

# Neural Pruning for 3D Scene Reconstruction: Efficient NeRF Acceleration

Tianqi (Kirk) Ding<sup>\*1</sup> , Dawei Xiang<sup>\*2</sup> , Pablo Rivas<sup>3</sup> , and Liang Dong<sup>1</sup> 

<sup>1</sup> Department of Electrical and Computer Engineering, Baylor University, USA  
{Kirk\_Ding1,Liang\_Dong}@Baylor.edu

<sup>2</sup> Department of Computer Science and Engineering, University of Connecticut, USA  
ieb24002@uconn.edu

<sup>3</sup> Department of Computer Science, Baylor University, USA  
Pablo\_Rivas@Baylor.edu

**Abstract.** Neural Radiance Fields (NeRF) have become a popular 3D reconstruction approach in recent years. While they produce high-quality results, they also demand lengthy training times, often spanning days. This paper studies neural pruning as a strategy to address these concerns. We compare pruning approaches, including uniform sampling, importance-based methods, and coreset-based techniques, to reduce the model size and speed up training. Our findings show that coreset-driven pruning can achieve a 50% reduction in model size and a 35% speedup in training, with only a slight decrease in accuracy. These results suggest that pruning can be an effective method for improving the efficiency of NeRF models in resource-limited settings.

**Keywords:** NeRF · Model Compression · Neural Pruning · 3D Reconstruction · Efficiency

## 1 Introduction

Neural Radiance Fields (NeRF) have become an influential approach for synthesizing 3D scenes from 2D images while preserving detailed visual quality. Despite their effectiveness, they often demand extensive computational resources, requiring long training times that can extend over multiple days [20]. Researchers have explored strategies to address these concerns, primarily by compressing the model or pruning less important parameters. Such methods seek to reduce computational overhead without severely affecting the ability to represent scene details. Existing work underscores the importance of this endeavor: long-running NeRF processes may impede real-world tasks like cultural heritage documentation and architectural modeling, where rapid generation of 3D content is vital [13,19].

One compression strategy uses uniform sampling. By evenly distributing sample points, this method decreases computational load in a straightforward manner. However, uniform sampling may cause a loss of detail in regions where small

---

\* These authors contributed equally to this work.

features matter [3,1], and it may not suffice for applications that require precise rendering of scene intricacies [21]. Another approach is importance-based sampling, which allocates more resources to parts of the input space that have a larger impact on the final image. In doing so, it can enhance training efficiency and preserve fine-grained features [3].

A further extension involves coreset methods, in which smaller, representative subsets of the data are selected to approximate the full dataset. This approach aligns with the ultimate goal of model compression by retaining essential information while discarding details that have negligible influence on output quality. Studies on coreset-driven pruning have shown promising efficiency improvements, suggesting that such techniques can maintain high-fidelity novel views while using fewer samples [3,27,17]. Compression ideas, including voxel-based representations, have also surfaced in other works such as TinyNeRF, which achieved significant size reductions [27]. While these methods vary in details, they share a common objective of conserving computational resources without undermining visual realism.

As additional pruning and compression methods gain traction in 3D reconstruction, evaluations reveal key trade-offs. Some strategies greatly shorten training cycles but may sacrifice the ability to represent subtle visual information [7,3]. Balancing compression and performance is crucial in domains that require both rapid 3D synthesis and detailed rendering. Although there have been efforts to integrate features of importance-based sampling with coreset ideas, there remains a pressing need for principled evaluations of these hybrid solutions [8,10].

This paper aims to systematically investigate pruning-based compression methods for NeRF. By comparing uniform, importance-based, and coreset approaches, we highlight how each one affects both efficiency and reconstruction fidelity. Our results indicate that it is possible to compress NeRF models substantially while preserving most of their scene representation capabilities. We believe that this study not only advances the understanding of pruning in NeRF but also informs broader research on compressing 3D reconstruction architectures [4,9]. The contributions of this paper are summarized as follows:

- Demonstrate the feasibility of neural pruning techniques on 3D Reconstruction networks.
- Achieve decent performance on compressing model size and accelerating computation speed.

The rest of the paper is organized as follows. Section 2 reviews related work on pruning-based strategies in NeRF and related 3D reconstruction models. Section 3 introduces the methodology and experimental setup. Section 4 discusses our results in detail, including analysis of accuracy and speed gains. Finally, Section 5 presents conclusions and future research directions.

## 2 Related work

### 2.1 Neural Network Pruning

Neural network pruning seeks to cut down unnecessary parameters—either connections or neurons—in a trained model, with the goal of reducing computational demands and memory usage while keeping acceptable performance. Common pruning approaches fall into two categories: unstructured and structured.

*Unstructured Pruning* Unstructured pruning eliminates individual weights in the network, typically those whose magnitudes are below a chosen threshold. Although this can dramatically reduce the parameter count, it also creates irregular memory patterns, which can hinder efficient implementation on hardware designed for dense operations. Our initial experimentation with “Edge Pruning,” where the smallest-magnitude connections are removed, belongs to this category.

*Structured Pruning* Structured pruning, by contrast, removes entire neurons, filters, or even layers. This approach focuses on pruning higher-level architectural components, resulting in networks that maintain more regular structures suitable for hardware acceleration. However, removing significant blocks of the network can lead to greater performance drops if done too aggressively. In our later experiments, structured pruning became the main approach for boosting NeRF’s training speed because it more naturally reduces the size of the fully connected layers.

### 2.2 Pruning Methods

*Uniform Sampling* Uniform sampling selects neurons from a given layer at equal probability, thereby reducing network size in a straightforward manner. This random strategy offers computational simplicity since it does not require additional metrics or calculations to determine which neurons to remove. However, because it does not account for the contribution that individual neurons make to network performance, its outcomes can be suboptimal. In practice, uniform sampling often removes valuable neurons alongside those that are genuinely redundant, leading to potential decreases in overall accuracy.

*Importance Pruning* Importance pruning targets neurons or connections deemed least critical to the network’s inference. This entails evaluating metrics such as weight magnitudes or estimated contributions to the network’s output, allowing for the selective removal of parameters while striving to maintain predictive capability. By eliminating less impactful elements, the model can become more efficient without a large decrease in quality. Importance pruning has proven effective under various settings, particularly when datasets are evenly distributed or when memory resources are restricted. For instance, [5] demonstrates how this technique can streamline network architectures while preserving performance across different data domains.

*Coreset* Coreset-based pruning constructs a smaller yet representative subset of points to approximate the original layer or dataset. Under this method, each candidate neuron is assigned a weight, and sampling probabilities are determined in proportion to these weights. As a result, the chosen subset reflects the most informative components of the network, reducing computational costs while retaining critical features. Section 3.6 will provide an algorithm that further illustrates the implementation details.

In the work of [15], neurons were viewed as coreset elements, with the methodology selectively removing those deemed redundant in a bottom-up manner, thus preserving accuracy while recalibrating remaining connections. Similarly, [2] introduced a coreset-focused scheme for compressing convolutional neural networks, exploiting internal redundancies to reduce storage footprints and expedite inference, all without additional retraining steps.

### 2.3 Neural Radiance Fields (NeRF)

Neural Radiance Fields (NeRF), first introduced by [14], have been widely adopted for producing high-fidelity 3D scene reconstructions from 2D image sets. At its core, NeRF relies on a multilayer perceptron (MLP) to map each 3D coordinate  $(x, y, z)$  and viewing direction  $(\theta, \phi)$  to a density value  $\sigma$  and a corresponding RGB color. By accumulating these densities and colors along any desired viewing ray, NeRF renders photorealistic images from novel perspectives. Fig. 1 illustrates the MLP architecture used in NeRF.

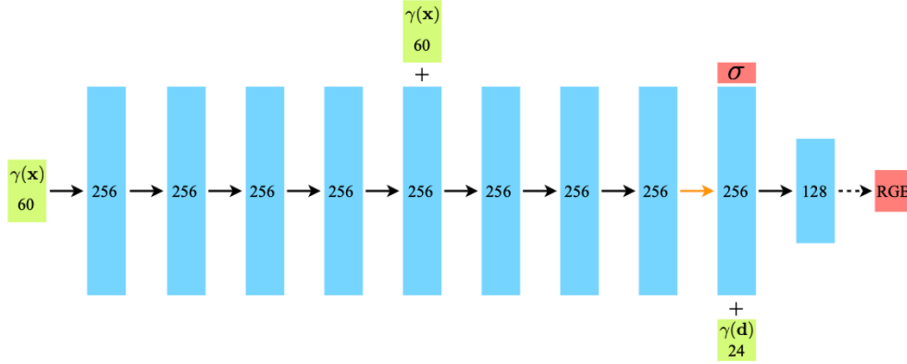


Fig. 1: A visualization of the MLP architecture from [14]. The network accepts  $\gamma(x)$ , a positional encoding of the 3D coordinates, along with  $\gamma(d)$ , an encoding of the viewing direction. It then outputs the density  $\sigma$  and RGB color of the point  $(x, y, z)$ .

*Model compression on NeRF* Efforts to compress or prune NeRF have been relatively limited. For instance, [6] explores edge-level pruning to highlight sparsity

in the network’s weights, whereas [18] leverages singular value decomposition (SVD) to yield lower-rank approximations of the NeRF representation. Additionally, [22] proposes pruning hash tables for 3D mesh-based reconstructions in Instant-NGP. Despite these advancements, the application of neuron-level pruning and coreset-based methods to NeRF remains underexplored, offering a potential avenue for more efficient 3D reconstructions without sacrificing realism.

In addition to prior studies on pruning and efficient NeRF representations, recent developments in parameter-efficient learning provide complementary insights. For example, VMT-Adapter [23] and MmAP [24] introduce efficient transfer learning methods for multi-task vision, which share structural similarities with importance-based pruning in our framework. Moreover, V-PETL [25] and the survey on parameter-efficient fine-tuning [26] highlight practical strategies to maintain model performance with fewer parameters, echoing the goals of our pruning techniques. These ideas inspire how pruning can be embedded into broader visual systems while retaining generalization. Additionally, reinforcement learning-based recognition of unordered robotic targets [12] demonstrates the importance of selecting representative visual features under constrained resources, which aligns with our coreset sampling approach.

### 3 Methods and Experiments

#### 3.1 Sparsity in NeRF MLP

Before trying to compress the MLP in NeRF, we need to prove that there is indeed sparsity in NeRF model. Therefore we first did the research on the neural representation of NeRF.

We have extracted the edge weights in the above MLP model. We first drew a frequency distribution Fig. 2. And we found that a large amount of edge weights are indeed very small (less than 0.05). This indicates that the MLP in NeRF is indeed sparse, and there is potential space for compression.

#### 3.2 Pruning the edges

Based on the discovery of the sparsity, we then tried to prune some low-weight edges: we discard the edges that have a low weight. When threshold is 0.05, only 40% edges will be kept and 60 % will be discarded. The edge pruning result is shown on Table 1:

The trends of increasing the remaining percentage and corresponding power-to-signal noise ratio (PSNR) indicate that eliminating redundant edges has minimal impact on the outcome. Hence, we will shift our focus from edge pruning to neuron pruning. The PSNR is defined as:

$$PSNR = 10 \log_{10} \left( \frac{MAX_I^2}{MSE} \right), \quad (1)$$

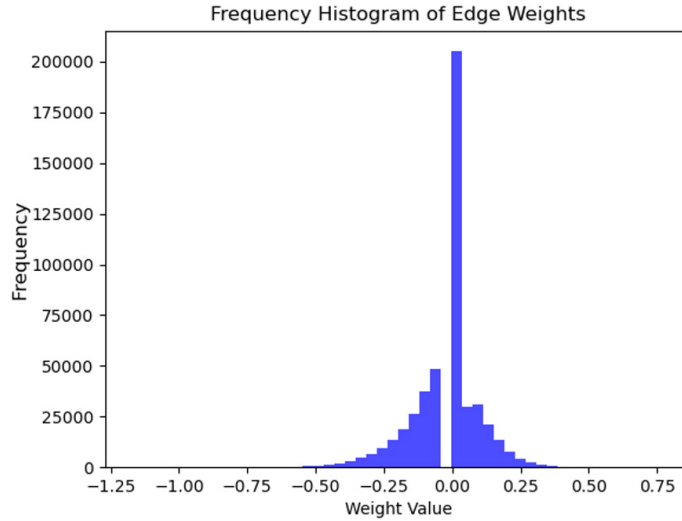


Fig. 2: Frequency Histogram of edge weights in NeRF. A large percent of edge weights fall into the range  $[0, 0.05]$ .

Table 1: Edge pruning comparison. PSNR denotes the peak signal-to-noise ratio, where higher values indicate closer similarity to ground truth.

	Threshold = 0	Threshold = 0.05	Threshold = 0.1
<b>Remaining Percentage</b>	100%	40%	20%
<b>PSNR (Test Set)</b>	21.5	21.3	20.8

where  $MAX_I$  is the maximum possible pixel value of the image, and Mean Squared Error (MSE) is defined as:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2, \quad (2)$$

where  $I(i, j)$  is the original image,  $K(i, j)$  is the reconstructed image, and  $m \times n$  are the dimensions of the images.

### 3.3 Problem of Unstructured Edge Pruning

While pruning edges can remove many low-magnitude connections, it does not reduce the overall size of the layer weights. Consequently, the training time remains largely unaffected, because the weight matrices between layers still have the same dimensions. To address this, we explored pruning at the neuron level. In particular, the original model utilizes seven fully connected layers, each sized  $256 \times 256$ , with one additional layer for incorporating the viewing direction. By

compressing these to  $64 \times 64$ , the network’s parameter count could theoretically be reduced to one-sixteenth of its initial size.

### 3.4 Uniform Sampling

A straightforward way to prune neurons is to select them at random. In this approach, each neuron has an equal probability of being retained, and the model is subsequently retrained. Table 2 illustrates the impact of uniform sampling, where we reduce each layer from  $256 \times 256$  neurons to  $64 \times 64$ . Although this technique does shrink the parameter count, it markedly degrades the peak signal-to-noise ratio (PSNR), indicating that random pruning can remove many crucial neurons.

Table 2: Performance with uniform sampling. We randomly select 64 neurons (out of 256) in each layer, then retrain the model.

	Baseline	Uniform Sampling
Connection Layer Size	$256 \times 256$	$64 \times 64$
PSNR	21.5	16.5
Model Size	2.38 MB	0.7 MB

The substantial drop in performance highlights two points: (i) random selection of neurons is ineffective at preserving important features, and (ii) a more informed pruning criterion is necessary to maintain quality while reducing model size.

### 3.5 Importance Pruning

The results in Table 2 show that randomly discarding neurons degrades performance, underscoring the need for a more selective pruning strategy. To address this, we compute an importance score for each neuron and then discard those with the lowest scores.

A neuron’s importance can be derived from its incoming and outgoing edge weights. Let the neuron  $i$  belong to layer  $L_k$ , with its preceding and subsequent layers denoted by  $L_{k-1}$  and  $L_{k+1}$ , respectively. We define:

$$w_{\text{in}}(i) = \frac{\sum_{j \in L_{k-1}} |e_{ji}|}{|L_{k-1}|}, \quad (3)$$

$$w_{\text{out}}(i) = \frac{\sum_{r \in L_{k+1}} |e_{ir}|}{|L_{k+1}|}, \quad (4)$$

where  $e_{ji}$  and  $e_{ir}$  represent the connection weights from neuron  $j$  to  $i$ , and from  $i$  to  $r$ , respectively. Fig. 3 illustrates this concept, showing incoming and outgoing connections for a target neuron in layer  $i$ .

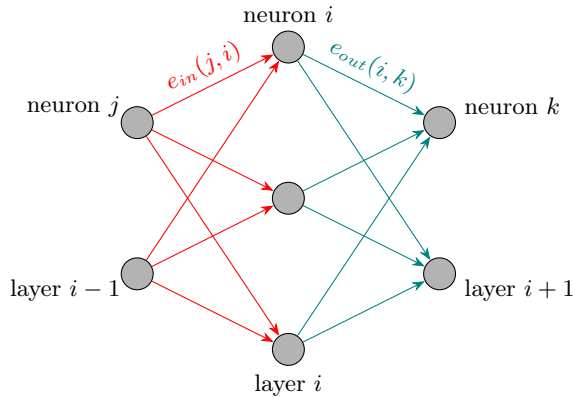


Fig. 3: An illustration of the importance weight calculation. Red arrows represent incoming edges for neurons in layer  $i$ , whereas green arrows indicate outgoing edges.

Using these definitions, we prune the neurons with the lowest importance scores and then retrain the resulting network. Table 3 presents the outcomes when using  $w_{\text{in}}$ ,  $w_{\text{out}}$ , or their product as the importance criterion.

Table 3: Results of different importance metrics. We compare pruning by  $w_{\text{in}}$ ,  $w_{\text{out}}$ , and their product, followed by retraining.

	Method	PSNR
No Pruning	Baseline	21.5
Importance Pruning	Select Neurons with $w_{\text{in}}$	19.5
	Select Neurons with $w_{\text{out}}$	20.0
	Select Neurons with $w_{\text{in}} \times w_{\text{out}}$	20.0

Notably, pruning based on  $w_{\text{out}}$  yields slightly higher PSNR values compared to using  $w_{\text{in}}$ . This finding suggests that focusing on how each neuron influences subsequent layers can be more effective in preserving overall model performance.

### 3.6 Coreset

Encouraged by the performance of importance pruning, we evaluated a coreset-based approach [16] to select a representative subset of neurons in each layer. This technique assigns a sampling probability to each neuron, then draws a smaller set of neurons that collectively preserve most of the model’s representational capacity. In practice, we follow a procedure similar to that of [16], using the product of  $w_{\text{in}}(i)$  and the average outgoing weight  $w_{\text{out}}(i)$  to guide neuron

selection. For simplicity, we replace the maximum outcoming edge weight with its average counterpart. The general pseudocode is shown in Algorithm 1.

---

**Algorithm 1** Coreset
 

---

**Input:**

$w_{in}(i)$  and  $w_{out}(i)$  for all the neurons in the layer  $k$  with  $i \in \{1, \dots, |L_k|\}$  where  $|L_k|$  is the total number of neurons in layer  $k$ , 256, in our case.  
 An integer (sample size)  $m \geq 1$ , 64, in our case.  
 An (activation) function  $\varphi: \mathbb{R} \rightarrow [0, \infty)$ ,  $\text{ReLU}()$ , in our case.  
 An upper bound controller  $\beta > 0$ , 3, in our case.

**Output:**

A weighted set  $(C, u)$  which corresponding to the coreset of the layer;

```

1: for all  $i \in \{1, \dots, |L_k|\}$  do
2:    $pr(i) := \frac{w_{in}(i)\varphi(\beta * w_{out}(i))}{\sum_{j \in L_k} w_{in}(j)\varphi(\beta * w_{out}(j))}$ 
3:    $u(i) := 0$ 
4: end for
5: for  $m$  iterations do
6:   Sample a point  $q$  from  $L_k$  with probability  $pr(q)$ .
7:    $C := C \cup \{q\}$ 
8:   for all  $i \in [L_k]$  do
9:      $u_i(q) := u_i(q) + \frac{w_{out}(q)}{m \cdot pr(q)}$ 
10:  end for
11: end for
12: return  $(C, u_1, \dots, u_k)$ 
  
```

---

After selecting neurons based on these coreset probabilities, we retrain the compressed network. Table 4 summarizes the outcomes of pruning each  $256 \times 256$  fully connected layer down to either  $128 \times 128$  or  $64 \times 64$ .

Table 4: Performance comparison using coreset sampling. Compressing from  $256 \times 256$  to  $128 \times 128$  preserves the highest PSNR while noticeably reducing both training time and model size.

Connection Layer Size	256 × 256	128 × 128	64 × 64
Model Parameters	595K	288K	177K
Model Size	2.38 MB	1.14 MB	0.7 MB
PSNR	21.5	<b>21.3</b>	20.1
Training Time (min / 100k iterations)	78.75	<b>51.25</b>	46.25

Notably, reducing the layer dimensions from  $256 \times 256$  to  $128 \times 128$  cuts the training time by roughly 35% and halves the model size, while maintaining a

PSNR of 21.3. This highlights the viability and effectiveness of neuron pruning in NeRF, particularly via coreset-based sampling.

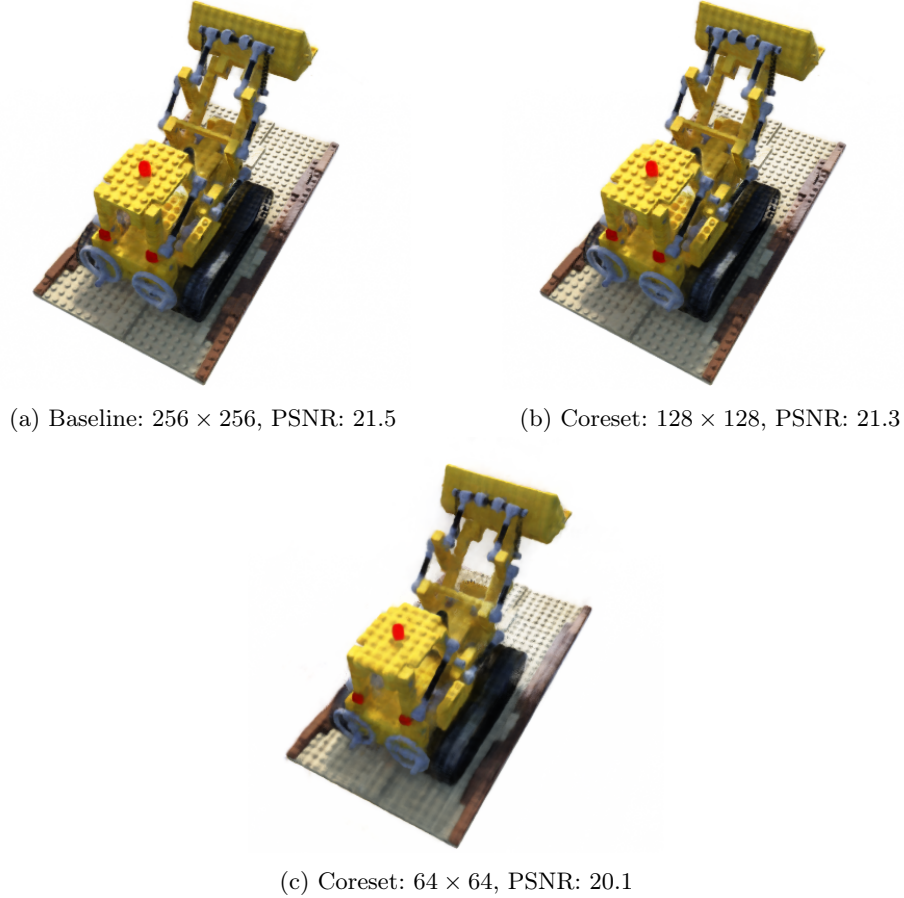


Fig. 4: Performance comparison visualization for different compression scales.

To visualize how pruning affects output quality, Fig. 4 illustrates sample renderings at different compression scales. As shown in the figure, the perceptual differences among the various compression scales are not dramatic, even though the numerical metrics do shift. The baseline model ( $256 \times 256$ ) produces the highest PSNR of 21.5, but compressing the layers to  $128 \times 128$  only slightly degrades performance (PSNR of 21.3), while providing substantial reductions in both training time and model size. By contrast, the more aggressive pruning to  $64 \times 64$  retains most of the scene’s overall structure yet reveals small losses of detail (PSNR of 20.1). These visual comparisons corroborate our quantitative

findings and highlight that coreset pruning can achieve a good balance between efficiency and scene fidelity.

## 4 Conclusion

In this paper, we compared multiple neural pruning strategies, including uniform sampling, importance pruning, and coreset-based methods, for compressing NeRF’s MLP layers. Our experiments showed that random approaches often degrade quality significantly, while importance-based pruning produces better results by targeting less influential neurons. Among the tested techniques, coreset pruning provides a balanced outcome, achieving a significant reduction in both model size and training time, with only a slight performance loss in terms of peak signal-to-noise ratio.

These findings highlight that neuron-level pruning is an effective way to accelerate NeRF training, taking advantage of latent sparsity within the network. Future research may also draw from parameter-efficient transfer learning benchmarks such as V-PETL [25] or causal representation learning like CausalSR [11], to integrate pruning with causal inference or visual generalization strategies. These directions may enable NeRF acceleration not only in synthetic benchmarks but also in more dynamic or interactive 3D environments. We expect that insights from this research will inform new developments in network compression for a broad range of 3D reconstruction and rendering applications.

**Acknowledgments.** The authors thank the Rivas.AI Lab (<https://lab.rivas.ai>) for the support and helpful feedback throughout this project. Part of this work was funded by the National Science Foundation under grants CNS-2210091, OPP-2146068, CHE-1905043, and CNS-2136961; and by the Department of Education under grant P116Z230151.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Arandjelović, R., Zisserman, A.: Nerf in detail: learning to sample for view synthesis (2021). <https://doi.org/10.48550/arxiv.2106.05264>
2. Dubey, A., Chatterjee, M., Ahuja, N.: Coreset-based neural network compression. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 454–470 (2018). [https://doi.org/10.1007/978-3-030-01234-2\\_28](https://doi.org/10.1007/978-3-030-01234-2_28)
3. Fang, J., Xie, L., Wang, X., Zhang, X., Liu, W., Tian, Q.: Neusample: neural sample field for efficient view synthesis (2021). <https://doi.org/10.48550/arxiv.2111.15552>
4. Guo, M., Fathi, A., Wu, J., Funkhouser, T.: Object-centric neural scene rendering (2020). <https://doi.org/10.48550/arxiv.2012.08503>
5. Hewahi, N.M.: Neural network pruning based on input importance. *Journal of Intelligent & Fuzzy Systems* **37**(2), 2243–2252 (2019). <https://doi.org/10.3233/jifs-182544>

6. Isik, B.: Neural 3d scene compression via model compression. arXiv preprint arXiv:2105.03120 (2021). <https://doi.org/10.48550/arxiv.2105.03120>
7. Jain, A., Tancik, M., Abbeel, P.: Putting nerf on a diet: semantically consistent few-shot view synthesis pp. 5865–5874 (2021). <https://doi.org/10.1109/iccv48922.2021.00583>
8. Jiang, C., Shao, H.: Fast 3d reconstruction of uav images based on neural radiance field. *Applied Sciences* **13**, 10174 (2023). <https://doi.org/10.3390/app131810174>
9. Kania, K., Yi, K., Kowalski, M., Trzcinski, T., Tagliasacchi, A.: Conerf: controllable neural radiance fields pp. 18602–18611 (2022). <https://doi.org/10.1109/cvpr52688.2022.01807>
10. Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J., Russell, B.: Editing conditional radiance fields (2021). <https://doi.org/10.48550/arxiv.2105.06466>
11. Lu, Z., Lu, B., Wang, F.: Causalsr: Structural causal model-driven super-resolution with counterfactual inference. arXiv preprint **arXiv:2501.15852** (2025)
12. Mao, Y., Tao, D., Zhang, S., Qi, T., Li, K.: Research and design on intelligent recognition of unordered targets for robots based on reinforcement learning. arXiv preprint (2025). <https://doi.org/10.48550/arXiv.2503.07340>
13. Mazzacca, G., Karami, A., Rigon, S., Farella, E., Trybala, P., Remondino, F.: Nerf for heritage 3d reconstruction. *The International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences* **XLVIII-M-2-2023**, 1051–1058 (2023). <https://doi.org/10.5194/isprs-archives-xxviii-m-2-2023-1051-2023>
14. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* **65**(1), 99–106 (2021). [https://doi.org/10.1007/978-3-030-58452-8\\_24](https://doi.org/10.1007/978-3-030-58452-8_24)
15. Mussay, B., Feldman, D., Zhou, S., Braverman, V., Osadchy, M.: Data-independent structured pruning of neural networks via coresets. *IEEE Transactions on Neural Networks and Learning Systems* **33**(12), 7829–7841 (2021). <https://doi.org/10.1109/tnnls.2021.3088587>
16. Mussay, B., Osadchy, M., Braverman, V., Zhou, S., Feldman, D.: Data-independent neural pruning via coresets. arXiv preprint arXiv:1907.04018 (2019). <https://doi.org/10.48550/arxiv.1907.04018>
17. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: neural radiance fields for dynamic scenes pp. 10313–10322 (2021). <https://doi.org/10.1109/cvpr46437.2021.01018>
18. Tang, J., Chen, X., Wang, J., Zeng, G.: Compressible-composable nerf via rank-residual decomposition. *Advances in Neural Information Processing Systems* **35**, 14798–14809 (2022). <https://doi.org/10.48550/arxiv.2205.14870>
19. Themistocleous, K., Abate, D.: The use of photogrammetry and nerf techniques to document built heritage p. 46 (2024). <https://doi.org/10.1117/12.3031605>
20. Tianqi, Ding, Xiang, D., Qi, Y., Yang, Z., Zhao, Z., Sun, T., Feng, P., Wang, H.: Nerf-based defect detection (2025), <https://arxiv.org/abs/2504.00270>
21. Wu, H., Sun, J., Duan, H., Cai, W.: Newbie at cuhksz: a voxel art puzzle game for campus orientation (2023). <https://doi.org/10.1109/icce56470.2023.10043459>
22. Xie, X., Gherardi, R., Pan, Z., Huang, S.: Hollownerf: Pruning hashgrid-based nerfs with trainable collision mitigation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 3480–3490 (2023). <https://doi.org/10.1109/iccv51070.2023.00322>

23. Xin, Y., Du, J., Wang, Q., Lin, Z., Yan, K.: VMT-adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 16085–16093 (2024). <https://doi.org/10.1609/aaai.v38i14.29541>
24. Xin, Y., Du, J., Wang, Q., Yan, K., Ding, S.: MmAP: Multi-modal alignment prompt for cross-domain multi-task learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 16076–16084 (2024). <https://doi.org/10.1609/aaai.v38i14.29540>
25. Xin, Y., Luo, S., Liu, X., Zhou, H., Cheng, X., Lee, C.E., Du, J., Wang, H., Chen, M., Liu, T.: V-petl bench: A unified visual parameter-efficient transfer learning benchmark. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 37, pp. 80522–80535 (2024)
26. Xin, Y., Luo, S., Zhou, H., Du, J., Liu, X., Fan, Y., Li, Q., Du, Y.: Parameter-efficient fine-tuning for pre-trained vision models: A survey. arXiv preprint (2024). <https://doi.org/10.48550/arXiv.2402.02242>
27. Zhao, T., Chen, J., Leng, C., Cheng, J.: Tinynerf: towards 100 x compression of voxel radiance fields. Proceedings of the Aaai Conference on Artificial Intelligence **37**, 3588–3596 (2023). <https://doi.org/10.1609/aaai.v37i3.25469>