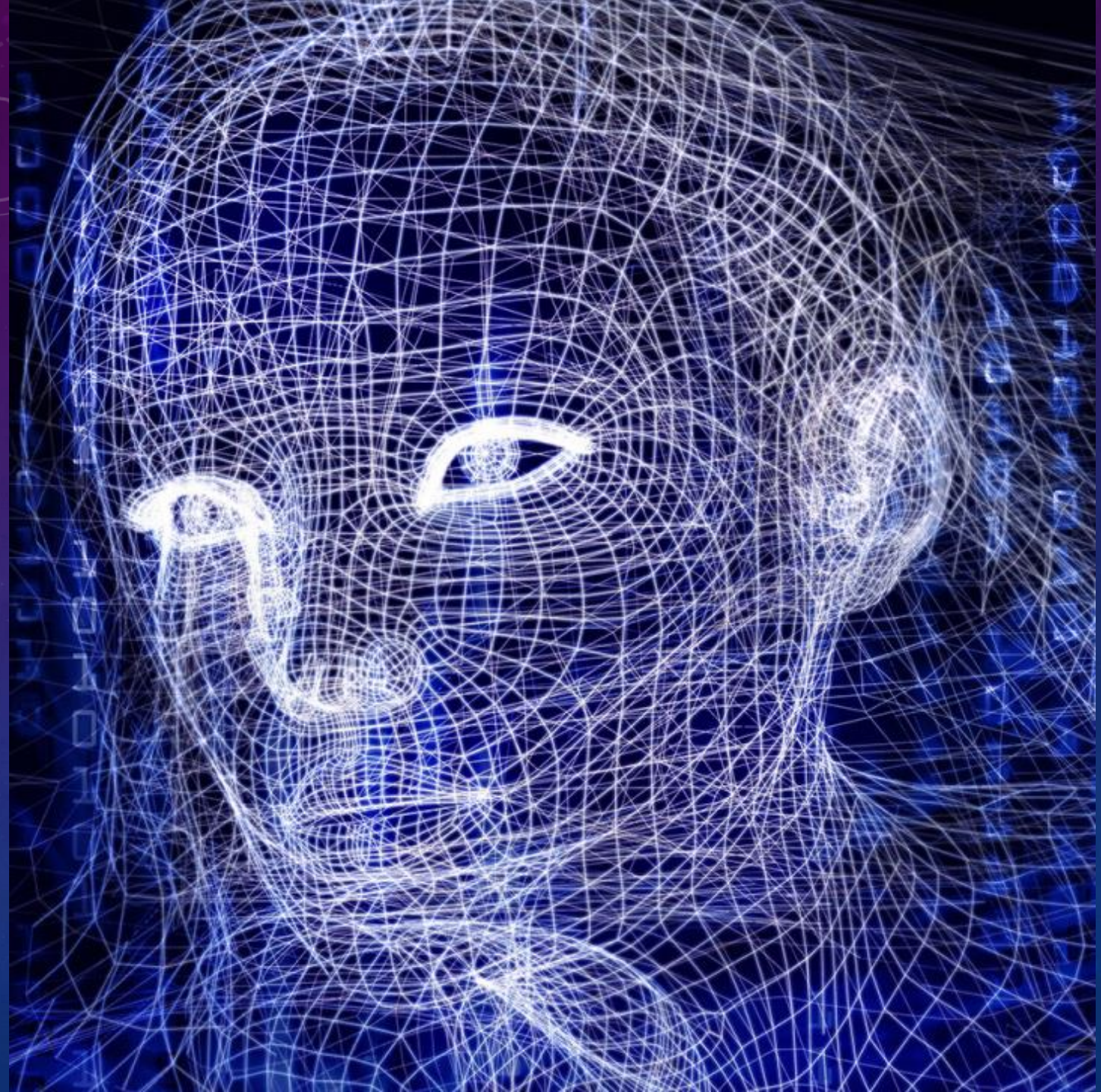
LECTURE SERIES FOR DIGITAL SURVEILLANCE SYSTEMS AND APPLICATION

CH4 EXPERIMENT FOR OBJECT DETECTION



徐繼聖

Gee-Sern Jison Hsu

National Taiwan University of Science and Technology

Example 4-1 – Faster RCNN

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}]}
```

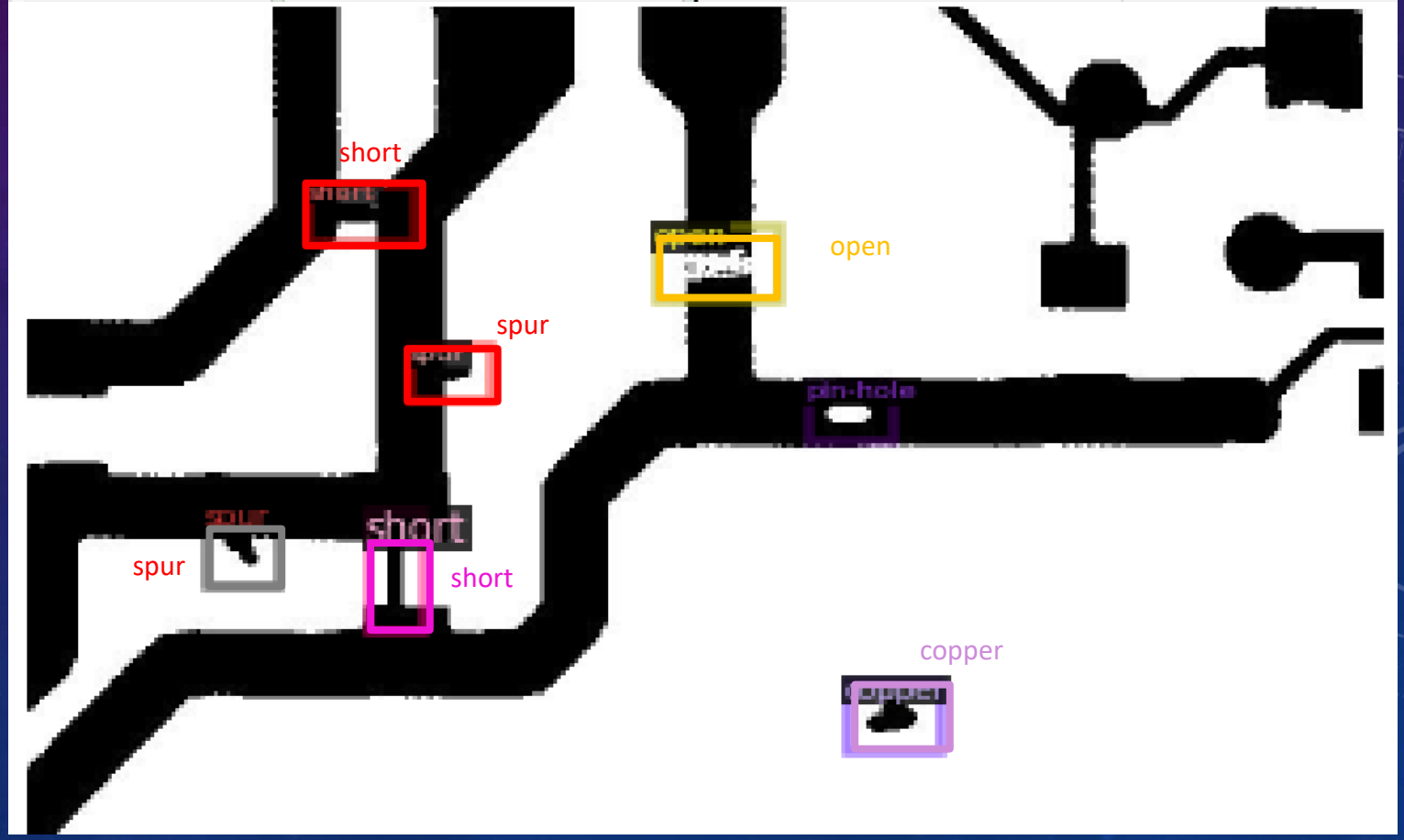

bbox: Defect bounding box coordinates
Bbox_mode: Bbox format
Category: Defect category
Iscrowd: 0: An object
1: A set of objects

Example 4-1 – Faster RCNN

Visualizing the Train Dataset

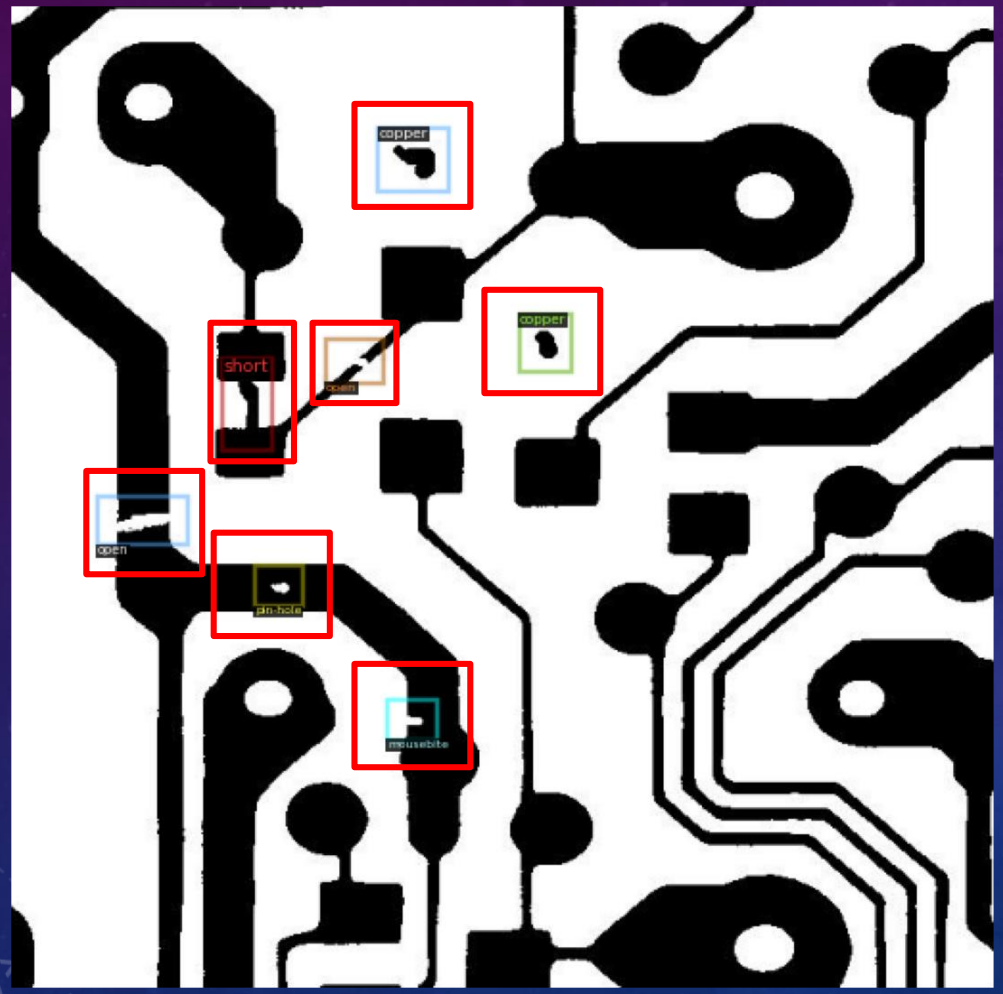
Randomly select 2 pictures from the train folder of the dataset and check the appearance of the bounding box.

```
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

PCB data sample:

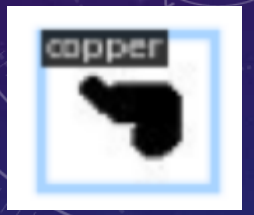


Category Type Sample:

Open

Spur

Short



Copper

Pin-hole

Copper

Mousebite

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

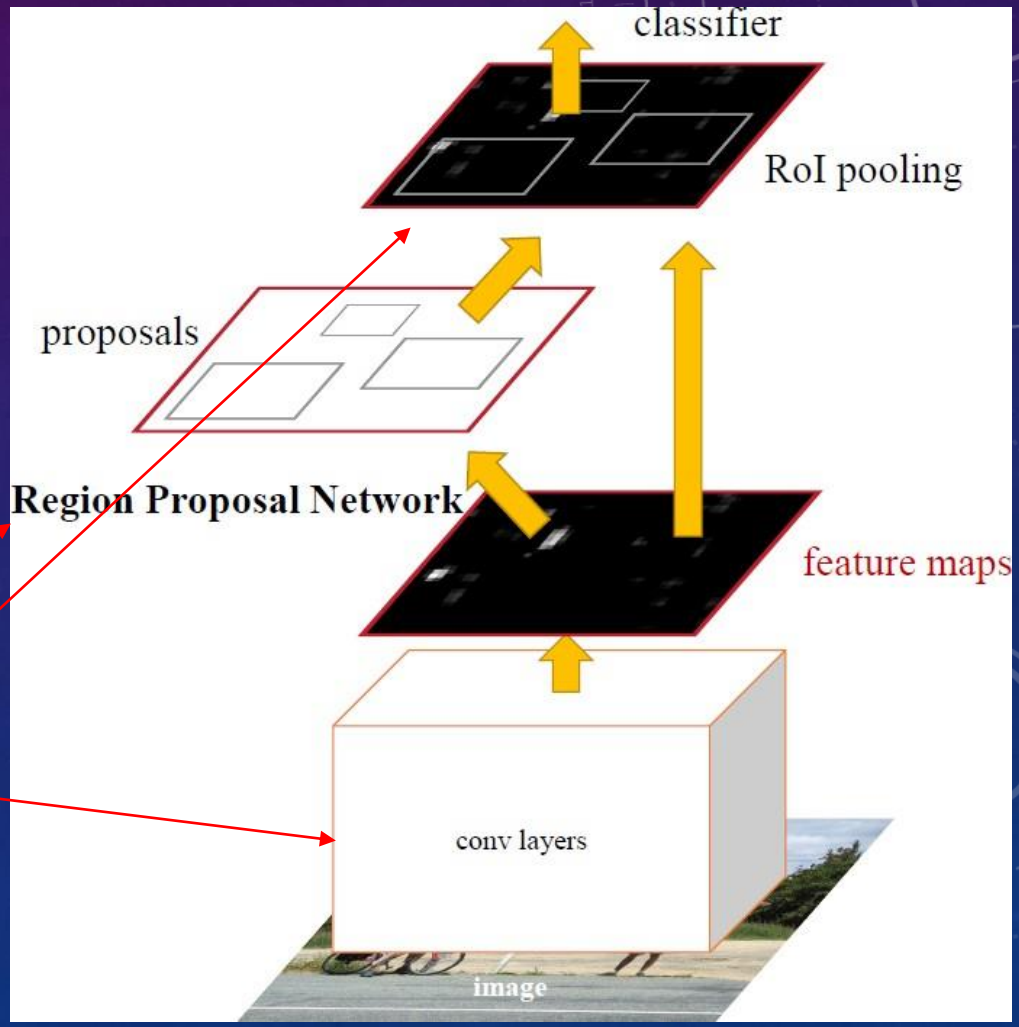
Example 4-1 – Faster RCNN

This conv layers architecture uses FPN

```python
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"
    ):
        """
```

```python
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)
```



Example 4-1 – Faster RCNN

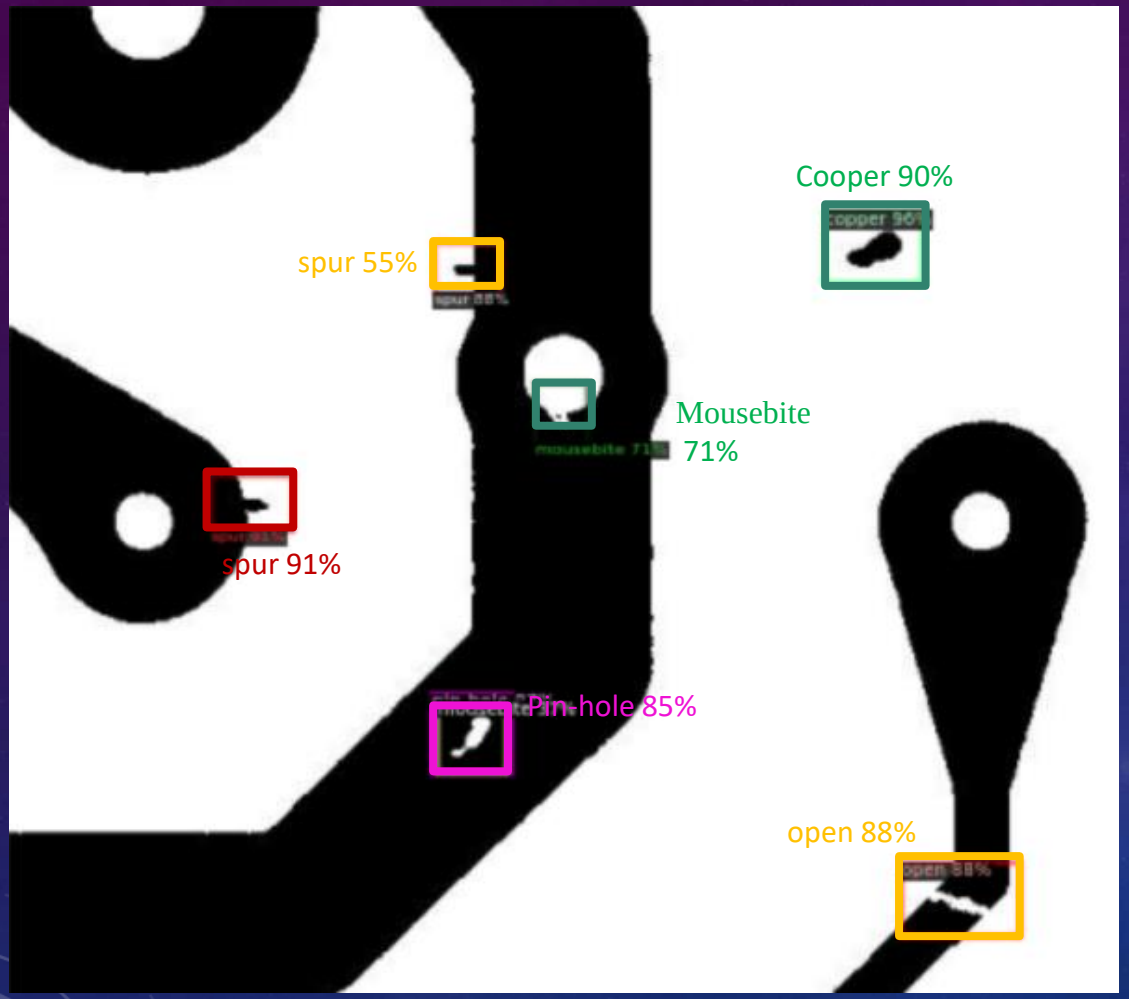
When starting to training , the colab will print the information for every 20 iterations

```
[08:57:18 d2.engine.train_loop]: Starting training from iteration 0
[08:57:26 d2.utils.events]:  eta: 0:18:07  iter: 19  total_loss: 2.585  loss_cls: 1.906  loss_box_reg: 0.042
[08:57:33 d2.utils.events]:  eta: 0:18:01  iter: 39  total_loss: 2.325  loss_cls: 1.703  loss_box_reg: 0.029
[08:57:40 d2.utils.events]:  eta: 0:17:50  iter: 59  total_loss: 1.812  loss_cls: 1.344  loss_box_reg: 0.034
[08:57:48 d2.utils.events]:  eta: 0:17:53  iter: 79  total_loss: 1.231  loss_cls: 0.869  loss_box_reg: 0.036
[08:57:55 d2.utils.events]:  eta: 0:17:50  iter: 99  total_loss: 0.912  loss_cls: 0.489  loss_box_reg: 0.038
[08:58:03 d2.utils.events]:  eta: 0:17:47  iter: 119  total_loss: 0.614  loss_cls: 0.247  loss_box_reg: 0.034
[08:58:11 d2.utils.events]:  eta: 0:17:48  iter: 139  total_loss: 0.466  loss_cls: 0.177  loss_box_reg: 0.068
[08:58:18 d2.utils.events]:  eta: 0:17:46  iter: 159  total_loss: 0.507  loss_cls: 0.176  loss_box_reg: 0.077
[08:58:27 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




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

```
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

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)
```

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 |
```

Small:
area<322
Medium:
322<area<962
Large:
area>962

maxDets:
Thresholds on max detections per image

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.