Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/45/20241047

Accelerating Neural Radiance Field by network pruning

Deng Yu Dong

Chinese International School, Hong Kong, China

michaeldeng580@gmail.com

Abstract. Novel view synthesis, a crucial problem in computer graphics with extensive applications, has gained significant attention due to its increasing demand in fields such as autonomous driving, virtual reality, film, and the visual arts. Recent advancements in machine learning have led to progress in generating photo-realistic imagery, particularly with the Neural Radiance Fields (NeRF) method. Despite its impressive results, NeRF requires high computational resources for training and synthesis, impeding its real-world applications. This work aims to investigate the potential of network pruning to reduce NeRF's computational complexity without compromising the quality of the generated images, addressing the limitations of the original model and paving the way for more efficient view synthesis techniques. We propose a novel pruning method tailored for NeRF, Coarse-Fine-Pruning (CFP). We note that the NeRF framework is divided into two parts: a coarse network that learns the global structure of the scene and a fine network that learns the local details. Crucially, our experiments show that the coarse and fine networks have different tolerance to pruning. As such, our method adapts the pruning rate to each network, achieving a higher compression rate without compromising the quality of the generated images. We evaluate CFP and show that it can reduce the number of parameters by 90% while maintaining the same level of image quality.

Keywords: View Synthesis, Scene Representation, Volume Rendering, NeRF, Network Pruning.

1. Introduction

Novel view synthesis is a long-standing problem in computer graphics and has wide-ranging applications in fields such as autonomous driving, virtual reality, film, and the visual arts. It aims to render novel views of a complex 3D scene given freely-placed viewpoints as input.

Recently, several studies [1-3] have explored the use of neural networks in view synthesis and have made significant progress towards generating photo-realistic imagery. In particular, the method of Neural Radiance Fields (NeRF) [4] has garnered considerable attention. NeRF consists of two neural networks - one "coarse" and one "fine". Both networks take a 3D coordinate and a 2D viewing direction as input and output a predicted 3D *rgb* color value and a single volume density value. Despite having identical input and output dimensions, the two networks perform different tasks and improve NeRF's efficiency and image quality. NeRF has been demonstrated to outperform previous methods both quantitatively and qualitatively.

However, NeRF has a large computational overhead. Synthesizing one image requires a considerable amount of passes through NeRF's neural networks. In fact, Garbin *et al.* [5] reports that just rendering one image pixel requires hundreds of MLP calls. This means that for high-end GPUs, rendering one

DOI: 10.54254/2755-2721/45/20241047

low-quality image already requires a few seconds; for devices with lower computational power, such as phones or laptops, NeRF is therefore completely impractical. NeRF's computational complexity is particularly inconvenient because a NeRF model cannot be generalized to other scenes, severely impeding its real-world applications.

In this work, we investigate whether network pruning can mitigate NeRF's computational complexity without compromising generation quality. Neural networks are typically over-parameterized [6, 7] leading to redundancies in the weights which unnecessarily increase computation and memory. Network pruning sets less important parameters to zero to prune the corresponding connections (see Figure 1), thereby saving computation and memory. Previous works [8-11] have shown that more than 90% parameters of a classification model can be pruned without accuracy degradation. However, it is still an open problem whether NeRF models can be pruned without compromising image generation quality. This work aims to answer the above question.

We introduce Coarse-Fine-Pruning (CFP), a compression method that is specifically designed for NeRF models. The intuition behind CFP is that a NeRF model consists of the coarse and fine networks which serve different purposes. To explore the different tolerance to pruning of the coarse and fine networks, we prune them individually with different sparsity levels and pruning strategies. For pruning strategies, we test the one-shot and iterative methods. The one-shot method prunes the neural network to the desired sparsity level directly, while the iterative method prunes the neural network multiple times until the desired sparsity level is reached.

We find that the coarse network is much more tolerant to pruning than its fine counterpart across different sparsity levels and pruning strategies. We observe that up to 95 percent of the parameters in the coarse network can be pruned without significant changes in generation quality. By contrast, for the fine network, the generation quality significantly degrades even when only 60 percent of the parameters are pruned. We also note that the iterative method proves to be better in most cases.

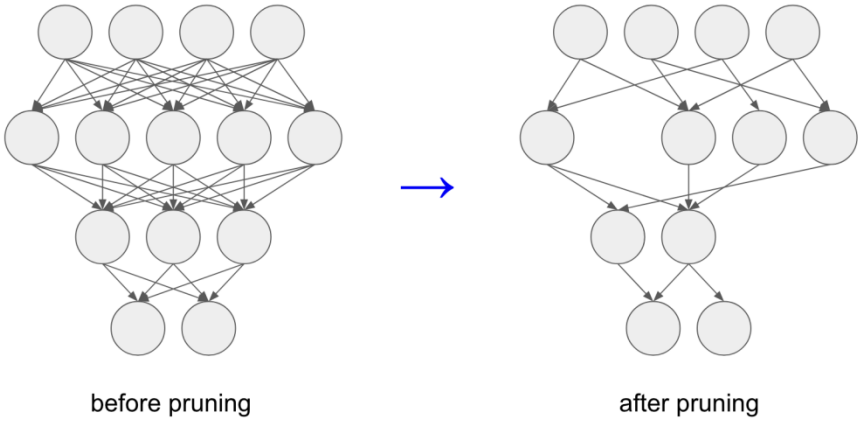

Figure 1. An illustration of network pruning where some unimportant weights (also known as connections) are removed after pruning [8].

Our method is better than previous NeRF acceleration methods in numerous ways. One important advantage of our method is its simplicity. While previous methods, such as NSVR [12] and KiloNeRF [13], have achieved impressive acceleration results, they fundamentally alter NeRF's architecture and require additional implementation resources. By contrast, our method performs simple modifications to NeRF and demonstrates good results without changing its architecture and training process. Furthermore, our method can improve both NeRF's computation and its memory, while other methods, such as FastNeRF [5], trade memory for more efficient computation. NeRF has large memory requirements as its neural networks have 8 hidden layers, each with 256 neurons. As we demonstrate

DOI: 10.54254/2755-2721/45/20241047

that a significant portion of parameters can be pruned, our method can significantly reduce memory and input/output (IO) cost. This work is the first work that systematically combines NeRF and network pruning, thereby paving the path for a novel way of improving NeRF.

2. Related Work

2.1. Network Pruning

Several studies [6, 7] have shown that neural networks are typically over-parameterized and significantly redundant, leading to unnecessary memory consumption and inefficient computation. Methods including matrix decomposition [14], features recombining [15], knowledge distillation [16-18], and reducing floating-point precision of weights [19] have been proposed to reduce this redundancy, to varying degrees of success. Currently, the method of network pruning is one of the most popular compression methods due to its high efficacy [20].

Network pruning is popularized by two influential papers in the 1990s [10, 11] as a way to reduce the size of a neural network by pruning less salient parameters. Here, saliency is a measure of how much the presence of an individual weight affects the final training error. One of the most influential network pruning algorithms is articulated by Han *et al.* [8] in 2015. This algorithm uses magnitude as a measure of saliency based on the intuition that weights with a smaller absolute value have less impact on the final output of the model. The algorithm is divided into three steps: first, the neural network model is trained until convergence; second, each parameter with a magnitude under a certain threshold is pruned, or changed into zero; third, the remaining weights are fine-tuned by retraining the model. This an iterative process where pruning followed by retraining is repeated several times. The paper demonstrates that 90 percent of weights in some models can be pruned without significant impacting task performance.

Several variants of the algorithm were also developed. The pruning proposed by Han *et al.* [8] is a form of unstructured pruning - the pruning of individual parameters. However, several papers [9, 18, 21] instead investigate the efficacy of structured pruning - that is, the pruning of entire neurons and layers - with some showing that structured pruning is more hardware-friendly. Both Optimal Brain Damage [10] and Optimal Brain Surgeon [11] criticize magnitude-based pruning and propose using the Hessian matrix of the objective function as a better measure of saliency. Frankle *et al.* [22] and Liu *et al.* [23] suggest that instead of fine-tuning weights after pruning, it is better to reset the weights - either randomly initializing them again or returning them to their original initialized values - and train them from scratch on the new, sparse architecture. However, Gale *et al.* [24] refutes this claim and instead suggests that fine-tuning is necessary.

As previously mentioned, the purpose of this work is to investigate how network pruning may increase NeRF's inference speed. In particular, this work takes heavy inspiration from the algorithm proposed by Han *et al.* [8] as it is influential and reliable.

2.2. Preliminaries on NeRF

NeRF fits ReLU-based multi-layer perceptrons (MLP) on a set of input views, encoding a continuous volume in the parameters of the MLP. The MLP encapsulates the radiance field of the scene to create an implicit neural representation of the scene's 3D structure. Specifically, it acts as a vector valued function, taking as its input a 5D vector with a 3D coordinate $x=(x, y, z)$ and a 2D viewing direction $d=(\alpha, \phi)$. The output of the function is a 4D output vector which represents view-dependent color values $c=(r, g, b)$ and a single volume density value α. The MLP $F_\theta: (x,d) \rightarrow (c,\alpha)$ aims to optimize the weights θ to better represent the structure of the scene. Differing from other volumetric techniques [25-27], NeRF aims to overfit a single scene to generate photo-realistic novel views and does not consider generalization across different scenes.

To synthesize a novel view, NeRF determines the color of each pixel in the 2D image. To estimate this color, a ray is marched through the pixel to sample a set of 3D points. The 3D coordinates of these points as well as the camera's viewing direction are passed to the MLP to obtain an output set of colors and densities. This output set is used to estimate a continuous integral through quadrature to determine

DOI: 10.54254/2755-2721/45/20241047

the color of the pixel to be rendered. This rendering process is inherently computational intensive because synthesizing one image requires numerous MLP passes for each sampled 3D point of each ray through each pixel.

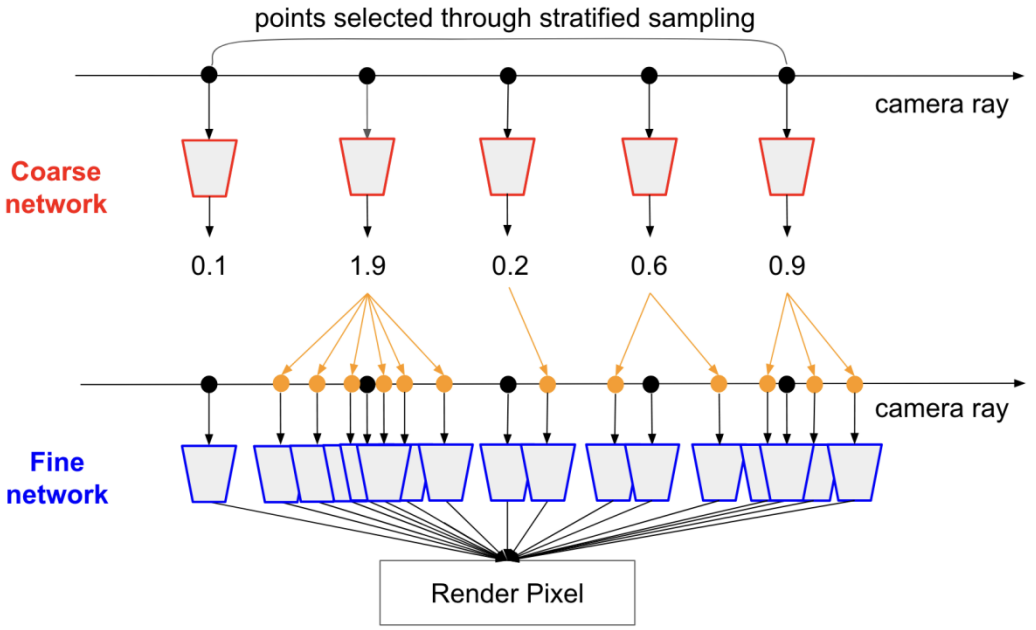

Figure 2. The function of the coarse and fine networks in NeRF's hierarchical sampling mechanism.

Central to this work is NeRF's distinction between its coarse and fine networks. These two networks carry out "hierarchical sampling," an efficient way of sampling 3D points along the camera ray which favors points with higher relevance to the scene. The coarse network, as input, receives points sampled equidistant along the ray. The network outputs a number which indicates how relevant the point (and points around it) are to the structure of the scene. These outputs provide a more informed sampling of points, in which more points are sampled from regions of higher relevance. The fine network then utilizes this informed sampling to arrive at a suitable prediction. This process is shown in Figure 2. Our pruning method, Coarse-Fine-Pruning (CFP), takes advantage of NeRF's simultaneous optimization of these two networks to improve NeRF's computational requirements. We discover that, because the coarse and fine networks perform different functions, they have different tolerance to pruning.

2.3. NeRF Acceleration

There are several existing works on NeRF's acceleration for real-time rendering.

Both KiloNeRF [13] and DeRF [28] split NeRF's single MLP into a set of smaller MLPs, each representing a fraction of the scene. However, unlike our method, these divide-and-conquer ideas are not hardware-friendly. Hedman *et al.* [29] stores four-dimensional feature vectors and diffuse color vectors in a novel data structure for efficient rendering. FastNeRF [5] factorizes view-dependent color as an inner product between a position-dependent deep radiance map and a direction-dependent feature vector. Yu *et al.* [30] instead represents view-dependent color as spherical harmonics, storing the scene as a sparse voxel octree which uses SH coefficients to represent appearance and density at each leaf. Despite the impressive results of these novel methods, they fundamentally reconstruct NeRF's architecture and require complex implementation. Our method, by contrast, preserves NeRF's architecture entirely and performs only slight modifications. The simplicity of our method makes it more accessible.

Lindell *et al.* [31] speeds up the NeRF's integral process by taking the derivative of its neural networks, enabling "automatic integration." Though this method requires merely two evaluations of the

DOI: 10.54254/2755-2721/45/20241047

network to render a pixel, it produces images with reduced quality. Neural Sparse Voxel Fields (NSVF) [12] takes advantage of the volumetric sparsity of real-world scenes, combining implicit neural functions with explicit voxel grids to create a sparse octree. DONeRF [32] includes a depth oracle network that predicts the locations of samples in relation to surfaces to render images with less samples per ray. These methods are orthogonal to our method.

3. Method

In this section, we describe in detail our methodology.

Evaluation Metric

We use the PSNR metric to evaluate the quality of the generated image, following previous work [4, 5,13]. A higher PSNR value means higher image quality. To track the progress of pruning, we record the PSNR value every 5,000 retraining steps.

When fully trained and not pruned, vanilla NeRF can generate images with a PSNR value of 37 on the lego dataset [26]. We compare generated images of the lego bulldozer with PSNR values of 30, 34, and 37 (see Figure 3). When zooming into a close-up of the bulldozer, the image with a PSNR value of 37 shows clearly distinguishable circular spots. While these spots are still discernible in the image with a PSNR value of 34, they are blurrier. In the image with a PSNR value of 30, the spots completely blur together. Through this visual comparison, we establish that, in most cases, a PSNR value of 34 or more is good generation quality while a PSNR value of 30 or more is acceptable generation quality.

3.1. Pruning Strategy

Separating the coarse and fine networks

In our experiments, we treat NeRF's coarse and fine networks separately. That is, each experiment is replicated two times - once for each network. As previously stated, despite the fact that the coarse and fine networks are structurally identical, they play different roles in the NeRF framework. The coarse network produces a more informed sampling along the ray, while the fine network uses this informed sampling to predict the pixel color. As such, we test their tolerance to pruning separately.

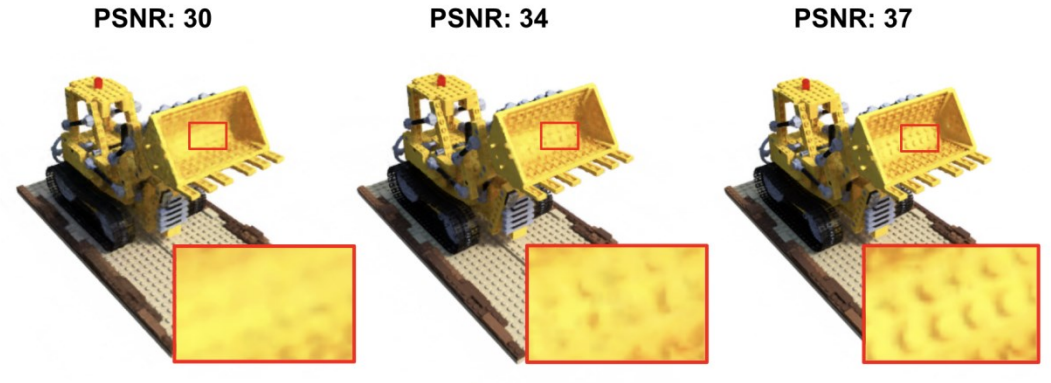

Figure 3. A qualitative comparison between generated lego images with PSNR values of 30, 34, and 37.

One-shot and iterative pruning

In order to test how different pruning methods impact the pruning tolerance of the coarse and fine networks, we consider two methods: one-shot pruning and iterative pruning. One-shot pruning prunes the weights in a network only once. Iterative pruning, on the other hand, prunes a network multiple times, and a cycle of pruning and fine-tuning is one iteration. The difference between one-shot and iterative pruning is shown in Figure 4. Typically, iterative pruning outperforms one-shot pruning [8, 22]. This is because the repeated process of pruning and fine-tuning allows the network to incrementally find

DOI: 10.54254/2755-2721/45/20241047

less salient parameters - which is often more accurate than finding less salient parameters in one go. We implement both types of pruning in our experiments to see whether this conclusion holds for NeRF.

Pruning ratios

In each experiment, we prune a different percentage of weights to test tolerance to pruning. The percentages we test are: 50%, 60%, 70%, 80%, 90%, and 95%, and they are tested for the "coarse" and "fine" networks separately. The reasons for these chosen percentages are as follows:

- We do not test pruning less than 50% of the parameters because it negligibly affects task performance. We also do not test pruning more than 95% of the parameters because task performance will be too significantly affected. These conclusions are empirically validated by our own experiments and also by many previous works [8, 22, 23].
- We space the percentages out by 10% with the exception of the 5% difference between 90% and 95%. The 10% gap allows us to cover a good range of percentages while not making too drastic leaps which may miss out on important percentages in between. There is, however, a big difference between 90% and 95% as the latter is pruning double as many parameters as the former. Therefore, we believe the inclusion of 95% is necessary for a complete set of experiments.

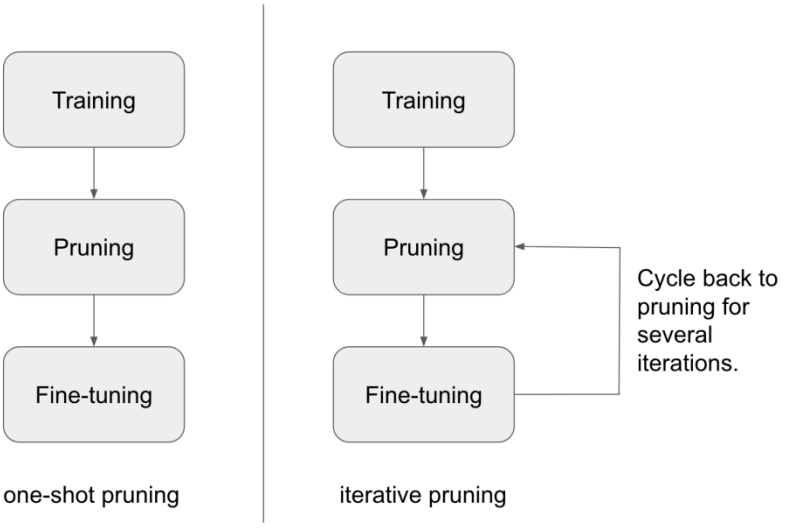

Figure 4. A figure showing the difference between one-shot and iterative-pruning.

Scheduling pruning

We conduct pruning on a pre-trained NeRF model which is obtained from 200,000 pre-trained steps. Here we describe how we schedule the pruning process.

- For one-shot pruning, we prune a certain percentage of weights on the pre-trained NeRF model and retrain the pruned model for an additional 20,000 steps. We empirically find that retraining more than 20,000 iterations will no longer offer a better result.
- For iterative pruning, we prune a total of ten times. Each pruning is followed by 1,000 steps of retraining. For instance, suppose we aim to prune 90% of the parameters. On each iteration, we prune 9% of the parameters, retrain for 1,000 steps, and on the next iteration prune another 9%. As such, pruning is complete after 10,000 steps. After pruning, we retrain for an additional 20,000 steps for the pruned model to converge.

3.2. Coarse-Fine-Pruning

Our novel pruning method, CFP, prunes the coarse and fine networks with significantly different percentages. In particular, we prune the fine network much less than the coarse network. This is because

DOI: 10.54254/2755-2721/45/20241047

the fine network is responsible for rendering pixels while the coarse network is only responsible for generating an informed sampling of points. Because the fine network fulfills a more important purpose, its parameters are more significant, which is why it is less tolerant to pruning. We prove these findings and the validity of CFP in the experiments section below.

4. Experiments

In order to study the impact of pruning on the NeRF model, we conduct experiments to answer the following questions,

- How tolerant to pruning are NeRF's coarse and fine networks?
- Which pruning method, one-shot pruning or iterative pruning, is better suited to prune NeRF's networks?
- How much can the coarse and fine networks be pruned in total without compromising generation quality, if the pruning of the two networks are combined?

4.1. Setup

We run all our experiments on Google Colab with an Apple M2 8-Core GPU. We use the lego dataset when conducting our experiments [4]. The dataset contains photographs of a lego bulldozer from different perspectives with a transparent background. It consists of 99 photographs in the train set, 99 photographs in the cross validation set, and 199 photographs in the test set. Figure 5 gives some sample images from this dataset.

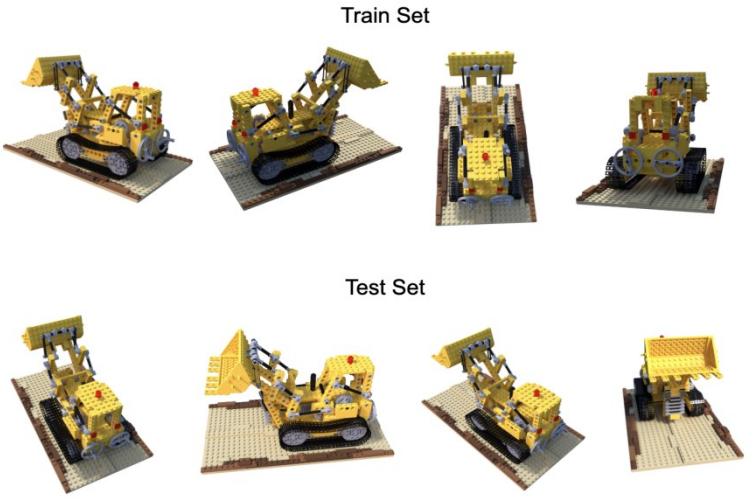

Figure 5. Four sample images from both the train and test sets: views from the train and test sets do not have any overlap.

We use a learning rate of 0.0005 and a learning rate decay rate of 250 in every 1,000 steps. We use mini-batch gradient descent with a batch size of 4,096. These hyperparameters have been empirically fine-tuned. We also use the ADAM optimization method.

4.2. Empirical Results

We summarize our results in Table 1, which shows, for each experiment, how good the generation quality is after all pruning is complete. We take the maximum PSNR value because the PSNR value can be unstable during fine-tuning. The results in Table 1 are visualized in Figure 6.

DOI: 10.54254/2755-2721/45/20241047

Table 1. The maximum peak signal-to-noise ratio (PSNR) values recorded for each experiment. We test various sparsities and two pruning methods: one-shot and iterative.

Weights Pruned (%)	Maximum PSNR Value			
	One-shot Coarse	One-shot Fine	Iterative Coarse	Iterative Fine
50%	36.31	35.71	36.57	34.91
60%	36.12	34.66	36.53	34.35
70%	35.39	31.05	35.90	32.32
80%	35.14	26.60	35.69	28.10
90%	34.79	22.88	35.25	26.69
95%	33.04	19.05	34.29	23.88

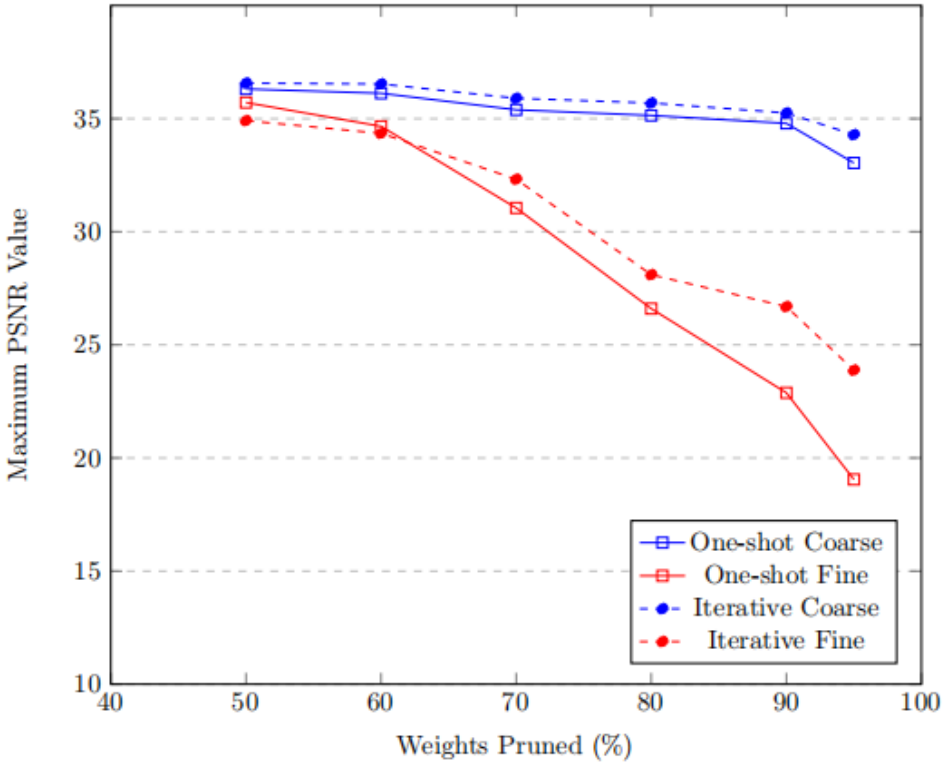

Figure 6. Visualizing the data from Table 1.

We first evaluate what these results reveal about the pruning tolerance of the two networks, then discuss which pruning method is better.

4.3. Pruning tolerance of the coarse and fine networks

Our findings suggest that NeRF's coarse network is much more tolerant to pruning than its fine counterpart. Referring to Table 1, even when 80% of the coarse network is pruned, the generation quality is negligibly impacted as the maximum PSNR value stays above 35 regardless of the pruning method.

DOI: 10.54254/2755-2721/45/20241047

Even when 95% of the parameters are pruned, the iterative pruning method still yields a maximum PSNR value of above 34, maintaining good generation quality.

However, for the fine network, the generation quality is good only when 60% or less of the parameters are pruned, with the maximum PSNR value staying above 34. When more parameters are pruned, the maximum PSNR value quickly decreases to less than 30, with the worst being only 19.05, when 95% of the network is pruned with the one-shot method. As such, when pruning more than 60% of the parameters, the generation quality is significantly impacted.

We give a qualitative visual comparison (Figure 7) of the images generated in four different experiments: pruning the parameters of the coarse and fine networks 60% and 95%. We chose these experiments because they best demonstrate the difference between the pruning tolerance of the two networks. For consistency, the pruning method is all one-shot; the effect of pruning on the generated image is similar regardless of the pruning method. We show the generated image at three different stages of retraining: retraining step 0 (when pruning has just completed and retraining has not started), step 5,000, and step 20,000. In the figure, we also show the PSNR value of each image below the image.

We observe that, without retraining, even when 95% of the coarse network is pruned, the smaller features in the scene are still recognizable and the colors are still correct. Despite the PSNR value dropping to 26.69, below acceptable quality, the general structure of the scene still remains intact. The bulldozer's tracks at step 0, despite not being completely generated, are still easily recognizable. By steps 5,000 and 20,000, the tracks are generated completely and realistically and the images are of good visual quality, with PSNR values above 35. When only 60% of the coarse network is pruned, there is almost no change to image quality as the PSNR value is 34.04, maintaining good results.

However, when the fine network is pruned without retraining, the smaller features in the scene are almost entirely lost even when only 60% of the parameters are pruned. The lego bulldozer is hardly distinguishable and the colors are completely wrong; quantitatively, the PSNR value is 18.12, way below the acceptable quality threshold of 30. At 95%, when even more of the parameters in the fine network are pruned, the image at step 0 becomes a pattern of pink and white that is entirely unidentifiable. The PSNR value becomes 5.53, indicating very bad generation quality. As such, comparing the four images before any retraining clearly demonstrates that the fine network is less tolerant to pruning than its coarse counterpart.

The fine network, however, shows bigger changes in generation quality before and after retraining. When 60% of the fine network is pruned, at step 0 the PSNR value is 18.12, but it increases significantly to 33.79 after only 5,000 steps of retraining. Visually, at step 0 only the rough structure of the bulldozer is identifiable, yet at step 5,000, the colors are completely correct and the finer details of the scene are rendered. In fact, when 60% of the parameters are pruned, the coarse and fine network generate images with similar visual quality at step 5,000, even though the image quality is drastically different at step 0. When 95% of the fine network is pruned, the PSNR value increases drastically as well, from 5.53 at step 0 to 13.99 and 18.78 at steps 5,000 and 10,000. Visually speaking, while the structure of the scene is completely unidentifiable at step 0, it is at least present at step 5,000 and continues to improve until step 20,000, though still far from being acceptable.

DOI: 10.54254/2755-2721/45/20241047

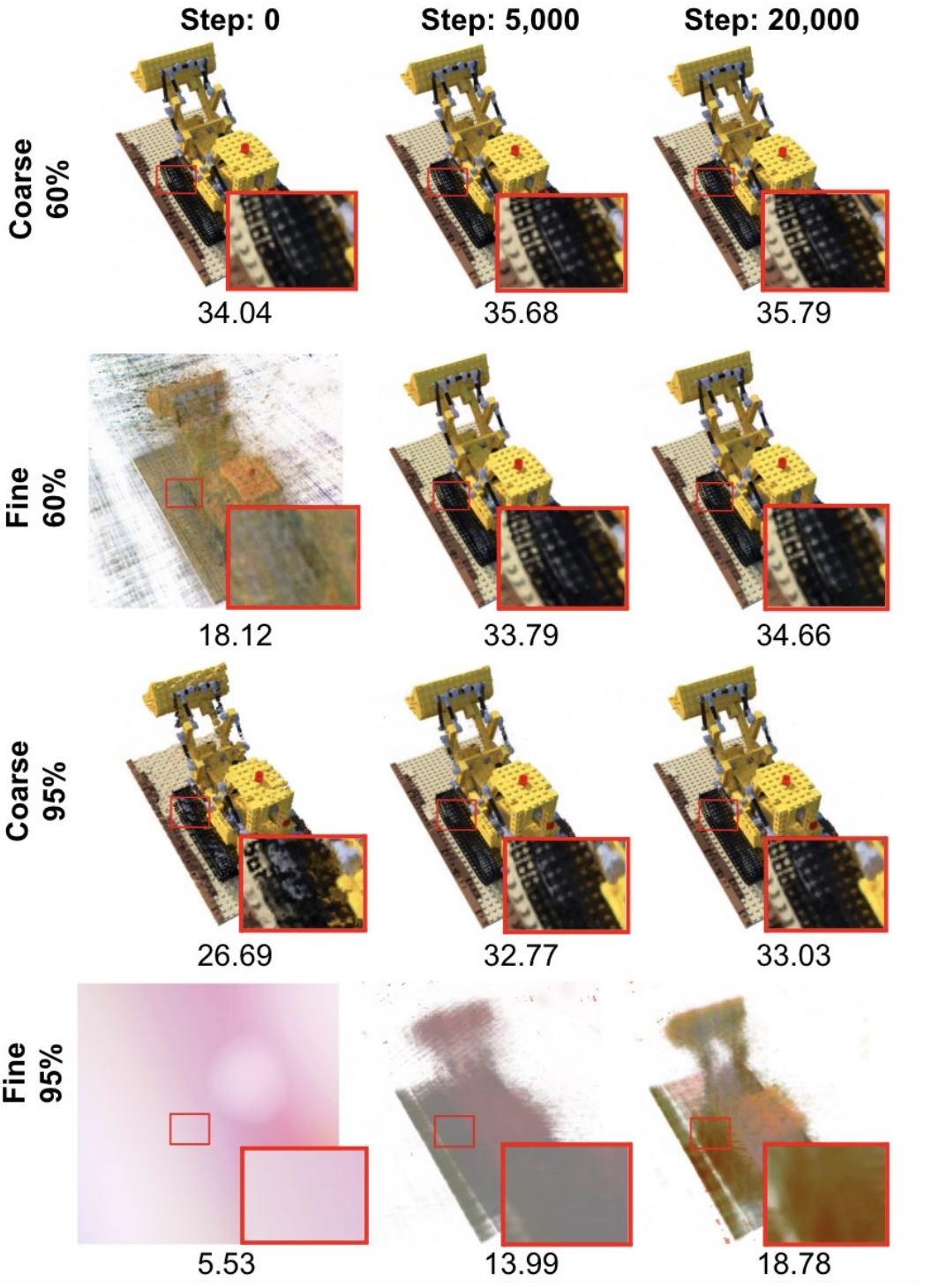

Figure 7. A qualitative comparison between the generated lego images of the coarse and fine networks after pruning.

DOI: 10.54254/2755-2721/45/20241047

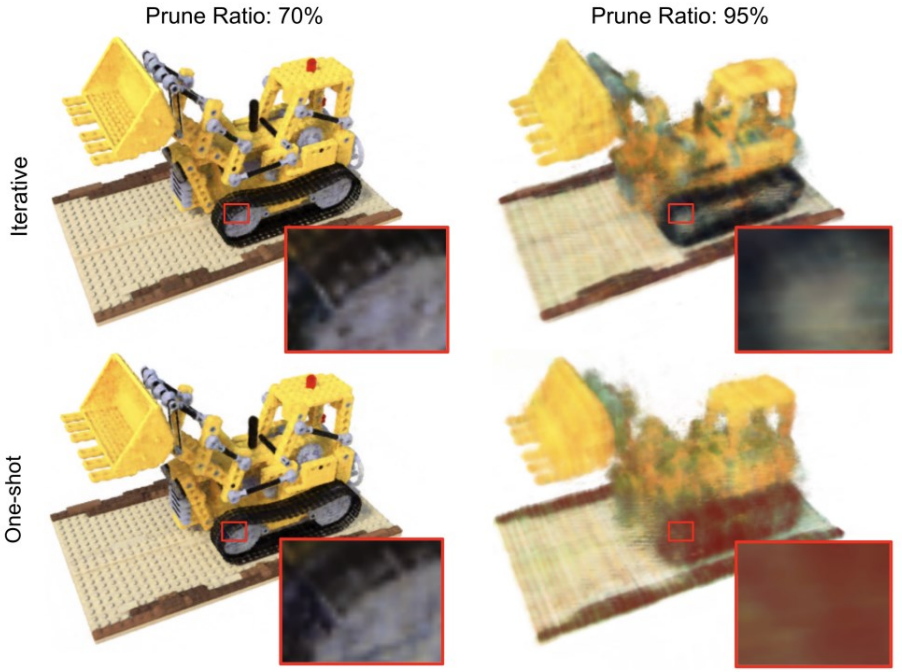

Figure 8. The one-shot and iterative pruning methods applied to prune 70% and 95% of the fine network.

4.4. One-shot versus iterative pruning

We observe that iterative pruning is typically better than one-shot pruning, particularly when a significant portion of the parameters are pruned, such as 90% or 95%. For instance, referring to Table 1, one-shot pruning 90% of the fine network significantly impacts generation quality, yielding a maximum PSNR value of only 22.88. However, switching to the iterative method increases the maximum PSNR value by almost 4, to 26.69. As such, the iterative method proves to be much better when the sparsity level is high.

However, when the sparsity level is low, the difference is marginal, if at all. For instance, one-shot pruning 50% of the coarse network already gives a high maximum PSNR value of 36.31, so iterative pruning improves this result only by 0.26, to 36.57. In fact, sometimes iterative pruning performs worse. One-shot pruning 50% of the fine network also gives a high maximum PSNR value, 35.71, but pruning the same percentage with the iterative method yields only a value of 34.91.

We show the qualitative difference between the two methods in a visual comparison (Figure 8). We use the two methods to prune 70% and 95% of the fine network, and show the final generated images after all pruning and retraining are complete.

We observe that when 70% of the parameters are pruned, the final generated images produced by one-shot and iterative pruning only have marginal qualitative differences. Iterative pruning yields slightly clearer results. However, when 95% of the parameters are pruned, the final generated images produced by the two methods have very different quality. While both images are much blurrier than their 70% counterparts, the one generated by the iterative method is much more detailed. Furthermore, the colors in the one created by the iterative method is significantly more realistic: for instance, the bulldozer's tracks are correctly black, while in the image created by the one-shot method, the tracks are red. This comparison visually proves that, when the sparsity level is high, the iterative method yields higher quality images than the one-shot method given the same pruning percentage, though not as much when the sparsity level is low.

DOI: 10.54254/2755-2721/45/20241047

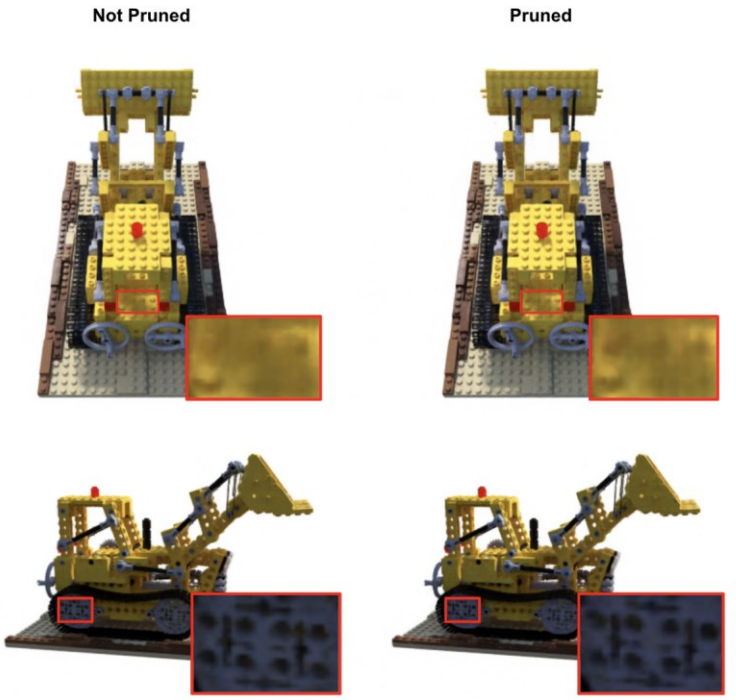

Figure 9. The iterative pruning method is used to prune 90% of the coarse network and 50% of the fine network simultaneously.

4.5. Combining the pruning of the coarse and fine networks

In the above experiments we pruned the coarse and fine networks separately. However, our method, CFP, performs better when we combine the pruning for both networks. In the below experiment, we prune 90% of the coarse network and 50% of the fine network simultaneously using the iterative method. We chose the respective percentages for the two networks because they maintain good generation quality, and we chose the iterative method over the one-shot method as we previously demonstrate it performs better in most cases. The result of this experiment is shown in Figure 9.

We observe that the images before and after pruning are nearly identical, meaning the generation quality is almost entirely preserved. As such, this experiment demonstrates that both the coarse and fine networks are significantly redundant, with the coarse network more so. With CFP, the coarse network can be pruned to one-tenth of its original capacity, while the fine network can be pruned to a half.

5. Conclusion

This work introduces Coarse-Fine-Pruning, a novel method tailored to enhancing the computation and memory efficiency of Neural Radiance Fields. We identify substantial redundancy in both the coarse and fine networks of NeRF, and our technique employs network pruning to effectively eliminate these superfluities. Our study reveals that the coarse network exhibits a higher tolerance to pruning compared to the fine network. Subsequently, our method prunes varying proportions from the two networks to preserve an optimal generation quality.

Empirical findings suggest that pruning up to 90% of the coarse network and 50% of the fine network does not compromise generation quality. Notably, our results show that iterative pruning outperforms the one-shot method, particularly at high sparsity levels. Our approach is both straightforward and highly adaptable, as it maintains the original NeRF architecture intact. The uncomplicated nature of our method, coupled with its capacity to decrease both memory and computation needs instead of employing a computation-memory trade-off, underscores its significant benefits.

DOI: 10.54254/2755-2721/45/20241047

As pioneers of integrating NeRF and network pruning in a systematic manner, we posit that our work lays the foundation for a fresh trajectory ofx NeRF optimization. The potential of our approach to transform the field of NeRF optimization is promising, and we hope that future research will continue to explore and refine this exciting new direction.

Appendix

We present more results in Tables A1, A2, A3, and A5, corresponding to one-shot and iterative pruning on the coarse and fine networks. In one-shot pruning, we prune the network and retrain the pruned network for 20,000 steps. For iterative pruning, in the first 10,000 steps we prune every 1,000 steps, each time pruning one-tenth of the total pruning percentage. After that, we fine-tune the pruned network for 20,000 steps. For both iterative and one-shot pruning, retraining more does not help with the final PSNR value because the networks have already converged.

In the two tables corresponding to iterative pruning (Tables A3 and A4), the PSNR value decreases in the first 10,000 steps because this is when pruning occurs. In addition, step 0 always has the same PSNR value, 37.02, because pruning hasn't started.

Table A1. One-shot pruning on the coarse network.

Weights Pruned(%)	Fine-Tuning Step				
	0	5000	10000	15000	20000
50	34.00	35.90	35.79	36.31	35.99
60	33.44	35.79	35.68	35.99	36.12
70	30.15	34.91	34.61	35.26	35.39
80	29.99	34.55	34.75	35.14	35.10
90	27.10	33.98	34.50	34.79	34.61
95	26.69	32.77	32.99	32.30	33.04

Table A2. One-shot pruning on the fine network.

Weights Pruned(%)	Fine-Tuning Step				
	0	5000	10000	15000	20000
50	22.86	34.48	35.55	35.29	35.71
60	18.12	33.79	34.66	34.42	34.63
70	15.98	28.19	30.69	31.05	31.01
80	12.01	21.22	24.75	26.44	26.60
90	7.67	18.81	22.76	22.74	22.88
95	5.53	13.99	18.78	18.76	19.05

DOI: 10.54254/2755-2721/45/20241047

Table A3. Iterative pruning on the coarse network.

Weights Pruned(%)	Fine-Tuning Step						
	0	5000	10000	15000	20000	25000	30000
50	37.02	36.79	34.23	36.16	35.19	36.52	36.57
60	37.02	36.33	33.14	35.70	36.47	36.38	36.53
70	37.02	36.19	32.63	35.60	35.57	35.83	35.90
80	37.02	35.82	31.96	35.07	35.69	35.57	35.60
90	37.02	35.59	31.23	34.74	35.15	35.13	35.25
95	37.02	35.01	30.77	33.91	34.21	34.29	34.18

Table A4. Iterative pruning on the fine network.

Weights Pruned(%)	Fine-Tuning Step						
	0	5000	10000	15000	20000	25000	30000
50	37.02	35.47	26.88	34.10	34.77	34.91	34.80
60	37.02	35.11	26.07	33.99	34.06	34.33	34.35
70	37.02	33.44	24.19	32.01	31.87	32.00	33.32
80	37.02	32.28	23.91	27.07	28.10	27.94	28.00
90	37.02	30.99	20.34	25.48	26.63	26.69	26.51
95	37.02	28.35	16.80	22.57	23.71	23.64	23.88

We observe that for both one-shot and iterative pruning, the network typically converges 10,000 steps before the end of retraining as the PSNR value typically does not improve significantly after that. In addition, sometimes the PSNR value decreases despite further fine-tuning because fine-tuning may be unstable.

References

[1] Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T., et al.: Local implicit grid representations for 3d scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6001–6010 (2020).

[2] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4460– 4470 (2019).

[3] Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learn- ing continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 165–174 (2019).

DOI: 10.54254/2755-2721/45/20241047

[4] 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).
[5] Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14346–14355 (2021).
[6] Ayinde, B.O., Inanc, T., Zurada, J.M.: On correlation of features extracted by deep neural networks. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2019). IEEE.
[7] Sankararaman, K.A., De, S., Xu, Z., Huang, W.R., Goldstein, T.: The impact of neural network overparameterization on gradient confusion and stochastic gra- dient descent. In: International Conference on Machine Learning, pp. 8469–8479 (2020). PMLR.
[8] Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015).
[9] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016).
[10] LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. Advances in neural information processing systems **2** (1989).
[11] Hassibi, B., Stork, D.G., Wolff, G.J.: Optimal brain surgeon and general network pruning. In: IEEE International Conference on Neural Networks, pp. 293–299 (1993). IEEE.
[12] Liu, L., Gu, J., Zaw Lin, K., Chua, T.-S., Theobalt, C.: Neural sparse voxel fields. Advances in Neural Information Processing Systems **33**, 15651–15663 (2020).
[13] Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radi- ance fields with thousands of tiny mlps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14335–14345 (2021).
[14] Yu, X., Liu, T., Wang, X., Tao, D.: On compressing deep models by low rank and sparse decomposition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7370–7379 (2017).
[15] Qiu, J., Chen, C., Liu, S., Zhang, H.-Y., Zeng, B.: Slimconv: Reducing chan- nel redundancy in convolutional neural networks by features recombining. IEEE Transactions on Image Processing **30**, 6434–6445 (2021).
[16] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).
[17] Liu, Y., Chen, J., Liu, Y.: Dccd: Reducing neural network redundancy via distillation. IEEE Transactions on Neural Networks and Learning Systems (2023).
[18] Lemaire, C., Achkar, A., Jodoin, P.-M.: Structured pruning of neural networks with budget-aware regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9108–9116 (2019).
[19] Dutta, D.: Optimization of neural networks by reducing floating-point precision of weights during training (2021).
[20] Blalock, D., Gonzalez Ortiz, J.J., Frankle, J., Guttag, J.: What is the state of neural network pruning? Proceedings of machine learning and systems **2**, 129–146 (2020).
[21] Wang, Z., Li, C., Wang, X.: Convolutional neural network pruning with struc- tural redundancy reduction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14913–14922 (2021).
[22] Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018).
[23] Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270 (2018).
[24] Gale, T., Elsen, E., Hooker, S.: The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574 (2019).

DOI: 10.54254/2755-2721/45/20241047

[25] Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: Deepview: View synthesis with learned gradient descent. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2367–2376 (2019).
[26] Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoor- thi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019).
[27] Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018).
[28] Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K.M., Tagliasacchi, A.: Derf: Decomposed radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14153–14161 (2021).
[29] Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Bak- ing neural radiance fields for real-time view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5875–5884 (2021).
[30] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real- time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5752–5761 (2021).
[31] Lindell, D.B., Martel, J.N., Wetzstein, G.: Autoint: Automatic integration for fast neural volume rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14556–14565 (2021).
[32] Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A., Steinberger, M.: Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In: Computer Graphics Forum, vol. 40, pp. 45–59 (2021). Wiley Online Library.

Acknowledgments
I express sincere gratitude towards my parents. My mother, who gave me enormous emotional support throughout the writing process, and my father, who frequently discussed computer sciences with me and inspired me to tackle this research project. I would like to thank my supervising teacher, Ms. Ward, whose teachings influenced the direction of this project, and also my advisors, who gave me suggestions and guidance throughout my journey.