RBE 549 Project 2: Buildings Built in Minutes - NeRF

Amrit Krishna Dayanand, Venkata Sai Krishna Bodda
MS Robotics Engineering
WPI
Email: adayanand@wpi.edu, vbodda@wpi.edu

I. Introduction

Neural radiance fields (NeRF) is a method to represent a scene using a fully-connected, non-convolutional deep neural network. Its 5-d input, **x**, is a spatial location and camera viewing direction, **d**, which are queried along a ray in the camera's viewing direction. Its output is the volume density and view-dependent radiance emitted. Classical, differentiable volume rendering is used to project the output color and density and subsequently train the network. NeRF can be used to generate novel views and can capture specular phenomena in a photorealistic manner. It outperformed state-of-the-art methods when it was published, and inspired various papers, including Nvidia's Instant-NeRFs.

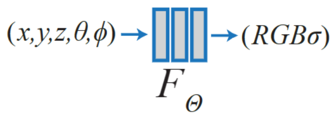


Fig. 1. High-level representation of NeRF pipeline

II. Volume Rendering

The volume density $\sigma(\mathbf{x})$ is interpreted as the differential probability of a ray terminating at an infinitesimal particle at **x**. In simple terms it represents the opacity of a point in space, which ranges from $[0., 1.]$, capturing translucence. The expected color, $C(\mathbf{r})$ of ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ with near and far bounds t_n and t_f is:

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

where, $T(t)$ is the transmittance defined as,

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

This continuous integral is estimated using quadrature and stratified sampling.

A. Hierarchical sampling

To reduce the number of computations with low contributions to the transmittance, the paper implements stratified sampling and importance-based sampling.

Stratified sampling is a method where the near and far bounds are divided into N-evenly spaced bins that are uniformly sampled. This initial sampling is coarse and includes empty space. By using the output of volume rendering to sample finely at regions more likely to contain salient parts of the scene.

B. Quadrature rule

Over the course of optimization, the discrete quadrature process can be used to represent a continuous scene. The quadrature rule modifies the volume rendering equation as follows:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i \tag{3}$$

where T_i is given by,

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \tag{4}$$

and where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples.

III. Optimization of a Neural Radiance Field

A. Positional encoding

The authors noted that the network was good at capturing low frequency features of the scene, but not high frequency ones, like specular phenomena. To improve the quality of the network in capturing these features, positional encoding is used. Positional encoding sends a low frequency signal to an L-dimensional high-frequency signal, where the position, and direction is concatenated with its high dimensional encoding. The dimensionality is $L = 10$ for position, and $L = 4$ for viewing direction.

The encoding is given by:

$$\gamma(p) = (sin(2^0\pi p), cos(2^0\pi p), ..., sin(2^{L-1}\pi p), cos(2^{L-1}\pi p)) \quad (5)$$

- Show results of novel views for both datasets (take frames from the gif you created). - Talk about any issues you faced and how you solved them. - Report the average PSNR and SSIM on the test set (You can use any third party code for these calculations, a sample can be found here.) - Compare results with and without positional encoding (for atleast any one of the datasets). Show a few good and bad examples of your model output with comparison with the ground truth images in test set

B. Network

The fully-connected multilayer perceptron (MLP) network has 8-layers. The network takes as input a positionally encoded position, and viewing direction. It outputs the 1-d opacity and 3-d RGB color. Following the DeepSDF architecture, the network includes a skip connection.

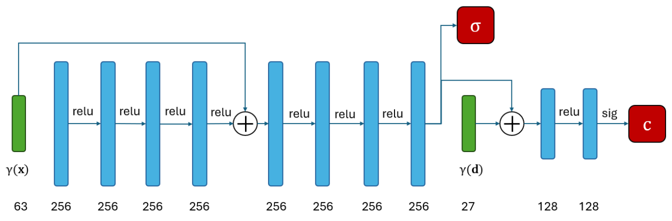


Fig. 2. Multilayer perceptron network for NeRF

We optimize using the following mean squared error loss function:

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2 \quad (6)$$

where $\mathcal{R}$ is the set of rays in a batch, $C(\mathbf{r})$ is the ground truth, and $\hat{C}(\mathbf{r})$ is the predicted output.

C. Hyperparameters and Data Engineering

To reduce the computational load on our GPUs we downsampled the input images by a factor of 8, reducing the size of images to $(100, 100)$. This resulted in poorer resolution of the continuous scene representation, yet the image and novel views were distinguishable. This likely contributes to our results of average PSNR and SSIM.

We used an Adam Optimizer with default values and a dynamically decaying learning rate from $5 * 10^{-4}$ to $5 * 10^{-5}$ over 10,000 epochs.

IV. Results

Our network was trained on the Lego and Ship datasets with positional encoding. We present the results of our test set for both datasets and the impact of not using positional encoding (PE) for our inputs. We use peak signal-to-noise ratio (PSNR)

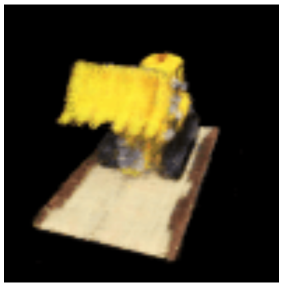


Fig. 3. Example of a good network output. Specular phenomena on the tracks of the Lego model are captured, albeit with some blur.

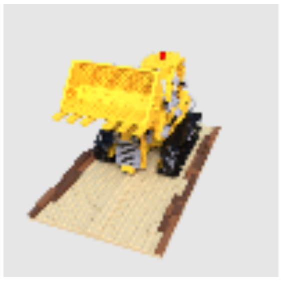


Fig. 4. Ground truth

and structural similarity (SSIM) to evaluate the quality of our network. The table summarizes the average PSNR and SSIM for both image datasets (higher is better).

Test Set	Loss	PSNR	SSIM
Lego	0.000235947	26.2719	0.949018
Lego (no PE)	0.00426547	23.7003	0.821065
Ship	0.00324482	24.8881	0.871624

We observed a 9.8% difference in PSNR, and 13.5% difference in SSIM between the render with positional encoding and without. Despite the lower resolution of the training images, the impact of positional encoding was significant.

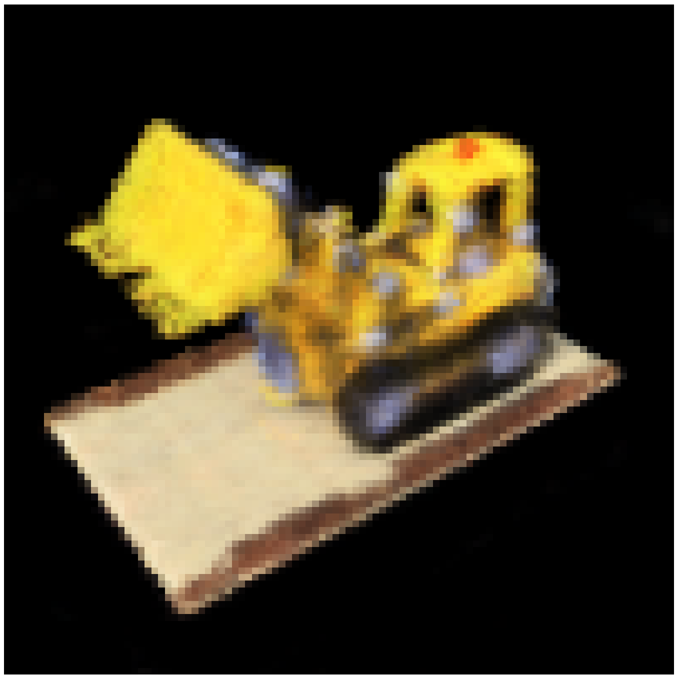


Fig. 5. Novel view rendered using NeRF of the Lego dataset

The main challenge we faced was training speed. This was greatly improved by reducing the resolution of the dataset and moving training to a GPU cluster. This reduced the number of model parameters, at the cost of accuracy.

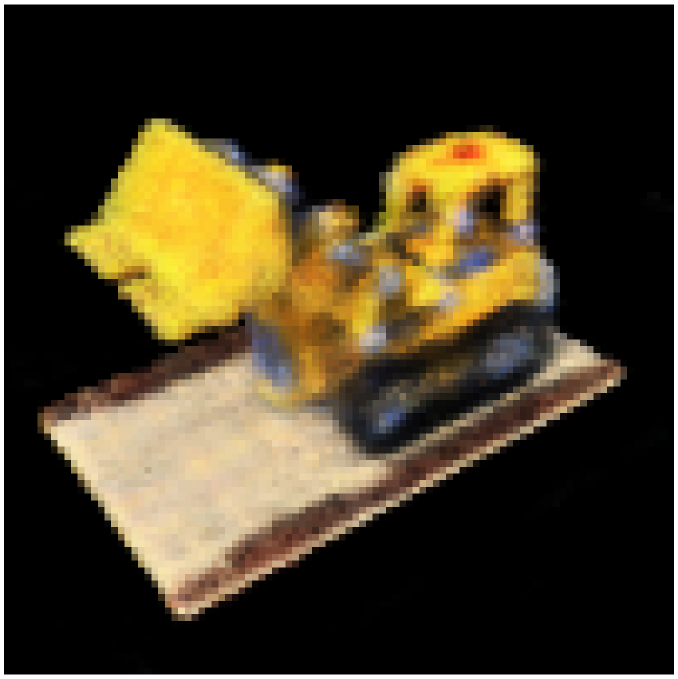


Fig. 6. Novel view rendered without implementing positional encoding

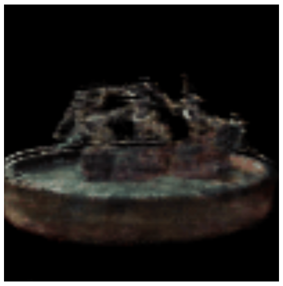


Fig. 8. Example of a poor network output. Specular information is not captured accurately, and there are more artifacts on the texture of the basin and ship's masts.

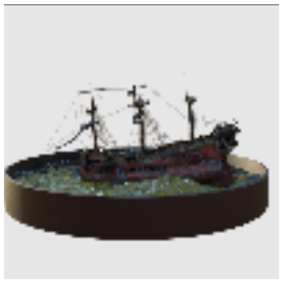
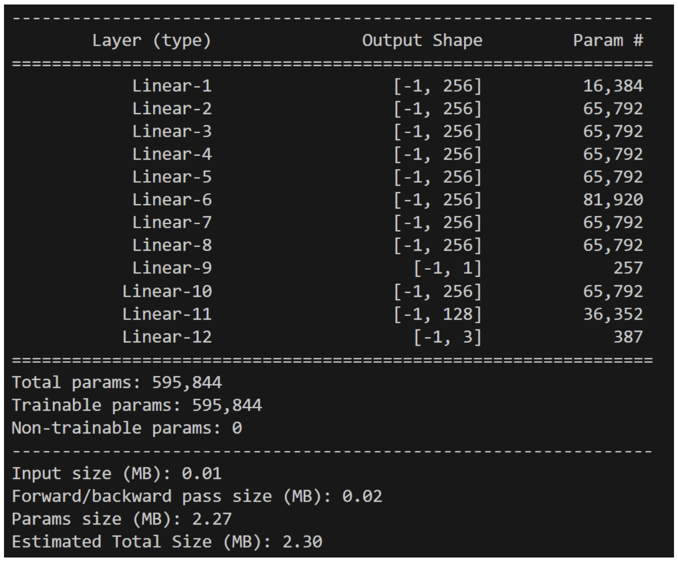


Fig. 9. Ground truth



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]          16,384
            Linear-2                  [-1, 256]          65,792
            Linear-3                  [-1, 256]          65,792
            Linear-4                  [-1, 256]          65,792
            Linear-5                  [-1, 256]          65,792
            Linear-6                  [-1, 256]          81,920
            Linear-7                  [-1, 256]          65,792
            Linear-8                  [-1, 256]          65,792
            Linear-9                    [-1, 1]             257
           Linear-10                  [-1, 256]          65,792
           Linear-11                  [-1, 128]          36,352
           Linear-12                    [-1, 3]             387
================================================================
Total params: 595,844
Trainable params: 595,844
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.02
Params size (MB): 2.27
Estimated Total Size (MB): 2.30
```

Fig. 7. Summary of the network parameters

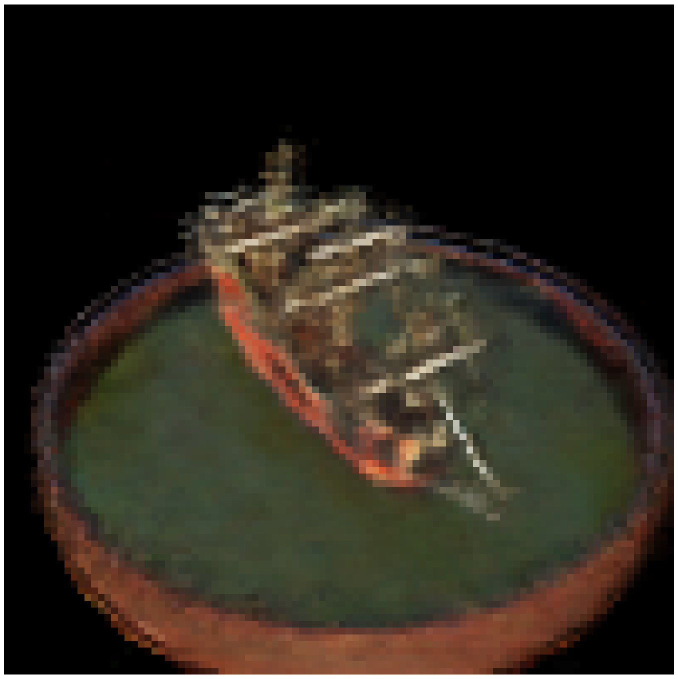


Fig. 10. Novel view rendered using NeRF on the Ship dataset