Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs

Deep Learning Project Update 1

Spring 2023

Team

Likhith Sai Chaitanya Vadlapatla

Ajay Kumar Ganipineni

Dataset:

Data Collection:

Images are Captured with mobile handset across the university of New Haven campus.

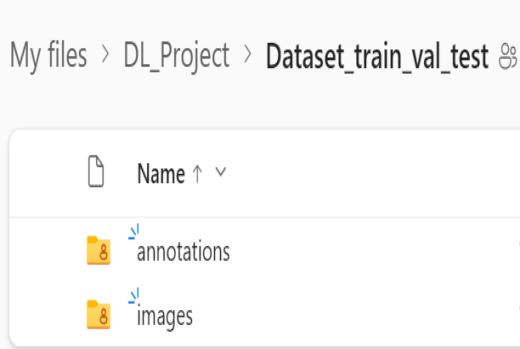
Data Processing:

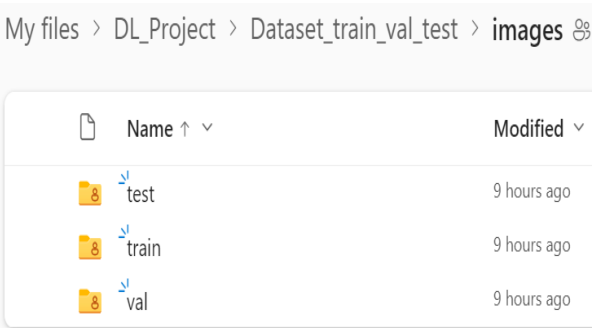
Our captured images are of size between 5MB to 12MB and of different mobile handset format, we pre-processed the images to reduce image size and to desired format. We annotated with 5 labeled classes (building, car, traffic sign, chair, person). We used MATLAB to annotate the images.

Then we divided the datasets into three parts :

training dataset (80% of the data), validation dataset (10% of the data), testing dataset (remaining 10% of the data).

Folder Structure:





Loading:

We also made our dataset available in UNH OneDrive link:(https://unhnewhaven-my.sharepoint.com/:f:/g/personal/agani3_unh_newhaven_edu/ErFKvLiA2j9Cu47SfKWfIIUB9mp7Z7gURsTFfKkOEhMOsQ?e=AEEwc9)

Methodology:

1. First, we have created the Custom Dataset class in which we loaded images from local system path.
2. In the default init method we have retrieved the path that we have received as an input and by using list comprehension we only retrieve the jpg images from the folder.
3. Then, we have created two functions inside the class:
   1. **len** - which retrieves the number of images present.
   2. **getitem** - In this function we have read both image and masked image and then we have converted it to tensor. Also we have resized and normalized the images using transform function.
4. Next we processed the RGB images and masked images in batch size of 8 and visualized the first image of the each batch.

Custom dataset class code Snippet:

```
# Creating a CustomDataset class which retrives the images and annotations.

class CustomDataset(Dataset):
    def __init__(self,path_of_image,path_of_maskedimage):
        super().__init__()
        images_path = Path(path_of_image)
        maskedimages_path = Path(path_of_maskedimage)
        self.images = [p for p in images_path.glob('*.jpg')]
        self.maskedimages = [p for p in maskedimages_path.glob('*.png')]
        self.transform1 = torchvision.transforms.Compose([torchvision.transforms.Resize((1080,1080)),
                                                          torchvision.transforms.Normalize((0.0,0.0,0.0),(255.,255.,255.))])
        self.transform2 = torchvision.transforms.Compose([torchvision.transforms.Resize((1080,1080)),
                                                          torchvision.transforms.Normalize((0.0),(255.))])

    def __len__(self):
        length = len(self.images)
        return length

    def __getitem__(self,index):
        img = torchvision.io.read_image(str(self.images[index]))
        masked_img = torchvision.io.read_image(str(self.maskedimages[index]))
        img = torch.tensor(img,dtype=torch.float)
        masked_img = torch.tensor(masked_img,dtype=torch.float)
        img = self.transform1(img)
        masked_img = self.transform2(masked_img)
        return img,masked_img
```

Visualization code snippet:

In [10]:
```
for i in range(0,3):
    dataset = d[i]
    train_dataloader = torch.utils.data.DataLoader(dataset,batch_size=8)
    if i == 0 :
        print_with_font_size("Images and Annotated Images of Training Images : ", font_size=6)
    elif i == 1 :
        print_with_font_size("Images and Annotated Images of Validation Images : ", font_size=6)
    else :
        print_with_font_size("Images and Annotated Images of Testing Images : ", font_size=6)
    for batch in train_dataloader:
        imgs,masked_imgs = batch
        img_np = imgs[0].permute([1,2,0]).numpy()
        maskedimage_np = masked_imgs[0].permute([1,2,0]).numpy()
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
        ax1.imshow(img_np)
        ax2.imshow(maskedimage_np)
        plt.show()
```

Images and Annotated Images of Training Images :

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