DLCV HW4

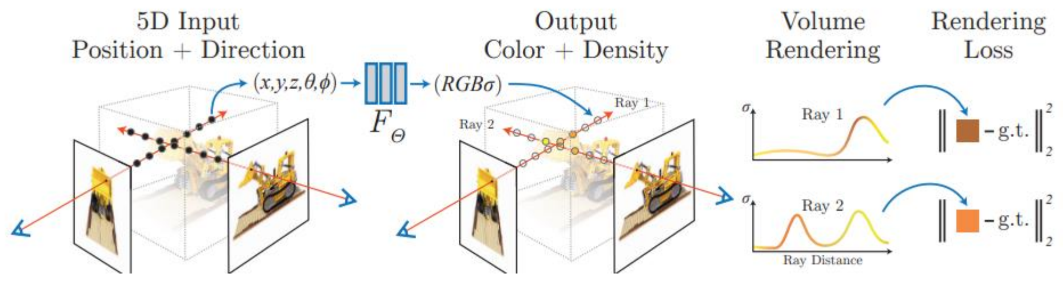
R12943010 林孟平

Problem: 3D Novel View Synthesis

Please explain:

a. the NeRF idea in your own words

If we know the scene from specific perspectives (in our training set), NeRF can help us to predict the scene from different perspective(image) that is unknown beforehand (not in our training set). We can first construct position axis and rotation axis of the scene. The ultimate goal our NeRF is: Given a particular position(x-y-z) and view dependence(theta-phi), NeRF can predict the color(r-g-b) and intensity(sigma) of the pixel on the image. To be more specific in the training stage, we will eject rays that pass through the positions that we consider, and then sample lots of points on these rays. For a single ray, the model will predict the color and intensity on each sample point, and we can use volume rendering to generate the predicted color and intensity of the pixel. In the end, we can use the predicted pixel and ground truth pixel to calculate MSE to optimize our model. After several training steps, the model will fully understand the scene and be able to generate images from different points of view.



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

I think the idea of volume rendering is probably the most fascinating part of NeRF. However, from the perspective of performance, I think the configuration of the model, especially MLP, is the most important part of NeRF. In the previous question, we explain that we use the color and intensity (model output) of each sample point on the ray to do volume rendering and then get the color of each pixel. It means that the how well the model can predict the color, how accurate the color of pixels will be, and I think the key components for color prediction is probably MLP layers and therefore act as the most important part of NeRF.

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

(1) NeRF vs DVGO

NeRF has a very good reconstruction quality and flexibility for reconstruction, but it requires large rendering time due to involving MLP queries (implicit representation). DVGO use another idea of voxel grid to store the information such as volume density and color (explicit representation). Unlike the rendering stage in NeRF, it uses “Fast and direct voxel grid optimization” with two steps to render the scene. The goal of the first step “Coarse geometry searching” is to find the 3D area of interest. After that, in the second stage “Fine Detail Reconstruction”, we can only consider the points that is occupied and thus reduce the number of queried points. The point of DVGO is that it uses voxel grid queries in the stage “Coarse geometry searching” and only involves smaller number of MLP queries in “Fine Detail Reconstruction” stage, so the speed and computing resource required for training and inference are much less than NeRF.

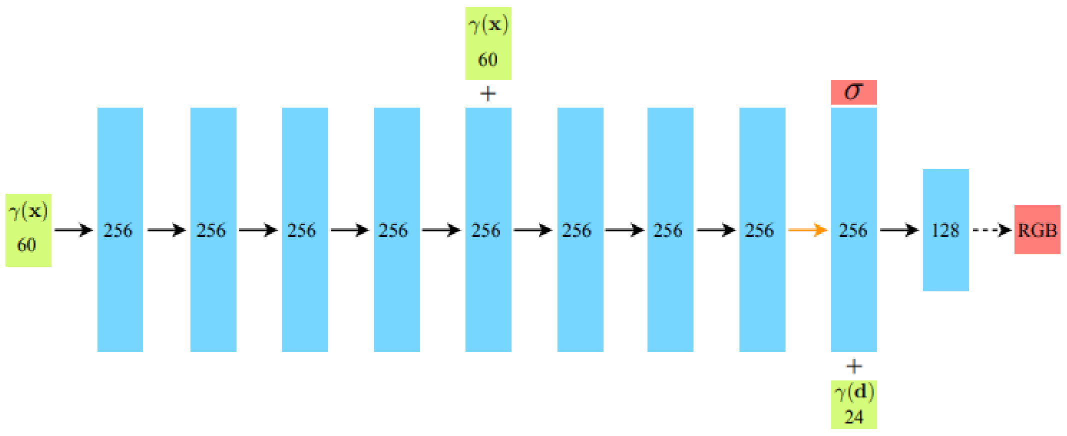
(2) NeRF vs InstantNGP

NeRF has a very good quality and flexibility for reconstruction, but the MLP size and number of layers are larger and thus requires lots amount of parameter update. InstantNGP not only uses a smaller MLP size and number of layers but also use a different encoding technique “multiresolution hash encoding” as a trainable encoding layer. Although this encoding method also requires large amounts of trainable parameters (~NeRF), the hash collisions in encoding enables the gradient relevant to the loss function dominate during training step, the hash table will then prioritize the sparse area with the most important fine detail. Therefore, no structural updates to the data structure are needed in the training step. Due to the sparsity of updates and the smaller MLP, the training speed of InstantNGP is way faster than the original NeRF (~8x).

2. Describe the implementation details of your NeRF model for the given dataset. You need to explain your ideas completely.

Model

My code is adapted from reference repo2:
https://github.com/kwea123/nerf_pl
The structure of model is the same as the original paper



The thing worth mentioning is: the view direction will only contribute to the calculation of last second layers of the model and it is independent from the intensity of the sample point.

The overall idea of how we do training has been mentioned in question 1(a), the implementation detail of volume rendering is also the same as the original paper. Note that we do 2-step sampling (coarse + fine) to do volume rendering and get more accurate color prediction on each pixel.

Volume rendering:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i \,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$$

C(r): Predicted colors of pixels
Ti: Weights of sample points, indicate how much light is blocked earlier along the ray
Sigma i: predicted intensites of sample points
Delta i: distance between sample points
Ci: Predicted colors of sample points

Loss function with corase and fine sampling:

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

Cc(r): Predicted color of pixels from coarse sampling

Cf(r): Predicted color of pixels from (coarse + fine) sampling

C(r): Color of ground truth pixels

We also add positional encoding layer. The input position and direction will first pass high-frequency function to reach higher dimension space and then be feed into the MLP layers. This method enables the MLPs to represent high-frequency functions and thus further improves the quality of generation patches with finer details.

Positional Encoding:

$$\gamma(p) = \left(\sin(2^0 \pi p), \cos(2^0 \pi p), \cdots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p) \right)$$

p: Input position

gamma(p): Output embedding

L: The largest power of frequency

Hyperparameters

Total Epoch: 16

Number of coarse samples: 64

Number of fine samples: 128

Batch size: 1024

Chunk size: 1024*32

Optimizer: Adam with learning rate = 5e-4, momentum = 0.9

Scheduler: StepLR with step = 20, gamma = 0.1

MLP size(W) = 256

Positional Embedding(L) = (10, 4) for position and direction.

We observe that the PSNR value increase by iterations, so we select the checkpoint from the last epoch in training.

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.

We modify the size of MLP and the size of Positional Embedding to see the impact.

Original:
MLP size (W) = 256, Positional embedding size(L) = (10, 4) for position and direction.
Reduce PE size:
MLP size (W) = 256, Positional embedding size(L) = (5, 2) for position and direction.
Reduce MLP size:
MLP size (W) = 128, Positional embedding size(L) = (10, 4) for position and direction.

Setting	PSNR	SSIM	LPIPS
Original	38.6783	0.9842	0.1694
Reudce PE size	38.4430	0.9815	0.1777
Reduce MLP size	38.9334	0.9850	0.1733

We can see that the smaller MLP size will slightly increase PSNR and LPIPS, which means that there exists a tradeoff between two evaluation indexes. We can also see that the smaller PE size will reduce PSNR and increase LPIPS, which means that positional encoding with larger frequency is the effective way to improve the quality of reconstructed image.

PSNR: PSNR stands for Peak signal-to-noise ratio, which represents the fidelity of image after reconstruction. It is calculated by dividing the largest value in pixel (ex:255) by the mean-square error between two images. The larger PSNR indicate that the reconstructed image has better quality.

SSIM: SSIM stands for structural similarity index, which evaluates the similarity between two images from the perspectives of luminance, contrast and structure. It is calculated by three functions. I(x, y) represents luminance, which is related to the mean values of two image. C(x, y) represents contrast, which is related to the standard deviations of two images. S(x, y) represents structure, which is related to the normalized value (mean value and standard deviation) of two images.

Three functions will be multiplied get SSIM index. The larger SSIM indicate that two images are more similar.

LPIPS: LPIPS stands for Learned Perceptual Image Patch Similarity, which represents the similarity between two images by the evaluation from Neural Networks. The Network is trained on large dataset and aim to distinguish two images from the perspective of human perception. The smaller LPIPS value indicate that two images are more similar.

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

We can see that the objects far from the camera has darker color (red), while the object close to the camera has lighter color (yellow).