In [3]:

```
import torch
from torchvision.models.detection import fasterrcnn_resnet50_fpn
from torchvision.transforms import functional as F
import matplotlib.pyplot as plt
import cv2

# Define the classes to be detected
CLASSES = [
    'Shoe', 'Sneaker', 'Bottle', 'Cup', 'Sandal', 'Perfume', 'Toy', 'Su
    'Chair', 'Office Chair', 'Can', 'Cap', 'Hat', 'Couch', 'Wristwatch'
    'Baggage', 'Suitcase', 'Headphones', 'Jar', 'Vase'
]

# Load the pre-trained YOLOv5 model
model = fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

# Function to perform object detection on an image
def detect_objects(image_path):
    # Load image
    img = cv2.imread(image_path)
    img_tensor = F.to_tensor(img).unsqueeze(0)

    # Perform object detection
    with torch.no_grad():
        predictions = model(img_tensor)

    # Display the image with bounding boxes
    plt.figure(figsize=(10, 6))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

    for box, label in zip(predictions[0]['boxes'], predictions[0]['labe
        box = box.tolist()
        label = int(label)
        class_name = CLASSES[label - 1]  # Adjusting index because YOLO

        # Draw bounding box
        cv2.rectangle(img, (int(box[0]), int(box[1])), (int(box[2]), in

        # Display class name
        cv2.putText(img, class_name, (int(box[0]), int(box[1] - 5)), cv

    plt.axis('off')
    plt.show()


```

In [4]:

```
1 image_path = './shoe.jpg'
2 detect_objects(image_path)
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[4], line 2
      1 image_path = './shoe.jpg'
----> 2 detect_objects(image_path)

Cell In[3], line 26, in detect_objects(image_path)
     24 # Perform object detection
     25 with torch.no_grad():
---> 26     predictions = model(img_tensor)
     28 # Display the image with bounding boxes
     29 plt.figure(figsize=(10, 6))

File ~\anaconda3\envs\dsml\lib\site-packages\torch\nn\modules\module.py:1130, in Module._call_impl(self, *input, **kwargs)
   1126 # If we don't have any hooks, we want to skip the rest of the logic in
   1127 # this function, and just call forward.
   1128 if not (self._backward_hooks or self._forward_hooks or self._forward_pre_hooks or _global_backward_hooks
   1129         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1130     return forward_call(*input, **kwargs)
   1131 # Do not call functions when jit is used
   1132 full_backward_hooks, non_full_backward_hooks = [], []

File ~\anaconda3\envs\dsml\lib\site-packages\torchvision\models\detection\generalized_rcnn.py:104, in GeneralizedRCNN.forward(self, images, targets)
    102 if isinstance(features, torch.Tensor):
    103     features = OrderedDict([("0", features)])
--> 104 proposals, proposal_losses = self.rpn(images, features, targets)
    105 detections, detector_losses = self.roi_heads(features, proposals, images.image_sizes, targets)
    106 detections = self.transform.postprocess(detections, images.image_sizes, original_image_sizes)  # type: ignore[operator]

File ~\anaconda3\envs\dsml\lib\site-packages\torch\nn\modules\module.py:1130, in Module._call_impl(self, *input, **kwargs)
   1126 # If we don't have any hooks, we want to skip the rest of the logic in
   1127 # this function, and just call forward.
   1128 if not (self._backward_hooks or self._forward_hooks or self._forward_pre_hooks or _global_backward_hooks
   1129         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1130     return forward_call(*input, **kwargs)
   1131 # Do not call functions when jit is used
   1132 full_backward_hooks, non_full_backward_hooks = [], []

File ~\anaconda3\envs\dsml\lib\site-packages\torchvision\models\detection\rpn.py:372, in RegionProposalNetwork.forward(self, images, features, targets)
    370 proposals = self.box_coder.decode(pred_bbox_deltas.detach(), anchors)
    371 proposals = proposals.view(num_images, -1, 4)
--> 372 boxes, scores = self.filter_proposals(proposals, objectness, images.image_sizes, num_anchors_per_level)
    374 losses = {}
    375 if self.training:
```

```
File ~\anaconda3\envs\dsml\lib\site-packages\torchvision\models\detection
\rpn.py:288, in RegionProposalNetwork.filter_proposals(self, proposals, ob
jectness, image_shapes, num_anchors_per_level)
    285 boxes, scores, lvl = boxes[keep], scores[keep], lvl[keep]
    287 # non-maximum suppression, independently done per level
--> 288 keep = box_ops.batched_nms(boxes, scores, lvl, self.nms_thresh)
    290 # keep only topk scoring predictions
    291 keep = keep[: self.post_nms_top_n()]

File ~\anaconda3\envs\dsml\lib\site-packages\torchvision\ops\boxes.py:73,
in batched_nms(boxes, scores, idxs, iou_threshold)
     70 # Benchmarks that drove the following thresholds are at
     71 # https://github.com/pytorch/vision/issues/1311#issuecomment-78132
9339 (https://github.com/pytorch/vision/issues/1311#issuecomment-78132933
9)
     72 if boxes.numel() > (4000 if boxes.device.type == "cpu" else 20000)
and not torchvision._is_tracing():
---> 73     return _batched_nms_vanilla(boxes, scores, idxs, iou_threshol
d)
     74 else:
     75     return _batched_nms_coordinate_trick(boxes, scores, idxs, iou_
threshold)

File ~\anaconda3\envs\dsml\lib\site-packages\torch\jit\_trace.py:1127, in
_script_if_tracing.<locals>.wrapper(*args, **kwargs)
   1123 @functools.wraps(fn)
   1124 def wrapper(*args, **kwargs):
   1125     if not is_tracing():
   1126         # Not tracing, don't do anything
-> 1127         return fn(*args, **kwargs)
   1129     compiled_fn = script(wrapper.__original_fn)  # type: ignore[at
tr-defined]
   1130     return compiled_fn(*args, **kwargs)

File ~\anaconda3\envs\dsml\lib\site-packages\torchvision\ops\boxes.py:109,
in _batched_nms_vanilla(boxes, scores, idxs, iou_threshold)
    107 for class_id in torch.unique(idxs):
    108     curr_indices = torch.where(idxs == class_id)[0]
--> 109     curr_keep_indices = nms(boxes[curr_indices], scores[curr_indic
es], iou_threshold)
    110     keep_mask[curr_indices[curr_keep_indices]] = True
    111 keep_indices = torch.where(keep_mask)[0]

File ~\anaconda3\envs\dsml\lib\site-packages\torchvision\ops\boxes.py:40,
in nms(boxes, scores, iou_threshold)
     38 if not torch.jit.is_scripting() and not torch.jit.is_tracing():
     39     _log_api_usage_once(nms)
---> 40 _assert_has_ops()
     41 return torch.ops.torchvision.nms(boxes, scores, iou_threshold)

File ~\anaconda3\envs\dsml\lib\site-packages\torchvision\extension.py:48,
in _assert_has_ops()
     46 def _assert_has_ops():
     47     if not _has_ops():
---> 48         raise RuntimeError(
     49             "Couldn't load custom C++ ops. This can happen if your
PyTorch and "
     50             "torchvision versions are incompatible, or if you had
errors while compiling "
     51             "torchvision from source. For further information on t
he compatible versions, check "
```

```
     52             "https://github.com/pytorch/vision#installation for th
e compatibility matrix. "
     53             "Please check your PyTorch version with torch.__versio
n__ and your torchvision "
     54             "version with torchvision.__version__ and verify if th
ey are compatible, and if not "
     55             "please reinstall torchvision so that it matches your
PyTorch install."
     56         )
```

RuntimeError: Couldn't load custom C++ ops. This can happen if your PyTorch and torchvision versions are incompatible, or if you had errors while compiling torchvision from source. For further information on the compatible versions, check https://github.com/pytorch/vision#installation (https://github.com/pytorch/vision#installation) for the compatibility matrix. Please check your PyTorch version with torch.__version__ and your torchvision version with torchvision.__version__ and verify if they are compatible, and if not please reinstall torchvision so that it matches your PyTorch install.

In [ ]:
```
1
```