

朱翔
2021211276
IIIS, Tsinghua University

张耕纶
2021270018
DCST, Tsinghua University

王彦仁
2021270027
IIIS, Tsinghua University

Framework

The following is our workflow for inference:

1. Create dataset for inference and evaluation. (See more in eval.py)

```
dataset = create_fasterrcnn_dataset(dataset_path, batch_size=config.test_batch_size,
                                      device_num=device_num, rank_id=rank,
                                      num_parallel_workers=config.num_parallel_workers,
                                      python_multiprocessing=config.python_multiprocessing,
                                      is_training=False)
dataset_coco = COCO(ann_file)
```

2. Load model. (See more in eval.py)

```
net = Faster_Rcnn_Resnet50(config=config)

load_path = ckpt_path
param_dict = load_checkpoint(load_path)
load_param_into_net(net, param_dict)

net = net.set_train(False)
```

3. For each batch, generate the bbox for each single image. (See more in eval.py)

```
for data in dataset.create_dict_iterator(num_epochs=1):
    eval_iter += 1
    print("{}-iteration.".format(eval_iter))

    img_data, img_shape = data['image'], data['image_shape']
    gt_bboxes, gt_labels, gt_num = data['box'], data['label'], data['valid_num']

    res = net(img_data, img_shape, gt_bboxes, gt_labels, gt_num)

    res_bboxes, res_labels, res_masks = res[0], res[1], res[2]

    for i in range(config.test_batch_size):
        res_mask = np.squeeze(res_masks.asnumpy()[i, :, :])

        res_bbox = np.squeeze(res_bboxes.asnumpy()[i, :, :])[res_mask, :]
        res_label = np.squeeze(res_labels.asnumpy()[i, :, :])[res_mask]

        if res_bbox.shape[0] > max_num:
            idx = np.argsort(-res_bbox[:, -1])[:max_num]
            res_bbox, res_label = res_bbox[idx], res_label[idx]

    output_1image = bbox2result_1image(res_bbox, res_label, config.num_classes)
    outputs.append(output_1image)
```

4. Store the result as json format and evaluate the performance. (See more in eval.py)

```
result_files = results2json(dataset_coco, outputs, "./output")
coco_eval(result_files, ["bbox"], dataset_coco, single_result=True)
```

5. Label the bbox in the original images. (See more in visualization.py)

Evaluation

Average Precision	(AP) @[IoU=0.50:0.95	area= all maxDets=100] = 0.431
Average Precision	(AP) @[IoU=0.50	area= all maxDets=100] = 0.636
Average Precision	(AP) @[IoU=0.75	area= all maxDets=100] = 0.488
Average Precision	(AP) @[IoU=0.50:0.95	area= small maxDets=100] = 0.281
Average Precision	(AP) @[IoU=0.50:0.95	area=medium maxDets=100] = 0.496
Average Precision	(AP) @[IoU=0.50:0.95	area= large maxDets=100] = 0.551
Average Recall	(AR) @[IoU=0.50:0.95	area= all maxDets= 1] = 0.379
Average Recall	(AR) @[IoU=0.50:0.95	area= all maxDets= 10] = 0.549
Average Recall	(AR) @[IoU=0.50:0.95	area= all maxDets=100] = 0.575
Average Recall	(AR) @[IoU=0.50:0.95	area= small maxDets=100] = 0.392
Average Recall	(AR) @[IoU=0.50:0.95	area=medium maxDets=100] = 0.608
Average Recall	(AR) @[IoU=0.50:0.95	area= large maxDets=100] = 0.685

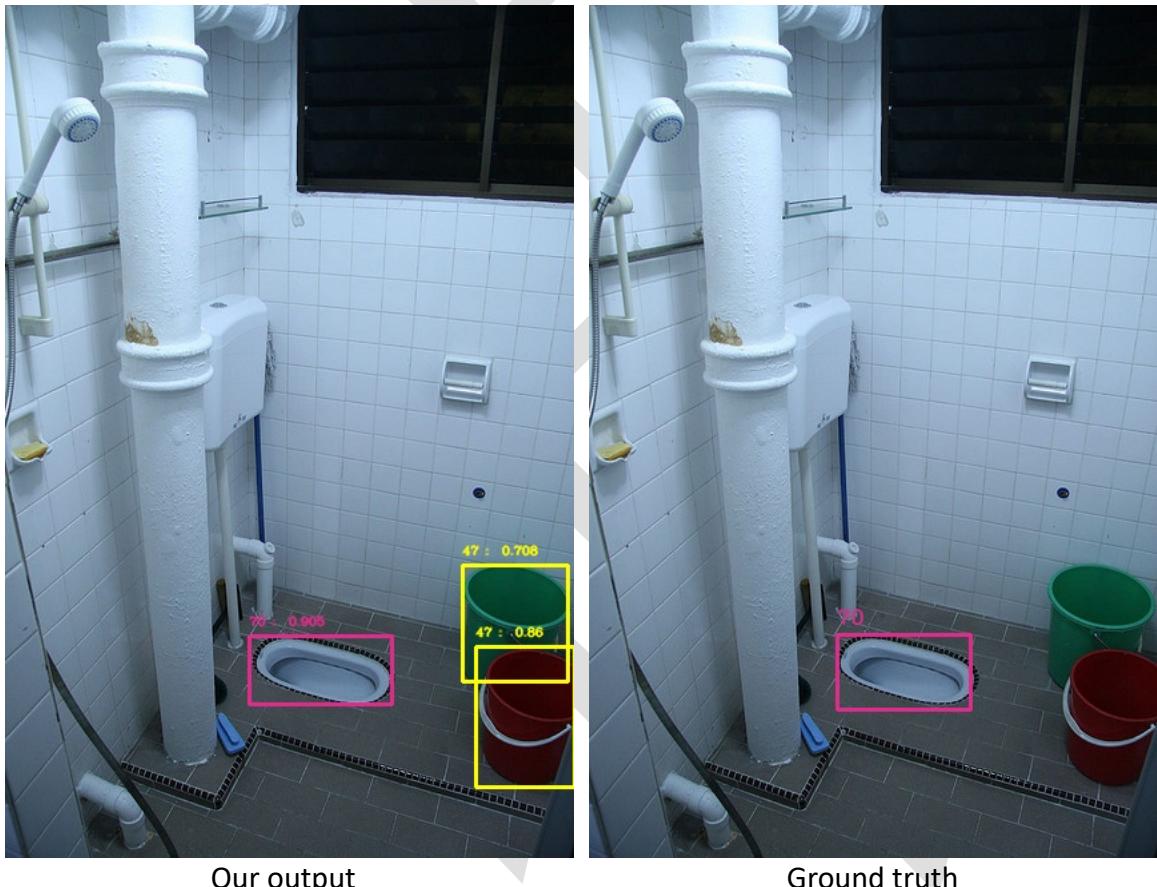
Detection Results

The following is our detection results and the ground-truth annotations, and we only pick 6 images. To see more results, you can go to ./result folder.

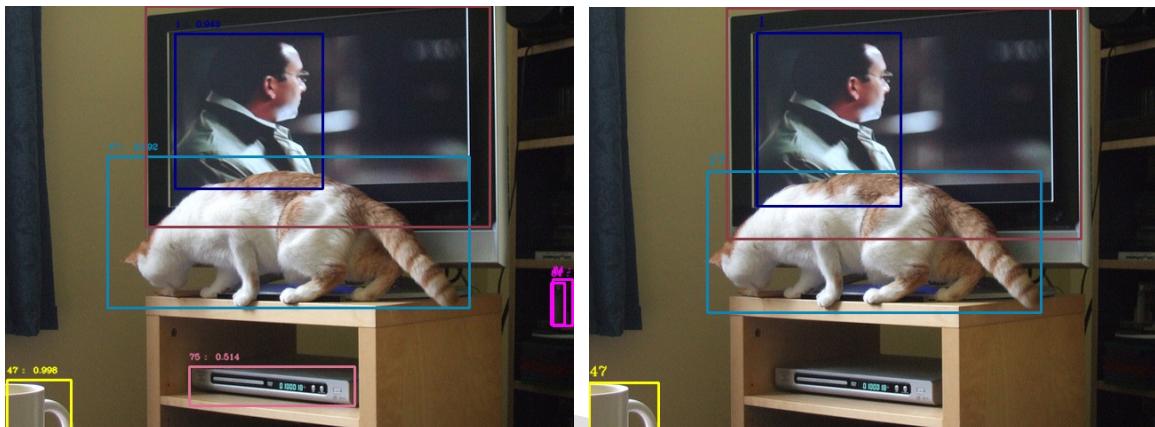
In our output, we only label the objects which has at least more than 0.5 probability, and we find our result is very close to the ground-truth.

However, for some objects, our result may have multiple overlapping rectangles. Although there contains ground-truth, overall, we would consider this imprecise. For this problem, maybe we can do a post-processing. To be more specific, we leave only the one with the largest area in the rectangles with high overlap.

000000006818



000000025560



Our output

Ground truth

000000041888



Our output

Ground truth

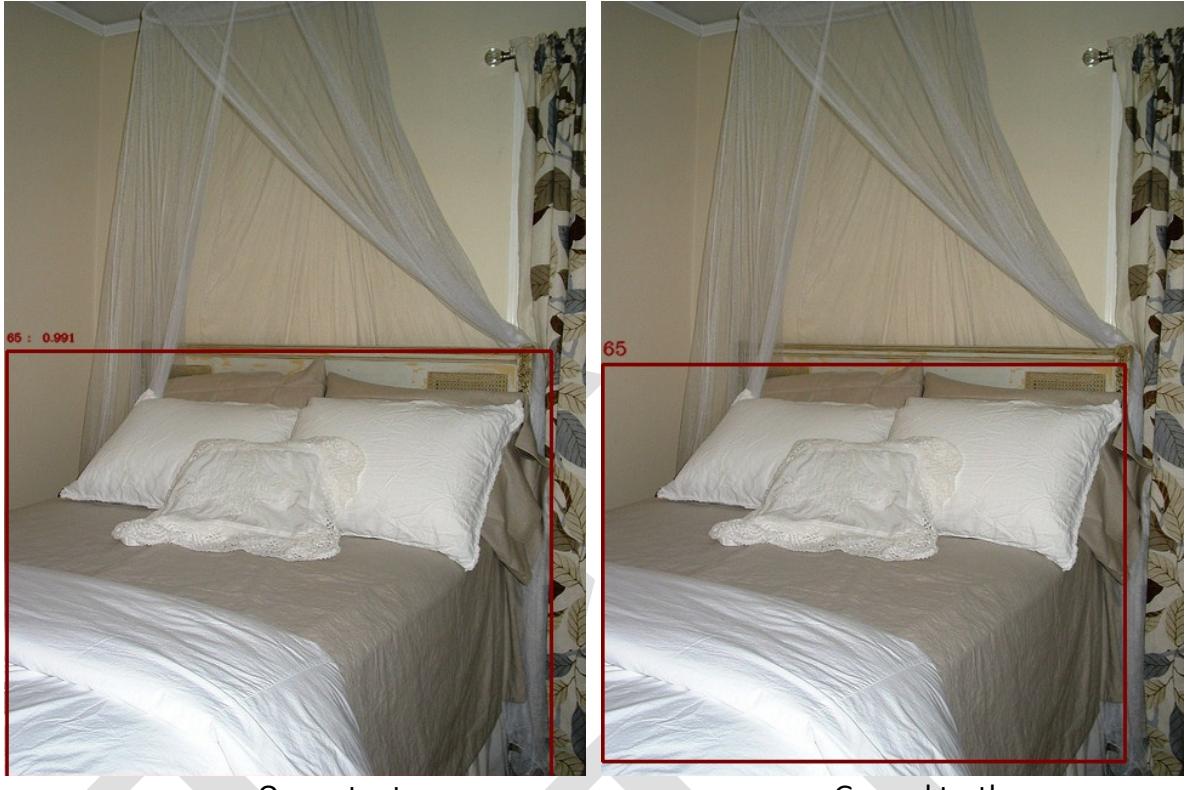
000000063154



Our output

Ground truth

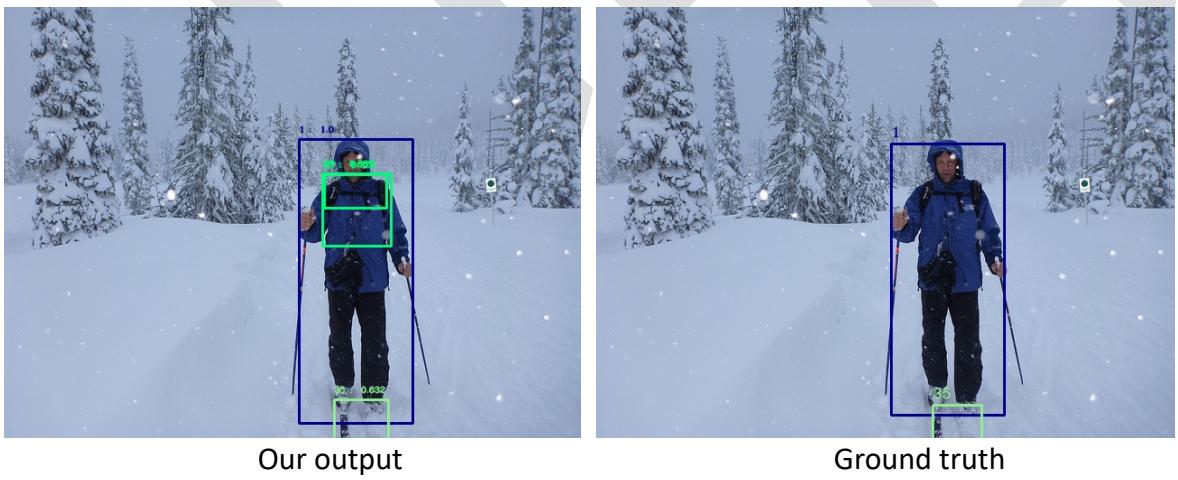
000000066523



Our output

Ground truth

000000266409



Our output

Ground truth