Homework 2

Zhangrui Huang (zh2680@nyu.edu)

Problem 1

(a). $L(w_1, w_2) = 0.5(aw_1^2 + bw_2^2),\ \frac{\partial L}{\partial w_1} = aw_1,\ \frac{\partial L}{\partial w_2} = bw_2,\ \nabla L(w_1, w_2) = [aw_1, bw_2]$
To achieve the minimum value of L, set $\nabla L(w_1, w_2) = [aw_1, bw_2] = 0 \Rightarrow w_1 = 0,\ w_2 = 0$

(b). Gradient descent should be $w(t+1) = w(t) - \eta \frac{\partial L}{\partial w}$, where η is learning rate.
$w_1(t+1) = w_1(t) - \eta \frac{\partial L}{\partial w_1} = w_1(t) - \eta a w_1(t) = (1 - \eta a) w_1(t) \Rightarrow \rho_1 = 1 - \eta a$
$w_2(t+1) = w_2(t) - \eta \frac{\partial L}{\partial w_2} = w_2(t) - \eta b w_2(t) = (1 - \eta b) w_2(t) \Rightarrow \rho_2 = 1 - \eta b$

(c). To converge the result of gradient descent, $-1 < \rho_1, \rho_2 < 1$, so $0 < \eta < \min(\frac{2}{a}, \frac{2}{b})$

(b). a/b is a very large ratio, which means a is much larger than b. In this case even if η is optimally set for w_1, it might be too small for w_2, so the the convergence rate of gradient descent for w_2 is very slow.

Problem 2

(a). The filter should be like

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

This is a vertical Sobel filter, it calculate the difference in intensity between the left and right pixels. Pixels in the middle are weighted more heavily. If there's a strong vertical edge, the difference in intensity will be high, and the output will be high.

(b). The filter should be like

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

This is a blurring filter, which is to replace each pixel value with the average value of its neighbors, including itself.

(c). The filter should be like

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 5 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

This is a horizontal sharpening filter, it increases the weight of the central pixel while subtracting some of its horizontal neighbors' intensities. The result is that horizontal features get enhanced.

(d). The filter should be like

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

This is a noise reduction filter, it gives more weight to pixels closer to the center. It averages out the values in a pixel's neighborhood, which can help in reducing high-frequency noise.

Problem 3

(a). $IOU = \frac{A \cap B}{A \cup B}$ is a metric defined as the ratio of the area of intersection of two bounding boxes to the area of their union.

If there is no intersection between two boxes, the area of $A \cap B = 0$, $IOU = 0$
If box A equals to box B, $A \cap B = A \cup B$, $IOU = 1$
By definition, $A \cup B = A + B - A \cap B$, so $A \cup B \geq A \cap B$, $0 \leq IOU \leq 1$

(b). When using the top-left and bottom-right coordinates to represent box,
$Box = (x_{top-left}, y_{top-left}, x_{bottom-right}, y_{bottom-right})$

The area of intersection, $A \cap B$, involves taking minimum and maximum operations to find the overlapping region.

$x_{intersection-top-left} = \max(x^A_{top-left}, x^B_{top-left})$
$y_{intersection-top-left} = \max(y^A_{top-left}, y^B_{top-left})$
$x_{intersection-bottom-right} = \min(x^A_{bottom-right}, x^B_{bottom-right})$
$y_{intersection-bottom-right} = \min(y^A_{bottom-right}, y^B_{bottom-right})$

Both the max and min operations are non-differentiable so IOU metric is non-differentiable and cannot be used as a loss function.

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.

```
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
```

```
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.

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

```
Files already downloaded and verified
```

```
# 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.

```
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.

```
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
```

```
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])
```

```
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.

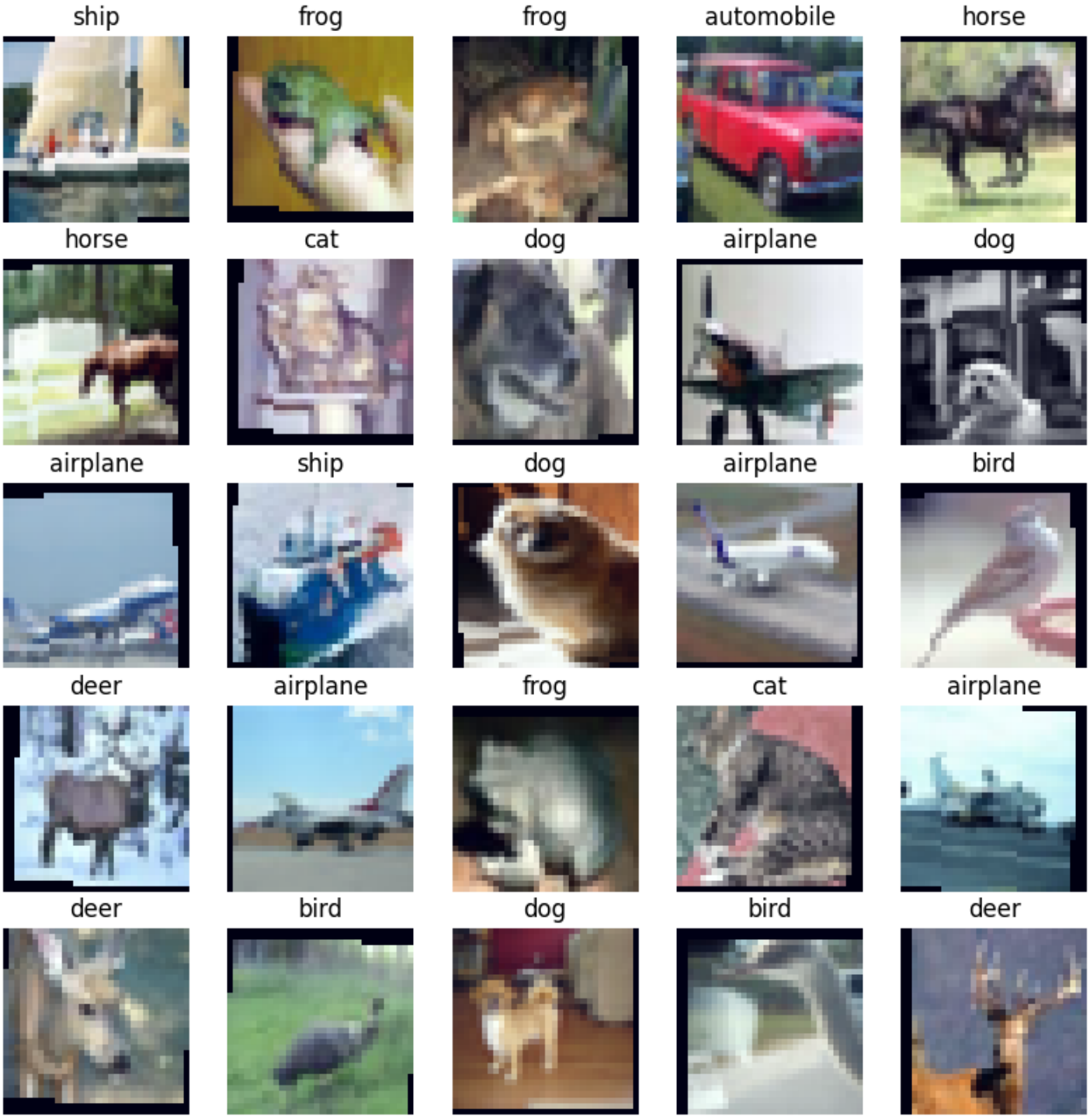
Note that by default PyTorch handles images that are arranged `[channel, height, width]`, but `matplotlib` expects images to be `[height, width, channel]`, hence we need to permute the dimensions of our images before plotting them.

```
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].

```
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
```

```
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.

```
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.

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


BATCH_SIZE = 256

train_iterator = data.DataLoader(train_data, shuffle=True, batch_size=BATCH_SIZE)

valid_iterator = data.DataLoader(valid_data, shuffle=False, batch_size=BATCH_SIZE)

test_iterator = data.DataLoader(test_data, shuffle=False, 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).

```
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=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.classifier = nn.Sequential(
            # define according to the steps described above
            nn.Dropout(0.5),
            nn.Linear(1024, 4096),
            nn.ReLU(inplace=True),

            nn.Dropout(0.5),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),

            nn.Linear(4096, output_dim)
        )


    def forward(self, x):
        x = self.features(x)
        h = x.view(x.shape[0], -1)
        x = self.classifier(h)
        return x, h
```

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

```
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.

```
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.

```
model.apply(initialize_parameters)
```

```
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)
  )
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
```

```
    (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.

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

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

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

We define a function to calculate accuracy...

```
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()`.

```
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()`.

```
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)
```

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

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

```
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.

```
# 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):

    start_time = time.time()

    # Training
    train_loss, train_acc = train(model, train_iterator, optimizer, criterion, device)

    # Validation
    valid_loss, valid_acc = evaluate(model, valid_iterator, criterion, device)

    end_time = time.time()

    epoch_mins, epoch_secs = epoch_time(start_time, end_time)

    # Saving the best model
    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model.state_dict(), 'best-model.pt')

    # Printing results after the current epoch
    print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
    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 | Epoch Time: 0m 17s
  Train Loss: 2.412 | Train Acc: 21.50%
   Val. Loss: 1.615 |  Val. Acc: 37.16%
Epoch: 02 | Epoch Time: 0m 10s
  Train Loss: 1.547 | Train Acc: 42.50%
   Val. Loss: 1.385 |  Val. Acc: 50.26%
Epoch: 03 | Epoch Time: 0m 10s
  Train Loss: 1.373 | Train Acc: 50.10%
   Val. Loss: 1.243 |  Val. Acc: 54.74%
Epoch: 04 | Epoch Time: 0m 10s
  Train Loss: 1.274 | Train Acc: 54.00%
   Val. Loss: 1.160 |  Val. Acc: 59.15%
Epoch: 05 | Epoch Time: 0m 10s
  Train Loss: 1.183 | Train Acc: 57.64%
   Val. Loss: 1.097 |  Val. Acc: 60.51%
Epoch: 06 | Epoch Time: 0m 10s
  Train Loss: 1.123 | Train Acc: 60.36%
   Val. Loss: 1.059 |  Val. Acc: 62.83%
Epoch: 07 | Epoch Time: 0m 10s
  Train Loss: 1.068 | Train Acc: 62.47%
   Val. Loss: 1.000 |  Val. Acc: 64.66%
Epoch: 08 | Epoch Time: 0m 10s
  Train Loss: 1.014 | Train Acc: 64.19%
   Val. Loss: 0.960 |  Val. Acc: 66.28%
Epoch: 09 | Epoch Time: 0m 10s
  Train Loss: 0.973 | Train Acc: 65.82%
   Val. Loss: 0.932 |  Val. Acc: 67.35%
Epoch: 10 | Epoch Time: 0m 10s
  Train Loss: 0.937 | Train Acc: 67.23%
   Val. Loss: 0.903 |  Val. Acc: 68.04%
Epoch: 11 | Epoch Time: 0m 10s
  Train Loss: 0.913 | Train Acc: 68.11%
   Val. Loss: 0.860 |  Val. Acc: 70.45%
Epoch: 12 | Epoch Time: 0m 10s
  Train Loss: 0.871 | Train Acc: 69.58%
   Val. Loss: 0.894 |  Val. Acc: 68.85%
Epoch: 13 | Epoch Time: 0m 10s
  Train Loss: 0.854 | Train Acc: 70.49%
   Val. Loss: 0.838 |  Val. Acc: 70.89%
Epoch: 14 | Epoch Time: 0m 10s
  Train Loss: 0.829 | Train Acc: 71.00%
   Val. Loss: 0.799 |  Val. Acc: 72.94%
Epoch: 15 | Epoch Time: 0m 10s
  Train Loss: 0.801 | Train Acc: 72.30%
   Val. Loss: 0.809 |  Val. Acc: 71.90%
Epoch: 16 | Epoch Time: 0m 10s
```

```
  Train Loss: 0.785 | Train Acc: 72.90%
   Val. Loss: 0.776 |  Val. Acc: 73.65%
Epoch: 17 | Epoch Time: 0m 10s
  Train Loss: 0.763 | Train Acc: 73.80%
   Val. Loss: 0.794 |  Val. Acc: 72.64%
Epoch: 18 | Epoch Time: 0m 10s
  Train Loss: 0.758 | Train Acc: 73.87%
   Val. Loss: 0.789 |  Val. Acc: 73.25%
Epoch: 19 | Epoch Time: 0m 10s
  Train Loss: 0.735 | Train Acc: 74.63%
   Val. Loss: 0.762 |  Val. Acc: 74.65%
Epoch: 20 | Epoch Time: 0m 10s
  Train Loss: 0.715 | Train Acc: 75.22%
   Val. Loss: 0.731 |  Val. Acc: 75.45%
Epoch: 21 | Epoch Time: 0m 10s
  Train Loss: 0.705 | Train Acc: 75.87%
   Val. Loss: 0.737 |  Val. Acc: 74.67%
Epoch: 22 | Epoch Time: 0m 10s
  Train Loss: 0.683 | Train Acc: 76.59%
   Val. Loss: 0.731 |  Val. Acc: 75.41%
Epoch: 23 | Epoch Time: 0m 10s
  Train Loss: 0.672 | Train Acc: 76.89%
   Val. Loss: 0.733 |  Val. Acc: 75.56%
Epoch: 24 | Epoch Time: 0m 10s
  Train Loss: 0.660 | Train Acc: 77.11%
   Val. Loss: 0.728 |  Val. Acc: 75.30%
Epoch: 25 | Epoch Time: 0m 10s
  Train Loss: 0.657 | Train Acc: 77.21%
   Val. Loss: 0.728 |  Val. Acc: 75.60%
```

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?

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

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

```
def get predictions(model, iterator, device):
```

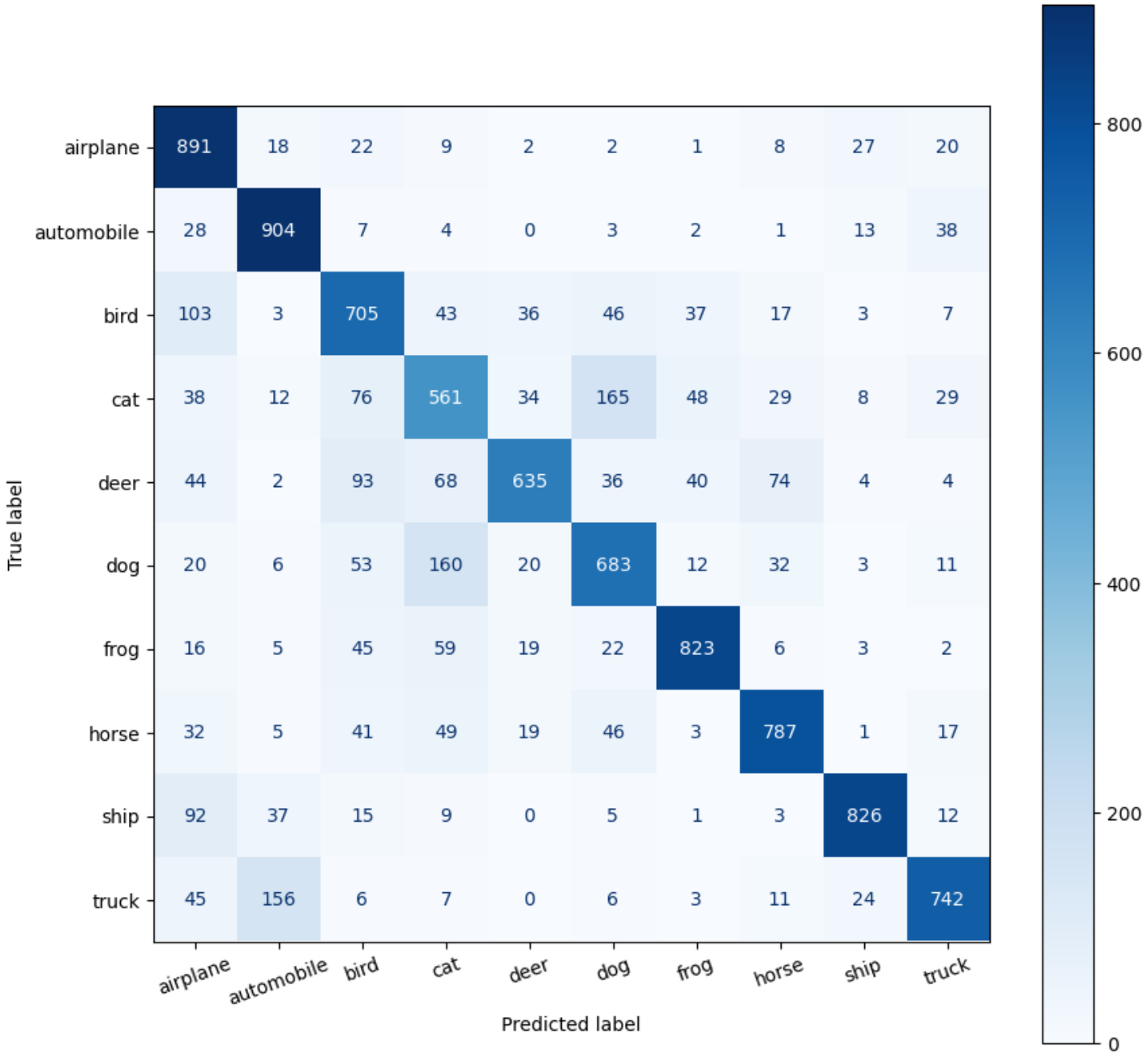
```
    model.eval()

    labels = []
    probs = []

    # Q4: Fill code here.
    with torch.no_grad():

        for (x, y) in iterator:

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

            y_pred, _ = model(x)

            labels.append(y)
            probs.append(y_pred)

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

    return labels, probs
```

```
labels, probs = get_predictions(model, test_iterator, device)
```

```
pred_labels = torch.argmax(probs, 1)
```

```
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.cpu(), pred_labels.cpu());
    cm = ConfusionMatrixDisplay(cm, display_labels = classes);
    cm.plot(values_format = 'd', cmap = 'Blues', ax = ax)
    plt.xticks(rotation = 20)
```

```
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.)

TorchVision Object Detection Finetuning Tutorial

Defining the Dataset

```
import os
import torch

from torchvision.io import read_image
from torchvision.ops.boxes import masks_to_boxes
from torchvision import tv_tensors
from torchvision.transforms.v2 import functional as F

import warnings
warnings.filterwarnings("ignore")

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 = read_image(img_path)
        mask = read_image(mask_path)
        # instances are encoded as different colors
        obj_ids = torch.unique(mask)
        # first id is the background, so remove it
        obj_ids = obj_ids[1:]
        num_objs = len(obj_ids)

        # split the color-encoded mask into a set
        # of binary masks
        masks = (mask == obj_ids[:, None, None]).to(dtype=torch.uint8)

        # get bounding box coordinates for each mask
        boxes = masks_to_boxes(masks)

        # there is only one class
        labels = torch.ones((num_objs,), dtype=torch.int64)


        image_id = idx
```

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

        # Wrap sample and targets into torchvision tv_tensors:
        img = tv_tensors.Image(img)

        target = {}
        target["boxes"] = tv_tensors.BoundingBoxes(boxes, format="XYXY",
canvas_size=F.get_size(img))
        target["masks"] = tv_tensors.Mask(masks)
        target["labels"] = labels
        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)
```

1 - Finetuning from a pretrained model

```
import torchvision
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor


def get_model_instance_segmentation(num_classes):
    # load an instance segmentation model pre-trained on COCO
    model = torchvision.models.detection.maskrcnn_resnet50_fpn(weights="DEFAULT")

    # 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
```

```
from torchvision.transforms import v2 as T


def get_transform(train):
    transforms = []
    if train:
        transforms.append(T.RandomHorizontalFlip(0.5))
    transforms.append(T.ToDtype(torch.float, scale=True))
    transforms.append(T.ToPureTensor())
    return T.Compose(transforms)
```

```
import utils
from engine import train_one_epoch, evaluate

# 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('PennFudanPed', get_transform(train=True))
dataset_test = PennFudanDataset('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
)

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

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

# construct an optimizer
params = [p for p in model.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
lr_scheduler = torch.optim.lr_scheduler.StepLR(
    optimizer,
    step_size=3,
    gamma=0.1
)

# let's train it for 5 epochs, modify to 10
num_epochs = 10

for epoch in range(num_epochs):
    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    evaluate(model, data_loader_test, device=device)

print("That's it!")
```

```
Epoch: [0]  [ 0/60]  eta: 0:07:02  lr: 0.000090  loss: 5.3390 (5.3390)  loss_classifier:
1.0105 (1.0105)  loss_box_reg: 0.1436 (0.1436)  loss_mask: 4.1724 (4.1724)  loss_objectness:
0.0037 (0.0037)  loss_rpn_box_reg: 0.0088 (0.0088)  time: 7.0445  data: 0.0234  max mem: 2258
Epoch: [0]  [10/60]  eta: 0:00:57  lr: 0.000936  loss: 1.8463 (2.7963)  loss_classifier:
0.5531 (0.5893)  loss_box_reg: 0.2991 (0.2884)  loss_mask: 1.2507 (1.8919)  loss_objectness:
0.0214 (0.0224)  loss_rpn_box_reg: 0.0041 (0.0043)  time: 1.1425  data: 0.0330  max mem: 2768
Epoch: [0]  [20/60]  eta: 0:00:34  lr: 0.001783  loss: 1.1690 (1.7923)  loss_classifier:
0.2832 (0.3868)  loss_box_reg: 0.2921 (0.2685)  loss_mask: 0.4370 (1.1116)  loss_objectness:
0.0203 (0.0202)  loss_rpn_box_reg: 0.0041 (0.0051)  time: 0.5459  data: 0.0343  max mem: 2768
Epoch: [0]  [30/60]  eta: 0:00:22  lr: 0.002629  loss: 0.5333 (1.3968)  loss_classifier:
```

```
0.0836 (0.2878)  loss_box_reg: 0.1943 (0.2556)  loss_mask: 0.2266 (0.8300)  loss_objectness:
0.0108 (0.0169)  loss_rpn_box_reg: 0.0054 (0.0065)  time: 0.5423  data: 0.0350  max mem: 2768
Epoch: [0]  [40/60]  eta: 0:00:14  lr: 0.003476  loss: 0.5064 (1.1767)  loss_classifier:
0.0605 (0.2309)  loss_box_reg: 0.1882 (0.2449)  loss_mask: 0.2233 (0.6801)  loss_objectness:
0.0070 (0.0142)  loss_rpn_box_reg: 0.0060 (0.0066)  time: 0.5498  data: 0.0359  max mem: 2768
Epoch: [0]  [50/60]  eta: 0:00:06  lr: 0.004323  loss: 0.4367 (1.0277)  loss_classifier:
0.0496 (0.1949)  loss_box_reg: 0.1796 (0.2326)  loss_mask: 0.1884 (0.5817)  loss_objectness:
0.0040 (0.0121)  loss_rpn_box_reg: 0.0051 (0.0064)  time: 0.5473  data: 0.0349  max mem: 2857
Epoch: [0]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.3930 (0.9301)  loss_classifier:
0.0411 (0.1716)  loss_box_reg: 0.1604 (0.2184)  loss_mask: 0.1720 (0.5232)  loss_objectness:
0.0018 (0.0105)  loss_rpn_box_reg: 0.0050 (0.0063)  time: 0.5446  data: 0.0337  max mem: 2857
Epoch: [0] Total time: 0:00:39 (0.6535 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:09  model_time: 0.1604 (0.1604)  evaluator_time: 0.0114 (0.0114)
time: 0.1811  data: 0.0086  max mem: 2857
Test:  [49/50]  eta: 0:00:00  model_time: 0.0941 (0.1085)  evaluator_time: 0.0055 (0.0091)
time: 0.1253  data: 0.0124  max mem: 2857
Test: Total time: 0:00:06 (0.1311 s / it)
Averaged stats: model_time: 0.0941 (0.1085)  evaluator_time: 0.0055 (0.0091)
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 ] = 0.619
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.990
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.749
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.425
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.524
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.627
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.263
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.675
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.675
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.644
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.677
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.667
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.984
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.870
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.300
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.547
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.676
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.276
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.704
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.709
 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 ] = 0.744
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.708
Epoch: [1]  [ 0/60]  eta: 0:00:27  lr: 0.005000  loss: 0.2231 (0.2231)  loss_classifier:
0.0209 (0.0209)  loss_box_reg: 0.0553 (0.0553)  loss_mask: 0.1395 (0.1395)  loss_objectness:
0.0016 (0.0016)  loss_rpn_box_reg: 0.0058 (0.0058)  time: 0.4534  data: 0.0213  max mem: 2857
Epoch: [1]  [10/60]  eta: 0:00:26  lr: 0.005000  loss: 0.2564 (0.2813)  loss_classifier:
0.0267 (0.0389)  loss_box_reg: 0.0777 (0.0952)  loss_mask: 0.1418 (0.1422)  loss_objectness:
0.0010 (0.0014)  loss_rpn_box_reg: 0.0026 (0.0036)  time: 0.5397  data: 0.0259  max mem: 2884
Epoch: [1]  [20/60]  eta: 0:00:22  lr: 0.005000  loss: 0.2787 (0.2950)  loss_classifier:
0.0319 (0.0378)  loss_box_reg: 0.0879 (0.1015)  loss_mask: 0.1418 (0.1493)  loss_objectness:
0.0012 (0.0018)  loss_rpn_box_reg: 0.0030 (0.0047)  time: 0.5784  data: 0.0385  max mem: 2884
Epoch: [1]  [30/60]  eta: 0:00:18  lr: 0.005000  loss: 0.3099 (0.3159)  loss_classifier:
0.0355 (0.0404)  loss_box_reg: 0.1161 (0.1106)  loss_mask: 0.1509 (0.1572)  loss_objectness:
0.0016 (0.0022)  loss_rpn_box_reg: 0.0061 (0.0055)  time: 0.6499  data: 0.0586  max mem: 2884
Epoch: [1]  [40/60]  eta: 0:00:12  lr: 0.005000  loss: 0.3249 (0.3128)  loss_classifier:
0.0407 (0.0409)  loss_box_reg: 0.1024 (0.1063)  loss_mask: 0.1509 (0.1576)  loss_objectness:
0.0011 (0.0021)  loss_rpn_box_reg: 0.0061 (0.0058)  time: 0.6338  data: 0.0513  max mem: 2884
Epoch: [1]  [50/60]  eta: 0:00:06  lr: 0.005000  loss: 0.2723 (0.3133)  loss_classifier:
0.0473 (0.0432)  loss_box_reg: 0.0887 (0.1065)  loss_mask: 0.1439 (0.1558)  loss_objectness:
0.0010 (0.0021)  loss_rpn_box_reg: 0.0033 (0.0056)  time: 0.6041  data: 0.0381  max mem: 2948
Epoch: [1]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.2641 (0.3072)  loss_classifier:
0.0395 (0.0423)  loss_box_reg: 0.0795 (0.1022)  loss_mask: 0.1405 (0.1555)  loss_objectness:
0.0008 (0.0019)  loss_rpn_box_reg: 0.0031 (0.0052)  time: 0.5998  data: 0.0361  max mem: 2948
Epoch: [1] Total time: 0:00:36 (0.6005 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:08  model_time: 0.1473 (0.1473)  evaluator_time: 0.0223 (0.0223)
time: 0.1800  data: 0.0097  max mem: 2948
Test:  [49/50]  eta: 0:00:00  model_time: 0.1074 (0.1131)  evaluator_time: 0.0051 (0.0091)
time: 0.1395  data: 0.0187  max mem: 2948
Test: Total time: 0:00:07 (0.1422 s / it)
Averaged stats: model_time: 0.1074 (0.1131)  evaluator_time: 0.0051 (0.0091)
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 ] = 0.791
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.923
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.533
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.590
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.801
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.334
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.834
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.834
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.800
 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 ] = 0.839
IoU metric: segm
```

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.733
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.947
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.300
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.586
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.742
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.310
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.773
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.773
 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 ] = 0.789
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.775
Epoch: [2]  [ 0/60]  eta: 0:00:33  lr: 0.005000  loss: 0.2981 (0.2981)  loss_classifier:
0.0507 (0.0507)  loss_box_reg: 0.0932 (0.0932)  loss_mask: 0.1490 (0.1490)  loss_objectness:
0.0018 (0.0018)  loss_rpn_box_reg: 0.0036 (0.0036)  time: 0.5648  data: 0.0228  max mem: 2948
Epoch: [2]  [10/60]  eta: 0:00:29  lr: 0.005000  loss: 0.2350 (0.2414)  loss_classifier:
0.0404 (0.0398)  loss_box_reg: 0.0616 (0.0669)  loss_mask: 0.1205 (0.1286)  loss_objectness:
0.0008 (0.0022)  loss_rpn_box_reg: 0.0022 (0.0039)  time: 0.5809  data: 0.0379  max mem: 2948
Epoch: [2]  [20/60]  eta: 0:00:23  lr: 0.005000  loss: 0.2350 (0.2525)  loss_classifier:
0.0371 (0.0375)  loss_box_reg: 0.0577 (0.0703)  loss_mask: 0.1294 (0.1382)  loss_objectness:
0.0008 (0.0018)  loss_rpn_box_reg: 0.0028 (0.0047)  time: 0.5999  data: 0.0384  max mem: 2949
Epoch: [2]  [30/60]  eta: 0:00:17  lr: 0.005000  loss: 0.2196 (0.2391)  loss_classifier:
0.0282 (0.0336)  loss_box_reg: 0.0498 (0.0626)  loss_mask: 0.1400 (0.1365)  loss_objectness:
0.0005 (0.0019)  loss_rpn_box_reg: 0.0022 (0.0045)  time: 0.5980  data: 0.0313  max mem: 2949
Epoch: [2]  [40/60]  eta: 0:00:12  lr: 0.005000  loss: 0.2196 (0.2449)  loss_classifier:
0.0285 (0.0343)  loss_box_reg: 0.0507 (0.0669)  loss_mask: 0.1400 (0.1373)  loss_objectness:
0.0005 (0.0017)  loss_rpn_box_reg: 0.0022 (0.0047)  time: 0.6028  data: 0.0285  max mem: 3130
Epoch: [2]  [50/60]  eta: 0:00:05  lr: 0.005000  loss: 0.2274 (0.2408)  loss_classifier:
0.0288 (0.0323)  loss_box_reg: 0.0716 (0.0667)  loss_mask: 0.1338 (0.1354)  loss_objectness:
0.0008 (0.0018)  loss_rpn_box_reg: 0.0041 (0.0045)  time: 0.5979  data: 0.0266  max mem: 3130
Epoch: [2]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.2327 (0.2408)  loss_classifier:
0.0240 (0.0322)  loss_box_reg: 0.0654 (0.0676)  loss_mask: 0.1318 (0.1350)  loss_objectness:
0.0006 (0.0016)  loss_rpn_box_reg: 0.0034 (0.0044)  time: 0.5847  data: 0.0278  max mem: 3130
Epoch: [2] Total time: 0:00:35 (0.5940 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:07  model_time: 0.1406 (0.1406)  evaluator_time: 0.0030 (0.0030)
time: 0.1527  data: 0.0084  max mem: 3130
Test:  [49/50]  eta: 0:00:00  model_time: 0.1040 (0.1116)  evaluator_time: 0.0055 (0.0088)
time: 0.1499  data: 0.0195  max mem: 3130
Test: Total time: 0:00:06 (0.1377 s / it)
Averaged stats: model_time: 0.1040 (0.1116)  evaluator_time: 0.0055 (0.0088)
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 ] = 0.723
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.991
```

```
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.901
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.326
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.566
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.733
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.308
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.768
 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 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.722
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.773
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.735
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.982
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.912
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.240
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.530
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.749
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.314
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.773
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.775
 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 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.779
Epoch: [3]  [ 0/60]  eta: 0:00:38  lr: 0.000500  loss: 0.2697 (0.2697)  loss_classifier:
0.0295 (0.0295)  loss_box_reg: 0.1028 (0.1028)  loss_mask: 0.1346 (0.1346)  loss_objectness:
0.0003 (0.0003)  loss_rpn_box_reg: 0.0024 (0.0024)  time: 0.6364  data: 0.0373  max mem: 3130
Epoch: [3]  [10/60]  eta: 0:00:31  lr: 0.000500  loss: 0.2443 (0.2586)  loss_classifier:
0.0288 (0.0316)  loss_box_reg: 0.0740 (0.0815)  loss_mask: 0.1346 (0.1399)  loss_objectness:
0.0006 (0.0012)  loss_rpn_box_reg: 0.0033 (0.0045)  time: 0.6381  data: 0.0391  max mem: 3130
Epoch: [3]  [20/60]  eta: 0:00:24  lr: 0.000500  loss: 0.2143 (0.2240)  loss_classifier:
0.0245 (0.0284)  loss_box_reg: 0.0530 (0.0645)  loss_mask: 0.1214 (0.1268)  loss_objectness:
0.0005 (0.0008)  loss_rpn_box_reg: 0.0026 (0.0036)  time: 0.6030  data: 0.0338  max mem: 3130
Epoch: [3]  [30/60]  eta: 0:00:17  lr: 0.000500  loss: 0.1807 (0.2138)  loss_classifier:
0.0225 (0.0274)  loss_box_reg: 0.0378 (0.0576)  loss_mask: 0.1160 (0.1248)  loss_objectness:
0.0003 (0.0007)  loss_rpn_box_reg: 0.0022 (0.0033)  time: 0.5729  data: 0.0283  max mem: 3130
Epoch: [3]  [40/60]  eta: 0:00:12  lr: 0.000500  loss: 0.1791 (0.2100)  loss_classifier:
0.0227 (0.0276)  loss_box_reg: 0.0350 (0.0544)  loss_mask: 0.1156 (0.1243)  loss_objectness:
0.0004 (0.0007)  loss_rpn_box_reg: 0.0016 (0.0031)  time: 0.6032  data: 0.0302  max mem: 3134
Epoch: [3]  [50/60]  eta: 0:00:05  lr: 0.000500  loss: 0.1774 (0.2055)  loss_classifier:
0.0219 (0.0271)  loss_box_reg: 0.0295 (0.0516)  loss_mask: 0.1086 (0.1230)  loss_objectness:
0.0003 (0.0008)  loss_rpn_box_reg: 0.0012 (0.0029)  time: 0.5961  data: 0.0309  max mem: 3134
Epoch: [3]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.1740 (0.2028)  loss_classifier:
0.0207 (0.0270)  loss_box_reg: 0.0321 (0.0505)  loss_mask: 0.1086 (0.1215)  loss_objectness:
0.0004 (0.0009)  loss_rpn_box_reg: 0.0021 (0.0029)  time: 0.5913  data: 0.0326  max mem: 3134
Epoch: [3] Total time: 0:00:35 (0.5932 s / it)
creating index...
index created!

Test:  [ 0/50]  eta: 0:00:07  model_time: 0.1401 (0.1401)  evaluator_time: 0.0027 (0.0027)
```

```
time: 0.1520  data: 0.0085  max mem: 3134
Test:  [49/50]  eta: 0:00:00  model_time: 0.0995 (0.1062)  evaluator_time: 0.0033 (0.0056)
time: 0.1283  data: 0.0141  max mem: 3134
Test: Total time: 0:00:06 (0.1259 s / it)
Averaged stats: model_time: 0.0995 (0.1062)  evaluator_time: 0.0033 (0.0056)
Accumulating evaluation results...
DONE (t=0.03s).
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.818
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.945
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.552
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.675
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.828
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.345
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.855
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.855
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.700
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.844
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.858
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.745
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.981
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.934
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.296
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.564
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.756
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.315
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.787
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.787
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.778
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.790
Epoch: [4]  [ 0/60]  eta: 0:00:45  lr: 0.000500  loss: 0.1845 (0.1845)  loss_classifier:
0.0239 (0.0239)  loss_box_reg: 0.0420 (0.0420)  loss_mask: 0.1169 (0.1169)  loss_objectness:
0.0003 (0.0003)  loss_rpn_box_reg: 0.0014 (0.0014)  time: 0.7579  data: 0.0463  max mem: 3134
Epoch: [4]  [10/60]  eta: 0:00:34  lr: 0.000500  loss: 0.1845 (0.1758)  loss_classifier:
0.0239 (0.0209)  loss_box_reg: 0.0420 (0.0374)  loss_mask: 0.1091 (0.1143)  loss_objectness:
0.0003 (0.0009)  loss_rpn_box_reg: 0.0027 (0.0023)  time: 0.6819  data: 0.0567  max mem: 3134
Epoch: [4]  [20/60]  eta: 0:00:25  lr: 0.000500  loss: 0.1957 (0.1950)  loss_classifier:
0.0262 (0.0271)  loss_box_reg: 0.0411 (0.0446)  loss_mask: 0.1121 (0.1196)  loss_objectness:
0.0004 (0.0012)  loss_rpn_box_reg: 0.0022 (0.0024)  time: 0.6363  data: 0.0437  max mem: 3134
Epoch: [4]  [30/60]  eta: 0:00:19  lr: 0.000500  loss: 0.1831 (0.1911)  loss_classifier:
0.0243 (0.0265)  loss_box_reg: 0.0411 (0.0443)  loss_mask: 0.1086 (0.1168)  loss_objectness:
0.0006 (0.0011)  loss_rpn_box_reg: 0.0017 (0.0024)  time: 0.6393  data: 0.0398  max mem: 3210

Epoch: [4]  [40/60]  eta: 0:00:12  lr: 0.000500  loss: 0.1830 (0.1968)  loss_classifier:
```

```
0.0230 (0.0275)  loss_box_reg: 0.0434 (0.0452)  loss_mask: 0.1099 (0.1204)  loss_objectness:
0.0006 (0.0009)  loss_rpn_box_reg: 0.0017 (0.0027)  time: 0.6220  data: 0.0386  max mem: 3210
Epoch: [4]  [50/60]  eta: 0:00:06  lr: 0.000500  loss: 0.2034 (0.1979)  loss_classifier:
0.0280 (0.0281)  loss_box_reg: 0.0434 (0.0455)  loss_mask: 0.1244 (0.1206)  loss_objectness:
0.0005 (0.0009)  loss_rpn_box_reg: 0.0036 (0.0028)  time: 0.5994  data: 0.0386  max mem: 3210
Epoch: [4]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.1706 (0.1935)  loss_classifier:
0.0244 (0.0269)  loss_box_reg: 0.0338 (0.0437)  loss_mask: 0.1080 (0.1192)  loss_objectness:
0.0005 (0.0009)  loss_rpn_box_reg: 0.0031 (0.0028)  time: 0.6260  data: 0.0436  max mem: 3210
Epoch: [4] Total time: 0:00:37 (0.6332 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:07  model_time: 0.1430 (0.1430)  evaluator_time: 0.0039 (0.0039)
time: 0.1595  data: 0.0119  max mem: 3210
Test:  [49/50]  eta: 0:00:00  model_time: 0.1010 (0.1149)  evaluator_time: 0.0036 (0.0076)
time: 0.1397  data: 0.0166  max mem: 3210
Test: Total time: 0:00:06 (0.1398 s / it)
Averaged stats: model_time: 0.1010 (0.1149)  evaluator_time: 0.0036 (0.0076)
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 ] = 0.823
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.943
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.350
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.672
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.834
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.345
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.861
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.861
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.700
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.833
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.866
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 ] = 0.981
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.932
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.325
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.563
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.762
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.315
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.792
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.792
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.794

Epoch: [5]  [ 0/60]  eta: 0:00:29  lr: 0.000500  loss: 0.2033 (0.2033)  loss_classifier:
0.0236 (0.0236)  loss_box_reg: 0.0361 (0.0361)  loss_mask: 0.1413 (0.1413)  loss_objectness:
```

```
0.0236 (0.0236)  loss_box_reg: 0.0361 (0.0361)  loss_mask: 0.1413 (0.1413)  loss_objectness:
0.0002 (0.0002)  loss_rpn_box_reg: 0.0021 (0.0021)  time: 0.4895  data: 0.0308  max mem: 3210
Epoch: [5]  [10/60]  eta: 0:00:28  lr: 0.000500  loss: 0.1743 (0.1834)  loss_classifier:
0.0233 (0.0246)  loss_box_reg: 0.0330 (0.0387)  loss_mask: 0.1176 (0.1175)  loss_objectness:
0.0003 (0.0003)  loss_rpn_box_reg: 0.0011 (0.0023)  time: 0.5638  data: 0.0297  max mem: 3210
Epoch: [5]  [20/60]  eta: 0:00:22  lr: 0.000500  loss: 0.1743 (0.1912)  loss_classifier:
0.0242 (0.0278)  loss_box_reg: 0.0345 (0.0416)  loss_mask: 0.1176 (0.1183)  loss_objectness:
0.0004 (0.0007)  loss_rpn_box_reg: 0.0018 (0.0028)  time: 0.5777  data: 0.0296  max mem: 3210
Epoch: [5]  [30/60]  eta: 0:00:17  lr: 0.000500  loss: 0.1816 (0.1929)  loss_classifier:
0.0242 (0.0270)  loss_box_reg: 0.0395 (0.0437)  loss_mask: 0.1194 (0.1182)  loss_objectness:
0.0005 (0.0010)  loss_rpn_box_reg: 0.0024 (0.0030)  time: 0.5798  data: 0.0298  max mem: 3210
Epoch: [5]  [40/60]  eta: 0:00:11  lr: 0.000500  loss: 0.1713 (0.1865)  loss_classifier:
0.0234 (0.0267)  loss_box_reg: 0.0378 (0.0413)  loss_mask: 0.1095 (0.1150)  loss_objectness:
0.0004 (0.0008)  loss_rpn_box_reg: 0.0017 (0.0027)  time: 0.5919  data: 0.0281  max mem: 3500
Epoch: [5]  [50/60]  eta: 0:00:05  lr: 0.000500  loss: 0.1708 (0.1864)  loss_classifier:
0.0233 (0.0261)  loss_box_reg: 0.0330 (0.0409)  loss_mask: 0.1058 (0.1158)  loss_objectness:
0.0003 (0.0008)  loss_rpn_box_reg: 0.0017 (0.0027)  time: 0.6047  data: 0.0297  max mem: 3500
Epoch: [5]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.1711 (0.1862)  loss_classifier:
0.0198 (0.0256)  loss_box_reg: 0.0343 (0.0405)  loss_mask: 0.1122 (0.1165)  loss_objectness:
0.0003 (0.0008)  loss_rpn_box_reg: 0.0026 (0.0028)  time: 0.5860  data: 0.0284  max mem: 3500
Epoch: [5] Total time: 0:00:35 (0.5854 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:07  model_time: 0.1435 (0.1435)  evaluator_time: 0.0025 (0.0025)
time: 0.1561  data: 0.0093  max mem: 3500
Test:  [49/50]  eta: 0:00:00  model_time: 0.0991 (0.1052)  evaluator_time: 0.0031 (0.0057)
time: 0.1220  data: 0.0134  max mem: 3500
Test: Total time: 0:00:06 (0.1260 s / it)
Averaged stats: model_time: 0.0991 (0.1052)  evaluator_time: 0.0031 (0.0057)
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 ] = 0.830
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.991
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.954
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.635
 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 ] = 0.840
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.355
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.869
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.869
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.750
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.822
 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 ] = 0.752
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.981
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.922
```

```
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.922
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.325
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.578
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.761
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.316
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.791
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.793
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.822
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.793
Epoch: [6]  [ 0/60]  eta: 0:00:41  lr: 0.000050  loss: 0.1881 (0.1881)  loss_classifier:
0.0245 (0.0245)  loss_box_reg: 0.0411 (0.0411)  loss_mask: 0.1181 (0.1181)  loss_objectness:
0.0002 (0.0002)  loss_rpn_box_reg: 0.0043 (0.0043)  time: 0.6954  data: 0.0222  max mem: 3500
Epoch: [6]  [10/60]  eta: 0:00:29  lr: 0.000050  loss: 0.1881 (0.1951)  loss_classifier:
0.0245 (0.0262)  loss_box_reg: 0.0335 (0.0410)  loss_mask: 0.1181 (0.1232)  loss_objectness:
0.0002 (0.0018)  loss_rpn_box_reg: 0.0011 (0.0030)  time: 0.5858  data: 0.0288  max mem: 3500
Epoch: [6]  [20/60]  eta: 0:00:24  lr: 0.000050  loss: 0.1755 (0.2008)  loss_classifier:
0.0256 (0.0273)  loss_box_reg: 0.0322 (0.0451)  loss_mask: 0.1131 (0.1236)  loss_objectness:
0.0002 (0.0014)  loss_rpn_box_reg: 0.0032 (0.0034)  time: 0.5979  data: 0.0331  max mem: 3500
Epoch: [6]  [30/60]  eta: 0:00:17  lr: 0.000050  loss: 0.1677 (0.1868)  loss_classifier:
0.0215 (0.0247)  loss_box_reg: 0.0289 (0.0393)  loss_mask: 0.1119 (0.1188)  loss_objectness:
0.0003 (0.0012)  loss_rpn_box_reg: 0.0019 (0.0028)  time: 0.5783  data: 0.0300  max mem: 3500
Epoch: [6]  [40/60]  eta: 0:00:11  lr: 0.000050  loss: 0.1590 (0.1862)  loss_classifier:
0.0209 (0.0248)  loss_box_reg: 0.0299 (0.0390)  loss_mask: 0.1084 (0.1186)  loss_objectness:
0.0003 (0.0009)  loss_rpn_box_reg: 0.0016 (0.0028)  time: 0.5592  data: 0.0272  max mem: 3500
Epoch: [6]  [50/60]  eta: 0:00:05  lr: 0.000050  loss: 0.1878 (0.1903)  loss_classifier:
0.0219 (0.0255)  loss_box_reg: 0.0304 (0.0406)  loss_mask: 0.1092 (0.1203)  loss_objectness:
0.0004 (0.0011)  loss_rpn_box_reg: 0.0025 (0.0029)  time: 0.5829  data: 0.0290  max mem: 3500
Epoch: [6]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.1732 (0.1877)  loss_classifier:
0.0232 (0.0254)  loss_box_reg: 0.0351 (0.0401)  loss_mask: 0.1071 (0.1185)  loss_objectness:
0.0003 (0.0009)  loss_rpn_box_reg: 0.0019 (0.0027)  time: 0.6117  data: 0.0417  max mem: 3500
Epoch: [6] Total time: 0:00:35 (0.5897 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:07  model_time: 0.1407 (0.1407)  evaluator_time: 0.0024 (0.0024)
time: 0.1521  data: 0.0083  max mem: 3500
Test:  [49/50]  eta: 0:00:00  model_time: 0.0979 (0.1042)  evaluator_time: 0.0032 (0.0049)
time: 0.1201  data: 0.0126  max mem: 3500
Test: Total time: 0:00:06 (0.1225 s / it)
Averaged stats: model_time: 0.0979 (0.1042)  evaluator_time: 0.0032 (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 ] = 0.831
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.943
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.635
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.667
```

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.842
 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 ] = 0.872
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.872
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.750
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.833
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.876
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.753
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.991
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.940
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.325
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.567
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.763
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.315
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.791
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.793
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.795
Epoch: [7]  [ 0/60]  eta: 0:00:41  lr: 0.000050  loss: 0.2865 (0.2865)  loss_classifier:
0.0411 (0.0411)  loss_box_reg: 0.0862 (0.0862)  loss_mask: 0.1506 (0.1506)  loss_objectness:
0.0004 (0.0004)  loss_rpn_box_reg: 0.0083 (0.0083)  time: 0.6939  data: 0.0432  max mem: 3500
Epoch: [7]  [10/60]  eta: 0:00:30  lr: 0.000050  loss: 0.1941 (0.2085)  loss_classifier:
0.0256 (0.0280)  loss_box_reg: 0.0464 (0.0523)  loss_mask: 0.1192 (0.1238)  loss_objectness:
0.0002 (0.0007)  loss_rpn_box_reg: 0.0021 (0.0037)  time: 0.6102  data: 0.0327  max mem: 3500
Epoch: [7]  [20/60]  eta: 0:00:23  lr: 0.000050  loss: 0.1610 (0.1853)  loss_classifier:
0.0190 (0.0242)  loss_box_reg: 0.0286 (0.0425)  loss_mask: 0.1046 (0.1150)  loss_objectness:
0.0003 (0.0007)  loss_rpn_box_reg: 0.0015 (0.0030)  time: 0.5880  data: 0.0289  max mem: 3500
Epoch: [7]  [30/60]  eta: 0:00:17  lr: 0.000050  loss: 0.1633 (0.1897)  loss_classifier:
0.0219 (0.0251)  loss_box_reg: 0.0300 (0.0434)  loss_mask: 0.1019 (0.1178)  loss_objectness:
0.0003 (0.0006)  loss_rpn_box_reg: 0.0015 (0.0028)  time: 0.5838  data: 0.0270  max mem: 3500
Epoch: [7]  [40/60]  eta: 0:00:11  lr: 0.000050  loss: 0.1785 (0.1851)  loss_classifier:
0.0243 (0.0249)  loss_box_reg: 0.0357 (0.0406)  loss_mask: 0.1096 (0.1165)  loss_objectness:
0.0003 (0.0006)  loss_rpn_box_reg: 0.0019 (0.0026)  time: 0.5719  data: 0.0275  max mem: 3500
Epoch: [7]  [50/60]  eta: 0:00:05  lr: 0.000050  loss: 0.1785 (0.1863)  loss_classifier:
0.0243 (0.0248)  loss_box_reg: 0.0377 (0.0407)  loss_mask: 0.1096 (0.1175)  loss_objectness:
0.0003 (0.0005)  loss_rpn_box_reg: 0.0020 (0.0027)  time: 0.5665  data: 0.0281  max mem: 3500
Epoch: [7]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.1808 (0.1837)  loss_classifier:
0.0232 (0.0239)  loss_box_reg: 0.0356 (0.0394)  loss_mask: 0.1155 (0.1173)  loss_objectness:
0.0002 (0.0005)  loss_rpn_box_reg: 0.0024 (0.0027)  time: 0.5784  data: 0.0276  max mem: 3500
Epoch: [7] Total time: 0:00:34 (0.5818 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:08  model_time: 0.1437 (0.1437)  evaluator_time: 0.0040 (0.0040)
time: 0.1603  data: 0.0119  max mem: 3500

Test:  [49/50]  eta: 0:00:00  model_time: 0.0981 (0.1066)  evaluator_time: 0.0032 (0.0056)
time: 0.1211  data: 0.0127  max mem: 3500
```

```
Test: Total time: 0:00:06 (0.1269 s / it)
Averaged stats: model_time: 0.0981 (0.1066)  evaluator_time: 0.0032 (0.0056)
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 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.943
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.635
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.667
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.836
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.350
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.868
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.868
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.750
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.833
 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 ] = 0.750
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.991
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.922
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.325
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.555
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.762
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.315
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.789
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.792
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.789
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.794
Epoch: [8]  [ 0/60]  eta: 0:00:38  lr: 0.000050  loss: 0.1496 (0.1496)  loss_classifier:
0.0159 (0.0159)  loss_box_reg: 0.0264 (0.0264)  loss_mask: 0.1060 (0.1060)  loss_objectness:
0.0001 (0.0001)  loss_rpn_box_reg: 0.0013 (0.0013)  time: 0.6364  data: 0.0282  max mem: 3500
Epoch: [8]  [10/60]  eta: 0:00:31  lr: 0.000050  loss: 0.1572 (0.1686)  loss_classifier:
0.0257 (0.0236)  loss_box_reg: 0.0264 (0.0314)  loss_mask: 0.1060 (0.1116)  loss_objectness:
0.0002 (0.0003)  loss_rpn_box_reg: 0.0013 (0.0018)  time: 0.6390  data: 0.0344  max mem: 3500
Epoch: [8]  [20/60]  eta: 0:00:25  lr: 0.000050  loss: 0.1640 (0.1732)  loss_classifier:
0.0263 (0.0256)  loss_box_reg: 0.0368 (0.0341)  loss_mask: 0.1095 (0.1111)  loss_objectness:
0.0002 (0.0007)  loss_rpn_box_reg: 0.0013 (0.0017)  time: 0.6406  data: 0.0463  max mem: 3500
Epoch: [8]  [30/60]  eta: 0:00:18  lr: 0.000050  loss: 0.1682 (0.1757)  loss_classifier:
0.0248 (0.0261)  loss_box_reg: 0.0359 (0.0342)  loss_mask: 0.1121 (0.1127)  loss_objectness:
0.0004 (0.0006)  loss_rpn_box_reg: 0.0016 (0.0021)  time: 0.6276  data: 0.0432  max mem: 3500
Epoch: [8]  [40/60]  eta: 0:00:12  lr: 0.000050  loss: 0.1817 (0.1818)  loss_classifier:
0.0250 (0.0265)  loss_box_reg: 0.0359 (0.0374)  loss_mask: 0.1151 (0.1147)  loss_objectness:
0.0004 (0.0007)  loss_rpn_box_reg: 0.0026 (0.0025)  time: 0.6003  data: 0.0287  max mem: 3500
Epoch: [8]  [50/60]  eta: 0:00:06  lr: 0.000050  loss: 0.2130 (0.1853)  loss_classifier:
0.0259 (0.0264)  loss_box_reg: 0.0491 (0.0393)  loss_mask: 0.1160 (0.1162)  loss_objectness:
```

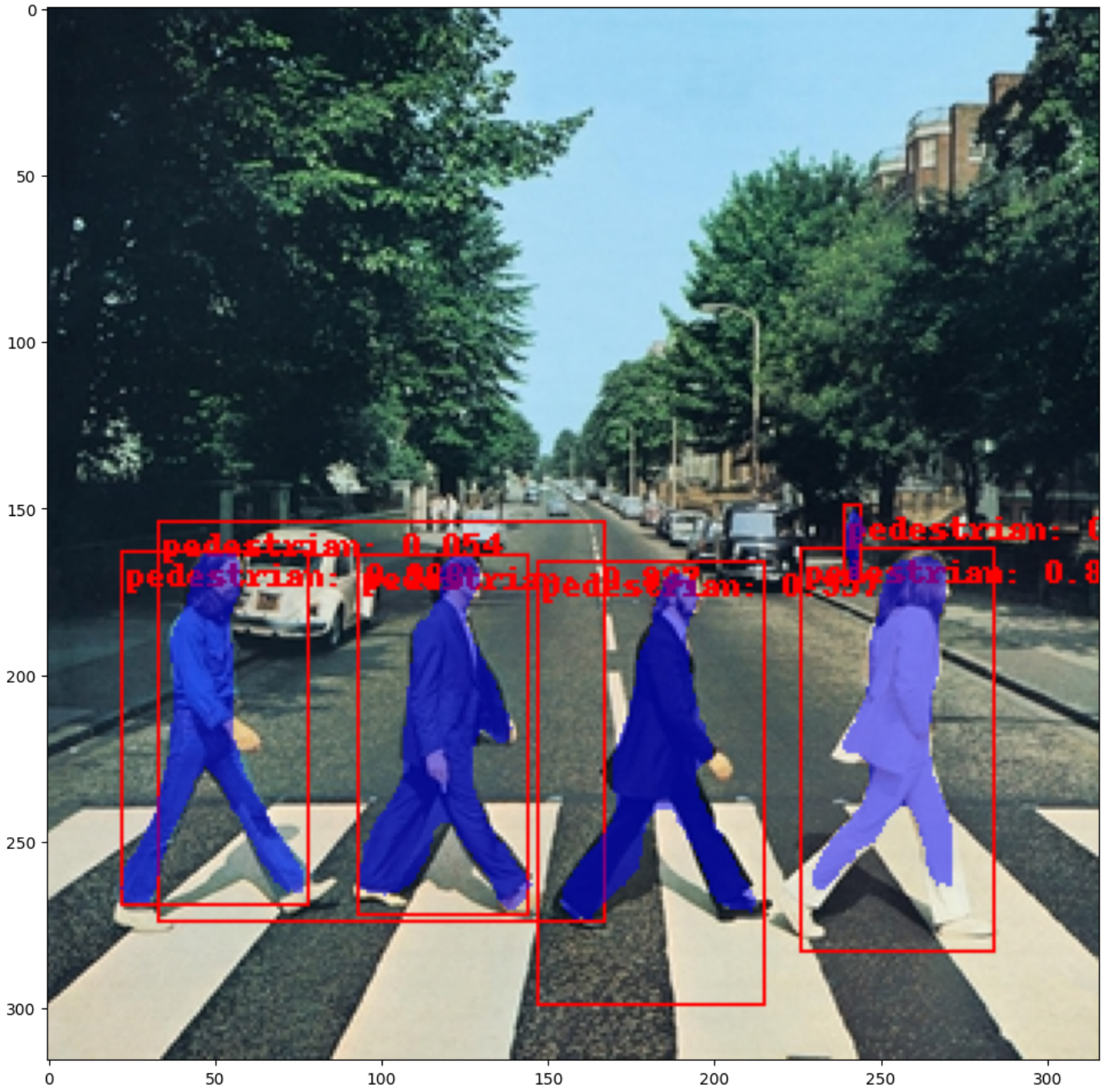
```
0.0004 (0.0009)  loss_rpn_box_reg: 0.0025 (0.0026)  time: 0.5974  data: 0.0297  max mem: 3500
Epoch: [8]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.1719 (0.1852)  loss_classifier:
0.0240 (0.0257)  loss_box_reg: 0.0347 (0.0389)  loss_mask: 0.1119 (0.1170)  loss_objectness:
0.0003 (0.0008)  loss_rpn_box_reg: 0.0020 (0.0027)  time: 0.5926  data: 0.0290  max mem: 3500
Epoch: [8] Total time: 0:00:36 (0.6119 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:07  model_time: 0.1387 (0.1387)  evaluator_time: 0.0025 (0.0025)
time: 0.1504  data: 0.0086  max mem: 3500
Test:  [49/50]  eta: 0:00:00  model_time: 0.0992 (0.1070)  evaluator_time: 0.0039 (0.0058)
time: 0.1278  data: 0.0155  max mem: 3500
Test: Total time: 0:00:06 (0.1281 s / it)
Averaged stats: model_time: 0.0992 (0.1070)  evaluator_time: 0.0039 (0.0058)
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 ] = 0.830
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.943
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.375
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.660
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.840
 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 ] = 0.871
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.871
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.750
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.822
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.876
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 ] = 0.991
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.940
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.325
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.557
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.762
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.315
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.790
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.792
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.789
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.795
Epoch: [9]  [ 0/60]  eta: 0:00:28  lr: 0.000005  loss: 0.1142 (0.1142)  loss_classifier:
0.0095 (0.0095)  loss_box_reg: 0.0138 (0.0138)  loss_mask: 0.0905 (0.0905)  loss_objectness:
0.0000 (0.0000)  loss_rpn_box_reg: 0.0004 (0.0004)  time: 0.4719  data: 0.0173  max mem: 3500

Epoch: [9]  [10/60]  eta: 0:00:26  lr: 0.000005  loss: 0.1785 (0.1816)  loss_classifier:
0.0195 (0.0227)  loss_box_reg: 0.0283 (0.0369)  loss_mask: 0.1098 (0.1188)  loss_objectness:
```

```
0.0003 (0.0009)  loss_rpn_box_reg: 0.0013 (0.0024)  time: 0.5334  data: 0.0247  max mem: 3500
Epoch: [9]  [20/60]  eta: 0:00:22  lr: 0.000005  loss: 0.1939 (0.1888)  loss_classifier:
0.0268 (0.0267)  loss_box_reg: 0.0384 (0.0405)  loss_mask: 0.1098 (0.1184)  loss_objectness:
0.0003 (0.0008)  loss_rpn_box_reg: 0.0019 (0.0024)  time: 0.5557  data: 0.0291  max mem: 3500
Epoch: [9]  [30/60]  eta: 0:00:17  lr: 0.000005  loss: 0.1848 (0.1884)  loss_classifier:
0.0268 (0.0271)  loss_box_reg: 0.0404 (0.0411)  loss_mask: 0.1083 (0.1173)  loss_objectness:
0.0003 (0.0007)  loss_rpn_box_reg: 0.0019 (0.0023)  time: 0.5903  data: 0.0302  max mem: 3500
Epoch: [9]  [40/60]  eta: 0:00:11  lr: 0.000005  loss: 0.1756 (0.1897)  loss_classifier:
0.0242 (0.0269)  loss_box_reg: 0.0373 (0.0409)  loss_mask: 0.1105 (0.1187)  loss_objectness:
0.0002 (0.0006)  loss_rpn_box_reg: 0.0024 (0.0025)  time: 0.5986  data: 0.0284  max mem: 3500
Epoch: [9]  [50/60]  eta: 0:00:05  lr: 0.000005  loss: 0.1688 (0.1854)  loss_classifier:
0.0226 (0.0263)  loss_box_reg: 0.0270 (0.0389)  loss_mask: 0.1113 (0.1172)  loss_objectness:
0.0002 (0.0006)  loss_rpn_box_reg: 0.0014 (0.0025)  time: 0.5685  data: 0.0276  max mem: 3500
Epoch: [9]  [59/60]  eta: 0:00:00  lr: 0.000005  loss: 0.1754 (0.1848)  loss_classifier:
0.0247 (0.0262)  loss_box_reg: 0.0336 (0.0389)  loss_mask: 0.1088 (0.1166)  loss_objectness:
0.0004 (0.0006)  loss_rpn_box_reg: 0.0020 (0.0026)  time: 0.5861  data: 0.0289  max mem: 3500
Epoch: [9] Total time: 0:00:34 (0.5770 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:07  model_time: 0.1411 (0.1411)  evaluator_time: 0.0025 (0.0025)
time: 0.1527  data: 0.0086  max mem: 3500
Test:  [49/50]  eta: 0:00:00  model_time: 0.0985 (0.1053)  evaluator_time: 0.0033 (0.0050)
time: 0.1215  data: 0.0130  max mem: 3500
Test: Total time: 0:00:06 (0.1243 s / it)
Averaged stats: model_time: 0.0985 (0.1053)  evaluator_time: 0.0033 (0.0050)
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 ] = 0.831
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.992
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.943
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.375
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.660
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.842
 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 ] = 0.872
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.872
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.750
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.822
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.878
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 ] = 0.991
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.940
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.325
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.554
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.762
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.315
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.790
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.792
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.789
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.795
That's it!
```

```
import matplotlib.pyplot as plt

from torchvision.utils import draw_bounding_boxes, draw_segmentation_masks


image = read_image("Beatles_-_Abbey_Road.jpg")
eval_transform = get_transform(train=False)

model.eval()
with torch.no_grad():
    x = eval_transform(image)
    # convert RGBA -> RGB and move to device
    x = x[:3, ...].to(device)
    predictions = model([x, ])
    pred = predictions[0]


image = (255.0 * (image - image.min()) / (image.max() - image.min())).to(torch.uint8)
image = image[:3, ...]
pred_labels = [f"pedestrian: {score:.3f}" for label, score in zip(pred["labels"],
pred["scores"])]
pred_boxes = pred["boxes"].long()
output_image = draw_bounding_boxes(image, pred_boxes, pred_labels, colors="red")

masks = (pred["masks"] > 0.7).squeeze(1)
output_image = draw_segmentation_masks(output_image, masks, alpha=0.5, colors="blue")


plt.figure(figsize=(12, 12))
plt.imshow(output_image.permute(1, 2, 0))
```

```
<matplotlib.image.AxesImage at 0x79c685b5a6e0>
```



2 - Modifying the model to add a different backbone

```
import torchvision
from torchvision.models.detection import FasterRCNN, MaskRCNN
from torchvision.models.detection.rpn import AnchorGenerator



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 Faster-RCNN model
model = FasterRCNN(
    backbone,
    num_classes=2,
    rpn_anchor_generator=anchor_generator,
    box_roi_pool=roi_pooler
)

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

# construct an optimizer
params = [p for p in model.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
```

```
lr_scheduler = torch.optim.lr_scheduler.StepLR(
    optimizer,
    step_size=3,
    gamma=0.1
)

# let's train it for 5 epochs, modify to 10
num_epochs = 10

for epoch in range(num_epochs):
    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    evaluate(model, data_loader_test, device=device)
```

```
Epoch: [0]  [ 0/60]  eta: 0:00:23  lr: 0.000090  loss: 1.4881 (1.4881)  loss_classifier:
0.7213 (0.7213)  loss_box_reg: 0.0230 (0.0230)  loss_objectness: 0.7056 (0.7056)
loss_rpn_box_reg: 0.0382 (0.0382)  time: 0.3888  data: 0.0220  max mem: 13577
Epoch: [0]  [10/60]  eta: 0:00:17  lr: 0.000936  loss: 1.4354 (1.3951)  loss_classifier:
0.6727 (0.6331)  loss_box_reg: 0.0234 (0.0278)  loss_objectness: 0.6981 (0.6935)
loss_rpn_box_reg: 0.0385 (0.0408)  time: 0.3545  data: 0.0276  max mem: 13577
Epoch: [0]  [20/60]  eta: 0:00:14  lr: 0.001783  loss: 1.1219 (1.2181)  loss_classifier:
0.4221 (0.4668)  loss_box_reg: 0.0411 (0.0594)  loss_objectness: 0.6578 (0.6497)
loss_rpn_box_reg: 0.0406 (0.0422)  time: 0.3573  data: 0.0313  max mem: 13577
Epoch: [0]  [30/60]  eta: 0:00:10  lr: 0.002629  loss: 0.9818 (1.1438)  loss_classifier:
0.2722 (0.4175)  loss_box_reg: 0.1333 (0.0981)  loss_objectness: 0.5217 (0.5853)
loss_rpn_box_reg: 0.0479 (0.0429)  time: 0.3508  data: 0.0305  max mem: 13577
Epoch: [0]  [40/60]  eta: 0:00:06  lr: 0.003476  loss: 0.8954 (1.0571)  loss_classifier:
0.2722 (0.3807)  loss_box_reg: 0.1931 (0.1188)  loss_objectness: 0.3562 (0.5176)
loss_rpn_box_reg: 0.0340 (0.0399)  time: 0.3377  data: 0.0257  max mem: 13577
Epoch: [0]  [50/60]  eta: 0:00:03  lr: 0.004323  loss: 0.7315 (0.9852)  loss_classifier:
0.2473 (0.3516)  loss_box_reg: 0.1951 (0.1326)  loss_objectness: 0.2558 (0.4615)
loss_rpn_box_reg: 0.0297 (0.0394)  time: 0.3446  data: 0.0283  max mem: 13577
Epoch: [0]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.5636 (0.9258)  loss_classifier:
0.2049 (0.3316)  loss_box_reg: 0.1632 (0.1387)  loss_objectness: 0.2007 (0.4181)
loss_rpn_box_reg: 0.0254 (0.0374)  time: 0.3414  data: 0.0275  max mem: 13577
Epoch: [0] Total time: 0:00:20 (0.3468 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0805 (0.0805)  evaluator_time: 0.0014 (0.0014)
time: 0.0916  data: 0.0091  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0666 (0.0692)  evaluator_time: 0.0018 (0.0025)
time: 0.0818  data: 0.0127  max mem: 13577
Test: Total time: 0:00:04 (0.0854 s / it)
Averaged stats: model_time: 0.0666 (0.0692)  evaluator_time: 0.0018 (0.0025)
Accumulating evaluation results...
```

```
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.061
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.188
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.010
 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 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.111
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.060
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.281
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.338
 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 ] = 0.022
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.368
Epoch: [1]  [ 0/60]  eta: 0:00:20  lr: 0.005000  loss: 0.7907 (0.7907)  loss_classifier:
0.2730 (0.2730)  loss_box_reg: 0.3139 (0.3139)  loss_objectness: 0.1803 (0.1803)
loss_rpn_box_reg: 0.0235 (0.0235)  time: 0.3491  data: 0.0262  max mem: 13577
Epoch: [1]  [10/60]  eta: 0:00:17  lr: 0.005000  loss: 0.5971 (0.6338)  loss_classifier:
0.2137 (0.2243)  loss_box_reg: 0.2479 (0.2201)  loss_objectness: 0.1608 (0.1604)
loss_rpn_box_reg: 0.0235 (0.0290)  time: 0.3586  data: 0.0342  max mem: 13577
Epoch: [1]  [20/60]  eta: 0:00:14  lr: 0.005000  loss: 0.5401 (0.5911)  loss_classifier:
0.1893 (0.2035)  loss_box_reg: 0.1889 (0.2079)  loss_objectness: 0.1467 (0.1501)
loss_rpn_box_reg: 0.0266 (0.0297)  time: 0.3531  data: 0.0316  max mem: 13577
Epoch: [1]  [30/60]  eta: 0:00:10  lr: 0.005000  loss: 0.4755 (0.5633)  loss_classifier:
0.1654 (0.1913)  loss_box_reg: 0.1571 (0.1996)  loss_objectness: 0.1304 (0.1437)
loss_rpn_box_reg: 0.0254 (0.0287)  time: 0.3436  data: 0.0276  max mem: 13577
Epoch: [1]  [40/60]  eta: 0:00:07  lr: 0.005000  loss: 0.4316 (0.5280)  loss_classifier:
0.1458 (0.1799)  loss_box_reg: 0.1333 (0.1885)  loss_objectness: 0.1165 (0.1326)
loss_rpn_box_reg: 0.0238 (0.0271)  time: 0.3484  data: 0.0295  max mem: 13577
Epoch: [1]  [50/60]  eta: 0:00:03  lr: 0.005000  loss: 0.4568 (0.5262)  loss_classifier:
0.1411 (0.1769)  loss_box_reg: 0.1767 (0.1920)  loss_objectness: 0.1101 (0.1293)
loss_rpn_box_reg: 0.0267 (0.0279)  time: 0.3489  data: 0.0286  max mem: 13577
Epoch: [1]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.4514 (0.5163)  loss_classifier:
0.1388 (0.1722)  loss_box_reg: 0.1767 (0.1905)  loss_objectness: 0.1074 (0.1253)
loss_rpn_box_reg: 0.0320 (0.0283)  time: 0.3407  data: 0.0262  max mem: 13577
Epoch: [1] Total time: 0:00:20 (0.3473 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0709 (0.0709)  evaluator_time: 0.0013 (0.0013)
time: 0.0812  data: 0.0084  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0620 (0.0687)  evaluator_time: 0.0014 (0.0023)
time: 0.0781  data: 0.0125  max mem: 13577
Test: Total time: 0:00:04 (0.0864 s / it)
Averaged stats: model_time: 0.0620 (0.0687)  evaluator_time: 0.0014 (0.0023)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.167
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.548
```

```
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.018
 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 ] = 0.191
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.112
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.337
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.352
 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 ] = 0.384
Epoch: [2]  [ 0/60]  eta: 0:00:22  lr: 0.005000  loss: 0.6456 (0.6456)  loss_classifier:
0.1880 (0.1880)  loss_box_reg: 0.2919 (0.2919)  loss_objectness: 0.1256 (0.1256)
loss_rpn_box_reg: 0.0402 (0.0402)  time: 0.3706  data: 0.0359  max mem: 13577
Epoch: [2]  [10/60]  eta: 0:00:17  lr: 0.005000  loss: 0.4584 (0.4519)  loss_classifier:
0.1518 (0.1423)  loss_box_reg: 0.2013 (0.1904)  loss_objectness: 0.0873 (0.0925)
loss_rpn_box_reg: 0.0289 (0.0267)  time: 0.3464  data: 0.0262  max mem: 13577
Epoch: [2]  [20/60]  eta: 0:00:13  lr: 0.005000  loss: 0.4640 (0.4822)  loss_classifier:
0.1518 (0.1485)  loss_box_reg: 0.2013 (0.2098)  loss_objectness: 0.0873 (0.0966)
loss_rpn_box_reg: 0.0254 (0.0272)  time: 0.3455  data: 0.0264  max mem: 13577
Epoch: [2]  [30/60]  eta: 0:00:10  lr: 0.005000  loss: 0.4017 (0.4432)  loss_classifier:
0.1267 (0.1367)  loss_box_reg: 0.1885 (0.1936)  loss_objectness: 0.0834 (0.0878)
loss_rpn_box_reg: 0.0203 (0.0252)  time: 0.3533  data: 0.0308  max mem: 13577
Epoch: [2]  [40/60]  eta: 0:00:07  lr: 0.005000  loss: 0.3251 (0.4346)  loss_classifier:
0.1002 (0.1323)  loss_box_reg: 0.1437 (0.1879)  loss_objectness: 0.0703 (0.0867)
loss_rpn_box_reg: 0.0215 (0.0278)  time: 0.3556  data: 0.0334  max mem: 13577
Epoch: [2]  [50/60]  eta: 0:00:03  lr: 0.005000  loss: 0.3320 (0.4233)  loss_classifier:
0.0987 (0.1296)  loss_box_reg: 0.1368 (0.1840)  loss_objectness: 0.0687 (0.0826)
loss_rpn_box_reg: 0.0266 (0.0272)  time: 0.3495  data: 0.0288  max mem: 13577
Epoch: [2]  [59/60]  eta: 0:00:00  lr: 0.005000  loss: 0.2990 (0.4011)  loss_classifier:
0.0873 (0.1230)  loss_box_reg: 0.1261 (0.1743)  loss_objectness: 0.0560 (0.0782)
loss_rpn_box_reg: 0.0202 (0.0256)  time: 0.3484  data: 0.0252  max mem: 13577
Epoch: [2] Total time: 0:00:21 (0.3503 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0782 (0.0782)  evaluator_time: 0.0013 (0.0013)
time: 0.0885  data: 0.0084  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0665 (0.0678)  evaluator_time: 0.0015 (0.0019)
time: 0.0814  data: 0.0127  max mem: 13577
Test: Total time: 0:00:04 (0.0835 s / it)
Averaged stats: model_time: 0.0665 (0.0678)  evaluator_time: 0.0015 (0.0019)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.172
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.555
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.032
 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 ] = 0.190
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.108
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.364
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.388
 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 ] = 0.424
Epoch: [3]  [ 0/60]  eta: 0:00:22  lr: 0.000500  loss: 0.6482 (0.6482)  loss_classifier:
0.1973 (0.1973)  loss_box_reg: 0.3128 (0.3128)  loss_objectness: 0.0833 (0.0833)
loss_rpn_box_reg: 0.0548 (0.0548)  time: 0.3675  data: 0.0290  max mem: 13577
Epoch: [3]  [10/60]  eta: 0:00:18  lr: 0.000500  loss: 0.4065 (0.4286)  loss_classifier:
0.1095 (0.1292)  loss_box_reg: 0.2072 (0.2068)  loss_objectness: 0.0707 (0.0688)
loss_rpn_box_reg: 0.0205 (0.0238)  time: 0.3614  data: 0.0307  max mem: 13577
Epoch: [3]  [20/60]  eta: 0:00:14  lr: 0.000500  loss: 0.3403 (0.3605)  loss_classifier:
0.0980 (0.1110)  loss_box_reg: 0.1423 (0.1704)  loss_objectness: 0.0562 (0.0607)
loss_rpn_box_reg: 0.0155 (0.0183)  time: 0.3550  data: 0.0291  max mem: 13577
Epoch: [3]  [30/60]  eta: 0:00:10  lr: 0.000500  loss: 0.3084 (0.3541)  loss_classifier:
0.0932 (0.1080)  loss_box_reg: 0.1388 (0.1659)  loss_objectness: 0.0537 (0.0604)
loss_rpn_box_reg: 0.0138 (0.0198)  time: 0.3498  data: 0.0274  max mem: 13577
Epoch: [3]  [40/60]  eta: 0:00:07  lr: 0.000500  loss: 0.2729 (0.3315)  loss_classifier:
0.0804 (0.1009)  loss_box_reg: 0.1281 (0.1543)  loss_objectness: 0.0518 (0.0577)
loss_rpn_box_reg: 0.0186 (0.0186)  time: 0.3528  data: 0.0281  max mem: 13577
Epoch: [3]  [50/60]  eta: 0:00:03  lr: 0.000500  loss: 0.2782 (0.3425)  loss_classifier:
0.0804 (0.1032)  loss_box_reg: 0.1396 (0.1606)  loss_objectness: 0.0470 (0.0583)
loss_rpn_box_reg: 0.0164 (0.0204)  time: 0.3660  data: 0.0325  max mem: 13577
Epoch: [3]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.3178 (0.3415)  loss_classifier:
0.1016 (0.1030)  loss_box_reg: 0.1470 (0.1588)  loss_objectness: 0.0548 (0.0590)
loss_rpn_box_reg: 0.0212 (0.0206)  time: 0.3662  data: 0.0307  max mem: 13577
Epoch: [3] Total time: 0:00:21 (0.3575 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0784 (0.0784)  evaluator_time: 0.0013 (0.0013)
time: 0.0889  data: 0.0086  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0651 (0.0665)  evaluator_time: 0.0015 (0.0020)
time: 0.0813  data: 0.0131  max mem: 13577
Test: Total time: 0:00:04 (0.0825 s / it)
Averaged stats: model_time: 0.0651 (0.0665)  evaluator_time: 0.0015 (0.0020)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.247
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.673
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.083
 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 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.276
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.168
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.390
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.408
 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 ] = 0.022
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.445
Epoch: [4]  [ 0/60]  eta: 0:00:19  lr: 0.000500  loss: 0.2327 (0.2327)  loss_classifier:
0.0778 (0.0778)  loss_box_reg: 0.1046 (0.1046)  loss_objectness: 0.0390 (0.0390)
loss_rpn_box_reg: 0.0114 (0.0114)  time: 0.3206  data: 0.0183  max mem: 13577
Epoch: [4]  [10/60]  eta: 0:00:18  lr: 0.000500  loss: 0.3444 (0.3246)  loss_classifier:
0.0968 (0.0970)  loss_box_reg: 0.1720 (0.1603)  loss_objectness: 0.0466 (0.0502)
loss_rpn_box_reg: 0.0146 (0.0172)  time: 0.3659  data: 0.0314  max mem: 13577
Epoch: [4]  [20/60]  eta: 0:00:14  lr: 0.000500  loss: 0.2942 (0.3074)  loss_classifier:
0.0941 (0.0938)  loss_box_reg: 0.1415 (0.1474)  loss_objectness: 0.0446 (0.0483)
loss_rpn_box_reg: 0.0148 (0.0178)  time: 0.3607  data: 0.0302  max mem: 13577
Epoch: [4]  [30/60]  eta: 0:00:10  lr: 0.000500  loss: 0.2816 (0.3297)  loss_classifier:
0.0858 (0.0991)  loss_box_reg: 0.1317 (0.1595)  loss_objectness: 0.0447 (0.0517)
loss_rpn_box_reg: 0.0178 (0.0193)  time: 0.3529  data: 0.0272  max mem: 13577
Epoch: [4]  [40/60]  eta: 0:00:07  lr: 0.000500  loss: 0.3514 (0.3424)  loss_classifier:
0.1076 (0.1034)  loss_box_reg: 0.1787 (0.1645)  loss_objectness: 0.0569 (0.0547)
loss_rpn_box_reg: 0.0214 (0.0198)  time: 0.3623  data: 0.0299  max mem: 13577
Epoch: [4]  [50/60]  eta: 0:00:03  lr: 0.000500  loss: 0.3037 (0.3354)  loss_classifier:
0.1046 (0.1019)  loss_box_reg: 0.1417 (0.1597)  loss_objectness: 0.0537 (0.0545)
loss_rpn_box_reg: 0.0183 (0.0193)  time: 0.3596  data: 0.0287  max mem: 13577
Epoch: [4]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.2808 (0.3273)  loss_classifier:
0.0823 (0.0992)  loss_box_reg: 0.1188 (0.1551)  loss_objectness: 0.0519 (0.0534)
loss_rpn_box_reg: 0.0173 (0.0196)  time: 0.3518  data: 0.0253  max mem: 13577
Epoch: [4] Total time: 0:00:21 (0.3577 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0780 (0.0780)  evaluator_time: 0.0012 (0.0012)
time: 0.0882  data: 0.0084  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0647 (0.0713)  evaluator_time: 0.0018 (0.0028)
time: 0.0810  data: 0.0130  max mem: 13577
Test: Total time: 0:00:04 (0.0898 s / it)
Averaged stats: model_time: 0.0647 (0.0713)  evaluator_time: 0.0018 (0.0028)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.238
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.609
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.132
 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 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.265
 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 ] = 0.398
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.425
 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 ] = 0.011
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.463
Epoch: [5]  [ 0/60]  eta: 0:00:20  lr: 0.000500  loss: 0.3760 (0.3760)  loss_classifier:
0.1191 (0.1191)  loss_box_reg: 0.1695 (0.1695)  loss_objectness: 0.0636 (0.0636)
```

```
loss_rpn_box_reg: 0.0238 (0.0238)  time: 0.3363  data: 0.0218  max mem: 13577
Epoch: [5]  [10/60]  eta: 0:00:17  lr: 0.000500  loss: 0.3568 (0.3450)  loss_classifier:
0.0978 (0.1029)  loss_box_reg: 0.1695 (0.1675)  loss_objectness: 0.0550 (0.0541)
loss_rpn_box_reg: 0.0179 (0.0204)  time: 0.3514  data: 0.0271  max mem: 13577
Epoch: [5]  [20/60]  eta: 0:00:13  lr: 0.000500  loss: 0.3011 (0.3082)  loss_classifier:
0.0862 (0.0902)  loss_box_reg: 0.1370 (0.1482)  loss_objectness: 0.0473 (0.0517)
loss_rpn_box_reg: 0.0152 (0.0181)  time: 0.3490  data: 0.0265  max mem: 13577
Epoch: [5]  [30/60]  eta: 0:00:10  lr: 0.000500  loss: 0.3011 (0.3117)  loss_classifier:
0.0825 (0.0910)  loss_box_reg: 0.1342 (0.1498)  loss_objectness: 0.0473 (0.0520)
loss_rpn_box_reg: 0.0171 (0.0189)  time: 0.3564  data: 0.0286  max mem: 13577
Epoch: [5]  [40/60]  eta: 0:00:07  lr: 0.000500  loss: 0.3152 (0.3197)  loss_classifier:
0.0937 (0.0942)  loss_box_reg: 0.1516 (0.1533)  loss_objectness: 0.0534 (0.0531)
loss_rpn_box_reg: 0.0205 (0.0192)  time: 0.3633  data: 0.0287  max mem: 13577
Epoch: [5]  [50/60]  eta: 0:00:03  lr: 0.000500  loss: 0.2916 (0.3270)  loss_classifier:
0.0986 (0.0979)  loss_box_reg: 0.1373 (0.1551)  loss_objectness: 0.0516 (0.0545)
loss_rpn_box_reg: 0.0163 (0.0194)  time: 0.3611  data: 0.0273  max mem: 13577
Epoch: [5]  [59/60]  eta: 0:00:00  lr: 0.000500  loss: 0.3065 (0.3231)  loss_classifier:
0.0896 (0.0967)  loss_box_reg: 0.1374 (0.1539)  loss_objectness: 0.0418 (0.0527)
loss_rpn_box_reg: 0.0167 (0.0198)  time: 0.3660  data: 0.0308  max mem: 13577
Epoch: [5] Total time: 0:00:21 (0.3597 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0827 (0.0827)  evaluator_time: 0.0013 (0.0013)
time: 0.0933  data: 0.0087  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0639 (0.0663)  evaluator_time: 0.0016 (0.0020)
time: 0.0792  data: 0.0125  max mem: 13577
Test: Total time: 0:00:04 (0.0818 s / it)
Averaged stats: model_time: 0.0639 (0.0663)  evaluator_time: 0.0016 (0.0020)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.269
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.637
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.137
 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 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.301
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.160
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.419
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.442
 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 ] = 0.480
Epoch: [6]  [ 0/60]  eta: 0:00:19  lr: 0.000050  loss: 0.1891 (0.1891)  loss_classifier:
0.0685 (0.0685)  loss_box_reg: 0.0786 (0.0786)  loss_objectness: 0.0355 (0.0355)
loss_rpn_box_reg: 0.0065 (0.0065)  time: 0.3305  data: 0.0256  max mem: 13577

Epoch: [6]  [10/60]  eta: 0:00:17  lr: 0.000050  loss: 0.3023 (0.3003)  loss_classifier:
0.0823 (0.0958)  loss_box_reg: 0.1401 (0.1395)  loss_objectness: 0.0473 (0.0460)
```

```
loss_rpn_box_reg: 0.0194 (0.0189)  time: 0.3567  data: 0.0293  max mem: 13577
Epoch: [6]  [20/60]  eta: 0:00:14  lr: 0.000050  loss: 0.2657 (0.3032)  loss_classifier:
0.0823 (0.0958)  loss_box_reg: 0.1380 (0.1448)  loss_objectness: 0.0437 (0.0449)
loss_rpn_box_reg: 0.0178 (0.0177)  time: 0.3575  data: 0.0290  max mem: 13577
Epoch: [6]  [30/60]  eta: 0:00:10  lr: 0.000050  loss: 0.2603 (0.3012)  loss_classifier:
0.0834 (0.0944)  loss_box_reg: 0.1290 (0.1412)  loss_objectness: 0.0457 (0.0476)
loss_rpn_box_reg: 0.0160 (0.0181)  time: 0.3573  data: 0.0260  max mem: 13577
Epoch: [6]  [40/60]  eta: 0:00:07  lr: 0.000050  loss: 0.2958 (0.3150)  loss_classifier:
0.0929 (0.0985)  loss_box_reg: 0.1340 (0.1493)  loss_objectness: 0.0474 (0.0479)
loss_rpn_box_reg: 0.0171 (0.0193)  time: 0.3646  data: 0.0266  max mem: 13577
Epoch: [6]  [50/60]  eta: 0:00:03  lr: 0.000050  loss: 0.3133 (0.3261)  loss_classifier:
0.0961 (0.0998)  loss_box_reg: 0.1451 (0.1554)  loss_objectness: 0.0474 (0.0511)
loss_rpn_box_reg: 0.0216 (0.0198)  time: 0.3688  data: 0.0321  max mem: 13577
Epoch: [6]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.2958 (0.3167)  loss_classifier:
0.0879 (0.0968)  loss_box_reg: 0.1273 (0.1506)  loss_objectness: 0.0440 (0.0496)
loss_rpn_box_reg: 0.0201 (0.0197)  time: 0.3575  data: 0.0289  max mem: 13577
Epoch: [6] Total time: 0:00:21 (0.3597 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0778 (0.0778)  evaluator_time: 0.0014 (0.0014)
time: 0.0896  data: 0.0098  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0676 (0.0673)  evaluator_time: 0.0019 (0.0022)
time: 0.0865  data: 0.0146  max mem: 13577
Test: Total time: 0:00:04 (0.0840 s / it)
Averaged stats: model_time: 0.0676 (0.0673)  evaluator_time: 0.0019 (0.0022)
Accumulating evaluation results...
DONE (t=0.03s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.256
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.632
 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 ] = 0.016
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.289
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.164
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.414
 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 ] = 0.044
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.492
Epoch: [7]  [ 0/60]  eta: 0:00:22  lr: 0.000050  loss: 0.3784 (0.3784)  loss_classifier:
0.1157 (0.1157)  loss_box_reg: 0.1815 (0.1815)  loss_objectness: 0.0602 (0.0602)
loss_rpn_box_reg: 0.0211 (0.0211)  time: 0.3757  data: 0.0321  max mem: 13577
Epoch: [7]  [10/60]  eta: 0:00:18  lr: 0.000050  loss: 0.3713 (0.3509)  loss_classifier:
0.1037 (0.1044)  loss_box_reg: 0.1815 (0.1706)  loss_objectness: 0.0561 (0.0538)
loss_rpn_box_reg: 0.0211 (0.0221)  time: 0.3666  data: 0.0316  max mem: 13577

Epoch: [7]  [20/60]  eta: 0:00:14  lr: 0.000050  loss: 0.2973 (0.3345)  loss_classifier:
0.0902 (0.1013)  loss_box_reg: 0.1461 (0.1609)  loss_objectness: 0.0478 (0.0516)
```

```
loss_rpn_box_reg: 0.0181 (0.0207)  time: 0.3618  data: 0.0293  max mem: 13577
Epoch: [7]  [30/60]  eta: 0:00:10  lr: 0.000050  loss: 0.3101 (0.3373)  loss_classifier:
0.0902 (0.1041)  loss_box_reg: 0.1538 (0.1602)  loss_objectness: 0.0503 (0.0521)
loss_rpn_box_reg: 0.0181 (0.0209)  time: 0.3556  data: 0.0256  max mem: 13577
Epoch: [7]  [40/60]  eta: 0:00:07  lr: 0.000050  loss: 0.3186 (0.3300)  loss_classifier:
0.0974 (0.1008)  loss_box_reg: 0.1454 (0.1577)  loss_objectness: 0.0514 (0.0517)
loss_rpn_box_reg: 0.0162 (0.0198)  time: 0.3558  data: 0.0286  max mem: 13577
Epoch: [7]  [50/60]  eta: 0:00:03  lr: 0.000050  loss: 0.2648 (0.3152)  loss_classifier:
0.0812 (0.0961)  loss_box_reg: 0.1153 (0.1500)  loss_objectness: 0.0435 (0.0504)
loss_rpn_box_reg: 0.0112 (0.0187)  time: 0.3572  data: 0.0298  max mem: 13577
Epoch: [7]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.2703 (0.3175)  loss_classifier:
0.0835 (0.0961)  loss_box_reg: 0.1105 (0.1514)  loss_objectness: 0.0430 (0.0507)
loss_rpn_box_reg: 0.0127 (0.0194)  time: 0.3554  data: 0.0271  max mem: 13577
Epoch: [7] Total time: 0:00:21 (0.3586 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0841 (0.0841)  evaluator_time: 0.0022 (0.0022)
time: 0.0987  data: 0.0117  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0614 (0.0672)  evaluator_time: 0.0016 (0.0023)
time: 0.0778  data: 0.0126  max mem: 13577
Test: Total time: 0:00:04 (0.0842 s / it)
Averaged stats: model_time: 0.0614 (0.0672)  evaluator_time: 0.0016 (0.0023)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.255
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.658
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.107
 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 ] = 0.001
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.285
 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 ] = 0.427
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.452
 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 ] = 0.492
Epoch: [8]  [ 0/60]  eta: 0:00:21  lr: 0.000050  loss: 0.2255 (0.2255)  loss_classifier:
0.0732 (0.0732)  loss_box_reg: 0.1081 (0.1081)  loss_objectness: 0.0320 (0.0320)
loss_rpn_box_reg: 0.0122 (0.0122)  time: 0.3532  data: 0.0234  max mem: 13577
Epoch: [8]  [10/60]  eta: 0:00:17  lr: 0.000050  loss: 0.2585 (0.2776)  loss_classifier:
0.0870 (0.0859)  loss_box_reg: 0.1270 (0.1322)  loss_objectness: 0.0390 (0.0416)
loss_rpn_box_reg: 0.0140 (0.0179)  time: 0.3519  data: 0.0271  max mem: 13577
Epoch: [8]  [20/60]  eta: 0:00:14  lr: 0.000050  loss: 0.2585 (0.2887)  loss_classifier:
0.0882 (0.0894)  loss_box_reg: 0.1270 (0.1377)  loss_objectness: 0.0411 (0.0443)
loss_rpn_box_reg: 0.0140 (0.0174)  time: 0.3521  data: 0.0271  max mem: 13577

Epoch: [8]  [30/60]  eta: 0:00:10  lr: 0.000050  loss: 0.2462 (0.3003)  loss_classifier:
0.0859 (0.0912)  loss_box_reg: 0.1124 (0.1417)  loss_objectness: 0.0479 (0.0482)
loss_rpn_box_reg: 0.0130 (0.0191)  time: 0.3544  data: 0.0272  max mem: 13577
```

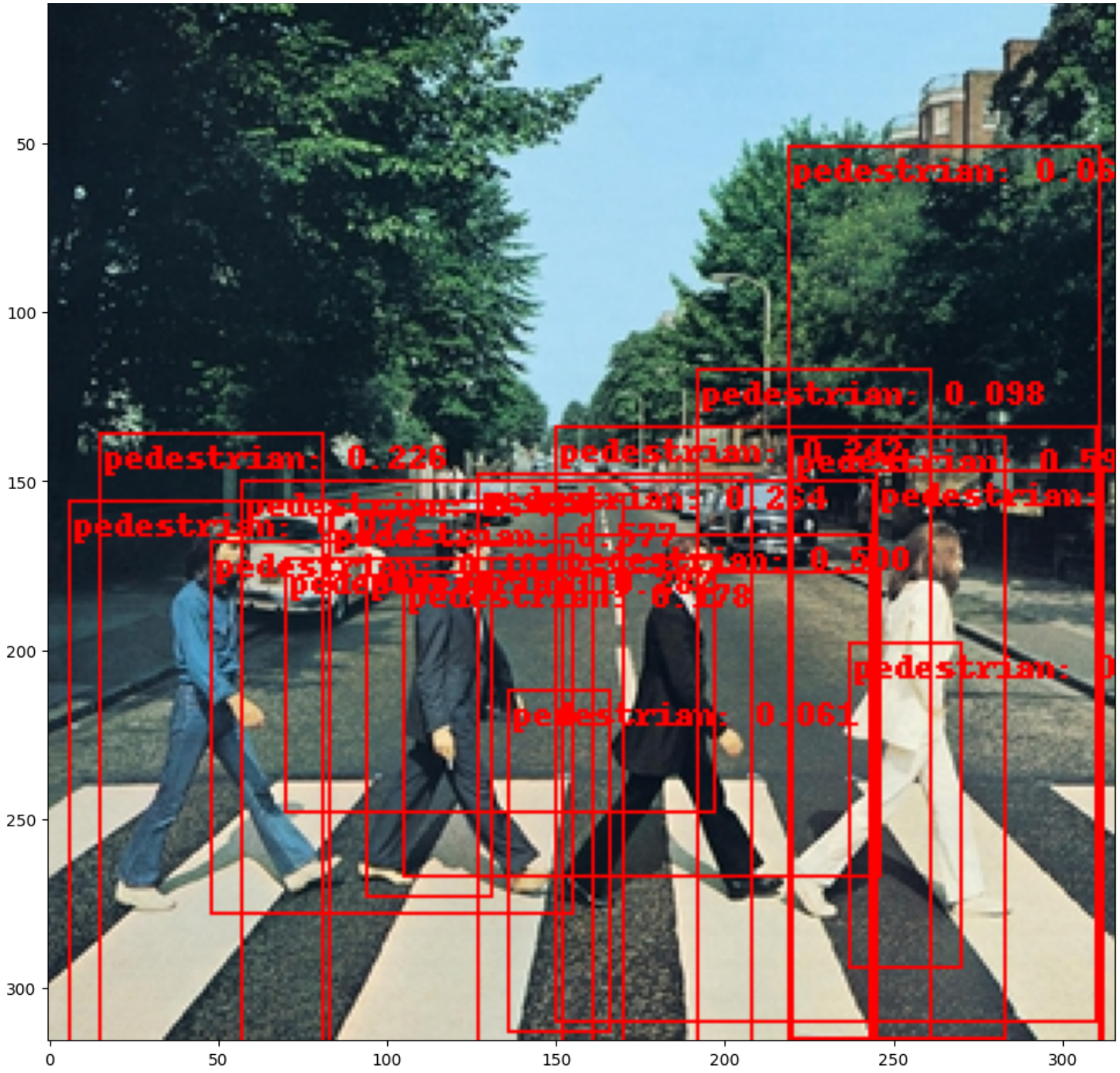
```
loss_rpn_box_reg: 0.0128 (0.0191)  time: 0.3544  data: 0.0272  max mem: 13577
Epoch: [8]  [40/60]  eta: 0:00:07  lr: 0.000050  loss: 0.3002 (0.3212)  loss_classifier:
0.0921 (0.0971)  loss_box_reg: 0.1475 (0.1531)  loss_objectness: 0.0475 (0.0509)
loss_rpn_box_reg: 0.0183 (0.0202)  time: 0.3622  data: 0.0288  max mem: 13577
Epoch: [8]  [50/60]  eta: 0:00:03  lr: 0.000050  loss: 0.3099 (0.3193)  loss_classifier:
0.0986 (0.0962)  loss_box_reg: 0.1511 (0.1521)  loss_objectness: 0.0475 (0.0508)
loss_rpn_box_reg: 0.0190 (0.0201)  time: 0.3685  data: 0.0298  max mem: 13577
Epoch: [8]  [59/60]  eta: 0:00:00  lr: 0.000050  loss: 0.3078 (0.3136)  loss_classifier:
0.0859 (0.0950)  loss_box_reg: 0.1388 (0.1489)  loss_objectness: 0.0431 (0.0502)
loss_rpn_box_reg: 0.0170 (0.0195)  time: 0.3680  data: 0.0306  max mem: 13577
Epoch: [8] Total time: 0:00:21 (0.3604 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0798 (0.0798)  evaluator_time: 0.0012 (0.0012)
time: 0.0901  data: 0.0086  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0641 (0.0668)  evaluator_time: 0.0016 (0.0020)
time: 0.0796  data: 0.0126  max mem: 13577
Test: Total time: 0:00:04 (0.0825 s / it)
Averaged stats: model_time: 0.0641 (0.0668)  evaluator_time: 0.0016 (0.0020)
Accumulating evaluation results...
DONE (t=0.02s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.260
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.668
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.110
 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 ] = 0.012
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.287
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.190
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.412
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.427
 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 ] = 0.022
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.465
Epoch: [9]  [ 0/60]  eta: 0:00:22  lr: 0.000005  loss: 0.3433 (0.3433)  loss_classifier:
0.1244 (0.1244)  loss_box_reg: 0.1495 (0.1495)  loss_objectness: 0.0454 (0.0454)
loss_rpn_box_reg: 0.0239 (0.0239)  time: 0.3696  data: 0.0260  max mem: 13577
Epoch: [9]  [10/60]  eta: 0:00:18  lr: 0.000005  loss: 0.3433 (0.3450)  loss_classifier:
0.1176 (0.1109)  loss_box_reg: 0.1586 (0.1660)  loss_objectness: 0.0454 (0.0504)
loss_rpn_box_reg: 0.0179 (0.0178)  time: 0.3737  data: 0.0329  max mem: 13577
Epoch: [9]  [20/60]  eta: 0:00:14  lr: 0.000005  loss: 0.2860 (0.3264)  loss_classifier:
0.0818 (0.1020)  loss_box_reg: 0.1315 (0.1515)  loss_objectness: 0.0453 (0.0546)
loss_rpn_box_reg: 0.0165 (0.0183)  time: 0.3614  data: 0.0295  max mem: 13577
Epoch: [9]  [30/60]  eta: 0:00:10  lr: 0.000005  loss: 0.2700 (0.3164)  loss_classifier:
0.0832 (0.0982)  loss_box_reg: 0.1310 (0.1485)  loss_objectness: 0.0417 (0.0515)
loss_rpn_box_reg: 0.0178 (0.0181)  time: 0.3585  data: 0.0274  max mem: 13577

Epoch: [9]  [40/60]  eta: 0:00:07  lr: 0.000005  loss: 0.2760 (0.3095)  loss_classifier:
0.0834 (0.0960)  loss_box_reg: 0.1424 (0.1460)  loss_objectness: 0.0359 (0.0495)
loss_rpn_box_reg: 0.0178 (0.0179)  time: 0.3583  data: 0.0271  max mem: 13577
```

```
loss_rpn_box_reg: 0.0178 (0.0179)  time: 0.3583  data: 0.0271  max mem: 13577
Epoch: [9]  [50/60]  eta: 0:00:03  lr: 0.000005  loss: 0.2891 (0.3193)  loss_classifier:
0.0834 (0.0970)  loss_box_reg: 0.1424 (0.1502)  loss_objectness: 0.0457 (0.0518)
loss_rpn_box_reg: 0.0194 (0.0203)  time: 0.3616  data: 0.0283  max mem: 13577
Epoch: [9]  [59/60]  eta: 0:00:00  lr: 0.000005  loss: 0.2806 (0.3094)  loss_classifier:
0.0703 (0.0939)  loss_box_reg: 0.1424 (0.1463)  loss_objectness: 0.0421 (0.0499)
loss_rpn_box_reg: 0.0175 (0.0193)  time: 0.3663  data: 0.0288  max mem: 13577
Epoch: [9] Total time: 0:00:21 (0.3625 s / it)
creating index...
index created!
Test:  [ 0/50]  eta: 0:00:04  model_time: 0.0782 (0.0782)  evaluator_time: 0.0013 (0.0013)
time: 0.0885  data: 0.0084  max mem: 13577
Test:  [49/50]  eta: 0:00:00  model_time: 0.0707 (0.0696)  evaluator_time: 0.0023 (0.0025)
time: 0.0906  data: 0.0161  max mem: 13577
Test: Total time: 0:00:04 (0.0876 s / it)
Averaged stats: model_time: 0.0707 (0.0696)  evaluator_time: 0.0023 (0.0025)
Accumulating evaluation results...
DONE (t=0.03s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.257
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.688
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.133
 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 ] = 0.013
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.287
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.164
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.422
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.441
 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 ] = 0.044
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.478
```

```python
import matplotlib.pyplot as plt

from torchvision.utils import draw_bounding_boxes, draw_segmentation_masks


image = read_image("Beatles_-_Abbey_Road.jpg")
eval_transform = get_transform(train=False)

model.eval()
with torch.no_grad():
    x = eval_transform(image)
    # convert RGBA -> RGB and move to device
    x = x[:3, ...].to(device)
    predictions = model([x, ])
    pred = predictions[0]
```

```
    pred = predictions[0]


image = (255.0 * (image - image.min()) / (image.max() - image.min())).to(torch.uint8)
image = image[:3, ...]
pred_labels = [f"pedestrian: {score:.3f}" for label, score in zip(pred["labels"],
pred["scores"])]
pred_boxes = pred["boxes"].long()
output_image = draw_bounding_boxes(image, pred_boxes, pred_labels, colors="red")

# masks = (pred["masks"] > 0.7).squeeze(1)
# output_image = draw_segmentation_masks(output_image, masks, alpha=0.5, colors="blue")


plt.figure(figsize=(12, 12))
plt.imshow(output_image.permute(1, 2, 0))
```

```
<matplotlib.image.AxesImage at 0x79c67ea2ed70>
```



(b). On the training data, option 1 perform much better than option 2.

(c). On the testing picture, option 1 also perform much better than option 2. Option 1 is a MaskRCNN model, it not only detects objects, gives their bounding boxes, but also provides a pixel-wise mask for each detected object. This mask can differentiate pixels belonging to the object from the background.