NeRFs for All - A probe into the accessibility of NeRFs for Educational Use

Anmol Mansingh
anmolmsg@umich.edu

Keagan Pinto
keaganjp@umich.edu

Raghav Mishra
imraghav@umich.edu

Abstract

Neural Radiance Fields (NeRFs) are increasingly being adopted for personal-use applications, such as creating views from amateur videos and reconstructing small artifacts in 3D. However, the complexity and inaccessibility of leading implementations like NVIDIA's InstantNGP and Luma AI at the granular level pose challenges for novice programmers. We take an easily-understandable NeRF architecture, written entirely in Python, and trained it on a synthetic dataset from the original NeRF paper, and then adapt it for more challenging data such as the Middlebury dino dataset. This would further our aims of using NeRF architectures, to positively impact and enhance the quality of education by increasing the accessibility of such 3D models to the wider public. Our findings show promising convergence and performance on synthetic data (max val PSNR: 27.3 dB), but a decline in effectiveness with the Middlebury dataset (max val PSNR: 23.1 dB). We then highlight our learnings and suggest ways to overcome the challenges we faced. We conclude with potential future work integrating InstantNGP's COLMAP for enhanced camera parameter generation in our model. This study contributes to making NeRF technology more accessible to a broader user base. Github

1 Executive Overview

The education sector has seen a dire need for boosting student engagemnent through interactive teaching tools. Recent work (1) argues that VR-augmented education helps students retain knowledge better. Hence, this sector has seen widespread adoption of VR equipment with the aim of providing students access for tools for 3D exploration. However, there has been a dearth of generation of such content, especially on an amateur level by the educators themselves. 3D cameras are prohibitively expensive (2), and often require extensive setup in generating reconstructions of ancient ruins/historic sites, for example.

Over the past three years, Neural Radiance Fields (NeRFs) (3) have emerged as a significant advancement in 3D rendering and processing. In the landmark paper, they were introduced with the primary aim of generating novel intermediate views from a given input view. NeRFs accomplished this task by representing a scene and approximating it by a continuous volumetric function underneath.

This is where other software approaches for 3D reconstruction come in, and NeRFs have emerged as a robust way of doing so. With the introduction of the InstantNGP offering (4), NVIDIA has been at the forefront of making 3D reconstruction from NeRFs incredibly easy. Also, mobile applications like Luma AI are intended for the same purpose, albeit on a However, we observe that the workings of such powerful, plug-and-play tools, by design, trade efficiency and speed for understandability and customization. Hence, there is a potential barrier to entry for wider adoption.

As we discuss throughout this report, we started with a simple, Pythonic NeRF implementation that is more understandable and powerful enough to work on well-processed data, like the synthetic Lego dataset given in the original paper. We then adapt it to less processed data (Middlebury dataset),

through a series of data processing and camera parameter transformations, and note the results and deficiencies with doing so.

2 Background and Impact

Since the inception of NeRFs (3), they have revolutionized the field of 3D rendering. NeRFs have been primarily used in computer vision and graphics for photorealistic scene reconstruction. Their ability to synthesize novel views of complex scenes from a sparse set of images has set a new standard in the field (5).

The application of NeRFs in educational settings is relatively new. (1) explored virtual reality (VR) in education, highlighting the need for immersive and interactive content to enhance learning experiences. However, they did not specifically address the use of NeRFs.

The challenges of content generation for educational purposes have been discussed extensively. (6) noted the high costs and technical expertise required for 3D content creation, particularly for historical and archaeological sites. This gap in accessible content creation tools has been a barrier to the widespread adoption of VR in classrooms.

On the software front, NVIDIA's InstantNGP has been a significant step forward in simplifying 3D reconstruction (4). However, its ease of use often comes at the expense of customizability. Similarly, mobile applications like Luma AI offer user-friendly interfaces but lack the depth required for educational purposes. In contrast, nerfstudio (7), a project by Berkeley AI researchers, has been working towards a more modular and accessible approach to NeRFs, which could potentially lower the barrier to entry for educational content creators.

Our research builds upon these foundations, aiming to strike a balance between simplicity, understandability, and functionality in NeRF implementation. By adopting a simple NeRF architecture that can be run on Colab's free tier, we address the specific needs of individual educators that previous works have identified but not fully resolved. Thus, our approach is geared towards enabling educators to create their 3D content with minimal technical overhead, thus democratizing access to high-quality educational resources.

3 Methodology

3.1 Datasets and Data Pre-processing

We used a synthetic dataset called the Lego Dataset (From the NeRF paper) consisting of train, validation and test views. for our baseline model. To extend our models, we selected the Middlebury Dino dataset for training and Middlebury DinoRing Dataset for testing. The testing dataset consisted of views collected in a hemispherical fashion. This dataset consisted of 100 views for training, 10 views for validation, and 60 views for testing which were randomly sampled.
The Data we used consisted of:

- RGB images (**.png** files)
- Camera Parameters (**.txt** files and **.json** files)

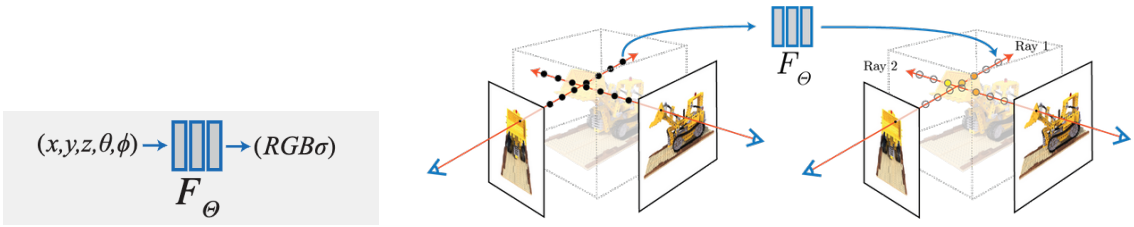
For the data pre-processing, since we aimed to train the NeRFs with compute restrictions we resized the images to 200x200. Originally, the Lego Dataset had a size of 800x800. So we scaled the extrinsic matrix using the scaling matrix as shown below. Where, s_x and s_y is the ratio of the new dimensions and the old dimensions. Similarly, we pre-processed the original Middlebury data from 640x480 to 200x200 and random sampling to generate similar sizes as the Lego dataset used.

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{3\times3} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$$

Above is the representation of the transforms applied on the image parameters.

3.2 Model Architecture

The NeRF algorithm represents a scene using a fully-connected (non-convolutional) deep network, whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)) and whose output is the volume density and view-dependent emitted radiance at that spatial location. Besides this, the input is positionally encoded with high-frequency representations of the input coordinates. This enhances the representations that the NeRF learns. NeRF uses a differentiable rendering technique to simulate the process of camera rays passing through a scene. This allows the use of gradient descent to optimize the neural network's parameters.



(a) Overall NeRF architecture

(b) NeRF View Synthesis

Figure 1: NeRF Architecture

The main components of the NeRF architecture are:

- **Input**:The input to the NeRF model is a set of 5D coordinates (x,y,z,θ,ϕ)[Fig.1a], which are the spatial locations of the scene.
- **Positional Encoding**: To allow the network to better represent high-frequency functions, NeRF applies a positional encoding to the input coordinates. This encoding maps each input coordinate to a higher-dimensional space using a set of sinusoidal functions.
- **Volume Rendering**: To render a 2D image from this representation, NeRF uses classical volume rendering techniques. It casts rays from the camera through each pixel of the image plane and into the scene. As the ray traverses the scene, it samples points along its path. The color and density at each point are determined by the neural network, and the final color of the pixel is computed by accumulating these values along the ray, taking into account the transmittance.
- **Multi-Layer Perceptron**: This fully-connected Neural Network consists of 8 layers with fixed dimensionality of 256.
- **Scene Representation**: After training, the neural network effectively becomes the scene representation. It encodes both the geometry (via volume density) and the appearance (via color) of the scene as a function of position and direction. [Fig.1b]

During training, NeRF optimizes the network weights using a collection of 2D images of a scene from known camera positions. It uses stochastic gradient descent to minimize the error between the rendered images and the actual images. After training, NeRF can synthesize novel views of the scene by rendering images from new camera positions. It does this by casting rays through the volume, querying the neural network, and applying the volume rendering equation.

3.3 Implementation Details

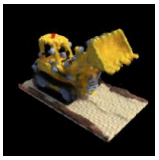
For training on the synthetic Lego dataset, we trained the model with a batch size of 10000 rays for 3000 iterations with a learning rate of *1e-3* using an Adam optimizer. This required relatively lower number of training steps as the data was synthetically generated and meant for NeRFs.

The Middlebury dataset training was far more challenging. We trained on a batch size of 20000 rays for 10000 iterations with a learning rate of *1e-5*. Although we trained it for a substantial amount of time the validation PSNR plateaued at around **23.1 dB** compared to the **27.3 dB** on the Lego dataset. Due to computational restrictions, we were unable to train the Middlebury datasets with larger batch size and for a longer duration. However, we were able to obtain noisy novel views with our training shown in the results section below.

4 Results

Here are the novel views generated by the model:

(a) Processed lego



(b) Novel View 1

(c) Novel View 2

Figure 2: Synthetic lego dataset

(a) Processed dino

(b) Novel View 1

(c) Novel View 2

Figure 3: Middlebury Dino dataset

Table 1: Maximum PSNR attained (on train and val datasets)

Dataset	val PSNR (dB)	Iterations	dataset size
Lego	27.3	3000	12.6 MB
Middlebury Dino	23.1	8000	98 MB

4.1 Conclusion and Future Work

We gained a lot of insights during the data processing stage. Initially, we found that the vanilla architecture expected a .npz file consisting of preprocessed 200x200 images, neatly packaged with their camera-to-world transformation matrix parameters. We wrote additional helper code to perform this preprocessing to fit the After we refactored the architecture to handle bigger images, we began to see CUDA out of memory errors, indicating that the Colab compute limits were reached. Next, we decided to resize the Middlebury Dino images to 200x200 (as they were originally 640x480). However, this resulted in very poor training performance. Upon further inspection, we realized that the camera parameters were not scaled accordingly, which we recalled as a key learning in the course. Then, we took care to scale them according to the resizing operation.

As this implementation requires the camera parameters (which we were given for both datasets), one extension we wish the perform as future work is to integrate InstantNGP's COLMAP utility to fetch the camera parameters for any video/set of images, which will be a significant usability improvement from the perspective of our users (educators).

References

[1] S. Kavanagh, A. Luxton-Reilly, B. Wuensche, and B. Plimmer, "A systematic review of virtual reality in education," *Themes in Science and Technology Education*, vol. 10, no. 2, pp. 85–119, 2017. [Online]. Available: https://www.learntechlib.org/p/182115/

[2] C. Nock, O. Taugourdeau, S. Delagrange, and C. Messier, "Assessing the potential of low-cost 3d cameras for the rapid measurement of plant woody structure," *Sensors*, vol. 13, no. 12, pp. 16 216–16 233, 2013. [Online]. Available: https://www.mdpi.com/1424-8220/13/12/16216

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[4] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[5] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 6498–6508.

[6] J. Hardesty, J. Johnson, J. Wittenberg, N. Hall, M. Cook, Z. Lischer-Katz, Z. Xie, and R. H. McDonald, "3d data repository features, best practices, and implications for preservation models: findings from a national forum," 2020.

[7] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. Mcallister, J. Kerr, and A. Kanazawa, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3588432.3591516