DLCV HW4 Report

資工所 呂兆凱 R11922098

Problem 1

1. Please explain:
    a. the NeRF idea in your own words

    NeRF 網路會使用一組五維的參數，裡面包含 location 與 view direction 的資訊，輸出則是 emitted color 與 volume density。NeRF 網路會先利用 location生成volume density，再將volume density與view direction 做 concat後生成emitted color，最後再利用積分產生某點的顏色。

    b. which part of NeRF do you think is the most important

    Positional Encoding :
    由於深度網路對於學習 high frequency 的函數較為不擅長，又實際場景的輸入即為 high frequency。因此改使用 positional encoding 作為輸入來 encode location & view direction 資訊。

    c. compare NeRF's pros/cons w.r.t. other novel view synthesis work
        - pros
            - 生成的圖片細緻度較其他方法來得高
            - 不對場景有任何假設，甚至半透明物體的圖片也可以生成
        - cons
            - 需要大量不同角度的圖片來作為訓練的輸入，使得準備資料成本與計算成本增加
            - 如果圖片有被遮擋就無法產生高品質的圖片

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.

    DVGO 對於 density 直接使用 Voxel grid 來表示，並利用差值法算出各個位置的 density ; 並在 Voxel 裡面儲存一個多維向量，這個向量可以經過 MLP decode 形成 rgb 值，以上方式可以使得 MLP 的使用減少很多，就可以加速訓練。

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.

a. Please report the PSNR/SSIM/LPIPS on the validation set. (refer to DVGO's github)
   - PSNR : 35.257
   - SSIM : 0.975
   - LPIPS : 0.040

b. You also need to explain the meaning of these metrics.
   i. PSNR : Peak Signal to Noise Ratio, 衡量最大的像素值和背景的像素值的比值，PSNR若>40db則表示圖片跟原圖差不多好。
   ii. SSIM : Structural Similarity Index, 會同時考慮到圖片的結構、對比度以及亮度，值的範圍為0~1, 數值越大則表示圖片越相似。
   iii. LPIPS : Learned Perceptual Image Patch Similarity, 會對於圖片所取出之特徵進行相似度的計算，此方法較為接近人類所感知的情況較為接近，數值越大則表示圖片越相似。

c.

Default Setting :

```
coarse_train = dict(
    N_iters=5000,
    N_rand=8192,
    lrate_density=1e-1,
    lrate_k0=1e-1,
    lrate_rgbnet=1e-3,
    lrate_decay=20,
    pervoxel_lr=True,
    pervoxel_lr_downrate=1,
    ray_sampler='random',
    weight_main=1.0,
    weight_entropy_last=0.01,
    weight_nearclip=0,
    weight_distortion=0,
    weight_rgbper=0.1,
    tv_every=1,
    tv_after=0,
    tv_before=0,
    tv_dense_before=0,
    weight_tv_density=0.0,
    weight_tv_k0=0.0,
    pg_scale=[],
    decay_after_scale=1.0,
    skip_zero_grad_fields=[],
    maskout_lt_nviews=0,
)
```

```
data = dict(
    datadir=None,
    dataset_type=None,
    inverse_y=False,
    flip_x=False,
    flip_y=False,
    annot_path='',
    split_path='',
    sequence_name='',
    load2gpu_on_the_fly=False,
    testskip=1,
    white_bkgd=False,
    rand_bkgd=False,
    half_res=False,
    bd_factor=.75,
    movie_render_kwargs=dict(),

    # Below are forward-facing
    ndc=False,
    spherify=False,
    factor=4,
    width=None,
    height=None,
    llffhold=8,
    load_depths=False,

    # Below are unbounded inwar
    unbounded_inward=False,
    unbounded_inner_r=1.0,
)
```

```
coarse_model_and_render = dict(
    num_voxels=1024000,
    num_voxels_base=1024000,
    density_type='DenseGrid',
    k0_type='DenseGrid',
    density_config=dict(),
    k0_config=dict(),
    mpi_depth=128,
    nearest=False,
    pre_act_density=False,
    in_act_density=False,
    bbox_thres=1e-3,
    mask_cache_thres=1e-3,
    rgbnet_dim=0,
    rgbnet_full_implicit=False,
    rgbnet_direct=True,
    rgbnet_depth=3,
    rgbnet_width=128,
    alpha_init=1e-6,
    fast_color_thres=1e-7,
    maskout_near_cam_vox=True,
    world_bound_scale=1,
    stepsize=0.5,
)
```

```
fine_train = deepcopy(coarse_train)
fine_train.update(dict(
    N_iters=20000,
    pervoxel_lr=False,
    ray_sampler='in_maskcache',
    weight_entropy_last=0.001,
    weight_rgbper=0.01,
    pg_scale=[1000, 2000, 3000, 4000],
    skip_zero_grad_fields=['density', 'k0'],
))
```

```
fine_model_and_render = deepcopy(coarse_model_and_render)
fine_model_and_render.update(dict(
    num_voxels=160**3,
    num_voxels_base=160**3,
    rgbnet_dim=12,
    alpha_init=1e-2,
    fast_color_thres=1e-4,
    maskout_near_cam_vox=False,
    world_bound_scale=1.05,
))
```

Setting	PSNR	SSIM	LPIPS
Default Setting	35.19	0.974	0.041
Change coarse_train.N_iters = 10000, Change fine_train.N_iters = 30000	35.23	0.975	0.041
Change coarse_model_and_render.num_voxels = 2048000, Change coarse_model_and_render.num_voxels_base = 2048000	35.25	0.975	0.040

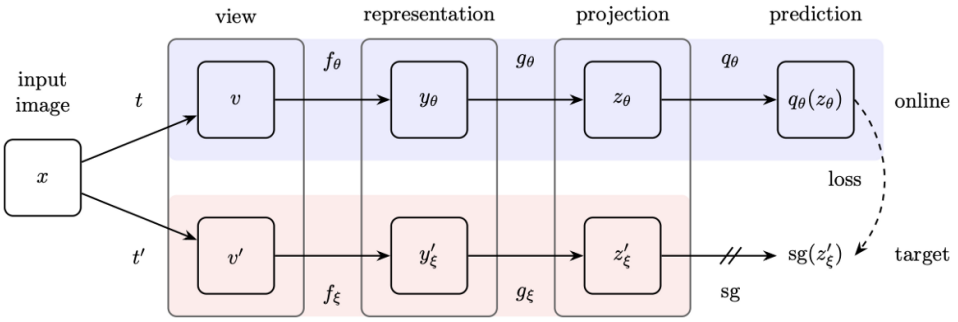
- discuss/analyze
  - Increase N_iters :
    增加 iteration 步數雖然會使得訓練網路時間加長，但也可以使產生的圖片的過程可以更為精細，產生的圖片品質也會更好一些。
  - Increase num_voxels :
    因為增加了 voxel 的數量，可以讓產生出來的圖片更加細緻，因此 PSNR 的分數有些微上升。

Problem 2

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)

- SSL method : BYOL
  BYOL 由兩個網路組成，上面為 online network，下面為 target network，輸入為一張圖片的兩種不同 data augmentation 結果，其中 target network 相較 online network 又少了一個 predictor。訓練目標為去縮小兩個 network 所輸出的 representation 距離，而在更新參數時，online

network 的參數會進行梯度下降的更新，target network 則不會透過梯度下降更新參數，而是藉由 online network 參數的 EMA 來更新參數。



- Data augmentation :
  - Resized (128 × 128)
  - Horizontal flip
  - Color distortion
    - brightness
    - contrast
    - saturation
    - hue adjustments
    - grayscale conversion
  - Gaussian blur and solarization
- Learning rate : 0.0003
- Optimizer : Adam
- Batch size : 128

2. Please conduct the Image classification on the 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	Fine-tuning	Validation accuracy
A	-	Train full model	29%
B	w/ label	Train full model	28%
C	w/o label	Train full model	43%
D	w/ label	Fix the backbone	23%
E	w/o label	Fix the backbone	40%

- discuss/analyze :
  可以看到在固定其他的情況下，fix backbone 會讓整體的表現降低，因此如果可以去 fine tune 整個包含 backbone 的網路，可以再提升整體的效果。

Reference :

- NeRF: Representing Scene as Neural Radiance Fields for View Synthesis
- Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction
- Bootstrap Your own Latent
- Guide: Neural Radiance Field (NeRF)
- 白話Neural Radiance Fields (NeRF): 類神經網路在View Synthesis的熱門新方向
- Neural Radiance Fields (NeRF)
- 有真实参照的图像质量的客观评估指标:SSIM、PSNR和LPIPS
- 綜述 | 基於神經輻射場的（NeRF-based）SLAM
- BYOL：无需负样本就可以做对比自监督学习
- 论文解读（BYOL）《Bootstrap Your Own Latent A New Approach to Self-Supervised Learning》