ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 38

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

NEURAL ARCHITECTURE SEARCH FOR COMPUTER VISION TASKS

Presentata da: Alessio Mingozzi

Coordinatore Dottorato

Paola Salomoni

Supervisore

Matteo Poggi

Co-supervisore

Luigi Di Stefano
Fabio Masci

Esame finale anno 2026

Abstract

Automated Machine Learning (AutoML) accelerates the deep model design cycle by reducing dependence on manual, expert-crafted architectures. Within this broader field, Neural Architecture Search (NAS) is the subcategory dedicated to automating the generation of neural network architectures. NAS has demonstrated success across various domains, including image classification, natural language processing, and reinforcement learning. Yet, a persistent gap remains between generic NAS benchmarks and the nuanced demands of specialized Computer Vision tasks that require strong generalization under data scarcity, geometric consistency, or rapid adaptation to novel scenes. Moreover, the high computational cost of NAS limits its practical adoption, especially in resource-constrained settings. Recent advances in zero-cost and near-zero-cost performance proxies offer promising avenues to alleviate this burden by estimating model quality without extensive training. However, these proxies often struggle to generalize across diverse tasks and complex architectures, highlighting the need for more robust and domain-agnostic metrics. This thesis investigates how to further automate the neural network design process for diverse vision problems through low-cost NAS. The central objective is to evaluate and extend zero-cost or near-zero-cost performance proxies, metrics that can be computed before or after minimal training, to guide search over architectural spaces tailored to three challenging domains: (i) deep stereo matching, where structural regularization and correspondence reasoning are critical; (ii) generalizable Neural Radiance Fields (NeRF), where view synthesis quality and cross-scene transfer dominate; and (iii) small object detection, where representational efficiency and scale-sensitive feature aggregation are decisive.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Neural Architecture Search

With the widespread adoption of machine learning, the design of neural network architectures has become a pivotal factor influencing both the accuracy and efficiency of models. The architecture determines not only the representational capacity of a network but also its computational requirements and ability to generalize across tasks. Traditionally, crafting neural architectures has been a manual, iterative process that demands substantial domain expertise and time investment. Researchers often rely on intuition, experience, and extensive trial-and-error to identify optimal configurations, which can significantly slow the pace of innovation and limit accessibility for non-experts.

To address these challenges, the field of Automated Machine Learning (AutoML) [1] has emerged, aiming to automate various stages of the machine learning pipeline, including model selection, hyperparameter tuning, and architecture design. Within AutoML, Neural Architecture Search (NAS) [8] specifically focuses on automating the discovery of effective neural network architectures. NAS frameworks systematically explore vast architectural spaces, leveraging search algorithms to identify high-performing models with minimal human intervention.

NAS has demonstrated remarkable success in domains such as image classification [9, 10], object detection [11] and semantic segmentation [12], often surpassing manually designed architectures. The main advantage of NAS lies in its ability to uncover novel and efficient designs that might be overlooked by human experts. However, despite these achievements, a notable gap persists between the generic NAS benchmarks and the nuanced requirements of specialized computer vision tasks. Many vision problems demand architectures that excel under constraints such as limited data availability, geometric consistency, or the need for rapid adaptation to

new environments. Addressing these challenges requires NAS methods that are not only efficient but also capable of tailoring architectures to the unique demands of each task, motivating ongoing research into more principled approaches.

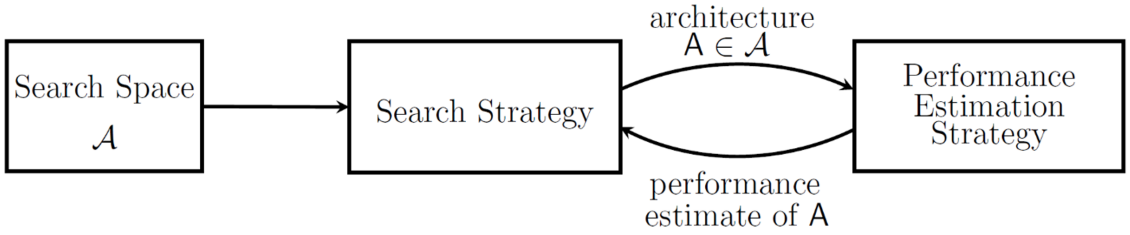


Figure 1.1: High-level NAS pipeline. High-level overview of the Neural Architecture Search pipeline. The search space defines candidate modules and connections, the search strategy explores candidate architectures, and the performance estimation strategy ranks candidates.

To explain how NAS works, we can break down the process into three main components: the search space, the search strategy and the performance estimation strategy, as illustrated in Figure 1.1. Each component plays a crucial role in determining the efficiency and effectiveness of the NAS process.

1.1.1 Search Space

The search space in Neural Architecture Search (NAS) defines the universe of possible architectures that can be constructed and evaluated. It encompasses all the components and design choices available for building neural networks, such as the types of layers (e.g., convolutional, pooling, normalization), the number and arrangement of these layers, the connectivity patterns (e.g., skip connections, residual links), and associated hyperparameters (e.g., kernel sizes, activation functions, number of filters). The richness and flexibility of the search space directly influence the diversity and quality of architectures that NAS can discover. An high level illustration of it is shown in Figure 1.2. Any network can be seen as a graph of nodes and edges, where nodes represent layers and edges represent connections between them. By parametrizing this graph, we can define a search space that includes various configurations of layers and connections.

A well-designed search space balances expressiveness with size. If the search space is too broad, incorporating every conceivable layer type and connection, the search process becomes computationally infeasible, as the number of candidate architectures grows exponentially. Conversely, an overly restrictive search space may prevent the discovery of novel or high-performing designs. Therefore, researchers often tailor the

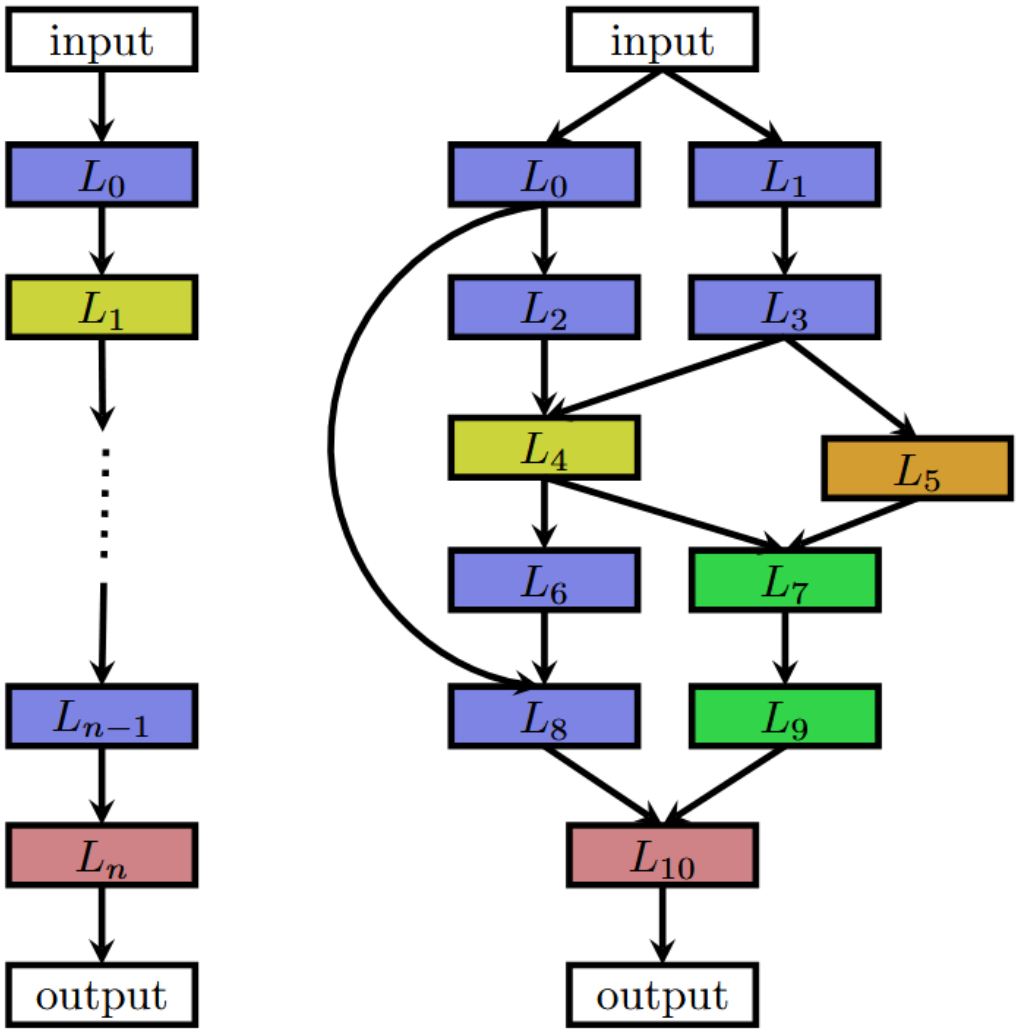


Figure 1.2: Search space design. Example of search space design [1]. Each node in the graph represents a possible layer. On the left, a simple chain architecture is displayed, while on the right, a more complex architecture with skip connections is shown. The search space defines the possible configurations of these layers and connections.

search space to the specific requirements of the target task or domain, incorporating prior knowledge to guide the search toward promising regions.

In the literature, two primary paradigms for defining the search space have emerged: global search space and cell-based (or modular) search space. The global search space treats the architecture as a sequence of operations, allowing the search algorithm to freely combine different layers and connections throughout the entire network. This approach offers maximum flexibility but can be prohibitively expensive due to the vast number of possible configurations.

The cell-based search space, on the other hand, focuses on discovering small, repeatable building blocks, referred to as "cells", that are stacked or arranged in a predefined manner to form the final architecture. Each cell is typically a directed acyclic graph composed of a limited set of operations, and the overall network is constructed by repeating these cells. This modular approach dramatically reduces the complexity of the search space while enabling efficient transferability of discovered cells across different tasks and datasets. An example of this paradigm is shown in Figure 1.3.

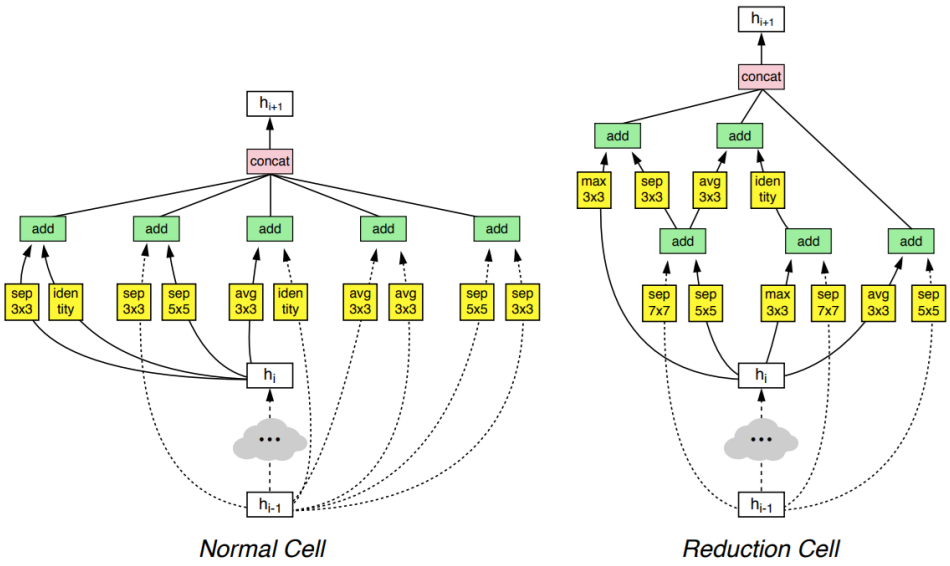


Figure 1.3: NASNet cell design. Example of NASNet [2] cell-based architecture design. The cell search space focuses on discovering small, repeatable building blocks (cells) that are stacked to form the final neural network, enabling efficient architecture search and transferability across tasks.

Beyond these two paradigms, recent research has explored hybrid and task-specific search spaces, introducing additional constraints or domain-specific modules to better address the unique challenges of specialized applications. For example, search spaces for vision tasks may incorporate multi-scale feature aggregation, attention

mechanisms, or geometric priors, while those for sequence modeling might emphasize recurrent or transformer-based components. The careful design of the search space is thus a critical factor in the success of NAS, as it determines both the feasibility of the search and the relevance of the resulting architectures to the problem at hand. While full automation of search space design, where there are no predefined structures or constraints, remains an open challenge, ongoing research continues to explore methods for freeing NAS search spaces from human biases and limitations.

1.1.2 Search Strategy

The search strategy defines how to explore the search space. This can be done through various methods, including random search, evolutionary algorithms, reinforcement learning, and gradient-based methods. Different strategies have been proposed to balance exploration and exploitation, as well as to handle the trade-off between search time and the quality of the discovered architectures. Bayesian optimization was used successfully in Hyperparameter Optimization and subsequently in NAS even before it was popular. The search strategy represents a critical aspect of the NAS process, as it directly influences the speed and success of finding optimal architectures within the defined search space. The simplest method is random search, which samples architectures uniformly at random from the search space. While this approach is easy to implement and can be effective in some cases, it may require a large number of evaluations to find high-performing architectures. One of the first popular approach was reinforcement learning [13], where a controller (often a recurrent neural network) generates architectures and receives feedback based on their performance. The controller is trained to maximize the expected reward, guiding the search towards promising regions of the search space. Other approaches included evolutionary algorithms, which use principles of natural selection to evolve a population of architectures over generations.

These algorithms typically involve mutation and crossover operations to explore the search space in order to generate diverse candidate architectures which form a population, from which a selection mechanisms retain the best-performing architectures, usually though the means of some tournament selection approach based on their validation performance. This approach became widely adopted after [14] showed how it performed comparability with reinforcement learning, while having better anytime performance and finding smaller architectures.

All these methods have their strengths and weaknesses, and the choice of search strategy often depends on the specific requirements of the task, the size of the search space, and the available computational resources.

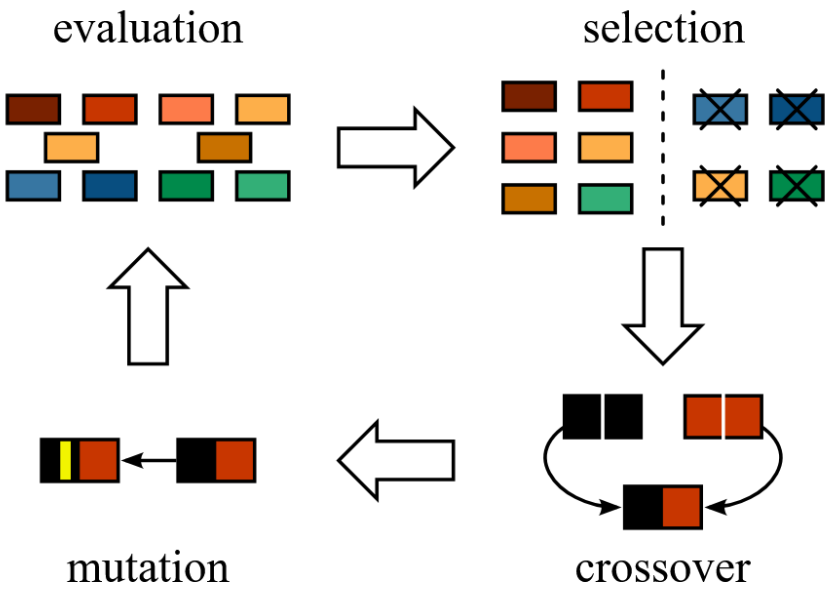


Figure 1.4: Evolutionary NAS. The evolutionary process. A population of candidates is evolved over generations using mutation and crossover operations, with selection based on performance to guide the search towards optimal candidates.

1.1.3 Performance Estimation Strategy

The performance estimation strategy determines how the quality of candidate architectures is assessed during NAS. Traditionally, this involves fully training each architecture and evaluating its accuracy on a validation set, a process that is computationally intensive and often impractical for large search spaces. Early NAS methods leveraged clusters of thousands of GPUs to parallelize and speed up the evaluation process. Also the choice of performance estimation strategy is crucial, as it directly affects both the reliability of the search and the computational resources required. All the candidate are evaluated and ranked based on their estimated performance, guiding the search strategy towards more promising architectures 1.5.

Researchers have proposed various efficient strategies to enable rapid exploration of the search space, using lower fidelity approximations (proxy metrics) to relax the need for exhaustive training, while maintaining a strong correlation with true model performance.

1.2 Efficient NAS

Training and evaluating candidate architectures in NAS is computationally expensive, often requiring thousands of GPU hours to identify optimal designs. This bottleneck

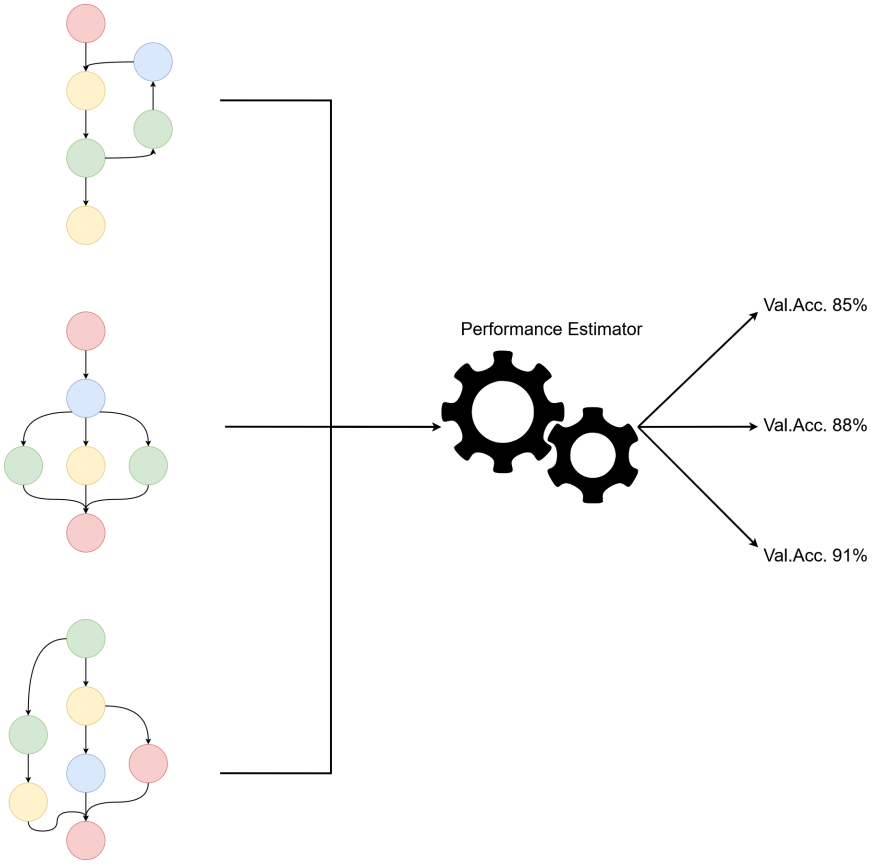


Figure 1.5: Performance estimation strategies. Evaluation of the fully trained model provides accurate evaluations but is computationally expensive. Proxy methods could offer faster estimates with varying degrees of accuracy.

arises because each candidate typically needs to be trained from scratch to reliably estimate its performance. To mitigate this challenge, several strategies have been developed to reduce the computational burden of NAS. Early methods focused on reducing the cost of performance estimation by using lower-fidelity approximations, such as training candidate architectures for fewer epochs, using smaller subsets of data, or employing reduced numerical precision. While these techniques accelerate the search, they can introduce additional noise and reduce the accuracy of performance estimates, sometimes resulting in sub-optimal architecture choices. Other approaches leverage early stopping or learning curve extrapolation, aiming to predict the final performance of a model based on its initial training trajectory. By estimating outcomes from early training signals, these methods further reduce the need for exhaustive training of each candidate.

One of the most influential approaches is One-Shot NAS. In this paradigm, a SuperNet is constructed to encompass all possible components and connections defined by the search space. Instead of training each candidate architecture independently,

the SuperNet is trained once, and its weights are shared among all candidate architectures. During the search, architectures are sampled from the SuperNet, inheriting the shared weights, which allows for rapid evaluation without retraining from scratch. This weight-sharing mechanism dramatically reduces the computational cost, enabling the exploration of large search spaces within feasible time-frames. Still, the weight-sharing approach can introduce biases, as the shared weights may not optimally represent all candidate architectures, potentially affecting the reliability of performance estimates.

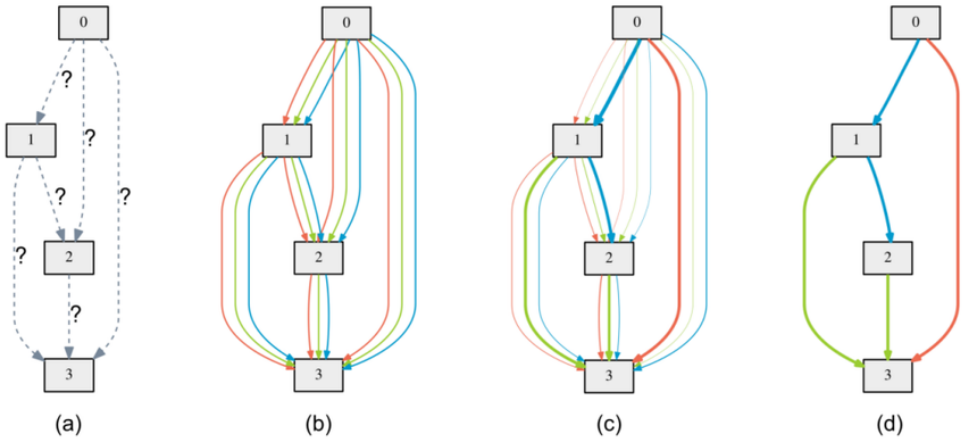


Figure 1.6: One-Shot NAS. Example of One-Shot NAS process from DARTS [3] approach. A SuperNet containing all possible components and connections is trained, and candidate architectures are derived from this shared representation, allowing for weight sharing and reduced training time.

Beyond One-Shot NAS, other efficiency-driven techniques have emerged, such as early stopping, network morphism, and the use of surrogate models. Early stopping halts training once a candidate's performance plateaus, while network morphism incrementally adapts existing architectures to new ones, reusing learned weights. Surrogate models predict the performance of architectures based on their structural characteristics, further reducing the need for exhaustive training. Collectively, these methods have shifted NAS from a prohibitively expensive process to one that is increasingly accessible and scalable. This reduced the time required to discover a candidate to the order of hundreds of GPU hours.

More recently, zero-cost and near-zero-cost performance proxies have emerged as a promising solution to further accelerate NAS. Unlike traditional approaches that require full or partial training, these proxies estimate the potential of candidate architectures without any training or after only a few gradient steps. The central idea is that certain intrinsic properties of an untrained or minimally trained network, such

as the distribution of gradients, the sharpness of the loss landscape, or the efficiency of information flow, can serve as reliable indicators of its eventual performance.

Zero-cost proxies typically operate by analyzing the architecture's response to true, random or synthetic data in its initial state, computing metrics such as the Jacobian spectrum, gradient norms, or measures of expressivity and trainability. For example, proxies like NASWOT (Neural Architecture Search Without Training) [15] evaluate the linear independence of activations, while others assess the sensitivity of outputs to input perturbations or the entropy of feature representations. Near-zero-cost proxies may involve a handful of training iterations to capture early learning dynamics, leveraging metrics like initial loss reduction or gradient alignment.

Speed-up method	**How is achieved?**
Lower fidelity estimates	Training candidate architectures for fewer epochs, using smaller subsets of data, or employing reduced numerical precision.
Learning Curve Extrapolation	Predicting final performance based on initial training trajectory.
Weight Inheritance/Network Morphism	Incrementally adapting existing architectures to new ones, reusing learned weights.
One-Shot NAS	Training a SuperNet that encompasses all candidate architectures, allowing weight sharing on subgraph derived models.
Zero-Cost Proxies	Estimating model quality using metrics computable with little or no training, often requiring only a single forward or backward pass.

Table 1.1: Summary of NAS speed-up methods. Summary of various methods to speed up Neural Architecture Search (NAS) by reducing the computational cost of performance estimation.

These methods dramatically reduce the computational burden of NAS, enabling the evaluation of thousands of candidate architectures in seconds or minutes rather than hours or days. By providing a fast yet informative estimate of model quality, zero-cost proxies facilitate rapid search and selection, making NAS feasible even for resource-constrained settings or highly specialized tasks. However, the effectiveness of these proxies can vary depending on the domain and the complexity of the search space, motivating ongoing research into more robust and generalizable metrics. In practice, the adoption of zero-cost and near-zero-cost proxies has shifted the computational requirements of NAS from hundreds or thousands of GPU hours to less than tens, democratizing access to automated architecture design.

1.3 Challenges and Motivation

Despite the remarkable progress of NAS in domains such as image classification, its broader adoption across other computer vision and machine learning tasks remains limited. This limitation stems from several interrelated challenges. First, tasks beyond image classification, such as stereo matching, 3D scene understanding, or small object detection, often demand more intricate architectural designs, including multi-branch structures, task-specific modules, or complex connectivity patterns. As a result, the corresponding search spaces become significantly larger and more heterogeneous, exacerbating the combinatorial explosion and making efficient exploration far more difficult. The increased complexity also amplifies the sensitivity of NAS outcomes to the definition of the search space, the choice of search strategy, and the performance estimation method. Small changes in any of these components can lead to vastly different results, complicating reproducibility and generalization across datasets and domains.

Another major bottleneck is the computational cost associated with NAS. Many state-of-the-art NAS algorithms require extensive GPU resources, often running for days or weeks to identify competitive architectures. This high barrier to entry restricts practical experimentation to well-funded research groups or industry labs, limiting the democratization of NAS and slowing its adoption in specialized or resource-constrained settings. While recent advances in zero-cost and near-zero-cost performance proxies have shown promise in alleviating this burden, their effectiveness is not universal. Most existing proxies are tailored to standard classification backbones and may not generalize well to architectures with non-standard modules, multi-task heads, or domain-specific layers. Furthermore, these proxies often focus on evaluating feature extraction components, neglecting the impact of downstream processing or task-specific adaptations, which are critical in complex applications.

A further challenge lies in the correlation between proxy metrics and true model performance. While some proxies demonstrate reasonable predictive power on standard benchmarks, their reliability often degrades in specialized domains where subtle architectural details can have great effects on generalization, robustness, or efficiency. This gap highlights the need for more robust, domain-agnostic proxies that can faithfully guide the search process across a wider range of tasks.

From a practical standpoint, the deployment and development of NAS methods are hindered by a lack of accessible, well maintained, and user-friendly software tools. The majority of open-source NAS frameworks are either research prototypes focused on novel algorithmic contributions or are no longer actively maintained. As a result, practitioners face significant obstacles in adapting these tools for real-

world applications, especially when dealing with custom datasets, non-standard architectures, or hardware constraints. The scarcity of modular, extensible, and well documented NAS libraries impedes both experimentation and technology transfer, slowing the pace of innovation and limiting the impact of NAS in applied settings.

Collectively, these challenges underscore the need for new methods and tools that can make NAS more efficient, generalizable, and accessible, particularly in domains where architectural requirements are complex, computational resources are limited, and practical deployment is a priority.

This thesis aims to investigate how to further automate the design process of neural networks for various vision problems through principled, low-cost NAS. Given the impractical cost of running Multi/One-shot NAS approaches, we choose to focus on zero-cost methodologies for this thesis work. The main goal is to evaluate and extend zero-cost or near-zero-cost performance proxies, metrics that can be computed before or after minimal training, to guide the search over architectural spaces tailored to three challenging domains. Specifically:

1. **Zero-Cost NAS for Deep Stereo Matching:** We introduce an application of zero-cost proxies to the domain of dense disparity estimation. By defining a search space around the feature extraction modules of the state-of-the-art RAFT-Stereo architecture, we utilize the AZ-Score proxy to discover efficient variants without full training. The resulting model, *RAFT-StereoZero*, achieves competitive accuracy with the original baseline while reducing the parameter count by approximately 90% (from 11M to 1.14M) and increasing inference speed by up to 5.4× on standard benchmarks like KITTI and Middlebury. This contribution demonstrates that zero-cost NAS is highly effective for model compression in geometric computer vision tasks.

2. **Multi-View Aware NAS for Generalizable NeRF:** We extend zero-cost NAS to the task of Generalizable Neural Radiance Fields (G-NeRF). Applying this to CaesarNeRF, we identify encoder architectures that maintain high-quality view synthesis on the LLFF and mip-NeRF 360 datasets. While emphasizing the importance of multi-view consistency in search, this study also highlights current limitations where purely architectural improvements in the encoder yield diminishing returns compared to changes in the rendering pipeline.

3. **Efficiency-Driven Search for Small Object Detection:** We tackle the challenging problem of detecting small objects (e.g., distant birds) on resource-constrained edge devices. We construct a search space based on the NanoDet

architecture, incorporating focus-oriented modifications to the Feature Pyramid Network (FPN) to preserve high-resolution details. Using zero-cost proxies combined with latency constraints, we discover an ultra-compact model (1.7M parameters) that achieves a 2× inference speedup compared to an RT-DETR baseline. Although this comes with an accuracy trade-off, this contribution provides critical insights into the balance between representational capacity and efficiency when targets occupy extremely few pixels.

These contributions collectively illustrate that the traditional development of neural architectures, often focused on maximizing accuracy first and compressing later, can be effectively complemented by automated methods that prioritize efficiency and compactness from the start.

Chapter 2

Related Work

2.1 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) has evolved rapidly, demonstrating its effectiveness across a wide range of tasks in computer vision and natural language processing. Early NAS approaches, such as those by Zoph and Le [13], leveraged reinforcement learning to train a recurrent neural network (RNN) controller that generates architectural descriptions. These pioneering methods achieved state-of-the-art results on image classification and language modeling tasks, but their practical impact was limited by the immense computational resources required, often thousands of GPU days for a single search.

Subsequent research sought to generalize and improve NAS for image classification, with works such as NASNet [2], PNAS [16], and ENAS [17] introducing new search spaces, progressive search strategies, and weight-sharing mechanisms. These innovations enabled NAS to discover competitive architectures for datasets like CIFAR-10 and ImageNet, but the computational cost remained a significant barrier to broader adoption. Other works, such as MnasNet [18], focused on optimizing architectures for target hardware platforms, such as mobile devices, balancing accuracy with latency constraints.

To address the computational cost challenge, the community developed more efficient NAS methodologies. Differentiable Architecture Search (DARTS) [19] introduced a continuous relaxation of the search space, enabling architecture optimization via gradient descent and reducing search times from thousands to just a few GPU days. DARTS inspired further efficient NAS methods, such as ProxylessNAS [9], which allows direct search on large-scale datasets and adapts architectures for specific hardware, and FBNet [10], which jointly optimizes architecture and weights. These advances have made NAS practical for real-world scenarios, including mobile and

edge devices. BANANAS [20] proposed a neural predictor-based approach, using an ensemble of neural networks and Bayesian optimization to efficiently model and search the performance landscape, achieving strong results with fewer evaluations and robust generalization across benchmarks.

The impact of efficient NAS extends beyond image classification. In stereo vision for example, DARTS-based approaches have enabled the automated discovery of architectures tailored to disparity estimation. For example, AutoDispNet [21] utilized a cell-based search space and differentiable search to produce competitive stereo matching networks in 42 GPU days. LEAStereo [22] further advanced this paradigm by extending differentiable NAS to volumetric stereo matching, achieving state-of-the-art results through hierarchical search spaces that consider both cell-level and global network structures. For object detection, a NAS discovered network EfficientNet [23] was used to create its detector counterparts, EfficientDet [11]. MobileNetV2 [24] was also discovered through NAS. Moreover YOLO-NAS applied NAS techniques to object detection, optimizing architectures for real-time performance while maintaining high accuracy.

Zero-Cost Proxies. Zero-cost proxies, also referred to as zero-shot NAS methods, have emerged as a transformative approach to drastically reduce the computational demands of Neural Architecture Search. Instead of fully training each candidate architecture, these methods estimate the potential performance of a model using metrics that can be computed with little or no training, often requiring only a single forward or backward pass. The central premise is that certain intrinsic properties of an architecture, such as its expressivity, or information flow, can serve as reliable indicators of its eventual test accuracy.

A variety of zero-cost proxies have been proposed in recent literature. For example, NASWOT [15] evaluates the linear independence of activations in untrained networks, while GradNorm [25], GradSign [26] and SynFlow [27] analyze gradient-based metrics to assess trainability. Other approaches, such as Jacobian-based proxies [28, 29], measure the sensitivity of outputs to input perturbations, and ZenNAS [30] quantifies architectural expressivity with minimal computation, enabling the identification of competitive ImageNet architectures in under 0.5 GPU days. ZiCo [31] further refines this paradigm by leveraging the inverse coefficient of variation of gradients, achieving top-tier results with only 0.4 GPU days.

Despite the promise of these methods, recent surveys [32] have revealed that many zero-cost proxies correlate with test set accuracy at levels comparable to simple structural metrics such as parameter count (#PARAMS) and floating-point operations (FLOPs). This finding highlights the challenge of designing proxies that

are both computationally efficient and highly predictive of real-world performance.

To address these limitations, AZ-NAS [33] introduces a suite of four distinct expressivity scores, each computed over a single forward pass. These scores are designed to capture complementary aspects of architectural quality, including feature diversity, activation dynamics, and information propagation. By combining multiple metrics, AZ-NAS improves the discrimination among top-performing architectures and achieves a higher correlation with test set accuracy than previous single-proxy approaches. This multi-faceted evaluation enables more robust selection of candidate models, particularly in large and diverse search spaces.

Zero-cost proxies represent a significant advancement in NAS methodology, enabling rapid and scalable architecture search even in resource-constrained environments. Ongoing research continues to refine these metrics, seeking greater generalizability across domains and improved alignment with downstream task performance.

2.2 Deep Stereo Matching

Deep stereo matching has undergone a significant transformation over the past decade, evolving from traditional hand-crafted algorithms [4] to sophisticated deep learning-based solutions. Early deep stereo methods relied on explicit feature engineering and cost aggregation, but the introduction of end-to-end architectures such as DispNetC [34] marked a paradigm shift. These models leveraged convolutional neural networks (CNNs) to directly predict disparity maps from stereo image pairs, eliminating the need for manual feature design. Subsequent advances explored both 2D [35] and 3D [36] convolutional architectures, which improved the ability to capture spatial context and geometric relationships, establishing deep learning as the dominant approach in stereo matching.

Building on these foundations, the field has diversified into several promising directions. Stereo video processing [37] extends matching to temporal sequences, enabling more robust and consistent depth estimation in dynamic scenes. Fusion with active sensors, such as LiDAR or structured light, has been explored to enhance depth accuracy and reliability, especially in challenging environments [38, 39]. Self-adapting models [40, 41, 42] leverage continual learning and online adaptation to maintain performance across changing domains and deployment scenarios. Specialized strategies have also been developed to address non-Lambertian surfaces [43, 44], depth discontinuities [45, 46], and visually imbalanced stereo pairs [47, 46, 48], each presenting unique challenges for robust disparity estimation.

Recent surveys [49] highlight the emergence of novel architectural paradigms.

Recurrent architectures, inspired by RAFT [50], have demonstrated strong generalization by iteratively refining disparity estimates using all-pair cost volumes [51]. Transformer-based models [52] capture long-range dependencies and global context, while fully data-driven Markov Random Field (MRF) approaches [53] offer flexible modeling of spatial relationships. These advances have shifted the focus from explicit design choices to architectures capable of learning complex matching functions directly from data, with RAFT-Stereo [51] exemplifying the state-of-the-art in generalization and accuracy.

Temporal consistency in stereo videos remains an active area of research, with methods such as [54] specifically targeting the alignment of depth estimates across frames. Domain generalization is another major challenge, addressed through domain-invariant feature learning [55, 56], hand-crafted matching costs [57], integration of geometric priors [58, 59], and the use of sparse depth measurements from active sensors [38, 39, 60]. Self-supervised approaches [61] have gained traction, leveraging pseudo-labels from traditional algorithms [62, 63] or neural radiance fields [62] to reduce reliance on annotated data.

Nonetheless, handling non-Lambertian surfaces remains particularly difficult due to complex reflective properties and limited ground truth data, with only a few works such as Depth4ToM [44] addressing this through semantic guidance. Integration of stereo with monocular cues [63, 64] is still limited, typically explored in self-supervised or loosely coupled frameworks. Furthermore, while deep learning has enabled remarkable improvements in accuracy and generalization, state-of-the-art architectures are often computationally intensive, making them unsuitable for deployment on low-power or real-time devices.

2.3 Generalizable NeRF

Neural Radiance Fields (NeRF). The foundational approach of Neural Radiance Fields (NeRF) [65] implicitly models the density (σ) and color (c) of points in a 3D scene as a function of their spatial position and viewing direction. This methodology enables the synthesis of photo-realistic images from arbitrary, unseen viewpoints by learning a continuous volumetric scene representation. Since its introduction, NeRF has driven significant progress in novel view synthesis, 3D scene reconstruction, and applications such as dynamic scene modeling, relighting, and scene editing. However, a key limitation of the original NeRF formulation is its scene-specific nature: each new scene requires training a separate model from scratch, which is computationally expensive and restricts scalability for large-scale or real-time applications.

Generalizable NeRF (GNeRF). To address the constraint of per-scene optimization, Generalizable Neural Radiance Fields (GNeRF) aim to decouple scene representation from the model, enabling a single network to render novel views across multiple scenes. Early GNeRF works, such as PixelNeRF [66] and GRF [67], introduced the use of image encoders to extract per-pixel feature embeddings from input images, which are then used by a NeRF-style decoder to predict density and color. These methods demonstrated the feasibility of generalizing across scenes by conditioning the radiance field on learned image features. Subsequent approaches further improved feature encoding and scene representation: MVSNeRF [68] incorporated 3D features via cost volumes constructed from multi-view stereo (MVSNet [69]), while IBRNet [70] leveraged self-attention across points along a ray to enhance point density predictions. Transformer-based models, including GNT [71], GeoNeRF [72], and GPNR [73], utilized attention mechanisms to aggregate pixel- and patch-level information, improving the model's ability to capture complex scene geometry and appearance. Other notable works include InsertNeRF [74], which employs hypernet modules for efficient parameter adaptation to novel scenes, and Mip-NeRF 360 [75], which extends NeRF to unbounded, outdoor environments using mipmap-based sampling and improved positional encoding. CaesarNeRF [5] further advances the field by enhancing feature extraction and cost volume construction, resulting in more accurate and efficient scene representations for novel view synthesis. Extensions to unbounded and in-the-wild scenes have also been proposed to address the limitations of the original NeRF in handling real-world variability and large-scale environments.

Despite these advancements, several open challenges remain in the development of generalizable NeRF models. These include improving the efficiency and scalability of feature extraction, enhancing the robustness of models to varying scene content and lighting conditions, and reducing the computational cost associated with rendering high-quality novel views. Additionally, achieving strong generalization to truly unseen scenes, especially in the presence of limited input views or significant domain shifts, remains an active area of research. Addressing these challenges is critical for enabling practical deployment of NeRF-based systems in real-world applications such as robotics, augmented reality, and autonomous navigation.

2.4 Small Object Detection

Small object detection is a critical yet challenging task in computer vision, with applications spanning autonomous driving, surveillance, medical imaging, remote sensing, and robotics. The primary difficulty arises from the limited pixel repre-

sentation of small objects, which often leads to insufficient feature extraction, poor localization accuracy, and increased susceptibility to noise. Small objects are also more prone to occlusion, low contrast, and background clutter, making their detection significantly more complex than that of larger objects. These challenges necessitate specialized approaches and architectural innovations to improve detector performance on small-scale targets.

The advent of deep learning, particularly Convolutional Neural Networks (CNNs), revolutionized the field by enabling end-to-end learning of hierarchical features. Modern detectors such as Faster R-CNN [76] and YOLO [77, 78, 79, 80] have achieved remarkable success on standard benchmarks, leveraging region proposal networks, anchor boxes, and unified detection pipelines. More recently, transformer-based models like DETR [81] and its variants have introduced attention mechanisms to capture global context and long-range dependencies, further enhancing detection capabilities. However, these general-purpose detectors often struggle with small objects due to inherent design limitations, such as coarse feature maps, large anchor sizes, and insufficient resolution in deeper layers.

To address the unique challenges of small object detection, several strategies have been proposed. Multi-scale feature representation techniques, such as Feature Pyramid Networks (FPN) [82] and PANet, construct hierarchical feature maps at multiple scales, enabling detectors to better capture objects of varying sizes by aggregating low-level spatial details with high-level semantic information. FocusDet [83] improves the retention of small objects by removing downsampling operations, which can cause loss of fine details. Incorporating contextual information from surrounding regions helps disambiguate small objects from background clutter, with attention modules and context-aware pooling improving both localization and classification. Slicing the image into smaller patches has also been explored to improve detection by focusing on local context; SAHI [84] demonstrated significant improvements using this approach. Small object detection is also a common challenge in aerial imagery, such as satellite and drone footage. Works such as [85, 86, 87, 88] have proposed specialized architectures and training strategies to enhance detection in these domains, which often contain a high density of small targets against complex backgrounds. In the context of bird detection, recent challenges [6, 89] have spurred the development of new methods, with various works proposing strategies to improve detection of small birds in images and videos [90, 91, 87, 88].

Speed and efficiency are critical for deploying object detectors in resource-constrained environments such as mobile devices, drones, and embedded systems. Lightweight architectures and model compression techniques are actively researched

to balance detection accuracy with computational demands. Notable examples include NanoDet [7], an anchor-free, lightweight detector designed for real-time applications on edge devices, and tiny or nano variants of state-of-the-art detectors such as YOLOv5n [78] and YOLOv8n [80], which are optimized for deployment on devices with limited resources while maintaining competitive performance. Model compression and quantization techniques, such as pruning, quantization, and knowledge distillation, are widely used to reduce model size and inference latency without significant loss in accuracy, making small object detection feasible on edge hardware.

Several open challenges remain in the field of small object detection. Achieving an optimal balance between detection accuracy and computational efficiency is still a significant issue, especially for real-time and resource-constrained applications. The scarcity of annotated datasets specifically focused on small objects limits effective training and evaluation. Furthermore, small object detection is highly sensitive to domain shifts, scale variations, and occlusions, highlighting the need for more robust architectures, adaptive training strategies, and improved data augmentation methods.

In this thesis, we explore how NAS can be employed to automatically discover efficient and effective architectures on different vision tasks, with a particular focus on deep stereo matching, generalizable NeRF, and small object detection. By leveraging zero-cost and near-zero-cost performance proxies, we aim to significantly reduce the computational burden of NAS while maintaining strong correlations with true model performance. Our goal is to demonstrate the potential of automated architecture design to address the unique challenges of these domains, ultimately advancing the research and practical applications in tasks where traditional NAS methods have been less explored.

Chapter 3

Zero-Cost Neural Architecture Search for Deep Stereo Matching

This chapter presents a detailed investigation into the application of Zero-Cost Neural Architecture Search (NAS) for deep stereo matching. Building on the foundations and challenges outlined in previous chapters, we focus on how zero-cost proxies can be leveraged to efficiently search for neural network architectures tailored to the unique demands of stereo vision tasks. More specifically, we explore the integration of zero-cost NAS techniques with state-of-the-art stereo matching frameworks, such as RAFT-Stereo, to identify architectures that excel in disparity estimation while minimizing computational overhead. Our goal was to leverage the potential of this approach to discover an architecture variant that was more compact and efficient, while still maintaining competitive performance with the original RAFT-Stereo model.

3.1 Introduction

Stereo matching is a fundamental problem in computer vision, with applications ranging from autonomous driving to 3D reconstruction. The task involves estimating depth information from a pair of stereo images by finding correspondences between pixels in the left and right images. Deep learning-based approaches have significantly advanced the state-of-the-art in stereo matching, with models such as RAFT-Stereo demonstrating impressive performance through iterative refinement of disparity estimates using all-pairs cost volumes. However, designing effective neural network architectures for stereo matching remains a challenging and time-consuming process, often requiring extensive experimentation and domain expertise. The latter delivered impressive results, achieving state-of-the-art performance on several benchmarks.

Even though RAFT-Stereo was already more efficient than previous methods, it still had a relatively high computational cost and memory footprint, which could limit its applicability in real-time or resource-constrained scenarios. For this reason we aimed to explore how zero-cost NAS could be employed to discover a more efficient architecture variant that retained competitive performance.

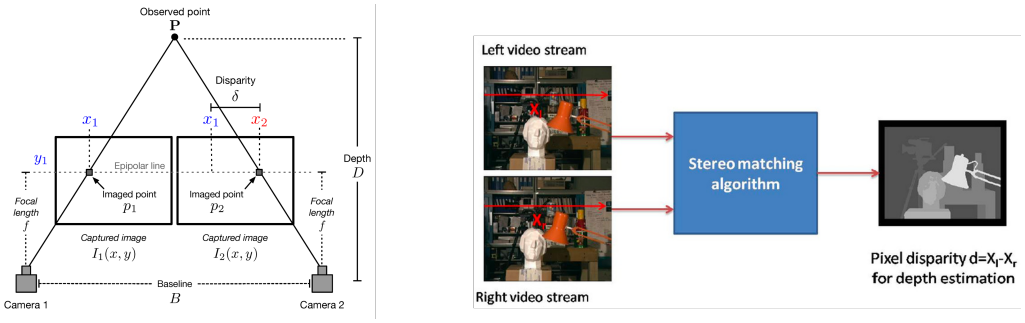


Figure 3.1: Stereo Matching. Illustration of the stereo matching process, highlighting the epipolar geometry involved in estimating depth from stereo images. Classic stereo matching algorithms [4] were hand-designed, exploiting the geometric relationship between views and exctracing features to compute disparity maps. Deep learning-based methods have since revolutionized the field by learning feature representations and matching functions directly from data.

3.2 Methodology

3.2.1 Search Space Design

The first step in building any NAS framework is to define a suitable search space that captures the architectural variations relevant to the target task. For our use case, since we aim to search for a RAFT-Stereo variant architecture, we designed a search space that encompass modifications to a key component of the architecture: the feature extractor. The feature extractor is responsible for generating multi-scale feature maps from the input stereo images, which are then used to compute the cost volume and refine disparity estimates. specifically, RAFT-Stereo is composed by a shared feature extractor, that is responsible to extract features from both the left and right images, and a context network, that is used to extract features only from the left image. Both these components are based on a ResNet-like architecture, with several residual blocks and downsampling layers. We defined a search space that allowed for variations in composition of layers, the number of channels, the kernel sizes, and the presence of skip connections within these components.

Each layer of the encoders were made searchable, replacing the layer by searchable

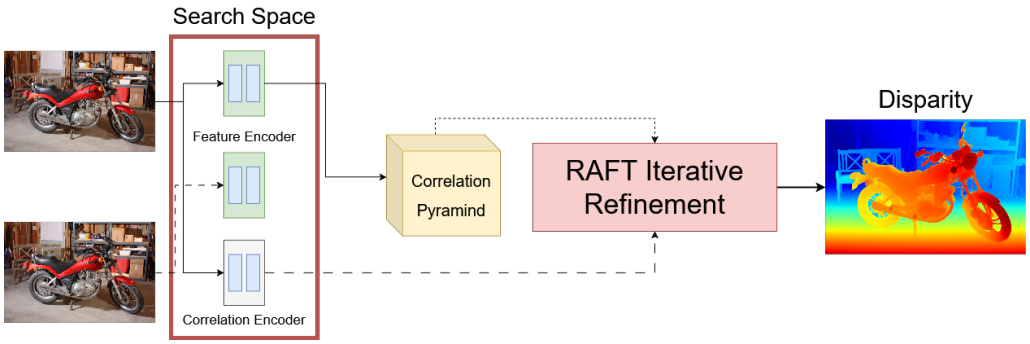


Figure 3.2: RAFT-Stereo Architecture and Search Space. Overview of the RAFT-Stereo architecture, highlighting the feature extractor and context network components that were targeted in our NAS search space.

cells, that could be selected from two different options, visible in Figure 3.3:

- A searchable ResBlock layer, which consist in the original RAFT-Stereo ResBlock layer but modified to be searchable in its hyperparameter values.
- A searchable GhostModule, which is a lightweight alternative to the standard convolutional layer, designed to reduce the number of parameters and computations while maintaining performance. It derived from the GhostNet architecture [92, 93], which introduces the concept of "ghost" features, redundant feature maps that can be generated using inexpensive operations. The GhostModule consists of two main components: a primary convolutional layer that generates a set of intrinsic feature maps, and a series of cheap linear operations (such as depthwise convolutions) that generate additional ghost feature maps from the intrinsic ones. The final output is obtained by concatenating the intrinsic and ghost feature maps, resulting in a feature representation that is both rich and computationally efficient.

3.2.2 Search Strategy and Performance Estimation

To efficiently explore the defined search space, in literature several search strategies have been proposed, as stated in previous chapters. We opted for regularized evolution [14], a population-based search strategy that has demonstrated strong performance in various NAS tasks. Regularized evolution maintains a population of candidate architectures, iteratively selecting the best-performing models as the parents to generate a new population of candidates through mutation and crossover operations. This approach balances exploration and exploitation, allowing for efficient navigation of large and complex search spaces.

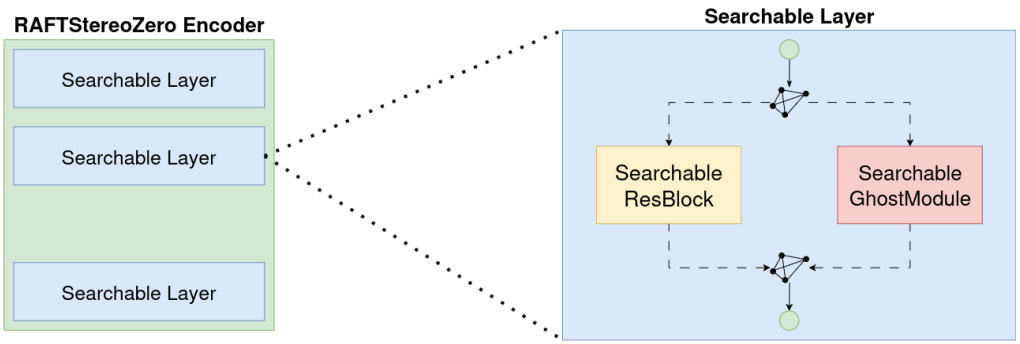


Figure 3.3: RAFTStereo-Zero Searchable Cells. The two types of searchable cells used in our NAS search space: (a) a searchable ResBlock layer, and (b) a residual block with customizable kernel sizes, strides, and output channels.

The selection of the top candidate happens through a performance estimation strategy. In our case, we opted for a zero-cost proxy, specifically the AZ-Score [94], which computes a set of expressivity scores based on a single forward pass through the untrained network. The AZ-Score combines multiple metrics that capture different aspects of architectural quality, including feature diversity, activation dynamics, and information propagation. By evaluating these scores, we can quickly estimate the potential performance of candidate architectures without the need for extensive training, significantly reducing the computational cost of the search process. AZ-Score has been shown to correlate well with test set accuracy across various tasks, making it a suitable choice for guiding the search in our stereo matching application. The score is composed by four different components:

Expressivity($\mathcal{E}$). It evaluates how features are distributed across orientations in the feature space, given randomly initialized network weights and Gaussian random inputs. Given the score of the l-th primary block, $s_l^{\mathcal{E}}$, as an entropy score, posing L1-normalized coefficients of PCs as probabilities:

$$s_l^{\mathcal{E}} = \sum_{i=1}^{c} -\hat{\lambda}_l(i) log \hat{\lambda}_l(i) \tag{3.1}$$

where $\hat{\lambda}_l(i)$ is a set of L1-normalized coefficient. The overall network score consists of the sum of all block-level scores:

$$s^{\mathcal{E}} = \sum_{l=1}^{L} s_l^{\mathcal{E}} \tag{3.2}$$

Progressivity($\mathcal{P}$). The difference of the block-level $\mathcal{E}$ scores from Eq.(3.1):

$$s^{\mathcal{P}} = \min_{lin2,\ldots,L} s_l^{\mathcal{E}} - s_{l-1}^{\mathcal{E}} \tag{3.3}$$

Trainability($\mathcal{T}$). Given the approximation of the Jacobian matrix A_l:

$$A_l^\top = \frac{1}{n}\sum_{p=1}^{n} g_{l-1}(p)g_l^\top(p), \tag{3.4}$$

The trainability score $s^{\mathcal{T}}$ is defined as:

$$s^{\mathcal{T}} = \frac{1}{L-1}\sum_{l=2}^{L} -\sigma_l - \frac{1}{\sigma_l} + 2, \tag{3.5}$$

which reaches its maximum value when the spectral norm of A_l, denoted as σ_l, is equal to 1 for all l.

Complexity($\mathcal{C}$). Given the correlation between network dimension and performance, [33] proposes using the FLOPs count as a complexity proxy.

$$s^{\mathcal{C}}(i) = log_{10}(FLOPs(i)) \tag{3.6}$$

We use the log_{10} to reduce the summed score dimension.

But if we just consider the AZ-Score alone, we will probably end up selecting architectures that are big, given that by construction, the more expressive model is usually one with more parameters. To avoid this and to drive the search towards more efficient architectures, we combined the AZ-Score with a dimensionality metric, specifically the number of parameters (#PARAMS) of the model. The final score used for selection was computed as follows:

Given a model i, the final score $s^{AZ}(i)$ is defined as:

$$s^{AZ}(i) = \sum_{\mathcal{M}\in\mathcal{E},\mathcal{T},\mathcal{P},\mathcal{C}} s^{\mathcal{M}}(i) \tag{3.7}$$

In order to account for the architectural dimensions in the scoring process, the combined final score is formulated as:

$$s^{\Omega}(i) = ln(s^{AZ}(i)) - ln(\#PARAMS(i)) \tag{3.8}$$

We compute all proxy scores for the possible candidate architectures during the evolutionary search process. Subsequently, to generate new candidates, we mutate the top-scoring architecture based on the s^{Ω} value. Finally we select the best-performing architecture using the AZ ranking formula:

$$S^{\Omega}(i) = \sum_{\mathcal{M}\in\{\mathcal{E},\mathcal{P},\mathcal{T},\mathcal{C}\}} \log\left(\frac{\text{Rank}\left(s^{\Omega}(i)\right)}{m}\right) \tag{3.9}$$

where m is the number of candidate architecture. The chosen architecture A^{Ω} will be the first ranked candidate.

With the scoring function and search strategy defined, the overall search process integrates these components by employing evolutionary search guided by the AZ-Score. Candidate architectures are evaluated using the AZ-Score, and the final selection is performed through a non-linear aggregation of the scores to identify the optimal architecture, as summarized in Algorithm 1.

Algorithm 1 Evolutionary search using the combined score

Input: search space Z; number of NAS iteration T.
Output: selected architecture A^*.
Randomly initialize first architecture A_1 for evolutionary search
while $i = 1 < T$ **do**
    Compute proxy scores $s^{\mathcal{E}}(i)$, $s^{\mathcal{T}}(i)$, $s^{\mathcal{P}}(i)$ for the architecture A_i.
    Append the architecture A_i to the population history $\mathbb{A}$.
    Compute #PARAMS of A_i.
    Compute the score $s(A_i)$.
    Generate a new candidate architecture A_{i+1} which belongs to the search space Z by mutating the top architecture based on s score.
end while
Compute the ranking over the history set $\mathbb{A}$ and select the best candidate architecture A^*

3.3 Experiments and Results

Upon concluding the search process, which took 0.1 GPU days, we applied the ranking process over the architectures discovered to identify the network with the best overall score. The final architecture configuration, dubbed RAFT-StereoZero and shown in Fig.3.4, counts 1.14M trainable parameters, in contrast to the original 11M, translating into lower memory occupancy. While RAFT-Stereo consists of ResBlocks of a variable number of channels, RAFT-StereoZero modules are mostly GhostModules counting 32 or 64 channels. Given the iterative nature of the prediction process, we further optimize RAFT-StereoZero complexity and speed by conducting a Hyperparameter Optimization (HPO) search to identify the best combination of training and validation iterations. Using the Tree-structured Parzen Estimator (TPE) algorithm [95], each combination was trained over a small subset of the training dataset, i.e., 400 samples for 5000 steps, with the error on Middlebury [96] and KITTI 2015 [97] used as evaluation scores, leading to 8 training and 10 validation iterations as the best trade-off between accuracy and speed.

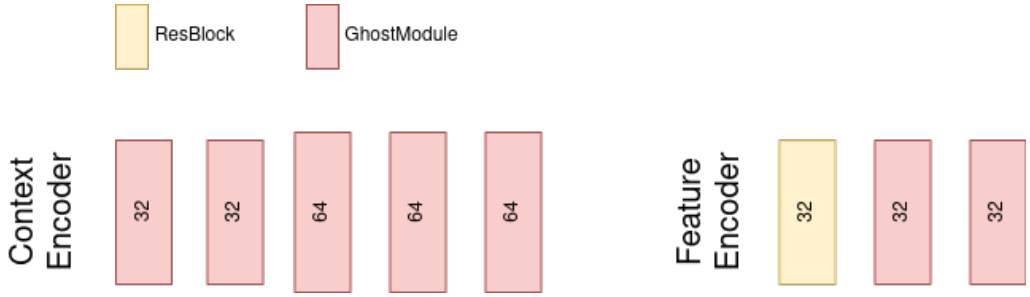


Figure 3.4: RAFT-StereoZero Architecture. The RAFT-StereoZero architecture discovered through our NAS framework, highlighting the use of GhostModules and ResBlocks in the feature extractor and context network.

3.3.1 Training

Once the search process is concluded, we trained the architecture using the same training algorithm as the original RAFT-Stereo, while exploring the use of more recent datasets, such as NERF-Stereo [62] and CREStereo [98], alongside SceneFlow [34], Sintel Stereo [99], and Falling Things [100]. After balancing, our training set counts 760,836 samples during training. We trained RAFT-StereoZero on four A100 GPUs for over 800,000 steps, with a total batch size 32. Given the substantial training data, we select this configuration to expedite the training process. Additionally, we observed that increasing the number of training steps relative to the original architecture's training improves evaluation results significantly. In the following section, we will evaluate the performance of the discovered architecture and compare it with the original model.

3.3.2 Evaluation

In order to assess the performance of RAFT-StereoZero compared to the original RAFT-Stereo, we run the evaluation over the zero-shot benchmark [62] composed of four standard different stereo datasets: Middlebury 2014 [96], ETH3D [101], KITTI 2012[102] and KITTI 2015 [97]. The performance are evaluated using two metrics: EPE (End Point Error) and D1 (frame image mismatch rate). Specifically EPE is used to measure the average disparity error in pixels, while D1 represents the percentage of pixels in the disparity map which are considered outliers (abs error $>$ 3 pixels and $>$ 5% disparity error). The lower this metrics are, more accurate is the model disparity prediction. The results are reported in Table 3.1, where we can see that RAFT-StereoZero achieves competitive performance compared to the original RAFT-Stereo, despite having significantly fewer parameters and lower computational complexity.

Model	Middlebury 2014		ETH3D		KITTI 2012		KITTI 2015	
	EPE	D1	EPE	D1	EPE	D1	EPE	D1
RAFT-Stereo	1.57	11.50	0.27	2.61	0.83	3.86	1.13	5.68
RAFT-Stereo (Setup C)	1.79	11.78	0.88	24.17	0.76	2.88	1.08	4.86
RAFT-StereoZero	1.87	13.06	0.31	4.32	0.78	3.57	1.12	5.69
RAFT-Stereo (RT)	2.53	15.57	0.58	5.76	0.92	4.31	1.15	5.82
RAFT-StereoZero (RT)	2.93	18.59	0.49	9.28	0.89	4.03	1.13	5.62

Table 3.1: Quantitative results. When compared with the original architecture, RAFT-StereoZero achieves competitive results across different datasets.

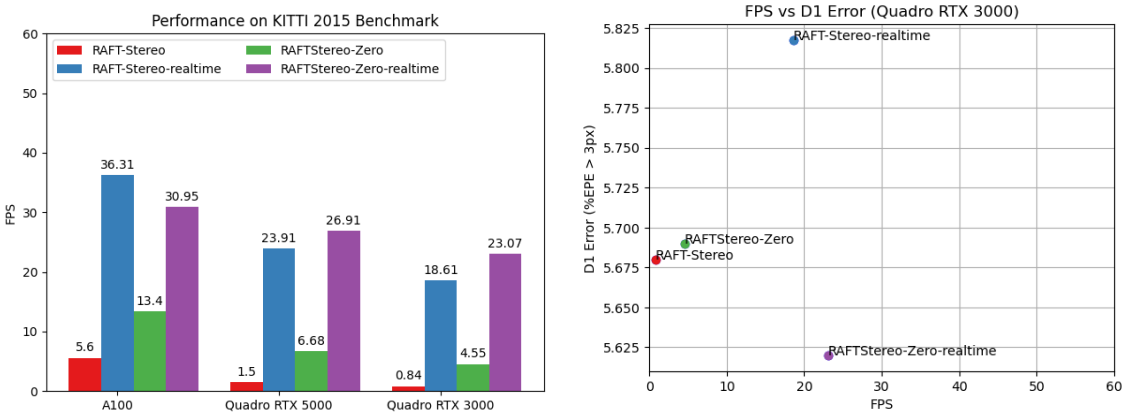


Figure 3.5: Runtime analysis. Left: framerate measured on three different GPUS; right: accuracy-framerate plot on KITTI 2015 using Quadro RTX 3000 GPU.

Impact of training datasets. We study the effect of each additional dataset we introduce for training. Purposely, we train different instances RAFT-StereoZero following the same protocol, yet using three training data setup: first, we use just Sceneflow, Falling Things and Sintel Stereo (Setup A), then add NERF-Stereo (Setup B), and finally, we add CREStereo (Setup C). Table 3.2 reports the outcome of this evaluation. Specifically, we can see an average 8% relative improvement when adding NeRF Stereo, and a further 10% when adding CREStereo. In particular, the accuracy over the ETH3D benchmark benefits the most, with an improvement of 31% EPE and a 50% D1.

Model	Middlebury 2014		ETH3D		KITTI 2012		KITTI 2015	
	EPE	D1	EPE	D1	EPE	D1	EPE	D1
Setup A	3.21	18.87	0.67	16.92	0.90	**4.26**	1.51	**6.34**
Setup B	3.13	**18.19**	0.58	12.29	**0.89**	4.37	1.29	6.72
Setup C	**2.91**	18.89	**0.40**	**6.13**	0.90	4.51	**1.22**	6.44

Table 3.2: Ablation study – impact of the training data. We evaluate the impact of different training data configurations.

Accuracy comparison with RAFT-Stereo. Table 3.1 reports a comparison between the original RAFT-Stereo and our model at the top. For a fair comparison, we also train a further instance of RAFT-Stereo on the same training data used for our model (Setup C): this yields some improvements on KITTI datasets, although being surprisingly ineffective on ETH3D. We can appreciate how RAFT-StereoZero achieves almost equivalent accuracy on any dataset with marginal drops only – lower than 0.1 EPE and about 1% D1 on Middlebury and KITTI datasets against RAFT-Stereo Setup C, about 0.03 EPE and 1.3% D1 on ETH3D against the original RAFT-Stereo.

At the bottom, we report a comparison between RAFT-Stereo (RT) variant and a *realtime* counterpart searched through our approach, referred to as RAFT-StereoZero (RT). Again, our model achieves marginal drops in accuracy on Middlebury and ETH3D, while gaining some accuracy on KITTI datasets.

The small drops exposed by both our models against their counterparts allow for training accuracy with efficiency, as we are going to discuss in the remainder.

Inference Speed. Figure 3.5 reports the accuracy-speed trade-off achieved by the original RAFT-Stereo models and our searched variants. On the left, the runtime required to process KITTI 2015 images when measured on three GPUs: nVidia A100, Quadro RTX 5000, and Quadro RTX 3000. RAFT-StereoZero achieves a speed-up ranging from 2.3× to 5.4× against RAFT-Stereo and an increase of 5FPS when considering (RT) variants. On the right, we report an accuracy-framerate plot concerning performance on the Quadro RTX 3000. It highlights that our models achieve faster and more accurate results on KITTI 2015 than their original counterparts.

3.3.3 Qualitative Results

We conclude by reporting a qualitative comparison between the predictions by RAFT-Stereo and RAFT-StereoZero. Figure 3.6 shows examples from KITTI 2015, Middlebury 2014 and ETH3D datasets. We can appreciate how our model retains the generalization capability of the original RAFT-Stereo, as well as fine-grained details in the predicted disparity maps.

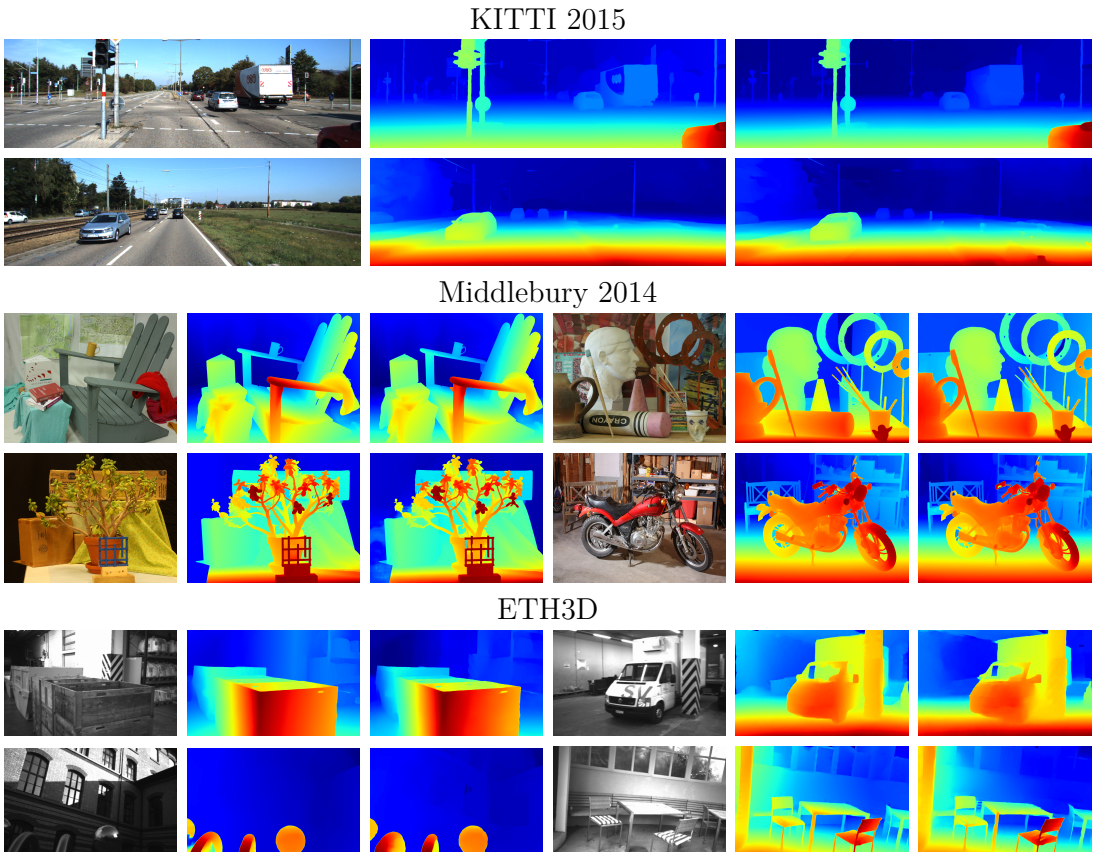


Figure 3.6: Qualitative results. We report examples from KITTI 2015, Middlebury 2014 and ETH3D, showing the reference color image (left) and the disparity maps predicted by RAFT-Stereo (middle) and RAFT-StereoZero (right).

Chapter 4

Zero-Cost Neural Architecture Search for Generalizable NeRF

In this chapter we explore the application of Zero-Cost Neural Architecture Search (NAS) to Generalizable Neural Radiance Fields (G-NeRF). Building on previous foundations and identified challenges, we investigate how zero-cost proxy metrics can be used to efficiently search for neural network architectures suited to generalizable NeRF tasks. By integrating zero-cost NAS techniques with state-of-the-art NeRF models, we aim to identify architectures that deliver high-quality novel view synthesis from sparse inputs while minimizing computational cost. The objective is to automatically discover compact and efficient architectures that retain competitive performance compared to existing methods.

4.1 Introduction

Neural Radiance Fields (NeRF) [65] have transformed novel view synthesis by enabling photorealistic rendering from sparse input views. NeRF aims to render a 3D scene by learning a continuous volumetric representation using a neural network, typically a multi-layer perceptron (MLP). The network tries to predict the color and density of points where hypothetically the rays intersect the scene, allowing for high-quality rendering from arbitrary viewpoints. Given a point $x \in \mathbb{R}^3$ and a viewing direction $d \in \mathbb{S}^2$ in the 3D space sphere, the NeRF model predicts the color c and density σ at that point:

$$(c, \sigma) = F(x, d) \tag{4.1}$$

After predicting all these values, the final pixel color is computed using volume rendering techniques, integrating the contributions of all sampled points along the ray. The original NeRF model demonstrated impressive results on various datasets, including synthetic and real-world scenes, showcasing its ability to capture fine details and complex lighting effects.

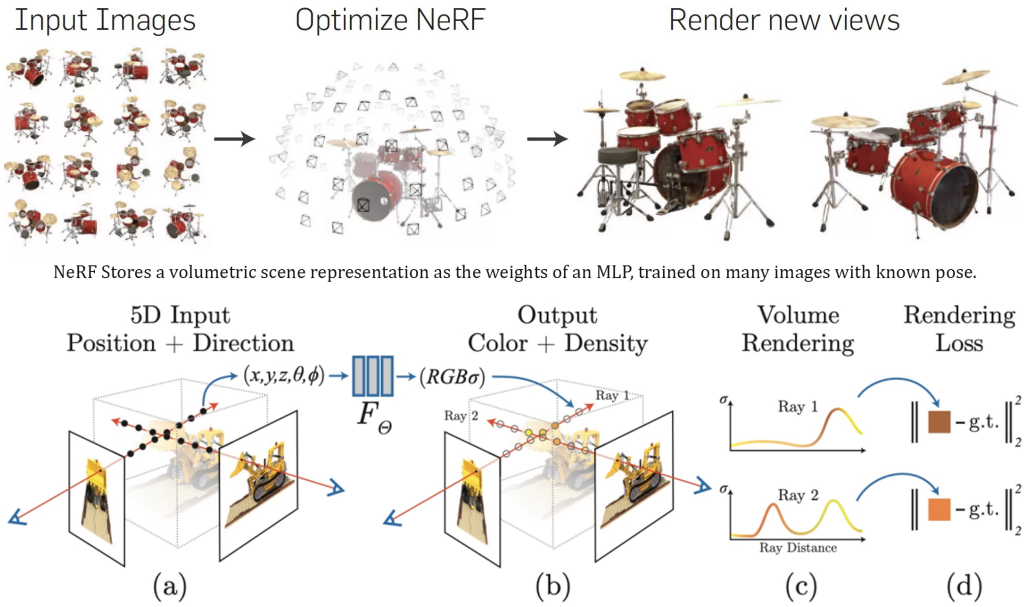


Figure 4.1: Neural Radiance Fields Overview. Overview of the Neural Radiance Fields (NeRF). NeRF represents a 3D scene using a neural network that predicts color and density values for points in space, enabling photorealistic rendering of novel views of the scene through volume rendering techniques.

Despite their success, traditional NeRF models require per-scene optimization, limiting their scalability and practicality for real-world applications. To address this, generalizable NeRF approaches have been developed, aiming to synthesize novel views of unseen scenes without retraining for each of them. To do that, encoder-based generalizable NeRF models use encoders to extract per-pixel feature maps in order to decouple the object representation from the neural rendering process, allowing the model to generalize across different scenes. Given a fused feature embedding $\tilde{F}$ and a ray direction d, the generalizable NeRF model predicts the color c and density σ as:

$$(c, \sigma) = F_G(x, d, \tilde{F}) \tag{4.2}$$

Among these, CaesarNeRF [5] stands out for its state-of-the-art performance, leveraging a CNN backbone alongside a transformer-based architecture to effectively capture both local and global scene information. Given its strong results and design, we selected CaesarNeRF as the baseline architecture for our NAS experiments targeting

generalizable NeRF.

4.2 Methodology

CaesarNeRF [103] employs a CNN backbone to extract multi-scale image features from input views, which are then processed by a transformer-based architecture to model complex scene geometry and appearance. The model utilizes a ray-based rendering approach, where rays are sampled from the input images and passed through the network to predict color and density values for volumetric rendering. This design allows CaesarNeRF to effectively capture both local and global scene information, making it well-suited for synthesizing novel views of unseen scenes from sparse input images. Moreover, it implements a calibration mechanism to align features from different views, enhancing the model's ability to generalize across diverse scenes. an overview of the architecture is shown in Figure 4.2.

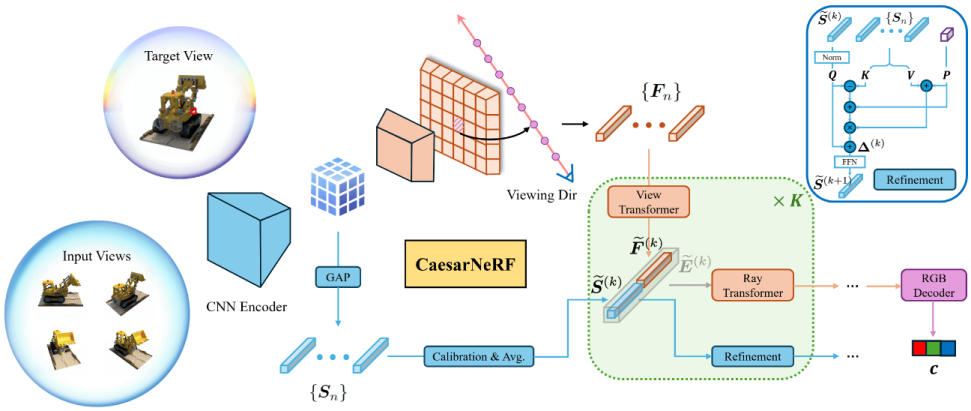


Figure 4.2: CaesarNeRF Architecture. Overview of the CaesarNeRF [5]. CaesarNeRF employs a shared encoder to capture two types of features from input views, including scene-level semantic representation $\{S_n\}$ and pixel-level feature representation $\{F_n\}$. We use the same encoder for both the scene-level semantic representation and the pixel-level embeddings. Following calibration and aggregation of $\{S_n\}$ from various views, we concatenate it with the pixel-level fused feature, processed by the view transformer. Subsequent use of the ray-transformer, coupled with sequential refinement, enables us to render the final RGB values for each pixel in the target view. The output features serve as the input for the next stage, indicated by matching line colors.

The model consists of three main components: a shared CNN encoder, a view transformer, and a ray transformer. The shared encoder extracts multi-scale features from the input images, which are then processed by the view transformer to aggregate information across different views. The ray transformer further refines the aggregated

features along sampled rays to predict color and density values for volumetric rendering. Getting inspiration from [70], the transformers in CaesarNeRF utilize attention mechanisms to effectively model long-range dependencies and complex interactions between different views and rays, enhancing the model's ability to capture intricate scene details and learn to render photorealistic images from novel viewpoints in an end-to-end manner. The CNN encoder plays a crucial role in extracting rich feature representations from the input images, which are essential for the subsequent transformer modules to perform effective aggregation and refinement, in order to boost the overall rendering quality.

For this application, given the constraint of Zero-Cost NAS, we explore the search space composed by the CNN encoder, to identify an architecture that can further improve the feature quality and thus the render quality. To do that, we define a search space that allows variations in the number of layers, the number of channels, kernel sizes, and the inclusion of skip connections within the CNN encoder. By systematically exploring these architectural choices within our NAS framework, we aim to identify configurations that offer superior accuracy for novel view synthesis over unseen scenes.

4.2.1 Search Space Design

To construct the search space for our NAS experiments, we started from the ResNet-34-based CNN encoder architecture originally used in CaesarNeRF. This encoder follows a three-stage design, progressively extracting multi-scale features from the input images. To enable automated architecture search while retaining compatibility with the overall CaesarNeRF pipeline, we preserved the decoder and focused our modifications on the encoder.

The search space was designed to allow flexible exploration of architectural variations within the encoder. Specifically, we made each stage of the encoder configurable in terms of the number of layers, the number of output channels, kernel sizes, and the presence or absence of skip connections. In addition to standard convolutional layers, we introduced alternative building blocks such as depthwise separable convolutions, GhostModules (for lightweight feature extraction), and ConvNeXt layers, reflecting recent advances in efficient CNN design. Each layer in the encoder could thus be instantiated as one of several candidate types, with its own set of tunable hyperparameters.

By structuring the search space in this way, we enabled the NAS process to discover encoder architectures that balance expressivity, computational efficiency, and the ability to generalize across diverse scenes. Figure 4.3 provides an overview of

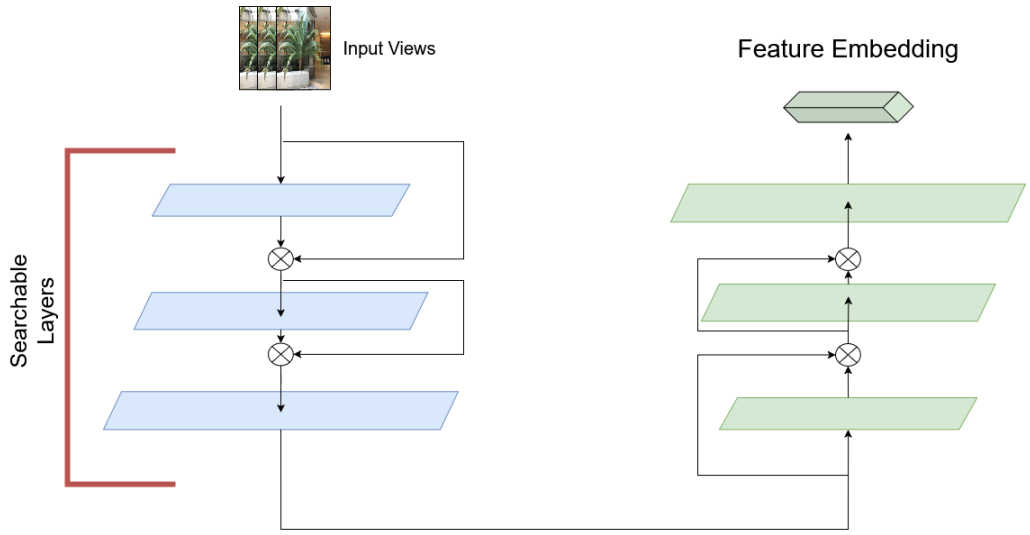


Figure 4.3: G-NeRF Search Space Structure. Overview of the search space structure for the CNN encoder in CaesarNeRF.

the search space structure, highlighting the configurable components and the range of architectural choices available for each encoder stage.

All the element composing a layer are searchable, including the type of layer, the number of channels, the kernel size. The layer can be chosen between standard convolutional layers, depthwise separable convolutional layers, and the previously mentioned GhostModules. Moreover we also added the ConvNeXt layer option, given its recent success in various computer vision tasks. The architecture of the search space is shown in figure 4.3.

4.2.2 Search Strategy and Scoring Function

For the search strategy and scoring function, we adopted the same evolutionary algorithm and zero-cost proxy approach detailed in Chapter 3.2.2. Specifically, we utilized regularized evolution to explore the defined search space, generating candidate architectures and evaluating them using a suite of zero-cost proxies that estimate model expressivity, trainability, and complexity without requiring full training. The evolutionary process iteratively selects top-performing candidates based on their aggregated proxy scores, applies mutations to introduce architectural diversity, and refines the population over successive generations. The final architecture is chosen through a non-linear aggregation of the proxy metrics, ensuring a balance between accuracy and computational efficiency.

During initial experiments, we observed that architectures discovered using this approach performed well when evaluated on single-view inputs but exhibited

limited generalization when multiple views were provided, a critical requirement for generalizable NeRF tasks. We hypothesized that this limitation arose because the proxy scores were computed using only single-view data, thus favoring architectures optimized for that scenario rather than for multi-view consistency.

To address this, we enhanced our search strategy by evaluating each candidate architecture across multiple input configurations. Specifically, for each candidate, we computed the zero-cost proxy scores using single-view, double-view, and triple-view inputs, then averaged the results to obtain a final score that better reflects the architecture's ability to generalize across varying numbers of views. Formally, for a candidate architecture A_i, let $s_n^{AZ}(i)$ denote the proxy score when evaluated with n input views. The aggregated score used for selection is then:

$$s^{AZ}(i) = \frac{1}{3} \sum_{n=1}^{3} s_n^{AZ}(i) \tag{4.3}$$

This modification ensures that the search process favors architectures that are robust and effective across different multi-view scenarios, rather than being over-specialized to single-view performance. As a result, the discovered architectures demonstrated improved generalization and stronger performance in generalizable NeRF tasks, particularly when synthesizing novel views from multiple input images.

4.3 Experimental Results

4.3.1 Training protocol

Using the described strategy, a candidate architecture was discovered after 0.2 GPU days of search. This model is substantially bigger than the original ResNet-34-based encoder, counting 40M parameters against 9M. However, it is important to note that the encoder is only a part of the overall CaesarNeRF architecture, which also includes the view and ray transformers. The latter component remains unchanged, and thus the comparison focuses on the encoder's contribution to the overall performance.

The training of both the original CaesarNeRF and our modified version with the searched encoder was conducted using the same protocol, ensuring a fair comparison. Both models were trained on the same datasets, with identical hyperparameters and training schedules proposed by the authors, which at the same time is the one proposed by [71]. Only the number of ray sampled at each step was reduced from 4096 to 1024 to fit the training within the memory limits of our hardware. To compensate for this, we increased the number of training steps by the same factor, from 500k to 2M. Both models were trained on four A100 GPUs until the completion

of the training schedule.

4.3.2 Evaluation

After training, we assessed both models on their ability to generalize to unseen scenes without any fine-tuning at inference time. Evaluation was performed on the LLFF [65] and mip-NeRF 360 [75] datasets, which offer a variety of scenes and viewpoints to test novel view synthesis from sparse inputs. We report results using Peak Signal-to-Noise Ratio (PSNR), which represents the reconstruction quality, Learned Perceptual Image Patch Similarity (LPIPS) that shows perceptual similarity, and Structural Similarity Index Measure (SSIM) representing structural similarity, providing a comprehensive assessment of image quality and fidelity.

For each dataset, we evaluated the models under different input conditions, specifically using 1, 2, and 3 input views to synthesize novel target views. This protocol allows us to analyze not only the overall rendering quality but also the models' ability to aggregate information from multiple views, a key requirement for generalizable NeRF systems. All metrics were computed on the standard test splits provided by the respective datasets, ensuring comparability with prior work. The results are summarized in the following tables.

	1 View			2 Views			3 Views		
Model	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM
CaesarNeRF (Original)	18.31	0.435	0.521	21.94	0.224	0.736	23.45	0.176	0.794
CaesarNeRF (Retrained)	17.41	0.429	0.485	20.68	0.250	0.690	22.46	0.189	0.769
CaesarNeRF (NAS)	17.67	0.403	0.493	20.28	0.266	0.663	21.84	0.209	0.744

Table 4.1: Results on LLFF dataset. Generalizable novel view synthesis rendering results on **LLFF** dataset (PSNR↑, LPIPS↓, SSIM↑).

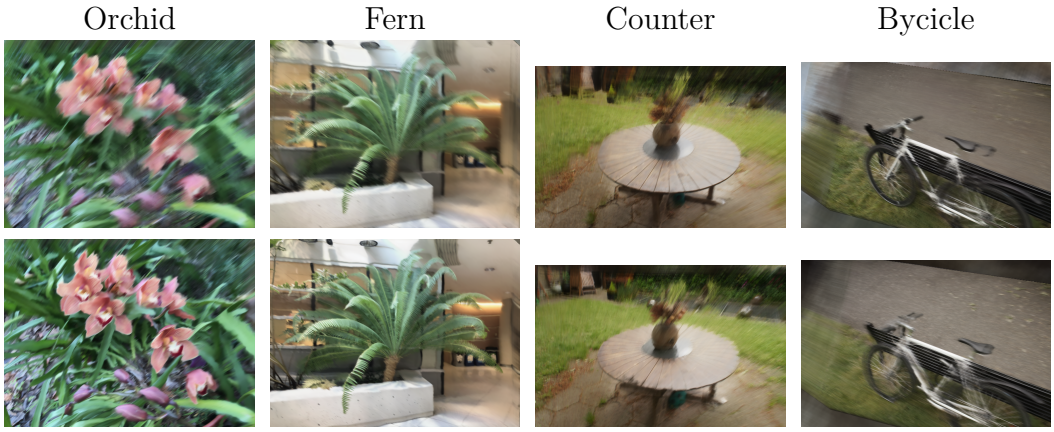
Table 4.1 presents the novel view synthesis results on the LLFF dataset for both the original CaesarNeRF (with a ResNet-34 encoder) and the encoder discovered via NAS. Notably, the retrained CaesarNeRF does not fully match the performance reported in the original paper, likely due to differences in the number of rays sampled during training, which appears to significantly impact results. Nevertheless, the retrained model demonstrates strong performance, particularly as the number of input views increases, indicating effective multi-view aggregation. The NAS-discovered encoder achieves similar results in the single-view setting, with a slight improvement in LPIPS, suggesting better perceptual quality in this scenario. However, with two or three input views, the NAS-based model shows a moderate decrease in PSNR and SSIM compared to the baseline, implying that while the searched encoder is effective

for single-view synthesis, it may not fully leverage additional views. This underscores the importance of evaluating candidate architectures under multi-view conditions during the search. Indeed, initial experiments using only single-view proxy scores led to architectures with weaker multi-view generalization, confirming the need for a multi-view-aware scoring strategy to guide the search toward architectures that generalize across different input configurations.

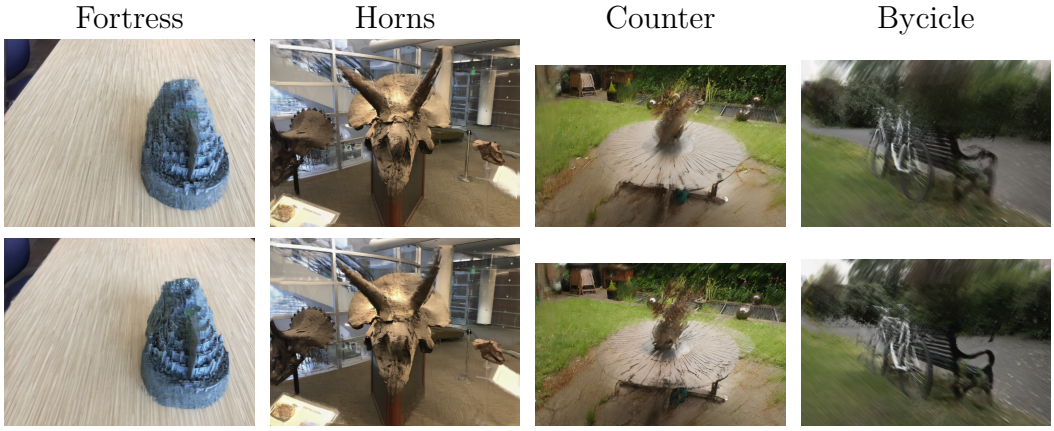
Model	1 View			2 Views			3 Views		
	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM	PSNR	LPIPS	SSIM
CaesarNeRF (Original)	15.20	0.592	0.350	17.05	0.538	0.403	17.55	0.512	0.430
CaesarNeRF (Retrained)	13.64	0.680	0.286	14.77	0.638	0.308	15.26	0.611	0.333
CaesarNeRF (NAS)	13.68	0.665	0.279	14.83	0.622	0.301	15.20	0.596	0.324

Table 4.2: Results on mip-NeRF 360 dataset. Generalizable novel view synthesis rendering results on **mip-NeRF 360** dataset (PSNR↑, LPIPS↓, SSIM↑).

A similar trend is evident in Table 4.2, which presents the evaluation results on the mip-NeRF 360 dataset, a benchmark characterized by greater scene diversity and complexity. In this setting, the retrained CaesarNeRF model continues to demonstrate robust, yet lower than the original, performance, especially as the number of input views increases. In contrast, the encoder architecture discovered via NAS achieves mixed results with respect to the baseline in the different scenarios. Still, even in the more challenging dataset, the performance on multi-view scenarios is consistent with the original architecture. This observation reinforces the importance of designing encoder architectures that are not only expressive for single-view synthesis but also capable of leveraging multi-view information to improve rendering quality. These results suggest that the contribution of the CNN encoder may be relatively limited compared to other components of the architecture. Notably, improvements in feature quality do not necessarily translate into enhanced overall performance. An alternative explanation is that the search space, constrained to variations of the encoder alone, may not be sufficiently expressive to yield architectures that perform significantly better on the target task. Addressing this challenge remains a key direction for future work in generalizable NeRF, as effective multi-view integration is essential for achieving high-fidelity novel view synthesis across diverse and complex scenes.



(a) Single reference view image.



(b) Two reference view images.

Figure 4.4: CaesarNeRF NAS qualitative comparison. Qualitative comparison between CaesarNeRF (Retrained) (top row of each subfigure) and CaesarNeRF (NAS) (bottom row of each subfigure) on LLFF and mip-NeRF 360 datasets. Each column corresponds to a different scene, showcasing the models' ability to synthesize novel views from sparse inputs.

Chapter 5

Zero-Cost Neural Architecture Search for Small Object Detection

This chapter presents a comprehensive study on the application of Zero-Cost Neural Architecture Search (NAS) for small object detection. Building upon the challenges and advancements discussed in previous chapters, we focus on how zero-cost proxies can be effectively utilized to search for neural network architectures that are specifically optimized for detecting small objects in images. We explore the integration of zero-cost NAS techniques with state-of-the-art object detection frameworks to identify architectures that excel in small object detection while minimizing computational overhead. Our objective is to leverage this approach to discover an architecture which is more compact and efficient, while still maintaining competitive performance with existing models.

5.1 Introduction

Small object detection is a challenging problem in computer vision, with applications in areas such as autonomous driving, surveillance, and robotics. Detecting and localizing small objects within images is difficult due to their limited pixel footprint, increased sensitivity to noise, and frequent occlusion or background clutter. Although deep learning-based detectors like Faster R-CNN, SSD, and YOLO have advanced the field, they often underperform on small objects because of architectural constraints such as coarse feature maps and insufficient resolution in deeper layers. As a result, designing neural network architectures that are both accurate and efficient for small object detection typically requires extensive manual tuning and domain expertise.

In this work, we investigate the use of zero-cost Neural Architecture Search (NAS) to automate and accelerate the discovery of efficient architectures for small

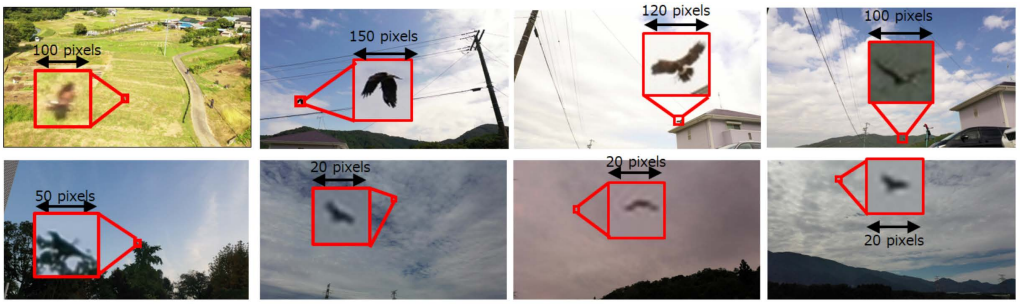


Figure 5.1: Small Object Detection Challenges. Examples of small object detection challenges [6]. In the case of birds, often they occupy only a few pixels in the image, making it difficult for detectors to extract meaningful features.

object detection. Our objective is to identify models that can accurately detect and localize small objects while maintaining low computational overhead, making them suitable for deployment on resource-constrained devices. By leveraging zero-cost NAS techniques, we systematically explore a diverse set of architectural configurations and select candidates that offer the best trade-off between accuracy and efficiency for small object detection.

5.2 Methodology

For our small object detection application, we selected NanoDet [7] as the baseline architecture due to its efficiency and suitability for real-time inference on edge devices. NanoDet is a one-stage, anchor-free detector that integrates lightweight backbone networks, such as MobileNetV2 [24] and ShuffleNetV2 [104], with a feature pyramid network (FPN) to enhance multi-scale feature extraction. The detection head employs a combination of focal loss and IoU loss to address class imbalance and improve localization accuracy. NanoDet's design prioritizes low computational cost and fast inference, making it well-suited for scenarios with limited hardware resources while maintaining strong detection performance.

The structure of NanoDet is illustrated in Figure 5.2. The model comprises three principal components: a backbone network, a feature pyramid network (FPN), and a detection head. For our application, we constructed a search space that enables variations in the composition of layers, the number of channels, kernel sizes, and the inclusion of skip connections within these modules.

The backbone is a critical element, as it determines the quality and granularity of features extracted from the input image. Given the dual requirements of lightweight design and enhanced sensitivity to small objects, we explored a hybrid backbone architecture combining GhostModules [92], which generate efficient feature representations

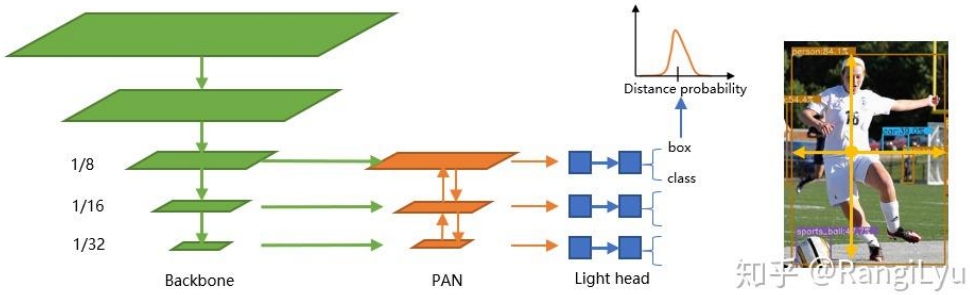


Figure 5.2: NanoDet Architecture. Overview of the NanoDet [7] architecture, highlighting the backbone, feature pyramid network (FPN), and detection head components.

with reduced computational cost, and classic depthwise convolutional layers. This combination aims to maximize representational efficiency while minimizing parameter count. Additionally, we varied kernel sizes throughout the backbone, hypothesizing that larger kernels could improve contextual awareness and help the network better capture small objects that might otherwise be lost in downsampled representations. The search space also includes different backbone stage configurations, allowing us to investigate the optimal number and arrangement of downsampling steps to preserve small object details.

The FPN is responsible for aggregating multi-scale features from the backbone, which is essential for robust small object detection. In our search space, we allowed for variations in the number of FPN layers, the number of channels per layer, and kernel sizes, enabling the discovery of architectures that best balance feature richness and computational efficiency. Importantly, the way the FPN combines features from different scales has a significant impact on the retention of small object information. To address this, we drew inspiration from FocusDet [83], which proposes removing the downsampling operations in the FPN and instead emphasizes upsampling and feature fusion. This approach is designed to maintain high-resolution representations throughout the network, thereby improving the detection of small objects that are easily lost in conventional FPN designs.

By systematically exploring these architectural choices within our NAS framework, we aim to identify configurations that offer superior accuracy for small object detection while remaining efficient enough for real-time deployment on edge devices.

5.2.1 Search Space Design

In order to define the search space, we leveraged the existing NanoDet architecture and modified it to make it searchable, beside introducing the modifications previously mentioned. An overview of the search space structure can be seen in figure 5.3.

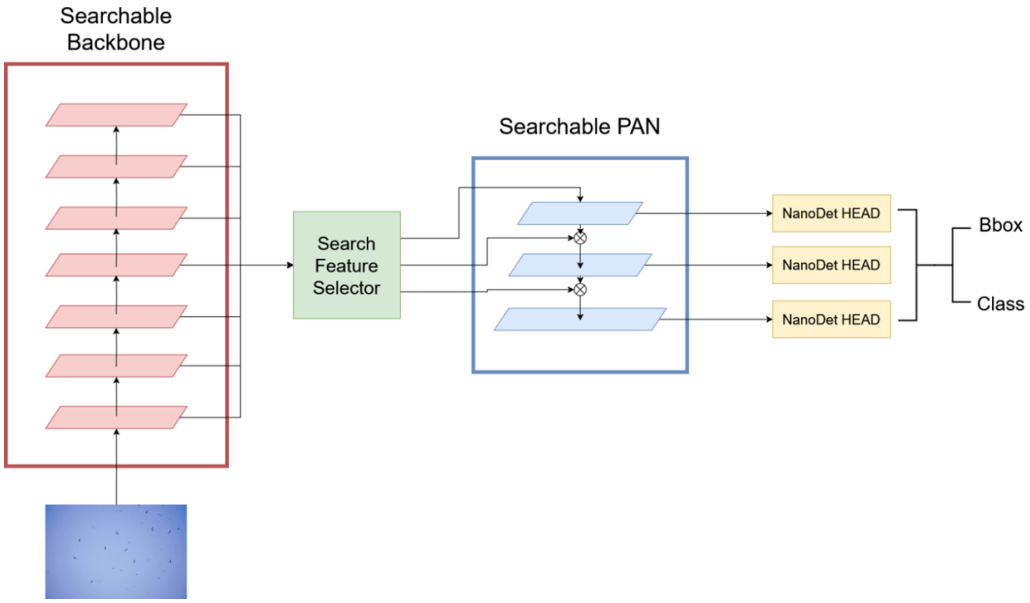


Figure 5.3: Search Space Design. Overview of the search space design for the NanoDet architecture, highlighting the searchable components in the backbone and FPN.

This search space was designed to allow for a wide range of architectural variations while maintaining the core principles of efficiency and effectiveness inherent to the NanoDet framework. By enabling modifications to key components such as the backbone and FPN, we aimed to discover architectures that are particularly well-suited for small object detection tasks. Given the confidentiality of the specific structure of the architecture, further details on the exact configurations and parameters within the search space cannot be disclosed.

5.2.2 Search Strategy and Performance Estimation

For the search strategy in our small object detection NAS framework, we adopted regularized evolution [14], a population-based algorithm that has proven effective in a variety of NAS applications. Regularized evolution maintains a pool of candidate architectures, iteratively selecting the highest-performing models as parents to generate new candidates via mutation and crossover. This evolutionary approach strikes a balance between exploration of the search space and exploitation of promising

regions, enabling efficient discovery of optimal architectures even in large and complex search spaces.

To rank and select candidate architectures, we employed a zero-cost performance estimation strategy. Specifically, we utilized the AZ-Score [94], which computes a suite of expressivity metrics from a single forward pass through the untrained network. The AZ-Score aggregates multiple indicators, such as feature diversity, activation dynamics, and information propagation, to provide a fast and informative proxy for model quality. This approach allows us to evaluate thousands of candidate architectures in seconds, dramatically reducing the computational burden compared to traditional training-based evaluation. As detailed in Chapter 3.2.2, the AZ-Score has demonstrated strong correlation with downstream accuracy across a range of tasks, making it a robust choice for guiding the search process in small object detection. Following the approach in 3.2.2, our goal was to identify an architecture that is both efficient and fast. To achieve this, we combined the AZ-Score with a latency penalty, estimated via a forward pass, so that the search favors models that balance accuracy with computational efficiency.

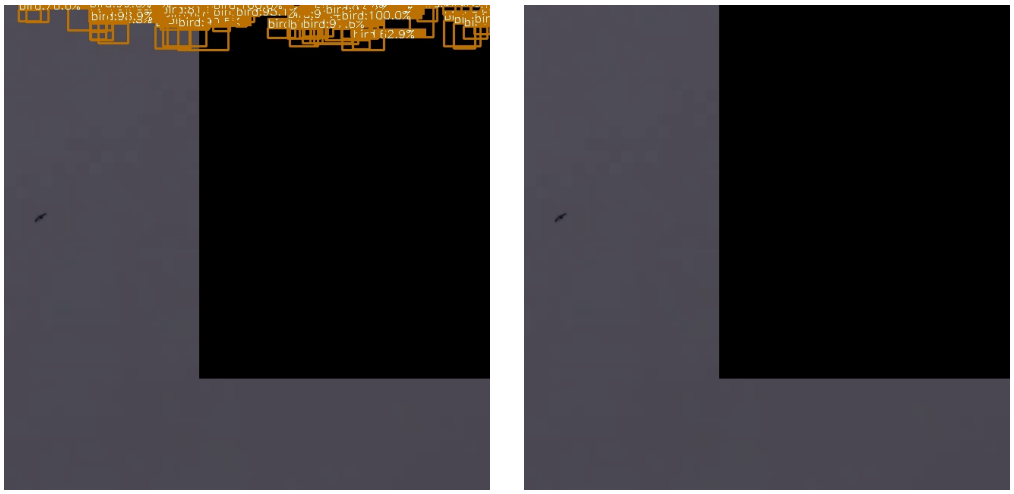
In addition to zero-cost proxies, we briefly explored early training-based performance estimation. In this alternative approach, each candidate architecture is trained for a few epochs on a small batch of data to assess its ability to quickly overfit. The hypothesis is that architectures which achieve rapid overfitting may possess greater representational capacity and thus higher potential accuracy. However, due to resource and time constraints, this method was not fully pursued in our experiments.

5.3 Experiments and Results

5.3.1 Training and Evaluation

To validate the effectiveness of the architectures discovered by our NAS framework, we trained the top-ranked models on a proprietary dataset consisting of thousands of annotated images featuring birds. In this dataset, the birds typically occupy less than 5% of the image area, with bounding boxes of as small as 10x10 pixels, presenting a challenging scenario for small object detection. While specific details about the dataset composition and size cannot be disclosed due to confidentiality, we confirm that it is sufficiently large and diverse to support robust training and evaluation. To further benchmark our approach in an open setting, we also leveraged an open dataset from a recent challenge focused on spotting small birds in the wild [6, 89], providing a public reference for performance comparison and reproducibility. For training, we followed a protocol similar to that used in actual model used by

the company, in order to compare the two model over a similar training setup. We trained each selected architecture for a sufficient number of epochs to ensure convergence, using standard data augmentation techniques to enhance generalization. Experiments showed that training for 300 epochs, which is the same as the original NanoDet, did not provide the best results, given the low number of classes and the characteristics of the object to be detected. Instead, we found that training for less than 100 epochs yielded better performance, likely due to the reduced risk of overfitting on our specific dataset.



(a) **(b)**

Figure 5.4: Training Dynamics. (a) Example of overfitting during training when using 300 epochs. (b) Improved false positives when training for fewer than 100 epochs.

As we can see in figure 5.4a, training for 300 epochs leads to overfitting, resulting in a significant gap between training and validation performance. The number of false positives greatly increases, indicating that the model is not generalizing well. Conversely, as shown in figure 5.4b, training for fewer than 100 epochs helps mitigate overfitting, allowing the model to generalize better to unseen data and achieve improved validation performance.

The architecture discovered through our NAS framework was evaluated against the current production model used by the company, which is based on RT-DETR [6]. RT-DETR is a state-of-the-art object detection model known for its efficiency and accuracy, making it a strong baseline for comparison. Specifically, the RT-DETR small variant is used, which is optimized for real-time performance on edge devices while maintaining competitive detection accuracy. The comparison was conducted on both the proprietary bird dataset and the open dataset from the small bird spotting challenge [6, 89], providing a comprehensive evaluation of the models' performance

in detecting small birds.

5.3.2 Qualitative Results

The architecture discovered via NAS was markedly more compact than the RT-DETR small baseline, containing approximately 1.7 million parameters versus RT-DETR's 20 million. This substantial reduction in model size offers clear advantages for deployment on resource-constrained devices, enabling faster inference and lower memory usage. Empirical tests demonstrated a 2x speedup in inference time, with potential for further optimization through hardware-specific tuning or quantization. In Figure 5.5, we present qualitative results showcasing successful detection scenarios by the NAS-discovered model. The model effectively identifies small birds in various poses and backgrounds.

However, this efficiency comes at a cost: in field evaluations conducted by the company using their standard test protocol, the NAS-discovered model exhibited an average decrease in detection accuracy of around 50% compared to RT-DETR small. This pronounced performance gap underscores the inherent trade-off between model compactness and detection capability, particularly for small object detection tasks where fine-grained features and high representational capacity are essential. The reduced expressive power of the compact model made it more susceptible to false negatives, especially in challenging scenarios where small objects blend into complex backgrounds or are visually similar to non-target objects (see Figure 5.6). These findings highlight the importance of carefully balancing efficiency and accuracy in NAS-driven model design, and suggest that further refinement of the search space or proxy metrics may be necessary to achieve competitive performance without sacrificing speed and deployability.

Figure 5.5: Qualitative results. Qualitative results for the NAS-discovered small bird detection model in successful detection scenarios.

Figure 5.6: Failure cases. Highlights of typical failure cases, including missed birds and false positives. The model occasionally confuses birds with similar-looking objects (a drone and a street lamp head) or fails to detect birds in flight or on the ground.

Chapter 6

Conclusions

This thesis has investigated the application of Zero-Cost Neural Architecture Search (NAS) across several computer vision domains, including deep stereo matching, generalizable NeRF, and small object detection. By leveraging zero-cost proxies, we addressed the significant computational demands of traditional NAS, enabling rapid evaluation and selection of candidate architectures without the need for full training cycles. This methodology facilitated the systematic exploration of diverse architectural spaces, allowing for the identification of models that offer a favorable balance between accuracy and efficiency.

Experimental results demonstrated that zero-cost NAS can successfully discover architectures that are competitive with manually engineered models, particularly when the search space and proxy metrics are well aligned with the task requirements. However, our findings also underscore several limitations and challenges inherent to this approach. Zero-cost proxies, while efficient, may not always capture the nuanced performance characteristics required for specialized tasks, and their predictive power can vary depending on the domain and architecture complexity. For example, in the context of generalizable NeRF, we observed that architectures optimized using single-view proxy scores did not generalize effectively to multi-view scenarios, highlighting the need for proxy evaluation strategies that reflect the intended deployment conditions.

Furthermore, zero-cost proxies are typically data-agnostic, relying on intrinsic architectural properties rather than dataset-specific information. This can limit their ability to account for domain shifts or dataset-driven variations in model performance. The design of the search space remains a critical factor: overly broad or restrictive spaces can hinder the discovery of optimal architectures, and careful tailoring to the target application is essential.

Modern zero-cost proxies such as AZ-Score have shown promising correlation

with downstream performance in certain settings, but their generalizability across tasks is not guaranteed. As such, zero-cost NAS should be viewed as a powerful tool for accelerating architecture search, but not as a universal solution. Continued research into more robust, task-aware proxies and adaptive search strategies will be necessary to fully realize the potential of automated architecture design in complex and evolving computer vision applications.

6.1 Future Works

Looking ahead, there are several promising directions for future research in this area. Most notably, the lack of a flexible and open-source framework on which develop and test NAS algorithms has been a significant bottleneck. Developing such a framework would facilitate broader experimentation and collaboration within the research community, accelerating advancements in NAS methodologies.

Additionally, further development of zero-cost proxies that are not limited to specific tasks or architectural characteristics would enhance the versatility and applicability of these methodologies across a wider range of problems, opening their usage on more complex architectures. This could involve exploring new metrics that capture different aspects of model performance or integrating multiple proxies to provide a more comprehensive evaluation.

Moreover, given the increasing popularity of Generative AI models, investigating their application in NAS could yield novel insights and techniques. For example, generative models could be used to propose new architectural configurations or to augment training data for performance estimation, potentially improving the efficiency and effectiveness of the search process.

Overall, while this thesis has made significant strides in applying Zero-Cost NAS to various computer vision tasks, there remains substantial opportunity for further exploration and innovation in this rapidly evolving field.

Bibliography

[1] F. Hutter, L. Kotthoff, and J. Vanschoren, eds., *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning, Springer, 2019.

[2] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," Apr. 2018.

[3] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search." https://arxiv.org/abs/1806.09055, June 2018.

[4] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, pp. 7–42, 2002.

[5] H. Zhu, T. Ding, T. Chen, I. Zharkov, R. Nevatia, and L. Liang, "CaesarNeRF: Calibrated semantic representation for few-shot generalizable neural rendering," *Conference/Arxiv (Self-Reference)*, p. 1, 2024.

[6] Y. Kondo, N. Ukita, T. Yamaguchi, H.-Y. Hou, M.-Y. Shen, C.-C. Hsu, E.-M. Huang, Y.-C. Huang, Y.-C. Xia, C.-Y. Wang, C.-Y. Lee, D. Huo, M. A. Kastner, T. Liu, Y. Kawanishi, T. Hirayama, T. Komamizu, I. Ide, Y. Shinya, X. Liu, G. Liang, and S. Yasui, "MVA2023 Small Object Detection Challenge for Spotting Birds: Dataset, Methods, and Results," in *2023 18th International Conference on Machine Vision and Applications (MVA)*, pp. 1–11, July 2023.

[7] RangiLyu, "RangiLyu/nanodet," Feb. 2025.

[8] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," 2019.

[9] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware," Feb. 2019.

[10] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search," May 2019.

[11] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," July 2020.

[12] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation," Apr. 2019.

[13] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," Feb. 2017.

[14] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, vol. 33 of *AAAI'19/IAAI'19/EAAI'19*, (Honolulu, Hawaii, USA), pp. 4780–4789, AAAI Press, Jan. 2019.

[15] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural Architecture Search without Training," June 2020.

[16] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive Neural Architecture Search," July 2018.

[17] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient Neural Architecture Search via Parameters Sharing," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 4095–4104, PMLR, July 2018.

[18] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-Aware Neural Architecture Search for Mobile," May 2019.

[19] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," Apr. 2019.

[20] C. White, W. Neiswanger, and Y. Savani, "BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search," *arXiv: Learning*, Oct. 2019.

[21] T. Saikia, Y. Marrakchi, A. Zela, F. Hutter, and T. Brox, "AutoDispNet: Improving Disparity Estimation With AutoML," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (Seoul, Korea (South)), pp. 1812–1823, IEEE, Oct. 2019.

[22] X. Cheng, Y. Zhong, M. Harandi, Y. Dai, X. Chang, T. Drummond, H. Li, and Z. Ge, "Hierarchical Neural Architecture Search for Deep Stereo Matching," Oct. 2020.

[23] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sept. 2020.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Mar. 2019.

[25] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks," June 2018.

[26] Z. Zhang and Z. Jia, "GradSign: Model Performance Inference with Theoretical Insights," June 2022.

[27] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 6377–6389, Curran Associates, Inc., 2020.

[28] L. Theis, I. Korshunova, A. Tejani, and F. Huszár, "Faster gaze prediction with dense networks and Fisher pruning," July 2018.

[29] V. Lopes, S. Alirezazadeh, and L. A. Alexandre, "EPE-NAS: Efficient Performance Estimation Without Training for Neural Architecture Search," in *Artificial Neural Networks and Machine Learning – ICANN 2021* (I. Farkaš, P. Masulli, S. Otte, and S. Wermter, eds.), (Cham), pp. 552–563, Springer International Publishing, 2021.

[30] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-NAS: A Zero-Shot NAS for High-Performance Deep Image Recognition," Feb. 2021.

[31] G. Li, Y. Yang, K. Bhardwaj, and R. Marculescu, "ZiCo: Zero-shot NAS via Inverse Coefficient of Variation on Gradients," Jan. 2023.

[32] C. White, M. Khodak, R. Tu, S. Shah, S. Bubeck, and D. Dey, "A Deeper Look at Zero-Cost Proxies for Lightweight NAS," in *ICLR Blog Track*, 2022.

[33] J. Lee and B. Ham, "AZ-NAS: Assembling Zero-Cost Proxies for Network Architecture Search," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Seattle, WA, USA), pp. 5893–5903, IEEE, June 2024.

[34] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, June 2016.

[35] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6044–6053, 2019.

[36] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 66–75, 2017.

[37] C. Zhang, K. Tian, B. Ni, G. Meng, B. Fan, Z. Zhang, and C. Pan, "Stereo depth estimation with echoes," in *European Conference on Computer Vision*, pp. 496–513, Springer, 2022.

[38] M. Poggi, D. Pallotti, F. Tosi, and S. Mattoccia, "Guided stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 979–988, 2019.

[39] L. Bartolomei, M. Poggi, F. Tosi, A. Conti, and S. Mattoccia, "Active stereo without pattern projector," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18470–18482, 2023.

[40] A. Tonioni, O. Rahnama, T. Joy, L. D. Stefano, T. Ajanthan, and P. H. Torr, "Learning to adapt for stereo," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9661–9670, 2019.

[41] M. Poggi, A. Tonioni, F. Tosi, S. Mattoccia, and L. Di Stefano, "Continual adaptation for deep stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4713–4729, 2021.

[42] M. Poggi and F. Tosi, "Federated online adaptation for deep stereo," in *CVPR*, 2024.

[43] P. Z. Ramirez, A. Costanzino, F. Tosi, M. Poggi, S. Salti, S. Mattoccia, and L. D. Stefano, "Booster: A benchmark for depth from images of specular and transparent surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pp. 85–102, 2024.

[44] A. Costanzino, P. Z. Ramirez, M. Poggi, F. Tosi, S. Mattoccia, and L. Di Stefano, "Learning depth estimation for transparent and mirror surfaces," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9244–9255, 2023.

[45] F. Tosi, Y. Liao, C. Schmitt, and A. Geiger, "SMD-nets: Stereo mixture density networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[46] F. Aleotti, F. Tosi, P. Z. Ramirez, M. Poggi, S. Salti, S. Mattoccia, and L. Di Stefano, "Neural disparity refinement for arbitrary resolution stereo," in *2021 International Conference on 3D Vision (3DV)*, pp. 207–217, IEEE, 2021.

[47] Y. Liu, J. Ren, J. Zhang, J. Liu, and M. Lin, "Visually imbalanced stereo matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2029–2038, 2020.

[48] F. Tosi, P. Zama Ramirez, and M. Poggi, "Diffusion models for monocular depth estimation: Overcoming challenging conditions," in *European Conference on Computer Vision (ECCV)*, 2024.

[49] F. Tosi, L. Bartolomei, and M. Poggi, "A survey on deep stereo matching in the twenties," *arXiv preprint arXiv:2407.07816*, 2024.

[50] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *16th European Conference on Computer Vision, ECCV*, 2020.

[51] L. Lipson, Z. Teed, and J. Deng, "RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching," arXiv, Sept. 2021.

[52] Z. Li, X. Liu, N. Drenkow, A. Ding, F. X. Creighton, R. H. Taylor, and M. Unberath, "Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6197–6206, Oct. 2021.

[53] T. Guan, C. Wang, and Y.-H. Liu, "Neural markov random field for stereo matching," 2024.

[54] J. Zhang, K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson, "Dispsegnet: Leveraging semantics for end-to-end learning of disparity estimation from stereo imagery," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1162–1169, 2019.

[55] F. Zhang, X. Qi, R. Yang, V. Prisacariu, B. Wah, and P. Torr, "Domain-invariant stereo matching networks," in *Europe Conference on Computer Vision (ECCV)*, 2020.

[56] I. Liu, E. Yang, J. Tao, R. Chen, X. Zhang, Q. Ran, Z. Liu, and H. Su, "Activezero: Mixed domain learning for active stereovision with zero annotation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13033–13042, 2022.

[57] C. Cai, M. Poggi, S. Mattoccia, and P. Mordohai, "Matching-space stereo networks for cross-domain generalization," in *2020 International Conference on 3D Vision (3DV)*, pp. 364–373, 2020.

[58] F. Aleotti, M. Poggi, F. Tosi, and S. Mattoccia, "Learning end-to-end scene flow by distilling single tasks knowledge," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 10435–10442, 2020.

[59] F. Tosi, F. Aleotti, P. Z. Ramirez, M. Poggi, S. Salti, S. Mattoccia, and L. Di Stefano, "Neural disparity refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[60] A. Li, A. Hu, W. Xi, W. Yu, and D. Zou, "Stereo-LiDAR depth estimation with deformable propagation and learned disparity-depth conversion," *arXiv preprint arXiv:2404.07545*, 2024.

[61] P. Liu, I. King, M. R. Lyu, and J. Xu, "Flow2Stereo: Effective self-supervised learning of optical flow and stereo matching," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[62] F. Tosi, A. Tonioni, D. De Gregorio, and M. Poggi, "NeRF-Supervised Deep Stereo," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Vancouver, BC, Canada), pp. 855–866, IEEE, June 2023.

[63] F. Aleotti, F. Tosi, L. Zhang, M. Poggi, and S. Mattoccia, "Reversing the cycle: Self-supervised deep stereo through enhanced monocular distillation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 614–632, Springer, 2020.

[64] J. Watson, O. M. Aodha, D. Turmukhambetov, G. J. Brostow, and M. Firman, "Learning stereo from single images," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pp. 722–740, Springer, 2020.

[65] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, p. 43, 2020.

[66] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "Pixelnerf: Neural radiance fields from one or few images," in *CVPR*, p. 76, 2021.

[67] A. Trevithick and B. Yang, "Grf: Learning a general radiance field for 3d representation and rendering," in *ICCV*, p. 59, 2021.

[68] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14104–14113, 2021.

[69] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *ECCV*, p. 75, 2018.

[70] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "IBRNet: Learning Multi-View Image-Based Rendering," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4688–4697, June 2021.

[71] M. V. T, P. Wang, X. Chen, T. Chen, S. Venugopalan, and Z. Wang, "Is Attention All That NeRF Needs?," Mar. 2023.

[72] M. Johari, Y. Lepoittevin, and F. Fleuret, "Geonerf: Generalizing nerf with geometry priors," in *CVPR*, p. 25, 2022.

[73] M. Suhail, C. Esteves, L. Sigal, and A. Makadia, "Generalizable patch-based neural rendering," in *16th European Conference on Computer Vision (ECCV 2022)*, pp. 156–174, 2022.

[74] Y. Bao, T. Ding, J. Huo, W. Li, Y. Li, and Y. Gao, "InsertNeRF: Instilling Generalizability into NeRF with HyperNet Modules," Mar. 2024.

[75] J. Barron, B. Mildenhall, D. Verbin, P. Srinivasan, and P. Hedman, "Mip-nerf 360: Un-bounded anti-aliased neural radiance fields," in *CVPR*, p. 4, 2022.

[76] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Oct. 2014.

[77] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 2016.

[78] R. Khanam and M. Hussain, "What is YOLOv5: A deep look into the internal features of the popular object detector," July 2024.

[79] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018.

[80] R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," in *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pp. 1–6, Apr. 2024.

[81] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," May 2020.

[82] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Honolulu, HI), pp. 936–944, IEEE, July 2017.

[83] Y. Shi, Y. Jia, and X. Zhang, "FocusDet: An efficient object detector for small object," *Scientific Reports*, vol. 14, p. 10697, May 2024.

[84] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection," in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 966–970, Oct. 2022.

[85] R. V. Sri Hari, R. Ambalam, B. Ruban Kumar, M. Ibrahim, and R. Ponnusamy, "Yolo5-Based UAV Surveillance for Tiny Object Detection on Airport Runways," in *2023 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, pp. 1–6, Dec. 2023.

[86] Y. Zhang, C. Wu, T. Zhang, Y. Liu, and Y. Zheng, "Self-Attention Guidance and Multiscale Feature Fusion-Based UAV Image Object Detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 20, pp. 1–5, 2023.

[87] A. Miri Rekavandi, S. Rashidi, F. Boussaid, S. Hoefs, E. Akbas, and M. Bennamoun, "Transformers in Small Object Detection: A Benchmark and Survey of State-of-the-Art," *ACM Comput. Surv.*, vol. 58, pp. 64:1–64:33, Sept. 2025.

[88] X. Yu, X. Liu, and G. Liang, "YOLOv8-SMOT: An Efficient and Robust Framework for Real-Time Small Object Tracking via Slice-Assisted Training and Adaptive Association," July 2025.

[89] Y. Kondo, N. Ukita, R. Kanayama, Y. Yoshida, T. Yamaguchi, X. Yu, G. Liang, X. Liu, G.-Z. Wang, W.-T. Chu, B.-C. Chuang, J.-H. Lee, P.-T. Kuo, I.-H. Chu, Y.-S. Hsiao, C.-H. Wu, P.-Y. Wu, J.-C. Tsou, H.-C. Liu, C.-Y. Lee, Y.-F. Yang, K. Shigematsu, A. Shin, and B. Tran, "MVA 2025 Small Multi-Object Tracking for Spotting Birds Challenge: Dataset, Methods, and Results," July 2025.

[90] H.-Y. Hou, M.-Y. Shen, C.-C. Hsu, E.-M. Huang, Y.-C. Huang, Y.-C. Xia, C.-Y. Wang, and C.-Y. Lee, "Ensemble Fusion for Small Object Detection," in *2023 18th International Conference on Machine Vision and Applications (MVA)*, pp. 1–6, July 2023.

[91] D. Huo, M. A. Kastner, T. Liu, Y. Kawanishi, T. Hirayama, T. Komamizu, and I. Ide, "Small Object Detection for Birds with Swin Transformer," in *2023 18th International Conference on Machine Vision and Applications (MVA)*, pp. 1–5, July 2023.

[92] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More Features from Cheap Operations," Mar. 2020.

[93] Y. Tang, K. Han, J. Guo, C. Xu, C. Xu, and Y. Wang, "GhostNetV2: Enhance Cheap Operation with Long-Range Attention," Nov. 2022.

[94] H. Lee, W. Jin, S.-H. Baek, and S. Cho, "Generalizable novel-view synthesis using a stereo camera," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[95] S. Watanabe, "Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance," May 2023.

[96] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Pattern Recognition: 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings 36*, pp. 31–42, Springer, 2014.

[97] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3061–3070, 2015.

[98] J. Li, P. Wang, P. Xiong, T. Cai, Z. Yan, L. Yang, J. Liu, H. Fan, and S. Liu, "Practical Stereo Matching via Cascaded Recurrent Network with Adaptive Correlation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (New Orleans, LA, USA), pp. 16242–16251, IEEE, June 2022.

[99] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A Naturalistic Open Source Movie for Optical Flow Evaluation," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 611–625, Springer, 2012.

[100] J. Tremblay, T. To, and S. Birchfield, "Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation," arXiv, July 2018.

[101] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3260–3269, 2017.

[102] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, IEEE, 2012.

[103] Q. Zhou, K. Sheng, X. Zheng, K. Li, Y. Tian, J. Chen, and R. Ji, "Training-Free Transformer Architecture Search With Zero-Cost Proxy Guided Evolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, pp. 6525–6541, Oct. 2024.

[104] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," July 2018.