cktran_09859713

November 10, 2021

1 EECS 442 Fall 2021 PS7: Object Detection

Calvin Tran, cktran

2 Description

In this problem set, you will be implementing parts of a single stage object detector closely related to YOLOv1. We will also evaluate the detection accuracy using the classic metric mean Average Precision (mAP).

Feel free to take a look at the paper linked above, or this comprehensive summary of the paper by Sik-Ho Tsang.

Wait a minute! Before you begin, check out this cool video of the YOLOv1 object detector watching sports.

3 Setup

Run the following code to import the modules you'll need. After your finish the assignment, remember to run all cells and save the note book as PDF (according to directions in the problem set) and submit the PDF file to Gradescope.

[1]:
```
!pip install git+https://github.com/deepvision-class/starter-code
```

```
Collecting git+https://github.com/deepvision-class/starter-code
  Cloning https://github.com/deepvision-class/starter-code to /tmp/pip-req-
build-80tiw8_f
  Running command git clone -q https://github.com/deepvision-class/starter-code
/tmp/pip-req-build-80tiw8_f
Requirement already satisfied: pydrive in /usr/local/lib/python3.7/dist-packages
(from Colab-Utils==0.1.dev0) (1.3.1)
Requirement already satisfied: PyYAML>=3.0 in /usr/local/lib/python3.7/dist-
packages (from pydrive->Colab-Utils==0.1.dev0) (3.13)
Requirement already satisfied: google-api-python-client>=1.2 in
/usr/local/lib/python3.7/dist-packages (from pydrive->Colab-Utils==0.1.dev0)
(1.12.8)
Requirement already satisfied: oauth2client>=4.0.0 in /usr/local/lib/python3.7
/dist-packages (from pydrive->Colab-Utils==0.1.dev0) (4.1.3)
```

```
Requirement already satisfied: google-auth-httplib2>=0.0.3 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.2->pydrive->Colab-Utils==0.1.dev0) (0.0.4)
Requirement already satisfied: google-auth>=1.16.0 in /usr/local/lib/python3.7
/dist-packages (from google-api-python-client>=1.2->pydrive->Colab-
Utils==0.1.dev0) (1.35.0)
Requirement already satisfied: six<2dev,>=1.13.0 in /usr/local/lib/python3.7
/dist-packages (from google-api-python-client>=1.2->pydrive->Colab-
Utils==0.1.dev0) (1.15.0)
Requirement already satisfied: httplib2<1dev,>=0.15.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.2->pydrive->Colab-Utils==0.1.dev0) (0.17.4)
Requirement already satisfied: google-api-core<2dev,>=1.21.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.2->pydrive->Colab-Utils==0.1.dev0) (1.26.3)
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.2->pydrive->Colab-Utils==0.1.dev0) (3.0.1)
Requirement already satisfied: setuptools>=40.3.0 in /usr/local/lib/python3.7
/dist-packages (from google-api-core<2dev,>=1.21.0->google-api-python-
client>=1.2->pydrive->Colab-Utils==0.1.dev0) (57.4.0)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-core<2dev,>=1.21.0
->google-api-python-client>=1.2->pydrive->Colab-Utils==0.1.dev0) (2.23.0)
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages
(from google-api-core<2dev,>=1.21.0->google-api-python-client>=1.2->pydrive
->Colab-Utils==0.1.dev0) (2018.9)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-core<2dev,>=1.21.0
->google-api-python-client>=1.2->pydrive->Colab-Utils==0.1.dev0) (1.53.0)
Requirement already satisfied: protobuf>=3.12.0 in /usr/local/lib/python3.7
/dist-packages (from google-api-core<2dev,>=1.21.0->google-api-python-
client>=1.2->pydrive->Colab-Utils==0.1.dev0) (3.17.3)
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.7/dist-
packages (from google-api-core<2dev,>=1.21.0->google-api-python-
client>=1.2->pydrive->Colab-Utils==0.1.dev0) (21.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7
/dist-packages (from google-auth>=1.16.0->google-api-python-client>=1.2->pydrive
->Colab-Utils==0.1.dev0) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-
packages (from google-auth>=1.16.0->google-api-python-client>=1.2->pydrive
->Colab-Utils==0.1.dev0) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth>=1.16.0->google-api-
python-client>=1.2->pydrive->Colab-Utils==0.1.dev0) (4.2.4)
Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.7/dist-
packages (from oauth2client>=4.0.0->pydrive->Colab-Utils==0.1.dev0) (0.4.8)
Requirement already satisfied: pyparsing<3,>=2.0.2 in /usr/local/lib/python3.7
```

```
/dist-packages (from packaging>=14.3->google-api-core<2dev,>=1.21.0->google-api-
python-client>=1.2->pydrive->Colab-Utils==0.1.dev0) (2.4.7)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7
/dist-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2dev,>=1.21.0
->google-api-python-client>=1.2->pydrive->Colab-Utils==0.1.dev0) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7
/dist-packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2dev,>=1.21.0
->google-api-python-client>=1.2->pydrive->Colab-Utils==0.1.dev0) (2021.10.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests<3.0.0dev,>=2.18.0->google-
api-core<2dev,>=1.21.0->google-api-python-client>=1.2->pydrive->Colab-
Utils==0.1.dev0) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests<3.0.0dev,>=2.18.0->google-api-core<2dev,>=1.21.0
->google-api-python-client>=1.2->pydrive->Colab-Utils==0.1.dev0) (2.10)
```

[2]:
```python
import math
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import coutils
from coutils import extract_drive_file_id, register_colab_notebooks, \
                    fix_random_seed, rel_error
import matplotlib.pyplot as plt
import numpy as np
import cv2
import copy
import time
import shutil
import os

# parameters for plotting
plt.rcParams['figure.figsize'] = (10.0, 8.0) # set default size of plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'

# data type and device for torch.tensor
# unpack as argument to torch functions, like so: **to_float
to_float = {'dtype': torch.float, 'device': 'cpu'}
to_float_cuda = {'dtype': torch.float, 'device': 'cuda'}
to_double = {'dtype': torch.double, 'device': 'cpu'}
to_double_cuda = {'dtype': torch.double, 'device': 'cuda'}
to_long = {'dtype': torch.long, 'device': 'cpu'}
to_long_cuda = {'dtype': torch.long, 'device': 'cuda'}

# for mAP evaluation
```

```
!rm -rf mAP
!git clone https://github.com/Cartucho/mAP.git
!rm -rf mAP/input/*
```

```
Cloning into 'mAP'...
remote: Enumerating objects: 908, done.
remote: Total 908 (delta 0), reused 0 (delta 0), pack-reused 908
Receiving objects: 100% (908/908), 14.71 MiB | 19.74 MiB/s, done.
Resolving deltas: 100% (321/321), done.
```

We will use GPUs to accelerate our computation in this notebook. Run the following to make sure GPUs are enabled:

[3]:
```
if torch.cuda.is_available():
  print('Good to go!')
else:
  print('Please set GPU via Edit -> Notebook Settings.')
```

```
Good to go!
```

3.1 Load PASCAL VOC 2007 dataset

Unlike the CIFAR-10 and the MiniPlaces dataset we used in previous problem sets for image classificaiton, PASCAL VOC is one of the standard datasets used for object detection.

Object detection datasets provide images which are annotated with a set of bounding boxes, where each box gives the category label and spatial extent of some object present in the image.

We will use the PASCAL VOC 2007 dataset, which provides annotations of this form. PASCAL VOC ran a series of yearly computer vision competitions from 2005 to 2012, predating the ImageNet challenge for image classification.

The data from the 2007 challenge used to be one of the most popular datasets for evaluating object detection. It is much smaller than more recent object detection datasets such as COCO, and thus easier to manage in a problem set.

The following function will download the PASCAL VOC 2007 dataset and return it as a PyTorch Dataset object:

[4]:
```
def get_pascal_voc2007_data(image_root, split='train'):
  """
  Use torchvision.datasets
  https://pytorch.org/docs/stable/torchvision/datasets.html#torchvision.
 ↪datasets.VOCDetection
  """
  from torchvision import datasets

  train_dataset = datasets.VOCDetection(image_root, year='2007',␣
 ↪image_set=split,
                                        download=True)

  return train_dataset
```

Run the following cell to download the training and validation sets for the PASCAL VOC 2007 dataset. Downloading might take a minute or two, but should not take much longer.

The `Dataset` objects returned from the above function returns annotations for each image as a nested set of dictionary objects:

[5]:
```
# uncomment below to use the mirror link if the original link is broken
# !wget http://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar
!wget https://www.eecs.umich.edu/courses/eecs442-ahowens/fa20/pascal_voc/
 ↪VOCtrainval_06-Nov-2007.tar
train_dataset = get_pascal_voc2007_data('/content', 'train')
val_dataset = get_pascal_voc2007_data('/content', 'val')
# an example on the raw annotation
import json
print("============== Raw Annotation Example ==============")
print(json.dumps(train_dataset[1][1]['annotation'], indent=2))
```

```
--2021-11-09 23:52:53--  https://www.eecs.umich.edu/courses/eecs442-ahowens/fa20
/pascal_voc/VOCtrainval_06-Nov-2007.tar
Resolving www.eecs.umich.edu (www.eecs.umich.edu)... 141.212.113.199
Connecting to www.eecs.umich.edu (www.eecs.umich.edu)|141.212.113.199|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 460032000 (439M) [application/x-tar]
Saving to: VOCtrainval_06-Nov-2007.tar.1

VOCtrainval_06-Nov- 100%[===================>] 438.72M  10.9MB/s    in 44s

2021-11-09 23:53:38 (9.91 MB/s) - VOCtrainval_06-Nov-2007.tar.1 saved
[460032000/460032000]

Using downloaded and verified file: /content/VOCtrainval_06-Nov-2007.tar
Extracting /content/VOCtrainval_06-Nov-2007.tar to /content
Using downloaded and verified file: /content/VOCtrainval_06-Nov-2007.tar
Extracting /content/VOCtrainval_06-Nov-2007.tar to /content
============== Raw Annotation Example ==============
{
  "folder": "VOC2007",
  "filename": "000017.jpg",
  "source": {
    "database": "The VOC2007 Database",
    "annotation": "PASCAL VOC2007",
    "image": "flickr",
    "flickrid": "228217974"
  },
  "owner": {
    "flickrid": "genewolf",
    "name": "whiskey kitten"
  },
```

```
  "size": {
    "width": "480",
    "height": "364",
    "depth": "3"
  },
  "segmented": "0",
  "object": [
    {
      "name": "person",
      "pose": "Left",
      "truncated": "0",
      "difficult": "0",
      "bndbox": {
        "xmin": "185",
        "ymin": "62",
        "xmax": "279",
        "ymax": "199"
      }
    },
    {
      "name": "horse",
      "pose": "Left",
      "truncated": "0",
      "difficult": "0",
      "bndbox": {
        "xmin": "90",
        "ymin": "78",
        "xmax": "403",
        "ymax": "336"
      }
    }
  ]
}
```


In order to use these annotations to train our model, we need to convert this nested dictionary data structure into a set of PyTorch tensors.

We also need to preprocess the image, converting it to a PyTorch tensor and resizing it to 224x224. Real object detection systems typically work with much higher-resolution images, but we will use a low resolution for computational efficiency in this problem set.

We also want to train our models using minibatches of data, so we need to group the annotations from several images into minibatches.

We perform both of these functions by using a customized PyTorch DataLoader object, which we have written for you:

[6]:
```
def pascal_voc2007_loader(dataset, batch_size, num_workers=0):
  """
  Data loader for Pascal VOC 2007.
  https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader
```

```
  """
  from torch.utils.data import DataLoader
  # turn off shuffle so we can index the original image
  train_loader = DataLoader(dataset,
                            batch_size=batch_size,
                            shuffle=False, pin_memory=True,
                            num_workers=num_workers,
                            collate_fn=voc_collate_fn)
  return train_loader


class_to_idx = {'aeroplane':0, 'bicycle':1, 'bird':2, 'boat':3, 'bottle':4,
                'bus':5, 'car':6, 'cat':7, 'chair':8, 'cow':9, 'diningtable':10,
                'dog':11, 'horse':12, 'motorbike':13, 'person':14,␣
 ↪'pottedplant':15,
                'sheep':16, 'sofa':17, 'train':18, 'tvmonitor':19
}
idx_to_class = {i:c for c, i in class_to_idx.items()}


from torchvision import transforms

def voc_collate_fn(batch_lst, reshape_size=224):
    preprocess = transforms.Compose([
      transforms.Resize((reshape_size, reshape_size)),
      transforms.ToTensor(),
      transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
 ↪225]),
      ])

    batch_size = len(batch_lst)

    img_batch = torch.zeros(batch_size, 3, reshape_size, reshape_size)

    max_num_box = max(len(batch_lst[i][1]['annotation']['object']) \
                      for i in range(batch_size))

    box_batch = torch.Tensor(batch_size, max_num_box, 5).fill_(-1.)
    w_list = []
    h_list = []
    img_id_list = []

    for i in range(batch_size):
      img, ann = batch_lst[i]
      w_list.append(img.size[0]) # image width
      h_list.append(img.size[1]) # image height
      img_id_list.append(ann['annotation']['filename'])
```

```
      img_batch[i] = preprocess(img)
      all_bbox = ann['annotation']['object']
      if type(all_bbox) == dict: # inconsistency in the annotation file
        all_bbox = [all_bbox]
      for bbox_idx, one_bbox in enumerate(all_bbox):
        bbox = one_bbox['bndbox']
        obj_cls = one_bbox['name']
        box_batch[i][bbox_idx] = torch.Tensor([float(bbox['xmin']),␣
↪float(bbox['ymin']),
          float(bbox['xmax']), float(bbox['ymax']), class_to_idx[obj_cls]])

    h_batch = torch.tensor(h_list)
    w_batch = torch.tensor(w_list)

    return img_batch, box_batch, w_batch, h_batch, img_id_list
```

Training with the entire PASCAL VOC will be too computationally expensive for this homework assignment, so we can subsample the dataset by wrapping each `Dataset` object in a `Subset` object:

[7]:
```
train_dataset = torch.utils.data.Subset(train_dataset, torch.arange(0, 2500)) #␣
↪use 2500 samples for training
train_loader = pascal_voc2007_loader(train_dataset, 10)
val_loader = pascal_voc2007_loader(val_dataset, 10)
```

The `DataLoader` objects return batches of data.

The first output from the `DataLoader` is a Tensor `img` of shape `(B, 3, 224, 224)`. This is a batch of `B` images, similar to what we have seen in classification datasets.

The second output from the `DataLoader` is a Tensor `ann` of shape `(B, N, 5)` giving information about all objects in all images of the batch. `ann[i, j] = (x_tl, y_tl, x_br, y_br, class)` gives information about the `j`th object in `img[i]`. The position of the top-left corner of the box is `(x_tl, y_tl)` and the position of the bottom-right corner of the box is `(x_br, y_br)`. These positions are in the coordinate system of the original image (before it was resized to 224 x 224). `class` is an integer giving the category label for this bounding box.

Each image can have different numbers of objects. If `img[i]` has N_i objects, then $N = \max_i N_i$ is the maximum number of objects per image among all objects in the batch; this value can vary from batch to batch. For the images that have fewer than N annotated objects, only the first N_i rows of $anns[i]$ contain annotations; the remaining rows are padded with -1.

[8]:
```
train_loader_iter = iter(train_loader)
img, ann, _, _, _ = train_loader_iter.next()

print('img has shape: ', img.shape)
print('ann has shape: ', ann.shape)

print('Image 1 has only two annotated objects, so ann[1] is padded with -1:')
print(ann[1])

print('\nImage 2 has six annotated objects:, so ann[2] is not padded:')
```

```
print(ann[2])

print('\nEach row in the annotation tensor indicates (x_tl, y_tl, x_br, y_br,␣
 ↪class).')
```

```
img has shape:  torch.Size([10, 3, 224, 224])
ann has shape:  torch.Size([10, 6, 5])
Image 1 has only two annotated objects, so ann[1] is padded with -1:
tensor([[185.,  62., 279., 199.,  14.],
        [ 90.,  78., 403., 336.,  12.],
        [ -1.,  -1.,  -1.,  -1.,  -1.],
        [ -1.,  -1.,  -1.,  -1.,  -1.],
        [ -1.,  -1.,  -1.,  -1.,  -1.],
        [ -1.,  -1.,  -1.,  -1.,  -1.]])

Image 2 has six annotated objects:, so ann[2] is not padded:
tensor([[  9., 230., 245., 500.,   1.],
        [230., 220., 334., 500.,   1.],
        [  2., 178.,  90., 500.,   1.],
        [  2.,   1., 117., 369.,  14.],
        [  3.,   2., 243., 462.,  14.],
        [225.,   1., 334., 486.,  14.]])

Each row in the annotation tensor indicates (x_tl, y_tl, x_br, y_br, class).
```

4 Coordinate Transformation

It's a good practice to use a consistent coordinate system for all the spatial-related computations (e.g., bboxes or proposals). In this problem set, **we use the coordinate system defined by the CNN activation map (of shape 7x7), where the top-left corner is (0, 0) and the bottom-right corner is (7, 7). The horizontal axis is the x axis and the vertical axis is the y axis.**

The following function defines the transformation from the original image coordinate system (pixels, and the top-left corner is (0, 0)) to the activation map coordinate system and vice versa.

Notes: All the coordinates are in float precision. In later sections, **we use the activation map coordinate system for all computations except for visualization.**

[9]:
```
def coord_trans(bbox, w_pixel, h_pixel, w_amap=7, h_amap=7, mode='a2p'):
    """
    Coordinate transformation function. It converts the box coordinate from
    the image coordinate system to the activation map coordinate system and vice␣
 ↪versa.
    In our case, the input image will have a few hundred pixels in
    width/height while the activation map is of size 7x7.

    Input:
    - bbox: Could be either bbox, anchor, or proposal, of shape Bx*x4
```

```
 - w_pixel: Number of pixels in the width side of the original image, of shape␣
↪B
 - h_pixel: Number of pixels in the height side of the original image, of␣
↪shape B
 - w_amap: Number of pixels in the width side of the activation map, scalar
 - h_amap: Number of pixels in the height side of the activation map, scalar
 - mode: Whether transfer from the original image to activation map ('p2a') or
         the opposite ('a2p')

 Output:
 - resized_bbox: Resized box coordinates, of the same shape as the input bbox
 """

 assert mode in ('p2a', 'a2p'), 'invalid coordinate transformation mode!'
 assert bbox.shape[-1] >= 4, 'the transformation is applied to the first 4␣
↪values of dim -1'

 if bbox.shape[0] == 0: # corner cases
   return bbox

 resized_bbox = bbox.clone()
 # could still work if the first dim of bbox is not batch size
 # in that case, w_pixel and h_pixel will be scalars
 resized_bbox = resized_bbox.view(bbox.shape[0], -1, bbox.shape[-1])
 invalid_bbox_mask = (resized_bbox == -1) # indicating invalid bbox

 if mode == 'p2a':
   # pixel to activation
   width_ratio = w_pixel * 1. / w_amap
   height_ratio = h_pixel * 1. / h_amap
   resized_bbox[:, :, [0, 2]] /= width_ratio.view(-1, 1, 1)
   resized_bbox[:, :, [1, 3]] /= height_ratio.view(-1, 1, 1)
 else:
   # activation to pixel
   width_ratio = w_pixel * 1. / w_amap
   height_ratio = h_pixel * 1. / h_amap
   resized_bbox[:, :, [0, 2]] *= width_ratio.view(-1, 1, 1)
   resized_bbox[:, :, [1, 3]] *= height_ratio.view(-1, 1, 1)

 resized_bbox.masked_fill_(invalid_bbox_mask, -1)
 resized_bbox.resize_as_(bbox)
 return resized_bbox
```

5 Data Visualizer

This function will help us visualize boxes on top of images.

[10]:
```
def data_visualizer(img, idx_to_class, bbox=None, pred=None):
  """
  Data visualizer on the original image. Support both GT box input and proposal
  input.

  Input:
  - img: PIL Image input
  - idx_to_class: Mapping from the index (0-19) to the class name
  - bbox: GT bbox (in red, optional), a tensor of shape Nx5, where N is
          the number of GT boxes, 5 indicates (x_tl, y_tl, x_br, y_br, class)
  - pred: Predicted bbox (in green, optional), a tensor of shape N'x6, where
          N' is the number of predicted boxes, 6 indicates
          (x_tl, y_tl, x_br, y_br, class, object confidence score)
  """

  img_copy = np.array(img).astype('uint8')

  if bbox is not None:
    for bbox_idx in range(bbox.shape[0]):
      one_bbox = bbox[bbox_idx][:4]
      cv2.rectangle(img_copy, (one_bbox[0], one_bbox[1]), (one_bbox[2],
                  one_bbox[3]), (255, 0, 0), 2)
      if bbox.shape[1] > 4: # if class info provided
        obj_cls = idx_to_class[bbox[bbox_idx][4].item()]
        cv2.putText(img_copy, '%s' % (obj_cls),
                  (one_bbox[0], one_bbox[1]+15),
                  cv2.FONT_HERSHEY_PLAIN, 1.0, (0, 0, 255), thickness=1)

  if pred is not None:
    for bbox_idx in range(pred.shape[0]):
      one_bbox = pred[bbox_idx][:4]
      cv2.rectangle(img_copy, (one_bbox[0], one_bbox[1]), (one_bbox[2],
                  one_bbox[3]), (0, 255, 0), 2)

      if pred.shape[1] > 4: # if class and conf score info provided
        obj_cls = idx_to_class[pred[bbox_idx][4].item()]
        conf_score = pred[bbox_idx][5].item()
        cv2.putText(img_copy, '%s, %.2f' % (obj_cls, conf_score),
                    (one_bbox[0], one_bbox[1]+15),
                    cv2.FONT_HERSHEY_PLAIN, 1.0, (0, 0, 255), thickness=1)

  plt.imshow(img_copy)
  plt.axis('off')
  plt.show()
```

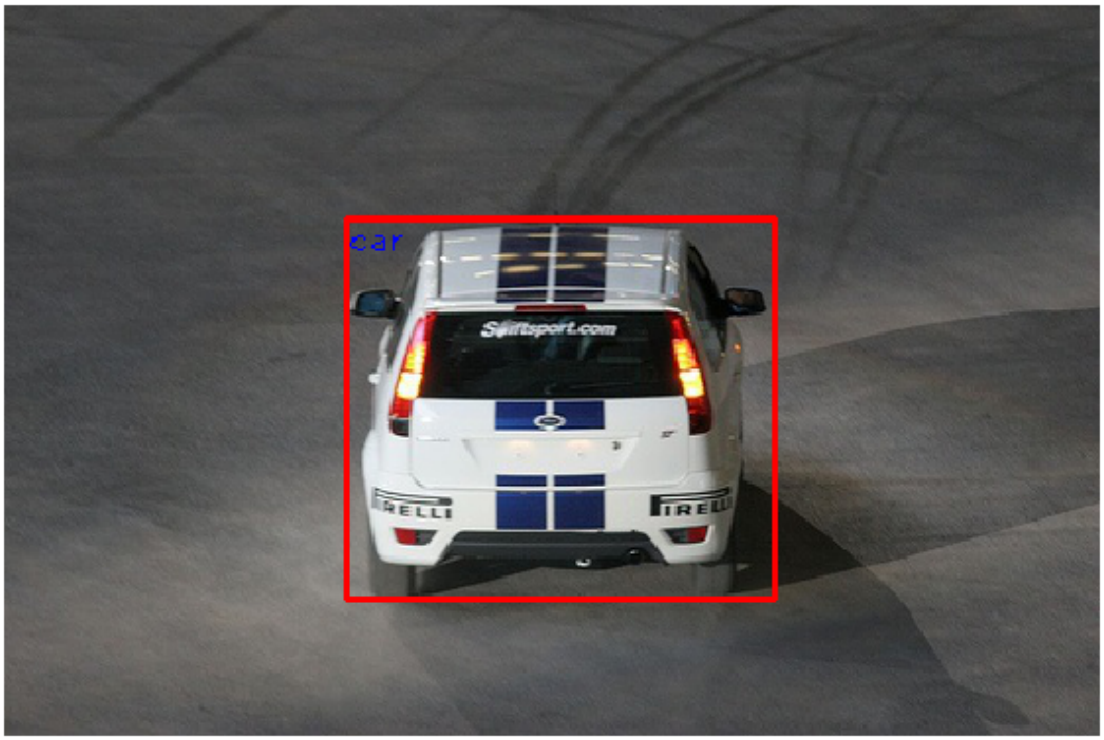
5.1 Visualize PASCAL VOC 2007

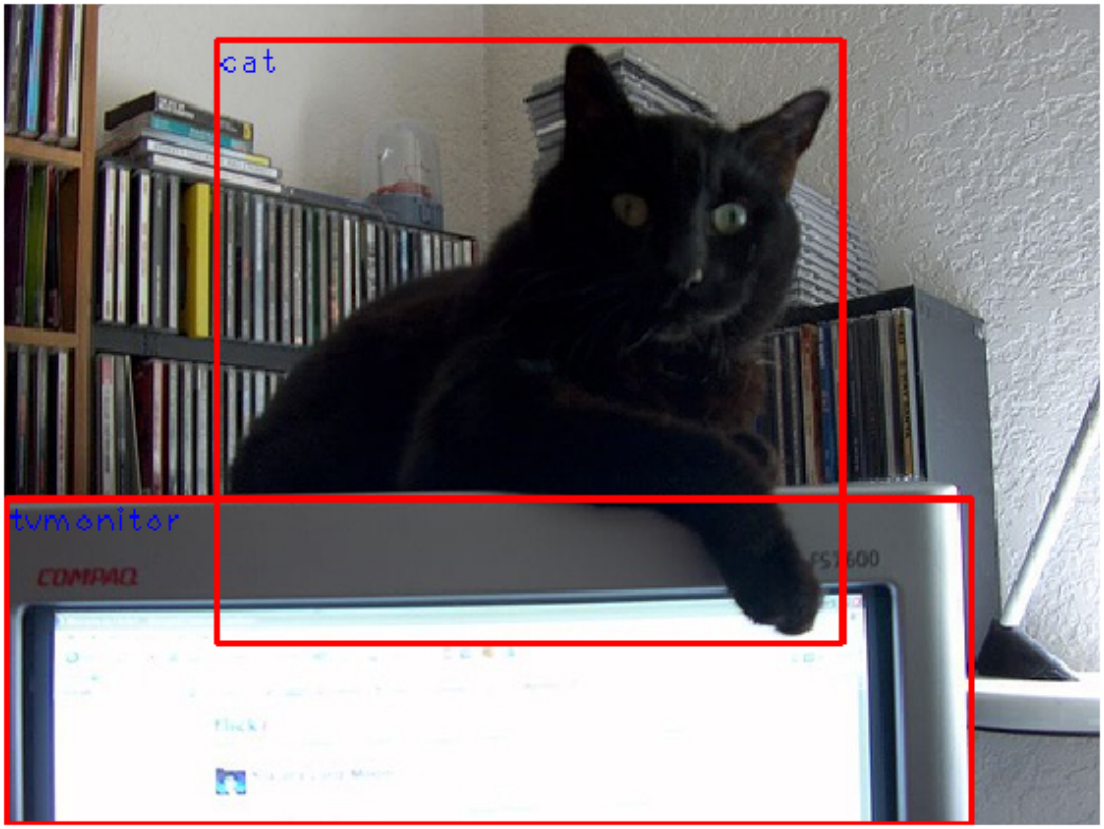
It is always good practice to try and visualize parts of your dataset before you build a model.

Here we sample some images from the PASCAL VOC 2007 training set, and visualize the ground-truth object boxes and category labels:

[11]:
```
# default examples for visualization
fix_random_seed(1)
batch_size = 3
sampled_idx = torch.linspace(0, len(train_dataset)-1, steps=batch_size).long()

# get the size of each image first
h_list = []
w_list = []
img_list = [] # list of images
MAX_NUM_BBOX = 40
box_list = torch.LongTensor(batch_size, MAX_NUM_BBOX, 5).fill_(-1) # PADDED GT␣
 ↪boxes

for idx, i in enumerate(sampled_idx):
  # hack to get the original image so we don't have to load from local again...
  img, ann = train_dataset.__getitem__(i)
  img_list.append(img)

  all_bbox = ann['annotation']['object']
  if type(all_bbox) == dict:
    all_bbox = [all_bbox]
  for bbox_idx, one_bbox in enumerate(all_bbox):
    bbox = one_bbox['bndbox']
    obj_cls = one_bbox['name']
    box_list[idx][bbox_idx] = torch.LongTensor([int(bbox['xmin']),␣
 ↪int(bbox['ymin']),
      int(bbox['xmax']), int(bbox['ymax']), class_to_idx[obj_cls]])

  # get sizes
  img = np.array(img)
  w_list.append(img.shape[1])
  h_list.append(img.shape[0])

w_list = torch.as_tensor(w_list, **to_float_cuda)
h_list = torch.as_tensor(h_list, **to_float_cuda)
box_list = torch.as_tensor(box_list, **to_float_cuda)
resized_box_list = coord_trans(box_list, w_list, h_list, mode='p2a') # on␣
 ↪activation map coordinate system
```

[12]:
```
# visualize GT boxes
for i in range(len(img_list)):
  valid_box = sum([1 if j != -1 else 0 for j in box_list[i][:, 0]])
  data_visualizer(img_list[i], idx_to_class, box_list[i][:valid_box])
```


car


cat
tvmonitor
COMPAQ

6 (a) Detector Backbone Network

Here, we use a MobileNet v2 for image feature extraction.

[13]:
```
class FeatureExtractor(nn.Module):
  """
  Image feature extraction with MobileNet.
  """
  def __init__(self, reshape_size=224, pooling=False, verbose=False):
    super().__init__()

    from torchvision import models
    from torchsummary import summary

    self.mobilenet = models.mobilenet_v2(pretrained=True)
    self.mobilenet = nn.Sequential(*list(self.mobilenet.children())[:-1]) #␣
↪Remove the last classifier
```

```
    # average pooling
    if pooling:
      self.mobilenet.add_module('LastAvgPool', nn.AvgPool2d(math.
↪ceil(reshape_size/32.))) # input: N x 1280 x 7 x 7

    for i in self.mobilenet.named_parameters():
      i[1].requires_grad = True # fine-tune all parameters

    if verbose:
      summary(self.mobilenet.cuda(), (3, reshape_size, reshape_size))

  def forward(self, img, verbose=False):
    """
    Inputs:
    - img: Batch of resized images, of shape Nx3x224x224

    Outputs:
    - feat: Image feature, of shape Nx1280 (pooled) or Nx1280x7x7
    """
    num_img = img.shape[0]

    img_prepro = img

    feat = []
    process_batch = 500
    for b in range(math.ceil(num_img/process_batch)):
      feat.append(self.mobilenet(img_prepro[b*process_batch:(b+1)*process_batch]
                                 ).squeeze(-1).squeeze(-1)) # forward and squeeze
    feat = torch.cat(feat)

    if verbose:
      print('Output feature shape: ', feat.shape)

    return feat
```

Now, let's see what's inside MobileNet v2. Assume we have a 3x224x224 image input.

[14]:
```
model = FeatureExtractor(verbose=True)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 112, 112]             864
       BatchNorm2d-2         [-1, 32, 112, 112]              64
             ReLU6-3         [-1, 32, 112, 112]               0
            Conv2d-4         [-1, 32, 112, 112]             288
       BatchNorm2d-5         [-1, 32, 112, 112]              64
             ReLU6-6         [-1, 32, 112, 112]               0
            Conv2d-7         [-1, 16, 112, 112]             512
```

BatchNorm2d-8	[-1, 16, 112, 112]	32
InvertedResidual-9	[-1, 16, 112, 112]	0
Conv2d-10	[-1, 96, 112, 112]	1,536
BatchNorm2d-11	[-1, 96, 112, 112]	192
ReLU6-12	[-1, 96, 112, 112]	0
Conv2d-13	[-1, 96, 56, 56]	864
BatchNorm2d-14	[-1, 96, 56, 56]	192
ReLU6-15	[-1, 96, 56, 56]	0
Conv2d-16	[-1, 24, 56, 56]	2,304
BatchNorm2d-17	[-1, 24, 56, 56]	48
InvertedResidual-18	[-1, 24, 56, 56]	0
Conv2d-19	[-1, 144, 56, 56]	3,456
BatchNorm2d-20	[-1, 144, 56, 56]	288
ReLU6-21	[-1, 144, 56, 56]	0
Conv2d-22	[-1, 144, 56, 56]	1,296
BatchNorm2d-23	[-1, 144, 56, 56]	288
ReLU6-24	[-1, 144, 56, 56]	0
Conv2d-25	[-1, 24, 56, 56]	3,456
BatchNorm2d-26	[-1, 24, 56, 56]	48
InvertedResidual-27	[-1, 24, 56, 56]	0
Conv2d-28	[-1, 144, 56, 56]	3,456
BatchNorm2d-29	[-1, 144, 56, 56]	288
ReLU6-30	[-1, 144, 56, 56]	0
Conv2d-31	[-1, 144, 28, 28]	1,296
BatchNorm2d-32	[-1, 144, 28, 28]	288
ReLU6-33	[-1, 144, 28, 28]	0
Conv2d-34	[-1, 32, 28, 28]	4,608
BatchNorm2d-35	[-1, 32, 28, 28]	64
InvertedResidual-36	[-1, 32, 28, 28]	0
Conv2d-37	[-1, 192, 28, 28]	6,144
BatchNorm2d-38	[-1, 192, 28, 28]	384
ReLU6-39	[-1, 192, 28, 28]	0
Conv2d-40	[-1, 192, 28, 28]	1,728
BatchNorm2d-41	[-1, 192, 28, 28]	384
ReLU6-42	[-1, 192, 28, 28]	0
Conv2d-43	[-1, 32, 28, 28]	6,144
BatchNorm2d-44	[-1, 32, 28, 28]	64
InvertedResidual-45	[-1, 32, 28, 28]	0
Conv2d-46	[-1, 192, 28, 28]	6,144
BatchNorm2d-47	[-1, 192, 28, 28]	384
ReLU6-48	[-1, 192, 28, 28]	0
Conv2d-49	[-1, 192, 28, 28]	1,728
BatchNorm2d-50	[-1, 192, 28, 28]	384
ReLU6-51	[-1, 192, 28, 28]	0
Conv2d-52	[-1, 32, 28, 28]	6,144
BatchNorm2d-53	[-1, 32, 28, 28]	64
InvertedResidual-54	[-1, 32, 28, 28]	0
Conv2d-55	[-1, 192, 28, 28]	6,144

```
     BatchNorm2d-56          [-1, 192, 28, 28]             384
           ReLU6-57          [-1, 192, 28, 28]               0
          Conv2d-58          [-1, 192, 14, 14]           1,728
     BatchNorm2d-59          [-1, 192, 14, 14]             384
           ReLU6-60          [-1, 192, 14, 14]               0
          Conv2d-61           [-1, 64, 14, 14]          12,288
     BatchNorm2d-62           [-1, 64, 14, 14]             128
InvertedResidual-63           [-1, 64, 14, 14]               0
          Conv2d-64          [-1, 384, 14, 14]          24,576
     BatchNorm2d-65          [-1, 384, 14, 14]             768
           ReLU6-66          [-1, 384, 14, 14]               0
          Conv2d-67          [-1, 384, 14, 14]           3,456
     BatchNorm2d-68          [-1, 384, 14, 14]             768
           ReLU6-69          [-1, 384, 14, 14]               0
          Conv2d-70           [-1, 64, 14, 14]          24,576
     BatchNorm2d-71           [-1, 64, 14, 14]             128
InvertedResidual-72           [-1, 64, 14, 14]               0
          Conv2d-73          [-1, 384, 14, 14]          24,576
     BatchNorm2d-74          [-1, 384, 14, 14]             768
           ReLU6-75          [-1, 384, 14, 14]               0
          Conv2d-76          [-1, 384, 14, 14]           3,456
     BatchNorm2d-77          [-1, 384, 14, 14]             768
           ReLU6-78          [-1, 384, 14, 14]               0
          Conv2d-79           [-1, 64, 14, 14]          24,576
     BatchNorm2d-80           [-1, 64, 14, 14]             128
InvertedResidual-81           [-1, 64, 14, 14]               0
          Conv2d-82          [-1, 384, 14, 14]          24,576
     BatchNorm2d-83          [-1, 384, 14, 14]             768
           ReLU6-84          [-1, 384, 14, 14]               0
          Conv2d-85          [-1, 384, 14, 14]           3,456
     BatchNorm2d-86          [-1, 384, 14, 14]             768
           ReLU6-87          [-1, 384, 14, 14]               0
          Conv2d-88           [-1, 64, 14, 14]          24,576
     BatchNorm2d-89           [-1, 64, 14, 14]             128
InvertedResidual-90           [-1, 64, 14, 14]               0
          Conv2d-91          [-1, 384, 14, 14]          24,576
     BatchNorm2d-92          [-1, 384, 14, 14]             768
           ReLU6-93          [-1, 384, 14, 14]               0
          Conv2d-94          [-1, 384, 14, 14]           3,456
     BatchNorm2d-95          [-1, 384, 14, 14]             768
           ReLU6-96          [-1, 384, 14, 14]               0
          Conv2d-97           [-1, 96, 14, 14]          36,864
     BatchNorm2d-98           [-1, 96, 14, 14]             192
InvertedResidual-99           [-1, 96, 14, 14]               0
         Conv2d-100          [-1, 576, 14, 14]          55,296
    BatchNorm2d-101          [-1, 576, 14, 14]           1,152
          ReLU6-102          [-1, 576, 14, 14]               0
         Conv2d-103          [-1, 576, 14, 14]           5,184
```

BatchNorm2d-104	[-1, 576, 14, 14]	1,152
ReLU6-105	[-1, 576, 14, 14]	0
Conv2d-106	[-1, 96, 14, 14]	55,296
BatchNorm2d-107	[-1, 96, 14, 14]	192
InvertedResidual-108	[-1, 96, 14, 14]	0
Conv2d-109	[-1, 576, 14, 14]	55,296
BatchNorm2d-110	[-1, 576, 14, 14]	1,152
ReLU6-111	[-1, 576, 14, 14]	0
Conv2d-112	[-1, 576, 14, 14]	5,184
BatchNorm2d-113	[-1, 576, 14, 14]	1,152
ReLU6-114	[-1, 576, 14, 14]	0
Conv2d-115	[-1, 96, 14, 14]	55,296
BatchNorm2d-116	[-1, 96, 14, 14]	192
InvertedResidual-117	[-1, 96, 14, 14]	0
Conv2d-118	[-1, 576, 14, 14]	55,296
BatchNorm2d-119	[-1, 576, 14, 14]	1,152
ReLU6-120	[-1, 576, 14, 14]	0
Conv2d-121	[-1, 576, 7, 7]	5,184
BatchNorm2d-122	[-1, 576, 7, 7]	1,152
ReLU6-123	[-1, 576, 7, 7]	0
Conv2d-124	[-1, 160, 7, 7]	92,160
BatchNorm2d-125	[-1, 160, 7, 7]	320
InvertedResidual-126	[-1, 160, 7, 7]	0
Conv2d-127	[-1, 960, 7, 7]	153,600
BatchNorm2d-128	[-1, 960, 7, 7]	1,920
ReLU6-129	[-1, 960, 7, 7]	0
Conv2d-130	[-1, 960, 7, 7]	8,640
BatchNorm2d-131	[-1, 960, 7, 7]	1,920
ReLU6-132	[-1, 960, 7, 7]	0
Conv2d-133	[-1, 160, 7, 7]	153,600
BatchNorm2d-134	[-1, 160, 7, 7]	320
InvertedResidual-135	[-1, 160, 7, 7]	0
Conv2d-136	[-1, 960, 7, 7]	153,600
BatchNorm2d-137	[-1, 960, 7, 7]	1,920
ReLU6-138	[-1, 960, 7, 7]	0
Conv2d-139	[-1, 960, 7, 7]	8,640
BatchNorm2d-140	[-1, 960, 7, 7]	1,920
ReLU6-141	[-1, 960, 7, 7]	0
Conv2d-142	[-1, 160, 7, 7]	153,600
BatchNorm2d-143	[-1, 160, 7, 7]	320
InvertedResidual-144	[-1, 160, 7, 7]	0
Conv2d-145	[-1, 960, 7, 7]	153,600
BatchNorm2d-146	[-1, 960, 7, 7]	1,920
ReLU6-147	[-1, 960, 7, 7]	0
Conv2d-148	[-1, 960, 7, 7]	8,640
BatchNorm2d-149	[-1, 960, 7, 7]	1,920
ReLU6-150	[-1, 960, 7, 7]	0
Conv2d-151	[-1, 320, 7, 7]	307,200

```
     BatchNorm2d-152            [-1, 320, 7, 7]             640
InvertedResidual-153            [-1, 320, 7, 7]               0
          Conv2d-154           [-1, 1280, 7, 7]         409,600
     BatchNorm2d-155           [-1, 1280, 7, 7]           2,560
           ReLU6-156           [-1, 1280, 7, 7]               0
================================================================
Total params: 2,223,872
Trainable params: 2,223,872
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 152.85
Params size (MB): 8.48
Estimated Total Size (MB): 161.91
----------------------------------------------------------------
```

7 (b) Activation and Proposal

7.1 Activation Grid Generator

After passing the input image through the backbone network, we have a convolutional feature map of shape $(D, 7, 7)$ which we interpret as a 7x7 grid of $D-$dimensional features. At each point in this grid, we predict a set of A bounding boxes. The total number of bounding boxes we predict for a single image are thus Ax7x7.

Centered at each position of the 7x7 activation grid, we predict A bounding boxes, where $A = 2$ in our case. In order to place bounding boxes centered at each position of the 7x7 grid of backbone features, we need to know the spatial position of the center of each cell in the grid of features.

This function will compute these center coordinates for us.

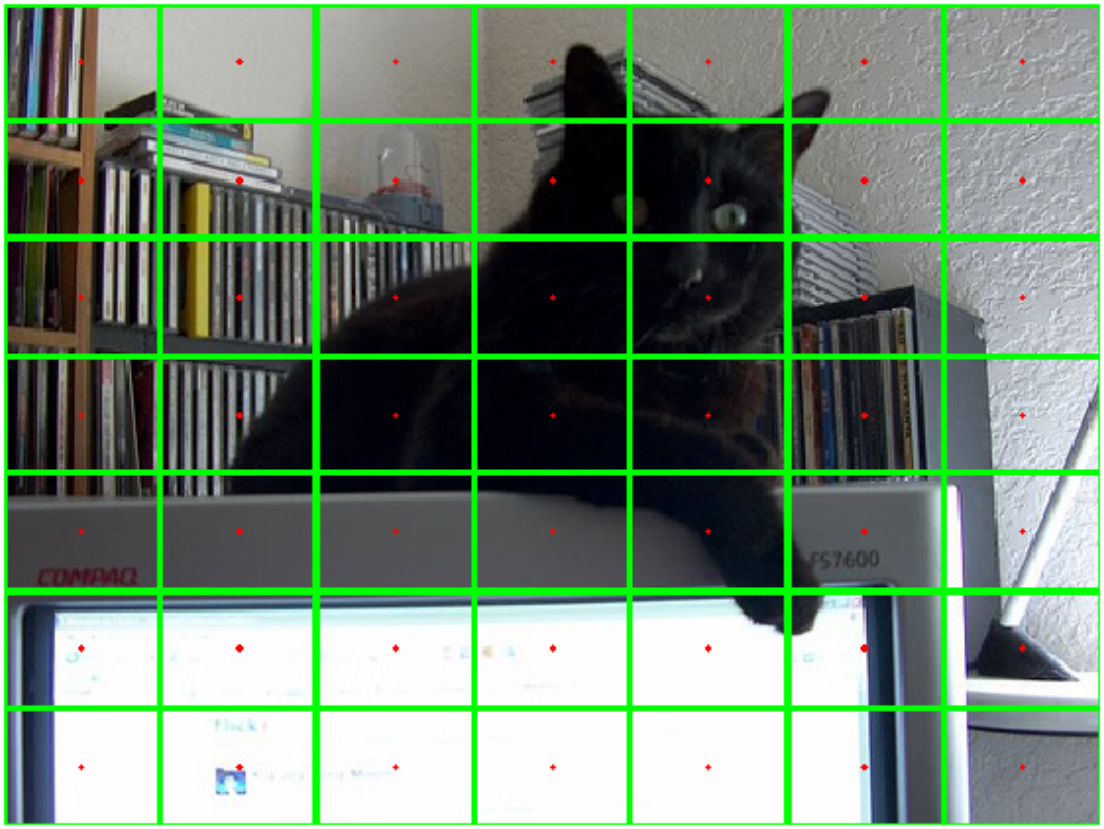
[15]:
```
def GenerateGrid(batch_size, w_amap=7, h_amap=7, dtype=torch.float32,␣
↪device='cuda'):
    """
    Return a grid cell given a batch size (center coordinates).

    Inputs:
    - batch_size, B
    - w_amap: or W', width of the activation map (number of grids in the␣
↪horizontal dimension)
    - h_amap: or H', height of the activation map (number of grids in the␣
↪vertical dimension)
    - W' and H' are always 7 in our case while w and h might vary.

    Outputs:
    grid: A float32 tensor of shape (B, H', W', 2) giving the (x, y) coordinates
          of the centers of each feature for a feature map of shape (B, D, H',␣
↪W')
    """
```

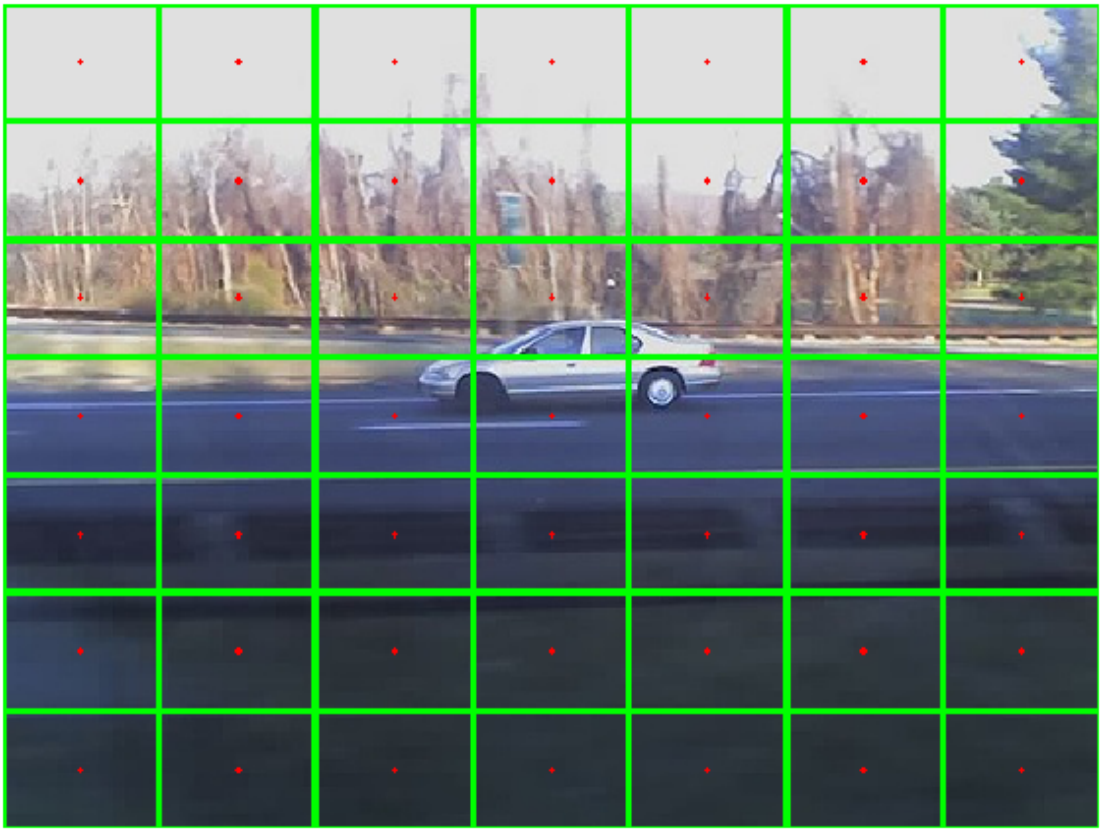
```
  w_range = torch.arange(0, w_amap, dtype=dtype, device=device) + 0.5
  h_range = torch.arange(0, h_amap, dtype=dtype, device=device) + 0.5

  w_grid_idx = w_range.unsqueeze(0).repeat(h_amap, 1)
  h_grid_idx = h_range.unsqueeze(1).repeat(1, w_amap)
  grid = torch.stack([w_grid_idx, h_grid_idx], dim=-1)
  grid = grid.unsqueeze(0).repeat(batch_size, 1, 1, 1)

  return grid
```

Visulaize this grid for a few images and notice the red dot at the center of each grid cell, which is the reference point from where we will be predicting our bouding boxes. (Remember, we will predict $A = 2$ bounding boxes per grid cell).

[16]:
```
# visualization
# simply create an activation grid where the cells are in green and the centers␣
 ↪in red
# you should see the entire image divided by a 7x7 grid, with no gaps on the␣
 ↪edges

grid_list = GenerateGrid(w_list.shape[0])

center = torch.cat((grid_list, grid_list), dim=-1)
grid_cell = center.clone()
grid_cell[:, :, :, [0, 1]] -= 1. / 2.
grid_cell[:, :, :, [2, 3]] += 1. / 2.
center = coord_trans(center, w_list, h_list)
grid_cell = coord_trans(grid_cell, w_list, h_list)

for img, anc, grid in zip(img_list, center, grid_cell):
  data_visualizer(img, idx_to_class, anc.reshape(-1, 4), grid.reshape(-1, 4))
```


Spiritsport.com
PIRELLI
PIRELLI


COMPAQ
FS7600



7.2 (b) TODO: Proposal Generator

Now, consider a grid with center, width and height (x_c^g, y_c^g). The Prediction Network that you will finish implementing later will predict *offsets* (t^x, t^y, t^w, t^h); applying this transformation yields the *bounding box or proposal* with center x, center y, width and height (x_c^p, y_c^p, w^p, h^p). To convert the offsets to the actual bounding box/proposal parameters, use the below mentioned transformation formulas.

Notice here we parametrize the boxes in form `(x, y, w, h)`, where `x, y` represent the center coordinates, and `w, h` represent the width and height respectively. This is in contrast to the `(xtl, ytl, xbr, ybr)` parametrization for the GT bounding boxes. While writing functions and performing operations, be mindful of which of these forms your bounding boxes are of. In this case, even though we apply the transformations to `(x, y, w, h)` parametrized form, we expect the final propsals to be in `(xtl, ytl, xbr, ybr)` form, so you will need to convert from `(x, y, w, h)` to `(xtl, ytl, xbr, ybr)` format before returning proposals. Below are the transformation formulations to help you get bounding boxes or proposals. Here we assume that the shape of the activation map is 7 x 7.

We assume that t^x and t^y are both in the range $-0.5 \leq t^x, t^y \leq 0.5$, while t^w and t^h are real numbers in the range $(0, 1)$. Then we have: - $x_c^p = x_c^g + t^x$ - $y_c^p = y_c^g + t^y$ - $w^p = (t^w) * 7$ -

$h^p = (t^h) * 7$

7.2.1 Training

During training, we compute the ground-truth transformation $(\hat{t^x}, \hat{t^y}, \hat{t^w}, \hat{t^h})$ that would, with the help of the grid center coordinates, yield the bounding box (x_c^p, y_c^p, w^p, h^p) which we expect to match the ground-truth box $(x_c^{gt}, y_c^{gt}, w^{gt}, h^{gt})$. We then apply a regression loss that penalizes differences between the predicted transform (t^x, t^y, t^w, t^h) and the ground-truth transform.

[17]:
```
def GenerateProposal(grids, offsets):
  """
  Proposal generator.

  Inputs:
  - grids: Activation grids, of shape (B, H', W', 2). Grid centers are
  represented by their coordinates in the activation map.
  - offsets: Transformations obtained from the Prediction Network
    of shape (B, A, H', W', 4) that will be used to generate proposals region
    proposals. The transformation offsets[b, a, h, w] = (tx, ty, tw, th) will␣
↪be
    applied to the grids[b, h, w].
    Assume that tx and ty are in the range
    (-0.5, 0.5) and h,w are normalized and thus in the range (0, 1).

  Outputs:
  - proposals: Region proposals of shape (B, A, H', W', 4), represented by the
    coordinates of their top-left and bottom-right corners. Using the
    transform offsets[b, a, h, w] and girds[b, a, h, w] should give the␣
↪proposals.
    The expected parametrization of the proposals is (xtl, ytl, xbr, ybr).

  CAUTION:
    Notice that the offsets here are parametrized as (x, y, w, h).
    The proposals you return need to be of the form (xtl, ytl, xbr, ybr).
  """
  proposals = None

  ##############################################################################
  # TODO: Given grid coordinates and the proposed offset for each bounding     #
  # box, compute the proposal coordinates using the transformation formulas    #
  # above.                                                                     #
  ##############################################################################
  # 1. Follow the formulas above to convert the grid centers into proposals.
  g = torch.unsqueeze(grids, 1).repeat(1, offsets.shape[1], 1, 1, 1).cuda()

  xgc = g[:, :, :, :, 0].cuda()
  ygc = g[:, :, :, :, 1].cuda()
```

```
    tx = offsets[:, :, :, :, 0].cuda()
    ty = offsets[:, :, :, :, 1].cuda()
    tw = offsets[:, :, :, :, 2].cuda()
    th = offsets[:, :, :, :, 3].cuda()

    xpc = xgc + tx
    ypc = ygc + ty
    wp = tw * 7
    hp = th * 7

    # 2. Convert the proposals into (xtl, ytl, xbr, ybr) coordinate format as
    # mentioned in the header and in the cell above that.
    proposals = torch.zeros(grids.shape[0], offsets.shape[1], grids.shape[1],␣
    ↪grids.shape[2], 4, device=torch.device('cuda'))

    proposals[:, :, :, :, 0] = xpc - wp/2   # xtl
    proposals[:, :, :, :, 1] = ypc - hp/2   # ytl
    proposals[:, :, :, :, 2] = xpc + wp/2   # xbr
    proposals[:, :, :, :, 3] = ypc + hp/2   # ybr


    ##############################################################################
    #                               END OF YOUR CODE                             #
    ##############################################################################␣
    ↪

    return proposals
```

8 (c-e) Prediction Networks

8.1 (c) TODO: Intersection Over Union (IoU)

You will implement the IoU function. Carefully read the expected inputs and outputs. This function will also be used later in the ObjectClassification module to calculate the IoU between the predicted bounding boxes/proposals and the ground truth bounding boxes. **NOTE**: Keep in mind the parametrization of the input and output bounding boxes. (Whether it is (x, y, w, h) or (xtl, ytl, xbr, ybr)). The definition of IoU can be found in the lecture slides (slides 46-47).

[18]:
```
def IoU(proposals, bboxes):
    """
    Compute intersection over union between sets of bounding boxes.

    Inputs:
    - proposals: Proposals of shape (B, A, H', W', 4). These should be␣
    ↪parametrized
      as (xtl, ytl, xbr, ybr).
    - bboxes: Ground-truth boxes from the DataLoader of shape (B, N, 5).
```

```
  Each ground-truth box is represented as tuple (x_tl, y_tl, x_br, y_br,␣
↪class).
  If image i has fewer than N boxes, then bboxes[i] will be padded with extra
  rows of -1.

Outputs:
- iou_mat: IoU matrix of shape (B, A*H'*W', N) where iou_mat[b, i, n] gives
  the IoU between one element of proposals[b] and bboxes[b, n].

For this implementation you DO NOT need to filter invalid proposals or boxes;
in particular you don't need any special handling for bboxes that are padded
with -1.
"""
iou_mat = None

##############################################################################
# TODO: Compute the Intersection over Union (IoU) on proposals and GT boxes.#
# No need to filter invalid proposals/bboxes (i.e., allow region area <= 0).#
# However, you need to make sure to compute the IoU correctly (it should be #
# 0 in those cases.                                                          #
# You need to ensure your implementation is efficient (no for loops).       #
# HINT:                                                                      #
# IoU = Area of Intersection / Area of Union, where                          #
# Area of Union = Area of Proposal + Area of BBox - Area of Intersection     #
# and the Area of Intersection can be computed using the top-left corner and#
# bottom-right corner of proposal and bbox. Think about their relationships.#
##############################################################################
B = proposals.shape[0]
AHW = proposals.shape[1] * proposals.shape[2] * proposals.shape[3]
N = bboxes.shape[1]

p = proposals.reshape(B, AHW, 4)
p = torch.unsqueeze(p, 2).repeat(1, 1, N, 1).cuda()

xp_tl = p[:,:,:,0].cuda()
yp_tl = p[:,:,:,1].cuda()
xp_br = p[:,:,:,2].cuda()
yp_br = p[:,:,:,3].cuda()

wp = xp_br - xp_tl
hp = yp_br - yp_tl

AoP = wp * hp

b = torch.unsqueeze(bboxes, 1).repeat(1, AHW, 1, 1).cuda()

xb_tl = b[:, :, :, 0].cuda()
```

```
  yb_tl = b[:, :, :, 1].cuda()
  xb_br = b[:, :, :, 2].cuda()
  yb_br = b[:, :, :, 3].cuda()

  wb = xb_br - xb_tl
  hb = yb_br - yb_tl

  AoB = wb * hb

  wi = xb_br - xp_tl
  hi = yb_br - yp_tl

  a0 = wi * hi
  AoI = a0

  AoU = AoP + AoB - AoI

  iou_mat = AoI / AoU
  iou_mat[iou_mat < 0] = 0
  ##############################################################################
  #                                END OF YOUR CODE                            #
  ##############################################################################

  return iou_mat
```

Run the following to check your implementation. You should see errors on the order of 1e-8 or less. Note that, this check doesn't completely ensure that your implementation is correct.

[19]:
```
# simple sanity check

width, height = torch.tensor([35, 35], **to_float_cuda), torch.tensor([40, 40],␣
 ↪**to_float_cuda)
sample_bbox = torch.tensor([[[1,1,11,11,0], [20,20,30,30,0]]], **to_float_cuda)
sample_proposals = torch.tensor([[[[5,5,15,15], [27,27,37,37]]]],␣
 ↪**to_float_cuda)

result = IoU(sample_proposals, sample_bbox)

# check 1
expected_result = torch.tensor([[[0.21951219, 0.00000000],
                                 [0.00000000, 0.04712042]]], **to_float_cuda)
print('simple iou_mat error: ', rel_error(expected_result, result))
```

```
simple iou_mat error:  0.0
```

After you are done with the problem set, delete the above cell with the image of the Prediction Network.

[20]:
```
class PredictionNetwork(nn.Module):
  def __init__(self, in_dim, hidden_dim=128, num_bboxes=2, num_classes=20,␣
↪drop_ratio=0.3):
    super().__init__()

    assert(num_classes != 0 and num_bboxes != 0)
    self.num_classes = num_classes
    self.num_bboxes = num_bboxes

    # Here we set up a network that will predict outputs for all bounding boxes.
    # This network has a 1x1 convolution layer with `hidden_dim` filters,
    # followed by a Dropout layer with `p=drop_ratio`, a Leaky ReLU
    # nonlinearity, and finally another 1x1 convolution layer to predict all
    # outputs. The network is stored in `self.net`, and has shape
    # (B, 5*A+C, 7, 7), where the 5 predictions are in the order
    # (x, y, w, h, conf_score), with A = `self.num_bboxes`
    # and C = `self.num_classes`.

    self.in_dim = in_dim
    self.hidden_dim = hidden_dim
    self.drop_ratio = drop_ratio
    out_dim = 5*self.num_bboxes + self.num_classes

    layers = [
        torch.nn.Conv2d(self.in_dim, self.hidden_dim, kernel_size=1),
        torch.nn.Dropout(p=self.drop_ratio),
        torch.nn.LeakyReLU(negative_slope=0.2),
        torch.nn.Conv2d(self.hidden_dim,out_dim,kernel_size=1),
    ]
    self.net = nn.Sequential(*layers)

  def forward(self, features):
    """
    Run the forward pass of the network to predict outputs given features
    from the backbone network.

    Inputs:
    - features: Tensor of shape (B, in_dim, 7, 7) giving image features␣
↪computed
      by the backbone network.

    Outputs:
    - bbox_xywh: Tensor of shape (B, A, 4, H, W) giving predicted offsets for
      all bounding boxes.
    - conf_scores: Tensor of shape (B, A, H, W) giving predicted classification
      scores for all bounding boxes.
    - cls_scores: Tensor of shape (B, C, H, W) giving classification scores for
```

```
    each spatial position.
  """
  bbox_xywh, conf_scores, cls_scores = None, None, None

  ##########################################################################
  # TODO: Implement the forward pass of the PredictionNetwork.             #
  # - Use features to predict bbox_xywh (offsets), conf_scores, and        #
  # class_scores.                                                          #
  # - Make sure conf_scores is between 0 and 1 by squashing the            #
  # network output with a sigmoid.                                         #
  # - The first two elements t^x and t^y of offsets should be between -0.5 #
  # and 0.5. You can achieve this by squashing with a sigmoid              #
  # and subtracting 0.5.                                                   #
  # - The last two elements of bbox_xywh w and h should be normalized,     #
  # i.e. squashed with a sigmoid between 0 and 1.                          #
  #                                                                        #
  # Note: In the 5A+C dimension, the first 5*A would be bounding box       #
  # offsets, and next C will be class scores.                              #
  ##########################################################################
  x = self.net(features)
  A = self.num_bboxes
  C = self.num_classes
  B = x.shape[0]
  H = x.shape[2]
  W = x.shape[3]

  offsets = torch.zeros(B, A, 4, H, W, device=torch.device('cuda'))
  conf = torch.zeros(B, A, H, W, device=torch.device('cuda'))
  cls = torch.zeros(B, C, H, W, device=torch.device('cuda'))

  for b in range(B):
    for a in range(A):
      offsets[b, a, :2] = torch.sigmoid(x[b, (a*5):(a*5)+2]).clone() - 0.5
      offsets[b, a, 2:4] = torch.sigmoid(x[b, (a*5)+2:(a*5)+4]).clone()
      conf[b, a] = torch.sigmoid(x[b, (a*5)+4]).clone()
    for c in range(C):
      cls[b, c] = x[b, 5*A+c]


  bbox_xywh = offsets
  conf_scores = conf
  cls_scores = cls
  ##########################################################################
  #                             END OF YOUR CODE                           #
  ##########################################################################
```

```
    # You can uncomment these lines when training for a few iterations to
    # check if your offets are within the expected bounds.

    # print("Checking offset bounds in Prediction Network...")
    # assert bbox_xywh[:, :, 0:2].max() <= 0.5 and bbox_xywh[:, :, 0:2].min()␣
↪>= -0.5, 'invalid offsets (x, y) values'
    # assert bbox_xywh[:, :, 2:4].max() <= 1 and bbox_xywh[:, :, 2:4].min() >=␣
↪0, 'invalid offsets (w, h) values'
    # print("Check passed!")

    return bbox_xywh, conf_scores, cls_scores
```

You can uncomment the assert statements above the return in the previous function when training for a few iterations to see if the offsets you predict are within the required bounds. Once you are sure of that, comment them again and continue training.

8.2 Activated (positive) and negative bounding boxes

During training, after we calculate the bbox_xywh (offset) values for all bounding boxes, we need to match the ground-truth boxes against the predicted bounding boxes to determine the classification labels for the bounding boxes -- which boxes should be classified as containing an object and which should be classified as background? Based on this, we will caluclate the 'expected' or 'Ground Truth' targets for offsets and class, which will be used by you to caluclate the loss. We have written this part for you.

Read and digest the input/output definition carefully. You are highly recommended to skim through the code as well, the ground targets are core to training an accurate model.

[21]:
```
def ReferenceOnActivatedBboxes(bboxes, gt_bboxes, grid, iou_mat, pos_thresh=0.
↪7, neg_thresh=0.3):
    """
    Determine the activated (positive) and negative bboxes for model training.

    A grid cell is responsible for predicting a GT box if the center of
    the box falls into that cell.
    Implementation details: First compute manhattan distance between grid cell␣
↪centers
    (BxHxW) and GT box centers (BxN). This gives us a matrix of shape Bx(H'xW')xN␣
↪and
    perform torch.min(dim=1)[1] on it gives us the indexes indicating activated␣
↪grids
    responsible for GT boxes (convert to x and y). Among all the bboxes␣
↪associated with
    the activate grids, the bbox with the largest IoU with the GT box is␣
↪responsible to
    predict (regress to) the GT box.
    Note: One bbox might match multiple GT boxes.

    Main steps include:
```

```
    i) Decide activated and negative bboxes based on the IoU matrix.
    ii) Compute GT confidence score/offsets/object class on the positive␣
↪proposals.
    iii) Compute GT confidence score on the negative proposals.

    Inputs:
    - bboxes: Bounding boxes, of shape BxAxHxWx4
    - gt_bboxes: GT boxes of shape BxNx5, where N is the number of PADDED GT␣
↪boxes,
              5 indicates (x_{lr}^{gt}, y_{lr}^{gt}, x_{rb}^{gt}, y_{rb}^{gt})␣
↪and class index
    - grid (float): A cell grid of shape BxH'xW'x2 where 2 indicate the (x, y)␣
↪coord
    - iou_mat: IoU matrix of shape Bx(AxHxW)xN
    - pos_thresh: Positive threshold value
    - neg_thresh: Negative threshold value

    Outputs:
    - activated_anc_ind: Index on activated bboxes, of shape M, where M indicates␣
↪the
                         number of activated bboxes
    - negative_anc_ind: Index on negative bboxes, of shape M
    - GT_conf_scores: GT IoU confidence scores on activated bboxes, of shape M
    - GT_offsets: GT offsets on activated bboxes, of shape Mx4. They are denoted␣
↪as
                  \hat{t^x}, \hat{t^y}, \hat{t^w}, \hat{t^h} in the formulation␣
↪earlier.
    - GT_class: GT class category on activated bboxes, essentially indexed from␣
↪gt_bboxes[:, :, 4],
                of shape M
    - activated_anc_coord: Coordinates on activated bboxes (mainly for␣
↪visualization purposes)
    - negative_anc_coord: Coordinates on negative bboxes (mainly for␣
↪visualization purposes)
    """


    B, A, h_amap, w_amap, _ = bboxes.shape
    N = gt_bboxes.shape[1]

    # activated/positive bboxes
    max_iou_per_anc, max_iou_per_anc_ind = iou_mat.max(dim=-1)

    bbox_mask = (gt_bboxes[:, :, 0] != -1) # BxN, indicate invalid boxes
    bbox_centers = (gt_bboxes[:, :, 2:4] - gt_bboxes[:, :, :2]) / 2. + gt_bboxes[:
↪, :, :2] # BxNx2
```

```
 mah_dist = torch.abs(grid.view(B, -1, 2).unsqueeze(2) - bbox_centers.
↪unsqueeze(1)).sum(dim=-1) # Bx(H'xW')xN
 min_mah_dist = mah_dist.min(dim=1, keepdim=True)[0] # Bx1xN
 grid_mask = (mah_dist == min_mah_dist).unsqueeze(1) # Bx1x(H'xW')xN

 reshaped_iou_mat = iou_mat.view(B, A, -1, N)
 anc_with_largest_iou = reshaped_iou_mat.max(dim=1, keepdim=True)[0] #␣
↪Bx1x(HxW)xN
 anc_mask = (anc_with_largest_iou == reshaped_iou_mat) # BxAx(HxW)xN
 activated_anc_mask = (grid_mask & anc_mask).view(B, -1, N)
 activated_anc_mask &= bbox_mask.unsqueeze(1)

 # one bbox could match multiple GT boxes
 activated_anc_ind = torch.nonzero(activated_anc_mask.view(-1)).squeeze(-1)
 GT_conf_scores = iou_mat.view(-1)[activated_anc_ind]
 gt_bboxes = gt_bboxes.view(B, 1, N, 5).repeat(1, A*h_amap*w_amap, 1, 1).
↪view(-1, 5)[activated_anc_ind]
 GT_class = gt_bboxes[:, 4].long()
 gt_bboxes = gt_bboxes[:, :4]
 activated_anc_ind = (activated_anc_ind.float() / activated_anc_mask.
↪shape[-1]).long()

 print('number of pos proposals: ', activated_anc_ind.shape[0])

 activated_anc_coord = bboxes.reshape(-1, 4)[activated_anc_ind]

 activated_grid_coord = grid.repeat(1,A,1,1,1).reshape(-1,␣
↪2)[activated_anc_ind]


 # GT offsets

 # bbox are x_tl, y_tl, x_br, y_br
 # offsets are t_x, t_y, t_w, t_h

 # Grid: (B, H, W, 2) -> This will be used to calculate center offsets
 # w, h offsets are not offsets but normalized w,h themselves.

 wh_offsets = torch.sqrt((gt_bboxes[:, 2:4] - gt_bboxes[:, :2])/7.)
 assert torch.max((gt_bboxes[:, 2:4] - gt_bboxes[:, :2])/7.) <= 1, "w and h␣
↪targets not normalised, should be between 0 and 1"

 xy_offsets = (gt_bboxes[:, :2] + gt_bboxes[:, 2:4])/(2.) -␣
↪activated_grid_coord
```

```
    assert torch.max(torch.abs(xy_offsets)) <= 0.5, \
      "x and y offsets should be between -0.5 and 0.5! Got {}".format( \
      torch.max(torch.abs(xy_offsets)))

    GT_offsets = torch.cat((xy_offsets, wh_offsets), dim=-1)

    # negative bboxes
    negative_anc_mask = (max_iou_per_anc < neg_thresh) # Bx(AxHxW)
    negative_anc_ind = torch.nonzero(negative_anc_mask.view(-1)).squeeze(-1)
    negative_anc_ind = negative_anc_ind[torch.randint(0, negative_anc_ind.
↪shape[0], (activated_anc_ind.shape[0],))]
    negative_anc_coord = bboxes.reshape(-1, 4)[negative_anc_ind.view(-1)]

    # activated_anc_coord and negative_anc_coord are mainly for visualization␣
↪purposes
    return activated_anc_ind, negative_anc_ind, GT_conf_scores, GT_offsets,␣
↪GT_class, \
           activated_anc_coord, negative_anc_coord
```

8.3 (e) Loss Function

The confidence score regression loss is for both activated/negative bounding boxes / proposals while the bounding box regression loss and the object classification loss are for activated bounding boxes only. These are implemented for you. Please go through and understand their inputs and outputs carefully as they are key to the implementation of the forward pass of the object detection module.

8.3.1 Confidence score regression

[22]:
```
def ConfScoreRegression(conf_scores, GT_conf_scores):
    """
    Use sum-squared error as in YOLO

    Inputs:
    - conf_scores: Predicted confidence scores
    - GT_conf_scores: GT confidence scores

    Outputs:
    - conf_score_loss
    """
    # the target conf_scores for negative samples are zeros
    GT_conf_scores = torch.cat((torch.ones_like(GT_conf_scores), \
                                torch.zeros_like(GT_conf_scores)), dim=0).
↪view(-1, 1)
```

```
    conf_score_loss = torch.sum((conf_scores - GT_conf_scores)**2) * 1. /␣
↪GT_conf_scores.shape[0]
    return conf_score_loss
```

8.3.2 Bounding box regression

[23]:
```
def BboxRegression(offsets, GT_offsets):
    """
    Use sum-squared error as in YOLO
    For both xy and wh.
    NOTE: In YOLOv1, the authors use sqrt(w) and sqrt(h) for normalized w and h
    (read paper for more details) and thus both offsets and GT_offsets will
    be having (x, y, sqrt(w), sqrt(h)) parametrization of the coodinates.


    Inputs:
    - offsets: Predicted box offsets
    - GT_offsets: GT box offsets

    Outputs:
    - bbox_reg_loss
    """

    bbox_reg_loss = torch.sum((offsets - GT_offsets)**2) * 1. / GT_offsets.
↪shape[0]
    return bbox_reg_loss
```

8.3.3 Object classification

[24]:
```
def ObjectClassification(class_prob, GT_class, batch_size, anc_per_img,␣
↪activated_anc_ind):
    """
    Use softmax loss

    Inputs:
    - class_prob: Predicted class logits
    - GT_class: GT box class label
    - batch_size: the batch size to compute loss over
    - anc_per_img: anchor indices for each image
    - activated_anc_ind: indices for positive anchors

    Outputs:
    - object_cls_loss, the classification loss for object detection
    """
    # average within sample and then average across batch
```

```
 # such that the class pred would not bias towards dense popular objects like␣
↪`person`

 all_loss = F.cross_entropy(class_prob, GT_class, reduction='none') # ,␣
↪reduction='sum') * 1. / batch_size
 object_cls_loss = 0
 for idx in range(batch_size):
   anc_ind_in_img = (activated_anc_ind >= idx * anc_per_img) &␣
↪(activated_anc_ind < (idx+1) * anc_per_img)
   object_cls_loss += all_loss[anc_ind_in_img].sum() * 1. / torch.
↪sum(anc_ind_in_img)
 object_cls_loss /= batch_size
 # object_cls_loss = F.cross_entropy(class_prob, GT_class, reduction='sum') *␣
↪1. / batch_size

 return object_cls_loss
```

9 (f) Object Detector Code

9.1 (f) TODO: Object detection module

We will now combine everything into the `SingleStageDetector` class:

We have implemented the `forward` function of the detector for you. This implements the training-time forward pass: it receives the input images and the ground-truth bounding boxes, and returns the total loss for the minibatch.

Below are the key steps in implementing the forward pass:

```
i)   Image feature extraction using Detector Backbone Network
ii)  Grid List generation using Grid Generator
iii) Compute offsets, conf_scores, cls_scores through the Prediction Network.
iv)  Calculate the proposals or actual bounding boxes using offsets stored in
     offsets and grid_list by passing these into the GenerateProposal function
     you wrote earlier.
v)   Compute IoU between grid_list and proposals and then determine activated
     negative bounding boxes/proposals, and GT_conf_scores, GT_offsets GT_class
     using the ReferenceOnActivatedBboxes function.
vi)  The loss function for BboxRegression which expects the parametrization as
     (x, y, sqrt(w), sqrt(h)) for the offsets and the GT_offsets also have
     sqrt(w), sqrt(h) in the offsets as part of GT_offsets. So before the next step,
     convert the bbox_xywh parametrization form (x, y, w, h) to (x, y, sqrt(w), sqrt(h)).
vii) Extract the confidence scores corresponding to the positive and negative
     activated bounding box indices, classification scores for positive box indices,
     offsets using positive box indices.  HINT: Set `neg_thresh=0.2` in ReferenceOnActivatedBb
     HINT: You can use the provided helper methods self._extract_bbox_data and
     self._extract_class_scores to extract information for positive and
     negative bounding boxes / proposals specified by activated_anc_ind and negative_anc_ind.
viii) Compute the total_loss which is formulated as:
```

```
    total_loss = w_conf * conf_loss + w_reg * reg_loss + w_cls * cls_loss,
    where conf_loss is determined by ConfScoreRegression, w_reg by BboxRegression,
    and w_cls by ObjectClassification.
```

[25]:
```python
class SingleStageDetector(nn.Module):
  def __init__(self):
    super().__init__()

    self.feat_extractor = FeatureExtractor()
    self.num_classes = 20
    self.num_bboxes = 2
    self.pred_network = PredictionNetwork(1280, num_bboxes=2, \
                                          num_classes=self.num_classes)


  def forward(self, images, bboxes):
    """
    Training-time forward pass for the single-stage detector.

    Inputs:
    - images: Input images, of shape (B, 3, 224, 224)
    - bboxes: GT bounding boxes of shape (B, N, 5) (padded)

    Outputs:
    - total_loss: Torch scalar giving the total loss for the batch.
    """
    # weights to multiple to each loss term
    w_conf = 1 # for conf_scores
    w_reg = 1 # for offsets
    w_cls = 1 # for class_prob

    total_loss = None

    # 1. Feature extraction
    features = self.feat_extractor(images)

    # 2. Grid generator
    grid_list = GenerateGrid(images.shape[0])

    # 3. Prediction Network
    bbox_xywh, conf_scores, cls_scores = self.pred_network(features)
    # (B, A, 4, H, W), (B, A, H, W), (B, C, H, W)

    B, A, _, H, W = bbox_xywh.shape
    bbox_xywh = bbox_xywh.permute(0, 1, 3, 4, 2) # (B, A, H, W, 4)
```

```
    assert bbox_xywh.max() <= 1 and bbox_xywh.min() >= -0.5, 'invalid offsets␣
↪values'

    # 4. Calculate the proposals
    proposals = GenerateProposal(grid_list, bbox_xywh)

    # 5. Compute IoU
    iou_mat = IoU(proposals, bboxes)

    # 7. Using the activated_anc_ind, select the activated conf_scores,␣
↪bbox_xywh, cls_scores
    activated_anc_ind, negative_anc_ind, GT_conf_scores, GT_offsets, GT_class,␣
↪_, _ \
      = ReferenceOnActivatedBboxes(bbox_xywh, bboxes, grid_list, iou_mat,␣
↪neg_thresh=0.3)

    conf_scores = conf_scores.view(B, A, 1, H, W)
    pos = self._extract_bbox_data(conf_scores, activated_anc_ind)
    neg = self._extract_bbox_data(conf_scores, negative_anc_ind)
    conf_scores = torch.cat([pos, neg], dim = 0)

    # 6. The loss function
    bbox_xywh[:, :, :, :, 2:4] = torch.sqrt(bbox_xywh[:, :, :, :, 2:4])

    assert bbox_xywh[:, :, :, :, :2].max() <= 0.5 and bbox_xywh[:, :, :, :, :2].
↪min() >= -0.5, 'invalid offsets values'
    assert bbox_xywh[:, :, :, :, :2:4].max() <= 1 and bbox_xywh[:, :, :, :, 2:
↪4].min() >= 0, 'invalid offsets values'

    offsets = self._extract_bbox_data(bbox_xywh.permute(0, 1, 4, 2, 3),␣
↪activated_anc_ind)
    cls_scores = self._extract_class_scores(cls_scores, activated_anc_ind)
    anc_per_img = torch.prod(torch.tensor(bbox_xywh.shape[1:-1])) # use as␣
↪argument in ObjectClassification
    ##########################################################################
    # TODO: Compute conf_loss, reg_loss, cls_loss, total_loss using the       #
    # functions defined in part (e).                                          ␣
↪#
    # - total_loss is the sum of the three other losses.                      #␣
↪
    ##########################################################################
    # 8. Compute losses
    conf_loss = ConfScoreRegression(conf_scores, GT_conf_scores)
    reg_loss = BboxRegression(offsets, GT_offsets)
    cls_loss = ObjectClassification(cls_scores, GT_class, B, anc_per_img,␣
↪activated_anc_ind)
```

```
    total_loss = w_conf * conf_loss + w_reg * reg_loss + w_cls * cls_loss
    ##############################################################################
    #                               END OF YOUR CODE                             #
    ##############################################################################

    print('(weighted) conf loss: {:.4f}, reg loss: {:.4f}, cls loss: {:.4f}'.
↪format(conf_loss, reg_loss, cls_loss))

    return total_loss

  def inference(self):
    raise NotImplementedError

  def _extract_bbox_data(self, bbox_data, bbox_idx):
    """
    Inputs:
    - bbox_data: Tensor of shape (B, A, D, H, W) giving a vector of length
      D for each of A bboxes at each point in an H x W grid.
    - bbox_idx: int64 Tensor of shape (M,) giving bbox indices to extract

    Returns:
    - extracted_bboxes: Tensor of shape (M, D) giving bbox data for each
      of the bboxes specified by bbox_idx.
    """
    B, A, D, H, W = bbox_data.shape
    bbox_data = bbox_data.permute(0, 1, 3, 4, 2).contiguous().view(-1, D)
    extracted_bboxes = bbox_data[bbox_idx]
    return extracted_bboxes

  def _extract_class_scores(self, all_scores, bbox_idx):
    """
    Inputs:
    - all_scores: Tensor of shape (B, C, H, W) giving classification scores for
      C classes at each point in an H x W grid.
    - bbox_idx: int64 Tensor of shape (M,) giving the indices of bboxes at
      which to extract classification scores

    Returns:
    - extracted_scores: Tensor of shape (M, C) giving the classification scores
      for each of the bboxes specified by bbox_idx.
    """
    B, C, H, W = all_scores.shape
    A = self.num_bboxes
    all_scores = all_scores.contiguous().permute(0, 2, 3, 1).contiguous()
    all_scores = all_scores.view(B, 1, H, W, C).expand(B, A, H, W, C)
    all_scores = all_scores.reshape(B * A * H * W, C)
    extracted_scores = all_scores[bbox_idx]
```

```
    return extracted_scores
```

9.2 Object detection solver

The DetectionSolver object runs the training loop to train an single stage detector.

[26]:
```
def DetectionSolver(detector, train_loader, learning_rate=3e-3,
                    lr_decay=1, num_epochs=20, **kwargs):
  """
  Run optimization to train the model.
  """

  # ship model to GPU
  detector.to(**to_float_cuda)

  # optimizer setup
  from torch import optim
  # optimizer = optim.Adam(
  optimizer = optim.SGD(
    filter(lambda p: p.requires_grad, detector.parameters()),
    learning_rate) # leave betas and eps by default
  lr_scheduler = optim.lr_scheduler.LambdaLR(optimizer,
                                             lambda epoch: lr_decay ** epoch)

  # sample minibatch data
  loss_history = []
  detector.train()
  for i in range(num_epochs):
    start_t = time.time()
    for iter_num, data_batch in enumerate(train_loader):
      images, boxes, w_batch, h_batch, _ = data_batch
      resized_boxes = coord_trans(boxes, w_batch, h_batch, mode='p2a')
      # print(resized_boxes)
      images = images.to(**to_float_cuda)
      resized_boxes = resized_boxes.to(**to_float_cuda)

      loss = detector(images, resized_boxes)
      optimizer.zero_grad()
      loss.backward()
      loss_history.append(loss.item())
      optimizer.step()

      print('(Iter {} / {})'.format(iter_num, len(train_loader)))

    end_t = time.time()
    print('(Epoch {} / {}) loss: {:.4f} time per epoch: {:.1f}s'.format(
        i, num_epochs, loss.item(), end_t-start_t))
```

```
    lr_scheduler.step()

  # plot the training losses
  plt.plot(loss_history)
  plt.xlabel('Iteration')
  plt.ylabel('Loss')
  plt.title('Training loss history')
  plt.show()

  return loss_history
```

10 (g) Train the Object Detector

10.1 Overfit small data

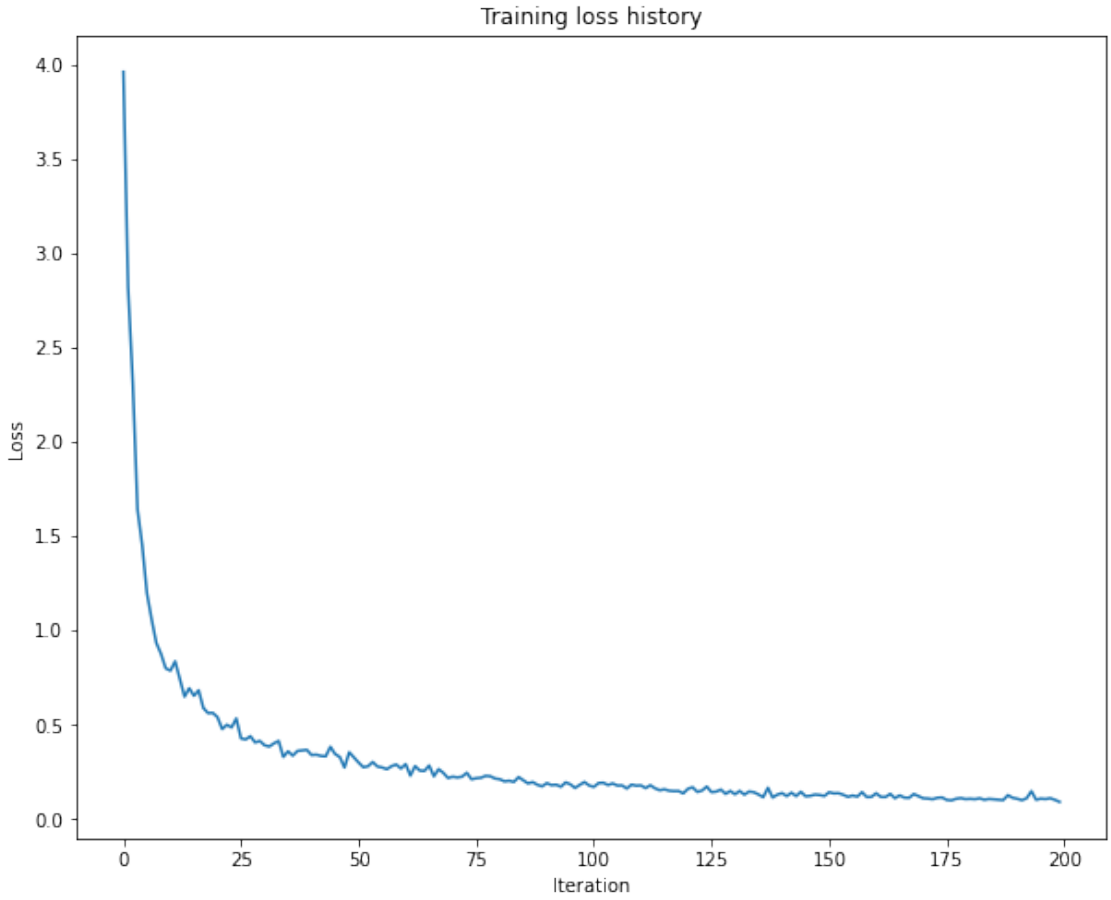
To make sure that everything is working as expected, we can try to overfit the detector to a small subset of data.

After 200 epochs of training you should see a total loss of around or less than 0.14.

After you are done with the problem set, delete the above cell since it will take up a lot of pages during the PDF conversion. To see your final loss and the loss curve, we add an additional cell at the bottom which you will keep instead of the cell above.

[28]:
```
print('Final loss obtained after overfitting is: %.4f'␣
 ↪%(loss_history_overfit[-1]))
plt.plot(loss_history_overfit)
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.title('Training loss history')
plt.show()
```

```
Final loss obtained after overfitting is: 0.0902
```



10.2 Train the neural net

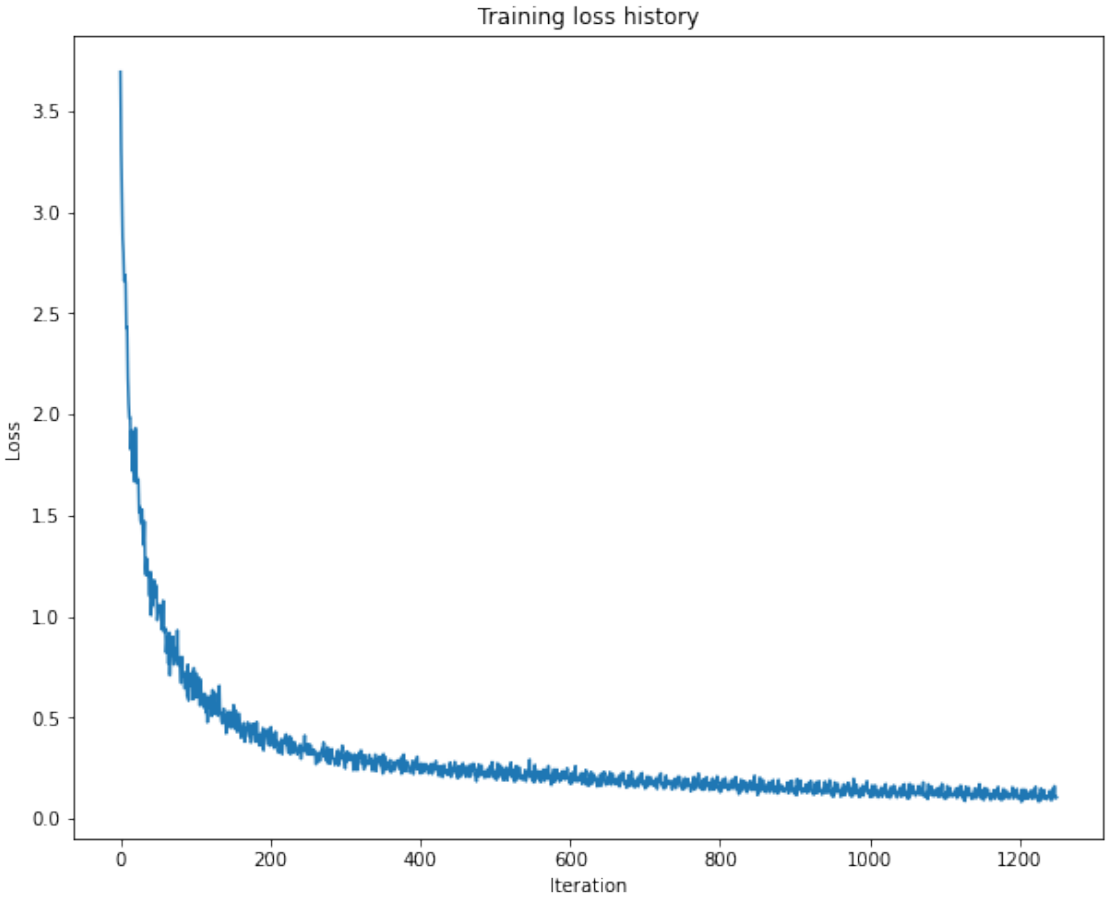
Now that we are confident that the training code is working properly, let's train the network on more data and for longer. We will train for 50 epochs; this should take about 35 minutes on a K80 GPU. You should see a total loss around or less than 0.15.

Note that real object detection systems typically train for 12-24 hours, distribute training over multiple GPUs, and use much faster GPUs. As such our result will be far from the state of the art, but it should give some reasonable results!

After you are done with the problem set, delete the above cell since it will take up a lot of pages during the PDF conversion. To see your final loss and the loss curve, we add an additional cell at the bottom which you will keep instead of the cell above.

[30]:
```
print('Final loss obtained after training is: %.4f' %(loss_history_train[-1]))
plt.plot(loss_history_train)
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.title('Training loss history')
plt.show()
```

```
Final loss obtained after training is: 0.1046
```



Run the cell below to save the checkpoint of the trained model and save it to your local system. You can then upload the model and load checkpoint later to colab incase you don't want to run training again.

[31]:
```
# (optional) load/save checkpoint
# torch.save(yolo_detector.state_dict(), 'yolo_detector.pt') # uncomment to␣
 ↪save your checkpoint
yolo_detector.load_state_dict(torch.load('yolo_detector.pt')) # uncomment to␣
 ↪load your previous checkpoint
```

11 (h-j) Use an object detector

11.1 (h) TODO: Non-Maximum Suppression (NMS)

The definition of NMS and instructions on how to compute NMS can be found in the lecture slides (p47-48)

[76]:
```
def nms(boxes, scores, iou_threshold=0.5, topk=None):
  """
  Non-maximum suppression removes overlapping bounding boxes.
```

```
    Inputs:
    - boxes: top-left and bottom-right coordinate values of the bounding boxes
      to perform NMS on, of shape Nx4
    - scores: scores for each one of the boxes, of shape N
    - iou_threshold: discards all overlapping boxes with IoU > iou_threshold;␣
    ↪float
    - topk: If this is not None, then return only the topk highest-scoring boxes.
      Otherwise if this is None, then return all boxes that pass NMS.

    Outputs:
    - keep: torch.long tensor with the indices of the elements that have been
      kept by NMS, sorted in decreasing order of scores; of shape␣
    ↪[num_kept_boxes]
    """

    if (not boxes.numel()) or (not scores.numel()):
      return torch.zeros(0, dtype=torch.long)

    keep = None
    ##########################################################################
    # TODO: Implement non-maximum suppression which iterates the following:  #
    #       1. Select the highest-scoring box among the remaining ones,      #
    #          which has not been chosen in this step before                 #
    #       2. Eliminate boxes with IoU > threshold                          #
    #       3. If any boxes remain, GOTO 1                                   #
    #       Your implementation should not depend on a specific device type; #
    #       you can use the device of the input if necessary.                #
    ##########################################################################
    keep = torch.empty(1)
    A = boxes
    S = scores

    while S.nelement() is not 0:
      x_ind = torch.argmax(S)
      x = S[x_ind]
      S = torch.cat((S[:x_ind], S[x_ind+1:]))
      if x > iou_threshold:
        keep = torch.cat((keep, torch.unsqueeze(x_ind, 0)))
      A = torch.cat((A[:x_ind], A[x_ind+1:]))

    if topk is not None:
      keep = torch.topk(keep, topk)
    ##########################################################################
    #                              END OF YOUR CODE                          #
    ##########################################################################
```

```
  return keep
```

We will now compare our implementation of NMS with the implementation in torchvision. Most likely, our implementation will be faster on CPU than on CUDA, and the torchvision implementation will likely be much faster than ours. This is expected, but our implementation should produce the same outputs as the torchvision version.

[77]:
```
fix_random_seed(0)
boxes = (100. * torch.rand(5000, 4)).round()
boxes[:,2] = boxes[:,2] + boxes[:,0] + 1.
boxes[:,3] = boxes[:,3] + boxes[:,1] + 1.
scores = torch.randn(5000)

names = ['your_cpu', 'torchvision_cpu', 'torchvision_cuda']
iou_thresholds = [0.3, 0.5, 0.7]
elapsed = dict(zip(names, [0.]*len(names)))
intersects = dict(zip(names[1:], [0.]*(len(names)-1)))

for iou_threshold in iou_thresholds:
  tic = time.time()
  my_keep = nms(boxes, scores, iou_threshold)
  elapsed['your_cpu'] += time.time() - tic

  tic = time.time()
  tv_keep = torchvision.ops.nms(boxes, scores, iou_threshold)
  elapsed['torchvision_cpu'] += time.time() - tic
  intersect = len(set(tv_keep.tolist()).intersection(my_keep.tolist())) /␣
 ↪len(tv_keep)
  intersects['torchvision_cpu'] += intersect

  tic = time.time()
  tv_cuda_keep = torchvision.ops.nms(boxes.cuda(), scores.cuda(),␣
 ↪iou_threshold).to(my_keep.device)
  torch.cuda.synchronize()
  elapsed['torchvision_cuda'] += time.time() - tic
  intersect = len(set(tv_cuda_keep.tolist()).intersection(my_keep.tolist())) /␣
 ↪len(tv_cuda_keep)
  intersects['torchvision_cuda'] += intersect

for key in intersects:
  intersects[key] /= len(iou_thresholds)

# You should see < 1% difference
print('Testing NMS:')
print('Your        CPU  implementation: %fs' % elapsed['your_cpu'])
print('torchvision CPU  implementation: %fs' % elapsed['torchvision_cpu'])
print('torchvision CUDA implementation: %fs' % elapsed['torchvision_cuda'])
print('Speedup CPU : %fx' % (elapsed['your_cpu'] / elapsed['torchvision_cpu']))
```

```
print('Speedup CUDA: %fx' % (elapsed['your_cpu'] / elapsed['torchvision_cuda']))
print('Difference CPU : ', 1. - intersects['torchvision_cpu']) # in the order␣
↪of 1e-3 or less
print('Difference CUDA: ', 1. - intersects['torchvision_cuda']) # in the order␣
↪of 1e-3 or less
```

```
Testing NMS:
Your         CPU  implementation: 1.183219s
torchvision CPU  implementation: 0.079952s
torchvision CUDA implementation: 0.012105s
Speedup CPU : 14.799204x
Speedup CUDA: 97.742545x
Difference CPU :  0.7297065492349235
Difference CUDA:  0.7299169854270626
```

11.2 (i) Inference

Now, implement the inference part of module `SingleStageDetector`.

[78]:
```
def detector_inference(self, images, thresh=0.5, nms_thresh=0.7):
    """
    Inference-time forward pass for the single stage detector.

    Inputs:
    - images: Input images
    - thresh: Threshold value on confidence scores
    - nms_thresh: Threshold value on NMS

    Outputs:
    - final_propsals: Keeped proposals after confidence score thresholding and␣
↪NMS,
                      a list of B (*x4) tensors
    - final_conf_scores: Corresponding confidence scores, a list of B (*x1)␣
↪tensors
    - final_class: Corresponding class predictions, a list of B  (*x1) tensors
    """
    final_proposals, final_conf_scores, final_class = [], [], []

    # Predicting the final proposal coordinates `final_proposals`,
    # confidence scores `final_conf_scores`, and the class index `final_class`.␣
↪

    # The overall steps are similar to the forward pass but now we do not need
    # to decide the activated nor negative bounding boxes.                    ␣
↪
    # We threshold the conf_scores based on the threshold value `thresh`.
    # Then, apply NMS (torchvision.ops.nms) to the filtered proposals given the␣
↪
```

```
    # threshold `nms_thresh`.
    # The class index is determined by the class with the maximal probability.
    # Note that `final_propsals`, `final_conf_scores`, and `final_class` are all
    # lists of B 2-D tensors.

    with torch.no_grad():
      # Feature extraction
      features = self.feat_extractor(images)

      # Grid  Generator
      grid_list = GenerateGrid(images.shape[0])

      # Prediction Network
      offsets, conf_scores, class_scores = self.pred_network(features)
      B, A, _, w_amap, h_amap = offsets.shape # B, A, 4, H, W
      C = self.num_classes
      conf_scores = conf_scores.view(B, -1) # B, A*H*W
      offsets = offsets.permute(0, 1, 3, 4, 2) # B, A, H, W, 4
      class_scores = class_scores.permute(0, 2, 3, 1).reshape(B, -1, C) # B, H*W, C

      most_conf_class_score, most_conf_class_idx = class_scores.max(dim=-1)

      # Proposal generator
      proposals = GenerateProposal(grid_list, offsets).view(B, -1, 4) # Bx(AxH'xW')x4

      # Thresholding and NMS
      for i in range(B):
        score_mask = torch.nonzero((conf_scores[i] > thresh)).squeeze(1) # (AxH'xW')
        prop_before_nms = proposals[i, score_mask]
        scores_before_nms = conf_scores[i, score_mask]
        class_idx_before_nms = most_conf_class_idx[i, score_mask%(h_amap*w_amap)]
        # class_prob_before_nms = most_conf_class_prob[i, score_mask/A]

        prop_keep = torchvision.ops.nms(prop_before_nms, scores_before_nms, nms_thresh).to(images.device)
        final_proposals.append(prop_before_nms[prop_keep])
        final_conf_scores.append(scores_before_nms[prop_keep].unsqueeze(-1))
        final_class.append(class_idx_before_nms[prop_keep].unsqueeze(-1))
```

```
    return final_proposals, final_conf_scores, final_class
  SingleStageDetector.inference = detector_inference
```

[80]:
```
def DetectionInference(detector, data_loader, dataset, idx_to_class, thresh=0.
↪8, nms_thresh=0.3, output_dir=None):

  # ship model to GPU
  detector.to(**to_float_cuda)

  detector.eval()
  start_t = time.time()

  if output_dir is not None:
    det_dir = 'mAP/input/detection-results'
    gt_dir = 'mAP/input/ground-truth'
    if os.path.exists(det_dir):
      shutil.rmtree(det_dir)
    os.mkdir(det_dir)
    if os.path.exists(gt_dir):
      shutil.rmtree(gt_dir)
    os.mkdir(gt_dir)

  for iter_num, data_batch in enumerate(data_loader):
    images, boxes, w_batch, h_batch, img_ids = data_batch
    images = images.to(**to_float_cuda)

    final_proposals, final_conf_scores, final_class = detector.
↪inference(images, thresh=thresh, nms_thresh=nms_thresh)

    # clamp on the proposal coordinates
    batch_size = len(images)
    for idx in range(batch_size):
      torch.clamp_(final_proposals[idx][:, 0::2], min=0, max=w_batch[idx])
      torch.clamp_(final_proposals[idx][:, 1::2], min=0, max=h_batch[idx])

      # visualization
      # get the original image
      # hack to get the original image so we don't have to load from local␣
↪again...
      i = batch_size*iter_num + idx
      img, _ = dataset.__getitem__(i)

      valid_box = sum([1 if j != -1 else 0 for j in boxes[idx][:, 0]])
      final_all = torch.cat((final_proposals[idx], \
        final_class[idx].float(), final_conf_scores[idx]), dim=-1).cpu()
      resized_proposals = coord_trans(final_all, w_batch[idx], h_batch[idx])
```

```
    # write results to file for evaluation (use mAP API https://github.com/
↪Cartucho/mAP for now...)
    if output_dir is not None:
      file_name = img_ids[idx].replace('.jpg', '.txt')
      with open(os.path.join(det_dir, file_name), 'w') as f_det, \
        open(os.path.join(gt_dir, file_name), 'w') as f_gt:
        print('{}: {} GT bboxes and {} proposals'.format(img_ids[idx],␣
↪valid_box, resized_proposals.shape[0]))
        for b in boxes[idx][:valid_box]:
          f_gt.write('{} {:.2f} {:.2f} {:.2f} {:.2f}\n'.
↪format(idx_to_class[b[4].item()], b[0], b[1], b[2], b[3]))
        for b in resized_proposals:
          f_det.write('{} {:.6f} {:.2f} {:.2f} {:.2f} {:.2f}\n'.
↪format(idx_to_class[b[4].item()], b[5], b[0], b[1], b[2], b[3]))
    else:
      data_visualizer(img, idx_to_class, boxes[idx][:valid_box],␣
↪resized_proposals)

 end_t = time.time()
 print('Total inference time: {:.1f}s'.format(end_t-start_t))
```

11.2.1 Inference - overfit small data

[81]:
```
# visualize the output from the overfitted model on small dataset
# the bounding boxes should be really accurate
DetectionInference(detector, small_train_loader, small_dataset, idx_to_class,␣
 ↪thresh=0.8)
```

```
Output hidden; open in https://colab.research.google.com to view.
```

[82]:
```
# visualize the same output from the model trained on the entire training set
# some bounding boxes might not make sense
DetectionInference(yolo_detector, small_train_loader, small_dataset,␣
 ↪idx_to_class)
```

```
Output hidden; open in https://colab.research.google.com to view.
```

11.3 (j) Evaluation

Compute the mean Average Precision (mAP). For a brief introduction to mAP see lecture slides (slides 48-50) or this repository.

Run the following to evaluate your detector on the PASCAL VOC validation set. You should see mAP at around 12% or above.

The state of the art on this dataset is >80% mAP! To achieve these results we would need to use a much bigger network, and train with more data and for much longer, but that is beyond the scope of this problme set.

(Optional) If you train the model longer (e.g., 100 epochs), you should see a better mAP. But make sure you revert the code back for grading purposes.

Before printing the PDF, delete the cell above. The image below (in the Convert Notebook to PDF section) will indicate the mAP as well as the class-wise APs obtained.

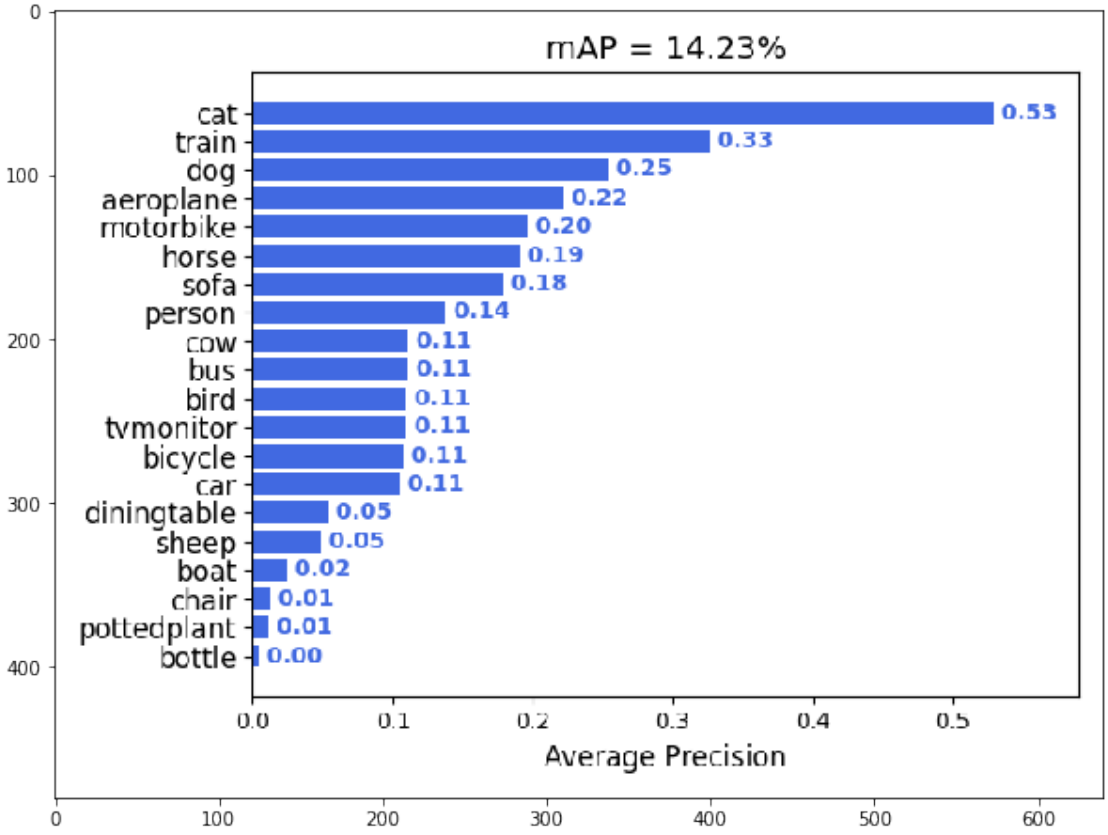
12 Convert Notebook to PDF

Read the instructions here carefully because you will need to delete certain cells.

[84]:
```
from PIL import Image
img = Image.open('mAP/output/mAP.png')
print('Final mAP obtained on the validation set')
plt.imshow(img)
```

```
Final mAP obtained on the validation set
```

[84]: <matplotlib.image.AxesImage at 0x7f4cafeb1110>



Use this cell to convert your notebook to PDF. Before doing that though, delete the text cell with the Prediction Network photo, and all the training cells showing outputs at every iteration (we have indicated these below those cells) and the mAP cell which has long outputs as well. So, you have to remove 4 cells in total. If you don't do this correctly, your PDF will not contain all your work.

[ ]:
```
# generate pdf
# %%capture
!git clone https://gist.github.com/bc5f1add34fef7c7f9fb83d3783311e2.git
!cp bc5f1add34fef7c7f9fb83d3783311e2/colab_pdf.py colab_pdf.py
from colab_pdf import colab_pdf
# change the name to your ipynb file name shown on the top left of Colab window
# Important: make sure that your file name does not contain spaces!
colab_pdf('cktran_09859713.ipynb')
```

```
fatal: destination path 'bc5f1add34fef7c7f9fb83d3783311e2' already exists and is
not an empty directory.
Get:1 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease
[3,626 B]
Get:2 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ Packages [73.0
kB]
Ign:3 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
InRelease
Ign:6 https://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1804/x86_64  InRelease
Get:7 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
Release [696 B]
Hit:8 https://developer.download.nvidia.com/compute/machine-
learning/repos/ubuntu1804/x86_64  Release
Get:9 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
Release.gpg [836 B]
Get:5 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:11 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:12
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64
Packages [808 kB]
Hit:13 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Hit:16 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:17 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64
Packages [26.8 kB]
Get:18 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages
[1,439 kB]
Get:19 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages
[2,426 kB]
Hit:20 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Get:15 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages
```

```
[2,867 kB]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages
[34.5 kB]
Get:4 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease
[15.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages
[2,218 kB]
Get:24 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages
[699 kB]
Get:25 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64
Packages [665 kB]
Get:26 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main Sources
[1,812 kB]
Get:27 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main amd64
Packages [928 kB]
Fetched 4,593 kB in 6s (723 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
57 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-latex-base texlive-latex-extra
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
```

```
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-generic-recommended
  texlive-latex-base texlive-latex-extra texlive-latex-recommended
  texlive-pictures texlive-plain-generic texlive-xetex tipa
0 upgraded, 47 newly installed, 0 to remove and 57 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.8 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.14 [5,092 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.14 [2,265 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
```

```
2.5.1-1ubuntu1.10 [48.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-did-you-mean all
1.2.0-2 [9,700 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-minitest all
5.10.3-1 [38.6 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all
0.3.0-1 [7,952 B]
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-test-unit all
3.2.5-1 [61.1 kB]
Get:28 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.10 [3,071 kB]
Get:29 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsynctex1
amd64 2017.20170613.44572-8ubuntu0.1 [41.4 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexlua52
amd64 2017.20170613.44572-8ubuntu0.1 [91.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexluajit2
amd64 2017.20170613.44572-8ubuntu0.1 [230 kB]
Get:32 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libzzip-0-13
amd64 0.13.62-3.1ubuntu0.18.04.1 [26.0 kB]
Get:33 http://archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3
[9,631 kB]
Get:34 http://archive.ubuntu.com/ubuntu bionic/main amd64 preview-latex-style
all 11.91-1ubuntu1 [185 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic/main amd64 t1utils amd64 1.41-2
[56.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all
20160520-1 [4,998 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 texlive-
binaries amd64 2017.20170613.44572-8ubuntu0.1 [8,179 kB]
Get:38 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-base all
2017.20180305-1 [18.7 MB]
Get:39 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-fonts-
recommended all 2017.20180305-1 [5,262 kB]
Get:40 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-plain-
generic all 2017.20180305-2 [23.6 MB]
Get:41 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-generic-
recommended all 2017.20180305-1 [15.9 kB]
Get:42 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-base all
2017.20180305-1 [951 kB]
Get:43 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-
recommended all 2017.20180305-1 [14.9 MB]
Get:44 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-pictures
```

```
all 2017.20180305-1 [4,026 kB]
Get:45 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-latex-
extra all 2017.20180305-2 [10.6 MB]
Get:46 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:47 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-xetex all
2017.20180305-1 [10.7 MB]
Fetched 146 MB in 14s (10.1 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages ...
Selecting previously unselected package fonts-droid-fallback.
(Reading database ... 155219 files and directories currently installed.)
Preparing to unpack .../00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb ...
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) ...
Selecting previously unselected package fonts-lato.
Preparing to unpack .../01-fonts-lato_2.0-2_all.deb ...
Unpacking fonts-lato (2.0-2) ...
Selecting previously unselected package poppler-data.
Preparing to unpack .../02-poppler-data_0.4.8-2_all.deb ...
Unpacking poppler-data (0.4.8-2) ...
Selecting previously unselected package tex-common.
Preparing to unpack .../03-tex-common_6.09_all.deb ...
Unpacking tex-common (6.09) ...
Selecting previously unselected package fonts-lmodern.
Preparing to unpack .../04-fonts-lmodern_2.004.5-3_all.deb ...
Unpacking fonts-lmodern (2.004.5-3) ...
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack .../05-fonts-noto-mono_20171026-2_all.deb ...
Unpacking fonts-noto-mono (20171026-2) ...
Selecting previously unselected package fonts-texgyre.
Preparing to unpack .../06-fonts-texgyre_20160520-1_all.deb ...
Unpacking fonts-texgyre (20160520-1) ...
Selecting previously unselected package javascript-common.
Preparing to unpack .../07-javascript-common_11_all.deb ...
Unpacking javascript-common (11) ...
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack .../08-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb ...
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) ...
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack .../09-libcupsimage2_2.2.7-1ubuntu2.8_amd64.deb ...
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.8) ...
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack .../10-libijs-0.35_0.35-13_amd64.deb ...
Unpacking libijs-0.35:amd64 (0.35-13) ...
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack .../11-libjbig2dec0_0.13-6_amd64.deb ...
Unpacking libjbig2dec0:amd64 (0.13-6) ...
Selecting previously unselected package libgs9-common.
```

```
Preparing to unpack .../12-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.14_all.deb
...
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.14) ...
Selecting previously unselected package libgs9:amd64.
Preparing to unpack .../13-libgs9_9.26~dfsg+0-0ubuntu0.18.04.14_amd64.deb ...
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.14) ...
Selecting previously unselected package libjs-jquery.
Preparing to unpack .../14-libjs-jquery_3.2.1-1_all.deb ...
Unpacking libjs-jquery (3.2.1-1) ...
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack .../15-libkpathsea6_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libpotrace0.
Preparing to unpack .../16-libpotrace0_1.14-2_amd64.deb ...
Unpacking libpotrace0 (1.14-2) ...
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack .../17-libptexenc1_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package rubygems-integration.
Preparing to unpack .../18-rubygems-integration_1.11_all.deb ...
Unpacking rubygems-integration (1.11) ...
Selecting previously unselected package ruby2.5.
Preparing to unpack .../19-ruby2.5_2.5.1-1ubuntu1.10_amd64.deb ...
Unpacking ruby2.5 (2.5.1-1ubuntu1.10) ...
Selecting previously unselected package ruby.
Preparing to unpack .../20-ruby_1%3a2.5.1_amd64.deb ...
Unpacking ruby (1:2.5.1) ...
Selecting previously unselected package rake.
Preparing to unpack .../21-rake_12.3.1-1ubuntu0.1_all.deb ...
Unpacking rake (12.3.1-1ubuntu0.1) ...
Selecting previously unselected package ruby-did-you-mean.
Preparing to unpack .../22-ruby-did-you-mean_1.2.0-2_all.deb ...
Unpacking ruby-did-you-mean (1.2.0-2) ...
Selecting previously unselected package ruby-minitest.
Preparing to unpack .../23-ruby-minitest_5.10.3-1_all.deb ...
Unpacking ruby-minitest (5.10.3-1) ...
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack .../24-ruby-net-telnet_0.1.1-2_all.deb ...
Unpacking ruby-net-telnet (0.1.1-2) ...
Selecting previously unselected package ruby-power-assert.
Preparing to unpack .../25-ruby-power-assert_0.3.0-1_all.deb ...
Unpacking ruby-power-assert (0.3.0-1) ...
Selecting previously unselected package ruby-test-unit.
Preparing to unpack .../26-ruby-test-unit_3.2.5-1_all.deb ...
Unpacking ruby-test-unit (3.2.5-1) ...
Selecting previously unselected package libruby2.5:amd64.
```

```
Preparing to unpack .../27-libruby2.5_2.5.1-1ubuntu1.10_amd64.deb ...
Unpacking libruby2.5:amd64 (2.5.1-1ubuntu1.10) ...
Selecting previously unselected package libsynctex1:amd64.
Preparing to unpack .../28-libsynctex1_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libtexlua52:amd64.
Preparing to unpack .../29-libtexlua52_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
.../30-libtexluajit2_2017.20170613.44572-8ubuntu0.1_amd64.deb ...
Unpacking libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack .../31-libzzip-0-13_0.13.62-3.1ubuntu0.18.04.1_amd64.deb ...
Unpacking libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) ...
Selecting previously unselected package lmodern.
Preparing to unpack .../32-lmodern_2.004.5-3_all.deb ...
Unpacking lmodern (2.004.5-3) ...
Selecting previously unselected package preview-latex-style.
Preparing to unpack .../33-preview-latex-style_11.91-1ubuntu1_all.deb ...
Unpacking preview-latex-style (11.91-1ubuntu1) ...
Selecting previously unselected package t1utils.
Preparing to unpack .../34-t1utils_1.41-2_amd64.deb ...
Unpacking t1utils (1.41-2) ...
Selecting previously unselected package tex-gyre.
Preparing to unpack .../35-tex-gyre_20160520-1_all.deb ...
Unpacking tex-gyre (20160520-1) ...
Selecting previously unselected package texlive-binaries.
Preparing to unpack .../36-texlive-
binaries_2017.20170613.44572-8ubuntu0.1_amd64.deb ...
Unpacking texlive-binaries (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package texlive-base.
Preparing to unpack .../37-texlive-base_2017.20180305-1_all.deb ...
Unpacking texlive-base (2017.20180305-1) ...
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack .../38-texlive-fonts-recommended_2017.20180305-1_all.deb ...
Unpacking texlive-fonts-recommended (2017.20180305-1) ...
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack .../39-texlive-plain-generic_2017.20180305-2_all.deb ...
Unpacking texlive-plain-generic (2017.20180305-2) ...
Selecting previously unselected package texlive-generic-recommended.
Preparing to unpack .../40-texlive-generic-recommended_2017.20180305-1_all.deb
...
Unpacking texlive-generic-recommended (2017.20180305-1) ...
Selecting previously unselected package texlive-latex-base.
Preparing to unpack .../41-texlive-latex-base_2017.20180305-1_all.deb ...
```

```
Unpacking texlive-latex-base (2017.20180305-1) ...
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack .../42-texlive-latex-recommended_2017.20180305-1_all.deb ...
Unpacking texlive-latex-recommended (2017.20180305-1) ...
Selecting previously unselected package texlive-pictures.
Preparing to unpack .../43-texlive-pictures_2017.20180305-1_all.deb ...
Unpacking texlive-pictures (2017.20180305-1) ...
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack .../44-texlive-latex-extra_2017.20180305-2_all.deb ...
Unpacking texlive-latex-extra (2017.20180305-2) ...
```

If the above cell doesn't work, try this alternative.