In [1]:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:

```
# Specify the path to your zip file
zip_file_path = '/content/sample_data/LAB_09.zip'

# Specify the directory where you want to extract the contents
extracted_folder_path = '/content/sample_data/folder'

# Unzip the file
import zipfile
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_folder_path)
```

In [4]:

```
# Define constants
img_height, img_width = 224, 224
num_classes = 7
batch_size = 32

# Define data paths
train_data_dir = '/content/sample_data/folder/LAB_09/TRAIN'
test_data_dir = '/content/sample_data/folder/LAB_09/TEST'

# Data preprocessing and augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

# Data generators
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

In [5]:

```
# Load pre-trained ResNet model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(img_height, im
g_width, 3))

# Freeze the layers of the pre-trained ResNet
for layer in base_model.layers:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet
/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 0s 0us/step
```

In [6]:

```
# Build your classification model on top of the pre-trained ResNet
model = models.Sequential()
model.add(base_model)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(7, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))
```

In [7]:

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [8]:

```
# Train the model
epochs = 5  # Adjust the number of epochs as needed
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=test_generator
)
```

```
Epoch 1/5
898/898 [==============================] - 401s 432ms/step - loss: 1.8797 - accuracy: 0.2
506 - val_loss: 1.8435 - val_accuracy: 0.2471
Epoch 2/5
898/898 [==============================] - 389s 433ms/step - loss: 1.8282 - accuracy: 0.2
513 - val_loss: 1.8233 - val_accuracy: 0.2471
Epoch 3/5
898/898 [==============================] - 390s 434ms/step - loss: 1.8160 - accuracy: 0.2
513 - val_loss: 1.8166 - val_accuracy: 0.2471
Epoch 4/5
898/898 [==============================] - 387s 431ms/step - loss: 1.8119 - accuracy: 0.2
513 - val_loss: 1.8142 - val_accuracy: 0.2471
Epoch 5/5
898/898 [==============================] - 386s 429ms/step - loss: 1.8104 - accuracy: 0.2
513 - val_loss: 1.8134 - val_accuracy: 0.2471
```

In [9]:

```
# Evaluate the model on the test set
accuracy = model.evaluate(test_generator)[1]
print('Test Accuracy: {:.2%}'.format(accuracy))
```

```
225/225 [==============================] - 25s 109ms/step - loss: 1.8134 - accuracy: 0.24
71
Test Accuracy: 24.71%
```

In [15]:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

image_index = 0

test_image_path = test_generator.filepaths[image_index]
img = image.load_img(test_image_path, target_size=(img_height, img_width))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.0

predictions = model.predict(img_array)
```

```
predicted_label = test_generator.classes[image_index]
predicted_class_name = list(test_generator.class_indices.keys())[predicted_label]

true_label = int(test_generator.classes[image_index])
true_class_name = list(test_generator.class_indices.keys())[true_label]

plt.imshow(img)
plt.axis('off')

print("True Label:", true_class_name)
print("Predicted Label:", predicted_class_name)

# Show the plot
plt.show()
```

```
1/1 [==============================] - 0s 67ms/step
True Label: angry
Predicted Label: angry
```

In [ ]:

In [2]:
```
!pip install ultralytics
```

```
Collecting ultralytics
  Downloading ultralytics-8.0.217-py3-none-any.whl (645 kB)
     ------------------------------------ 645.7/645.7 kB 2.5 MB/s eta
0:00:00
Requirement already satisfied: matplotlib>=3.3.0 in c:\users\dummy\anacon
da3\anaconda\lib\site-packages (from ultralytics) (3.7.0)
Requirement already satisfied: numpy>=1.22.2 in c:\users\dummy\anaconda3
\anaconda\lib\site-packages (from ultralytics) (1.23.5)
Requirement already satisfied: pandas>=1.1.4 in c:\users\dummy\anaconda3
\anaconda\lib\site-packages (from ultralytics) (1.5.3)
Requirement already satisfied: seaborn>=0.11.0 in c:\users\dummy\anaconda
3\anaconda\lib\site-packages (from ultralytics) (0.12.2)
Requirement already satisfied: tqdm>=4.64.0 in c:\users\dummy\anaconda3\a
naconda\lib\site-packages (from ultralytics) (4.64.1)
Requirement already satisfied: scipy>=1.4.1 in c:\users\dummy\anaconda3\a
naconda\lib\site-packages (from ultralytics) (1.10.0)
Requirement already satisfied: psutil in c:\users\dummy\anaconda3\anacond
a\lib\site-packages (from ultralytics) (5.9.0)
Collecting thop>=0.1.1
  Downloading thop-0.1.1.post2209072238-py3-none-any.whl (15 kB)
Requirement already satisfied: pillow>=7.1.2 in c:\users\dummy\anaconda3
\anaconda\lib\site-packages (from ultralytics) (9.4.0)
Requirement already satisfied: torch>=1.8.0 in c:\users\dummy\anaconda3\a
naconda\lib\site-packages (from ultralytics) (2.0.1)
Requirement already satisfied: torchvision>=0.9.0 in c:\users\dummy\anaco
nda3\anaconda\lib\site-packages (from ultralytics) (0.15.2)
Requirement already satisfied: opencv-python>=4.6.0 in c:\users\dummy\ana
conda3\anaconda\lib\site-packages (from ultralytics) (4.8.0.76)
Collecting py-cpuinfo
  Downloading py_cpuinfo-9.0.0-py3-none-any.whl (22 kB)
Requirement already satisfied: requests>=2.23.0 in c:\users\dummy\anacond
a3\anaconda\lib\site-packages (from ultralytics) (2.28.1)
Requirement already satisfied: pyyaml>=5.3.1 in c:\users\dummy\anaconda3
\anaconda\lib\site-packages (from ultralytics) (6.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dummy\ana
conda3\anaconda\lib\site-packages (from matplotlib>=3.3.0->ultralytics)
(2.8.2)
Requirement already satisfied: packaging>=20.0 in c:\users\dummy\anaconda
3\anaconda\lib\site-packages (from matplotlib>=3.3.0->ultralytics) (22.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\dummy\anacon
da3\anaconda\lib\site-packages (from matplotlib>=3.3.0->ultralytics) (1.
4.4)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\dummy\anacond
a3\anaconda\lib\site-packages (from matplotlib>=3.3.0->ultralytics) (3.0.
9)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dummy\anacon
da3\anaconda\lib\site-packages (from matplotlib>=3.3.0->ultralytics) (4.2
5.0)
Requirement already satisfied: cycler>=0.10 in c:\users\dummy\anaconda3\a
naconda\lib\site-packages (from matplotlib>=3.3.0->ultralytics) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\dummy\anacond
a3\anaconda\lib\site-packages (from matplotlib>=3.3.0->ultralytics) (1.0.
5)
Requirement already satisfied: pytz>=2020.1 in c:\users\dummy\anaconda3\a
naconda\lib\site-packages (from pandas>=1.1.4->ultralytics) (2022.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dummy\anaco
nda3\anaconda\lib\site-packages (from requests>=2.23.0->ultralytics) (202
3.7.22)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\dummy
\anaconda3\anaconda\lib\site-packages (from requests>=2.23.0->ultralytic
s) (2.0.4)
```

```
Requirement already satisfied: idna<4,>=2.5 in c:\users\dummy\anaconda3\a
naconda\lib\site-packages (from requests>=2.23.0->ultralytics) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\dummy\an
aconda3\anaconda\lib\site-packages (from requests>=2.23.0->ultralytics)
(1.26.14)
Requirement already satisfied: typing-extensions in c:\users\dummy\anacon
da3\anaconda\lib\site-packages (from torch>=1.8.0->ultralytics) (4.4.0)
Requirement already satisfied: sympy in c:\users\dummy\anaconda3\anaconda
\lib\site-packages (from torch>=1.8.0->ultralytics) (1.11.1)
Requirement already satisfied: filelock in c:\users\dummy\anaconda3\anaco
nda\lib\site-packages (from torch>=1.8.0->ultralytics) (3.12.2)
Requirement already satisfied: networkx in c:\users\dummy\anaconda3\anaco
nda\lib\site-packages (from torch>=1.8.0->ultralytics) (2.8.4)
Requirement already satisfied: jinja2 in c:\users\dummy\anaconda3\anacond
a\lib\site-packages (from torch>=1.8.0->ultralytics) (3.1.2)
Requirement already satisfied: colorama in c:\users\dummy\anaconda3\anaco
nda\lib\site-packages (from tqdm>=4.64.0->ultralytics) (0.4.6)
Requirement already satisfied: six>=1.5 in c:\users\dummy\anaconda3\anaco
nda\lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.3.0->ultr
alytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\dummy\anaconda
3\anaconda\lib\site-packages (from jinja2->torch>=1.8.0->ultralytics) (2.
1.1)
Requirement already satisfied: mpmath>=0.19 in c:\users\dummy\anaconda3\a
naconda\lib\site-packages (from sympy->torch>=1.8.0->ultralytics) (1.2.1)
Installing collected packages: py-cpuinfo, thop, ultralytics
Successfully installed py-cpuinfo-9.0.0 thop-0.1.1.post2209072238 ultraly
tics-8.0.217
```

In [ ]:
```
import cv2
import math
from ultralytics import YOLO

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

model = YOLO("yolo-Weights/yolov8n.pt")

lab_asset_classes = ["lab_equipment", "chemical_flask", "microscope", "comp

while True:
    ret, img = cap.read()

    results = model(img, stream=True)

    for r in results.xyxy[0]:
        x1, y1, x2, y2 = map(int, r[:4])
        confidence = math.ceil(r[4] * 100) / 100
        cls = int(r[5])
        class_name = lab_asset_classes[cls] if cls < len(lab_asset_classes)

        cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 255), 3)

        print(f"Confidence: {confidence}, Class: {class_name}")

        org = (x1, y1 - 10)
        font = cv2.FONT_HERSHEY_SIMPLEX
        font_scale = 1
        color = (255, 9, 0)
        thickness = 2
        cv2.putText(img, class_name, org, font, font_scale, color, thicknes


    cv2.imshow('Lab Assets Detection', img)

    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

In [ ]:

Import libraries

In [1]:
```
import numpy as np
import pandas as pd
import os
import torch
import torchvision
from torchvision import datasets, models
from torchvision.transforms import functional as FT
from torchvision import transforms as T
from torch import nn, optim
from torch.nn import functional as F
from torch.utils.data import DataLoader, sampler, random_split, Dataset
import copy
import math
from PIL import Image
import cv2
import albumentations as A

import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
from collections import defaultdict, deque
import datetime
import time
from tqdm import tqdm
from torchvision.utils import draw_bounding_boxes
```

In [3]:
```
print(torch.__version__)
print(torchvision.__version__)
```

```
1.9.1
0.10.1
```

In [4]:
```
!pip install pycocotools
from pycocotools.coco import COCO
```

```
Collecting pycocotools
  Downloading pycocotools-2.0.7-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(403 kB)
     |████████████████████████████████| 403 kB 5.9 MB/s
Requirement already satisfied: matplotlib>=2.1.0 in /opt/conda/lib/python3.7/site-packages
(from pycocotools) (3.5.1)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from pycoc
otools) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-packages
(from matplotlib>=2.1.0->pycocotools) (1.3.2)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (fr
om matplotlib>=2.1.0->pycocotools) (8.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (fro
m matplotlib>=2.1.0->pycocotools) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/site-packages
(from matplotlib>=2.1.0->pycocotools) (4.28.2)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.7/site-packa
ges (from matplotlib>=2.1.0->pycocotools) (2.8.0)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/site-packages
(from matplotlib>=2.1.0->pycocotools) (3.0.6)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-packages
(from matplotlib>=2.1.0->pycocotools) (21.3)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from py
thon-dateutil>=2.7->matplotlib>=2.1.0->pycocotools) (1.16.0)
Installing collected packages: pycocotools
Successfully installed pycocotools-2.0.7
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting b
ehaviour with the system package manager. It is recommended to use a virtual environment i
nstead: https://pip.pypa.io/warnings/venv (https://pip.pypa.io/warnings/venv)
```

In [5]:
```
from albumentations.pytorch import ToTensorV2
```

CODE

In [6]:
```
def get_transforms(train=False):
    if train:
        transform = A.Compose([
            A.Resize(600, 600),
            A.HorizontalFlip(p=0.3),
            A.VerticalFlip(p=0.3),
            A.RandomBrightnessContrast(p=0.1),
            A.ColorJitter(p=0.1),
            ToTensorV2()
        ], bbox_params=A.BboxParams(format='coco'))
    else:
        transform = A.Compose([
            A.Resize(600, 600),
            ToTensorV2()
        ], bbox_params=A.BboxParams(format='coco'))
    return transform
```

In [7]:
```
class AquariumDetection(datasets.VisionDataset):
    def __init__(self, root, split='train', transform=None, target_transform=None, transfor
        super().__init__(root, transforms, transform, target_transform)
        self.split = split
        self.coco = COCO(os.path.join(root, split, "_annotations.coco.json"))
        self.ids = list(sorted(self.coco.imgs.keys()))
        self.ids = [id for id in self.ids if (len(self._load_target(id)) > 0)]

    def _load_image(self, id: int):
        path = self.coco.loadImgs(id)[0]['file_name']
        image = cv2.imread(os.path.join(self.root, self.split, path))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        return image
    def _load_target(self, id):
        return self.coco.loadAnns(self.coco.getAnnIds(id))

    def __getitem__(self, index):
        id = self.ids[index]
        image = self._load_image(id)
        target = self._load_target(id)
        target = copy.deepcopy(self._load_target(id))

        boxes = [t['bbox'] + [t['category_id']] for t in target]
        if self.transforms is not None:
            transformed = self.transforms(image=image, bboxes=boxes)

        image = transformed['image']
        boxes = transformed['bboxes']

        new_boxes = []
        for box in boxes:
            xmin = box[0]
            xmax = xmin + box[2]
            ymin = box[1]
            ymax = ymin + box[3]
            new_boxes.append([xmin, ymin, xmax, ymax])

        boxes = torch.tensor(new_boxes, dtype=torch.float32)

        targ = {}
        targ['boxes'] = boxes
        targ['labels'] = torch.tensor([t['category_id'] for t in target], dtype=torch.int64
        targ['image_id'] = torch.tensor([t['image_id'] for t in target])
        targ['area'] = (boxes[:, 3] - boxes[:, 1]) * (boxes[:, 2] - boxes[:, 0])
        targ['iscrowd'] = torch.tensor([t['iscrowd'] for t in target], dtype=torch.int64)
        return image.div(255), targ
    def __len__(self):
        return len(self.ids)
```

In [8]:
```
dataset_path = "/kaggle/input/aquarium-dataset/Aquarium Combined"
```

In [9]:
```
coco = COCO(os.path.join(dataset_path, "train", "_annotations.coco.json"))
categories = coco.cats
n_classes = len(categories.keys())
categories
```

```
loading annotations into memory...
Done (t=0.06s)
creating index...
index created!
```

Out[9]:
```
{0: {'id': 0, 'name': 'creatures', 'supercategory': 'none'},
 1: {'id': 1, 'name': 'fish', 'supercategory': 'creatures'},
 2: {'id': 2, 'name': 'jellyfish', 'supercategory': 'creatures'},
 3: {'id': 3, 'name': 'penguin', 'supercategory': 'creatures'},
 4: {'id': 4, 'name': 'puffin', 'supercategory': 'creatures'},
 5: {'id': 5, 'name': 'shark', 'supercategory': 'creatures'},
 6: {'id': 6, 'name': 'starfish', 'supercategory': 'creatures'},
 7: {'id': 7, 'name': 'stingray', 'supercategory': 'creatures'}}
```

In [10]:
```
classes = [i[1]['name'] for i in categories.items()]
classes
```

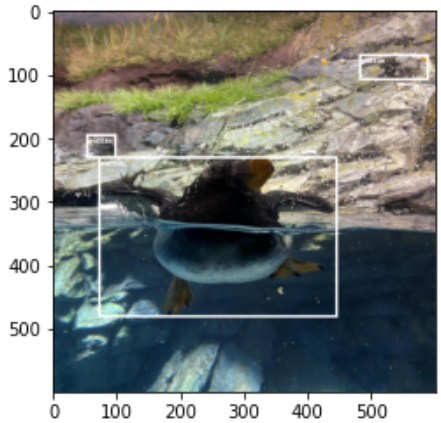
Out[10]:
```
['creatures',
 'fish',
 'jellyfish',
 'penguin',
 'puffin',
 'shark',
 'starfish',
 'stingray']
```

In [11]:
```
train_dataset = AquariumDetection(root=dataset_path, transforms=get_transforms(True))
```

```
loading annotations into memory...
Done (t=0.02s)
creating index...
index created!
```

In [12]:
```
sample = train_dataset[2]
img_int = torch.tensor(sample[0] * 255, dtype=torch.uint8)
plt.imshow(draw_bounding_boxes(
    img_int, sample[1]['boxes'], [classes[i] for i in sample[1]['labels']], width=4
).permute(1, 2, 0))
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x7fa2667f3650>
```



In [13]:
```
len(train_dataset)
```

Out[13]:
```
447
```

In [14]:
```
model = models.detection.fasterrcnn_mobilenet_v3_large_fpn(pretrained=True)
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = models.detection.faster_rcnn.FastRCNNPredictor(in_features,
```

```
Downloading: "https://download.pytorch.org/models/fasterrcnn_mobilenet_v3_large_fpn-fb6a3c
c7.pth" to /root/.cache/torch/hub/checkpoints/fasterrcnn_mobilenet_v3_large_fpn-fb6a3cc7.p
th

  0%|          | 0.00/74.2M [00:00<?, ?B/s]
```

In [15]:
```
def collate_fn(batch):
    return tuple(zip(*batch))
```

In [16]:
```
train_loader = DataLoader(train_dataset, batch_size=4, shuffle=True, num_workers=4, collate
```

In [17]:
```
images,targets = next(iter(train_loader))
images = list(image for image in images)
targets = [{k:v for k, v in t.items()} for t in targets]
output = model(images, targets)
```

In [18]:
```
device = torch.device("cuda")
```

In [19]:
```
model = model.to(device)
```

In [20]:
```
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.01, momentum=0.9, nesterov=True, weight_decay=1e-4
```

In [21]:
```
import sys
```

In [22]:
```
def train_one_epoch(model, optimizer, loader, device, epoch):
    model.to(device)
    model.train()

    all_losses = []
    all_losses_dict = []

    for images, targets in tqdm(loader):
        images = list(image.to(device) for image in images)
        targets = [{k: torch.tensor(v).to(device) for k, v in t.items()} for t in targets]

        loss_dict = model(images, targets)
        losses = sum(loss for loss in loss_dict.values())
        loss_dict_append = {k: v.item() for k, v in loss_dict.items()}
        loss_value = losses.item()

        all_losses.append(loss_value)
        all_losses_dict.append(loss_dict_append)

        if not math.isfinite(loss_value):
            print(f"Loss is {loss_value}, stopping trainig")
            print(loss_dict)
            sys.exit(1)

        optimizer.zero_grad()
        losses.backward()
        optimizer.step()

    all_losses_dict = pd.DataFrame(all_losses_dict) # for printing
    print("Epoch {}, lr: {:.6f}, loss: {:.6f}, loss_classifier: {:.6f}, loss_box: {:.6f}, lo
        epoch, optimizer.param_groups[0]['lr'], np.mean(all_losses),
        all_losses_dict['loss_classifier'].mean(),
        all_losses_dict['loss_box_reg'].mean(),
        all_losses_dict['loss_rpn_box_reg'].mean(),
        all_losses_dict['loss_objectness'].mean()
    ))
```

In [23]:
```
num_epochs=10

for epoch in range(num_epochs):
    train_one_epoch(model, optimizer, train_loader, device, epoch)
```

```
100%|██████████| 112/112 [00:22<00:00,  4.90it/s]

Epoch 0, lr: 0.010000, loss: 0.995030, loss_classifier: 0.464037, loss_box: 0.402295, loss
_rpn_box: 0.033534, loss_object: 0.095164

100%|██████████| 112/112 [00:16<00:00,  6.96it/s]

Epoch 1, lr: 0.010000, loss: 0.789345, loss_classifier: 0.354852, loss_box: 0.344495, loss
_rpn_box: 0.028941, loss_object: 0.061057

100%|██████████| 112/112 [00:15<00:00,  7.10it/s]

Epoch 2, lr: 0.010000, loss: 0.728536, loss_classifier: 0.311697, loss_box: 0.344034, loss
_rpn_box: 0.026197, loss_object: 0.046608

100%|██████████| 112/112 [00:15<00:00,  7.18it/s]

Epoch 3, lr: 0.010000, loss: 0.699221, loss_classifier: 0.280837, loss_box: 0.352691, loss
_rpn_box: 0.024920, loss_object: 0.040773

100%|██████████| 112/112 [00:15<00:00,  7.24it/s]

Epoch 4, lr: 0.010000, loss: 0.681142, loss_classifier: 0.262584, loss_box: 0.356135, loss
_rpn_box: 0.023867, loss_object: 0.038556

100%|██████████| 112/112 [00:15<00:00,  7.33it/s]

Epoch 5, lr: 0.010000, loss: 0.669380, loss_classifier: 0.255460, loss_box: 0.357264, loss
_rpn_box: 0.022854, loss_object: 0.033802

100%|██████████| 112/112 [00:15<00:00,  7.40it/s]

Epoch 6, lr: 0.010000, loss: 0.651492, loss_classifier: 0.245125, loss_box: 0.353056, loss
_rpn_box: 0.022852, loss_object: 0.030459

100%|██████████| 112/112 [00:15<00:00,  7.46it/s]

Epoch 7, lr: 0.010000, loss: 0.603885, loss_classifier: 0.225554, loss_box: 0.329301, loss
_rpn_box: 0.021540, loss_object: 0.027490

100%|██████████| 112/112 [00:14<00:00,  7.50it/s]

Epoch 8, lr: 0.010000, loss: 0.625815, loss_classifier: 0.226839, loss_box: 0.352307, loss
_rpn_box: 0.020505, loss_object: 0.026165

100%|██████████| 112/112 [00:14<00:00,  7.57it/s]

Epoch 9, lr: 0.010000, loss: 0.624672, loss_classifier: 0.227926, loss_box: 0.349201, loss
_rpn_box: 0.020486, loss_object: 0.027060
```

In [25]:
```
model.eval()
torch.cuda.empty_cache()
```

In [26]:
```
test_dataset = AquariumDetection(root=dataset_path, split="test", transforms=get_transforms
```

```
loading annotations into memory...
Done (t=0.02s)
creating index...
index created!
```

In [27]:
```
img, _ = test_dataset[5]
img_int = torch.tensor(img*255, dtype=torch.uint8)
with torch.no_grad():
    prediction = model([img.to(device)])
    pred = prediction[0]
```

In [28]:
```
fig = plt.figure(figsize=(14, 10))
plt.imshow(draw_bounding_boxes(img_int,
    pred['boxes'][pred['scores'] > 0.8],
    [classes[i] for i in pred['labels'][pred['scores'] > 0.8].tolist()], width=4
).permute(1, 2, 0))
```

Out[28]: <matplotlib.image.AxesImage at 0x7fa22e675f10>