Homework Assignment - 3

Name: Sagar Patel

NETID: sp5894

Question 1

Exercises in convolution. Suppose that the input is a 1D array (or signal; call it x). Find (i.e., design by hand) a convolutional filter that produces the following output (call it y). All you need to do is to specify the filter weights.

1.a

(Approximate) derivatives:

$y[n] = x[n+1] - x[n-1]$

Solution:

The filter weights W are $[-1, 0, 1]$

$y[n] = \sum_{k=-1}^{1} x[n+k] * W[k]$

1.b

(Approximate) second derivatives:

$y[n] = x[n+2] - 2x[n] + x[n-2]$

Solution:

The filter weights W are $[1, 0, -2, 0, 1]$

$y[n] = \sum_{k=0}^{4} x[n+(k-2)] * W[k]$

1.c

(Approximate) integrals:

$y[n] = x[n-\Delta] + x[n-\Delta+1] + ... + x[n+\Delta-1] + x[n+\Delta]$

Solution:

The filter weights W are $[1]$

$y[n] = \sum_{k=-\Delta}^{\Delta} x[n+k] * W[0]$

1.d

Cross-correlations:

$y[n] = \sum_i x[i]x[i+n]$

Solution:

The filter weights W are $W = [x[k]]$

$y[n] = \sum_{k=-\infty}^{\infty} x[i+k] * W[x[k]]$

Question 2

1 × 1 convolution. In the definition of a convolutional layer, if we have I input channels, J output channels, and the filter size is chosen to be $\Delta = 0$, show that the operation is equivalent to applying a regular dense layer in the channel domain. What is the number of trainable parameters in this layer?

Solution:

We know that the total number of trainable parameters in a convolution network with I input layers and J output layers are given by $(2\Delta + 1)^2.I.J$

If $\Delta = 0$, then the total number of trainable parameters would be equal to $I.J$ which is just the number of input channels on to the number of output channels. Now, that is exactly what a dense layer does.

Therefore, a 1×1 convolution is the same as that of a dense layer.

Question 3

The IoU metric. Recall the definition of the IoU metric (or the Jaccard similarity index) for comparing bounding boxes.

3.a

Using elementary properties of sets, prove that the IoU metric between any two pair of bounding boxes is always a non-negative real number in [0, 1].

Solution:

Given two sets 'A' and 'B', we know that - IoU $= \frac{\text{A} \cap \text{B}}{\text{A} \cup \text{B}}$

The Jaccard similarity index is a statistic which is said to be the ratio of the size of the Intersection of sets over their Union (in this case, we have two sets A and B).

Should the elements in sets be the same, the size of the intersection will then be equal to the size of the union. In that case, the Jaccard similarity index will be 1 (unit).

However, if none of the elements in the sets are same (or intersecting), the size of the intersection will be zero! The Jaccard similarity index will be 0. ($A \cap B = 0$)

From this, we can conclude that the IoU metric between any (two) pair of Bounding Boxes will always be a non-negative real number which ranges from $[0, 1]$

3.b

If we represent each bounding box as a function of the top-left and bottom-right coordinates (assume all coordinates are real numbers) then argue that the IoU metric is *non-differentiable* and hence cannot be directly optimized by gradient descent.

Solution:

If no two objects (bounded by the box) that are identified using the top-left, bottom-right co-ordinate approach overlap(/intersect), then the IoU value is said to be zero (0).

Now, if the IoU value is used as a loss function, the gradient will be zero and that will not be differentiable. Therefore, it can not be directly optimized by Gradient Descent.

Question 4

In this programming exercise, we will explore the performance of three different object detection networks. We will be using Detectron2, Facebook AI's object detector library; here is the repository. It will be helpful to go through the excellent tutorial here.

Solution:

Link to the Notebook

Homework Assignment - 3

Name: Sagar Patel

NETID: sp5894

References

1. Tutorial: https://gilberttanner.com/blog/detectron-2-object-detection-with-pytorch
2. Reference Notebook: https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5#scrollTo=b-i4hmGYk1dL

Question 4

In this programming exercise, we will explore the performance of three different object detection networks. We will be using Detectron2, Facebook AI's object detector library; here is the repository. It will be helpful to go through the excellent tutorial here.

Installing dependancies and libraries

We shall be installing the dependencies that we need (torch version, pyyaml, etc.). OpenCV will not need to be installed separately as it comes pre-installed on Google Colab.

In [1]:

```
!pip install pyyaml==5.1
!pip install torch==1.7.1 torchvision==0.8.2
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
```

```
Requirement already satisfied: pyyaml==5.1 in /usr/local/lib/python3.7/dist-pack
ages (5.1)
Requirement already satisfied: torch==1.7.1 in /usr/local/lib/python3.7/dist-pac
kages (1.7.1)
Requirement already satisfied: torchvision==0.8.2 in /usr/local/lib/python3.7/di
st-packages (0.8.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from torch==1.7.1) (1.19.5)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dis
t-packages (from torch==1.7.1) (3.7.4.3)
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.7/dist-pa
ckages (from torchvision==0.8.2) (8.1.2)
1.7.1 True
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Now, we install `Detectron2` and the necessary torch version.

Refer https://detectron2.readthedocs.io/tutorials/install.html for instructions

NOTE:

- Google Colab has CUDA 10.1 and torch 1.7 versions.
- After installation, we might need to restart the runtime in Google Colab. Or, we could add `exit(0)` on the last line.

In [2]:

```
import torch
assert torch.__version__.startswith("1.7")
!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu10

# exit(0)
```

```
Looking in links: https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.
7/index.html
Requirement already satisfied: detectron2 in /usr/local/lib/python3.7/dist-packa
ges (0.3+cu101)
Requirement already satisfied: termcolor>=1.1 in /usr/local/lib/python3.7/dist-p
ackages (from detectron2) (1.1.0)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.7/dist-pack
ages (from detectron2) (2.4.1)
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-package
s (from detectron2) (0.8.9)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages
(from detectron2) (0.16.0)
Requirement already satisfied: tqdm>4.29.0 in /usr/local/lib/python3.7/dist-pack
ages (from detectron2) (4.41.1)
Requirement already satisfied: fvcore>=0.1.2 in /usr/local/lib/python3.7/dist-pa
ckages (from detectron2) (0.1.3.post20210311)
Requirement already satisfied: Pillow>=7.1 in /usr/local/lib/python3.7/dist-pack
ages (from detectron2) (8.1.2)
Requirement already satisfied: pycocotools>=2.0.2 in /usr/local/lib/python3.7/di
st-packages (from detectron2) (2.0.2)
Requirement already satisfied: pydot in /usr/local/lib/python3.7/dist-packages
(from detectron2) (1.3.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-pack
ages (from detectron2) (1.3.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packa
ges (from detectron2) (3.2.2)
Requirement already satisfied: yacs>=0.1.6 in /usr/local/lib/python3.7/dist-pack
ages (from detectron2) (0.1.8)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/li
b/python3.7/dist-packages (from tensorboard->detectron2) (0.4.3)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/di
st-packages (from tensorboard->detectron2) (54.0.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dis
t-packages (from tensorboard->detectron2) (1.0.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/d
ist-packages (from tensorboard->detectron2) (2.23.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/p
ython3.7/dist-packages (from tensorboard->detectron2) (1.8.0)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-p
ackages (from tensorboard->detectron2) (1.32.0)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.7/dist-pa
ckages (from tensorboard->detectron2) (1.19.5)
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.7/dist-
packages (from tensorboard->detectron2) (3.12.4)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in /usr/local/
lib/python3.7/dist-packages (from tensorboard->detectron2) (0.36.2)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-
```

```
packages (from tensorboard->detectron2) (3.3.4)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-pack
ages (from tensorboard->detectron2) (1.15.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.
7/dist-packages (from tensorboard->detectron2) (1.27.1)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-pac
kages (from tensorboard->detectron2) (0.10.0)
Requirement already satisfied: iopath>=0.1.2 in /usr/local/lib/python3.7/dist-pa
ckages (from fvcore>=0.1.2->detectron2) (0.1.6)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-pack
ages (from fvcore>=0.1.2->detectron2) (5.1)
Requirement already satisfied: cython>=0.27.3 in /usr/local/lib/python3.7/dist-p
ackages (from pycocotools>=2.0.2->detectron2) (0.29.22)
Requirement already satisfied: pyparsing>=2.1.4 in /usr/local/lib/python3.7/dist
-packages (from pydot->detectron2) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/
dist-packages (from matplotlib->detectron2) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dis
t-packages (from matplotlib->detectron2) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-pac
kages (from matplotlib->detectron2) (0.10.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python
3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard->detectron
2) (1.3.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pac
kages (from requests<3,>=2.21.0->tensorboard->detectron2) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/l
ocal/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard->detectr
on2) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/di
st-packages (from requests<3,>=2.21.0->tensorboard->detectron2) (2020.12.5)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
t-packages (from requests<3,>=2.21.0->tensorboard->detectron2) (3.0.4)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /us
r/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensorboard->detectro
n2) (3.7.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.
7/dist-packages (from google-auth<2,>=1.6.3->tensorboard->detectron2) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" in /usr/lo
cal/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard->detect
ron2) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.
7/dist-packages (from google-auth<2,>=1.6.3->tensorboard->detectron2) (4.2.1)
Requirement already satisfied: portalocker in /usr/local/lib/python3.7/dist-pack
ages (from iopath>=0.1.2->fvcore>=0.1.2->detectron2) (2.2.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-
packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tenso
rboard->detectron2) (3.1.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packag
es (from importlib-metadata; python_version < "3.8"->markdown>=2.6.8->tensorboar
d->detectron2) (3.4.1)
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8"
in /usr/local/lib/python3.7/dist-packages (from importlib-metadata; python_versi
on < "3.8"->markdown>=2.6.8->tensorboard->detectron2) (3.7.4.3)
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-auth<2,>=1.6.3->tensorboard->d
etectron2) (0.4.8)
```

Moving on, we have some basic set up for `detectron2` logger, importing some other libraries and utlities for `detectron2`

In [3]:

```python
# Setting up the detectron2 logger
import detectron2
from detectron2.utils.logger import setup_logger
```

```
setup_logger()

# Importing other libraries
import numpy as np
import os, json, cv2, random
from google.colab.patches import cv2_imshow

# Importing 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
```

```
** fvcore version of PathManager will be deprecated soon. **
** Please migrate to the version in iopath repo. **
https://github.com/facebookresearch/iopath

** fvcore version of PathManager will be deprecated soon. **
** Please migrate to the version in iopath repo. **
https://github.com/facebookresearch/iopath
```

Problem 4.a

Download the following test image (a picture of pedestrians in Central Park). We will run two different detectors on this image.

NOTE: For convenience, I have changed the name of the image to `input.jpeg`

In [4]:

```
im = cv2.imread("input.jpeg")
cv2_imshow(im)
```

Problem 4.b

First, consider the COCO Keypoint Person Detector model with a ResNet50-FPN base network, which is trained to detect human silhouettes. This can be found in the Detectron2 Model Zoo in the "COCO Keypoint" table. Use this model to detect as many silhouttes of people in the test image as you can. You may have to play around with the thresholds to optimize performance.

In [5]:

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Keypoints/keypoint_rcnn_R_50
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.6
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Keypoints/keypoint_rcnn_R
predictor = DefaultPredictor(cfg)
outputs = predictor(im)
```

```
model_final_a6e10b.pkl: 237MB [00:09, 25.5MB/s]
/usr/local/lib/python3.7/dist-packages/detectron2/modeling/roi_heads/fast_rcnn.p
y:124: UserWarning: This overload of nonzero is deprecated:
        nonzero()
Consider using one of the following signatures instead:
        nonzero(*, bool as_tuple) (Triggered internally at  /pytorch/torch/csrc/
utils/python_arg_parser.cpp:882.)
  filter_inds = filter_mask.nonzero()
```

We can visualize using `Visualizer` by running the next cell

In [6]:

```
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```

As we can see, the output image gives us the expected result. All the moving objects (people) in the given image have been marked by a silhouette.

For reference, we have used the COCO Person Keypoint R-CNN model **R50-FPN 3x** model

`cfg.merge_from_file(model_zoo.get_config_file("COCO-Keypoints/keypoint_rcnn_R_50_FPN_3x.yaml"))`

COCO Person Keypoint Detection Baselines with Keypoint R-CNN

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	kp. AP	model id	download
R50-FPN	1x	0.315	0.072	5.0	53.6	64.0	137261548	model | metrics
R50-FPN	3x	0.316	0.066	5.0	55.4	65.5	137849621	model | metrics
R101-FPN	3x	0.390	0.076	6.1	56.4	66.1	138363331	model | metrics
X101-FPN	3x	0.738	0.121	8.7	57.3	66.0	139686956	model | metrics

We set the threshold value to `0.6`

Problem 4.c

Second, repeat the above procedure, but with the Mask R-CNN model with ResNet50- FPN backbone, available in the Model Zoo in the “COCO Instance Segmentation” table. This time, you should be able to detect both people as well as other objects in the scene. Comment on your findings.

In [7]:

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rc
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.6
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask
predictor = DefaultPredictor(cfg)
outputs = predictor(im)
```

```
model_final_4ce675.pkl: 144MB [00:06, 23.5MB/s]
```

We can visualize using `Visualizer` by running the next cell

In [8]:

```
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```

From the output image, we are now be able to detect and distinguish objects visually. However, the catch is that the objects were detected based on the model that was being trained on them.

Unfortunately, the model was not trained JUST for balloons (there's other stuff too). This also explains why the "balloons" were being detected as 'sports balls' or 'apples'.

We use the **R50-C4 3x** model from the COCO Instance Segmentation Mask R-CNN list of models.

`cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_C4_3x.yaml"))`

The threshold value is set to 0.6

COCO Instance Segmentation Baselines with Mask R-CNN

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	mask AP	model id	download
R50-C4	1x	0.584	0.110	5.2	36.8	32.2	137259246	model | metrics
R50-DC5	1x	0.471	0.076	6.5	38.3	34.2	137260150	model | metrics
R50-FPN	1x	0.261	0.043	3.4	38.6	35.2	137260431	model | metrics
R50-C4	3x	0.575	0.111	5.2	39.8	34.4	137849525	model | metrics
R50-DC5	3x	0.470	0.076	6.5	40.0	35.9	137849551	model | metrics
R50-FPN	3x	0.261	0.043	3.4	41.0	37.2	137849600	model | metrics
R101-C4	3x	0.652	0.145	6.3	42.6	36.7	138363239	model | metrics
R101-DC5	3x	0.545	0.092	7.6	41.9	37.3	138363294	model | metrics
R101-FPN	3x	0.340	0.056	4.6	42.9	38.6	138205316	model | metrics
X101-FPN	3x	0.690	0.103	7.2	44.3	39.5	139653917	model | metrics

Problem 4.d

It appears that the balloons in the test image are not being properly detected in either model. This is because the COCO dataset used to train the above models does not contain balloons! Following the tutorial code above, start with the above pre-trained Mask R-CNN model and train a balloon detector using the (fine-tuning) balloon image dataset provided here. Test it on the original test image and show that you are now able to identify all the balloons.

Preparing the dataset

Downloading and extracting the dataset.

In [9]:

```
!wget https://github.com/matterport/Mask_RCNN/releases/download/v2.1/balloon_dat
!unzip balloon_dataset.zip > /dev/null
```

```
--2021-03-12 22:45:48--  https://github.com/matterport/Mask_RCNN/releases/downlo
ad/v2.1/balloon_dataset.zip
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-releases.githubusercontent.com/107595270/737339e2-2b83-
11e8-856a-188034eb3468?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWN
JYAX4CSVEH53A%2F20210312%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20210312T224
548Z&X-Amz-Expires=300&X-Amz-Signature=dedaec07230501544f91ebee16ebd00c966980ad9
07e5d2824c88d0a6446e991&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=107
595270&response-content-disposition=attachment%3B%20filename%3Dballoon_dataset.z
ip&response-content-type=application%2Foctet-stream [following]
--2021-03-12 22:45:48--  https://github-releases.githubusercontent.com/10759527
0/737339e2-2b83-11e8-856a-188034eb3468?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Cr
edential=AKIAIWNJYAX4CSVEH53A%2F20210312%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-D
ate=20210312T224548Z&X-Amz-Expires=300&X-Amz-Signature=dedaec07230501544f91ebee1
6ebd00c966980ad907e5d2824c88d0a6446e991&X-Amz-SignedHeaders=host&actor_id=0&key_
id=0&repo_id=107595270&response-content-disposition=attachment%3B%20filename%3Db
alloon_dataset.zip&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-releases.githubusercont
```

```
nt.com)... 185.199.108.154, 185.199.109.154, 185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-releases.githubuserc
ontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38741381 (37M) [application/octet-stream]
Saving to: ‘balloon_dataset.zip’

balloon_dataset.zip 100%[===================>]  36.95M   105MB/s    in 0.4s

2021-03-12 22:45:49 (105 MB/s) - ‘balloon_dataset.zip’ saved [38741381/38741381]
```

Here, the dataset is in its custom format, therefore we write a function to parse it and prepare it into detectron2's standard format.

In [10]:

```
import os
import numpy as np
import json
from detectron2.structures import BoxMode
import itertools

# Defining the function
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 _, v in imgs_anns.items():
    record = {}
    filename = os.path.join(img_dir, v["filename"])
    height, width = cv2.imread(filename).shape[:2]

    record["file_name"] = filename
    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 = list(itertools.chain.from_iterable(poly))
      obj = {"bbox": [np.min(px), np.min(py), np.max(px), np.max(py)],
             "bbox_mode": BoxMode.XYXY_ABS,
             "segmentation": [poly],
             "category_id": 0,
             "iscrowd": 0}
      objs.append(obj)
      record["annotations"] = objs
      dataset_dicts.append(record)
  return dataset_dicts

from detectron2.data import DatasetCatalog, MetadataCatalog
for d in ["train", "val"]:
```

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

In [11]:

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

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rc
cfg.DATASETS.TRAIN = ("balloon/train",)
cfg.DATASETS.TEST = () # no metrics implemented for this dataset
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 300 # 300 iterations seems good enough, but you can certai
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128 # faster, and good enough for thi
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1 # only has one class (ballon)

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

```
[03/12 22:45:57 d2.engine.defaults]: Model:
GeneralizedRCNN(
  (backbone): 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)
        )
        (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)
        )
      )
    )
    (res3): Sequential(
      (0): BottleneckBlock(
        (shortcut): Conv2d(
          256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
        (conv1): Conv2d(
          256, 128, kernel_size=(1, 1), stride=(2, 2), bias=False
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv2): Conv2d(
          128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv3): Conv2d(
          128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
      )
      (1): BottleneckBlock(
        (conv1): Conv2d(
          512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv2): Conv2d(
          128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv3): Conv2d(
          128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
      )
      (2): BottleneckBlock(
        (conv1): Conv2d(
          512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv2): Conv2d(
          128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv3): Conv2d(
```

```
          128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
      )
      (3): BottleneckBlock(
        (conv1): Conv2d(
          512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv2): Conv2d(
          128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
        )
        (conv3): Conv2d(
          128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
      )
    )
    (res4): Sequential(
      (0): BottleneckBlock(
        (shortcut): Conv2d(
          512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False
          (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
        )
        (conv1): Conv2d(
          512, 256, kernel_size=(1, 1), stride=(2, 2), bias=False
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv2): Conv2d(
          256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv3): Conv2d(
          256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
        )
      )
      (1): BottleneckBlock(
        (conv1): Conv2d(
          1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv2): Conv2d(
          256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv3): Conv2d(
          256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
        )
      )
      (2): BottleneckBlock(
        (conv1): Conv2d(
          1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv2): Conv2d(
          256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
```

```
        (conv3): Conv2d(
          256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
        )
      )
      (3): BottleneckBlock(
        (conv1): Conv2d(
          1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv2): Conv2d(
          256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv3): Conv2d(
          256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
        )
      )
      (4): BottleneckBlock(
        (conv1): Conv2d(
          1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv2): Conv2d(
          256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv3): Conv2d(
          256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
        )
      )
      (5): BottleneckBlock(
        (conv1): Conv2d(
          1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv2): Conv2d(
          256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
        )
        (conv3): Conv2d(
          256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
        )
      )
    )
  )
  (proposal_generator): RPN(
    (rpn_head): StandardRPNHead(
      (conv): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1,
 1))
      (objectness_logits): Conv2d(1024, 15, kernel_size=(1, 1), stride=(1, 1))
      (anchor_deltas): Conv2d(1024, 60, kernel_size=(1, 1), stride=(1, 1))
    )
    (anchor_generator): DefaultAnchorGenerator(
      (cell_anchors): BufferList()
    )
  )
  (roi_heads): Res5ROIHeads(
    (pooler): ROIPooler(
```

```
      (level_poolers): ModuleList(
        (0): ROIAlign(output_size=(14, 14), spatial_scale=0.0625, sampling_ratio
=0, aligned=True)
      )
    )
    (res5): Sequential(
      (0): BottleneckBlock(
        (shortcut): Conv2d(
          1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False
          (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
        )
        (conv1): Conv2d(
          1024, 512, kernel_size=(1, 1), stride=(2, 2), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
        (conv2): Conv2d(
          512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
        (conv3): Conv2d(
          512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
        )
      )
      (1): BottleneckBlock(
        (conv1): Conv2d(
          2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
        (conv2): Conv2d(
          512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
        (conv3): Conv2d(
          512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
        )
      )
      (2): BottleneckBlock(
        (conv1): Conv2d(
          2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
        (conv2): Conv2d(
          512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
        )
        (conv3): Conv2d(
          512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
          (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
        )
      )
    )
    (box_predictor): FastRCNNOutputLayers(
      (cls_score): Linear(in_features=2048, out_features=2, bias=True)
      (bbox_pred): Linear(in_features=2048, out_features=4, bias=True)
    )
    (mask_head): MaskRCNNConvUpsampleHead(
      (deconv): ConvTranspose2d(2048, 256, kernel_size=(2, 2), stride=(2, 2))
      (deconv_relu): ReLU()
      (predictor): Conv2d(256, 1, kernel_size=(1, 1), stride=(1, 1))
    )
```

```
  )
)
[03/12 22:45:58 d2.data.build]: Removed 0 images with no usable annotations. 255
images left.
[03/12 22:45:58 d2.data.build]: Distribution of instances among all 1 categorie
s:
|  category   | #instances   |
|:----------:|:-------------|
|  balloon    | 2741         |
|             |              |
[03/12 22:45:58 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used in t
raining: [ResizeShortestEdge(short_edge_length=(640, 672, 704, 736, 768, 800), m
ax_size=1333, sample_style='choice'), RandomFlip()]
[03/12 22:45:58 d2.data.build]: Using training sampler TrainingSampler
[03/12 22:45:58 d2.data.common]: Serializing 255 elements to byte tensors and co
ncatenating them all ...
[03/12 22:45:58 d2.data.common]: Serialized dataset takes 1.61 MiB
Skip loading parameter 'roi_heads.box_predictor.cls_score.weight' to the model d
ue to incompatible shapes: (81, 2048) in the checkpoint but (2, 2048) in the mod
el! You might want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.cls_score.bias' to the model due
to incompatible shapes: (81,) in the checkpoint but (2,) in the model! You might
want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.weight' to the model d
ue to incompatible shapes: (320, 2048) in the checkpoint but (4, 2048) in the mo
del! You might want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.bias' to the model due
to incompatible shapes: (320,) in the checkpoint but (4,) in the model! You migh
t want to double check if this is expected.
Skip loading parameter 'roi_heads.mask_head.predictor.weight' to the model due t
o incompatible shapes: (80, 256, 1, 1) in the checkpoint but (1, 256, 1, 1) in t
he model! You might want to double check if this is expected.
Skip loading parameter 'roi_heads.mask_head.predictor.bias' to the model due to
incompatible shapes: (80,) in the checkpoint but (1,) in the model! You might wa
nt to double check if this is expected.
[03/12 22:46:00 d2.engine.train_loop]: Starting training from iteration 0
[03/12 22:46:11 d2.utils.events]:  eta: 0:02:32  iter: 19  total_loss: 2.486  lo
ss_cls: 0.6779  loss_box_reg: 0.8294  loss_mask: 0.6985  loss_rpn_cls: 0.193  lo
ss_rpn_loc: 0.0562  time: 0.5457  data_time: 0.0155  lr: 4.9953e-06  max_mem: 22
92M
[03/12 22:46:22 d2.utils.events]:  eta: 0:02:22  iter: 39  total_loss: 2.428  lo
ss_cls: 0.6537  loss_box_reg: 0.8405  loss_mask: 0.6792  loss_rpn_cls: 0.2565  l
oss_rpn_loc: 0.05173  time: 0.5474  data_time: 0.0066  lr: 9.9902e-06  max_mem:
 2292M
[03/12 22:46:34 d2.utils.events]:  eta: 0:02:12  iter: 59  total_loss: 2.284  lo
ss_cls: 0.5958  loss_box_reg: 0.8178  loss_mask: 0.6341  loss_rpn_cls: 0.2081  l
oss_rpn_loc: 0.05285  time: 0.5496  data_time: 0.0092  lr: 1.4985e-05  max_mem:
 2292M
[03/12 22:46:45 d2.utils.events]:  eta: 0:02:02  iter: 79  total_loss: 2.203  lo
ss_cls: 0.5378  loss_box_reg: 0.8324  loss_mask: 0.5856  loss_rpn_cls: 0.1973  l
oss_rpn_loc: 0.06615  time: 0.5548  data_time: 0.0059  lr: 1.998e-05  max_mem: 2
292M
[03/12 22:46:56 d2.utils.events]:  eta: 0:01:51  iter: 99  total_loss: 2.035  lo
ss_cls: 0.4843  loss_box_reg: 0.8481  loss_mask: 0.5147  loss_rpn_cls: 0.1808  l
oss_rpn_loc: 0.04441  time: 0.5556  data_time: 0.0079  lr: 2.4975e-05  max_mem:
 2292M
[03/12 22:47:07 d2.utils.events]:  eta: 0:01:40  iter: 119  total_loss: 1.965  l
oss_cls: 0.4276  loss_box_reg: 0.8404  loss_mask: 0.448  loss_rpn_cls: 0.2039  l
oss_rpn_loc: 0.04959  time: 0.5562  data_time: 0.0053  lr: 2.997e-05  max_mem: 2
292M
[03/12 22:47:19 d2.utils.events]:  eta: 0:01:29  iter: 139  total_loss: 1.819  l
oss_cls: 0.3822  loss_box_reg: 0.7888  loss_mask: 0.403  loss_rpn_cls: 0.1445  l
oss_rpn_loc: 0.05537  time: 0.5581  data_time: 0.0083  lr: 3.4965e-05  max_mem:
 2292M
[03/12 22:47:30 d2.utils.events]:  eta: 0:01:18  iter: 159  total_loss: 1.661  l
```

```
oss_cls: 0.2933  loss_box_reg: 0.8417  loss_mask: 0.3202  loss_rpn_cls: 0.1498
 loss_rpn_loc: 0.03767  time: 0.5570  data_time: 0.0060  lr: 3.996e-05  max_mem:
2292M
[03/12 22:47:41 d2.utils.events]:  eta: 0:01:06  iter: 179  total_loss: 1.702  l
oss_cls: 0.3057  loss_box_reg: 0.8512  loss_mask: 0.2882  loss_rpn_cls: 0.1662
 loss_rpn_loc: 0.07733  time: 0.5584  data_time: 0.0066  lr: 4.4955e-05  max_me
m: 2292M
[03/12 22:47:53 d2.utils.events]:  eta: 0:00:55  iter: 199  total_loss: 1.476  l
oss_cls: 0.2537  loss_box_reg: 0.7892  loss_mask: 0.2405  loss_rpn_cls: 0.1078
 loss_rpn_loc: 0.05624  time: 0.5595  data_time: 0.0078  lr: 4.995e-05  max_mem:
2292M
[03/12 22:48:04 d2.utils.events]:  eta: 0:00:44  iter: 219  total_loss: 1.342  l
oss_cls: 0.1934  loss_box_reg: 0.7421  loss_mask: 0.2281  loss_rpn_cls: 0.1358
 loss_rpn_loc: 0.03112  time: 0.5605  data_time: 0.0064  lr: 5.4945e-05  max_me
m: 2292M
[03/12 22:48:16 d2.utils.events]:  eta: 0:00:33  iter: 239  total_loss: 1.2  los
s_cls: 0.1662  loss_box_reg: 0.6713  loss_mask: 0.2173  loss_rpn_cls: 0.1196  lo
ss_rpn_loc: 0.03137  time: 0.5621  data_time: 0.0052  lr: 5.994e-05  max_mem: 22
92M
[03/12 22:48:27 d2.utils.events]:  eta: 0:00:22  iter: 259  total_loss: 1.152  l
oss_cls: 0.1444  loss_box_reg: 0.5962  loss_mask: 0.162  loss_rpn_cls: 0.1405  l
oss_rpn_loc: 0.03627  time: 0.5628  data_time: 0.0079  lr: 6.4935e-05  max_mem:
 2322M
[03/12 22:48:38 d2.utils.events]:  eta: 0:00:11  iter: 279  total_loss: 1.175  l
oss_cls: 0.1614  loss_box_reg: 0.6515  loss_mask: 0.1503  loss_rpn_cls: 0.09248
loss_rpn_loc: 0.05811  time: 0.5628  data_time: 0.0088  lr: 6.993e-05  max_mem:
 2322M
[03/12 22:48:51 d2.utils.events]:  eta: 0:00:00  iter: 299  total_loss: 0.9233
 loss_cls: 0.1275  loss_box_reg: 0.5383  loss_mask: 0.1319  loss_rpn_cls: 0.0911
8  loss_rpn_loc: 0.03744  time: 0.5633  data_time: 0.0053  lr: 7.4925e-05  max_m
em: 2322M
[03/12 22:48:51 d2.engine.hooks]: Overall training speed: 298 iterations in 0:0
2:47 (0.5633 s / it)
[03/12 22:48:51 d2.engine.hooks]: Total training time: 0:02:49 (0:00:01 on hook
s)
```

In [12]:

```
from detectron2.utils.visualizer import ColorMode

# Loading the weights
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

# Setting the training dataset path
cfg.DATASETS.TEST = ("balloon/val", )

# Creating predictor (model) for inference
predictor = DefaultPredictor(cfg)
dataset_dicts = get_balloon_dicts("balloon/val")
for d in random.sample(dataset_dicts, 4):
    im = cv2.imread("input.jpeg")
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1],
                   metadata=balloon_metadata,
                   scale=0.8,
                   instance_mode=ColorMode.IMAGE_BW
                  )
    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(v.get_image()[:, :, ::-1])
```

balloon 99%
balloon 99%
balloon 83%
balloon 86%
fine art
america

balloon 99%
balloon 99%
balloon 83%
balloon 86%
fine art
america

For the final segment, we start with the pre-trained COCO Instance segmentation Mask R-CNN model **R-50 C4 3x** and we trained the balloon detector using the image dataset provided to us which we import.

We can finally see the results in the output image(s) and now the model does predict balloons as they are and not 'sports goods' or 'apples'

In [ ]:

```
```