Surrogate Modeling for Efficient Evolutionary Multi-Objective Neural Architecture Search in Super Resolution Image Restoration

Sergio Sarmiento-Rosales, Jesús Leopoldo Llano García, Jesús Guillermo Falcón-Cardona, Raúl Monroy, Manuel Iván Casillas del Llano and Víctor Adrián Sosa Hernández

Tecnologico de Monterrey, School of Engineering and Sciences, Mexico
{*a01801059, a01748867, jfalcon, raulm, manuel_casillas,vsosa*}*@tec.mx*

Keywords: Neural Architecture Search, Multi-Objective Optimization, Surrogate Models, Super-Resolution.

Abstract: Fully training each candidate architecture generated during the Neural Architecture Search (NAS) process is computationally expensive. To overcome this issue, surrogate models approximate the performance of a Deep Neural Network (DNN), considerably reducing the computational cost and, thus, democratizing the utilization of NAS techniques. This paper proposes an XGBoost-based surrogate model to predict the Peak-Signal-to-Noise Ratio (PSNR) of DNNs for Super-Resolution Image Restoration (SRIR) tasks. In addition to maximizing PSNR, we also focus on minimizing the number of learnable parameters and the total number of floating-point operations. We use the Non-dominated Sorting Genetic Algorithm III (NSGA-III) to tackle this three-objective optimization NAS problem. Our experimental results indicate that NSGA-III using our XGBoost-based surrogate model is significantly faster than using full or partial training of the candidate architectures. Moreover, some selected architectures are comparable in quality to those found using partial training. Consequently, our XGBoost-based surrogate model offers a promising approach to accelerate the automatic design of architectures for SRIR, particularly in resource-constrained environments, decreasing computing time.

1 INTRODUCTION

Neural Architecture Search (NAS) automates the design of neural networks, discovering near-optimal architectures for specific tasks (Wistuba et al., 2019). However, its high computational demands often require access to powerful hardware and large architectural datasets (Xie et al., 2023). This constraint limits the broader application of NAS (Elsken et al., 2018).

To address this challenge, surrogate-assisted NAS has emerged as a promising solution. This approach uses surrogate models to estimate the performance of neural architectures without full training (Hutter et al., 2011). By minimizing reliance on costly training procedures, surrogate-assisted NAS makes the search process more accessible (Kandasamy et al., 2018).

Recent advances in performance estimation within NAS have introduced various innovative methodologies, such as Gaussian processes, graph neural networks, and recurrent neural networks (White et al., 2021; Luo et al., 2018; Real et al., 2019). These techniques work in reducing computational requirements (Falkner et al., 2018).

The integration of surrogate models into NAS presents significant potential for advancing applications in areas such as Super-Resolution (SR) (Ahn and Cho, 2021; Huang et al., 2022). These tasks are challenging due to their ill-posed nature, making it difficult to determine the characteristics of a network that captures sufficient information during training. Moreover, as dense prediction problems, they require considerable computational resources, escalating the computational cost of the NAS process. However, models capable of performing SR have diverse applications in fields such as medical imaging, biometric recognition, surveillance, and remote sensing (Wistuba et al., 2019).

Recent NAS research focuses on optimizing evaluation within the search pipeline, enhancing efficiency under limited data and constrained conditions (Ahn and Cho, 2021). Surrogate models improve NAS applicability, making it more versatile and efficient (Lu et al., 2022; Elsken et al., 2018). Despite their success in other areas (Sun et al., 2019; Xue et al., 2024; White et al., 2021), the use of surrogate models in image super-resolution remains limited. This gap underscores the need to develop and validate surrogate approaches for rapid and accurate

Sarmiento-Rosales, S., Llano García, J., Falcón-Cardona, J., Monroy, R., Casillas del Llano, M. and Sosa-Hernández, V.
Surrogate Modeling for Efficient Evolutionary Multi-Objective Neural Architecture Search in Super Resolution Image Restoration.
DOI: 10.5220/0012949000003837

In *Proceedings of the 16th International Joint Conference on Computational Intelligence (IJCCI 2024)*, pages 242-249
ISBN: 978-989-758-721-4; ISSN: 2184-3236

performance evaluation in SR tasks.

Given the success of surrogate assisted NAS (Xie et al., 2023), we here investigate its use in SR, but using an Evolutionary Multi-Objective Algorithm (EMOA) that makes use of a regressor model for estimating the performance of candidate neural networks. For our EMOA, we used NSGA-III for solving an optimization problem that includes several conflicting objectives, including maximizing the Peak-Signal-to-Noise Ratio (PSNR), minimizing the number of learnable parameters, and minimizing the total floating-point operations (FLOPs). For the regressor, we have trained XGBoost on a reduced dataset, created with Latin hypercube sampling.

Our experiments demonstrate the significant difference that surrogate-assisted NAS has in reducing computational overhead and accelerating evaluation speed, highlighted by a decrease in evaluation time from approximately 30 GPU hours per generation using partial training to just 3 CPU hours for 500 generations.

Section 2 provides the fundamental background, Section 3 describes our surrogate model and methodology, Section 4 highlights our results and Section 5 presents our conclusions and future work.

2 RELATED WORKS

This section defines SR task, explores NAS architecture evaluation strategies, and reviews surrogate models for NAS.

2.1 Super-Resolution Image Restoration

SR aims to obtain a high-resolution image from a degraded low-resolution sample. SR systems compensate for inadequate image acquisition conditions. Mathematically, SR is modeled as $\mathbf{y} = H\mathbf{x} + e$, where $\mathbf{x}$ is the non-degraded image, $\mathbf{y}$ is the low-resolution observation, H is a degradation matrix, and e is additive noise. Solving this problem efficiently requires models that can extract high-resolution information from low-resolution samples. Deep learning offers efficient solutions by mapping low-resolution samples to higher-resolution reconstructions, and NAS can be helpful in discovering architectures that enhance SR models.

2.2 Architecture Evaluation Strategies

Optimal architecture search requires exploration guidance, typically through performance evaluation across multiple objectives. However, training for this evaluation is often challenging and computationally expensive. (Real et al., 2017; Real et al., 2019; Zoph et al., 2018).The traditional method of training and evaluating each model on validation data is impractical due to its high computational demands. To address this, various techniques have been developed to estimate model performance more efficiently.

Some early and recent works in NAS have explored using lower fidelity estimates to approximate model performance. These estimates are typically obtained by training models for shorter duration (Chu et al., 2020; Zoph et al., 2018) or on partial datasets (Chen et al., 2020b). While these approaches can provide a quick indication of a model's potential performance, they do not always guarantee accurate rankings compared to full training. Then, these methods have become less popular in contemporary NAS research but remain useful for reducing time costs.

Weight sharing and inheritance are techniques used to expedite the estimation of model performance by reducing or eliminating the need for training from scratch (Zhu et al., 2019; Chen et al., 2020a; Wei et al., 2016). Network morphism, introduced by (Wei et al., 2016), simplifies the process of weight inheritance among models, leading to more effective networks and reducing the total training time needed for NAS (Zhu et al., 2019; Chen et al., 2020a).

Performance predictors, such as Gaussian processes and regression tree models, offer an indirect estimation of model performance, by evaluating candidate architectures without full training (Xie et al., 2023). In surrogate-assisted NAS, these models predict architecture performance based on network encodings and performance metrics, significantly reducing computational costs. In NAS, surrogate models enhance both efficiency and effectiveness by replacing computationally intensive processes with faster, cost-effective alternatives. Their use has led to innovative approaches and methodologies, driving advancements in NAS research and enabling more efficient exploration of architectural spaces.

2.3 Surrogate-Assisted NAS

Several notable approaches in NAS have leveraged surrogate models to improve efficiency and effectiveness. In (Hutter et al., 2011) introduced sequential model-based optimization (SMBO), employing Gaussian processes to predict algorithm configuration performance. Within (Kandasamy et al., 2018) developed Neural Architecture Search with Bayesian Optimization and Optimal Transport (NASBOT), using Gaussian processes for performance modeling and optimizing the architecture search space. In (White

et al., 2021) combined neural networks and Gaussian processes in Bayesian Optimization with Neural Architectures for NAS (BANANAS) to enhance NAS efficiency through architecture space representation and uncertainty management. Authors in (Falkner et al., 2018) integrated Bayesian Optimization with HyperBand (BOHB) for robust and efficient NAS evaluation. Luo et al. (Luo et al., 2018) proposed Neural Architecture Optimization (NAO), using graph neural networks and recurrent neural networks to map architecture topological structures into higher dimensions and optimize architectures in a continuous latent space, respectively. These works demonstrate a diverse range of surrogate modeling techniques that have significantly enhanced the efficiency and effectiveness of NAS.

In domains with limited published NAS works, such as SR, integrating surrogate models presents an opportunity. This focus on surrogate models is evident in the work of (Lu et al., 2022), Surrogate-assisted Multi-objective Neural Architecture Search for Real-time Semantic Segmentation (MoSegNAS), which applies sparse coding, classification loss, and synthetic data to a multi-layer perceptron model as a way to predict the segmentation performance of neural architectures. Additionally, (Huang et al., 2022) demonstrate innovative approaches using unsupervised learning and differentiable search levels to enhance efficiency and performance in SR tasks. In (Ahn and Cho, 2021) showcases a simplified and fast representation of the original neural architecture search system, trained with a limited dataset to predict new architecture performance, avoiding exhaustive evaluation of each one. These advancements align to expand knowledge and tools in this dynamic field of research.

3 METHODOLOGY

Our methodology comprises three integral components: a search space tailored for DNNs in SR tasks, an evolutionary search strategy, and a performance evaluation mechanism. Although our focus is the last two components, we assume there exists a suitable search space.[1]

3.1 Optimizing Architectures

NSGA-III serves as the cornerstone of our approach to NAS for SR by considering multiple critical objectives in architecture design. Our goal is to craft networks that achieve the highest performance within our search space while minimizing complexity and memory requirements. To achieve this, we formalize three objectives. Let A be an architectural search space and $\alpha \in A$ an architecture with optimized learning parameters or weights $\mathbf{w}^*(\alpha)$ achieving the minimal loss. We aim to find an α that solves the following multi-objective problem, given an SR dataset $\mathcal{D}$ split into D_{train} and D_{valid}:

$$\mathbf{F}(\alpha, \mathbf{w}^*(\alpha)) = (f_1, f_2, f_3)$$

where:

$$f_1 = \max\left(\text{PSNR}_{\mathcal{D}_{\text{valid}}}\right) = 20 \cdot \log_{10}\left(\frac{L-1}{\sqrt{\text{MSE}}}\right)$$

$$\text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(O_{i,j} - D_{i,j})^2$$

The objective f_1 maximizes the PSNR between a super-resolved image and its high-resolution counterpart, where L represents the maximum possible intensity of RGB pixels (0 to 255), and MSE measures the average squared difference between the original and super-resolved images. In the MSE equation, O denotes the original high-resolution image pixels, and D represents the SR image produced by $\alpha, \mathbf{w}^*(\alpha)$. m and n are the dimensions of the image.

The second and third objectives are defined as:

$$f_2 = \min(\text{FLOPs}), \quad f_3 = \min(\text{Parameters}),$$

where f_2 represents the minimization of the total number of floating-point operations required for prediction, and f_3 represents the minimization of the number of learnable parameters within the model.

3.2 Accelerating Evaluation

Evaluating architectural designs in SR tasks is computationally intensive due to the high volume of predictions involved. To address this, we replace direct PSNR calculation from objective f_1 with a machine learning surrogate-based approach, employing a regression technique to approximate the PSNR of untrained architectures. This surrogate model leverage 28 features derived from architectural configurations. These features encompass various architectural aspects, including operation types, kernel sizes, repetition patterns, and channel numbers. The surrogate model shall use these features to predict how architectural variations influence final performance, utilizing a dataset of architecture-performance pairs.

To generate the dataset for surrogate training, we trained a diverse set of 541 neural architectures for

[1]For the sake of reproducible research, it is available upon request by sending an e-mail to raulm@tec.mx

Table 1: The different Regression Models tested to find the best surrogate alternative and Hyperparameters for tuning.

Model	**Hyperparameters**
Linear Regression	'fit_intercept': [True, False]
Ridge Regression	'alpha': [0.1, 1, 10]
Lasso Regression	'alpha': [0.1, 1, 10]
ElasticNet Regression	'alpha': [0.1, 1, 10], 'l1_ratio': [0.2, 0.5, 0.8]
Decision Tree Regression	'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]
Random Forest Regression	'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]
XGboost Regression	'n_estimators': [100, 200, 300], 'max_depth': [3, 4, 5], 'learning_rate': [0.01, 0.1, 0.2], 'subsample': [0.8, 0.9, 1.0]
AdaBoost Regression	'n_estimators': [50, 100, 200], 'learning_rate': [0.5, 1.0, 1.5]
Extra Trees Regression	'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]
KNN Regression	'n_neighbors': [1, 3, 5, 7, 9, 11], 'weights': ['uniform', 'distance'], 'p': [1, 2, 3, 4], 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
Support Vector Regression	'C': [0.1, 1, 10], 'gamma': ['scale', 'auto'], 'kernel': ['linear', 'rbf', 'poly']

Table 2: The table shows the time taken, average MSE, and average R^2 for each algorithm. The best on each column are highlighted in **bold text**. The one that achieved a good balance between time and performance is highlighted with *italics*.

Algorithm	Time (sec)	Avg MSE	Avg R^2
Linear Regression	0.11	18.5383	0.4053
Ridge Regression	0.16	18.5229	0.4034
Lasso Regression	**0.10**	16.9799	0.2130
ElasticNet Regression	0.15	16.9799	0.2130
Decision Tree Regression	1.55	21.1536	**0.7481**
Random Forest Regression	413.10	17.1595	0.2937
XGBoost Regression	170.99	**16.9618**	0.2327
AdaBoost Regression	9.25	19.3816	0.6329
Extra Trees Regression	255.00	17.0304	0.2383
KNN Regression	23.05	17.4596	0.3468
SVM Regression	22.46	17.3736	0.1090

SR, strategically sampled using a Latin hypercube from our search space. These architectures are uniformly distributed, ensuring the representation of different regions of the space. Training was performed on a high-resolution image subset of the DIV2K dataset, consisting of 522,000 training patches from 800 training instances and 66,700 validation patches from 100 validation instances. Due to time constraints, only 541 architectures were evaluated.

We tested 11 different regression algorithms in predicting the PSNR of untrained models, experimenting with various hyper-parameters to identify the best configuration for each[2]. The efficacy of our surrogate model was validated by comparing its predictions with the outcomes of partially trained architectures. This validation method was chosen to accommodate time constraints, as fully training the architectures would be exceedingly time-consuming and computationally demanding. By comparing the surrogate model's predictions with P.T. results, we gain valuable insights into its performance and accuracy without incurring exhaustive computational costs associated with full training.

[2]The code and dataset required to replicate the training results of each model can be found at: https://github.com/SergioSarmientoRosales/Training-Regression_Models

4 EXPERIMENTAL PROTOCOL AND RESULTS

To thoroughly validate our Surrogate-assisted ENAS pipeline, we conducted a series of experiments. Our goal was to compare its performance with partially trained, fully trained, and standard NAS approaches, taking into account the constraints of time, and computational and environmental resources.

4.1 Regression Model Comparison

To determine the most suitable regression model for our application, we conducted a comprehensive grid search using the hyperparameters delineated in Table 1. This systematic exploration was undertaken to ascertain a good configuration for each model. We selected the best-performing configurations for each model and subjected them to a rigorous 10-fold cross-validation separating the dataset into 10 pieces with instances allocated randomly. The results, as summarized in Table 2 and endorse XGBoost as the most effective model across all metrics considered.

Among the 11 regressors, XGBoost exhibited a training time of 170.99 seconds, an average MSE of 16.9618, and an average R^2 of 0.2327. While XGBoost did not attain the highest accuracy in isolation, its balance between computational efficiency and performance makes it a surrogate candidate for our application. Future research endeavors could potentially delve into even more refinement of hyperparameters or the exploration of ensemble techniques that further augment the model's performance.

4.2 Surrogate vs. Partial Training

To validate the efficacy of our XGBoost-based surrogate model as a NAS evaluation protocol, we compare it against partial training (P.T.). This approach was chosen to accommodate time constraints, as fully training the architectures would be exceedingly time-consuming and computationally demanding. By comparing the NAS results using surrogate model with the

NAS results using P.T., we gain valuable insights into the performance and accuracy of our approach.

In our initial NAS experiment, we utilized NSGA-III to optimize a population of 20 individuals over 500 generations, resulting in 10,000 evaluation functions. Then, we tested the two evaluation approaches under the same conditions. The first approach focused on P.T. to evaluate architecture performance. This involved training with the following parameters: epochs = 5, learning rate = 3e-4, epsilon = 1e-7, and weight decay = 1e-8, based on established standards (Huang et al., 2022). Limiting the training epochs to five reduced the training time, aiming for a more agile and efficient search. Despite this reduction, the need for numerous training processes still demanded significant computational resources.

The second approach leverages an XGBoost-based surrogate model to predict an architecture's PSNR from its design features, dramatically speeding up the NAS process compared to P.T.. This optimization reduced evaluation time from 30 GPU hours per generation to just 3 CPU hours for 500 generations on an Intel® Xeon® Gold CPU, demonstrating the efficiency of surrogate-assisted techniques in evolutionary algorithms.

To ensure robustness, 30 runs with different seeds were performed, showing the surrogate-assisted approach's superiority over P.T., which only evolved 9 full generations in its initial run. The top 20 models from the architecture-performance dataset were selected using Pareto dominance for a fair baseline, providing a benchmark for comparing algorithm performance.

Our primary quality indicator is the hypervolume (hv), which gauges the proximity and distribution of solutions relative to the Pareto frontier. Tables 3 and 4 summarize the hv, average normalized approximated PSNR, average parameters, and average FLOPs for each of the different seeds, P.T. and our baseline.

Among all the seeds, seed 25 achieved the best balance between parameter count, FLOPs, and PSNR. Specifically, seed 25 reached a normalized PSNR of 0.1037 with a standard deviation of 0.1200, maintaining a moderate parameter count of 2.95e+5 and a FLOPs count of 1.21e+9. The hv indicator is vital for assessing both the convergence and diversity of solutions. Seed 25 achieved a hv of 1.2088, significantly higher than the baseline's 1.0949, indicating that the surrogate-assisted approach not only better converged toward the Pareto front but also preserved a diverse set of high-quality solutions.

This approach also drastically reduced computational demands, cutting down from 30 GPU-hours per generation in the P.T. approach to just 3 CPU hours for 500 generations. This underscores the efficiency and practicality of surrogate models in large-scale NAS tasks.

To validate the surrogate model's accuracy, we partially trained the last generation using the median seed, achieving a real PSNR of 33.25 compared to the predicted 34.7, with a low error rate of approximately 4.27%, confirming the model's predictive accuracy.

These findings emphasize the effectiveness of surrogate-assisted NAS in enhancing the efficiency of evolutionary algorithms, achieving high-quality solutions with significantly reduced computational resources.

Table 3: Summary for the 30 seeds of the NAS algorithm going for 500 generations.

Quality indicator	Mean	SD	Minimum	Maximum
Hypervolume	1.1400	0.0400	1.0195	1.2088
Normalized PSNR	0.1720	0.0790	0.0586	0.5215
Parameters	119k	102k	3k	732k
FLOPs	556M	826M	277M	3020M

Table 4: Quality indicators for the best, worst, median seeds, P.T., and Baseline.

Quality indicator	Best Seed	Worst Seed	Median Seed	P.T.	Baseline
Hypervolume	1.2088	1.0195	1.1977	1.0240	1.0949
Normalized PSNR	0.0551	0.5215	0.0586	0.4932	0.2804
Parameters	35k	732k	360k	78k	335k
FLOPs	146M	3020M	1470M	322M	1380M

4.3 Additional Generations

To assess performance with extended computing time, we tested key populations from 30 experiments across 1250 generations, totaling 25,000 function evaluations, while keeping resource consumption low and striving for optimal results. This extended analysis examines the algorithm's long-term behavior and ability to approach the global optimum, thoroughly evaluating its efficiency and effectiveness. The convergence plots in Figures 1, and 2 display the hypervolume, inverted normalized PSNR, learnable parameters, and FLOPs, illustrating these findings.

Analyzing the aggregated results from the 30 seeds enables a more accurate evaluation of the proposed method's effectiveness and facilitates comparison with other approaches. This methodology is critical to ensuring the study's conclusions are valid and broadly applicable.

The results indicate that approximately 800 generations are sufficient to achieve hv stability. No significant improvement is observed in the different objectives beyond this point, and some objectives may even worsen, making the evaluation of new architectures unnecessary. This conclusion is corroborated by

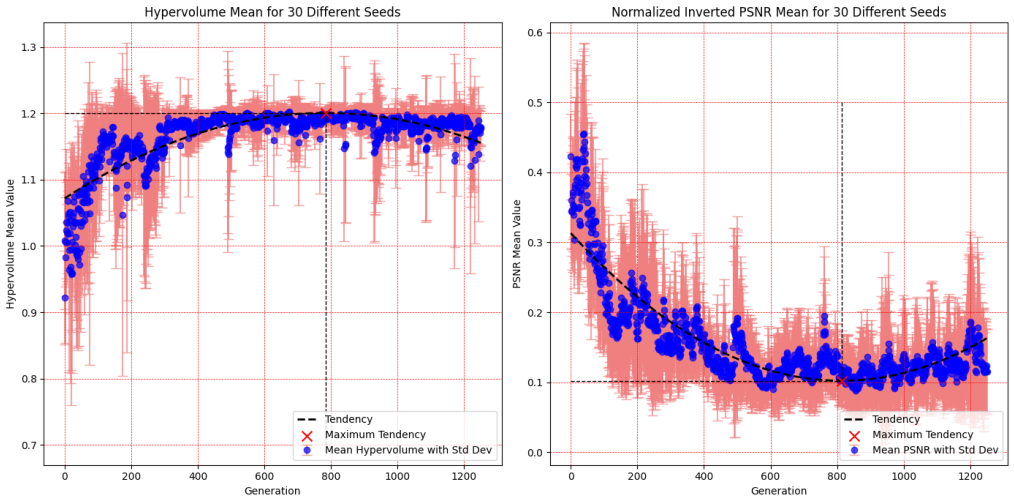

Figure 1: The figure shows hypervolume improvement with surrogate-assisted NSGA-III, including average (blue dots), standard deviation (red lines), central tendency (dashed line), and maximum average (red 'x'). Average predicted PSNR and its statistics are also depicted.

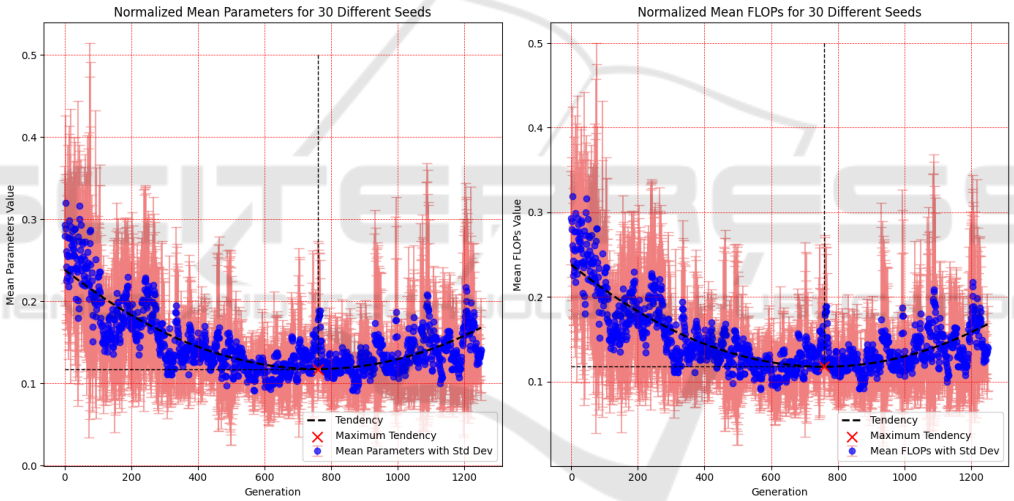

Figure 2: The figure shows progress in discovering SR deep neural networks with surrogate-assisted NSGA-III. It includes average parameters (blue dots), standard deviation (red lines), central tendency (dashed line), and maximum average (red 'x'). Average FLOPs and their statistics are also shown.

Table 5: Summary for the 30 seeds of the NAS algorithm going for 1250 generations.

Quality indicator	Mean	SD	Minimum	Maximum
Hypervolume	1.1972	0.0378	1.1149	1.2059
Normalized PSNR	0.1108	0.0957	0.0628	0.2370
Parameters	1172k	572k	253k	1840k
FLOPs	1470M	1330M	28M	2800M

comparing the hypervolumes of random seeds. Table 5 shows the behavior of the 30 seeds, and it can be concluded that, although in some cases there may be an improvement, this is not generalizable and in certain cases the results may worsen.

4.4 Tradeoff Analysis

Based on the analysis of our architectures compared to the baseline summarized in Table 6, our models exhibit a nuanced performance profile. The average PSNR (34.2097) of our architectures is slightly higher than the baseline's average PSNR (34.3201). This suggests that our models are more effective at maintaining fine textures and other subtle features in images, resulting in better overall quality in these aspects compared to the baseline.

In terms of model complexity, our architectures generally feature a higher number of parameters, indicating the potential for more detailed representations.

Table 6: Comparison of Final Architectures.

Architecture	PSNR (Predicted)	PSNR (Real)	Model Size	FLOPs
Surrogate 1	34.0966	34.8696	4,592	19,562,496
Surrogate 2	34.4465	34.9078	3,207,264	13,132,562,432
Surrogate 3	33.8144	34.2638	2,272	10,321,920
Surrogate 4	34.4816	35.0541	35,712	146,866,176
Surrogate Avg	34.2097	34.7738	812,460	3,327,328,256
Baseline 1	-	34.1874	4,992	21,268,480
Baseline 2	-	34.9011	1,380,192	5,699,436,544
Baseline 3	-	34.2617	3,904	17,009,664
Baseline 4	-	33.9305	22,784	95,478,784
Baseline Avg.	-	34.3201	352,968	1,458,298,368

This additional complexity can be related to our models ability to capture intricate features in the data.

However, this increased complexity also leads to a higher computational cost, as more parameters require more computation during training and inference. Similarly, our architectures tend to have higher FLOPs, indicating an increased computational demand.

Despite these complexities, the number of parameters and FLOPs in our models remains relatively low, with several model examples having at most thousands of parameters rather than millions. This suggests that our models remain computationally efficient given the balance in objectives used during the search. This, also means that our models have a limitation to the maximum PSNR they can achieve due to this diminishment of parameters and operations.

In conclusion, our architectures demonstrate competitive performance and even surpass that of the baseline in terms of PSNR. However, they come with a tradeoff of higher model complexity and computational cost. Further analysis could focus on understanding the specific architectural differences that lead to these tradeoffs, as well as their implications for practical deployment and scalability.

5 CONCLUSIONS AND FUTURE WORK

This research presents a novel NAS approach that combines an EMOA with a regressor model for predicting performance. The methodology effectively manages conflicting optimization objectives, producing architectures with thousands of parameters that demonstrate competitive results on the DIV2K validation dataset. This advancement is significant as it allows for deploying DNNs across various hardware platforms, thereby reducing computational and energy costs.

Extensive experimentation identified XGBoost as the most effective regression algorithm due to its performance in terms of MSE and training time. This has led to a substantial reduction in computational needs, cutting the training time from approximately 30 GPU-hours per generation to about 3 CPU-hours for 500 generations of a full search.

A key strength of this framework is its use of surrogate models, which predict architecture performance without exhaustive evaluations. However, the quality of these surrogate models is currently suboptimal, potentially impacting prediction accuracy. The existing approach involves sampling and training a subset of architectures, which is more feasible than evaluating each architecture individually. Future improvements in dataset sampling are expected to enhance the understanding of the relationship between architectural features and performance.

The P.T. approach used for 30 seeds in the surrogate model could either be time-consuming or resource-intensive. While the focus has been on SR, the NAS pipeline has potential for broader applications if initial architectural data is sufficient and the framework can be adapted for different tasks.

Future research should address the limitations of current surrogate models and explore enhancements. Incorporating online learning to continuously update the model with new data and transfer learning to leverage knowledge from related tasks could improve the surrogate model's accuracy and adaptability. This study lays the groundwork for more efficient neural architecture design, with future work aimed at refining surrogate models and exploring additional learning paradigms to advance the field of machine learning.

ACKNOWLEDGEMENTS

We thank the members of the Advanced Artificial Intelligence group at Tecnologico de Monterrey for providing feedback on this paper. The first author want to thank the National Council of Humanities, Science, and Technology (CONAHCyT) for the financial support given under scholarship grants 850203 and 829049, respectively. This research project is supported by CONAHCyT Ciencia de Frontera 2023 under grant CF-2023-I-801. V. A. Sosa Hernández and Raul Monroy thank the Microsoft Azure initiative called Microsoft's AI for Cultural Heritage program for providing cloud resources that greatly enhanced our development process.

REFERENCES

Ahn, J. Y. and Cho, N. I. (2021). Neural architecture search for image super-resolution using densely constructed search space: Deconas. In *2020 25th Inter-*

national Conference on Pattern Recognition (ICPR), pages 4829–4836. IEEE.

Chen, Y., Pan, T., He, C., and Cheng, R. (2020a). *Efficient Evolutionary Deep Neural Architecture Search (NAS) by Noisy Network Morphism Mutation*, pages 497–508.

Chen, Y.-C., Gao, C., Robb, E., and Huang, J.-B. (2020b). Nas-dip: Learning deep image prior with neural architecture search. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 442–459, Cham. Springer International Publishing.

Chu, X., Zhang, B., Ma, H., Xu, R., and Li, Q. (2020). Fast, accurate and lightweight super-resolution with neural architecture search.

Elsken, T., Metzen, J. H., and Hutter, F. (2018). Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*.

Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pages 1437–1446. PMLR.

Huang, H., Shen, L., He, C., Dong, W., and Liu, W. (2022). Differentiable neural architecture search for extremely lightweight image super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 507–523. Springer.

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. P. (2018). Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31.

Lu, Z., Cheng, R., Huang, S., Zhang, H., Qiu, C., and Yang, F. (2022). Surrogate-assisted multiobjective neural architecture search for real-time semantic segmentation. *IEEE Transactions on Artificial Intelligence*.

Luo, R., Tian, F., Qin, T., Chen, E., and Liu, T.-Y. (2018). Neural architecture optimization. *Advances in neural information processing systems*, 31.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. (2017). Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR.

Sun, Y., Wang, H., Xue, B., Jin, Y., Yen, G. G., and Zhang, M. (2019). Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation*, 24(2):350–364.

Wei, T., Wang, C., Rui, Y., and Chen, C. W. (2016). Network morphism. In *International conference on machine learning*, pages 564–572. PMLR.

White, C., Neiswanger, W., and Savani, Y. (2021). Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10293–10301.

Wistuba, M., Rawat, A., and Pedapati, T. (2019). A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*.

Xie, X., Song, X., Lv, Z., Yen, G. G., Ding, W., and Sun, Y. (2023). Efficient evaluation methods for neural architecture search: A survey. *arXiv preprint arXiv:2301.05919*.

Xue, Y., Zhang, Z., and Neri, F. (2024). Similarity surrogate-assisted evolutionary neural architecture search with dual encoding strategy. *Electronic research archive*, 32(2):1017–1043.

Zhu, H., An, Z., Yang, C., Xu, K., Zhao, E., and Xu, Y. (2019). Eena: efficient evolution of neural architecture. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.