Mobile NeRF: Real-Time On-device Neural Radiance Field

KEWAL MISHRA, Worcester Polytechnic Institute
SWAPNEEL WAGHOLIKAR, Worcester Polytechnic Institute
SHOUNAK NAIK, Worcester Polytechnic Institute

Abstract: Through the use of implicit neural representations to describe three-dimensional scenes, Neural Rendering Fields, or NeRF, have shown remarkable abilities in novel view synthesis. But volumetric rendering's intrinsic computing requirements makes NeRF unsuitable for real-time applications on hardware with limited resources, especially mobile devices. Despite the fact that several approaches have been developed for reducing NeRF's latency problems[Agarap 2018], many still rely on expensive GPUs or large amounts of storage capacity, neither of which are available on mobile devices.

This project explores an effective neural rendering technique designed for mobile device real-time operation in response to these difficulties. We explored some new network architectures optimized for mobile operation, based on the idea of neural light fields (NeLF)[Attal et al. 2022] , which simplifies rendering by using one forward pass per ray for pixel color prediction. On mobile devices, the proposed model shows real-time inference for both artificial and real-world scenarios. An interesting rendering latency of 20 FPS (iPad) was achieved for a real 3D scene.

Moreover, our project explores the use of pruning algorithms to maximise the neural network's storage footprint. Our tests show that rendering quality can be harmed by checkpoint pruning, and it can be recovered a little by using some techniques like iterative pruning.

ACM Reference Format:
Kewal Mishra, Swapneel Wagholikar, and Shounak Naik. 2018. Mobile NeRF: Real-Time On-device Neural Radiance Field. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The field of computer graphics has completely changed with the introduction of Neural Rendering Fields (NeRF), especially in the area of novel-view synthesis. These methods produce high-quality 3D scene renderings by using implicit neural representations. Unfortunately, NeRF's built-in volumetric rendering approach places heavy computational demands on rendering speed and real-time performance[Barron et al. 2021]. This restriction is most noticeable when attempting to use NeRF on hardware with limited resources, such mobile devices, which are lacking of high-end GPUs' computing capacity.

The growing uses of virtual reality and 3D scanning, where smooth, real-time rendering on mobile devices may enable engaging experiences for consumers, are the reason for the importance and interest in tackling this topic. This is a fascinating area to explore since it has the potential to simplify asset generation and rendering by doing away with the requirement for precise mesh, texture, or material parameters[Barron et al. 2022].

We address the issue of real-time neural rendering on mobile devices in this project, with a special focus on novel-view synthesis with NeRF. Having acknowledged the drawbacks of volumetric rendering on mobile platforms, our goal is to go over these challenges and help make neural rendering on devices with limited resources a reality.

Our approach makes use of two essential on-device learning techniques: iterative pruning checkpoints and knowledge distillation using pseudo images. In addition to addressing the computational constraints imposed by mobile devices, these techniques open the door for effective storage utilisation. Unlike previous efforts that frequently depend on expensive GPUs or sacrifice rendering quality in favour of speed, our initiative aims to find a fair compromise. We present a unique network architecture that retains high-resolution rendering for both synthetic and real-world situations while achieving real-time inference and drastically lowering storage needs over earlier approaches.

Our goal in conducting this project is to present a strong case for the implementation of neural rendering on mobile devices, opening the way for applications like interactive 3D object manipulation and virtual try-on. Our approach contributes to the larger objective of enabling sophisticated graphics on resource-constrained devices by straddling the line between effective neural rendering, on-device learning, and storage optimisation.

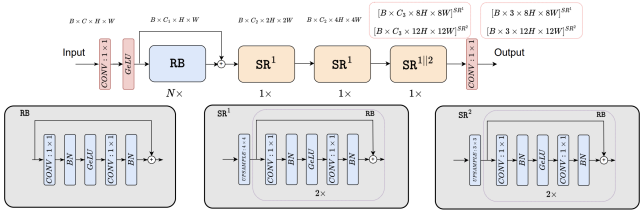

Fig. 1. Overview of MobileR2L Network

2 PROPOSED APPLICATION

We took inspiration from the MobileR2L[Cao et al. 2023] framework, a real-time neural rendering model created with mobile devices in mind, in our attempt to achieve real-time neural rendering on mobile devices. The MobileR2L model is a perfect fit for our project as it effectively strikes a compromise between rendering quality, inference speed, and storage requirements.

2.1 Knowledge Distillation from NeLF Teacher Model

After undergoing extensive training on a variety of settings, the NeLF teacher model can produce high-quality images with just

Authors' addresses: Kewal Mishra, Worcester Polytechnic Institute; Swapneel Wagholikar, Worcester Polytechnic Institute; Shounak Naik, Worcester Polytechnic Institute.

ACM 0730-0301/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

one forward pass for each ray. For real-time applications, when reducing processing demands is crucial, this feature is very helpful. Through the use of the NeLF teacher model, we aimed to capture the rich information that it learned throughout its training process, therefore capturing its capacity to produce accurate renderings with efficiency.

The NeLF teacher model was trained on the LEGO dataset to make sure it could handle a range of spatial layouts, texturing, and lighting conditions that were typical of real-world situations. The resilience and adaptability of the learned representations of the NeLF teacher model were enhanced by the use of such a diverse and readily available dataset.

The NeLF teacher model was used to create pseudo visuals as part of the knowledge distillation process. The core of the complex knowledge represented in the teacher's model was captured by these pseudo images, which functioned as a surrogate dataset. The teacher model's comprehension of scene representations, lighting interactions, and view synthesis was captured in the pseudo images, which served as a condensed yet rich training set for our lightweight student model, MobileR2L.

By using the pseudo images for training, the MobileR2L student model was able to pick up and replicate the insightful knowledge extracted from the NeLF teacher model. This procedure successfully transmitted the capability of low-complexity real-time scene rendering, which is appropriate for mobile device deployment.

To do this, we adopted the NeLF teacher model's knowledge distillation in an attempt to balance rendering quality with model efficiency. By using this method, we were able to drastically lower the number of computations and input parameters needed for inference, which opened up the possibility for real-time neural rendering on mobile devices without sacrificing the visual quality of the pictures that were produced. Our MobileR2L student model is now well-suited for real-world applications when computing resources are few, thanks in large part to the knowledge extracted from the NeLF instructor model.

2.2 Hyperparameter Considerations

Super-Resolution (SR) modules for high-resolution rendering and an effective backbone comprise our network architecture, which closely resembles the MobileR2L concept. The efficient backbone is modelled after residual blocks from the original R2L model and consists of 60 convolutional layers with activation and normalization algorithms. Notably, for improved optimization and mobile device compatibility, we swapped out the fully connected (FC) layers in the backbone for convolutional (CONV) layers.

We used SR modules after the backbone to overcome memory limitations on mobile devices and provide high-resolution images with lower latency. With the help of these modules, we may only send a portion of the rays, upsampling the output to create a picture with high resolution. Three SR modules and sixty CONV layers make up the efficient backbone of our model, which we refer to as D60-SR3.

We sought to achieve a considerable reduction in the number of input parameters and calculations by integrating knowledge distillation from a NeLF teacher model with this adaption of the MobileR2L architecture. This reduced the size of the model and made it easier to make decisions in real time on mobile devices with limited resources. Our dedication to tackling the difficulties of neural rendering on mobile platforms while preserving rendering quality and efficiency is demonstrated by the selection of our customized architecture and knowledge distillation method.

3 METHODOLOGY

A key component of our process was building a reliable and effective pipeline, which presented a number of problems in the development of an on-device deep neural network for real-time neural rendering. The process of building this pipeline was quite meticulous, and much of our work was focused on solving different technological problems.

3.1 Knowledge Distillation with AWS Cluster

We first addressed the critical process of knowledge distillation by using the checkpoint of a pre-trained Neural Light Field (NeLF) model to provide pseudo data that we needed to train our MobileR2L student model. The computational intensity of this complex procedure made it necessary to use specialised libraries like Tiny Cudann in order to efficiently extract the information contained in the NeLF teacher model.

We utilised the power of a powerful cloud computing infrastructure, an AWS cluster, to expedite this process of knowledge distillation. Using GPU instances in the AWS environment was crucial for assuring effective installs and faster computations during the complex pseudo data generation process. Using GPU instances greatly accelerated the amount of processing power needed for the computationally demanding parts of knowledge distillation.

It required careful attention to detail to set up and configure the environment for this task. A special attention to the integration of Tiny Cudann and other dependencies was paid to make sure that everything ran smoothly and in sync inside the AWS cluster. This complex setup was necessary to maximise GPU parallelization efficiency and optimize the distillation process.

3.2 Student Training on WPI's Turing Cluster

With the pseudo data successfully generated through the knowledge distillation process, the subsequent critical phase involved the training of our MobileR2L student model. This training phase was indispensable for enabling the model to learn from the informative dataset derived from the NeLF teacher model's pseudo images.

We needed to complete this training process as quickly and effectively as possible, therefore we used the powerful Turing Cluster computer at WPI University. This powerful GPU-enhanced high-performance computing cluster was crucial in greatly accelerating the training process. Faster convergence and model refining were made possible by the parallelization of calculations made possible by the availability of these potent GPUs.

One reason for the accelerated training schedule was the deliberate use of the Turing Cluster and its dedicated GPU hardware. The training procedure, which usually took a while, was effectively finished in around one and a half days. This expedited schedule

was evidence of the Turing Cluster's optimal use of its parallel processing power and computing resources.

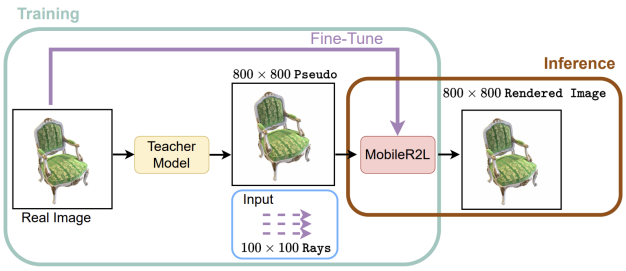

Fig. 2. Training and Inference Pipeline.

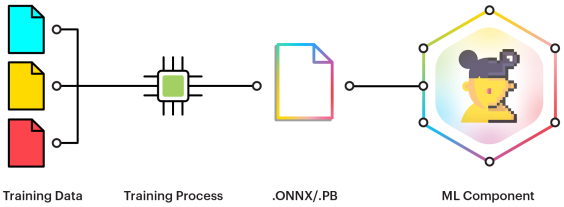

Fig. 3. Lens Studio Pipeline.

3.3 Application Development with Lens Studio

The following critical phases required getting the MobileR2L student model ready for on-device deployment and developing an application for real-time rendering after the model had been successfully trained. To guarantee compliance with on-device deployment, this complex approach involved converting the trained model's checkpoint to the ONNX format. Next, an application was developed using Lens Studio[Len [n. d.]] utilising SnapML libraries.

This phase started with the checkpoint of the trained student model being converted to the ONNX format. Because of its adaptability and compatibility, this format was selected to enable on-device distribution and smooth interaction with a variety of systems. The conversion procedure made sure that the learned capabilities of the trained model could be conveniently used on a mobile device.

Then the application started to be developed with Lens Studio[NeR [n. d.]], a robust platform for making augmented reality experiences. Lens Studio was the perfect setting for rendering the Neural Rendering Fields (NeRF) model on Snapchat because of its features, especially its interaction with SnapML libraries. In order to provide a seamless and optimal integration of the trained model into the Lens Studio environment, the development process necessitated careful consideration of the distinct characteristics and requirements of SnapML.

One of the main goals of the application development process was to optimize the model for mobile device real-time performance. This required optimizing rendering settings, fine-tuning parameters, and putting in place other adjustments to make sure the neural rendering process could run smoothly within the limitations of mobile technology. The MobileR2L model's integration with Lens Studio[Sna [n. d.]] made it possible to create an immersive augmented reality experience for Snapchat users. Users may now immediately experience real-time neural rendering on their mobile devices thanks to the application, which was created to take advantage of SnapML's capabilities for effective model inference on the device.

3.4 Inference on iPad (Apple M1v Pro Chip)

We installed the model on an iPad with the Apple M1v Pro processor after the program was complete. Selecting the right hardware was essential to getting the best inference performance. Neural rendering might be carried out effectively and with high-quality output if the model was deployed on a mobile device, particularly one with a potent processor like the M1v Pro.

4 RESULTS

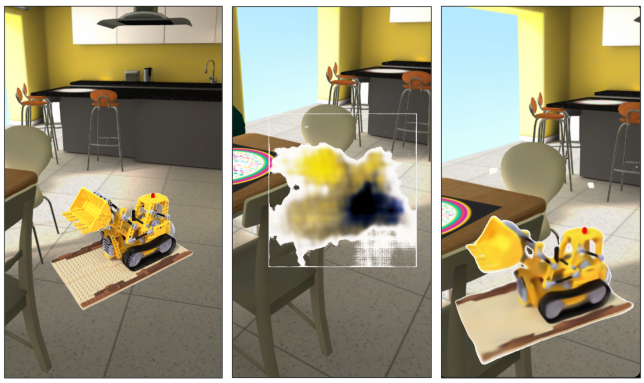

Fig. 4. (a) Result from the model compression (b) Result with 50% pruned model (c) Result from the 30% pruned fine-tuned model.

4.1 Performance Analysis

This section provides an analysis of the performance characteristics of our on-device learning approach, highlighting the performance of iterative pruning and knowledge distillation methods. Figure 4 shows results from these methods.

4.2 Knowledge Distillation with Pseudo Images

Knowledge distillation using pseudo images has proven to be highly effective, with the distilled model preserving the core competencies of the larger parent model. The utilization of pseudo images for distillation allows the condensed model to retain a robust feature set, enabling it to perform well. The resulting model, with a size of 13 MB, strikes an optimal balance between size and performance, making it well-suited for on-device deployment where resources are limited.

4.3 Model Pruning and Trade-offs

Our approach involved iterative pruning at two levels: 50% and 30%, followed by further fine-tuning to recover performance losses typically associated with the pruning process. The 50% pruned model, while being the smallest at 8 MB, exhibited a performance trade-off that was noticeable in compute-intensive tasks. Although the reduction in size was significant, this came at the cost of reduced accuracy and increased latency.

Conversely, the 30% pruned model, sized at 9 MB, presented a more favorable trade-off. The modest reduction in size from the knowledge distillation baseline allowed for a lesser impact on performance, maintaining closer fidelity to the original model's capabilities. This level of pruning, complemented by subsequent fine-tuning, proved to be a viable approach for on-device deployment, offering a more balanced compromise between resource utilization and functional performance.

In summary, our findings underscore the intricate balance between model size and performance in on-device machine learning applications. While knowledge distillation with pseudo images emerged as the superior technique in preserving performance, the pruning strategies provided valuable insights into the trade-offs required for achieving a minimal model footprint.

4.4 Comparison Analysis

In our comparison analysis, we evaluate the performance and efficiency of various model optimization strategies applied to the MobileNeRF[Chen et al. 2022] framework. We focus on three main metrics: the size of the model, its latency, and RAM usage.

From Table 1, Knowledge Distillation stands out for significantly reducing the model size from 125 MB in the original paper to 13 MB, with a marginal impact on latency, achieving 20 frames per second (FPS), and a slight reduction in RAM usage. This indicates a high level of efficiency in model compression, achieving substantial size reduction with minimal performance loss.

Conversely, the 30% Pruned + finetuned model demonstrates the effectiveness of pruning techniques, reducing the model size even further to 9 MB. This is accompanied by an improvement in latency to 22 FPS, suggesting a faster model despite its reduced size. The RAM usage for this model is slightly decreased to 680 KB, representing a negligible trade-off given the benefits of reduced size and increased speed.

The findings reveal that while Knowledge Distillation with pseudo images strikes an optimal balance between size reduction and performance, model pruning—particularly when combined with fine-tuning—offers the most significant reduction in size and an increase in processing speed, albeit with potential performance trade-offs. Aggressive pruning strategies, such as a 50% reduction, must be carefully considered due to their potential impact on the model's accuracy and generalizability.

Ultimately, the decision to employ Knowledge Distillation or model pruning should be guided by the specific requirements of the application. If maintaining performance is paramount, Knowledge Distillation is the superior choice. However, if minimizing model size and enhancing speed are more critical, pruning is the more appropriate strategy.

Metric	MobileNeRF	Knowledge Dist	30% Pruned
Model Size (MB)	125	13	9
Latency (FPS)	-	20	22
RAM (KB)	-	700	680

Table 1. Comparison of Model Performance

Note: The '-' symbol indicates metrics not implemented for MobileNeRF or not reported in the original paper, as these metrics are device-specific.

5 DISCUSSION, CHALLENGES AND LIMITATIONS:

Our study presents a comprehensive examination of model optimization techniques applied to MobileNeRF, focusing on the trade-offs between model size, latency, and memory usage.

5.1 Discussion

The application of Knowledge Distillation with pseudo images has proven to be an effective strategy for size reduction while maintaining a high level of performance. The distilled models retain a significant portion of the original model's capabilities, suggesting that this technique is well-suited for scenarios where performance cannot be compromised.

Model Pruning, particularly when combined with fine-tuning, offers a viable solution for applications where storage and speed are at a premium. The 30% pruned model strikes a balance, offering a reduced footprint while preserving a degree of performance. However, when pruning is increased to 50%, there is a notable decline in accuracy, highlighting the delicate balance between size reduction and model capability.

5.2 Challenges

Many difficulties were encountered while constructing this extensive pipeline, especially when setting up the AWS cluster, streamlining the training procedure on the Turing Cluster, and incorporating the model into a Lens Studio real-time application. Iterative testing, debugging, and teamwork were necessary to overcome these obstacles. Our project's timetable was mostly consumed by the pipeline's creation, which required painstaking attention to detail from knowledge distillation to on-device deployment. The intricacy of the pipeline highlighted how crucial each stage is to the effective implementation of a real-time neural rendering model on mobile devices.

5.3 Limitations

While our findings contribute valuable insights into model optimization for on-device neural networks, several limitations must be acknowledged:

- The metrics for the original MobileNeRF were not available, limiting our ability to compare across all dimensions of performance. Future studies could address this gap by implementing these metrics on standardized devices to ensure comparability.
- Our evaluation was constrained to specific hardware configurations. The generalizability of our results to other devices with different specifications remains to be tested.

- The pseudo images used for Knowledge Distillation were generated from a limited dataset. The model's performance on a broader range of real-world images could differ from our reported results.
- The pruning percentages were chosen based on heuristics and prior literature. An exhaustive search over pruning percentages might yield a more finely tuned model, potentially offering better performance.
- The training computation is highly resource-intensive, requiring approximately 1.5 days of training on a single GPU for one object. This involves the computation for both the teacher and the student models, which may be prohibitive for researchers with limited access to computational resources.
- The realism of the models may be insufficient for certain applications, as the current approach does not incorporate plane fitting or other advanced computer vision techniques. This could affect the applicability of our models in scenarios where high fidelity to real-world objects is crucial.

In conclusion, the trade-offs identified in this study are essential for guiding future work on neural network optimization for mobile devices. As the field advances, further research is needed to explore these techniques in diverse operational contexts and with varying model architectures.

6 CONCLUSION AND FUTURE WORK

In conclusion, this work has presented a detailed analysis of optimization techniques for the MobileNeRF framework, with a focus on Knowledge Distillation and Model Pruning strategies. Our investigation reveals that while Knowledge Distillation provides a substantial reduction in model size with minimal performance loss, Pruning offers additional size and speed benefits but with certain trade-offs in performance.

Through this study, we have identified critical insights that not only aid in understanding the balance between model efficiency and effectiveness but also illuminate the practical challenges associated with deploying neural network models on mobile devices. Despite the limitations related to training computation demands and model realism, our findings pave the way for future research to refine these optimization strategies further.

Future work must continue to address these challenges, aiming to reduce the computational overhead of model training and enhance the realism of the optimized models, thereby extending the practical applicability of MobileNeRF and similar frameworks in real-world mobile applications. Looking ahead, several avenues appear promising for extending the capabilities of our work:

- **Custom Object 3D Model:** Development of tailored 3D models for custom objects for specific use cases.
- **Top Layer Fine-tuning:** Implementing fine-tuning of only the top layers of the network to reduce the overall training time while maintaining the integrity of the learned features.
- **Pruning-aware Fine-tuning:** Exploring pruning-aware training methods that anticipate the effects of pruning and adjust the training process to compensate for potential losses in performance.
- **Model Compression Techniques:** Investigating other model compression techniques such as quantization, which could provide additional performance benefits and further reduce the computational footprint of the models.

7 CONTRIBUTIONS

This research was made possible through the collaborative efforts of our team, where responsibilities were allocated to leverage individual expertise effectively. The setup and development of the pipeline, as well as the network configuration and optimization, were adeptly handled by Shounak Naik and Swapneel Wagholikar. The Lens Studio setup for viewing the asset generated from the model and seamless performance was handled by Kewal Mishra.

A critical component of our project was the integration of our model with SnapML within Lens Studio, ensuring that the assets generated by our optimized MobileNeRF model were not only viewable but also interacted with high performance and fidelity in a real-world augmented reality context. This effort was handled by Kewal Mishra.

REFERENCES

[n. d.]. Lens studio. https://ar.snap.com/en-US/lens-studio. 1.

[n. d.]. Nerf-factory. https://github.com/kakaobrain/NeRF-Factory. 5.

[n. d.]. Snap ml. https://docs.snap.com/lens-studio/references/guides/lens-features/machine-learning/ml-overview. 1.

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). (2018). arXiv:arXiv:1803.08375 7.

Benjamin Attal, Jia-Bin Huang, Michael Zollhofer, Johannes Kopf, and Changil Kim. 2022. Learning neural light fields with ray-space embedding. In *CVPR*. 2, 3.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5855–5864. 2, 7.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5470–5479. 2.

Junli Cao, Huan Wang, Pavlo Chemerys, Vladislav Shakhrai, Ju Hu, Yun Fu, Denys Makoviichuk, Sergey Tulyakov, and Jian Ren. 2023. Real-Time Neural Light Field on Mobile Devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8328–8337.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2022. MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. *arXiv preprint arXiv:2208.00277* (2022).