Student ID: R10625016

Name: 許致銓

Department: 森林所

Problem: 3D Novel View Synthesis

1. NeRF Explanations (15%)

a. the NeRF idea in your own words

The key idea is using MLP to make discretized multi-view image information continuous and further optimized by gradient descent. For the neural network, the inputs are 3D spatial location (x, y, z) and direction information ($\theta, \emptyset$) (polar angle & azimuthal angle), and the outputs are color (r, g, b) and density (σ). The above method is based on "ray tracing". There are different methods to implement "ray tracing", and the main method in NeRF is stratified sampling. Besides, the positional encoding in Transformer is also added to largely improve the optimization process.

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

In the optimization process, there is a method called Hierarchical Volume Sampling. As I know above, stratified sampling uses an approximated method to generate the density and probability distribution between each discretized point, and we can sample points based on the distribution. However, we may sample some points with useless information such as background, and we cannot precisely represent the 3D shape of objects. As a result, we need Hierarchical Volume Sampling, including a coarse network and a fine network, to sample more points in the high-density regions from the probability distribution, leading to a high-resolution synthesis.

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

- PROS:
    1. Based on the above, we can understand that the Hierarchical Volume Sampling and positional encoding can lead to a high-resolution rendering.
    2. First continuous scene representation with high performance.
    3. No specific constraints for input images
    4. Efficient sampling method with fewer inputs.

- CONS
    1. Computational complexity is high, which results in longer training time.
    2. Model interpretability for scene representation is challenging, which is mentioned in the conclusion of the paper.

Please read through ALL reference paper to realize their ideas.

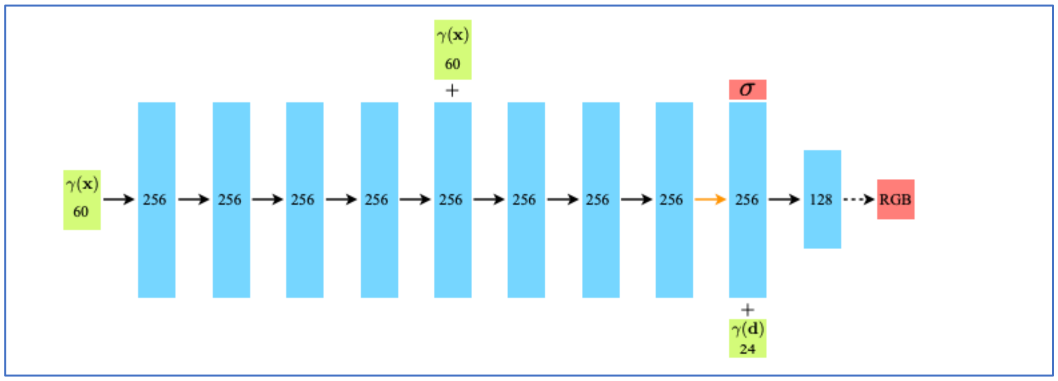
2. Describe the implementation details of *your NeRF model* for the given dataset. You need to explain your ideas completely. (15%)

My NeRF model is composed of most of the code in [1]. There are two stages of my NeRF model. First, embedding with positional encoding by sinusoidal functions can capture different frequency components, which helps generate high-resolution synthesis. Second, the neural network part is as below, there is a skip connection for the spatial location (x, y, z). Besides, one branch of the model concatenates spatial location and direction information ($\theta, \emptyset$) to predict the RGB, and the other branch predicts density through only spatial location.

The above is the main idea of my model.

Besides, for optimization in hierarchical sampling, I included the coarse and fine neural networks to sample in different scales. In this way, I can generate a more detailed and high-resolution rendering.

Hyper-parameters are defined in the next section.



3. Given novel view camera pose from metadata.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 three different hyperparameter settings and discuss/analyze the results. (15%)

- **Please report the PSNR/SSIM/LPIPS on the validation set.**
- **You also need to explain the meaning of these metrics.**
- **Different configuration settings such as MLP and embedding size, etc.**

a. Validation Set

Setting	**PSNR**	**SSIM**	**LPIPS (vgg)**
A	43.71	0.994	0.101
B	43.15	0.993	0.103
C	33.35	0.953	0.208

b. Meaning of Metrics

- **PSNR: Peak Signal-to-Noise Ratio**

  PSNR represents the ratio between the maximum value of a signal and the noise. It can estimate the fidelity of the representation of a signal. Defined by: $10 \cdot \log_{10} \frac{MAX_I^2}{MSE}$. As we can see, the higher PSNR is, the higher the quality of an image (lower MSE between noise approximation and image) is.

- **SSIM: Structural Similarity Index Measure**

  SSIM can estimate the similarity between two images based on their comparisons of structure, luminance, and contrast.

  Given two images x, y.

  SSIM(x, y) = $[s(x,y)]^{\alpha}[l(x,y)]^{\beta}[c(x,y)]^{\gamma}$ where s(x, y), l(x, y), and c(x, y) stand for structure function, luminance function and contrast function. The above function calculates the covariate of two images. The higher SSIM is, the higher the similarity comes.

- **LPIPS: Learned Perceptual Image Patch Similarity**

  Because the above metrics sometimes contradict the perception of images from humans, the paper “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric” (2018, R. Zhang et al.) designed LPIPS to better generalize the “perceptual distance” for image similarity.

The method uses the pre-defined neural network to extract the features from two images, and then calculate the similarity. The result outperforms traditional metrics in the perceptual similarity dataset. The lower the LPIPS is, the more similar the two images are.

c. Different Configuration Setting

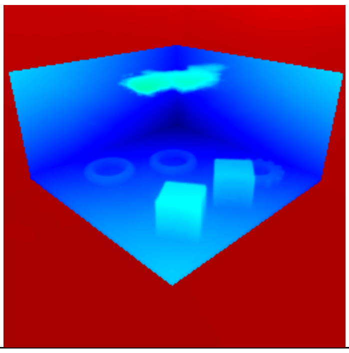
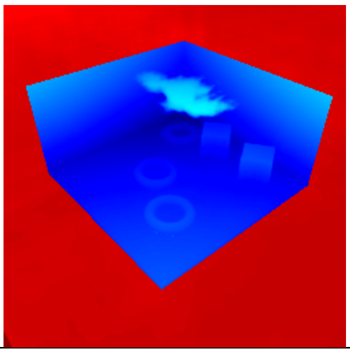
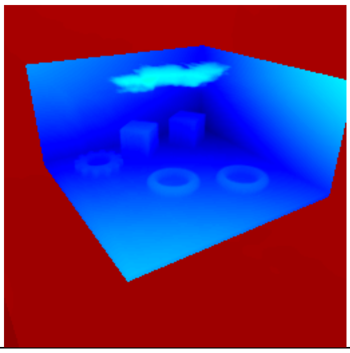
- A
  - n_epochs: 16
  - batch_size: 1024
  - optimizer: Adam
  - learning rate: 5e-4
  - lr_schedulre: stepLR
  - decay_gamma: 0.5
  - N_importance: 64 (for hierarchical sampling)
  - encoding_layers_num: 8
  - embedding_xyz_num: 10 (n_freq)
  - embedding_dir_num: 4 (n_freq)
  -
- B
  - n_epochs: 8
  - batch_size: 512
  - optimizer: Adam
  - learning rate: 5e-4
  - lr_schedulre: stepLR
  - decay_gamma: 0.5
  - N_importance: 64 (for hierarchical sampling)
  - encoding_layers_num: 4
  - embedding_xyz_num: 10 (n_freq)
  - embedding_dir_num: 4 (n_freq)
- C
  - n_epochs: 8
  - batch_size: 512
  - optimizer: Adam

- learning rate: 5e-4
- lr_schedulre: stepLR
- decay_gamma: 0.5
- N_importance: 64 (for hierarchical sampling)
- encoding_layers_num: 8
- embedding_xyz_num: 7 (n_freq)
- embedding_dir_num: 7 (n_freq)

The performance above is reasonable, setting A has the best performance with more parameters and layers. For setting B, I lowered the training epoch and batch_size to check if the features and performance were affected. With fewer parameters, the performance is a bit worse than setting A, but reduces training time to only half. As for setting C based on setting B, this time I changed the render_rays function with different embedding frequency to check if the different embeddings affect the performance. As we can see, setting C has a large drop in performance. I think it stemmed from the lower frequency range, which cannot make the embedding identifiable by positional encoding. However, as the author mentioned in the paper, it is hard to explain how the embedding features affect the performance. In my opinion, this is a kind of black box.

4. With your trained NeRF, please implement depth rendering in your own way and visualize your results. (15%)

I chose the better results of my depth rendering. However, there is a weird part in the above of the objects in each rendering.

00005.png	00235.png	00165.png
		

Reference

[1] https://github.com/kwea123/nerf_pl

[2] https://zhuanlan.zhihu.com/p/641420683

[3] https://medium.com/ai-blog-tw/%E7%99%BD%E8%A9%B1neural-radiance-fields-nerf-%E9%A1%9E%E7%A5%9E%E7%B6%93%E7%B6%B2%E8%B7%AF%E5%9C%A8view-synthesis%E7%9A%84%E7%86%B1%E9%96%80%E6%96%B0%E6%96%B9%E5%90%91-23be9411d399

[4] https://zhuanlan.zhihu.com/p/360365941

[5] https://lightning.ai/docs/pytorch/stable/common/lightning_module.html#logger

[6] https://lightning.ai/docs/pytorch/stable/api/lightning.pytorch.trainer.trainer.Trainer.html#lightning.pytorch.trainer.trainer.Trainer

[7] https://arxiv.org/abs/2003.08934