CSCI 1430 Final Project Report: NeRF

NeRF: Jianxin Gu, Ning Li.
Brown University

Abstract

In this project, we reimplement NeRF, a method which can synthesize novel views of complex scenes with state-of-art result. NeRF takes a set of input images of a scene and renders the complete scene by interpolating between the scenes. NeRF mainly includes three steps: first, we match the camera rays through the scene to sample 3D points. Then, we use the points from previous step and corresponding view directions as input for MLP to get output set of RBG and densities. Finally, we use volume rendering to accumulate these color and densities into 2D images. Since it came out, NeRF revolutionized its field, spawning a plethora of subsequent works inspired by it. With this project, we hope to explore the details within NeRF and reimplement the NeRF.

1. Introduction

The problem that Neural Radiance Field attempts to solve is called Novel View Synthesis, which aims to generate novel views from one or more given source views from given input images. As described in original paper [3], NeRF consists in synthesizing a target image with an arbitrary target camera pose from given source images and their camera poses. Unlike other current top-performing techniques for view synthesis, NeRF generate novel views of complex 3D scenes, based on a partial set of 2D images. NeRF has dramatically increased the state of the art by starting a new line of research comparable to that started by GANs. NeRF shows impressive because it enables a wide range of applications from cinematography to virtual reality and more.

2. Related Work

NeRF (Neural Radiance Fields) is a method for rendering 3D scenes using a neural network, which was introduced in the paper "Neural Radiance Fields for Scene Rendering" by Matthew J. M. Schwartz, Ziyu Wang, and Kavita Bala [3]. The paper describes a method for representing 3D scenes using a multi-layer perceptron (MLP) that regresses density and color in a 3D volume. NeRF contributes mainly in two

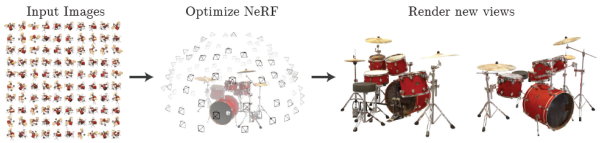

Figure 1. General Idea of NeRF.

Figure 2. Sample images from NeRF synthetic dataset

fields: Neural 3D shape represenation and View synthesis and image-based rendering. NeRF synthesize images by sampling 5D coordinates, feeding those locations into an MLP to produce color and volume density. And then we use volume rendering to composite these value into an image. Then we would use volume rendering techniques to accumulate these color and densities into 2D images.

We use NeRF synthetic dataset as our training dataset to train our neural radiance fields. This dataset contains synthetically rendered images. This dataset includes 8 scenes, with 100 training images, 100 validation images and 200 test images. Here is some examples from NeRF synthetic dataset shown in Figure 2. The original paper also provides real images dataset which is not included in our project.

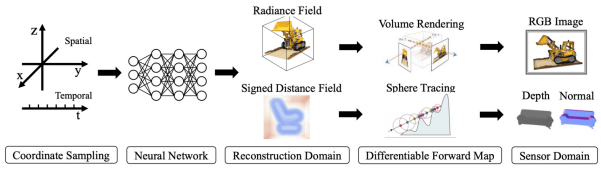

Figure 3. NeRF Pipeline.

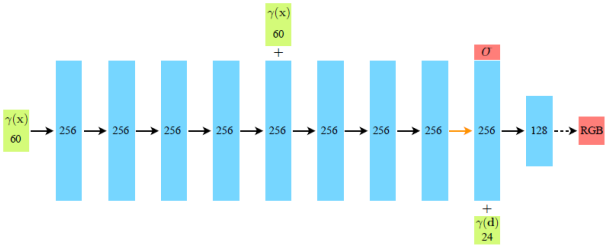

Figure 4. NeRF MLP Architecture.

3. Method

A NeRF model uses a multi-layer perceptron (MLP) to regress density and color in a 3D volume based on a set of posed images. This creates a representation of the 3D scene, which can be used to approximate a true volumetric rendering step through the use of a differentiable numerical integration method. The pipeline is show in Figure 3

3.1. Input and output

In NeRF, we represent a continuous scene as a 5D vector input which include a 3D location $x = (x, y, z)$ and 2D viewing direction (θ, ϕ), and whose output is an emitted color $c = (r, g, b)$ and volume density σ. In our implementation, the viewing direction is expressed as a 3D Cartesian unit vector d

Volume density models how opaque the object is. Since the objects are not completely opaque, we can see through them and the color we see is influenced by the color of the lens we are using. To determine the color observed in a given pixel, we have to analyze the characteristics of all points in "camera ray" that goes from the optical center of our camera and continues indefinitely along our viewing direction.

3.2. Model architecture

To map the inputs to output, NeRF uses simple Multi-layer Perceptron (MLP) $F_\Theta : (x, d) \rightarrow (c, \sigma)$. The MLP processes the input 3D coordinate x with 8 fully-connected layers, and output σ and a 256-dimensional feature vector. The feature vector is concatenated with cameras ray's view direction, then passed into adiitional fully-connected layer whose output is RGB color and viewing direction.

We use a fully-connected network in NeRF whose architecture is shown in Figure 4. All layers are standard fully-connected layers, with black arrows indicating ReLU activation, orange arrows indicating layers with no activation and "+" denotes vector concatenation.

3.3. Volume Rendering with Radiance Fields

From MLP model, we can get the colors and densities, so the next step is to use these to calculate the expected color at given pixel. Then we will use volume rendering techniques to accumulate those colors and densities into a 2D image. To map them back to the image, all we have to do is integrate these rays and acquire the color of pixel equation 1.

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt, \tag{1}$$

Given that t_n and t_f are the bound of the ray and $T(t)$ its transmittance. The transmittance is measure of how much the ray can penetrate the 3D space to a certain point and is defined as equation 2. This aforementioned technique is also referred as neural rendering or differentiable rendering in bibliography. This function thus defined is differentiable, a key property because in this case, we could use gradient descent to optimize the parameters.

$$T(t) = exp(-\int_{t_n}^{t_f} \sigma(r(s)))ds, \tag{2}$$

3.4. Optimizing a Neural Radiance Field

Positional encoding Using a high-frequency positional encoding could largely benefit our model. If we do not employ positional encoding in NeRF, high frequency geometries and textures are actually blurred. The original paper use the following encoding function 3, where p is a scalar. This function is applied separately to each of the three coordinate values in x as well as to those of the viewing direction. Positional encoding is not a new concept and a very similar version of this mapping is used in the popular Transformer architecture. However, in NeRF, the purpose is to map the input to higher dimensional space so that MLP can approximate higher frequency functions much more easily.

$$\gamma(p) = (sin(2^0\pi p), cos(2^0\pi p), ..., sin(2^{L-1}\pi p), cos(2^{L-1}\pi p)), \tag{3}$$

Hierarchical volume sampling In order to improve the efficiency of NeRF, the researchers use two networks: a coarse network and a fine network. The strategy consists of two steps: 1) stratified sampling is used to sample a set of Nc locations and evaluate the coarse network at these locations. 2) the output of the first step is used to produce a more informed sampling that allocates more samples to regions that are expected to contain visible content. So we can rewrite the equation like 4. This allows the allocation of more samples to regions that are expected to contribute to the final image, rather than uniformly sampling points along

Method	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

Figure 5. Comparison between different methods.

the camera ray and potentially wasting samples in free space or occluded regions.

$$\hat{C}_c(\mathbf{r}) = \Sigma_{i=1}^{N_c} w_i c_i, w_i = T_i(1 - exp(-\sigma_i \delta_i)), \quad (4)$$

4. Results

We train the NeRF on the synthetic blender dataset that is also used in the original NeRF paper. The MLP model we used has the exact same architecture as described in the paper. However, we noticed that there is a tiny difference between the positional encoding between the official implementation and the paper. We choose to use the version from the official code release.

4.1. Technical Discussion

Most training parameters are the same as the configurations provided in the official database release. We uses two models during training, one is called coarse model and the other is called fine model. The coarse model sample 64 points along each ray, while the fine model samples 128 points along each ray. The size of the training images are all 800×800. However, during testing, we sometimes half the resolution to speed up the process. We also change the number of sample rays to 4096 rather than 1024, hoping to speed up the training process. In addition, we increase the ray batch and point batch as much as possible to fully utilize the GPU power.

The main GPU resource we use is the CS department Grid Engine, which has multiple NVIDIA GTX 1080/1080ti GPUs available. It roughly takes 30 minutes to train 2000 epochs, which can give us preliminary results so that we can quickly verify our implementation. For generating test images and videos, we eventually choose to train for 40000 epochs to get our final results, which takes about 5 to 6 hours.

Given the limitation of time and group members, we do not have the time to try and run some of the comparison tests described in the NeRF paper. As the paper stated, also shown in Figure 5, NeRF almost consistently out-performs the other methods on both synthetic and real images. Moreover, given that it has been widely acknowledged that using view directions and using positional or even more sophisticated encoding significantly help with model training, we did not attempt to run any our experiment without using view direction or positional encoding to compare the results.

Some of the results are shown in Figure 6. The corresponding epochs from left to right is 10000, 20000, 30000, and 40000. We can see that the quality of the images from the first columns are significantly worse than the others. For example, the plate in the hotdog scene is less smooth, the details of the bucket of the truck are missing, and the surface of the spheres in the scene at bottom is blurred. As the training epochs increases, the results get much better, we found that the 30000 to 40000 epochs is a good point for us to stop training as it can provides visually qualitative results while still in a reasonable time. However, it's worth noting that the original NeRF paper claims that it actually takes 100k to 300k for a model to converge on a single scene.

We also have multiple videos rendered with a camera orbiting above the scene. The three videos corresponding to the scenes shown in Figure 6 are included in the submission on Gradescope. The Lego scene is rendered at resolution 800×800, while the hotdog and the materials scenes are rendered with resolution 400×400.

4.2. Social and Impacts

NeRF has inspire a ton of subsequent works and many of them have promising usage in the future. For example, scene relighting [5] can probably help CG artists to have a quick preview of the scene when performing editing of the lights and materials in the scene. In addition, more recent work [4] has proven the possibility of having almost realtime NeRF rendering at 1080p resolution using multi-resolution hash encoding with fully-fused CUDA kernels, which can potentially be used for realtime preview for scene editing.

There are also works [2] that expand the NeRF to become aware of temporal information. Now, instead of only being able to render a static scene with new view directions, the model can render a dynamic scene with different view angles which can then be composed into a video clip. NeRF has also been proved useful in the field of generalization. StyleNeRF[1] is capable to provide photo-realistic rendering results with high multi-view consistency, which can be potentially helpful for many kinds of computer graphics works.

However, being able to render something that does not originally exist can also introduce problems. If not used properly, NeRF and its variants can be easily used for generating fake images and videos, which can then be used to spread fake information to fool and trick the public.

In general, Neural Radiance Field can be considered as a complex representation of a 3D scene. Currently, a common approach to represent and exchange a 3D scene is using OBJ files and then render it with path tracing algorithms. With NeRF, we've unlocked a new possibility for scene representation. It's possible that in the future, NeRF will become the carrier when exchanging for 3D scenes and we can render the scene using volumetric rendering techniques.

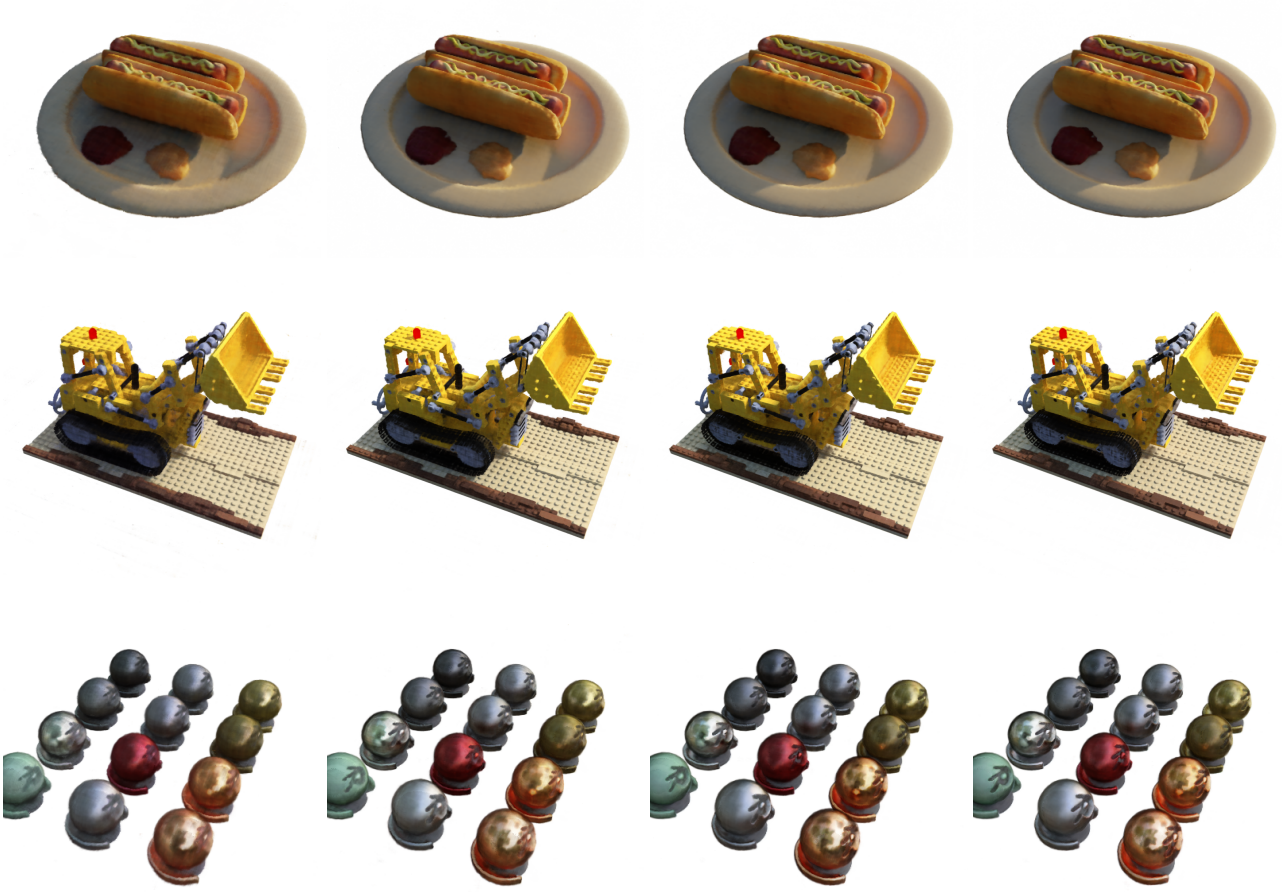

Figure 6. *Epochs from left to right:* 10000, 20000, 30000, 40000

5. Conclusion

Neural Radiance Field introduces a new approach for novel view synthesis and has become the backbone of countless subsequent works since it came out. It also has the potential to become a new way of becoming a standard way of representing and rendering 3D scene in the future.

In this project, we explored the details of NeRF and attempted to reimplement the NeRF. We reused the code for loading the dataset and implemented the rest core functionalities by ourselves, including the MLP model, the positional encoding, the volumetric rendering and the training loop for generating rays and points in the 3D space. Here is the link of our repository for reference.

We trained the Neural Radiance Field on multiple scenes in the blender synthetic datasets at its original resolution 800×800. When rendering test images and videos, we used a mixed resolution of 800×800 and 400×400 to speed up the process.

In general, we find that the volumetric rendering algorithm is the hardest part when reimplementing the NeRF model, it requires us to have a very thorough understanding of the concept behind and it also contains many detailed numeric operations. And missing a single one can mess up the whole output.

References

[1] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *CoRR*, abs/2110.08985, 2021. 3

[2] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. *CoRR*, abs/2103.02597, 2021. 3

[3] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020. 1

[4] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 3

[5] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. *CoRR*, abs/2012.03927, 2020. 3

Appendix

Team contributions

Please describe in one paragraph (3 - 4 sentences) per team member what each of you contributed to the project.

Person 1 Read the original NeRF paper and the code release. Implemented the MLP model for NeRF. Implemented the volumetric rendering and integrate it into the training loop. Ran multiple experiments with on different scenes with different parameters. Wrote the final project report.

Ning Li Read the original NeRF paper and explore the NeRF synthetic dataset. Completed the Positional Encoding to map input coordinates to high dimensional space before passing into MLP. Ran multiple experiments with on different scenes with different parameters. Finished final poster and final report.