NTU DLCV (Autumn, 2022) HW4 Report

R10921069　沈郁鈞

Problem 1

1

a. The key idea behind NeRF is to use a neural network to model the 3D structure and appearance of a scene. The network takes as input a set of view parameters (e.g., the location and orientation of the camera) and outputs an estimate of the radiance (i.e., the amount of light) at each point in the scene. This allows NeRF to generate high-quality, photorealistic images of the scene from any viewpoint.

b. I thought **inference algorithm** is the most important part. NeRF uses a special inference algorithm that allows it to generate images from any viewpoint, not just the ones used in the training data.

c.

- 😀 Pros:
  - Excellent quality & flexibility.
  - Able to handle complex, realistic scenes with many **still** objects and materials.
- 😕 Cons:
  - The inference speed is very slow because MLP queries.
  - Not able to handle scenes with moving objects.

2

In DVGO, the 3D scene is represented as a voxel grid, which is a 3D array of voxels. Each voxel encodes the color and depth information of the scene at that point. The neural network takes as input a set of view parameters and outputs a predicted voxel grid for the novel view.

3

- **PSNR** (*Peak signal-to-noise ratio*): The ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Often used for evaluating quantized difference between two images.
- **SSIM** (*Structural Similarity Index Measure*): A method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. It's often used for measuring the similarity between two images.

- **LPIPS** (*Learned Perceptual Image Patch Similarity*): Used to judge the perceptual similarity between two images. It essentially computes the similarity between the activations of two image patches for some pre-defined network.

Setting	PSNR	SSIM	LPIPS(VGG)	LPIPS(Alexnet)
Default setting (`config/default.py`)	35.18	0.9742	0.041	0.022
fine_model_and_render.num_voxels = 256^3	**35.38**	**0.9754**	**0.038**	**0.019**

Problem 2

1

- Method Used for Seif-Supervised Learning: **BYOL** (https://github.com/lucidrains/byol-pytorch)
- Data Augmentation: None
  - Image Transformation: Same as `TRANSFORM_IMG` provided in slide.

```
TRANSFORM_IMG = transforms.Compose([
    transforms.Resize(128),
    transforms.CenterCrop(128),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225] )
    ])
```

- Batch-Size: 256 (Pretrain on Mini-ImageNet) / 64 (Finetune on Office-Home)
- Learning Rate: Constant 3×10^{-4} without scheduling
- Optimizer: `Adam`
- Weight Decay (Setting in Optimizer): None (Pretrain on Mini-ImageNet) / 5×10^{-4} (Finetune on Office-Home)
- Pretrain Epochs on mini-ImageNet: 300
- Finetune Epochs on Office-Home: 30

2

Setting	Pre-training (Mini-ImageNet)	Fine-tuning (Office-Home dataset)	Validation accuracy (Office-Home dataset)
A	-	Train Full Model	0.268473
B	with label	Train Full Model	0.389163
C	without label	Train Full Model	0.433498
D	with label	Train Classifier Only	0.263547
E	without label	Train Classifier Only	0.258621