NAS-NeRF: Generative Neural Architecture Search for Neural Radiance Fields

Saeejith Nair[1] **Yuhao Chen**[1] **Mohammad Javad Shafiee**[1,2,3] **Alexander Wong**[1,2,3]
[1] Vision and Image Processing Research Group, University of Waterloo
[2] Waterloo Artificial Intelligence Institute, Waterloo, ON
[3] DarwinAI Corp., Waterloo, ON
{smnair, yuhao.chen1, mjshafiee, a28wong}@uwaterloo.ca

Abstract

Neural radiance fields (NeRFs) enable high-quality novel view synthesis, but their high computational complexity limits deployability. While existing neural-based solutions strive for efficiency, they use one-size-fits-all architectures regardless of scene complexity. The same architecture may be unnecessarily large for simple scenes but insufficient for complex ones. Thus, there is a need to dynamically optimize the neural network component of NeRFs to achieve a balance between computational complexity and specific targets for synthesis quality. We introduce NAS-NeRF, a generative neural architecture search strategy that generates compact, scene-specialized NeRF architectures by balancing architecture complexity and target synthesis quality metrics. Our method incorporates constraints on target metrics and budgets to guide the search towards architectures tailored for each scene. Experiments on the Blender synthetic dataset show the proposed NAS-NeRF can generate architectures up to 5.74× smaller, with 4.19× fewer FLOPs, and 1.93× faster on a GPU than baseline NeRFs, without suffering a drop in SSIM. Furthermore, we illustrate that NAS-NeRF can also achieve architectures up to 23× smaller, with 22× fewer FLOPs, and 4.7× faster than baseline NeRFs with only a 5.3% average SSIM drop. Our source code is also made publicly available[1].

1 Introduction

Neural radiance fields (NeRFs) [1] can enable photorealistic novel view synthesis of complex 3D scenes by using multilayer perceptrons (MLPs) to represent continuous volumetric radiance and density fields. However, converging to a sufficiently high-resolution representation is computationally demanding, as NeRF requires hundreds of millions of costly neural network queries per rendered image. However, converging to a sufficiently high-resolution representation is computationally demanding, as NeRF requires hundreds of millions of costly neural network queries per rendered image. For example, Mildenhall et al. report that training NeRF [1] on their Blender synthetic dataset takes 640k rays per image, with each ray requiring 256 sampled points that need to be passed through the fields in order to accumulate colour and density. This results in over 150 million network queries per rendered image, which severely limits deployability, especially on resource constrained platforms.

Recent advancements have succeeded in enhancing the efficiency of NeRF by employing techniques like caching [2, 3], baking [4], tensor decomposition [5], efficient sampling [6], spatial decomposition [7, 8, 2] or multi-resolution hash encodings [3]. However, these methods typically adopt one-size-fits-all architectures, irrespective of scene complexity and target metrics, making them impractical for realistic deployment constraints.

[1]Project website: https://saeejithnair.github.io/NAS-NeRF

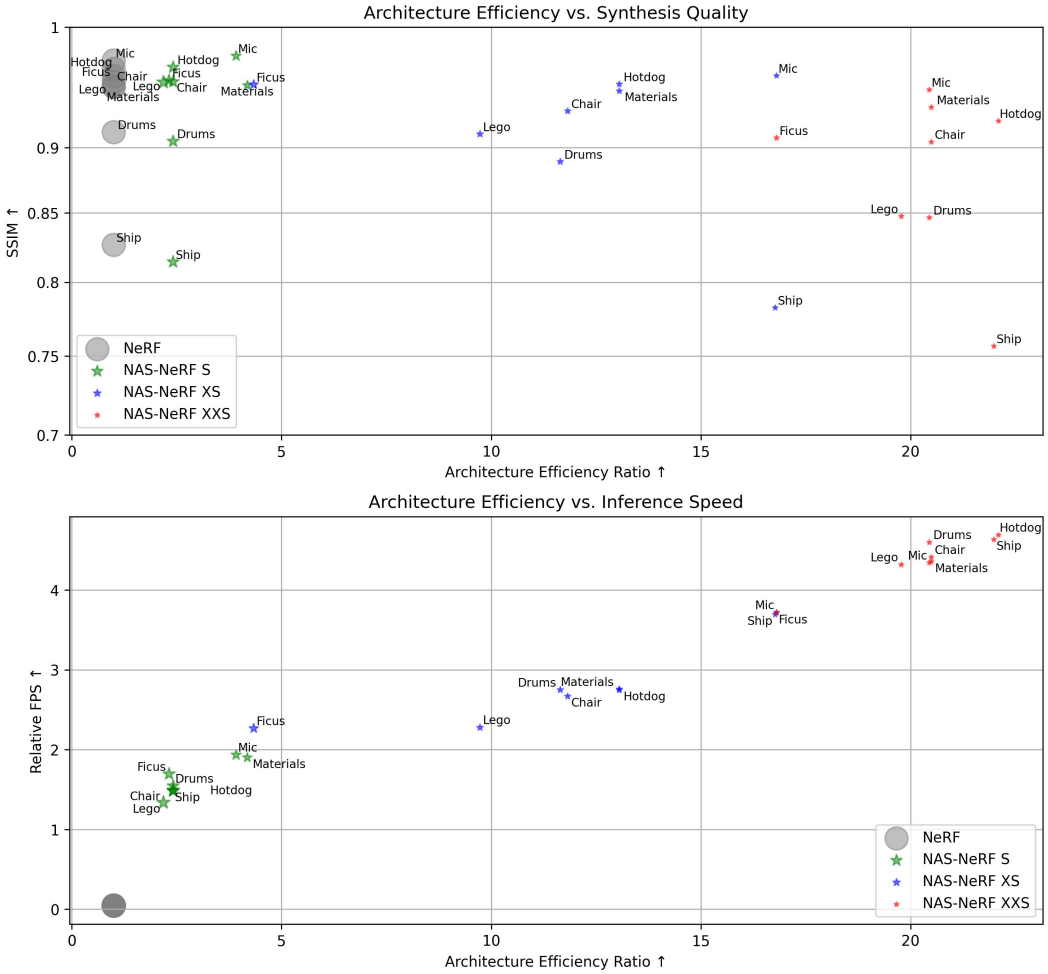

Figure 1: Architecture efficiency ratio (a measure of generated architecture FLOPs relative to baseline NeRF) vs. (top) synthesis quality (SSIM) and (bottom) inference speed; size $\propto$ parameter count.

We propose NAS-NeRF, a generative architecture search strategy [9] that tailors scene-specific architectures. Using the NAS-NeRF field cell as our core, we modularize the neural radiance field representation in terms of a cell [10]. This enables us to reduce neural network complexity to meet deployment constraints without sacrificing synthesis quality. Experiments demonstrate that NAS-NeRF generates architectures up to 23$\times$ smaller, 22$\times$ fewer FLOPs, and 4.7$\times$ faster than baseline NeRFs with only a 5.3% average SSIM drop across scenes on the Blender synthetic dataset [1].

The remainder of this paper is organized as follows. Section 2 describes the methodology behind the creation of the proposed NAS-NeRF via generative neural architecture search, as well as a description of the resulting NAS-NeRF architectures. Section 3 describes the dataset used in this study, the training and evaluation setup, as well as the experimental results and complexity comparisons.

2 Methods

2.1 NAS-NeRF Field Cell

The NAS-NeRF field cell modularizes the classic NeRF field [1] into stages to enable efficient architecture search. The classic NeRF field maps 3D spatial coordinates $\mathbf{x}$ and 2D viewing directions $\mathbf{d}$ to volume density $\sigma(\mathbf{x})$ and view-dependent emitted radiance $c(\mathbf{x}, \mathbf{d})$. This mapping is learned by training an 8-layer density estimator MLP and smaller single layer radiance estimator MLP. We focus

our architecture search on the density estimator since it contributes the most parameters. As shown in Figure 2, we decompose the density estimator into 3 stages with configurable parameters:

1. D_1 fully connected layers with C_1 channels
2. $D_2 = 1$ fully connected layer with C_2 channels and a skip connection concatenating to the input
3. D_3 fully connected layers with C_3 channels

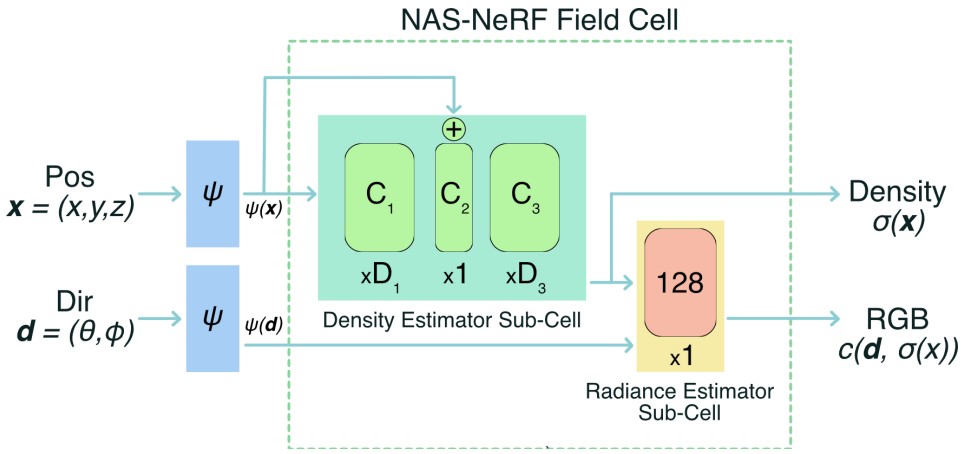

Figure 2: High level overview of the NAS-NeRF field cell architecture. The NAS-NeRF pipeline comprises the original NeRF components [1], utilizing two NAS-NeRF field cells for coarse and fine hierarchical sampling. Each cell is parameterized by depth stages (D_1, D_3) and channels (C_1, C_2, C_3), where ψ denotes the positional encoder for input coordinates or viewing directions. The nature of our parameterization ensures that the NAS-NeRF field cell can be plugged into most other NeRF methods, as our optimizations revolve entirely around the core network architectures.

Decomposing the networks for neural radiance fields into standalone blocks enables reformulating NeRF architecture search as learning specialized cells, with the NeRF field acting analogously to an optimizable cell [10]. Furthermore, this allows viewing the overall NAS-NeRF architecture as two configurable field cells in series - one coarse field cell to learn global structure, and one fine field cell to learn high-frequency details. This modular parameterization focuses optimization on the core NeRF network while decoupling peripheral components like samplers or encoders. Our search strategy can thus function as a plugin applied to various NeRF methods by optimizing the architecture of the density estimator sub-cell.

2.2 Generative Neural Architecture Search

To discover NAS-NeRF architectures that balance efficiency and novel view synthesis quality, we employ a generative neural architecture search approach based on generative synthesis [9]. Given operational constraints like compute budgets and target rendering metrics, generative synthesis executes an architectural exploration process to discover highly tailored architectures fitting the constraints. This discovery process can be formulated as a constrained optimization problem:

$$\mathcal{G} = \max_{\mathcal{G}} \mathcal{U}(\mathcal{G}(s)) \quad \text{subject to} \quad 1_r(G(s)) = 1, \;\; \forall \in \mathcal{S}. \tag{1}$$

where the underlying objective is to learn an expression $\mathcal{G}(\cdot)$ that, given seeds $\{s|s \in S\}$, can generate network architectures $\{N_s|s \in S\}$ that maximizes a universal performance metric U (e.g., [11]) while adhering to operational constraints set by the indicator function $1_r(\cdot)$. This constrained optimization is solved iteratively through a collaboration between a generator G and an inquisitor I which inspects the generated network architectures and guides the generator to improve its generation performance towards operational requirements (see [9] for details).

Although the NAS-NeRF search space is flexible, we enforce design constraints through $1_r(\cdot)$ in Eq. 1 to achieve the desired balance between i) accuracy, ii) architectural complexity, and iii)

Scene	Architecture	Coarse Field		Fine Field		FPS	Parameters (M)	FLOPs (G)
		Depths	Channels	Depths	Channels	(Speedup)		
	NeRF	8 ×	256	8×	256	1×	1.09 (1×)	574.14 (1×)
Chair	NAS-NeRF S	[2, 1, 1]	[9, 11, 12]	[3, 1, 2]	[200, 207, 214]	1.49×	0.32 (3.46×)	237.57 (2.42×)
Chair	NAS-NeRF XS	[2, 1, 1]	[12, 12, 12]	[3, 1, 2]	[53, 57, 61]	2.67×	0.08 (14.33×)	48.56 (11.82×)
Chair	NAS-NeRF XXS	[2, 1, 1]	[9, 11, 12]	[2, 1, 1]	[16, 18, 20]	4.41×	0.05 (21.92×)	28.01 (20.49×)
Drums	NAS-NeRF S	[2, 1, 1]	[9, 11, 12]	[3, 1, 2]	[200, 207, 214]	1.55×	0.32 (3.46×)	237.57 (2.42×)
Drums	NAS-NeRF XS	[2, 1, 1]	[20, 20, 20]	[3, 1, 2]	[53, 57, 61]	2.75×	0.08 (13.81×)	49.29 (11.65×)
Drums	NAS-NeRF XXS	[2, 1, 1]	[12, 12, 12]	[2, 1, 1]	[16, 18, 20]	4.60×	0.05 (21.82×)	28.08 (20.45×)
Ficus	NAS-NeRF S	[2, 1, 1]	[12, 12, 12]	[3, 1, 2]	[214, 214, 214]	1.70×	0.33 (3.32×)	247.57 (2.32×)
Ficus	NAS-NeRF XS	[2, 1, 1]	[9, 11, 12]	[2, 1, 1]	[167, 174, 180]	2.27×	0.18 (5.99×)	132.33 (4.34×)
Ficus	NAS-NeRF XXS	[2, 1, 1]	[9, 11, 12]	[2, 1, 1]	[33, 36, 39]	3.72×	0.06 (18.94×)	34.17 (16.80×)
Hotdog	NAS-NeRF S	[2, 1, 1]	[9, 11, 12]	[3, 1, 2]	[200, 207, 214]	1.49×	0.32 (3.46×)	237.57 (2.42×)
Hotdog	NAS-NeRF XS	[2, 1, 1]	[12, 12, 12]	[3, 1, 2]	[51, 51, 51]	2.75×	0.07 (15.51×)	43.98 (13.05×)
Hotdog	NAS-NeRF XXS	[2, 1, 1]	[12, 12, 12]	[2, 1, 1]	[9, 11, 12]	4.69×	0.05 (23.05×)	25.98 (22.10×)
Lego	NAS-NeRF S	[2, 1, 1]	[100, 104, 109]	[3, 1, 2]	[214, 214, 214]	1.34×	0.39 (2.83×)	262.61 (2.19×)
Lego	NAS-NeRF XS	[2, 1, 1]	[12, 12, 12]	[4, 1, 3]	[64, 64, 64]	2.28×	0.09 (12.19×)	58.98 (9.73×)
Lego	NAS-NeRF XXS	[2, 1, 1]	[16, 18, 20]	[2, 1, 1]	[20, 20, 20]	4.32×	0.05 (20.64×)	29.03 (19.78×)
Materials	NAS-NeRF S	[2, 1, 1]	[16, 18, 20]	[2, 1, 1]	[180, 180, 180]	1.90×	0.19 (5.74×)	137.17 (4.19×)
Materials	NAS-NeRF XS	[2, 1, 1]	[12, 12, 12]	[3, 1, 2]	[51, 51, 51]	2.76×	0.07 (15.51×)	43.98 (13.05×)
Materials	NAS-NeRF XXS	[2, 1, 1]	[9, 11, 12]	[2, 1, 1]	[16, 18, 20]	4.36×	0.05 (21.92×)	28.01 (20.49×)
Mic	NAS-NeRF S	[4, 1, 3]	[56, 60, 64]	[2, 1, 1]	[180, 180, 180]	1.93×	0.23 (4.83×)	146.53 (3.92×)
Mic	NAS-NeRF XS	[2, 1, 1]	[9, 11, 12]	[2, 1, 1]	[33, 36, 39]	3.72×	0.06 (18.94×)	34.17 (16.80×)
Mic	NAS-NeRF XXS	[2, 1, 1]	[12, 12, 12]	[2, 1, 1]	[16, 18, 20]	4.34×	0.05 (21.82×)	28.08 (20.45×)
Ship	NAS-NeRF S	[2, 1, 1]	[9, 11, 12]	[3, 1, 2]	[200, 207, 214]	1.48×	0.32 (3.46×)	237.57 (2.42×)
Ship	NAS-NeRF XS	[2, 1, 1]	[12, 12, 12]	[2, 1, 1]	[33, 36, 39]	3.69×	0.06 (18.86×)	34.23 (16.77×)
Ship	NAS-NeRF XXS	[2, 1, 1]	[9, 11, 12]	[2, 1, 1]	[12, 12, 12]	4.63×	0.05 (23.05×)	26.11 (21.99×)

Table 1: The architecture configuration and performance metrics of each generated NAS-NeRF is shown, along with the baseline NeRF [1]. The architecture configurations for GEN-NeRF are listed as $[D_1, D_2, D_3]$ for the depth columns and $[C_1, C_2, C_3]$ for the channels columns. The architecture configuration for the baseline NeRF model is also listed as each field has an MLP with 8 layers and 256 channels. The FPS and Parameters columns show the frames per second and the number of parameters of each model, along with the speedup and architecture efficiency ratio relative to the baseline NeRF architecture.

computational complexity to yield low-footprint, compact NeRF architectures that are tailored for a target performance metric. Specifically, our constraints encourage:

1. Fields with both uniform and variable number of channels across different MLP stages.
2. Fields that uniformly expand such that stages of the MLP get progressively wider.
3. Meeting minimum scene-specific targets T for performance metrics.

Our experiments use SSIM [12] as the performance metric. For each scene, we sample 3 SSIM targets and generate 3 NAS-NeRF variants per scene (XXS, XS, S) that achieve different trade-offs between efficiency and quality.

Determining suitable targets for a new scene is challenging, as we can't know *a priori* what performance can be achieved. We circumvent this using an efficient heuristic which is fast and extensible to any scene or NeRF method. Specifically, we train two boundary architectures A, a maximum size architecture A_{max} and minimum size A_{min} ($> 23.2\times$ fewer parameters than the baseline) for 16k iterations each (this takes 15-60mins on an Nvidia RTX A6000 GPU). Using their SSIM evaluations $SSIM_{min}$ and $SSIM_{max}$, we linearly interpolate targets T_{XXS}, T_{XS}, and T_S at 10%, 50%, and 90% between the extremes. Although SSIM versus size may be non-linear along the true pareto frontier, we find that our heuristic provides a reasonable performance bound quickly. After formulating scene-specific SSIM targets and constraining the search, we then leverage generative synthesis to iteratively discover tailored models across a range of complexities specific to each scene's requirements.

PSNR ↑

Architecture	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
NeRF	**31.259**	**24.607**	**29.086**	34.137	31.319	29.428	31.089	**27.669**
NAS-NeRF S	31.205	24.129	28.382	**34.312**	**31.720**	**29.526**	**31.711**	26.985
NAS-NeRF XS	29.423	23.215	28.100	32.307	28.266	29.135	29.313	24.876
NAS-NeRF XXS	27.756	20.849	24.526	28.921	25.047	27.627	27.887	22.525

SSIM ↑

Architecture	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
NeRF	0.953	**0.912**	**0.958**	0.964	0.950	0.949	0.971	**0.827**
NAS-NeRF S	**0.953**	0.905	0.954	**0.966**	**0.953**	**0.950**	**0.975**	0.814
NAS-NeRF XS	0.929	0.889	0.951	0.951	0.911	0.946	0.958	0.782
NAS-NeRF XXS	0.904	0.847	0.908	0.921	0.848	0.932	0.947	0.757

LPIPS ↓

Architecture	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship
NeRF	0.043	**0.089**	**0.039**	0.039	0.027	0.042	0.033	**0.179**
NAS-NeRF S	**0.041**	0.100	0.045	**0.035**	**0.026**	**0.038**	**0.025**	0.191
NAS-NeRF XS	0.071	0.124	0.046	0.062	0.057	0.044	0.052	0.266
NAS-NeRF XXS	0.110	0.213	0.086	0.127	0.131	0.061	0.076	0.320

Table 2: Per-scene quantitative results on the Blender synthetic dataset comparing the generated architectures with the baseline NeRF model.

2.3 Network Architecture

The generated NAS-NeRF architectures, as shown in Table 1, are much more compact compared to the NeRF [1] baseline, with up to 23× fewer parameters for the XXS variants. This compactness leads to substantial speedups of up to 4.7× frames per second. The architectures also achieve major reductions in computational complexity, with the S, XS, and XXS variants requiring 2.19-4.19×, 4.34-16.8×, and 16.8-22.1× fewer FLOPs respectively.

The architectures tend to converge on a minimal coarse field, while exhibiting more diversity in the fine fields, which aligns with the fine field's role in capturing higher frequency details. The network depths of each stage remain fairly consistent for both coarse and fine fields across scenes. However, the channel widths vary substantially, especially in the fine fields. Most coarse fields have uniform stage widths, whereas many fine fields have non-uniform widths with progressive channel expansion across stages.

The variation in fine field widths and channel growth demonstrates the scene-specific customization enabled by NAS-NeRF's generative architecture search. Notably, the same architecture satisfied constraints for both the Ficus scene at the XXS scale and the Mic scene at the XS scale. This highlights the flexibility of NAS-NeRF in finding tailored architectures across scenes and target performance levels.

Figure 3: Qualitative results of each of our NAS-NeRF architectures evaluated on the corresponding scene it was generated for, along with the ground truth image and NeRF baseline. Red box shows a magnified view of an image patch.

3 Experiments

We evaluate NAS-NeRF on the Blender synthetic dataset [1], comparing SSIM [12], PSNR, and LPIPS [13] against the NeRF [1] baseline. To ensure that all architectures are trained to a comparable level, we scale the number of training iterations proportional to the architecture parameter efficiency ratio ER_{params} and relative to the number of training iterations required for the baseline NeRF architecture (200k iterations). Note that we define architecture efficiency ratio as $ER_{metric} = Metric(A_{baseline}) : Metric(A_{generated})$.

NAS-NeRF is built on the Nerfstudio framework (v0.2.1) [14] and all architectures are trained using the default hyperparameters for the *Vanilla NeRF* model (i.e. RAdam optimizer with a learning rate of 5×10^{-4}). To maximize reproducibility and extensibility, we also open source our repo[2].

As seen in Figure 1 and Table 2, our generated architectures demonstrate strong performance given their compact size. The NAS-NeRF S variants match or exceed the baseline NeRF in terms of performance metrics, despite having 2-4$\times$ fewer parameters and running up to 1.93$\times$ faster. The NAS-NeRF XXS variants achieve SSIM scores only 5.3% lower than baseline NeRF, while requiring 23$\times$ fewer parameters and achieving 4.7$\times$ speedup. Qualitative comparisons in Figure 3 shows the NAS-NeRF S and XS variants produce rendering quality on par with the baseline, while the smallest XXS models lose some high-frequency details. However, combining our architecture search strategy with newer NeRF techniques for geometric encoding, sampling, and hierarchical scene representation could overcome this limitation and enable high-quality novel view synthesis from ultra-compact models. Overall, the results highlight the ability of NAS-NeRF to find scene-specialized architectures along the complexity-quality spectrum.

4 Conclusion

Our work introduces NAS-NeRF, a family of NeRF architectures that are tailored per scene and compute budget. By executing a generative neural architecture search strategy, we show that our method can discover architectures at varying scales of efficiency (NAS-NeRF XXS, XS, and S), thus introducing a way for developers and researchers to systematically make trade offs between performance target quality and architecture complexity, while still remaining near the pareto frontier. By developing NeRF models that achieve competitive performance with baseline NeRF architectures while being a fraction of the size, we hope to open the door to enabling more widespread deployment of novel view synthesis techniques on resource constrained devices like mobile phones and embedded systems.

References

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[2] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. FastNeRF: High-fidelity neural rendering at 200fps. pages 14346–14355. URL https://openaccess.thecvf.com/content/ICCV2021/html/Garbin_FastNeRF_High-Fidelity_Neural_Rendering_at_200FPS_ICCV_2021_paper.html.

[3] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. 41(4):1–15. ISSN 0730-0301, 1557-7368. doi: 10.1145/3528223.3530127. URL https://dl.acm.org/doi/10.1145/3528223.3530127.

[4] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. pages 5875–5884. URL https://openaccess.thecvf.com/content/ICCV2021/html/Hedman_Baking_Neural_Radiance_Fields_for_Real-Time_View_Synthesis_ICCV_2021_paper.html.

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, Lecture Notes in Computer Science, pages 333–350. Springer Nature Switzerland. ISBN 978-3-031-19824-3. doi: 10.1007/978-3-031-19824-3_20.

[2] https://saeejithnair.github.io/NAS-NeRF/

[6] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. Kaplanyan, and M. Steinberger. DONeRF: Towards real-time rendering of compact neural radiance fields using depth oracle networks. 40(4):45–59. ISSN 1467-8659. doi: 10.1111/cgf.14340. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14340. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14340.

[7] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. DeRF: Decomposed radiance fields. pages 14153–14161. URL https://openaccess.thecvf.com/content/CVPR2021/html/Rebain_DeRF_Decomposed_Radiance_Fields_CVPR_2021_paper.html.

[8] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. pages 14335–14345. URL https://openaccess.thecvf.com/content/ICCV2021/html/Reiser_KiloNeRF_Speeding_Up_Neural_Radiance_Fields_With_Thousands_of_Tiny_ICCV_2021_paper.html.

[9] Alexander Wong, Mohammad Javad Shafiee, Brendan Chwyl, and Francis Li. FermiNets: Learning generative machines to generate efficient neural networks via generative synthesis. URL http://arxiv.org/abs/1809.05989.

[10] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. URL http://arxiv.org/abs/1611.01578.

[11] Alexander Wong. NetScore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage. In Fakhri Karray, Aurélio Campilho, and Alfred Yu, editors, *Image Analysis and Recognition*, Lecture Notes in Computer Science, pages 15–26. Springer International Publishing. ISBN 978-3-030-27272-2. doi: 10.1007/978-3-030-27272-2_2.

[12] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. 13(4):600–612. ISSN 1941-0042. doi: 10.1109/TIP.2003.819861. Conference Name: IEEE Transactions on Image Processing.

[13] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. pages 586–595. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

[14] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, pages 1–12. doi: 10.1145/3588432.3591516. URL http://arxiv.org/abs/2302.04264.