DLCV hw4

ID: r12922075

Name: 吳韋論

3D Novel View Synthesis

1. Explain the following questions

- NeRF idea in your own words

  NeRF uses a MLP network to predict colors and density for a scene based on position and direction inputs. These predictions are then used to create the scene through volume rendering, and NeRF can calculate loss based on ground truth and predicted scene.

  Optimization details:

  - To improve scene quality, NeRF uses positional encoding to change position to higher dimensional space.
  - For faster rendering, NeRF employs a two-step process: a coarse network followed by a fine network, which is importance sampling.
  - In the neural network, density is calculated only by position, while RGB color is affected by both position and direction.

- Which part of NeRF do you think is the most important

  I think the most important part is the design of the MLP network. It takes position and direction as inputs, uses positional encoding twice to map them to higher dimensions, resulting in high-resolution scene generation. This simple network structure achieves SOTA performance.

- Compare NeRF’s pros/cons w.r.t. other novel view synthesis work

  Compare to InstantNGP

  - Pros: NeRF only requires position and directions as inputs
  - cons: InstantNGP uses a mixture of local planar surfaces to speedup rendering

Compare to DVGO

- Pros: NeRF only requires position and directions as inputs
- Cons: NeRF needs long training time and inference time, DVGO uses voxel grid to achieve fast convergence

2. Describe the implementation details of *your NeRF model*

Most of the codes are from nerf_pl github link. Below are the files in which the code has been modified to fit the dataset and the environment provided by the TA.

- Requirement: Only uses TA requirement.txt, and version in my pc is below

```
torch==2.1.1
torchvision==0.16.1
matplotlib==3.8.2
pytorch-lightning==2.1.2
opencv-python==4.8.1.78
kornia==0.7.0
scipy==1.11.4
imageio==2.33.0
lpips==0.1.4
```

- rendering.py
  - Use torch.searchsorted instead of searchsorted github
    - inds = torch.searchsorted(cdf, u, right=True)
  - Use softplus calculate alpha instead of relu
    - alphas = 1-torch.exp(-deltas*torch.nn.Softplus()(sigmas+noise))
- metric.py
  - Add SSIM and LPIPS metric
    - Reference: https://github.com/sunset1995/DirectVoxGO/blob/main/lib/utils.py#L69
- train.py
  - Turn off TestTubeLogger because of pytorch_lightning version
  - Use KlevrDataset (TA given dataset)
- dataset.py and opt.py
  - Use TA dataset.py

- Change root_dir, dataset name, image_wh, batch_size, use_disp, perturb, noise_std

All hyperparameter settings are the same instead of numbers of layers / skip / embedding sizes

3. Evaluate images with the following three metrics. Try to use at least three different hyperparameter settings and discuss/analyze the results. You also need to explain the meaning of these metrics

PSNR:

Peak Signal-to-Noise Ratio, is a metric used to quantify the quality of an image, particularly in the context of image restoration. It is calculated by comparing the original image to a restored version.

The formula for PSNR is:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

where:

- Max_I is the maximum possible pixel value in the image
- MSE is the Mean Squared Error

A higher PSNR value means less noise or distortion in the restored image, implying better quality.

SSIM:

Structural Similarity Index, is a score that calculate how similar two images are. It looks at things like brightness, contrast, and overall structure. The score ranges from -1 to 1, where 1 means the images are exactly the same. So, a higher SSIM score means the images are more similar, while a lower score indicates more differences.

The formula for SSIM is:

$$\text{SSIM}(x, y) = l(x, y)^{\alpha} \cdot c(x, y)^{\beta} \cdot s(x, y)^{\gamma}$$

LPIPS:

Learned Perceptual Image Patch Similarity with VGG is a metric that measures the similarity between two images based on features learned by a VGG neural network. Instead of focusing on pixel-wise differences like traditional metrics, LPIPS considers how humans perceive images.

A lower LPIPS score indicates higher similarity between the images, while a higher score implies more differences.

Settings

- Layers: number of layers for density (sigma) encoder
- skips: add skip connection in the Dth layer
- embedding size: number of hidden units in each layer

Settings	PSNR	SSIM	LPIPS (vgg)
layers: 8, skips: 4, embedding: 256	43.40	0.9941	0.0986
layers: 8, skips: 4, embedding: 512	43.73	0.9945	0.0991
layers: 6, skips: 3, embedding: 256	43.82	0.9943	0.1004

4. Depth rendering in your own way and visualize your results

Use depth_final and visualize_depth in visualization.py to calculate draw depth

The results have wall effect, probably because the rays do not intersect with the bounding box of the scene, so near_far is set to epsilon.

Results