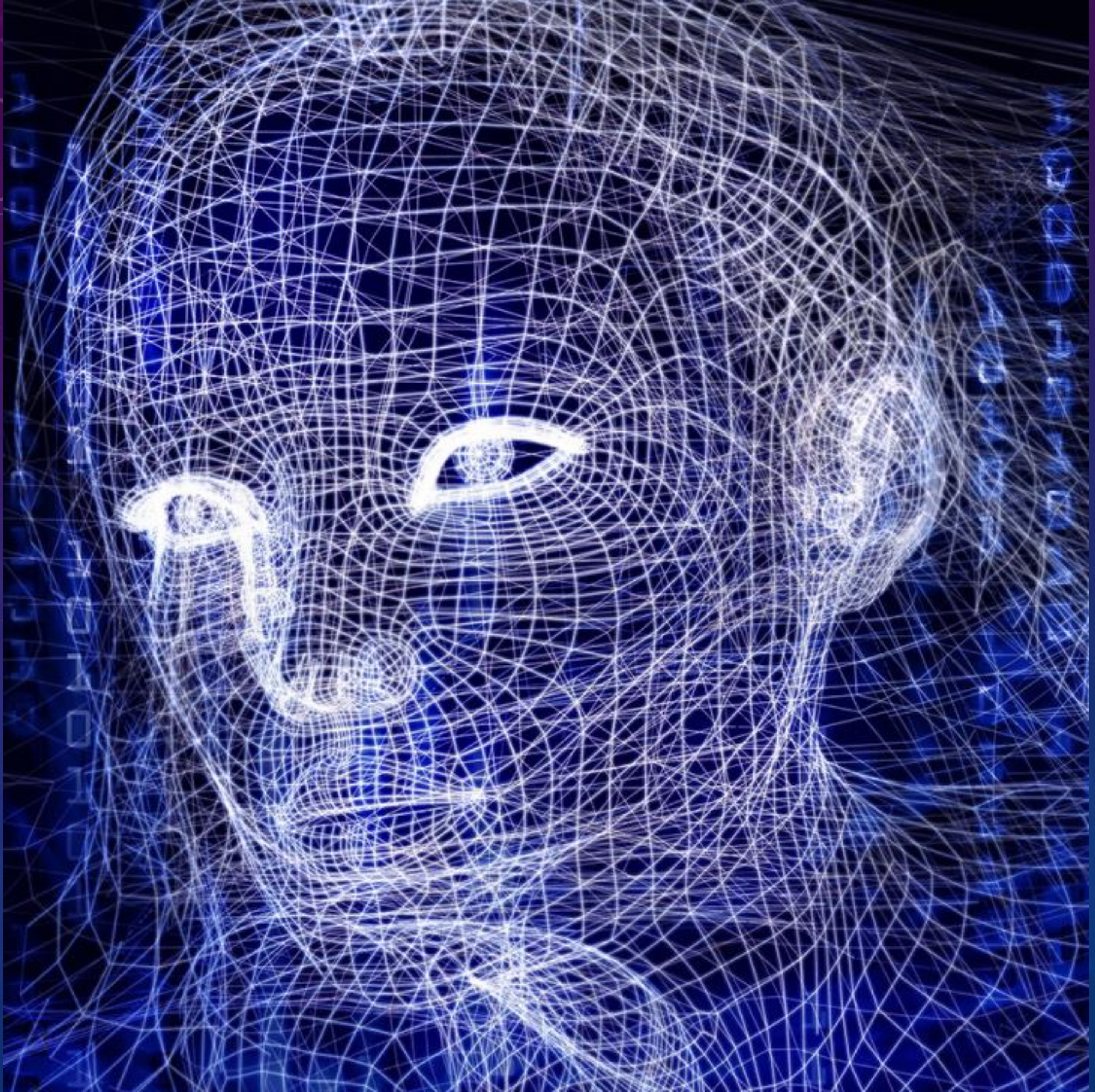
LECTURE SERIES FOR DIGITAL SURVEILLANCE SYSTEMS AND APPLICATION

CH4 EXPERIMENT FOR OBJECT DETECTION

徐繼聖

Gee-Sern Jison Hsu

National Taiwan University of Science and Technology

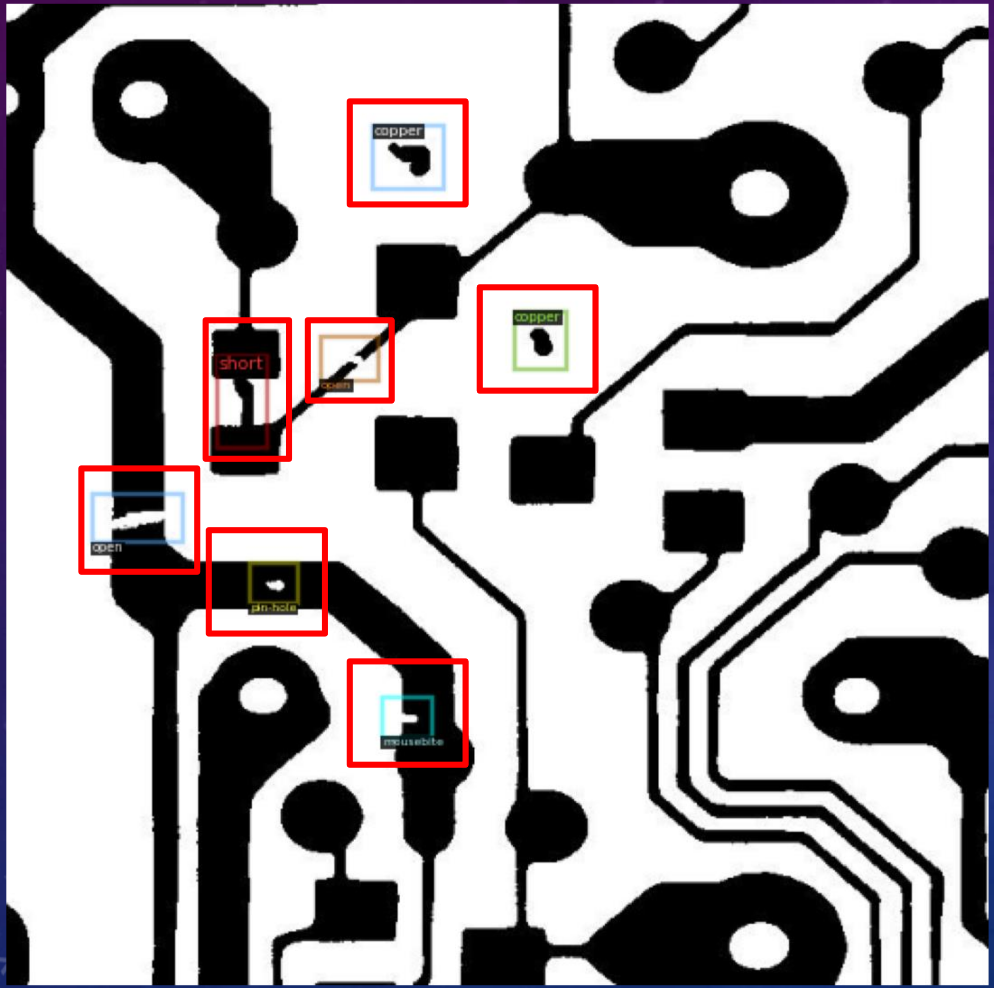


Example 4-1 – Faster RCNN

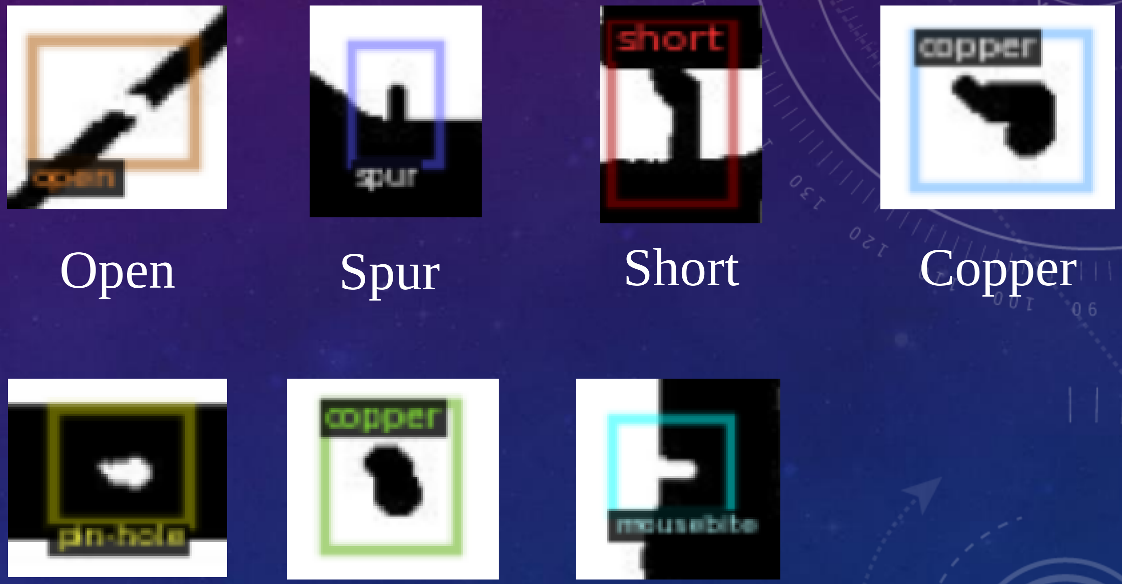
In this example, we want to use Faster RCNN to detect defects on a Printed Circuit Board (PCB). The problem is a classification of 6 different defect types. The defect types can be a short circuit or Pin-hole on the circuits for example. You can see an overview of all defect types in the following slide. We will use pretrained weights, which were trained on the COCO dataset and retrain on our own data.

Example 4-1 – Faster RCNN

PCB data sample:



Category Type Sample:



Open Spur Short Copper

Pin-hole Copper Mousebite

Training data :1000 images
Testing data : 500 images

Example 4-1 – Faster RCNN

Step 1 : Set up environment

```
!pip install -U torch==1.5 torchvision==0.6 -f https://download.pytorch.org/whl/cu101/torch_stable.html
!pip install cython pyyaml==5.1
!pip install -U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
```

Result :

```
Successfully installed torch-1.5.0+cu101 torchvision-0.6.0+cu101
Requirement already satisfied: cython in /usr/local/lib/python3.6/dist-packages (0.29.20)
```

```
Successfully built pyyaml
```

```
Successfully installed pyyaml-5.1
```

```
Successfully installed pycocotools-2.0
```

Example 4-1 – Faster RCNN

Step 1 : Set up environment

```
!git clone https://github.com/tangsanli5201/DeepPCB
# install detectron2:
!pip install detectron2==0.1.3 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.5/index.html
```

Result :

```
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.6/dist-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensc
Building wheels for collected packages: fvcore
  Building wheel for fvcore (setup.py) ... done
  Created wheel for fvcore: filename=fvcore-0.1.1.post20200630-cp36-none-any.whl size=41299 sha256=0973f00611a6330425c434401316cea811cc11c9d0ffe9159f9c
  Stored in directory: /root/.cache/pip/wheels/80/eb/49/83b9d20a804f1b4b163d1c1451c670a2067a00175662516f01
Successfully built fvcore
Installing collected packages: yacs, portalocker, fvcore, mock, detectron2
Successfully installed detectron2-0.1.3+cu101 fvcore-0.1.1.post20200630 mock-4.0.2 portalocker-1.7.0 yacs-0.1.7
```

Example 4-1 – Faster RCNN

Register PCB dataset

```
def get_PCB_dict(data_list):
    dataset_dicts = []

    for i,path in enumerate(data_list):
        filename = path[0]
        height, width = cv2.imread(filename).shape[:2]
        record = {}
        record['file_name'] = filename
        record['image_id'] = i
        record['height']= height
        record['width']= width

        objs = []
        with open(path[1]) as t:
            print(path[1])
            lines = t.readlines()
            print(lines)
```

The txt contains all the information in each image with all defect locations and types

Example 4-1 – Faster RCNN

We need to convert PCB-data into COCO format(.JSON) for training.
Here are some important information.

```
        for line in lines:
            if line[-1]=="\n":
              box = line[:-1].split(' ')
            else:
              box = line.split(' ')
            print(box)
            boxes = list(map(float,[box[0],box[1],box[2],box[3]]))
            category = int(box[4])
            print(boxes)
            obj = {
                "bbox": boxes,
                "bbox_mode": BoxMode.XYXY_ABS,
                #"segmentation": [poly], To draw a line, along to ballon
                #you will need this for mask RCNN
                "category_id": category-1,
                "iscrowd": 0
            }
            print(obj)
            objs.append(obj)
        record["annotations"] = objs
    dataset_dicts.append(record)
return dataset_dicts #list of dicts
```

The location and type

The location(point from top left to bottom right) [x1,y1,x2,y2]

Type:0-open 1-short 2-mousebite 3-spur 4-copper 5-pin-hole

Example 4-1 – Faster RCNN

Take a photo and print the information


```
{'file_name': './DeepPCB/PCBData/group20085/20085/20085000_test.jpg', 'image_id': 0, 'height': 640, 'width': 640, 'annotations':

[{'bbox': [409.0, 394.0, 435.0, 422.0], 'bbox_mode': <BoxMode.XYXY_ABS: 0>, 'category_id': 2, 'iscrowd': 0},
{'bbox': [275.0, 383.0, 319.0, 417.0], 'bbox_mode': <BoxMode.XYXY_ABS: 0>, 'category_id': 2, 'iscrowd': 0},
{'bbox': [8.0, 163.0, 36.0, 191.0], 'bbox_mode': <BoxMode.XYXY_ABS: 0>, 'category_id': 3, 'iscrowd': 0},
{'bbox': [244.0, 151.0, 270.0, 182.0], 'bbox_mode': <BoxMode.XYXY_ABS: 0>, 'category_id': 4, 'iscrowd': 0},
{'bbox': [338.0, 519.0, 364.0, 543.0], 'bbox_mode': <BoxMode.XYXY_ABS: 0>, 'category_id': 5, 'iscrowd': 0},
{'bbox': [476.0, 460.0, 502.0, 481.0], 'bbox_mode': <BoxMode.XYXY_ABS: 0>, 'category_id': 3, 'iscrowd': 0}]}
```

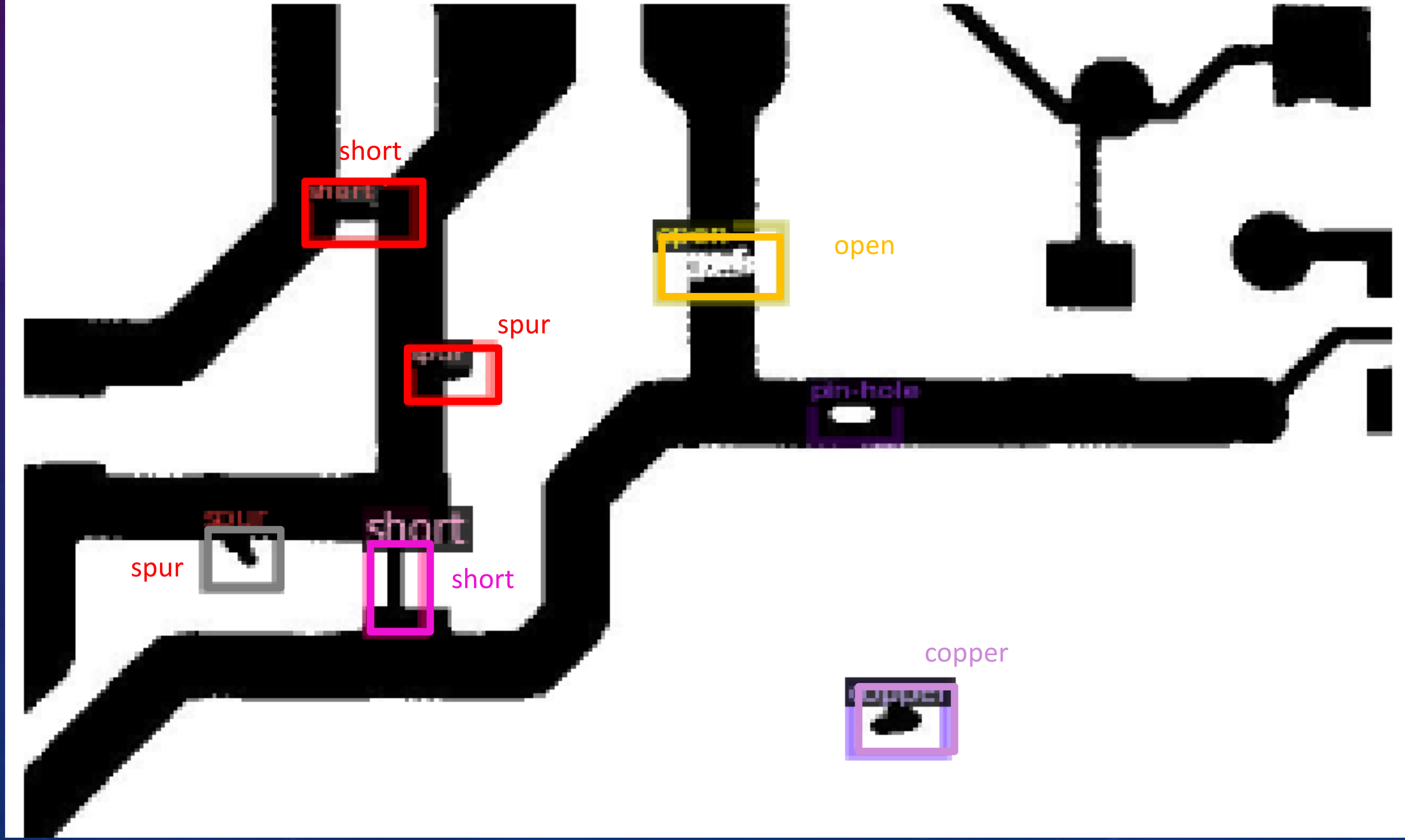

bbox: Defect bounding box coordinates
Bbox_mode: Bbox format
Category: Defect category
Iscrowd: 0: An object
1: A set of objects

Example 4-1 – Faster RCNN

Visualizing the Train Dataset

Randomly select 2 pictures from the train folder of the dataset and check the appearance of the bounding box.

```python
for d in random.sample(dataset_dicts, 2):
    img = cv2.imread(d["file_name"])
    visualizer = Visualizer(img[:, :, ::-1], metadata=PCB_metadata, scale=0.5)
    vis = visualizer.draw_dataset_dict(d)
    cv2_imshow(vis.get_image()[:, :, ::-1])
```



Example 4-1 – Faster RCNN

Define Hyper-parameter

In this part we select the faster_rcnn_R_50_FPN_3x pre-trained model in the model library. The model has been pre-trained on the COCO dataset.

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

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("PCB_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 0
cfg.MODEL.WEIGHTS = "detectron2://COCO-Detection/faster_rcnn_R_50_FPN_3x/137849458/model_final_280758.pkl"  # Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025  # pick a good LR
cfg.SOLVER.MAX_ITER = 300    # 300 iterations seems good enough for this toy dataset; you may need to train longer for a practical dataset
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 4096   # faster, and good e
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 6

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

Pretrained model

Define the max iterations of training

Set the trainer

Load last checkpoint or model.weights

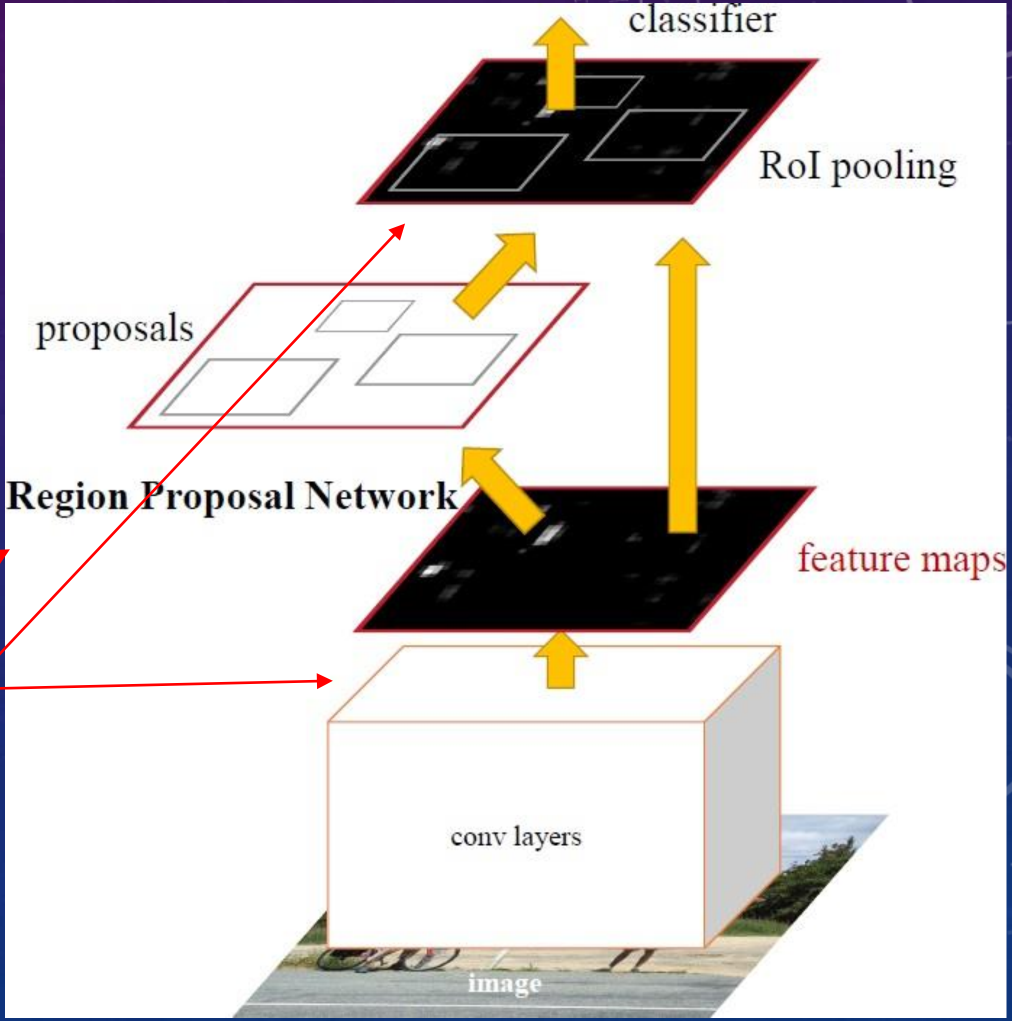
Start to train

Example 4-1 – Faster RCNN

This conv layers architecture uses FPN

```
class FPN(Backbone):
    """
    This module implements :paper:`FPN`.
    It creates pyramid features built on top of some input feature maps.
    """

    def __init__(
        self, bottom_up, in_features, out_channels, norm="", top_block=None, fuse_type="sum"
    ):
        """
```

```
features = self.backbone(images.tensors)
if isinstance(features, torch.Tensor):
    features = OrderedDict([('0', features)])
proposals, proposal_losses = self.rpn(images, features, targets)
detections, detector_losses = self.roi_heads(features, proposals, images.image_sizes, targets)
detections = self.transform.postprocess(detections, images.image_sizes, original_image_sizes)
```



Perform NMS and map the box back to the original image through original_images_size

Example 4-1 – Faster RCNN

rpn.py

```
def forward(self, images, features, gt_instances=None):
    """
    Args:
        images (ImageList): input images of length `N`
        features (dict[str: Tensor]): input data as a mapping from feature
            map name to tensor. Axis 0 represents the number of images `N` in
            the input data; axes 1-3 are channels, height, and width, which may
            vary between feature maps (e.g., if a feature pyramid is used).
        gt_instances (list[Instances], optional): a length `N` list of `Instances`s.
            Each `Instances` stores ground-truth instances for the corresponding image.

    Returns:
        proposals: list[Instances]: contains fields "proposal_boxes", "objectness_logits"
        loss: dict[Tensor] or None
    """
    features = [features[f] for f in self.in_features]
    pred_objectness_logits, pred_anchor_deltas = self.rpn_head(features)
    anchors = self.anchor_generator(features)
```

Realize the core function of the RPN network, output the classification vector objectness, and realize the background classification. Store the four offsets of the candidate box in pred_anchor_deltas

Generate bounding box

```
with torch.no_grad():
    # Find the top proposals by applying NMS and removing boxes that
    # are too small. The proposals are treated as fixed for approximate
    # joint training with roi heads. This approach ignores the derivative
    # w.r.t. the proposal boxes’ coordinates that are also network
    # responses, so is approximate.
    proposals = find_top_rpn_proposals(
            outputs.predict_proposals(),
            outputs.predict_objectness_logits(),
            images,
            self.nms_thresh,
            self.pre_nms_topk[self.training],
            self.post_nms_topk[self.training],
            self.min_box_side_len,
            self.training,
    )

return proposals, losses
```

Use NMS to pick bounding box

Example 4-1 – Faster RCNN

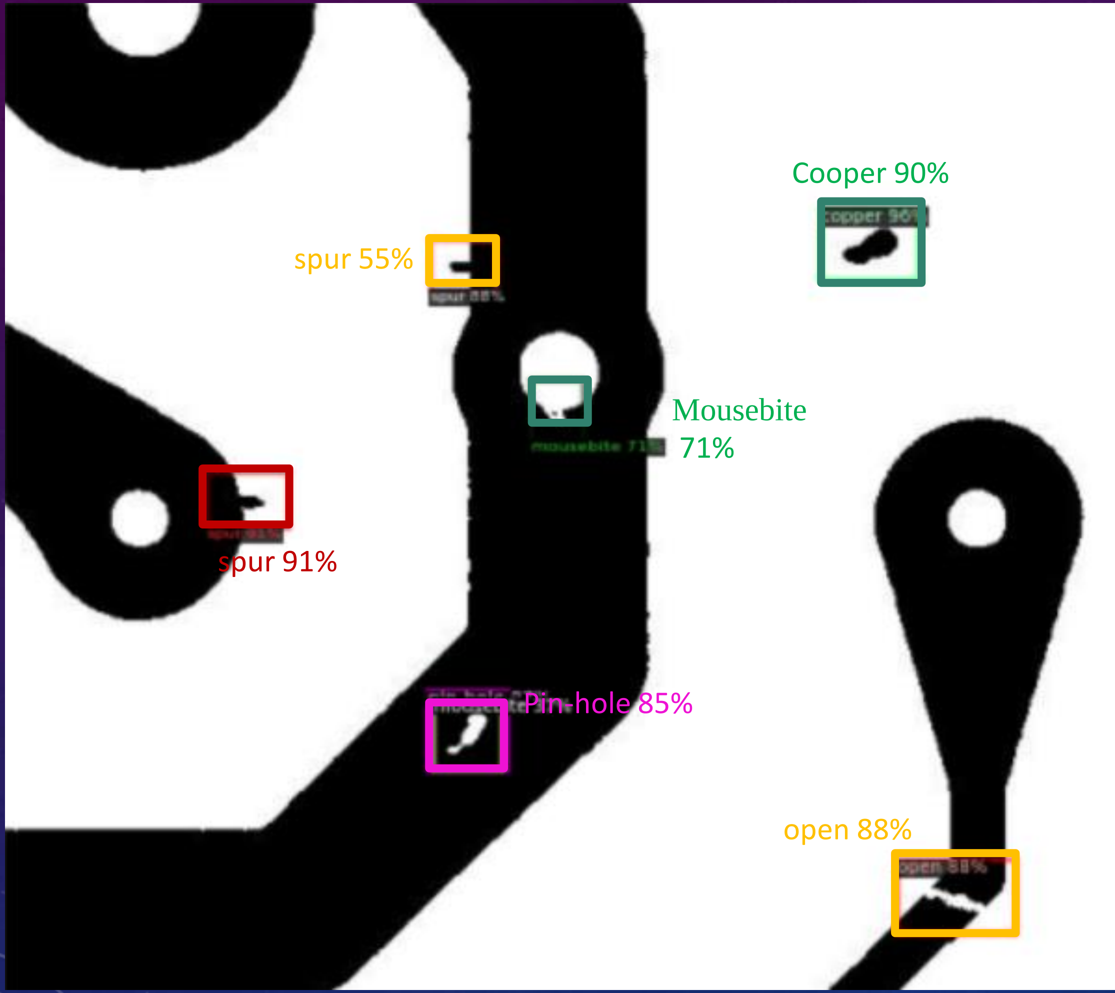
When starting to training , the colab will print the information for every 20 iterations

```
)8:57:18 d2.engine.train_loop]: Starting training from iteration 0
)8:57:26 d2.utils.events]:  eta: 0:18:07  iter: 19  total_loss: 2.585  loss_cls: 1.906  loss_box_reg: 0.042
)8:57:33 d2.utils.events]:  eta: 0:18:01  iter: 39  total_loss: 2.325  loss_cls: 1.703  loss_box_reg: 0.029
)8:57:40 d2.utils.events]:  eta: 0:17:50  iter: 59  total_loss: 1.812  loss_cls: 1.344  loss_box_reg: 0.034
)8:57:48 d2.utils.events]:  eta: 0:17:53  iter: 79  total_loss: 1.231  loss_cls: 0.869  loss_box_reg: 0.036
)8:57:55 d2.utils.events]:  eta: 0:17:50  iter: 99  total_loss: 0.912  loss_cls: 0.489  loss_box_reg: 0.038
)8:58:03 d2.utils.events]:  eta: 0:17:47  iter: 119  total_loss: 0.614  loss_cls: 0.247  loss_box_reg: 0.034
)8:58:11 d2.utils.events]:  eta: 0:17:48  iter: 139  total_loss: 0.466  loss_cls: 0.177  loss_box_reg: 0.068
)8:58:18 d2.utils.events]:  eta: 0:17:46  iter: 159  total_loss: 0.507  loss_cls: 0.176  loss_box_reg: 0.077
)8:58:27 d2.utils.events]:  eta: 0:17:47  iter: 179  total_loss: 0.521  loss_cls: 0.213  loss_box_reg: 0.098
```

```
loss_rpn_cls: 0.469  loss_rpn_loc: 0.154  time: 0.3619  data_time: 0.0438  lr: 0.000005  max_mem: 1826M
loss_rpn_cls: 0.416  loss_rpn_loc: 0.143  time: 0.3653  data_time: 0.0426  lr: 0.000010  max_mem: 1826M
loss_rpn_cls: 0.243  loss_rpn_loc: 0.135  time: 0.3647  data_time: 0.0437  lr: 0.000015  max_mem: 1826M
loss_rpn_cls: 0.162  loss_rpn_loc: 0.140  time: 0.3689  data_time: 0.0447  lr: 0.000020  max_mem: 1826M
loss_rpn_cls: 0.209  loss_rpn_loc: 0.122  time: 0.3682  data_time: 0.0422  lr: 0.000025  max_mem: 1826M
 loss_rpn_cls: 0.185  loss_rpn_loc: 0.132  time: 0.3712  data_time: 0.0452  lr: 0.000030  max_mem: 1826M
 loss_rpn_cls: 0.096  loss_rpn_loc: 0.119  time: 0.3742  data_time: 0.0428  lr: 0.000035  max_mem: 1826M
 loss_rpn_cls: 0.098  loss_rpn_loc: 0.094  time: 0.3757  data_time: 0.0424  lr: 0.000040  max_mem: 1826M
 loss_rpn_cls: 0.104  loss_rpn_loc: 0.084  time: 0.3790  data_time: 0.0440  lr: 0.000045  max_mem: 1826M
```

Example 4-1 – Faster RCNN

Visualizing the Testing Result



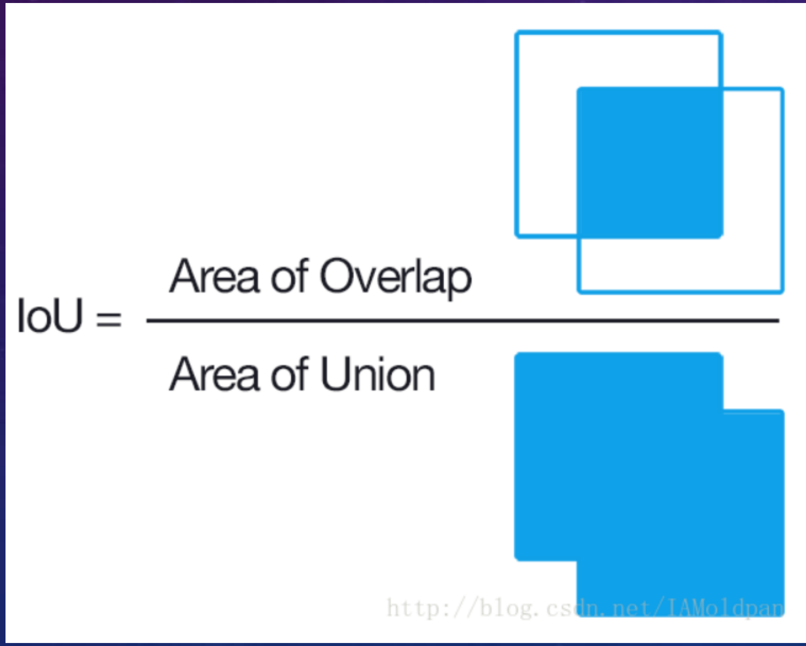

```python
from detectron2.utils.visualizer import ColorMode
dataset_dicts = get_PCB_dict(test)
```

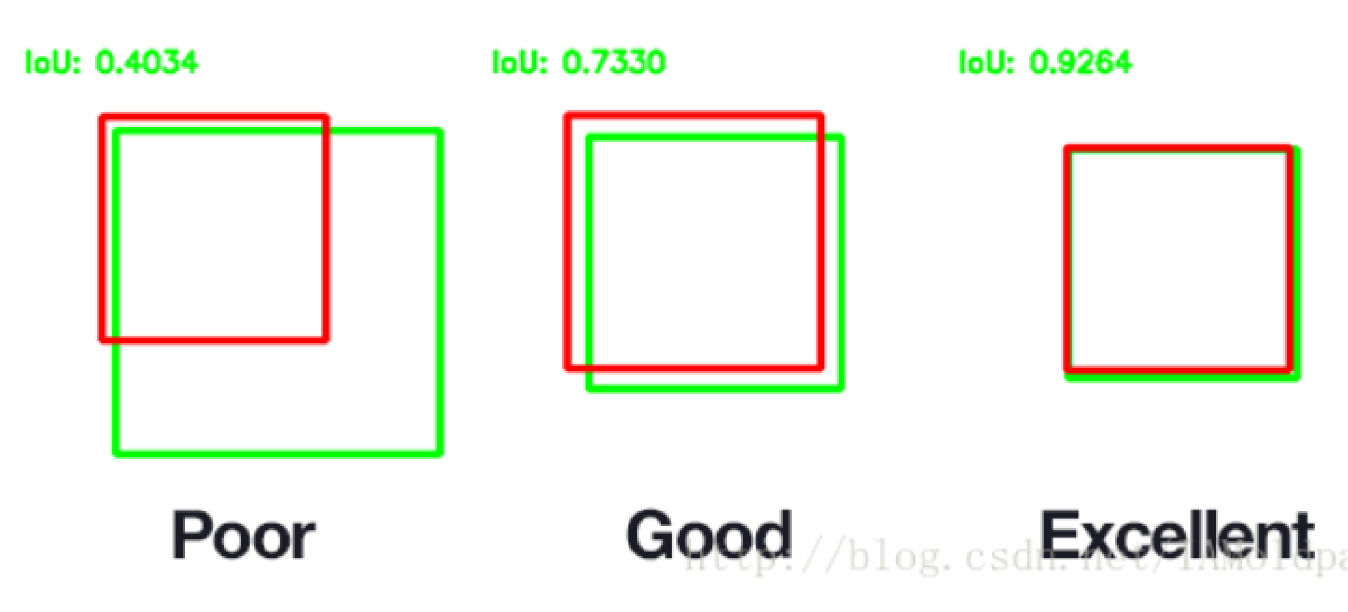
```python
for d in random.sample(dataset_dicts, 10):
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)
    v = Visualizer(im,
                   metadata=PCB_metadata,
                   scale=0.8,
                   instance_mode = ColorMode.IMAGE
    )
                   # remove the colors of unsegmented pixels
    print(outputs['instances'].pred_classes)
    print(outputs["instances"].pred_boxes)

    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(v.get_image())
```

Example 4-1 – Faster RCNN

- Formally we define confidence as *Pr*(Object) * *IOU*(pred, truth) . If no object exists in that cell, the confidence score should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

- IOU





Example 4-1 – Faster RCNN

```
if thresholds is None:
    step = 0.05
    thresholds = torch.arange(0.5, 0.95 + 1e-5, step, dtype=torch.float32)
recalls = torch.zeros_like(thresholds)
# compute recall for each iou threshold
for i, t in enumerate(thresholds):
    recalls[i] = (gt_overlaps >= t).float().sum() / float(num_pos)
# ar = 2 * np.trapz(recalls, thresholds)
ar = recalls.mean()
return {
    "ar": ar,
    "recalls": recalls,
    "thresholds": thresholds,
    "gt_overlaps": gt_overlaps,
    "num_pos": num_pos,
}
```

```
# Compute per-category AP
# from https://github.com/facebookresearch/Detectron/blob/a6a835f5b8208c45d0dce217
precisions = coco_eval.eval["precision"]
# precision has dims (iou, recall, cls, area range, max dets)
assert len(class_names) == precisions.shape[2]

results_per_category = []
for idx, name in enumerate(class_names):
    # area range index 0: all area ranges
    # max dets index -1: typically 100 per image
    precision = precisions[:, :, idx, 0, -1]
    precision = precision[precision > -1]
    ap = np.mean(precision) if precision.size else float("nan")
    results_per_category.append(("{}".format(name), float(ap * 100)))
```

Compute recall for each IOU threshold

Compute per-category AP

Different IOU thresholds, from 0.5 to 0.95, step 0.05

Example 4-1 – Faster RCNN

Evaluate the trained model

```
from detectron2.evaluation import COCOEvaluator, inference_on_dataset, LVISEvaluator
from detectron2.data import build_detection_test_loader

evaluator = COCOEvaluator("PCB_test", cfg, False, output_dir="./output/")
val_loader = build_detection_test_loader(cfg, "PCB_test")
inference_on_dataset(trainer.model, val_loader, evaluator)
```

Area : target detection area
Small : area<32*32
Medium : 32*32<area<96*96
Large : area>96*96

Result:

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.555
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.862
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.623
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.531
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.568
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.600
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.508
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.650
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.650
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.649
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.645
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.600
[10/26 07:08:07 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 55.525 | 86.206 | 62.343 | 53.122 | 56.825 | 60.000 |
[10/26 07:08:07 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category   | AP     | category   | AP     |
|:-----------|:-------|:-----------|:-------|:-----------|:-------|
| open       | 45.682 | short      | 30.675 | mousebite  | 53.424 |
| spur       | 56.391 | copper     | 82.105 | pin-hole   | 64.871 |
```

AP of each category

Exercise 4-1

- Please download the “PCBdata_fasterRCNN_colab.ipynb” from the Moodle and open by the Colab.
- Follow the Colab code and pip install packages for environments.
- Visualize the PCB data and define the hyper-parameter for training.
- Compare different iterations and comment on the results.

Please crop your results and code and paste to a MS Word, discuss the results, and then upload to the Moodles.

Example 4-2 – License Plate Detector

In this example, we want to use Faster RCNN to detect license plate. We have to learn how to label the license plate and we have to change the data into COCO data format. We will use pretrained weights, which were trained on the COCO dataset and retrain on our own data and test our own data.

Example 4-2 : License Plate - labelme

Install labelme

1. Python2 (Windows)

   ```
   pip install pyqt
   pip install labelme
   ```

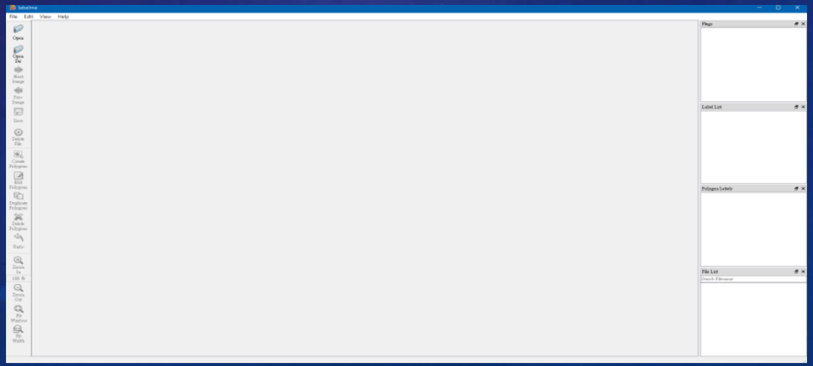
   Python 3 (Windows)

   ```
   pip install pyqt5
   pip install labelme
   ```

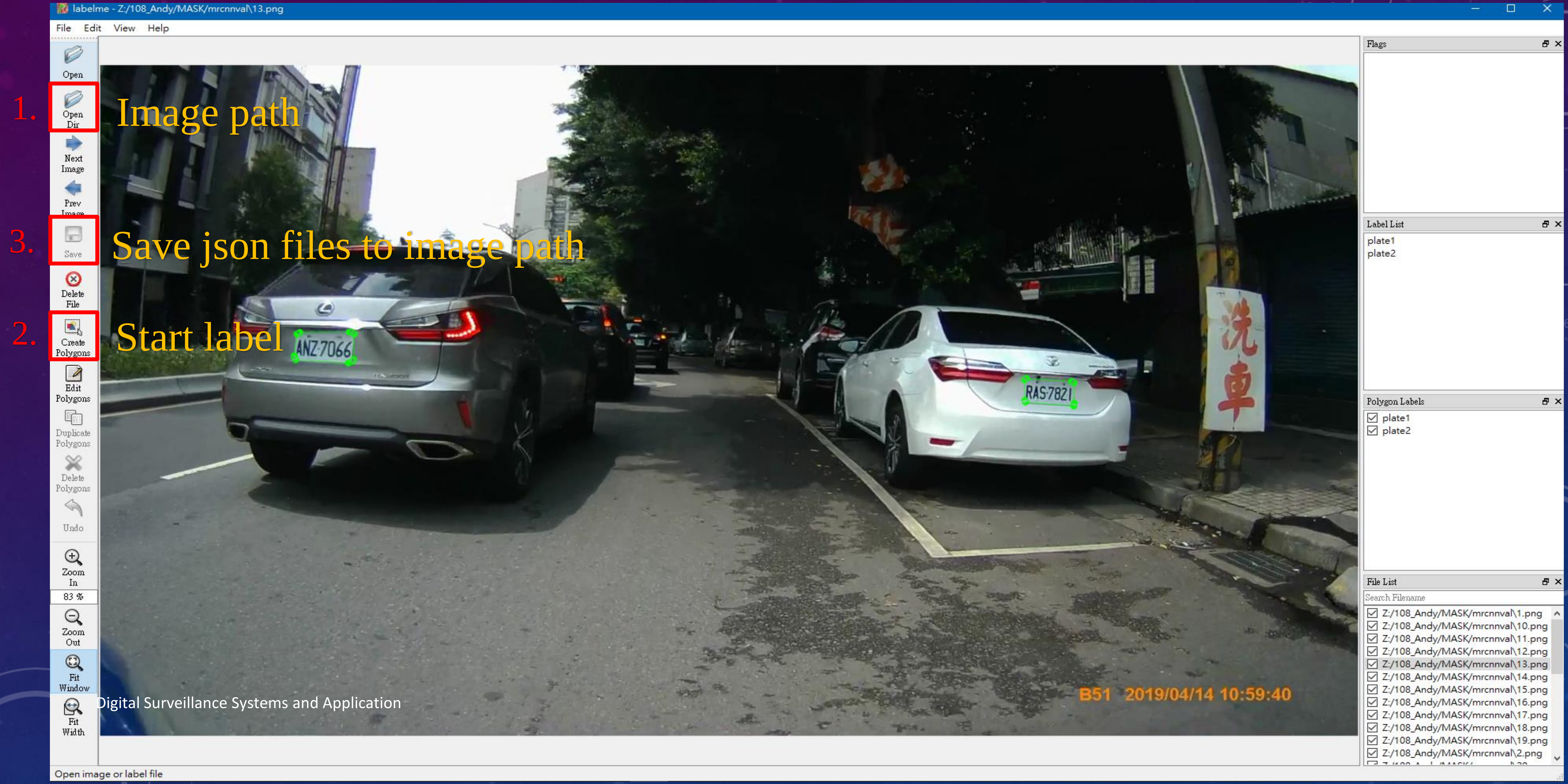
2. Cmd : labelme

   ```
   (pythonGPU) C:\Users\andy>labelme
   ```

3. Start to label



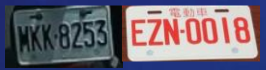
Example 4-2 : Start to label License Plate



Example 4-2 : How to label License Plate

For example

Naming method:
(name+order)

Label list: 

(normal class)plate1 plate2 ….

(yellow class)yellow_plate1 yellow_plate2 ….

(green class)green_plate1 green_plate2 ….

(red class)red_plate1 red_plate2 ….

Example 4-2 : Label Process

Step1:check the number of clear license plates in this image

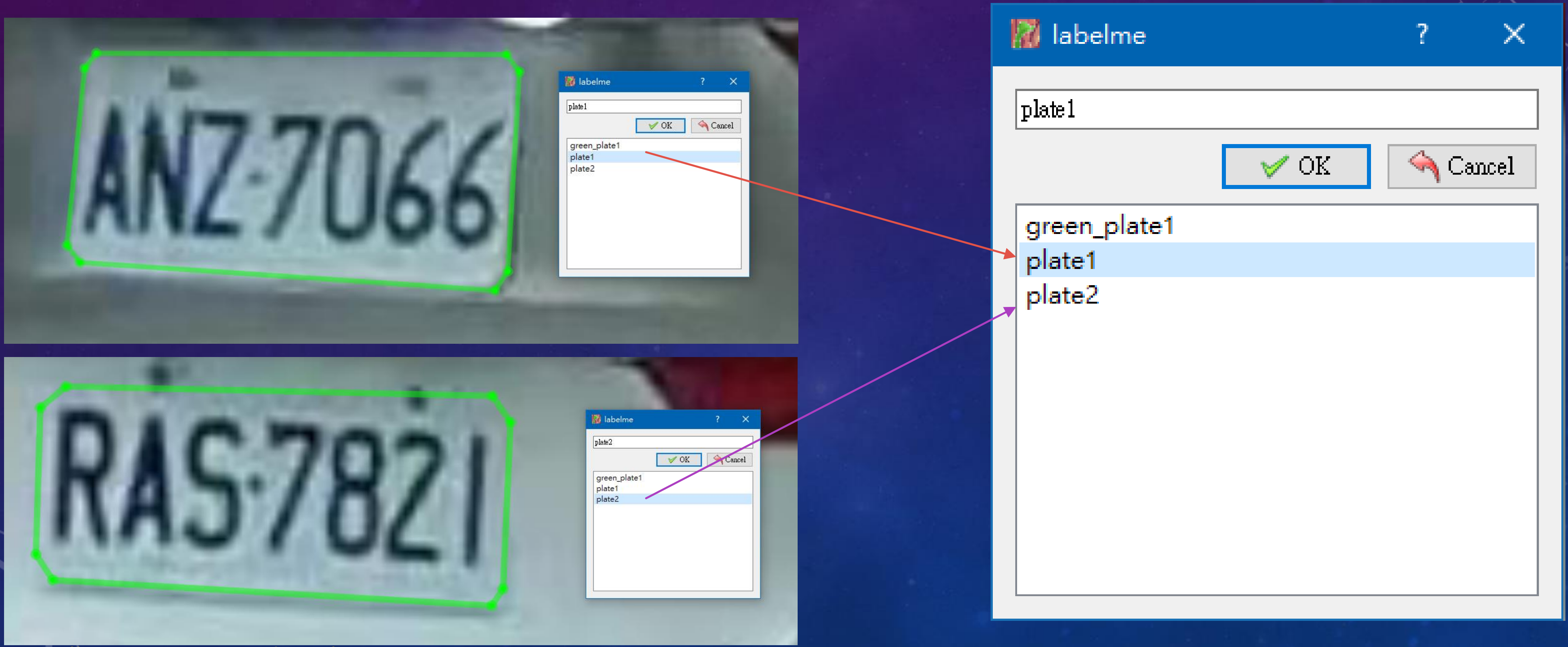
We can get two samples from this image

- There're two clear license plates
- All the license plate are normal class so we can named plate1 and plate 2

Example 4-2 : Label Process

Step2:Partial enlargement the license plate and use the dots to bound it

Step3:Give the license plate their name and order



Example 4-2 : Label Process

From the labelme we can get several json files that contain bounding boxes in the images (1 json/img). If you want to use multiple json files for combining multiple images, please consider using labelme2coco.py (given in the labelme2coco.7z from Moodle)

Step1: Put your image folder in the labelme2coco folder

Step2: Make sure your json files and images are in the same folder

Step3: Open cmd and run labelme2coco.py [folder name], as shown below

```
(C:\Users\Micky\Anaconda3\envs\Pytorch10) Y:\108_Andy\henry\labelme2coco-master>python labelme2coco.py images
save coco json
via_region_data.json
```

The images folder name

Result : The file via_region_data.json combines all json files with the associated images.

via_region_data.json

Example 4-2：License Plate Detector

Step 1 : Set up environment

Install pytorch and detectron2

Step 2 : Prepare the “license plate dataset”

Download the dataset from https://drive.google.com/file/d/14wXRkpow5vlVc2ROhHbWelEjwCJCx9b5/view?usp=sharing

Step 3 : Write a function that loads the dataset into detectron2's standard format

Step 4 : Verify the data loading is correct

Step 5 : Fine-tune a pretrained model on the license plate dataset

Step 6 : Visualize the prediction results.

Example 4-2：License Plate Detector

Step 1 : Set up environment

```
!pip install -U torch torchvision
!pip install git+https://github.com/facebookresearch/fvcore.git
import torch, torchvision
torch.__version__
```

Result :

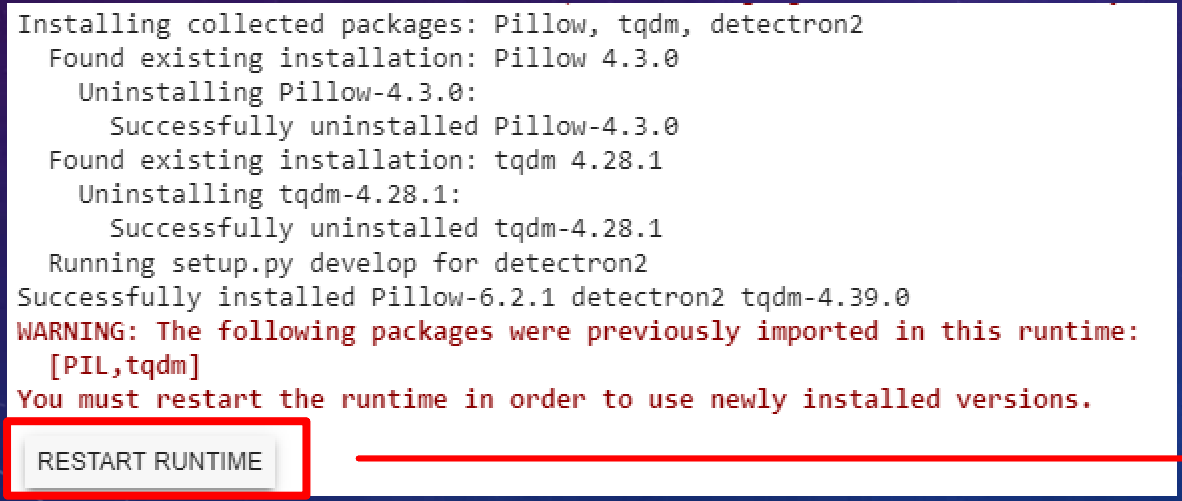
```
Successfully built fvcore pyyaml
Installing collected packages: pyyaml, yacs, portalocker, fvcore
  Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed fvcore-0.1 portalocker-1.5.2 pyyaml-5.1.2 yacs-0.1.6
'1.3.1'
```

Example 4-2：License Plate Detector

Step 1 : Set up environment

```
!git clone https://github.com/facebookresearch/detectron2 detectron2_repo
!pip install -e detectron2_repo
```

Result :



```
Installing collected packages: Pillow, tqdm, detectron2
  Found existing installation: Pillow 4.3.0
    Uninstalling Pillow-4.3.0:
      Successfully uninstalled Pillow-4.3.0
  Found existing installation: tqdm 4.28.1
    Uninstalling tqdm-4.28.1:
      Successfully uninstalled tqdm-4.28.1
  Running setup.py develop for detectron2
Successfully installed Pillow-6.2.1 detectron2 tqdm-4.39.0
WARNING: The following packages were previously imported in this runtime:
  [PIL,tqdm]
You must restart the runtime in order to use newly installed versions.
```


If you have already installed the detectron2, you need to "restart runtime".

Example 4-2：License Plate Detector

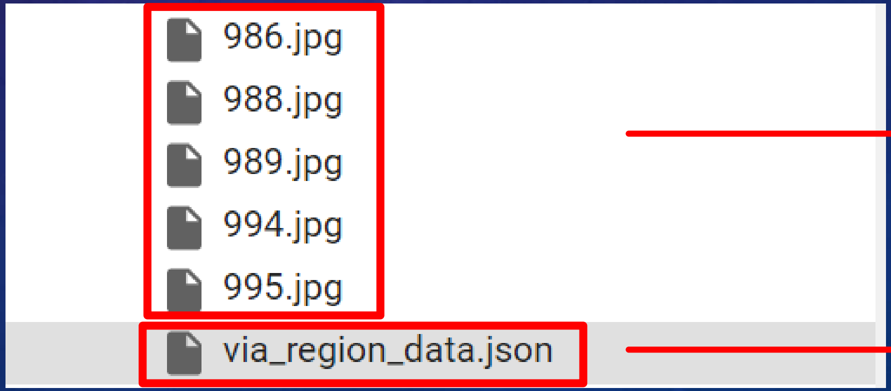
Step 2 : Prepare the “license plate dataset”

Option 1 : run !wget https://www.dropbox.com/s/rjra44iyzoh19te/plate.zip?dl=0 -O plate.zip

Option 2 : Download the plate.zip from
https://drive.google.com/file/d/14wXRkpow5vlVc2ROhHbWelEjwCJCx9b5/view?usp=sharing and upload it
Unzip plate.zip.
!unzip plate.zip

Result :

In plate/train



Training data

Our label file

Example 4-2：License Plate Detector

Step 3 : Write a function that loads the dataset into detectron2's standard format

```python
def get_plate_dicts(img_dir):
    json_file = os.path.join(img_dir, "via_region_data.json")  # via_region_data.json
    with open(json_file) as f:
        imgs_anns = json.load(f)
```

Open the “via_region_data.json”

```python
for i in range(len(filename_list)):
    record = {}
    filename = os.path.join(img_dir, filename_list[i]["file_name"])
    height, width = cv2.imread(filename).shape[:2]
    record["file_name"] = filename
    record["image_id"] = i
    record["height"] = 1080
    record["width"] = 1920
```

In plate/train/via_region_data.json


```
"images": [
    {
        "height": 1080,
        "width": 1920,
        "id": 0,
        "file_name": "10.jpg"
    },
```

Example 4-2：License Plate Detector

Step 3 : Write a function that loads the dataset into detectron2's standard format

```python
for j in range(k,k+bbox_num[i]):
    if bbox_list[j]["image_id"] != record["image_id"]:
        assert False
    bbox = []
    bbox = [int(bbox_list[j]["bbox"][0]),int(bbox_list[j]["bbox"][1]),
            int(bbox_list[j]["bbox"][0])+int(bbox_list[j]["bbox"][2]),
            int(bbox_list[j]["bbox"][1])+int(bbox_list[j]["bbox"][3])]
    obj = {
        "bbox":bbox,
        "bbox_mode": BoxMode.XYXY_ABS,
        "segmentation":bbox_list[j]["segmentation"],
        "category_id":0,
        "iscrowd": 0
    }
    objs.append(obj)
record["annotations"] = objs
k += bbox_num[i]
dataset_dicts.append(record)
```

In plate/train/via_region_data.json

```
"bbox": [
    710.0,
    283.0,
    128.0,
    81.0
],
```

The width and height of plates

```
"segmentation": [
    [
        732.9621380846324,
        283.5189309576837,
        723.8307349665924,
        283.2962138084632,
        710.467706013363,
        327.8396436525612,
        714.6993318485523,
        331.62583518930956,
        815.5902004454342,
        364.81069042316255,
        821.826280623608,
        361.2472160356347,
        838.3073496659242,
        316.9265033407572,
        835.1893095768373,
        312.6948775055679
    ]
```

Example 4-2：License Plate Detector

Step 4 : Verify the data in the training folder

```
import random
dataset_dicts = get_plate_dicts("plate/train")
for d in random.sample(dataset_dicts, 3):
    img = cv2.imread(d["file_name"])
    visualizer = Visualizer(img[:, :, ::-1], metadata=plate_metadata, scale=0.5)
    vis = visualizer.draw_dataset_dict(d)
    cv2_imshow(vis.get_image()[:, :, ::-1])
```

Randomly select 3 samples to verify.

Result :

Example 4-2：License Plate Detector

Step 5 : Fine-tune a pretrained model on the license plate dataset

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

cfg = get_cfg()
cfg.merge_from_file("./detectron2_repo/configs/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
cfg.DATASETS.TRAIN = ("plate_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 5
cfg.MODEL.WEIGHTS = "detectron2://COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x/137849600/model_final_f10217.pkl"
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 2000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 256
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1  # only has one class

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

Define the max iterations of training

Pretrained model

Example 4-2：License Plate Detector

Step 6 : Visualize the prediction results.

```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set the testing threshold for this model
cfg.DATASETS.TEST = ("plate_val", )
predictor = DefaultPredictor(cfg)
```

Load trained model

```
from detectron2.utils.visualizer import ColorMode
dataset_dicts = get_plate_dicts("plate/val")
for d in random.sample(dataset_dicts, 10):
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1],
                   metadata=plate_metadata,
                   scale=0.8,
                   instance_mode=ColorMode.IMAGE_BW
    )
    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(v.get_image()[:, :, ::-1])
```

Randomly select 10 samples to test.

Example 4-2：License Plate Detector

Result :

Exercise 4-2：License Plate Detector

- Please download “Detectron2_license_plate.ipynb” and “test_plate.zip” and open “Detectron2_license_plate.ipynb” by Colab.
- Use labelme tool to label the 10 images by yourself and test it on trained model.

  ※Hint: If you want to test, you need to upload your images and json file into plate/val folder.

- Please write down result and your code in MS Word to Moodle

Example 4-3 – Detectron2 Tutorial

In this example, we will use detectron2 to fine-tune the balloon dataset, and then detect it. In addition, we will also test different pre-trained model, keypoint detection model, and try to run keypoint detection on a video.

Example 4-3 – Detectron2 Tutorial

Set up environment

```
# install dependencies:
!pip install pyyaml==5.1
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
# opencv is pre-installed on colab
```

```
# install detectron2: (Colab has CUDA 10.1 + torch 1.7)
# See https://detectron2.readthedocs.io/tutorials/install.html for instructions
import torch
assert torch.__version__.startswith("1.7")
!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.7/index.html
# exit(0)  # After installation, you need to "restart runtime" in Colab. This line can also restart runtime
```

```
# 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
```

Example 4-3 – Detectron2 Tutorial

Train on a custom dataset:

In this section, we show how to train an existing detectron2 model on a custom dataset in a new format.

We use the balloon segmentation dataset which only has one class: balloon.

We'll train a balloon segmentation model from an existing model pre-trained on COCO dataset, available in detectron2's model zoo.

Prepare the dataset:

```
# download, decompress the data
!wget https://github.com/matterport/Mask_RCNN/releases/download/v2.1/balloon_dataset.zip
!unzip balloon_dataset.zip > /dev/null
```

Example 4-3 – Detectron2 Tutorial

Register the balloon dataset to detectron2:

```
from detectron2.structures import BoxMode

def get_balloon_dicts(img_dir):
    json_file = os.path.join(img_dir, "via_region_data.json")
    with open(json_file) as f:
        imgs_anns = json.load(f)

    dataset_dicts = []
    for idx, v in enumerate(imgs_anns.values()):
        record = {}

        filename = os.path.join(img_dir, v["filename"])
        height, width = cv2.imread(filename).shape[:2]

        record["file_name"] = filename
        record["image_id"] = idx
        record["height"] = height
        record["width"] = width

        annos = v["regions"]
        objs = []
        for _, anno in annos.items():
            assert not anno["region_attributes"]
            anno = anno["shape_attributes"]
            px = anno["all_points_x"]
            py = anno["all_points_y"]
            poly = [(x + 0.5, y + 0.5) for x, y in zip(px, py)]
            poly = [p for x in poly for p in x]
```

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

for d in ["train", "val"]:
    DatasetCatalog.register("balloon_" + d, lambda d=d: get_balloon_dicts("balloon/" + d))
    MetadataCatalog.get("balloon_" + d).set(thing_classes=["balloon"])
balloon_metadata = MetadataCatalog.get("balloon_train")
```

Open "via_region_data.json" in the train and val folder

Example 4-3 – Detectron2 Tutorial

Verify the data in the training folder

```
import random
dataset_dicts = get_plate_dicts(“ballon/train")
for d in random.sample(dataset_dicts, 3):
    img = cv2.imread(d["file_name"])
    visualizer = Visualizer(img[:, :, ::-1], metadata=plate_metadata, scale=0.5)
    vis = visualizer.draw_dataset_dict(d)
    cv2_imshow(vis.get_image()[:, :, ::-1])
```

Randomly select 3 samples to verify.

Result :

Example 4-3 – Detectron2 Tutorial

Fine-tune a pretrained model on the ballon dataset

```
from detectron2.engine import DefaultTrainer

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("balloon_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025  # pick a good LR
cfg.SOLVER.MAX_ITER = 300    # 300 iterations seems good enough for this toy dataset; you wi
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128   # faster, and good enough for this toy dataset
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1  # only has one class (ballon). (see https://detectron2.readt
cfg.MODEL.ANCHOR_GENERATOR.ASPECT_RATIOS=[[0.5, 1.0, 2.0]]
cfg.MODEL.ANCHOR_GENERATOR.SIZES=[[32], [64], [128], [256], [512]]
print(cfg)
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

Pretrained model

Define the max iterations of training

Only one class (ballon)

Define different anchor ratios

Define different anchor size

Example 4-3 – Detectron2 Tutorial

You can **print (cfg)** and find that you can adjust more different parameters for training.

```
CUDNN_BENCHMARK: False
DATALOADER:
  ASPECT_RATIO_GROUPING: True
  FILTER_EMPTY_ANNOTATIONS: True
  NUM_WORKERS: 2
  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: ('balloon_train',)
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]]
```

cfg.MODEL.ANCHOR_GENERATOR.ASPECT_RATIOS=[[0.5, 1.0, 2.0]]

cfg.MODEL.ANCHOR_GENERATOR.SIZES=[[32], [64], [128], [256], [512]]

You can get 15 different anchor boxes(3*5)

Example 4-3 – Detectron2 Tutorial

Visualize the prediction results

Load trained model

```
# Inference should use the config with parameters that are used in training
# cfg now already contains everything we've set previously. We changed it a lit
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")  # path to the
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set a custom testing threshold
predictor = DefaultPredictor(cfg)
```

Set a testing threshlod

```
from detectron2.utils.visualizer import ColorMode
dataset_dicts = get_balloon_dicts("balloon/val")
for d in random.sample(dataset_dicts, 3):
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)  # format is documented at https://dete
    v = Visualizer(im[:, :, ::-1],
                   metadata=balloon_metadata,
                   scale=0.5,
                   instance_mode=ColorMode.IMAGE_BW   # rem
    )
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(out.get_image()[:, :, ::-1])
```

Randomly select 3 samples to test.

Example 4-3 – Detectron2 Tutorial

Evaluate the trained model

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

Area : target detection area
Small : area<32*32
Medium : 32*32<area<96*96
Large : area>96*96

Result:

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.691
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.845
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.825
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.029
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.550
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.837
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.234
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.730
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.766
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.233
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.659
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.880
[11/20 15:14:37 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 69.075 | 84.529 | 82.475 | 2.871 | 54.958 | 83.651 |
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.778
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.842
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.833
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.012
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.581
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.957
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.252
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.792
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.824
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.200
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.682
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.967
[11/20 15:14:37 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 77.839 | 84.201 | 83.293 | 1.238 | 58.077 | 95.700 |
```

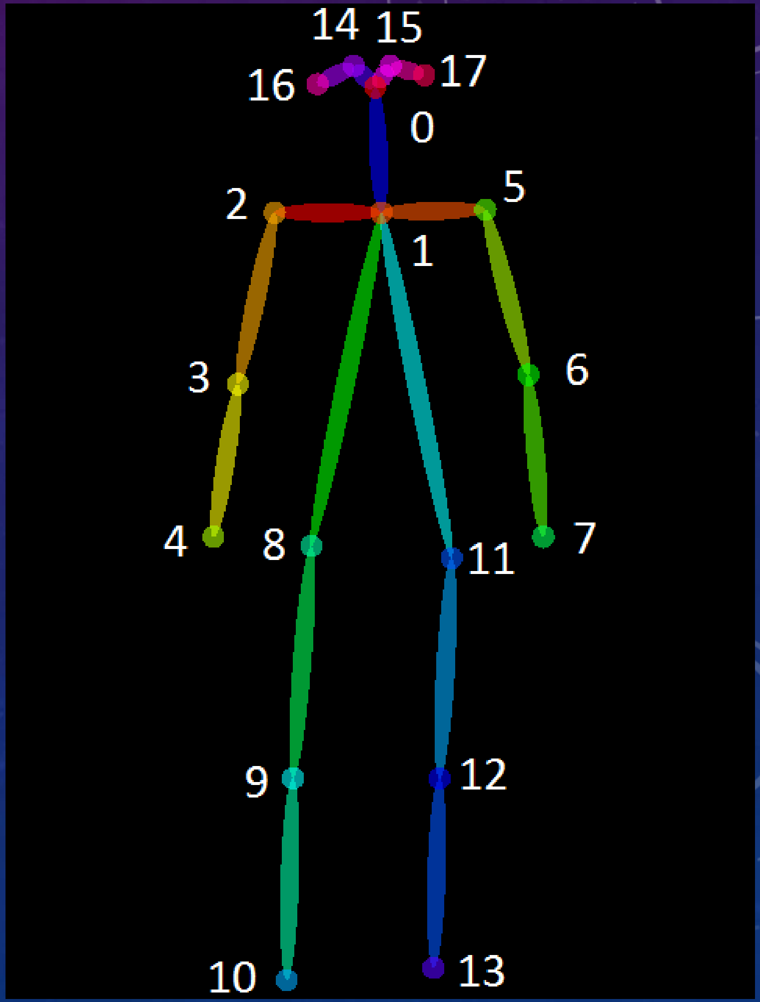
Example 4-3 – Detectron2 Tutorial

Keypoint detection model

- Train in COCO dataset (250,000 people with keypoints)
- 17 types of keypoints

Contains:

“nose", "left_eye", "right_eye", "left_ear", "right_ear", "left_shoulder", "right_shoulder", "left_elbow", "right_elbow", "left_wrist", "right_wrist", "left_hip", "right_hip", "left_knee", "right_knee", "left_ankle", "right_ankle“.



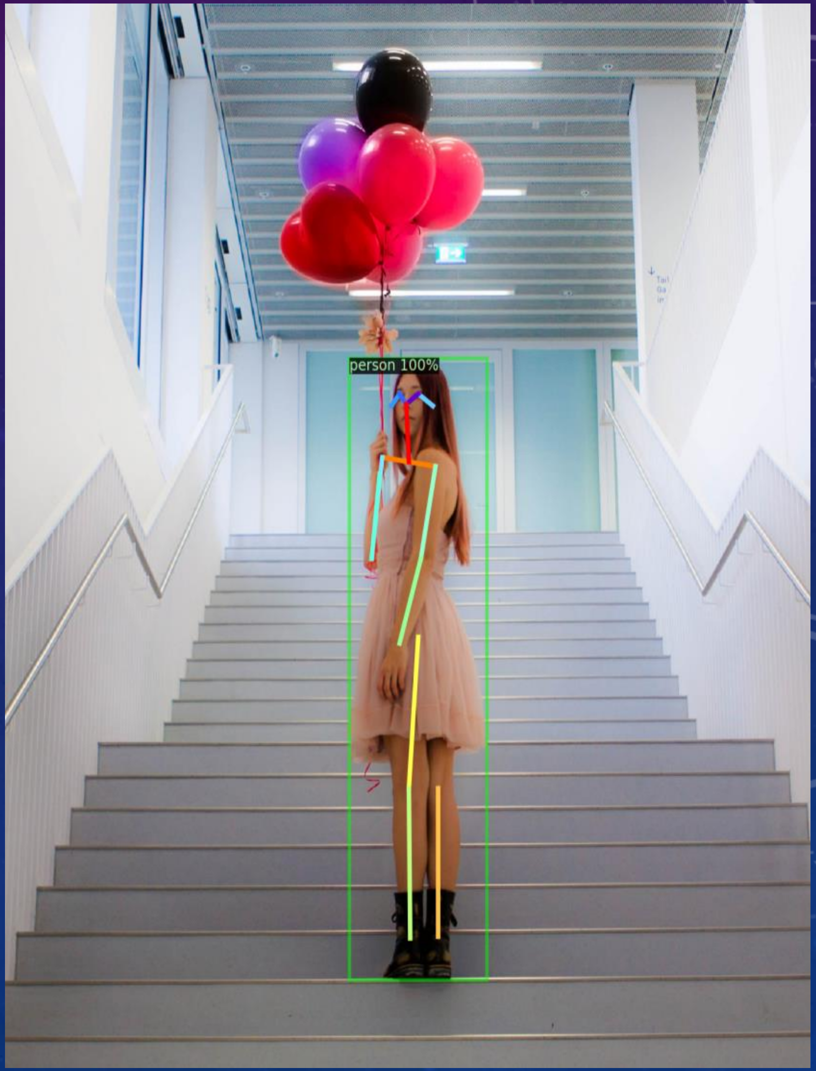
Example 4-3 – Detectron2 Tutorial

Other types of models- keypoint detection model

```
# Inference with a keypoint detection model
cfg = get_cfg()    # get a fresh new config
cfg.merge_from_file(model_zoo.get_config_file("COCO-Keypoints/keypoint_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set threshold for this model
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Keypoints/keypoint_rcnn_R_50_FPN_3x.yaml")
predictor = DefaultPredictor(cfg)
outputs = predictor(im)
v = Visualizer(im[:,:,::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```

Set a testing threshlod

Load pretrain model train on COCO



Example 4-3 – Detectron2 Tutorial

Run panoptic segmentation on a video:

Prepare the video URL

```
!pip  install  youtube-dl
!pip  uninstall  -y  opencv-python-headless  opencv-contrib-python
!apt  install  python3-opencv    #  the  one  pre-installed  have  some  issues
!youtube-dl  https://www.youtube.com/watch?v=ll8TgCZ0plk  -f  22  -o  video.mp4
!ffmpeg  -i  video.mp4  -t  00:00:06  -c:v  copy  video-clip.mp4
```

Youtube URL

Use demo.py tool run frame-by-frame inference demo on this video

```
#  Run  frame-by-frame  inference  demo  on  this  video  (takes  3-4  minutes)  with  the  "demo.py"  tool  we  provided  in  the  repo.
!git  clone  https://github.com/facebookresearch/detectron2
!python  detectron2/demo/demo.py  --config-file  detectron2/configs/COCO-Keypoints/keypoint_rcnn_R_101_FPN_3x.yaml  --video-input  video-clip.mp4  --confidence-threshold  0.6
    --opts  MODEL.WEIGHTS  detectron2://COCO-Keypoints/keypoint_rcnn_R_101_FPN_3x/138363331/model_final_997cc7.pkl
```

Download the results

```
from  google.colab  import  files
files.download('video-output.mkv')
```

Model architecture

Model weight

Example 4-3 – Detectron2 Tutorial

Example 4-3 – Detectron2 Tutorial

Model_zoo.py

```
# COCO Detection with RetinaNet
"COCO-Detection/retinanet_R_50_FPN_1x.yaml": "190397773/model_final_bfca0b.pkl",
"COCO-Detection/retinanet_R_50_FPN_3x.yaml": "190397829/model_final_5bd44e.pkl",
"COCO-Detection/retinanet_R_101_FPN_3x.yaml": "190397697/model_final_971ab9.pkl",
# COCO Detection with RPN and Fast R-CNN
"COCO-Detection/rpn_R_50_C4_1x.yaml": "137258005/model_final_450694.pkl",
"COCO-Detection/rpn_R_50_FPN_1x.yaml": "137258492/model_final_02ce48.pkl",
"COCO-Detection/fast_rcnn_R_50_FPN_1x.yaml": "137635226/model_final_e5f7ce.pkl",
# COCO Instance Segmentation Baselines with Mask R-CNN
"COCO-InstanceSegmentation/mask_rcnn_R_50_C4_1x.yaml": "137259246/model_final_9243eb.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_50_DC5_1x.yaml": "137260150/model_final_4f86c3.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_1x.yaml": "137260431/model_final_a54504.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_50_C4_3x.yaml": "137849525/model_final_4ce675.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_50_DC5_3x.yaml": "137849551/model_final_84107b.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml": "137849600/model_final_f10217.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_101_C4_3x.yaml": "138363239/model_final_a2914c.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_101_DC5_3x.yaml": "138363294/model_final_0464b7.pkl",
"COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml": "138205316/model_final_a3ec72.pkl",
"COCO-InstanceSegmentation/mask_rcnn_X_101_32x8d_FPN_3x.yaml": "139653917/model_final_2d9806.pkl",  # noqa
# COCO Person Keypoint Detection Baselines with Keypoint R-CNN
"COCO-Keypoints/keypoint_rcnn_R_50_FPN_1x.yaml": "137261548/model_final_04e291.pkl",
"COCO-Keypoints/keypoint_rcnn_R_50_FPN_3x.yaml": "137849621/model_final_a6e10b.pkl",
"COCO-Keypoints/keypoint_rcnn_R_101_FPN_3x.yaml": "138363331/model_final_997cc7.pkl",
"COCO-Keypoints/keypoint_rcnn_X_101_32x8d_FPN_3x.yaml": "139686956/model_final_5ad38f.pkl",
# COCO Panoptic Segmentation Baselines with Panoptic FPN
"COCO-PanopticSegmentation/panoptic_fpn_R_50_1x.yaml": "139514544/model_final_dbfeb4.pkl",
"COCO-PanopticSegmentation/panoptic_fpn_R_50_3x.yaml": "139514569/model_final_c10459.pkl",
"COCO-PanopticSegmentation/panoptic_fpn_R_101_3x.yaml": "139514519/model_final_cafdb1.pkl",
# LVIS Instance Segmentation Baselines with Mask R-CNN
"LVISv0.5-InstanceSegmentation/mask_rcnn_R_50_FPN_1x.yaml": "144219072/model_final_571f7c.pkl",  # noqa
"LVISv0.5-InstanceSegmentation/mask_rcnn_R_101_FPN_1x.yaml": "144219035/model_final_824ab5.pkl",  # noqa
"LVISv0.5-InstanceSegmentation/mask_rcnn_X_101_32x8d_FPN_1x.yaml": "144219108/model_final_5e3439.pkl",  # noqa
```

Model weight

Model architecture

Exercise 4-3 – Detectron2 Tutorial

- Please download “Detectron2_tutorial.ipynb” and open “Detectron2_tutorial.ipynb” by Colab.

1.Try to use **your own** pictures to test on keypoint detection models.

※ Hints: You must change random.sample to the specified picture.

2.Following the previous question change the cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST and compare the result.

3.Use different types of videos for key points model (8sec) .

4.Following the previous question choose different pretrain model to test.

※ Hints: Different weights and networks can be found at the following URL.

https://github.com/facebookresearch/detectron2/blob/master/detectron2/model_zoo/model_zoo.py

Exercise 4-3 – Detectron2 Tutorial

5. Retrain the model and change anchor box size and ratio then compare the result.

6. Following the previous question, retrain the new model on different cfg on the model and compare the influence on the model (In addition to the existing cfg, need 3 different cfg changes).

- Please write down result and your code in MS Word to Moodle