# Dual-Level Cross-Modality Neural Architecture Search for Guided Image Super-Resolution

Zhiwei Zhong, Xianming Liu, *Member, IEEE*, Junjun Jiang, *Senior Member, IEEE*, Debin Zhao, *Member, IEEE*, and Shiqi Wang, *Senior Member, IEEE*

***Abstract*—Guided image super-resolution (GISR) aims to reconstruct a high-resolution (HR) target image from its low-resolution (LR) counterpart with the guidance of a HR image from another modality. Existing learning-based methods typically employ symmetric two-stream networks to extract features from both the guidance and target images, and then fuse these features at either an early or late stage through manually designed modules to facilitate joint inference. Despite significant performance, these methods still face several issues: i) the symmetric architectures treat images from different modalities equally, which may overlook the inherent differences between them; ii) lower-level features contain detailed information while higher-level features capture semantic structures. However, determining which layers should be fused and which fusion operations should be selected remain unresolved; iii) most methods achieve performance gains at the cost of increased computational complexity, so balancing the trade-off between computational complexity and model performance remains a critical issue. To address these issues, we propose a Dual-level Cross-modality Neural Architecture Search (DCNAS) framework to automatically design efficient GISR models. Specifically, we propose a dual-level search space that enables the NAS algorithm to identify effective architectures and optimal fusion strategies. Moreover, we propose a supernet training strategy that employs a pairwise ranking loss trained performance predictor to guide the supernet training process. To the best of our knowledge, this is the first attempt to introduce the NAS algorithm into GISR tasks. Extensive experiments demonstrate that the discovered model family, DCNAS-Tiny and DCNAS, achieve significant improvements on several GISR tasks, including guided depth map super-resolution, guided saliency map super-resolution, guided thermal image super-resolution, and pan-sharpening. Furthermore, we analyze the architectures searched by our method and provide some new insights for future research.**

***Index Terms*—Guided image super-resolution, neural architecture search, ranking loss.**

Received 12 September 2024; revised 9 May 2025; accepted 1 June 2025. Date of publication 10 June 2025; date of current version 6 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 92270116, and in part by the RGC General Research Fund under Grant 11200323 and Grant 11203220. Recommended for acceptance by Y. Fu. *(Corresponding author: Xianming Liu.)*

Zhiwei Zhong is with the Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China, and also with the Department of Computer Science, City University of Hong Kong, Hong Kong, SAR 999077, China (e-mail: zhwzhong.cs@gmail.com).

Xianming Liu is with the Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China, and also with Peng Cheng Laboratory, Shenzhen 518052, China (e-mail: csxm@hit.edu.cn).

Junjun Jiang and Debin Zhao are with the Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China (e-mail: jiangjunjun@hit.edu.cn; dbzhao@hit.edu.cn).

Shiqi Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, SAR 999077, China (e-mail: shiqwang@cityu.edu.hk).

The codes and trained models are available at here.

Digital Object Identifier 10.1109/TPAMI.2025.3578468

## I. Introduction

IMAGE super-resolution (SR) is a fundamental problem in computer vision that focuses on estimating high-resolution (HR) images from low-resolution (LR) ones. Over the years, this topic has received considerable attention, and various approaches have emerged [1], [2]. However, most of these algorithms are tailored for single-modality images and do not exploit the potential of utilizing other modalities as guidance. In many practical scenarios, it is common to capture a specific scene using different sensors to obtain a more comprehensive representation. For instance, in robotics, an RGB-D camera is commonly used to simultaneously capture color and depth images. Similarly, in remote sensing, multiple images are acquired with different spectral bands. To balance the trade-off between bandwidth and computational efficiency, some of these images are captured at notably low-resolution, such as depth maps, which limits their practical applications.

Considering that these cross-modality images originate from the same scene, they inherently exhibit shared characteristics, such as edges and corners. This naturally raises the question: Can we leverage HR images from other modalities to enhance the SR performance of current LR images? To answer this question, researchers have developed various methods to incorporate cross-modality information into the SR pipeline. One prominent solution is guided image super-resolution (GISR) [3], [4], [5], which uses a HR image from a different modality to guide the SR pipeline. The guidance image contains rich and diverse structures that are relevant to the target image. By transferring this information to the LR image, GISR models typically achieve better performance.

Conventional GISR tasks include guided depth map SR [6], [7], [8], guided saliency map SR [9], [10], guided thermal image SR [11], [12] and pan-sharpening [13], [14]. Specifically, guided depth map SR leverages HR RGB images to enhance the resolution of LR depth maps, addressing the resolution limitations of depth sensors. Guided saliency map SR improves the resolution and quality of saliency maps by using HR RGB images to highlight visually significant regions more accurately. Guided thermal image SR employs HR RGB images as guidance to refine the resolution of thermal images, enhancing structural and textural clarity for applications such as surveillance and medical diagnostics. Pan-sharpening combines HR panchromatic images with LR multispectral images to generate HR multispectral images, which is widely used in remote sensing for applications such as land use analysis and environmental monitoring.

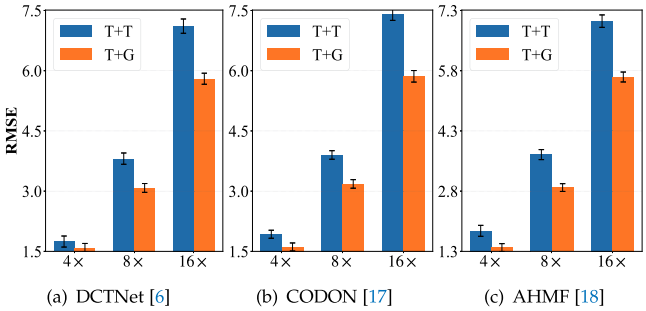



Fig. 1. RMSE comparisons for guided depth map super-resolution. The notation "T+G" indicates that the model takes both target and guidance images as input, while "T+T" refers to a model employing two target images as inputs.

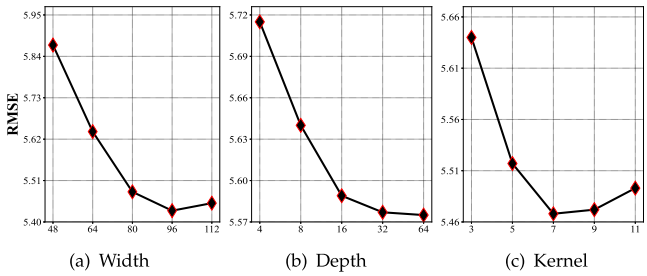



Fig. 2. Adjust a baseline model with different network width (a), network depth (b) and kernel size (c).

Recently, deep learning techniques have gained renewed interest due to their remarkable feature extraction capabilities. As a result, an increasing number of learning-based GISR models [15], [16], [17], [18] are proposed. These methods are mainly based on symmetric two-stream architectures, wherein the guidance and target images are independently processed, and then the extracted features are fused at either an early or late stage by manually designed modules to enable joint inference. Although these methods have achieved significant performance, there are still some limitations that need to be carefully addressed.

First, as the most ubiquitous solution, the symmetric two-stream architectures may not fully exploit the inherent differences between the guidance and target images. In GISR, the guidance and target images are usually obtained from different sensors, due to intrinsically different imaging mechanisms, they exhibit distinct characteristics. For example, the depth maps offer information about object distances in a scene, while the color image contains valuable color and texture details. The symmetric architectures may ignore the inherent differences between these images. Additionally, as illustrated in Fig. 1, the guidance images have different impacts on different tasks. For $16\times$ GISR, the guidance images contribute significantly for improving the model performance. However, its impact is minimal in simpler cases (e.g., $4\times$). To further optimize the model performance, we can allocate more computational resources to the guidance stream for $16\times$ GISR, and consider reducing or even removing the guidance stream to conserve computational resources for simpler tasks. **Consequently, how to discover efficient feature extractors to effectively exploit the unique characteristics of the guidance and target images is crucial**.

In addition to feature extraction, researchers have explored various fusion strategies to further enhance the performance of GISR models. Early studies [15], [19] employ simple concatenation or summation to fuse cross-modality features at a single level. Nevertheless, these fusion strategies treat the features of different streams equally, which can result in unwanted artifacts and blurred edges [10]. Moreover, in convolution neural networks, the lower layers capture local and basic features such as edges and textures, while the higher layers detect more complex and abstract features such as semantic concepts and objects. Manually selecting optimal intermediate features for cross-modality fusion is a time-consuming and laborious task. To address these challenges, the most straightforward way is to design sophisticated fusion strategies [18], [20] and fuse cross-modality features at each level in a brute force manner. However, fusing features from all levels is a time-consuming process, and developing an elegant fusion strategy relies heavily on human expertise through extensive trial and error. **Consequently, devising an elegant fusion strategy to seamlessly integrate multilevel and cross-modality features is a non-trivial task**.

Last but not least, it has been observed that improving the performance of GISR models usually results in higher computational complexity. For example, JIIF [21] comprises 10.83 M parameters, while DADA [22] reaches up to 32.53 M parameters. Even with the utilization of a powerful RTX 3090 GPU server, the inference time for a $480 \times 640$ image in the DADA model [22] requires approximately 13 seconds. Such large model sizes and prohibitive inference latencies make them unsuitable for real-time or resource-constrained applications, such as mobile phones and driving assistance systems. **Therefore, exploring how to design lightweight GISR models is worth investigating**.

In light of the above observations, we endeavor to take a step further towards developing efficient GISR models. On the one hand, designing such architectures manually can be a tedious and time-consuming task, since it requires consideration of numerous influential factors. As shown in Fig. 2, while increasing network width and kernel size can initially boost model performance, it may lead to overfitting beyond a certain threshold. Similarly, deepening the network improves capability, but the performance gains plateau as models grow larger. On the other hand, neural architecture search (NAS) have emerged as a promising approach for automatically discovering high-performing architectures [23]. Motivated by this, we propose to leverage NAS to design efficient GISR models. Considering that GISR requires nested combinations of cross-modality features and high-resolution output, directly transferring existing NAS ideas from high-leve tasks to GISR is inadequate. Therefore, we propose a dual-level search space comprised of a unimodality feature extraction space and a cross-modality feature fusion space. The former consists of lightweight building blocks, which facilitates the NAS algorithm in exploring compact and efficient architectures, whereas the latter integrates various popular fusion operations, empowering the model to automatically determine the optimal fusion strategies (i.e., which features should be

fused and which fusion operations should be chosen). Based on this search space, we propose a **D**ual-level **C**ross-modal **N**eural **A**rchitecture **S**earch framework (DCNAS) for GISR. Our framework is built upon one-shot NAS methods [24] which break down the NAS process into two separate stages: supernet training and architecture search. Moreover, we propose using pairwise ranking loss to train a performance predictor that guides the supernet training process, thereby improving the performance of the supernet and bridge the gap between training and real-world application. The contributions are as follows:

- We systematically analyze recent learning-based GISR methods and identify that designing effective feature extractors and fusion strategies is crucial for GISR tasks. We also show that manually designing optimal GISR models is a complex process, demanding significant effort and domain expertise. To address these challenges, we propose a Dual-level Cross-modality Neural Architecture Search (DCNAS) framework for automatically designing GISR models. To the best of our knowledge, this is the first attempt to explore the potential of applying NAS algorithms to GISR.
- We propose a dual-level search space specifically designed for GISR tasks, comprising a unimodality feature extraction space and a cross-modality feature fusion space. The extraction space employs lightweight building blocks, such as depthwise separable convolutions and inverted residual structures, enabling the NAS algorithm to explore more compact and efficient architectures. The fusion space includes a variety of fusion operations, allowing the model to autonomously identify optimal fusion strategies.
- We propose a supernet training strategy that utilizes a performance predictor trained with pairwise ranking loss to guide the training process. This strategy focuses on optimizing the most promising paths within the search space, thereby improving the efficiency and effectiveness of supernet training and facilitating the discovery of superior architectures.
- We conduct extensive experiments across various GISR tasks, demonstrating that the proposed method achieves a superior performance-to-complexity trade-off compared to existing hand-crafted approaches. Additionally, we analyze the architectures identified by our NAS algorithm, providing new insights into the design of efficient GISR models.

## II. Related Work

### A. Cross-Modality Image Processing

Cross-modality image processing aims to integrate information from different image modalities, such as RGB, depth, and thermal images, to overcome the limitations of individual modal. By leveraging complementary information across modalities, it has been widely applied in tasks like image super-resolution [3], denoising [25], and fusion [14], with use cases in autonomous driving [26], video surveillance [27], and more. Existing methods for cross-modality image processing can be broadly categorized into traditional optimization-based approaches and deep learning-based approaches. Traditional methods rely on hand-crafted filter kernels or objective functions. However, these methods often suffer from limited flexibility and accuracy when processing diverse image modalities due to their reliance on assumptions like structural consistency [28].

In recent years, learning-based methods have emerged as the dominant paradigm in various computer vision tasks [29], surpassing traditional hand-crafted approaches due to their ability to autonomously learn complex features. To this end, more and more advanced neural architectures are proposed [15], [30], [31], [32]. For example, Li et al. [15] propose a learnable image filter which can selectively transfer salient structures from the guidance to the target image. Lutio et al. [8] propose to explicitly learn graph regularization to effectively leverage contextual information from the guidance image. Zhou et al. [14] introduce a spatial frequency information intergration network that fuses cross-modality features in both the spatial and frequency domains. Dian et al. [33] propose a method that integrates an imaging model with CNNs and employs cross-fusion strategies to effectively learn spatial and spectral priors, achieving superior performance on both simulated and real datasets. Yuan et al. [34] propose a flow-guided upsampling network to align cross-modality structures and a flow-enhanced pyramid edge attention network for edge refinement.

### B. Efficient Neural Network Design

In the literature, various methods have been proposed for designing lightweight and fast models. For instance, MobileNet [35] and ShuffleNet [36] utilize depthwise separable convolutions and group convolutions to reduce parameter counts and computational costs. EfficientNet [37] introduces a compound scaling method that simultaneously optimizes network depth, width, and resolution using a unified scaling coefficient. Besides improving network designs, techniques such as network pruning [38], quantization [39], and knowledge distillation [40] are widely used to make neural networks more compact and efficient. However, a major drawback of these methods is the reliance on labor-intensive experimentation to identify optimal network structures.

Neural Architecture Search (NAS) aims to automatically develop an optimal neural network architecture for a specific task, thus reducing the reliance on human expertise. Typically, NAS is formulated as a bi-level optimization problem:

$$\begin{aligned} &\min_{\alpha\in\mathcal{A}} \mathcal{L}_{\text{val}}\left(D_{\text{val}}, \alpha, w_{\alpha}^{*}\right) \\ &\text{s.t.} \quad w_{\alpha}^{*} = \arg\min_{w} \mathcal{L}_{\text{train}}\left(D_{\text{train}}, \alpha, w\right), \end{aligned} \tag{1}$$

where $D_{train}$ and $D_{val}$ represent the training and validation datasets, respectively. $\mathcal{L}$ is loss function. Given a search space $\mathcal{A}$, the objective of NAS is to identify an optimal architecture $\alpha \in \mathcal{A}$ such that after training its weights $w_{\alpha}^{*}$, it minimizes $\mathcal{L}_{val}$. Current research in NAS can be broadly categorized into two dimensions: search space and search strategy.

The search space defines the set of all possible architectures that can be explored during NAS. Early studies [42], [43] focus on searching entire networks, achieving remarkable results.

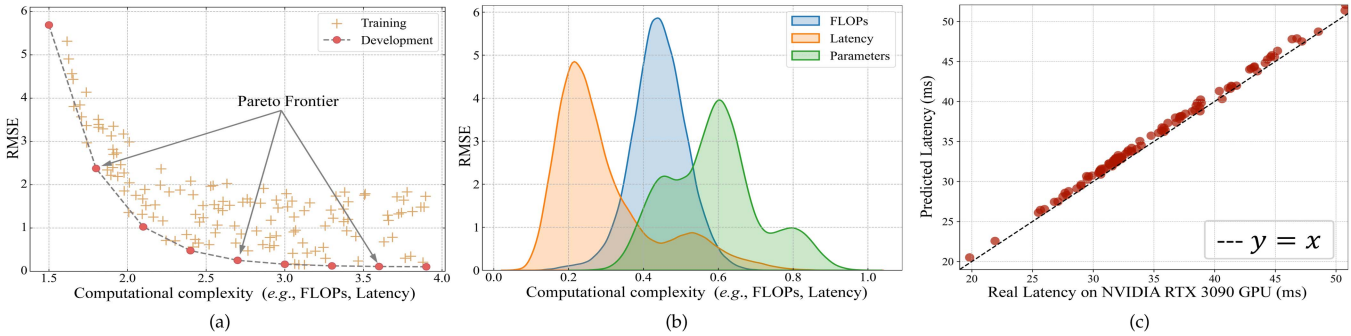



Fig. 3. (a) Illustration of the gap between supernet training and practical development in single path methods. During training, existing methods [24], [41] treat each candidate architecture equally and utilize a uniform sampling strategy to select one architecture for weight updates in each iteration. Conversely, during development, the focus shifts to the architectures that lie on the Pareto frontier (circle marker), while disregarding less efficient candidates (plus marker); (b) Probability density distribution of computational complexities across 100,000 randomly sampled paths from the proposed search space. For better presentation, the computational complexity metrics have been normalized to a 0-1 scale (Better view in color version); (c) The Latency calculated by the Lookup Table (LUT) is very accurate, with an average error (RMSE) of 0.7994.

However, the significant computational overhead of these approaches limits their practicality for real-world applications. To mitigate this issue, the cell-based search space is introduced [44], where architectures are constructed by repeating a predefined cell structure. Subsequent works [45], [46] further optimize this approach, enabling the development of efficient architectures tailored to resource-constrained devices. However, cell-based approaches struggle with low-level vision tasks that require complex architectures for multi-scale information extraction. This prompts researchers to explore diverse search spaces beyond cell-based design, such as hybrid designs [47] and multi-scale architectures [48]. However, these methods mainly focus on single-modality tasks, neglecting the potential benefits of integrating multiple modalities.

The search strategy refers to the method used to explore the search space and identify optimal architectures. Initially, most NAS methods rely on reinforcement learning (RL) [49] or evolutionary algorithm (EA) [50]. These methods require a significant computational budget to train thousands of architectures. To reduce computational costs, one-shot methods are proposed, which leverage an over-parameterized supernet to encode all candidate architectures within the search space and share weights across different architectures. Unlike traditional NAS, one-shot methods train the supernet only once, significantly reducing computational demands and enabling high-performance architecture search on standard hardware. A widely adopted variant is the gradient-based approach, which formulates the NAS as a differentiable optimization problem. For example, DARTS [51] relaxes the search space to a continuous one and jointly optimizes the network parameters and the architecture parameters through backpropagation. E-DNAS [52] introduces a latency-aware approach for embedded systems, which incorporates meta-kernel optimization, feedback mechanisms, and gradient-based search to design efficient neural networks. DE-DARTS [53] further enhances gradient-based NAS by utilizing dynamic attention networks, which adapt architectures based on input data to reduce bias during search process. However, these methods often suffer from performance degradation due to inadequate architecture discretization.

Another widely used one-shot NAS approach is the single-path method, which mitigates weight-sharing instability by training only one sampled path at a time. Since the search space is usually very large, selecting appropriate paths for optimization is crucial. SPOS [24] employs a uniform sampling strategy, where each path is selected with equal probability. However, as shown in Fig. 3(a), a gap often exists between supernet training and practical development. To bridge this gap and accelerate the supernet training process, Wang et al. [54] propose an attentive sampling strategy. Specifically, during each iteration, instead of sampling only one path, they first sample multiple paths and then use a pre-trained performance predictor to select the top-performing paths for optimization. This strategy ensures that paths near the Pareto frontier receive sufficient optimization while mitigating the adverse effects caused by optimizing weaker paths, thereby further enhancing the supernet's performance. Despite its significant performance, how to efficiently sample candidate architectures and how to accurately identify the high-performing ones still require further exploration.

### C. Learning to Rank

Learning to Rank (LTR) is a machine learning technique employed in information retrieval systems to optimize the ranking of items, such as documents, images, or search results, based on their relevance to a given query [55]. The goal of LTR is to automatically learn a scoring function from training data, which can subsequently be applied to new, unseen data to generate an ordered list of items that best align with the user's intent. Due to its significance, LTR has garnered substantial attention from the academic community.

Pointwise approaches [56] treat the ranking problem as a regression or classification problem by assigning an absolute score or label to each item independently. For example, Li et al. [57] introduce a classification-based ranking scheme, where class probabilities are learned and transformed into ranking scores using expected relevance framework. A key limitation of pointwise approaches is that they evaluate items individually without considering their relative ordering, which is essential

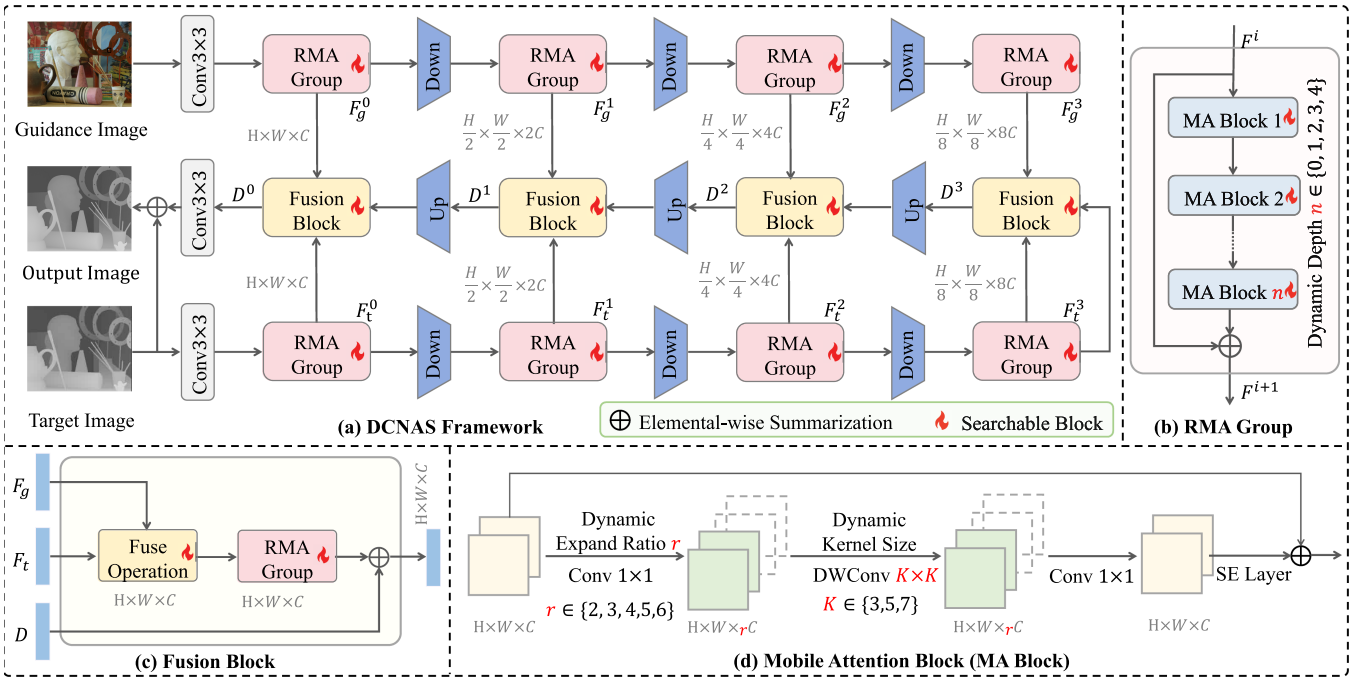



Fig. 4. An overview of the proposed Dual-level Cross-modality Neural Architecture Search (DCNAS) framework.

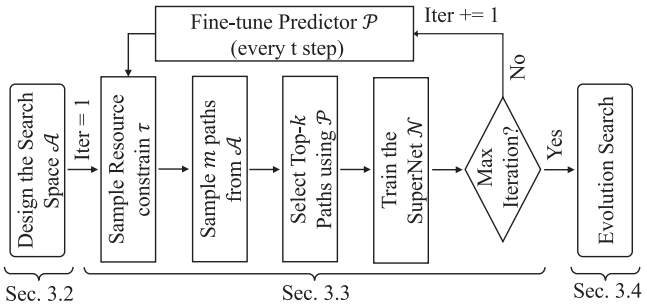



Fig. 5. Illustration of the proposed DCNAS workflow.

for ranking tasks. In contrast, pairwise approaches focus on learning the relative ranking between item pairs rather than assigning absolute scores. These methods aim to reduce ranking errors by optimizing pairwise preferences, making them more effective in capturing item relationships. For example, Cao et al. [58] formulate the ranking as a classification problem, and employ Support Vector Machines (SVM) to learn a ranking function that maximizes the margin between correctly and incorrectly ordered pairs. Burges et al. [59] propose RankNet, a neural network-based ranking model that optimizes ranking order using a probabilistic cost function and gradient descent.

## III. Proposed Method

In this section, we provide a detailed introduction of the proposed method. To offer a quick overview, we present the framework diagram of DCNAS in Fig. 4 and illustrate the architecture search workflow in Fig. 5.

### A. Problem Formulation

Given a low-resolution (LR) target image $\tilde{\boldsymbol{L}} \in \mathbb{R}^{m\times n\times C}$ and its corresponding high-resolution (HR) guidance image $\boldsymbol{G} \in \mathbb{R}^{M\times N\times c}$, the goal of GISR is to reconstruct a HR target image $\boldsymbol{H}_{\text{SR}} \in \mathbb{R}^{M\times N\times C}$ leveraging $\boldsymbol{G}$ as guidance. Here, $M = s\cdot m$, $N = s\cdot n$, and $s$ is the up-scale factor. To address the resolution disparity between $\tilde{\boldsymbol{L}}$ and $\boldsymbol{G}$, the LR image is typically interpolated to the same size as $G$. Then, the learning-based GISR methods can be formulated as:

$$\boldsymbol{H}_{\text{SR}} = \mathcal{F}_{\text{GISR}}(\boldsymbol{L}, \boldsymbol{G}, \alpha; \boldsymbol{W}_\alpha), \tag{2}$$

where $\mathcal{F}_{\text{GISR}}(\cdot)$ denotes the GISR model, which learns the nonlinear mapping from the interpolated LR image $\boldsymbol{L}$ to the desired HR target image $\boldsymbol{H}_{\text{GT}}$; $\alpha$ is the network architecture and $\boldsymbol{W}_\alpha$ means its weights. In this work, we propose to automatically search for an optimal architecture $\alpha$ to achieve $\mathcal{F}_{\text{GISR}}(\cdot)$. Specifically, building upon recent advances in one-shot NAS [24], [54], we formulate the process of finding the desired GISR models as a two-stage optimzation problem. The first stage is to optimize the weights of the supernet:

$$\boldsymbol{W}^*_{\mathcal{A}} = \arg\min_{\boldsymbol{W}_{\mathcal{A}}} \mathbb{E}_{\alpha\sim\widetilde{\mathcal{A}}}[\![\mathcal{L}_{\text{train}}(\mathcal{F}_{\text{GISR}}(\alpha, \boldsymbol{W}_\alpha))]\!], \tag{3}$$

where $\widetilde{\mathcal{A}} \subset \mathcal{A}$ means the subset of paths that lie on the Pareto frontier; $\mathcal{A}$ is the search space and $\boldsymbol{W}_{\mathcal{A}}$ means its weights. We adopt a sampling-based method to train the supernet. Existing methods utilize regression loss to train a performance predictor for $\widetilde{\mathcal{A}}$. However, this method focuses on predicting absolute performance scores, which may fail to preserve the relative ranking of candidate architectures. To address this limitation, we reformulate the training of the performance predictor from a regression problem to a ranking problem and propose to use

a pairwise ranking loss to train the predictor. Additionally, we propose an optimized sampling strategy that enhances sampling speed and ensures a more uniform distribution across architectures with varying computational complexities.

The second stage is to search for the optimal candidate architectures under different resource constraints.

$$\alpha^* = \arg\min_{\alpha \in \mathcal{A}} \mathcal{L}_{\text{val}}(\mathcal{F}_{\text{GISR}}(\alpha, \boldsymbol{W}^*_\alpha)), \quad \text{s.t. } C(\alpha) \leq \tau, \qquad (4)$$

where $\alpha^*$ is the desired network architecture; $C(\alpha)$ measures the computational cost for a given architecture $\alpha$; $\tau$ denotes the computation complexity threshold. Since the supernet is well-trained in the first stage, the performance of all paths can be efficiently estimated by directly inheriting the weights $\boldsymbol{W}^*_\alpha$ from the supernet. The optimization of (3) and (4) is addressed through two different stages of DCNAS, which are detailed in Sections III-C and III-D, respectively.

### B. Search Space Design

The search space is a basic component in NAS algorithms, as it defines the range of potential neural architectures that the algorithm can explore. To optimize the performance of search algorithms, an effective search space must satisfy two key criteria: i) it should allow for a broad range of architectural configurations with different levels of complexity to meet the demands of diverse development scenarios, ii) it should be constrained within a reasonable range to reduce computational and time costs. Based on these principles, we propose a novel search space tailored for GISR tasks. As illustrated in Fig. 4(a), it consists of two main components: a searchable feature extractor and a searchable fusion decoder. The feature extractor takes a LR image and a guidance image as inputs, aiming to extract multi-level features. The fusion decoder integrates the extracted features from both images of the same level and generates the desired HR target image.

For the proposed search space, the macro-architecture is designed based on prior knowledge (*i.e.*, an encoder-decoder structure with multi-level feature fusion) derived from well-established GISR models [5], [17], [18]. This design reduces the search space, enabling the search algorithm to be more focused and efficient. By adjusting the detailed configurations of the feature extractor and fusion decoder, various architectures can be obtained to meet different requirements. In the following, we will provide a comprehensive explanation of each component mentioned above.

*Searchable Feature Extractor:* As illustrated in Fig. 4(a), the proposed feature extractor comprises two parallel image encoders: one dedicated to processing the LR target image and the other handling the guidance image. Each encoder consists of four hierarchical feature learning stages separated by three downsampling operations. Each stage contains several basic searchable units, and by adjusting the configurations of these units, we can obtain different feature extractors. In this work, we use the Mobile block [35] with slightly modifications as the basic unit. Specifically, we exclude all batch normalization (BN) [60] layers, as recent studies [61], [62] have indicated that the BN layers may impact the performance of SR models. Moreover, the residual learning and squeeze-and-excitation (SE) block [63] are added to facilitate information flow and enhance the capabilities of the network architecture. As shown in Fig. 4(d), several Mobile blocks are stacked sequentially to form the Residual Mobile Attention Group (RMAG), where three key architectural dimensions are optimized during neural search:

TABLE I
THE CHOICES OF NUMBER OF BLOCKS, EXPAND RATIOS AND KERNEL SIZES OF DEPTH-WISE CONVOLUTION IN THE PROPOSED RESIDUAL MOBILE ATTENTION GROUP (RMAG)

| Variable | Depth | Expand Ratio | Kernel Size |
|---|---|---|---|
| Choices | {0, 1, 2, 3, 4} | {2, 3, 4, 5, 6} | {3, 5, 7} |

*i) Elastic Depth:* The number of Mobile blocks in each stage (RMAG) is an important factor that affects both the computational cost and capacity of the model. To optimize these aspects, we search for number of blocks within a broader range of choices. ii) Elastic Expand Ratio: The Mobile block employs a point-wise convolution to expand the feature channel by a certain expansion ratio, followed by a depth-wise convolution, and then reduces the feature channel through another point-wise convolution. The expansion ratio determines the number of feature channels that the depth-wise convolution operates on, thereby influencing the trade-off between model accuracy and computational efficiency. Previous approaches have employed a fixed expansion ratio for all blocks. In contrast, we conduct a search to determine a unique expansion ratio for each block. iii) Elastic Kernel Size: The size of the kernel used in the depth-wise convolution is a critical factor that impacts model complexity and performance. Although most existing methods have relied on fixed $3 \times 3$ convolution kernels, we argue that this simple choice may not be optimal. Therefore, we include the kernel size as a variable in our search space to explore different sizes and find the most suitable choice. The detailed choices for these search variables are listed in Table I.

*Searchable Fusion Decoder:* The fusion decoder is an essential component in GISR models, which aims to fuse the multi-modality features and generate the HR target image. As shown in Fig. 4, the fusion decoder is composed of several fusion blocks (Fig. 4(c)), and each block is dedicated to fuse the extracted guidance and target features of the same level. We search on two dimensions for it: **feature fusion stage** and **fusion operation**. The feature fusion stage search aims to automatically identify which stages of the features should be fused. To achieve this, we define the following two operators as feature selection operations:

(1) Identity $(\boldsymbol{x}) = \boldsymbol{x}$, *i.e.,* the feature $\boldsymbol{x}$ is selected.

(2) Zero$(\boldsymbol{x}) = 0$, *i.e.,* the feature $\boldsymbol{x}$ is discarded.

The fusion operation search aims to automatically discover the suitable fusion operations for the selected features. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the input features, the set of possible fusion operations used in our search space are as follows:

*1) Sum $(\boldsymbol{x}, \boldsymbol{y})$:* The element-wise summation is the most widely fusion operation in GISR models, and it can be defined as: Sum$(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} + \boldsymbol{y}$. (2) ConcatPW$(\boldsymbol{x}, \boldsymbol{y})$: In this operation,

**Algorithm 1:** Supernet Training Algorithm.

**Input:** Supernet $\mathcal{F}_{\text{GISR}}$ with parameter $\boldsymbol{W}$, search space $\mathcal{A}$, training dataset $\boldsymbol{D}_{\text{train}}$, validation dataset $\boldsymbol{D}_{\text{val}}$, maximum iteration $T$, computational resource distribution $\mathcal{R}$, number of sampled paths $m$, number of selected paths $k$, pre-trained performance predictor $\mathcal{P}$, predictor update interval $t$, batch size $n$

**for** $i$=*1 to* $T$ **do**
- Sample a target resource constraint $\tau$ according to the resource distribution $\mathcal{R}$;
- Uniformly sample $m$ paths $\{\alpha_i\}_{i=1}^{m}$ from the search space $\mathcal{A}$ following the constraint $\tau$;
- Get the top-$k$ paths $\{\alpha_{t_i}\}_{i=1}^{k}$ using the path predictor $\mathcal{P}$ ; `/* {t_i}_{i=1}^{k} are indices for top-k paths */`
- **for** $j$=*1 to* $k$ **do**
  - Randomly sample a training batch from $\boldsymbol{D}_{\text{train}}$;
  - Update the weights $\boldsymbol{W}_{\alpha_{t_i}}$ of path $\alpha_{t_i}$ using stochastic gradient descent;
- **end**
- **if** $i$ % $t$ == *0* **then**
  - Randomly sample a bacth of paths $\{\beta\}_{j=1}^{n}$ under the resource constrain $\tau$;
  - Calculate the loss $\{l\}_{j=1}^{n}$ of $\{\beta_j\}_{j=1}^{n}$ on $\boldsymbol{D}_{\text{val}}$;
  - Fine-tune the performance predictor $\mathcal{P}$ using $\{(\beta_j, l_j), |1 \le j \le n\}$;
- **end**

**end**

the two input features are first concatenated along the channel dimension to create a fused feature. Then the fused feature is passed through a point-wise (PW) convolution for feature reduction:

$$\text{ConcatPW}(\boldsymbol{x}, \boldsymbol{y}) = \text{PW}(\text{Concat}(\boldsymbol{x}, \boldsymbol{y})). \tag{5}$$

*3) Attention ($\boldsymbol{x}, \boldsymbol{y}$):* Attention mechanisms have been widely used in GISR models [17], [18] to enhance their ability to focus on important regions of input features. Generally, they first send the input features to a small network to generate the attention map, then the attention maps are used to re-weight the input features:

$$[\boldsymbol{m}_x, \boldsymbol{m}_y] = \text{H}(\text{Concat}(\boldsymbol{x}, \boldsymbol{y})), \tag{6}$$

$$\text{Attention}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{m}_x * \boldsymbol{x} + \boldsymbol{m}_y * \boldsymbol{y}, \tag{7}$$

where H is the network to learn the attention maps $\boldsymbol{m}_x$ and $\boldsymbol{m}_y$. We use the same architecture as in [18] to implement H.

*4) Identity ($\boldsymbol{x}, \boldsymbol{y}$):* If one of the features is not selected at the feature selection step, we will use this operation:

$$\text{Identity}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}, \boldsymbol{y}, \tag{8}$$

*Search Complexity:* The proposed search space contains two independent search space: feature extractor and fusion decoder. As shown in Fig. 4, the feature extractor contains two feature encoder, and each encoder have four RMAG. Correspondingly, the fusion decoder also has four stages, and each stage have a fusion block and a RMAG. Thus, the overall search complexity of the proposed search space is:

$$\underbrace{(5 \times 5 \times 8)^{4 \times 2}}_{\text{Feature Extractors}} \times \underbrace{(6 \times 5 \times 5 \times 8)^{4}}_{\text{Fusion Decoder}} \approx 5.31 \times 10^{30}. \tag{9}$$

## C. Supernet Training

In this subsection, we introduce the proposed supernet training pipeline. As outlined in Algorithm 1, each iteration begins by sampling a resource constraint from the resource distribution. Given this constraint, we uniformly sample $m$ paths from the search space. The Top-$k$ paths are then selected by a performance predictor and optimized on the training dataset. Additionally, we fine-tune the predictor every $t$ steps to maintain its accuracy. Our contributions are twofold: first, we propose an efficient path sampling strategy; and second, we propose using a pairwise ranking loss to train the predictor. In the following, we provide a detailed explanation of the key steps of the algorithm.

*Resource Constraint Sampling:* Generally, we cannot directly access the true distribution $\mathcal{R}$ of computational resources. However, we know the distribution $\mathcal{A}$ of all candidate paths and the transformation function $C$ that maps distribution $\mathcal{A}$ to $\mathcal{R}$. Therefore, we employ transformation sampling to draw a resource constraint $\tau$ from $\mathcal{R}$:

$$\tau = C(\alpha), \quad \text{s.t.} \quad \alpha \sim \mathcal{A}, \tag{10}$$

where $C(a)$ means a function that calculate the computational cost for the path $\alpha$. Compared to the method proposed in [54], our method is more efficient since it require a large number of samples and iterations to estimate distribution $\mathcal{R}$, while ours only requires a simple transformation function.

*Lookup Table (LUT) Construction:* To sample paths under specific constraints, it is necessary to understand the computational complexity of each path within the search space. However, since the search space is typically very large, measuring the computational complexity for each path would be a time-consuming task, especially for Latency, which requires testing on the target device. Considering that the number of basic operations (e.g., $3 \times 3$ convolution) is relatively small compared to the number of candidate paths, we build up a operation-level lookup table (LUT) to accelerate this process. Once the LUT is established, the computational cost of a path $\alpha$ can be quickly estimated as:

$$C(\alpha) = \sum_{l} (o_l^{\alpha}), \tag{11}$$

where $o_i^{\alpha}$ means the operation at the $l$-th layer of path $\alpha$.

*Resource-constrained Path Sampling:* In this work, we adopt the acceptance-rejection algorithm to sample candidate paths.

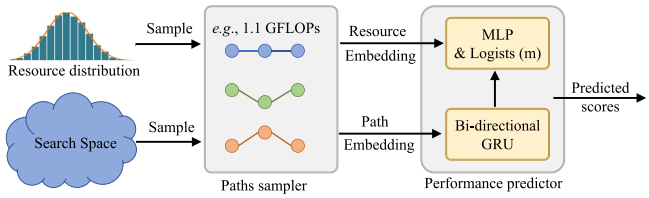



Fig. 6. The pipline for the performance predictor training.

Specifically, given $\tau$ as a constraint, this algorithm iteratively samples paths from the search space. Paths that satisfy the constraint are accepted, while others are rejected:

$$\mathbb{1}(\alpha_i) = \begin{cases} 1 & , \quad \text{if } 0 \leq \tau - \mathcal{T}(\alpha_i) \leq \sigma, \\ 0 & , \quad \text{otherwise}, \end{cases} \tag{12}$$

where $\mathbb{1}(\cdot)$ is an indicator function, and $\mathbb{1}(\alpha_i) = 1$ means that the path $\alpha_i$ will be accepted; otherwise, it is rejected. $\sigma$ is a predefined constant. This process continues until the desired number of candidates is reached.

*Performance Predictor Training:* An overview of the proposed performance predictor training process is presented in Fig. 6. Given a resource constraint $\tau_0$ and a batch of paths $\{\alpha_i\}_{i=1}^n$ as inputs, we first employ a learned embedding layer convert them into a unidied feature vector representation:

$$\tau_0^{\mathrm{e}} = \mathcal{E}_\tau(\tau_0), \tag{13}$$

$$\alpha_i^{\mathrm{e}} = \mathcal{E}_\alpha(\alpha_i), \quad 0 \leq i \leq n, \tag{14}$$

where $\mathcal{E}$ represents the embedding layer. Subsequently, the embedded paths vectors $\{\alpha_i^{\mathrm{e}}\}_{i=1}^n$ are processed through a bidirectional gated recurrent unit (GRU) $\mathcal{G}$ to integrate cross-layer information within each path:

$$\hat{\alpha}_i = \mathcal{G}(\alpha_i^{\mathrm{e}}), \quad 0 \leq i \leq n. \tag{15}$$

Finally, we concatenate the path features $\hat{\alpha}_i$ with the resource features $\tau_0^{\mathrm{e}}$ and feed them into a multi-layer perceptron (MLP) layer to predict the performance of each path:

$$s_i = \mathcal{M}(\mathrm{Concat}(\hat{\alpha}_i, \tau_0^{\mathrm{e}})), \quad 0 \leq i \leq n. \tag{16}$$

Here, we propose to use the pair-wise ranking loss to train the performance predictor:

$$\mathcal{L}_{\mathrm{rank}} = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \aleph\left((s_i - s_j) * \mathrm{sign}(\hat{s}_i - \hat{s}_j)\right), \tag{17}$$

where $\aleph(x) = (x - \epsilon)_+$ is a hinge funtion with a margin $\epsilon$. $\hat{s}_i$ means the actual performance of path $\alpha_i$, and we set it as the negative loss onvalidation set, *i.e.*, $\hat{s}_i = -\mathcal{L}(\boldsymbol{W}_\alpha, \boldsymbol{D}_{\mathrm{val}})$.

*Discussion. Resource Constraint Sampling:* A resource-constrained sampling strategy was proposed in [64], where the total computational cost is discretized into multiple intervals for uniform sampling. While effective in their context, we find this approach is not well-suited for our algorithm. As shown in Fig. 3(b), although the paths are uniformly sampled, the computational resource distribution of our search space is inherently non-uniform. Consequently, this sampling method may lead to an over-sampling of networks with excessively high or low computational complexities, while networks in the intermediate range may be under-sampled.

*Performance Predictor Training:* The method proposed in [54] also uses a performance predictor to select the most promising paths. However, our method differs in two key aspects. First, we reformulate the training of the performance predictor from a regression problem to a ranking problem and propose to use ranking loss to train the predictor. This is because the actual goal of the predictor is to determine which path will perform better, rather than precisely predicting their exact performance. Second, we replace the random forest regressor with a bidirectional GRU to better capture the cross-layer relationships within each path. Here, we use a toy example to demonstrate that regression loss (e.g., $\mathcal{L}_1$ or $\mathcal{L}_2$) may not be suitable for addressing this problem: consider two paths with ground-truth classification accuracies of 0.8 and 0.85 on the validation set. In the first scenario, their predicted accuracies are 0.85 and 0.8, respectively, reversing the true ranking. In the second scenario, the predictions are 0.75 and 0.9, respectively, maintaining the correct order. Although the regression losses are the same in both cases, the first scenario is problematic as it incorrectly swaps the rankings, leading to the selection of a suboptimal path.

*Local Optimum:* In this paper, we fine-tune the performance predictor at regular intervals to avoid trapping the supernet training into local optimum. Additionally, if the network still falls into a local optimum, the following two strategies can also be applied to address the issue: i) Mixed Sampling Strategy: Combining different sampling methods, such as uniform sampling or random search with the proposed strategy, can enhance exploration and exploitation. This approach balances the need to discover new architectures while leveraging promising ones based on current predictions. ii) Ensemble of Predictors: Utilizing multiple predictors can provide more reliable estimates and reduce reliance on any single potentially unreliable predictor. By leveraging the strengths of different models, this strategy enhances robustness and mitigates the risk of being misled by an inaccurate predictor.

### D. Constrained Evolutionary Search

Once the supernet is trained, we can evaluate the performance of different paths by directly inheriting the weights from the supernet. In this paper, following [24], [54], we utilize the evolutionary algorithm to search the top-performing paths, as the resource constraints can be easily integrated into the evolution process. The searching procedure is presented in Algorithm 2. For all experiments, the population size is set as $N = 50$, and the top number $k = 10$. Instead of randomly initializing the population, we use the trained performance predictor $\mathcal{P}$ for initialization: we first randomly select 1000 paths, then use $\mathcal{P}$ to select the best 50 from this subset. In the crossover stage, we randomly select two parent paths from the Top-$k$ paths and generate a new path by randomly choosing each layer from either parent. During mutation, a path is randomly selected from the Top-$k$ paths and each layer is randomly altered with a probability of 0.1. These steps are repeated until convergence

**Algorithm 2:** Evolutionary Architectrure Search.

**Input:** Supernet $\mathcal{F}_{\text{GISR}}(\mathcal{A}, \boldsymbol{W})$, performance predictor $\mathcal{P}$, population size $N$, validation dataset $\boldsymbol{D}_{\text{val}}$, resource constrain $\tau$, max interations $T_1$

**Output:** The path $\alpha^*$ that performs the best under the constrain $\tau$

$\mathrm{P}_0$ := InitializePopulation($\mathcal{P}, N, \tau$);
$n := N/2$ ; /* Crossover number */
$m := N/2$ ; /* Mutation number */
$prob := 0.1$ ; /* Mutation probability */
**for** $i$=1 *to* $T_1$ **do**
  $\text{RMSE}_{i-1}$ = Forward($\boldsymbol{W}, \boldsymbol{D}_{\text{val}}, \mathrm{P}_{i-1}$);
  Top-$k$ = UpdataTopK(Top-$k, n, \tau$);
  $\mathrm{P}_{\text{crossover}}$ = Crossover(Top-$k, n, \tau$);
  $\mathrm{P}_{\text{mutation}}$ = Mutation(Top-$k, n, prob, \tau$);
  $\mathrm{P}_i = \mathrm{P}_{\text{crossover}} \cup \mathrm{P}_{\text{mutation}}$;
**end**
Return the path $\alpha^*$ with the lowest RMSE in Top-$k$;

or the maximum number of iterations $T_1$ is reached, and we set $T_1 = 200$.

## IV. Experiments

In this section, we evaluate the effectiveness of the proposed method on four representative GISR tasks, including RGB guided depth map SR (Section IV-B), RGB guided saliency map SR (Section IV-C), Pan-sharpening (Section IV-D), and RGB guided thermal SR (Section IV-E). For the RGB-guided depth map super-resolution task, we perform three distinct experiments to evaluate our method: i) simulated depth map super-resolution; ii) joint depth map super-resolution and denoising; iii) real-world depth map super-resolution.

### A. Implementation Details

The proposed framework is implemented with PyTorch and trained on two NVIDIA RTX 3090 GPUs. It includes three stages: supernet training, architecture search, and retraining. The supernet is trained in about 30 hours, while the architecture search completes in under 3 hours. In the following, we will describe the training settings shared between different tasks. The specific settings for each task will be detailed in their respective sections.

*Performance Predictor:* We use pairwise ranking loss (17) to train the performance predictor. Following [54], we first conduct constraint-free training of the supernet for 30 epochs. We then randomly sample 1,024 architectures and evaluate their performance on the validation dataset. Finally, we use the sampled architectures and their evaluated scores to train the performance predictor. Additionally, we fine-tune the predictor every 20 epochs to maintain its performance.

*Supernet Training:* Our model has three hyperparameters: the number of sampled paths ($m$), the number of selected paths ($k$), and the performance predictor update interval ($t$), which are set to $m = 20$, $k = 1$, and $t = 20$, respectively. We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$ to optimize our model. The mini-batch size is set to 16, and data augmentation strategies, such as random horizontal flips and $90^\circ$ rotations, are employed to avoid overfitting. The number of feature channels of the first layer (C in Fig. 4) is set to 8 for DCNAS-Tiny and 16 for DCNAS. We employ the $\mathcal{L}_1$ loss as the default loss function.

*Architecture Search:* Following [24], [54], we adopt the evolutionary algorithm [50] to search for the desired architectures, and the detailed settings can be found in Section III-D. However, unlike these approaches, we initialize the population with the trained performance predictor to accelerate the search process. We provide two different models, **DCNAS-Tiny** and **DCNAS**, with constraints of **#Params** $\leq 0.6$**M** and **#Params** $\leq 2.0$**M**, respectively. For different scenarios, such as different tasks or up-scale factors, we search for different architectures. Since we use Lookup Table (LUT) to compute the computational complexity of the candidate architectures, which is very efficient, other constraints such as FLOPs and Latency can also be easily incorporated into this process.

*Retraining:* We apply the same settings as in the supernet training strategy to train the discovered architectures.

### B. RGB Guided Depth Map Super-Resolution

*Datasets and Settings:* Following the experimental settings of [3], [5], [14], we evaluate the performance of our proposed method on two widely used benchmark datasets: the NYU v2 dataset [65] and the RGB-D-D dataset [69]. The NYU v2 dataset [65] is a large scale indoor dataset containing 1,449 RGB-D image pairs. We use the first 1,000 image pairs as the training set and the remaining 449 image pairs as the testing set. To verify the generalization ability of the proposed method, we further incorporate five additional datasets into our evaluation: 1) 1,064 RGB-D pairs from Sintel dataset [66]; 2) the testset of DIDOE indoor dataset [67]; 3) the first 500 RGB-D pairs from SUN RGBD testset [68]; 4) the testset of RGB-D-D dataset [69]; 5) the testset of DIML indoor dataset [70]. For RGB-D-D dataset [69], we use the official training and testing splits as the training and test sets.

For the NYU v2 dataset [65], we conduct two experiments: i) depth map super-resolution, where bicubic downsampling is used to generate low-resolution (LR) depth maps with scaling factors of $4\times, 8\times$, and $16\times$; ii) joint super-resolution and denoising, where we first downsample the depth maps using bicubic interpolation and then add Gaussian noise with a noise level of 25. For the RGB-D-D dataset [69], it is a real-world dataset that contains paired LR-HR depth images. We directly use these LR depth images as input. During training, we set the initial learning rate at $1 \times 10^{-4}$ and decrease it by multipliing it by 0.5 every 100 epochs until the model converges. The small patch for the HR depth images is fixed as $256 \times 256$. For this task, we use the Root Mean Square Error (RMSE) as the evaluation metric, where a lower RMSE value indicates better performance.

*Experimental Results on NYU v2 Dataset:* We compare our method with 15 manually designed methods. Notably, all learning-based approaches are trained and tested under the same configurations to ensure a fair comparison.

TABLE II
RMSE COMPARISON WITH MANUALLY DESIGNED METHODS FOR DEPTH MAP SUPER-RESOLUTION

| Method | NYU v2 [65] | | | Sintel [66] | | | DIDOE [67] | | | SUN RGB [68] | | | RGB-D-D [69] | | | DIML [70] | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 4× | 8× | 16× | 4× | 8× | 16× | 4× | 8× | 16× | 4× | 8× | 16× | 4× | 8× | 16× | 4× | 8× | 16× | 4× | 8× | 16× |
| DJFR [15] | 2.38 | 4.94 | 9.18 | 4.90 | 7.39 | 10.33 | 5.63 | 8.24 | 9.89 | 0.81 | 1.54 | 2.80 | 1.50 | 2.72 | 5.05 | 1.27 | 2.34 | 4.13 | 2.75 | 4.53 | 6.90 |
| CUNet [16] | 1.65 | 3.35 | 6.64 | 4.27 | 5.96 | 8.51 | 3.80 | 6.98 | 9.44 | 0.63 | 1.13 | 2.20 | 1.20 | 1.86 | 3.27 | 1.18 | 1.88 | 3.25 | 2.12 | 3.53 | 5.55 |
| SVLRM [7] | 1.51 | 3.21 | 6.98 | 4.05 | 5.83 | 8.60 | 3.58 | 6.96 | 9.55 | 0.59 | 1.10 | 2.33 | 1.22 | 1.88 | 3.55 | 1.19 | 1.93 | 3.49 | 2.02 | 3.48 | 5.75 |
| FDKN [9] | 1.86 | 3.58 | 6.96 | 4.23 | 5.92 | 8.73 | 4.04 | 7.07 | 9.50 | 0.67 | 1.13 | 2.23 | 1.18 | 1.91 | 3.41 | 1.13 | 1.84 | 3.29 | 2.18 | 3.58 | 5.69 |
| DKN [9] | 1.62 | 3.26 | 6.51 | 4.38 | 5.89 | 8.40 | 3.49 | 6.96 | 9.31 | 0.63 | 1.10 | 2.16 | 1.31 | 1.87 | 3.26 | 1.27 | 1.86 | 3.22 | 2.12 | 3.49 | 5.48 |
| JIIF [21] | 1.37 | 2.76 | 5.27 | 3.82 | 5.50 | 7.46 | 2.94 | 6.17 | 8.58 | 0.54 | 0.95 | 1.79 | 1.15 | 1.77 | 2.79 | 1.17 | 1.79 | 2.86 | 1.83 | 3.16 | 4.79 |
| FDSR [69] | 1.61 | 3.18 | 5.86 | 4.14 | 5.67 | 7.86 | 3.62 | 6.54 | 8.85 | 0.64 | 1.05 | 1.97 | 1.16 | 1.82 | 3.06 | 1.10 | 1.71 | 2.87 | 2.05 | 3.33 | 5.08 |
| GraphSR [8] | 1.79 | 3.04 | 6.02 | 4.29 | 5.56 | 7.93 | 3.93 | 6.37 | 9.06 | 0.67 | 1.05 | 2.03 | 1.30 | 1.83 | 3.12 | 1.25 | 1.79 | 3.03 | 2.20 | 3.27 | 5.20 |
| CODON [17] | 1.40 | 2.77 | 5.92 | 3.76 | 5.37 | 7.86 | 3.03 | 6.17 | 9.02 | 0.53 | 0.99 | 2.09 | 1.12 | 1.79 | 3.05 | 1.18 | 1.85 | 3.06 | 1.82 | 3.17 | 5.17 |
| AHMF [18] | 1.40 | 2.89 | 5.64 | 3.84 | 5.62 | 7.55 | 2.93 | 6.14 | 8.54 | 0.57 | 0.99 | 1.82 | 1.10 | 1.73 | 3.04 | 1.10 | 1.70 | 2.83 | 1.82 | 3.18 | 4.90 |
| DAGF [10] | 1.36 | 2.87 | 6.06 | 3.93 | 5.37 | 7.66 | 3.45 | 6.09 | 8.71 | 0.57 | 0.95 | 1.96 | 1.13 | 1.76 | 2.93 | 1.15 | 1.73 | 2.84 | 1.93 | 3.13 | 5.03 |
| DCTNet [6] | 1.59 | 3.08 | 5.80 | 4.18 | 6.31 | 9.16 | 3.84 | 7.20 | 9.69 | 0.65 | 1.28 | 2.43 | 1.08 | 1.74 | 3.05 | 1.07 | 1.71 | 2.99 | 1.88 | 3.34 | 5.32 |
| SSDNet [5] | 1.60 | 3.14 | 5.86 | 3.84 | 5.72 | 8.31 | 3.04 | 6.54 | 9.46 | 0.62 | 1.11 | 2.15 | **1.04** | 1.72 | 2.92 | 1.15 | 1.83 | 3.21 | 1.98 | 3.32 | 5.65 |
| SFINet++ [14] | 1.52 | 3.04 | 6.21 | 3.91 | 5.66 | 8.78 | 3.51 | 6.47 | 9.50 | 0.56 | 1.07 | 2.34 | 1.16 | 1.87 | 3.47 | 1.21 | 1.83 | 3.60 | 1.98 | 3.32 | 5.65 |
| **DCNAS-Tiny** | 1.33 | 2.76 | 5.37 | 3.83 | 5.35 | 7.45 | 2.98 | 6.07 | 8.59 | 0.54 | 0.93 | 1.81 | 1.14 | 1.72 | 2.64 | 1.15 | 1.76 | 2.81 | 1.83 | 3.10 | 4.78 |
| **DCNAS** | **1.21** | **2.55** | **5.06** | **3.68** | **5.24** | **7.19** | **2.82** | **5.88** | **8.36** | **0.51** | **0.91** | **1.70** | 1.06 | **1.70** | **2.61** | **1.04** | **1.64** | **2.65** | **1.72** | **2.99** | **4.60** |

The best performance is indicated in **bold**, while the second-best and third-best performances are underscored and waved, respectively.

TABLE III
RMSE COMPARISONS WITH MANUALLY DESIGNED METHODS FOR JOINT DEPTH MAP SUPER-RESOLUTION AND DENOISING

| Method | DJFR [15] | CUNet [16] | SVLRM [7] | DKN [9] | DAGF [10] | **DCNAS-Tiny** Ours | **DCNAS** Ours |
|---|---|---|---|---|---|---|---|
| 4× | 4.01 | 3.84 | 3.47 | 3.45 | 3.25 | 3.19 | **3.06** |
| 8× | 6.21 | 6.00 | 5.27 | 5.29 | 4.96 | 4.82 | **4.51** |
| 16× | 9.83 | 9.23 | 8.82 | 8.55 | 7.76 | 7.58 | **7.41** |

Tables II and III present the RMSE comparisons for depth map super-resolution and joint depth map super-resolution with denoising, respectively. The results demonstrate that the models discovered by our NAS algorithm, DCNAS and DCNAS-Tiny, consistently achieve superior performance across various scaling factors and datasets. Specifically, considering the average RMSE values (the last three columns of Table II), DCNAS outperforms all competitors, while DCNAS-Tiny ranks the second-best scores for 8× and 16× . Moreover, although DCNAS-Tiny and JIIF [21] exhibit similar RMSE values, JIIF [21] requires approximately 20 times more parameters (10.83M *vs*. 0.59M). In contrast, SSDNet [5] (0.57M) has a comparable parameter count to DCNAS-Tiny but is outperformed by margins of 0.15, 0.22, and 0.87 RMSE in 4×, 8×, and 16× super-resolution, respectively. This superior performance can be attributed to our method's ability to automatically identify optimal network architectures, thereby eliminating subjective biases and the limitations inherent in manual design. Consequently, compared to handcrafted methods relying on fixed or heuristic strategies, our approach achieves superior performance with a smaller model size, thereby striking a more favorable balance between model complexity and performance.

Figs. 7 and 8 further show visual comparisons of different methods on the NYU v2 [65] and DIML [70] datasets, respectively. It is evident that both DCNAS and DCNAS-Tiny produce depth maps with significantly fewer artifacts and more accurate depth boundaries compared to other methods. Taking Fig. 7 as an example, for larger objects, such as tables and doors, all the compared methods are able to generate relatively good results. However, for smaller objects, such as the lamp stand within the orange box, most methods produce very blurry results and even fail to reconstruct the object's boundary information. In contrast, our methods excel in preserving fine details and maintaining the structural integrity of small objects. This is because our approach, unlike manually designed networks, automatically searches and combines features from different network stages to effectively capture cross-modality information. In this way, it adaptively selects the optimal feature extraction and fusion strategies for both the target and guidance images, thereby enhancing edge and detail restoration.

*Comparison of Model Complexity:* To further demonstrate the effectiveness of the proposed method, we show the number of parameters, Latency and RMSE of different methods in Fig. 9. We present five different depth SR models that are obtained by different constraints (*i.e.,* #Params $\leq$ 0.3 M, 0.6M, 1.0M, 1.5M, and 2.0M), and mark these models with red pentagrams. Based on Fig. 9, we can draw the following coclusions: i) In most cases, as the model complexity increases, model performance also improves; ii) Models discovered by the proposed DCNAS algorithm achieve a better tradeoff between computational complexity and reconstruction performance. Fig. 10 presents speed comparisons of different methods on resource-limitted devices. We can see that our DCNAS-Tiny consistently outperforms the compared methods across all tested devices. Notably, on Xiaomi Mi 9 and Intel Movidius NCS2, our method achieves real-time speed ( i.e., $>$ 25 *fps*). These speed improvements demonstrate the effectiveness of the proposed method in identifying more compact and efficient models.

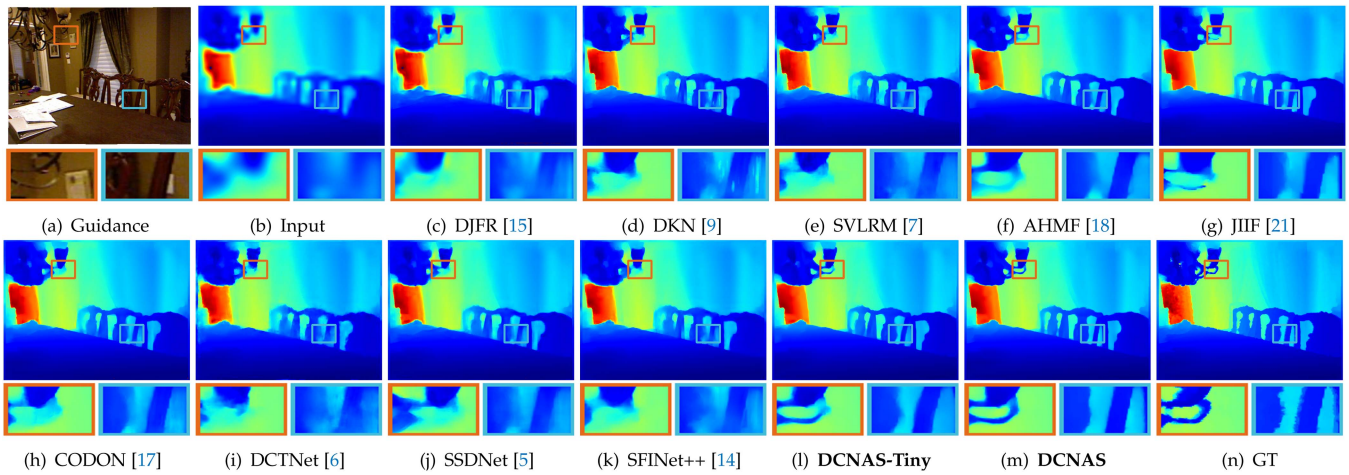



Fig. 7. Visual comparison with manually designed methods for 16× depth map super-resolution on NYU v2 dataset [65].

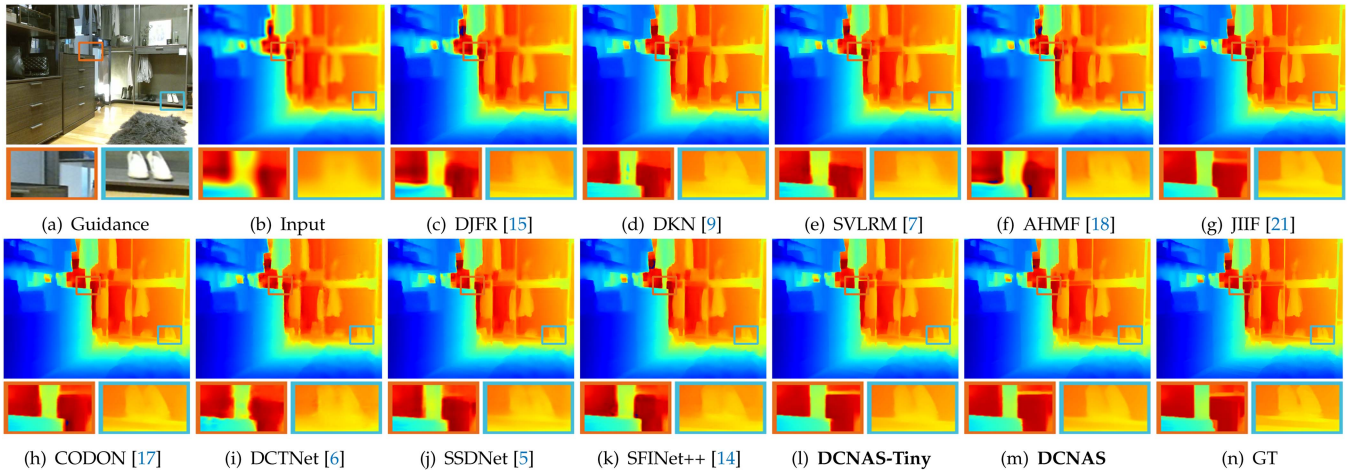



Fig. 8. Visual comparison with manually designed methods for 16× depth map super-resolution on DIML dataset [70].

TABLE IV
RMSE COMPARISON WITH MANUALLY DESIGNED METHODS ON RGB-D-D [69] DATASET FOR REAL-WORLD DEPTH MAP SR

| Method | GF [71] | DJFR [15] | DKN [9] | FDSR [69] | DADA [22] | DCTNet [6] | SGNet [72] | DAGF [10] | SSDNet [5] | SFINet++ [14] | **DCNAS-Tiny** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RMSE↓ | 6.97 | 6.24 | 5.74 | 5.49 | 5.48 | 5.38 | 5.32 | 5.36 | 5.40 | 5.38 | **5.17** |
| #Params↓ | – | 0.082M | 1.16M | 0.60M | 32.53M | 0.48M | 25.33M | 2.50M | 0.57M | 0.85M | 0.60M |

*Experimental Results on RGB-D-D Dataset:* To validate the practical effectiveness of our approach, we evaluate it on the real-world RGB-D-D dataset [69]. We compare our approach with ten state-of-the-art approaches. As shown in Table IV, the proposed approach, DCNAS-Tiny, achieves the best performance with a relatively small number of parameters. Fig. 11 presents a visual comparison of the upsampled depth maps generated by various methods. Please see the first image in this figure, the method of FDSR [69] struggles to distinguish between the wall and the tabletop barrier, resulting in unclear boundaries and a blending of the two elements. Similarly, DCTNet [6] and SFINet [14] also fail to effectively separate the wall from the tabletop barrier. Although DAGF [10] performs better in distinguishing between the wall and the barrier, it still produces relatively blurry edges. In contrast, our proposed method clearly distinguishes between the wall and the tabletop barrier, with much sharper and more precise edges.

### C. RGB Guided Saliency Map Super-Resolution

*Datasets and Settings:* Following the experimental protocols of [9], [10], we conduct experiments on saliency map SR to validate the generalization capability of the proposed method. Specifically, we utilize the DUT-OMRON dataset [74] as the

TABLE V
QUANTITATIVE COMPARISON WITH MANUALLY DESIGNED METHODS FOR 8× SALIENCY MAP SUPER-RESOLUTION

| Methods | Bicubic | GF [71] | DJFR | DMSG [73] | AHMF [18] | DKN [9] | DAGF [10] | DCTNet [6] | SFINet++ [14] | **DCNAS-Tiny** |
|---|---|---|---|---|---|---|---|---|---|---|
| F-score↑ | 0.853 | 0.821 | 0.901 | 0.910 | 0.928 | 0.926 | 0.932 | 0.921 | 0.917 | **0.937** |

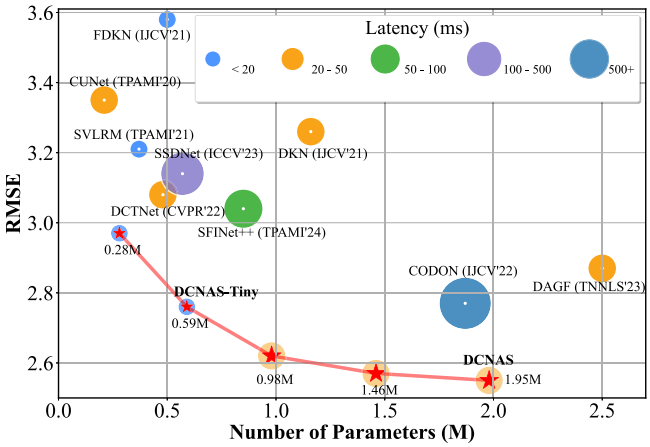



Fig. 9. Comparison with manually designed methods for 8× guided depth map super-resolution on NYU v2 [65] dataset in terms of RMSE, number of parameters and Latency (ms). For a fair comparison, we measure the Latency of different models on the same NVIDIA RTX 3090 GPU with a low-resolution depth map size of 60 × 60 as input.

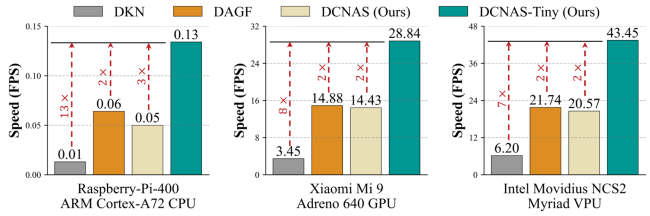



Fig. 10. Inference latency comparison of different methods on resource-limitted platforms. The size of ground truth depth map is fixed as 128 × 128, and the scale factor is set as 8.

testing set and employ bicubic downsampling with a scale factor of 8 to generate LR saliency maps. For this task, we directly apply the models trained on the NYU v2 dataset [65] to upsample the LR saliency maps, and use F-socre (higher values indicate better performance) as the evaluation metric.

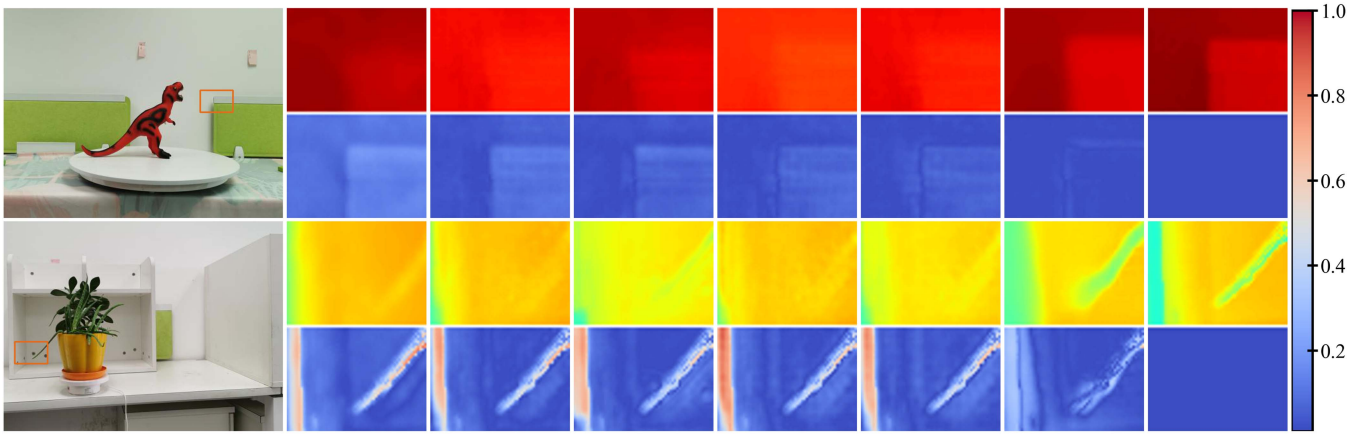
*Experimental Results:* The quantitative results are shown in Table V. Our method outperforms all compared methods, achieving a score of 0.937, which is significantly higher than the second-best score of 0.932 obtained by DAGF [10]. This indicates the superior generalization capability of our method. Furthermore, Fig. 12 provides a visual comparison of the super-resolved saliency maps produced by different methods. The images illustrate that Bicubic interpolation results in oversmoothed outputs with notable loss of structural details. Methods such as DJFR [15] and SFINet++ [14] fail to generate clear boundaries, while DKN [9] introduces artifacts around the edges. In contrast, our method not only preserves high-quality details but also maintains sharp and distinct boundaries, effectively leveraging the guidance image to transfer meaningful structural information.

### D. Pan-Sharpening

*Datasets and Settings:* For this task, we evaluate the proposed method on two widely recognized datasets: WorldView II and Gaofen-2. Both WorldView II and Gaofen-2 provide high-resolution satellite imagery. WorldView II captures a broader range of geospatial features, while Gaofen-2 focuses on providing detailed imagery for specific regions, allowing for a comprehensive evaluation of the model's performance. Specifically, following [13], [14], we first split the MS and PAN images into 128 × 128 small pathes and then downsample the MS with a scale factor 8 to produce LRMS images. During training, we set the initial learning rate at $2 \times 10^{-4}$ and decrease it by multipling 0.5 ervery 100 epochs until the model converges. Since most of the compared methods have parameter counts less than 1M, we only provide results for our smaller model, DCNAS-Tiny.

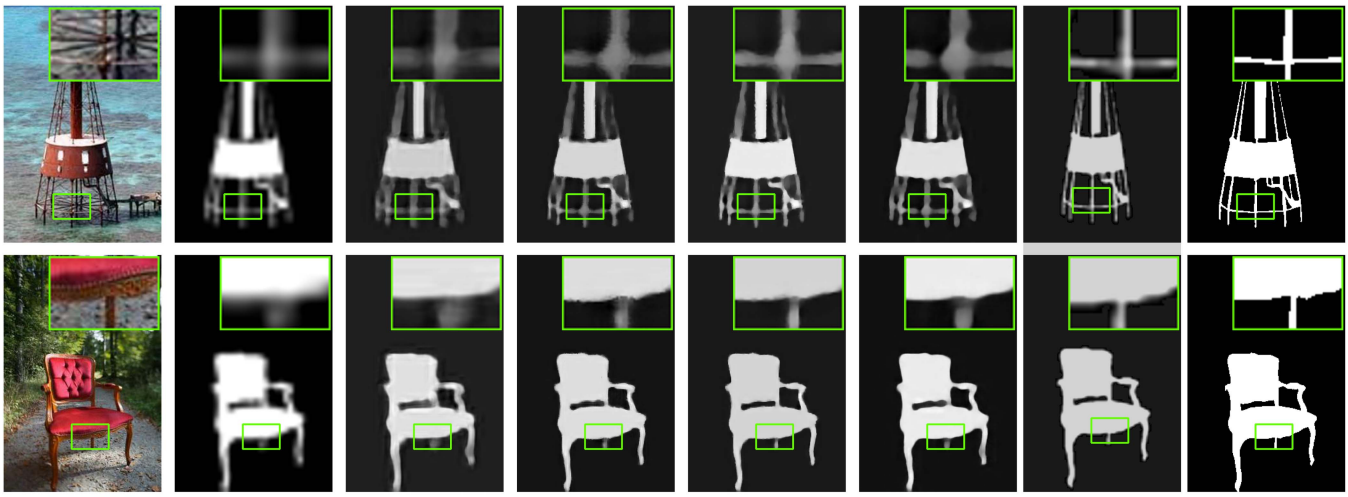
To evaluate the performance of different methods, we use seven evaluation metrics: PSNR, SSIM, SAM, ERGAS, SCC, Q index, and LPIPS. Among these metrics, PSNR, SSIM, SCC, and Q index are better when their values are higher, whereas lower values of SAM, ERGAS, and LPIPS are preferable.

*Experimental Results:* We compare our method with ten state-of-the-art methods and present the quantitative results in Table VI. From these results, it is evident that traditional methods such as GS [75] and IHS [76] generally exhibit lower performance across all metrics compared to deep learning–based methods. Among these learning–based methods, our DCNAS-Tiny achieves near–optimal results across all metrics. Notably, our method also has the smallest number of parameters compared to all other methods. We attribute this superior performance to the proposed search space and architectural search algorithm, which effectively leverage complementary information across different modalities. Fig. 13 provides a visual comparison of different pan-sharpening methods on two representative satellite images from the WorldView II dataset. The initial LRMS images display significant blurring and lack of detail. IHS [76] exhibits notable spectral distortions, and the error maps reveal significant inaccuracies. Pannet [19] improves spatial details compared to IHS [76], but it still struggles to maintain spectral integrity. Although the error maps show moderate improvements, but certain areas still exhibit notable errors. GPPNN [80] and SFINet++ [14] further improve both spatial and spectral quality. However, some distortions remain visible on the error maps. In comparison, our approach can better preserve both spectral information and spatial details.




(a) RGB (b) Input (c) FDSR [69] (d) DAGF [10] (e) DCTNet [6] (f) SFINet++ [14] (g) **DCNAS-Tiny** (h) GT

Fig. 11. Visual comparisons with manually designed methods for real-world depth super-resolution. For better visualization, a small area is selected for magnification, and the error map of this area is also displayed. Best viewed by zooming.



(a) Guidance (b) Input (c) DJFR [15] (d) DKN [9] (e) DAGF [10] (f) SFINet++ [14] (g) **DCNAS-Tiny** (h) GT

Fig. 12. Visual comparison for 8× saliency map super-resolution on DUT-OMRON dataset [74]. Best viewed by zooming.

TABLE VI
QUANTITATIVE COMPARISON WITH MANUALLY DESIGNED METHODS ON WV-II AND GF-2 DATASET FOR PANSHARPENING

| Method | WorldView II | | | | | | | | Gaofen-2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | SAM↓ | ERGAS↓ | SCC↑ | Q↑ | LPIPS↓ | Params↓ | PSNR↑ | SSIM↑ | SAM↓ | ERGAS↓ | SCC↑ | Q↑ | LPIPS↓ | Params↓ |
| GS [75] | 35.6376 | 0.9176 | 0.0423 | 1.8774 | 0.9225 | 0.6307 | 0.0829 | – | 37.2260 | 0.9034 | 0.0309 | 1.6736 | 0.7851 | 0.4211 | 0.0559 | – |
| IHS [76] | 35.2962 | 0.9027 | 0.0461 | 2.0278 | 0.8534 | 0.5704 | 0.0847 | – | 38.1754 | 0.9100 | 0.0243 | 1.5336 | 0.6738 | 0.3682 | 0.0549 | – |
| PNN [77] | 40.7550 | 0.9624 | 0.0259 | 1.0646 | 0.9677 | 0.7426 | 0.0827 | 0.69M | 43.1208 | 0.9704 | 0.0172 | 0.8528 | 0.9400 | 0.7390 | 0.0546 | 0.69M |
| PanNet [19] | 40.8176 | 0.9626 | 0.0257 | 1.0557 | 0.9768 | 0.7437 | 0.0811 | 0.69M | 43.0659 | 0.9685 | 0.0178 | 0.8577 | 0.9402 | 0.7309 | 0.0538 | 0.69M |
| MSDCNN [78] | 41.3355 | 0.9664 | 0.0242 | 0.9940 | 0.9721 | 0.7577 | 0.0782 | 2.40M | 45.6874 | 0.9827 | 0.0135 | 0.6389 | 0.9526 | 0.7759 | 0.0529 | 2.40M |
| SRPPNN [79] | 41.4538 | 0.9679 | 0.0257 | 1.0557 | 0.9680 | 0.7437 | 0.0768 | 17.11M | 47.1998 | 0.9877 | 0.0106 | 0.5586 | 0.9564 | 0.7900 | 0.0486 | 17.11M |
| GPPNN [80] | 41.1622 | 0.9684 | 0.0244 | 1.0315 | 0.9722 | 0.7627 | 0.0755 | 1.20M | 44.2145 | 0.9815 | 0.0137 | 0.7361 | 0.9510 | 0.7721 | 0.0461 | 1.20M |
| MutNet [20] | 41.6773 | 0.9705 | 0.0224 | 0.9519 | 0.9724 | 0.7741 | 0.0748 | 0.71M | 47.3042 | 0.9892 | 0.0102 | 0.5481 | 0.9603 | 0.8025 | 0.0455 | 0.71M |
| MaDUN [13] | 41.8577 | 0.9697 | 0.0229 | 0.9420 | 0.9745 | 0.7740 | 0.0721 | 0.70M | 47.2668 | 0.9890 | 0.0102 | 0.5472 | 0.9597 | 0.7973 | 0.0439 | 0.70M |
| SFINet++ [14] | 41.8115 | **0.9731** | 0.0220 | 0.9489 | 0.9735 | 0.7762 | 0.0734 | 0.85M | 47.5344 | **0.9906** | 0.0100 | 0.5356 | 0.9611 | 0.8227 | 0.0434 | 0.85M |
| **DCNAS-Tiny** | **42.1534** | 0.9717 | **0.0209** | **0.8944** | **0.9779** | **0.7956** | **0.0689** | 0.50M | **48.1405** | 0.9904 | **0.0096** | **0.4908** | **0.9673** | **0.8289** | **0.0427** | 0.58M |

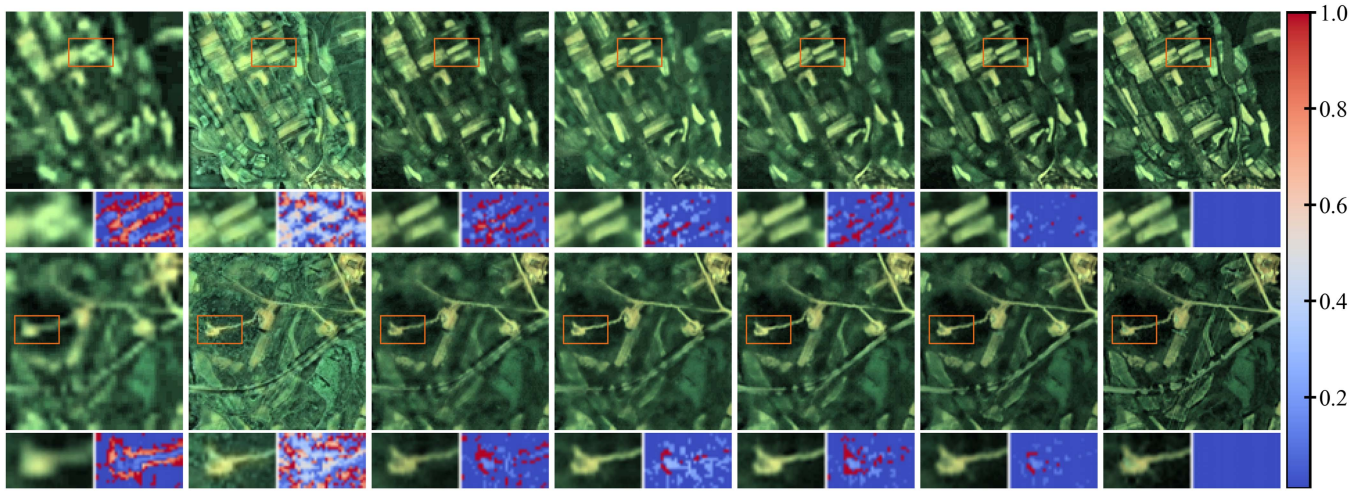



Fig. 13. Visual comparison of different pansharpening methods on two typical satellite image pairs from the WorldView II dataset. For better visualization, a small area is selected for magnification, and the error map of this area is also displayed.

TABLE VII
QUANTITATIVE COMPARISON WITH MANUALLY DESIGNED METHODS FOR RGB-GUIDED THERMAL IMAGE SUPER-RESOLUTION

| Method | #Params (8×) | 4× PSNR↑/SSIM↑ | 8× PSNR↑/SSIM↑ | 16× PSNR↑/SSIM↑ |
|---|---|---|---|---|
| Bicubic | – | 31.41/0.9076 | 25.48/0.7771 | 22.21/0.6992 |
| MultiNet [81] | 8.72M | 33.19/0.9339 | 26.94/0.8247 | 23.73/0.7459 |
| PAG-SR [11] | 7.64M | 33.96/0.9421 | 27.95/0.8540 | 24.64/0.7796 |
| UGSR [82] | 4.50M | 33.48/0.9372 | 27.19/0.8318 | 23.92/0.7509 |
| AHMF [18] | 3.36M | 34.42/0.9460 | 28.15/0.8581 | 24.59/0.7717 |
| DAGF [10] | 2.50M | 34.16/0.9439 | 27.98/0.8540 | 24.60/0.7748 |
| GuidedSR [12] | 2.13M | 34.31/0.9433 | 28.20/0.8629 | 24.89/0.7805 |
| DCTNet [6] | 0.48M | 33.10/0.9337 | 27.08/0.8295 | 24.00/0.7533 |
| SFINet++ [14] | 0.85M | 33.24/0.9349 | 27.56/0.8449 | 24.38/0.7685 |
| **DCNAS-Tiny** | 0.60M | 34.35/0.9455 | 28.16/0.8624 | 24.89/0.7955 |
| **DCNAS** | 1.98M | **34.55/0.9473** | **28.45/0.8696** | **25.11/0.7912** |

### E. RGB Guided Thermal Image Super-Resolution

*Datasets and Settings:* Masi et al. [12] provide a large-scale, high-quality RGB-T dataset to validate the performance of near-infrared image super-resolution algorithms. For this task, we use the training set provided by the authors as our training set. Since they do not provide ground-truth images for their testing set, we employ the validation set as the testing set and randomly select 100 image pairs from the training set to serve as our validation set. We use PSNR and SSIM as evaluation metrics. The initial learning rate is set to $2 \times 10^{-4}$ and is reduced by a factor of 0.5 every 100 epochs. We train our models on small patches of size $256 \times 256$ for the HR thermal images, and the downsampling factors are set to $4\times$, $8\times$, and $16\times$, respectively.

*Experimental Results:* We compare our method with eight state-of-the-art methods, For GuidedSR [12], we adjust its channel number to ensure its parameter count is approximately within the same range as other comparison methods. The quantitative results are presented in Table VII. We observe that the proposed DCNAS achieves the best objective results across all upsampling cases. Notably, DCNAS-Tiny, with only 0.6 M parameters, also achieves competitive performance. Additionally, we conduct another experiment to show the effectiveness of our method. The experimental results are presented in Fig. 15, from which we can see that the proposed method significantly outperforms the hand-crafted methods under different complexity constraints.

Fig. 14 illustrates the visual comparison of different methods at $16\times$ upscaling. Taking the first image as an example, the LR image is heavily blurred and lacks detail, making it difficult to distinguish the four wires clearly. Methods like MultiNet [81] and SFINet++ [14] show moderate improvements over the LR image, but the wires remain mostly indistinguishable and blurred. AHMF [18] and DCTNet [6] show better results than SFINet++ [14], with more distinct edges, but the wires are not clearly separated and remain somewhat blurred. In contrast, the proposed method significantly enhances clarity and definition, with the four wires becoming more distinguishable.

## V. ABLATION STUDY

In this section, we first analyze the hyperparameter settings of our framework and then conduct ablation studies to evaluate the effectiveness of our main contributions. Unless otherwise specified, all experiments are conducted on the NYU v2 dataset [65] for $8\times$ depth map super-resolution, and the complexity constraint is set as #Params $\leq$ 2.0M.

*Hyperparameters Analyses:* Our model has three hyperparameters: the number of sampled paths $m$, the number of selected paths $k$, and the performance predictor update interval $t$ (Algorithm 1). Since increasing $m$ has a similar effect to reducing $k$, we set $k$ to 1 and focus our analysis on the effect of $m$ and $t$. The experimental results are depicted in Fig. 16, from which we can see that the RMSE values exhibits a significant decrease when the $m$ increases from 1 to 20. However, beyond $m = 20$,

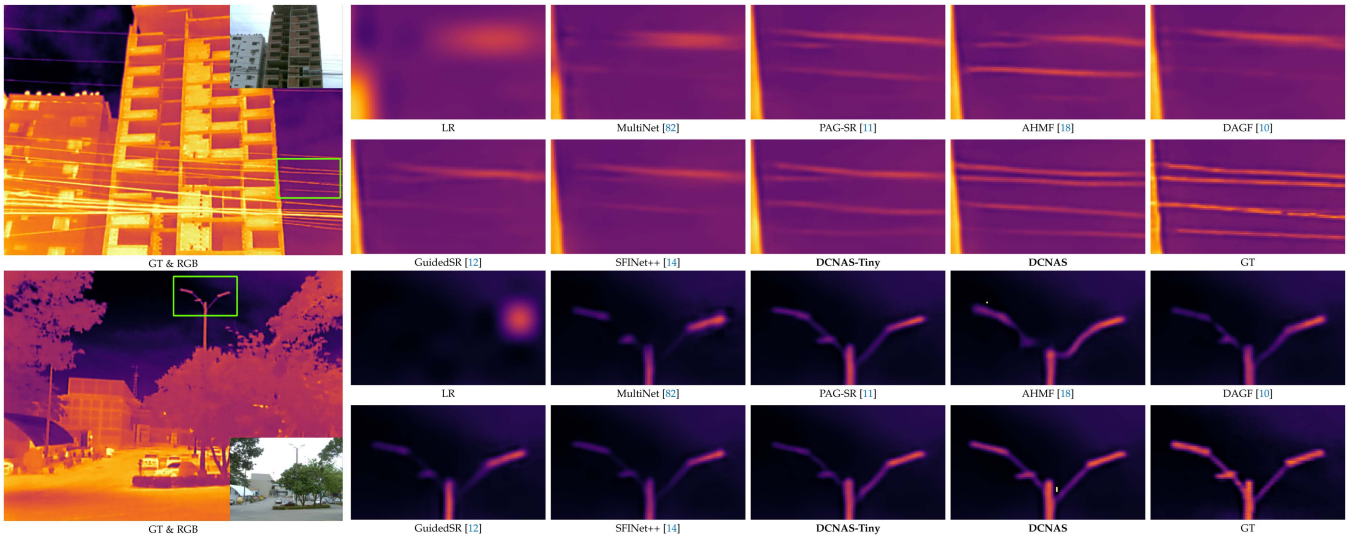



Fig. 14. Visual comparison of different RGB guided thermal image super-resolution (16×) methods. Best viewed by zooming.

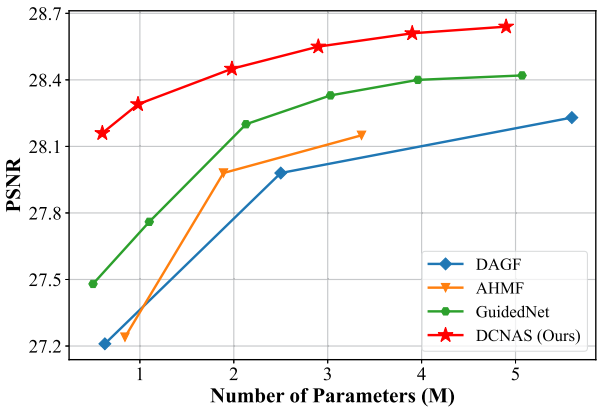



Fig. 15. PSNR and number of parameters comparison between the proposed DCNAS and recent state-of-the-art thermal image super-resolution methods (8×). For the compared methods, we obtain the models with different parameter counts by adjusting the channel number configurations.

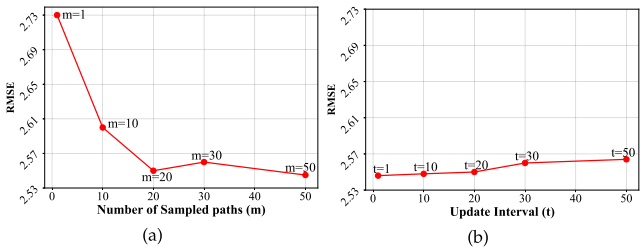



Fig. 16. Hyperparameters Analyses. RMSE comparison on NYU v2 dataset [65] (8× SR) by adjusting the number of (a) sampled paths $m$, and (b) update interval $t$.

the RMSE reduction flattens, with some instances of increase. Thus, we set $m = 20$. For $t$, its impact on RMSE is minimal. To balance computational complexity and algorithm performance, we set $t = 20$.

*Search Space:* In this paper, we propose a dual-level search space to jointly optimize the feature extractor and fusion decoder. To evaluate the effectiveness of the proposed search space, we decompose it into three distinct components: the encoder, the decoder, and the fusion operation, and implement three variants of our models:

- `Model1`: where the encoder is fixed and the algorithm searches for the fusion operation and decoder;
- `Model2`: where the fusion operation is fixed while the encoder and decoder are searched;
- `Model3`: where the decoder is fixed and the encoder and fusion operation are optimized;
- `DCNAS`: our final model.

For fixed components, we initialize them by randomly selecting operations from the search space. To ensure a fair comparison, each experiment is repeated 10 times, and the average performance is recorded.

TABLE VIII
ABLATION STUDY

| | Encoder | Fusion | Decoder | 4× | 8× | 16× |
|---|---|---|---|---|---|---|
| `Model1` | ✘ | ✓ | ✓ | 1.27 | 2.61 | 5.15 |
| `Model2` | ✓ | ✘ | ✓ | 1.24 | 2.58 | 5.10 |
| `Model3` | ✓ | ✓ | ✘ | 1.29 | 2.65 | 5.17 |
| `DCNAS` | ✓ | ✓ | ✓ | **1.21** | **2.55** | **5.06** |

Quantitative comparison of different search space on NYU v2 dataset [65].

The experimental results are presented in Table VIII, from which we can see that `Model3` achieves the worst performance, indicating that the decoder plays a crucial role in the overall performance of the GISR models. `Model2` performs better than `Model1` and `Model3`. This could be attributed to the relatively small number of fusion operations available in the framework (a maximum of four), making it less sensitive to the search process. `DCNAS`, which utilizes the entire search space, obtains the best results, demonstrating the effectiveness of the proposed search space. In addition, as shown in Figs. 9 and 15, compared to existing methods, our models, which are derived

TABLE IX
ABLATION STUDY

| | $\mathcal{L}_1$ | $\mathcal{L}_2$ | $\mathcal{L}_{\text{rank}}$ | 4× | 8× | 16× |
|---|---|---|---|---|---|---|
| Model4 | ✘ | ✘ | ✘ | 1.30 | 2.73 | 5.30 |
| Model5 | ✓ | ✘ | ✘ | 1.26 | 2.64 | 5.18 |
| Model6 | ✘ | ✓ | ✘ | 1.29 | 2.70 | 5.28 |
| DCNAS | ✘ | ✘ | ✓ | **1.21** | **2.55** | **5.06** |

Comparison of different performance predictor training strategies on NYU v2 dataset [65].

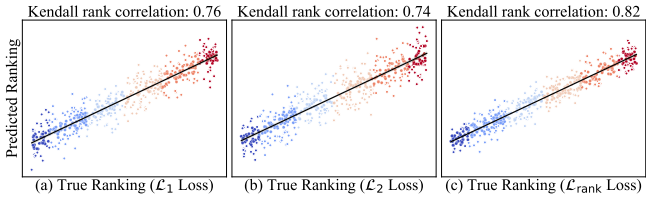



Fig. 17. Ablation Study. The predicted ranking and the true ranking. We randomly select 500 architectures for exhibition.

from the proposed search space, achieve better performance with lower computational complexity. This further validates the effectiveness of the search space introduced in this paper.

*Supernet Training:* In one-shot NAS, a common issue is the gap between supernet training and practical development. To reduce this gap, we propose to use pairwise ranking loss to train a performance predictor, which is then employed to guide the supernet training process. To validate the effectiveness of the proposed method, we implement three variants of our models:

- Model4: it does not use the predictor;
- Model5: it uses $\mathcal{L}_1$ loss to train the predictor;
- Model6: it uses $\mathcal{L}_2$ loss to train the predictor;
- DCNAS: it uses $\mathcal{L}_{\text{rank}}$ loss to train the predictor.

The experimental results are summarized in Table IX. Model4, which does not use the performance predictor, exhibits the poorest performance. Both Model2 and Model3, which utilize regression losses ($\mathcal{L}_1$ or $\mathcal{L}_2$) to train the performance predictor, show significant improvements over Model4, highlighting the importance of the predictor for supernet training. Among all methods, our approach (DCNAS), which employs rank loss to train the performance predictor, achieves the best results. This demonstrates the effectiveness of rank loss in accurately capturing the relative performance of different architectures. Additionally, we randomly select 500 architectures from the search space and compute the rankings predicted by models trained with different losses. The Kendall rank correlation between these predicted rankings and the true rankings is presented in Fig. 17. It is clear that the model trained with rank loss achieves the best performance, further demonstrating the effectiveness of our method.

*Architecture Search:* We use the evolution algorithm to find the desired architectures. Moreover, to accelerate the searching process, we initialize the population using the trained performance predictor. This strategy allows us to start the search from a more informed position, potentially reducing the time needed to discover high-performing architectures. To evaluate the effectiveness of our approach, we compare it with a baseline

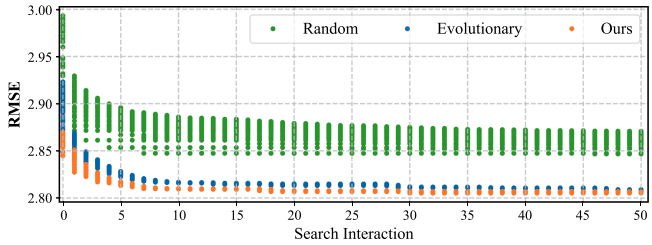



Fig. 18. Abaltion Study. The performance of Top-50 candidate architectures at each search iteration. We calculate the RMSE values by inheriting weights from the supernet.

evolution algorithm and a random search method. The RMSE curves for different strategies during the searching process are plotted in Fig. 18. As shown in the figure, our approach consistently outperforms both the standard evolution algorithm and the random search, demonstrating the benefits of using the predictor to guide the initial population in the evolutionary search process.

*Searched Architecture Analyses:* The searched architectures for 4× and 16× depth map super-resolution on the NYU v2 dataset [65] are shown in Table X. We observe several interesting phenomena:

1) in the shallow layers of the network, we can see a lot of $7 \times 7$ convolutions, while in the deeper layers, $3 \times 3$ convolutions are more common. This trend can be attributed to the need for larger receptive fields in the initial stages of the network, where the input features are more localized. The $7 \times 7$ convolutions help to quickly expand the receptive field and capture broader context. But as the features go deeper, the receptive field is already large enough, so smaller $3 \times 3$ convolutions can be used, which helps to reduce the number of parameters and computational load;

2) for multi-modality feature fusion, the fusion operations in the early stages of the network are relatively straightforward, often involving simple summation or concatenation of features. As the network goes deeper, more sophisticated fusion strategies, such as attention mechanisms, are used. This indicates that while early-stage fusion focuses on combining basic features, later stages require more complex operations to effectively integrate the richer and more intricate features;

3) for different tasks, the allocation of computational resources to various branches varies. As shown in the last column of Table X, for 4× SR, the Target branch has more parameters than the Guidance branch. In contrast, for 16× SR, the Guidance branch has more parameters. This is likely because, for 4× SR, the input target image contains more informative content. However, for 16× SR, the structural details in the target image are severely degraded, necessitating the guidance image to provide additional structural information. Consequently, more parameters are allocated to the guidance branch. Additionally, we observe that the Decoder branch consistently has the highest number of parameters, indicating that the fused features are particularly beneficial for the GISR task.

Based on the above observations, we find that the network architectures discovered through the proposed method differ significantly from those manually designed, which further confirms the necessity of the search algorithm.

TABLE X
SEARCHED ARCHITECTURES FOR $4\times$ AND $16\times$ DEPTH MAP SUPER-RESOLUTION ON NYU V2 DATASET [65]

| Model | Module | $(k, r)$, where $k$ means the kernel size, and $r$ represents the expand ratio | | | | #Params (Average) |
|---|---|---|---|---|---|---|
| | | $i=0$ | $i=1$ | $i=2$ | $i=3$ | |
| $\text{DCNAS}_{4\times}$ | Guidance ($\boldsymbol{F}_{\text{g}}^i$) | (5, 6)(7, 6)(3, 5)(5, 6) | (5, 4)(5, 5)(7, 6)(5, 6) | (3, 5) (0, 0) (0, 0) (3, 3) | (3, 5)(5, 6)(5, 5)(3, 6) | 0.46M |
| | Target ($\boldsymbol{F}_{\text{t}}^i$) | (7, 5)(5, 3)(7, 4)(7, 4) | (5, 4)(7, 2)(3, 6)(3, 3) | (5, 4)(5, 5)(7, 4)(3, 6) | (5, 3)(3, 4)(3, 3)(5, 4) | 0.65M |
| | Decoder ($\boldsymbol{D}^i$) | (3, 4)(7, 3)(5, 4)(7, 4) | (5, 6)(7, 6)(5, 6)(7, 6) | (3, 6)(7, 5)(7, 5)(5, 6) | (3, 5)(7, 6)(5, 6)(3, 6) | 0.66M |
| $\text{DCNAS}_{16\times}$ | Guidance($\boldsymbol{F}_{\text{g}}^i$) | (7, 5)(7, 6)(7, 5)(7, 2) | (7, 4)(3, 3)(5, 3)(3, 6) | (3, 3)(5, 3)(3, 5)(5, 6) | (3, 4)(5, 4)(3, 4)(5, 2) | 0.53M |
| | Target ($\boldsymbol{F}_{\text{t}}^i$) | (7, 3)(7, 2)(5, 3)(3, 6) | (3, 6)(3, 6)(5, 3)(7, 2) | (3, 5)(3, 5)(5, 4)(3, 2) | (3, 5)(5, 3)(3, 2)(3, 4) | 0.48M |
| | Decoder ($\boldsymbol{D}^i$) | (5, 5)(7, 3)(7, 5)(7, 3) | (7, 2)(5, 3)(7, 2)(5, 6) | (5, 6)(7, 6)(3, 6)(7, 5) | (7, 4) (0, 0) (5, 3)(3, 4) | 0.63M |
| $\text{Tiny}_{16\times}$ | Guidance($\boldsymbol{F}_{\text{g}}^i$) | (7, 5)(7, 6)(7, 5)(5, 5) | (7, 4)(5, 3)(5, 4)(5, 6) | (5, 6)(3, 6)(5, 2)(3, 6) | (3, 2)(3, 4)(3, 3) (0, 0) | 0.16M |
| | Target ($\boldsymbol{F}_{\text{t}}^i$) | (7, 4)(7, 2)(7, 2)(7, 3) | (7, 4)(5, 5)(7, 3)(5, 6) | (5, 5)(5, 6)(3, 6)(3, 5) | (5, 2)(3, 3)(3, 6)(3, 4) | 0.18M |
| | Decoder ($\boldsymbol{D}^i$) | (7, 6)(7, 5)(7, 3)(7, 3) | (7, 4)(5, 2)(3, 2)(3, 5) | (5, 6)(7, 3)(3, 3)(5, 5) | (3, 2)(3, 2)(3, 4)(3, 2) | 0.20M |

Here, $\boldsymbol{F}_{\text{g}}^i$, $\boldsymbol{F}_{\text{t}}^i$, and $\boldsymbol{D}^i$ denote the $i$-th block of the guidance branch, target branch, and decoder branch, respectively (see Fig. 4). The last row represents the average number of parameters for the top 50 architectures. The parameter counts for upsampling, downsampling, and fusion operations are not included in these numbers.

## VI. CONCLUSION

In this paper, we first analyze the challenges faced by current learning-based guided image super-resolution (GISR) methods, such as insufficient feature extraction and high computational complexity. We then highlight the difficulties in addressing these issues through hand-crafted network design. To this end, we propose a novel Dual-level Cross-Modality Neural Architecture Search (DCNAS) framework to automatically design optimal GISR models. Our contributions are threefold. First, we propose a dual-level search space that consists of a unimodality feature extraction space and a cross-modality feature fusion space. The unimodality feature extraction space incorporates several lightweight modules as primary building blocks, allowing the neural architecture search (NAS) algorithm to explore more compact and efficient neural architectures. The cross-modality feature fusion space includes various widely used fusion operators, allowing the model to automatically identify the optimal fusion strategy. Second, we propose to use the pairwise ranking loss to train a performance predictor, which is then employed to assist in the supernet training process, thereby improving overall performance. Finally, we provide a visual analysis of the searched networks, offering valuable insights for future network design. The experimental results demonstrate that the models discovered by DCNAS outperform state-of-the-art hand-crafted models and other NAS baselines in terms of both performance and efficiency.

## REFERENCES


[1] L. Wang et al., "Exploring fine-grained sparsity in convolutional neural networks for efficient inference," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4474–4493, Apr. 2023.

[2] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: A technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21–36, May 2003.

[3] Z. Zhong, X. Liu, J. Jiang, D. Zhao, and X. Ji, "Guided depth map super-resolution: A survey," *ACM Comput. Surv.*, vol. 55, no. 14s, pp. 1–36, 2023.

[4] X. Qiao, M. Poggi, P. Deng, H. Wei, C. Ge, and S. Mattoccia, "RGB guided tof imaging system: A survey of deep learning-based methods," *Int. J. Comput. Vis.*, vol. 132, pp. 4954–4991, 2024.

[5] Z. Zhao et al., "Spherical space feature decomposition for guided depth map super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 12547–12558.

[6] Z. Zhao, J. Zhang, S. Xu, Z. Lin, and H. Pfister, "Discrete cosine transform network for guided depth map super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5697–5707.

[7] J. Dong, J. Pan, J. Ren, L. Lin, J. Tang, and M.-H. Yang, "Learning spatially variant linear representation models for joint filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8355–8370, Nov. 2022.

[8] R. de Lutio, A. Becker, S. D'Aronco, S. Russo, J. D. Wegner, and K. Schindler, "Learning graph regularisation for guided super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1969–1978.

[9] B. Kim, J. Ponce, and B. Ham, "Deformable kernel networks for joint image filtering," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 579–600, 2021.

[10] Z. Zhong, X. Liu, J. Jiang, D. Zhao, and X. Ji, "Deep attentional guided image filtering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 12236–12250, Sep. 2024.

[11] H. Gupta and K. Mitra, "Pyramidal edge-maps and attention based guided thermal super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 698–715.

[12] R. E. Rivadeneira et al., "Thermal image super-resolution challenge results-PBVS 2024," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 3113–3122.

[13] M. Zhou, K. Yan, J. Pan, W. Ren, Q. Xie, and X. Cao, "Memory-augmented deep unfolding network for guided image super-resolution," *Int. J. Comput. Vis.*, vol. 131, no. 1, pp. 215–242, 2023.

[14] M. Zhou et al., "A general spatial-frequency learning framework for multimodal image fusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 7, pp. 5281–5298, Jul. 2025.

[15] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Joint image filtering with deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1909–1923, Aug. 2019.

[16] X. Deng and P. L. Dragotti, "Deep convolutional neural network for multi-modal image restoration and fusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3333–3348, Oct. 2021.

[17] Y. Yang, Q. Cao, J. Zhang, and D. Tao, "Codon: On orchestrating cross-domain attentions for depth super-resolution," *Int. J. Comput. Vis.*, vol. 130, no. 2, pp. 267–284, 2022.

[18] Z. Zhong, X. Liu, J. Jiang, D. Zhao, Z. Chen, and X. Ji, "High-resolution depth maps imaging via attention-based hierarchical multi-modal fusion," *IEEE Trans. Image Process.*, vol. 31, pp. 648–663, 2022.

[19] J. Yang, X. Fu, Y. Hu, Y. Huang, and J. Paisley, "PanNet: A deep network architecture for pan-sharpening," in *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 5449–5457, 2017.

[20] M. Zhou, K. Yan, J. Huang, Z. Yang, X. Fu, and F. Zhao, "Mutual information-driven pan-sharpening," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1798–1808.

[21] J. Tang, X. Chen, and G. Zeng, "Joint implicit image function for guided depth super-resolution," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 4390–4399.

[22] N. Metzger, R. C. Daudt, and K. Schindler, "Guided depth super-resolution by deep anisotropic diffusion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 18237–18246.

[23] P. Ren et al., "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, 2021.

[24] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 544–560.

[25] Q. Yan et al., "Cross-field joint image restoration via scale map," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1537–1544.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[27] R. T. Collins, A. J. Lipton, and T. Kanade, "Introduction to the special section on video surveillance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 745–746, Aug. 2000.

[28] S. Gu et al., "Learned dynamic guidance for depth image reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2437–2452, Oct. 2020.

[29] A. Liu, Y. Liu, J. Gu, Y. Qiao, and C. Dong, "Blind image super-resolution: A survey and beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5461–5480, May 2023.

[30] X. Ye et al., "PMBANet: Progressive multi-branch aggregation network for scene depth super-resolution," *IEEE Trans. Image Process.*, vol. 29, pp. 7427–7442, 2020.

[31] Q. Tang et al., "BridgeNet: A joint learning network of depth map super-resolution and monocular depth estimation," in *Proc. ACM Int. Conf. Multimedia*, 2021, pp. 2148–2157.

[32] W. Shi, M. Ye, and B. Du, "Symmetric uncertainty-aware feature transmission for depth super-resolution," in *Proc. ACM Int. Conf. Multimedia*, 2022, pp. 3867–3876.

[33] R. Dian, T. Shan, W. He, and H. Liu, "Spectral super-resolution via model-guided cross-fusion network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 10059–10070, Jul. 2024.

[34] J. Yuan, H. Jiang, X. Li, J. Qian, J. Li, and J. Yang, "Structure flow-guided network for real depth super-resolution," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 3340–3348.

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[36] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[37] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[38] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10558–10578, Dec. 2024.

[39] J. Yang et al., "Quantization networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7308–7316.

[40] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, 2021.

[41] X. Chu, B. Zhang, and R. Xu, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 12239–12248.

[42] E. Real et al., "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2902–2911.

[43] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[44] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.

[45] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[46] M. Tan et al., "Mnasnet: Platform-aware neural architecture search for mobile," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2820–2828.

[47] H. Huang, L. Shen, C. He, W. Dong, and W. Liu, "Differentiable neural architecture search for extremely lightweight image super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 6, pp. 2672–2682, Jun. 2023.

[48] X. Hu et al., "Pyramid architecture search for real-time image deblurring," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 4278–4287.

[49] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[50] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 4780–4789.

[51] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[52] J. G. López, A. Agudo, and F. Moreno-Noguer, "E-DNAS: Differentiable neural architecture search for embedded systems," in *Proc. Int. Conf. Pattern Recognit.*, 2021, pp. 4704–4711.

[53] J. Mun, S. Ha, and J. Lee, "De-darts: Neural architecture search with dynamic exploration," *ICT Exp.*, vol. 9, no. 3, pp. 379–384, 2023.

[54] D. Wang, M. Li, C. Gong, and V. Chandra, "Attentivenas: Improving neural architecture search via attentive sampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6418–6427.

[55] T.-Y. Liu et al., "Learning to rank for information retrieval," *Foundations Trends Inf. Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[56] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li, "Ranking measures and loss functions in learning to rank," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 315–323.

[57] P. Li, Q. Wu, and C. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 897–904.

[58] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking SVM to document retrieval," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieva*, 2006, pp. 186–193.

[59] C. Burges et al., "Learning to rank using gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 89–96.

[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[61] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 136–144.

[62] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 286–301.

[63] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[64] C. Tang et al., "Elasticvit: Conflict-aware supernet training for deploying fast vision transformer on diverse mobile devices," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 5829–5840.

[65] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.

[66] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 611–625.

[67] I. Vasiljevic et al., "Diode: A dense indoor and outdoor depth dataset," 2019, *arXiv: 1908.00463*.

[68] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun RGB-D: A rgb-d scene understanding benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 567–576.

[69] L. He et al., "Towards fast and accurate real-world depth super-resolution: Benchmark dataset and baseline," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9229–9238.

[70] J. Cho, D. Min, Y. Kim, and K. Sohn, "Deep monocular depth estimation leveraging a large-scale outdoor stereo dataset," *Expert Syst. Appl.*, vol. 178, 2021, Art. no. 114877.

[71] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[72] Z. Wang, Z. Yan, and J. Yang, "Sgnet: Structure guided network via gradient-frequency awareness for depth map super-resolution," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 5823–5831.

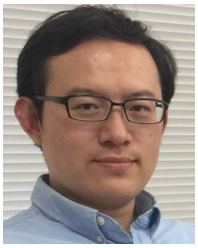
[73] T. W. Hui, C. C. Loy, and X. Tang, "Depth map super-resolution by deep multi-scale guidance," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 353–369.

[74] C. Yang, L. Zhang, H. Lu, X. Ruan, and M. Yang, "Saliency detection via graph-based manifold ranking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3166–3173.

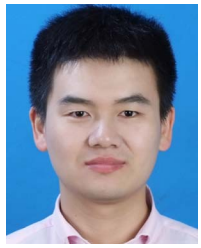
[75] C. A. Laben and B. V. Brower, "Process for enhancing the spatial resolution of multispectral imagery using pan-sharpening," US Patent. 6,011,875, Jan. 4, 2000.

[76] R. Haydn, G. W. Dalke, J. Henkel, and J. E. Bare, "Application of the IHS color transform to the processing of multisensor data and image enhancement," in *Proc. Int. Symp. Remote Sens. Arid SemiArid Lands*, Cairo, Egypt, 1982, pp. 599–616.
[77] G. Masi, D. Cozzolino, L. Verdoliva, and G. Scarpa, "Pansharpening by convolutional neural networks," *Remote Sens.*, vol. 8, no. 7, 2016, Art. no. 594.
[78] Q. Yuan, Y. Wei, X. Meng, H. Shen, and L. Zhang, "A multiscale and multidepth convolutional neural network for remote sensing imagery pan-sharpening," *IEEE J. Sel. Topics Appl. Earth Obser. Remote Sens.*, vol. 11, no. 3, pp. 978–989, Mar. 2018.
[79] J. Cai and B. Huang, "Super-resolution-guided progressive pansharpening based on a deep convolutional neural network," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 6, pp. 5206–5220, Jun. 2021.
[80] S. Xu, J. Zhang, Z. Zhao, K. Sun, J. Liu, and C. Zhang, "Deep gradient projection networks for pan-sharpening," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1366–1375.
[81] T. Y. Han, Y. J. Kim, and B. C. Song, "Convolutional neural network-based infrared image super resolution under low light environment," in *Proc. 25th Eur. Signal Process. Conf.*, 2017, pp. 803–807.
[82] H. Gupta and K. Mitra, "Toward unaligned guided thermal super-resolution," *IEEE Trans. Image Process.*, vol. 31, pp. 433–445, 2021.

**Zhiwei Zhong** received the BS degree in computer science from the Heilongjiang University, Harbin, China, in 2017, and the PhD degrees in computer science from the Harbin Institute of Technology, Harbin, in 2023. He is currently a postdoctoral researcher with the Department of Computer Science, City University of Hong Kong. His research interests include image processing, computer vision, and deep learning.



**Xianming Liu** (Member, IEEE) received the BS, MS, and PhD degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 2006, 2008, and 2012, respectively. He is currently a professor with the School of Computer Science and Technology, HIT. He has published more than 200 top-tier international conference and journal publications. He was a recipient of the IEEE ICME 2016 Best Student Paper Award.



**Junjun Jiang** (Senior Member, IEEE) received the BS degree from the Department of Mathematics, Huaqiao University, Quanzhou, China, in 2009, and the PhD degree from the School of Computer, Wuhan University, Wuhan, China, in 2014. He is currently a professor with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. He won the Finalist of the World's FIRST 10 K Best Paper Award at ICME 2017, and the Best Student Paper Runner-up Award at MMM 2015. His research interests include image processing and computer vision.

**Debin Zhao** (Member, IEEE) received the BS, MS, and PhD degrees in computer science from the Harbin Institute of Technology, China, in 1985, 1988, and 1998, respectively. He is now a professor with the Department of Computer Science, Harbin Institute of Technology. He has published more than 200 technical articles in the areas of image and video coding, video processing, video streaming, and pattern recognition.

**Shiqi Wang** (Senior Member, IEEE) received the BS degree in computer science from the Harbin Institute of Technology, in 2008, and the PhD degree in computer application technology from Peking University, in 2014. He is currently an associate professor with the Department of Computer Science, City University of Hong Kong. His research interests include video compression, image/video quality assessment, and image/video search and analysis. He received the Best Paper Award from IEEE VCIP 2019, ICME 2019, IEEE Multimedia 2018, and PCM 2017. His coauthored article received the Best Student Paper Award from the IEEE ICIP 2018. He serves as an associate editor for *IEEE Transactions on Image Processing*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Circuits and Systems for Video Technology*, etc.