Report 5

LabelBox Instance Segmentation

Vibhav Joshi

2th June, 2020

Two google colab file with different Hyperparameters

1)https://colab.research.google.com/drive/1dSO78HUsUmdTQpp38v_F3Pd7mXGHRJZo?usp=sharing

2)https://colab.research.google.com/drive/1PqDRq562N0R5iV8IKKjo6_7RZ4d4o_Ir?usp=sharing

Standard Dictionaries needed for training

Standard Dataset Dicts

For standard tasks (instance detection, instance/semantic/panoptic segmentation, keypoint detection), we load the original dataset into `list[dict]` with a specification similar to COCO's json annotations. This is our standard representation for a dataset.

Each dict contains information about one image. The dict may have the following fields, and the required fields vary based on what the dataloader or the task needs (see more below).

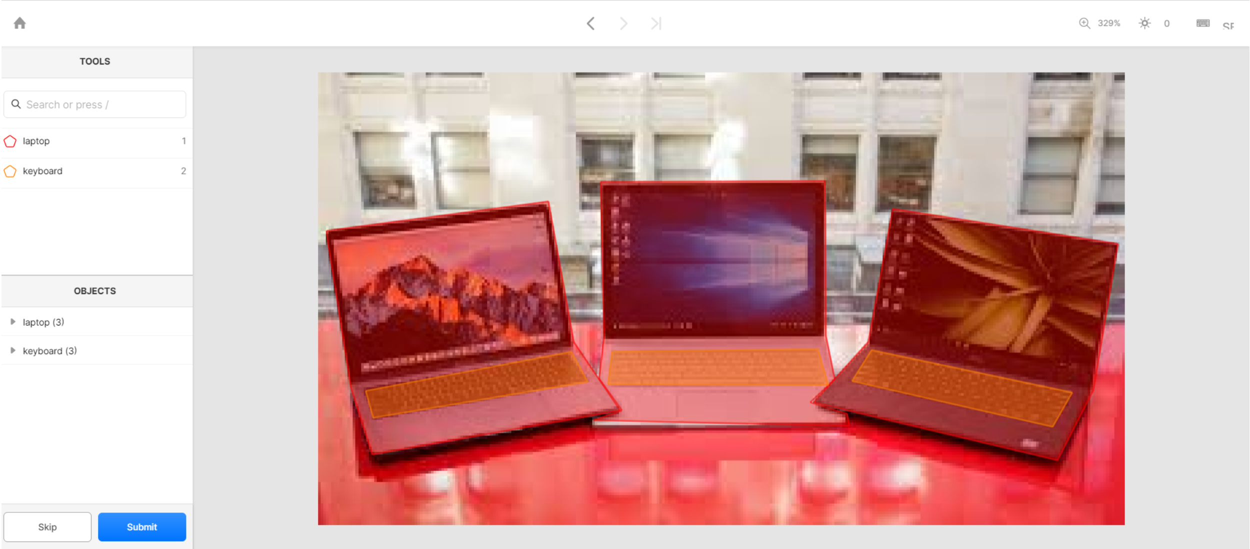
`file_name`: the full path to the image file. Will apply rotation and flipping if the image has such exif information.
`height`, `width`: integer. The shape of image.
`image_id` (str or int): a unique id that identifies this image. Used during evaluation to identify the images, but a dataset may use it for different purposes.
`annotations` (list[dict]): each dict corresponds to annotations of one instance in this image. Required by instance detection/segmentation or keypoint detection tasks.
Images with empty `annotations` will by default be removed from training, but can be included using `DATALOADER.FILTER_EMPTY_ANNOTATIONS`.
Each dict contains the following keys, of which `bbox`,`bbox_mode` and `category_id` are required:
`bbox` (list[float]): list of 4 numbers representing the bounding box of the instance.
`bbox_mode` (int): the format of bbox. It must be a member of structures.BoxMode. Currently supports: `BoxMode.XYXY_ABS`, `BoxMode.XYWH_ABS`.
`category_id` (int): an integer in the range [0, num_categories-1] representing the category label. The value num_categories is reserved to represent the "background" category, if applicable.
`segmentation` (list[list[float]] or dict): the segmentation mask of the instance.
If `list[list[float]]`, it represents a list of polygons, one for each connected component of the object. Each `list[float]` is one simple polygon in the format

of `[x1, y1, ..., xn, yn]`. The Xs and Ys are either relative coordinates in [0, 1], or absolute coordinates, depend on whether “bbox_mode” is relative.
If `dict`, it represents the per-pixel segmentation mask in COCO’s RLE format. The dict should have keys “size” and “counts”. You can convert a uint8 segmentation mask of 0s and 1s into RLE format by `pycocotools.mask.encode(np.asarray(mask, order="F"))`.
`keypoints` (list[float]): in the format of [x1, y1, v1,…, xn, yn, vn]. v[i] means the visibility of this keypoint. `n` must be equal to the number of keypoint categories. The Xs and Ys are either relative coordinates in [0, 1], or absolute coordinates, depend on whether “bbox_mode” is relative.
Note that the coordinate annotations in COCO format are integers in range [0, H-1 or W-1]. By default, detectron2 adds 0.5 to absolute keypoint coordinates to convert them from discrete pixel indices to floating point coordinates.
`iscrowd`: 0 (default) or 1. Whether this instance is labeled as COCO’s “crowd region”. Don’t include this field if you don’t know what it means.
`sem_seg_file_name`: the full path to the ground truth semantic segmentation file. Required by semantic segmentation task. It should be an image whose pixel values are integer labels.

- We need This annotations to be built For Training The model
- Our Aim is to build a custom function that Transforms LabelBox json data to these dictionaries that can be used by model for training

LabelBox

- Labelled images in LabelBox with mask and two labels laptop and keyboard



The Format of LabelBox Data is of one image :

{"ID":"ckaqe2ji65g5r0853fjiqbwqw","DataRow ID":"ckaqdswf792iv0blthckr80it","Labeled Data":"https://storage.labelbox.com/ckaqd3h5q6t9107003u5s1sva%2F6a53c5ac-d94e-f588-81b4-45206ecf7073-24.jpg?Expires=1591871848555&KeyName=labelbox-assets-key-1&Signature=ePSK842udWwizCmhTftVdl9Yuxc","Label":{"objects":[{"featureId":"ckaqe24yv09xf0z95wn8h6fr2","schemaId":"ckaqdxkgl07y80z5u80ntzj5a","title":"laptop","value":"laptop","color":"#FF0000","polygon":[{"x":14.764,"y":15.693},{"x":16.632,"y":130.029},{"x":39.051,"y":198.032},{"x":228.489,"y":179.35},{"x":186.641,"y":118.819},{"x":190.004,"y":5.605},{"x":20.368,"y":11.209}],"instanceURI":"https://api.labelbox.com/masks/feature/ckaqe24yv09xf0z95wn8h6fr2?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJja2FxZDNoNmE3dzFrMDczODBqbzhwbmY1Iiwib3JnYW5pemF0aW9uSWQiOiJja2FxZDNoNXE2dDkxMDcwMDN1NXMxc3ZhIiwiaWF0IjoxNTkwNjYyMjQ4LCJleHAiOjE1OTMyNTQyNDh9.ejU0R7GSbdbkcEk9sXcXk1WNGJNDFuHE7WzMN1j9TIo"},{"featureId":"ckaqe2d3g09zw0z6o3zmpssvs","schemaId":"ckaqdxkgh07y70z5uz1ykwahf","title":"keyboard","value":"keyboard","color":"#FF8000","polygon":[{"x":38.303,"y":138.996},{"x":46.15,"y":166.273},{"x":194.861,"y":153.569},{"x":176.926,"y":125.171}],"instanceURI":"https://api.labelbox.com/masks/feature/ckaqe2d3g09zw0z6o3zmpssvs?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJja2FxZDNoNmE3dzFrMDczODBqbzhwbmY1Iiwib3JnYW5pemF0aW9uSWQiOiJja2FxZDNoNXE2dDkxMDcwMDN1NX

Mxc3ZhIiwiaWF0IjoxNTkwNjYyMjQ4LCJleHAiOjE1OTMyNTQyNDh9.ejU0R7GSbdbkcEk9sXcXk1WNGJNDFuHE7WzMN1j9TIo"}],"classifications":[]},"Created By":"vibhav031998@gmail.com","Project Name":"InstanceSegmentation","Created At":"2020-05-28T06:20:59.000Z","Updated At":"2020-05-28T06:21:01.000Z","Seconds to Label":61.087,"External ID":"24.jpg","Agreement":-1,"Benchmark Agreement":-1,"Benchmark ID":null,"Dataset Name":"laptop","Reviews":[],"View Label":"https://editor.labelbox.com?project=ckaqdbiwa6rv20833x6eh96su&label=ckaqe2ji65g5r0853fjiqbwqw"},

- Need to transform this data in Standard Dicts Format

Custom Functions

```
def get_lap_dicts(img_dir):
    classes = ['keyboard', 'laptop']
    json_file = os.path.join(img_dir, "data.json")
    with open(json_file) as f:
        imgs_anns = json.load(f)

    dataset_dicts = []
    for  v in imgs_anns:
        record = {}
        idx=v["ID"]
        if idx=="ckaqdwgom79bc07003t6sj7bi":
          continue

        filename =v["Labeled Data"]

        record["file_name"] = filename
        image = io.imread(filename)
        '''
        image_2 = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        final_frame = cv2.hconcat((image, image_2))
        cv2_imshow(final_frame)
```

```
'''


height, width = image.shape[:2]

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

annos = v["Label"]
annos=annos["objects"]



objs = []
for anno in annos:
    px = [a["x"] for a in anno['polygon']]
    py = [a["y"] for a in anno['polygon']]
    poly = [(x, y) 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": classes.index(anno['value']),
                "iscrowd": 0
            }
            objs.append(obj)
        record["annotations"] = objs

        dataset_dicts.append(record)
    return dataset_dicts
```

- This was the custom function that was written by me to transform data from LabelBox data to Standard Dicts Format
- The annotations from Labelbox data was put in record dictionary with classes=[laptop,keyboard]
- The function returns dictionaries in which every element is a dictionary of annotation for each image

Tuning Hyperparameters

- Trained Hyperparameters to see variation in accuracy

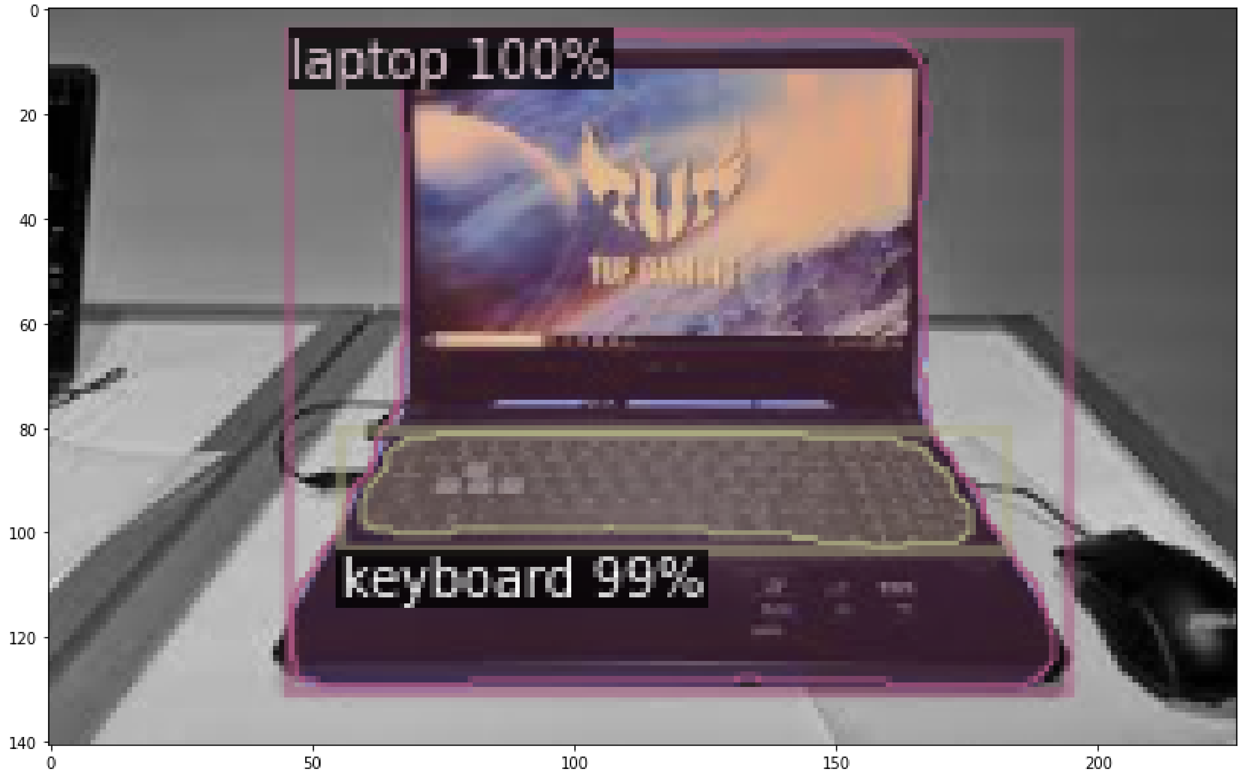
In first colab File parameters were

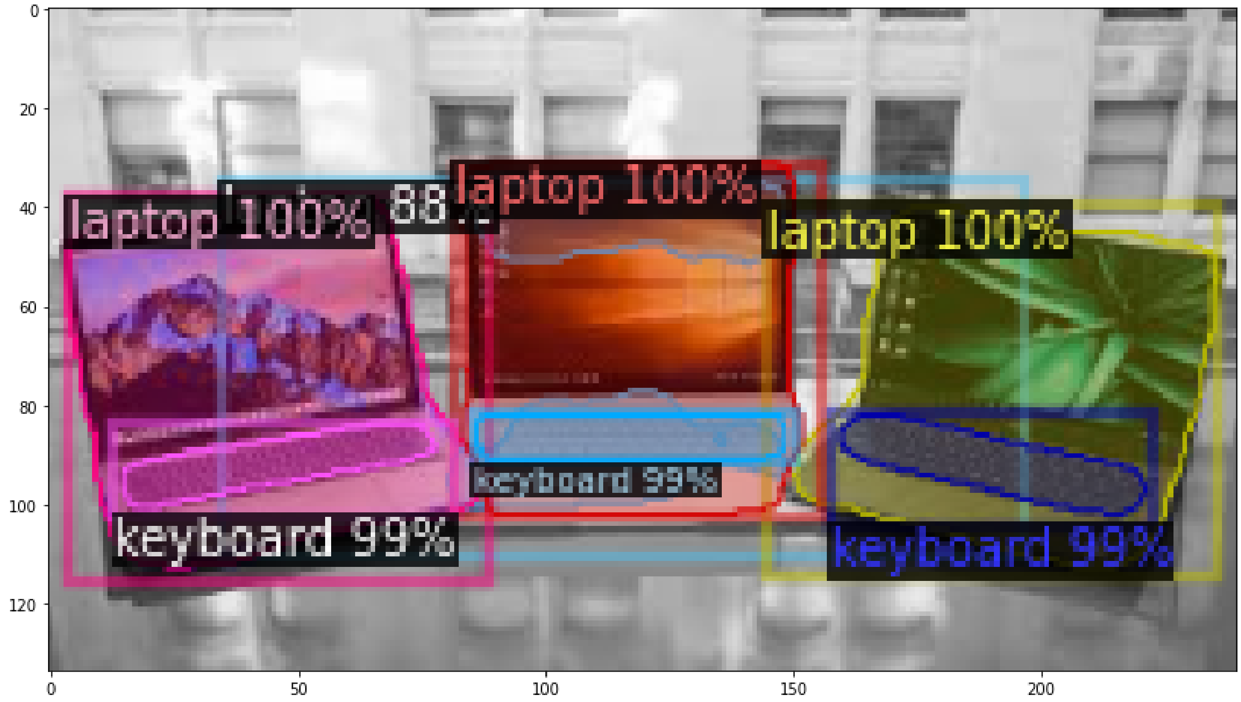
```
cfg.SOLVER.IMS_PER_BATCH = 2

cfg.SOLVER.BASE_LR = 0.00025

cfg.SOLVER.MAX_ITER = 1000
```

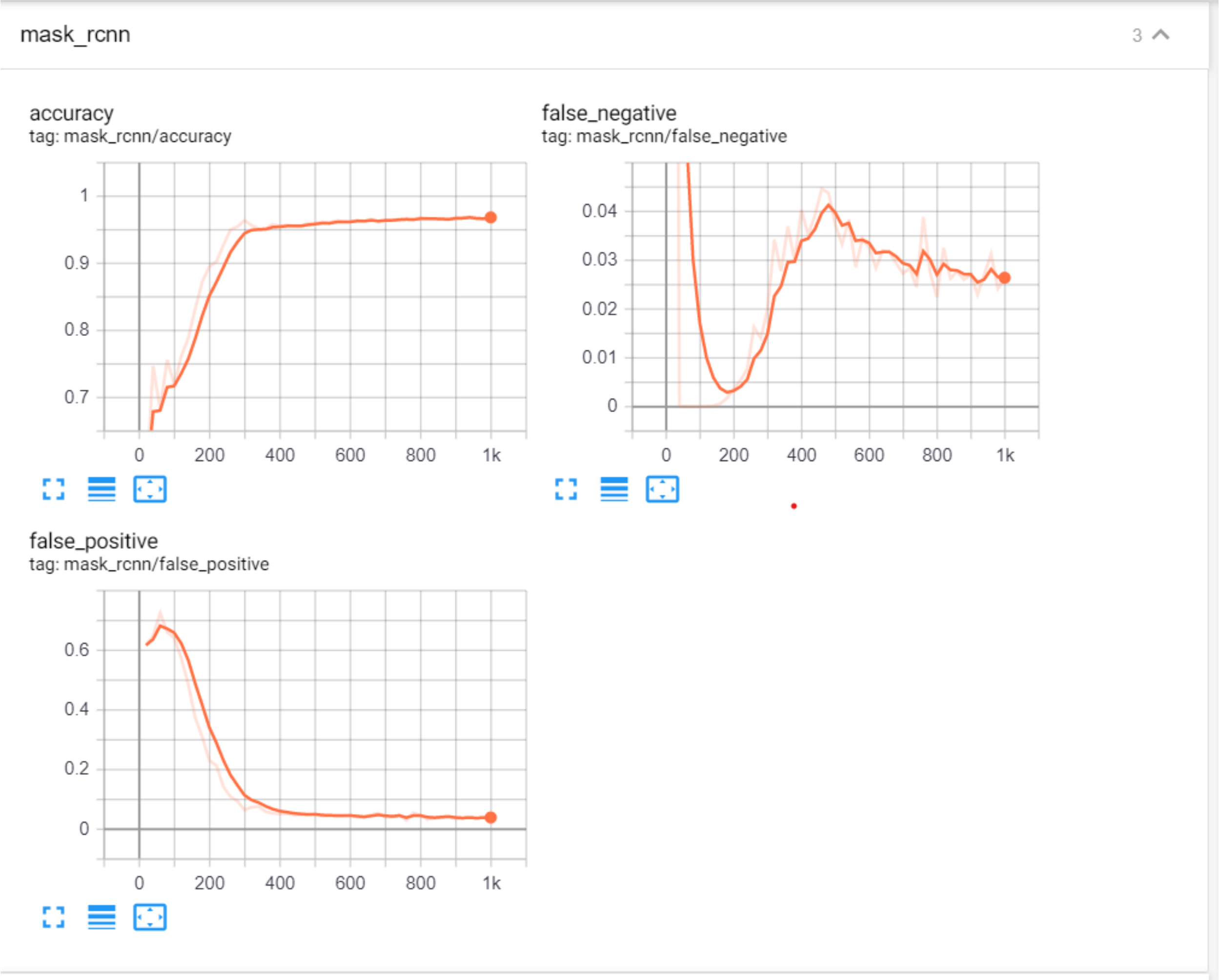
In Second colab file

```
cfg.SOLVER.IMS_PER_BATCH = 2

cfg.SOLVER.BASE_LR = 0.0002

cfg.SOLVER.MAX_ITER = 800
```

The best accuracy was in 1 colab file The results Were As Follows:


laptop 100%
keyboard 99%
0
20
40
60
80
100
120
140
0
50
100
150
200


laptop 100%
88.
laptop 100%
laptop 100%
keyboard 99%
keyboard 99%
keyboard 99%
0
20
40
60
80
100
120
0
50
100
150
200


mask_rcnn
3
accuracy
tag: mask_rcnn/accuracy
1
0.9
0.8
0.7
0
200
400
600
800
1k
false_negative
tag: mask_rcnn/false_negative
0.04
0.03
0.02
0.01
0
0
200
400
600
800
1k
false_positive
tag: mask_rcnn/false_positive
0.6
0.4
0.2
0
0
200
400
600
800
1k