ViLD_demo

June 3, 2022

1 Copyright

[1]:
```
#@title Collapsed copyright
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

2 Demo for open-vocabulary ViLD object detector

Open-vocabulary Object Detection via Vision and Language Knowledge Distillation, Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, Yin Cui.

The ViLD model in this colab has a backbone of ResNet-152 and the distillation weight is 0.1. It achieves 16.8 APr and 26.4 overall AP on LVIS.

For faster inference speed, please use a GPU runtime. You can use your local GPU runtime or change the runtime type to “GPU” from Runtime -> Change Runtime Type in the menu.

3 Preparation

[1]:
```
#@title Install dependencies
! pip install git+https://github.com/openai/CLIP.git
```

```
Collecting git+https://github.com/openai/CLIP.git
  Cloning https://github.com/openai/CLIP.git to
/private/var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/pip-req-build-ex0vw8ka
  Running command git clone -q https://github.com/openai/CLIP.git
```

```
/private/var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/pip-req-build-ex0vw8ka
  Resolved https://github.com/openai/CLIP.git to commit
b46f5ac7587d2e1862f8b7b1573179d80dcdd620
Collecting ftfy
  Downloading ftfy-6.1.1-py3-none-any.whl (53 kB)
     |                 | 53 kB 894 kB/s eta 0:00:01
Collecting regex
  Downloading regex-2022.4.24-cp39-cp39-macosx_10_9_x86_64.whl (288 kB)
     |                 | 288 kB 1.8 MB/s eta 0:00:01
Collecting tqdm
  Using cached tqdm-4.64.0-py2.py3-none-any.whl (78 kB)
Collecting torch
  Downloading torch-1.11.0-cp39-none-macosx_10_9_x86_64.whl (129.9 MB)
     |                 | 129.9 MB 10.4 MB/s eta 0:00:01
Collecting torchvision
  Downloading torchvision-0.12.0-cp39-cp39-macosx_10_9_x86_64.whl (1.2 MB)
     |                 | 1.2 MB 10.4 MB/s eta 0:00:01
Requirement already satisfied: wcwidth>=0.2.5 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
ftfy->clip==1.0) (0.2.5)
Requirement already satisfied: typing-extensions in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
torch->clip==1.0) (4.2.0)
Requirement already satisfied: requests in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
torchvision->clip==1.0) (2.27.1)
Requirement already satisfied: numpy in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
torchvision->clip==1.0) (1.22.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
torchvision->clip==1.0) (9.1.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
requests->torchvision->clip==1.0) (1.26.9)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
requests->torchvision->clip==1.0) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
requests->torchvision->clip==1.0) (2022.5.18.1)
Requirement already satisfied: idna<4,>=2.5 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
requests->torchvision->clip==1.0) (3.3)
Building wheels for collected packages: clip
  Building wheel for clip (setup.py) … done
  Created wheel for clip: filename=clip-1.0-py3-none-any.whl size=1369378
sha256=dd49a4bedfc2f83cb83fc3922704be6fba7b7c15853250ca41d700ed3a25ea4a
```

```
  Stored in directory:
/private/var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/pip-ephem-wheel-cache-
vss1gqjg/wheels/c8/e4/e1/11374c111387672fc2068dfbe0d4b424cb9cdd1b2e184a71b5
Successfully built clip
Installing collected packages: torch, tqdm, torchvision, regex, ftfy, clip
Successfully installed clip-1.0 ftfy-6.1.1 regex-2022.4.24 torch-1.11.0
torchvision-0.12.0 tqdm-4.64.0
```

[3]:
```
! pip install gsutil
```

```
Requirement already satisfied: gsutil in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (5.10)
Requirement already satisfied: monotonic>=1.4 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(1.6)
Requirement already satisfied: google-reauth>=0.1.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(0.1.1)
Requirement already satisfied: retry-decorator>=1.0.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(1.1.1)
Requirement already satisfied: gcs-oauth2-boto-plugin>=3.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(3.0)
Requirement already satisfied: google-apitools>=0.5.32 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(0.5.32)
Requirement already satisfied: argcomplete>=1.9.4 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(2.0.0)
Requirement already satisfied: httplib2>=0.20.4 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(0.20.4)
Requirement already satisfied: fasteners>=0.14.1 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(0.17.3)
Requirement already satisfied: crcmod>=1.7 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(1.7)
Requirement already satisfied: google-auth[aiohttp]>=2.5.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(2.6.6)
Requirement already satisfied: pyOpenSSL>=0.13 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(22.0.0)
Requirement already satisfied: six>=1.12.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gsutil)
(1.16.0)
```

```
Requirement already satisfied: oauth2client>=2.2.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gcs-
oauth2-boto-plugin>=3.0->gsutil) (4.1.3)
Requirement already satisfied: rsa==4.7.2 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gcs-
oauth2-boto-plugin>=3.0->gsutil) (4.7.2)
Requirement already satisfied: boto>=2.29.1 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from gcs-
oauth2-boto-plugin>=3.0->gsutil) (2.49.0)
Requirement already satisfied: pyasn1>=0.1.3 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
rsa==4.7.2->gcs-oauth2-boto-plugin>=3.0->gsutil) (0.4.8)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from google-
auth[aiohttp]>=2.5.0->gsutil) (5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from google-
auth[aiohttp]>=2.5.0->gsutil) (0.2.8)
Requirement already satisfied: requests<3.0.0dev,>=2.20.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from google-
auth[aiohttp]>=2.5.0->gsutil) (2.27.1)
Requirement already satisfied: aiohttp<4.0.0dev,>=3.6.2 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from google-
auth[aiohttp]>=2.5.0->gsutil) (3.8.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
aiohttp<4.0.0dev,>=3.6.2->google-auth[aiohttp]>=2.5.0->gsutil) (6.0.2)
Requirement already satisfied: frozenlist>=1.1.1 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
aiohttp<4.0.0dev,>=3.6.2->google-auth[aiohttp]>=2.5.0->gsutil) (1.3.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
aiohttp<4.0.0dev,>=3.6.2->google-auth[aiohttp]>=2.5.0->gsutil) (1.2.0)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
aiohttp<4.0.0dev,>=3.6.2->google-auth[aiohttp]>=2.5.0->gsutil) (2.0.12)
Requirement already satisfied: yarl<2.0,>=1.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
aiohttp<4.0.0dev,>=3.6.2->google-auth[aiohttp]>=2.5.0->gsutil) (1.7.2)
Requirement already satisfied: attrs>=17.3.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
aiohttp<4.0.0dev,>=3.6.2->google-auth[aiohttp]>=2.5.0->gsutil) (21.4.0)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
aiohttp<4.0.0dev,>=3.6.2->google-auth[aiohttp]>=2.5.0->gsutil) (4.0.2)
Requirement already satisfied: pyu2f in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from google-
reauth>=0.1.0->gsutil) (0.1.5)
```

```
Requirement already satisfied:
pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
httplib2>=0.20.4->gsutil) (3.0.9)
Requirement already satisfied: cryptography>=35.0 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
pyOpenSSL>=0.13->gsutil) (37.0.2)
Requirement already satisfied: cffi>=1.12 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
cryptography>=35.0->pyOpenSSL>=0.13->gsutil) (1.15.0)
Requirement already satisfied: pycparser in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
cffi>=1.12->cryptography>=35.0->pyOpenSSL>=0.13->gsutil) (2.21)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
requests<3.0.0dev,>=2.20.0->google-auth[aiohttp]>=2.5.0->gsutil) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
requests<3.0.0dev,>=2.20.0->google-auth[aiohttp]>=2.5.0->gsutil) (2022.5.18.1)
Requirement already satisfied: idna<4,>=2.5 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from
requests<3.0.0dev,>=2.20.0->google-auth[aiohttp]>=2.5.0->gsutil) (3.3)
```

[4]:
```
#@title Download files
!gsutil cp -r gs://cloud-tpu-checkpoints/detection/projects/vild/colab/
 ↪image_path_v2 ./
#@ About Pb File : https://zhuanlan.zhihu.com/p/32887066
#// image_path_v2
#//    saved_model.pb
#//    variables
#//       variables.data-00000-of-00001
#//       variables.index

!gsutil cp -r gs://cloud-tpu-checkpoints/detection/projects/vild/colab/examples␣
 ↪./
#@ examples have jpg img

#@ Download and unzip the LVIS v1.0 validation sets to DATA_DIR.
```

```
Copying gs://cloud-tpu-
checkpoints/detection/projects/vild/colab/image_path_v2/saved_model.pb…
Copying gs://cloud-tpu-checkpoints/detection/projects/vild/colab/image_path_v2/v
ariables/variables.data-00000-of-00001…
If you experience problems with multiprocessing on MacOS, they might be related
to https://bugs.python.org/issue33725. You can disable multiprocessing by
editing your .boto config or by adding the following flag to your command: `-o
"GSUtil:parallel_process_count=1"`. Note that multithreading is still available
even if you disable multiprocessing.
```

```
Copying gs://cloud-tpu-checkpoints/detection/projects/vild/colab/image_path_v2/v
ariables/variables.index…
- [3 files][323.7 MiB/323.7 MiB]  770.2 KiB/s
Operation completed over 3 objects/323.7 MiB.
Copying gs://cloud-tpu-checkpoints/detection/projects/vild/colab/examples/five_w
omen_and_umbrellas.jpg…
Copying gs://cloud-tpu-
checkpoints/detection/projects/vild/colab/examples/three_toys.jpg…
\ [2 files][447.5 KiB/447.5 KiB]
Operation completed over 2 objects/447.5 KiB.
```

[11]:
```
! pip install easyDict
! pip install scipy
! pip install pyyaml
```

```
Requirement already satisfied: easyDict in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (1.9)
Requirement already satisfied: scipy in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (1.8.1)
Requirement already satisfied: numpy<1.25.0,>=1.17.3 in
/Applications/anaconda3/envs/zeroShot/lib/python3.9/site-packages (from scipy)
(1.22.4)
Collecting pyyaml
  Downloading PyYAML-6.0-cp39-cp39-macosx_10_9_x86_64.whl (197 kB)
     |                  | 197 kB 1.3 MB/s eta 0:00:01
Installing collected packages: pyyaml
Successfully installed pyyaml-6.0
```

[12]:
```
#@title Import libraries
from easydict import EasyDict

import numpy as np
import torch
import clip

from tqdm import tqdm

from matplotlib import pyplot as plt
from matplotlib import patches

import collections
import json
import numpy as np

import os
import os.path as osp
```

```
from PIL import Image
from pprint import pprint
from scipy.special import softmax
import yaml


import tensorflow.compat.v1 as tf

import cv2
```

[13]:
```
#@title Define hyperparameters
FLAGS = {
    'prompt_engineering': True,
    'this_is': True,

    'temperature': 100.0,
    'use_softmax': False,
}
FLAGS = EasyDict(FLAGS)


# Global matplotlib settings
SMALL_SIZE = 16#10
MEDIUM_SIZE = 18#12
BIGGER_SIZE = 20#14

plt.rc('font', size=MEDIUM_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=BIGGER_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=MEDIUM_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title


# Parameters for drawing figure.
display_input_size = (10, 10)
overall_fig_size = (18, 24)

line_thickness = 2
fig_size_w = 35
# fig_size_h = min(max(5, int(len(category_names) / 2.5) ), 10)
mask_color =   'red'
alpha = 0.5
```

3.1 Build text embeddings

We use the CLIP model from OpenAI: https://github.com/openai/CLIP.

[14]:
```
def article(name):
  return 'an' if name[0] in 'aeiou' else 'a'

def processed_name(name, rm_dot=False):
  # _ for lvis
  # / for obj365
  res = name.replace('_', ' ').replace('/', ' or ').lower()
  if rm_dot:
    res = res.rstrip('.')
  return res

single_template = [
    'a photo of {article} {}.'
]

multiple_templates = [
    'There is {article} {} in the scene.',
    'There is the {} in the scene.',
    'a photo of {article} {} in the scene.',
    'a photo of the {} in the scene.',
    'a photo of one {} in the scene.',


    'itap of {article} {}.',
    'itap of my {}.',  # itap: I took a picture of
    'itap of the {}.',
    'a photo of {article} {}.',
    'a photo of my {}.',
    'a photo of the {}.',
    'a photo of one {}.',
    'a photo of many {}.',

    'a good photo of {article} {}.',
    'a good photo of the {}.',
    'a bad photo of {article} {}.',
    'a bad photo of the {}.',
    'a photo of a nice {}.',
    'a photo of the nice {}.',
    'a photo of a cool {}.',
    'a photo of the cool {}.',
    'a photo of a weird {}.',
    'a photo of the weird {}.',

    'a photo of a small {}.',
    'a photo of the small {}.',
    'a photo of a large {}.',
    'a photo of the large {}.',
```

```
    'a photo of a clean {}.',
    'a photo of the clean {}.',
    'a photo of a dirty {}.',
    'a photo of the dirty {}.',

    'a bright photo of {article} {}.',
    'a bright photo of the {}.',
    'a dark photo of {article} {}.',
    'a dark photo of the {}.',

    'a photo of a hard to see {}.',
    'a photo of the hard to see {}.',
    'a low resolution photo of {article} {}.',
    'a low resolution photo of the {}.',
    'a cropped photo of {article} {}.',
    'a cropped photo of the {}.',
    'a close-up photo of {article} {}.',
    'a close-up photo of the {}.',
    'a jpeg corrupted photo of {article} {}.',
    'a jpeg corrupted photo of the {}.',
    'a blurry photo of {article} {}.',
    'a blurry photo of the {}.',
    'a pixelated photo of {article} {}.',
    'a pixelated photo of the {}.',

    'a black and white photo of the {}.',
    'a black and white photo of {article} {}.',

    'a plastic {}.',
    'the plastic {}.',

    'a toy {}.',
    'the toy {}.',
    'a plushie {}.',
    'the plushie {}.',
    'a cartoon {}.',
    'the cartoon {}.',

    'an embroidered {}.',
    'the embroidered {}.',

    'a painting of the {}.',
    'a painting of a {}.',
]
```

[15]:
```
clip.available_models()
model, preprocess = clip.load("ViT-B/32")
```

```
100%|                    | 338M/338M [00:39<00:00, 9.05MiB/s]
```

[16]:
```
def build_text_embedding(categories):
  if FLAGS.prompt_engineering:
    templates = multiple_templates
  else:
    templates = single_template

  run_on_gpu = torch.cuda.is_available()

  with torch.no_grad():
    all_text_embeddings = []
    print('Building text embeddings...')
    for category in tqdm(categories):
      texts = [
        template.format(processed_name(category['name'], rm_dot=True),
                        article=article(category['name']))
        for template in templates]
      if FLAGS.this_is:
        texts = [
                 'This is ' + text if text.startswith('a') or text.
 ↪startswith('the') else text
                 for text in texts
                 ]
      texts = clip.tokenize(texts) #tokenize
      if run_on_gpu:
        texts = texts.cuda()
      text_embeddings = model.encode_text(texts) #embed with text encoder
      text_embeddings /= text_embeddings.norm(dim=-1, keepdim=True)
      text_embedding = text_embeddings.mean(dim=0)
      text_embedding /= text_embedding.norm()
      all_text_embeddings.append(text_embedding)
    all_text_embeddings = torch.stack(all_text_embeddings, dim=1)
    if run_on_gpu:
      all_text_embeddings = all_text_embeddings.cuda()
  return all_text_embeddings.cpu().numpy().T
```

3.2 Load ViLD model

[17]:
```
session = tf.Session(graph=tf.Graph())
```

```
2022-06-03 03:15:10.777447: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
```

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

[18]:
```
saved_model_dir = './image_path_v2' #@param {type:"string"}

_ = tf.saved_model.loader.load(session, ['serve'], saved_model_dir)
```

```
WARNING:tensorflow:From /var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/ipykern
el_57573/3084755540.py:3: load (from tensorflow.python.saved_model.loader_impl)
is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as
tf.compat.v1.saved_model.loader.load or tf.compat.v1.saved_model.load. There
will be a new function for importing SavedModels in Tensorflow 2.0.
INFO:tensorflow:Restoring parameters from ./image_path_v2/variables/variables
```

```
2022-06-03 03:15:20.606310: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:354] MLIR V1
optimization pass is not enabled
```

3.3 Helper functioins

[19]:
```
numbered_categories = [{'name': str(idx), 'id': idx,} for idx in range(50)]
numbered_category_indices = {cat['id']: cat for cat in numbered_categories}
```

[21]:
```
#@title NMS
def nms(dets, scores, thresh, max_dets=1000):
  """Non-maximum suppression.
  Args:
    dets: [N, 4]
    scores: [N,]
    thresh: iou threshold. Float
    max_dets: int.
  """
  y1 = dets[:, 0]
  x1 = dets[:, 1]
  y2 = dets[:, 2]
  x2 = dets[:, 3]

  areas = (x2 - x1) * (y2 - y1)
  order = scores.argsort()[::-1]

  keep = []
  while order.size > 0 and len(keep) < max_dets:
    i = order[0]
    keep.append(i)

    xx1 = np.maximum(x1[i], x1[order[1:]])
```

```
    yy1 = np.maximum(y1[i], y1[order[1:]])
    xx2 = np.minimum(x2[i], x2[order[1:]])
    yy2 = np.minimum(y2[i], y2[order[1:]])

    w = np.maximum(0.0, xx2 - xx1)
    h = np.maximum(0.0, yy2 - yy1)
    intersection = w * h
    overlap = intersection / (areas[i] + areas[order[1:]] - intersection +␣
↪1e-12)

    inds = np.where(overlap <= thresh)[0]
    order = order[inds + 1]
  return keep
```

[22]:
```
#@title Visualization
import PIL.ImageColor as ImageColor
import PIL.ImageDraw as ImageDraw
import PIL.ImageFont as ImageFont

STANDARD_COLORS = [
    'AliceBlue', 'Chartreuse', 'Aqua', 'Aquamarine', 'Azure', 'Beige', 'Bisque',
    'BlanchedAlmond', 'BlueViolet', 'BurlyWood', 'CadetBlue', 'AntiqueWhite',
    'Chocolate', 'Coral', 'CornflowerBlue', 'Cornsilk', 'Cyan',
    'DarkCyan', 'DarkGoldenRod', 'DarkGrey', 'DarkKhaki', 'DarkOrange',
    'DarkOrchid', 'DarkSalmon', 'DarkSeaGreen', 'DarkTurquoise', 'DarkViolet',
    'DeepPink', 'DeepSkyBlue', 'DodgerBlue', 'FloralWhite',
    'ForestGreen', 'Fuchsia', 'Gainsboro', 'GhostWhite', 'Gold', 'GoldenRod',
    'Salmon', 'Tan', 'HoneyDew', 'HotPink', 'Ivory', 'Khaki',
    'Lavender', 'LavenderBlush', 'LawnGreen', 'LemonChiffon', 'LightBlue',
    'LightCoral', 'LightCyan', 'LightGoldenRodYellow', 'LightGray', 'LightGrey',
    'LightGreen', 'LightPink', 'LightSalmon', 'LightSeaGreen', 'LightSkyBlue',
    'LightSlateGray', 'LightSlateGrey', 'LightSteelBlue', 'LightYellow', 'Lime',
    'LimeGreen', 'Linen', 'Magenta', 'MediumAquaMarine', 'MediumOrchid',
    'MediumPurple', 'MediumSeaGreen', 'MediumSlateBlue', 'MediumSpringGreen',
    'MediumTurquoise', 'MediumVioletRed', 'MintCream', 'MistyRose', 'Moccasin',
    'NavajoWhite', 'OldLace', 'Olive', 'OliveDrab', 'Orange',
    'Orchid', 'PaleGoldenRod', 'PaleGreen', 'PaleTurquoise', 'PaleVioletRed',
    'PapayaWhip', 'PeachPuff', 'Peru', 'Pink', 'Plum', 'PowderBlue', 'Purple',
    'RosyBrown', 'RoyalBlue', 'SaddleBrown', 'Green', 'SandyBrown',
    'SeaGreen', 'SeaShell', 'Sienna', 'Silver', 'SkyBlue', 'SlateBlue',
    'SlateGray', 'SlateGrey', 'Snow', 'SpringGreen', 'SteelBlue', 'GreenYellow',
    'Teal', 'Thistle', 'Tomato', 'Turquoise', 'Violet', 'Wheat', 'White',
    'WhiteSmoke', 'Yellow', 'YellowGreen'
]

def draw_bounding_box_on_image(image,
                               ymin,
```

```
                               xmin,
                               ymax,
                               xmax,
                               color='red',
                               thickness=4,
                               display_str_list=(),
                               use_normalized_coordinates=True):
"""Adds a bounding box to an image.

Bounding box coordinates can be specified in either absolute (pixel) or
normalized coordinates by setting the use_normalized_coordinates argument.

Each string in display_str_list is displayed on a separate line above the
bounding box in black text on a rectangle filled with the input 'color'.
If the top of the bounding box extends to the edge of the image, the strings
are displayed below the bounding box.

Args:
  image: a PIL.Image object.
  ymin: ymin of bounding box.
  xmin: xmin of bounding box.
  ymax: ymax of bounding box.
  xmax: xmax of bounding box.
  color: color to draw bounding box. Default is red.
  thickness: line thickness. Default value is 4.
  display_str_list: list of strings to display in box
                    (each to be shown on its own line).
  use_normalized_coordinates: If True (default), treat coordinates
    ymin, xmin, ymax, xmax as relative to the image.  Otherwise treat
    coordinates as absolute.
"""
draw = ImageDraw.Draw(image)
im_width, im_height = image.size
if use_normalized_coordinates:
  (left, right, top, bottom) = (xmin * im_width, xmax * im_width,
                                ymin * im_height, ymax * im_height)
else:
  (left, right, top, bottom) = (xmin, xmax, ymin, ymax)
draw.line([(left, top), (left, bottom), (right, bottom),
           (right, top), (left, top)], width=thickness, fill=color)
try:
  font = ImageFont.truetype('arial.ttf', 24)
except IOError:
  font = ImageFont.load_default()

# If the total height of the display strings added to the top of the bounding
# box exceeds the top of the image, stack the strings below the bounding box
```

```
  # instead of above.
  display_str_heights = [font.getsize(ds)[1] for ds in display_str_list]
  # Each display_str has a top and bottom margin of 0.05x.
  total_display_str_height = (1 + 2 * 0.05) * sum(display_str_heights)

  if top > total_display_str_height:
    text_bottom = top
  else:
    text_bottom = bottom + total_display_str_height
  # Reverse list and print from bottom to top.
  for display_str in display_str_list[::-1]:
    text_left = min(5, left)
    text_width, text_height = font.getsize(display_str)
    margin = np.ceil(0.05 * text_height)
    draw.rectangle(
        [(left, text_bottom - text_height - 2 * margin), (left + text_width,
                                                          text_bottom)],
        fill=color)
    draw.text(
        (left + margin, text_bottom - text_height - margin),
        display_str,
        fill='black',
        font=font)
    text_bottom -= text_height - 2 * margin

def draw_bounding_box_on_image_array(image,
                                     ymin,
                                     xmin,
                                     ymax,
                                     xmax,
                                     color='red',
                                     thickness=4,
                                     display_str_list=(),
                                     use_normalized_coordinates=True):
  """Adds a bounding box to an image (numpy array).

  Bounding box coordinates can be specified in either absolute (pixel) or
  normalized coordinates by setting the use_normalized_coordinates argument.

  Args:
    image: a numpy array with shape [height, width, 3].
    ymin: ymin of bounding box.
    xmin: xmin of bounding box.
    ymax: ymax of bounding box.
    xmax: xmax of bounding box.
    color: color to draw bounding box. Default is red.
    thickness: line thickness. Default value is 4.
```

```
    display_str_list: list of strings to display in box
                      (each to be shown on its own line).
    use_normalized_coordinates: If True (default), treat coordinates
      ymin, xmin, ymax, xmax as relative to the image.  Otherwise treat
      coordinates as absolute.
  """
  image_pil = Image.fromarray(np.uint8(image)).convert('RGB')
  draw_bounding_box_on_image(image_pil, ymin, xmin, ymax, xmax, color,
                             thickness, display_str_list,
                             use_normalized_coordinates)
  np.copyto(image, np.array(image_pil))


def draw_mask_on_image_array(image, mask, color='red', alpha=0.4):
  """Draws mask on an image.

  Args:
    image: uint8 numpy array with shape (img_height, img_height, 3)
    mask: a uint8 numpy array of shape (img_height, img_height) with
      values between either 0 or 1.
    color: color to draw the keypoints with. Default is red.
    alpha: transparency value between 0 and 1. (default: 0.4)

  Raises:
    ValueError: On incorrect data type for image or masks.
  """
  if image.dtype != np.uint8:
    raise ValueError('`image` not of type np.uint8')
  if mask.dtype != np.uint8:
    raise ValueError('`mask` not of type np.uint8')
  if np.any(np.logical_and(mask != 1, mask != 0)):
    raise ValueError('`mask` elements should be in [0, 1]')
  if image.shape[:2] != mask.shape:
    raise ValueError('The image has spatial dimensions %s but the mask has '
                     'dimensions %s' % (image.shape[:2], mask.shape))
  rgb = ImageColor.getrgb(color)
  pil_image = Image.fromarray(image)

  solid_color = np.expand_dims(
      np.ones_like(mask), axis=2) * np.reshape(list(rgb), [1, 1, 3])
  pil_solid_color = Image.fromarray(np.uint8(solid_color)).convert('RGBA')
  pil_mask = Image.fromarray(np.uint8(255.0*alpha*mask)).convert('L')
  pil_image = Image.composite(pil_solid_color, pil_image, pil_mask)
  np.copyto(image, np.array(pil_image.convert('RGB')))

def visualize_boxes_and_labels_on_image_array(
    image,
```

```
    boxes,
    classes,
    scores,
    category_index,
    instance_masks=None,
    instance_boundaries=None,
    use_normalized_coordinates=False,
    max_boxes_to_draw=20,
    min_score_thresh=.5,
    agnostic_mode=False,
    line_thickness=4,
    groundtruth_box_visualization_color='black',
    skip_scores=False,
    skip_labels=False,
    mask_alpha=0.4,
    plot_color=None,
):
  """Overlay labeled boxes on an image with formatted scores and label names.

  This function groups boxes that correspond to the same location
  and creates a display string for each detection and overlays these
  on the image. Note that this function modifies the image in place, and returns
  that same image.

  Args:
    image: uint8 numpy array with shape (img_height, img_width, 3)
    boxes: a numpy array of shape [N, 4]
    classes: a numpy array of shape [N]. Note that class indices are 1-based,
      and match the keys in the label map.
    scores: a numpy array of shape [N] or None.  If scores=None, then
      this function assumes that the boxes to be plotted are groundtruth
      boxes and plot all boxes as black with no classes or scores.
    category_index: a dict containing category dictionaries (each holding
      category index `id` and category name `name`) keyed by category indices.
    instance_masks: a numpy array of shape [N, image_height, image_width] with
      values ranging between 0 and 1, can be None.
    instance_boundaries: a numpy array of shape [N, image_height, image_width]
      with values ranging between 0 and 1, can be None.
    use_normalized_coordinates: whether boxes is to be interpreted as
      normalized coordinates or not.
    max_boxes_to_draw: maximum number of boxes to visualize.  If None, draw
      all boxes.
    min_score_thresh: minimum score threshold for a box to be visualized
    agnostic_mode: boolean (default: False) controlling whether to evaluate in
      class-agnostic mode or not.  This mode will display scores but ignore
      classes.
    line_thickness: integer (default: 4) controlling line width of the boxes.
```

```
  groundtruth_box_visualization_color: box color for visualizing groundtruth
    boxes
  skip_scores: whether to skip score when drawing a single detection
  skip_labels: whether to skip label when drawing a single detection

Returns:
  uint8 numpy array with shape (img_height, img_width, 3) with overlaid boxes.
"""
# Create a display string (and color) for every box location, group any boxes
# that correspond to the same location.
box_to_display_str_map = collections.defaultdict(list)
box_to_color_map = collections.defaultdict(str)
box_to_instance_masks_map = {}
box_to_score_map = {}
box_to_instance_boundaries_map = {}

if not max_boxes_to_draw:
  max_boxes_to_draw = boxes.shape[0]
for i in range(min(max_boxes_to_draw, boxes.shape[0])):
  if scores is None or scores[i] > min_score_thresh:
    box = tuple(boxes[i].tolist())
    if instance_masks is not None:
      box_to_instance_masks_map[box] = instance_masks[i]
    if instance_boundaries is not None:
      box_to_instance_boundaries_map[box] = instance_boundaries[i]
    if scores is None:
      box_to_color_map[box] = groundtruth_box_visualization_color
    else:
      display_str = ''
      if not skip_labels:
        if not agnostic_mode:
          if classes[i] in list(category_index.keys()):
            class_name = category_index[classes[i]]['name']
          else:
            class_name = 'N/A'
          display_str = str(class_name)
      if not skip_scores:
        if not display_str:
          display_str = '{}%'.format(int(100*scores[i]))
        else:
          float_score = ("%.2f" % scores[i]).lstrip('0')
          display_str = '{}: {}'.format(display_str, float_score)
        box_to_score_map[box] = int(100*scores[i])

      box_to_display_str_map[box].append(display_str)
      if plot_color is not None:
        box_to_color_map[box] = plot_color
```

```
        elif agnostic_mode:
          box_to_color_map[box] = 'DarkOrange'
        else:
          box_to_color_map[box] = STANDARD_COLORS[
              classes[i] % len(STANDARD_COLORS)]

  # Handle the case when box_to_score_map is empty.
  if box_to_score_map:
    box_color_iter = sorted(
        box_to_color_map.items(), key=lambda kv: box_to_score_map[kv[0]])
  else:
    box_color_iter = box_to_color_map.items()

  # Draw all boxes onto image.
  for box, color in box_color_iter:
    ymin, xmin, ymax, xmax = box
    if instance_masks is not None:
      draw_mask_on_image_array(
          image,
          box_to_instance_masks_map[box],
          color=color,
          alpha=mask_alpha
      )
    if instance_boundaries is not None:
      draw_mask_on_image_array(
          image,
          box_to_instance_boundaries_map[box],
          color='red',
          alpha=1.0
      )
    draw_bounding_box_on_image_array(
        image,
        ymin,
        xmin,
        ymax,
        xmax,
        color=color,
        thickness=line_thickness,
        display_str_list=box_to_display_str_map[box],
        use_normalized_coordinates=use_normalized_coordinates)

  return image


def paste_instance_masks(masks,
                         detected_boxes,
                         image_height,
```

```
                          image_width):
  """Paste instance masks to generate the image segmentation results.

  Args:
    masks: a numpy array of shape [N, mask_height, mask_width] representing the
      instance masks w.r.t. the `detected_boxes`.
    detected_boxes: a numpy array of shape [N, 4] representing the reference
      bounding boxes.
    image_height: an integer representing the height of the image.
    image_width: an integer representing the width of the image.

  Returns:
    segms: a numpy array of shape [N, image_height, image_width] representing
      the instance masks *pasted* on the image canvas.
  """

  def expand_boxes(boxes, scale):
    """Expands an array of boxes by a given scale."""
    # Reference: https://github.com/facebookresearch/Detectron/blob/master/
↪detectron/utils/boxes.py#L227  # pylint: disable=line-too-long
    # The `boxes` in the reference implementation is in [x1, y1, x2, y2] form,
    # whereas `boxes` here is in [x1, y1, w, h] form
    w_half = boxes[:, 2] * .5
    h_half = boxes[:, 3] * .5
    x_c = boxes[:, 0] + w_half
    y_c = boxes[:, 1] + h_half

    w_half *= scale
    h_half *= scale

    boxes_exp = np.zeros(boxes.shape)
    boxes_exp[:, 0] = x_c - w_half
    boxes_exp[:, 2] = x_c + w_half
    boxes_exp[:, 1] = y_c - h_half
    boxes_exp[:, 3] = y_c + h_half

    return boxes_exp

  # Reference: https://github.com/facebookresearch/Detectron/blob/master/
↪detectron/core/test.py#L812  # pylint: disable=line-too-long
  # To work around an issue with cv2.resize (it seems to automatically pad
  # with repeated border values), we manually zero-pad the masks by 1 pixel
  # prior to resizing back to the original image resolution. This prevents
  # "top hat" artifacts. We therefore need to expand the reference boxes by an
  # appropriate factor.
  _, mask_height, mask_width = masks.shape
  scale = max((mask_width + 2.0) / mask_width,
```

```
              (mask_height + 2.0) / mask_height)

  ref_boxes = expand_boxes(detected_boxes, scale)
  ref_boxes = ref_boxes.astype(np.int32)
  padded_mask = np.zeros((mask_height + 2, mask_width + 2), dtype=np.float32)
  segms = []
  for mask_ind, mask in enumerate(masks):
    im_mask = np.zeros((image_height, image_width), dtype=np.uint8)
    # Process mask inside bounding boxes.
    padded_mask[1:-1, 1:-1] = mask[:, :]

    ref_box = ref_boxes[mask_ind, :]
    w = ref_box[2] - ref_box[0] + 1
    h = ref_box[3] - ref_box[1] + 1
    w = np.maximum(w, 1)
    h = np.maximum(h, 1)

    mask = cv2.resize(padded_mask, (w, h))
    mask = np.array(mask > 0.5, dtype=np.uint8)

    x_0 = min(max(ref_box[0], 0), image_width)
    x_1 = min(max(ref_box[2] + 1, 0), image_width)
    y_0 = min(max(ref_box[1], 0), image_height)
    y_1 = min(max(ref_box[3] + 1, 0), image_height)

    im_mask[y_0:y_1, x_0:x_1] = mask[
        (y_0 - ref_box[1]):(y_1 - ref_box[1]),
        (x_0 - ref_box[0]):(x_1 - ref_box[0])
    ]
    segms.append(im_mask)

  segms = np.array(segms)
  assert masks.shape[0] == segms.shape[0]
  return segms
```

[23]:
```
#@title Plot instance masks
def plot_mask(color, alpha, original_image, mask):
  rgb = ImageColor.getrgb(color)
  pil_image = Image.fromarray(original_image)

  solid_color = np.expand_dims(
      np.ones_like(mask), axis=2) * np.reshape(list(rgb), [1, 1, 3])
  pil_solid_color = Image.fromarray(np.uint8(solid_color)).convert('RGBA')
  pil_mask = Image.fromarray(np.uint8(255.0*alpha*mask)).convert('L')
  pil_image = Image.composite(pil_solid_color, pil_image, pil_mask)
  img_w_mask = np.array(pil_image.convert('RGB'))
  return img_w_mask
```

3.4 Main functions

[24]:
```
%matplotlib inline
def display_image(path_or_array, size=(10, 10)):
  if isinstance(path_or_array, str):
    image = np.asarray(Image.open(open(image_path, 'rb')).convert("RGB"))
  else:
    image = path_or_array

  plt.figure(figsize=size)
  plt.imshow(image)
  plt.axis('off')
  plt.show()
```

[25]:
```
def main(image_path, category_name_string, params):
  ######################################################################
  # Preprocessing categories and get params
  category_names = [x.strip() for x in category_name_string.split(';')]
  category_names = ['background'] + category_names
  categories = [{'name': item, 'id': idx+1,} for idx, item in␣
 ↪enumerate(category_names)]
  category_indices = {cat['id']: cat for cat in categories}

  max_boxes_to_draw, nms_threshold, min_rpn_score_thresh, min_box_area = params
  fig_size_h = min(max(5, int(len(category_names) / 2.5) ), 10)


  ######################################################################
  # Obtain results and read image
  roi_boxes, roi_scores, detection_boxes, scores_unused, box_outputs,␣
 ↪detection_masks, visual_features, image_info = session.run(
        ['RoiBoxes:0', 'RoiScores:0', '2ndStageBoxes:0', '2ndStageScoresUnused:
 ↪0', 'BoxOutputs:0', 'MaskOutputs:0', 'VisualFeatOutputs:0', 'ImageInfo:0'],
        feed_dict={'Placeholder:0': [image_path,]})

  roi_boxes = np.squeeze(roi_boxes, axis=0)  # squeeze
  # no need to clip the boxes, already done
  roi_scores = np.squeeze(roi_scores, axis=0)

  detection_boxes = np.squeeze(detection_boxes, axis=(0, 2))
  scores_unused = np.squeeze(scores_unused, axis=0)
  box_outputs = np.squeeze(box_outputs, axis=0)
  detection_masks = np.squeeze(detection_masks, axis=0)
  visual_features = np.squeeze(visual_features, axis=0)

  image_info = np.squeeze(image_info, axis=0)  # obtain image info
  image_scale = np.tile(image_info[2:3, :], (1, 2))
```

```
image_height = int(image_info[0, 0])
image_width = int(image_info[0, 1])

rescaled_detection_boxes = detection_boxes / image_scale # rescale

# Read image
image = np.asarray(Image.open(open(image_path, 'rb')).convert("RGB"))
assert image_height == image.shape[0]
assert image_width == image.shape[1]


##################################################################
# Filter boxes

# Apply non-maximum suppression to detected boxes with nms threshold.
nmsed_indices = nms(
    detection_boxes,
    roi_scores,
    thresh=nms_threshold
    )

# Compute RPN box size.
box_sizes = (rescaled_detection_boxes[:, 2] - rescaled_detection_boxes[:, 0])␣
↪* (rescaled_detection_boxes[:, 3] - rescaled_detection_boxes[:, 1])

# Filter out invalid rois (nmsed rois)
valid_indices = np.where(
    np.logical_and(
      np.isin(np.arange(len(roi_scores), dtype=np.int), nmsed_indices),
      np.logical_and(
          np.logical_not(np.all(roi_boxes == 0., axis=-1)),
          np.logical_and(
            roi_scores >= min_rpn_score_thresh,
            box_sizes > min_box_area
            )
      )
    )
)[0]
print('number of valid indices', len(valid_indices))

detection_roi_scores = roi_scores[valid_indices][:max_boxes_to_draw, ...]
detection_boxes = detection_boxes[valid_indices][:max_boxes_to_draw, ...]
detection_masks = detection_masks[valid_indices][:max_boxes_to_draw, ...]
detection_visual_feat = visual_features[valid_indices][:max_boxes_to_draw, ...
↪]
rescaled_detection_boxes = rescaled_detection_boxes[valid_indices][:
↪max_boxes_to_draw, ...]
```

```
  ######################################################################
  # Compute text embeddings and detection scores, and rank results
  text_features = build_text_embedding(categories)

  raw_scores = detection_visual_feat.dot(text_features.T)
  if FLAGS.use_softmax:
    scores_all = softmax(FLAGS.temperature * raw_scores, axis=-1)
  else:
    scores_all = raw_scores

  indices = np.argsort(-np.max(scores_all, axis=1))  # Results are ranked by␣
↪scores
  indices_fg = np.array([i for i in indices if np.argmax(scores_all[i]) != 0])

  ######################################################################
  # Plot detected boxes on the input image.
  ymin, xmin, ymax, xmax = np.split(rescaled_detection_boxes, 4, axis=-1)
  processed_boxes = np.concatenate([xmin, ymin, xmax - xmin, ymax - ymin],␣
↪axis=-1)
  segmentations = paste_instance_masks(detection_masks, processed_boxes,␣
↪image_height, image_width)

  if len(indices_fg) == 0:
    display_image(np.array(image), size=overall_fig_size)
    print('ViLD does not detect anything belong to the given category')

  else:
    image_with_detections = visualize_boxes_and_labels_on_image_array(
        np.array(image),
        rescaled_detection_boxes[indices_fg],
        valid_indices[:max_boxes_to_draw][indices_fg],
        detection_roi_scores[indices_fg],
        numbered_category_indices,
        instance_masks=segmentations[indices_fg],
        use_normalized_coordinates=False,
        max_boxes_to_draw=max_boxes_to_draw,
        min_score_thresh=min_rpn_score_thresh,
        skip_scores=False,
        skip_labels=True)

    plt.figure(figsize=overall_fig_size)
    plt.imshow(image_with_detections)
    plt.axis('off')
    plt.title('Detected objects and RPN scores')
    plt.show()
```

```
 ##################################################################
 # Plot
 cnt = 0
 raw_image = np.array(image)
 n_boxes = rescaled_detection_boxes.shape[0]

 for anno_idx in indices[0:int(n_boxes)]:
   rpn_score = detection_roi_scores[anno_idx]
   bbox = rescaled_detection_boxes[anno_idx]
   scores = scores_all[anno_idx]
   if np.argmax(scores) == 0:
     continue

   y1, x1, y2, x2 = int(np.floor(bbox[0])), int(np.floor(bbox[1])), int(np.
↪ceil(bbox[2])), int(np.ceil(bbox[3]))
   img_w_mask = plot_mask(mask_color, alpha, raw_image,␣
↪segmentations[anno_idx])
   crop_w_mask = img_w_mask[y1:y2, x1:x2, :]


   fig, axs = plt.subplots(1, 4, figsize=(fig_size_w, fig_size_h),␣
↪gridspec_kw={'width_ratios': [3, 1, 1, 2]}, constrained_layout=True)

   # Draw bounding box.
   rect = patches.Rectangle((x1, y1), x2-x1, y2-y1, linewidth=line_thickness,␣
↪edgecolor='r', facecolor='none')
   axs[0].add_patch(rect)

   axs[0].set_xticks([])
   axs[0].set_yticks([])
   axs[0].set_title(f'bbox: {y1, x1, y2, x2} area: {(y2 - y1) * (x2 - x1)} rpn␣
↪score: {rpn_score:.4f}')
   axs[0].imshow(raw_image)

   # Draw image in a cropped region.
   crop = np.copy(raw_image[y1:y2, x1:x2, :])
   axs[1].set_xticks([])
   axs[1].set_yticks([])

   axs[1].set_title(f'predicted: {category_names[np.argmax(scores)]}')
   axs[1].imshow(crop)

   # Draw segmentation inside a cropped region.
   axs[2].set_xticks([])
   axs[2].set_yticks([])
```

```
    axs[2].set_title('mask')
    axs[2].imshow(crop_w_mask)

    # Draw category scores.
    fontsize = max(min(fig_size_h / float(len(category_names)) * 45, 20), 8)
    for cat_idx in range(len(category_names)):
      axs[3].barh(cat_idx, scores[cat_idx],
                  color='orange' if scores[cat_idx] == max(scores) else 'blue')
    axs[3].invert_yaxis()
    axs[3].set_axisbelow(True)
    axs[3].set_xlim(0, 1)
    plt.xlabel("confidence score")
    axs[3].set_yticks(range(len(category_names)))
    axs[3].set_yticklabels(category_names, fontdict={
        'fontsize': fontsize})

    cnt += 1
    # fig.tight_layout()


  print('Detection counts:', cnt)
```

4 Start playing

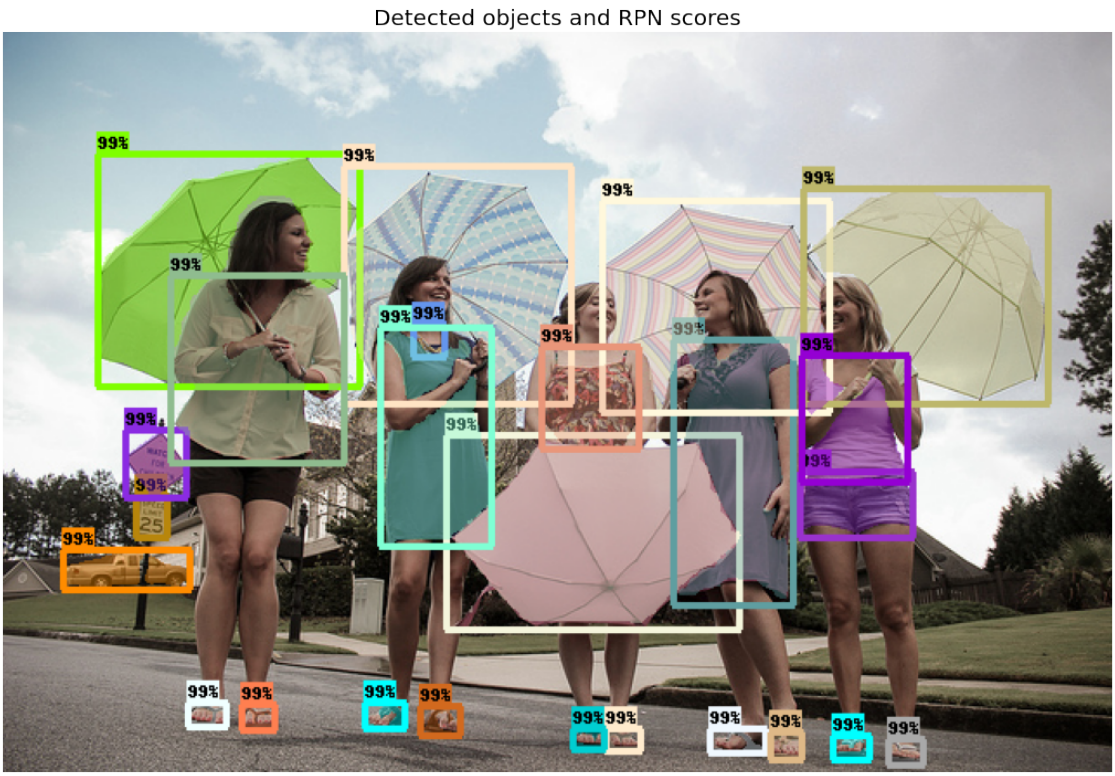
4.1 Five women and umbrellas

[26]:
```
image_path = './examples/five_women_and_umbrellas.jpg'  #@param {type:"string"}
display_image(image_path, size=display_input_size)
```

[27]:
```
category_name_string = ';'.join(['flipflop', 'street sign', 'bracelet',
      'necklace', 'shorts', 'floral camisole', 'orange shirt',
      'purple dress', 'yellow tee', 'green umbrella', 'pink striped umbrella',
      'transparent umbrella', 'plain pink umbrella', 'blue patterned umbrella',
      'koala', 'electric box','car', 'pole'])
max_boxes_to_draw = 25 #@param {type:"integer"}

nms_threshold = 0.6 #@param {type:"slider", min:0, max:0.9, step:0.05}
min_rpn_score_thresh = 0.9  #@param {type:"slider", min:0, max:1, step:0.01}
min_box_area = 220 #@param {type:"slider", min:0, max:10000, step:1.0}


params = max_boxes_to_draw, nms_threshold, min_rpn_score_thresh, min_box_area
main(image_path, category_name_string, params)

#@markdown Orange bar means the prediction with maximum score over text inputs.
```

```
/var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/ipykernel_57573/3119204168.py:5
8: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To
silence this warning, use `int` by itself. Doing this will not modify any
behavior and is safe. When replacing `np.int`, you may wish to use e.g.
`np.int64` or `np.int32` to specify the precision. If you wish to review your
current use, check the release note link for additional information.
```

```
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  np.isin(np.arange(len(roi_scores), dtype=np.int), nmsed_indices),

number of valid indices 34
Building text embeddings…

100%|      | 19/19 [01:36<00:00,  5.09s/it]
```




```
/var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/ipykernel_57573/3119204168.py:1
39: RuntimeWarning: More than 20 figures have been opened. Figures created
through the pyplot interface (`matplotlib.pyplot.figure`) are retained until
explicitly closed and may consume too much memory. (To control this warning, see
the rcParam `figure.max_open_warning`).
  fig, axs = plt.subplots(1, 4, figsize=(fig_size_w, fig_size_h),
gridspec_kw={'width_ratios': [3, 1, 1, 2]}, constrained_layout=True)

Detection counts: 25
```

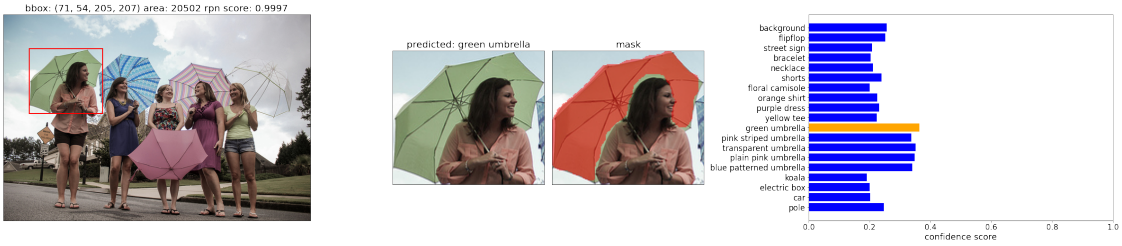

bbox: (268, 76, 292, 96) area: 480 rpn score: 0.9978
predicted: street sign
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (233, 255, 345, 426) area: 19152 rpn score: 0.9994
predicted: plain pink umbrella
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (169, 236, 188, 256) area: 380 rpn score: 0.9988
predicted: necklace
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (230, 70, 269, 107) area: 1443 rpn score: 0.9992
predicted: street sign
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (178, 386, 331, 457) area: 10863 rpn score: 0.9992
predicted: purple dress
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (71, 54, 205, 207) area: 20502 rpn score: 0.9997
predicted: green umbrella
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

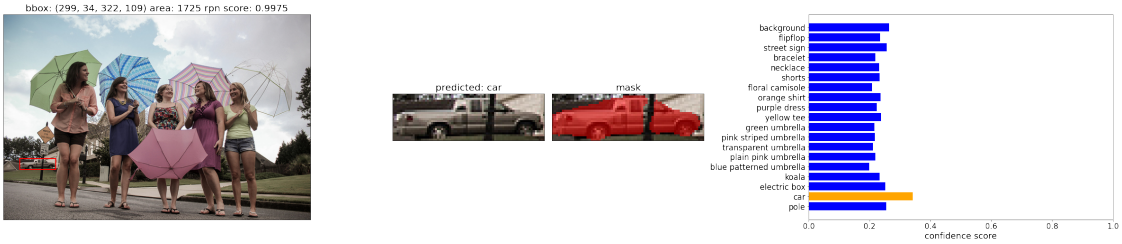

bbox: (78, 196, 215, 329) area: 18221 rpn score: 0.9994
predicted: blue patterned umbrella
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (171, 217, 297, 283) area: 8316 rpn score: 0.9997
predicted: purple dress
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

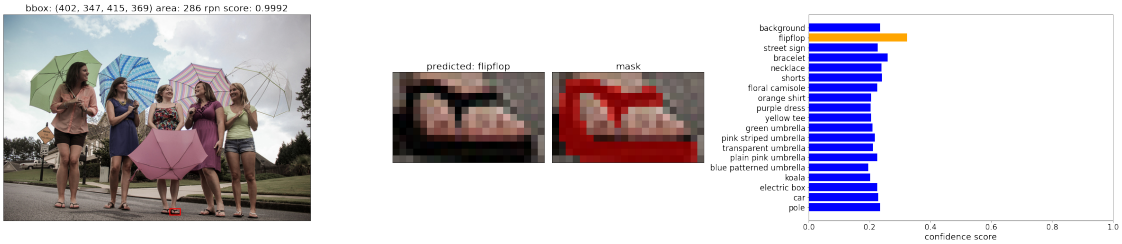

bbox: (254, 459, 292, 526) area: 2546 rpn score: 0.9966
predicted: shorts
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

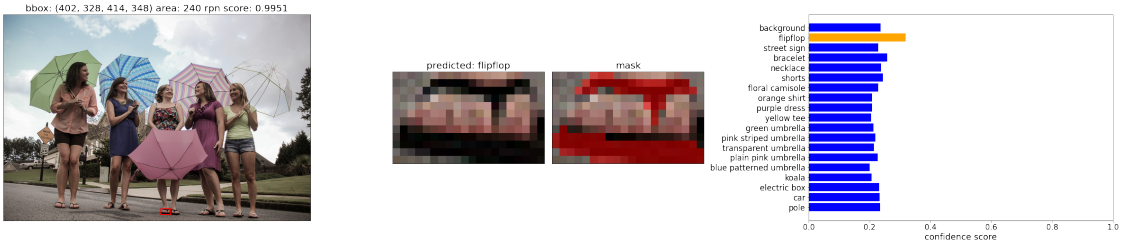

bbox: (98, 345, 220, 478) area: 16226 rpn score: 0.9986
predicted: pink striped umbrella
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (91, 461, 215, 604) area: 17732 rpn score: 0.9975
predicted: transparent umbrella
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

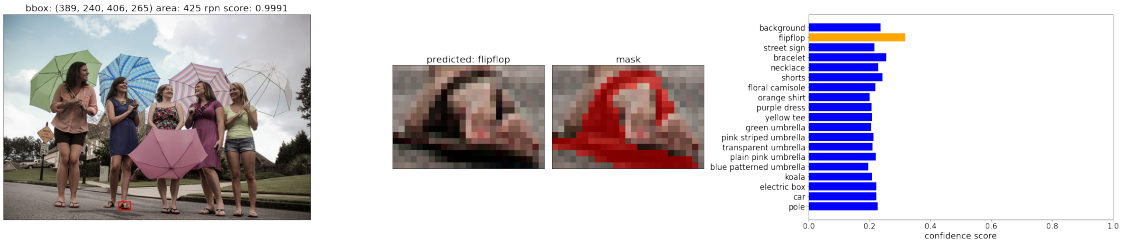

bbox: (299, 34, 322, 109) area: 1725 rpn score: 0.9975
predicted: car
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

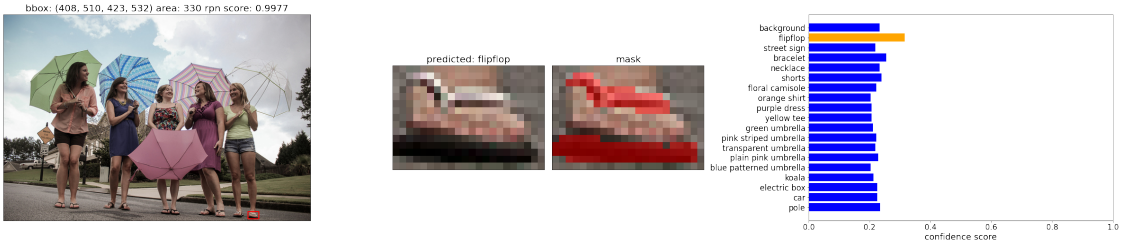

bbox: (182, 310, 241, 368) area: 3422 rpn score: 0.9965
predicted: floral camisole
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

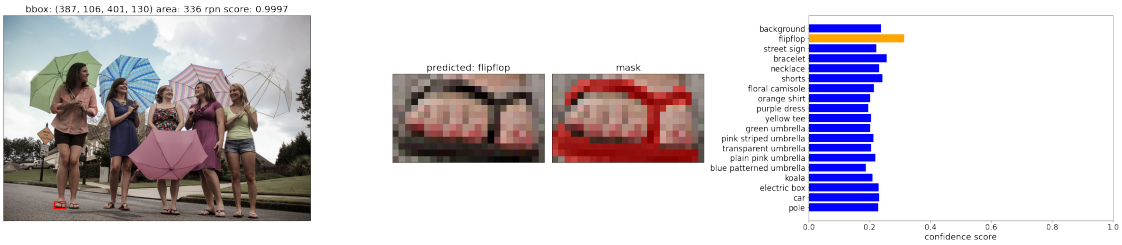

bbox: (402, 347, 415, 369) area: 286 rpn score: 0.9992
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

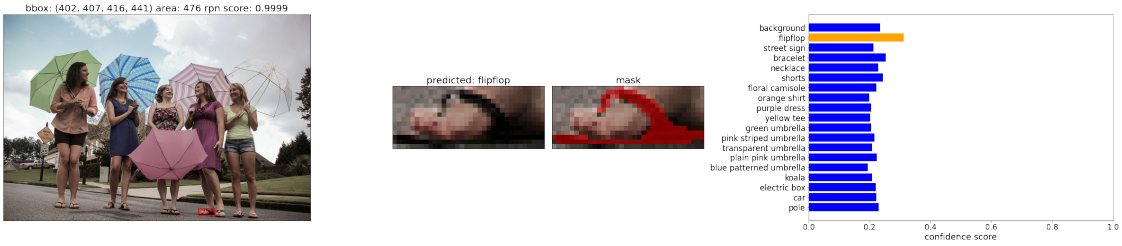

bbox: (402, 328, 414, 348) area: 240 rpn score: 0.9951
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (406, 478, 420, 501) area: 322 rpn score: 0.9985
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

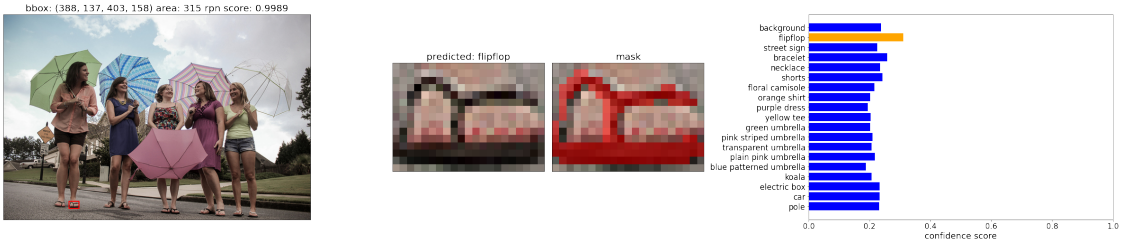

bbox: (389, 240, 406, 265) area: 425 rpn score: 0.9991
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (408, 510, 423, 532) area: 330 rpn score: 0.9977
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

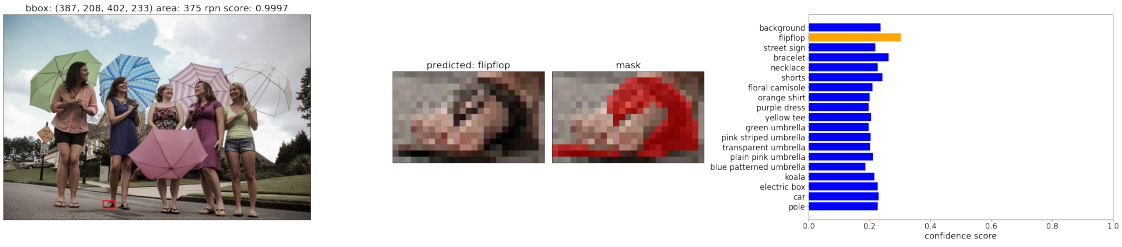

bbox: (387, 106, 401, 130) area: 336 rpn score: 0.9997
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

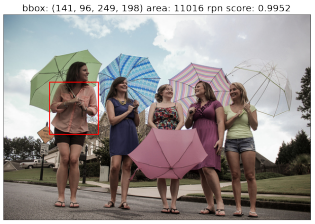

bbox: (402, 407, 416, 441) area: 476 rpn score: 0.9999
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score

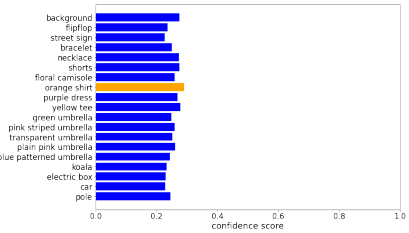

bbox: (187, 460, 260, 523) area: 4599 rpn score: 0.9951
predicted: yellow tee
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (388, 137, 403, 158) area: 315 rpn score: 0.9989
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (404, 442, 420, 462) area: 320 rpn score: 0.9992
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score


bbox: (387, 208, 402, 233) area: 375 rpn score: 0.9997
predicted: flipflop
mask
background
flipflop
street sign
bracelet
necklace
shorts
floral camisole
orange shirt
purple dress
yellow tee
green umbrella
pink striped umbrella
transparent umbrella
plain pink umbrella
blue patterned umbrella
koala
electric box
car
pole
0.0
0.2
0.4
0.6
0.8
1.0
confidence score





4.2 ViLD is able to detect nothing when there is no matched category.

[28]:
```
#@title Same image, detect koalas
category_name_string = ';'.join(['koala'])
max_boxes_to_draw = 25 #@param {type:"integer"}

nms_threshold = 0.6 #@param {type:"slider", min:0, max:0.9, step:0.05}
min_rpn_score_thresh = 0.9  #@param {type:"slider", min:0, max:1, step:0.01}
min_box_area = 220 #@param {type:"slider", min:0, max:10000, step:1.0}


params = max_boxes_to_draw, nms_threshold, min_rpn_score_thresh, min_box_area
main(image_path, category_name_string, params)
```

```
/var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/ipykernel_57573/3119204168.py:5
8: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To
silence this warning, use `int` by itself. Doing this will not modify any
behavior and is safe. When replacing `np.int`, you may wish to use e.g.
`np.int64` or `np.int32` to specify the precision. If you wish to review your
current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  np.isin(np.arange(len(roi_scores), dtype=np.int), nmsed_indices),

number of valid indices 34
Building text embeddings…

100%|      | 2/2 [00:09<00:00,  4.90s/it]
```


```
ViLD does not detect anything belong to the given category
Detection counts: 0
```

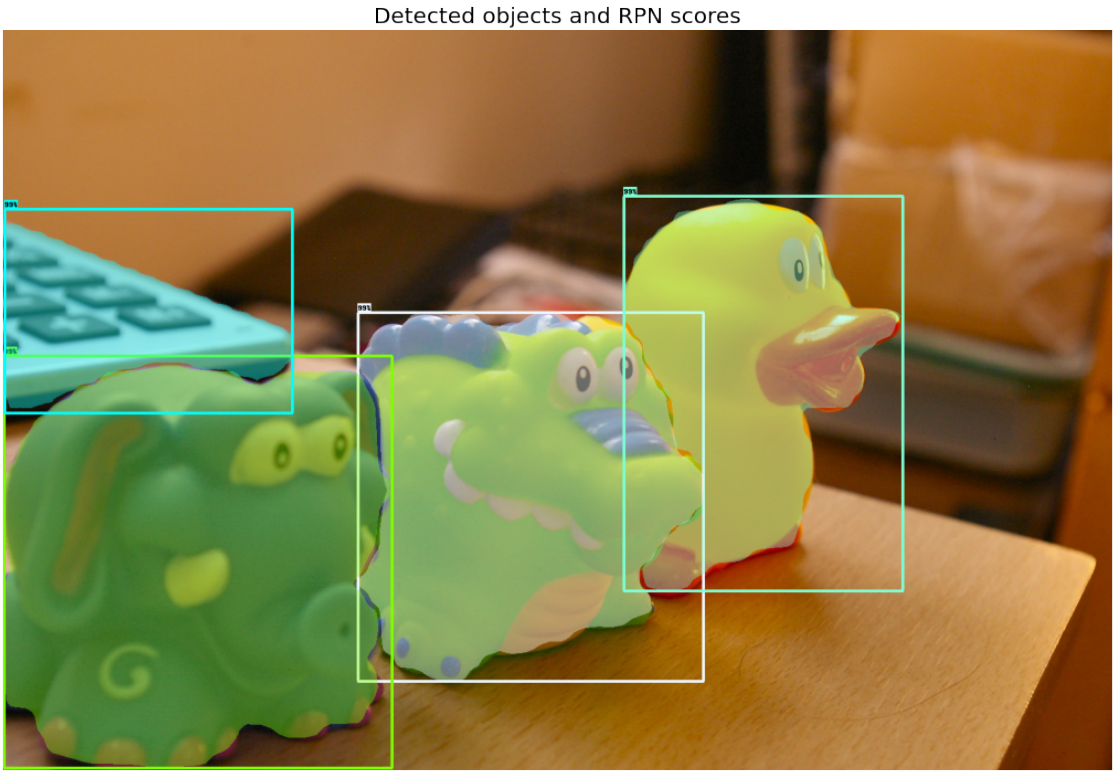
4.3 Three toys

[29]:
```
image_path = './examples/three_toys.jpg'  #@param {type:"string"}
display_image(image_path, size=display_input_size)
```

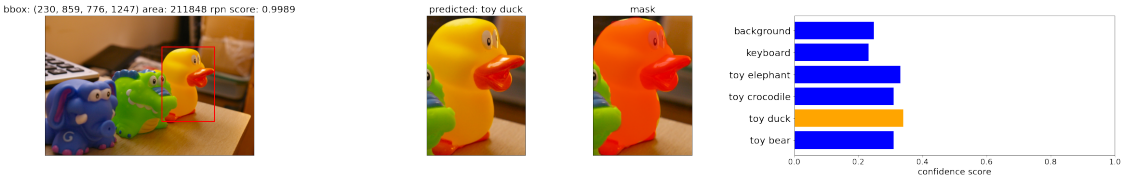
[31]:
```
category_name_string = ';'.join(['keyboard', 'toy elephant', 'toy crocodile',␣
 ↪'toy duck', 'toy bear'])
max_boxes_to_draw = 4 #@param {type:"integer"}

nms_threshold = 0.6 #@param {type:"slider", min:0, max:0.9, step:0.05}
min_rpn_score_thresh = 0.9  #@param {type:"slider", min:0, max:1, step:0.01}
min_box_area = 220 #@param {type:"slider", min:0, max:10000, step:1.0}


params = max_boxes_to_draw, nms_threshold, min_rpn_score_thresh, min_box_area
main(image_path, category_name_string, params)
```

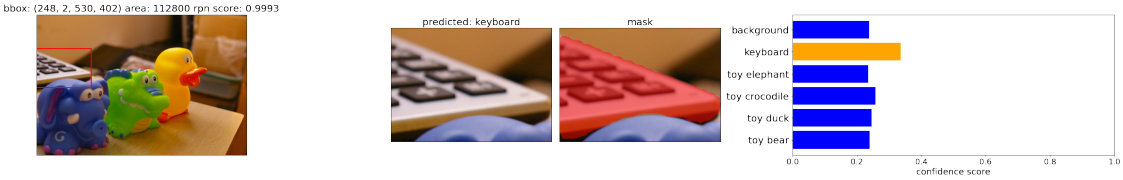
```
/var/folders/fc/n5827zt91vb65jnqcsgpv5cr0000gn/T/ipykernel_57573/3119204168.py:5
8: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To
silence this warning, use `int` by itself. Doing this will not modify any
behavior and is safe. When replacing `np.int`, you may wish to use e.g.
`np.int64` or `np.int32` to specify the precision. If you wish to review your
current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  np.isin(np.arange(len(roi_scores), dtype=np.int), nmsed_indices),

number of valid indices 15
Building text embeddings…
```

```
100%|          | 6/6 [00:30<00:00,  5.05s/it]
```




```
Detection counts: 4
```





5 Try your own image by replacing the `image_path` field!