Exploring NeRF: Final Project Report

Yi-ting (Taryn) Chiang, Lucas Martinez, and Oona Wood

Abstract: This project experiments with simple methods to improve the results generated by the Neural Radiance Fields (NeRF) Model architecture, focusing on some architectural modifications, diverse loss functions, and image pre-processing techniques. Also, we evaluate the impact of these adjustments on the accuracy, using SSIM, PSNR, and LPIPS indicators in order to evaluate the similarity between our outputs and the original target image.

1. Introduction

The purpose of this project is to expand upon the research around NeRFs. In particular, this project chose to build on the research in what is largely considered the ”original” NeRF paper ”NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis” [4]. In this effort, we explored potential improvements through model modifications and image preprocessing. Our goals include improving the quality of 3D scene reconstructions and understanding how different loss functions, model architectures, and contrast adjustments impact the results.

1.1. Datasets

Due to hardware limitations, capturing and creating a new dataset was not feasible. However, utilizing the synthetic dataset provided by the original NeRF authors instead offered a controlled environment to assess our modifications against a known baseline, ensuring that improvements can be attributed to our changes rather than dataset variability.

1.2. Literature Review

A targeted literature review was conducted, focusing on papers that cite the original NeRF work. From this review, two primary avenues for enhancing NeRF emerged: model modifications and image preprocessing [1, 2, 3, 5]. Each of these areas presented several opportunities to potentially improve the quality and efficiency of 3D scene reconstruction.

2. Related Work: NeRF

NeRF representation is an innovative approach to 3D scene reconstruction that uses deep learning in a unique way to synthesize novel views from a set of 2D images. At its core, NeRF represents a scene as a continuous 3D function, parameterized by a neural network, which learns to predict the color and density at any point in the 3D space given an input view direction and spatial coordinates. In essence, the neural network consists of 8 fully-connected layers (using ReLU activations and 256 channels per layer), and the MSE as its loss function. Here’s how NeRF works:

1. **Input Data**: NeRF takes as input multiple images of a scene captured from different viewpoints, along with the camera parameters associated with each image. In other words, the neural network that takes a 5D input (spatial location (x, y, z) and viewing direction (θ, φ)).

2. **Training**: During training, NeRF uses the input images to learn a mapping function that predicts the color and density of any point in the 3D space, given the 3D coordinates and the camera view direction.

3. **Rendering**: After training, NeRF can synthesize new, unseen views of the scene by querying this mapping function and rendering the 3D scene from any virtual camera perspective. It does this by sampling rays through the scene and using volume rendering techniques to accumulate the predicted color and density along each ray.

NeRF’s ability to accurately reconstruct a continuous volumetric representation allows it to generate highly detailed and realistic images, even from limited input data. This has broad applications in computer graphics, virtual reality, and other fields that require realistic 3D scene rendering.

3. Our Approach: NeRF Research and Experimentation

3.1. Model Modification

3.1.1. Different Loss Functions:

The original NeRF model works with the MSE loss function. To assess the impact of loss functions on the quality of the final 3D reconstructions, we experimented with the following alternatives:

1. **Huber Loss:** A robust loss function that balances the sensitivity to outliers and the need for a smooth gradient. It's effective for tasks where the presence of outliers can significantly impact the performance of the model.
2. **Log Cosh Loss:** Works mostly like MSE, but will not be so strongly affected by the occasional wildly incorrect prediction, hence it might offer improved convergence properties.
3. **Structural Similarity Index (SSIM) Loss:** A perceptual loss function that assesses the structural similarity between two images, providing a measure closer to human perception. The loss function is represented as $1 - SSIM(x, y)$

3.2. Model Architectures and Distillation:

In order to understand how the model prediction changes, we experimented with the following variations of the model architecture and hyperparameters:

1. Model with **2 layers**, **128 filters**, trained for **10,000** and **100,000** iterations
2. Model with **8 layers**, **256 filters**, trained for **10,000** and **100,000** iterations

Also, we experimented with the distillation technique, expecting it would enhance the prediction quality of a smaller model. We used a model with 2 layers, 128 filters and trained it for 10,000 iterations with the following models as teacher:

1. Teacher model: **8 layers**, **256 filters**, previously trained for **10,000** iterations (later referenced as *'Teacher 10k'*)
2. Teacher model: **8 layers**, **256 filters**, previously trained for **100,000** iterations (later referenced as *'Teacher 100k'*)

3.3. Image Preprocessing

Image preprocessing was considered as another potential lever for enhancing reconstruction quality. We adjusted image contrast using the following formula:

$$enhanced\_image = original\_image * \mathbf{gain} + \mathbf{bias}$$

to enhance visibility in darker regions and bring out more detail. This adjustment was hypothesized to improve NeRF's ability to reconstruct fine details and maintain structural consistency.

3.3.1. Pipeline

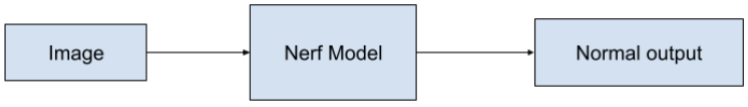

Fig. 1. The original NeRF model pipeline.

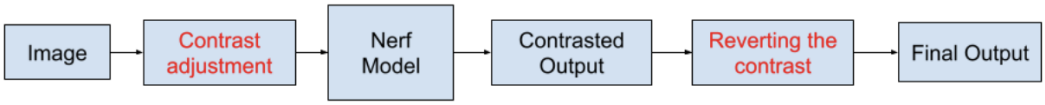

Fig. 2. The new pipeline with contrast adjustment.

3.3.2. Image Contrast Formula

$$enhanced\_image = original\_image * \mathbf{gain} + \mathbf{bias}$$

Gain Adjustment:

- **Increasing the Gain:** Amplifies the difference between higher and lower pixel values. This results in lighter areas becoming significantly brighter, while darker areas become relatively darker.
- **Decreasing the Gain:** Reduces the difference between light and dark areas, lowering the contrast and creating a more uniform appearance.

Bias Adjustment:

- **Adding a Positive Value:** Shifts all pixel values upward, making the overall image appear brighter.
- **Adding a Negative Value:** Shifts pixel values downward, darkening the image and enhancing shadows if the gain is adjusted appropriately.

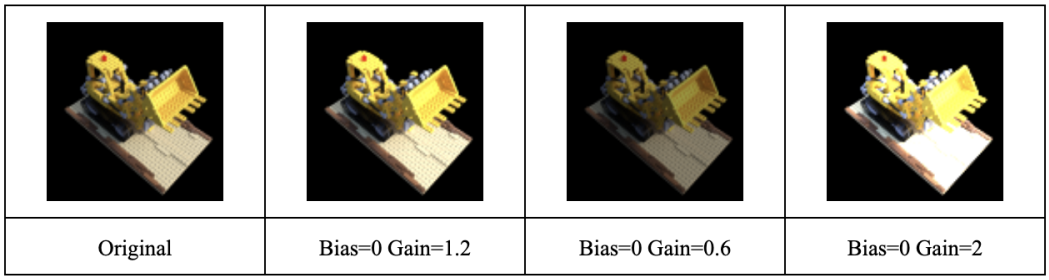

Fig. 3. Effect of changing gain on image contrast.

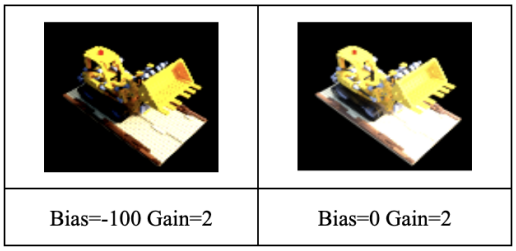

Fig. 4. Effect of changing bias with same gain on image contrast.

4. Results

Below, we used the SSIM, PSNR, and LPIPS index to compare our results with the original image.

4.1. Results with Different Loss Functions

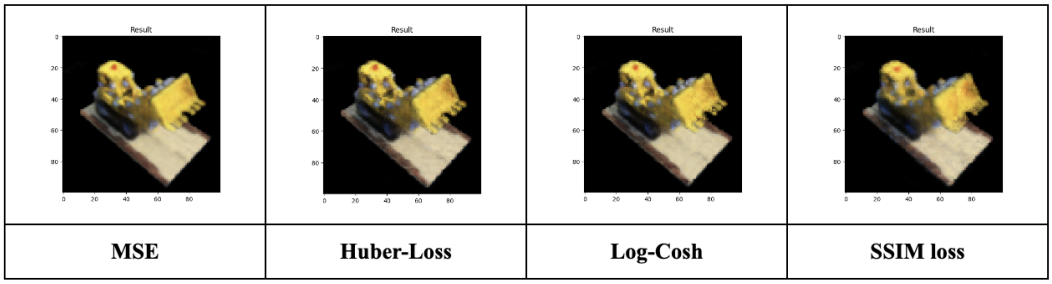

Fig. 5. Results from implementing different loss functions.

Table 1. We report PSNR (higher is better), SSIM (higher is better) and LPIPS (lower is better) for the loss functions tested. MSE outperforms the others in 2 of the 3 indexes used.

	MSE	Huber-Loss	Log-Cosh	SSIM loss
SSIM	**0.833**	0.829	0.831	0.803
PSNR	**34.05**	33.96	34.02	33.92
LPIPS	0.132	**0.119**	**0.119**	0.150

4.2. Results with Different Model Architectures and Distillation

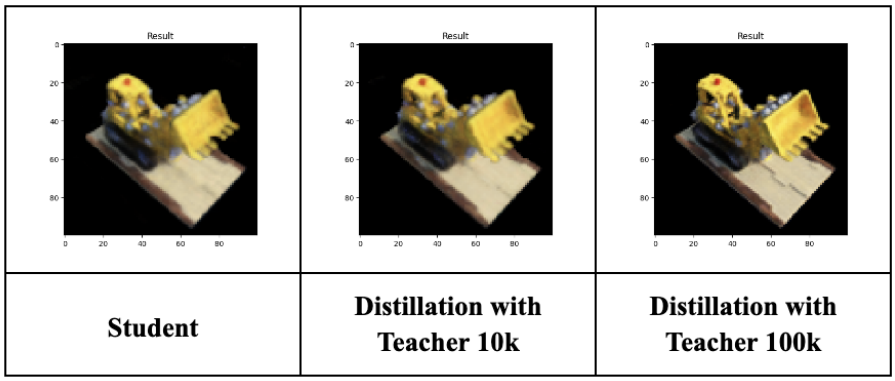

Table 2. Training time, RAM usage, and loss (MSE) across different architectures and distillation.

	2 Layers & 128 Filters		8 Layers & 256 Filters		Distillation	
	10k iters.	100k iters.	10k iters.	100k iters.	Teacher 10k	Teacher 100k
Training time (hs)	00:38:38	06:43:44	01:50:41	20:34:38	06:50:22	06:52:01
RAM usage	9.6 GB	9.6 GB	38 GB	38 GB	9.6 GB	9.6 GB
Loss (MSE)	0.0039	0.0016	0.0024	0.0010	0.0037	0.0015

Table 3. Performance comparison of different model architecture and distillation

	2 Layers & 128 Filters		8 Layers & 256 Filters		Distillation	
	10k iters.	100k iters.	10k iters.	100k iters.	Teacher 10k	Teacher 100k
SSIM	0.833	0.903	0.878	0.948	0.840	0.904
PSNR	34.05	35.06	34.63	36.26	34.17	35.06
LPIPS	0.132	0.080	0.090	0.030	0.130	0.080

4.3. Result with our Image Contrast

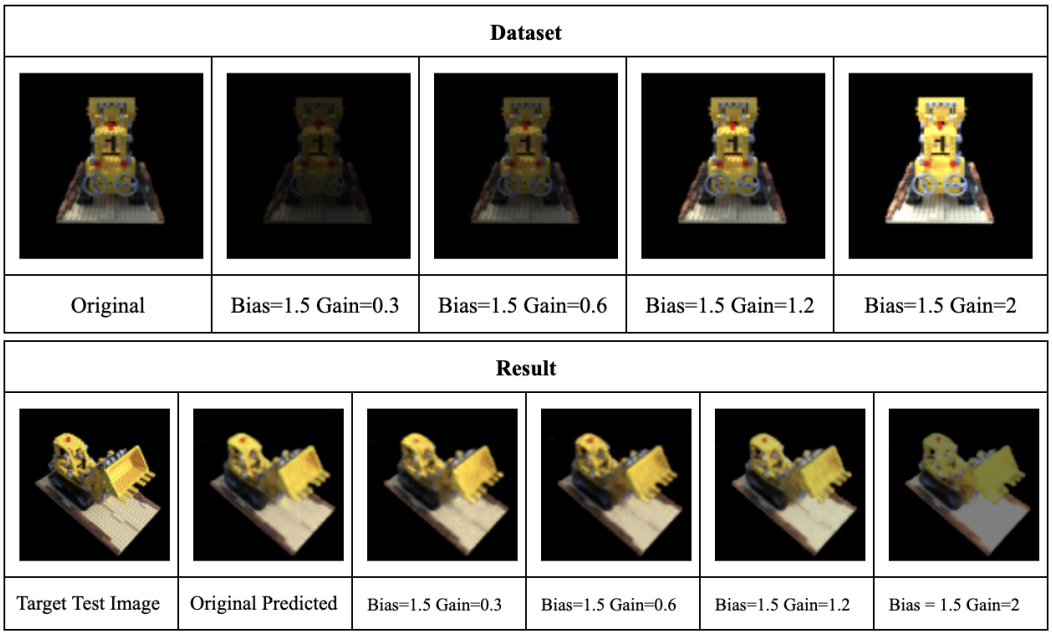

Fig. 6. The dataset used for training and its corresponding results.

Table 4. Performance comparison of different contrast levels

	Original	Bias=1.5 Gain=0.3	Bias=1.5 Gain=0.6	Bias=1.5 Gain=1.2	Bias=1.5 Gain=2
SSIM	0.853	0.850	0.847	0.842	0.764
PSNR	32.83	32.86	32.81	32.83	32.57
LPIPS	0.090	0.102	0.099	0.096	0.174

5. Limitations

1. **Resource Constraints:** The NeRF model is resource-intensive, so we limited training iterations to 10,000 in some experiments. For optimal results, 300,000 iterations would be needed, requiring approximately three days of training per experiment. With more resources, parallelism could be used during training to expedite the process.

2. **Dataset Bias:** The project relied on a synthetic, well-constructed dataset, which may not accurately reflect less-structured, real-world data. Therefore, the model might not perform similarly in other environments.

3. **Generalization Issues:**The results may not generalize well to other datasets due to differences in data quality, structure, or size. Thus, the limited dataset scope could affect the model's applicability to other scenarios.

6. Conclusion

Our experiments reveal several important findings about optimizing the NeRF model. Regarding the **impact of loss functions**, different loss functions lead to significant visual differences in the final output. Among them, Mean Squared Error produced the best overall results. Additionally, we found NeRF to be **resource-intensive during training**. Training the NeRF model is demanding in terms of both time and hardware resources. The 8-layer models required around 38 GB of GPU RAM and took over 20 hours for 100,000 iterations. Which is not surprising given the ongoing research efforts to compress the input data. Effectiveness of Distillation: Model distillation proved effective with NeRF, yielding slightly improved results over training smaller models directly. We also found insights regarding **training progression**: rapid learning was observed in the initial 10,000 to 20,000 iterations, but progress significantly slowed afterward. Finally, in terms of **image contrast adjustments**, we found out that using contrast-enhanced images did not lead to significant improvements in reconstruction quality.Excessive brightness could even worsen results.

7. References

References

[1] Matteo Bonotto et al. *CombiNeRF: A Combination of Regularization Techniques for Few-Shot Neural Radiance Field View Synthesis*. 2024. arXiv: 2403.14412 [cs.CV].

[2] Junyi Cao et al. *Lightning NeRF: Efficient Hybrid Scene Representation for Autonomous Driving*. 2024. arXiv: 2403.05907 [cs.CV].

[3] Yunhao Li et al. *SCINeRF: Neural Radiance Fields from a Snapshot Compressive Image*. 2024. arXiv: 2403.20018 [eess.IV].

[4] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV].

[5] Thomas Müller et al. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM Trans. on Graph.* 41.4 (July 2022), pp. 1–15. ISSN: 1557-7368. DOI: 10.1145/3528223.3530127. URL: http://dx.doi.org/10.1145/3528223.3530127.

8. Author Contributions

The individual contributions to this project are as follows:

- **Yi-ting Chiang:** Contributed to developing the research design, data collection, image preprocessing research, and manuscript writing.
- **Lucas Martinez:** Contributed to model modification, including testing different loss functions, experimenting with model architectures and distillation, and manuscript writing.
- **Oona Wood:** Contributed to the research idea, literature review and manuscript writing.