Problem1

B09901104 翁瑋杉

1.Please explain

- **Nerf idea in your word:**
  - The core concept of Nerf is to encode 5D information into MLP, and then apply volume rendering methods to project 3D object to 2D images.
- **Which part of Nerf is the most important**
  - I find two parts very novel and equally important. First, they come up with 5D representation of 3D objects, and each dimension is well-defined and has its own essential purposes. Second, I also think “Hierarchical Volume Sampling” plays a significant role in this paper. Splitting the network to coarse and fine ones, the authors were able to focus on regions where object exists when sampling. I think it is a smart way to optimize image quality.
- **Compare NeRF’s pros/cons w.r.t. other novel view synthesis work**
  - Pros:
    - Better image resolution
    - represent fine geometry more consistently across rendered views
    - correctly reconstructs partially occluded regions that LLFF struggles to render cleanly
    - requires only 5 MB for the network weights
  - Cons:
    - Training time was way too long.

2. Describe the implementation details of Direct Voxel Grid Optimization(DVGO) for the given dataset. You need to explain DVGO's method in your own ways.

- Coarse geometry searching:
  - Recalling "Hierarchical Volume Sampling" method adopted in original Nerf paper, DVGO further optimize this step. In coarse stage, DVGO searches for coarse scenes, allocates coarse voxels and sample points. In fine stage, DVGO first densely query coarse density voxel grid for fine voxels allocation, then a higher-resolution density voxel grid is utilized for progressive scaling, after which DVGO samples fine-stage points.

3. Given novel view camera pose from **transforms_val.json**, your model should render novel view images.
Please evaluate your generated images and ground truth images with the following three metrics (mentioned in the NeRF paper).
Try to use at least two different hyperparameter settings and discuss/analyze the results.

- Please report the PSNR/SSIM/LPIPS on the validation set
  - PSNR: 35.19
  - SSIM: 0.9745
  - LPIPS: 0.0412637

- ***You also need to explain the meaning of these metrics (ref)***
  - PSNR shows a ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.However, PSNR is a variation of the MSE and still concentrates on pixel-by-pixel comparison.
  - SSIM is correlated with the quality and perception of the human visual system (HVS color model).SSIM models image distortion as a combination of three factors that are loss of correlation, luminance distortion, and contrast distortion.
  - The Learned Perceptual Image Patch Similarity (*LPIPS*) is used to judge the perceptual similarity between two images. LPIPS essentially computes the similarity between the activations of two image patches for some pre-defined network. This measure has been shown to match human perception well.

- ***Different configuration settings such as iteration number/number of voxel/ stepsize … lead to different performance.***

Setting	PSNR	SSIM	LPIPS
N_iters(coarse) : 4000 N_iters(fine): 22000	35.2057	0.97467	0.04100
step_size: 0.8 (origin: 0.5)	35.0128	0.97344	0.04260

By increasing the number of iterations for fine model, we can get better score on all three metrics. I think it is because the model is trained more, and thus performs better. However, there is a trade off for training time and performance.

By increasing step size, we get lower score on each metrics. I think it is reasonable because we don’t want the sampling size during volume rendering be too large, which could lead to coarse rendering and low image equality.

Problem2

B09901104 翁瑋杉

1. Describe the implementation details of your SSL method for pre-training the ResNet50 backbone. (Include but not limited to the name of the SSL method you used, data augmentation for SSL, learning rate schedule, optimizer, and batch size setting for this pre-training phase)

```
"num_epochs": 500,
"batch_size": 32,
"lr": 2e-5,
```

```
# learner=BYOL(resnet,image_size=128,hidden_layer="avgpool")
learner = BYOL(resnet, image_size=128, hidden_layer="avgpool")
learner = learner.to(device)
opt = torch.optim.Adam(learner.parameters(), lr=config["lr"])
```

```
class ImgDataset(Dataset):
    def __init__(self, data_dir):
        super(ImgDataset, self).__init__()
        self.data_dir = data_dir
        self.files = sorted([p for p in os.listdir(data_dir)])
        self.tfm = T.ToTensor()

    def __len__(self):
        return len(self.files)

    def __getitem__(self, idx):
        fname = self.files[idx]
        img = Image.open(os.path.join(self.data_dir, fname))
        img = self.tfm(img)
        return img
```

2. Please conduct the Image classification on **Office-Home** dataset as the downstream task. Also, please complete the following Table, which contains different image classification setting, and **discuss/analyze** the results.

Setting	Pre-training (Mini-ImageNet)	Fine-tuning (Office-Home dataset	Validation accuracy
A	-	Train full model (backbone + classifier)	0.033
B	w/ label	Train full model (backbone + classifier)	0.44
C	w/o label	Train full model (backbone + classifier)	0.413
D	w/ label	Fix the backbone. Train classifier only	0.423
E	w/o label	Fix the backbone. Train classifier only	0.401

Setting A has the worst result because it trains from scratch and has so little image to learn.
Setting B has the highest score, because its backbone is trained by TAs, and I think they do know some tricks to get better performance.
Setting C has higher score the setting E, because backbone is trained on mini, but we fine-tune and office dataset, so there's domain difference. Fixing the parameters will prevent model from fitting more in office.