

## Assignment 4

CMSC 691 — Computer Vision

Faisal Rasheed Khan

VB02734

[vb02734@umbc.edu](mailto:vb02734@umbc.edu)

### Question 1-----

#### Task 1

```
import torch
import torchvision
import torchvision.transforms as transforms

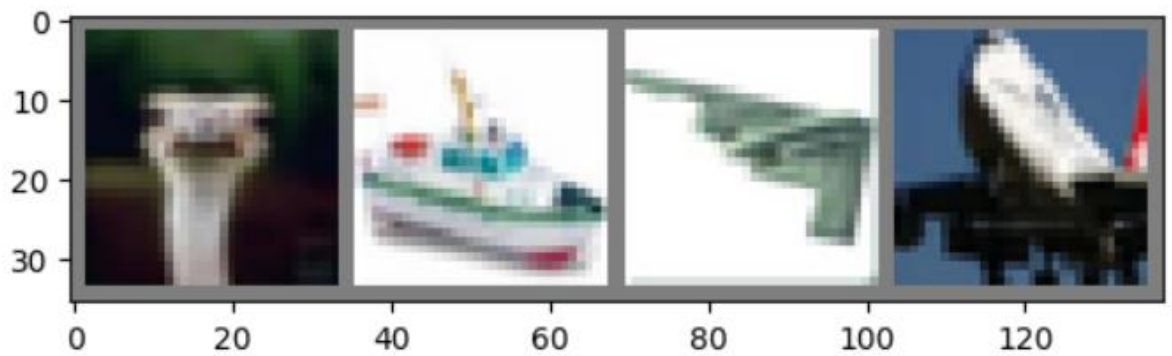
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
batch_size = 4
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                         download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                           shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                         download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                          shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

import matplotlib.pyplot as plt
import numpy as np
def imshow(img):
    img = img / 2 + 0.5     # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

# get some random training images
dataiter = iter(trainloader)
images, labels = next(dataiter)
# show images
imshow(torchvision.utils.make_grid(images))
# print labels
print(' '.join(f'{classes[labels[j]]:5s}' for j in range(batch_size)))
```



bird ship plane plane

```
import torch.nn as nn
import torch.nn.functional as F
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
net = Net()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
for epoch in range(2): # loop over the dataset multiple times
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        if i % 2000 == 1999: # print every 2000 mini-batches
            print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / 2000:.3f}')
            running_loss = 0.0
print('Finished Training')
```

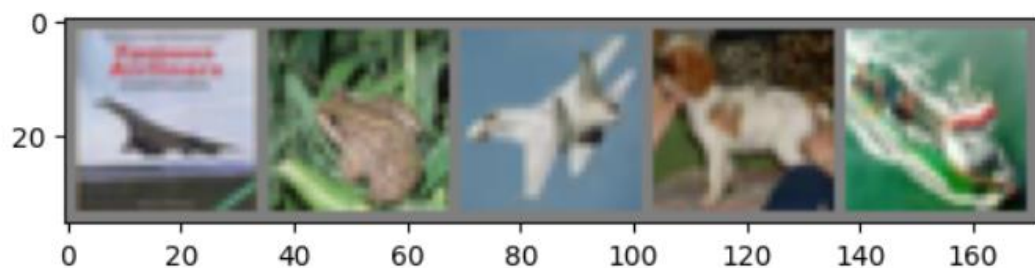
```
[1, 2000] loss: 2.206
[1, 4000] loss: 1.849
[1, 6000] loss: 1.670
[1, 8000] loss: 1.584
[1, 10000] loss: 1.513
[1, 12000] loss: 1.489
[2, 2000] loss: 1.427
[2, 4000] loss: 1.368
[2, 6000] loss: 1.327
[2, 8000] loss: 1.347
[2, 10000] loss: 1.307
[2, 12000] loss: 1.289
Finished Training
```

```
[ ] correct = 0
total = 0
with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f'Accuracy of the network on the 10000 test images: {100 * correct // total} %')
```

Accuracy of the network on the 10000 test images: 55 %

## Task 2



GroundTruth: plane frog plane dog ship

```
exclude_indices = {0, 1, 2, 5, 6, 7, 8, 9, 11, 13, 14}
outputs = net(images)
_, predicted = torch.max(outputs, 1)

print('Predicted: ', ' '.join(f'{classes[predicted[j]]:5s}'
                                for j in range(16) if j not in exclude_indices))
```

Predicted: ship deer deer frog car

### Task 3

- A. The learning rate 0.001 is good learning rate, so tuned with various epochs and batch size increasing and the at epoch 30 and batch size 25 got the best accuracy of 65%.
- B.

	Epochs	Batch Size	Learning Rate	Accuracy
1.	2	4	0.001	54
2.	4	4	0.001	59
3.	2	6	0.001	53
4.	4	6	0.001	61
5.	4	10	0.001	54
6.	10	50	0.001	55
7.	10	50	0.01	63
8.	30	25	0.01	57
9.	30	25	0.001	65
10.	50	30	0.001	61
11.	15	20	0.001	63

### Task 4

- A. The better compared to the normal without augmentation is

```

transform = transforms.Compose([
    transforms.RandomRotation(2),
    transforms.RandomHorizontalFlip(),
    transforms.RandomAffine(degrees=0, translate=(0.01, 0.01)),
    transforms.ToTensor(), # Convert images to PyTorch tensors
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # Normalize the data
])

```

```

Sample 100000
[1, 2000] loss: 2.129
[1, 4000] loss: 1.784
[1, 6000] loss: 1.657
[1, 8000] loss: 1.590
[1, 10000] loss: 1.500
[1, 12000] loss: 1.462
[2, 2000] loss: 1.424
[2, 4000] loss: 1.365
[2, 6000] loss: 1.361
[2, 8000] loss: 1.330
[2, 10000] loss: 1.340
[2, 12000] loss: 1.321
Finished Training

```

The accuracy I got is 55% which is just 1% better where the normal setting gave 54%. The rotation is 2 in magnitude, horizontal flip, random affine is 0 degree and 1% magnitude.

I have also tried for the rotation of 5,7 with the same affine of degree 0 and 10% magnitude, the 5 rotation gave 51% and 7 rotation gave 48%.

After some tries with better transformations I got better by 1% with 2 rotation and the accuracy would also be increased with increasing epochs

- B. The performance improved by 1%
- C. The data augmentation provides the diverse range of the training instances to the model to be trained.

The data augmentation helps in improving accuracy for the unseen data by those transformations.

The data augmentation help in avoiding overfitting by just having nice images, instead with data augmentation helps in testing the challenge images

- D. Adding noise to the image

Resizing the image

Data Augmenting

Testing on the ensemble methods such that the training images learns various models

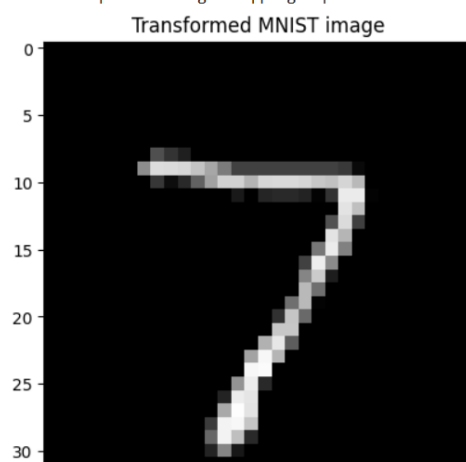
#### Task 5

1. The classifiers prediction will be of one of the classes of 10 CIFAR classes based on their trained weights.
2. The confidence for the unseen class might be low compared to the seen classes, as the model weights are trained on the seen weights. The model predicts for the unseen images using the trained weights of the seen classes.
3. The computer vision models to overcome these failures are to have transfer learning, ensemble learning, Data Augmentation.

```
# Visualize the image
# Squeeze the batch dimension and transpose the image for correct display
img_to_show = mnist_img_cifar.squeeze().permute(1, 2, 0) # From (C, H, W) to (H, W, C)

plt.imshow(img_to_show)
plt.title('Transformed MNIST image')
plt.show()
```

WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



```


# MNIST to CIFAR transform
transform_cifar = transforms.Compose([
    transforms.Resize(32), # Resize to 32x32 to match CIFAR-10 dimensions
    transforms.ToPILImage(), # Convert to PIL image to apply the next transformation
    transforms.Grayscale(num_output_channels=3), # Convert grayscale to RGB by replicating channels
    transforms.ToTensor(), # Convert back to tensor
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # Normalize like CIFAR-10
])

# Apply the transform
mnist_img_cifar = transform_cifar(mnist_img)

net.eval()
with torch.no_grad():
    mnist_img_cifar = mnist_img_cifar.unsqueeze(0) # Add batch dimension
    outputs = net(mnist_img_cifar) # Predict
    _, predicted = torch.max(outputs, 1) # Get the predicted class

# Output the prediction
print(f'Predicted CIFAR-10 class: {predicted.item()}')
print(classes[predicted])

```

 Predicted CIFAR-10 class: 2  
bird

## Question 2-----

### Task 1

```

import os
import numpy as np
import torch
import torchvision
import cv2
from PIL import Image
from matplotlib import pyplot as plt
from torchvision import transforms as T

```

### 2. Task 1

```

[ ] # Download a pretrained model
model0 = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model1 = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
model2 = torchvision.models.detection.retinanet_resnet50_fpn(pretrained=True)
model3 = torchvision.models.detection.ssd_lite320_mobilenet_v3_large(pretrained=True)

# Inference
model0.eval()
model1.eval()
model2.eval()
model3.eval()

(1): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
(2): ReLU6(inplace=True)
)
(1): Conv2d(128, 546, kernel_size=(1, 1), stride=(1, 1))
)
)

```

### Task 2

```
[ ] # Defining PyTorch Transform
transform = T.Compose([T.ToTensor()])
# Define class names from MS-COCO dataset
COCO_INSTANCE_CATEGORY_NAMES = [
    '__background__', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
    'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'N/A', 'stop sign',
    'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
    'elephant', 'bear', 'zebra', 'giraffe', 'N/A', 'backpack', 'umbrella', 'N/A', 'N/A',
    'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball',
    'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket',
    'bottle', 'N/A', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl',
    'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
    'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'N/A', 'dining table',
    'N/A', 'N/A', 'toilet', 'N/A', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
    'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'N/A', 'book',
    'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush'
]
```

```
def get_detection(img, model, threshold=0.5):
    pred = model([img]) # Pass the image to the model
    # pred is a list and each element of that list is a dictionary with keys: "labels", "scores", and "boxes"
    pred_class = [COCO_INSTANCE_CATEGORY_NAMES[i] for i in list(pred[0]['labels'].numpy())] # Get the Prediction Classes
    # !!!! Complete the following
    pred_boxes = [(i[0], i[1]), (i[2], i[3])] for i in list(pred[0]['boxes'].detach().numpy()) # Get the Prediction Boxes
    pred_score = list(pred[0]['scores'].detach().numpy()) # Get the Prediction Scores
    pred_t = [pred_score.index(x) for x in pred_score if x > threshold][-1] # Get list of index with score greater than threshold.
    pred_boxes = pred_boxes[:pred_t+1]
    pred_class = pred_class[:pred_t+1]

    return pred_boxes, pred_class
```

```
def show_detections(img_path, model, threshold=0.5):
    img = Image.open(img_path) # Load the image
    img = img.convert("RGB") # Convert image to RGB format
    img = transform(img) # Apply the transform to the image
    boxes, pred_cls = get_detection(img, model, threshold) # Get predictions
    img = cv2.imread(img_path) # Read image with cv2
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # Convert to RGB
    # !!!! Complete the following
    # write code to display the image, overlay the bounding boxes and predicted classes on top of the image
    plt.figure(figsize=(8,14)) # display the output image
    plt.imshow(img)
    plt.xticks([])
    plt.yticks([])
    # !!!!
    # Overlay bounding boxes and predicted classes
    ax = plt.gca()
    for i in range(len(boxes)):
        x1, y1 = boxes[i][0]
        x2, y2 = boxes[i][1]
        width = x2 - x1
        height = y2 - y1
        box = plt.Rectangle((x1, y1), width, height, linewidth=2, edgecolor='r', facecolor='none')
        ax.add_patch(box)
        plt.text(x1, y1, s=pred_cls[i], color='white', verticalalignment='top', bbox={'color': 'red', 'pad': 0})

    plt.show()
```

### Task 3



```
!wget https://www.tejasgokhale.com/images/vehicle.png -O vehicle.jpg
# code to use show_detections to display results for `model0,model1,model2,model3`
# !!!! Complete this

import cv2

# Read the image using OpenCV
img = cv2.imread("vehicle.jpg")

# Convert the image from BGR to RGB (OpenCV reads images in BGR format)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

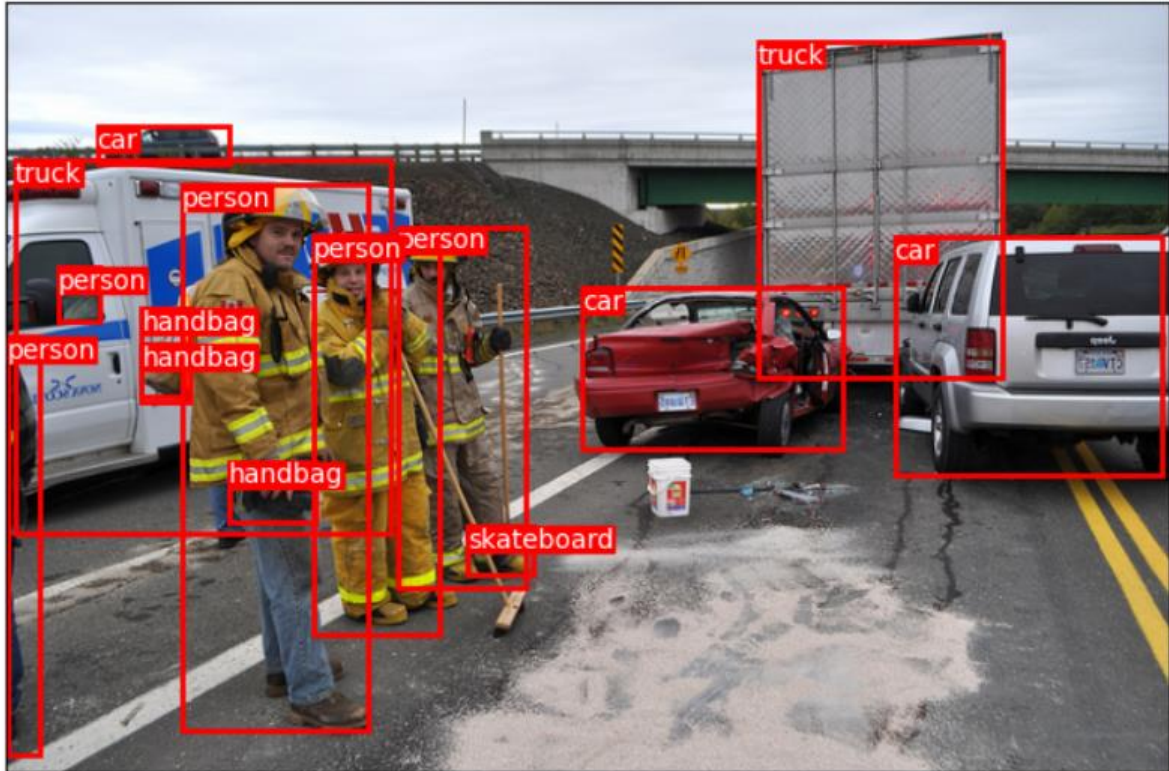
# Display the image using matplotlib
plt.imshow(img_rgb)
plt.axis('off') # Hide axes
plt.show()
```



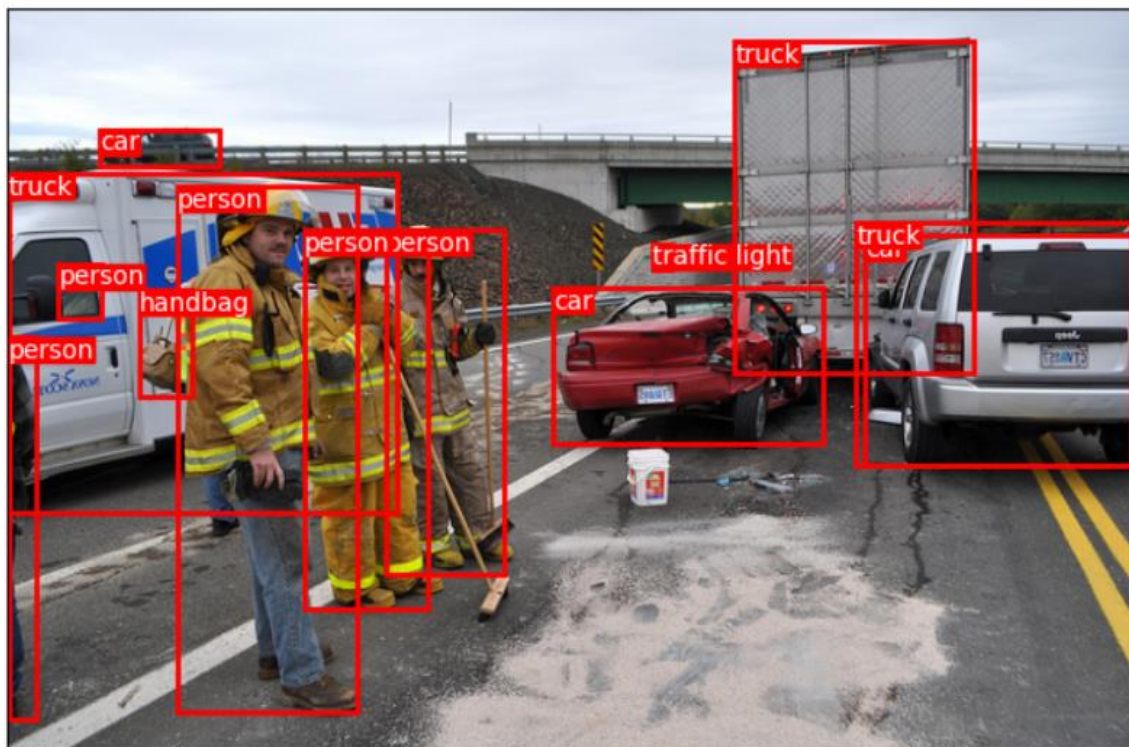
```
# Download a pretrained model
model0 = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model1 = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
model2 = torchvision.models.detection.retinanet_resnet50_fpn(pretrained=True)
model3 = torchvision.models.detection.ssd_lite320_mobilenet_v3_large(pretrained=True)
```



```
threshold = 0.5
img="vehicle.jpg"
show_detections(img, model0, threshold)
```



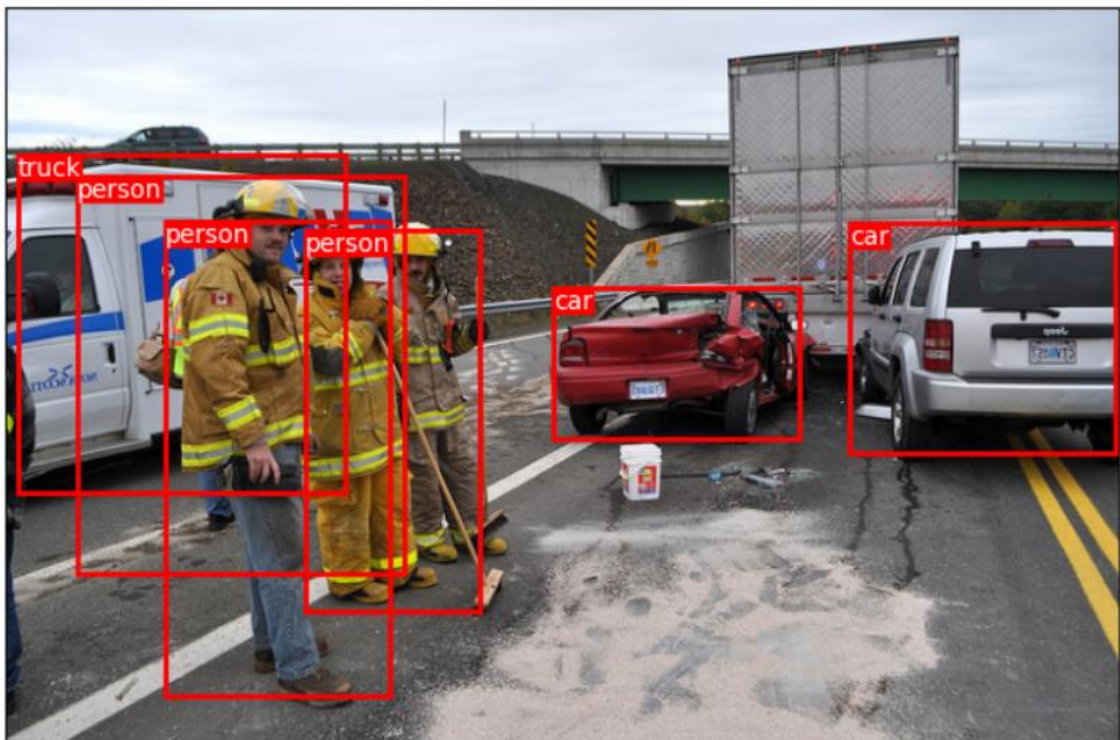
```
show_detections(img, model1, threshold)
```



▶ `show_detections(img, model2, threshold)`

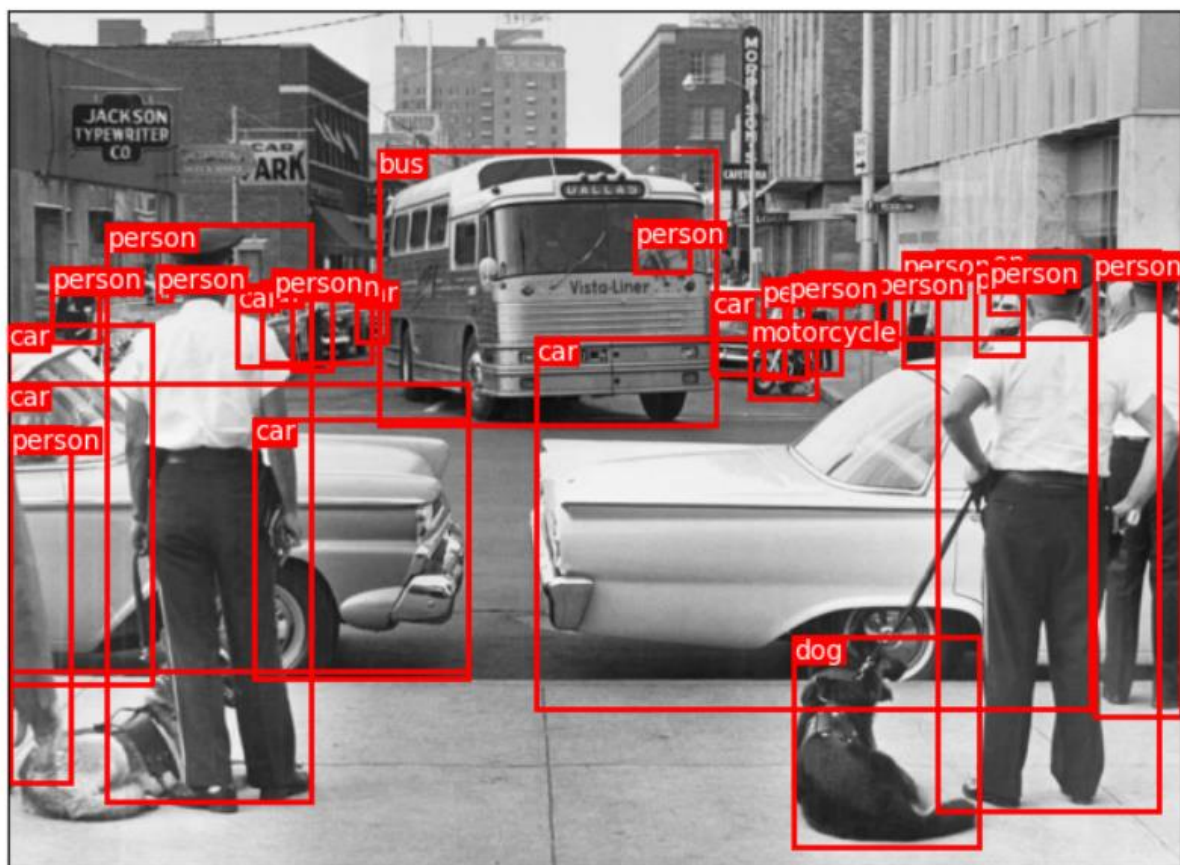


▶ `show_detections(img, model3, threshold)`

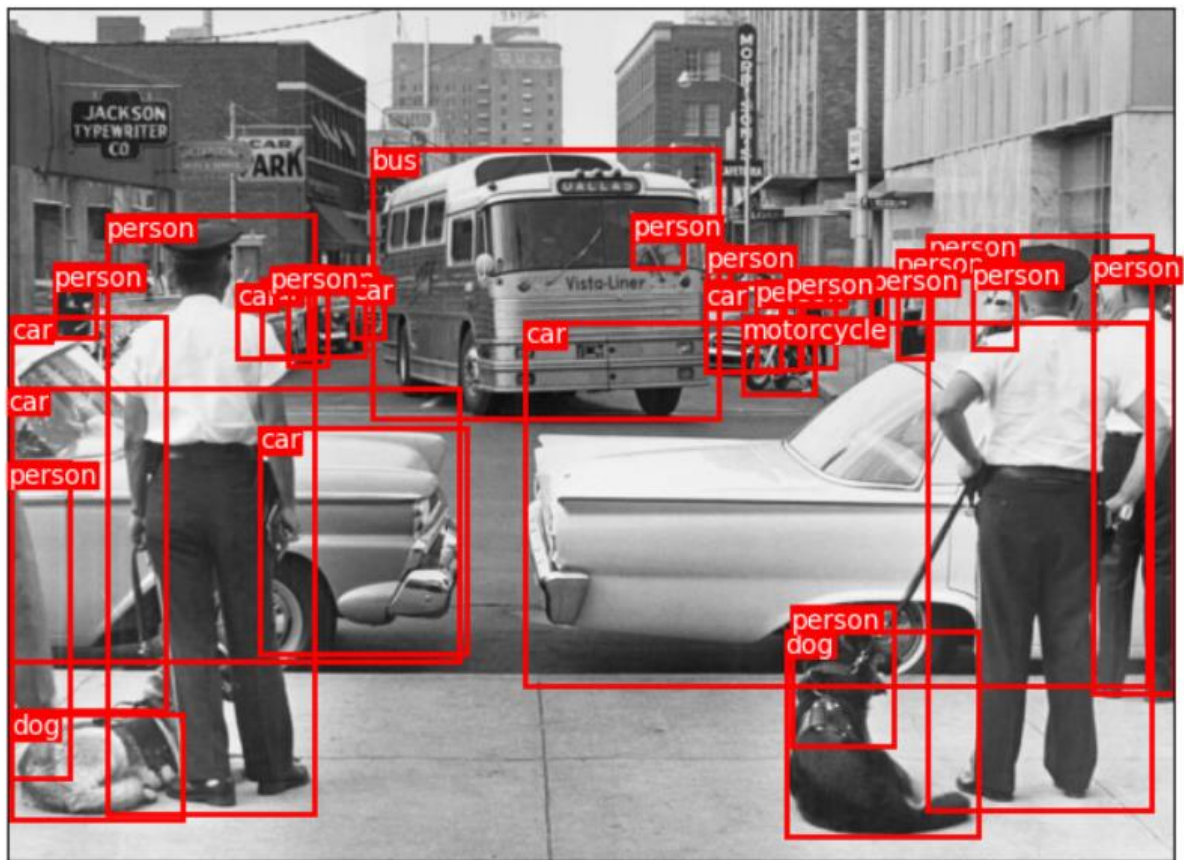


Model 0:

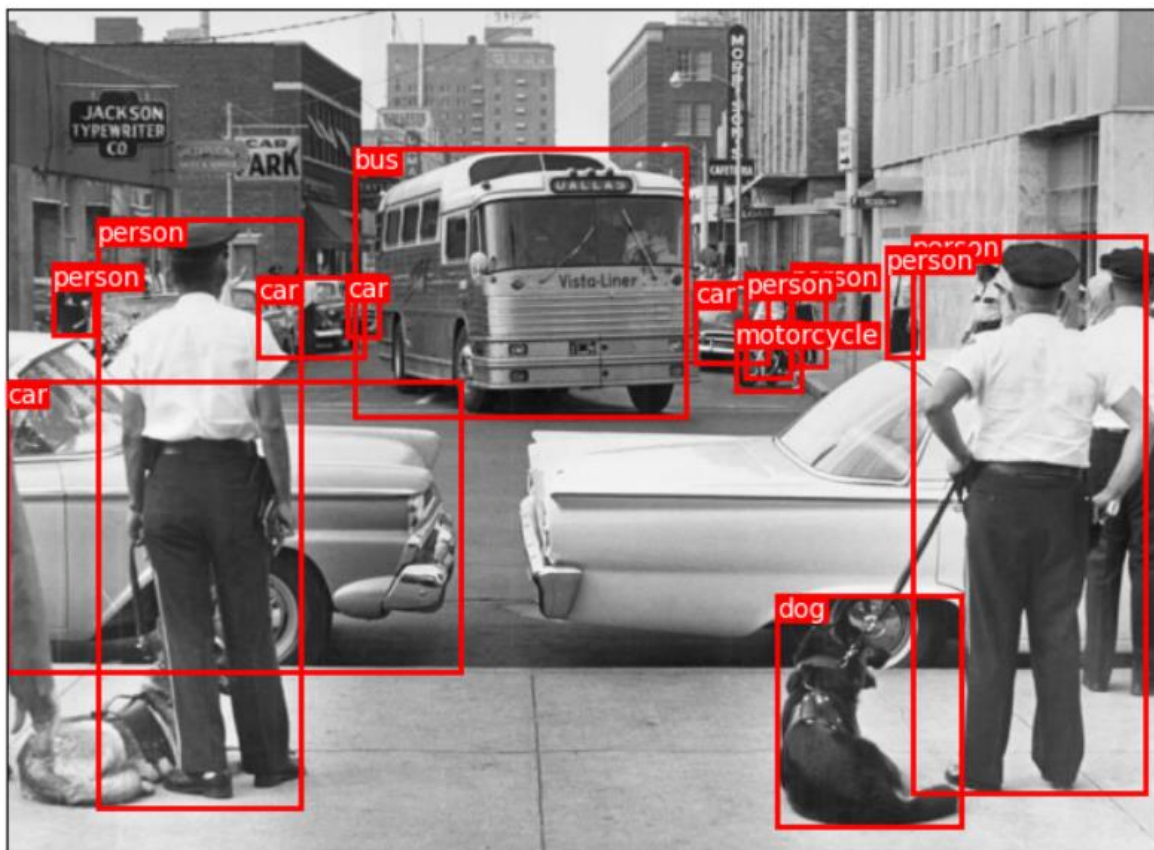




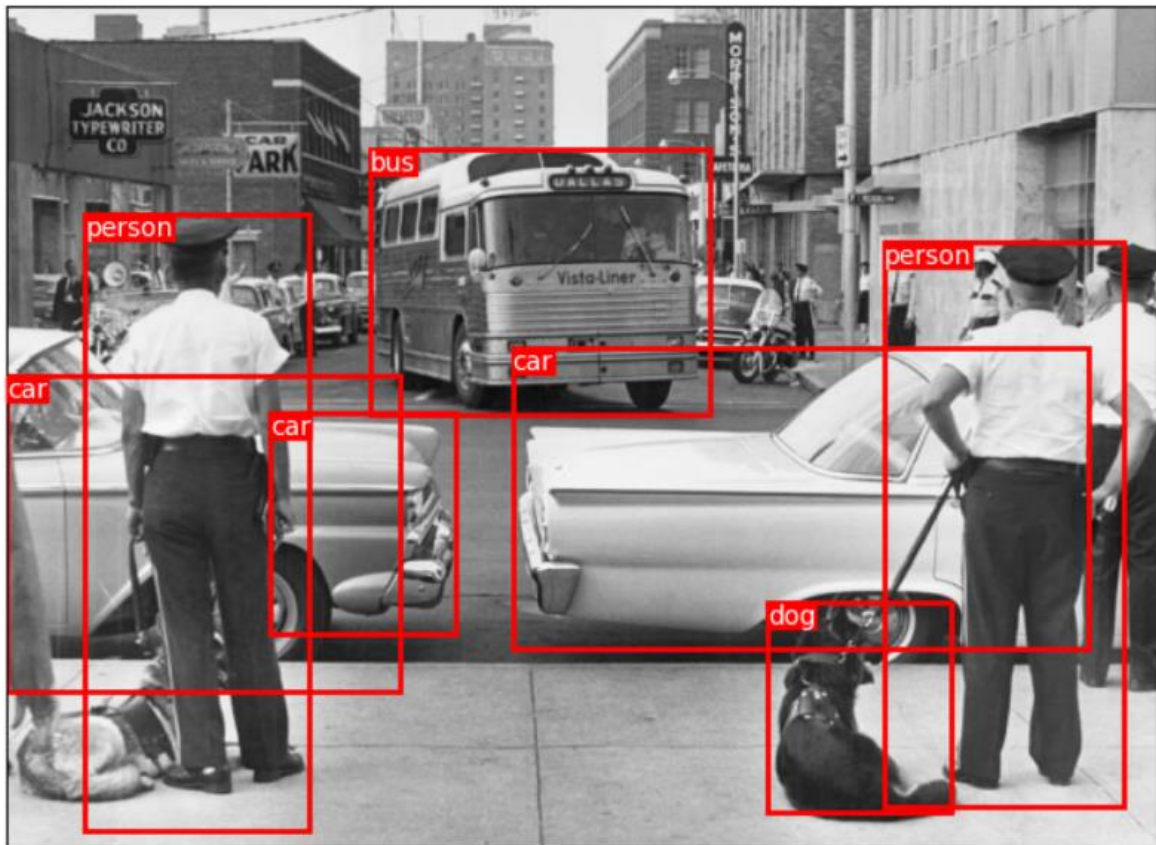
```
show_detections(img, model1, threshold)
```



```
show_detections(img, model2, threshold)
```



```
show_detections(img, model3, threshold)
```



Task 4



```

def eval_iou(pred_box, gt_box):
    # Convert the predicted and ground-truth boxes to lists
    pred_box = list(pred_box)
    gt_box = list(gt_box)
    #print(pred_box,gt_box)
    # Calculate intersection coordinates
    x_i = max(pred_box[0], gt_box[0])
    y_i = max(pred_box[1], gt_box[1])
    w_i = max(0, min(pred_box[0] + pred_box[2], gt_box[0] + gt_box[2]) - x_i)
    h_i = max(0, min(pred_box[1] + pred_box[3], gt_box[1] + gt_box[3]) - y_i)

    # Calculate areas of predicted and ground-truth boxes
    area_pred = pred_box[2] * pred_box[3]
    area_gt = gt_box[2] * gt_box[3]

    # Calculate area of intersection and union
    area_intersection = w_i * h_i
    area_union = area_pred + area_gt - area_intersection

    # Calculate IoU
    iou = area_intersection / (area_union + 1e-5)

    return iou

```

```

coco_annotations_file="./coco_ann2017/annotations/instances_val2017.json";coco_images_dir="./coco_val2017/val2017"
coco= COCOParser(coco_annotations_file, coco_images_dir);img_ids = coco.get_imgIds();img_ids = img_ids[:100];
iou_threshold=0.5;iou_scores_model0 = [];precision_scores = [];recall_scores = []
for i, im in enumerate(img_ids):
    image = Image.open(f"{coco_images_dir}/{str(im).zfill(12)}.jpg")
    image = transform(image)
    pred_boxes, pred_cls = get_detection(image, model0)
    ann_ids = coco.get_annIds(im)
    annotations = coco.load_anns(ann_ids)
    tp, fp, fn = 0, 0, 0;iou_scores_image = []
    for ann in annotations:
        gt_bbox = ann['bbox']
        gt_class_id = ann["category_id"]
        max_iou = 0
        for pred_box, pred_cls in zip(pred_boxes, pred_cls):
            iou = eval_iou(pred_box, gt_bbox)
            iou_scores_image.append(iou)
        mean_iou_image = np.argmax(iou_scores_image)
        iou_scores_model0.append(iou_scores_image[mean_iou_image])
        if iou_scores_image[mean_iou_image] >= iou_threshold:
            tp += 1
        else:
            fn += 1
    fp = len(pred_boxes) - tp
    precision_image = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall_image = tp / (tp + fn) if (tp + fn) > 0 else 0
    precision_scores.append(precision_image)
    recall_scores.append(recall_image)
    mean_iou_image = np.mean(iou_scores_image)
    iou_scores_model0.append(mean_iou_image)
mean_iou_model0 = np.nanmean(iou_scores_model0)
mean_precision = np.mean(precision_scores)
mean_recall = np.mean(recall_scores)
print(f"Mean IoU for model0: {mean_iou_model0}")
print(f"Mean Precision for model0: {mean_precision}")
print(f"Mean Recall for model0: {mean_recall}")

```



Mean IoU for model0: 0.3824964032292413  
Mean Precision for model0: 0.28058503332416374  
Mean Recall for model0: 0.36738095238095236

Mean IoU for model1: 0.3963833513268418  
Mean Precision for model1: 0.2829211714623283  
Mean Recall for model1: 0.3798809523809523

Mean IoU for model2: 0.3599976840119967  
Mean Precision for model2: 0.42337254901960786  
Mean Recall for model2: 0.3216428571428572

---

Mean IoU for model3: 0.3249436235323119  
Mean Precision for model3: 0.7909999999999999  
Mean Recall for model3: 0.2923571428571429

### Question 3-----

#### 3.1-----

1. Summarize the talk in a couple of paragraphs. You can write about the big-picture topic of the talk and specific projects (tasks, methods, results, etc.) (4 points)

The talk focuses on the evaluation metrics for Text-to-image models and its efficiency over the years. The main problem for these tasks are to integrate are multi-linguality, generative tasks and evaluation metrics. The author Micheal Sexon introduced the evaluation metrics T2IScoreScore which addresses the issue while evaluating the text-to-image for the improvements which lacked earlier. The evaluation performance are very important for the text-to-image tasks, the talk presented the technical comparisons between models and its metrics such as TS2. The author mentions the weakness of the TS2 metrics. With the proposed TSIScoreScore provides good evaluation compared to the previous methods which do not see those mistakes. TS2 aims for the development of better T2I prompt faithfulness metrics through improvements.

2. Questions/Discussions: If you asked a question, please summarize that question and the answer from the speaker. If you didnt ask a question, please summarize any question that was asked and the answer from the speaker. (3 points)

The question I asked was how important the evaluation metric is for the Text-to-image tasks, as the text give the output?

The answer I got was the improvement of the prompts to get the better image, the speaker gave comparison of the mistakes which was shown with the previous metrics and shown the improvements with their TS2 metric.

3. What was your favorite part of the talk / discussion?

My favorite part of the talk / discussion was the improvement of the prompts to generate the images based on the texts. And the details presented while showing the examples. The results shown giving the improved results was very nice to see. The critical examinations of benchmarks and metrics

4. Cite one of the speaker's published paper that they discussed in detail during the talk. (2 points) (1 points)

Saxon, M., Jahara, F., Khoshnoodi, M., Lu, Y., Sharma, A., & Wang, W. Y. (2024). Who Evaluates the Evaluations? Objectively Scoring Text-to-Image Prompt Coherence Metrics with T2IScoreScore (TS2). arXiv preprint arXiv:2404.04251. <https://arxiv.org/pdf/2404.04251>

### 3.2-----

1. Summarize the talk in a couple of paragraphs. You can write about the big-picture topic of the talk and specific projects (tasks, methods, results, etc.) (4 points)

The talk focuses on the effective Scene Composing of the image by performing the image synthesis. The speaker Y. Zeng proposes the novel approach for generating the image. The text-to-image prompt is converted to the semantic-to-image to generate semantic synthesis of image called SceneComposer. Traditional methods focus on the low level details but the SceneComposer aims to focus on the semantic description to match the described objects to fine the details of the image. They map the text details into the 2D canvas to have the appropriate details in order to get the fine results. The results based on the evaluation metric T2I and S2I, both of them gave good results when they applied SceneComposee which is better than the Text-to-Image

2. Questions/Discussions: If you asked a question, please summarize that question and the answer from the speaker. If you didnt ask a question, please summarize any question that was asked and the answer from the speaker. (3 points)

The question I asked was the chance of handling the synthesis of hypothetical scenarios?

The answer I got was SceneComposer is designed to have flexibility in synthesizing semantic images at any level. The model is trained to handled on different types of data and can be handled based on the text provided and each has its own trained images and then they are combined based on the SceneComposer effectively.

3. What was your favorite part of the talk / discussion?

My favorite part of the talk / discussion was the approach of solving the image synthesis generating text to 2D Canvas and then generating the images and with this the images can be generated accurately which are compares with the metrics compared to the other techniques.

4. Cite one of the speaker's published paper that they discussed in detail during the talk. (2 points) (1 points)

Y. Zeng, Z. Lin, J. Zhang, Q. Liu, J. Collomosse, J. Kuen, V. Patel, "Scenecomposer: Any-level semantic image synthesis," CVPR, 2023 (Hightlight, top 2.5% ).  
<https://arxiv.org/pdf/2403.09632>

#### References:

- [1] [COCO - Common Objects in Context \(cocodataset.org\)](https://cocodataset.org)
- [2] [Training a Classifier — PyTorch Tutorials 2.3.0+cu121 documentation](#)
- [3] [CMSC491/691 HW4 Starter Code and Instructions - Colab \(google.com\)](#)
- [4] Saxon, M., Jahara, F., Khoshnoodi, M., Lu, Y., Sharma, A., & Wang, W. Y. (2024). Who Evaluates the Evaluations? Objectively Scoring Text-to-Image Prompt Coherence Metrics with T2IScoreScore (TS2). arXiv preprint arXiv:2404.04251. <https://arxiv.org/pdf/2404.04251>
- [5] Y. Zeng, Z. Lin, J. Zhang, Q. Liu, J. Collomosse, J. Kuen, V. Patel, "Scenecomposer: Any-level semantic image synthesis," CVPR, 2023 (Hightlight, top 2.5% ). <https://arxiv.org/pdf/2403.09632>
- [6] [Matplotlib — Visualization with Python](#)
- [7] [Python Documentation contents — Python 3.12.2 documentation](#)
- [8] [NumPy —](#)
- [9] [OpenCV - Open Computer Vision Library](#)