Problem: building an algorithm to detect a visual signal for pneumonia in medical images. Specifically, this algorithm needs to automatically locate lung opacities on chest radiographs.

We have to predict bounding boxes on CXR images that are locating suspicious areas. It is aboslutely an object detection task.

We have 3 classes in this dataset but only the **Lung Opacity** class is important for us because the other two do not have boxes.

Our chosen model for this task is a state-of-the-art architecture called Detectron2.

```
## install pydicom
!pip install pydicom
```

```
    Collecting pydicom
      Downloading pydicom-2.2.2-py3-none-any.whl (2.0 MB)
         |████████████████████████████████| 2.0 MB 5.4 MB/s
    Installing collected packages: pydicom
    Successfully installed pydicom-2.2.2
```

```
!pip install pyyaml==5.1

import torch
TORCH_VERSION = ".".join(torch.__version__.split(".")[:2])
CUDA_VERSION = torch.__version__.split("+")[-1]
print("torch: ", TORCH_VERSION, "; cuda: ", CUDA_VERSION)
# Install detectron2 that matches the above pytorch version

!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/$CUDA_VERSION/torch$TORCH_VERSION/index.html
```

```
    Requirement already satisfied: black==21.4b2 in /usr/local/lib/python3.7/dist-packages (from detectron2) (21.4b2)
    Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from detectron2) (0.8.9)
    Requirement already satisfied: tensorboard in /usr/local/lib/python3.7/dist-packages (from detectron2) (2.7.0)
    Requirement already satisfied: omegaconf>=2.1 in /usr/local/lib/python3.7/dist-packages (from detectron2) (2.1.1)
    Requirement already satisfied: Pillow>=7.1 in /usr/local/lib/python3.7/dist-packages (from detectron2) (7.1.2)
    Requirement already satisfied: termcolor>=1.1 in /usr/local/lib/python3.7/dist-packages (from detectron2) (1.1.0)
    Requirement already satisfied: fvcore<0.1.6,>=0.1.5 in /usr/local/lib/python3.7/dist-packages (from detectron2) (0.1.5.post20211023)
    Requirement already satisfied: tqdm>4.29.0 in /usr/local/lib/python3.7/dist-packages (from detectron2) (4.62.3)
    Requirement already satisfied: pycocotools>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from detectron2) (2.0.3)
    Requirement already satisfied: pydot in /usr/local/lib/python3.7/dist-packages (from detectron2) (1.3.0)
    Requirement already satisfied: hydra-core>=1.1 in /usr/local/lib/python3.7/dist-packages (from detectron2) (1.1.1)
    Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-packages (from detectron2) (1.3.0)
    Requirement already satisfied: yacs>=0.1.8 in /usr/local/lib/python3.7/dist-packages (from detectron2) (0.1.8)
    Requirement already satisfied: regex>=2020.1.8 in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (2021.11.10
    Requirement already satisfied: mypy-extensions>=0.4.3 in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (0.4
    Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (3
    Requirement already satisfied: typed-ast>=1.4.2 in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (1.5.1)
    Requirement already satisfied: click>=7.1.2 in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (7.1.2)
    Requirement already satisfied: toml>=0.10.1 in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (0.10.2)
    Requirement already satisfied: appdirs in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (1.4.4)
    Requirement already satisfied: pathspec<1,>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from black==21.4b2->detectron2) (0.9.0)
    Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from fvcore<0.1.6,>=0.1.5->detectron2) (1.19.5)
    Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from fvcore<0.1.6,>=0.1.5->detectron2) (5.1)
    Requirement already satisfied: importlib-resources in /usr/local/lib/python3.7/dist-packages (from hydra-core>=1.1->detectron2) (5.4.
    Requirement already satisfied: antlr4-python3-runtime==4.8 in /usr/local/lib/python3.7/dist-packages (from hydra-core>=1.1->detectron
    Requirement already satisfied: portalocker in /usr/local/lib/python3.7/dist-packages (from iopath<0.1.10,>=0.1.7->detectron2) (2.3.2)
    Requirement already satisfied: setuptools>=18.0 in /usr/local/lib/python3.7/dist-packages (from pycocotools>=2.0.2->detectron2) (57.4
    Requirement already satisfied: cython>=0.27.3 in /usr/local/lib/python3.7/dist-packages (from pycocotools>=2.0.2->detectron2) (0.29.2
    Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->detectron2) (2.8.2)
    Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->d
    Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->detectron2) (0.11.0)
    Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->detectron2) (1.3.2)
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->detectron2)
    Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.7/dist-packages (from importlib-resources->hydra-core>=1.1->dete
    Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectro
    Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectron2) (1.0.1)
    Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectron2) (1.42.0)
    Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectron2) (0.12.0)
    Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectron2)
    Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectron2) (0.37.0)
    Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectron2) (1.35.0
    Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard->detectron2) (3.17.3)
    Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard->det
    Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-aut
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard->detectr
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard->d
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard->de
    Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-
```

```python
# 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
```

```python
import os
import sys
import pandas as pd
import pickle
import sys
from collections import defaultdict
import math
import random
import skimage.io
import skimage.transform
from skimage.transform import SimilarityTransform, AffineTransform
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec
from tqdm.auto import tqdm, trange
import pydicom
import cv2
from PIL import Image
import torch
from torch.utils.data import Dataset, DataLoader
from io import BytesIO
from io import StringIO
import scipy.misc
from torch import nn, optim
```

1. Load Data

```python
from google.colab import files
## Upload cookies.txt
files.upload()
```

```
Choose Files No file chosen          Upload widget is only available when the cell has
been executed in the current browser session. Please rerun this cell to enable.
Saving cookies.txt to cookies.txt
{'cookies.txt': b'https://www.kaggle.com/GCP-Credits-Form-RSNA-Pneumonia\r'}
```

```
!wget -x --load-cookies ./cookies.txt "https://storage.googleapis.com/kaggle-competitions-data/kaggle-v2/10338/862042/bundle/archive.zip?Goog
```

```
--2021-12-08 05:48:05--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-v2/10338/862042/bundle/archive.zip?GoogleAccessI
Resolving storage.googleapis.com (storage.googleapis.com)... 173.194.194.128, 173.194.195.128, 173.194.197.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|173.194.194.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3932287530 (3.7G) [application/zip]
Saving to: ‘data.zip’

data.zip            100%[===================>]   3.66G  46.8MB/s    in 60s

2021-12-08 05:49:05 (62.6 MB/s) - ‘data.zip’ saved [3932287530/3932287530]
```

```
## Unzip data
!unzip data.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: stage_2_train_images/d5231546-354e-4071-9af1-6644beabfd86.dcm
  inflating: stage_2_train_images/d5252a78-3ea1-48e9-9ffb-e7535be3ce80.dcm
  inflating: stage_2_train_images/d525eafb-8908-45fd-a942-48d07c435487.dcm
  inflating: stage_2_train_images/d5265640-17db-4880-866d-d2952e32941c.dcm
  inflating: stage_2_train_images/d5277276-f8f8-40e9-b8e1-791cf5d96ac0.dcm
  inflating: stage_2_train_images/d528d9e9-647a-4e2e-a16c-bd5e32a5bbf5.dcm
  inflating: stage_2_train_images/d5293a3e-f050-4b98-8bbf-1f40e25bced5.dcm
  inflating: stage_2_train_images/d52cbb5a-1d0a-457d-8c72-0f7aeec21ca7.dcm
  inflating: stage_2_train_images/d52ce67b-be7c-4349-8dc4-38562928d208.dcm
  inflating: stage_2_train_images/d535a3c8-c4a4-4856-b5cd-17f6332eac8b.dcm
  inflating: stage_2_train_images/d5360dc4-6bea-4a7b-bc49-5b2547ad7877.dcm
  inflating: stage_2_train_images/d5364bc1-bc2a-4bd0-a1bd-0cfb5a369ccc.dcm
  inflating: stage_2_train_images/d539e101-5662-445c-9f6a-381e674f0aed.dcm
  inflating: stage_2_train_images/d53cee27-787e-4136-aaf6-03bcff985ac9.dcm
  inflating: stage_2_train_images/d53ebae4-9b96-4a05-b066-4635d52e3ac2.dcm
  inflating: stage_2_train_images/d54150ef-1739-4002-aaef-e4e8441038b1.dcm
  inflating: stage_2_train_images/d54240c5-1375-42c8-85b5-e77968f6befc.dcm
  inflating: stage_2_train_images/d5431b84-9bf2-4758-badc-569e71ee1f9a.dcm
  inflating: stage_2_train_images/d54d9912-1e3d-4660-abf0-a0f95caa31c7.dcm
  inflating: stage_2_train_images/d54e2889-e703-469a-9bea-dd90c34d38a3.dcm
  inflating: stage_2_train_images/d55144f1-7b8b-424a-b460-5a7051abd301.dcm
  inflating: stage_2_train_images/d5523ffc-1c8a-4250-939a-3f5215397ff2.dcm
  inflating: stage_2_train_images/d554803a-b040-4952-b21d-9fbe9faf53a7.dcm
  inflating: stage_2_train_images/d5555a76-3191-4a05-a1d9-3e11f4c8d045.dcm
  inflating: stage_2_train_images/d5566f6b-26c7-4610-82af-a29a80ac6525.dcm
  inflating: stage_2_train_images/d557290b-bd46-4038-8056-c90688655890.dcm
  inflating: stage_2_train_images/d55c3e2b-327a-4d3e-83d3-059ec3132363.dcm
  inflating: stage_2_train_images/d55c7487-d4d7-4f9b-8f14-dbdd3a2217b1.dcm
  inflating: stage_2_train_images/d55d7d6f-f869-45f8-8cf3-a176e97bb71b.dcm
  inflating: stage_2_train_images/d55f67db-c835-43a1-a4a1-a80468eb7b75.dcm
  inflating: stage_2_train_images/d55faf33-fa7a-4d88-8050-abf7686577bf.dcm
  inflating: stage_2_train_images/d55fda52-2efc-449c-a9c5-5066a16ea85b.dcm
  inflating: stage_2_train_images/d561ede1-9992-4d20-a935-53a32a8c05d1.dcm
  inflating: stage_2_train_images/d563628b-cbde-4367-b71d-275634867e64.dcm
  inflating: stage_2_train_images/d564b24d-50c8-418c-bd27-6368d2f3e733.dcm
  inflating: stage_2_train_images/d5679f97-4534-4e1a-9fd6-758297d3fa98.dcm
  inflating: stage_2_train_images/d567d023-434f-4714-815c-f45556873cfc.dcm
  inflating: stage_2_train_images/d56aba53-2936-456a-a77f-2c758f2c9d41.dcm
  inflating: stage_2_train_images/d5727f57-0ad9-406c-8b58-8968ed91b100.dcm
  inflating: stage_2_train_images/d5746713-b462-446c-a526-9f705794377a.dcm
  inflating: stage_2_train_images/d5750138-776e-47e3-80c9-a413c2ea72cc.dcm
  inflating: stage_2_train_images/d57802e4-da7a-4a7b-b5ca-c662ab5b1411.dcm
  inflating: stage_2_train_images/d578cc91-9c7e-4f5e-b18a-f855a7d6b4b2.dcm
  inflating: stage_2_train_images/d5791a7c-c294-4bba-a10d-66064f247793.dcm
  inflating: stage_2_train_images/d57aaee0-caa9-46ad-b0a1-d34bebf2a2bc.dcm
  inflating: stage_2_train_images/d57b47b7-2324-4bdc-a0ee-27ad6925d9e0.dcm
  inflating: stage_2_train_images/d57ea738-4067-4bef-a751-088905ca5889.dcm
  inflating: stage_2_train_images/d57ecb68-cbf1-46cf-993e-a3763d1917da.dcm
  inflating: stage_2_train_images/d57f327a-a211-4e95-9bf9-0c190c9ae0ab.dcm
  inflating: stage_2_train_images/d58012a2-a845-472d-8f49-336a3cc977c1.dcm
  inflating: stage_2_train_images/d5835755-52dd-4a9d-a70e-6a49fcbbe1c8.dcm
  inflating: stage_2_train_images/d584a479-4020-46f5-bc34-62f6d8de8962.dcm
  inflating: stage_2_train_images/d585321b-d6a9-4a7a-ac36-c95234f61cd2.dcm
  inflating: stage_2_train_images/d58893fa-b9e9-4d93-a75f-c0b982aff3c1.dcm
  inflating: stage_2_train_images/d589f629-f614-4e32-9a69-ecd674a34676.dcm
  inflating: stage_2_train_images/d58a922b-97c8-4ee8-ad84-1c4c6c883c86.dcm
  inflating: stage_2_train_images/d58bd53a-c69c-443e-851d-479a78d1c9b2.dcm
```

Save data

```
for dcm in os.listdir('stage_2_train_images'):
    #### covert dicom to png
    ds = pydicom.read_file('./stage_2_train_images/'+dcm)
    img = ds.pixel_array.astype(float)
    #### resize
    img_scaled = skimage.transform.resize(img, (512, 512), anti_aliasing=True)
    img_scaled = np.uint8(img_scaled)
    #### save image
    cv2.imwrite('./stage_2_train_images/' + dcm.replace('.dcm','.png'), img_scaled)
    os.system('rm stage_2_train_images/'+ dcm)
```

Preparation

```
df1 = pd.read_csv('./stage_2_train_labels.csv')
```

```
df2 = pd.read_csv('./stage_2_detailed_class_info.csv')
```

```
print(f'Ground Truth boxes size: {len(df1)}, Patient class size: {len(df2)}')
```

```
Ground Truth boxes size: 30227, Patient class size: 30227
```

```
df1.head()
```

	patientId	x	y	width	height	Target
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN	0
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN	0
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN	0
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1

```
df2.head()
```

	patientId	class
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	No Lung Opacity / Not Normal
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	No Lung Opacity / Not Normal
2	00322d4d-1c29-4943-afc9-b6754be640eb	No Lung Opacity / Not Normal
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	Normal
4	00436515-870c-4b36-a041-de91049b9ab4	Lung Opacity

```
samples = None
## merge
df = pd.merge(df1, df2, on='patientId')

## keep Lung Opacity
samples = pd.DataFrame(df[df['class']=='Lung Opacity'])
samples.drop_duplicates(inplace = True,  ignore_index=True)
print(len(samples))
```

```
9555
```

Split patients to validation and train

```
train_samples = None
val_samples = None
## Split
msk = np.random.rand(len(samples)) < 0.8
train_df = samples[msk]
val_df = samples[~msk]


#Finding indexes of validation rows which has the same 'patientId' as train data's.
intersect = pd.merge(train_df, val_df, on = 'patientId')
inter_idx = np.array([val_df.index[val_df['patientId']==value].tolist() for value in intersect['patientId'].values], dtype=object)

idx_flat = []
for l in inter_idx:
    idx_flat.extend(l)

#Removing duplicates
idx_flat = list(dict.fromkeys(idx_flat))


# Adding back to train datasets
train_samples = train_df.append(val_df.loc[idx_flat], ignore_index=True)
```

```
val_samples = val_df.drop(idx_flat)
val_samples.reset_index(drop=True, inplace=True)
print('Train and Validation size (Before) :({},{})'.format(len(train_df), len(val_df)))
print('Train and Validation size (After) :({},{})'.format(len(train_samples), len(val_samples)))
```

```
     Train and Validation size (Before) :(7631,1924)
     Train and Validation size (After) :(8747,808)
```

```
# Group by Patient ID
train_samples = train_samples.groupby(['patientId'], dropna=True)
val_samples = val_samples.groupby(['patientId'], dropna=True)
```

```
train_samples.head()
```

	patientId	x	y	width	height	Target	class
0	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1	Lung Opacity
1	00436515-870c-4b36-a041-de91049b9ab4	562.0	152.0	256.0	453.0	1	Lung Opacity
2	00704310-78a8-4b38-8475-49f4573b2dbb	323.0	577.0	160.0	104.0	1	Lung Opacity
3	00704310-78a8-4b38-8475-49f4573b2dbb	695.0	575.0	162.0	137.0	1	Lung Opacity
4	00aecb01-a116-45a2-956c-08d2fa55433f	288.0	322.0	94.0	135.0	1	Lung Opacity
...	...	...	...	...	...	...	...
8826	c033f66d-900e-4ba3-8da4-a6823ea89d09	547.0	369.0	177.0	285.0	1	Lung Opacity
8827	c0654897-6bfe-4b7c-abd3-dda76d0fcac2	131.0	369.0	270.0	460.0	1	Lung Opacity

Dataset

When we rotate or scale an image, we have to do all transformation on bounding boxes too.

```
from detectron2.structures import BoxMode

whole = {'train': train_samples, 'val': val_samples}

def get_CXR_dicts(string, img_size = 512):

    if string == 'train':
        is_training = True
    else:
        is_training = False

    samples = whole[string]
    dataset_dicts = []
    for name, group in samples:
        record = {}
        ## filename is the address of patient image (I already saved the preprocessed images on my google drive)
        filename = str(f'./stage_2_train_images/{name}.png')
        record["file_name"] = filename
        ## fill None with appropriate values
        record["image_id"] = name
        record["height"] = img_size # Resized shape of image
        record["width"] = img_size

        objs = []
        for _, row in group.iterrows():
            ###  each group represents a patient, and the rows of the specific group shows bounding boxes for that patient
            resize_ratio = 0.5 # 512/1024
            x = int(row['x'])
            y = int(row['y'])
            width = int(row['width'])
            height = int(row['height'])

            x = int(round(x*resize_ratio))
            y = int(round(y*resize_ratio))
            w = int(round(width*resize_ratio))
            h = int(round(height*resize_ratio))
            bbox_resized = [x, y, w, h]

            obj = {
                "bbox": bbox_resized,
                "bbox_mode": BoxMode.XYWH_ABS,
                "category_id": 0,
            }
            ### objs is list of bounding boxes of a particular patient
            objs.append(obj)

        record["annotations"] = objs

        dataset_dicts.append(record)

    return dataset_dicts

### fill the attributes
for d in ["train", "val"]:
    DatasetCatalog.register("CXR_" + d, lambda d=d: get_CXR_dicts(d))
    MetadataCatalog.get("CXR_" + d).set(thing_classes=["opacity"])

CXR_metadata = MetadataCatalog.get("CXR_train")
```

Visualizing three train patients with their annotations

```
dataset_dicts = get_CXR_dicts("train")
```

```
for d in random.sample(dataset_dicts, 3):
    img = cv2.imread(d["file_name"])
    visualizer = Visualizer(img[:, :, ::-1], metadata=CXR_metadata, scale=0.5)
    out = visualizer.draw_dataset_dict(d)
    cv2_imshow(out.get_image()[:, :, ::-1])
```

we can change settings like LR or iterations here to get better performance.

We train detectron2 with two different baselines:

1. RetinaNet
2. FasterRCNN

and then we compare the results.

```
from detectron2.engine import DefaultTrainer

cfg = get_cfg()

cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_FPN_1x.yaml"))


cfg.DATASETS.TRAIN = ("CXR_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2

cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_FPN_1x.yaml")  # Let training initialize from model zoo

cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 500
cfg.SOLVER.STEPS = []
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128   # faster, and good enough for this toy dataset (default: 512)



cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1

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

```
[12/08 08:49:02 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)
          )
```

```
cfg1 = get_cfg()

cfg1.merge_from_file(model_zoo.get_config_file("COCO-Detection/retinanet_R_50_FPN_1x.yaml"))


cfg1.DATASETS.TRAIN = ("CXR_train",)
cfg1.DATASETS.TEST = ()
cfg1.DATALOADER.NUM_WORKERS = 2


cfg1.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/retinanet_R_50_FPN_1x.yaml")

cfg1.SOLVER.IMS_PER_BATCH = 2
cfg1.SOLVER.BASE_LR = 0.00025
cfg1.SOLVER.MAX_ITER = 500
cfg1.SOLVER.STEPS = []
cfg1.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128



cfg1.MODEL.ROI_HEADS.NUM_CLASSES = 1

cfg1.OUTPUT_DIR = './output_Retina'
os.makedirs(cfg1.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg1)
trainer.resume_or_load(resume=False)
trainer.train()
```

```
Loading config /usr/local/lib/python3.7/dist-packages/detectron2/model_zoo/configs/COCO-Detection/../Base-RetinaNet.yaml with yaml.un
[12/08 09:02:51 d2.engine.defaults]: Model:
RetinaNet(
  (backbone): FPN(
    (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): LastLevelP6P7(
      (p6): Conv2d(2048, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
      (p7): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    )
    (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(
```

```
#RCNN
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7
predict_RCNN = DefaultPredictor(cfg)
```

```
#Retina
cfg1.MODEL.WEIGHTS = os.path.join(cfg1.OUTPUT_DIR, "model_final.pth")
cfg1.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7
predictor_Retina = DefaultPredictor(cfg1)
```

Results of Faster-RCNN:

```
######
import pandas as pd
metrics_df = pd.read_json("./output_RCNN/metrics.json", orient="records", lines=True)
mdf = metrics_df.sort_values("iteration")
mdf.head(10).T
```

	0	1	2	3	4
data_time	0.007171	0.007291	0.006866	0.008601	0.006941
eta_seconds	447.492329	418.477463	387.019334	366.321021	357.461218
fast_rcnn/cls_accuracy	0.804688	0.900391	0.945312	0.919922	0.921875
fast_rcnn/false_negative	0.727941	1.000000	1.000000	1.000000	1.000000
fast_rcnn/fg_cls_accuracy	0.272059	0.000000	0.000000	0.000000	0.000000
iteration	19.000000	39.000000	59.000000	79.000000	99.000000
loss_box_reg	0.160706	0.250752	0.161676	0.285909	0.296601
loss_cls	0.623545	0.530271	0.381372	0.318220	0.289679
loss_rpn_cls	0.109928	0.104686	0.084949	0.073647	0.032388
loss_rpn_loc	0.017459	0.017746	0.018727	0.017098	0.014541
lr	0.000010	0.000020	0.000030	0.000040	0.000050
roi_head/num_bg_samples	121.250000	118.750000	121.000000	117.750000	118.000000
roi_head/num_fg_samples	6.750000	9.250000	7.000000	10.250000	10.000000
rpn/num_neg_anchors	249.750000	249.250000	250.500000	249.500000	250.250000
rpn/num_pos_anchors	6.250000	6.750000	5.500000	6.500000	5.750000
time	0.932276	0.857385	0.855247	0.871812	0.926520
total_loss	0.912309	0.902253	0.653780	0.738799	0.611257

Results of Retinanet:

```
########
import pandas as pd
metrics_df = pd.read_json("./output_Retina/metrics.json", orient="records", lines=True)
mdf = metrics_df.sort_values("iteration")
mdf.head(10).T
```

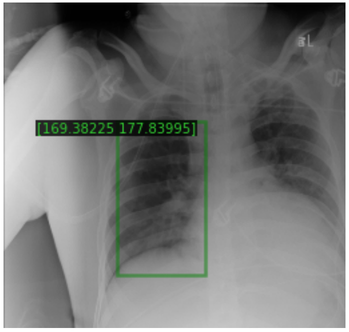
	0	1	2	3	4	5
data_time	0.009426	0.007446	0.007679	0.007109	0.008862	0.008071
eta_seconds	869.992080	724.388817	571.552633	542.887304	515.520277	462.878659
iteration	19.000000	39.000000	59.000000	79.000000	99.000000	119.000000
loss_box_reg	0.616985	0.637741	0.555257	0.513169	0.514836	0.451709
loss_cls	1.272475	1.017066	0.559201	0.510193	0.460335	0.384606
lr	0.000010	0.000020	0.000030	0.000040	0.000050	0.000060
num_pos_anchors	46.250000	50.750000	46.500000	48.750000	47.500000	44.750000
time	1.812483	1.287472	1.289599	1.208725	1.146682	1.141542
total_loss	1.867653	1.715738	1.078183	1.044905	0.995176	0.834199

Visualize some validation images with their predicted bounding boxes

```
val_metadata = MetadataCatalog.get("CXR_val")
val_dataset_dicts = get_CXR_dicts("val")
```

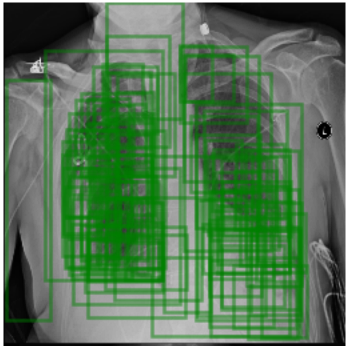
```
##### visualize
# RCNN
for d1 in random.sample(val_dataset_dicts, 3):
    im1 = cv2.imread(d1["file_name"])
    outputs1 = predict_RCNN(im1)
    v = Visualizer(im1[:, :, ::-1], metadata = val_metadata, scale = 0.5)
    for box in outputs1["instances"].pred_boxes.to('cpu'):
        v.draw_box(box)
        v.draw_text(str(box[:2].numpy()), tuple(box[:2].numpy()))
    v = v.get_output()
    img = v.get_image()[:, :, ::-1]
    cv2_imshow(img)
```

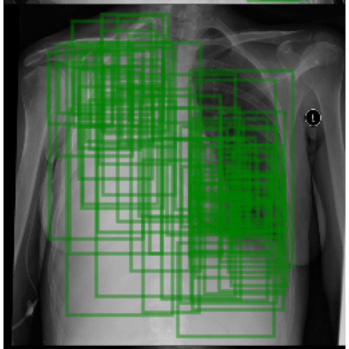
```
/usr/local/lib/python3.7/dist-packages/detectron2/structures/image_list.py:88: UserWarni
  max_size = (max_size + (stride - 1)) // stride * stride
```

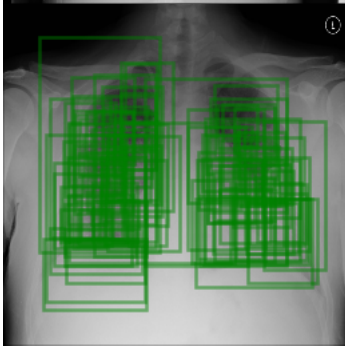


```
for d2 in random.sample(val_dataset_dicts, 3):
    im2 = cv2.imread(d2["file_name"])
    outputs2 = predictor_Retina(im2)
    v1 = Visualizer(im2[:, :, ::-1], metadata = val_metadata, scale = 0.5)
    for box in outputs2["instances"].pred_boxes.to('cpu'):
        v1.draw_box(box)
        #v1.draw_text(str(box[:2].numpy()), tuple(box[:2].numpy()))
    v1 = v1.get_output()
    img1 = v1.get_image()[:, :, ::-1]
    cv2_imshow(img1)
```

```
/usr/local/lib/python3.7/dist-packages/detectron2/structures/image_list.py:88: UserWarni
  max_size = (max_size + (stride - 1)) // stride * stride
```







Inference

```
#RCNN
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader
evaluator = COCOEvaluator("CXR_val", output_dir="./output_RCNN")
val_loader = build_detection_test_loader(cfg, "CXR_val")
print(inference_on_dataset(predict_RCNN.model, val_loader, evaluator))
```

```
[12/08 08:58:16 d2.evaluation.coco_evaluation]: Trying to convert 'CXR_val' to COCO format ...
[12/08 08:58:16 d2.data.datasets.coco]: Converting annotations of dataset 'CXR_val' to COCO format ...)
[12/08 08:58:16 d2.data.datasets.coco]: Converting dataset dicts into COCO format
[12/08 08:58:16 d2.data.datasets.coco]: Conversion finished, #images: 663, #annotations: 808
[12/08 08:58:16 d2.data.datasets.coco]: Caching COCO format annotations at './output_RCNN/CXR_val_coco_format.json' ...
[12/08 08:58:16 d2.data.build]: Distribution of instances among all 1 categories:
|  category  | #instances   |
|:----------:|:-------------|
|  opacity   | 808          |
|            |              |
[12/08 08:58:16 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in inference: [ResizeShortestEdge(short_edge_length=(800,
[12/08 08:58:16 d2.data.common]: Serializing 663 elements to byte tensors and concatenating them all ...
[12/08 08:58:16 d2.data.common]: Serialized dataset takes 0.18 MiB
[12/08 08:58:16 d2.evaluation.evaluator]: Start inference on 663 batches
/usr/local/lib/python3.7/dist-packages/detectron2/structures/image_list.py:88: UserWarning: __floordiv__ is deprecated, and its behav
  max_size = (max_size + (stride - 1)) // stride * stride
[12/08 08:58:20 d2.evaluation.evaluator]: Inference done 11/663. Dataloading: 0.0019 s/iter. Inference: 0.2721 s/iter. Eval: 0.0003 s
[12/08 08:58:25 d2.evaluation.evaluator]: Inference done 30/663. Dataloading: 0.0024 s/iter. Inference: 0.2667 s/iter. Eval: 0.0003 s
[12/08 08:58:30 d2.evaluation.evaluator]: Inference done 49/663. Dataloading: 0.0025 s/iter. Inference: 0.2660 s/iter. Eval: 0.0003 s
[12/08 08:58:35 d2.evaluation.evaluator]: Inference done 68/663. Dataloading: 0.0025 s/iter. Inference: 0.2660 s/iter. Eval: 0.0003 s
[12/08 08:58:40 d2.evaluation.evaluator]: Inference done 87/663. Dataloading: 0.0024 s/iter. Inference: 0.2660 s/iter. Eval: 0.0003 s
[12/08 08:58:45 d2.evaluation.evaluator]: Inference done 106/663. Dataloading: 0.0025 s/iter. Inference: 0.2661 s/iter. Eval: 0.0003
[12/08 08:58:50 d2.evaluation.evaluator]: Inference done 125/663. Dataloading: 0.0024 s/iter. Inference: 0.2661 s/iter. Eval: 0.0003
[12/08 08:58:55 d2.evaluation.evaluator]: Inference done 144/663. Dataloading: 0.0025 s/iter. Inference: 0.2663 s/iter. Eval: 0.0003
[12/08 08:59:01 d2.evaluation.evaluator]: Inference done 163/663. Dataloading: 0.0025 s/iter. Inference: 0.2664 s/iter. Eval: 0.0003
[12/08 08:59:06 d2.evaluation.evaluator]: Inference done 182/663. Dataloading: 0.0025 s/iter. Inference: 0.2665 s/iter. Eval: 0.0003
[12/08 08:59:11 d2.evaluation.evaluator]: Inference done 201/663. Dataloading: 0.0025 s/iter. Inference: 0.2666 s/iter. Eval: 0.0003
[12/08 08:59:16 d2.evaluation.evaluator]: Inference done 220/663. Dataloading: 0.0025 s/iter. Inference: 0.2668 s/iter. Eval: 0.0003
[12/08 08:59:21 d2.evaluation.evaluator]: Inference done 239/663. Dataloading: 0.0025 s/iter. Inference: 0.2668 s/iter. Eval: 0.0003
[12/08 08:59:26 d2.evaluation.evaluator]: Inference done 258/663. Dataloading: 0.0025 s/iter. Inference: 0.2669 s/iter. Eval: 0.0003
[12/08 08:59:31 d2.evaluation.evaluator]: Inference done 277/663. Dataloading: 0.0025 s/iter. Inference: 0.2670 s/iter. Eval: 0.0003
[12/08 08:59:37 d2.evaluation.evaluator]: Inference done 296/663. Dataloading: 0.0024 s/iter. Inference: 0.2671 s/iter. Eval: 0.0003
[12/08 08:59:42 d2.evaluation.evaluator]: Inference done 315/663. Dataloading: 0.0024 s/iter. Inference: 0.2672 s/iter. Eval: 0.0003
[12/08 08:59:47 d2.evaluation.evaluator]: Inference done 334/663. Dataloading: 0.0025 s/iter. Inference: 0.2672 s/iter. Eval: 0.0003
[12/08 08:59:52 d2.evaluation.evaluator]: Inference done 353/663. Dataloading: 0.0025 s/iter. Inference: 0.2673 s/iter. Eval: 0.0003
[12/08 08:59:57 d2.evaluation.evaluator]: Inference done 372/663. Dataloading: 0.0024 s/iter. Inference: 0.2674 s/iter. Eval: 0.0003
[12/08 09:00:02 d2.evaluation.evaluator]: Inference done 391/663. Dataloading: 0.0024 s/iter. Inference: 0.2674 s/iter. Eval: 0.0003
[12/08 09:00:08 d2.evaluation.evaluator]: Inference done 410/663. Dataloading: 0.0024 s/iter. Inference: 0.2675 s/iter. Eval: 0.0003
[12/08 09:00:13 d2.evaluation.evaluator]: Inference done 429/663. Dataloading: 0.0024 s/iter. Inference: 0.2675 s/iter. Eval: 0.0003
[12/08 09:00:18 d2.evaluation.evaluator]: Inference done 448/663. Dataloading: 0.0024 s/iter. Inference: 0.2675 s/iter. Eval: 0.0003
[12/08 09:00:23 d2.evaluation.evaluator]: Inference done 467/663. Dataloading: 0.0024 s/iter. Inference: 0.2676 s/iter. Eval: 0.0003
[12/08 09:00:28 d2.evaluation.evaluator]: Inference done 486/663. Dataloading: 0.0024 s/iter. Inference: 0.2676 s/iter. Eval: 0.0003
[12/08 09:00:33 d2.evaluation.evaluator]: Inference done 505/663. Dataloading: 0.0024 s/iter. Inference: 0.2676 s/iter. Eval: 0.0003
[12/08 09:00:38 d2.evaluation.evaluator]: Inference done 524/663. Dataloading: 0.0024 s/iter. Inference: 0.2676 s/iter. Eval: 0.0003
[12/08 09:00:44 d2.evaluation.evaluator]: Inference done 543/663. Dataloading: 0.0024 s/iter. Inference: 0.2677 s/iter. Eval: 0.0003
[12/08 09:00:49 d2.evaluation.evaluator]: Inference done 562/663. Dataloading: 0.0024 s/iter. Inference: 0.2677 s/iter. Eval: 0.0003
[12/08 09:00:54 d2.evaluation.evaluator]: Inference done 581/663. Dataloading: 0.0024 s/iter. Inference: 0.2677 s/iter. Eval: 0.0003
[12/08 09:00:59 d2.evaluation.evaluator]: Inference done 600/663. Dataloading: 0.0023 s/iter. Inference: 0.2677 s/iter. Eval: 0.0003
[12/08 09:01:04 d2.evaluation.evaluator]: Inference done 619/663. Dataloading: 0.0023 s/iter. Inference: 0.2677 s/iter. Eval: 0.0003
[12/08 09:01:09 d2.evaluation.evaluator]: Inference done 638/663. Dataloading: 0.0023 s/iter. Inference: 0.2678 s/iter. Eval: 0.0003
[12/08 09:01:14 d2.evaluation.evaluator]: Inference done 657/663. Dataloading: 0.0023 s/iter. Inference: 0.2678 s/iter. Eval: 0.0003
[12/08 09:01:16 d2.evaluation.evaluator]: Total inference time: 0:02:58.152556 (0.270749 s / iter per device, on 1 devices)
[12/08 09:01:16 d2.evaluation.evaluator]: Total inference pure compute time: 0:02:56 (0.267798 s / iter per device, on 1 devices)
[12/08 09:01:16 d2.evaluation.coco_evaluation]: Preparing results for COCO format ...
[12/08 09:01:16 d2.evaluation.coco_evaluation]: Saving results to ./output_RCNN/coco_instances_results.json
[12/08 09:01:16 d2.evaluation.coco_evaluation]: Evaluating predictions with unofficial COCO API...
Loading and preparing results...
```

```
#Retina
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader
evaluator = COCOEvaluator("CXR_val", output_dir="./output_Retina")
val_loader = build_detection_test_loader(cfg1, "CXR_val")
print(inference_on_dataset(predictor_Retina.model, val_loader, evaluator))
```

```
[12/08 09:16:14 d2.evaluation.evaluator]: Inference done 447/663. Dataloading: 0.0024 s/iter. Inference: 0.3114 s/iter. Eval: 0.0004
[12/08 09:16:19 d2.evaluation.evaluator]: Inference done 463/663. Dataloading: 0.0024 s/iter. Inference: 0.3114 s/iter. Eval: 0.0004
[12/08 09:16:24 d2.evaluation.evaluator]: Inference done 479/663. Dataloading: 0.0024 s/iter. Inference: 0.3114 s/iter. Eval: 0.0004
[12/08 09:16:29 d2.evaluation.evaluator]: Inference done 495/663. Dataloading: 0.0024 s/iter. Inference: 0.3114 s/iter. Eval: 0.0004
[12/08 09:16:34 d2.evaluation.evaluator]: Inference done 511/663. Dataloading: 0.0024 s/iter. Inference: 0.3114 s/iter. Eval: 0.0004
[12/08 09:16:39 d2.evaluation.evaluator]: Inference done 527/663. Dataloading: 0.0024 s/iter. Inference: 0.3114 s/iter. Eval: 0.0004
[12/08 09:16:44 d2.evaluation.evaluator]: Inference done 543/663. Dataloading: 0.0024 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:16:49 d2.evaluation.evaluator]: Inference done 559/663. Dataloading: 0.0024 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:16:54 d2.evaluation.evaluator]: Inference done 575/663. Dataloading: 0.0025 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:16:59 d2.evaluation.evaluator]: Inference done 591/663. Dataloading: 0.0024 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:17:05 d2.evaluation.evaluator]: Inference done 607/663. Dataloading: 0.0024 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:17:10 d2.evaluation.evaluator]: Inference done 623/663. Dataloading: 0.0025 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:17:15 d2.evaluation.evaluator]: Inference done 639/663. Dataloading: 0.0024 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:17:20 d2.evaluation.evaluator]: Inference done 655/663. Dataloading: 0.0024 s/iter. Inference: 0.3115 s/iter. Eval: 0.0004
[12/08 09:17:22 d2.evaluation.evaluator]: Total inference time: 0:03:27.106742 (0.314752 s / iter per device, on 1 devices)
[12/08 09:17:22 d2.evaluation.evaluator]: Total inference pure compute time: 0:03:24 (0.311532 s / iter per device, on 1 devices)
[12/08 09:17:23 d2.evaluation.coco_evaluation]: Preparing results for COCO format ...
[12/08 09:17:23 d2.evaluation.coco_evaluation]: Saving results to ./output_Retina/coco_instances_results.json
[12/08 09:17:23 d2.evaluation.coco_evaluation]: Evaluating predictions with unofficial COCO API...
Loading and preparing results...
DONE (t=0.34s)
creating index...
index created!
[12/08 09:17:23 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[12/08 09:17:24 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.45 seconds.
[12/08 09:17:24 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[12/08 09:17:24 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.12 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.100
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.339
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.030
 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 ] = 0.055
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.123
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.125
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.338
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.431
 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 ] = 0.386
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.451
[12/08 09:17:24 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm  |  APl   |
|:------:|:------:|:------:|:-----:|:-----:|:------:|
| 10.003 | 33.948 | 2.972  |  nan  | 5.530 | 12.308 |
[12/08 09:17:24 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
OrderedDict([('bbox', {'AP': 10.003349873303502, 'AP50': 33.948092006676625, 'AP75': 2.9722191996031806, 'APs': nan, 'APm': 5.5301468
```