6824  loss
[01/21 07:52:51 d2.utils.events]:  eta: 0:10:44  iter: 79  total_loss: 2.477  loss_cls: 1.275  loss_box_reg: 0.2214  loss_mask: 0.6677  loss
[01/21 07:53:07 d2.utils.events]:  eta: 0:10:30  iter: 99  total_loss: 2.191  loss_cls: 0.9907  loss_box_reg: 0.2292  loss_mask: 0.6481  los
[01/21 07:53:22 d2.utils.events]:  eta: 0:10:16  iter: 119  total_loss: 1.955  loss_cls: 0.8549  loss_box_reg: 0.2098  loss_mask: 0.6252  lo
[01/21 07:53:38 d2.utils.events]:  eta: 0:10:02  iter: 139  total_loss: 1.903  loss_cls: 0.8119  loss_box_reg: 0.2323  loss_mask: 0.6019  lo
[01/21 07:53:54 d2.utils.events]:  eta: 0:09:56  iter: 159  total_loss: 1.873  loss_cls: 0.8139  loss_box_reg: 0.2586  loss_mask: 0.5739  lo
[01/21 07:54:09 d2.utils.events]:  eta: 0:09:38  iter: 179  total_loss: 1.845  loss_cls: 0.802  loss_box_reg: 0.2433  loss_mask: 0.5451  los
[01/21 07:54:25 d2.utils.events]:  eta: 0:09:28  iter: 199  total_loss: 1.783  loss_cls: 0.7934  loss_box_reg: 0.269  loss_mask: 0.5186  los
[01/21 07:54:42 d2.utils.events]:  eta: 0:09:11  iter: 219  total_loss: 1.787  loss_cls: 0.7808  loss_box_reg: 0.2764  loss_mask: 0.4888  lo
[01/21 07:54:57 d2.utils.events]:  eta: 0:09:00  iter: 239  total_loss: 1.737  loss_cls: 0.7811  loss_box_reg: 0.2765  loss_mask: 0.4605  lo
[01/21 07:55:13 d2.utils.events]:  eta: 0:08:51  iter: 259  total_loss: 1.725  loss_cls: 0.7766  loss_box_reg: 0.2873  loss_mask: 0.4333  lo
[01/21 07:55:29 d2.utils.events]:  eta: 0:08:38  iter: 279  total_loss: 1.697  loss_cls: 0.7629  loss_box_reg: 0.3119  loss_mask: 0.4193  lo
[01/21 07:55:50 d2.utils.events]:  eta: 0:08:33  iter: 299  total_loss: 1.653  loss_cls: 0.7603  loss_box_reg: 0.3051  loss_mask: 0.3987  lo
[01/21 07:56:05 d2.utils.events]:  eta: 0:08:13  iter: 319  total_loss: 1.67  loss_cls: 0.7643  loss_box_reg: 0.3368  loss_mask: 0.3876  los
[01/21 07:56:21 d2.utils.events]:  eta: 0:08:00  iter: 339  total_loss: 1.665  loss_cls: 0.7546  loss_box_reg: 0.3584  loss_mask: 0.3702  lo
```

```
[01/21 07:56:36 d2.utils.events]:  eta: 0:07:45  iter: 359  total_loss: 1.589  loss_cls: 0.7219  loss_box_reg: 0.3326  loss_mask: 0.3563  lo
[01/21 07:56:51 d2.utils.events]:  eta: 0:07:31  iter: 379  total_loss: 1.577  loss_cls: 0.7217  loss_box_reg: 0.3581  loss_mask: 0.3418  lo
[01/21 07:57:07 d2.utils.events]:  eta: 0:07:16  iter: 399  total_loss: 1.526  loss_cls: 0.6992  loss_box_reg: 0.3541  loss_mask: 0.326  los
[01/21 07:57:22 d2.utils.events]:  eta: 0:06:59  iter: 419  total_loss: 1.508  loss_cls: 0.6751  loss_box_reg: 0.3671  loss_mask: 0.3076  lo
[01/21 07:57:37 d2.utils.events]:  eta: 0:06:46  iter: 439  total_loss: 1.467  loss_cls: 0.6459  loss_box_reg: 0.3833  loss_mask: 0.2852  lo
[01/21 07:57:52 d2.utils.events]:  eta: 0:06:30  iter: 459  total_loss: 1.39  loss_cls: 0.621  loss_box_reg: 0.3636  loss_mask: 0.2686  loss
[01/21 07:58:08 d2.utils.events]:  eta: 0:06:19  iter: 479  total_loss: 1.379  loss_cls: 0.6084  loss_box_reg: 0.3836  loss_mask: 0.2465  lo
[01/21 07:58:24 d2.utils.events]:  eta: 0:06:04  iter: 499  total_loss: 1.357  loss_cls: 0.5713  loss_box_reg: 0.3852  loss_mask: 0.2407  lo
[01/21 07:58:39 d2.utils.events]:  eta: 0:05:49  iter: 519  total_loss: 1.286  loss_cls: 0.5526  loss_box_reg: 0.3806  loss_mask: 0.2291  lo
[01/21 07:58:54 d2.utils.events]:  eta: 0:05:35  iter: 539  total_loss: 1.257  loss_cls: 0.5274  loss_box_reg: 0.3726  loss_mask: 0.2227  lo
[01/21 07:59:10 d2.utils.events]:  eta: 0:05:21  iter: 559  total_loss: 1.219  loss_cls: 0.5142  loss_box_reg: 0.3823  loss_mask: 0.2201  lo
[01/21 07:59:25 d2.utils.events]:  eta: 0:05:05  iter: 579  total_loss: 1.199  loss_cls: 0.4881  loss_box_reg: 0.3788  loss_mask: 0.2128  lo
[01/21 07:59:41 d2.utils.events]:  eta: 0:04:51  iter: 599  total_loss: 1.14  loss_cls: 0.464  loss_box_reg: 0.3596  loss_mask: 0.2085  loss
[01/21 07:59:57 d2.utils.events]:  eta: 0:04:36  iter: 619  total_loss: 1.109  loss_cls: 0.4489  loss_box_reg: 0.3523  loss_mask: 0.2075  lo
[01/21 08:00:12 d2.utils.events]:  eta: 0:04:21  iter: 639  total_loss: 1.078  loss_cls: 0.4362  loss_box_reg: 0.3395  loss_mask: 0.2034  lo
[01/21 08:00:27 d2.utils.events]:  eta: 0:04:07  iter: 659  total_loss: 1.083  loss_cls: 0.4187  loss_box_reg: 0.3491  loss_mask: 0.2045  lo
[01/21 08:00:43 d2.utils.events]:  eta: 0:03:53  iter: 679  total_loss: 1.023  loss_cls: 0.3993  loss_box_reg: 0.3235  loss_mask: 0.2018  lo
[01/21 08:00:58 d2.utils.events]:  eta: 0:03:38  iter: 699  total_loss: 0.9916  loss_cls: 0.3891  loss_box_reg: 0.3193  loss_mask: 0.1989  l
[01/21 08:01:13 d2.utils.events]:  eta: 0:03:24  iter: 719  total_loss: 0.9541  loss_cls: 0.3613  loss_box_reg: 0.3065  loss_mask: 0.1891  l
[01/21 08:01:29 d2.utils.events]:  eta: 0:03:09  iter: 739  total_loss: 0.9566  loss_cls: 0.358  loss_box_reg: 0.3021  loss_mask: 0.1972  lo
[01/21 08:01:44 d2.utils.events]:  eta: 0:02:55  iter: 759  total_loss: 0.9427  loss_cls: 0.3315  loss_box_reg: 0.3029  loss_mask: 0.1947  l
[01/21 08:01:59 d2.utils.events]:  eta: 0:02:40  iter: 779  total_loss: 0.8965  loss_cls: 0.3189  loss_box_reg: 0.2837  loss_mask: 0.1913  l
[01/21 08:02:14 d2.utils.events]:  eta: 0:02:25  iter: 799  total_loss: 0.8515  loss_cls: 0.3044  loss_box_reg: 0.2648  loss_mask: 0.1832  l
[01/21 08:02:29 d2.utils.events]:  eta: 0:02:11  iter: 819  total_loss: 0.8281  loss_cls: 0.2944  loss_box_reg: 0.2664  loss_mask: 0.1842  l
[01/21 08:02:45 d2.utils.events]:  eta: 0:01:56  iter: 839  total_loss: 0.8076  loss_cls: 0.2825  loss_box_reg: 0.2624  loss_mask: 0.1821  l
[01/21 08:03:00 d2.utils.events]:  eta: 0:01:41  iter: 859  total_loss: 0.8368  loss_cls: 0.306  loss_box_reg: 0.2439  loss_mask: 0.1828  lo
[01/21 08:03:15 d2.utils.events]:  eta: 0:01:26  iter: 879  total_loss: 0.785  loss_cls: 0.2736  loss_box_reg: 0.2434  loss_mask: 0.1799  lo
[01/21 08:03:30 d2.utils.events]:  eta: 0:01:12  iter: 899  total_loss: 0.7491  loss_cls: 0.2496  loss_box_reg: 0.2399  loss_mask: 0.177  lo
[01/21 08:03:46 d2.utils.events]:  eta: 0:00:57  iter: 919  total_loss: 0.7159  loss_cls: 0.2252  loss_box_reg: 0.2345  loss_mask: 0.1789  l
[01/21 08:04:01 d2.utils.events]:  eta: 0:00:43  iter: 939  total_loss: 0.6785  loss_cls: 0.2095  loss_box_reg: 0.2272  loss_mask: 0.1748  l
[01/21 08:04:16 d2.utils.events]:  eta: 0:00:28  iter: 959  total_loss: 0.6898  loss_cls: 0.2067  loss_box_reg: 0.2185  loss_mask: 0.1715  l
[01/21 08:04:32 d2.utils.events]:  eta: 0:00:14  iter: 979  total_loss: 0.6716  loss_cls: 0.201  loss_box_reg: 0.2281  loss_mask: 0.1733  lo
[01/21 08:04:48 d2.utils.events]:  eta: 0:00:00  iter: 999  total_loss: 0.6857  loss_cls: 0.2266  loss_box_reg: 0.2044  loss_mask: 0.1698  l
[01/21 08:04:48 d2.engine.hooks]: Overall training speed: 998 iterations in 0:12:54 (0.7761 s / it)
[01/21 08:04:48 d2.engine.hooks]: Total training time: 0:13:00 (0:00:05 on hooks)
```

```
# Look at training curves in tensorboard:
%load_ext tensorboard
%tensorboard --logdir output
```

403. That’s an error.

That’s all we know.

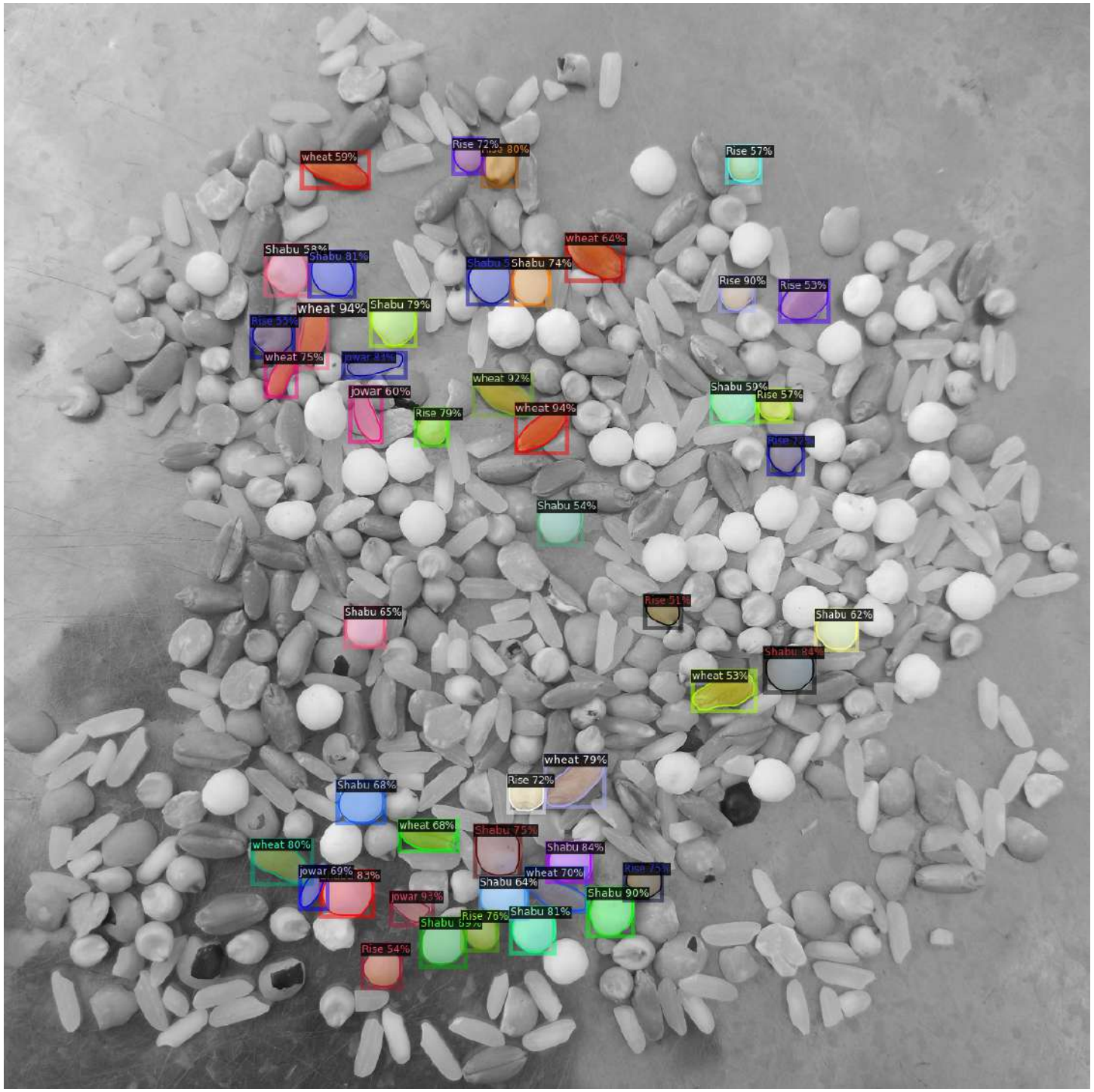
```
import yaml
# Save the configuration to a config.yaml file
# Save the configuration to a config.yaml file
config_yaml_path = "/content/drive/MyDrive/Train 1/config.yaml"
with open(config_yaml_path, 'w') as file:
    yaml.dump(cfg, file)


# Inference should use the config with parameters that are used in training
# cfg now already contains everything we've set previously. We changed it a little bit for inference:
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")  # path to the model we just trained
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5   # set a custom testing threshold
predictor = DefaultPredictor(cfg)
```

```
[01/21 08:05:37 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from /content/drive/MyDrive/Train 1/model_final.pth ...
```

```
from detectron2.utils.visualizer import ColorMode

for d in random.sample(val_dataset_dicts, 1):    #select number of images for display
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1],
                   metadata=val_metadata,
                   scale=0.5,
                   instance_mode=ColorMode.IMAGE_BW   # remove the colors of unsegmented pixels. This option is only available for segmentation
    )
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(out.get_image()[:, :, ::-1])
```


wheat 59%
Rise 72%
Rise 57%
Shabu 58%
Shabu 81%
wheat 64%
Shabu 74%
Rise 90%
Rise 53%
wheat 94%
Shabu 79%
wheat 75%
wheat 92%
jowar 60%
Rise 79%
wheat 94%
Shabu 59%
Rise 57%
Shabu 54%
Shabu 65%
Rise 51%
Shabu 62%
Shabu 84%
wheat 53%
wheat 79%
Rise 72%
Shabu 68%
wheat 68%
Shabu 75%
wheat 80%
Shabu 84%
jowar 69%
wheat 70%
Shabu 64%
Shabu 90%
Shabu 81%
Rise 76%
Rise 54%

```
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader
evaluator = COCOEvaluator("dataset_val", output_dir="./output")
val_loader = build_detection_test_loader(cfg, "dataset_val")
print(inference_on_dataset(predictor.model, val_loader, evaluator))
# another equivalent way to evaluate the model is to use `trainer.test`
```

```
DONE (t=0.03s).
Accumulating evaluation results...
```

```
DONE (t=0.00s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *segm*
DONE (t=0.03s).
Accumulating evaluation results...
DONE (t=0.02s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.042
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.048
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.048
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.063
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.015
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.023
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.067
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.118
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.147
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.113
[01/21 08:05:51 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl  |
|:-----:|:------:|:------:|:-----:|:-----:|:-----:|
| 4.212 | 4.815  | 4.815  |  nan  | 6.283 | 1.497 |
[01/21 08:05:51 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[01/21 08:05:51 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category   | AP     | category   | AP    | category   | AP    |
|:-----------|:-------|:-----------|:------|:-----------|:------|
| Shabu      | 12.857 | wheat      | 1.964 | Rise       | 0.000 |
| Dal        | 6.238  | jowar      | 0.000 |            |       |
OrderedDict([('bbox', {'AP': 3.9552594253136, 'AP50': 4.814548121478814, 'AP75': 4.814548121478814, 'APs': nan, 'APm': 13.786798679867987,
```

```
new_im = cv2.imread("/content/drive/MyDrive/Test/20240120_091015.jpg")
outputs  = predictor(new_im)

# We can use `Visualizer` to draw the predictions on the image.
v = Visualizer(new_im[:, :, ::-1], metadata=train_metadata)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))

cv2_imshow(out.get_image()[:, :, ::-1])
```


jowar 55%
jowar 70%
Dal 83%
wheat 88%
jowar 61%
Shabu 55%
jowar 52%
wheat
jowar 65%
Dal 66%
wheat 69%
wheat 65%
Rise 57%
jowar 59%
jowar 84%
wheat 74%
Shabu 67%
Dal 60%
wheat 96%
jowar 58%
wheat 96%
wheat 56%
jowar 54%
jowar 61%
Rise 74%
wheat 59%
jowar 56%
jowar 73%
Shabu 86%
jowar 51%
Rise 94%
Shabu 66%
jowar 62%
Shabu 83%
Rise 67%
Shabu 87%
Shabu 52%
wheat 82%
Shabu 70%
Rise 93%
jowar 75%
jowar 51%
Rise 93%
Rise 78%
jowar 50%
jowar 65%
Shabu 66%
Rise 62%
Dal 74%
Rise 95%
Shabu 51%
Shabu 71%
Rise 65%
Rise 86%