CSE274 Project Report
Neural Radiance Fields

Teja Sai Dhondu

1 Introduction

Neural Radiance Fields (NeRF) tackles the problem of novel view interpolation. Given a set of sparse views of a scene, it attempts to generate new views of that scene. Unlike previous deep learning methods for novel view interpolation, which perform transformations on 2D images to generate new views, NeRF falls in the new class of techniques that learn a 3D volumetric scene representation which is used to generate new views. This adds a level of geometric accuracy in novel view interpolation not seen before. However, the two major downsides of such methods have been the need for ground truth 3D data and the large size of the volumetric scene representation. NeRF addresses both problems and is able to have a compact volumetric scene representation while being trained with end-to-end supervision without the need for ground truth 3D data. This comes down to its two key ideas: a neural network as a volumetric scene representation, and volume rendering. In this project, I attempted to implement the NeRF paper and achieve comparable results on the dataset used in the paper.

2 Methods

The pipeline for generating a view from NeRF is as follows: Shoot rays for each pixel out of the camera into the scene and sample points along the rays, feed the points to the 3D volumetric scene representation, and perform volume rendering to composite the values of the samples along each ray to obtain the color of each ray.

2.1 Scene Representation

NeRF uses a neural network as its 3D volumetric scene representation. As opposed to similar novel view interpolation methods that used N-Dimensional arrays as scene representations, NeRF has a vastly more compact scene representation. Where N-Dimensional arrays have memory requirements of 1-10 GB, neural networks containing only fully connected layers have memory requirements of only 1-10 MB. The neural network acts as a 5D continuous function that represents the radiance field of the scene:

$$F : (x, y, z, \theta, \phi) \rightarrow (r, g, b, \sigma)$$

Where x, y, z are the 3D positional coordinates of a sample point in the scene; θ, ϕ are the 2D directional coordinates of the sample point from the center of the camera; r, g, b is the RGB color of the radiance at the sample point; σ is the volume density of the sample point

2.2 Volume Rendering

Volume rendering is used to perform a continuous variant of alpha compositing to composite the RGB values of the samples along each ray to obtain the color of that ray. Volume rendering is fully differentiable, which enables NeRF to be trained with end-to-end supervision without the need for 3D ground truth data. The formula for volume rendering is:

$$C = \sum_{i=1}^{N} T_i \alpha_i c_i \tag{1}$$

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{2}$$

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i} \tag{3}$$

Where C is the RGB color of the ray, c_i is the RGB color for sample i, T_i is the transmittance of the sample i, α_i is the alpha value of sample i, σ_i is the volume density of sample i, and δt_i is the distance between sample i and the previous sample along the ray. The transmittance is effectively how much light is blocked earlier along the ray before reaching the sample, and α is the continuous variant of the alpha value from standard alpha compositing since it takes into account differing distances between consecutive samples. The alpha value is effectively how much light is contributed by, or the opacity of, the sample.

2.3 Hierarchical Rendering

NeRF uses a 2-pass hierarchical rendering method in order to sample points along rays efficiently and effectively. The first pass is a coarse pass and the second pass is a fine pass. During the first pass, points are uniformally sampled along each ray. These points are then used to generated a coarse view. The weights of the samples calculated during volume rendering, which is the product of the transmittance and alpha value, are used as a probability distribution function from which the fine points during the second pass are sampled. The reason why the weights are used as the probability distribution function is because they allow points to be sampled in areas that contributed more towards the color of the ray during the first pass. Both the coarse and fine samples are fed to the network during the second pass in order to generate the final output image. Additionally, separate networks are trained using the first pass and second pass samples.

2.4 Optimization

Since the entire pipeline is end-to-end differentiable, NeRF can be trained using a mean squared error rendering loss:

$$Loss_c = \sum_i ||render^{(i)}(F_c) - I_{gt}^{(i)}||^2 \tag{4}$$

$$Loss_f = \sum_i ||render^{(i)}(F_f) - I_{gt}^{(i)}||^2 \tag{5}$$

$$Loss = Loss_c + Loss_f \tag{6}$$

Where F_c is the network trained with the first pass samples and F_f is the network trained with the second pass samples, $render(F_c)$ is the rendered image using volume rendering on the first pass network with just the coarse samples, $render(F_f)$ is the final rendered image using volume rendering on the second pass network with the coarse and fine samples, and I_{gt} is the ground truth image for that particular view. The loss is minimized with respect to the parameters of the first and second pass networks using gradient descent.

2.5 Input Encoding

In actuality, the input to the scene representation neural network is greater than 5D since an encoding is applied on the inputs to shift them to a higher dimensional space. This is done because of the difficulty for neural networks in general to learn high frequency functions from the low dimensional data. This relates to neural tangent kernel theory which states that neural networks can be approximated as neural tangent kernels. Neural tangent kernels can be transformed to learn high frequency functions. A kernel can be transformed by transforming or encoding its input data. One of the contributions of the NeRF paper was a generic encoding to input data that enables neural networks to learn high frequency functions in general and not just for novel view synthesis applications. The encoding is a fourier encoding:

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

Where p is each of the coordinates in the 5D input to the scene representation neural network, and L is the parameter that decides the length of the encoding.

3 Implementation Details

3.1 Architecture

The architecture is given in figure 1. The network is made up of two sub-networks. The first sub-network takes the position coordinates as input and outputs the volume density. The second sub-network takes as input the viewing direction coordinates, as well as a 256D vector output from the previous sub-network, and outputs the RGB color. This means that the volume density is only dependent on the position coordinates, thus promoting multi-view consistency, and that the color is dependent on viewing direction as well, which enables view dependent effects like specular highlights. The neural network was implemented using PyTorch. My implementation code is available at https://github.com/TD87/CSE-274-NeRF.

3.2 Important Parameters

- Number of Coarse Samples: 64
- Number of Fine Samples: 128
- Position encoding, L = 12
- Viewing direction encoding, L = 4
- Batch size: 5000 rays
- Number of epochs: 1000
- Optimizer: Adams
- Learning Rate = 5e-4
- Hidden Layer Channels = 256

4 Results

Results were generated for the synthetic Lego dataset. The dataset consisted of 84 images used for training and 22 used for testing. The images had dimensions of 100x100. Video results and all the image results are available at https://drive.google.com/drive/folders/

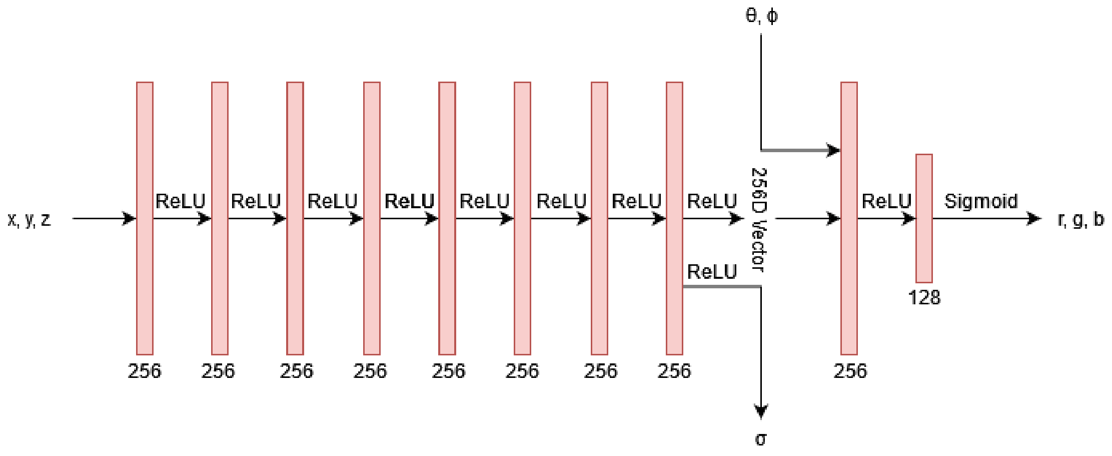


Figure 1: Neural Network Architecture

12gCgdVQk08qVQV4J9ISTx84xGNGgQPD7?usp=sharing. Losses are in figure 2. Select results are at the end of the report. Training was performed on a GTX 1080 Ti. Training time was 20 hours.

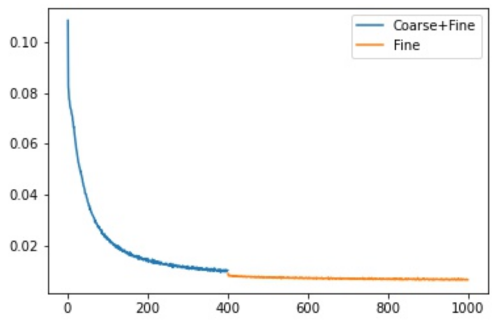


Figure 2: Losses

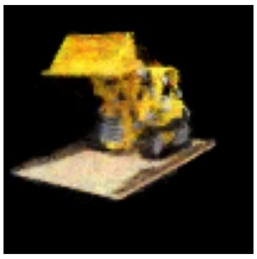
5 Discussion

As can be seen from the results, for the views that the network was able to learn well, the network was able to learn accurate geometric detail and color detail. Additionally, as can be seen from the video results, there is multi-view consistency, for the views that the model was able to learn well. However, there is still some noise and the images are of low resolution, which is likely due to the fact that the input images were only of 100x100 dimensions. For the views that the model is not able to learn well, it outputs a highly noisy, low frequency and geometrically inaccurate image. Since all the views that the model struggles with fall within the testing set, it is reasonable to assume that the model has overfitted and hence struggles with views that are very different from the views it trained on. Thus, it is able to do reasonably well on test views similar to the train views, but struggles with test views that are very different. This is most probably due to one key difference between my implementation and the paper. Instead of using separate network for the two hierarchical rendering passes, I used the same network for both passes. I made this decision due to the difficulties I faced having the models converge when I used separate networks. However, because of this, the same network is learning to perform two roles which can interfere with each other and should be learnt by separate networks: providing weights for the probability distribution to sample from, and generating the final output image from the second pass samples. My attempt to make up for this shortcoming can be seen in the loss graph. For the first 400 epochs, the losses were optimized over both passes, but for the last 600 epochs, losses were optimized over only the second pass. I did this in order to make the network transition from performing the first role to performing the second role.

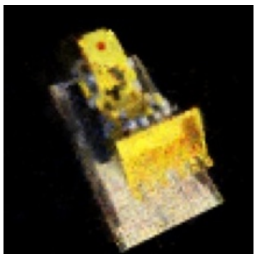
Conclusions

NeRF is a revolutionary method for novel view synthesis as it is able to generate new views with highly accurate geometric detail by learning a 3D volumetric scene representation, which is both compact and end-to-end differentiable, allowing purely 2D supervision. It is state of the art for not only novel view synthesis but also for volumetric scene representations. However, it is still not ideal due to its very high training and inference times as well as its high memory requirements during training, as the number of queries to the network can explode into the hundreds of millions for high resolution images.

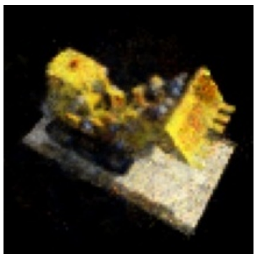
(a) Ground Truth

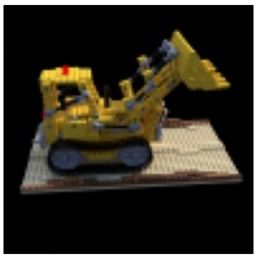

(b) Train View

(c) Ground Truth


(d) Train View

(a) Ground Truth


(b) Test View


(c) Ground Truth

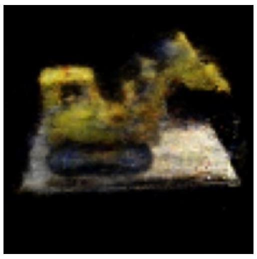
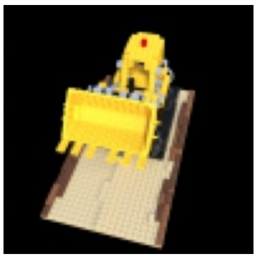
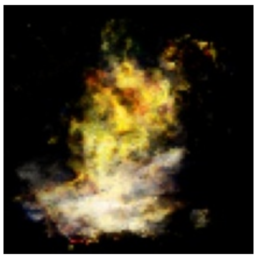
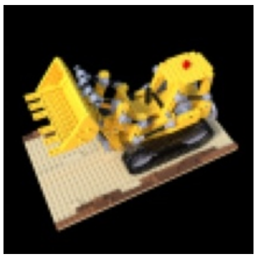

(d) Test View

Figure 3: Good Results


(a) Ground Truth


(b) Train View


(c) Ground Truth

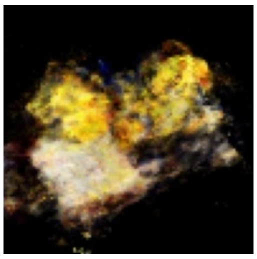

(d) Train View

Figure 4: Bad Results