Karan Vora kv2154
ECE-GY 7123 Introduction to Deep Learning

Problem 1):

Solution a):

$$c = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Solution b):

$$c = \begin{bmatrix} -1 & 0 & -1 \\ -1 & 0 & -1 \\ -1 & 0 & -1 \end{bmatrix}$$

Solution c):

$$c = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Solution d):

$$c = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

Solution e):

One example of an image operation that cannot be implemented using a 3x3 convolutional filter is a non-linear edge detector. A non-linear edge detector aims to highlight the edges in an image using non-linear operations. This can be achieved using techniques such as morphological operations or non-linear filtering, which involve operations such as dilation, erosion, and median filtering. These non-linear operations cannot be implemented using a 3x3 convolutional filter because a convolutional filter is a linear operator that operates on a local neighborhood of pixels in a fixed way. It applies the same linear transformation to each pixel in the neighborhood, regardless of its value or position. Therefore, a 3x3 convolutional filter cannot capture the complex non-linear relationships between pixels that are necessary for non-linear edge detection.

Problem 2):

Solution a):

The formula for l_2 loss is given by

$$L(w)=\frac{1}{2}\|y-Wx\|^2$$

So the new l_2 loss with λ parameter for regularization is

$$\bar{L}(w)=L(w)+\lambda\|w\|_2^2$$

Solution b):

If the learning rate η, then the general gradient update rule is,

$$w_{t+1}=w_t-\eta\nabla\bar{L}(w)$$

$$w_{t+1}=w_t-\lambda\nabla L(w_t)-2\eta\lambda w_t$$

$$w_{t+1}=(1-2\eta\lambda)w_t-\eta\nabla L(w_t)$$

Solution c):

From the equation mentioned in Solution B, we can see that the updated *w* consists of shrinking/decaying gradient by a factor of $(1-2\eta\lambda)$ and then updating in the direction of the gradient

Solution d):

Increasing λ penalizes the l_2 norm of the weight vector, thus enforcing smaller weights on average. In order for the gradient to be stable, the constraining factor should be smaller than 1, i.e. $\eta<\frac{1}{2\lambda}$

Problem 3):

Solution a):

The definition of IOU for any two bounding boxes A and B is given by:

$$IOU(A,B)=\frac{|A\cap B|}{|A\cup B|}$$

Since the RHS is non-negative, the number has to be bigger than or equal to 0. Moreover, $A\cap B\subseteq A\cup B$ and hence the numerator has to be no bigger than the denominator. Therefore IOU is bounded between 0 and 1 (inclusive).

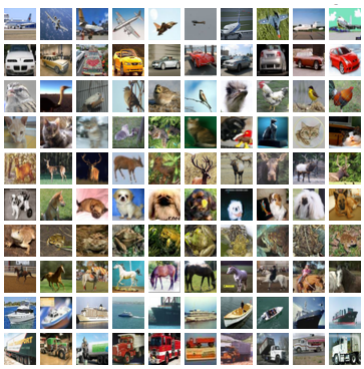
Solution b):

Consider two identical size square boxes A and B, both aligned at the same horizontal level. Fix B and then imagine sliding A from left to right. As A moves, the IOU will start from 0, increase until perfect overlap and then decrease until no overlap. The graph we will get is a step funtion i.e., it jumps from 0 to 1 when the boxes overlap and stays at 0 otherwise, the change in I)U will be discontinuous and will not have a well-defined derivative.

AlexNet

In this problem, you are asked to train a deep convolutional neural network to perform image classification. In fact, this is a slight variation of a network called *AlexNet*. This is a landmark model in deep learning, and arguably kickstarted the current (and ongoing, and massive) wave of innovation in modern AI when its results were first presented in 2012. AlexNet was the first real-world demonstration of a *deep* classifier that was trained end-to-end on data and that outperformed all other ML models thus far.

We will train AlexNet using the CIFAR10 dataset, which consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.



A lot of the code you will need is already provided in this notebook; all you need to do is to fill in the missing pieces, and interpret your results.

Warning : AlexNet takes a good amount of time to train (~1 minute per epoch on Google Colab). So please budget enough time to do this homework.

In [ ]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.optim.lr_scheduler import _LRScheduler
import torch.utils.data as data

import torchvision.transforms as transforms
import torchvision.datasets as datasets

from sklearn import decomposition
from sklearn import manifold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import numpy as np

import copy
import random
import time
```

In [ ]:
```
SEED = 1234

random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
torch.backends.cudnn.deterministic = True
```

Loading and Preparing the Data

Our dataset is made up of color images but three color channels (red, green and blue), compared to MNIST's black and white images with a single color channel. To normalize our data we need to calculate the means and standard deviations for each of the color channels independently, and normalize them.

In [ ]:
```
ROOT = '.data'
train_data = datasets.CIFAR10(root = ROOT,
                              train = True,
                              download = True)
```

```
Files already downloaded and verified
```

In [ ]:
```
# Compute means and standard deviations along the R,G,B channel

means = train_data.data.mean(axis = (0,1,2)) / 255
stds = train_data.data.std(axis = (0,1,2)) / 255
```

Next, we will do data augmentation. For each training image we will randomly rotate it (by up to 5 degrees), flip/mirror with probability 0.5, shift by +/-1 pixel. Finally we will normalize each color channel using the means/stds we calculated above.

In [ ]:
```
train_transforms = transforms.Compose([
                            transforms.RandomRotation(5),
                            transforms.RandomHorizontalFlip(0.5),
                            transforms.RandomCrop(32, padding = 2),
                            transforms.ToTensor(),
                            transforms.Normalize(mean = means,
                                                 std = stds)
                        ])

test_transforms = transforms.Compose([
                            transforms.ToTensor(),
                            transforms.Normalize(mean = means,
                                                 std = stds)
                        ])
```

Next, we'll load the dataset along with the transforms defined above.

We will also create a validation set with 10% of the training samples. The validation set will be used to monitor loss along different epochs, and we will pick the model along the optimization path that performed the best, and report final test accuracy numbers using this model.

In [ ]:
```
train_data = datasets.CIFAR10(ROOT,
                              train = True,
                              download = True,
                              transform = train_transforms)

test_data = datasets.CIFAR10(ROOT,
                             train = False,
                             download = True,
                             transform = test_transforms)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

In [ ]:
```
VALID_RATIO = 0.9

n_train_examples = int(len(train_data) * VALID_RATIO)
n_valid_examples = len(train_data) - n_train_examples

train_data, valid_data = data.random_split(train_data,
                                           [n_train_examples, n_valid_examples])
```

In [ ]:
```
valid_data = copy.deepcopy(valid_data)
valid_data.dataset.transform = test_transforms
```

Now, we'll create a function to plot some of the images in our dataset to see what they actually look like.

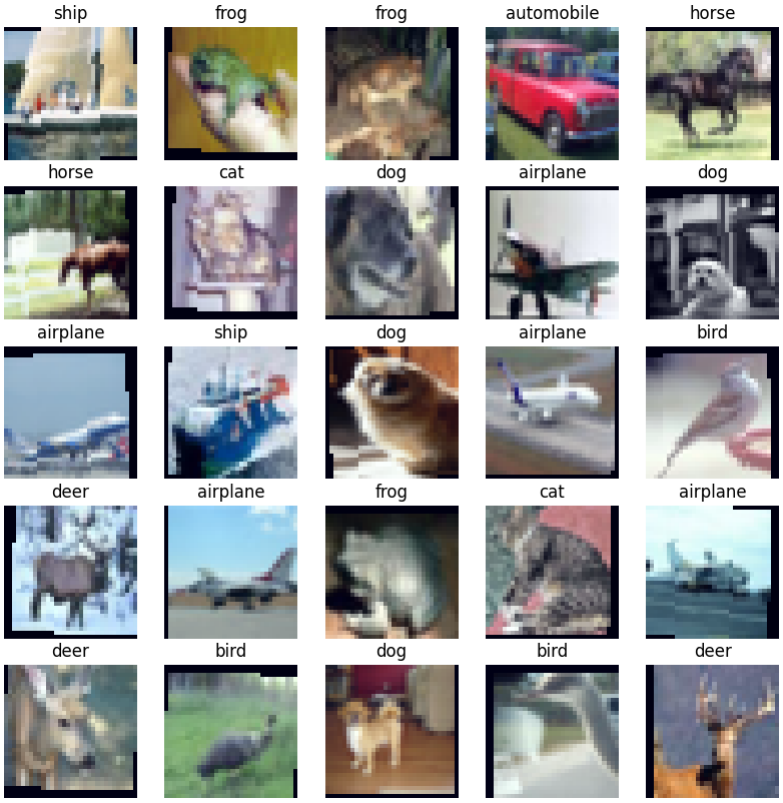
Note that by default PyTorch handles images that are arranged `[channel, height, width]`, but `matplotlib` expects images to be `[height, width, channel]`, hence we need to permute the dimensions of our images before plotting them.

In [ ]:
```
def plot_images(images, labels, classes, normalize = False):

    n_images = len(images)

    rows = int(np.sqrt(n_images))
    cols = int(np.sqrt(n_images))

    fig = plt.figure(figsize = (10, 10))

    for i in range(rows*cols):

        ax = fig.add_subplot(rows, cols, i+1)

        image = images[i]

        if normalize:
            image_min = image.min()
            image_max = image.max()
            image.clamp_(min = image_min, max = image_max)
            image.add_(-image_min).div_(image_max - image_min + 1e-5)

        ax.imshow(image.permute(1, 2, 0).cpu().numpy())
        ax.set_title(classes[labels[i]])
        ax.axis('off')
```

One point here: `matplotlib` is expecting the values of every pixel to be between $[0, 1]$, however our normalization will cause them to be outside this range. By default `matplotlib` will then clip these values into the $[0, 1]$ range. This clipping causes all of the images to look a bit weird - all of the colors are oversaturated. The solution is to normalize each image between [0,1].

In [ ]:
```
N_IMAGES = 25

images, labels = zip(*[(image, label) for image, label in
                       [train_data[i] for i in range(N_IMAGES)]])

classes = test_data.classes
```

In [ ]:
```
plot_images(images, labels, classes, normalize = True)
```



We'll be normalizing our images by default from now on, so we'll write a function that does it for us which we can use whenever we need to renormalize an image.

In [ ]:
```
def normalize_image(image):
    image_min = image.min()
    image_max = image.max()
    image.clamp_(min = image_min, max = image_max)
    image.add_(-image_min).div_(image_max - image_min + 1e-5)
    return image
```

The final bit of the data processing is creating the iterators. We will use a large. Generally, a larger batch size means that our model trains faster but is a bit more susceptible to overfitting.

In [ ]:
```
# Q1: Create data loaders for train_data, valid_data, test_data
# Use batch size 256


# BATCH_SIZE =

# train_iterator =

# valid_iterator =

# test_iterator =

BATCH_SIZE = 256
train_iterator = data.DataLoader(train_data, batch_size=BATCH_SIZE, shuffle=True)
valid_iterator = data.DataLoader(valid_data, batch_size=BATCH_SIZE)
test_iterator = data.DataLoader(test_data, batch_size=BATCH_SIZE)
```

Defining the Model

Next up is defining the model.

AlexNet will have the following architecture:

- There are 5 2D convolutional layers (which serve as *feature extractors*), followed by 3 linear layers (which serve as the *classifier*).
- All layers (except the last one) have `ReLU` activations. (Use `inplace=True` while defining your ReLUs.)
- All convolutional filter sizes have kernel size 3 x 3 and padding 1.
- Convolutional layer 1 has stride 2. All others have the default stride (1).
- Convolutional layers 1,2, and 5 are followed by a 2D maxpool of size 2.
- Linear layers 1 and 2 are preceded by Dropouts with Bernoulli parameter 0.5.
- For the convolutional layers, the number of channels is set as follows. We start with 3 channels and then proceed like this:
  - $3 \rightarrow 64 \rightarrow 192 \rightarrow 384 \rightarrow 256 \rightarrow 256$

  In the end, if everything is correct you should get a feature map of size $2 \times 2 \times 256 = 1024$.
- For the linear layers, the feature sizes are as follows:
  - $1024 \rightarrow 4096 \rightarrow 4096 \rightarrow 10$.

  (The 10, of course, is because 10 is the number of classes in CIFAR-10).

In [ ]:
```
# class AlexNet(nn.Module):
#     def __init__(self, output_dim):
#         super().__init__()

#         self.features = nn.Sequential(
```

```
#             # Define according to the steps described above
#             nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
#             nn.ReLU(inplace=True),
#             nn.MaxPool2d(kernel_size=3, stride=2),
#             nn.Conv2d(64, 192, kernel_size=5, padding=2),
#             nn.ReLU(inplace=True),
#             nn.MaxPool2d(kernel_size=3, stride=2),
#             nn.Conv2d(192, 384, kernel_size=3, padding=1),
#             nn.ReLU(inplace=True),
#             nn.Conv2d(384, 256, kernel_size=3, padding=1),
#             nn.ReLU(inplace=True),
#             nn.Conv2d(256, 256, kernel_size=3, padding=1),
#             nn.ReLU(inplace=True),
#             nn.MaxPool2d(kernel_size=3, stride=2)
#         )
#         self.avgpool = nn.AdaptiveAvgPool2d((6, 6))

#         self.classifier = nn.Sequential(
#             # define according to the steps described above
#             nn.Dropout(),
#             nn.Linear(256 * 6 * 6, 4096),
#             nn.ReLU(inplace=True),
#             nn.Dropout(),
#             nn.Linear(4096, 4096),
#             nn.ReLU(inplace=True),
#             nn.Linear(4096, output_dim)
#         )

#     def forward(self, x):
#         x = self.features(x)
#         x = self.avgpool(x)
#         # h = x.view(x.shape[0], -1)
#         h = x.view(x.size(0), 256 * 6 * 6)
#         x = self.classifier(h)
#         return x, h
```

In [ ]:
```
class AlexNet(nn.Module):
    def __init__(self, num_classes=10):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, stride=2, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
            nn.Conv2d(64, 192, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((2, 2))
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 2 * 2, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x):
        x = self.features(x)
        x = self.avgpool(x)
        h = x.view(x.size(0), 256 * 2 * 2)
        x = self.classifier(h)
        return x, h
```

We'll create an instance of our model with the desired amount of classes.

In [ ]:
```
OUTPUT_DIM = 10
model = AlexNet(OUTPUT_DIM)
```

Training the Model

We first initialize parameters in PyTorch by creating a function that takes in a PyTorch module, checking what type of module it is, and then using the `nn.init` methods to actually initialize the parameters.

For convolutional layers we will initialize using the *Kaiming Normal* scheme, also known as *He Normal*. For the linear layers we initialize using the *Xavier Normal* scheme, also known as *Glorot Normal*. For both types of layer we initialize the bias terms to zeros.

In [ ]:
```
def initialize_parameters(m):
    if isinstance(m, nn.Conv2d):
        nn.init.kaiming_normal_(m.weight.data, nonlinearity = 'relu')
        nn.init.constant_(m.bias.data, 0)
    elif isinstance(m, nn.Linear):
        nn.init.xavier_normal_(m.weight.data, gain = nn.init.calculate_gain('relu'))
        nn.init.constant_(m.bias.data, 0)
```

We apply the initialization by using the model's `apply` method. If your definitions above are correct you should get the printed output as below.

In [ ]:
```
model.apply(initialize_parameters)
```

Out[ ]:
```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(2, 2))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=1024, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=10, bias=True)
  )
)
```

We then define the loss function we want to use, the device we'll use and place our model and criterion on to our device.

In [ ]:
```
optimizer = optim.Adam(model.parameters(), lr = 1e-3)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
criterion = nn.CrossEntropyLoss()

model = model.to(device)
criterion = criterion.to(device)
```

```
    # This is formatted as code
```

We define a function to calculate accuracy...

In [ ]:
```
def calculate_accuracy(y_pred, y):
    top_pred = y_pred.argmax(1, keepdim = True)
    correct = top_pred.eq(y.view_as(top_pred)).sum()
    acc = correct.float() / y.shape[0]
    return acc
```

As we are using dropout we need to make sure to "turn it on" when training by using `model.train()`.

In [ ]:
```
def train(model, iterator, optimizer, criterion, device):

    epoch_loss = 0
    epoch_acc = 0

    model.train()

    for (x, y) in iterator:

        x = x.to(device)
        y = y.to(device)

        optimizer.zero_grad()

        y_pred, _ = model(x)

        loss = criterion(y_pred, y)

        acc = calculate_accuracy(y_pred, y)

        loss.backward()

        optimizer.step()

        epoch_loss += loss.item()
        epoch_acc += acc.item()

    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

We also define an evaluation loop, making sure to "turn off" dropout with `model.eval()`.

In [ ]:
```
def evaluate(model, iterator, criterion, device):

    epoch_loss = 0
    epoch_acc = 0

    model.eval()

    with torch.no_grad():

        for (x, y) in iterator:

            x = x.to(device)
            y = y.to(device)

            y_pred, _ = model(x)

            loss = criterion(y_pred, y)

            acc = calculate_accuracy(y_pred, y)

            epoch_loss += loss.item()
            epoch_acc += acc.item()

    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

Next, we define a function to tell us how long an epoch takes.

In [ ]:
```
def epoch_time(start_time, end_time):
    elapsed_time = end_time - start_time
    elapsed_mins = int(elapsed_time / 60)
```

```
    elapsed_secs = int(elapsed_time - (elapsed_mins * 60))
    return elapsed_mins, elapsed_secs
```

Then, finally, we train our model.

Train it for 25 epochs (using the train dataset). At the end of each epoch, compute the validation loss and keep track of the best model. You might find the command `torch.save` helpful.

At the end you should expect to see validation losses of ~76% accuracy.

In [ ]:
```
# Q3: train your model here for 25 epochs.
# Print out training and validation loss/accuracy of the model after each epoch
# Keep track of the model that achieved best validation loss thus far.

EPOCHS = 25

# Fill training code here

best_valid_loss = float('inf')
for epoch in range(EPOCHS):
    train_loss, train_acc = train(model, train_iterator, optimizer, criterion, device)
    valid_loss, valid_acc = evaluate(model, valid_iterator, criterion, device)

    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model.state_dict(), 'best_model.pt')

    print(f'Epoch: {epoch+1:02}')
    print(f'\tTrain Loss: {train_loss:.3f} | Train Acc: {train_acc*100:.2f}%')
    print(f'\t Val. Loss: {valid_loss:.3f} |  Val. Acc: {valid_acc*100:.2f}%')
```

```
Epoch: 01
        Train Loss: 2.385 | Train Acc: 21.71%
         Val. Loss: 1.613 |  Val. Acc: 39.04%
Epoch: 02
        Train Loss: 1.539 | Train Acc: 43.00%
         Val. Loss: 1.351 |  Val. Acc: 50.60%
Epoch: 03
        Train Loss: 1.357 | Train Acc: 50.75%
         Val. Loss: 1.241 |  Val. Acc: 54.72%
Epoch: 04
        Train Loss: 1.256 | Train Acc: 54.90%
         Val. Loss: 1.141 |  Val. Acc: 58.73%
Epoch: 05
        Train Loss: 1.173 | Train Acc: 58.41%
         Val. Loss: 1.083 |  Val. Acc: 61.63%
Epoch: 06
        Train Loss: 1.112 | Train Acc: 60.57%
         Val. Loss: 1.059 |  Val. Acc: 62.47%
Epoch: 07
        Train Loss: 1.066 | Train Acc: 62.07%
         Val. Loss: 1.018 |  Val. Acc: 65.02%
Epoch: 08
        Train Loss: 1.008 | Train Acc: 64.62%
         Val. Loss: 0.997 |  Val. Acc: 64.91%
Epoch: 09
        Train Loss: 0.966 | Train Acc: 66.29%
         Val. Loss: 0.936 |  Val. Acc: 67.48%
Epoch: 10
        Train Loss: 0.929 | Train Acc: 67.48%
         Val. Loss: 0.919 |  Val. Acc: 68.54%
Epoch: 11
        Train Loss: 0.899 | Train Acc: 68.67%
         Val. Loss: 0.922 |  Val. Acc: 67.75%
Epoch: 12
        Train Loss: 0.876 | Train Acc: 69.48%
         Val. Loss: 0.835 |  Val. Acc: 71.28%
Epoch: 13
        Train Loss: 0.841 | Train Acc: 70.71%
         Val. Loss: 0.849 |  Val. Acc: 70.74%
Epoch: 14
        Train Loss: 0.827 | Train Acc: 71.38%
         Val. Loss: 0.832 |  Val. Acc: 72.24%
Epoch: 15
        Train Loss: 0.800 | Train Acc: 72.07%
         Val. Loss: 0.804 |  Val. Acc: 72.87%
Epoch: 16
        Train Loss: 0.770 | Train Acc: 73.30%
         Val. Loss: 0.802 |  Val. Acc: 72.40%
Epoch: 17
        Train Loss: 0.756 | Train Acc: 73.56%
         Val. Loss: 0.784 |  Val. Acc: 73.14%
Epoch: 18
        Train Loss: 0.739 | Train Acc: 74.43%
         Val. Loss: 0.753 |  Val. Acc: 74.50%
Epoch: 19
        Train Loss: 0.721 | Train Acc: 75.17%
         Val. Loss: 0.767 |  Val. Acc: 73.64%
Epoch: 20
        Train Loss: 0.710 | Train Acc: 75.37%
         Val. Loss: 0.754 |  Val. Acc: 74.73%
Epoch: 21
        Train Loss: 0.692 | Train Acc: 76.04%
         Val. Loss: 0.757 |  Val. Acc: 74.90%
Epoch: 22
        Train Loss: 0.679 | Train Acc: 76.66%
         Val. Loss: 0.748 |  Val. Acc: 75.09%
Epoch: 23
        Train Loss: 0.673 | Train Acc: 76.63%
         Val. Loss: 0.727 |  Val. Acc: 75.59%
Epoch: 24
        Train Loss: 0.650 | Train Acc: 77.80%
         Val. Loss: 0.726 |  Val. Acc: 75.35%
Epoch: 25
        Train Loss: 0.644 | Train Acc: 78.15%
         Val. Loss: 0.756 |  Val. Acc: 74.38%
```

Evaluating the model

We then load the parameters of our model that achieved the best validation loss. You should expect to see ~75% accuracy of this model on the test dataset.

Finally, plot the confusion matrix of this model and comment on any interesting patterns you can observe there. For example, which two classes are confused the most?

In [ ]:
```
# Q4: Load the best performing model, evaluate it on the test dataset, and print test accuracy.

# Also, print out the confusion matrox.
```

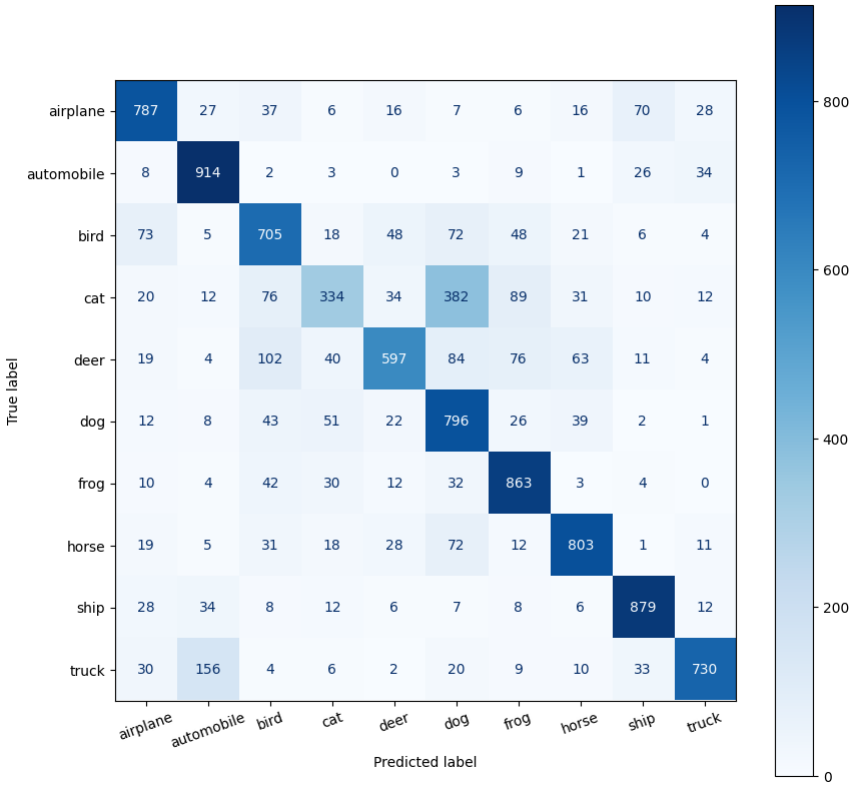
In [ ]:
```
# def get_predictions(model, iterator, device):

#     model.eval()

#     labels = []
#     probs = []

#     # Q4: Fill code here.


#     labels = torch.cat(labels, dim = 0)
#     probs = torch.cat(probs, dim = 0)

#     return labels, probs

def get_predictions(model, iterator, device):

    model.eval()

    labels = []
    probs = []

    with torch.no_grad():
        for batch in iterator:
            x, y = batch
            x = x.to(device)
            y = y.to(device)
            y_pred, _ = model(x)
            labels.append(y)
            probs.append(torch.softmax(y_pred, dim=1))

    labels = torch.cat(labels, dim=0)
    probs = torch.cat(probs, dim=0)

    return labels, probs
```

In [ ]:
```
labels, probs = get_predictions(model, test_iterator, device)
```

In [ ]:
```
pred_labels = torch.argmax(probs, 1)
```

In [ ]:
```
def plot_confusion_matrix(labels, pred_labels, classes):

    fig = plt.figure(figsize = (10, 10));
    ax = fig.add_subplot(1, 1, 1);
    cm = confusion_matrix(labels, pred_labels);
    cm = ConfusionMatrixDisplay(cm, display_labels = classes);
    cm.plot(values_format = 'd', cmap = 'Blues', ax = ax)
    plt.xticks(rotation = 20)
```

In [ ]:
```
plot_confusion_matrix(labels, pred_labels, classes)
```



Conclusion

That's it! As a side project (this is not for credit and won't be graded), feel free to play around with different design choices that you made while building this network.

- Whether or not to normalize the color channels in the input.
- The learning rate parameter in Adam.
- The batch size.
- The number of training epochs.

- (and if you are feeling brave -- the AlexNet architecture itself.)

In [8]:
```
import os
import numpy as np
import torch
from PIL import Image


class PennFudanDataset(torch.utils.data.Dataset):
    def __init__(self, root, transforms):
        self.root = root
        self.transforms = transforms
        # load all image files, sorting them to
        # ensure that they are aligned
        self.imgs = list(sorted(os.listdir(os.path.join(root, "PNGImages"))))
        self.masks = list(sorted(os.listdir(os.path.join(root, "PedMasks"))))

    def __getitem__(self, idx):
        # load images and masks
        img_path = os.path.join(self.root, "PNGImages", self.imgs[idx])
        mask_path = os.path.join(self.root, "PedMasks", self.masks[idx])
        img = Image.open(img_path).convert("RGB")
        # note that we haven't converted the mask to RGB,
        # because each color corresponds to a different instance
        # with 0 being background
        mask = Image.open(mask_path)
        # convert the PIL Image into a numpy array
        mask = np.array(mask)
        # instances are encoded as different colors
        obj_ids = np.unique(mask)
        # first id is the background, so remove it
        obj_ids = obj_ids[1:]

        # split the color-encoded mask into a set
        # of binary masks
        masks = mask == obj_ids[:, None, None]

        # get bounding box coordinates for each mask
        num_objs = len(obj_ids)
        boxes = []
        for i in range(num_objs):
            pos = np.where(masks[i])
            xmin = np.min(pos[1])
            xmax = np.max(pos[1])
            ymin = np.min(pos[0])
            ymax = np.max(pos[0])
            boxes.append([xmin, ymin, xmax, ymax])

        # convert everything into a torch.Tensor
        boxes = torch.as_tensor(boxes, dtype=torch.float32)
        # there is only one class
        labels = torch.ones((num_objs,), dtype=torch.int64)
        masks = torch.as_tensor(masks, dtype=torch.uint8)

        image_id = torch.tensor([idx])
        area = (boxes[:, 3] - boxes[:, 1]) * (boxes[:, 2] - boxes[:, 0])
        # suppose all instances are not crowd
        iscrowd = torch.zeros((num_objs,), dtype=torch.int64)

        target = {}
        target["boxes"] = boxes
        target["labels"] = labels
        target["masks"] = masks
        target["image_id"] = image_id
        target["area"] = area
        target["iscrowd"] = iscrowd

        if self.transforms is not None:
            img, target = self.transforms(img, target)

        return img, target

    def __len__(self):
        return len(self.imgs)
```

In [9]:
```
import torchvision
import torchvision
from torchvision.models.detection import FasterRCNN
from torchvision.models.detection.rpn import AnchorGenerator
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor

def get_model_instance_segmentation_option1(num_classes):

    # load an instance segmentation model pre-trained pre-trained on COCO
    model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)

    # get number of input features for the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features

    # replace the pre-trained head with a new one
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    # now get the number of input features for the mask classifier
    in_features_mask = model.roi_heads.mask_predictor.conv5_mask.in_channels
    hidden_layer = 256

    # and replace the mask predictor with a new one
    model.roi_heads.mask_predictor = MaskRCNNPredictor(in_features_mask,
                                                       hidden_layer,
                                                       num_classes)

    return model

def get_model_instance_segmentation_option2(num_classes):

    # load a pre-trained model for classification and return
    # only the features
    backbone = torchvision.models.mobilenet_v2(weights="DEFAULT").features

    # FasterRCNN needs to know the number of
    # output channels in a backbone. For mobilenet_v2, it's 1280
    # so we need to add it here
    backbone.out_channels = 1280
```

```
    # let's make the RPN generate 5 x 3 anchors per spatial
    # location, with 5 different sizes and 3 different aspect
    # ratios. We have a Tuple[Tuple[int]] because each feature
    # map could potentially have different sizes and
    # aspect ratios
    anchor_generator = AnchorGenerator(sizes=((32, 64, 128, 256, 512),),
                                       aspect_ratios=((0.5, 1.0, 2.0),))

    # let's define what are the feature maps that we will
    # use to perform the region of interest cropping, as well as
    # the size of the crop after rescaling.
    # if your backbone returns a Tensor, featmap_names is expected to
    # be [0]. More generally, the backbone should return an
    # OrderedDict[Tensor], and in featmap_names you can choose which
    # feature maps to use.
    roi_pooler = torchvision.ops.MultiScaleRoIAlign(featmap_names=['0'],
                                                    output_size=7,
                                                    sampling_ratio=2)
    # put the pieces together inside a FasterRCNN model
    model = FasterRCNN(backbone,
                       num_classes=2,
                       rpn_anchor_generator=anchor_generator,
                       box_roi_pool=roi_pooler)
    return model
```

In [10]:
```
import transforms as T

def get_transform(train):
    transforms = []
    transforms.append(T.PILToTensor())
    transforms.append(T.ConvertImageDtype(torch.float))
    if train:
        transforms.append(T.RandomHorizontalFlip(0.5))
    return T.Compose(transforms)
```

In [11]:
```
from engine import train_one_epoch, evaluate
import utils
from PIL import Image
import torch
import torchvision.transforms as transforms
import requests
import utils
#%matplotlib inline
import matplotlib.pyplot as plt

# train on the GPU or on the CPU, if a GPU is not available
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')

# our dataset has two classes only - background and person
num_classes = 2

# use our dataset and defined transformations
dataset = PennFudanDataset('/content/drive/MyDrive/Colab Notebooks/PennFudanPed', get_transform(train=True))
dataset_test = PennFudanDataset('/content/drive/MyDrive/Colab Notebooks/PennFudanPed', get_transform(train=False))

# split the dataset in train and test set
indices = torch.randperm(len(dataset)).tolist()
dataset = torch.utils.data.Subset(dataset, indices[:-50])
dataset_test = torch.utils.data.Subset(dataset_test, indices[-50:])

# define training and validation data loaders
data_loader = torch.utils.data.DataLoader(
                       dataset, batch_size=2, shuffle=True, num_workers=0,
                       collate_fn=utils.collate_fn)

data_loader_test = torch.utils.data.DataLoader(
                       dataset_test, batch_size=1, shuffle=False, num_workers=0,
                       collate_fn=utils.collate_fn)
```

In [12]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [13]:
```
#MODEL_OPTION1

# get the model using our helper function
model_option1 = get_model_instance_segmentation_option1(num_classes)

# move model to the right device
model_option1.to(device)

# construct an optimizer for option1
params = [p for p in model_option1.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,
momentum=0.9, weight_decay=0.0005)

# and a learning rate scheduler for the same
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
step_size=3,
gamma=0.1)

# let's train it for 10 epochs
num_epochs = 10
print ("MODEL OPTION 1")

for epoch in range(num_epochs):
    print("Model 1 Epoch: {}".format(epoch))
    # train for one epoch, printing every 10 iterations
    train_one_epoch(model_option1, optimizer, data_loader, device, epoch, print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    evaluate(model_option1, data_loader_test, device=device)

print ("\n")
print("That's it! Model 1")
```

```
/usr/local/lib/python3.9/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future,
please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 a
nd may be removed in the future. The current behavior is equivalent to passing `weights=MaskRCNN_ResNet50_FPN_Weights.COCO_V1`. You can also use `weights=MaskRCNN_ResNet50_
FPN_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
```

```
MODEL OPTION 1
Model 1 Epoch: 0
Epoch: [0]  [ 0/60]  eta: 0:01:56  lr: 0.000090  loss: 5.8660 (5.8660)  loss_classifier: 0.7740 (0.7740)  loss_box_reg: 0.3408 (0.3408)  loss_mask: 4.7247 (4.7247)  loss_ob
jectness: 0.0241 (0.0241)  loss_rpn_box_reg: 0.0023 (0.0023)  time: 1.9364  data: 1.3563  max mem: 2579
Epoch: [0]  [10/60]  eta: 0:01:33  lr: 0.000936  loss: 2.3789 (3.1197)  loss_classifier: 0.5083 (0.4910)  loss_box_reg: 0.3408 (0.2984)  loss_mask: 1.4818 (2.3071)  loss_ob
jectness: 0.0192 (0.0185)  loss_rpn_box_reg: 0.0028 (0.0047)  time: 1.8733  data: 1.3209  max mem: 3528
Epoch: [0]  [20/60]  eta: 0:01:14  lr: 0.001783  loss: 0.9242 (2.0173)  loss_classifier: 0.2323 (0.3516)  loss_box_reg: 0.3054 (0.2896)  loss_mask: 0.4465 (1.3515)  loss_ob
jectness: 0.0192 (0.0187)  loss_rpn_box_reg: 0.0030 (0.0059)  time: 1.8659  data: 1.3193  max mem: 3528
Epoch: [0]  [30/60]  eta: 0:00:56  lr: 0.002629  loss: 0.5756 (1.5340)  loss_classifier: 0.0937 (0.2659)  loss_box_reg: 0.2108 (0.2590)  loss_mask: 0.2391 (0.9873)  loss_ob
jectness: 0.0099 (0.0160)  loss_rpn_box_reg: 0.0059 (0.0058)  time: 1.9024  data: 1.3500  max mem: 3554
Epoch: [0]  [40/60]  eta: 0:00:39  lr: 0.003476  loss: 0.4931 (1.2783)  loss_classifier: 0.0713 (0.2166)  loss_box_reg: 0.2089 (0.2468)  loss_mask: 0.2166 (0.7948)  loss_ob
jectness: 0.0081 (0.0141)  loss_rpn_box_reg: 0.0059 (0.0059)  time: 2.0443  data: 1.4791  max mem: 3554
Epoch: [0]  [50/60]  eta: 0:00:19  lr: 0.004323  loss: 0.4387 (1.1157)  loss_classifier: 0.0552 (0.1853)  loss_box_reg: 0.1454 (0.2312)  loss_mask: 0.1952 (0.6806)  loss_ob
jectness: 0.0045 (0.0122)  loss_rpn_box_reg: 0.0057 (0.0065)  time: 2.0565  data: 1.4892  max mem: 3559
Epoch: [0]  [59/60]  eta: 0:00:01  lr: 0.005000  loss: 0.3701 (0.9989)  loss_classifier: 0.0367 (0.1630)  loss_box_reg: 0.1329 (0.2163)  loss_mask: 0.1797 (0.6026)  loss_ob
jectness: 0.0016 (0.0106)  loss_rpn_box_reg: 0.0056 (0.0063)  time: 1.9448  data: 1.3731  max mem: 3742
Epoch: [0] Total time: 0:01:57 (1.9522 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1674 (0.1674)  evaluator_time: 0.0149 (0.0149)  time: 0.1996  data: 0.0165  max mem: 3742
Test:  [49/50]  eta: 0:00:00  model_time: 0.1023 (0.1111)  evaluator_time: 0.0067 (0.0120)  time: 0.1457  data: 0.0200  max mem: 3742
Test: Total time: 0:00:07 (0.1417 s / it)
Averaged stats: model_time: 0.1023 (0.1111)  evaluator_time: 0.0067 (0.0120)
Accumulating evaluation results...
DONE (t=0.03s).
Accumulating evaluation results...
DONE (t=0.03s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.593
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.968
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.766
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.381
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.499
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.604
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.260
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.660
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.660
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.633
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.666
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.611
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.972
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.762
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.330
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.337
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.626
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.272
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.658
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.663
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.533
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.678
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.665
Model 1 Epoch: 1
Epoch: [1]  [ 0/60]  eta: 0:00:35  lr: 0.005000  loss: 0.4237 (0.4237)  loss_classifier: 0.0566 (0.0566)  loss_box_reg: 0.1427 (0.1427)  loss_mask: 0.2189 (0.2189)  loss_ob
jectness: 0.0003 (0.0003)  loss_rpn_box_reg: 0.0052 (0.0052)  time: 0.5921  data: 0.0361  max mem: 3742
Epoch: [1]  [10/60]  eta: 0:00:31  lr: 0.005000  loss: 0.3257 (0.3524)  loss_classifier: 0.0435 (0.0462)  loss_box_reg: 0.1320 (0.1303)  loss_mask: 0.1620 (0.1690)  loss_ob
jectness: 0.0017 (0.0024)  loss_rpn_box_reg: 0.0031 (0.0045)  time: 0.6349  data: 0.0409  max mem: 3742
Epoch: [1]  [20/60]  eta: 0:00:25  lr: 0.005000  loss: 0.3009 (0.3246)  loss_classifier: 0.0401 (0.0439)  loss_box_reg: 0.1076 (0.1142)  loss_mask: 0.1506 (0.1588)  loss_ob
jectness: 0.0018 (0.0025)  loss_rpn_box_reg: 0.0040 (0.0053)  time: 0.6325  data: 0.0400  max mem: 3742
Epoch: [1]  [30/60]  eta: 0:00:18  lr: 0.005000  loss: 0.2838 (0.3094)  loss_classifier: 0.0400 (0.0424)  loss_box_reg: 0.0880 (0.1051)  loss_mask: 0.1452 (0.1543)  loss_ob
jectness: 0.0012 (0.0023)  loss_rpn_box_reg: 0.0053 (0.0054)  time: 0.6196  data: 0.0378  max mem: 3742
Epoch: [1]  [40/60]  eta: 0:00:12  lr: 0.005000  loss: 0.2544 (0.2860)  loss_classifier: 0.0311 (0.0379)  loss_box_reg: 0.0549 (0.0915)  loss_mask: 0.1316 (0.1493)  loss_ob
jectness: 0.0008 (0.0021)  loss_rpn_box_reg: 0.0033 (0.0052)  time: 0.6021  data: 0.0364  max mem: 3742
Epoch: [1]  [50/60]  eta: 0:00:06  lr: 0.005000  loss: 0.2330 (0.2792)  loss_classifier: 0.0262 (0.0377)  loss_box_reg: 0.0546 (0.0882)  loss_mask: 0.1263 (0.1461)  loss_ob
jectness: 0.0007 (0.0022)  loss_rpn_box_reg: 0.0032 (0.0049)  time: 0.6087  data: 0.0352  max mem: 3742
Epoch: [1]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.2324 (0.2717)  loss_classifier: 0.0275 (0.0368)  loss_box_reg: 0.0553 (0.0841)  loss_mask: 0.1263 (0.1440)  loss_ob
jectness: 0.0004 (0.0020)  loss_rpn_box_reg: 0.0032 (0.0049)  time: 0.6104  data: 0.0345  max mem: 3742
Epoch: [1] Total time: 0:00:36 (0.6159 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:10  model_time: 0.1657 (0.1657)  evaluator_time: 0.0120 (0.0120)  time: 0.2010  data: 0.0225  max mem: 3742
Test:  [49/50]  eta: 0:00:00  model_time: 0.0985 (0.1064)  evaluator_time: 0.0037 (0.0055)  time: 0.1263  data: 0.0167  max mem: 3742
Test: Total time: 0:00:06 (0.1297 s / it)
Averaged stats: model_time: 0.0985 (0.1064)  evaluator_time: 0.0037 (0.0055)
Accumulating evaluation results...
DONE (t=0.01s).
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.768
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.982
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.942
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.363
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.618
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.785
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.336
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.808
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.808
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.733
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.822
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.741
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.982
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.926
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.370
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.506
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.759
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.321
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.770
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.770
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.600
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.689
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.781
Model 1 Epoch: 2
Epoch: [2]  [ 0/60]  eta: 0:00:43  lr: 0.005000  loss: 0.2319 (0.2319)  loss_classifier: 0.0396 (0.0396)  loss_box_reg: 0.0734 (0.0734)  loss_mask: 0.1158 (0.1158)  loss_ob
jectness: 0.0003 (0.0003)  loss_rpn_box_reg: 0.0029 (0.0029)  time: 0.7233  data: 0.0464  max mem: 3742
Epoch: [2]  [10/60]  eta: 0:00:30  lr: 0.005000  loss: 0.2233 (0.2445)  loss_classifier: 0.0276 (0.0330)  loss_box_reg: 0.0562 (0.0700)  loss_mask: 0.1351 (0.1349)  loss_ob
jectness: 0.0006 (0.0012)  loss_rpn_box_reg: 0.0046 (0.0054)  time: 0.6173  data: 0.0452  max mem: 3742
Epoch: [2]  [20/60]  eta: 0:00:23  lr: 0.005000  loss: 0.1867 (0.2278)  loss_classifier: 0.0240 (0.0292)  loss_box_reg: 0.0483 (0.0615)  loss_mask: 0.1240 (0.1324)  loss_ob
jectness: 0.0004 (0.0009)  loss_rpn_box_reg: 0.0029 (0.0038)  time: 0.5888  data: 0.0386  max mem: 3742
Epoch: [2]  [30/60]  eta: 0:00:18  lr: 0.005000  loss: 0.2387 (0.2389)  loss_classifier: 0.0289 (0.0308)  loss_box_reg: 0.0599 (0.0674)  loss_mask: 0.1259 (0.1353)  loss_ob
jectness: 0.0004 (0.0012)  loss_rpn_box_reg: 0.0026 (0.0042)  time: 0.6107  data: 0.0351  max mem: 3742
Epoch: [2]  [40/60]  eta: 0:00:12  lr: 0.005000  loss: 0.2450 (0.2316)  loss_classifier: 0.0295 (0.0310)  loss_box_reg: 0.0661 (0.0635)  loss_mask: 0.1324 (0.1319)  loss_ob
```

```
jectness: 0.0007 (0.0012)  loss_rpn_box_reg: 0.0047 (0.0041)  time: 0.6250  data: 0.0382  max mem: 3748
Epoch: [2]  [50/60]  eta: 0:00:06  lr: 0.005000  loss: 0.2110 (0.2333)  loss_classifier: 0.0314 (0.0322)  loss_box_reg: 0.0511 (0.0652)  loss_mask: 0.1215 (0.1306)  loss_ob
jectness: 0.0004 (0.0011)  loss_rpn_box_reg: 0.0037 (0.0042)  time: 0.5832  data: 0.0360  max mem: 3748
Epoch: [2]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.2345 (0.2314)  loss_classifier: 0.0341 (0.0316)  loss_box_reg: 0.0635 (0.0641)  loss_mask: 0.1213 (0.1306)  loss_ob
jectness: 0.0003 (0.0010)  loss_rpn_box_reg: 0.0034 (0.0041)  time: 0.5916  data: 0.0374  max mem: 3748
Epoch: [2] Total time: 0:00:36 (0.6053 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1700 (0.1700)  evaluator_time: 0.0067 (0.0067)  time: 0.1945  data: 0.0171  max mem: 3748
Test:  [49/50]  eta: 0:00:00  model_time: 0.0988 (0.1064)  evaluator_time: 0.0035 (0.0054)  time: 0.1285  data: 0.0173  max mem: 3748
Test: Total time: 0:00:06 (0.1296 s / it)
Averaged stats: model_time: 0.0988 (0.1064)  evaluator_time: 0.0035 (0.0054)
Accumulating evaluation results...
DONE (t=0.02s).
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.788
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.982
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.945
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.374
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.612
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.805
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.343
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.828
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.828
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.533
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.842
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.731
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.983
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.897
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.400
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.542
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.744
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.322
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.766
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.768
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.633
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.744
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.773
Model 1 Epoch: 3
Epoch: [3]  [ 0/60]  eta: 0:00:36  lr: 0.000500  loss: 0.2125 (0.2125)  loss_classifier: 0.0267 (0.0267)  loss_box_reg: 0.0480 (0.0480)  loss_mask: 0.1368 (0.1368)  loss_ob
jectness: 0.0001 (0.0001)  loss_rpn_box_reg: 0.0009 (0.0009)  time: 0.6009  data: 0.0399  max mem: 3748
Epoch: [3]  [10/60]  eta: 0:00:30  lr: 0.000500  loss: 0.2125 (0.2237)  loss_classifier: 0.0285 (0.0297)  loss_box_reg: 0.0511 (0.0574)  loss_mask: 0.1352 (0.1306)  loss_ob
jectness: 0.0006 (0.0014)  loss_rpn_box_reg: 0.0041 (0.0047)  time: 0.6178  data: 0.0428  max mem: 3748
Epoch: [3]  [20/60]  eta: 0:00:23  lr: 0.000500  loss: 0.1779 (0.1978)  loss_classifier: 0.0258 (0.0269)  loss_box_reg: 0.0359 (0.0461)  loss_mask: 0.1114 (0.1203)  loss_ob
jectness: 0.0005 (0.0010)  loss_rpn_box_reg: 0.0027 (0.0035)  time: 0.5927  data: 0.0367  max mem: 3748
Epoch: [3]  [30/60]  eta: 0:00:18  lr: 0.000500  loss: 0.1763 (0.1988)  loss_classifier: 0.0246 (0.0268)  loss_box_reg: 0.0328 (0.0486)  loss_mask: 0.1049 (0.1188)  loss_ob
jectness: 0.0005 (0.0010)  loss_rpn_box_reg: 0.0024 (0.0036)  time: 0.6036  data: 0.0373  max mem: 3748
Epoch: [3]  [40/60]  eta: 0:00:12  lr: 0.000500  loss: 0.1909 (0.1962)  loss_classifier: 0.0260 (0.0266)  loss_box_reg: 0.0494 (0.0467)  loss_mask: 0.1148 (0.1185)  loss_ob
jectness: 0.0004 (0.0010)  loss_rpn_box_reg: 0.0030 (0.0033)  time: 0.6157  data: 0.0384  max mem: 3748
Epoch: [3]  [50/60]  eta: 0:00:06  lr: 0.000500  loss: 0.1962 (0.1967)  loss_classifier: 0.0260 (0.0270)  loss_box_reg: 0.0474 (0.0466)  loss_mask: 0.1155 (0.1185)  loss_ob
jectness: 0.0004 (0.0011)  loss_rpn_box_reg: 0.0021 (0.0035)  time: 0.6055  data: 0.0383  max mem: 3748
Epoch: [3]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.1837 (0.1939)  loss_classifier: 0.0250 (0.0267)  loss_box_reg: 0.0397 (0.0455)  loss_mask: 0.1110 (0.1174)  loss_ob
jectness: 0.0005 (0.0011)  loss_rpn_box_reg: 0.0028 (0.0034)  time: 0.6071  data: 0.0379  max mem: 3748
Epoch: [3] Total time: 0:00:36 (0.6063 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:10  model_time: 0.1773 (0.1773)  evaluator_time: 0.0087 (0.0087)  time: 0.2051  data: 0.0183  max mem: 3748
Test:  [49/50]  eta: 0:00:00  model_time: 0.0974 (0.1074)  evaluator_time: 0.0032 (0.0066)  time: 0.1257  data: 0.0172  max mem: 3748
Test: Total time: 0:00:06 (0.1344 s / it)
Averaged stats: model_time: 0.0974 (0.1074)  evaluator_time: 0.0032 (0.0066)
Accumulating evaluation results...
DONE (t=0.01s).
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.808
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.987
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.949
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.384
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.627
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.826
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.352
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.840
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.840
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.533
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.855
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.741
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.978
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.878
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.396
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.531
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.755
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.328
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.772
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.772
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.778
Model 1 Epoch: 4
Epoch: [4]  [ 0/60]  eta: 0:00:29  lr: 0.000500  loss: 0.1836 (0.1836)  loss_classifier: 0.0257 (0.0257)  loss_box_reg: 0.0415 (0.0415)  loss_mask: 0.1111 (0.1111)  loss_ob
jectness: 0.0040 (0.0040)  loss_rpn_box_reg: 0.0013 (0.0013)  time: 0.4944  data: 0.0278  max mem: 3748
Epoch: [4]  [10/60]  eta: 0:00:28  lr: 0.000500  loss: 0.1576 (0.1762)  loss_classifier: 0.0249 (0.0266)  loss_box_reg: 0.0334 (0.0379)  loss_mask: 0.1037 (0.1091)  loss_ob
jectness: 0.0009 (0.0012)  loss_rpn_box_reg: 0.0009 (0.0013)  time: 0.5626  data: 0.0343  max mem: 3748
Epoch: [4]  [20/60]  eta: 0:00:23  lr: 0.000500  loss: 0.1803 (0.1852)  loss_classifier: 0.0242 (0.0269)  loss_box_reg: 0.0369 (0.0417)  loss_mask: 0.1173 (0.1138)  loss_ob
jectness: 0.0003 (0.0008)  loss_rpn_box_reg: 0.0018 (0.0020)  time: 0.5829  data: 0.0385  max mem: 3748
Epoch: [4]  [30/60]  eta: 0:00:17  lr: 0.000500  loss: 0.1872 (0.1918)  loss_classifier: 0.0242 (0.0262)  loss_box_reg: 0.0403 (0.0427)  loss_mask: 0.1211 (0.1197)  loss_ob
jectness: 0.0003 (0.0008)  loss_rpn_box_reg: 0.0024 (0.0025)  time: 0.6006  data: 0.0399  max mem: 3748
Epoch: [4]  [40/60]  eta: 0:00:11  lr: 0.000500  loss: 0.1794 (0.1886)  loss_classifier: 0.0227 (0.0258)  loss_box_reg: 0.0361 (0.0418)  loss_mask: 0.1128 (0.1175)  loss_ob
jectness: 0.0006 (0.0008)  loss_rpn_box_reg: 0.0024 (0.0027)  time: 0.6085  data: 0.0374  max mem: 3748
Epoch: [4]  [50/60]  eta: 0:00:05  lr: 0.000500  loss: 0.1797 (0.1870)  loss_classifier: 0.0268 (0.0263)  loss_box_reg: 0.0384 (0.0411)  loss_mask: 0.1041 (0.1159)  loss_ob
jectness: 0.0007 (0.0012)  loss_rpn_box_reg: 0.0022 (0.0027)  time: 0.6102  data: 0.0356  max mem: 3748
Epoch: [4]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.1908 (0.1876)  loss_classifier: 0.0295 (0.0266)  loss_box_reg: 0.0431 (0.0418)  loss_mask: 0.1082 (0.1150)  loss_ob
jectness: 0.0008 (0.0013)  loss_rpn_box_reg: 0.0022 (0.0029)  time: 0.6307  data: 0.0368  max mem: 3748
Epoch: [4] Total time: 0:00:36 (0.6054 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1630 (0.1630)  evaluator_time: 0.0052 (0.0052)  time: 0.1859  data: 0.0170  max mem: 3748
Test:  [49/50]  eta: 0:00:00  model_time: 0.0977 (0.1057)  evaluator_time: 0.0028 (0.0049)  time: 0.1254  data: 0.0165  max mem: 3748
Test: Total time: 0:00:06 (0.1282 s / it)
```

```
Averaged stats: model_time: 0.0977 (0.1057)  evaluator_time: 0.0028 (0.0049)
Accumulating evaluation results...
DONE (t=0.01s).
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.826
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.987
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.950
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.413
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.650
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.843
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.357
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.857
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.857
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.872
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.752
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.987
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.901
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.385
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.523
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.331
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.780
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.780
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.788
Model 1 Epoch: 5
Epoch: [5]  [ 0/60]  eta: 0:00:28  lr: 0.000500  loss: 0.1180 (0.1180)  loss_classifier: 0.0121 (0.0121)  loss_box_reg: 0.0125 (0.0125)  loss_mask: 0.0923 (0.0923)  loss_ob
jectness: 0.0000 (0.0000)  loss_rpn_box_reg: 0.0011 (0.0011)  time: 0.4785  data: 0.0294  max mem: 3748
Epoch: [5]  [10/60]  eta: 0:00:30  lr: 0.000500  loss: 0.1657 (0.1810)  loss_classifier: 0.0221 (0.0235)  loss_box_reg: 0.0321 (0.0388)  loss_mask: 0.1072 (0.1156)  loss_ob
jectness: 0.0003 (0.0005)  loss_rpn_box_reg: 0.0021 (0.0026)  time: 0.6149  data: 0.0409  max mem: 3748
Epoch: [5]  [20/60]  eta: 0:00:23  lr: 0.000500  loss: 0.1657 (0.1786)  loss_classifier: 0.0221 (0.0248)  loss_box_reg: 0.0321 (0.0373)  loss_mask: 0.1064 (0.1133)  loss_ob
jectness: 0.0003 (0.0006)  loss_rpn_box_reg: 0.0021 (0.0025)  time: 0.5997  data: 0.0365  max mem: 3748
Epoch: [5]  [30/60]  eta: 0:00:18  lr: 0.000500  loss: 0.1862 (0.1854)  loss_classifier: 0.0262 (0.0254)  loss_box_reg: 0.0383 (0.0392)  loss_mask: 0.1127 (0.1176)  loss_ob
jectness: 0.0003 (0.0006)  loss_rpn_box_reg: 0.0027 (0.0027)  time: 0.5969  data: 0.0366  max mem: 3748
Epoch: [5]  [40/60]  eta: 0:00:12  lr: 0.000500  loss: 0.1862 (0.1862)  loss_classifier: 0.0262 (0.0256)  loss_box_reg: 0.0368 (0.0403)  loss_mask: 0.1148 (0.1170)  loss_ob
jectness: 0.0004 (0.0006)  loss_rpn_box_reg: 0.0028 (0.0027)  time: 0.6123  data: 0.0406  max mem: 3748
Epoch: [5]  [50/60]  eta: 0:00:06  lr: 0.000500  loss: 0.1724 (0.1849)  loss_classifier: 0.0255 (0.0256)  loss_box_reg: 0.0304 (0.0402)  loss_mask: 0.1068 (0.1159)  loss_ob
jectness: 0.0004 (0.0006)  loss_rpn_box_reg: 0.0022 (0.0027)  time: 0.6083  data: 0.0412  max mem: 3748
Epoch: [5]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.1579 (0.1812)  loss_classifier: 0.0213 (0.0250)  loss_box_reg: 0.0283 (0.0389)  loss_mask: 0.1046 (0.1140)  loss_ob
jectness: 0.0004 (0.0006)  loss_rpn_box_reg: 0.0021 (0.0027)  time: 0.6049  data: 0.0379  max mem: 3748
Epoch: [5] Total time: 0:00:36 (0.6040 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1698 (0.1698)  evaluator_time: 0.0062 (0.0062)  time: 0.1938  data: 0.0171  max mem: 3748
Test:  [49/50]  eta: 0:00:00  model_time: 0.0985 (0.1068)  evaluator_time: 0.0040 (0.0067)  time: 0.1281  data: 0.0181  max mem: 3748
Test: Total time: 0:00:06 (0.1331 s / it)
Averaged stats: model_time: 0.0985 (0.1068)  evaluator_time: 0.0040 (0.0067)
Accumulating evaluation results...
DONE (t=0.02s).
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.830
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.988
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.950
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.413
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.652
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.848
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.859
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.859
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.874
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.752
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.979
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.892
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.385
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.546
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.333
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.782
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.782
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.790
Model 1 Epoch: 6
Epoch: [6]  [ 0/60]  eta: 0:00:43  lr: 0.000050  loss: 0.1812 (0.1812)  loss_classifier: 0.0300 (0.0300)  loss_box_reg: 0.0430 (0.0430)  loss_mask: 0.1058 (0.1058)  loss_ob
jectness: 0.0002 (0.0002)  loss_rpn_box_reg: 0.0022 (0.0022)  time: 0.7193  data: 0.0380  max mem: 3748
Epoch: [6]  [10/60]  eta: 0:00:29  lr: 0.000050  loss: 0.1659 (0.1628)  loss_classifier: 0.0232 (0.0218)  loss_box_reg: 0.0334 (0.0313)  loss_mask: 0.1058 (0.1075)  loss_ob
jectness: 0.0002 (0.0005)  loss_rpn_box_reg: 0.0014 (0.0016)  time: 0.5810  data: 0.0329  max mem: 3748
Epoch: [6]  [20/60]  eta: 0:00:24  lr: 0.000050  loss: 0.1659 (0.1718)  loss_classifier: 0.0220 (0.0237)  loss_box_reg: 0.0270 (0.0344)  loss_mask: 0.1067 (0.1108)  loss_ob
jectness: 0.0002 (0.0007)  loss_rpn_box_reg: 0.0021 (0.0023)  time: 0.5953  data: 0.0363  max mem: 3748
Epoch: [6]  [30/60]  eta: 0:00:18  lr: 0.000050  loss: 0.1665 (0.1756)  loss_classifier: 0.0245 (0.0252)  loss_box_reg: 0.0295 (0.0366)  loss_mask: 0.1018 (0.1108)  loss_ob
jectness: 0.0003 (0.0006)  loss_rpn_box_reg: 0.0021 (0.0024)  time: 0.6139  data: 0.0381  max mem: 3748
Epoch: [6]  [40/60]  eta: 0:00:12  lr: 0.000050  loss: 0.1640 (0.1768)  loss_classifier: 0.0218 (0.0244)  loss_box_reg: 0.0311 (0.0360)  loss_mask: 0.1035 (0.1134)  loss_ob
jectness: 0.0003 (0.0005)  loss_rpn_box_reg: 0.0020 (0.0025)  time: 0.6043  data: 0.0376  max mem: 3748
Epoch: [6]  [50/60]  eta: 0:00:06  lr: 0.000050  loss: 0.1698 (0.1787)  loss_classifier: 0.0218 (0.0248)  loss_box_reg: 0.0318 (0.0373)  loss_mask: 0.1164 (0.1136)  loss_ob
jectness: 0.0003 (0.0006)  loss_rpn_box_reg: 0.0019 (0.0024)  time: 0.6026  data: 0.0370  max mem: 3748
Epoch: [6]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.1795 (0.1817)  loss_classifier: 0.0246 (0.0254)  loss_box_reg: 0.0394 (0.0381)  loss_mask: 0.1164 (0.1148)  loss_ob
jectness: 0.0005 (0.0008)  loss_rpn_box_reg: 0.0025 (0.0026)  time: 0.6173  data: 0.0388  max mem: 3748
Epoch: [6] Total time: 0:00:36 (0.6076 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1704 (0.1704)  evaluator_time: 0.0053 (0.0053)  time: 0.1924  data: 0.0160  max mem: 3748
Test:  [49/50]  eta: 0:00:00  model_time: 0.0999 (0.1063)  evaluator_time: 0.0045 (0.0061)  time: 0.1340  data: 0.0207  max mem: 3748
Test: Total time: 0:00:06 (0.1314 s / it)
Averaged stats: model_time: 0.0999 (0.1063)  evaluator_time: 0.0045 (0.0061)
Accumulating evaluation results...
DONE (t=0.02s).
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.829
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.988
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.950
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.413
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.652
```

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.847
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.859
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.859
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.874
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.752
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.978
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.892
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.385
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.546
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.766
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.331
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.781
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.781
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.789
Model 1 Epoch: 7
Epoch: [7]  [ 0/60]  eta: 0:00:44  lr: 0.000050  loss: 0.1408 (0.1408)  loss_classifier: 0.0232 (0.0232)  loss_box_reg: 0.0253 (0.0253)  loss_mask: 0.0902 (0.0902)  loss_ob
jectness: 0.0004 (0.0004)  loss_rpn_box_reg: 0.0017 (0.0017)  time: 0.7354  data: 0.0668  max mem: 3748
Epoch: [7]  [10/60]  eta: 0:00:30  lr: 0.000050  loss: 0.1746 (0.1848)  loss_classifier: 0.0262 (0.0266)  loss_box_reg: 0.0324 (0.0415)  loss_mask: 0.1022 (0.1133)  loss_ob
jectness: 0.0003 (0.0007)  loss_rpn_box_reg: 0.0017 (0.0026)  time: 0.6174  data: 0.0494  max mem: 3748
Epoch: [7]  [20/60]  eta: 0:00:24  lr: 0.000050  loss: 0.1710 (0.1811)  loss_classifier: 0.0247 (0.0256)  loss_box_reg: 0.0276 (0.0371)  loss_mask: 0.1077 (0.1152)  loss_ob
jectness: 0.0003 (0.0006)  loss_rpn_box_reg: 0.0017 (0.0026)  time: 0.6016  data: 0.0415  max mem: 3748
Epoch: [7]  [30/60]  eta: 0:00:18  lr: 0.000050  loss: 0.1710 (0.1803)  loss_classifier: 0.0246 (0.0247)  loss_box_reg: 0.0276 (0.0382)  loss_mask: 0.1095 (0.1138)  loss_ob
jectness: 0.0004 (0.0010)  loss_rpn_box_reg: 0.0023 (0.0027)  time: 0.5976  data: 0.0362  max mem: 3748
Epoch: [7]  [40/60]  eta: 0:00:12  lr: 0.000050  loss: 0.1876 (0.1803)  loss_classifier: 0.0246 (0.0251)  loss_box_reg: 0.0425 (0.0383)  loss_mask: 0.1071 (0.1133)  loss_ob
jectness: 0.0005 (0.0010)  loss_rpn_box_reg: 0.0024 (0.0026)  time: 0.6119  data: 0.0368  max mem: 3748
Epoch: [7]  [50/60]  eta: 0:00:06  lr: 0.000050  loss: 0.1731 (0.1800)  loss_classifier: 0.0230 (0.0252)  loss_box_reg: 0.0323 (0.0372)  loss_mask: 0.1144 (0.1143)  loss_ob
jectness: 0.0003 (0.0009)  loss_rpn_box_reg: 0.0020 (0.0025)  time: 0.6093  data: 0.0351  max mem: 3748
Epoch: [7]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.1665 (0.1781)  loss_classifier: 0.0218 (0.0247)  loss_box_reg: 0.0281 (0.0369)  loss_mask: 0.1071 (0.1130)  loss_ob
jectness: 0.0003 (0.0009)  loss_rpn_box_reg: 0.0021 (0.0026)  time: 0.5948  data: 0.0325  max mem: 3748
Epoch: [7] Total time: 0:00:36 (0.6048 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1646 (0.1646)  evaluator_time: 0.0053 (0.0053)  time: 0.1884  data: 0.0178  max mem: 3748
Test:  [49/50]  eta: 0:00:00  model_time: 0.0976 (0.1067)  evaluator_time: 0.0040 (0.0063)  time: 0.1302  data: 0.0191  max mem: 3748
Test: Total time: 0:00:06 (0.1327 s / it)
Averaged stats: model_time: 0.0976 (0.1067)  evaluator_time: 0.0040 (0.0063)
Accumulating evaluation results...
DONE (t=0.01s).
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.829
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.988
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.950
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.413
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.641
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.847
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.859
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.859
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.874
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.754
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.978
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.887
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.385
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.546
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.769
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.333
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.784
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.784
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.792
Model 1 Epoch: 8
Epoch: [8]  [ 0/60]  eta: 0:00:31  lr: 0.000050  loss: 0.1698 (0.1698)  loss_classifier: 0.0236 (0.0236)  loss_box_reg: 0.0266 (0.0266)  loss_mask: 0.1169 (0.1169)  loss_ob
jectness: 0.0003 (0.0003)  loss_rpn_box_reg: 0.0024 (0.0024)  time: 0.5304  data: 0.0425  max mem: 3748
Epoch: [8]  [10/60]  eta: 0:00:29  lr: 0.000050  loss: 0.1698 (0.1759)  loss_classifier: 0.0217 (0.0220)  loss_box_reg: 0.0330 (0.0352)  loss_mask: 0.1169 (0.1159)  loss_ob
jectness: 0.0003 (0.0008)  loss_rpn_box_reg: 0.0018 (0.0020)  time: 0.5902  data: 0.0359  max mem: 3748
Epoch: [8]  [20/60]  eta: 0:00:24  lr: 0.000050  loss: 0.1806 (0.1819)  loss_classifier: 0.0219 (0.0245)  loss_box_reg: 0.0330 (0.0391)  loss_mask: 0.1092 (0.1150)  loss_ob
jectness: 0.0003 (0.0010)  loss_rpn_box_reg: 0.0018 (0.0024)  time: 0.6041  data: 0.0398  max mem: 3748
Epoch: [8]  [30/60]  eta: 0:00:18  lr: 0.000050  loss: 0.1670 (0.1746)  loss_classifier: 0.0226 (0.0235)  loss_box_reg: 0.0266 (0.0363)  loss_mask: 0.1026 (0.1115)  loss_ob
jectness: 0.0002 (0.0009)  loss_rpn_box_reg: 0.0025 (0.0024)  time: 0.6140  data: 0.0411  max mem: 3748
Epoch: [8]  [40/60]  eta: 0:00:12  lr: 0.000050  loss: 0.1483 (0.1742)  loss_classifier: 0.0201 (0.0235)  loss_box_reg: 0.0257 (0.0365)  loss_mask: 0.1026 (0.1111)  loss_ob
jectness: 0.0002 (0.0008)  loss_rpn_box_reg: 0.0017 (0.0024)  time: 0.6188  data: 0.0392  max mem: 3748
Epoch: [8]  [50/60]  eta: 0:00:06  lr: 0.000050  loss: 0.1573 (0.1716)  loss_classifier: 0.0160 (0.0228)  loss_box_reg: 0.0240 (0.0349)  loss_mask: 0.1080 (0.1108)  loss_ob
jectness: 0.0002 (0.0007)  loss_rpn_box_reg: 0.0018 (0.0024)  time: 0.5940  data: 0.0368  max mem: 3748
Epoch: [8]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.1598 (0.1730)  loss_classifier: 0.0234 (0.0240)  loss_box_reg: 0.0267 (0.0351)  loss_mask: 0.1055 (0.1107)  loss_ob
jectness: 0.0004 (0.0008)  loss_rpn_box_reg: 0.0021 (0.0025)  time: 0.5995  data: 0.0352  max mem: 4110
Epoch: [8] Total time: 0:00:36 (0.6078 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1705 (0.1705)  evaluator_time: 0.0090 (0.0090)  time: 0.1971  data: 0.0168  max mem: 4110
Test:  [49/50]  eta: 0:00:00  model_time: 0.0985 (0.1068)  evaluator_time: 0.0040 (0.0052)  time: 0.1266  data: 0.0168  max mem: 4110
Test: Total time: 0:00:06 (0.1301 s / it)
Averaged stats: model_time: 0.0985 (0.1068)  evaluator_time: 0.0040 (0.0052)
Accumulating evaluation results...
DONE (t=0.01s).
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.829
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.988
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.950
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.413
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.641
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.847
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.860
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.860
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.875
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.754
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.978
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.899
```

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.385
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.539
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.769
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.331
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.783
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.783
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.744
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.791
Model 1 Epoch: 9
Epoch: [9]  [ 0/60]  eta: 0:00:36  lr: 0.000005  loss: 0.1867 (0.1867)  loss_classifier: 0.0263 (0.0263)  loss_box_reg: 0.0442 (0.0442)  loss_mask: 0.1138 (0.1138)  loss_ob
jectness: 0.0003 (0.0003)  loss_rpn_box_reg: 0.0021 (0.0021)  time: 0.6014  data: 0.0356  max mem: 4110
Epoch: [9]  [10/60]  eta: 0:00:30  lr: 0.000005  loss: 0.1539 (0.1587)  loss_classifier: 0.0241 (0.0209)  loss_box_reg: 0.0286 (0.0302)  loss_mask: 0.1004 (0.1047)  loss_ob
jectness: 0.0004 (0.0008)  loss_rpn_box_reg: 0.0020 (0.0021)  time: 0.6057  data: 0.0414  max mem: 4110
Epoch: [9]  [20/60]  eta: 0:00:23  lr: 0.000005  loss: 0.1513 (0.1632)  loss_classifier: 0.0174 (0.0212)  loss_box_reg: 0.0232 (0.0310)  loss_mask: 0.1020 (0.1082)  loss_ob
jectness: 0.0003 (0.0006)  loss_rpn_box_reg: 0.0019 (0.0021)  time: 0.5890  data: 0.0369  max mem: 4110
Epoch: [9]  [30/60]  eta: 0:00:17  lr: 0.000005  loss: 0.1691 (0.1729)  loss_classifier: 0.0248 (0.0228)  loss_box_reg: 0.0368 (0.0339)  loss_mask: 0.1126 (0.1133)  loss_ob
jectness: 0.0004 (0.0007)  loss_rpn_box_reg: 0.0023 (0.0023)  time: 0.5811  data: 0.0346  max mem: 4110
Epoch: [9]  [40/60]  eta: 0:00:12  lr: 0.000005  loss: 0.1996 (0.1778)  loss_classifier: 0.0282 (0.0243)  loss_box_reg: 0.0393 (0.0364)  loss_mask: 0.1126 (0.1139)  loss_ob
jectness: 0.0004 (0.0007)  loss_rpn_box_reg: 0.0023 (0.0025)  time: 0.6141  data: 0.0396  max mem: 4110
Epoch: [9]  [50/60]  eta: 0:00:05  lr: 0.000005  loss: 0.1686 (0.1771)  loss_classifier: 0.0267 (0.0235)  loss_box_reg: 0.0333 (0.0359)  loss_mask: 0.1075 (0.1146)  loss_ob
jectness: 0.0004 (0.0008)  loss_rpn_box_reg: 0.0020 (0.0023)  time: 0.6066  data: 0.0383  max mem: 4110
Epoch: [9]  [59/60]  eta: 0:00:00  lr: 0.000005  loss: 0.1673 (0.1787)  loss_classifier: 0.0214 (0.0241)  loss_box_reg: 0.0308 (0.0371)  loss_mask: 0.1094 (0.1142)  loss_ob
jectness: 0.0005 (0.0008)  loss_rpn_box_reg: 0.0021 (0.0025)  time: 0.6094  data: 0.0357  max mem: 4110
Epoch: [9] Total time: 0:00:36 (0.6017 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1697 (0.1697)  evaluator_time: 0.0051 (0.0051)  time: 0.1918  data: 0.0162  max mem: 4110
Test:  [49/50]  eta: 0:00:00  model_time: 0.0994 (0.1072)  evaluator_time: 0.0041 (0.0064)  time: 0.1346  data: 0.0209  max mem: 4110
Test: Total time: 0:00:06 (0.1336 s / it)
Averaged stats: model_time: 0.0994 (0.1072)  evaluator_time: 0.0041 (0.0064)
Accumulating evaluation results...
DONE (t=0.01s).
Accumulating evaluation results...
DONE (t=0.01s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.829
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.988
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.950
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.413
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.641
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.847
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.860
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.860
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.767
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.875
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.751
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.978
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.899
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.385
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.539
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.766
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.330
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.780
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.780
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.744
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.789


That's it! Model 1
```

In [14]:

```
#MODEL_OPTION2

model_option2 = get_model_instance_segmentation_option2(num_classes)
model_option2.to(device)

# construct an optimizer for option2
params = [p for p in model_option2.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,
momentum=0.9, weight_decay=0.0005)

# and a learning rate scheduler for the same
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
step_size=3,
gamma=0.1)

print ("MODEL OPTION 2")

for epoch in range(num_epochs):

    print("Model 2 Epoch: {}".format(epoch))

    # train for one epoch, printing every 10 iterations
    train_one_epoch(model_option2, optimizer, data_loader, device, epoch, print_freq=10)

    # update the learning rate
    lr_scheduler.step()

    # evaluate on the test dataset
    evaluate(model_option2, data_loader_test, device=device)

print ("\n")
print("That's it! Model 2")
```

```
Downloading: "https://download.pytorch.org/models/mobilenet_v2-7ebf99e0.pth" to /root/.cache/torch/hub/checkpoints/mobilenet_v2-7ebf99e0.pth
  0%|          | 0.00/13.6M [00:00<?, ?B/s]
```

```
MODEL OPTION 2
Model 2 Epoch: 0
Epoch: [0]  [ 0/60]  eta: 0:00:36  lr: 0.000090  loss: 1.5483 (1.5483)  loss_classifier: 0.7352 (0.7352)  loss_box_reg: 0.0793 (0.0793)  loss_objectness: 0.7081 (0.7081)  l
oss_rpn_box_reg: 0.0257 (0.0257)  time: 0.6098  data: 0.0252  max mem: 4110
Epoch: [0]  [10/60]  eta: 0:00:20  lr: 0.000936  loss: 1.4805 (1.4755)  loss_classifier: 0.6844 (0.6521)  loss_box_reg: 0.0793 (0.0757)  loss_objectness: 0.6987 (0.6954)  l
oss_rpn_box_reg: 0.0506 (0.0523)  time: 0.4007  data: 0.0353  max mem: 5714
Epoch: [0]  [20/60]  eta: 0:00:15  lr: 0.001783  loss: 1.1994 (1.2825)  loss_classifier: 0.4023 (0.4896)  loss_box_reg: 0.0807 (0.0973)  loss_objectness: 0.6620 (0.6526)  l
oss_rpn_box_reg: 0.0349 (0.0430)  time: 0.3801  data: 0.0346  max mem: 6171
Epoch: [0]  [30/60]  eta: 0:00:11  lr: 0.002629  loss: 1.0186 (1.1833)  loss_classifier: 0.2883 (0.4310)  loss_box_reg: 0.1316 (0.1218)  loss_objectness: 0.5236 (0.5883)  l
oss_rpn_box_reg: 0.0349 (0.0421)  time: 0.3802  data: 0.0377  max mem: 6171
Epoch: [0]  [40/60]  eta: 0:00:07  lr: 0.003476  loss: 0.8882 (1.1042)  loss_classifier: 0.2730 (0.3972)  loss_box_reg: 0.1691 (0.1437)  loss_objectness: 0.3652 (0.5223)  l
oss_rpn_box_reg: 0.0381 (0.0410)  time: 0.3797  data: 0.0381  max mem: 6171
Epoch: [0]  [50/60]  eta: 0:00:03  lr: 0.004323  loss: 0.7166 (1.0127)  loss_classifier: 0.2349 (0.3635)  loss_box_reg: 0.1691 (0.1471)  loss_objectness: 0.2655 (0.4636)  l
oss_rpn_box_reg: 0.0324 (0.0384)  time: 0.3751  data: 0.0342  max mem: 6171
Epoch: [0]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.5928 (0.9590)  loss_classifier: 0.2266 (0.3439)  loss_box_reg: 0.1759 (0.1565)  loss_objectness: 0.1994 (0.4216)  l
oss_rpn_box_reg: 0.0231 (0.0370)  time: 0.3756  data: 0.0379  max mem: 6171
Epoch: [0] Total time: 0:00:22 (0.3823 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0546 (0.0546)  evaluator_time: 0.0107 (0.0107)  time: 0.0823  data: 0.0163  max mem: 6171
Test:  [49/50]  eta: 0:00:00  model_time: 0.0328 (0.0359)  evaluator_time: 0.0025 (0.0035)  time: 0.0537  data: 0.0166  max mem: 6171
Test: Total time: 0:00:02 (0.0566 s / it)
Averaged stats: model_time: 0.0328 (0.0359)  evaluator_time: 0.0025 (0.0035)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.033
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.105
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.004
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.097
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.036
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.208
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.337
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.033
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.370
Model 2 Epoch: 1
Epoch: [1]  [ 0/60]  eta: 0:00:22  lr: 0.005000  loss: 0.5127 (0.5127)  loss_classifier: 0.1759 (0.1759)  loss_box_reg: 0.1739 (0.1739)  loss_objectness: 0.1374 (0.1374)  l
oss_rpn_box_reg: 0.0256 (0.0256)  time: 0.3745  data: 0.0383  max mem: 6171
Epoch: [1]  [10/60]  eta: 0:00:18  lr: 0.005000  loss: 0.5127 (0.5067)  loss_classifier: 0.1759 (0.1746)  loss_box_reg: 0.1603 (0.1513)  loss_objectness: 0.1390 (0.1538)  l
oss_rpn_box_reg: 0.0284 (0.0270)  time: 0.3690  data: 0.0334  max mem: 6171
Epoch: [1]  [20/60]  eta: 0:00:14  lr: 0.005000  loss: 0.4856 (0.5248)  loss_classifier: 0.1535 (0.1770)  loss_box_reg: 0.1603 (0.1709)  loss_objectness: 0.1360 (0.1477)  l
oss_rpn_box_reg: 0.0284 (0.0292)  time: 0.3737  data: 0.0333  max mem: 6171
Epoch: [1]  [30/60]  eta: 0:00:11  lr: 0.005000  loss: 0.4856 (0.5197)  loss_classifier: 0.1565 (0.1767)  loss_box_reg: 0.1639 (0.1730)  loss_objectness: 0.1248 (0.1423)  l
oss_rpn_box_reg: 0.0244 (0.0277)  time: 0.3826  data: 0.0362  max mem: 6171
Epoch: [1]  [40/60]  eta: 0:00:07  lr: 0.005000  loss: 0.4909 (0.5159)  loss_classifier: 0.1580 (0.1708)  loss_box_reg: 0.1813 (0.1795)  loss_objectness: 0.1248 (0.1374)  l
oss_rpn_box_reg: 0.0235 (0.0282)  time: 0.3854  data: 0.0378  max mem: 6171
Epoch: [1]  [50/60]  eta: 0:00:03  lr: 0.005000  loss: 0.4838 (0.5030)  loss_classifier: 0.1422 (0.1662)  loss_box_reg: 0.1787 (0.1787)  loss_objectness: 0.1128 (0.1309)  l
oss_rpn_box_reg: 0.0235 (0.0273)  time: 0.3765  data: 0.0347  max mem: 6171
Epoch: [1]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.4595 (0.5034)  loss_classifier: 0.1332 (0.1642)  loss_box_reg: 0.1787 (0.1828)  loss_objectness: 0.1061 (0.1283)  l
oss_rpn_box_reg: 0.0236 (0.0281)  time: 0.3814  data: 0.0405  max mem: 6171
Epoch: [1] Total time: 0:00:22 (0.3798 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0600 (0.0600)  evaluator_time: 0.0039 (0.0039)  time: 0.0827  data: 0.0181  max mem: 6171
Test:  [49/50]  eta: 0:00:00  model_time: 0.0363 (0.0379)  evaluator_time: 0.0020 (0.0027)  time: 0.0566  data: 0.0168  max mem: 6171
Test: Total time: 0:00:02 (0.0585 s / it)
Averaged stats: model_time: 0.0363 (0.0379)  evaluator_time: 0.0020 (0.0027)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.158
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.510
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.055
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.176
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.116
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.339
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.370
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.409
Model 2 Epoch: 2
Epoch: [2]  [ 0/60]  eta: 0:00:24  lr: 0.005000  loss: 0.5657 (0.5657)  loss_classifier: 0.1616 (0.1616)  loss_box_reg: 0.2584 (0.2584)  loss_objectness: 0.1075 (0.1075)  l
oss_rpn_box_reg: 0.0383 (0.0383)  time: 0.4106  data: 0.0463  max mem: 6171
Epoch: [2]  [10/60]  eta: 0:00:19  lr: 0.005000  loss: 0.3937 (0.4336)  loss_classifier: 0.1302 (0.1313)  loss_box_reg: 0.1717 (0.1799)  loss_objectness: 0.0883 (0.0938)  l
oss_rpn_box_reg: 0.0226 (0.0286)  time: 0.3803  data: 0.0364  max mem: 6171
Epoch: [2]  [20/60]  eta: 0:00:15  lr: 0.005000  loss: 0.3937 (0.4440)  loss_classifier: 0.1302 (0.1346)  loss_box_reg: 0.1717 (0.1912)  loss_objectness: 0.0815 (0.0914)  l
oss_rpn_box_reg: 0.0189 (0.0266)  time: 0.3802  data: 0.0395  max mem: 6171
Epoch: [2]  [30/60]  eta: 0:00:11  lr: 0.005000  loss: 0.4298 (0.4288)  loss_classifier: 0.1327 (0.1300)  loss_box_reg: 0.2017 (0.1866)  loss_objectness: 0.0814 (0.0868)  l
oss_rpn_box_reg: 0.0234 (0.0255)  time: 0.3804  data: 0.0395  max mem: 6264
Epoch: [2]  [40/60]  eta: 0:00:07  lr: 0.005000  loss: 0.4020 (0.4208)  loss_classifier: 0.1207 (0.1266)  loss_box_reg: 0.1513 (0.1819)  loss_objectness: 0.0774 (0.0865)  l
oss_rpn_box_reg: 0.0253 (0.0258)  time: 0.3727  data: 0.0346  max mem: 6264
Epoch: [2]  [50/60]  eta: 0:00:03  lr: 0.005000  loss: 0.3682 (0.4048)  loss_classifier: 0.1056 (0.1216)  loss_box_reg: 0.1464 (0.1741)  loss_objectness: 0.0764 (0.0835)  l
oss_rpn_box_reg: 0.0227 (0.0255)  time: 0.3695  data: 0.0342  max mem: 6264
Epoch: [2]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.3682 (0.4099)  loss_classifier: 0.1056 (0.1238)  loss_box_reg: 0.1556 (0.1793)  loss_objectness: 0.0668 (0.0812)  l
oss_rpn_box_reg: 0.0215 (0.0255)  time: 0.3782  data: 0.0391  max mem: 6264
Epoch: [2] Total time: 0:00:22 (0.3782 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0564 (0.0564)  evaluator_time: 0.0048 (0.0048)  time: 0.0909  data: 0.0288  max mem: 6264
Test:  [49/50]  eta: 0:00:00  model_time: 0.0338 (0.0363)  evaluator_time: 0.0037 (0.0043)  time: 0.0618  data: 0.0224  max mem: 6264
Test: Total time: 0:00:03 (0.0639 s / it)
Averaged stats: model_time: 0.0338 (0.0363)  evaluator_time: 0.0037 (0.0043)
Accumulating evaluation results...
DONE (t=0.03s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.206
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.552
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.063
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.228
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.141
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.381
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.412
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.455
Model 2 Epoch: 3
Epoch: [3]  [ 0/60]  eta: 0:00:22  lr: 0.000500  loss: 0.4481 (0.4481)  loss_classifier: 0.1347 (0.1347)  loss_box_reg: 0.2129 (0.2129)  loss_objectness: 0.0810 (0.0810)  l
oss_rpn_box_reg: 0.0195 (0.0195)  time: 0.3754  data: 0.0440  max mem: 6264
```

```
Epoch: [3]  [10/60]  eta: 0:00:19  lr: 0.000500  loss: 0.3439 (0.3628)  loss_classifier: 0.1153 (0.1113)  loss_box_reg: 0.1754 (0.1707)  loss_objectness: 0.0569 (0.0573)  l
oss_rpn_box_reg: 0.0234 (0.0235)  time: 0.3869  data: 0.0380  max mem: 6264
Epoch: [3]  [20/60]  eta: 0:00:15  lr: 0.000500  loss: 0.3216 (0.3389)  loss_classifier: 0.0992 (0.1041)  loss_box_reg: 0.1535 (0.1563)  loss_objectness: 0.0542 (0.0577)  l
oss_rpn_box_reg: 0.0201 (0.0208)  time: 0.3821  data: 0.0383  max mem: 6264
Epoch: [3]  [30/60]  eta: 0:00:11  lr: 0.000500  loss: 0.3216 (0.3517)  loss_classifier: 0.1030 (0.1079)  loss_box_reg: 0.1612 (0.1644)  loss_objectness: 0.0571 (0.0585)  l
oss_rpn_box_reg: 0.0184 (0.0209)  time: 0.3786  data: 0.0372  max mem: 6264
Epoch: [3]  [40/60]  eta: 0:00:07  lr: 0.000500  loss: 0.3549 (0.3523)  loss_classifier: 0.1067 (0.1080)  loss_box_reg: 0.1705 (0.1640)  loss_objectness: 0.0620 (0.0580)  l
oss_rpn_box_reg: 0.0228 (0.0224)  time: 0.3842  data: 0.0362  max mem: 6264
Epoch: [3]  [50/60]  eta: 0:00:03  lr: 0.000500  loss: 0.3358 (0.3526)  loss_classifier: 0.1067 (0.1090)  loss_box_reg: 0.1589 (0.1649)  loss_objectness: 0.0577 (0.0572)  l
oss_rpn_box_reg: 0.0208 (0.0215)  time: 0.3863  data: 0.0402  max mem: 6264
Epoch: [3]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.3065 (0.3492)  loss_classifier: 0.1011 (0.1070)  loss_box_reg: 0.1562 (0.1631)  loss_objectness: 0.0480 (0.0581)  l
oss_rpn_box_reg: 0.0183 (0.0211)  time: 0.3807  data: 0.0394  max mem: 6264
Epoch: [3] Total time: 0:00:22 (0.3819 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:03  model_time: 0.0543 (0.0543)  evaluator_time: 0.0025 (0.0025)  time: 0.0744  data: 0.0169  max mem: 6264
Test:  [49/50]  eta: 0:00:00  model_time: 0.0326 (0.0352)  evaluator_time: 0.0017 (0.0019)  time: 0.0515  data: 0.0161  max mem: 6264
Test: Total time: 0:00:02 (0.0540 s / it)
Averaged stats: model_time: 0.0326 (0.0352)  evaluator_time: 0.0017 (0.0019)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.222
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.599
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.084
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.246
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.157
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.386
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.414
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.457
Model 2 Epoch: 4
Epoch: [4]  [ 0/60]  eta: 0:00:23  lr: 0.000500  loss: 0.3060 (0.3060)  loss_classifier: 0.0976 (0.0976)  loss_box_reg: 0.1501 (0.1501)  loss_objectness: 0.0435 (0.0435)  l
oss_rpn_box_reg: 0.0148 (0.0148)  time: 0.3883  data: 0.0459  max mem: 6264
Epoch: [4]  [10/60]  eta: 0:00:19  lr: 0.000500  loss: 0.3060 (0.3268)  loss_classifier: 0.0976 (0.1039)  loss_box_reg: 0.1409 (0.1477)  loss_objectness: 0.0590 (0.0565)  l
oss_rpn_box_reg: 0.0148 (0.0188)  time: 0.3977  data: 0.0474  max mem: 6264
Epoch: [4]  [20/60]  eta: 0:00:15  lr: 0.000500  loss: 0.3483 (0.3647)  loss_classifier: 0.1080 (0.1158)  loss_box_reg: 0.1570 (0.1722)  loss_objectness: 0.0538 (0.0567)  l
oss_rpn_box_reg: 0.0173 (0.0201)  time: 0.3888  data: 0.0413  max mem: 6264
Epoch: [4]  [30/60]  eta: 0:00:11  lr: 0.000500  loss: 0.3649 (0.3614)  loss_classifier: 0.1056 (0.1116)  loss_box_reg: 0.1827 (0.1712)  loss_objectness: 0.0523 (0.0576)  l
oss_rpn_box_reg: 0.0194 (0.0211)  time: 0.3780  data: 0.0349  max mem: 6264
Epoch: [4]  [40/60]  eta: 0:00:07  lr: 0.000500  loss: 0.3387 (0.3632)  loss_classifier: 0.0948 (0.1108)  loss_box_reg: 0.1673 (0.1717)  loss_objectness: 0.0498 (0.0586)  l
oss_rpn_box_reg: 0.0232 (0.0222)  time: 0.3841  data: 0.0390  max mem: 6264
Epoch: [4]  [50/60]  eta: 0:00:03  lr: 0.000500  loss: 0.3387 (0.3586)  loss_classifier: 0.1016 (0.1096)  loss_box_reg: 0.1673 (0.1700)  loss_objectness: 0.0504 (0.0577)  l
oss_rpn_box_reg: 0.0192 (0.0213)  time: 0.3847  data: 0.0403  max mem: 6264
Epoch: [4]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.2903 (0.3500)  loss_classifier: 0.0981 (0.1075)  loss_box_reg: 0.1418 (0.1655)  loss_objectness: 0.0504 (0.0563)  l
oss_rpn_box_reg: 0.0150 (0.0207)  time: 0.3776  data: 0.0356  max mem: 6264
Epoch: [4] Total time: 0:00:22 (0.3829 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:03  model_time: 0.0543 (0.0543)  evaluator_time: 0.0024 (0.0024)  time: 0.0760  data: 0.0186  max mem: 6264
Test:  [49/50]  eta: 0:00:00  model_time: 0.0343 (0.0363)  evaluator_time: 0.0033 (0.0050)  time: 0.0662  data: 0.0219  max mem: 6264
Test: Total time: 0:00:03 (0.0621 s / it)
Averaged stats: model_time: 0.0343 (0.0363)  evaluator_time: 0.0033 (0.0050)
Accumulating evaluation results...
DONE (t=0.03s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.279
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.645
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.157
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.002
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.308
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.182
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.429
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.454
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.067
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.497
Model 2 Epoch: 5
Epoch: [5]  [ 0/60]  eta: 0:00:23  lr: 0.000500  loss: 0.6110 (0.6110)  loss_classifier: 0.1749 (0.1749)  loss_box_reg: 0.2841 (0.2841)  loss_objectness: 0.1050 (0.1050)  l
oss_rpn_box_reg: 0.0470 (0.0470)  time: 0.3990  data: 0.0475  max mem: 6264
Epoch: [5]  [10/60]  eta: 0:00:18  lr: 0.000500  loss: 0.3202 (0.3145)  loss_classifier: 0.1059 (0.0987)  loss_box_reg: 0.1407 (0.1428)  loss_objectness: 0.0445 (0.0554)  l
oss_rpn_box_reg: 0.0142 (0.0177)  time: 0.3774  data: 0.0368  max mem: 6264
Epoch: [5]  [20/60]  eta: 0:00:15  lr: 0.000500  loss: 0.3202 (0.3502)  loss_classifier: 0.1026 (0.1043)  loss_box_reg: 0.1407 (0.1639)  loss_objectness: 0.0498 (0.0616)  l
oss_rpn_box_reg: 0.0161 (0.0204)  time: 0.3790  data: 0.0359  max mem: 6264
Epoch: [5]  [30/60]  eta: 0:00:11  lr: 0.000500  loss: 0.3115 (0.3456)  loss_classifier: 0.1009 (0.1048)  loss_box_reg: 0.1383 (0.1634)  loss_objectness: 0.0521 (0.0579)  l
oss_rpn_box_reg: 0.0183 (0.0196)  time: 0.3765  data: 0.0356  max mem: 6264
Epoch: [5]  [40/60]  eta: 0:00:07  lr: 0.000500  loss: 0.3016 (0.3376)  loss_classifier: 0.0947 (0.1021)  loss_box_reg: 0.1300 (0.1581)  loss_objectness: 0.0525 (0.0580)  l
oss_rpn_box_reg: 0.0164 (0.0194)  time: 0.3746  data: 0.0389  max mem: 6264
Epoch: [5]  [50/60]  eta: 0:00:03  lr: 0.000500  loss: 0.3181 (0.3369)  loss_classifier: 0.0858 (0.1008)  loss_box_reg: 0.1355 (0.1588)  loss_objectness: 0.0574 (0.0569)  l
oss_rpn_box_reg: 0.0168 (0.0203)  time: 0.3805  data: 0.0384  max mem: 6264
Epoch: [5]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.3293 (0.3430)  loss_classifier: 0.0954 (0.1028)  loss_box_reg: 0.1463 (0.1619)  loss_objectness: 0.0472 (0.0579)  l
oss_rpn_box_reg: 0.0162 (0.0204)  time: 0.3846  data: 0.0346  max mem: 6698
Epoch: [5] Total time: 0:00:22 (0.3801 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0558 (0.0558)  evaluator_time: 0.0051 (0.0051)  time: 0.0865  data: 0.0248  max mem: 6698
Test:  [49/50]  eta: 0:00:00  model_time: 0.0344 (0.0365)  evaluator_time: 0.0020 (0.0034)  time: 0.0565  data: 0.0183  max mem: 6698
Test: Total time: 0:00:03 (0.0614 s / it)
Averaged stats: model_time: 0.0344 (0.0365)  evaluator_time: 0.0020 (0.0034)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.244
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.621
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.106
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.004
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.269
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.159
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.394
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.418
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.067
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.457
Model 2 Epoch: 6
Epoch: [6]  [ 0/60]  eta: 0:00:20  lr: 0.000050  loss: 0.3441 (0.3441)  loss_classifier: 0.1117 (0.1117)  loss_box_reg: 0.1667 (0.1667)  loss_objectness: 0.0453 (0.0453)  l
oss_rpn_box_reg: 0.0204 (0.0204)  time: 0.3436  data: 0.0268  max mem: 6698
Epoch: [6]  [10/60]  eta: 0:00:18  lr: 0.000050  loss: 0.3446 (0.3596)  loss_classifier: 0.1105 (0.1085)  loss_box_reg: 0.1865 (0.1809)  loss_objectness: 0.0481 (0.0506)  l
oss_rpn_box_reg: 0.0204 (0.0197)  time: 0.3729  data: 0.0367  max mem: 6698
Epoch: [6]  [20/60]  eta: 0:00:15  lr: 0.000050  loss: 0.2979 (0.3221)  loss_classifier: 0.0930 (0.0967)  loss_box_reg: 0.1443 (0.1573)  loss_objectness: 0.0454 (0.0483)  l
oss_rpn_box_reg: 0.0179 (0.0197)  time: 0.3775  data: 0.0367  max mem: 6698
```

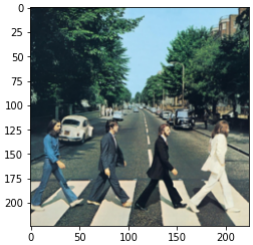
```
Epoch: [6]  [30/60]  eta: 0:00:11  lr: 0.000050  loss: 0.2815 (0.3187)  loss_classifier: 0.0869 (0.0985)  loss_box_reg: 0.1379 (0.1533)  loss_objectness: 0.0386 (0.0475)  l
oss_rpn_box_reg: 0.0167 (0.0194)  time: 0.3818  data: 0.0380  max mem: 6698
Epoch: [6]  [40/60]  eta: 0:00:07  lr: 0.000050  loss: 0.2880 (0.3140)  loss_classifier: 0.0921 (0.0972)  loss_box_reg: 0.1379 (0.1516)  loss_objectness: 0.0386 (0.0470)  l
oss_rpn_box_reg: 0.0147 (0.0183)  time: 0.3778  data: 0.0371  max mem: 6698
Epoch: [6]  [50/60]  eta: 0:00:03  lr: 0.000050  loss: 0.2981 (0.3280)  loss_classifier: 0.0908 (0.1001)  loss_box_reg: 0.1379 (0.1578)  loss_objectness: 0.0450 (0.0508)  l
oss_rpn_box_reg: 0.0191 (0.0193)  time: 0.3813  data: 0.0365  max mem: 6698
Epoch: [6]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.3217 (0.3301)  loss_classifier: 0.1031 (0.1014)  loss_box_reg: 0.1474 (0.1591)  loss_objectness: 0.0502 (0.0502)  l
oss_rpn_box_reg: 0.0192 (0.0194)  time: 0.3892  data: 0.0359  max mem: 6698
Epoch: [6] Total time: 0:00:22 (0.3816 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:03  model_time: 0.0550 (0.0550)  evaluator_time: 0.0026 (0.0026)  time: 0.0791  data: 0.0207  max mem: 6698
Test:  [49/50]  eta: 0:00:00  model_time: 0.0347 (0.0363)  evaluator_time: 0.0018 (0.0021)  time: 0.0548  data: 0.0173  max mem: 6698
Test: Total time: 0:00:02 (0.0565 s / it)
Averaged stats: model_time: 0.0347 (0.0363)  evaluator_time: 0.0018 (0.0021)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.248
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.654
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.074
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.273
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.156
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.404
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.432
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.056
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.473
Model 2 Epoch: 7
Epoch: [7]  [ 0/60]  eta: 0:00:21  lr: 0.000050  loss: 0.2458 (0.2458)  loss_classifier: 0.0645 (0.0645)  loss_box_reg: 0.1382 (0.1382)  loss_objectness: 0.0258 (0.0258)  l
oss_rpn_box_reg: 0.0173 (0.0173)  time: 0.3662  data: 0.0361  max mem: 6698
Epoch: [7]  [10/60]  eta: 0:00:18  lr: 0.000050  loss: 0.2629 (0.3183)  loss_classifier: 0.0960 (0.0969)  loss_box_reg: 0.1356 (0.1473)  loss_objectness: 0.0418 (0.0530)  l
oss_rpn_box_reg: 0.0176 (0.0211)  time: 0.3693  data: 0.0323  max mem: 6698
Epoch: [7]  [20/60]  eta: 0:00:15  lr: 0.000050  loss: 0.3018 (0.3290)  loss_classifier: 0.0960 (0.1004)  loss_box_reg: 0.1356 (0.1556)  loss_objectness: 0.0425 (0.0525)  l
oss_rpn_box_reg: 0.0190 (0.0206)  time: 0.3780  data: 0.0356  max mem: 6698
Epoch: [7]  [30/60]  eta: 0:00:11  lr: 0.000050  loss: 0.3247 (0.3496)  loss_classifier: 0.0976 (0.1062)  loss_box_reg: 0.1588 (0.1684)  loss_objectness: 0.0499 (0.0544)  l
oss_rpn_box_reg: 0.0190 (0.0207)  time: 0.3895  data: 0.0439  max mem: 6698
Epoch: [7]  [40/60]  eta: 0:00:07  lr: 0.000050  loss: 0.3315 (0.3473)  loss_classifier: 0.0988 (0.1055)  loss_box_reg: 0.1758 (0.1672)  loss_objectness: 0.0500 (0.0539)  l
oss_rpn_box_reg: 0.0204 (0.0207)  time: 0.3878  data: 0.0421  max mem: 6698
Epoch: [7]  [50/60]  eta: 0:00:03  lr: 0.000050  loss: 0.3005 (0.3331)  loss_classifier: 0.0882 (0.1012)  loss_box_reg: 0.1401 (0.1601)  loss_objectness: 0.0481 (0.0524)  l
oss_rpn_box_reg: 0.0173 (0.0194)  time: 0.3751  data: 0.0333  max mem: 6698
Epoch: [7]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.2915 (0.3343)  loss_classifier: 0.0870 (0.1021)  loss_box_reg: 0.1539 (0.1606)  loss_objectness: 0.0485 (0.0523)  l
oss_rpn_box_reg: 0.0156 (0.0194)  time: 0.3839  data: 0.0375  max mem: 6698
Epoch: [7] Total time: 0:00:22 (0.3824 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:03  model_time: 0.0543 (0.0543)  evaluator_time: 0.0025 (0.0025)  time: 0.0749  data: 0.0173  max mem: 6698
Test:  [49/50]  eta: 0:00:00  model_time: 0.0332 (0.0362)  evaluator_time: 0.0018 (0.0022)  time: 0.0539  data: 0.0171  max mem: 6698
Test: Total time: 0:00:02 (0.0563 s / it)
Averaged stats: model_time: 0.0332 (0.0362)  evaluator_time: 0.0018 (0.0022)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.268
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.640
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.166
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.004
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.296
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.181
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.433
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.459
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.056
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.503
Model 2 Epoch: 8
Epoch: [8]  [ 0/60]  eta: 0:00:22  lr: 0.000050  loss: 0.4425 (0.4425)  loss_classifier: 0.1317 (0.1317)  loss_box_reg: 0.1979 (0.1979)  loss_objectness: 0.0853 (0.0853)  l
oss_rpn_box_reg: 0.0276 (0.0276)  time: 0.3714  data: 0.0305  max mem: 6698
Epoch: [8]  [10/60]  eta: 0:00:18  lr: 0.000050  loss: 0.3537 (0.3539)  loss_classifier: 0.1093 (0.1074)  loss_box_reg: 0.1728 (0.1715)  loss_objectness: 0.0553 (0.0527)  l
oss_rpn_box_reg: 0.0218 (0.0222)  time: 0.3745  data: 0.0329  max mem: 6698
Epoch: [8]  [20/60]  eta: 0:00:15  lr: 0.000050  loss: 0.3457 (0.3768)  loss_classifier: 0.1065 (0.1140)  loss_box_reg: 0.1633 (0.1859)  loss_objectness: 0.0543 (0.0545)  l
oss_rpn_box_reg: 0.0191 (0.0225)  time: 0.3882  data: 0.0393  max mem: 6698
Epoch: [8]  [30/60]  eta: 0:00:11  lr: 0.000050  loss: 0.2902 (0.3482)  loss_classifier: 0.0924 (0.1072)  loss_box_reg: 0.1319 (0.1685)  loss_objectness: 0.0415 (0.0518)  l
oss_rpn_box_reg: 0.0168 (0.0207)  time: 0.3912  data: 0.0410  max mem: 6698
Epoch: [8]  [40/60]  eta: 0:00:07  lr: 0.000050  loss: 0.2577 (0.3386)  loss_classifier: 0.0835 (0.1037)  loss_box_reg: 0.1242 (0.1621)  loss_objectness: 0.0408 (0.0526)  l
oss_rpn_box_reg: 0.0162 (0.0202)  time: 0.3834  data: 0.0374  max mem: 6698
Epoch: [8]  [50/60]  eta: 0:00:03  lr: 0.000050  loss: 0.2839 (0.3280)  loss_classifier: 0.0816 (0.1001)  loss_box_reg: 0.1317 (0.1564)  loss_objectness: 0.0482 (0.0519)  l
oss_rpn_box_reg: 0.0156 (0.0195)  time: 0.3818  data: 0.0397  max mem: 6698
Epoch: [8]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.2839 (0.3250)  loss_classifier: 0.0880 (0.0991)  loss_box_reg: 0.1423 (0.1543)  loss_objectness: 0.0434 (0.0526)  l
oss_rpn_box_reg: 0.0146 (0.0190)  time: 0.3831  data: 0.0416  max mem: 6698
Epoch: [8] Total time: 0:00:23 (0.3840 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:03  model_time: 0.0550 (0.0550)  evaluator_time: 0.0024 (0.0024)  time: 0.0759  data: 0.0178  max mem: 6698
Test:  [49/50]  eta: 0:00:00  model_time: 0.0337 (0.0360)  evaluator_time: 0.0017 (0.0021)  time: 0.0529  data: 0.0164  max mem: 6698
Test: Total time: 0:00:02 (0.0555 s / it)
Averaged stats: model_time: 0.0337 (0.0360)  evaluator_time: 0.0017 (0.0021)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.274
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.674
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.159
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.002
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.303
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.177
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.402
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.433
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.067
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.473
Model 2 Epoch: 9
Epoch: [9]  [ 0/60]  eta: 0:00:21  lr: 0.000005  loss: 0.2460 (0.2460)  loss_classifier: 0.0772 (0.0772)  loss_box_reg: 0.0851 (0.0851)  loss_objectness: 0.0695 (0.0695)  l
oss_rpn_box_reg: 0.0143 (0.0143)  time: 0.3595  data: 0.0282  max mem: 6698
Epoch: [9]  [10/60]  eta: 0:00:19  lr: 0.000005  loss: 0.3128 (0.3490)  loss_classifier: 0.1043 (0.1090)  loss_box_reg: 0.1557 (0.1661)  loss_objectness: 0.0507 (0.0528)  l
oss_rpn_box_reg: 0.0184 (0.0211)  time: 0.3812  data: 0.0373  max mem: 6698
Epoch: [9]  [20/60]  eta: 0:00:15  lr: 0.000005  loss: 0.3055 (0.3252)  loss_classifier: 0.0996 (0.1000)  loss_box_reg: 0.1423 (0.1527)  loss_objectness: 0.0504 (0.0517)  l
oss_rpn_box_reg: 0.0183 (0.0208)  time: 0.3827  data: 0.0378  max mem: 6698
Epoch: [9]  [30/60]  eta: 0:00:11  lr: 0.000005  loss: 0.3133 (0.3458)  loss_classifier: 0.1029 (0.1053)  loss_box_reg: 0.1559 (0.1669)  loss_objectness: 0.0531 (0.0533)  l
oss_rpn_box_reg: 0.0151 (0.0202)  time: 0.3792  data: 0.0380  max mem: 6698
Epoch: [9]  [40/60]  eta: 0:00:07  lr: 0.000005  loss: 0.3147 (0.3327)  loss_classifier: 0.0928 (0.1006)  loss_box_reg: 0.1422 (0.1570)  loss_objectness: 0.0547 (0.0553)  l
oss_rpn_box_reg: 0.0164 (0.0197)  time: 0.3742  data: 0.0354  max mem: 6698
```

```
Epoch: [9]  [50/60]  eta: 0:00:03  lr: 0.000005  loss: 0.3147 (0.3339)  loss_classifier: 0.0905 (0.1009)  loss_box_reg: 0.1416 (0.1595)  loss_objectness: 0.0440 (0.0539)  l
oss_rpn_box_reg: 0.0171 (0.0196)  time: 0.3785  data: 0.0387  max mem: 6698
Epoch: [9]  [59/60]  eta: 0:00:00  lr: 0.000005  loss: 0.3137 (0.3334)  loss_classifier: 0.0933 (0.1010)  loss_box_reg: 0.1546 (0.1593)  loss_objectness: 0.0440 (0.0534)  l
oss_rpn_box_reg: 0.0176 (0.0197)  time: 0.3840  data: 0.0408  max mem: 6698
Epoch: [9] Total time: 0:00:22 (0.3805 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:03  model_time: 0.0545 (0.0545)  evaluator_time: 0.0024 (0.0024)  time: 0.0752  data: 0.0176  max mem: 6698
Test:  [49/50]  eta: 0:00:00  model_time: 0.0351 (0.0366)  evaluator_time: 0.0019 (0.0022)  time: 0.0553  data: 0.0174  max mem: 6698
Test: Total time: 0:00:02 (0.0572 s / it)
Averaged stats: model_time: 0.0351 (0.0366)  evaluator_time: 0.0019 (0.0022)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.273
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.652
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.135
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.003
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.302
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.184
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.431
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.465
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.089
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.507


That's it! Model 2
```

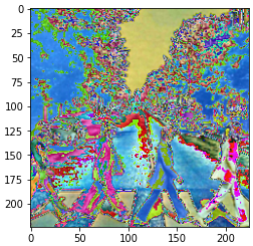
In [15]:

```
import torch
import torchvision
from PIL import Image
import matplotlib.pyplot as plt

url = "https://upload.wikimedia.org/wikipedia/en/4/42/Beatles_-_Abbey_Road.jpg"
img = Image.open(requests.get(url, stream=True).raw)

transform = transforms.Compose([transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    ])

img = transform(img).to(device)

# Plot the original image
plt.imshow(transforms.ToPILImage()(img))
print ("Original Image")
plt.show()

transform = torchvision.transforms.Compose([
    torchvision.transforms.Resize(256),
    torchvision.transforms.CenterCrop(224),

    torchvision.transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
        )
    ])

img = transform(img).to(device)

print("OPTION 1")

# Make a prediction
with torch.no_grad():
    output = model_option1(img.unsqueeze(0))
    print('Prediction for model_option1: \n', output)

# Filter the output to keep only the top scoring boxes
threshold = 0.5
_boxes = output[0]['boxes']

# Draw the boxes on the image
fig, ax = plt.subplots()
ax.imshow(transforms.ToPILImage()(img))

for box in _boxes:
    x1, y1, x2, y2 = box.tolist()
    w, h = x2 - x1, y2 - y1
    rect = plt.Rectangle((x1, y1), w, h, linewidth=1,edgecolor='r', facecolor='none')
    ax.add_patch(rect)
plt.show()

print("OPTION 2")

# Make a prediction
with torch.no_grad():
    output = model_option2(img.unsqueeze(0))
    print('Prediction for model_option2: \n', output)

# Filter the output to keep only the top scoring boxes
threshold = 0.5
_boxes = output[0]['boxes']

# Draw the boxes on the image
fig, ax = plt.subplots()
ax.imshow(transforms.ToPILImage()(img))

for box in _boxes:
    x1, y1, x2, y2 = box.tolist()
    w, h = x2 - x1, y2 - y1
    rect = plt.Rectangle((x1, y1), w, h, linewidth=1,edgecolor='r', facecolor='none')
    ax.add_patch(rect)
plt.show()
```

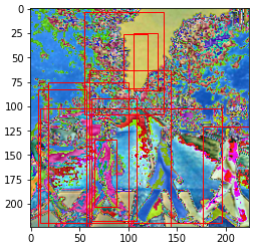
```
Original Image
```



```
OPTION 1
Prediction for model_option1:
 [{'boxes': tensor([], device='cuda:0', size=(0, 4)), 'labels': tensor([], device='cuda:0', dtype=torch.int64), 'scores': tensor([], device='cuda:0'), 'masks': tensor([], d
evice='cuda:0', size=(0, 1, 224, 224))}]
```



```
OPTION 2
Prediction for model_option2:
 [{'boxes': tensor([[ 61.7875,  91.8146, 108.8170, 217.9420],
        [  7.7791,  82.5294,  56.8638, 224.0000],
        [176.3702, 120.4357, 223.9798, 224.0000],
        [ 64.7320, 133.9791,  87.6553, 202.6269],
        [ 60.2962,  63.2871, 143.0251, 224.0000],
        [ 17.7155,  75.2567,  99.8058, 223.7091],
        [ 95.4509,  26.1168, 119.9425,  81.9522],
        [105.6704,  25.2083, 129.5414,  83.2234],
        [  9.1040, 102.0064, 196.3956, 219.8465],
        [ 54.6946,   3.4210, 136.3288, 107.8350]], device='cuda:0'), 'labels': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0'), 'scores': tensor([0.3437, 0.2554, 0.
1789, 0.1712, 0.1502, 0.1249, 0.1001, 0.0676, 0.0614,
        0.0581], device='cuda:0')}]
```



In [15]: