DLCV HW4

r11922a05 資工AI碩一 林聖硯

Problem 1: 3D Novel View Synthesis

1. Please explain:

a. the NeRF idea in your own words

Neural Radiance Fields (NeRF) is a kind of deep learning model that takes a sparse set of input views (including 3D location x = (x,y,z) and 2D viewing direction (θ,φ)) to opimtimize a continuous volumetric scene function. In the training stage, the optimization process uses gradient descent to minimize errors between each observed image and all corresponding views rendered from the representation. NeRFs **do not** use convolutional layers and instead rely on deep, fully-connected multi-layer perceptrons (MLPs). In the inference stage, we'll generate a set of 3D points, inputting these points with their corresponding 2D viewing directions into the neural network, and then using classical volume rendering techniques to accumulate the densities and colors into a 2D image.

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

I think the following two parts of NeRF paper are the most important

- Section 3: Neural Radiance Field Scene Representation
- Section 5: Optimizing a Neural Radiance Field (Positional Encoding and Hierarchical volume sampling)

The first part covers the most essential idea of this paper, which is using a neural radiance field to model scenes. The second part of the paper further improve this idea by designing two methods to optimize the neural radiance field

c. compare NeRF's pros/cons w.r.t. other novel view synthesis work

	NeRF	NV (Neural Volumes)	SRN (Scene Representation Networks)	LLFF (Local Light Field Fusion)
Pros	Good at recover fine details in both geometry and appearance	Good at capture reasonably detailed volumetric		Fast (can process a small input dataset in

		geometry and appearance		under 10 minutes)
Cons	Slow (comparing to other methods)	Fail to represent fine details at high resolutions due to its explicit voxel grid	Bad synthetic reresults, produces heavily smoothed geometry and texture	1. Fail to estimate correct geometry in the synthetic datasets which contain up to 400-500 pixels of disparity between views 2. Require enormous memory

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 improves the work of NeRF by optimizing the volume density model in a dense voxel grid **directly**. Also, the authors of DVGO propose two methods to speed up NeRF convergence and produce high quality output. First, they found that **post-activation interpolation** on voxel density could produce sharp surfaces in lower grid resolution. Second, they use several priors (low-density initialization and view-count-based learning rate) to robustify the optimization process.

Implementation details on NeRF dataset:

- In the "coarse" stage
  - The expected number of voxels is 100^3
  - The activated alpha values are initialized to 10^{-6}
- In the "fine" stage
  - The expected number of voxels is 160^3, the activated alpha values are initialized to 0.01.
  - The shallow MLP layer has two hidden layers with 128 channels.
  - The number of channels of the feature voxel grid is set to D = 12. The number of frequencies of positional embedding for the query position x and the viewing direction are $k_x = 5, k_d = 4$.
  - Threshold τ(c) = 10−3 is used to define the known free space and τ(f) = 10−4 is used to skip the low-density query points in unknown space.

- The sampling step sizes are half of the voxel sizes
- During training, the query points are randomly shifted on a ray via the following function

$$\left(\bar{p} \cdot \delta^{(\cdot)} \cdot \bar{d} / \|d\|^2\right)$$

- Adam optimizer, batch size = 8192 rays
- Base learning rate = 0.1 for all voxel grids and 10^{-3} for MLP
- The exponential learning rate decay is applied such that the learning rates are downscaled by 0.1 at 20k iterations.

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.

Meaning of the metrics

- Peak signal-to-noise ratio (PSNR): computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.
- Structural Similarity Index (SSIM): A metric that assesses the perceived quality of an image or video based on its structural similarity to a reference image. It is often used as a quality assessment tool in image and video processing, and it is particularly useful for comparing images that have undergone lossy compression or other types of distortion
- Learned Perceptual Image Patch Similarity (LPIPS): Evaluate the distance between image patches, via computing the similarity between the activations of two image patches for some pre-defined network. Higher means further/more different. Lower means more similar.

Setting	PSNR	SSIM	LPIPS
default settings in DVGO	35.1771	0.9742	0.0408
default settings + N_iters: 5000 → 10000 (in coarse stage)	35.2235	0.9746	0.0413

default settings + N_iters: 5000 → 10000 (in coarse stage) + rgbnet_depth = 3 → 6 + rgbnet_width = 128 → 256	35.662	0.9775	0.0359
default settings + N_iters: 5000 → 10000 (in coarse stage) + N_iters: 20000 → 40000 (in fine stage) + rgbnet_depth = 3 → 6 + rgbnet_width = 128 → 256	35.656	0.9775	0.0354

Observation

The default setting of DVGO could already pass the validation baseline, but I still did some experiment on hyperparameter tuning since the paper told us that “setting N_iters, N_rand, num_voxels, rgbnet_depth, rgbnet_width to larger values or setting stepsize to smaller values typically leads to better quality but need more computation”.
From the experiment we could conclude that setting `rbgnet_depth` and `rgbnet_width` to a higher value could improve the model’s performance. However, under this setting, when I change the `N_iters` in fine stage from 20000 to 40000, the training PSNR is augmented to 37.07. However, the validation PSNR decreases to 35.656. The model is prone to overfit the training dataset if we increase the number of optimization steps too much. Therefore, I could choose the third setting as my final submitted model.

Problem 2: Self-Supervised Pre-training for Image Classification

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)

- **Data augmentation** (in both pretraining and fine-tuning)

```
transform = transforms.Compose([
    transforms.Resize(128),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=.5, hue=.3),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

- Pretraining
  - Optimizer: Adam
  - learning rate: 3e-4
  - No learning rate scheduling
  - batch size: 128
  - number of epoch: 500
- Fine-tuning:
  - Optimizer: Adam
  - learning rate: 1e-3
  - No learning rate scheduling
  - batch size: 128
  - number of epoch: 30

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-Image Net)	Fine-tuning (Office-Home dataset)	Validation Accuracy (Office-Home dataset)
A	-	Train full model (backbone + classifier)	24.81%
B	w/ label (TA's model)	Train full model (backbone + classifier)	39.69%
C	w/o label (SSL)	Train full model (backbone + classifier)	54.43%
D	w/ label (TA's model)	Fix the backbone. Train classifier only	32.11%

E	w/o label (SSL)	Fix the backbone. Train classifier only	37.62%

Discussion

- The model perfomance is better when we train full model instead of fixing the backbone. The reason causing this situation is that we pretrained our model on a domain different from that we do validation/testing on. Thus, if we let the whole model update when training on new domain data, the feature extractor (backbone) could learn how to fit on the new domain data well as will as keeping the information that it learned from pre-training dataset.
- The model perfomance is better when trained with self-supervised setting instead of training in a supervised fashion. It shows that self-supervised is quite useful for pretraining in visual task.