IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version February 11, 2025.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Beyond Performance: Designing a Super-Resolution Architecture Search Space and a Hybrid Multi-Objective Approach for Neural Architecture Optimization

JESÚS LEOPOLDO LLANO GARCÍA[1], RAÚL MONROY[1], VÍCTOR ADRIÁN SOSA HERNÁNDEZ[1], and KALYANMOY DEB[2], (Fellow, IEEE)

[1]Tecnologico de Monterrey, School of Engineering and Sciences, Av. Lago de Guadalupe Km 3.5, Atizapán de Zaragoza, Edo. Mex. 52926 Mexico
[2]Michigan State University, Electrical and Computer Eng., 428 S. Shaw Lane East Lansing, MI 48824-1226 USA

Corresponding author: Raúl Monroy (e-mail: raulm@tec.mx).

This work was supported in part by the National Council of Humanities, Science, and Technology (CONAHCyT) under Grant CF-2023-I-801 (Ciencia de Frontera 2023). The first author acknowledges financial support from CONAHCyT through Scholarship Grant 829049. This work was also supported in part by Azure sponsorship credits granted by Microsoft's AI for Good Research Lab.

ABSTRACT Multi-objective neural architecture search (NAS) for super-resolution image restoration (SRIR) targets models that simultaneously deliver high-fidelity reconstructions and respect strict computational budgets—requirements that single-objective searches routinely overlook. In response, we introduce BASS, a versatile multi-branch search space, together with a hybrid optimisation framework that couples NSGA-III's global exploration with fine-grained local search. The local phase is preliminarily instantiated with hill climbing, tabu search and simulated annealing, enabling a systematic comparison of their ability to refine candidate architectures. We cast the task as a tri-objective problem that maximises predicted PSNR while minimising floating-point operations and parameter count, ensuring that discovered networks remain both accurate and efficient. Extensive experiments show that hybridising NSGA-III with local search accelerates convergence and consistently yields superior architectures; the hill-climbing variant offers the best overall trade-off. The resulting BASSN models attain competitive peak signal-to-noise ratios across $\times 2$, $\times 3$ and $\times 4$ upscaling factors while fitting diverse resource envelopes, providing a practical and scalable route toward deployable SRIR solutions.

INDEX TERMS Automated Machine Learning, Deep Learning, Evolutionary Computation, Image Restoration, Neural Architecture Search, Pareto Optimization, Superresolution.

I. INTRODUCTION

Neural Architecture Search (NAS) automates deep neural network design, optimizing architectures for tasks like Super-Resolution Image Restoration (SRIR) [1], [2]. However, NAS is computationally expensive, as evaluating each candidate requires significant resources [3]. Traditional NAS prioritizes accuracy, often neglecting efficiency, making it impractical for real-world applications where computational constraints matter. Multi-Objective NAS (MoNAS), particularly Evolutionary NAS (ENAS), addresses this by optimizing accuracy and efficiency, although at high evaluation costs. SRIR exemplifies a domain that requires this trade-off, as models must balance performance with deployability on hardware-limited platforms [4]–[6].

Hybrid NAS strategies improve ENAS efficiency by combining broad exploration with targeted refinement [7]. Although evolutionary multi-objective optimization Algorithms (EMOAs) with local search have proven effective in related tasks [8], they remain underutilized in NAS. Their integration could improve NAS by reducing evaluations while preserving the quality of the solution.

This work introduces the Branching Architecture Search Space (BASS), a framework that allows dynamic layer selection and repetition to optimize SRIR architectures [9], [10]. BASS supports multi-depth feature extraction, balancing computational cost and performance, making it suitable for constrained environments [5]. We also propose a hybrid NAS approach, integrating EMOA-based global search

with local refinements to accelerate convergence and improve Pareto front approximations [11].

Our results demonstrate that this approach reduces function evaluations while improving search efficiency. A hybrid variant achieved the highest hypervolume (1.572) while reducing FLOPs to 223 billion—substantially improving over the 982 billion FLOPs in the standard EMOA baseline. These findings highlight the method's ability to balance Peak Signal-to-Noise Ratio (PSNR), FLOPs, and trainable parameters, advancing NAS toward more efficient, high-performance architectures applicable across multiple domains.

II. BACKGROUND AND RELATED WORK

Neural Architecture Search (NAS) automates deep neural network design, optimizing architectures for tasks like Super-Resolution Image Restoration (SRIR) [1], [2]. While traditional NAS maximizes accuracy, its high computational demands challenge real-world deployment. Multi-Objective NAS (MoNAS) addresses this by balancing accuracy with efficiency, although at a high evaluation cost. SRIR exemplifies this trade-off, requiring models that perform well under strict computational constraints [4]–[6].

A. EVOLUTIONARY NAS IN SRIR

Early applications of Evolutionary Algorithms (EAs) in SRIR, such as Suganuma et al. [12], used genetic programming to evolve convolutional autoencoders, though their limited search space restricted architectural diversity. Van Wyk and Bosman [13] expanded the search with more flexible layers but faced inconsistent performance, highlighting the need for more sophisticated strategies beyond search space expansion.

B. HYBRID EVOLUTIONARY ALGORITHMS IN NAS

Hybrid approaches improve EAs by integrating reinforcement learning (RL), local search, or other optimization methods. Grosan and Abraham [8] classify these into four types: global-local, relay, cooperative, and embedded hybrids. Zhou et al. [14] demonstrated how local search refines promising solutions. Glorieux [15] introduced cooperative hybrids that exchange information between algorithms, and Robles et al. [16] embedded estimation-of-distribution models within genetic algorithms to improve efficiency.

C. REINFORCEMENT LEARNING IN ENAS

Chu et al. [17] combined NSGA-II [18] with RL to refine mutation strategies, effectively balancing exploration and exploitation. However, RL's increased complexity and reliance on predefined components limited architectural diversity. Their later work [19] introduced a hierarchical search space, improving granularity but increasing computational demands. The need for a uniform evaluation between candidates further constrained scalability.

D. LOCAL SEARCH WITHIN ENAS

Hybrid approaches that integrate local search improve NAS efficiency by refining architectures mid-search. GENAS [20] combined genetic algorithms with differentiable NAS to iteratively fine-tune architectures, reducing evaluation costs. However, its reliance on differentiable models restricted architectural flexibility. Evans et al. [21] incorporated gradient descent to optimize post-evolution hyperparameters, improving accuracy but increasing the risk of overfitting. Wistuba [22] applied function-preserving mutations to accelerate convergence, but its task-specific nature limited generalizability.

E. ZERO-COST PROXIES: SYNAPTIC FLOW FOR NAS

To mitigate evaluation costs, zero-cost proxies such as Synaptic Flow estimate performance using early-stage network dynamics [23]. By analyzing gradients and weight updates, SynFlow eliminates complete training cycles, significantly reducing computational overhead. However, its accuracy varies with architectural complexity [24], and its generalization across diverse tasks remains an area for improvement [25]. Despite these limitations, SynFlow represents a promising direction for accelerating NAS in resource-constrained applications.

F. SRIR-NAS METHODS

Several NAS strategies have been tailored specifically to SRIR. Song et al. [26] produced the models named ESRN, using evolutionary search over three handcrafted residual blocks. While effective under PSNR and computational constraints, its fixed cell types limit expressiveness and novelty.

The NAS-DIP model is obtained by [27] leveraging Deep Image Priors to search over U-Net-like structures using reinforcement learning without full supervision. Though its results trail fully trained models, it showcases the inductive power of architecture alone.

MoreMNAS [28] applies a hybrid NSGA-II and RL approach, optimizing across accuracy, FLOPs, and parameter count via hierarchical mutation operators. FALSR [29] extends this with a two-level search space and a refined composed mutation strategy, improving efficiency and solution granularity.

HNAS [30] uses a scalarized reward in an RL framework to explore a dual-cell supernet. This allows controllable trade-offs but requires multiple runs and manual tuning of reward weights. Finally, DLSR [31] adopts differentiable NAS over residual distillation blocks and learned skip connections, guided by handcrafted loss functions targeting both perceptual and efficiency metrics.

These representative methods illustrate the diverse directions explored in SRIR-focused NAS. While each contributes important innovations, many retain strong dependencies on fixed cell structures, handcrafted components, or scalarized formulations. In this paper, we shall see how to get around such limitations looking into more flexible search spaces and optimization schemes.

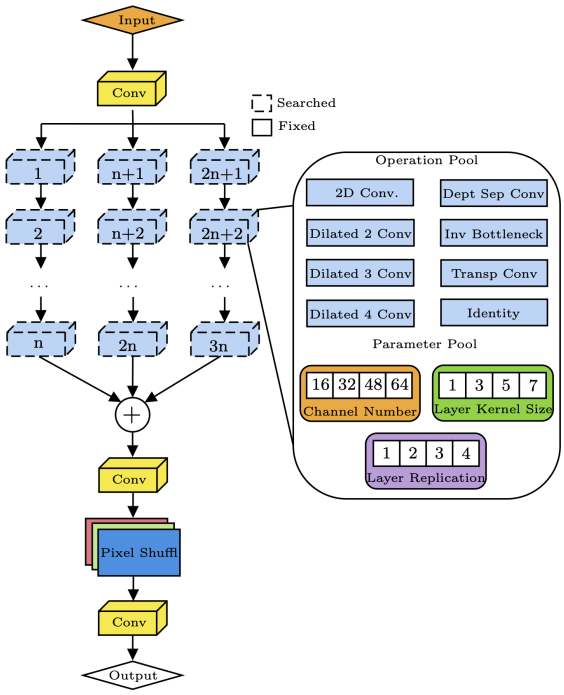


FIGURE 1: The diagram illustrates the BASS search space, showcasing three distinct pathways within the network's backbone. It highlights the diverse operations available, including 2D, depth-wise separable, and dilated convolutions. Searchable operations, indicated by dashed lines and blue shading, can be selected from the operation pool, while fixed operations are shown in solid yellow. The parameter pool allows for adjustments to channel numbers, kernel sizes, and layer replication, enabling fine-tuning of the model's architecture.

III. DESIGN OF THE BRANCHING ARCHITECTURE SEARCH SPACE

In this section, we introduce BASS, a search space specifically developed to facilitate the exploration and refinement of neural architectures tailored for SRIR applications. As illustrated in Figure 1, BASS architectures are based on an architectural design that establishes a main pathway branching into multiple distinct feature extraction paths. These branches enable independent processing of features at varying scales, converging into a unified structure that leads to an upsampling stage. This layout supports the construction of architectures with varying depths and complexities, allowing BASS-generated models to span a wide range of potential performance and efficiency profiles. The choice of a multi-branch configuration is motivated by the demonstrated effectiveness of multi-scale feature fusion in super-resolution [32]–[34], which is essential for capturing both fine-grained textures and broader structural coherence in high-fidelity image reconstruction.

In an attempt to extract fine-grained patterns while preserving global context, we shall see that in the design of BASS we aimed at architectures that process information in parallel across branches of varying complexity. This strategy aligns with recent findings indicating that multi-branch networks improve representational flexibility and exhibit smoother loss surfaces and lower non-convexity, favoring more stable optimization [35], [36]. In contrast to Transformer- and GAN-based approaches, which often emphasize perceptual quality at the cost of computational load, BASS emphasizes a favorable trade-off between reconstruction accuracy (e.g., PSNR) and computational efficiency. Its modular and repeatable structure supports dynamic architectural composition, aiming at deployment even under resource constraints [37]. Nevertheless, the BASS topology cannot express architectures that depend on extensive, long-range skip connections—such as deep residual networks (e.g., ResNets)—because its design emphasizes modular, convolution-centric, multi-branch pathways rather than backbone-spanning residual links. As a result, entire families of residual models remain outside BASS's representational scope.

The Multi-Depth Branch Network (MDBN) design forms a critical aspect of BASS, creating multiple branches of varying depths within the main network backbone. This structure leverages the hierarchical nature of neural networks to optimize feature extraction, with deeper branches focusing on detailed information and shallower branches capturing broader contextual elements. The variations in depth within a single network address SRIR's demand for precision and efficiency, enabling BASS to balance feature detail with computational constraints. MDBNs also help mitigate the duality gap, the discrepancy between training objectives and deployment performance, particularly pronounced in SRIR tasks requiring processing of high-resolution images [38], [39].

A. OPERATION POOL SELECTION

The main components of BASS are selected from a curated pool of operations, each contributing to specific aspects of feature extraction, spatial resolution, and computational economy:

- *2D Convolution*: A cornerstone operation in neural networks, 2D convolutions are extensively used in SRIR for feature extraction. They serve as the building blocks for more complex transformations within the network [40].
- *dil_conv_d2*, *dil_conv_d3*, *dil_conv_d4*: These dilated convolutions increase the receptive field of the standard convolution, enhancing the model's ability to incorporate broader contextual information essential in SRIR.
- *Dsep_conv*: A depthwise separable convolution that optimizes computational load by dividing spatial and depth operations, supporting efficient feature extraction.
- *invert_Bot_Conv_E2*: An Inverted Bottleneck structure expands and compresses channels to maximize feature extraction efficiency, drawing from mobile architecture principles [41].

- *conv_transpose*: A transposed convolution operation that reverses the convolution process for upsampling, crucial in SRIR for enhancing output resolution [42].
- *identity*: A pass-through operation that preserves data flow without alteration, improving gradient flow and stability in deeper networks [43].

Each of these operations can be assigned to nodes over BASS's architectural backbone, allowing for multiple configurations that can adjust to SRIR's requirements for complexity and efficiency.

B. CONFIGURABLE PARAMETERS IN BASS

The BASS design incorporates three primary configurable parameters that influence the behavior of layers and, consequently, the branching characteristics of the architectures: channel size, kernel size, and operation repetition. Each parameter plays a distinct role in defining network depth, resolution capacity, and feature extraction capability.

- Channel size dictates the number of feature maps in each layer, ranging from 16 to 64. Larger channel sizes increase representational capacity and raise computational costs, providing a tradeoff between richness and efficiency [44].
- Kernel size choices include 1×1, 3×3, 5×5, and 7×7, allowing adjustment of each layer's spatial extent. Smaller kernels capture finer details, while larger kernels enhance contextual understanding. This variety supports BASS's ability to handle SRIR's spatial resolution needs [31].
- Operation repetition manages the depth of each layer by controlling sequential applications of the same operation. Repetition of up to four times the same operation typically results in richer feature representations without adding excessive computational loads.

These parameters, combined with the diverse types of layers incorporated in BASS, enable the creation of deeper and broader architectures that capture a range of trade-offs between performance and computational cost. This variability in the multi-branch structure of architectures within BASS allows it to serve as a foundation for search strategies to identify architectures suited to specific challenges within the SRIR domain.

C. ENCODING AND DECODING OF ARCHITECTURES

Each architecture inside BASS is represented by an 84-bit binary sequence structured through Gray codes to facilitate BASS exploration and compatibility with different search strategies. This encoding scheme is organized into 28 groups of 3 bits, where each group corresponds to specific architectural parameters: the initial 3 bits encode a single channel number applied across the architecture, while the remaining groups define characteristics for nine operations. Each operation segment specifies its type, kernel size, and repetition count (from 1 to 4), allowing each architecture in BASS to represent a unique configuration with a global channel size and specific operational details.

Gray codes guarantee that neighbouring architectures differ by exactly one bit, giving a Hamming distance of $d_{\mathrm{H}} = 1$. This produces smooth, incremental mutations and prevents disruptive jumps. Because every Gray bit-string decodes to a valid network, configuration collisions are impossible, making this encoding ideal for BASS.

For instance, incrementing the index from 3 to 4 flips three bits in binary ($011 \rightarrow 100$, $d_{\mathrm{H}} = 3$), but only one bit in Gray code ($010 \rightarrow 110$, $d_{\mathrm{H}} = 1$). As a result, each mutation in BASS affects only one architectural component, enabling controlled, fine-grained navigation through the discrete search space.

Mathematically, the encoding maps Gray sequences to architectural parameters:

- Let $G = \{g_1, g_2, \dots\}$ be the Gray code sequence.
- *Channel-size encoding:*

$$C = f(g_1),$$

  where $f(g_1)$ returns the global channel count.
- *Operation encoding:* for each operation in branch j and block k,

$$\mathrm{Operation}_{j,k} = \big(f(g_n), f(g_{n+1}), f(g_{n+2})\big),$$

  where n runs from 2 to 28; the three function calls map to operation type, kernel size, and repetition count.

This scheme maintains a well-structured search space, where each bit-wise change on the chromosome of a candidate architecture is valid and corresponds to a precise, single-step transition.

Central to BASS's functionality, the decoding process transforms each binary sequence into the network's structural components. The Gray code encoding reduces disruptive shifts between configurations, allowing smooth transitions and stable exploration within the search space [45]. Each decoded integer corresponds to a specific node in the MDBN, mapping architectural features like channel numbers, operation types, kernel sizes, and repetition counts to create a fully reconstructed network tailored to BASS's requirements for SRIR.

Our focus is primarily on the decoding process within the BASS, as encoding preexisting architectures is inherently complex and context-specific. Given the vast variability in neural network designs, creating a universal encoding scheme that accommodates all possible architectures is beyond the scope of this work. This approach emphasizes the search techniques, positioning the decoding process as a critical component for exploring BASS and optimizing architectures.

IV. HYBRID APPROACH FOR MULTI-OBJECTIVE NEURAL ARCHITECTURE SEARCH

The search for optimal neural network architectures in MoNAS involves navigating complex, multidimensional spaces with competing objectives like maximizing accuracy and minimizing computational costs. EMOAs are effective

in these contexts, leveraging global exploration to maintain diversity across solutions. Notably, NSGA-III's reference point-based selection helps distribute solutions along the Pareto front, balancing conflicting objectives [46], [47]. However, it and similar algorithms often struggle to exploit local, high-potential regions where finer optimizations could yield substantial gains. Critical configurations may be missed without such localized adjustments, reducing solution quality.

To address these limitations, we augmented NSGA-III's global exploration with targeted local search refinements, enhancing its ability to push solutions closer to Pareto optimality. Our hybrid approach improves convergence speed and solution quality by periodically applying a local search technique to refine architectures within the NSGA-III population. This addition of localized optimization allows the algorithm to capture configurations that might otherwise be missed.

We explore our hybrid ENAS methodology, starting with the problem formulation that drives our search objectives and leveraging NSGA-III to manage multidimensional trade-offs with diverse solutions. The integration of local search techniques within NSGA-III is examined, highlighting how each contributes to refining candidate architectures. We then outline the algorithm's configuration and workflow, focusing on how this hybrid approach effectively balances broad exploration and focused local optimization, ultimately producing robust neural architectures for resource-constrained applications.

A. OPTIMIZATION PROBLEM DEFINITION

The goal of our NAS approach is to discover a set of optimal neural architectures, α, with associated weights $w^*(\alpha)$ that maximize the PSNR between super-resolution images and their high-resolution counterparts while minimizing computational demands, quantified by FLOPs and the number of trainable model parameters. Formally, we define this as the following multi-objective optimization problem:

$$F(\alpha, w^*(\alpha)) = \begin{cases} f_1 = \max\limits_{\mathcal{D}_{\text{valid}}} \text{PSNR}, \\ f_2 = \min\limits_{\alpha} \text{FLOPs}(\alpha), \\ f_3 = \min\limits_{\alpha} \text{Parameters}(\alpha) \end{cases} \tag{1}$$

$$\text{subject to} \quad \begin{aligned} g_1(\alpha, w^*(\alpha)) &= \text{PSNR} \geq 20, \\ g_2(\alpha, w^*(\alpha)) &= \text{FLOPs} \leq \text{limit}. \end{aligned} \tag{2}$$

The PSNR is computed using:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{(L-1)^2}{\text{MSE}}\right), \quad \text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(O_{i,j} - D_{i,j})^2 \tag{3}$$

where $O_{i,j}$ and $D_{i,j}$ represent the original and predicted pixel intensities at position i, j in an image, $L = 255$ denotes the maximum pixel value, and $m \times n$ defines the image dimensions. Here, objective f_1 focuses on maximizing PSNR (equivalent to minimizing Mean Squared Error), while objectives f_2 and f_3 ensure that solutions meet computational constraints suitable for deployment.

The constraints impose specific requirements on the architectures. The first constraint ensures a minimum PSNR of 20 dB, guaranteeing a baseline level of image quality in the super-resolution output, as PSNR values below this threshold generally indicate poor visual quality. The second constraint, limiting the number of FLOPs, ensures that the architecture remains computationally feasible, allowing for deployment in resource-constrained environments.

B. HYBRIDIZATION RATIONALE AND DESIGN

Hybrid metaheuristics that combine population-based search with localized refinement are widely used to balance exploration and exploitation in optimization. In our approach, NSGA-III ensures diversity across multiple objectives—such as performance (e.g., PSNR) and computational cost (e.g., FLOPs, parameter count)—but may overlook small-scale structural changes in neural architectures that yield significant improvements.

To address this, we incorporate periodic local searches to refine candidate architectures without excessive computational overhead. Unlike standard selection mechanisms that favor high-performing solutions, our approach broadens the refinement scope by selecting a random individual every L generations ($L = 50$). This ensures broader exploration and enhances diversity by allowing any individual to undergo refinement, including those not on the Pareto front. To intensify local exploration, the selected architecture is modified with a mutation probability of $P_{lm} = 0.25$, significantly higher than the global mutation rate. If, after 10 modification steps, the new variant improves at least one objective without degrading others, it replaces its predecessor.

Algorithm 1 begins by initializing a population of N architectures, evaluated on multiple objectives, and maintaining diversity via NSGA-III's reference-point niching (lines 1–3). In each generation, offspring Q are generated via crossover (P_c) and mutation (P_m) (line 5), evaluated, and merged with the parent population (lines 6–8). The next generation is then selected using non-dominated sorting.

Every L generations, local search modifies a randomly chosen individual (line 11). Over I evaluations, the refined individual replaces its predecessor if it dominates it; otherwise, it is discarded. The updated individual is then reintegrated into the population (line 19). This hybrid strategy preserves NSGA-III as the primary driver of global exploration while leveraging periodic refinements to enhance architectural discovery.

The local search strategy applies a bit-flip mutation to individual bits of the candidate chromosome. While all possible bitstrings correspond to structurally valid architectures by design—eliminating the possibility of syntactic collisions or invalid configurations—only those variants that satisfy the computational constraints specified in Eq. (2) (e.g., FLOP and parameter limits) are considered feasible. After each

mutation, the resulting architecture is verified against these constraints. Infeasible variants are immediately discarded, the current chromosome is reset (a fresh bit-flip is applied to the last feasible individual) and the local-search loop continues. This back-track strategy caps wasted effort on persistently infeasible regions.

C. PARAMETER CONFIGURATION

To balance computational feasibility and solution diversity, the population size is set to 20, providing sufficient search space coverage while keeping evaluations manageable—an approach commonly used in NAS studies [48], [49]. A total of 1,250 generations ($G = 1250$) ensures the necessary depth to explore complex architecture spaces and identify high-performance solutions [50], [51].

The evolutionary process is guided by a crossover probability of $P_c = 0.9$ and a mutation rate of $P_m = 0.01$, balancing diversity with search stability [52], [53]. Local search occurs at fixed intervals ($L = 50$ generations), selectively refining individuals with $P_{lm} = 0.25$ [54], [55]. Unlike baseline mutations that promote gradual variation, local search intensifies exploration in targeted regions of the architecture space, complementing the broader search dynamics of NSGA-III [56], [57].

This configuration effectively trades global exploration and local precision, ensuring performance gains while maintaining computational efficiency.

V. FROM LOCAL SEARCH TO HYBRID SEARCH: EVALUATING OUR OPTIMIZATION STRATEGY

This section compares the proposed hybrid MOENAS strategy against standard MOENAS, focusing on convergence speed and computational efficiency, assuming a reliable estimation of solution quality. We evaluate our Hybrid ENAS approach for super-resolution tasks using the BASS framework for architectural exploration.

We begin by examining three local search methods—hill climbing (HC), tabu search (TS), and simulated annealing (SA)—to understand their respective strengths for refining candidate architectures within BASS. We then build hybrid NSGA-III variants that incorporate each local search technique into the evolutionary process, comparing them against the standard NSGA-III baseline. In the following sections, we detail the performance gains and trade-offs that hybridization introduces when searching for optimal architectures under multi-objective constraints.

A. SEARCH OBJECTIVES AND ARCHITECTURE RANKING CRITERIA

This subsection defines the objectives and evaluation criteria used during our multi-objective neural architecture search. Several strategies have been proposed to estimate the potential performance of candidate architectures without requiring training or testing. Among these, the most promising for lightweight, training-free evaluation fall into two main categories: zero-cost proxies and learning-based surrogates.

Zero-cost proxies such as GraSP [58] and SNIP [59] estimate architectural saliency at initialization—GraSP through gradient magnitude, and SNIP by measuring loss sensitivity to weight pruning. Though effective in classification, these methods overlook the structural and topological needs of image restoration tasks. Learning-based surrogates [60], which regress performance metrics like PSNR from prior architecture-performance data, offer more task-specific accuracy and stable rankings. However, they require large datasets, careful calibration, and introduce computational overhead unsuited to early-stage, training-free evaluation. As our goal is rapid candidate filtering, we rely solely on zero-cost proxies and defer surrogate-based modeling to future work.

To score architectural candidates during the search, we employ Synaptic Flow (SynFlow) as a zero-cost proxy for network performance [61]. SynFlow evaluates a network by analyzing weight gradients under a uniform weight initialization, ensuring unbiased gradient flow throughout the architecture. Models with disrupted gradient propagation tend to struggle during optimization, making SynFlow an effective indicator of trainability.

Within BASS, which defines architectures with dynamic, multi-branch topologies, SynFlow serves as a structural diagnostic: it reveals whether gradient signals are preserved across parallel branches, a key factor for ensuring effective spatial feature integration in super-resolution [62]. This property makes SynFlow particularly well-suited for our search space, where architectural responsiveness depends on seamless signal flow across heterogeneous paths. We shall see that, in the context of SRIR, using SynFlow suffices to identify architectures that are either competitive or state-of-the-art when evaluated under a multi-objective setting, serving as a zero-cost proxy to filter out designs with disrupted gradient flow before further evaluation.

In addition to SynFlow, our search considers two secondary objectives:

- **Total Number of Trainable Parameters**: Indicates the memory demands of each architecture.
- **Number of FLOPs**: Captures computational complexity.

B. COMPARISON OF LOCAL SEARCH APPROACHES

HC, TS, and SA were tested on 30 initial populations (each with 20 individuals) under identical starting conditions to ensure a fair comparison. Table 1 summarizes the key experimental parameters.

Each method refined a randomly selected architecture by applying bit-flip mutations with probability $P_{lm} = 0.25$. For every 1,000 evaluations, a new individual was selected, leading to 25,000 total evaluations per run. Performance was assessed using SynFlow, parameter count, and FLOPs, with results summarized in Table 2.

HC achieved the highest SynFlow score (9.13), suggesting superior gradient flow preservation.

TS produced the most compact architectures, minimizing both parameters (0.96M) and FLOPs (4.75B), but at the cost

Algorithm 1 Hybrid NSGA-III with Periodic Local Search

Require: Population size N, total generations G, local search interval L, local search evaluations I, crossover probability $P_c = 0.9$, mutation probability $P_m = 0.01$, local search mutation probability $P_{lm} = 0.25$
Ensure: Final population P containing Pareto-optimal solutions

```
Initialize population P with N randomly generated architectures
Evaluate each individual in P on all objectives (e.g., PSNR, FLOPs)
Generate reference points for NSGA-III to guide selection
for g ← 1 to G do
    Q ← GenerateOffspring(P, P_c, P_m)          ▷ Apply two-point crossover and bit-flip mutation
    Evaluate each individual in Q on the objectives
    R ← P ∪ Q
    P ← SelectNextGeneration(R)                 ▷ Apply non-dominated sorting and reference-point niching
    if g mod L = 0 then
        individual ← RandomSelection(P)         ▷ Select a random individual for local search
        for i ← 1 to I do
            mutated_ind ← ApplyLocalSearch(individual, P_lm)
            Evaluate mutated_ind on the objectives
            if mutated_ind dominates individual then
                individual ← mutated_ind        ▷ Replace if improved
            end if
        end for
        UpdatePopulation(P, individual)         ▷ Reintegrate refined individual
    end if
end for
    return P
```

Algorithm 2 Apply Local Search

Require: Individual *individual*, mutation probability P_{lm}, total evaluations $E = 1000$
Ensure: Mutated individual

```
candidate ← Duplicate(individual)
evals ← 0
while evals < E do
    success ← 0
    for each parameter in candidate do
        if Random() < P_lm then
            Mutate parameter; success ← success + 1
            if success = 10 then break
            end if
        end if
    end for
    Evaluate candidate; evals ← evals + 1
end while
    return candidate
```

of reduced gradient flow (8.85 SynFlow), potentially affecting optimization stability.

SA balanced exploration and exploitation, achieving a SynFlow score of 9.05 while producing larger models (1.28M parameters, 5.45B FLOPs). While it did not showcased bad performance, its higher resource demands did not offer a clear advantage over HC or TS.

Given the focus on deep learning architectures, where trainability is critical for achieving competitive performance, HC is preferred. While compactness is relevant, strong gradient propagation is essential for efficient optimization. TS offers better efficiency, but the lower SynFlow scores obtained by the models it found suggest potential training difficulties, making HC the more robust choice in this context.

C. HYBRID NSGA-III PERFORMANCE EVALUATION

In the context of NAS, we evaluated how hybridizing ENAS with each of the three local searches (HC, TS, SA) impacts the evolutionary process within the BASS framework. As a baseline, we selected NSGA-III for its strong track record

TABLE 1: Experimental parameters for evaluating HC, TS, and SA under consistent initial conditions.

Parameter	Description
Initial Population	20 individuals (random)
Selected Individual	Random
Neighborhood Search	Bit-flip mutation
Mutation Probability	$P_{lm} = 0.25$
Evaluations	25,000
Sampling Interval	Every 10 steps
Selection Criteria	Non-strict Pareto dominance

in handling high-dimensional objective spaces, maintaining diverse solutions, and employing reference points to locate Pareto-optimal fronts. These properties make NSGA-III particularly well-suited for complex multi-objective tasks, as it balances convergence and diversity effectively.

Each hybrid variant was evaluated against the standard NSGA-III algorithm, with all methods beginning from a randomly initialized population and executing a total of 50,000 function evaluations—25,000 for NSGA-III and 25,000 for the respective local search. This setup allowed us to assess whether hybridization improves convergence speed, solution quality, and computational efficiency. Table 3 details the standard initialization parameters, ensuring a consistent baseline across approaches.

Table 4 summarizes the SynFlow scores, trainable parameters, and FLOPs across 30 runs. The hybrid methods consistently outperformed NSGA-III.

NSGA-III-HC achieved the highest SynFlow score (6.45×10^4) and the lowest FLOP count (2.23×10^8), indicating strong gradient preservation and efficient computation. NSGA-III-TS produced the most compact architectures, minimizing parameters (8.18×10^4).

NSGA-III-SA, however, exhibited lower SynFlow scores (5.80×10^4) and higher FLOP counts (5.36×10^8), suggesting that the introduction of SA did not really benefit the search process in this context.

We also employed the hypervolume metric—an indicator of quality and diversity along the Pareto front—to compare solution sets (Table 5). NSGA-III-HC yielded the highest hypervolume, indicating broader coverage of the objective space and reinforcing the advantages of hybridization for super-resolution tasks.

Figure 2 illustrates how average SynFlow scores progress over 1250 generations for each variant, showing convergence patterns and final performance. Notably, NSGA-III-HC and NSGA-III-TS achieved Pareto optimality levels similar to standard NSGA-III but required fewer function evaluations, demonstrating faster convergence. Given the same computational budget, these hybrid approaches refined the Pareto front beyond standard NSGA-III, delivering higher quality and diversity solutions.

Figure 3 presents FLOP reductions across algorithms. NSGA-III-TS achieved the lowest FLOP values by consistently selecting compact architectures while maintaining competitive performance. NSGA-III-HC rapidly improved SynFlow through targeted local refinements, whereas NSGA-III-TS resulted on a steady exploitation of small architectures through the search.

Overall, hybridization accelerated convergence and yielded more diverse Pareto-optimal sets, making hybrid NSGA-III variants particularly well-suited for computationally intensive tasks like super-resolution NAS. Among the three hybrid strategies explored, NSGA-III-HC consistently outperformed the others in both convergence speed and solution diversity. Statistical analyses (Wilcoxon Signed-Rank and paired t-tests with $\alpha = 0.05$) confirmed the significance of these differences, as highlighted by the asterisks in the corresponding tables.

Based on these results, we select NSGA-III-HC as the hybrid strategy for the remainder of this study. Having validated that our hybrid approach outperforms standard NSGA-III in both convergence and solution quality, we now turn to evaluating the architectures it discovers against state-of-the-art SRIR methods. From its resulting Pareto front, we extract representative architectures located at both extremes and the knee region—collectively referred to as BASSN variants. These models serve to benchmark our NAS-generated solutions against machine-crafted SRIR approaches using a unified training and evaluation protocol. To ensure robustness and avoid selection bias, the selected architectures were drawn from the Pareto front corresponding to the median approximation seed across 30 independent NSGA-III-HC runs, thus excluding both overly optimistic and overly pessimistic cases.

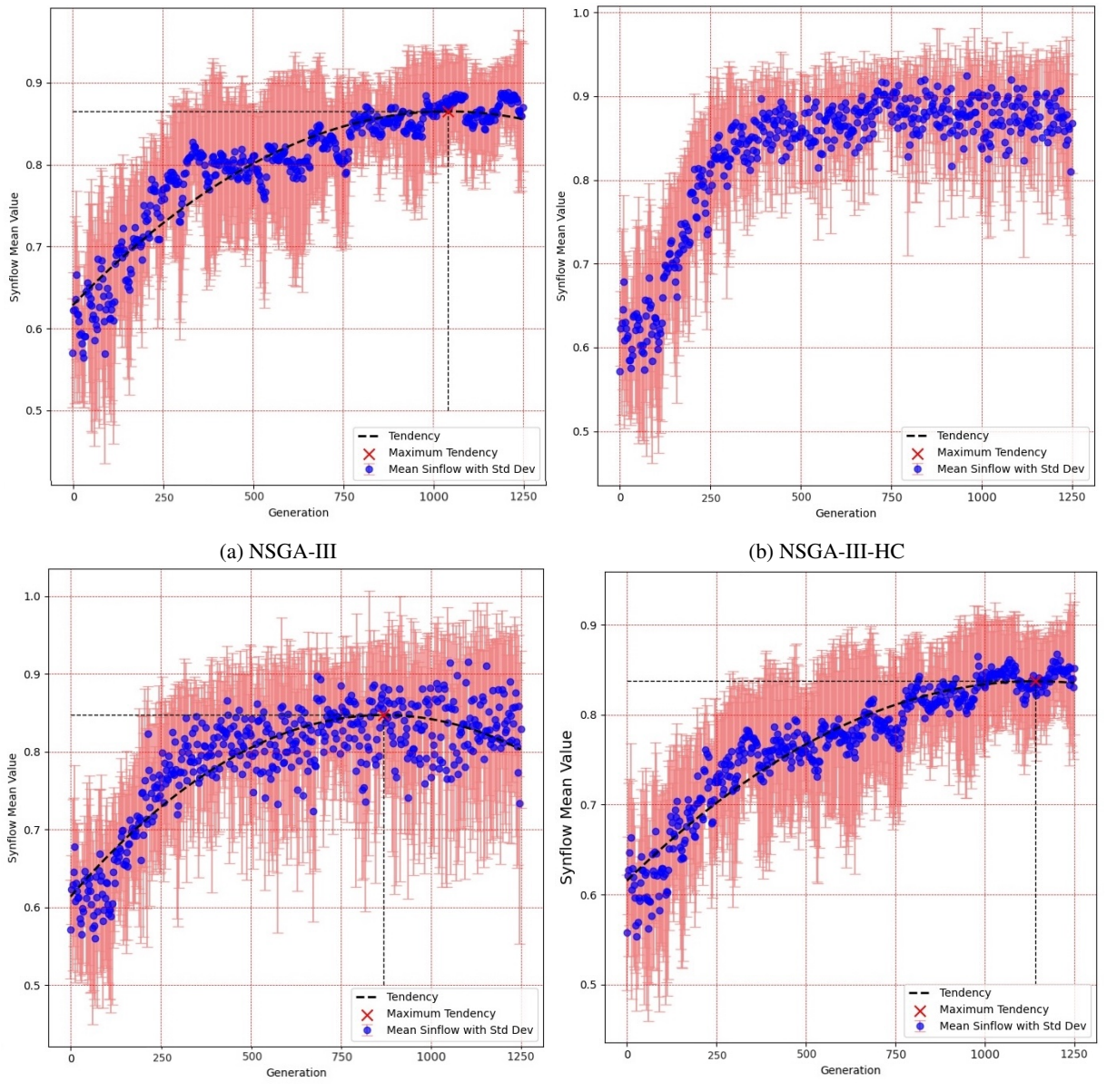
D. COMPUTATIONAL SETUP AND RUNTIME

To support reproducibility and assess the practicality of our method, we report the system configuration and runtime required for architecture search. All search experiments were conducted on a high-memory virtual machine (Standard FX48mds) hosted on Microsoft Azure, provisioned with 48 vCPUs and 1008 GiB of RAM. No GPU acceleration was used during this phase. The system was configured to run up to five independent NAS processes in parallel, each corresponding to a full hybrid NSGA-III search. Under this CPU-only setup, a single search run—comprising 1250 generations and 50,000 architecture evaluations—took on average 7.4 hours ($\pm$0.2 h) for the complete parallel batch. This runtime reflects only the architecture search stage and excludes the training of selected models.

Although the framework was not explicitly optimized for intra-process parallelism, the Python runtime and host OS exploited concurrency, running multiple search processes and evaluations in parallel. This improved throughput despite the absence of fine-grained scheduling. Furthermore, SynFlow—as a zero-cost proxy—enabled rapid, training-free evaluation of gradient flow quality, avoiding full dataset passes. This combination of concurrency and proxy-based

TABLE 2: Performance comparison of local search methods applied within the BASS. The table presents the average SynFlow scores (scaled by 10^2), number of parameters (in millions), and FLOPs (in billions) of architectures discovered by each method over 30 independent runs. Bold font highlights the best performance for each metric.

Local Search Method	**SynFlow** ($\times 10^2$)		**# Params** ($\times 10^6$)		**FLOPs** ($\times 10^9$)	
	Average	Std. Dev.	Average	Std. Dev.	Average	Std. Dev.
HC	**9.13**	1.21	1.07	0.88	5.18	2.95
TS	8.85	1.14	**0.96**	0.85	**4.75**	2.80
SA	9.05	1.15	1.28	0.89	5.45	2.95



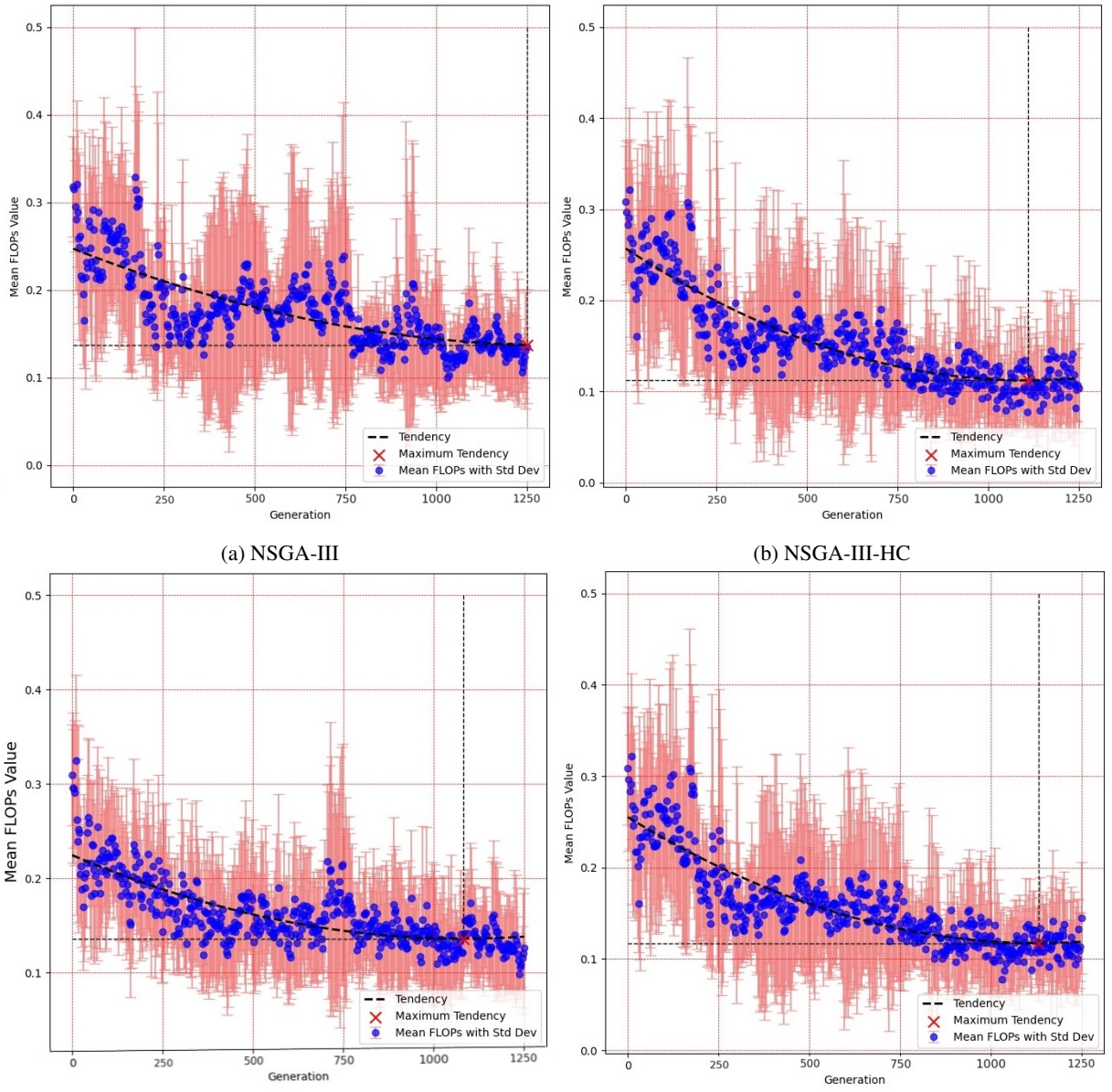
(a) NSGA-III

(b) NSGA-III-HC

(c) NSGA-III-TS

(d) NSGA-III-SA

FIGURE 2: Progression of Average SynFlow Scores over 1250 Generations



(a) NSGA-III

(b) NSGA-III-HC

(c) NSGA-III-TS

(d) NSGA-III-SA

FIGURE 3: Progression of Average Floating point operations Scores over 1250 Generations

TABLE 3: Initial parameters and genetic operators shared across all ENAS approaches evaluated, including NSGA-III. The parameters provide a consistent baseline for fair comparison among the algorithms in searching for optimal architectures within the BASS.

NSGA-III Genetic Operators	
Crossover	Two-point crossover (probability: 0.9)
Mutation	Bit-flip mutation (probability: 0.01)
Initialization	Stochastic initialization
Reference Points	Normalized hyperplane
Population Details	
Population Size	20 individuals (randomly initialized)
Generations	1250
Evaluation Process	
Total Function Evaluations	50,000 evaluations
Hybrid Evaluations	Split equally between NSGA-III and local search

evaluation supports the practical feasibility of our method under CPU-only constraints.

VI. BENCHMARKING THE RESULTS OF NSGA-III-HC VS SOTA

While Section V focused on validating the search process itself, this section shifts the focus to the quality of its outputs—specifically, the BASSN architectures derived from NSGA-III-HC—and benchmarks them against recent SRIR approaches. Having adopted NSGA-III-HC as our hybrid search method and identified the extreme and knee architectures on its Pareto front (collectively denoted as *BASSN variants*), we now describe the unified protocol used to

TABLE 4: Performance comparison of NSGA-III and its hybrid variants NSGA-III-HC, -TS, and -SA. The table shows the average and standard deviation of SynFlow scores, the number of trainable parameters, and FLOPs of architectures discovered over 30 runs. The bold text highlights the best score for each objective. An asterisk (*) indicates statistically significant results ($p < 0.05$) based on the Wilcoxon Signed-Rank and Paired t-tests.

Algorithm	SynFlow		Parameters		FLOPs	
	Average	Std. Dev.	Average	Std. Dev.	Average	Std. Dev.
NSGA-III*	6.16×10^4	7.63×10^3	2.39×10^5	6.29×10^5	9.82×10^8	2.58×10^9
NSGA-III-HC*	$\mathbf{6.45 \times 10^4}$	3.39×10^3	2.97×10^5	6.19×10^4	$\mathbf{2.23 \times 10^8}$	2.54×10^8
NSGA-III-TS*	6.36×10^4	4.95×10^3	$\mathbf{8.18 \times 10^4}$	4.72×10^4	4.36×10^8	1.26×10^9
NSGA-III-SA*	5.80×10^4	5.73×10^3	2.36×10^5	4.24×10^5	5.36×10^8	1.74×10^9

TABLE 5: Average hypervolume values achieved by NSGA-III and its hybrid variants over 30 runs. The hypervolume metric assesses the quality and diversity of the approximated Pareto fronts, with higher values indicating better performance. The bold text highlights the best average performance. An asterisk (*) indicates statistically significant results (p < 0.05) based on the Wilcoxon Signed-Rank and Paired t-tests.

Algorithm	Average Hypervolume	Std. Dev.
NSGA-III*	1.497	0.531
NSGA-III-HC*	**1.572**	0.557
NSGA-III-TS*	1.542	0.547
NSGA-III-SA*	1.331	0.489

benchmark these models against existing NAS-based SRIR methods.

A. REFERENCE METHODS

The comparison follows the experimental protocol proposed in [63], which standardizes datasets, training routines, and statistical evaluations for machine-crafted architectures. The comparison set comprises NAS-DIP [27], FALSR (A, B) [29], HNAS (A, C) [30], MoreMNAS (A) [28], ESRN / ESRN-V [26], and DLSR [31]. These baselines were selected based on the availability of source code and comparable parameter budgets. All models are evaluated under identical training and testing conditions, and performance is reported in terms of PSNR and total number of trainable parameters.

B. DATASETS AND PRE-PROCESSING

We employ the DIV2K dataset [64] for both the training and validation stages, comprising 800 training and 100 validation images. Following [63], three SRIR tasks are generated using bicubic degradation at scales $\times 2$, $\times 3$, and $\times 4$, resulting in approximately 522K training and 66K validation patches per scale. During final training (not during search), we apply horizontal/vertical flips and 90° rotations to augment each dataset eightfold.

Test results are reported on Set5 [65], Set14 [66], BSD100 [67], and Urban100 [68], using consistent bicubic downsampling. These datasets represent varying levels of texture complexity and semantic content, facilitating a more robust comparison across architectures.

C. TRAINING PROTOCOL

All models are trained over the course of 25 full epochs using the Adam optimizer, configured with an initial learning rate of 3×10^{-4}, a small numerical stability constant $\epsilon = 10^{-7}$, and a weight decay regularization term of 10^{-8} to mitigate overfitting. To ensure a stable training process and facilitate early convergence, we implement a linear warm-up strategy during the first 5 epochs, gradually increasing the learning rate from zero to its peak value. This is subsequently followed by a cosine annealing schedule, which smoothly decays the learning rate over the remaining 20 epochs. Training is conducted using a uniform mini-batch size of 32 samples across all experiments, providing consistent gradient estimation and comparable optimization dynamics between architectures.

D. EVALUATING THE DISCOVERED BASSN VARIANTS IN SRIR

These variants were selected based on their Pareto front positions with respect to three objectives: maximizing SynFlow (a proxy for network trainability), minimizing the number of parameters, and minimizing Floating Point Operations (FLOPs).

To summarize the outcome of the 30 search runs, Figures 4 and 5 present the combined Pareto front approximation formed by aggregating all non-dominated solutions identified across the different seeds based on their contribution to the overall hypervolume. These visualizations reflect the diversity and trade-off coverage achieved by the hybrid NSGA-III-HC strategy in terms of SynFlow, parameter count, and FLOPs. While Figure 4 shows the 2D projections among objective pairs, Figure 5 highlights the full 3D trade-off landscape explored during the search.

Each BASSN architecture's performance was assessed across $\times 2$, $\times 3$, and $\times 4$ super-resolution tasks using the Peak Signal-to-Noise Ratio (PSNR) metric (Tables 6–8). The tables highlight the highest PSNR scores, best performances among the smallest architectures, and top outcomes from models under 500k parameters. Table 9 presents a Bayesian

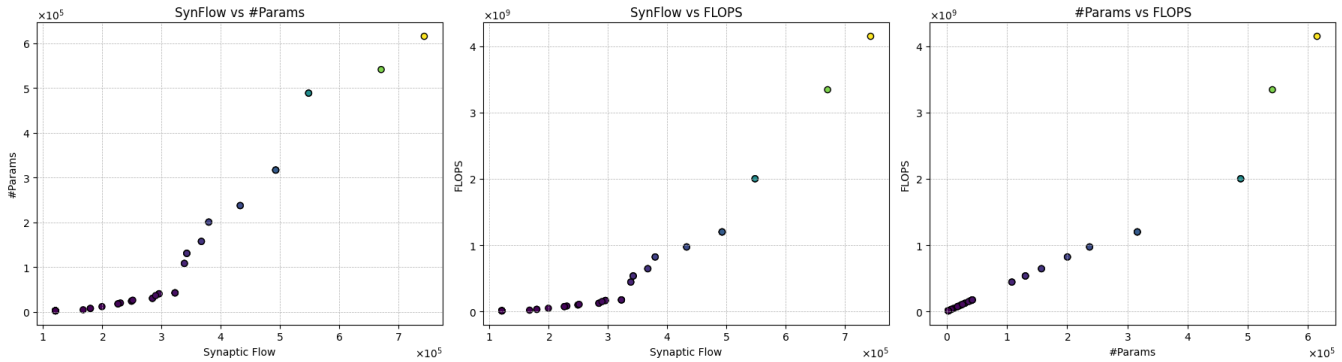


FIGURE 4: 2D projections of the global Pareto front approximation formed by aggregating all non-dominated solutions across 30 runs of NSGA-III-HC. Each plot shows pairwise relationships among SynFlow, parameter count, and FLOPs.

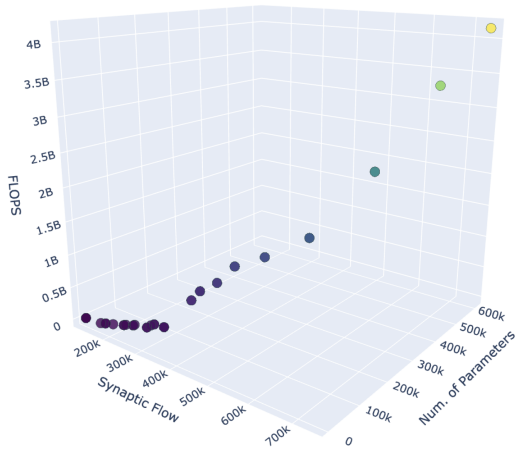


FIGURE 5: 3D visualization of the global Pareto front across all 30 NSGA-III-HC runs. Points represent non-dominated architectures in the SynFlow–Parameters–FLOPs space, capturing the structure of trade-offs explored during optimization.

analysis of pairwise model comparisons, reinforcing our findings by directly comparing each model's performance based on posterior probabilities:

- $\ll$ indicates that the row model outperforms the column model practically.
- $\gg$ signifies that the row model underperforms compared to the column model.
- A star (*) denotes high confidence (posterior probability > 0.95) in the comparison.

BASSN-A minimizes FLOPs with only 40k parameters, making it one of the most compact models. However, its small size results in lower PSNR values across all super-resolution scales, highlighting an efficiency-accuracy trade-off consistent with [43]. Thus, BASSN-A is ideal for deployment on highly resource-constrained devices like mobile platforms or embedded systems.

BASSN-B prioritizes SynFlow performance over strict resource constraints. With 615k parameters, it achieves high PSNR values—exceeding 37.8 dB on Set5 for $\times 2$ upscaling (Table 6)—making it suitable for high-performance servers or desktop applications. However, it occasionally underperforms compared to similarly sized models like ESRN-V, suggesting room for optimization.

BASSN-C adopts an extreme approach to parameter minimization, containing only 2k parameters and exhibiting the lowest FLOPs among the variants. Despite its efficiency, BASSN-C yields the lowest PSNR scores across all tasks, emphasizing the significant performance cost of such compactness. This model is best suited for applications where minimal model size is critical and some reconstruction quality reduction is acceptable, such as in simple Internet of Things (IoT) devices.

BASSN-D balances SynFlow maximization, parameter minimization, and FLOPs reduction, using 316k parameters and delivering competitive PSNR values (Tables 6–8). It approaches the performance of ESRN-V (324k) without similar resource demands, making it ideal for applications requiring both efficiency and high-quality outputs, such as real-time video streaming. Future improvements could enhance its PSNR without significantly increasing its size.

The BASSN variants demonstrate robust optimization tailored to varying resource constraints and performance requirements in SRIR tasks. Employing multi-branching strategies to define the architecture structure, our approach enables the search process to autonomously select operations and blocks while utilizing Mean Squared Error (MSE) for training. These strategies enhance model flexibility by allowing diverse computational paths and improve efficiency by optimizing resource usage. Additionally, using MSE simplifies training while effectively minimizing reconstruction errors, enabling BASSN architectures to maintain high performance without significant computational overhead.

As a result, BASSN architectures achieve competitive PSNR values across various super-resolution scales (see Tables 6–8). For instance, BASSN-B outperforms several studied models while maintaining manageable parameter counts and FLOPs, making it suitable for environments with moder-

TABLE 6: This table extends the Performance comparison of SRIR machine-crafted architectures results from [63] to the super-resolution $\times 2$ task, incorporating four additional models while maintaining the same evaluation criteria. A higher Peak-Signal-to-Noise Ratio indicates better performance. The best models, regardless of size, are in bold. Gray highlights the two smallest architectures, while darker gray with white text marks the best result per task for models under 400k parameters.

Model	**#Params (K)**	**Set5**		**Set14**		**B100**		**Urban100**	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
NAS-DIP	1 800	35.32	0.18	31.58	0.19	29.99	0.20	29.81	0.17
FALSR-A	1 021	37.82	0.08	33.52	0.09	32.12	0.07	31.93	0.08
FALSR-B	326	37.61	0.09	33.29	0.07	31.97	0.08	31.28	0.09
HNAS-A	139	37.84	0.09	33.39	0.08	32.06	0.08	31.50	0.09
HNAS-C	380	**38.11**	0.08	33.60	0.07	32.07	0.08	31.73	0.08
MoreMNAS-A	1 039	37.63	0.08	33.23	0.08	31.95	0.08	31.24	0.08
ESRN	1 039	38.04	0.05	**33.69**	0.05	32.23	0.05	**32.37**	0.06
ESRN-V	324	37.85	0.06	33.42	0.06	32.10	0.06	31.79	0.06
DLSR	322	38.04	0.05	33.67	0.05	32.21	0.05	32.26	0.05
BASSN-A	40	34.18	0.14	30.92	0.15	29.54	0.13	29.66	0.15
BASSN-B	615	37.85	0.10	33.64	0.09	32.12	0.10	32.16	0.09
BASSN-C	2	33.71	0.24	30.50	0.23	29.30	0.25	28.78	0.24
BASSN-D	316	37.35	0.08	33.62	0.09	**32.25**	0.08	32.31	0.09

TABLE 7: This table extends the Performance comparison of SRIR machine-crafted architectures results from [63] to the super-resolution $\times 3$ task, incorporating four additional models while maintaining the same evaluation criteria. A higher Peak-Signal-to-Noise Ratio indicates better performance. The best models, regardless of size, are in bold. Gray highlights the two smallest architectures, while darker gray with white text marks the best result per task for models under 400k parameters.

Model	**#Params (K)**	**Set5**		**Set14**		**B100**		**Urban100**	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
NAS-DIP	1 800	30.81	0.17	27.84	0.18	26.16	0.17	26.00	0.18
FALSR-A	1 021	32.97	0.09	29.65	0.08	28.41	0.07	28.14	0.08
FALSR-B	326	32.80	0.09	29.34	0.08	27.88	0.09	27.31	0.10
HNAS-A	139	32.35	0.09	29.32	0.08	27.65	0.08	27.17	0.09
HNAS-C	380	33.01	0.09	28.52	0.10	27.30	0.09	27.10	0.09
MoreMNAS-A	1 039	32.82	0.09	29.41	0.09	27.85	0.08	27.23	0.09
ESRN	1 039	**34.46**	0.05	**30.43**	0.05	**29.15**	0.05	**28.42**	0.05
ESRN-V	324	34.23	0.06	30.27	0.06	29.03	0.06	27.95	0.07
DLSR	322	**34.49**	0.05	30.39	0.05	29.13	0.05	28.26	0.05
BASSN-A	40	31.83	0.15	28.68	0.16	26.99	0.14	26.14	0.15
BASSN-B	615	34.14	0.11	29.82	0.09	28.55	0.10	27.46	0.09
BASSN-C	2	31.20	0.24	28.17	0.23	26.50	0.24	25.97	0.25
BASSN-D	316	33.96	0.08	29.66	0.09	28.40	0.08	27.32	0.09

ate resource availability.

While the leading models—ESRN, which utilizes handcrafted blocks proven effective in SRIR tasks [26], and DLSR, which employs a differentiable Neural Architecture Search (NAS) strategy incorporating both cell-level and network-level search spaces to enhance SRIR performance alongside a handcrafted loss function considering distortion, high-frequency reconstruction, and lightweight regularization [31]—demonstrate superior performance across all tasks, BASSN variants hold their own against these more complex strategies. The simplicity of BASSN's multi-branching and MSE-based training facilitates more straightforward implementation and adaptability, making BASSN architectures an effective and flexible choice for scenarios where resource optimization and straightforward training processes are paramount. This highlights BASSN's effective methodologies and their potential for further refinement.

The Bayesian comparative analysis in Table 9 confirms that BASSN variants effectively balance performance and resource efficiency, offering competitive trade-offs against handcrafted and NAS-based models. While some aspects still leave room for refinement, these results align with PSNR-based evaluations, demonstrating that BASSN architectures are well-suited for constrained applications and continue to evolve toward top-tier performance.

BASSN-B consistently outperforms smaller and less optimized models such as BASSN-A and BASSN-C, as evidenced by numerous $\ll^*$ entries. This confirms its ability to achieve high PSNR while maintaining reasonable parameter counts and FLOPs. Similarly, BASSN-D demonstrates com-

TABLE 8: This table extends the Performance comparison of SRIR machine-crafted architectures results from [63]to the super-resolution $\times 4$ task, incorporating four additional models while maintaining the same evaluation criteria. A higher Peak-Signal-to-Noise Ratio indicates better performance. The best models, regardless of size, are in bold. Gray highlights the two smallest architectures, while darker gray with white text marks the best result per task for models under 400k parameters.

Model	**#Params (K)**	**Set5**		**Set14**		**B100**		**Urban100**	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
NAS-DIP	1 800	26.41	0.18	24.59	0.17	22.42	0.18	22.28	0.17
FALSR-A	1 021	30.33	0.08	28.21	0.08	26.11	0.07	25.36	0.08
FALSR-B	326	28.12	0.09	26.92	0.09	23.90	0.09	23.38	0.10
HNAS-A	139	28.22	0.09	25.43	0.09	23.40	0.08	23.27	0.09
HNAS-C	380	28.44	0.09	25.16	0.10	24.48	0.09	23.91	0.10
MoreMNAS-A	1 039	28.72	0.09	25.87	0.09	23.93	0.08	23.40	0.09
ESRN	1 039	32.26	0.05	28.63	0.06	**27.62**	0.05	**26.24**	0.05
ESRN-V	324	31.99	0.06	28.49	0.06	27.50	0.05	25.87	0.06
DLSR	322	**32.33**	0.05	**28.68**	0.05	27.61	0.05	26.19	0.05
BASSN-A	40	27.74	0.14	24.42	0.15	23.49	0.13	22.52	0.14
BASSN-B	615	31.93	0.09	28.10	0.09	27.03	0.08	25.60	0.09
BASSN-C	2	27.21	0.23	24.12	0.24	23.20	0.23	22.17	0.24
BASSN-D	316	31.76	0.08	27.95	0.08	26.88	0.07	25.46	0.08

TABLE 9: Bayesian analysis comparing the performance of different models. The symbol $\ll$ indicates that the model in the row performs better than the model in the column, while $\gg$ indicates worse performance. A star (*) denotes high confidence (posterior probability > 0.95).

Models	NAS-DIP	FALSR-A	FALSR-B	HNAS-A	HNAS-C	More-MNAS-A	ESRN	ESRN-V	DLSR	BASSN -A	BASSN -B	BASSN -C	BASSN -D
NAS-DIP	–	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg$	$\gg^*$	$\ll$	$\gg^*$
FALSR-A	$\ll^*$	–	$\ll^*$	$\ll^*$	$\ll$	$\ll^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\ll^*$	$\gg^*$	$\ll^*$	$\gg^*$
FALSR-B	$\ll^*$	$\gg^*$	–	$\ll$	$\gg$	$\ll$	$\gg^*$	$\gg^*$	$\gg^*$	$\ll^*$	$\gg^*$	$\ll^*$	$\gg^*$
HNAS-A	$\ll^*$	$\gg^*$	$\gg$	–	$\gg$	$\gg$	$\gg^*$	$\gg^*$	$\gg^*$	$\ll^*$	$\gg^*$	$\ll^*$	$\gg^*$
HNAS-C	$\ll^*$	$\gg$	$\ll$	$\ll$	–	$\ll$	$\gg^*$	$\gg^*$	$\gg^*$	$\ll^*$	$\gg^*$	$\ll^*$	$\gg^*$
MoreMNAS-A	$\ll^*$	$\gg^*$	$\gg$	$\ll$	$\gg$	–	$\gg^*$	$\gg^*$	$\gg^*$	$\ll^*$	$\gg^*$	$\ll^*$	$\gg^*$
ESRN	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	–	$\ll^*$	$=$	$\ll^*$	$\gg^*$	$\ll^*$	$\gg$
ESRN-V	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\gg^*$	–	$\gg^*$	$\ll^*$	$\gg^*$	$\ll^*$	$\gg^*$
DLSR	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$==$	$\ll^*$	–	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$
BASSN-A	$\ll$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	–	$\gg^*$	$\ll^*$	$\gg^*$
BASSN-B	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	–	$\ll^*$	$\ll^*$
BASSN-C	$\gg$	$\ll^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\gg^*$	$\ll^*$	$\gg^*$	–	$\gg^*$
BASSN-D	$\ll^*$	$\ll$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\gg^*$	$\ll^*$	$\ll^*$	$\ll^*$	$\ll^*$	–

petitive performance, often surpassing models like NAS-DIP and FALSR variants, as $\ll^*$ symbols indicate. However, it does not surpass top-tier models like ESRN and DLSR, which align with PSNR-based findings and highlight its role as a balanced yet not leading solution.

Conversely, BASSN-A and BASSN-C underperform compared to larger models, with multiple $\gg^*$ symbols reflecting their lower PSNR and higher resource consumption relative to other models. This underscores the limitations of extreme compactness in achieving high-quality super-resolution. On the other hand, ESRN and DLSR remain dominant, consistently outperforming BASSN variants and other machine-crafted architectures, thereby underscoring their robustness and efficiency in balancing performance and computational demands.

The practical applications of these findings are significant. BASSN-B is ideal for high-performance environments where superior PSNR is essential and computational resources are ample, such as servers or desktop applications. BASSN-D serves well in scenarios requiring a balance between efficiency and quality, like real-time video processing or interactive graphics, where resource usage and reconstruction fidelity are essential. Meanwhile, BASSN-A and BASSN-C are best suited for deployment on highly resource-constrained devices, where minimizing memory and computational overhead is critical despite some compromises in PSNR.

VII. CONCLUSIONS AND FUTURE WORK

This work introduced a Hybrid Evolutionary Neural Architecture Search (ENAS) approach for super-resolution tasks, integrating the Balanced Architecture Search Space (BASS) to optimize neural network architectures efficiently. By combining NSGA-III's global exploration with local search refinements, we identified architectures that balance performance, model complexity, and computational efficiency.

The experimental results demonstrate that hybridization improves NAS performance, with NSGA-III-HC achieving the highest SynFlow score (6.45×10^4) and lowest FLOP

count (2.23×10^8), making it the most effective approach for trainability and computational efficiency. NSGA-III-TS produced the most compact architectures, minimizing parameters (8.18×10^4) and FLOPs (4.36×10^8), making it an attractive option for resource-constrained environments. However, NSGA-III-SA exhibited lower SynFlow scores (5.80×10^4) and higher FLOP counts (5.36×10^8), indicating that simulated annealing is less effective in this context.

From a multi-objective optimization perspective, these results confirm that higher SynFlow scores correlate with better model trainability and increased resource demands, highlighting the efficiency-accuracy trade-off in NAS. The hypervolume analysis further validated these improvements, with NSGA-III-HC achieving the highest hypervolume (1.572), demonstrating its ability to generate diverse, high-quality solutions while accelerating convergence.

Integrating BASS played a key role in enhancing NAS efficiency, allowing dynamic layer selection and fine-grained model adaptation. This feature facilitated architectural flexibility, enabling models to adjust their complexity based on task-specific constraints, making them suitable for real-world applications such as medical imaging and satellite image processing.

A. FUTURE WORK

Several avenues for future research are identified:

- **Further Characterization of BASS**: Conduct an in-depth study of the properties of the BASS search space to better understand its structure, limitations, and potential for guiding NAS towards improved architectures.
- **Expansion of Architectural Variations**: Extend BASS to incorporate a broader range of architectural designs, including novel branching patterns, dynamic depth adjustments, and specialized layer configurations.
- **Adaptive Local Search**: Develop adaptive mutation and selection mechanisms to dynamically adjust local search behavior based on search progress, improving efficiency and convergence speed.
- **Hardware-Aware NAS**: Optimize architectures considering hardware constraints, targeting TPUs, GPUs, and edge devices for energy-efficient deployment.
- **Real-World Deployment**: Validate optimized architectures in practical applications to bridge the gap between NAS-generated models and real-world usability.
- **Cross-Task Generalization**: Investigate whether the architectures discovered using BASS can generalize beyond SRIR. Given their modular and lightweight multi-branch structure, these models appear well-suited for adaptation to related image restoration tasks such as denoising, deblurring, or even video super-resolution in constrained environments. A systematic evaluation is needed to assess their generalization capacity and determine how well the architectural patterns learned by BASS transfer across diverse restoration domains.
- **Integration of Learning-Based Predictors**: Investigate the incorporation of regression surrogates and predictive models that estimate downstream performance metrics (e.g., PSNR or perceptual quality) from architecture descriptors [60]. These models have shown promise for improving ranking stability and task-specific accuracy but require careful calibration and labeled data. Their integration could enhance search quality while maintaining low computational cost.

B. FINAL REMARKS

This study demonstrated that hybridizing NSGA-III with local search accelerates convergence, enhances diversity, and improves NAS efficiency. NSGA-III-HC emerged as the most effective approach for discovering trainable, high-performance architectures, while NSGA-III-TS offered the most compact and computationally efficient solutions. Additionally, the BASS search space proved highly adaptable, highlighting its potential for guiding NAS toward architectures that meet real-world performance and resource constraints. Future research will focus on refining BASS to explore a broader range of architectural designs, improving its ability to generate competitive, efficient models for diverse applications.

ACKNOWLEDGMENT

We extend our gratitude to the members of the Advanced Artificial Intelligence group at Tecnológico de Monterrey for their valuable feedback on an earlier version of this paper. J.L. Llano Garcia acknowledges the financial support provided by the National Council of Humanities, Science, and Technology (CONAHCyT) through scholarship grant 829049. This research was funded by CONAHCyT Ciencia de Frontera 2023 under grant CF-2023-I-801. Additionally, this work benefited from Azure sponsorship credits awarded by Microsoft's AI for Good Research Lab to V. A. Sosa Hernández and R. Monroy.

REFERENCES

[1] Zhihao Wang, Jian Chen, and Steven C. H. Hoi. Deep learning for image super-resolution: A survey, 2020.

[2] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[3] Edgar Galván and Peter Mooney. Neuroevolution in deep neural networks: Current trends and future challenges, 2020.

[4] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*, 2019.

[5] Huixin Tan, Jiewei Lai, Yunbi Liu, Yuzhang Song, Jinliang Wang, Mingyang Chen, Yong Yan, Liming Zhong, Qianjin Feng, and Wei Yang. Neural architecture search for real-time quality assessment of wearable multi-lead ECG on mobile devices. *Biomedical Signal Processing and Control*, 74:103495, 2022.

[6] Yuanbiao Gou, Boyun Li, Zitao Liu, Songfan Yang, and Xi Peng. CLEARER: Multi-scale neural architecture search for image restoration. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17129–17140. Curran Associates, Inc., 2020.

[7] Philippe Preux and E-G Talbi. Towards hybrid evolutionary algorithms. *International transactions in operational research*, 6(6):557–570, 1999.

[8] Crina Grosan and Ajith Abraham. *Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[9] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, 2017.

[10] Haiwei Wu and Jiantao Zhou. IID-Net: Image inpainting detection network via neural architecture search and attention. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1172–1185, 2022.

[11] Huiyu Kuang. M-FasterSeg: An efficient semantic segmentation network based on neural architecture search, 2021.

[12] Masanori Suganuma, Mete Ozay, and Takayuki Okatani. Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4771–4780. PMLR, 10–15 Jul 2018.

[13] Gerard Jacques van Wyk and Anna Sergeevna Bosman. Evolutionary neural architecture search for image restoration. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[14] Fuqing Zhao, Haizhu Bao, Ling Wang, Xuan He, and Jonrinaldi. A hybrid cooperative differential evolution assisted by cma-es with local search mechanism. *Neural Computing and Applications*, 34(9):7173–7197, 2022.

[15] Emile Glorieux, Bo Svensson, Fredrik Danielsson, and Bengt Lennartson. Constructive cooperative coevolution for large-scale global optimisation. *Journal of Heuristics*, 23(6):449–469, 2017.

[16] Victor Robles, Jose M Pena, Pedro Larranaga, María S Pérez, and Vanessa Herves. Ga-eda: A new hybrid cooperative search evolutionary algorithm. *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, pages 187–219, 2006.

[17] X. Chu, B. Zhang, and X. Xu. Multi-objective reinforced evolution in mobile neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7840–7849, 2019.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[19] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12239–12248, 2021.

[20] Yu Xue, Xiaolong Han, Ferrante Neri, Jiafeng Qin, and Danilo Pelusi. A gradient-guided evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2024.

[21] Benjamin Patrick Evans, Harith Al-Sahaf, Bing Xue, and Mengjie Zhang. Genetic programming and gradient descent: A memetic approach to binary image classification. *arXiv preprint arXiv:1909.13030*, 2019.

[22] M. Wistuba. Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 5409–5413, 2018.

[23] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight NAS. *arXiv preprint arXiv:2101.08134*, 2021.

[24] Jovita Lukasik, Michael Moeller, and Margret Keuper. An evaluation of zero-cost proxies-from neural architecture performance prediction to model robustness. In *DAGM German Conference on Pattern Recognition*, pages 624–638. Springer, 2023.

[25] Guihong Li, Duc Hoang, Kartikeya Bhardwaj, Ming Lin, Zhangyang Wang, and Radu Marculescu. Zero-shot neural architecture search: Challenges, solutions, and opportunities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[26] Dehua Song, Chang Xu, Xu Jia, Yiyi Chen, Chunjing Xu, and Yunhe Wang. Efficient residual dense block search for image super-resolution, 2019.

[27] Yun-Chun Chen, Chen Gao, Esther Robb, and Jia-Bin Huang. Nas-dip: Learning deep image prior with neural architecture search. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 442–459, Cham, 2020. Springer International Publishing.

[28] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Hailong Ma. Multi-objective reinforced evolution in mobile neural architecture search, 2019.

[29] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search, 2020.

[30] Yong Guo, Yongsheng Luo, Zhenhao He, Jin Huang, and Jian Chen. Hierarchical neural architecture search for single image super-resolution. *IEEE Signal Processing Letters*, 27:1255–1259, 2020.

[31] Han Huang, Li Shen, Chaoyang He, Weisheng Dong, Haozhi Huang, and Guangming Shi. Lightweight image super-resolution with hierarchical and differentiable neural architecture search, 2021.

[32] Peng Liu, Ying Hong, and Yan Liu. Multi-branch deep residual network for single image super-resolution. *Algorithms*, 11(10):144, 2018.

[33] Xiang Gao, Lijuan Xu, Fan Wang, and Xiaopeng Hu. Multi-branch aware module with channel shuffle pixel-wise attention for lightweight image super-resolution. *Multimedia Syst.*, 29(1):289–303, September 2022.

[34] Guanqiang Wang, Mingsong Chen, Y.C. Lin, Xianhua Tan, Chizhou Zhang, Wenxin Yao, Baihui Gao, Kai Li, Zehao Li, and Weidong Zeng. Efficient multi-branch dynamic fusion network for super-resolution of industrial component image. *Displays*, 82:102633, 2024.

[35] Hongyang Zhang, Junru Shao, and Ruslan Salakhutdinov. Deep neural networks with multi-branch architectures are less non-convex. *arXiv preprint arXiv:1806.01845*, 2018.

[36] Chunying Liu, Xujie Wan, and Guangwei Gao. A multi-branch feature extraction residual network for lightweight image super-resolution. *Mathematics*, 12(17):2736, 2024.

[37] Xintao Lu, Ying Wang, Kai Zhang, Jingyun Liang, Chao Dong, and Chen Change Loy. Efficient super-resolution transformer (esrt). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1–10, 2022.

[38] Hongyang Zhang, Junru Shao, and Ruslan Salakhutdinov. Deep neural networks with multi-branch architectures are intrinsically less non-convex. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1099–1109. PMLR, 16–18 Apr 2019.

[39] Huiyuan Tian, Li Zhang, Shijian Li, Min Yao, and Gang Pan. Multi-depth branch network for efficient image super-resolution. *Image and Vision Computing*, 144:104949, 2024.

[40] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 38, pages 295–307, 2015.

[41] Mark Sandler et al. MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[42] Wenzhe Shi et al. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[44] Sarah L. Thomson, Gabriela Ochoa, Nadarajen Veerapen, and Krzysztof Michalak. Channel configuration for neural architecture: Insights from the search space. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 1267–1275, New York, NY, USA, 2023. Association for Computing Machinery.

[45] Hongwei Dai and Yu Yang. Gray-coded clonal selection algorithm for optimization problem. In *International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2016)*, pages 280–288. Springer, Cham, 2016.

[46] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[47] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, 2014.

[48] Mingxing Tan and Quoc Le. EfficientNetV2: Smaller models and faster training. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 10096–10106, 2021.

[49] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[50] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[51] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.
[52] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(1):1997–2017, jan 2019.
[53] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6105–6114, 2022.
[54] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.
[55] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*, 2019.
[56] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 544–560, 2020.
[57] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
[58] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.
[59] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.
[60] Sergio Sarmiento-Rosales, Jesús Leopoldo Llano García, Jesús Guillermo Falcón-Cardona, Raúl Monroy, Manuel Iván Casillas del Llano, and Víctor Adrián Sosa-Hernández. Surrogate modeling for efficient evolutionary multi-objective neural architecture search in super resolution image restoration. In *Proceedings of the 16th International Joint Conference on Computational Intelligence (IJCCI)*, pages 242–249, 2024.
[61] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.
[62] Ngoc Hoang Luong, Quan Minh Phan, An Vo, Tan Ngoc Pham, and Dzung Tri Bui. Lightweight multi-objective evolutionary neural architecture search with low-cost proxy metrics. *Information Sciences*, 655:119856, 2024.
[63] Jesús Leopoldo Llano García, Raúl Monroy, and Víctor Adrián Sosa Hernández. An experimental protocol for neural architecture search in super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 4139–4146, October 2023.
[64] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1122–1131, 2017.
[65] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *23rd British Machine Vision Conference (BMVC)*, pages 135.1–135.10. BMVA press, 2012.
[66] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In Jean-Daniel Boissonnat, Patrick Chenin, Albert Cohen, Christian Gout, Tom Lyche, Marie-Laurence Mazure, and Larry Schumaker, editors, *Curves and Surfaces*, pages 711–730, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
[67] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.
[68] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.

JESÚS LEOPOLDO LLANO GARCÍA is a Ph.D. candidate in Computer Science at Tecnológico de Monterrey, specializing in optimization techniques for Machine Learning, AutoML, and Computer Vision. He is currently an adjunct professor in Computing at Tecnológico de Monterrey. His research focuses on Neural Architecture Search (NAS) for Super-Resolution Image Restoration, employing multi-objective evolutionary optimization and heuristic-based search. He has co-authored publications in venues such as Neurocomputing, ICCV, and Swarm and Evolutionary Computation. He holds an M.Sc. in Computer Science from Tecnológico de Monterrey, where he worked on constraint-based evolutionary optimization.

RAÚL MONROY received a Ph.D. degree in artificial intelligence (AI) from Edinburgh University, in 1998, under the supervision of Alan Bundy. He is currently a Professor in AI with the Tecnológico de Monterrey, a member of CONAHCYT's Mexican Research System, currently rank three (top), a member of the Mexican Academy of Sciences, and a Founding Member of the Mexican Academy of Computing. He is the author of more than 120 publications. His research interests include the design and development of novel machine learning models, which he often applies in the domain of cyber security.

VÍCTOR ADRIÁN SOSA HERNÁNDEZ received a B.Sc. degree in computer systems from the Instituto Politécnico Nacional, Mexico City, Mexico, in 2011, and an M.Sc. degree in computer science and a Ph.D. degree from CINVESTAV-IPN, Mexico City, in 2013 and 2017, respectively. His current research interests include machine learning, deep learning, neuroevolution, and multi-objective optimization through global and local searchers. He is the head of the Graduate Programme in Computing Computer Science and a full-time professor in the computer science department at the Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México (ITESM-CEM).

KALYANMOY DEB (F'11) received the bachelor's degree in mechanical engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1985, and the master's and Ph.D. degrees from the University of Alabama, Tuscaloosa, AL, USA, in 1989 and 1991, respectively.,He is the Koenig Endowed Chair Professor with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA. He is largely known for his seminal research in evolutionary multicriterion optimization. He has published over 544 international journal and conference research papers. His current research interests include evolutionary optimization and its application in design, modeling, AI, and machine learning.,Dr. Deb is a recipient of the IEEE CIS EC Pioneer Award in 2018, the Lifetime Achievement Award from Clarivate Analytics in 2017, the Infosys Prize in 2012, and the Edgeworth-Pareto Award in 2008. He is a Fellow of ASME.