

Detectron2 & TFOD2 | Assignment

Question 1: What is Detectron2 and how does it differ from previous object detection frameworks?

Answer:

Detectron2 is an advanced, open-source computer vision library developed by Facebook AI Research. It is built with PyTorch and, generally, is applied to object detection, instance segmentation, semantic segmentation, keypoint detection, and panoptic segmentation. Detectron2 has a very modular design that easily allows researchers and developers to build, train, and deploy state-of-the-art vision models.

How is Detectron2 different from previous object detection frameworks

1. Built on PyTorch (Not Caffe2)

- The previous Detectron was based on Caffe2.
- Detectron2 now has a fully PyTorch implementation, making it easier to use and integrate.

2. Better Modularity and Customization

- Easy modification of backbone networks, heads, and data pipelines.
- Researchers can add custom layers or models with minimal code.

3. Faster Training and Inference

- Uses PyTorch JIT and optimized CUDA kernels.
- Improved training loops for better speed and efficiency.

4. Supports More Advanced Models

Includes latest architectures such as:

- Mask R-CNN
- Cascade R-CNN
- RetinaNet
- PointRend
- Panoptic FPN
- DensePose

Previous frameworks had fewer options.

5. Easy Deployment

- Easily export models to ONNX or TorchScript.
- Better compatibility with modern deployment tools such as TensorRT.

6. Strong Community & Frequent Updates

- Detectron2 has an active developer community.
- Receives frequent improvements and feature updates, unlike old Detectron.

Question 2: Explain the process and importance of data annotation when working with Detectron2.

Answer:

Data annotation consists of labeling images with bounding boxes, segmentation masks, class labels, or keypoints. For a Detectron2 model, annotated data is highly necessary because this model learns patterns directly from such annotated examples.

Importance of Data Annotation

1. Teaches the Model What to Detect

- Detectron2 learns object boundaries, shapes, and positions only from annotations.
- It cannot learn what objects look like without correct labels.

2. Improves Accuracy and Performance

High-quality annotations lead to:

- Better detection accuracy
- Fewer false positives/negatives
- Stronger generalization to new images

3. Required for Training Custom Datasets

- If you want Detectron2 to detect your own objects, such as helmets, fruits, and documents, it requires properly annotated data.

4. Helps Choose the Right Model

Annotations define what task you're solving:

- Bounding boxes → object detection
- Masks → instance segmentation
- Keypoints → pose estimation
- Panoptic labels → panoptic segmentation

How Data Annotation Works in Detectron2

1. Collect Images

- Collect images that are real-world representations of your objects.

2. Annotate Images Using Tools

Common annotation tools

- LabelMe → creates polygon/mask annotations
- CVAT → supports bounding boxes, masks, keypoints
- LabelImg → bounding box annotation (in Pascal VOC / YOLO format)
- Roboflow → online annotation + dataset management

3. Export Annotations in JSON Format

Detectron2 prefers formats such as:

- COCO JSON (most recommended)
- Detectron2 custom dictionary format

4. Register the Dataset in Detectron2

- Use `DatasetCatalog.register()`
- Provide paths to the image folder and annotation file

5. Preprocess and Visualize Annotations

- Detectron2 allows you to visualize annotations and confirm correctness.

6. Train the Model

- The model uses the annotations as ground truth during training.
- Loss functions compare predictions against annotated labels.

Question 3: Describe the steps involved in training a custom object detection model using Detectron2.

Answer:

Training a custom object detection model in Detectron2 involves several structured steps.

Below is the complete process:

1. Install Detectron2

- Install with pip: `pip install detectron2`
- Or install from source if using GPU and specific CUDA versions.

2. Prepare and Annotate Your Dataset

- Collect images of objects you want to detect.
- Annotate them using tools like:
 - **CVAT**
 - **LabelMe**
 - **Roboflow**
 - **LabelImg**
- Export annotations in **COCO JSON format** (best for Detectron2).

3. Organize the Dataset

4. Register the Dataset in Detectron2

Use `DatasetCatalog` and `MetadataCatalog`

This tells Detectron2 where your images and labels are.

5. Choose a Model and Configuration

- Detectron2 provides many pretrained models:
 - **Faster R-CNN**
 - **Mask R-CNN**
 - **RetinaNet**
 - **Cascade R-CNN**

6. Update Config for Custom Training

7. Start Training

Detectron2 will automatically create logs, checkpoints, and metrics.

8. Evaluate the Model

After training, evaluate performance

9. Run Inference on New Images

Load trained weights

Predict on images and visualize results.

10. Deploy or Use the Model

- Convert to ONNX/TensorRT
- Integrate into application or API
- Use in Streamlit, Flask, or mobile apps

Question 4: What are evaluation curves in Detectron2, and how are metrics like mAP and IoU interpreted?

Answer:

Evaluation Curves in Detectron2

Evaluation curves represent graphs on the performance of your object detection model. By default, Detectron2 will provide this when evaluating a trained model. It visualizes model behaviors and performances quite clearly.

Common evaluation curves:

1. Precision–Recall Curve

- It shows how precise your model is at different confidence levels.
- Helps you understand if the model detects too few objects or makes too many mistakes.

2. IoU-based Curves

- Performance when the overlap requirement between prediction and ground truth is increased.
- Shows how precisely your model draws bounding boxes.

3. Loss curves

- Track the decrease of training loss over time.
- Used to detect overfitting or underfitting.

These curves are helpful in diagnosing problems and comparing different models.

Interpretation of Key Metrics

1. IoU: Intersection over Union

What is IoU?

- IoU basically calculates the overlapping of a predicted bounding box with the real object box.
- It checks how much they overlap.

How to interpret IoU:

- High IoU → Your box is drawn accurately around the object.
- Low IoU → predicted box is far off from the actual object.
- IoU is used to decide whether a detection should be counted as correct.

2. mAP (Mean Average Precision) What is mAP?

- mAP is the chief accuracy score for object detection models.
- It reflects how well the model detects objects across various difficulty levels and thresholds.

Types in Detectron2 (COCO Metrics):

- **mAP@[0.50:0.95]** → The most important metric; checks performance from easy to strict levels.
- **AP50** → Easy setting; requires only moderate accuracy.
- **AP75** → Stricter; requires very accurate bounding boxes.

How to interpret mAP:

- **Higher mAP = better detection performance.**
- A model with high mAP detects objects reliably and precisely.
- A big gap between AP50 and AP75 means the model finds objects but with weak accuracy in box placement.

Question 5: Compare Detectron2 and TFOD2 in terms of features, performance, and ease of use.

Answer:

Detectron2 (by Facebook/Meta) and TFOD2 (TensorFlow Object Detection API by Google) are two of the most widely used frameworks for object detection. Both are powerful, but they differ in flexibility, speed, and user experience.

1. Feature Comparison

Detectron2

- Built on **PyTorch**
- Supports advanced state-of-the-art models:
 - Mask R-CNN
 - Faster/Cascade R-CNN
 - RetinaNet
 - Panoptic FPN
 - DensePose, PointRend, keypoints
- Highly modular and customizable
- Strong research focus → frequently updated with new models

TFOD2 (TensorFlow Object Detection API)

- Built on **TensorFlow 2.x**
- Supports:
 - SSD
 - EfficientDet
 - CenterNet
 - Faster R-CNN
- Good for production pipelines using TensorFlow ecosystem
- More tutorials and beginner support available

2. Performance Comparison

Detectron2

- Faster training due to optimized PyTorch kernels
- Better performance on segmentation tasks

- Often higher accuracy for advanced models (Mask R-CNN, Cascade R-CNN)
- Better use of GPU resources

TFOD2

- Excellent performance for lightweight models like SSD, MobileNet
- EfficientDet family is highly optimized and fast
- Typically slightly slower to train compared to Detectron2 for heavy models
- Great performance on mobile/edge devices with TensorFlow Lite

3. Ease of Use

Detectron2

- Easy to customize model architecture
- Simple dataset registration system
- YAML-based configurations are cleaner
- Requires understanding of PyTorch but very developer-friendly

TFOD2

- Easier for complete beginners because of many tutorials
- More ready-made pipelines (TFRecords, model configs)
- Harder to customize deep architecture
- TFRecord creation can be tedious for new users

Question 6: Write Python code to install Detectron2 and verify the installation

Answer :

CODE FOR INSTALLING DETECTRON2 :

```
!python -m pip install pyyaml==5.1

import sys, os, distutils.core

!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'))
```

OUTPUT :

Collecting pyyaml==5.1

Downloading PyYAML-5.1.tar.gz (274 kB)

274.2/274.2 kB 8.3 MB/s eta 0:00:00

error: subprocess-exited-with-error

× python setup.py egg_info did not run successfully.

| exit code: 1

↳ See above for output.

note: This error originates from a subprocess, and is likely not a problem with pip.

Preparing metadata (setup.py) ... error

error: metadata-generation-failed

× Encountered error while generating package metadata.

↳ See above for output.

note: This is an issue with the package mentioned above, not pip.

hint: See above for details.

Cloning into 'detectron2'...

remote: Enumerating objects: 15943, done.

remote: Counting objects: 100% (14/14), done.

remote: Compressing objects: 100% (10/10), done.

remote: Total 15943 (delta 5), reused 5 (delta 4), pack-reused 15929 (from 2)

Receiving objects: 100% (15943/15943), 6.70 MiB | 20.29 MiB/s, done.

Resolving deltas: 100% (11337/11337), done.

Ignoring dataclasses: markers 'python_version < "3.7"' don't match your environment

Requirement already satisfied: Pillow>=7.1 in /usr/local/lib/python3.12/dist-packages (11.3.0)

Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)

Requirement already satisfied: pycocotools>=2.0.2 in /usr/local/lib/python3.12/dist-packages (2.0.10)

Requirement already satisfied: termcolor>=1.1 in /usr/local/lib/python3.12/dist-packages (3.2.0)

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/python3.12/dist-packages (0.9.0)

Requirement already satisfied: cloudpickle in /usr/local/lib/python3.12/dist-packages (3.1.2)

Requirement already satisfied: tqdm>4.29.0 in /usr/local/lib/python3.12/dist-packages (4.67.1)

Requirement already satisfied: tensorboard in /usr/local/lib/python3.12/dist-packages (2.19.0)

Collecting fvcore<0.1.6,>=0.1.5

Downloading fvcore-0.1.5.post20221221.tar.gz (50 kB)

— 50.2/50.2 kB 3.1 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)

Requirement already satisfied: omegaconf<2.4,>=2.1 in /usr/local/lib/python3.12/dist-packages (2.3.0)

Collecting hydra-core>=1.1

Downloading hydra_core-1.3.2-py3-none-any.whl.metadata (5.5 kB)

Collecting black

Downloading black-25.11.0-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (85 kB)

— 85.2/85.2 kB 7.4 MB/s eta 0:00:00

Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (25.0)

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)

Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)

Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)

Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)

Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)

Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)

Requirement already satisfied: PyYAML in /usr/local/lib/python3.12/dist-packages (from yacs>=0.1.8) (6.0.3)

Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (1.4.0)

Requirement already satisfied: grpcio>=1.48.2 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (1.76.0)

Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (3.10)

Requirement already satisfied: protobuf!=4.24.0,>=3.19.6 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (5.29.5)

Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (75.2.0)

Requirement already satisfied: six>1.9 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (1.17.0)

Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (0.7.2)

Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard) (3.1.3)

Collecting portalocker (from iopath<0.1.10,>=0.1.7)

Downloading portalocker-3.2.0-py3-none-any.whl.metadata (8.7 kB)

Requirement already satisfied: antlr4-python3-runtime==4.9.* in /usr/local/lib/python3.12/dist-packages (from omegaconf<2.4,>=2.1) (4.9.3)

Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist-packages (from black) (8.3.1)

Collecting mypy-extensions>=0.4.3 (from black)

Downloading mypy_extensions-1.1.0-py3-none-any.whl.metadata (1.1 kB)

Collecting pathspec>=0.9.0 (from black)

Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)

Requirement already satisfied: platformdirs>=2 in /usr/local/lib/python3.12/dist-packages (from black) (4.5.0)

Collecting pytokens<=0.3.0 (from black)

Downloading pytokens-0.3.0-py3-none-any.whl.metadata (2.0 kB)

Requirement already satisfied: typing-extensions<=4.12 in /usr/local/lib/python3.12/dist-packages (from grpcio<=1.48.2->tensorboard) (4.15.0)

Requirement already satisfied: MarkupSafe<=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug<=1.0.1->tensorboard) (3.0.3)

Downloading yacs-0.1.8-py3-none-any.whl (14 kB)

Downloading iopath-0.1.9-py3-none-any.whl (27 kB)

Downloading hydra_core-1.3.2-py3-none-any.whl (154 kB)

154.5/154.5 kB 17.4 MB/s eta 0:00:00

Downloading black-25.11.0-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (1.7 MB)

1.7/1.7 MB 61.9 MB/s eta 0:00:00

Downloading mypy_extensions-1.1.0-py3-none-any.whl (5.0 kB)

Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)

Downloading pytokens-0.3.0-py3-none-any.whl (12 kB)

Downloading portalocker-3.2.0-py3-none-any.whl (22 kB)

Building wheels for collected packages: fvcore

Building wheel for fvcore (setup.py) ... done

Created wheel for fvcore: filename=fvcore-0.1.5.post20221221-py3-none-any.whl size=61397 sha256=7e4067b86750b02e73986055848c0b5b2cd2aa4e6e3b327302d20c042b7ce419

Stored in directory:
/root/.cache/pip/wheels/ed/9f/a5/e4f5b27454ccd4596bd8b62432c7d6b1ca9fa22aef9d70a16a

Successfully built fvcore

Installing collected packages: yacs, pytokens, portalocker, pathspec, mypy-extensions, iopath, hydra-core, black, fvcore

Successfully installed black-25.11.0 fvcore-0.1.5.post20221221 hydra-core-1.3.2 iopath-0.1.9 mypy-extensions-1.1.0 pathspec-0.12.1 portalocker-3.2.0 pytokens-0.3.0 yacs-0.1.8

IMPORTS :

```
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__)
```

OUTPUT :

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Thu_Jun__6_02:18:23_PDT_2024
Cuda compilation tools, release 12.5, V12.5.82
Build cuda_12.5.r12.5/compiler.34385749_0
torch: 2.9 ; cuda: cu126
detectron2: 0.6
```

```
import detectron2

from detectron2.utils.logger import setup_logger
setup_logger()
```

Verify that installation is working by importing important modules

```
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2 import model_zoo
```

Question 7: Annotate a dataset using any tool of your choice and convert the annotations to COCO format for Detectron2.

Answer :

For annotating the dataset I used makesense.ai tool.

Steps:

- Upload images in MakeSense.AI
- Start drawing bounding boxes or polygons for each object in the images.
- After annotating, export the annotations

I have exported Annotations in YOLO format from makesense.ai tool

Code for converting annotations to COCO format

CODE :

```
import os

import json

import xml.etree.ElementTree as ET

from pathlib import Path


def voc_to_coco(voc_dir, output_file):

    # Initialize COCO format structure

    coco_dict = {

        "images": [],

        "annotations": [],

        "categories": []

    }


    categories = {}

    annotation_id = 1

    image_id = 1


    for image_file in Path(voc_dir).rglob('*.jpg'):

        image_info = {

            "id": image_id,

            "file_name": image_file.name,
```

```

        "width": 800, # Set your image width here

        "height": 600 # Set your image height here
    }

    coco_dict["images"].append(image_info)


# Parse the corresponding XML file
xml_file = image_file.with_suffix('.xml')
tree = ET.parse(xml_file)
root = tree.getroot()


for obj in root.findall('object'):

    category_name = obj.find('name').text
    if category_name not in categories:
        categories[category_name] = len(categories) + 1
        coco_dict["categories"].append({
            "id": categories[category_name],
            "name": category_name,
            "supercategory": "none"
        })


    category_id = categories[category_name]
    bndbox = obj.find('bndbox')
    xmin = int(bndbox.find('xmin').text)
    ymin = int(bndbox.find('ymin').text)
    xmax = int(bndbox.find('xmax').text)
    ymax = int(bndbox.find('ymax').text)


    annotation_info = {
        "id": annotation_id,
        "image_id": image_id,
        "category_id": category_id,

```

```

        "bbox": [xmin, ymin, xmax - xmin, ymax - ymin],
        "area": (xmax - xmin) * (ymax - ymin),
        "iscrowd": 0,
        "segmentation": []
    }
    coco_dict["annotations"].append(annotation_info)
    annotation_id += 1

image_id += 1

# Write the COCO JSON file
with open(output_file, 'w') as f:
    json.dump(coco_dict, f, indent=4)

print(f"COCO annotations saved to {output_file}")

# Usage
voc_dir = "path_to_voc_annotations" # Path where Pascal VOC annotations are stored
output_file = "output_coco.json" # Output COCO format JSON file
voc_to_coco(voc_dir, output_file)

```

By executing this code we can have annotations in COCO format in .json file.

Now you can load it to Detectron2 using coco dataset loader.

Register the Dataset in Detectron2

Code :

```

from detectron2.data import DatasetCatalog, MetadataCatalog
from detectron2.data.datasets import register_coco_instances
register_coco_instances(
    "my_dataset_final",
    {},

```

```
    "/content/drive/MyDrive/Dataset/fixed_coco_annotations.json",  
    "/content/drive/MyDrive/Dataset/images"  
)
```

```
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 8
```

This will register dataset in Detectron2

Training with Detectron2

Code :

```
from detectron2.config import get_cfg  
from detectron2.engine import DefaultTrainer, default_setup, launch  
from detectron2 import model_zoo  
from detectron2.data.datasets import register_coco_instances
```

```
cfg = get_cfg()  
cfg.merge_from_file(model_zoo.get_config_file(  
    "COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"  
))
```

```
cfg.DATASETS.TRAIN = ("my_dataset_final",)
```

```
cfg.DATASETS.TEST = ()
```

```
cfg.DATALOADER.NUM_WORKERS = 2
```

```
cfg.SOLVER.IMS_PER_BATCH = 2
```

```
cfg.SOLVER.BASE_LR = 2.5e-4
```

```
cfg.SOLVER.MAX_ITER = 500
```

```
cfg.SOLVER.STEPS = [] # no LR drops
```

```
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 8
```

```
cfg.MODEL.DEVICE = "cuda"
```

```
cfg.OUTPUT_DIR = "./output_frcnn"
```

```
default_setup(cfg, {})
```

```
trainer = DefaultTrainer(cfg)
```

```
trainer.resume_or_load(resume=False)
```

```
print("FINAL NUM_CLASSES:", cfg.MODEL.ROI_HEADS.NUM_CLASSES)
```

```
trainer.train()
```

Output :

```
[12/09 14:40:19 detectron2]: Rank of current process: 0. World size: 1
```

```
[12/09 14:40:20 detectron2]: Environment info:
```

```
sys.platform          linux
```

```
Python                3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
```

```
numpy                 2.0.2
```

```
detectron2            0.6 @/usr/local/lib/python3.12/dist-packages/detectron2
```

```
Compiler              GCC 11.4
```

```
CUDA compiler         CUDA 12.5
```

```
detectron2 arch flags 7.5
```

```
DETECTRON2_ENV_MODULE <not set>
```

```
PyTorch               2.9.0+cu126 @/usr/local/lib/python3.12/dist-packages/torch
```

```
PyTorch debug build   False
```

```
torch._C._GLIBCXX_USE_CXX11_ABI True
```

```
GPU available         Yes
```

```
GPU 0                 Tesla T4 (arch=7.5)
```

```
Driver version        550.54.15
```

```
CUDA_HOME             /usr/local/cuda
```

```
Pillow               11.3.0
```

```
torchvision           0.24.0+cu126 @/usr/local/lib/python3.12/dist-packages/torchvision
```

```
torchvision arch flags 5.0, 6.0, 7.0, 7.5, 8.0, 8.6, 9.0
```

fvcore 0.1.5.post20221221

iopath 0.1.9

cv2 4.12.0

PyTorch built with:

- GCC 13.3

- C++ Version: 201703

- Intel(R) oneAPI Math Kernel Library Version 2024.2-Product Build 20240605 for Intel(R) 64 architecture applications

- Intel(R) MKL-DNN v3.7.1 (Git Hash 8d263e693366ef8db40acc569cc7d8edf644556d)

- OpenMP 201511 (a.k.a. OpenMP 4.5)

- LAPACK is enabled (usually provided by MKL)

- NNPACK is enabled

- CPU capability usage: AVX2

- CUDA Runtime 12.6

- NVCC architecture flags: -gencode;arch=compute_50,code=sm_50;-
gencode;arch=compute_60,code=sm_60;-gencode;arch=compute_70,code=sm_70;-
gencode;arch=compute_75,code=sm_75;-gencode;arch=compute_80,code=sm_80;-
gencode;arch=compute_86,code=sm_86;-gencode;arch=compute_90,code=sm_90

- CuDNN 91.0.2 (built against CUDA 12.9)

- Magma 2.6.1

- Build settings: BLAS_INFO=mkl, BUILD_TYPE=Release,
COMMIT_SHA=0fab3ba44823f257e70ce397d989c8de5e362c1, CUDA_VERSION=12.6,
CUDNN_VERSION=9.10.2, CXX_COMPILER=/opt/rh/gcc-toolset-13/root/usr/bin/c++, CXX_FLAGS= -
fvisibility-inlines-hidden -DUSE_PTHREADPOOL -DNDEBUG -DUSE_KINETO -
DLIBKINETO_NOROCTRACER -DLIBKINETO_NOXPUPTI=ON -DUSE_FBGEMM -
DUSE_PYTORCH_QNNPACK -DUSE_XNNPACK -DSYMBOLICATE_MOBILE_DEBUG_HANDLE -O2 -fPIC -
DC10_NODEPRECATED -Wall -Wextra -Werror=return-type -Werror=non-virtual-dtor -Werror=range-
loop-construct -Werror=bool-operation -Wnarrowing -Wno-missing-field-initializers -Wno-unknown-
pragmas -Wno-unused-parameter -Wno-strict-overflow -Wno-strict-aliasing -Wno-stringop-overflow
-Wsuggest-override -Wno-psabi -Wno-error=old-style-cast -faligned-new -Wno-maybe-uninitialized -
fno-math-errno -fno-trapping-math -Werror=format -Wno-dangling-reference -Wno-error=dangling-
reference -Wno-stringop-overflow, LAPACK_INFO=mkl, PERF_WITH_AVX=1, PERF_WITH_AVX2=1,
TORCH_VERSION=2.9.0, USE_CUDA=ON, USE_CUDNN=ON, USE_CUSPARSELT=1, USE_GFLAGS=OFF,
USE_GLOG=OFF, USE_GLOO=ON, USE_MKL=ON, USE_MKLDNN=ON, USE_MPI=OFF, USE_NCCL=1,
USE_NNPACK=ON, USE_OPENMP=ON, USE_ROCM=OFF, USE_ROCM_KERNEL_ASSERT=OFF,
USE_XCCL=OFF, USE_XPU=OFF,

[12/09 14:40:20 detectron2]: Command line arguments: {}

[12/09 14:40:20 detectron2]: Running with full config:

CUDNN_BENCHMARK: false

DATALOADER:

ASPECT_RATIO_GROUPING: true

FILTER_EMPTY_ANNOTATIONS: true

NUM_WORKERS: 2

REPEAT_SQRT: true

REPEAT_THRESHOLD: 0.0

SAMPLER_TRAIN: TrainingSampler

DATASETS:

PRECOMPUTED_PROPOSAL_TOPK_TEST: 1000

PRECOMPUTED_PROPOSAL_TOPK_TRAIN: 2000

PROPOSAL_FILES_TEST: []

PROPOSAL_FILES_TRAIN: []

TEST: []

TRAIN:

- my_dataset_final

FLOAT32_PRECISION: "

GLOBAL:

HACK: 1.0

INPUT:

CROP:

ENABLED: false

SIZE:

- 0.9

- 0.9

TYPE: relative_range

FORMAT: BGR

MASK_FORMAT: polygon

MAX_SIZE_TEST: 1333

MAX_SIZE_TRAIN: 1333

MIN_SIZE_TEST: 800

MIN_SIZE_TRAIN:

- 640

- 672

- 704

- 736

- 768

- 800

MIN_SIZE_TRAIN_SAMPLING: choice

RANDOM_FLIP: horizontal

MODEL:

ANCHOR_GENERATOR:

ANGLES:

-- -90

- 0

- 90

ASPECT_RATIOS:

-- 0.5

- 1.0

- 2.0

NAME: DefaultAnchorGenerator

OFFSET: 0.0

SIZES:

-- 32

-- 64

-- 128

-- 256

-- 512

BACKBONE:

FREEZE_AT: 2

NAME: build_resnet_fpn_backbone

DEVICE: cuda

FPN:

FUSE_TYPE: sum

IN_FEATURES:

- res2

- res3

- res4

- res5

NORM: "

OUT_CHANNELS: 256

KEYPOINT_ON: false

LOAD_PROPOSALS: false

MASK_ON: false

META_ARCHITECTURE: GeneralizedRCNN

PANOPTIC_FPN:

COMBINE:

ENABLED: true

INSTANCES_CONFIDENCE_THRESH: 0.5

OVERLAP_THRESH: 0.5

STUFF_AREA_LIMIT: 4096

INSTANCE_LOSS_WEIGHT: 1.0

PIXEL_MEAN:

- 103.53

- 116.28

- 123.675

PIXEL_STD:

- 1.0

- 1.0

- 1.0

PROPOSAL_GENERATOR:

MIN_SIZE: 0

NAME: RPN

RESNETS:

DEFORM_MODULATED: false

DEFORM_NUM_GROUPS: 1

DEFORM_ON_PER_STAGE:

- false

- false

- false

- false

DEPTH: 50

NORM: FrozenBN

NUM_GROUPS: 1

OUT_FEATURES:

- res2

- res3

- res4

- res5

RES2_OUT_CHANNELS: 256

RES5_DILATION: 1

STEM_OUT_CHANNELS: 64

STRIDE_IN_1X1: true

WIDTH_PER_GROUP: 64

RETINANET:

BBOX_REG_LOSS_TYPE: smooth_l1

BBOX_REG_WEIGHTS: &id002

- 1.0

- 1.0

- 1.0

- 1.0

FOCAL_LOSS_ALPHA: 0.25

FOCAL_LOSS_GAMMA: 2.0

IN_FEATURES:

- p3

- p4

- p5

- p6

- p7

IOU_LABELS:

- 0

- -1

- 1

IOU_THRESHOLDS:

- 0.4

- 0.5

NMS_THRESH_TEST: 0.5

NORM: "

NUM_CLASSES: 80

NUM_CONVS: 4

PRIOR_PROB: 0.01

SCORE_THRESH_TEST: 0.05

SMOOTH_L1_LOSS_BETA: 0.1

TOPK_CANDIDATES_TEST: 1000

ROI_BOX_CASCADE_HEAD:

BBOX_REG_WEIGHTS:

- &id001

- 10.0

- 10.0

- 5.0

- 5.0

- - 20.0

- 20.0

- 10.0

- 10.0

- - 30.0

- 30.0

- 15.0

- 15.0

IOUS:

- 0.5

- 0.6

- 0.7

ROI_BOX_HEAD:

BBOX_REG_LOSS_TYPE: smooth_l1

BBOX_REG_LOSS_WEIGHT: 1.0

BBOX_REG_WEIGHTS: *id001

CLS_AGNOSTIC_BBOX_REG: false

CONV_DIM: 256

FC_DIM: 1024

FED_LOSS_FREQ_WEIGHT_POWER: 0.5

FED_LOSS_NUM_CLASSES: 50

NAME: FastRCNNConvFCHead

NORM: "

NUM_CONV: 0

NUM_FC: 2

POOLER_RESOLUTION: 7

POOLER_SAMPLING_RATIO: 0

POOLER_TYPE: ROIAlignV2

SMOOTH_L1_BETA: 0.0

TRAIN_ON_PRED_BOXES: false

USE_FED_LOSS: false

USE_SIGMOID_CE: false

ROI_HEADS:

BATCH_SIZE_PER_IMAGE: 512

IN_FEATURES:

- p2

- p3

- p4

- p5

IOU_LABELS:

- 0

- 1

IOU_THRESHOLDS:

- 0.5

NAME: StandardROIHeads

NMS_THRESH_TEST: 0.5

NUM_CLASSES: 8

POSITIVE_FRACTION: 0.25

PROPOSAL_APPEND_GT: true

SCORE_THRESH_TEST: 0.05

ROI_KEYPOINT_HEAD:

CONV_DIMS:

- 512

- 512

- 512

- 512

- 512

- 512

- 512

- 512

LOSS_WEIGHT: 1.0

MIN_KEYPOINTS_PER_IMAGE: 1

NAME: KRCNNConvDeconvUpsampleHead

NORMALIZE_LOSS_BY_VISIBLE_KEYPOINTS: true

NUM_KEYPOINTS: 17

POOLER_RESOLUTION: 14

POOLER_SAMPLING_RATIO: 0

POOLER_TYPE: ROIAlignV2

ROI_MASK_HEAD:

CLS_AGNOSTIC_MASK: false

CONV_DIM: 256

NAME: MaskRCNNConvUpsampleHead

NORM: "

NUM_CONV: 4

POOLER_RESOLUTION: 14

POOLER_SAMPLING_RATIO: 0

POOLER_TYPE: ROIAlignV2

RPN:

BATCH_SIZE_PER_IMAGE: 256

BBOX_REG_LOSS_TYPE: smooth_l1

BBOX_REG_LOSS_WEIGHT: 1.0

BBOX_REG_WEIGHTS: *id002

BOUNDARY_THRESH: -1

CONV_DIMS:

- -1

HEAD_NAME: StandardRPNHead

IN_FEATURES:

- p2

- p3

- p4

- p5

- p6

IOU_LABELS:

- 0

- -1

- 1

IOU_THRESHOLDS:

- 0.3

- 0.7

LOSS_WEIGHT: 1.0

NMS_THRESH: 0.7

POSITIVE_FRACTION: 0.5

POST_NMS_TOPK_TEST: 1000

POST_NMS_TOPK_TRAIN: 1000

PRE_NMS_TOPK_TEST: 1000

PRE_NMS_TOPK_TRAIN: 2000

SMOOTH_L1_BETA: 0.0

SEM_SEG_HEAD:

COMMON_STRIDE: 4

CONVS_DIM: 128

IGNORE_VALUE: 255

IN_FEATURES:

- p2

- p3

- p4

- p5

LOSS_WEIGHT: 1.0

NAME: SemSegFPNHead

NORM: GN

NUM_CLASSES: 54

WEIGHTS: detectron2://ImageNetPretrained/MSRA/R-50.pkl

OUTPUT_DIR: ./output_frcnn

SEED: -1

SOLVER:

AMP:

ENABLED: false

BASE_LR: 0.00025

BASE_LR_END: 0.0

BIAS_LR_FACTOR: 1.0

CHECKPOINT_PERIOD: 5000

CLIP_GRADIENTS:

CLIP_TYPE: value

CLIP_VALUE: 1.0

ENABLED: false

NORM_TYPE: 2.0

GAMMA: 0.1

IMS_PER_BATCH: 2

LR_SCHEDULER_NAME: WarmupMultiStepLR

MAX_ITER: 500

MOMENTUM: 0.9

NESTEROV: false

NUM_DECAYS: 3

REFERENCE_WORLD_SIZE: 0

RESCALE_INTERVAL: false

STEPS: []

WARMUP_FACTOR: 0.001

WARMUP_ITERS: 1000

WARMUP_METHOD: linear

WEIGHT_DECAY: 0.0001

WEIGHT_DECAY_BIAS: null

WEIGHT_DECAY_NORM: 0.0

TEST:

AUG:

ENABLED: false

FLIP: true

MAX_SIZE: 4000

MIN_SIZES:

- 400

- 500
- 600
- 700
- 800
- 900
- 1000
- 1100
- 1200

DETECTIONS_PER_IMAGE: 100

EVAL_PERIOD: 0

EXPECTED_RESULTS: []

KEYPOINT_OKS_SIGMAS: []

PRECISE_BN:

ENABLED: false

NUM_ITER: 200

VERSION: 2

VIS_PERIOD: 0

[12/09 14:40:20 detectron2]: Full config saved to ./output_frcnn/config.yaml

[12/09 14:40:20 d2.utils.env]: Using a generated random seed 20265896

[12/09 14:40:21 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, aligned=True)
      (1): ROIAlign(output_size=(7, 7), spatial_scale=0.125, sampling_ratio=0, aligned=True)
      (2): ROIAlign(output_size=(7, 7), spatial_scale=0.0625, sampling_ratio=0, aligned=True)
      (3): ROIAlign(output_size=(7, 7), spatial_scale=0.03125, sampling_ratio=0, aligned=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=9, bias=True)
  (bbox_pred): Linear(in_features=1024, out_features=32, bias=True)
)
)
)
)

```

WARNING [12/09 14:40:21 d2.data.datasets.coco]:

Category ids in annotations are not in [1, #categories]! We'll apply a mapping for you.

[12/09 14:40:21 d2.data.datasets.coco]: Loaded 10 images in COCO format from
/content/drive/MyDrive/Dataset/fixed_coco_annotations.json

[12/09 14:40:21 d2.data.build]: Removed 0 images with no usable annotations. 10 images left.

[12/09 14:40:21 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in training:
[ResizeShortestEdge(short_edge_length=(640, 672, 704, 736, 768, 800), max_size=1333,
sample_style='choice'), RandomFlip()]

[12/09 14:40:21 d2.data.build]: Using training sampler TrainingSampler

[12/09 14:40:21 d2.data.common]: Serializing the dataset using: <class
'detectron2.data.common._TorchSerializedList'>

[12/09 14:40:21 d2.data.common]: Serializing 10 elements to byte tensors and concatenating them
all ...

[12/09 14:40:21 d2.data.common]: Serialized dataset takes 0.00 MiB

[12/09 14:40:21 d2.data.build]: Making batched data loader with batch_size=2

[12/09 14:40:21 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from
detectron2://ImageNetPretrained/MSRA/R-50.pkl ...

[12/09 14:40:21 fvcore.common.checkpoint]: [Checkpoint] Loading from /root/.torch/iopath_cache/detectron2/ImageNetPretrained/MSRA/R-50.pkl ...

[12/09 14:40:21 d2.checkpoint.c2_model_loading]: Renaming Caffe2 weights

[12/09 14:40:21 d2.checkpoint.c2_model_loading]: Following weights matched with submodule backbone.bottom_up - Total num: 54

WARNING [12/09 14:40:21 fvcore.common.checkpoint]: Some model parameters or buffers are not found in the checkpoint:

backbone.fpn_lateral2.{bias, weight}

backbone.fpn_lateral3.{bias, weight}

backbone.fpn_lateral4.{bias, weight}

backbone.fpn_lateral5.{bias, weight}

backbone.fpn_output2.{bias, weight}

backbone.fpn_output3.{bias, weight}

backbone.fpn_output4.{bias, weight}

backbone.fpn_output5.{bias, weight}

proposal_generator.rpn_head.anchor_deltas.{bias, weight}

proposal_generator.rpn_head.conv.{bias, weight}

proposal_generator.rpn_head.objectness_logits.{bias, weight}

roi_heads.box_head.fc1.{bias, weight}

roi_heads.box_head.fc2.{bias, weight}

roi_heads.box_predictor.bbox_pred.{bias, weight}

roi_heads.box_predictor.cls_score.{bias, weight}

WARNING [12/09 14:40:21 fvcore.common.checkpoint]: The checkpoint state_dict contains keys that are not used by the model:

fc1000.{bias, weight}

stem.conv1.bias

FINAL NUM_CLASSES: 8

[12/09 14:40:21 d2.engine.train_loop]: Starting training from iteration 0

[12/09 14:40:31 d2.utils.events]: eta: 0:03:46 iter: 19 total_loss: 2.975 loss_cls: 2.24 loss_box_reg: 0.01144 loss_rpn_cls: 0.6863 loss_rpn_loc: 0.02043 time: 0.4840 last_time: 0.5131 data_time: 0.0180 last_data_time: 0.0054 lr: 9.7405e-06 max_mem: 3567M

[12/09 14:40:41 d2.utils.events]: eta: 0:03:41 iter: 39 total_loss: 1.298 loss_cls: 0.5988 loss_box_reg: 0.01375 loss_rpn_cls: 0.6744 loss_rpn_loc: 0.02106 time: 0.4831 last_time: 0.5155 data_time: 0.0061 last_data_time: 0.0062 lr: 1.9731e-05 max_mem: 3567M

[12/09 14:40:51 d2.utils.events]: eta: 0:03:36 iter: 59 total_loss: 0.8967 loss_cls: 0.2034
loss_box_reg: 0.02883 loss_rpn_cls: 0.644 loss_rpn_loc: 0.02087 time: 0.4900 last_time: 0.4968
data_time: 0.0070 last_data_time: 0.0064 lr: 2.972e-05 max_mem: 3567M

[12/09 14:41:01 d2.utils.events]: eta: 0:03:32 iter: 79 total_loss: 0.8344 loss_cls: 0.1673
loss_box_reg: 0.0441 loss_rpn_cls: 0.6003 loss_rpn_loc: 0.02155 time: 0.4972 last_time: 0.4182
data_time: 0.0064 last_data_time: 0.0073 lr: 3.9711e-05 max_mem: 3619M

[12/09 14:41:11 d2.utils.events]: eta: 0:03:21 iter: 99 total_loss: 0.8019 loss_cls: 0.1682
loss_box_reg: 0.06875 loss_rpn_cls: 0.5352 loss_rpn_loc: 0.02036 time: 0.4968 last_time: 0.4996
data_time: 0.0065 last_data_time: 0.0039 lr: 4.9701e-05 max_mem: 3620M

[12/09 14:41:21 d2.utils.events]: eta: 0:03:10 iter: 119 total_loss: 0.6766 loss_cls: 0.1397
loss_box_reg: 0.07637 loss_rpn_cls: 0.4436 loss_rpn_loc: 0.01916 time: 0.4978 last_time: 0.4614
data_time: 0.0080 last_data_time: 0.0040 lr: 5.9691e-05 max_mem: 3620M

[12/09 14:41:31 d2.utils.events]: eta: 0:02:59 iter: 139 total_loss: 0.5579 loss_cls: 0.1254
loss_box_reg: 0.06758 loss_rpn_cls: 0.3531 loss_rpn_loc: 0.0205 time: 0.4975 last_time: 0.4905
data_time: 0.0071 last_data_time: 0.0058 lr: 6.9681e-05 max_mem: 3620M

[12/09 14:41:41 d2.utils.events]: eta: 0:02:49 iter: 159 total_loss: 0.4598 loss_cls: 0.1162
loss_box_reg: 0.06489 loss_rpn_cls: 0.2619 loss_rpn_loc: 0.02155 time: 0.4955 last_time: 0.4495
data_time: 0.0062 last_data_time: 0.0077 lr: 7.9671e-05 max_mem: 3620M

[12/09 14:41:50 d2.utils.events]: eta: 0:02:38 iter: 179 total_loss: 0.4154 loss_cls: 0.1222
loss_box_reg: 0.07334 loss_rpn_cls: 0.1887 loss_rpn_loc: 0.01893 time: 0.4950 last_time: 0.5249
data_time: 0.0066 last_data_time: 0.0064 lr: 8.966e-05 max_mem: 3620M

[12/09 14:42:01 d2.utils.events]: eta: 0:02:29 iter: 199 total_loss: 0.4108 loss_cls: 0.1445
loss_box_reg: 0.1004 loss_rpn_cls: 0.1386 loss_rpn_loc: 0.02202 time: 0.4962 last_time: 0.3627
data_time: 0.0079 last_data_time: 0.0076 lr: 9.9651e-05 max_mem: 3620M

[12/09 14:42:10 d2.utils.events]: eta: 0:02:19 iter: 219 total_loss: 0.4204 loss_cls: 0.1551
loss_box_reg: 0.1153 loss_rpn_cls: 0.1145 loss_rpn_loc: 0.01798 time: 0.4954 last_time: 0.5245
data_time: 0.0069 last_data_time: 0.0059 lr: 0.00010964 max_mem: 3621M

[12/09 14:42:20 d2.utils.events]: eta: 0:02:09 iter: 239 total_loss: 0.385 loss_cls: 0.1569
loss_box_reg: 0.1215 loss_rpn_cls: 0.0943 loss_rpn_loc: 0.02084 time: 0.4953 last_time: 0.5050
data_time: 0.0062 last_data_time: 0.0054 lr: 0.00011963 max_mem: 3621M

[12/09 14:42:30 d2.utils.events]: eta: 0:01:59 iter: 259 total_loss: 0.4389 loss_cls: 0.1758
loss_box_reg: 0.1537 loss_rpn_cls: 0.08966 loss_rpn_loc: 0.02368 time: 0.4960 last_time: 0.4720
data_time: 0.0072 last_data_time: 0.0061 lr: 0.00012962 max_mem: 3621M

[12/09 14:42:40 d2.utils.events]: eta: 0:01:49 iter: 279 total_loss: 0.4386 loss_cls: 0.192
loss_box_reg: 0.1617 loss_rpn_cls: 0.07111 loss_rpn_loc: 0.01713 time: 0.4966 last_time: 0.5798
data_time: 0.0063 last_data_time: 0.0056 lr: 0.00013961 max_mem: 3621M

[12/09 14:42:50 d2.utils.events]: eta: 0:01:39 iter: 299 total_loss: 0.4624 loss_cls: 0.2042
loss_box_reg: 0.1863 loss_rpn_cls: 0.06305 loss_rpn_loc: 0.02112 time: 0.4963 last_time: 0.5177
data_time: 0.0075 last_data_time: 0.0083 lr: 0.0001496 max_mem: 3621M

[12/09 14:43:00 d2.utils.events]: eta: 0:01:29 iter: 319 total_loss: 0.527 loss_cls: 0.2257
loss_box_reg: 0.2108 loss_rpn_cls: 0.05718 loss_rpn_loc: 0.02005 time: 0.4963 last_time: 0.5132
data_time: 0.0076 last_data_time: 0.0105 lr: 0.00015959 max_mem: 3621M

[12/09 14:43:10 d2.utils.events]: eta: 0:01:19 iter: 339 total_loss: 0.5532 loss_cls: 0.2054
loss_box_reg: 0.2258 loss_rpn_cls: 0.05123 loss_rpn_loc: 0.02067 time: 0.4964 last_time: 0.3917
data_time: 0.0064 last_data_time: 0.0034 lr: 0.00016958 max_mem: 3621M

[12/09 14:43:20 d2.utils.events]: eta: 0:01:09 iter: 359 total_loss: 0.5377 loss_cls: 0.2277
loss_box_reg: 0.2257 loss_rpn_cls: 0.04799 loss_rpn_loc: 0.01908 time: 0.4965 last_time: 0.5227
data_time: 0.0067 last_data_time: 0.0061 lr: 0.00017957 max_mem: 3621M

[12/09 14:43:30 d2.utils.events]: eta: 0:00:59 iter: 379 total_loss: 0.5442 loss_cls: 0.2447
loss_box_reg: 0.2387 loss_rpn_cls: 0.0416 loss_rpn_loc: 0.01802 time: 0.4965 last_time: 0.3967
data_time: 0.0065 last_data_time: 0.0053 lr: 0.00018956 max_mem: 3621M

[12/09 14:43:40 d2.utils.events]: eta: 0:00:49 iter: 399 total_loss: 0.5084 loss_cls: 0.2161
loss_box_reg: 0.2466 loss_rpn_cls: 0.03484 loss_rpn_loc: 0.01671 time: 0.4957 last_time: 0.5339
data_time: 0.0066 last_data_time: 0.0081 lr: 0.00019955 max_mem: 3621M

[12/09 14:43:50 d2.utils.events]: eta: 0:00:39 iter: 419 total_loss: 0.5084 loss_cls: 0.2043
loss_box_reg: 0.2508 loss_rpn_cls: 0.0288 loss_rpn_loc: 0.0163 time: 0.4957 last_time: 0.4866
data_time: 0.0066 last_data_time: 0.0053 lr: 0.00020954 max_mem: 3621M

[12/09 14:43:59 d2.utils.events]: eta: 0:00:29 iter: 439 total_loss: 0.4849 loss_cls: 0.1915
loss_box_reg: 0.2404 loss_rpn_cls: 0.03094 loss_rpn_loc: 0.01564 time: 0.4950 last_time: 0.4394
data_time: 0.0071 last_data_time: 0.0055 lr: 0.00021953 max_mem: 3621M

[12/09 14:44:09 d2.utils.events]: eta: 0:00:19 iter: 459 total_loss: 0.4663 loss_cls: 0.1867
loss_box_reg: 0.2394 loss_rpn_cls: 0.02546 loss_rpn_loc: 0.01737 time: 0.4952 last_time: 0.5910
data_time: 0.0063 last_data_time: 0.0091 lr: 0.00022952 max_mem: 3621M

[12/09 14:44:19 d2.utils.events]: eta: 0:00:09 iter: 479 total_loss: 0.4854 loss_cls: 0.2068
loss_box_reg: 0.2575 loss_rpn_cls: 0.02275 loss_rpn_loc: 0.01892 time: 0.4947 last_time: 0.4835
data_time: 0.0061 last_data_time: 0.0057 lr: 0.00023951 max_mem: 3621M

[12/09 14:44:29 fvcore.common.checkpoint]: Saving checkpoint to ./output_frcnn/model_final.pth

[12/09 14:44:29 d2.utils.events]: eta: 0:00:00 iter: 499 total_loss: 0.4257 loss_cls: 0.1594
loss_box_reg: 0.2384 loss_rpn_cls: 0.01939 loss_rpn_loc: 0.0178 time: 0.4948 last_time: 0.5379
data_time: 0.0081 last_data_time: 0.0063 lr: 0.0002495 max_mem: 3621M

[12/09 14:44:30 d2.engine.hooks]: Overall training speed: 498 iterations in 0:04:06 (0.4948 s / it)

[12/09 14:44:30 d2.engine.hooks]: Total training time: 0:04:08 (0:00:01 on hooks)

Question 8 : Write a script to download pretrained weights and configure paths for training in Detectron2.

Answer:

I chose Faster R-CNN R50-FPN from Detectron2 model zoo for downloading pretrained weights

Code:

```
import os

import urllib.request

from detectron2.config import get_cfg

from detectron2 import model_zoo


# MODEL FROM DETECTRON2 MODEL ZOO

# Faster R-CNN R50-FPN

MODEL_ZOO_CONFIG = "COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"

MODEL_URL = model_zoo.get_checkpoint_url(MODEL_ZOO_CONFIG) # pretrained weights URL


# DEFINE SAVE LOCATIONS

OUTPUT_DIR = "/content/drive/MyDrive/COCO 128.v2-640x640.yolov11/output"

WEIGHTS_DIR = "/content/drive/MyDrive/COCO 128.v2-640x640.yolov11/weights"

PRETRAINED_WEIGHT_PATH = os.path.join(WEIGHTS_DIR, MODEL_URL.split("/")[-1])


os.makedirs(OUTPUT_DIR, exist_ok=True)

os.makedirs(WEIGHTS_DIR, exist_ok=True)


# DOWNLOAD PRETRAINED WEIGHTS

if not os.path.exists(PRETRAINED_WEIGHT_PATH):

    print(f"Downloading pretrained weights from:\n{MODEL_URL}")

    urllib.request.urlretrieve(MODEL_URL, PRETRAINED_WEIGHT_PATH)

    print(f"Downloaded to: {PRETRAINED_WEIGHT_PATH}")

else:

    print(f"Weights already exist at: {PRETRAINED_WEIGHT_PATH}")


# CONFIGURE DETECTRON2
```

```

cfg = get_cfg()

cfg.merge_from_file(model_zoo.get_config_file(MODEL_ZOO_CONFIG))

#DATASET

cfg.DATASETS.TRAIN = ("/content/drive/MyDrive/COCO 128.v2-640x640.yolov11/train",)
cfg.DATASETS.TEST = ("/content/drive/MyDrive/COCO 128.v2-640x640.yolov11/valid",)

# Number of classes in my dataset

cfg.MODEL.ROI_HEADS.NUM_CLASSES = 3

# Set pretrained weights

cfg.MODEL.WEIGHTS = PRETRAINED_WEIGHT_PATH

# Training parameters

cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 5000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128

cfg.OUTPUT_DIR = OUTPUT_DIR

# Save config

config_path = os.path.join(OUTPUT_DIR, "config.yaml")
with open(config_path, "w") as f:
    f.write(cfg.dump())

print(f"Configuration written to: {config_path}")

```

Output :

Downloading pretrained weights from:

https://dl.fbaipublicfiles.com/detectron2/COCO-Detection/faster_rcnn_R_50_FPN_3x/137849458/model_final_280758.pkl

Downloaded to: /content/drive/MyDrive/COCO 128.v2-640x640.yolov11/weights/model_final_280758.pkl

Configuration written to: /content/drive/MyDrive/COCO 128.v2-640x640.yolov11/output/config.yaml

Question 9: Show the steps and code to run inference using a trained Detectron2 model on a new image.

Answer :

- Install Detectron2
- Prepare your environment

```
import cv2

import torch

from detectron2.config import get_cfg

from detectron2.engine import DefaultPredictor

from detectron2.utils.visualizer import Visualizer, ColorMode

from detectron2.data import MetadataCatalog
```
- Load the trained model + confi

```
cfg = get_cfg()

# Load the config used during training
cfg.merge_from_file("configs/my_config.yaml")

# Path to trained weights
cfg.MODEL.WEIGHTS = "output/model_final.pth"

# Set inference threshold (optional)
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5

# Force CPU if needed
# cfg.MODEL.DEVICE = "cpu"

predictor = DefaultPredictor(cfg)
```
- Load image

```
image_path = "input.jpg"
image = cv2.imread(image_path)
```
- Run Inference

```
outputs = predictor(image)
```

```
print(outputs)
```

- Visualise the predications
 - Save the output
- ```
result = out.get_image()[:, :, :-1]
cv2.imwrite("prediction_output.jpg", result)
```

Code :

```
import cv2
```

```
import torch
```

```
from detectron2.config import get_cfg
```

```
from detectron2.engine import DefaultPredictor
```

```
from detectron2.utils.visualizer import Visualizer, ColorMode
```

```
from detectron2.data import MetadataCatalog
```

# 1. CONFIGURE DETECTRON2

```
cfg = get_cfg()
```

```
cfg.merge_from_file("/content/drive/MyDrive/Dataset/output/config.yaml") # path to your saved
config
```

```
cfg.MODEL.WEIGHTS = "/content/drive/MyDrive/Dataset/weights/model_final_280758.pkl" # path
to trained model
```

# 2. FORCE CPU USAGE

```
cfg.MODEL.DEVICE = "cpu" # <- IMPORTANT: use CPU only
```

```
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5 # confidence threshold
```

# Create predictor

```
predictor = DefaultPredictor(cfg)
```

# 3. LOAD IMAGE

```
image_path = "/content/drive/MyDrive/Dataset/Val/images/5.jpg" # path to your input image
```

```
image = cv2.imread(image_path)
```

if image is None:

```

 raise FileNotFoundError(f"Image not found: {image_path}")

4. RUN INFERENCE

outputs = predictor(image)

print("Detected classes:", outputs["instances"].pred_classes)
print("Scores:", outputs["instances"].scores)

5. VISUALIZE RESULTS

dataset_name = cfg.DATASETS.TRAIN[0] # dataset name used during training
metadata = MetadataCatalog.get(dataset_name)

visualizer = Visualizer(
 image[:, :, ::-1], # BGR -> RGB
 metadata=metadata,
 scale=1.2,
 instance_mode=ColorMode.IMAGE
)

vis_output = visualizer.draw_instance_predictions(outputs["instances"].to("cpu"))
result_image = vis_output.get_image()[:, :, ::-1] # RGB -> BGR

6. SAVE VISUALIZED IMAGE

out_path = "/content/drive/MyDrive/Dataset/output.jpg"
cv2.imwrite(out_path, result_image)
print(f"Saved visualized result to {out_path}")

```

Output:

[12/10 17:35:19 d2.checkpoint.detection\_checkpoint]: [DetectionCheckpointer] Loading from /content/drive/MyDrive/Dataset/weights/model\_final\_280758.pkl ...

roi\_heads.box\_predictor.bbox\_pred.{bias, weight}

roi\_heads.box\_predictor.cls\_score.{bias, weight}

Detected classes: tensor([3], dtype=torch.int64)

Scores: tensor([3])

Saved visualized result to /content/drive/MyDrive/Dataset/output.jpg

**Question 10: You are assigned to build a wildlife monitoring system to detect and track different animal species in a forest using Detectron2. Describe the end-to-end pipeline from data collection to deploying the model, and how you would handle challenges like occlusion or nighttime detection.**

**Answer :**

This is an end to end pipeline for building a **wildlife detection and tracking system using Detectron2**.

- **Problem Definition and System Scope**

**Goal:** Detect, classify, and track multiple wildlife species (e.g., deer, boar, elephants, predators) in forest environments.

**Outputs:**

- Bounding boxes and/or instance masks
- Species labels
- Object IDs

**Constraints:**

- Variable lighting
- Heavy occlusion
- Motion blur
- Limited labeled data

- **Data Collection**

Animalas 151 dataset

- **Data Annotation**

Annotation Types

- **Bounding boxes** → faster, good for detection
- **Instance segmentation masks** → better under occlusion

- **Dataset Structure (COCO Format)**

Detectron2 works best with **COCO-style JSON**:

- **Data Preprocessing & Augmentation**

- Image Preprocessing
- Data Augmentation

- **Model Selection (Detectron2)**

- **Model Architecture**

**Mask R-CNN** (instance segmentation handles occlusion better)

Backbone: ResNet-50 / ResNet-101 + FPN

Pretrained Weights

Start from COCO-pretrained weights

- **Training Pipeline**

Register custom dataset

Adjust:

- NUM\_CLASSES
- Learning rate (lower for fine-tuning)
- Batch size (GPU-dependent)
- Anchor sizes (important for small animals)
- Freeze backbone initially
- Gradually unfreeze

- **Handling Key Challenges**

**A. Occlusion**

Animals hidden by vegetation

Partial visibility

**Solutions:**

Instance Segmentation

- Mask R-CNN > Faster R-CNN

Occlusion-aware Augmentation

- Random crops
- CutOut

Multi-frame reasoning

- Use tracking to recover identity

Multi-camera fusion

- Same animal seen from different angles

## **B.Nighttime Detection**

### **Problems:**

- Low contrast
- Infrared images differ from RGB

### **Solutions:**

#### **Thermal / IR Cameras**

- Train separate model or domain-adapted model

#### **Domain Adaptation**

- Fine-tune with nighttime data

#### **Two-Stream Model**

- RGB stream + IR stream

#### **Preprocessing**

- Histogram equalization
- Noise reduction

#### **Separate Day/Night Models**

- Switch based on timestamp or brightness

## **Code :**

### **For COCO conversion**

```
import os, json
import cv2
from pathlib import Path

DATA_DIR = "/content/drive/MyDrive/Zoo"
out = {"images": [], "annotations": [], "categories": []}

create a small subset of classes
classes = sorted(os.listdir(DATA_DIR))[:10] # choose first 10 classes
for idx, cat in enumerate(classes):
```

```
out["categories"].append({"id": idx, "name": cat})
```

```
img_id = 0
```

```
ann_id = 0
```

```
for cat_id, cat in enumerate(classes):
```

```
 folder = Path(DATA_DIR) / cat
```

```
 for img_file in folder.glob("*.jpg"):
```

```
 image = cv2.imread(str(img_file))
```

```
 if image is None: continue
```

```
 h, w = image.shape[:2]
```

```
 out["images"].append({
```

```
 "file_name": str(img_file),
```

```
 "height": h,
```

```
 "width": w,
```

```
 "id": img_id
```

```
 })
```

```
 # auto full-image box
```

```
 out["annotations"].append({
```

```
 "id": ann_id,
```

```
 "image_id": img_id,
```

```
 "category_id": cat_id,
```

```
 "bbox": [0, 0, w, h],
```

```
 "area": w*h,
```

```
 "iscrowd": 0
```

```
 })
```

```
 img_id += 1
```

```
 ann_id += 1
```

```
with open("animals141_detection.json", "w") as f:
```

```
json.dump(out, f, indent=2)
```

```
print("Generated detection JSON!")
```

### **OUTPUT :**

Generated detection JSON!

### **Dataset Registration**

```
from detectron2.data.datasets import register_coco_instances

register_coco_instances(
 "animals141_det",
 {},
 "animals141_detection.json",
 "animals141"
)
```

### **Model Train**

```
import detectron2

from detectron2.config import get_cfg
from detectron2 import model_zoo
from detectron2.engine import DefaultTrainer

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file(
 "COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"
))
cfg.DATASETS.TRAIN = ("animals141_det",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 4
cfg.SOLVER.IMS_PER_BATCH = 2
```

```

cfg.SOLVER.BASE_LR = 0.00025

cfg.SOLVER.MAX_ITER = 500

cfg.MODEL.ROI_HEADS.NUM_CLASSES = 10 # Corrected from 30 to 10 to match the dataset

cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(
 "COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"
)

cfg.OUTPUT_DIR = "/content/drive/MyDrive/Zoo/output/output_animals141"

trainer = DefaultTrainer(cfg)

trainer.resume_or_load(resume=False)

trainer.train()

```

## OUTPUT :

```

roi_heads.box_predictor.bbox_pred.{bias, weight}

roi_heads.box_predictor.cls_score.{bias, weight}

[12/11 10:14:26 d2.engine.train_loop]: Starting training from iteration 0

[12/11 10:14:32 d2.utils.events]: eta: 0:02:29 iter: 19 total_loss: 2.641 loss_cls: 2.52 loss_box_reg:
0.07662 loss_rpn_cls: 0.005229 loss_rpn_loc: 0.0342 time: 0.3182 last_time: 0.3049 data_time:
0.0185 last_data_time: 0.0065 lr: 9.7405e-06 max_mem: 2382M

[12/11 10:14:39 d2.utils.events]: eta: 0:02:26 iter: 39 total_loss: 2.277 loss_cls: 2.164
loss_box_reg: 0.07016 loss_rpn_cls: 0.00481 loss_rpn_loc: 0.02578 time: 0.3173 last_time:
0.3113 data_time: 0.0061 last_data_time: 0.0051 lr: 1.9731e-05 max_mem: 2382M

[12/11 10:14:45 d2.utils.events]: eta: 0:02:20 iter: 59 total_loss: 1.42 loss_cls: 1.324 loss_box_reg:
0.06864 loss_rpn_cls: 0.004785 loss_rpn_loc: 0.02915 time: 0.3199 last_time: 0.3210 data_time:
0.0054 last_data_time: 0.0055 lr: 2.972e-05 max_mem: 2382M

[12/11 10:14:52 d2.utils.events]: eta: 0:02:14 iter: 79 total_loss: 0.5663 loss_cls: 0.4571
loss_box_reg: 0.07447 loss_rpn_cls: 0.007792 loss_rpn_loc: 0.02808 time: 0.3236 last_time:
0.2778 data_time: 0.0064 last_data_time: 0.0058 lr: 3.9711e-05 max_mem: 2383M

[12/11 10:14:58 d2.utils.events]: eta: 0:02:08 iter: 99 total_loss: 0.2384 loss_cls: 0.1525
loss_box_reg: 0.07523 loss_rpn_cls: 0.003457 loss_rpn_loc: 0.02363 time: 0.3254 last_time:
0.3723 data_time: 0.0052 last_data_time: 0.0063 lr: 4.9701e-05 max_mem: 2383M

[12/11 10:15:05 d2.utils.events]: eta: 0:02:03 iter: 119 total_loss: 0.197 loss_cls: 0.0992
loss_box_reg: 0.07299 loss_rpn_cls: 0.00522 loss_rpn_loc: 0.02151 time: 0.3280 last_time:
0.2933 data_time: 0.0067 last_data_time: 0.0079 lr: 5.9691e-05 max_mem: 2383M

```

[12/11 10:15:12 d2.utils.events]: eta: 0:01:57 iter: 139 total\_loss: 0.2103 loss\_cls: 0.09845  
loss\_box\_reg: 0.08498 loss\_rpn\_cls: 0.003108 loss\_rpn\_loc: 0.0222 time: 0.3288 last\_time:  
0.3710 data\_time: 0.0056 last\_data\_time: 0.0052 lr: 6.9681e-05 max\_mem: 2383M

[12/11 10:15:19 d2.utils.events]: eta: 0:01:51 iter: 159 total\_loss: 0.195 loss\_cls: 0.09013  
loss\_box\_reg: 0.08025 loss\_rpn\_cls: 0.002197 loss\_rpn\_loc: 0.02127 time: 0.3303 last\_time:  
0.3221 data\_time: 0.0058 last\_data\_time: 0.0054 lr: 7.9671e-05 max\_mem: 2383M

[12/11 10:15:26 d2.utils.events]: eta: 0:01:45 iter: 179 total\_loss: 0.1816 loss\_cls: 0.08201  
loss\_box\_reg: 0.07383 loss\_rpn\_cls: 0.002429 loss\_rpn\_loc: 0.01989 time: 0.3315 last\_time:  
0.3334 data\_time: 0.0068 last\_data\_time: 0.0062 lr: 8.966e-05 max\_mem: 2383M

[12/11 10:15:33 d2.utils.events]: eta: 0:01:39 iter: 199 total\_loss: 0.1816 loss\_cls: 0.07921  
loss\_box\_reg: 0.0757 loss\_rpn\_cls: 0.003534 loss\_rpn\_loc: 0.02149 time: 0.3338 last\_time:  
0.3774 data\_time: 0.0063 last\_data\_time: 0.0057 lr: 9.9651e-05 max\_mem: 2383M

[12/11 10:15:39 d2.utils.events]: eta: 0:01:32 iter: 219 total\_loss: 0.1692 loss\_cls: 0.07479  
loss\_box\_reg: 0.07653 loss\_rpn\_cls: 0.0009282 loss\_rpn\_loc: 0.01598 time: 0.3325 last\_time:  
0.3764 data\_time: 0.0053 last\_data\_time: 0.0047 lr: 0.00010964 max\_mem: 2383M

[12/11 10:15:46 d2.utils.events]: eta: 0:01:25 iter: 239 total\_loss: 0.1633 loss\_cls: 0.06993  
loss\_box\_reg: 0.07196 loss\_rpn\_cls: 0.0008387 loss\_rpn\_loc: 0.01599 time: 0.3332 last\_time:  
0.3066 data\_time: 0.0067 last\_data\_time: 0.0048 lr: 0.00011963 max\_mem: 2383M

[12/11 10:15:53 d2.utils.events]: eta: 0:01:19 iter: 259 total\_loss: 0.1669 loss\_cls: 0.06859  
loss\_box\_reg: 0.07382 loss\_rpn\_cls: 0.001848 loss\_rpn\_loc: 0.01961 time: 0.3338 last\_time:  
0.3310 data\_time: 0.0053 last\_data\_time: 0.0052 lr: 0.00012962 max\_mem: 2383M

[12/11 10:16:00 d2.utils.events]: eta: 0:01:13 iter: 279 total\_loss: 0.1548 loss\_cls: 0.06408  
loss\_box\_reg: 0.07051 loss\_rpn\_cls: 0.001219 loss\_rpn\_loc: 0.01659 time: 0.3352 last\_time:  
0.3399 data\_time: 0.0064 last\_data\_time: 0.0071 lr: 0.00013961 max\_mem: 2383M

[12/11 10:16:06 d2.utils.events]: eta: 0:01:06 iter: 299 total\_loss: 0.1551 loss\_cls: 0.06393  
loss\_box\_reg: 0.07471 loss\_rpn\_cls: 0.0008734 loss\_rpn\_loc: 0.01586 time: 0.3350 last\_time:  
0.3843 data\_time: 0.0057 last\_data\_time: 0.0056 lr: 0.0001496 max\_mem: 2383M

[12/11 10:16:13 d2.utils.events]: eta: 0:01:00 iter: 319 total\_loss: 0.1479 loss\_cls: 0.06097  
loss\_box\_reg: 0.06966 loss\_rpn\_cls: 0.0005414 loss\_rpn\_loc: 0.01652 time: 0.3351 last\_time:  
0.3449 data\_time: 0.0054 last\_data\_time: 0.0053 lr: 0.00015959 max\_mem: 2383M

[12/11 10:16:20 d2.utils.events]: eta: 0:00:53 iter: 339 total\_loss: 0.1452 loss\_cls: 0.05892  
loss\_box\_reg: 0.07221 loss\_rpn\_cls: 0.0006309 loss\_rpn\_loc: 0.01397 time: 0.3348 last\_time:  
0.3408 data\_time: 0.0058 last\_data\_time: 0.0051 lr: 0.00016958 max\_mem: 2383M

[12/11 10:16:27 d2.utils.events]: eta: 0:00:46 iter: 359 total\_loss: 0.1404 loss\_cls: 0.0559  
loss\_box\_reg: 0.06902 loss\_rpn\_cls: 0.0005295 loss\_rpn\_loc: 0.01294 time: 0.3351 last\_time:  
0.3410 data\_time: 0.0063 last\_data\_time: 0.0060 lr: 0.00017957 max\_mem: 2385M

[12/11 10:16:33 d2.utils.events]: eta: 0:00:40 iter: 379 total\_loss: 0.1446 loss\_cls: 0.05828  
loss\_box\_reg: 0.07195 loss\_rpn\_cls: 0.0004184 loss\_rpn\_loc: 0.0133 time: 0.3350 last\_time:  
0.3322 data\_time: 0.0057 last\_data\_time: 0.0060 lr: 0.00018956 max\_mem: 2385M

[12/11 10:16:40 d2.utils.events]: eta: 0:00:33 iter: 399 total\_loss: 0.1378 loss\_cls: 0.0555  
loss\_box\_reg: 0.06646 loss\_rpn\_cls: 0.001035 loss\_rpn\_loc: 0.01501 time: 0.3360 last\_time:  
0.3026 data\_time: 0.0058 last\_data\_time: 0.0053 lr: 0.00019955 max\_mem: 2385M

[12/11 10:16:47 d2.utils.events]: eta: 0:00:26 iter: 419 total\_loss: 0.136 loss\_cls: 0.05484  
loss\_box\_reg: 0.06724 loss\_rpn\_cls: 0.0005201 loss\_rpn\_loc: 0.01298 time: 0.3364 last\_time:  
0.3874 data\_time: 0.0051 last\_data\_time: 0.0051 lr: 0.00020954 max\_mem: 2385M

[12/11 10:16:54 d2.utils.events]: eta: 0:00:20 iter: 439 total\_loss: 0.1478 loss\_cls: 0.05605  
loss\_box\_reg: 0.07357 loss\_rpn\_cls: 0.0006399 loss\_rpn\_loc: 0.01326 time: 0.3370 last\_time:  
0.3368 data\_time: 0.0056 last\_data\_time: 0.0048 lr: 0.00021953 max\_mem: 2385M

[12/11 10:17:01 d2.utils.events]: eta: 0:00:13 iter: 459 total\_loss: 0.1329 loss\_cls: 0.05354  
loss\_box\_reg: 0.06561 loss\_rpn\_cls: 0.0003979 loss\_rpn\_loc: 0.01358 time: 0.3377 last\_time:  
0.3962 data\_time: 0.0060 last\_data\_time: 0.0092 lr: 0.00022952 max\_mem: 2385M

[12/11 10:17:08 d2.utils.events]: eta: 0:00:06 iter: 479 total\_loss: 0.1345 loss\_cls: 0.05481  
loss\_box\_reg: 0.06885 loss\_rpn\_cls: 0.0006563 loss\_rpn\_loc: 0.01314 time: 0.3378 last\_time:  
0.2918 data\_time: 0.0054 last\_data\_time: 0.0048 lr: 0.00023951 max\_mem: 2385M

[12/11 10:17:17 d2.utils.events]: eta: 0:00:00 iter: 499 total\_loss: 0.1325 loss\_cls: 0.0513  
loss\_box\_reg: 0.06613 loss\_rpn\_cls: 0.0006371 loss\_rpn\_loc: 0.01286 time: 0.3381 last\_time:  
0.3873 data\_time: 0.0061 last\_data\_time: 0.0055 lr: 0.0002495 max\_mem: 2385M

[12/11 10:17:17 d2.engine.hooks]: Overall training speed: 498 iterations in 0:02:48 (0.3381 s / it)

[12/11 10:17:17 d2.engine.hooks]: Total training time: 0:02:50 (0:00:02 on hooks)

## **Inference and Visualisation**

```
import cv2

from detectron2.engine import DefaultPredictor
from detectron2.utils.visualizer import Visualizer
import matplotlib.pyplot as plt

cfg.MODEL.WEIGHTS = "/content/drive/MyDrive/Zoo/output/output_animals141/model_final.pth"
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
predictor = DefaultPredictor(cfg)

img = cv2.imread("/content/drive/MyDrive/Zoo/correlolphus-ciliatus/correlolphus-
ciliatus_11_fc8510c2.jpg")

outputs = predictor(img)

v = Visualizer(img[:, :, ::-1])
```

```
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
```

```
plt.imshow(out.get_image())
```

```
plt.axis("off")
```

```
plt.show()
```

### OUTPUT :

[12/11 10:08:03 d2.checkpoint.detection\_checkpoint]: [DetectionCheckpoint] Loading from /content/drive/MyDrive/Zoo/output/output\_animals141/model\_final.pth ...

