Improving Neural Radiance Fields for More Efficient, Tailored, View-Synthesis

by

Saeejith Muralidharan Nair

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2024

© Saeejith Muralidharan Nair 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Neural radiance fields (NeRFs) have revolutionized novel view synthesis, enabling high-quality 3D scene reconstruction from sparse 2D images. However, their computational intensity often hinders real-time applications and deployment on resource-constrained devices. Traditional NeRF models can require days of training for a single scene and demand significant computational resources for rendering, with some implementations necessitating over 150 million network evaluations per rendered image.

While various approaches have been proposed to improve NeRF efficiency, they often employ fixed network architectures that may not be optimal for all scenes. This research introduces NAS-NeRF, an new approach that employs generative neural architecture search (NAS) to discover compact, scene-specialized NeRF architectures. NAS, a technique for automatically designing neural network architectures, is investigated as a potential method for optimizing NeRFs by tailoring network architectures to the specific complexities of individual scenes.

NAS-NeRF reformulates the NeRF architecture into configurable field cells, enabling efficient exploration of the architecture space while maintaining compatibility with various NeRF variants. Our method incorporates a scene-specific optimization strategy that considers the unique characteristics of each 3D environment to guide architecture search. We also introduce a quality-constrained generation approach that allows for the specification of target performance metrics within the search process.

Experiments on the Blender synthetic dataset demonstrate the effectiveness of NAS-NeRF in generating a family of architectures tailored to different efficiency-quality trade-offs. Our most efficient models (NAS-NeRF XXS) achieve up to 23× reduction in parameters and 22× fewer FLOPs compared to baseline NeRF, with only a 5.3% average drop in structural similarity (SSIM). Meanwhile, our high-quality models (NAS-NeRF S) match or exceed baseline performance while reducing parameters by $2 - 4\times$ and offering up to 1.93× faster inference.

These results suggest that high-quality novel view synthesis can be achieved with more compact models, particularly when architectures are tailored to specific scenes. NAS-NeRF contributes to the ongoing research into efficient 3D scene representation methods, helping enable applications in resource-constrained environments and real-time scenarios.

Acknowledgements

I am extremely grateful for the mentorship and guidance I've received throughout my academic journey. My deepest appreciation goes to my co-supervisors, Prof. Alex Wong and Prof. Javad Shafiee, as well as to Prof. Yuhao Chen, whose collective support has been instrumental in shaping my research experience. I am deeply appreciative of your seemingly boundless patience, your infectious enthusiasm for research, moral support, and your encouragement to pursue my curiosities.

To Prof. Wong: Your willingness to involve me in many research projects and encourage the exploration of my own ideas has been invaluable. You've provided me with countless opportunities to dive into various areas of deep learning and computer vision beyond my core projects. Your exemplary leadership in managing multiple responsibilities and time management has been truly inspirational, setting a standard I aspire to emulate in my future endeavors. Additionally, I am deeply thankful for your substantial investment in research infrastructure, which has been crucial in making our projects possible.

Prof. Shafiee: I am thankful for your support and advice on balancing life and research. Your encouragement to scrutinize data and think critically about results helped instill in me a more rigorous approach to scientific inquiry.

Prof. Chen: Your efforts in building and maintaining our research infrastructure have been the bedrock of our work. From constructing new GPU servers to addressing outages at all hours, our experiments would not have been possible without your dedication. Moreover, your initiatives in fostering a collaborative and enjoyable research environment have greatly enhanced our ability to share ideas and work together effectively.

I'm grateful to my thesis committee for their time and feedback. Thanks to my Vision and Image Processing Lab colleagues for the inspiring discussions and friendship. Lastly, I appreciate the friends, family friends, and former colleagues who've shaped my journey.

Last but most importantly, I would like to thank God, and my family for all of the continuous support and tremendous sacrifices they've made for me. I cannot thank you enough.

Dedication

To my parents.
Thank you for everything.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures ix

List of Tables x

1 Introduction 1

1.1 Overview of Contributions and Thesis Structure 2

2 Background 4

2.1 Neural Radiance Fields 5

2.1.1 NeRF Architecture 6

2.1.2 Scene Representation 6

2.1.3 Differentiable Rendering Pipeline 8

2.1.4 Datasets 10

2.1.5 NeRF Evaluation 11

2.2 Extensions to NeRF . . . 12
2.2.1 Improvements in Efficiency . . . 13
2.3 Adapting to Diverse Scenes . . . 15
2.3.1 Modifications to the Neural Architectures in NeRF . . . 17
2.4 Neural Architecture Search . . . 18
2.4.1 Techniques in NAS . . . 18
2.4.2 Generative Architecture Search . . . 19

3 NeRF Optimization Framework **21**
3.1 The Axes of Optimization in NeRF Efficiency . . . 22
3.1.1 Training Duration . . . 22
3.1.2 Architecture Complexity . . . 24
3.1.3 Search Complexity and Scene-Specific Optimization . . . 25
3.1.4 Quality-Constrained Optimization . . . 28
3.2 Research Objectives . . . 28

4 NAS-NeRF: Generative Neural Architecture Search for Neural Radiance Fields **31**
4.1 Methodology . . . 32
4.1.1 Reparameterizing NeRF for Architecture Search . . . 32
4.1.2 NAS-NeRF Field Cell . . . 32
4.1.3 Generative Neural Architecture Search . . . 35
4.2 Experiments and Results . . . 40
4.2.1 Experimental Setup . . . 40
4.2.2 Performance Analysis . . . 41
4.2.3 Architectural Insights . . . 44
4.2.4 Architectural Patterns and Scene Adaptation . . . 45
4.2.5 Discussion and Implications . . . 46

5 Conclusions **48**

5.1 Summary of Contributions 48

5.2 Future Research Directions 49

References **51**

List of Figures

2.1 High-level overview of the NeRF pipeline 5
2.2 Illustration of the radiance field MLP architecture in NeRF 7

3.1 Impact of training duration on NeRF optimization 23
3.2 Impact of architectural complexity on NeRF optimization 25
3.3 Scene-adaptive optimization in NeRF 27
3.4 Tiered approach to NeRF optimization 29

4.1 NAS-NeRF field cell architecture overview 34
4.2 NAS-NeRF pipeline with configurable field cells 35
4.3 Iterative process of generative neural architecture search for NAS-NeRF . . 38
4.4 Heuristic for determining scene-specific performance targets 40
4.5 Architecture efficiency vs. synthesis quality and inference speed 42
4.6 Qualitative results of NAS-NeRF architectures 43

List of Tables

4.1 Per-scene quantitative results on the Blender synthetic dataset comparing the generated architectures with the baseline NeRF model. 44

4.2 The architecture configuration and performance metrics of each generated NAS-NeRF, along with the baseline NeRF 47

Chapter 1

Introduction

The field of computer vision has long sought to recreate three-dimensional scenes from limited two-dimensional inputs. This challenge, known as novel view synthesis, has applications ranging from virtual and augmented reality to robotics and autonomous navigation [1]. Traditional approaches often struggled with sparse input data or required extensive computational resources. However, the introduction of neural radiance fields (NeRF) [2] in 2020 marked a significant leap forward in this domain, leveraging the power of deep learning to achieve unprecedented levels of realism and flexibility in 3D scene representation and rendering.

NeRF represents a paradigm shift in addressing the classic problem of image-based rendering (IBR) [3]. At its core, NeRF utilizes a multi-layer perceptron (MLP) to learn a continuous volumetric scene representation, mapping five-dimensional inputs—spatial coordinates (x, y, z) and viewing directions (θ, ϕ)—to output color and volume density. This innovative approach allows NeRF to render high-quality, photorealistic images from novel viewpoints by integrating the learned radiance field along camera rays.

The success of NeRF lies in its ability to capture complex geometries, lighting conditions, and view-dependent effects with remarkable fidelity. By encoding the entire scene in the weights of a neural network, NeRF can synthesize new views with a level of detail and consistency that was previously unattainable with traditional computer graphics techniques.

However, the impressive visual quality of NeRF comes at a significant computational cost. The process of rendering a single image requires thousands of network evaluations for each ray cast through the scene. This computational intensity poses challenges for real-time applications and deployment on resource-constrained devices. Moreover, training a NeRF

model to convergence often requires days of computation on high-end GPUs, limiting its accessibility and scalability.

A critical, yet often overlooked, aspect of current NeRF implementations is the use of a uniform neural network architecture across all scenes, regardless of their individual complexities. This one-size-fits-all approach, while simplifying implementation, potentially leads to inefficiencies. Complex scenes with intricate geometries and lighting may benefit from larger, more expressive networks. In contrast, simpler scenes might be adequately represented by more compact architectures, achieving similar quality with reduced computational overhead.

This observation motivates the central question addressed in this thesis: Can we develop a method to automatically tailor NeRF architectures to the specific requirements of individual scenes, optimizing for both rendering quality and computational efficiency?

1.1 Overview of Contributions and Thesis Structure

To address this challenge, this thesis introduces NAS-NeRF, a novel approach that employs generative neural architecture search to dynamically optimize NeRF architectures for specific scenes. The key contributions of this work include:

1. A modular reformulation of the NeRF architecture into configurable field cells, allowing for flexible adaptation of network complexity.

2. A generative neural architecture search framework tailored for NeRF, capable of balancing rendering quality and computational efficiency.

3. A scene-specific optimization strategy that considers the unique characteristics of each 3D environment to guide architecture search.

4. Extensive experimental validation demonstrating significant reductions in model size and inference time while maintaining high-quality renderings.

The remainder of this thesis is structured as follows:

Chapter 2 provides a comprehensive background on neural radiance fields and neural architecture search, establishing the theoretical foundations for our work. We discuss the evolution of NeRF techniques, highlighting current limitations and opportunities for improvement.

Chapter 3 dives deeper into the motivation behind NAS-NeRF, analyzing the relationship between scene complexity, network architecture, and rendering quality. We formulate

the problem of NeRF optimization as a multi-objective search task and discuss the challenges involved.

Chapter 4 presents the core methodology of NAS-NeRF. We detail our approach to modularizing NeRF architectures, the design of our generative search algorithm, and our strategies for efficient exploration of the architecture space. This chapter also includes a thorough experimental setup and an in-depth analysis of our results, comparing NAS-NeRF against the baseline NeRF implementation across various metrics.

Finally, Chapter 5 summarizes the key findings and contributions of this work. We discuss the broader implications of our research for the field of neural rendering and outline promising directions for future investigation, including potential extensions to dynamic scenes and integration with other advanced NeRF variants.

Through this work, we aim to advance the state-of-the-art in neural radiance fields, making this powerful technique more accessible and practical for a wide range of applications. By demonstrating the potential of scene-specific architectural optimization, we hope to inspire new approaches to efficient, high-quality 3D scene representation and rendering.

Chapter 2

Background

Novel view synthesis is a fundamental challenge in computer vision and computer graphics which aims to generate new perspectives of a scene from a limited set of input images. This capability is important for applications spanning 3D reconstruction, virtual reality, robotics, and autonomous navigation [1]. Traditional approaches such as light fields or lumigraph rendering [3] laid important groundwork but often struggled with sparse inputs or demanded substantial computational resources. The introduction of neural radiance fields (NeRFs) [2] in 2020 marked a significant leap forward in this domain, leveraging deep learning techniques to achieve unprecedented levels of realism and flexibility in 3D scene representation and rendering.

This chapter provides a comprehensive overview of NeRFs, their theoretical foundations, core architecture, and evaluation methodologies. We also explore recent extensions to the NeRF framework, with a particular focus on approaches that enhance efficiency and adaptability to diverse scene complexities. Through this exploration, we identify a critical gap in the current research landscape: the lack of methods that simultaneously improve neural network architecture efficiency and tailor these architectures to specific scenes. This observation leads us to propose a new research direction: the application of neural architecture search techniques to discover not only more efficient NeRF architectures but also those that can be customized for individual scenes.

To lay the groundwork for this proposal, we conclude the chapter with an overview of relevant work in the field of neural architecture search (NAS), setting the stage for our contributions in the subsequent chapters.

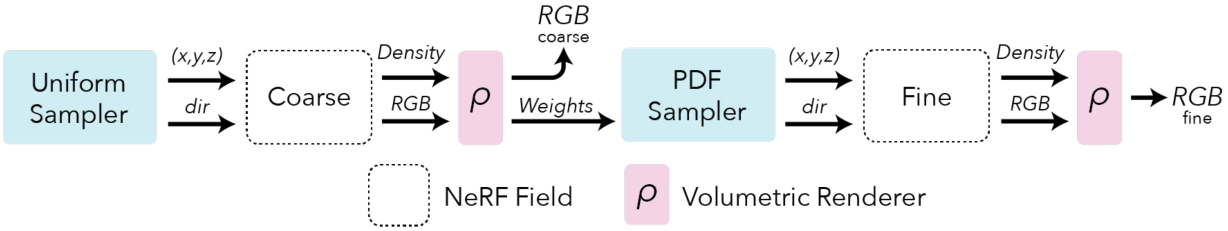


Figure 2.1: High-level overview of the NeRF pipeline, obtained from the documentation of the Nerfstudio framework [7].

2.1 Neural Radiance Fields

Introduced by Mildenhall *et al.* in 2020, neural radiance fields (NeRF) [2] presented a new way of addressing the classic problem of Image-Based Rendering (IBR) [3]. The core challenge of IBR is to generate new views of a scene from a collection of existing images captured from diverse perspectives. While previous approaches struggled with sparse input data or required dense sampling of the light field, NeRF leverages deep learning techniques [4] to synthesize photorealistic images from a remarkably sparse set of input views.

The key innovation of NeRF lies in its ability to learn a continuous volumetric scene representation and store it in the weights of a neural network. This approach enables the synthesis of realistic and continuous representations of the scene from angles that were not captured in the original dataset. At its core, NeRF utilizes a multi-layer perceptron (MLP) [5, 6] to learn a radiance field – a mapping of spatial coordinates and viewing directions to colour and volume density [2]. Figure 2.1 provides an overview of the NeRF pipeline, highlighting the 2 stage hierarchical sampling strategy and volumetric rendering.

This innovative approach allows NeRF to render high-quality images from novel viewpoints by integrating the learned radiance field along camera rays. The following sections dive into the technical details of NeRF's architecture, training process, and rendering pipeline, providing a thorough understanding of this groundbreaking technique in novel view synthesis.

2.1.1 NeRF Architecture

Positional Encoding

One of the main contributions of NeRF that enables the creation of detailed renderings is its positional encoding scheme. This encoding transforms 3D spatial coordinates $\mathbf{x} = (x, y, z)$ and 2D viewing directions $\mathbf{d} = (\theta, \phi)$ using periodic activation functions. By mapping the input coordinates through a series of sine and cosine functions at different frequencies, NeRF enhances the ability of the neural network to represent high-frequency variations in the scene, which is essential for capturing detailed geometry and complex lighting effects.

Specifically, NeRF applies the following encoding function $\gamma(p)$ to each of the three coordinate values in $\mathbf{x}$ and to the three components of the Cartesian viewing direction unit vector $\mathbf{d}$:

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \ldots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)) \tag{2.1}$$

2.1.2 Scene Representation

At its core, NeRF trains a neural network to represent a continuous scene as a 5D function $F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$. This function maps a 3D spatial coordinate $\mathbf{x} = (x, y, z)$ and 2D viewing directions $\mathbf{d} = (\theta, \phi)$ to an emitted color $\mathbf{c} = (r, g, b)$ and volume density σ. In the original work by Mildenhall *et al.* this mapping is learned by optimizing a deep fully-connected neural network (a multi-layer perceptron (MLP)).

Rather than using a single monolithic neural network, NeRF employs a hierarchical sampling strategy consisting of two separate neural networks: one *coarse* and one *fine*. As shown in Figure 2.2:

- Both the coarse and fine MLPs consist of a base network with 8 fully-connected layers, each with a width of 256 channels.
- The base network processes the input 3D coordinate $\mathbf{x}$ and outputs σ along with a 256-dimensional feature vector.
- The head network consists of a single fully-connected layer with a width of 128 channels.

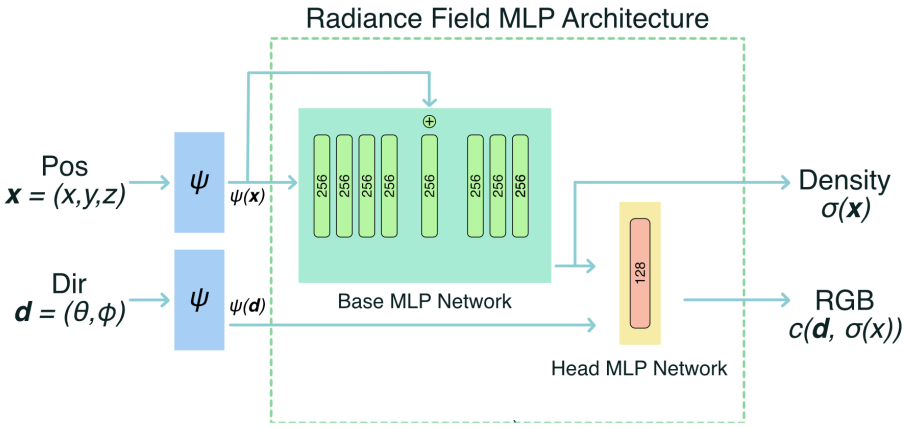


Figure 2.2: A visualization of the neural network architecture used for the coarse and fine radiance fields in NeRF [2], as well as the inputs and outputs of each subnetwork.

- This feature vector output from the base network gets concatenated with the camera ray's viewing direction and is passed to the head network, which outputs the view-dependent RGB colour.

The two-stage hierarchical sampling strategy is a key innovation that "improves rendering efficiency by allocating samples proportionally to their expected effect on the final rendering" [2]. This approach addresses a significant drawback of naive implementations, which would require evaluating the MLP at N query points along each ray, leading to unnecessary computations in free space and occluded regions.

Instead, NeRF's strategy works as follows:

1. The coarse network is first queried at uniformly sampled points N_c along each ray to obtain an initial estimate of the radiance and density distribution.
2. This initial estimate guides the sampling of additional points N_f for the fine network, focusing on the relevant parts of the volume.
3. The fine network is then evaluated at all sampled $N_c + N_f$ points, allowing it to concentrate on regions with higher predicted density or radiance change.

This hierarchical approach enables NeRF to capture high-frequency details more accurately while maintaining computational efficiency. By focusing computational resources on the most relevant parts of the scene, NeRF achieves a balance between rendering quality and performance that has made it a cornerstone of modern novel view synthesis techniques.

2.1.3 Differentiable Rendering Pipeline

Another key innovation in NeRF is its differentiable rendering pipeline, which adapts classical volume rendering techniques for use with neural networks. This approach enables the generation of photorealistic images from novel viewpoints through end-to-end optimization.

Volume Rendering Fundamentals

Central to NeRF's rendering process is the concept of volume density, denoted as $\sigma(\mathbf{x})$. This quantity represents the differential probability of a ray terminating at an infinitesimal particle located at position $\mathbf{x} \in \mathbb{R}^3$. To compute the expected color $C(\mathbf{r})$ of a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o} \in \mathbb{R}^3$ is the ray origin and $\mathbf{d} \in \mathbb{R}^3$ is the ray direction, NeRF employs the following integral:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt \tag{2.2}$$

Here, t_n and t_f represent the near and far bounds of the ray, respectively. The accumulated transmittance $T(t)$ is defined as:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right) \tag{2.3}$$

This function describes the probability that the ray travels from t_n to t without encountering any other particles, which is necessary for determining the visibility of different parts of the scene.

Numerical Approximation

Rendering a view from the continuous neural radiance field requires estimating the integral in Equation 2.2 for each camera ray. However, computing this continuous integral directly

is computationally infeasible. NeRF addresses this challenge using a numerical integration technique based on quadrature. Specifically, NeRF adopts a stratified sampling approach where:

1. The interval $[t_n, t_f]$ is partitioned into N evenly-spaced bins.
2. One point is randomly sampled from within each bin.

Using these samples, NeRF approximates $C(\mathbf{r})$ using the following quadrature rule:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i \tag{2.4}$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j)$ and $\delta_i = t_{i+1} - t_i$ represents the distance between adjacent samples.

Differentiability and Optimization

The formulation for calculating $\hat{C}(\mathbf{r})$ from the set of color and density values $(\mathbf{c}_i, \sigma_i)$ in Equation 2.4 is inherently differentiable. This property facilitates gradient-based optimization, allowing the neural network to learn from the rendering process itself.

The optimization process in NeRF involves minimizing a loss function that quantifies the difference between the rendered images and the ground truth:

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\|\hat{C}_c(\mathbf{r}) - C(\mathbf{r})\|_2^2 + \|\hat{C}_f(\mathbf{r}) - C(\mathbf{r})\|_2^2 \right] \tag{2.5}$$

Here, $\mathcal{R}$ represents the set of rays in each batch, while $C(\mathbf{r})$, $\hat{C}_c(\mathbf{r})$, and $\hat{C}_f(\mathbf{r})$ denote the ground truth, coarse volume predicted, and fine volume predicted RGB colors for ray $\mathbf{r}$, respectively.

Using gradient descent, NeRF minimizes the loss associated with both $\hat{C}_c(\mathbf{r})$ and $\hat{C}_f(\mathbf{r})$, despite the final rendering being derived solely from $\hat{C}_f(\mathbf{r})$. This dual optimization strategy allows the weight distribution from the coarse network to inform sample allocation in the fine network, enhancing overall rendering quality.

2.1.4 Datasets

Several datasets have been introduced to assess the performance and robustness of NeRF and other related methods for novel view synthesis under various conditions.

One of the earliest datasets tailored specifically for NeRF is the Synthetic-NeRF Dataset [2], introduced by Mildenhall *et al.* in their seminal 2020 paper. Generated using Blender, this dataset features 8 synthetic objects such as drums and legos, and was one of the first datasets to feature more complex geometry and materials with more realistic non-Lambertian effects [2]. While beneficial for initial evaluations, the dataset's simplicity and lack of real-world complexity limit its applicability for advanced scene complexities.

The DTU Dataset [8], a multi-view dataset comprising 124 real-world objects, was initially created for evaluating multi-view stereo (MVS) algorithms and later adapted for evaluating NeRF models [2]. While it features scenes with controlled lighting and high-resolution ground truth [9], certain scenes have incomplete areas due to self-occlusion [10].

Introduced in 2020, the BlendedMVS dataset [11], created by Yao *et al.* offers a diverse collection of viewpoints covering landscapes, buildings, and small objects designed for multi-view stereo networks. BlendedMVS provides over 17k high-resolution images, significantly enhancing the generalization ability of trained models by incorporating ambient lighting information during training. However, the dataset's reliance on synthetic blending techniques introduces artifacts which can affect the realism of the generated 3D structures.

The Local Light Field Fusion (LLFF) dataset [12], introduced by Mildenhall *et al.* in 2019 is another widely used real-world dataset for evaluating novel view synthesis methods. It consists of 20 forward-facing scenes captured with a handheld camera under natural illumination conditions. Each scene contains high-resolution images with their corresponding camera poses and a sparse set of scene depth maps.

The Tanks and Temples dataset [13], introduced by Knapitsch *et al.* in 2017, is a large-scale multi-view stereo dataset comprising complex real-world scenes. It consists of two subsets: the *Intermediate* set with 8 scenes and the *Advanced* set with 6 scenes, captured under varying lighting conditions and viewpoints. The scenes depict intricate structures like temples, sculptures, and machinery, posing significant challenges for 3D reconstruction due to their complex geometry, materials, and lighting.

Recent advancements have introduced datasets that address specific limitations of earlier collections. For instance, the NeRF in the Wild dataset [14], introduced by Martin-Brualla *et al.* in 2021, is a large-scale real-world dataset designed to evaluate the generalization capabilities of NeRF models on challenging, uncontrolled environments. It comprises

88 scenes and over 200,000 images captured using mobile phone cameras in various indoor and outdoor settings, such as offices, streets, and parks. The dataset features diverse lighting conditions, dynamic objects, and complex geometries, making it a challenging benchmark for novel view synthesis methods.

The Mip-NeRF 360 dataset [15], introduced by Barron *et al.* in 2022, is tailored for challenging unbounded scenes where the camera captures 360-degree views. It consists of nine scenes, split between five outdoors and four indoors, each focused on a complex central object with detailed backgrounds. This dataset has recently been pivotal for testing NeRF models on their ability to handle complex scenes captured at various scales.

2.1.5 NeRF Evaluation

After detailing some of the datasets that are commonly used to train NeRF models, we now turn to how these models are evaluated. Effective evaluation is crucial to verify that the rendered images are both visually appealing and accurately depict the scenes.

Quantitative Metrics

NeRF models are primarily evaluated using metrics like peak signal-to-noise ratio (PSNR), Structural Similarity Index (SSIM) [16], and Learned Perceptual Image Patch Similarity (LPIPS) [17]. These metrics quantify both the fidelity and perceptual quality of rendered images compared to the ground truth real-world photographs or synthetic images.

PSNR quantifies the error levels between generated images and ground truth, serving as an indicator of image quality. It is calculated using the equation:

$$\text{PSNR} = 20 \cdot \log_{10} \left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}} \right) \tag{2.6}$$

where MAX_I is the maximum possible pixel value of the image, and MSE is the mean squared error between the generated and ground truth images.

SSIM measures the visual impact of changes in luminance, contrast, and structure between the rendered and actual images. The formula for SSIM is given by:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.7}$$

where μ_x and μ_y are the average values of x and y, σ_x^2 and σ_y^2 are the variances of x and y, σ_{xy} is the covariance of x and y, and c_1, c_2 are constants to stabilize the division with weak denominator.

LPIPS provides a deeper dive by evaluating similarity based on high-level features extracted via deep learning, focusing on aspects that closely align with human visual perception. The LPIPS metric is calculated using the following equation:

$$\text{LPIPS} = \sum_l w_l \cdot \frac{1}{H_l W_l} \sum_{h,w} \|\phi_l(\hat{y})_{hw} - \phi_l(y)_{hw}\|^2 \tag{2.8}$$

where ϕ_l denotes the feature map extracted from layer l of a pretrained network, w_l is the weight for layer l, and H_l, W_l are the dimensions of the feature map at layer l.

Qualitative Assessment

While quantitative metrics offer valuable insights into NeRF's technical performance, qualitative assessments are crucial for evaluating the perceptual quality and realism of rendered images. Given the 3D nature of the task, interactive visualization has become a primary method for qualitative evaluation.

Tools like Nerfstudio [7] enable real-time navigation through synthesized scenes, offering several advantages:

- Observation of view-dependent effects and scene consistency across different angles.
- Assessment of smooth transitions between viewpoints, critical for applications like virtual reality.
- Easy identification of common artifacts such as floaters, localized blurring, or inconsistencies in specular reflections that might be overlooked in static images.

2.2 Extensions to NeRF

The original NeRF architecture, introduced by Mildenhall *et al.* [2], showcased impressive capabilities in novel view synthesis. However, it also revealed significant computational challenges. The need to evaluate a fully connected neural network for each sample along a ray resulted in a rendering process that was both computationally expensive and time-consuming. In response to these limitations, researchers have explored various innovations, aiming to enhance NeRF's efficiency and adaptability across diverse scene complexities.

2.2.1 Improvements in Efficiency

Efforts to improve NeRF's efficiency have primarily followed two approaches: refining the core neural network architecture to reduce computational complexity, and developing new techniques to streamline the rendering process. These methods seek to strike an optimal balance between rendering quality and computational efficiency. By doing so, they aim to make NeRF more viable for real-time applications and deployment on resource-constrained devices.

Caching and Baking

One of the major bottlenecks in NeRF rendering is the computational cost of evaluating the neural network for each ray sample during the volume rendering process. To address this, researchers have explored techniques to improve efficiency during both training by optimizing the sampling process and during rendering by using techniques like caching and baking.

Caching involves storing and reusing the network outputs for specific input coordinates, avoiding redundant computations. This can significantly reduce the number of network evaluations required, especially for scenes with overlapping views or static regions. Garbin *et al.* introduced a caching strategy in their FastNeRF [18] method, which caches the network outputs in a spatial hash grid and reuses them for subsequent rays passing through the same locations. This approach achieved up to 3000× speedup in rendering time compared to the original NeRF implementation.

Building on these advancements, the concept of baking neural radiance fields into a Sparse Neural Radiance Grid (SNeRG) [19] was introduced by Hedman *et al.* which further enhances real-time rendering capabilities. SNeRG precomputes and stores the volumetric scene as a sparse voxel grid containing learned feature vectors and diffuse colors, thus reducing the need for real-time network evaluations. By focusing computational efforts on pre-baking data and utilizing a smaller MLP for real-time view-dependent effects, SNeRG enables efficient, high-quality rendering on commodity hardware, making it feasible to achieve more than 30 frames per second even on a laptop GPU. This approach not only retains the high-resolution detail and view-dependent appearance characteristic of NeRF but also significantly improves the feasibility of using NeRF models for interactive applications.

Advancing further, Wadhwani *et al.* introduced SqueezeNeRF, which optimizes the caching approach by further factorizing the neural network architecture used in FastNeRF

into even smaller, more specialized networks [20]. This method enhances memory efficiency over 60× compared to FastNeRF, allowing for real-time rendering at over 190 frames per second on a high-end GPUs without sacrificing image quality. SqueezeNeRF achieves this by splitting the position-dependent network into three separate networks, each handling a distinct subset of the position coordinates, thereby reducing the redundancy and computational overhead typically associated with neural radiance fields.

Tensor Decomposition

Building on the strides made by caching and baking technologies in NeRF, Chen *et al.* introduced TensoRF [21], which enhances the efficiency of neural radiance field rendering through tensor decomposition, specifically employing CP [22] and vector-matrix (VM) decompositions. Unlike the traditional NeRF [2], which relies heavily on MLPs to encode the volumetric scene, TensoRF treats the entire volume as a 4D tensor, facilitating the factorization into multiple low-rank tensor components. These factorizations not only enhance memory efficiency but also substantially decrease the computational demands, as demonstrated by TensoRF's ability to achieve realistic scene rendering with model sizes under 4MB and reconstruction times reduced to less than ten minutes using standard PyTorch implementations.

Efficient Sampling

Efficient sampling strategies aim to reduce the number of ray samples required for accurate rendering, thereby reducing the overall computational burden. Two notable advancements in this area are DONeRF [23] and NerfAcc [24], which introduce sophisticated methods to enhance sampling efficiency.

DONeRF (Depth Oracle Neural Radiance Fields) optimizes the sampling process by integrating a depth oracle network that predicts optimal sampling locations along each ray. This network employs a classification approach to discretize depth into logarithmically spaced bins, allowing it to focus computational resources on the most significant features of the scene. By predicting depth values before shading computations, DONeRF efficiently reduces the number of samples needed, particularly in complex and large-scale scenes. This approach not only accelerates rendering but also maintains high-quality outputs by focusing on regions with high informational value.

On the other hand, NerfAcc introduces a flexible Python toolbox that enhances sampling efficiency across various NeRF implementations. It leverages an advanced transmittance estimator to optimize where samples should be taken along each ray. This method

significantly decreases the number of required samples by focusing on the most impactful regions of the volume. NerfAcc's framework allows for rapid integration into existing NeRF models, reducing computational time by up to 20x without sacrificing rendering quality.

2.3 Adapting to Diverse Scenes

Handling Dynamic Scenes and Non-Rigid Objects

The original NeRF formulation was designed for static scenes, limiting its applicability to dynamic environments or non-rigid objects. Since then, various extensions have been proposed to address this limitation.

D-NeRF [25], developed by Pumarola *et al.* extends the static scene capability of NeRF to dynamic environments by introducing a deformation network alongside the canonical radiance field. This approach allows the model to handle non-rigid transformations by predicting changes over time, thus enabling the rendering of scenes with moving and deforming objects.

Nerfies [26], introduced by Park *et al.* takes a step further by not only accommodating changes in the scene geometry over time but also optimizing a continuous volumetric deformation field. This field maps each observation to a canonical form, which is then used to synthesize novel views of non-rigidly deforming scenes. This method demonstrates robustness in handling casual captures from mobile phones, turning them into high-quality 3D renderings.

Lastly, ST-NeRF [27] by Xian *et al.* proposes a layered neural representation to address both spatial and temporal scene dynamics. This model partitions the scene into multiple dynamic entities, each represented as a spatio-temporal coherent neural radiance field. It supports complex edits such as affine transformations and temporal retiming, offering extensive control over the dynamic elements in free-viewpoint videos.

Generalization to Unseen Scenes and Categories

While NeRF achieves impressive results in novel view synthesis, one of its limitations is the need to train a separate model for each individual scene. This can be computationally expensive and impractical for large-scale applications or scenarios where the scene is not known in advance. To address this, researchers have explored techniques to enable NeRF models to generalize to unseen scenes or categories, leveraging principles from meta-learning, few-shot learning, and conditional neural processes.

One approach is to train NeRF models on a diverse set of scenes from a specific category (e.g., indoor scenes, outdoor landscapes, or human faces) and learn a shared representation that can be fine-tuned or adapted to new scenes from the same category with minimal additional training. Tancik *et al.* explored learning initializations [28] based on the underlying class of signals being represented, enabling better generalization when only partial observations of a signal are available. Similarly, Yu *et al.* proposed PixelNeRF [29], which leverages a fully convolutional network to learn a generalizable NeRF representation conditioned on a sparse set of input images.

Another line of work explores the use of hypernetworks or neural processes to generate NeRF models on-the-fly from input data. Sen *et al.* proposed HyP-NeRF [30], which uses a hypernetwork to dynamically generate both the multi-resolution hash encodings and the weights of a NeRF model, enabling it to generalize across unseen objects within a given object category. Additionally, Niemeyer and Geiger developed GIRAFFE [31], a method that models scenes as compositional generative neural feature fields. This approach enhances controllability by allowing for the individual manipulation of objects within a scene in terms of position, rotation, and appearance, using separate neural feature fields for each object.

Handling Sparse Input Views and Regularization Techniques

Another challenge in using NeRF is the requirement for a dense set of input views to achieve high-quality reconstructions. In many practical scenarios, such as casual photography or robotics applications, only a sparse set of views may be available, leading to potential artifacts or incomplete reconstructions.

To address this, researchers have explored various regularization techniques and strategies for handling sparse input data in NeRF. One approach is to incorporate additional priors or regularization terms into the NeRF optimization objective. Deng *et al.* proposed Depth-Supervised NeRF (DS-NeRF) [32], which incorporates depth supervision from sparse depth maps or LiDAR scans to improve NeRF reconstruction in areas with limited view coverage. Niemeyer *et al.* introduced RegNeRF [33], which specifically addresses the issues of view synthesis from sparse inputs by regularizing both the geometry and appearance of generated scenes. This method not only optimizes the scene's radiance fields but also employs a normalizing flow model to adjust the color of unobserved viewpoints, significantly enhancing the fidelity of the reconstructions.

Another notable development is FreeNeRF [34], proposed by Yang *et al.* , which introduces a frequency regularization to the NeRF training process. FreeNeRF gradually

increases the visible frequency range of inputs during training, starting with only low-frequency components and slowly adding higher frequencies, which prevents early overfitting and allows NeRF to learn coarse structure before fine details.

Recently, Somraj and Soundararajan introduced ViP-NeRF (Visibility Prior for Sparse Input Neural Radiance Fields) [35] which addresses the sparse input view challenge by incorporating a visibility prior computed using plane sweep volumes [36]. This visibility prior estimates whether each pixel in one view is visible in the other view and can be reliably estimated without pre-training, unlike previous works which use depth-based priors.

2.3.1 Modifications to the Neural Architectures in NeRF

As discussed previously, while much of the research on NeRF has focused on optimizing the rendering pipeline (such as by improving sampling techniques and caching strategies), relatively few studies (which we shall now discuss) have explored innovations in the core neural network architecture.

KiloNeRF [37], introduced by Reiser *et al.* , decomposes NeRF's single large radiance field into thousands of tiny MLPs, each responsibile for representing a small portion of the scene. This parallelization strategy is particularly friendly to modern GPUs, and when combined with methods like empty space skipping and early ray termination, allows KiloNeRF to achieve real-time rendering speeds up to three orders of magnitude faster than the original NeRF, while maintaining comparable visual quality.

Instant-NGP [38], developed by Müller *et al.* , introduces a novel multiresolution hash encoding that allows for the use of a much smaller neural network without sacrificing quality. This method uses a hash table to store the encoded features of the input coordinates, allowing for fast lookups and efficient computation. When combined with carefully optimized CUDA implementations, Instant-NGP manages to achieve real-time training and rendering speeds that are several orders of magnitude faster than the original NeRF, while achieving greater synthesis quality across a variety of neural graphics tasks.

Also as previously discussed, while many works have proposed ways of conditioning the radiance field on signals in the scene, there have been no works dedicated to tailoring the neural network architecture based on the input set of images. However, we believe that by taking into account the complexity of each scene and tailoring the architecture to it, we can achieve a better balance between computational demands and rendering quality. But how can we find the right architecture for a given scene? This is where we propose to use neural architecture search (NAS) and apply it to the domain of this thesis — NeRF.

2.4 Neural Architecture Search

neural architecture search (NAS) as a field seeks to come up with methods for automating the design of neural network architectures. Instead of the manual trial-and-error approaches that have traditionally dominated neural network design, NAS leverages strategies like reinforcement learning, evolutionary algorithms, and gradient-based methods to explore the vast and complex space of architectures in an automated fashion.

2.4.1 Techniques in NAS

Reinforcement Learning

Initially introduced by Zoph and Le [39], the classic reinforcement learning approach to NAS involves training a controller, typically a recurrent neural network, to generate promising neural network architectures. The controller predicts architectural hyperparameters sequentially, which are then used to construct a child network. This child network is trained and evaluated, and the performance feedback is used to update the controller via policy gradient methods. This method's iterative nature allows it to explore complex architectural landscapes and has been successfully applied to tasks such as image classification and sequence modeling.

Evolutionary Algorithms

Evolutionary algorithms like AmoebaNet [40] optimize neural architectures by mimicking the process of natural selection. They start with a population of random architectures and evolve them over time through operations like mutation, crossover, and selection based on performance metrics. This approach is particularly effective in exploring diverse architectural spaces without prior knowledge, making it suitable for tasks where architectural patterns are less well-understood.

Gradient-Based Methods

Gradient-based NAS methods, such as DARTS [41], optimize a continuous relaxation of the architecture representation, allowing the use of gradient descent for efficient search in the architecture space. These methods significantly reduce the computational resources required by directly optimizing on the architecture parameters.

2.4.2 Generative Architecture Search

Generative Architecture Search (GAS) aims to automate the design of neural network architectures by leveraging a machine-driven exploration strategy to optimize network design under predefined constraints. This section explores the foundational methodologies of GAS, which is the method we leverage to design more efficient neural network architectures in this thesis.

Generative Synthesis

Generative synthesis [42] is the core technique in GAS. It involves a dynamic interplay between a generative model (the generator) and an evaluative model (the inquisitor). As discussed by Wong *et al.* [42], this process is formulated as an optimization problem designed to maximize a universal performance function while adhering to operational constraints:

$$G = \max_{G} U(G(s)) \quad \text{subject to} \quad 1_r(G(s)) = 1, \forall s \in S, \tag{2.9}$$

where G denotes the generator, U represents the performance function, and 1_r is an indicator function that ensures compliance with the requirements. The goal is to develop a versatile generator capable of producing a variety of architectures optimized for different operational contexts.

Methodology

The methodology of GAS encompasses several key phases: defining the search space, generating architectures, evaluating their performance, and refining the generative process based on feedback from the inquisitor.

Defining the Search Space The first phase in GAS is the careful definition of the search space, which includes all potential macro-architectural configurations that the generator might consider. This space is typically vast and includes various architectural elements such as layer types, connections, activation functions, and more. The richness of this search space is crucial as it determines the breadth of possible solutions and directly impacts the ability of the GAS to find a highly optimized architecture. Constraints based on prior knowledge about architectural efficiencies, such as depth and width limitations or specific

layer arrangements that are known to perform well for similar tasks, are also integrated during this stage.

Architecture Generation Once the search space is defined, the machine-driven design exploration phase begins. In this stage, the generator proposes new architectures. Each generated architecture is a potential solution to the problem at hand and is constructed according to the operational constraints set out in the initial phase. The generator's capability to propose viable and diverse architectures is pivotal, as it feeds directly into the subsequent evaluation phase.

Performance Evaluation After architecture generation, each candidate is evaluated to determine how well it meets the predefined performance criteria. This evaluation is typically performed using a validation dataset where the architecture's efficiency, effectiveness, and compliance with operational requirements are assessed. Performance metrics such as accuracy, architecture complexity, and computational complexity are considered. The inquisitor plays a vital role here, ensuring that only architectures that meet certain thresholds advance to the next phase.

Refinement Through Feedback Critical to the GAS methodology is the iterative refinement process. Feedback from the evaluation phase is used to update the generative model, enhancing its ability to produce more effective architectures in subsequent iterations. This feedback loop is central to the generative synthesis approach, as it allows the model to learn from past outcomes and continuously improve.

Chapter 3

NeRF Optimization Framework

While neural radiance fields (NeRF) [2] have demonstrated remarkable capabilities in novel view synthesis, their computational demands often render them impractical for real-time applications or deployment on resource-constrained devices. This chapter introduces a multi-dimensional optimization framework for NeRF, aiming to balance rendering quality with computational efficiency.

The computational burden of NeRF is significant. For example, Mildenhall et al. [2] report that training NeRF on their Blender synthetic dataset necessitates processing 640,000 rays per image, with each ray sampled at 256 points. This translates to over 150 million network evaluations per rendered image, posing substantial challenges for real-time applications and deployment on resource-constrained platforms.

To quantify this computational burden, we can estimate the floating-point operations (FLOPs) required for a single forward pass through the NeRF network. Assuming a typical NeRF architecture with 8 fully-connected layers of 256 neurons each, we can approximate the FLOPs as follows:

$$\text{FLOPs}_{\text{forward}} \approx 8 \times 256 \times 256 \times 2 = 1,048,576 \tag{3.1}$$

Multiplying this by the number of samples per ray (256) and the number of rays per image (640,000), we arrive at approximately 1.7×10^{14} FLOPs per image. This immense computational demand has spurred research into more efficient NeRF variants.

rendering a single image using the original NeRF implementation requires over 150 million network evaluations, translating to approximately 1.7×10^{14} floating-point operations

(FLOPs). This immense computational demand has spurred research into more efficient NeRF variants, primarily focusing on optimizing sampling and rendering aspects of the pipeline.

However, comparatively little attention has been given to optimizing the core neural network architecture itself. Moreover, existing approaches tend to employ a one-size-fits-all architecture, regardless of the complexity of the scene being represented or the specific performance requirements of the application. This observation leads us to a crucial question: Can we tailor the neural network architecture of NeRF to better suit the complexity of individual scenes and the computational constraints of different deployment scenarios? This inquiry forms the foundation of our research into neural architecture search for NeRF, which we will explore in depth in this chapter.

3.1 The Axes of Optimization in NeRF Efficiency

To systematically explore the trade-offs in NeRF performance and computational efficiency, we propose a multi-dimensional optimization framework. This approach allows us to analyze and balance various factors that influence NeRF's effectiveness and practicality.

Our primary goal in NeRF optimization is to enhance efficiency while maintaining or achieving a specific target constraint, typically related to synthesis quality. This quality can be quantified using metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), or Learned Perceptual Image Patch Similarity (LPIPS).

For our analysis, we introduce a simplified model where the synthesis quality target represents one axis in a multi-dimensional space. The other axes correspond to key variables that can be manipulated to optimize NeRF performance:

3.1.1 Training Duration

Historically, training NeRF models has been a time-intensive process. The original NeRF implementation by Mildenhall et al. [2] required over 3 days of training, while subsequent improvements like Mip-NeRF-360 [43] still needed more than a day. Recent advancements, such as Instant-NGP [38], have significantly reduced training times, achieving high-quality results in minutes rather than days.

Despite these improvements in efficiency, the relationship between training duration and synthesis quality remains an important consideration in NeRF optimization. Empirical



Figure 3.1: Radar plot illustrating the multi-dimensional optimization space for NeRF. The plot shows four axes: Scene Complexity, Architecture Complexity, Training Duration, and Synthesis Quality. Three overlapping polygons (green, yellow, and blue) represent different optimization strategies or configurations. The strategies illustrate how given a fixed architecture and scene, the synthesis quality can be improved by increasing the training duration.

observations across various NeRF implementations suggest a general trend: extending the training duration often leads to enhanced synthesis quality as the model refines its representation of the scene. However, this relationship is not linear and can vary based on factors such as scene complexity, network architecture, and optimization algorithm. Typically, the quality improvement follows a pattern of diminishing returns:

1. **Initial rapid improvement:** In the early stages of training, the model quickly learns to capture the broad structure of the scene, resulting in a steep increase in synthesis quality.

2. **Gradual refinement:** As training progresses, the rate of improvement slows. The model begins to capture finer details and more subtle aspects of the scene.

3. **Plateau:** Eventually, the quality improvements become minimal, and the model may even risk overfitting if trained for too long without proper regularization.

Figure 3.1 illustrates this general trend of how increasing training duration can affect synthesis quality for a fixed architecture. Understanding this relationship is crucial for optimizing NeRF models, as it allows practitioners to make informed decisions about the trade-off between training time and quality improvements.

3.1.2 Architecture Complexity

Another approach to enhancing synthesis quality involves increasing the complexity of the neural network architecture, typically by expanding its width or depth, resulting in an architecture with a greater number of parameters.

Several studies and implementations have explored the impact of architecture complexity on NeRF performance. For instance, the Nerfacto series of models in the Nerfstudio framework [7] exemplifies a strategy of scaling model size. This framework offers variants with hidden dimensions of 64, 128, and 256, corresponding to model sizes of approximately 6GB, 12GB, and 24GB, respectively. These different scales allow practitioners to choose a model size appropriate for their specific requirements and computational resources.

The relationship between architecture complexity and synthesis quality typically exhibits the following characteristics:

1. **Improved representation capacity:** Larger models generally have a greater capacity to capture complex scene details, potentially leading to higher synthesis quality.

2. **Diminishing returns:** As model size increases, the improvements in quality often follow a pattern of diminishing returns. This means that doubling the model size may not result in a proportional increase in quality.

3. **Scene-dependent benefits:** The impact of increased model complexity can vary significantly depending on the complexity of the scene being modeled. Simpler scenes may not benefit as much from larger models compared to more complex scenes with intricate geometry or lighting conditions.

4. **Computational trade-offs:** While larger models may yield improved synthesis quality, they also incur increased computational costs. This includes longer training times, higher memory requirements, and slower inference speeds.



Figure 3.2: This radar plot expands on the concept of NeRF optimization strategies, focusing on the impact of architectural complexity. While maintaining the same four axes as Figure 3.1 (Scene Complexity, Architecture Complexity, Training Duration, and Synthesis Quality), this visualization emphasizes how increasing the complexity of the neural network architecture can lead to improved synthesis quality. The three polygons (green, yellow, and blue) represent progressively more complex architectures, with the green polygon extending furthest along the Architecture Complexity and Synthesis Quality axes. This illustration underscores the potential benefits and trade-offs of scaling up NeRF model complexity while keeping other factors constant.

Figure 3.2 conceptually illustrates the general trend of the trade-off between architecture complexity and synthesis quality. Recent advancements have begun to challenge the conventional wisdom that equates larger models with superior performance. For instance, the work of Müller *et al.* [38] has demonstrated that skillfully engineered compact models, when coupled with efficient encoding schemes, can yield results of remarkable quality. Nevertheless, even their work shows that additional synthesis quality gains can be achieved by increasing the size of the MLP.

3.1.3 Search Complexity and Scene-Specific Optimization

The process of finding optimal NeRF configurations introduces an additional dimension to our optimization problem: search complexity. This aspect becomes particularly significant

when considering the diverse range of scenes that NeRF models are expected to represent accurately.

Search complexity refers to the computational effort required to explore the vast space of possible NeRF configurations in order to find an optimal or near-optimal solution for a given scene. This exploration process creates a crucial balance between the effort invested in finding an optimal architecture and the performance gains achieved. The methods employed can range from simple grid searches to sophisticated neural architecture search (NAS) techniques, each with its own trade-offs between search efficiency and solution quality.

The complexity of the scene being modeled introduces another layer of variability to this optimization challenge. Different scenes may require substantially different architectural configurations to achieve optimal results:

1. **Simple scenes:** These may benefit from efficient, compact models with shorter training times. The relative simplicity of the scene allows for a more streamlined architecture without sacrificing rendering quality.

2. **Complex scenes:** Such scenes might require larger models and longer training durations to accurately capture intricate details. The increased complexity necessitates a more sophisticated architectural approach to effectively represent fine-grained features.

3. **Dynamic scenes:** These could necessitate specialized architectures capable of handling temporal variations. The added dimension of time introduces unique challenges that may require novel architectural solutions.

This variability in scene complexity necessitates adaptive NeRF optimization strategies. By tailoring both the architecture and search process to specific scene characteristics, we can potentially improve NeRF efficiency and effectiveness across diverse applications. The challenge lies in efficiently navigating the vast architectural search space while adapting to each scene's unique demands, motivating our exploration of advanced, adaptive optimization techniques.

Multi-dimensional Optimization

The interplay between training duration, architecture complexity, search effort, and scene characteristics creates a multi-dimensional optimization problem. This complexity high-

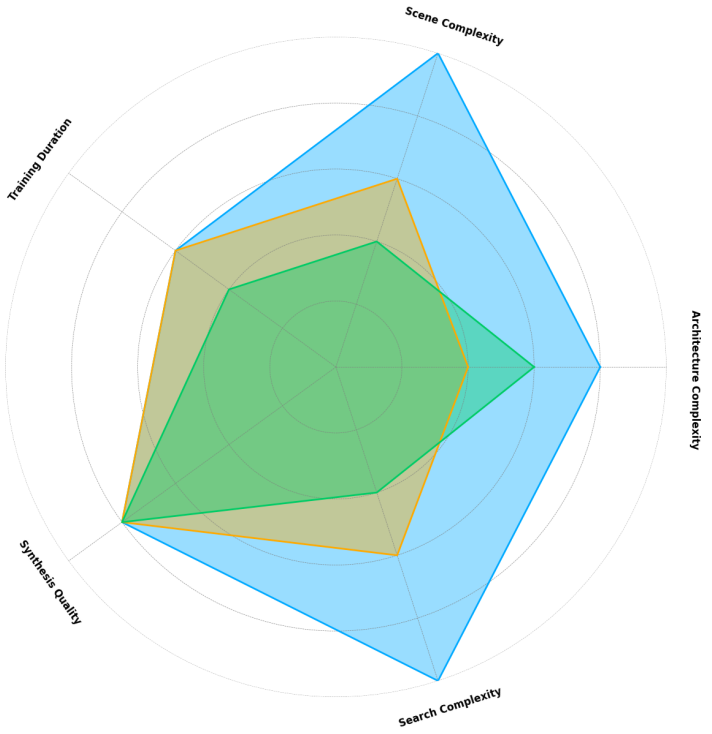


Figure 3.3: Five-dimensional radar plot illustrating NeRF optimization strategies across Scene Complexity, Architecture Complexity, Training Duration, Synthesis Quality, and Search Complexity. The three nested polygons (blue, yellow, green) demonstrate how achieving a fixed synthesis quality requires varying levels of architecture and search complexity depending on scene complexity. The blue polygon shows that highly complex scenes demand greater architectural and search complexity to reach the target quality. The yellow and green polygons represent intermediate and simpler scenes respectively, requiring progressively less complex architectures and search strategies to achieve the same quality target. This visualization highlights the importance of scene-adaptive optimization in NeRF, emphasizing how resource allocation can be tailored to scene complexity while maintaining consistent synthesis quality.

lights the potential benefits of adaptive or scene-specific NeRF models, capable of adjusting their structure and training approach based on the unique demands of each scene.

Figure 3.3 illustrates the core mental model guiding our investigations. It demonstrates how, for a given synthesis quality target, we can arrive at three distinct NeRF models, each satisfying the required quality but optimized along different axes.

3.1.4 Quality-Constrained Optimization

Building on our multi-dimensional optimization framework, we propose a strategy for optimizing NeRF models under varying quality constraints. We define three quality tiers: S (high), XS (medium), and XXS (low), as illustrated in Figure 3.4.

This tiered approach is analogous to graphics settings in video games, where different quality levels accommodate a range of hardware capabilities. In the context of NeRF, this allows for the deployment of novel view synthesis methods across devices with varying computational resources, ensuring optimal performance across a wide range of scenarios and hardware constraints.

3.2 Research Objectives

Based on our analysis of the NeRF optimization space, we formulate two primary research objectives:

1. **Scene-Specific Optimization:** Can we develop a method to automatically determine and construct the most suitable architecture for a given scene, taking into account its unique characteristics and complexity?
2. **Quality-Constrained Architecture Generation:** Can we incorporate target quality specifications directly into the search process, allowing for controlled architecture generation that meets specific performance requirements while optimizing resource utilization?

To address these objectives, we frame our problem mathematically. Let A represent a NeRF architecture, defined by its structural parameters (e.g., number of layers, neurons per layer). For a given scene S, we aim to find an optimal architecture A^* that maximizes

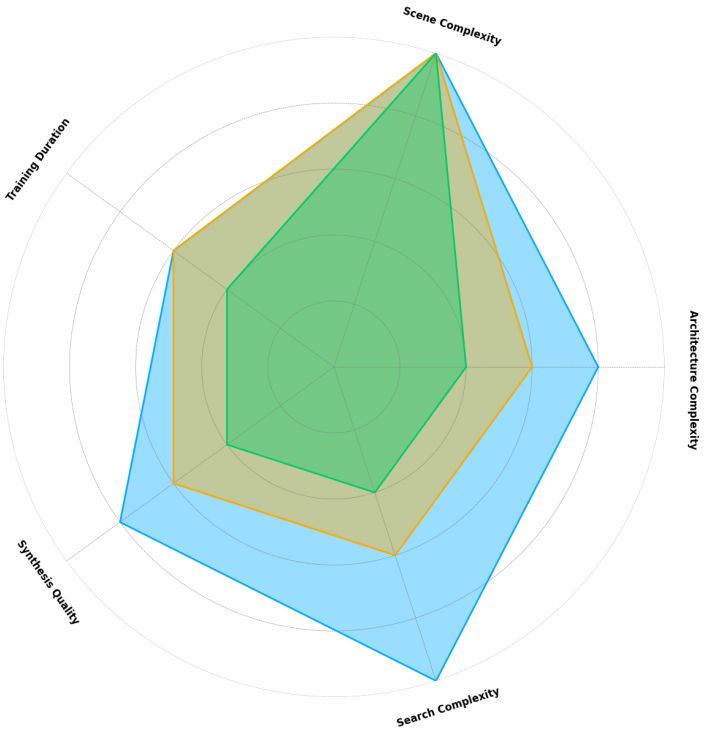


Figure 3.4: Radar plot illustrating a tiered approach to NeRF optimization across five dimensions: Scene Complexity, Architecture Complexity, Training Duration, Synthesis Quality, and Search Complexity. The three nested polygons represent our S/XS/XXS quality tiers for a fixed scene complexity. The blue (outermost) polygon corresponds to the S tier, demonstrating that achieving high synthesis quality may require greater architecture complexity and more extensive search. The yellow (middle) polygon represents the XS tier, offering a balanced approach. The green (innermost) polygon indicates the XXS tier, where lower synthesis quality targets can be met with simpler architectures and less intensive search. This visualization highlights how NeRF models can be tailored to different quality requirements (S/XS/XXS) while adapting to scene complexity, illustrating the trade-offs between synthesis quality and computational resources.

a quality metric Q (e.g., PSNR, SSIM) while minimizing computational resources R (e.g., FLOPs, memory usage):

$$A^* = \arg\max_A Q(A, S) \quad \text{subject to} \quad R(A) \leq R_{\max} \tag{3.2}$$

where $R_{\max}$ represents the maximum allowable computational resources.

For our quality-constrained objective, we reformulate the problem to minimize resources while meeting a target quality Q_{target}:

$$A^* = \arg\min_A R(A) \quad \text{subject to} \quad Q(A, S) \geq Q_{\text{target}} \tag{3.3}$$

To solve these optimization problems, we propose neural architecture search for NeRF (NAS-NeRF), a framework that leverages generative architecture search to automatically discover efficient, scene-specialized NeRF architectures. In the following chapter, we will detail our NAS-NeRF methodology, explaining how we parameterize the NeRF architecture into modular cells and employ a generative search strategy to balance complexity and target synthesis quality.

Chapter 4

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

In this chapter, we introduce NAS-NeRF, a novel approach that addresses the fundamental challenges in optimizing neural radiance fields (NeRFs) as outlined in Section 3.2. Our work focuses on two primary objectives: first, to generate more efficient architectures for neural radiance fields, and second, to develop a method that allows specifying a target quality within the search process, thereby controlling the architecture generation.

The motivation for this research stems from the inherent limitations of existing NeRF implementations. While NeRFs have demonstrated remarkable capabilities in novel view synthesis, their computational demands often render them impractical for real-time applications or deployment on resource-constrained devices. Moreover, the one-size-fits-all approach adopted by many NeRF variants fails to account for the varying complexities of different scenes, potentially leading to overparameterized models for simple scenes or underperforming ones for complex scenarios.

NAS-NeRF aims to bridge this gap by introducing a flexible, scene-adaptive framework that can generate optimal architectures tailored to specific scenes and computational constraints. By leveraging the power of neural architecture search, we seek to automate the process of finding the most efficient NeRF configuration for a given scenario, potentially revolutionizing the field of 3D scene representation and rendering.

4.1 Methodology

The cornerstone of our approach lies in reformulating the NeRF architecture to make it amenable to neural architecture search techniques. This reformulation is crucial not only for enabling efficient search but also for ensuring that our method can be generalized to various NeRF variants, not just the classic NeRF model proposed by Mildenhall et al. [2].

4.1.1 Reparameterizing NeRF for Architecture Search

To facilitate effective architecture search, we must first reparameterize the NeRF model into a form that is conducive to optimization. Our approach is inspired by the high-level NeRF pipeline illustrated in Figure 2.1. A careful analysis of this pipeline reveals that across all NeRF methods, the core radiance field is the component that houses the neural network architecture, while the remaining components are responsible for sampling or volumetric rendering.

This observation is crucial because it allows us to focus our optimization efforts on the core neural network component, which is the primary contributor to the computational complexity of NeRF models. By isolating this component, we can apply techniques from neural architecture search without interfering with the fundamental principles of volumetric rendering that underpin NeRF's success.

Drawing inspiration from seminal works in neural architecture search, particularly the cell-based approach introduced by Zoph et al. [39], we propose to parameterize NeRF into modular cells for the core radiance field. This cell-based approach offers several advantages:

1. **Modularity:** It allows us to break down the complex NeRF architecture into manageable, interchangeable components.
2. **Flexibility:** The cell-based structure can be easily adapted to different NeRF variants.
3. **Efficiency:** By focusing the search on repeatable cell structures, we can significantly reduce the search space while still exploring a wide range of architectural possibilities.

4.1.2 NAS-NeRF Field Cell

The heart of our methodology lies in the NAS-NeRF field cell, a novel construct that modularizes the classic NeRF field into distinct stages. This modularization is key to

enabling efficient architecture search while preserving the essential functionality of NeRF.

In the classic NeRF model [2], the radiance 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 typically learned through an 8-layer density estimator MLP and a smaller single-layer radiance estimator MLP. In our approach, we focus our architecture search primarily on the density estimator, as it is the major contributor to the overall parameter count and computational complexity.

As illustrated in Figure 4.1, we decompose the density estimator into three configurable stages:

1. **Initial Feature Extraction:** Fully connected layers with $\{D_1, C_1\}$ depth and width. This stage is responsible for initial feature extraction from the input coordinates.
2. **Skip Connection and Intermediate Processing:** A single fully connected layer with $\{1, C_2\}$ depth and width, augmented with a skip connection that concatenates the output with the input. This stage allows the network to preserve and process both low-level and high-level features.
3. **Final Feature Refinement:** Fully connected layers with $\{D_3, C_3\}$ depth and width. This stage performs the final refinement of features before outputting the density values.

This decomposition offers several significant advantages:

1. **Flexibility:** By parameterizing the depth and width of each stage, we create a highly flexible architecture that can adapt to the specific requirements of different scenes.
2. **Efficiency:** The modular structure allows for more efficient exploration of the architecture space during the search process.
3. **Interpretability:** The distinct stages make it easier to analyze and understand the role of different parts of the network in the overall scene representation.
4. **Generalizability:** This structure can be easily adapted to other NeRF variants, making our approach widely applicable.

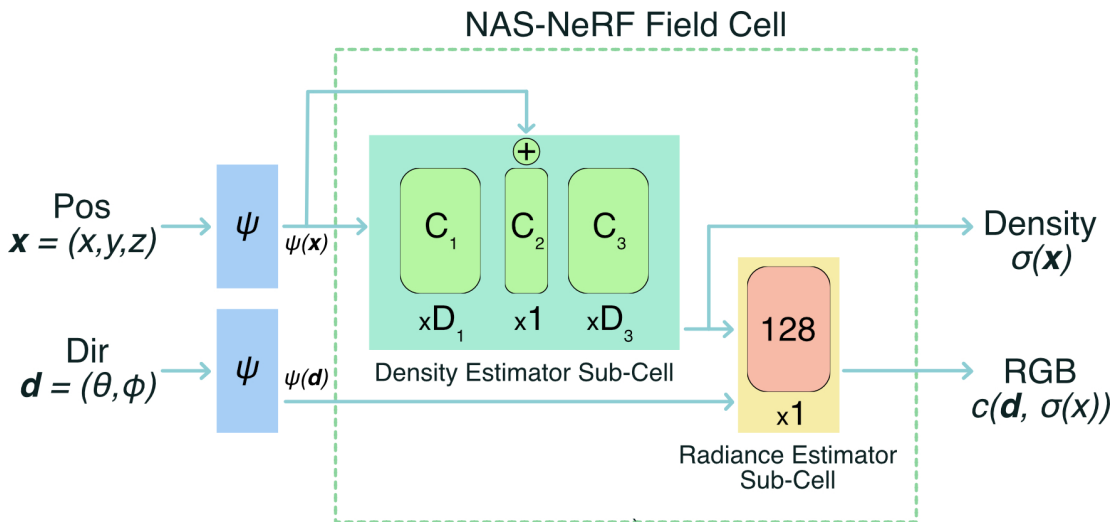


Figure 4.1: High-level overview of the NAS-NeRF field cell architecture. The NAS-NeRF pipeline incorporates the original NeRF components [2], utilizing two NAS-NeRF field cells for coarse and fine hierarchical sampling. Each cell is parameterized by depth stages (D_1, D_3) and channels (C_1, C_2, C_3), where ψ denotes the positional encoder for input coordinates or viewing directions. This parameterization ensures that the NAS-NeRF field cell can be integrated into most NeRF variants, as our optimizations focus entirely on the core network architectures.

By reformulating NeRF architecture search as the task of learning specialized cells, we create a powerful framework for optimization. The NeRF field effectively becomes an optimizable cell, similar to the approach used in other successful neural architecture search applications [39].

Furthermore as shown in Figure 4.2, this cell-based approach allows us to view the overall NAS-NeRF architecture as a series of two configurable field cells: a coarse field cell responsible for learning the global structure of the scene, and a fine field cell that captures high-frequency details. This hierarchical structure aligns well with the multi-scale nature of 3D scenes and allows for efficient allocation of computational resources.

A key advantage of our modular parameterization is that it focuses optimization efforts on the core NeRF network while decoupling peripheral components such as samplers or encoders. This means that our search strategy can function as a plugin, applicable to various NeRF methods by optimizing the architecture of the density estimator sub-cell. This plug-and-play nature significantly enhances the potential impact and adoption of our approach across different NeRF implementations.

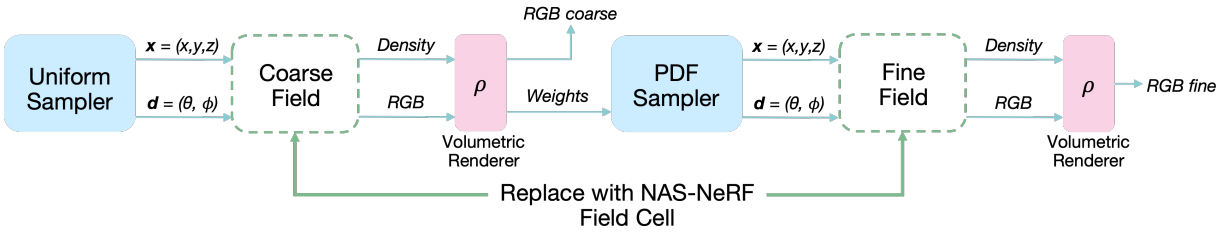


Figure 4.2: The NAS-NeRF pipeline incorporating two configurable field cells (coarse and fine) within the broader NeRF framework. This modular approach allows for targeted optimization of the core network architecture while maintaining compatibility with various NeRF implementations through a plug-and-play design. Figure is modified based on the pipeline presented in the documentation for the Nerfstudio [7] framework.

In the following sections, we will explore the details of our generative neural architecture search strategy and how it leverages this modular cell structure to discover efficient, scene-specific NeRF architectures.

4.1.3 Generative Neural Architecture Search

To discover NAS-NeRF architectures that optimally balance efficiency and novel view synthesis quality, we employ a generative neural architecture search approach based on generative synthesis [44]. This method represents a significant departure from traditional neural architecture search techniques, offering a more flexible and targeted approach to architecture optimization.

Formulation of the Search Problem

The core of our approach lies in formulating the architecture search as a constrained optimization problem:

$$\mathcal{G} = \max_{\mathcal{G}} \mathcal{U}(\mathcal{G}(s)) \quad \text{subject to} \quad 1_r(G(s)) = 1, \ \ \forall s \in \mathcal{S}. \tag{4.1}$$

In this formulation:

- $\mathcal{G}(\cdot)$ represents the generator function we aim to learn.

- $s \in S$ are seeds from which network architectures are generated.
- $\mathcal{U}$ is a universal performance metric (e.g., NetScore [45]).
- $1_r(\cdot)$ is an indicator function enforcing operational constraints.

This approach allows us to generate network architectures $\{N_s | s \in S\}$ that maximize performance while adhering to specific operational requirements. The optimization process involves an iterative collaboration between a generator G and an inquisitor I. The inquisitor evaluates generated architectures and guides the generator towards improving its performance in meeting operational requirements.

Tailoring the Search Space for NeRF

While the NAS-NeRF search space is inherently flexible, we introduce carefully crafted design constraints through the indicator function $1_r(\cdot)$ in Eq. 4.1. These constraints are crucial in achieving the desired balance between:

1. **Accuracy:** Ensuring high-quality novel view synthesis.
2. **Architectural complexity:** Controlling the overall size and structure of the network by putting constraints on the number of parameters.
3. **Computational complexity:** Optimizing for efficient inference and rendering by putting constraints on the number of floating-point operations (FLOPs).

Specifically, our constraints encourage:

1. **Diversity in channel distribution:** We allow for both uniform and variable numbers of channels across different MLP stages. This flexibility enables the network to allocate resources more efficiently based on the complexity of different parts of the scene representation task.
2. **Progressive width expansion:** We favor architectures where MLP stages become progressively wider. This design choice is motivated by the observation that deeper layers in neural networks often benefit from increased width to capture more complex, high-level features.

3. **Scene-specific performance targets:** We enforce minimum scene-specific targets T for performance metrics. This ensures that the generated architectures are not only efficient but also meet the quality requirements for each specific scene.

These constraints guide the search towards low-footprint, compact NeRF architectures that are tailored for specific performance targets, addressing the core challenges of efficiency and adaptability in NeRF models.

Iterative Search Process

Our search process, illustrated in Figure 4.3, is iterative and adaptive. This approach allows us to refine our architecture candidates progressively, ensuring that we explore a wide range of possibilities while converging towards optimal solutions.

In our experiments, we use the Structural Similarity Index (SSIM) [16] as the primary performance metric. SSIM is particularly suitable for this task as it correlates well with human perception of image quality, which is crucial for novel view synthesis applications.

For each scene, we sample three SSIM targets, corresponding to three different efficiency-quality trade-off points:

- **NAS-NeRF XXS:** Optimized for maximum efficiency with acceptable quality.
- **NAS-NeRF XS:** Balanced trade-off between efficiency and quality.
- **NAS-NeRF S:** Prioritizing quality while still maintaining improved efficiency over the baseline.

This multi-target approach allows us to generate a family of NAS-NeRF variants for each scene, providing flexibility in deployment based on specific computational constraints and quality requirements.

Addressing the Challenge of Scene-Specific Targets

A key challenge in our approach is determining suitable performance targets for new, unseen scenes. We cannot know a priori what level of performance is achievable for a given scene with a given computational budget. To address this, we developed an efficient proxy heuristic that is both fast and extensible to any scene or NeRF method.

Our heuristic, illustrated in Figure 4.4, involves the following steps:

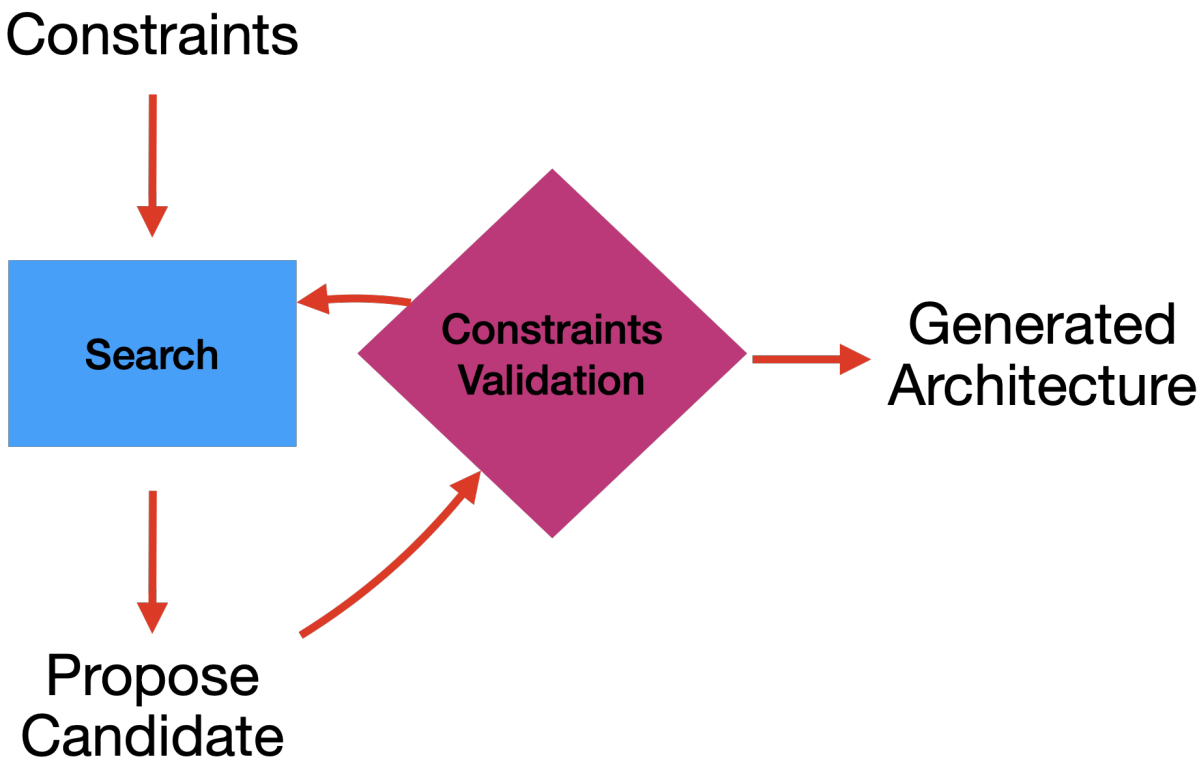


Figure 4.3: Iterative process of generative neural architecture search for NAS-NeRF. The generator proposes architectures, which are then evaluated by the inquisitor. This feedback loop allows for progressive refinement of the generated architectures, balancing performance and efficiency constraints.

1. Train two boundary architectures for a limited number of iterations (16k in our experiments):
    - A_{max}**:** A maximum size architecture
    - A_{min}**:** A minimum size architecture ($> 23.2\times$ fewer parameters than the baseline)
2. Evaluate the SSIM for both architectures: $SSIM_{min}$ and $SSIM_{max}$
3. Use linear interpolation to establish a performance spectrum between $SSIM_{min}$ and $SSIM_{max}$
4. Set target SSIM values along this spectrum for our three model variants:
    - $T_{XXS} = SSIM_{min} + 0.1 \times (SSIM_{max} - SSIM_{min})$
    - $T_{XS} = SSIM_{min} + 0.5 \times (SSIM_{max} - SSIM_{min})$
    - $T_S = SSIM_{min} + 0.9 \times (SSIM_{max} - SSIM_{min})$

This linear interpolation allows us to quickly estimate achievable performance levels across the range of possible architectures. By setting T_{XXS}, T_{XS}, and T_S at 10%, 50%, and 90% of this range respectively, we create targeted constraints for our search process that are tailored to the specific scene.

The XXS target aims for a highly efficient model with acceptable quality, the XS target balances efficiency and quality, while the S target prioritizes quality while still maintaining improved efficiency over the baseline. This approach enables us to explore a range of efficiency-quality trade-offs specific to each scene, guiding our generative search towards architectures that meet these targeted performance levels.

This process typically takes 15-60 minutes on an Nvidia RTX A6000 GPU, making it a practical approach for initializing our search process. While we acknowledge that the relationship between SSIM and model size may not be strictly linear along the true Pareto frontier, our experiments show that this heuristic provides a reasonable approximation of achievable performance bounds, allowing us to quickly bootstrap our search process.

We note that this heuristic approach represents a trade-off between search efficiency and optimality. A more exhaustive search might yield marginally better results but at the cost of significantly increased computational time. Our approach strikes a balance, allowing for rapid adaptation to new scenes while still producing high-quality, efficient architectures.

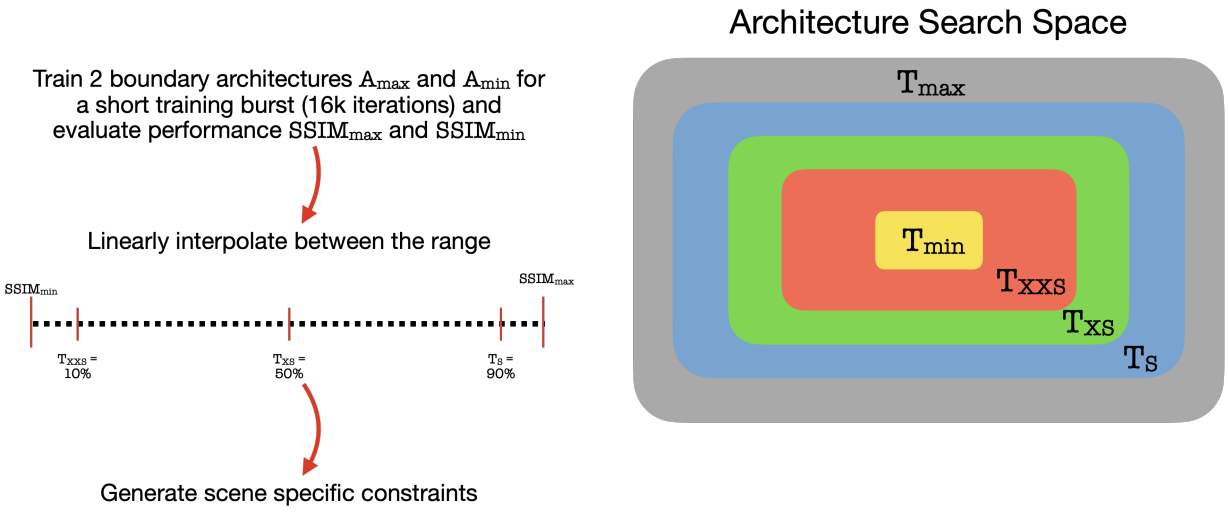


Figure 4.4: Illustration of our efficient heuristic for determining scene-specific performance targets. By training boundary architectures and interpolating between their performance, we quickly establish reasonable targets for our search process.

After establishing these scene-specific SSIM targets and incorporating them into our search constraints, we leverage the power of generative synthesis to iteratively discover tailored models. This process allows us to explore a range of architectural complexities, each optimized for the specific requirements of the scene at hand.

This scene-adaptive approach represents a significant advancement over traditional one-size-fits-all NeRF architectures. By tailoring the network structure to the complexity and characteristics of each individual scene, NAS-NeRF can achieve superior efficiency without compromising on the quality of novel view synthesis.

4.2 Experiments and Results

4.2.1 Experimental Setup

We evaluated NAS-NeRF on the Blender synthetic dataset [2], a standard benchmark in the NeRF literature. Our evaluation metrics included Structural Similarity Index (SSIM) [16], Peak Signal-to-Noise Ratio (PSNR), and Learned Perceptual Image Patch Similarity (LPIPS) [17], comparing our results against the original NeRF [2] baseline.

To ensure fair comparisons, we scaled the number of training iterations for each architecture proportionally to its parameter efficiency ratio (ER_{params}), relative to the 200,000 iterations used for the baseline NeRF. We define the architecture efficiency ratio as:

$$ER_{metric} = \frac{Metric(A_{baseline})}{Metric(A_{generated})} \tag{4.2}$$

This approach ensures that each architecture, regardless of its size, receives a comparable amount of training relative to its complexity.

We implemented NAS-NeRF using the Nerfstudio framework (v0.2.1) [7], maintaining consistency by using the default hyperparameters for the Vanilla NeRF model, including the RAdam optimizer with a learning rate of 5×10^{-4}. To promote reproducibility and further research, we have made our implementation publicly available[1].

4.2.2 Performance Analysis

As illustrated in Figure 4.5 and detailed in Table 4.1, the NAS-NeRF generated architectures demonstrate remarkable efficiency while maintaining high performance:

- **NAS-NeRF S variants:** These models match or surpass the baseline NeRF in terms of performance metrics, despite having 2-4× fewer parameters and achieving rendering speeds up to 1.93× faster.
- **NAS-NeRF XXS variants:** The most compact models in our suite, these architectures achieve SSIM scores only 5.3% lower than the baseline NeRF, while requiring 23× fewer parameters and offering a 4.7× speedup in rendering.

Qualitative comparisons, as shown in Figure 4.6, reveal that the NAS-NeRF S and XS variants can produce rendering quality comparable to the baseline. The XXS models, while showing some loss in high-frequency details, still maintain impressive quality considering their drastically reduced parameter count.

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

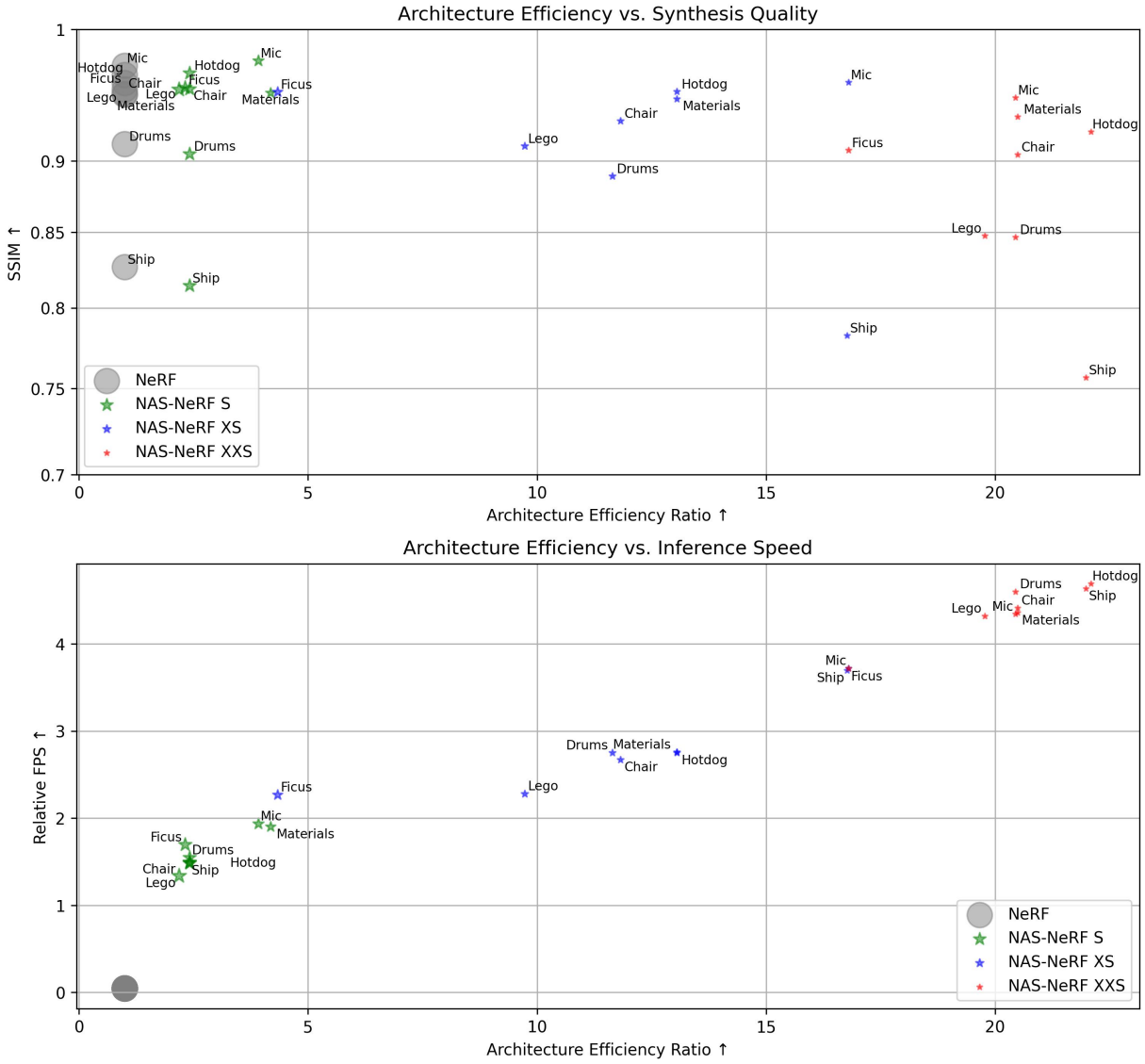


Figure 4.5: Architecture efficiency ratio (a measure of generated architecture FLOPs relative to baseline NeRF) vs. (top) synthesis quality (SSIM) and (bottom) inference speed; size $\propto$ parameter count.



Figure 4.6: 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.

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

4.2.3 Architectural Insights

Table 4.2 provides a detailed breakdown of the generated NAS-NeRF architectures. Key observations include:

- **Compactness:** The XXS variants achieve up to a 23× reduction in parameters compared to the baseline NeRF.

- **Computational Efficiency:** Significant reductions in FLOPs are observed across all variants:
  - **S variants:** 2.19-4.19× fewer FLOPs
  - **XS variants:** 4.34-16.8× fewer FLOPs
  - **XXS variants:** 16.8-22.1× fewer FLOPs
- **Rendering Speed:** The most compact models achieve speedups of up to 4.7× in frames per second.

4.2.4 Architectural Patterns and Scene Adaptation

Analysis of the generated architectures reveals intriguing patterns:

- **Coarse Field Consistency:** Across different scenes and model sizes, the coarse field tends to converge on a minimal structure. This suggests that a simple representation is sufficient for capturing global scene structure.
- **Fine Field Diversity:** In contrast, the fine fields exhibit more variation, particularly in channel widths. This aligns with the fine field's role in capturing high-frequency details, which are more scene-dependent.
- **Depth Stability:** The network depths for each stage remain relatively consistent across scenes and model sizes, indicating that the optimal depth for NeRF architectures may be more universal than width.
- **Progressive Expansion:** Many fine fields demonstrate non-uniform widths with progressive channel expansion across stages. This pattern suggests that later stages benefit from increased capacity to process and refine features.

The variation in fine field architectures across scenes demonstrates NAS-NeRF's ability to adapt to scene-specific requirements. A notable example is the architectural similarity between the Ficus scene at XXS scale and the Mic scene at XS scale, highlighting the method's flexibility in finding efficient solutions across different scenes and quality targets.

4.2.5 Discussion and Implications

The results of our experiments have several important implications for the field of neural radiance fields and 3D scene representation:

- **Efficiency-Quality Trade-off:** NAS-NeRF demonstrates that significant reductions in model size and computational requirements are possible without substantial sacrifices in rendering quality. This challenges the assumption that large, complex models are necessary for high-quality novel view synthesis.
- **Scene-Specific Optimization:** The variability in generated architectures across scenes underscores the potential benefits of moving away from one-size-fits-all approaches in NeRF models. Scene-specific optimization could lead to more efficient deployment of NeRF technology across a wide range of applications.
- **Scalability:** The ability to generate a spectrum of models (S, XS, XXS) for each scene provides a flexible framework for deploying NeRF in various computational environments, from high-performance servers to resource-constrained edge devices.
- **Generalizability:** The success of NAS-NeRF in finding efficient architectures for the diverse scenes in the Blender dataset suggests that this approach could be valuable for a wide range of 3D scene types and complexities.
- **Computational Resource Allocation:** The consistent pattern of minimal coarse fields and more variable fine fields suggests that computational resources in NeRF models might be more efficiently allocated by focusing on the refinement of high-frequency details rather than global structure representation.

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 4.2: The architecture configuration and performance metrics of each generated NAS-NeRF is shown, along with the baseline NeRF [2]. The architecture configurations for NAS-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.

Chapter 5

Conclusions

5.1 Summary of Contributions

Neural radiance fields (NeRFs) have revolutionized the field of novel view synthesis by enabling high-quality 3D scene representation and rendering. However, their substantial computational complexity hinders widespread deployment, particularly in resource-constrained environments. This thesis addresses a gap in the current state of NeRF research: the lack of scene-specific optimization in neural network architectures.

We introduced NAS-NeRF, a novel approach that employs generative neural architecture search to discover compact, scene-specialized NeRF architectures. Our method makes several key contributions:

1. **Scene-Specific Optimization:** NAS-NeRF moves beyond the one-size-fits-all approach common in existing NeRF implementations, tailoring network architectures to the complexity of individual scenes.

2. **Quality-Constrained Architecture Generation:** Our method incorporates constraints on target metrics and computational budgets, guiding the search towards architectures that balance efficiency and synthesis quality.

3. **Modular Parameterization:** We reformulated the NeRF architecture into configurable field cells, enabling efficient exploration of the architecture space while maintaining compatibility with various NeRF variants.

4. **Efficient Search Strategy:** NAS-NeRF employs a generative synthesis approach, allowing for rapid discovery of optimal architectures without the need for extensive training of candidate models.

Experiments on the Blender synthetic dataset demonstrated the effectiveness of our approach:

- NAS-NeRF S variants achieved comparable or superior performance to baseline NeRF while being up to 5.74× smaller, with 4.19× fewer FLOPs, and 1.93× faster inference on a GPU.
- NAS-NeRF XXS variants, optimized for maximum efficiency, resulted in architectures up to 23× smaller, with 22× fewer FLOPs, and 4.7× faster than baseline NeRF, with only a 5.3% average drop in SSIM.

These results challenge the assumption that large, complex models are necessary for high-quality novel view synthesis and demonstrate the potential for significant efficiency gains through scene-specific optimization.

5.2 Future Research Directions

The success of NAS-NeRF opens up several promising avenues for future research:

1. **Meta-Learning for Target Selection:** While our current approach uses a proxy heuristic to guide the search and establish target performance constraints, future work could leverage meta-learning techniques to dynamically learn these targets. By conditioning the target selection on scene-specific signals, we could potentially improve the accuracy of the search process and generate more Pareto-optimal architectures.
2. **Integration with Advanced NeRF Variants:** Applying the NAS-NeRF methodology to more sophisticated NeRF models, such as those using Instant-NGP as a backbone, could demonstrate the complementary nature of our approach. This would show that NAS-NeRF can work in conjunction with advanced sampling mechanisms and encodings, further pushing the boundaries of efficiency in novel view synthesis.

3. **Exploration of Dynamic Architectures:** Future research could investigate the potential for dynamically adjusting network architectures based on the complexity of different regions within a scene. This could lead to even more efficient resource allocation and potentially enable real-time adaptation of model capacity.

4. **Application to Diverse Datasets:** While our experiments focused on the Blender synthetic dataset, future work should explore the effectiveness of NAS-NeRF on more diverse and challenging real-world scenes. This could provide insights into how the method scales to different types of scene complexities and lighting conditions.

5. **Multi-Task Optimization:** Extending NAS-NeRF to jointly optimize for multiple tasks beyond novel view synthesis, such as semantic segmentation or depth estimation, could lead to more versatile and efficient 3D scene understanding models.

By addressing these research directions, we can further advance the field of neural radiance fields, making them more efficient, adaptable, and applicable to a wider range of practical scenarios. The principles behind NAS-NeRF have the potential to influence not only the development of NeRF models but also the broader field of neural network optimization for 3D vision tasks.

References

[1] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[2] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 405–421. Springer International Publishing. ISBN 978-3-030-58452-8. doi: 10.1007/978-3-030-58452-8_24.

[3] SC Chan, Heung-Yeung Shum, and King-To Ng. Image-based rendering and synthesis. *IEEE Signal Processing Magazine*, 24(6):22–33, 2007.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[5] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[6] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

[7] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023.

[8] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, 120:153–168, 2016.

[9] Oleg Voynov, Gleb Bobrovskikh, Pavel Karpyshev, Saveliy Galochkin, Andrei-Timotei Ardelean, Arseniy Bozhenko, Ekaterina Karmanova, Pavel Kopanev, Yaroslav Labutin-Rymsho, Ruslan Rakhimov, Aleksandr Safin, Valerii Serpiva, Alexey Artemov, Evgeny Burnaev, Dzmitry Tsetserukou, and Denis Zorin. Multi-sensor large-scale dataset for multi-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21392–21403, June 2023.

[10] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022.

[11] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1790–1799, 2020.

[12] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[13] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[14] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Krishna Wadhwani and Tamaki Kojima. Squeezenerf: Further factorized fastnerf for memory-efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2717–2725, 2022.

[21] 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.

[22] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[23] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *Computer Graphics Forum*, volume 40, pages 45–59. Wiley Online Library, 2021.

[24] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. NerfAcc: Efficient sampling accelerates NeRFs. URL http://arxiv.org/abs/2305.04966.

[25] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[26] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[27] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021.

[28] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2846–2855, 2021.

[29] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[30] Bipasha Sen, Gaurav Singh, Aditya Agarwal, Rohith Agaram, Madhava Krishna, and Srinath Sridhar. Hyp-nerf: Learning improved nerf priors using a hypernetwork. *Advances in Neural Information Processing Systems*, 36, 2024.

[31] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021.

[32] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[33] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022.

[34] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8254–8263, 2023.

[35] Nagabhushan Somraj and Rajiv Soundararajan. Vip-nerf: Visibility prior for sparse input neural radiance fields. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[36] Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE computer society conference on computer vision and pattern recognition*, pages 358–363. Ieee, 1996.

[37] 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.

[38] 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.

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

[40] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[41] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[42] Alexander Wong, Mohammad Javad Shafiee, Brendan Chwyl, and Francis Li. Ferminets: Learning generative machines to generate efficient neural networks via generative synthesis. *arXiv preprint arXiv:1809.05989*, 2018.

[43] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. pages 5855–5864. URL https://openaccess.thecvf.com/content/ICCV2021/html/Barron_Mip-NeRF_

A_Multiscale_Representation_for_Anti-Aliasing_Neural_Radiance_Fields_ICCV_2021_paper.html.

[44] 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.

[45] 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.

[46] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[47] Donald Knuth. *The TeXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[48] Leslie Lamport. *LaTeX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[49] NVIDIA Corporation. Nvidia tensorrt, 2022.

[50] The Linux Kernel Organization. perf: Linux profiling with performance counters, 2022.

[51] Saad Abbasi, Alexander Wong, and Mohammad Javad Shafiee. Maple: Microprocessor a priori for latency estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[52] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[53] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[54] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, et al. Chamnet: Towards efficient network design through platform-aware model adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11398–11407, 2019.

[55] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[56] Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. *Advances in Neural Information Processing Systems*, 33:10480–10490, 2020.

[57] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2961–2969, 2017.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[59] Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. Hardware-adaptive efficient latency prediction for nas via meta-learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[60] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, and Yingyan Lin. Hw-nas-bench: Hardware-aware neural architecture search benchmark. *arXiv preprint arXiv:2103.10584*, 2021.

[61] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

[62] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[63] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[65] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[66] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[67] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[68] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12965–12974, 2020.

[69] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*, 2020.

[70] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[71] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Bowen Shi, Qi Tian, and Hongkai Xiong. Latency-aware differentiable neural architecture search. *arXiv preprint arXiv:2001.06392*, 2020.

[72] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. Nn-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–93, 2021.

[73] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[74] 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.

[75] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. pages 5501–5510. URL https://openaccess.thecvf.com/content/CVPR2022/html/Fridovich-Keil_Plenoxels_Radiance_Fields_Without_Neural_Networks_CVPR_2022_paper.html.

[76] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. EfficientNeRF efficient neural radiance fields. pages 12902–12911. URL https://openaccess.thecvf.com/content/CVPR2022/html/Hu_EfficientNeRF_Efficient_Neural_Radiance_Fields_CVPR_2022_paper.html.

[77] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[78] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers and distillation through attention, 2021.

[79] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit, 2022.

[80] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[81] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution, 2022.

[82] Ali Hatamizadeh, Hongxu Yin, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Global context vision transformers, 2023.

[83] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, Jitendra Malik, Yanghao Li, and Christoph Feichtenhofer. Hiera: A hierarchical vision transformer without the bells-and-whistles, 2023.

[84] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers, 2021.

[85] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer, 2022.

[86] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers, 2022.

[87] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers, 2021.

[88] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers, 2021.

[89] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization, 2023.

[90] Ali Hatamizadeh, Greg Heinrich, Hongxu Yin, Andrew Tao, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. Fastervit: Fast vision transformers with hierarchical attention, 2023.

[91] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for on-device semantic segmentation, 2023.

[92] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention, 2023.

[93] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers, 2021.

[94] Alexander Wong, Yifan Wu, Saad Abbasi, Saeejith Nair, Yuhao Chen, and Mohammad Javad Shafiee. Fast graspnext: A fast self-attention neural network architecture for multi-task learning in computer vision tasks for robotic grasping on the edge, 2023.

[95] Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Fast vision transformers with hilo attention, 2023.

[96] Alexander Wong. Netscore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage. *arXiv preprint arXiv:1806.05512*, 2018.