Neural Networks for Image Segementation and Classification

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.

Set up `Detectron2`

In [1]:
```
# install dependencies:
!pip install pyyaml==5.1 pycocotools>=2.0.1
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
# opencv is pre-installed on colab
```

```
1.6.0+cu101 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.
```

In [2]:
```
# install detectron2: (Colab has CUDA 10.1 + torch 1.6)
# See https://detectron2.readthedocs.io/tutorials/install.html for instructions
assert torch.__version__.startswith("1.6")
!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.6/index.html
```

```
Looking in links: https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.6/index.html
Requirement already satisfied: detectron2 in /usr/local/lib/python3.6/dist-packages (0.2.1+cu101)
Requirement already satisfied: Pillow>=7.1 in /usr/local/lib/python3.6/dist-packages (from detectron
2) (8.0.0)
Requirement already satisfied: tqdm>4.29.0 in /usr/local/lib/python3.6/dist-packages (from detectron
2) (4.41.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from detectron
2) (3.2.2)
Requirement already satisfied: termcolor>=1.1 in /usr/local/lib/python3.6/dist-packages (from detect
ron2) (1.1.0)
Requirement already satisfied: pydot in /usr/local/lib/python3.6/dist-packages (from detectron2) (1.
3.0)
Requirement already satisfied: fvcore>=0.1.1 in /usr/local/lib/python3.6/dist-packages (from detectr
on2) (0.1.2.post20201016)
Requirement already satisfied: pycocotools>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from de
tectron2) (2.0.2)
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from detectron2)
(0.16.0)
Requirement already satisfied: yacs>=0.1.6 in /usr/local/lib/python3.6/dist-packages (from detectron
2) (0.1.8)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.6/dist-packages (from detectron
2) (1.3.0)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.6/dist-packages (from detectron
2) (2.3.0)
Requirement already satisfied: mock in /usr/local/lib/python3.6/dist-packages (from detectron2) (4.
0.2)
Requirement already satisfied: tabulate in /usr/local/lib/python3.6/dist-packages (from detectron2)
(0.8.7)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/
dist-packages (from matplotlib->detectron2) (2.4.7)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.6/dist-packages (from matplotli
b->detectron2) (1.18.5)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotl
ib->detectron2) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from mat
plotlib->detectron2) (1.2.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from
matplotlib->detectron2) (2.8.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.6/dist-packages (from fvcore>=
0.1.1->detectron2) (5.1)
Requirement already satisfied: portalocker in /usr/local/lib/python3.6/dist-packages (from fvcore>=
0.1.1->detectron2) (2.0.0)
Requirement already satisfied: cython>=0.27.3 in /usr/local/lib/python3.6/dist-packages (from pycoco
```

```
tools>=2.0.1->detectron2) (0.29.21)
Requirement already satisfied: setuptools>=18.0 in /usr/local/lib/python3.6/dist-packages (from pyco
cotools>=2.0.1->detectron2) (50.3.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-packages (from ten
sorboard->detectron2) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packages (from tenso
rboard->detectron2) (3.2.2)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.6/dist-packages (from tensor
board->detectron2) (1.32.0)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in /usr/local/lib/python3.6/dist-p
ackages (from tensorboard->detectron2) (0.35.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from tensorboa
rd->detectron2) (1.15.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.6/dist-packag
es (from tensorboard->detectron2) (1.7.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.6/dist-pac
kages (from tensorboard->detectron2) (0.4.1)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.6/dist-packages (from
tensorboard->detectron2) (1.17.2)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.6/dist-packages (from t
ensorboard->detectron2) (2.23.0)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.6/dist-packages (from tensorbo
ard->detectron2) (0.10.0)
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.6/dist-packages (from tenso
rboard->detectron2) (3.12.4)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/local/lib/python3.
6/dist-packages (from markdown>=2.6.8->tensorboard->detectron2) (2.0.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.6/dist-packages (f
rom google-auth-oauthlib<0.5,>=0.4.1->tensorboard->detectron2) (1.3.0)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3" in /usr/local/lib/python3.6/dist
-packages (from google-auth<2,>=1.6.3->tensorboard->detectron2) (4.6)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dist-packages (fro
m google-auth<2,>=1.6.3->tensorboard->detectron2) (4.1.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.6/dist-packages (from
google-auth<2,>=1.6.3->tensorboard->detectron2) (0.2.8)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests
<3,>=2.21.0->tensorboard->detectron2) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from req
uests<3,>=2.21.0->tensorboard->detectron2) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/d
ist-packages (from requests<3,>=2.21.0->tensorboard->detectron2) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from re
quests<3,>=2.21.0->tensorboard->detectron2) (2020.6.20)
```

```
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from importlib-m
etadata; python_version < "3.8"->markdown>=2.6.8->tensorboard->detectron2) (3.2.0)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from reque
sts-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard->detectron2) (3.1.0)
Requirement already satisfied: pyasn1>=0.1.3 in /usr/local/lib/python3.6/dist-packages (from rsa<5,>
=3.1.4; python_version >= "3"->google-auth<2,>=1.6.3->tensorboard->detectron2) (0.4.8)
```

1. COCO Keypoint Person Detector model with a ResNet50-FPN backbone

In [26]:
```
# import some common detectron2 utilities
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog
import cv2
from detectron2 import model_zoo

# get image
!wget https://images.fineartamerica.com/images-medium-large-5/central-park-balloon-man-madeline-elli
s.jpg -O input_1.jpg
im = cv2.imread("./input_1.jpg")

# Create config
cfg = get_cfg()
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 = 0.16  # set threshold for this model
cfg.MODEL.WEIGHTS = "https://dl.fbaipublicfiles.com/detectron2/COCO-Keypoints/keypoint_rcnn_R_50_FPN_
3x/137849621/model_final_a6e10b.pkl"
# Create predictor
predictor = DefaultPredictor(cfg)

# Make prediction
outputs = predictor(im)
```

```
--2020-10-16 16:09:51--  https://images.fineartamerica.com/images-medium-large-5/central-park-balloo
n-man-madeline-ellis.jpg
Resolving images.fineartamerica.com (images.fineartamerica.com)... 99.86.38.45, 99.86.38.81, 99.86.3
8.84, ...
Connecting to images.fineartamerica.com (images.fineartamerica.com)|99.86.38.45|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 248003 (242K) [image/jpeg]
Saving to: ‘input_1.jpg’

input_1.jpg         100%[===================>] 242.19K  --.-KB/s    in 0.02s

2020-10-16 16:09:51 (9.93 MB/s) - ‘input_1.jpg’ saved [248003/248003]
```

Here is the result of using `COCO Keypoint Person Detector` model with a `ResNet50-FPN` backbone. The threshold is `0.16`, and the performance is good. Most of the human sihouettes are deteced with high confidence. And it also decteced the small human sihouettes which are in the background.

In [27]:
```
from google.colab.patches import cv2_imshow
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(v.get_image()[:, :, ::-1])
```

2. Mask R-CNN model with ResNet50- FPN backbone

The result is shown below. As we can see, this model recongize multiply items in the image. For example, it recongizes `Person, Backpack, Sport Ball, Apple, Umbrella` . However, it mistakly regards `Ballon` as `Umbrella` or `Sport Ball` . The reason of this is that `the COCO dataset used to train the above models does not contain balloons` .

In [3]:
```
# Create config
cfg_2 = get_cfg()

cfg_2.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg_2.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5  # set threshold for this model

cfg_2.MODEL.WEIGHTS = "https://dl.fbaipublicfiles.com/detectron2/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x/137849600/model_final_f10217.pkl"
# Create predictor
predictor_2 = DefaultPredictor(cfg_2)

# Make prediction
outputs_2 = predictor_2(im)

# Visualization the result
v_2 = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg_2.DATASETS.TRAIN[0]), scale=1.2)
v_2 = v_2.draw_instance_predictions(outputs_2["instances"].to("cpu"))
cv2_imshow(v_2.get_image()[:, :, ::-1])
```

3. Train `Mask R-CNN model` to get a `Balloon Detector`

First, get the balloon image dataset.

In [6]:
```
# download, decompress the data
!wget https://github.com/matterport/Mask_RCNN/releases/download/v2.1/balloon_dataset.zip
!unzip balloon_dataset.zip > /dev/null
```

```
--2020-10-16 15:24:30--  https://github.com/matterport/Mask_RCNN/releases/download/v2.1/balloon_data
set.zip
Resolving github.com (github.com)... 192.30.255.112
Connecting to github.com (github.com)|192.30.255.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/107595270/737339e2-2b83-11
e8-856a-188034eb3468?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F202010
16%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20201016T152430Z&X-Amz-Expires=300&X-Amz-Signature=8b6
bfe37cf9a5c8d2e01c0154d3820ec531c9ecc28acf3c85876d4203153308c&X-Amz-SignedHeaders=host&actor_id=0&ke
y_id=0&repo_id=107595270&response-content-disposition=attachment%3B%20filename%3Dballoon_dataset.zip
&response-content-type=application%2Foctet-stream [following]
--2020-10-16 15:24:30--  https://github-production-release-asset-2e65be.s3.amazonaws.com/107595270/7
37339e2-2b83-11e8-856a-188034eb3468?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CS
VEH53A%2F20201016%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20201016T152430Z&X-Amz-Expires=300&X-Am
z-Signature=8b6bfe37cf9a5c8d2e01c0154d3820ec531c9ecc28acf3c85876d4203153308c&X-Amz-SignedHeaders=hos
t&actor_id=0&key_id=0&repo_id=107595270&response-content-disposition=attachment%3B%20filename%3Dball
oon_dataset.zip&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2
e65be.s3.amazonaws.com)... 52.217.90.204
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-ass
et-2e65be.s3.amazonaws.com)|52.217.90.204|: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  28.0MB/s    in 1.3s

2020-10-16 15:24:32 (28.0 MB/s) - 'balloon_dataset.zip' saved [38741381/38741381]
```

Second, change the `custom dataset format` into `detectron2's standard format`.

In [28]:
```
from detectron2.structures import BoxMode
from detectron2.data import MetadataCatalog, DatasetCatalog

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")
```

Third, train the pre-trained `Mask R-CNN` model on the `Ballon dataset`.

In [29]:
```
from detectron2.engine import DefaultTrainer
import os, json
import numpy as np

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.yam
l")  # 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 will need t
o train longer for a practical dataset
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128   # faster, and good enough for this toy dataset (defa
ult: 512)
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()
```

```
[10/16 16:12:00 d2.engine.defaults]: Model:
GeneralizedRCNN(
  (backbone): FPN(
    (fpn_lateral2): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (fpn_lateral3): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (fpn_lateral4): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output4): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (fpn_lateral5): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output5): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (top_block): LastLevelMaxPool()
    (bottom_up): 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=False
        (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=False
          (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=False
          (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=False
          (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=False
          (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=False
          (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=False
          (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=False
      (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=False
      (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=False
      (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)
    )
  )
)
(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=False
      (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=False
      (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=False
            (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)
          )
        )
      )
    )
  )
  (proposal_generator): RPN(
    (rpn_head): StandardRPNHead(
      (conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (objectness_logits): Conv2d(256, 3, kernel_size=(1, 1), stride=(1, 1))
      (anchor_deltas): Conv2d(256, 12, kernel_size=(1, 1), stride=(1, 1))
    )
    (anchor_generator): DefaultAnchorGenerator(
      (cell_anchors): BufferList()
    )
  )
  (roi_heads): StandardROIHeads(
    (box_pooler): ROIPooler(
      (level_poolers): ModuleList(
        (0): ROIAlign(output_size=(7, 7), spatial_scale=0.25, sampling_ratio=0, aligned=True)
        (1): ROIAlign(output_size=(7, 7), spatial_scale=0.125, sampling_ratio=0, aligned=True)
        (2): ROIAlign(output_size=(7, 7), spatial_scale=0.0625, sampling_ratio=0, aligned=True)
        (3): ROIAlign(output_size=(7, 7), spatial_scale=0.03125, sampling_ratio=0, aligned=True)
      )
    )
    (box_head): FastRCNNConvFCHead(
      (fc1): Linear(in_features=12544, out_features=1024, bias=True)
      (fc2): Linear(in_features=1024, out_features=1024, bias=True)
    )
    (box_predictor): FastRCNNOutputLayers(
      (cls_score): Linear(in_features=1024, out_features=2, bias=True)
      (bbox_pred): Linear(in_features=1024, out_features=4, bias=True)
    )
    (mask_pooler): ROIPooler(
      (level_poolers): ModuleList(
        (0): ROIAlign(output_size=(14, 14), spatial_scale=0.25, sampling_ratio=0, aligned=True)
        (1): ROIAlign(output_size=(14, 14), spatial_scale=0.125, sampling_ratio=0, aligned=True)
        (2): ROIAlign(output_size=(14, 14), spatial_scale=0.0625, sampling_ratio=0, aligned=True)
```

```
        (3): ROIAlign(output_size=(14, 14), spatial_scale=0.03125, sampling_ratio=0, aligned=True)
      )
    )
    (mask_head): MaskRCNNConvUpsampleHead(
      (mask_fcn1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (mask_fcn2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (mask_fcn3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (mask_fcn4): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (deconv): ConvTranspose2d(256, 256, kernel_size=(2, 2), stride=(2, 2))
      (predictor): Conv2d(256, 1, kernel_size=(1, 1), stride=(1, 1))
    )
  )
)
[10/16 16:12:02 d2.data.build]: Removed 0 images with no usable annotations. 61 images left.
[10/16 16:12:02 d2.data.build]: Distribution of instances among all 1 categories:
|  category  | #instances   |
|:----------:|:-------------|
|  balloon   | 255          |
|            |              |
[10/16 16:12:02 d2.data.common]: Serializing 61 elements to byte tensors and concatenating them all
 ...
[10/16 16:12:02 d2.data.common]: Serialized dataset takes 0.17 MiB
[10/16 16:12:02 d2.data.dataset_mapper]: Augmentations used in training: [ResizeShortestEdge(short_e
dge_length=(640, 672, 704, 736, 768, 800), max_size=1333, sample_style='choice'), RandomFlip()]
[10/16 16:12:02 d2.data.build]: Using training sampler TrainingSampler

Skip loading parameter 'roi_heads.box_predictor.cls_score.weight' to the model due to incompatible s
hapes: (81, 1024) in the checkpoint but (2, 1024) in the model! You might want to double check if th
is is expected.
Skip loading parameter 'roi_heads.box_predictor.cls_score.bias' to the model due to incompatible sha
pes: (81,) in the checkpoint but (2,) in the model! You might want to double check if this is expect
ed.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.weight' to the model due to incompatible s
hapes: (320, 1024) in the checkpoint but (4, 1024) in the model! You might want to double check if t
his is expected.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.bias' to the model due to incompatible sha
pes: (320,) in the checkpoint but (4,) in the model! You might want to double check if this is expec
ted.
Skip loading parameter 'roi_heads.mask_head.predictor.weight' to the model due to incompatible shape
s: (80, 256, 1, 1) in the checkpoint but (1, 256, 1, 1) in the 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 want to double check if this is expected.
```

```
[10/16 16:12:04 d2.engine.train_loop]: Starting training from iteration 0
[10/16 16:12:14 d2.utils.events]:  eta: 0:02:15  iter: 19  total_loss: 2.117  loss_cls: 0.760  loss_
box_reg: 0.645  loss_mask: 0.680  loss_rpn_cls: 0.026  loss_rpn_loc: 0.011  time: 0.4788  data_time:
0.0310  lr: 0.000005  max_mem: 2965M
[10/16 16:12:24 d2.utils.events]:  eta: 0:02:04  iter: 39  total_loss: 2.067  loss_cls: 0.704  loss_
box_reg: 0.663  loss_mask: 0.651  loss_rpn_cls: 0.034  loss_rpn_loc: 0.009  time: 0.4824  data_time:
0.0096  lr: 0.000010  max_mem: 2965M
[10/16 16:12:34 d2.utils.events]:  eta: 0:02:00  iter: 59  total_loss: 1.872  loss_cls: 0.639  loss_
box_reg: 0.581  loss_mask: 0.599  loss_rpn_cls: 0.031  loss_rpn_loc: 0.006  time: 0.4925  data_time:
0.0089  lr: 0.000015  max_mem: 3086M
[10/16 16:12:43 d2.utils.events]:  eta: 0:01:48  iter: 79  total_loss: 1.821  loss_cls: 0.517  loss_
box_reg: 0.629  loss_mask: 0.528  loss_rpn_cls: 0.023  loss_rpn_loc: 0.008  time: 0.4827  data_time:
0.0075  lr: 0.000020  max_mem: 3086M
[10/16 16:12:52 d2.utils.events]:  eta: 0:01:37  iter: 99  total_loss: 1.597  loss_cls: 0.485  loss_
box_reg: 0.648  loss_mask: 0.462  loss_rpn_cls: 0.034  loss_rpn_loc: 0.006  time: 0.4823  data_time:
0.0077  lr: 0.000025  max_mem: 3086M
[10/16 16:13:02 d2.utils.events]:  eta: 0:01:28  iter: 119  total_loss: 1.490  loss_cls: 0.415  loss
_box_reg: 0.611  loss_mask: 0.410  loss_rpn_cls: 0.027  loss_rpn_loc: 0.005  time: 0.4814  data_tim
e: 0.0106  lr: 0.000030  max_mem: 3086M
[10/16 16:13:12 d2.utils.events]:  eta: 0:01:18  iter: 139  total_loss: 1.428  loss_cls: 0.386  loss
_box_reg: 0.648  loss_mask: 0.362  loss_rpn_cls: 0.019  loss_rpn_loc: 0.008  time: 0.4811  data_tim
e: 0.0078  lr: 0.000035  max_mem: 3086M
[10/16 16:13:21 d2.utils.events]:  eta: 0:01:07  iter: 159  total_loss: 1.373  loss_cls: 0.326  loss
_box_reg: 0.645  loss_mask: 0.310  loss_rpn_cls: 0.013  loss_rpn_loc: 0.008  time: 0.4796  data_tim
e: 0.0063  lr: 0.000040  max_mem: 3086M
[10/16 16:13:31 d2.utils.events]:  eta: 0:00:58  iter: 179  total_loss: 1.327  loss_cls: 0.304  loss
_box_reg: 0.632  loss_mask: 0.282  loss_rpn_cls: 0.037  loss_rpn_loc: 0.012  time: 0.4789  data_tim
e: 0.0092  lr: 0.000045  max_mem: 3086M
[10/16 16:13:40 d2.utils.events]:  eta: 0:00:48  iter: 199  total_loss: 1.191  loss_cls: 0.255  loss
_box_reg: 0.667  loss_mask: 0.250  loss_rpn_cls: 0.023  loss_rpn_loc: 0.006  time: 0.4794  data_tim
e: 0.0064  lr: 0.000050  max_mem: 3086M
[10/16 16:13:50 d2.utils.events]:  eta: 0:00:39  iter: 219  total_loss: 1.060  loss_cls: 0.238  loss
_box_reg: 0.597  loss_mask: 0.194  loss_rpn_cls: 0.018  loss_rpn_loc: 0.004  time: 0.4789  data_tim
e: 0.0080  lr: 0.000055  max_mem: 3086M
[10/16 16:14:00 d2.utils.events]:  eta: 0:00:29  iter: 239  total_loss: 1.119  loss_cls: 0.218  loss
_box_reg: 0.634  loss_mask: 0.225  loss_rpn_cls: 0.023  loss_rpn_loc: 0.011  time: 0.4804  data_tim
e: 0.0101  lr: 0.000060  max_mem: 3086M
[10/16 16:14:09 d2.utils.events]:  eta: 0:00:19  iter: 259  total_loss: 0.930  loss_cls: 0.174  loss
_box_reg: 0.564  loss_mask: 0.160  loss_rpn_cls: 0.018  loss_rpn_loc: 0.009  time: 0.4803  data_tim
e: 0.0060  lr: 0.000065  max_mem: 3086M
[10/16 16:14:19 d2.utils.events]:  eta: 0:00:10  iter: 279  total_loss: 0.924  loss_cls: 0.168  loss
_box_reg: 0.565  loss_mask: 0.161  loss_rpn_cls: 0.016  loss_rpn_loc: 0.008  time: 0.4793  data_tim
e: 0.0065  lr: 0.000070  max_mem: 3086M
```

```
[10/16 16:14:30 d2.utils.events]:  eta: 0:00:00  iter: 299  total_loss: 0.827  loss_cls: 0.136  loss
_box_reg: 0.484  loss_mask: 0.131  loss_rpn_cls: 0.009  loss_rpn_loc: 0.007  time: 0.4795  data_tim
e: 0.0087  lr: 0.000075  max_mem: 3086M
[10/16 16:14:30 d2.engine.hooks]: Overall training speed: 297 iterations in 0:02:22 (0.4811 s / it)
[10/16 16:14:30 d2.engine.hooks]: Total training time: 0:02:25 (0:00:02 on hooks)
```

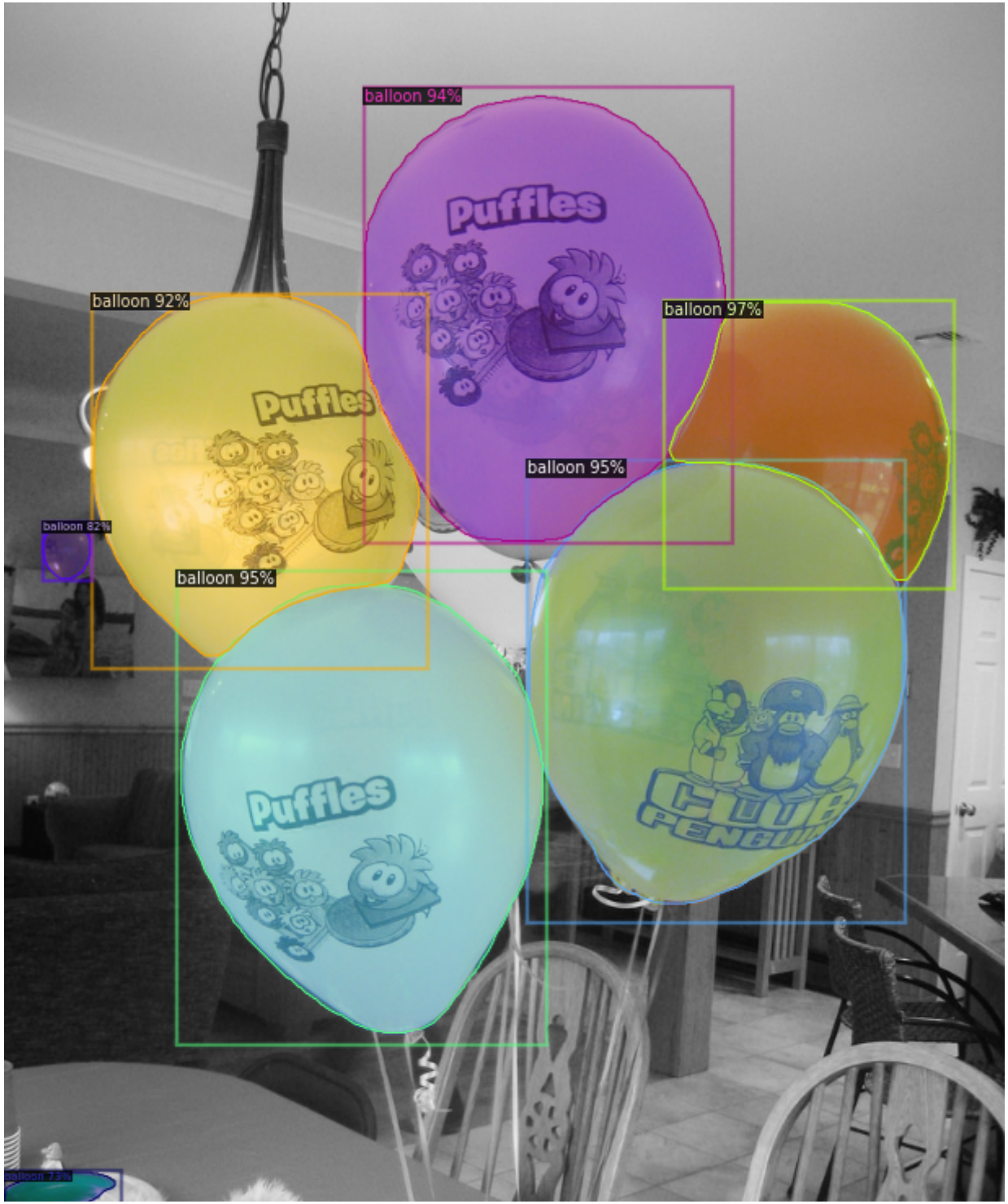
Now that the model is trained it can be used for inference on the `validation set` .

In [31]:
```
from detectron2.utils.visualizer import ColorMode
import random

# load 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 model
# Set training data-set path
cfg.DATASETS.TEST = ("balloon/val", )
# Create predictor (model for inference)
predictor = DefaultPredictor(cfg)

dataset_dicts = get_balloon_dicts("balloon/val")
for d in random.sample(dataset_dicts, 3):
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1],
                   metadata=balloon_metadata,
                   scale=0.8,
                   instance_mode=ColorMode.IMAGE_BW   # remove the colors of unsegmented pixels
    )
    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2_imshow(v.get_image()[:, :, ::-1])
```


balloon 96%
T★A★P★S
MENTOR
Remember the Love,
Celebrate the Life,
Share the journey
T★A★P★S
MENTOR
Remember the Love,
Celebrate the Life,
Share the journey


balloon 94%
balloon 92%
balloon 97%
balloon 95%
balloon 82%
balloon 95%
Puffles
Puffles
Puffles
CLUB PENGUIN


balloon 95%

Finally, use our `Ballon Detector` model that we just trained on our `input image` . As we can see, the `ballons` are successfully dectected.

In [33]:
```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")  # path to the model we just trai
ned
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set a custom testing threshold
predictor_3 = DefaultPredictor(cfg)

# Make prediction
im = cv2.imread("./input_1.jpg")
outputs_3 = predictor_3(im)

# Visualization the result
v_3 = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
v_3 = v_3.draw_instance_predictions(outputs_3["instances"].to("cpu"))
cv2_imshow(v_3.get_image()[:, :, ::-1])
```


balloon 78%
balloon 80%
balloon 73%
balloon 89%
balloon 79%
fine art
america