Received 12 June 2024, accepted 11 August 2024, date of publication 19 August 2024, date of current version 2 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3446039

RESEARCH ARTICLE

QuantNAS for Super Resolution: Searching for Efficient Quantization-Friendly Architectures Against Quantization Noise

EGOR SHVETSOV[1], DMITRY OSIN[1], ALEXEY ZAYTSEV[1], IVAN KORYAKOVSKIY[2,3], VALENTIN BUCHNEV[4], ILYA TROFIMOV[1], AND EVGENY BURNAEV[1,2]

[1]Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
[2]Artificial Intelligence Research Institute, 105064 Moscow, Russia
[3]Toloka AI, 11000 Belgrade, Serbia
[4]Yandex Self Driving Group LLC, 123112 Moscow, Russia

Corresponding author: Egor Shvetsov (e.shvetsov@skoltech.ru)

This work was supported by the Analytical Center under the RF Government (subsidy agreement 000000D730321P5Q0002) under Grant 70-2021-00145 02.11.2021.

ABSTRACT This work aims to develop an automated procedure for discovering new, efficient solutions that can be effectively quantized in mixed-precision mode with minimal degradation. While our primary focus is on Super-Resolution (SR), our proposed procedure is applicable beyond this domain. To achieve our goals, we first develop an efficient Neural Architecture Search (NAS) procedure for full-precision (in this paper, "full-precision" or **FP** refers to floating point with a 32-bit data format) models, surpassing existing NAS solutions for SR. We then adapt this procedure for quantization-aware search. By introducing Quantization Noise (QN) during the search phase, we approximate the model degradation after quantization. Additionally, we improve search performance by implementing entropy regularization, which prioritizes operations and its precision within each search space block. Our experiments confirm the superiority of quantization-aware NAS compared to the two-step process: NAS followed by quantization. Furthermore, approximating quantization with QN offers a 30% speed improvement over direct weight quantization. We validate our approach by developing and applying it to two search spaces inspired by state-of-the-art SR models. Our code is publicly available (github.com/On-Point-RND/QuantNAS).

INDEX TERMS Single image super resolution, quantization, neural architecture search, regularization.

I. INTRODUCTION

Neural networks (NNs) have become a default solution for many problems because of their superior performance. However, wider adoption of NNs is often hindered by their high computational complexity, which poses challenges, particularly for mobile devices. Ensuring computational efficiency is crucial, especially in tasks like super-resolution [1], where deep learning models are employed on devices with limited energy capacity.

Computational efficiency can be measured using various metrics such as memory usage, latency, energy consumption, FLOPs,[1] BitOps,[2] and throughput. In our work we focus on FLOPs and BitOps but our approach can be naturally extended to other efficiency constrains. Further, we define two strategies to obtain computationally efficient models: **preventive and corrective.**

Corrective strategy: This strategy focuses on modifying pre-trained or existing architectures to enhance their computational efficiency. While this strategy is commonly used, it is limited by the fixed initial architecture, which restricts the number of possible solutions.

The associate editor coordinating the review of this manuscript and approving it for publication was Bing Li.

[1]Number of floating-point operation.
[2]Number of operations multiplied by bit value used.

Preventive strategy: On the other hand, the preventive strategy involves designing the model's architecture with hardware constraints in mind right from the beginning. This proactive approach allows for the optimization of computational efficiency and performance on the target hardware. The goal of the preventive strategy is to bridge the gap between hardware and architecture design. Even though, computational constrains become more and more important, the preventive strategy is less common in the literature.

Our work falls into the preventive category and our goal is to bring architecture design closer to hardware.

Accounting for the impact of architecture design on computational efficiency can be challenging. Therefore, one way to consider hardware constrains is to automatically search for a specific hardware-friendly architecture via NAS, simultaneously optimizing target metrics and computational efficiency. Hardware constrained NAS was introduced in [2]. In our work we search for specific operations at a given layer, the number of layers, sets of possible operations and how they connect with each other define our search space.

Another way to consider the physical aspect of a computer system is by leveraging various data types. Quantization [3], [4], [5], [6], [7], [8]] is a technique that allows for the discretization of real-valued values and reducing the **bits width**[3] for each parameter, reducing the bit width helps with memory bandwidth and integer-based arithmetic requires fewer resources than floating-point arithmetic. Reducing the bit width helps with memory bandwidth, and integer-based arithmetic requires fewer resources than floating-point arithmetic. Therefore, it allows for a smaller model size and improved inference time. Additionally, mixed-precision quantization offers the flexibility to assign different bit widths to specific layers, which allows more accurately balance between model performance and its computational efficiency [9].

Strategies to Combining NAS and Quantization: To further reduce the gap between hardware and architecture design, we can combine NAS and quantization. There are two possible ways to achieve this:

1) **Two-fold Procedure:** This approach involves performing a hardware-aware architecture search [2], optimizing factors such as FLOPs and model performance. Once an architecture is found, it is quantized using mixed-precision quantization, further optimizing computational cost and performance.
2) **Joint Procedure:** In this approach, we perform a joint hardware and quantization-aware architecture search simultaneously, optimizing computational cost and model performance. It allows for the search of optimal solutions by simultaneously exploring operations at each layer and their corresponding bit values. While this approach offers greater flexibility, it is also more challenging due to the increased complexity of the search process.

[3] In this paper, "bit width" refers to the number of bits used with the integer data format.

To evaluate the effectiveness of these scenarios, we developed a NAS procedure and a specific search space for full precision NAS, focusing on super-resolution domain. We extended this procedure to perform quantization-aware search and demonstrated that the joint procedure yields superior results. To further enhance the effectiveness of our joint procedure, we introduced Quantization Noise to approximate quantization degradation during the search.

Contributions: Firstly, we developed a differentiable Neural Architecture Search procedure specifically for the super-resolution domain. This NAS procedure incorporates entropy regularization, the importance of which is demonstrated in Section IV-F.

Furthermore, we are the first to incorporate Quantization Noise into the NAS procedure, enabling the efficient exploration of quantization-friendly architectures. We refer to this approach as Search Against Noise (SAN). We also envision the potential for SAN to be extended to search for architectures that are robust to noise or adversarial attacks. In Section IV-E0a, we empirically demonstrate that approximating quantization with QN, instead of direct quantization, leads to superior architectures.

To address the challenges of combining NAS and mixed-precision quantization, we developed two SR-specific search spaces inspired by an analysis of efficient SR models. We found that straightforward combination of NAS and mixed-precision quantization leads to unstable and slow convergence due to the large search space size and non-differentiable quantization operations. Additionally, the presence of Batch Normalization (BN) in SR models can negatively impact final performance, and training models without BN significantly slows down convergence. To overcome these issues, we introduce the Adaptive Deviation for Quantization (ADQ) module and incorporate it into our search spaces.

We demonstrate that the Joint Procedure, which combines our search spaces with ADQ and NAS with SAN, outperforms the Two-fold Procedure by a significant margin. We refer to this combination as QuantNAS.

II. RELATED WORKS

This section serves as the preliminaries for different components of our work: Differentiable NAS and co-adaptation problem, Quantization, Search space design for SR. Furthermore, we will briefly review relevant works to conduct a comparative study.

A. DIFFERENTIABLE NAS (DNAS)

DARTS [10] introduces a continuous relaxation of the discrete architecture choices and formulates the search problem as an optimization task. By using the relaxation, the search space becomes differentiable, enabling the use of gradient-based optimization algorithms. This relaxation is achieved via supernet construction. The search can be formulated as selection of a directed acyclic sub-graph (DAG) from an over-parameterized supernet. Supernet includes all possible variations of architecture that we aim to select from. Specifically, it consists of a number of layers, for each of which we have a set of nodes such that each node corresponds to a specific operation. Output of a layer is a weighted sum of nodes within this layer. Weights used in such operation are called importance weights.

During the search procedure, we aim to assign importance weights for each edge and consequently select a sub-graph using edges with the highest importance weights. An example of such selection is in Figure 1. The weights assignment can be done in several ways. The main idea of DNAS is to update importance weights $\boldsymbol{\alpha}$ with respect to a loss function parameterized on supernet weights W.

DNAS has been proven to be efficient to search for computationally-optimized models. In this case, hardware constraints are introduced as an extension of an initial loss function. FBnet [2] focuses on optimizing FLOPs and latency with the main focus on classification problems. AGD [11] and TrilevelNAS [12] apply resource constrained NAS for super resolution problem (SR) by minimizing FLOPs during search procedure.

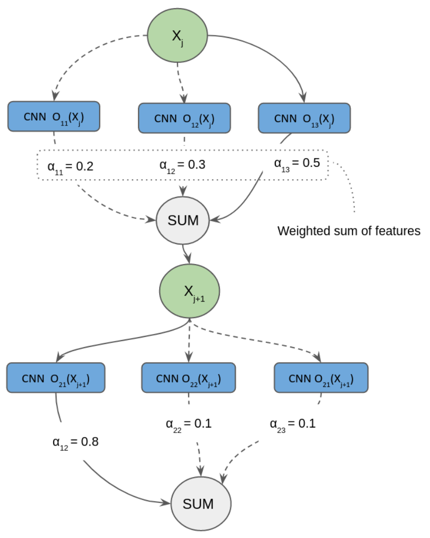

FIGURE 1. An overparametrized supernet is a graph. In this graph, multiple possible operation edges connect nodes that are outputs of each layer. The α values represent the edge importance. The joint training of operation parameters and their importance allow for differentiable NAS. The final architecture is the result of the selection of edges with the highest importance between each consecutive pair of nodes. The selected edges are marked with solid lines, composing a final neural network architecture.

B. SUPERNET CO-ADAPTION DURING DIFFERENTIABLE ARCHITECTURE SEARCH

makes it difficult to a select final architecture from the supernet because selected operations depend on all the left in the supernet operations. Therefore, we need to explicitly enforce operations independence during search phase. Below, we dicuss available solutions.

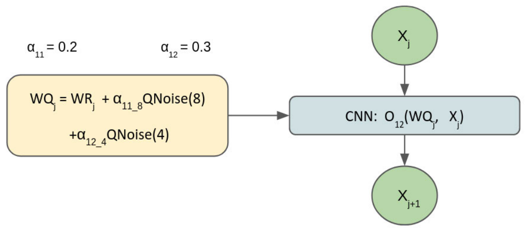

FIGURE 2. SAN approach for a single layer A function *QNoise*(*b*) generates quantization noise. *WR* are real valued weights, *WQ* are output pseudo quantized weights, and α is a vector of trainable parameters. By adjusting α, we search for acceptable model degradation caused by quantization procedure. *QNoise*(*b*) is independent of weights and allows for propagation of gradients. For quantization-aware search, each blue operation on Figure 1 becomes SAN operation with noisy weights.

Enforcing operations independence depends on the graph structure of a final model. In our work, we use the Single-Path graph - one possible edge between two nodes (more in Appendix J-A). For this structure, the sum of node outputs is a weighted sum of features (see Figure 1), the co-adaptation problem becomes obvious. Second layer convolutions are trained on a weighted sum of features, but after selecting a subgraph via discretization, only one source of features remains. Therefore, enforced independence for the vector of $\boldsymbol{\alpha}$ is necessary. In BATS [13], independence is achieved via scheduled temperature for softmax. Entropy regularization is proposed in Discretization-Aware search [14]. In [15], authors proposed an ensemble of Gumbels to sample sparse architectures for the Mixed-Path strategy, and in [16], Sparse Group Lasso (SGL) regularization is used. In ISTA-NAS [17], authors tackle sparsification as a sparse coding problem. Trilvel NAS [12] proposed sorted Sparsestmax.

C. QUANTIZATION

Quantization is a non-differentiable mapping that converts a range of values into a discrete range of values of fixed length. This conversion allows significant improvements in computational efficiency by allowing faster integer arithmetic and reduced memory consumption. However, quantization imposes limitations on the expressivity of the model, which can potentially lead to quality degradation. There are various methods available to perform this mapping, we utilize linear quantization methods introduced in LSQ [5] and HWGQ [4].

It has been observed that different layers in a model may require different levels of expressivity. As a result, they can be mapped into different bit ranges, where lower bit ranges correspond to lower expressivity. This insight has led to the emergence of mixed-precision quantization techniques [9], enabling more flexible and efficient model designs.

We use following quantization methods to compare our approach with Two-fold Procedure: LSQ, HWGQ and EdMIPS [9]

Since quantization is inherently a non-differentiable operation, we need to employ approximations in order to incorporate it into the training process. One commonly used approximation is the Straight Through Estimation (STE) [18] technique. The use of STE [18] can introduce oscillations during optimization due to rounding errors. DiffQ [7] addresses this issue by introducing differentiable Quantization Noise (QN) to approximate the degradation caused by quantization. Notably, DiffQ only applies QN to model weights. In NIPQ [19], authors combine QN and LSQ (Learned Step Size Quantization) [5] by sharing the same parameter corresponding to the same bit levels. This approach facilitates an easy transition from QN to LSQ during the later stages of training, allowing for improved quantization performance. While application of QN is not novel we are the first to demonstrate and study its advantages for NAS.

D. EFFICIENT SUPER RESOLUTION ARCHITECTURES

Many current models for SR suffer from high computational costs, making them impractical for resource-constrained devices and applications. To address this issue, lightweight SR networks have been proposed. One such network is the Information Distillation Network (IDN) [20], which splits features and processes them separately. Inspired by IDN and IMDB [21], the RFDN [22] improves the IMDB architecture by using RFDB blocks [22]. These blocks utilize feature distillation connections and cascade 1×1 convolutions towards a final layer.

While there are numerous ideas for making SR models lightweight, developing such methods can be labor-intensive due to the trial-and-error process. In our work, we aim to improve existing architectures - specifically the RFDN network, which was the winning solution in the AIM 2020 Challenge on Efficient Super-Resolution [23]. We focus on modifying RFDN to be more amenable to quantization by constructing a quantization-aware, RFDN-based space.

Search space design is crucial. It should be both flexible and contain known best-performing solutions. Even a random search can be a reasonable method with a good search space design. In AGD [11], authors apply NAS for SR, and search for **(1) a cell** - a block which is repeated several times, and **(2) kernel size**, along with other hyperparameters like the number of input and output channels. TrilevelNAS [12] extends the previous work by adding **(3) network level** that optimizes the position of the network upsampling layer. We compare our full precision NAS for SR with TrilevelNAS and AGD.

In [24], the authors applied NAS to an RFDN-based search space, which is similar to our work.[4] In Sections V-B1 and VI, we compare and discuss our method and results with the approaches mentioned above, specifically for the non-quantized setting.

E. HARDWARE-AWARE DNAS

DNAS can be employed to search for architectures with desired properties. In OQAT [3], authors performed a search for architectures that perform well when quantized. They used uniform quantization, where the same number of bits is used for each layer of the neural network. The architectures discovered through quantization-aware search performed better when quantized compared to architectures found without considering quantization.

In the studies conducted in [25] and [26], similar areas of research were explored, with some techniques closely related to ours. In [25], the authors focused on a joint search for neural architecture, layer-wise weights, activations precision, and accelerator design. In [26], the authors used DNAS to search for quantization-friendly architectures for the super-resolution problem, with U-net as the backbone for their search space. In Section VI, we will further compare our approach with the latter one.

In a more recent study [27], the authors applied NAS to search for binary neural networks, which represents the lowest and most extreme level of quantization. To achieve high-performing results, they enhanced their search space by incorporating recent advancements in binary neural networks, including new types of activation functions [28] and approximations of the sign function used for quantization. However, it is important to note that their focus was on the image classification task, which is outside the scope of our work.

F. SUMMARY

While the approaches mentioned above provide valuable insights into the integration of mixed-precision quantization and NAS, it is important to note that they do not address all the challenges associated with SR, DNAS, or quantization, nor do they explore their combinations comprehensively. In our work, we go beyond simply combining existing methods and mainly focus on addressing the problems that arise from their combination.

III. METHODOLOGY

The description of the methodology consists of four parts. We start with Subsection III-A that describes our search spaces. Subsection III-B describes our ADQ module, which is specifically designed to substitute Batch Norm and make the search space more robust. In Subsection III-C we introduce mixed precision search and provide details on Search Against Noise technique. The complete QuantNAS search procedure is described in subsection III-H. It includes the description of the used loss function.

A. SEARCH SPACE DESIGN

The work considers two approaches to design the search space. For designing the first search space, which we call *Basic search space*, we take into account recent results in this area and, in particular, the SR quantization challenge [29]. We combine most of these ideas in the search design depicted in Figure 3. The second search space RFDN expands a recent computationally efficient architecture RFDN [22].

Basic search space consists of head, body, upsample, and tail blocks. The **head** block is composed of two searchable convolutional layers. These layers play a crucial role in the model and are responsible for capturing important features

[4]It's worth noting that our choice of RFDN was independent from [24].

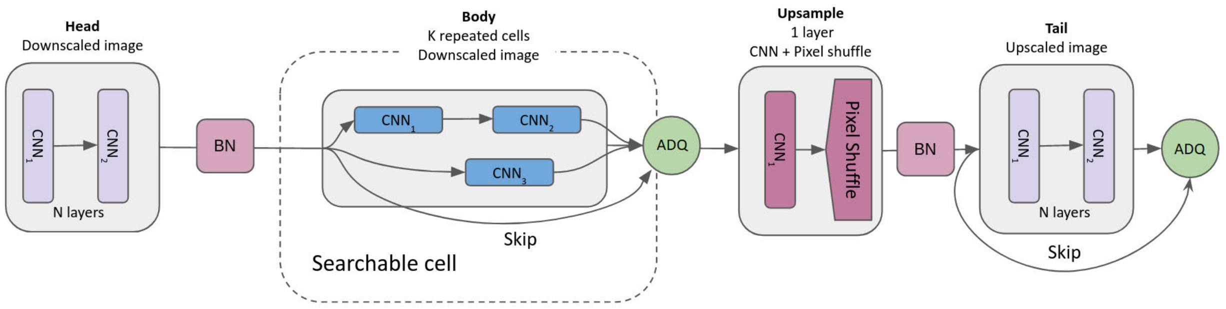

FIGURE 3. **The search space design. We separate the whole architecture into 4 parts: head, body, upsample, and tail. The head and the tail have $N = 2$ convolutional layers. The identical body part is repeated $K = 3$ times, unless specified otherwise. The number of channels for all the blocks equals 36, except for the head's first layer, upsample, and the tail's first layers. All the blocks with skip connections incorporate ADQ.**

at the beginning of the network. The **body** block consists of three layers. It includes two consecutive layers and one parallel layer, along with a skip connection. This block can be repeated multiple times to enhance the model's performance. Each body block is followed by ADQ. The **upsample** block consists of one searchable convolutional layer and one upsampling layer. The upsampling operation is performed using the Pixel Shuffle technique, as described in ESPCN [30]. This block is responsible for increasing the resolution of the image. The **tail** block consists of two searchable convolutional layers with a skip connection. This block is located at the end of the network and is responsible for refining the features and generating the final output. *Basic search space* is depicted in Figure 3.

The deterministic part of our search space includes the position of upsample block and the number of channels in convolutions. The ADQ block is used only in quantization-aware search. **The variable part** refers to quantization bit values and operations within head, body, upsample, and tail blocks. All possible operations are defined in Appendix section J-C. We perform all experiments with 3 body blocks, unless specified otherwise.

To create the *RFDN search space*, we start with the RFDN architecture and replace all convolutional layers with searchable operations. The possible operations are listed in the Appendix, specifically in section J-C. Each operation has different bit values that can be searched. The key difference between the *Basic search space* and the *RFDN search space* is that the latter uses repeated RFDB blocks [22] instead of body blocks defined above, the tail block has only 1 layer, and there is no head block. Instead, 3 input channels are repeated and concatenated to have a desirable shape for RFDB block.

If we were to substitute the body block in the *Basic search space* with the RFDB block, it would result in an architecture very similar to RFDN [22]. The *Basic search space* can be easily modified to create various popular SR architectures by adjusting the structure of the inner blocks, which is why it is called the *Basic search space*.

B. ADQ MODULE

Variation in a signal is crucial for identifying small details for the SR preventing usage of normalizations like batch

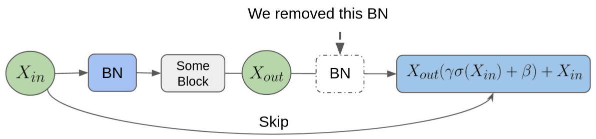

FIGURE 4. **Comparison of ADQ with AdaDM [31]. *Some Block* represents any residual block with several layers within, $\sigma(X_{in})$ is a variance of input signal, γ and β are learnable scalars. We remove the second BN after X_{out} from original AdaDM.**

norm (BN). After normalization layers, the residual feature's standard deviation shrinks, causing the performance degradation in SR task [31]. On the other hand, training a neural network without BN is unstable and requires more iterations. The issue is even more severe for differentiable NAS, as it requires training an overparameterized supernet.

The authors of AdaDM [31] proposed to rescale the signal after BN, based on its variation before BN layers. We empirically proved that removing the second BN in AdaDM scheme, keeping only the first one in each block, leads to better results for quantized models. We call this block ADQ. Original AdaDM block and our modification are depicted in Figure 4. All the body blocks during search have the ADQ module.

C. QUANTIZATION-AWARE TRAINING—QAT

Our aim is to find quantization-friendly architectures that perform well after quantization. A standard approach to obtain a trained and quantized model is the Quantization-Aware Training [6]. For QAT, we sequentially perform the following: (a) quantize full precision weights and activations during forward pass; (b) compute gradients using STE [18] by bypassing non differentiable quantization operation; and (c) update full precision weights.

Let consider the following one-layer neural network (NN) with input x,

$$y = f(a(x)) = Wa(x), \tag{1}$$

where a is a non linear activation function and f is a function parametrized by a tensor of parameters W. While in (1) f is a linear function, a convolutional operation is also a linear function, so the structure is general enough. To decrease the computational complexity of the network, we replace expensive float-point operations with quantized operations. Quantization occurs for both weights W and activation a.

The quantized output has the following form:

$$y_q = f_q(a_q(x)) = o(G(x, b), Q(W, b)), \tag{2}$$

where quantization bit width is denoted as b and a convolution layer is denoted as o.

$Q(W, b)$ is a quantization function for weights.

D. WEIGHTS AND ACTIVATION QUANTIZERS

There are various choices for quantizers, and weights and activations can use different quantizers. Below, we will describe our settings for both weight and activation quantization.

For weight quantization, we utilize linear uniform quantization, which means that the quantization steps are evenly spaced throughout the dynamic range of the weights. More details on uniform quantizers can be found in Section A. Specifically, we opted to use Learned Step Quantization (LSQ) [5] with a trainable step value for weight quantization, as it has demonstrated good performance in our initial experiments. Additionally, uniform quantizers are hardware-friendly.

Models are more sensitive to activation quantization rather than weight quantization [32]. It is well known that non-uniform quantization methods may lead to higher compression rates with better accuracy. Therefore, for activation quantization, we use a non-uniform quantizer. Specifically, we opted for the HWGQ (Hardware-Guided Quantization) method [4], as it has shown better performance than LSQ for activation quantization in our initial experiments and is simple to implement. In our notation, we use $G(W, b)$ for the HWGQ quantizer. More details on HWGQ can be found in Section B.

We need to note that while there are other possible options for quantizers, we did not study them since they are outside the scope of our work.

E. MIXED PRECISION SEARCH AND BitMixer

The task of mixed-precision quantization is to find optimal bit width for each layer in a neural network. In this scenario, we replace each convolution layer with an operation that we call BitMixer. BitMixer's purpose is to model a weighted sum of the same convolutional operation quantized to different bit width during search.

The straightforward approach is to have an independent set of weights for each convolutional operation. Let $\boldsymbol{\alpha}$ be vector of importance weights corresponding to different bit width. Then, for convolution o and input x_l, the output of l-th layer is:

$$BitMixer(\boldsymbol{\alpha}, o, x_l) = \sum_{b \in B} \alpha_b \cdot o\Big(G\big(x_l, b\big), Q(W_b^o, b)\Big) \tag{3}$$

This approach requires computing the same convolutional operation $|B|$ times.

F. QUANTIZATION-AWARE SEARCH WITH SHARED WEIGHTS (SW)

To improve computational efficiency we can quantize weights of identical operations with different quantization bits instead of using different weights for each quantization bit, this idea was studied in [9]. Then (3) becomes:

$$BitMixer(\boldsymbol{\alpha}, o, x_l) = \Big(\sum_{b \in B} \alpha_b\Big) \cdot o\Big(\sum_{b \in B} \hat{\alpha}_b G\big(x_l, b\big), \sum_{b \in B} \hat{\alpha}_b Q(W^o, b)\Big), \tag{4}$$

where $\hat{\alpha}_b = \frac{\alpha_b}{\sum_{b \in B} \alpha_b}$.

Note that it is not necessary to use the first term of the product as well as alpha-scale when $\sum_{b \in B} \alpha_b = 1$. This is the case when we only try to find optimal bit-width for a layer but **do not** search for convolutional operation, like in [9]. QuantNAS, however, searches for different bit width and operation simultaneously, which is why we perform such adjustments. Without it, $\sum_{b \in B} \alpha_b$ is significantly smaller than 1, with forward signal magnitude being drastically reduced after going through BitMixer.

The effectiveness of SW can be seen from (4): it requires fewer convolutional operations and less memory to store the weights.

G. SAN - QUANTIZATION-AWARE SEARCH AGAINST NOISE

To further improve computational efficiency and performance of search phase, we introduce SAN. Model degradation caused by weights quantization is equivalent to adding the quantization noise $QNoise_b(W) = Q(W, b) - W$. Then, quantized weights is $Q(W, b) = W + QNoise_b(W)$ and (4) is:

$$BitMixer(\boldsymbol{\alpha}, o, x_l) = \Big(\sum_{b \in B} \alpha_b\Big) \cdot o\Big(\sum_{b \in B} \hat{\alpha}_b QNoise_b\big(x_l\big) + x_l, \sum_{b \in B} \hat{\alpha}_b QNoise_b\big(W^o\big) + W^o\Big) \tag{5}$$

This procedure does not require weights quantization and is differentiable, unlike straightforward quantization. $QNoise_b$ is a function of W because it depends on its shape and magnitude of values. Given the quantization noise, we can more efficiently run forward and backward passes for our network, similar to the reparametrization trick.

Adding quantization noise is similar to adding independent uniform variables from $[-\Delta/2, \Delta/2]$ with $\Delta = \frac{1}{2^b - 1}$, as explained in in [8]. However, for the noise sampling, we use the following procedure as in [7]:

$$QNoise(b) = \frac{\Delta}{2} z, z \sim \mathcal{N}(0, 1), \tag{6}$$

as it performs slightly better than the uniform distribution [7].

H. THE SEARCH PROCEDURE

The search and training procedures are carried out as two separate steps. First, we search for an architecture and bit width, and then we conduct another training session for the selected architecture. We assign individual α-importance values to each possible operation with a specific bit. This means that the number of α values is equal to the number of operations multiplied by the number of possible bits. For l-th layer, there are $|O_l|$ possible operations and $|B|$ bit widths.

For search step, we alternately update supernet's weights W and edge importances $\boldsymbol{\alpha}$. Two different subsets of training data are used to calculate the loss function and derivatives for updating W and $\boldsymbol{\alpha}$, similar to [11]. Hardware constraints and entropy regularisation are applied as additional terms in the loss function for updating $\boldsymbol{\alpha}$.

To calculate the output of l-th layer x_{l+1} we sum the outputs of BitMixer taking as inputs: importance values $\boldsymbol{\alpha}_i^l$, convolutional operation o_i^l, and input x_l.

$$x_{l+1} = \sum_{i=1}^{|O^l|} BitMixer(\boldsymbol{\alpha}_i^l, o_i^l, x_l), \tag{7}$$

where $\sum_{i=1}^{|O^l|} \sum_{b=1}^{|B|} \alpha_{ib}^l = 1$ and all $\alpha_{ib}^l \geq 0$.

Note that when $|B| = 1$, $\hat{\alpha}_b$ used in (4) and (5) becomes 1, and (7) will give us the standard DNAS procedure for searching operations.

Then, the final architecture is derived by choosing a single operator with the maximal α_{ib}^l among the ones for this layer. Finally, we train the obtained architecture from scratch.

To optimize $\boldsymbol{\alpha}$, we compute the following loss that consists of three terms:

$$L(\boldsymbol{\alpha}) = L_1(\boldsymbol{\alpha}) + \eta L_{cq}(\boldsymbol{\alpha}) + \mu(t) L_e(\boldsymbol{\alpha}),$$

where η and $\mu(t)$ are regularization constants. $\mu(t)$ increases with each iteration t, details are covered in Appendix section IV-F. $L_1(\boldsymbol{\alpha})$ is the l_1-distance between high resolution and restored images averaged over a batch. $L_{cq}(\boldsymbol{\alpha})$ is the hardware constraint and $L_e(\boldsymbol{\alpha})$ is the entropy loss that enforces sparsity of the vector $\boldsymbol{\alpha}$. The last two losses are defined in two subsections below.

1) HARDWARE CONSTRAINT REGULARIZATION

The hardware constraint is proportional to the number of floating point operations FLOPs for full precision models and the number of quantized operations BitOps for mixed-precision models. $F_{fp}(o, x)$ is the function computing FLOPs value based on the input image size x and the properties of a convolutional layer o: kernel size, number of channels, stride, and the number of groups. We use the same number of bits for weights and activations in our setup. Therefore, BitOps can be computed as $F_q(o, x) = b^2 F_{fp}(o, x)$, where b is the number of bits. Then, the corresponding hardware part of the loss L_{cq} is:

$$L_{cq}(\boldsymbol{\alpha}) = \sum_{l=1}^{|S|} \sum_{i=1}^{|O^l|} \sum_{b=1}^{|B|} \alpha_{ib}^l b^2 F_{fp}(o_i^l, x_l), \tag{8}$$

where S is a supernet's block or layer consisting of several operations, the layer-wise structure is presented in Figure 1, and x_l is the input to l-th layer. We normalize $L_{cq}(\boldsymbol{\alpha})$ value by the value of this loss at initialization with the uniform assignment of α, as the scale of the unnormalized hardware constraint reaches 10^{12}.

2) ENTROPY REGULARIZATION

We use entropy regularization such that after the architecture search, the model keeps only one edge between two nodes, we call this procedure sparsification. Let us denote as $\boldsymbol{\alpha}_l$ all alphas that correspond to edges that connect a particular pair of nodes. They include different operations and different bits. At the end of the search, we want $\boldsymbol{\alpha}_l$ to be a vector with one value close to 1 and all remaining values close to 0.

The sparsification loss $L_e(\boldsymbol{\alpha})$ for $\boldsymbol{\alpha}$ update step has the following form:

$$L_e(\boldsymbol{\alpha}) = \sum_{l=1}^{|S|} H(\boldsymbol{\alpha}_l), \tag{9}$$

where H is the entropy function, that we can calculate, as $\boldsymbol{\alpha}$ admits interpretation as the categorical distribution. The coefficient before this loss $\mu(t)$ depends on the training epoch t. The detailed procedure for regularization scheduling is given in Appendix IV-F.

I. SUMMARY

We present the summary of our mixed-precision quantization NAS approach in this subsection. The algorithm outlining our procedure can be found in Appendix 1.

- We consider two search spaces that take origin from SR competition and from a recent RFDN [22] architecture. To make the search procedure more stable and efficient, we use ADQ.
- For different edges in a single layer that have different bit values and identical operations, we share weights making training more efficient.
- As a loss function, we use a three-term function. The first term is a standard SR loss, the second one constrains FLOPs of a model forcing it to be more efficient, and the last one leads that importance weights converge to a single non-zero value for each layer.
- We perform Quantization-Aware search, so our architecture in the end would be quantization-friendly. The idea is to substitute non-differentiable quantization with additive differentiable quantization noise. In this way, we ensure good quantization property of a final architecture.

IV. RESULTS

The section is organized as follows:

- Initially, we provide an overview of the protocol and introduce the competitor methods. This segment also includes technical specifications for both our approach and the alternative methods.
- We commence by conducting a comparative analysis between our approach and existing methods in the field of NAS and quantization for super-resolution.
- To conclude, we present the findings of an ablation study, offering insights into how different contributions have improved our approach.

We provide the code for our experiments here.

A. EVALUATION PROTOCOL

The evaluation protocol follows that from [12] with DIV2K [33] the training dataset. The test datasets are Set14 [34], Set5 [35], Urban100 [36], and Manga109 [37] The super-resolution scale is 4.

In the main body of the paper, we present results on Set14. The results for other datasets are presented in Appendix. For training, we use RGB images. For PSNR score calculation, we use only the Y channel similarly to [11] and [12]. Evaluation of FLOPs and BitOPs is done for fixed image sizes 256 × 256 and 32 × 32, respectively.

To illustrate the effectiveness of our approach, we present the application of QuantNAS in two distinct settings. The first setting involves our carefully crafted custom-designed search space BasicSpace, which has been developed through a comprehensive analysis of the state-of-the-art (SOTA) architectures. Furthermore, we demonstrate the versatility of QuantNAS by applying it to the champion of the AIM 2020 Efficient Super-Resolution Challenge, namely RFDN [22].

a: BASIC SEARCH SPACE

For all experiments, we consider the following setup if not stated otherwise. A number of body blocks is set to 3. For quantization-aware search, we limit the number of operations to 4 to obtain a search space of a reasonable size. Following others, our setup considers two options as possible quantization bits: 4 or 8 bits for activations and weights.

b: RFDN SEARCH SPACE

In our approach, we substitute each convolutional layer within RFDN with a search block consisting of six possible operations. Each operation can be configured to use either 4 or 8 bits. So, 12 edges constitute the search block. Furthermore, we apply the ADQ module around each RFDN block. Notably, the ESA block remains consistently quantized to 8 bits.

c: PERFORMANCE EVALUATION

QuantNAS has the capability to discover models that exhibit varying levels of computational complexity and quality. By adjusting the hardware constraint regularization, we identify several distinct models. When these points are plotted on a graph, they form a Pareto plot, which serves as a means to assess the method's quality. Visual evaluation of such a graph can be conducted as follows: the more points situated to the left and higher up on the graph, the better the overall performance, as they have higher quality and lower complexity.

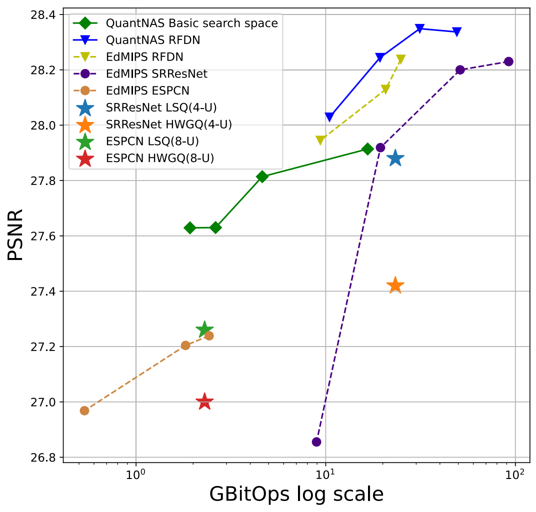

FIGURE 5. Our quantization-aware QuantNAS approach vs. fixed quantized architectures. PSNR is for Set14 dataset and BitOPs is for image size 32 × 32. We aim at the upper left corner that corresponds to smaller GBitOps and higher quality measure via PSNR.

B. QuantNAS VS. QUANTIZATION OF FIXED ARCHITECTURES

a: COMPARED METHODS

To compare QuantNAS with other mixed and uniform architectures, we consider the following fixed models: SRResNet [38], ESPCN [30], and RFDN [22]. For mixed precision quantization, we use EdMIPS [9]. Our setup for EdMIPS is matching the original setup and search is performed for different quantization bits for weights and activations. For uniform quantization, we use LSQ [5] and HWGQ [4].

Our QuantNAS with ADQ and SAN has the following hardware penalties: 0, 1e-4, 1e-3, 5e-5 to produce distinct points at the Pareto frontier. Mixed precision quantization by EdMIPS [9] for SRResNet [38], ESPCN [30], and RFDN [22] used hardware penalties 0, 1e-3, 1e-2, 1e-1 respectively.

b: MAIN TABLE

We start with comparison of different quantized models and results of QuantNAS. ESPCN model is quantized to 8 bits, and SRResNet is quantized to 4 bits to match the desired model size.

Table 1 presents the results. QuantNAS outputs architectures with a better PSNR/BitOps trade-off than uniform quantization techniques for both considered GBitOPs values about 5 and about 20.

TABLE 1. **Quantitative results for different quantization methods for different models. "U" - stands for uniform quantization - all bits are the same for all layers. GBitOPs were computed for 32 × 32 image size.**

Method	Model	GBitOPs	PSNR
LSQ (8-U)	ESPCN	2.3	27.26
HWGQ (8-U)	ESPCN	2.3	27.00
LSQ (4-U)	SRResNet	23.3	27.88
HWGQ (4-U)	SRResNet	23.3	27.42
EdMIPS (4,8)	SRResNet	19.4	27.92
EdMIPS (4,8)	RFDN	20.7	28.13
QuantNAS (4,8)	**RFDN**	**19.3**	**28.24**
QuantNAS (4,8)	RFDN w/o SAN	16.8	28.23
QuantNAS (4,8)	RFDN w/o ADQ	17.2	28.16
QuantNAS (4,8)	**Basic**	**2.6**	**27.81**
QuantNAS (4,8)	Basic w/o SAN	4.1	27.65
QuantNAS (4,8)	Basic w/o ADQ	4.4	27.65

c: PARETO FRONTIER

Figure 5 complements the table above, showcasing the complete Pareto frontier of architectures obtained using QuantNAS and EdMIPS.

QuantNAS excels in discovering architectures with more favorable PSNR/BitOps trade-offs, particularly within the range where BitOps values overlap, when compared to SRResNet and ESPCN. Additionally, our approach demonstrates a notable performance improvement, especially when compared to quantized ESPCN. Moreover, it is evident that QuantNAS for RFDN delivers superior results in comparison to EdMIPS RFDN.

Due to computational limits, our search space is bounded in terms of the number of layers. We cannot extend our results beyond SRResNet or RFDN in terms of BitOps to provide a more detailed comparison.

C. QuantNAS VS. NAS + FIXED QUANTIZATION

We also look at whether a joint selection of architecture and bit level - *mix precision setting* is better than neural architecture search for a single fixed bit level - *uniform quantization setting*.

We apply QuantNAS to the RFDN architecture in three distinct settings, each varying in the available bit options for each block. The first setting exclusively searches for 4-bit blocks, the second explores 8-bit blocks, and the third provides the flexibility to select either 4- or 8-bit operations for each block.

Results are shown in Figure 6. The graph clearly demonstrates that broadening the search space to include mixed bit width (4/8 bits) consistently leads to the discovery of superior models. It is worth noting that the Pareto plots for various metrics, such as SSIM and PSNR, exhibit remarkably similar results. This trend persists across all experiments.

D. ADAPTIVE DEVIATION FOR QUANTIZATION

We start with comparing the effect of AdaDM [31] and ADQ on three architectures randomly sampled from our search space. Table 2 shows that both original AdaDM and Batch Normalization hurt the final performance, while ADQ improves PSNR scores.

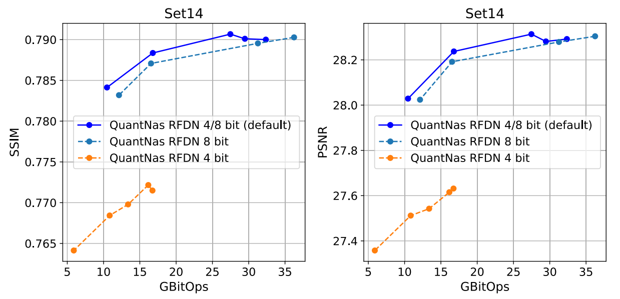

FIGURE 6. **NAS + mixed precision vs. NAS + uniform quantization. We conduct identical search for QuantNAS RFDN, but with the flexibility to search for blocks using fixed 4 bits, 8 bits, or both 4 and 8 bits simultaneously. Results are presented using the Set14 dataset via SSIM and PSNR metrics.**

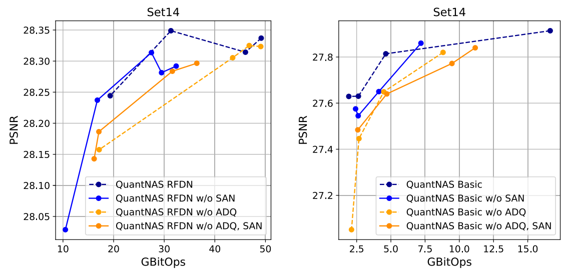

FIGURE 7. **Comparison of different NAS options: vanilla, without SAN, without ADQ, and without SAN and ADQ settings. Without SAN means that we use quantization with shared weights. Metrics are for the Set14 dataset. Left - QuantNAS RFDN, right - QuantNas Basic search space.**

TABLE 2. **PSNR of SR models with scaling factor 4 for Set14 dataset. M1 and M2 are two arbitrary mixed precision models randomly sampled from our search space.**

Model	Model M1	Model M2
Without Batch Norm	27.55	28.00
With Batch Norm	27.00	27.16
Original AdaDM	27.33	27.84
Our AdaDM	**27.68**	**28.05**

In Figure 7, we can see that architectures found with ADQ perform better in terms of both PSNR and BitOPs, highlighting the clear advantage of using ADQ in the search procedure for both our custom search space and RFDNs.

E. SEARCH AGAINST NOISE

a: QUALITY

The results shown in Figure 7 also demonstrate the contribution of SAN to our method.

Provided metrics demonstrate that SAN serves as a reasonable and effective replacement for direct quantization. Furthermore, in the setting involving our custom search space, SAN consistently enhances the search procedure, and when combined with ADQ, it yields a distinct improvement for RFDN.

b: TIME EFFICIENCY

To demonstrate the time efficiency of our approach, we measured the average training time for three quantization

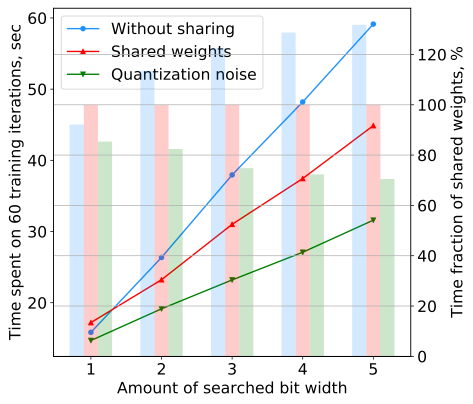

FIGURE 8. Time comparison of quantization noise and weights sharing strategy during the search phase of quantization-aware NAS. Y-axis (on the left) shows time spent on 60 training iterations (line plot). The secondary Y-axis (on the right) presents the time fraction of SW strategy (bar plot).

methods: without weight sharing, with weight sharing used by EdMIPS, and employing search against quantization noise used by QuantNAS with SAN.

SAN reduces computation during the search phase, avoiding the need for quantizing each bit level individually. We ran the same experiment with varying numbers of searched quantization bits. For uniform quantization, the number of searched bit widths is 1, while for mixed precision (4 or 8 bits for each block) it is 2.

Figure 8 shows the advantage of SAN in training time. As the number of searched bits grows, so does the advantage. On average, we get up to 30% speedup.

The search procedure with SAN, two optional bits and **Basic search space** takes about 24 hours to finish for a single GPU TITAN RTX. To train the final model takes about 6 hours on a single GPU.

F. ENTROPY REGULARIZATION

In this section, we provide evidence that the entropy regularization helps a NAS procedure and give details on the source of these improvements.

We consider three settings to compare QuantNas with and without Entropy regularization: (A) small search space, SGD optimizer; (B) big search space, Adam [39] optimizer; and (C) small search space, Adam [39] optimizer. All the experiments were performed for full precision search. For small and big search spaces, we refer to Appendix J. We perform the search without hardware penalty to analyze the effect of the entropy penalty.

Quantitative results for Entropy regularization are in Table 3. Entropy regularization improves performance in terms of PSNR for all the experiments.

Figure 9 demonstrates dynamics of operations importance for joint NAS with quantization for 4 and 8 bits. 4 bits edges are depicted in dashed lines. Only two layers are depicted: the first layer for the head (HEAD) block and the skip (SKIP) layer for the body block. With entropy regularization, the most important block is evident from its important weight value(α from (7)). Without entropy regularization, we have no clear most important block. So, our search procedure has two properties: (a) the input to the following layer is mostly produced as the output of a single operation from the previous layer; (b) an architecture at final search epochs is very close to the architecture obtained after selecting only one operation per layer with the highest importance value.

TABLE 3. PSNR/GFLOPs values of search procedure with and without entropy regularization. Models were searched in different settings A, B, and C.

Training settings	without Entropy	with Entropy
A	27.99 / 111	**28.10** / 206
B	28.00 / 30	**28.12** / 19
C	27.92 / 61	**28.11** / 321

V. COMPARISON WITH OTHER NAS APPROACHES AND LIMITATIONS

A. LIMITATIONS

The main limitation of this work is a lack of proper benchmarking with other NAS methods. As demonstrated in [40] there are no NAS-Benchmark for SR problem with a unified search space due to the fact that NAS procedures usually encompasses a big number of interdependent components which may affect final performance. Moreover, there are no NAS-Benchmark for quantized models as well.

To ensure a fair comparison, it would be necessary to adapt existing methods to the SR domain or add quantization. However, this adaptation would require significant modifications and could potentially lead to strong divergence from existing methods. Therefore, our evaluation is limited to three methods for NAS in the SR domain that operate in full precision settings such as AGD [11], TrilevelNAS [12] and [24]. For quantization we compare our approach with the Two-fold Procedure (as described in Section I) where we use our method to find architectures and mixed-precision [9] and uniform [4], [5] quantization methods from the literature. While there are works on quantization aware search such as [26] the authors demonstrate different metrics and use different search space. We discuss all these details in the following section.

Obtaining the full Pareto frontier requires running the same experiment multiple times, which are computationally and time demanding. In Figure 7, all most right points (within one experiment/color) have 0 hardware penalty. It clearly shows that limited search space creates an upper bound for the top model performance. Therefore, results for our search space do not fall within the same BitOps range as SRResNet.

Even with the lack of comparison with other NAS approaches our work brings valuable observations and our results demonstrate effectiveness of novel techniques within our own settings, which can further facilitate development in this area.

B. COMPARISON WITH OTHER NAS APPROACHES

1) FULL-PRECISION NAS FOR SR

Here we examine the quality of our procedure for full precision NAS without ADQ and SAN, since both were

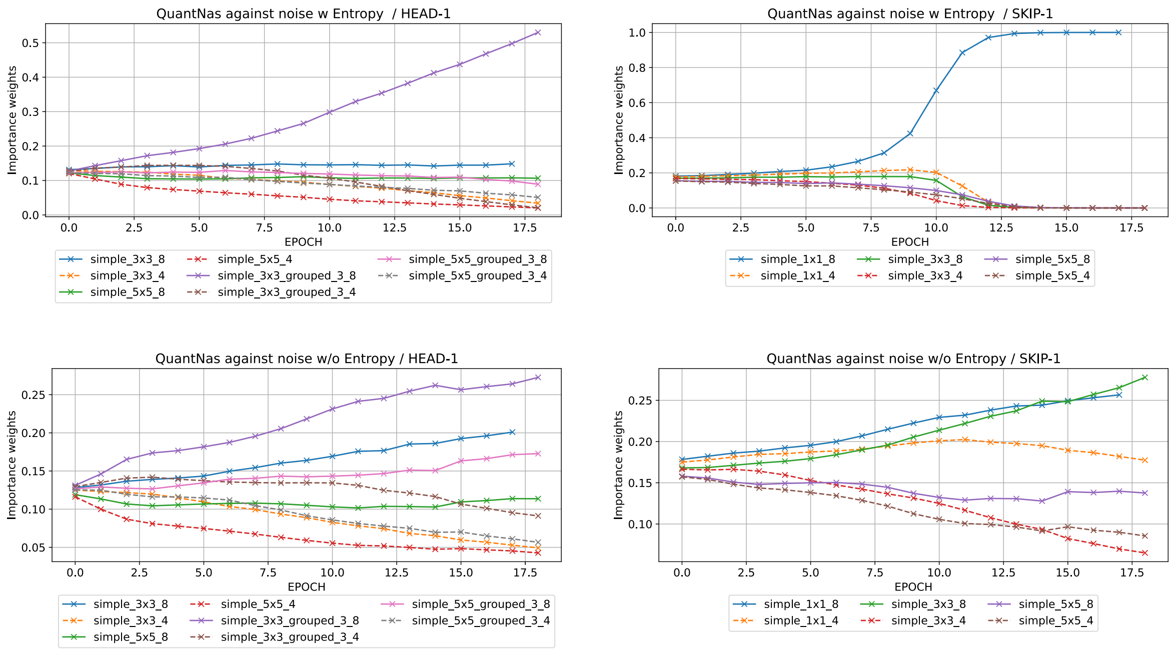

FIGURE 9. Dynamics of importance weights for different operations through epochs for QuantNAS. For 8 and 4 bits, we use solid and dashed lines, respectively. Usage of entropy sparsification (top) allows for selecting a single most relevant block with high importance compared to variants without entropy sparsification (bottom).

developed for quantized models. The results are presented in Table 4.

Our search procedure with the *Basic search space* achieves comparable results with TrilevelNAS [12] with a relatively simpler search space design and about 5 times faster search time. However, our procedure with the RFDN-based search space is slightly inferior to DLSR. Both our approach and DLSR develop search spaces based on RFDN and apply DNAS, but there are some key differences:

- DLSR uses different search blocks, specifically separable convolutional layers, which have fewer parameters but are slower.
- DLSR uses an additional loss function - High Frequency Error Norm.
- In terms of the search procedure, similar to us, DLSR uses DARTS but authors do not apply entropy regularization.
- Finally, different datasets are used for training the models.

While it is difficult to conclude which component affected the results the most, we believe that the reduction in GFLOPs was most likely achieved due to the usage of separable convolutions, and better performance was due to the different dataset and possibly the additional loss function.

Our best performing full precision architecture with *Basic search space* was found with a hardware penalty of value $1e-3$. This architecture is depicted in Appendix Figure 17. Visual examples of the obtained super-resolution pictures are presented in Figure 19 for Set14 [34], Set5 [34], Urban100 [36], and Manga109 [37] with scale factor 4.

2) MIXED-PRECISION NAS FOR SR

In terms of Mixed-precision NAS for SR our work very closely aligns with [26]. While it is difficult to make the direct comparison due to usage of different search space, datasets and computational efficiency metrics, it is not even necessary.

Our approach without SAN, ADQ and entropy regularization, would lead to the same procedure with minor differences in hyperparameters and with the main difference in search space. So the methods can be compared within the same search.

VI. DISCUSSION

We demonstrate that with SAN, we are able to achieve a close approximation of direct quantization. Additionally, SAN produces superior results, potentially attributed to its differentiable reparametrization. Moreover, we believe that application of SAN during NAS is not limited to only quantization but also can be extended to other problems for example noise and adversarial robustness.

However, the stochastic nature introduced by randomly sampled quantization noise makes the SAN procedure less stable. Interestingly, our findings reveal that when combined with ADQ, SAN consistently delivers improved outcomes, whereas using SAN alone may result in suboptimal solutions. In the subsequent section (Section E), we conduct a thorough analysis of the architectures and delve into further insights.

We have successfully showcased the efficacy of our procedure in two search spaces, indicating its potential applicability to other search spaces as well. *RFDN search space* consistently outperforms our *Basic search space* due to the incorporation of various technical solutions, including

TABLE 4. Quantitative results of PSNR-oriented models with SR scaling factor 4 for Set14 dataset. * results are from paper [12].

Method	GFLOPs	PSNR	Search cost
SRResNet	166.0	28.49	Manual
RFDN	27.14	28.37	Manual
AGD*	140.0	28.40	1.8 GPU days
Trilevel NAS*	33.3	28.26	6 GPU days
DLSR*	20.41	28.67	2 GPU days
QuantNAS Our SP	29.3	28.22	1.2 GPU days
QuantNAS RFDN	23.4	28.3	1.6 GPU days

Residual Feature Distillation. It is worth noting that the development of such search spaces requires considerable effort. However, our results demonstrate that it is possible to design a customized search space based on an existing architecture, resulting in improved quality and efficiency.

We found that our procedure is sensitive to hyperparameters. In particular, optimal coefficients for hardware penalty and entropy regularization can vary across different search settings. Moreover, we expect that there is a connection between optimal coefficients for the hardware penalty, entropy regularization, and search space size. Different strategies or search settings require different values of hardware penalties. Applying the same set of values for different settings might not be the best option, but it is not straightforward as how to determine them beforehand.

VII. CONCLUSION

In this work we demonstrate that approximating direct quantization with QN during architecture search is favorable and has many advantages.

We introduce an entire framework to search for efficient hardware friendly SR architectures and release our code. Our search procedure includes: (1) The entropy regularization to avoid co-adaptation in supernets during differentiable search; (2) differentiable SAN procedure; and (3) ADQ module which helps to alleviate problems caused by Batch Norm blocks in super-resolution models.

We demonstrate the versatility of our method by applying it to various search spaces. In particular, we conduct experiments using search space based on the computationally efficient SR model RFDN.

Our experiments clearly indicate that the joint NAS and mixed-precision quantization procedure outperforms using NAS or mixed-precision quantization alone.

Furthermore, when compared or approach with Two-Fold procedure where we used mixed-precision quantization with EdMIPS [9], our search consistently yields better solutions.

APPENDIX A
UNIFORM QUANTIZATION

Quantization is a mapping that converts a range of full-precision values into a discrete range of values allowing usage of integer arithmetic and reduced memory consumption. For example, Figure 10 depicts a uniform mapping with the quantization scale size $\Delta = \frac{1}{4}$ of float values from the interval (0, 1) into integer values.

In our work we apply uniform symmetric quantization with channel-wise quantization step size $\mathbf{\Delta}$ for **weights**. In this case, computations of quantization, dequantization and estimation of $\mathbf{\Delta}$ are performed for the bit-width b as below:

$$q_{\min} = -2^{b-1}, \quad q_{\max} = 2^{b-1} - 1 \tag{10}$$

$$\text{clamp}(x; q_{\min}, q_{\max}) = \max(q_{\min}, \min(x, q_{\max})) \tag{11}$$

$$\Delta = (\Delta_1, \dots, \Delta_n)^{\mathrm{T}}, \quad \Delta_i = \frac{\alpha_i}{q_{\max}} \tag{12}$$

$$\mathbf{W}_i^{\text{int}} = \text{clamp}\left(\left\lfloor \frac{\mathbf{W}_i}{\Delta_i} \right\rceil; q_{\min}, q_{\max}\right) \tag{13}$$

$$\mathbf{W} \approx Q(\mathbf{W}) = \Delta \odot \mathbf{W}^{\text{int}} \tag{14}$$

where Δ_i is the scale factor for i channel of $\mathbf{W}_i$, $\mathbf{W}^{int}$ denotes the matrix of the quantized weights, $\odot$ is element-wise product. We need to note, that while Δ can be computed, we follow LSQ [5] and set it as a trainable parameter. One of the advantages of learnable step size is that it helps to avoid outlier values (q_{min}, q_{max}) which may occur in the during training weights.

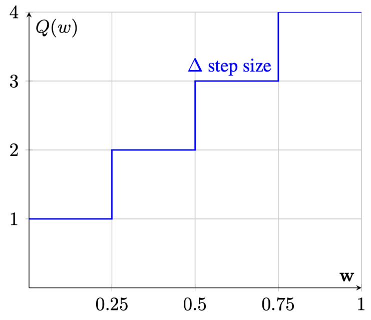

FIGURE 10. Uniform quantization step function with real valued one dimensional w and integer valued $Q(w)$.

APPENDIX B
HALF WAVE GAUSSIAN QUANTIZATION (HWGQ)

Models are more sensitive to activation quantization rather than weight quantization [32] and it is well known that non-uniform quantization may lead to higher compression rates with better accuracy, therefore, for activations quantization we use non-uniform HWGQ method [4]. The idea is illustrated in Equation 15, here each t_i is precomputed with Lloyd's algorithm considering Gaussian distribution of zero mean and unit variance. Number of intervals is obtained as $I = 2^{bit}$.

$$Q(\mathbf{x}) = \begin{cases} \mathbf{q_i} \text{ if } x \in (t_i, t_{i+1}], \\ \mathbf{0} \text{ if } x \leq 0 \end{cases} \tag{15}$$

Since we use ReLU activation functions we are interested in positive values only.

APPENDIX C
STRAIGHT THROUGH ESTIMATOR

STE can be described in two steps:

- Obtain quantized weights $Q(\mathbf{W})$ from the real-valued parameters $\mathbf{W}$ with some quantization function Q, which is usually is non differentiable.
- Compute gradients at quantized weights $Q(\mathbf{W})$ and update real valued weights $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \tau \nabla f(Q(\mathbf{W}))$

STE makes a particular choice of a quantization function to obtain the discrete weights from the real-valued weights. This approximation can be justified in some settings [41] but in general the reasons behind its effectiveness are unknown.

APPENDIX D TECHNICAL DETAILS

During the search phase, we consider architectures with a fixed number of channels for each operation unless channel size is changed due to operations properties. For *Basic search space*, number of channels is set to 36, and for *RFDN search space* number of channels is set to 48. The search is performed for 20 epochs. To update the weights of the supernet, we utilize the following hyperparameters: batch size of 16, an initial learning rate (lr) of 1e-3, a cosine learning rate scheduler, SGD with a momentum of 0.9, and a weight decay of 3e-7. When updating the alphas, we employ a fixed lr of 3e-4 and no weight decay.

During the training phase, an obtained architecture is optimized for 30 epochs with the following hyperparameters: batch size 16, initial lr 1e-3, and lr scheduler with the weight decay of 3e-7.

We did not perform any hyper parameter search to obtain the parameters.

Algorithm 1 QuantNAS algorithm

```
1:  Initialize parameters $W$ and edge values $\alpha$
2:  for $iteration = 1, 2, \ldots, N$ do
3:      Add QN to $W$ as in 6 and 5
4:      Compute the loss function $L(\alpha)$ as in III-H
5:      Run backpropagation to get derivatives for $\alpha$
6:      Update $\alpha$
7:      Add QN to $W$ as in 6 and 5
8:      Compute the loss function $L_1(W)$
9:      Run backpropagation to get derivatives for $W$
10:     Update $W$
11: end for
12: Select edges with the highest $\alpha$
13: Train the final architecture from scratch
```

APPENDIX E ANALYSIS OF FOUND ARCHITECTURES

We conducted an analysis of architectures discovered within our *Basic search space*, and exemplary architectures are presented in Figures 17 and 18 for full precision and quantized models, respectively. Our observations indicate that architectures with higher performance tend to have higher bit values for the first and last layers. Notably, the quantization of the first layer has a significant impact on model performance, as it results in substantial information loss due to the quantization of incoming signal. Additionally, we found that intermediate body blocks typically exhibit lower bit values.

APPENDIX F RANDOM SEARCH

In Figure 13, we conducted a comparison between our procedure and randomly sampled architectures on the *Basic search space*. The results indicate that our procedure significantly outperforms random search. Notably, there are two distinct clusters above and below 26 PSNR line, which correspond to models with 8- and 4-bit quantization of the first layers.

APPENDIX G ENTROPY SCHEDULE

For entropy regularization, we gradually increase the regularization value α according to Figure 11, and for the first two epochs, regularization is zero. Entropy regularization is multiplied by an initial coefficient and coefficient factor. Initial coefficients are 1e-3 and 1e-4 for experiments with full precision and the quantization-aware search.

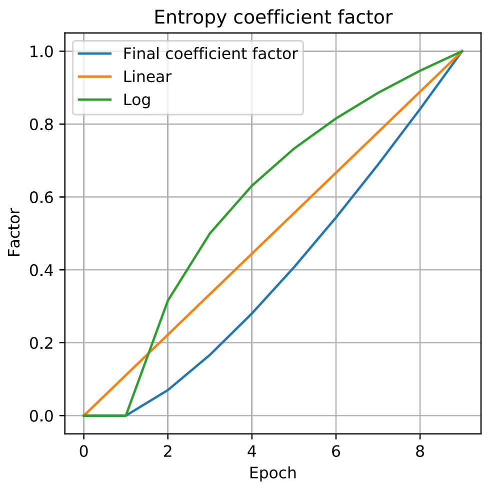

FIGURE 11. Entropy coefficient regularization is a product of log and linear functions.

APPENDIX H SCALING SR MODELS WITH INITIAL UP-SAMPLING

To maintain good computational efficiency, it is common for SR models to operate on down-sampled images and then up-sample them with some up-sampling layers. This idea was introduced first in ESPCN [30]. Since then, there were not many works in the literature exploring SR models on initially up-scaled images.

Therefore, we were interested in how this approach scales in terms of quality and computational efficiency given arbitrary many layers. Results are presented in Figure 12. We start with one fixed block, similar to our body block in Figure 3, and then increase it by one each time. We compare our results with SRResNet [38] and SRCNN [42]. As we can see, SRResNet [38] operates on down-scaled images

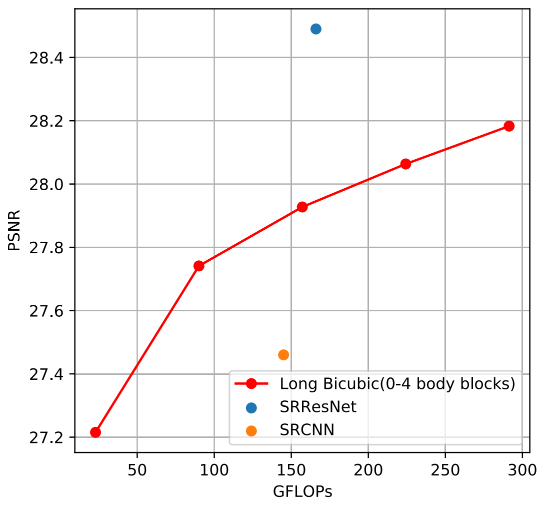

FIGURE 12. **Black point is the original SRCNN [42], and blue point is SRResNet [38]. For Long Bicubic, we initially upscale an image with bicubic interpolation and then add an efficient block found in our experiments. The block consists of 3 convolutions layers with 32 filters and is added 1, 2, 3, 4 times. PSNR is reported on Set14.**

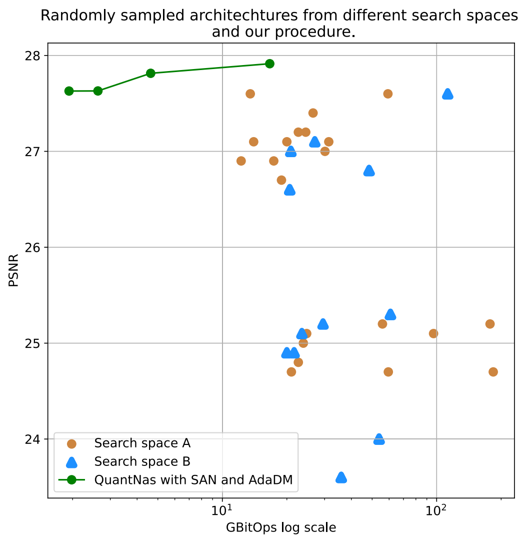

FIGURE 13. **Randomly sampled architectures from two search spaces. The search spaces are described in the corresponding section. PSNR was computed on Set14 and BitOPs for image size 256 × 256. We observe that two search spaces provide slightly different results with random sampling. Results in green for architecture search were obtained with Big search space - A. Two clusters above and below 26 PSNR line attribute to 8- and 4-bit quantization of the first layer.**

and yields better results given the same computational complexity.

APPENDIX I
DETERMINING THE IMPORTANCE OF BITS AND CHANNELS

In Figure 14, we conducted an analysis to determine the relative importance of bits and channels in model performance. Our findings are as follows. The results reveal that using 8-bit quantization yields comparable performance to that of 16- and 32-bit quantization, while providing marginal computational efficiency gains. This suggests that

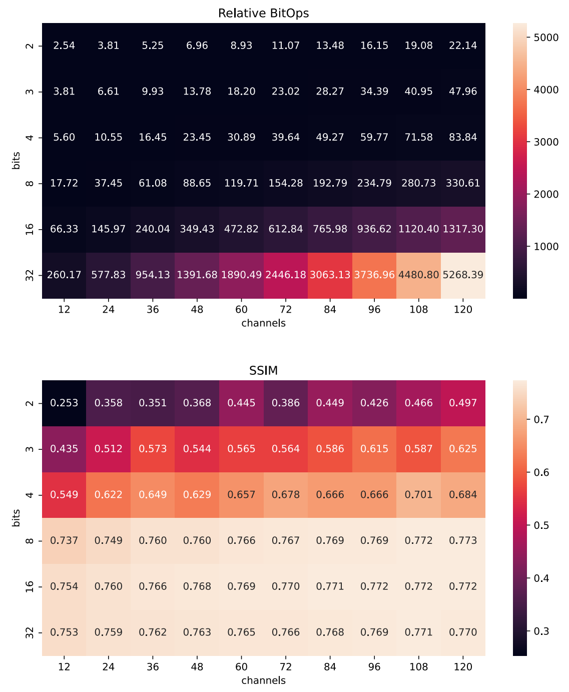

FIGURE 14. **Performance comparison with different bit values and number of channels on the ESPCN model. All layers are uniformly quantized, except for the first layer, which is fixed with 32 bits. BitOps values are scaled and relative values are reported.**

8-bit quantization is a viable option for achieving efficient performance. Additionally, we observed that increasing the number of channels in a model comes at a higher cost and is not practical. As a result, it is essential to explore alternative optimization approaches to enhance model performance, rather than rely solely on channel scaling. One such approaches is feature distillation used in RFDN.

Considering these findings, we decided not to include the number of channels in our search space, since it has a less significant impact on model performance.

APPENDIX J
SEARCH SPACE
A. SINGLE-PATH SEARCH SPACE

There are several ways to select directed acyclic sub-graph from a supernet. DARTS [10] uses Multi-Path strategy - one node can have several input edges. Such a strategy makes a

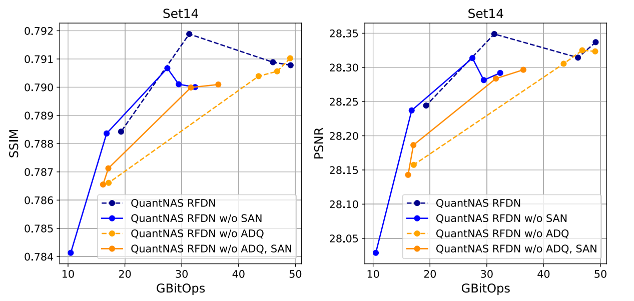

FIGURE 15. **Comparison of results from Fig. 7 for different metrics: SSIM and PSNR. As we can see, each metric gives a similar result.**

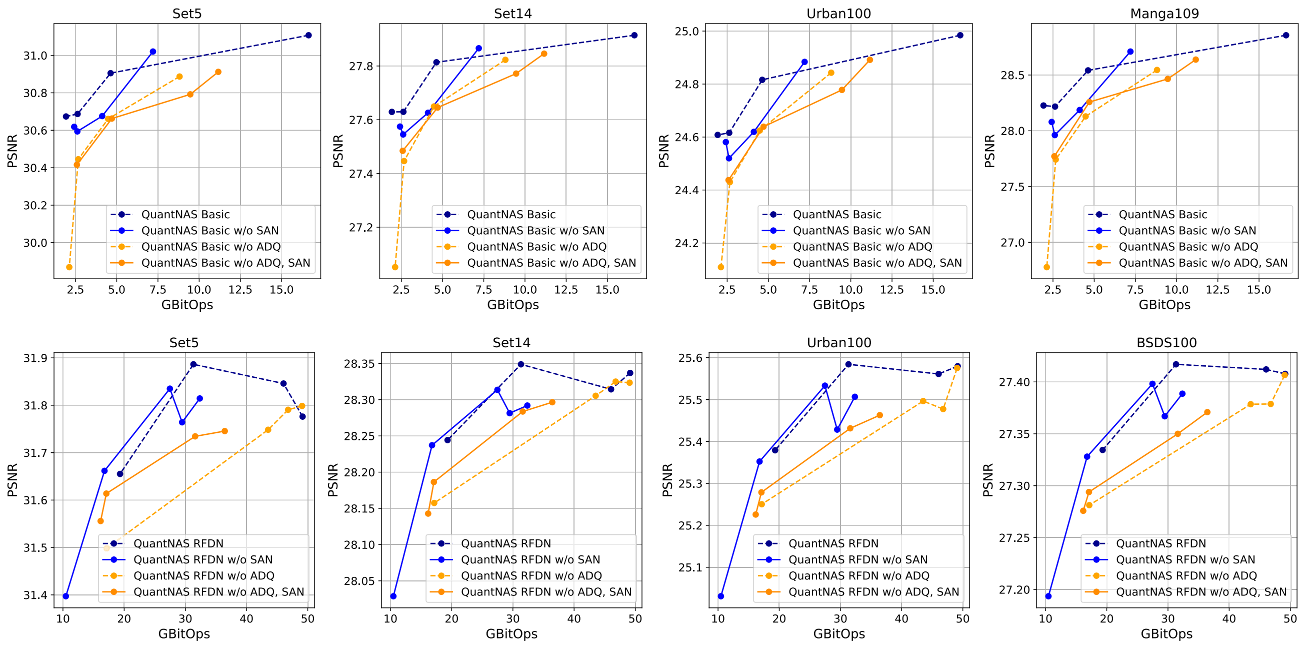

FIGURE 16. The same as Figure 7 but for different datasets.

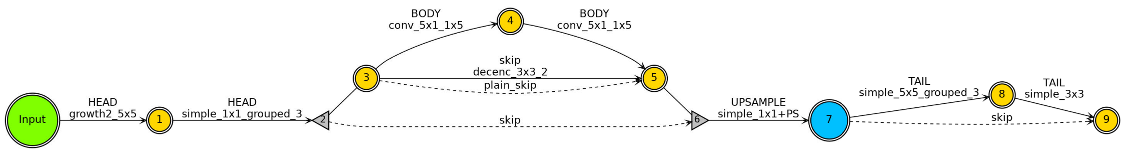

FIGURE 17. Our best FP (full precision) architecture, 29.3 GFLOPs (image size 265 × 265), PSNR: 28.22 dB. PSNR was computed on Set14. Body block is repeated three times for both architectures.

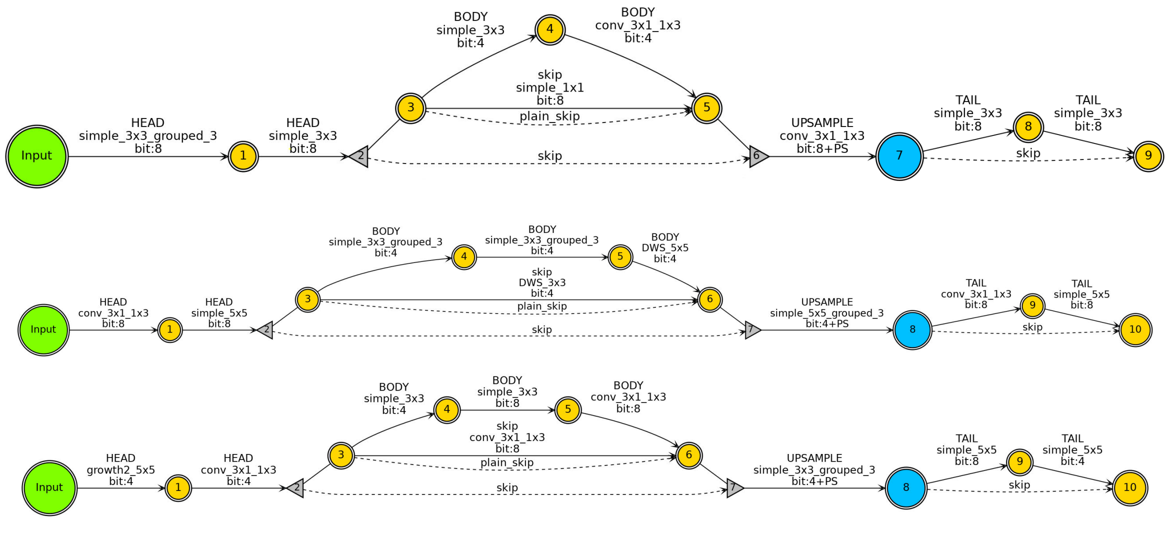

FIGURE 18. Examples of quantized architechtures. PSNR from top to bottom: 27.814 dB, 27.2 db, 24.8 db. On the top is our quantized architecture (body 3), more details are given in Table 1. PSNR was computed on Set14 with scale 4. Body block is repeated three times for all the architectures. Architecture on the bottom was sampled randomly.

search space significantly larger. In our work, we use Single-Path strategy - each searchable layer in the network can choose only one operation from the layer-wise search space (Figure 1). It has been shown in FBNet [2] that simpler Single-Path approach yields are comparable with Multi-Path approach results for classification problems. Additionally, since it aligns more with SR search design in our work, we use Single-Path approach.

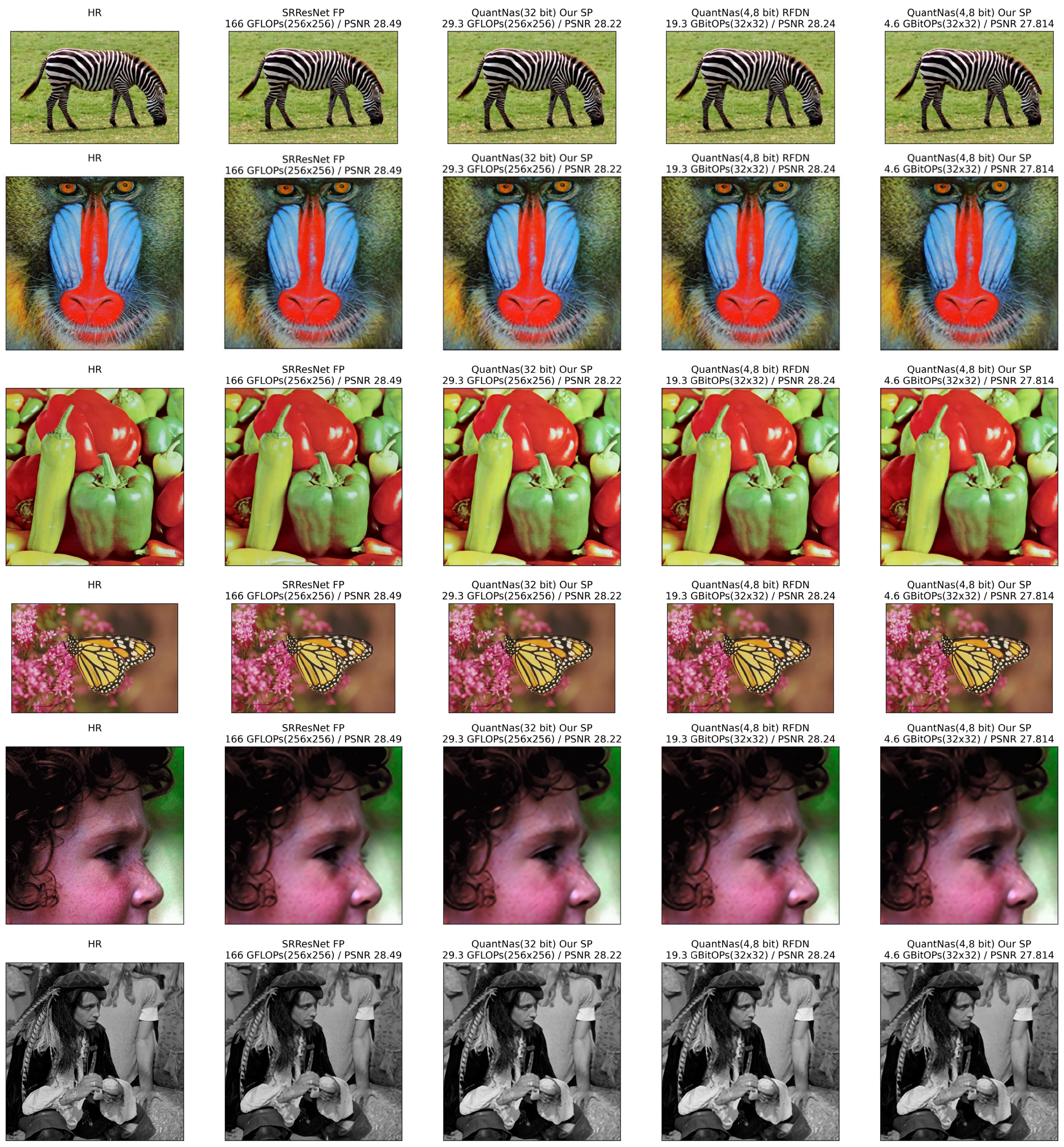
HR
SRResNet FP
166 GFLOPs(256x256) / PSNR 28.49
QuantNas(32 bit) Our SP
29.3 GFLOPs(256x256) / PSNR 28.22
QuantNas(4,8 bit) RFDN
19.3 GBitOPs(32x32) / PSNR 28.24
QuantNas(4,8 bit) Our SP
4.6 GBitOPs(32x32) / PSNR 27.814

FIGURE 19. Visual comparison of results for Set14. Better view in zoom. Note: we present results for quantized models with the body block repeated 3 times. Model with the body block repeated 6 times has better PSNR values (see in Table 1). Our SP denotes *Basic search space*.

We have a fixed number of channels for all the layers unless specified. For detailed operations description, we refer to our code.

B. SEARCH SPACE (BIG - A)

This search space was used for full precision experiments, unless specified. Possible operations block-wise:

- **Head** 8 operations: simple 3×3, simple 5×5, growth2 5×5, growth2 3×3, simple 3×3 grouped 3, simple 5×5 grouped 3, simple 1×1 grouped 3, simple 1×1;
- **Body** 7 operations: simple 3×3, simple 5×5, simple 3×3 grouped 3, simple 5×5 grouped 3, decenc 3×3 2, decenc 5×5 2, simple 1×1 grouped 3;
- **Skip** 4 operations: decenc 3×3 2, decenc 5×5 2, simple 3×3, simple 5×5;
- **Upsample** 12 operations: conv 5×1 1×5, conv 3×1 1×3, simple 3×3, simple 5×5, growth2

5 × 5, growth2 3 × 3, decenc 3 × 3 2, decenc 5 × 5 2, simple 3 × 3 grouped 3, simple 5 × 5 grouped 3, simple 1 × 1 grouped 3, simple 1 × 1;

- **Tail** 8 operations: simple 3 × 3, simple 5 × 5, growth2 5 × 5, growth2 3 × 3, simple 3 × 3 grouped 3, simple 5 × 5 grouped 3, simple 1 × 1 grouped 3, simple 1 × 1;

C. SEARCH SPACE (SMALL - B)

This search space was mainly used for all Quantization experiments Possible operations block-wise:

- **Head** 5 operations: simple 3 × 3, simple 5 × 5, simple 3 × 3 grouped 3, simple 5 × 5 grouped 3;
- **Body** 4 operations: conv 5 × 1 1 × 5, conv 3 × 1 1 × 3, simple 3 × 3, simple 5 × 5;
- **Skip** 3 operations: simple 1 × 1, simple 3 × 3, simple 5 × 5;
- **Upsample** 4 operations: conv 5 × 1 1 × 5, conv 3 × 1 1 × 3, simple 3 × 3, simple 5 × 5;
- **Tail** 3 operations: simple 1 × 1, simple 3 × 3, simple 5 × 5;

Conv 5 × 1 1 × 5 and conv 3 × 1 1 × 3 are depth-wise separable convolution convolutions. For operations description, we refer to our code.

APPENDIX K
RESULTS ON OTHER DATASETS

In Figure 16, we provide quantative results obtained on different test datasets: Set14 [34], Set5 [35], Urban100 [36], Manga109 [37] with scale 4.

In Figure 19, we provide with visual results for quantized and full precision models.

ACKNOWLEDGMENT

(Egor Shvetsov and Dmitry Osin contributed equally to this work.)

REFERENCES

[1] S. Anwar, S. Khan, and N. Barnes, "A deep journey into super-resolution: A survey," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, May 2021.

[2] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10726–10734.

[3] M. Shen, F. Liang, R. Gong, Y. Li, C. Li, C. Lin, F. Yu, J. Yan, and W. Ouyang, "Once quantization-aware training: High performance extremely low-bit architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5320–5329.

[4] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5406–5414.

[5] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," 2019, *arXiv:1902.08153*.

[6] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[7] A. Défossez, Y. Adi, and G. Synnaeve, "Differentiable model compression via pseudo quantization noise," 2021, *arXiv:2104.09987*.

[8] B. Widrow, I. Kollar, and M.-C. Liu, "Statistical theory of quantization," *IEEE Trans. Instrum. Meas.*, vol. 45, no. 2, pp. 353–361, Apr. 1996.

[9] Z. Cai and N. Vasconcelos, "Rethinking differentiable search for mixed-precision neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2346–2355.

[10] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–13.

[11] Y. Fu, W. Chen, H. Wang, H. Li, Y. Lin, and Z. Wang, "AutoGAN-distiller: Searching to compress generative adversarial networks," in *Proc. ICML*, 2020, pp. 1–12.

[12] Y. Wu, Z. Huang, S. Kumar, R. S. Sukthanker, R. Timofte, and L. Van Gool, "Trilevel neural architecture search for efficient single image super-resolution," in *Proc. Comput. Vis. Pattern Recognit.*, 2021, pp. 1–13.

[13] A. Bulat, B. Martinez, and G. Tzimiropoulos, "Bats: Binary architecture search," in *Proc. ECCV*, 2020, pp. 309–325.

[14] Y. Tian, C. Liu, L. Xie, J. Jiao, and Q. Ye, "Discretization-aware architecture search," *Pattern Recognit.*, vol. 120, Dec. 2021, Art. no. 108186.

[15] J. Chang, Y. Guo, G. Meng, S. Xiang, and C. Pan, "Data: Differentiable architecture approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[16] Y. Wu, A. Liu, Z. Huang, S. Zhang, and L. Van Gool, "Neural architecture search as sparse supernet," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 12, pp. 10379–10387.

[17] Y. Yang, H. Li, S. You, F. Wang, C. Qian, and Z. Lin, "ISTA-NAS: Efficient and consistent neural architecture search by sparse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1–11.

[18] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[19] J. Shin, J. So, S. Park, S. Kang, S. Yoo, and E. Park, "NIPQ: Noise proxy-based integrated pseudo-quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 3852–3861.

[20] Z. Hui, X. Wang, and X. Gao, "Fast and accurate single image super-resolution via information distillation network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 723–731.

[21] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight image super-resolution with information multi-distillation network," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 2024–2032.

[22] J. Liu, J. Tang, and G. Wu, "Residual feature distillation network for lightweight image super-resolution," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Glasgow, U.K. Springer, Jan. 2020, pp. 41–55.

[23] K. Zhang, M. Danelljan, Y. Li, R. Timofte, J. Liu, J. Tang, G. Wu, Y. Zhu, X. He, W. Xu, "Aim 2020 challenge on efficient super-resolution: Methods and results," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Glasgow, U.K. Springer, Aug. 2020, pp. 5–40.

[24] 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.

[25] K. Teja Chitty-Venkata, Y. Bian, M. Emani, V. Vishwanath, and A. K. Somani, "Differentiable neural architecture, mixed precision and accelerator co-search," *IEEE Access*, vol. 11, pp. 106670–106687, 2023.

[26] K. T. Chitty-Venkata, A. K. Somani, and S. Kothandaraman, "Searching architecture and precision for U-Net based image restoration tasks," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 1989–1993.

[27] M. Tan, W. Gao, H. Li, J. Xie, and M. Gong, "Universal binary neural networks design by improved differentiable neural architecture search," *IEEE Trans. Circuits Syst. Video Technol.*, early access, May 9, 2024, doi: 10.1109/TCSVT.2024.3398691.

[28] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "ReActNet: Towards precise binary neural network with generalized activation functions," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K. Springer, Aug. 2020, pp. 143–159.

[29] A. Ignatov et al., "Real-time quantized image super-resolution on mobile NPUs, mobile AI 2021 challenge: Report," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2525–2534.

[30] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1874–1883.

[31] J. Liu, J. Tang, and G. Wu, "AdaDM: Enabling normalization for image super-resolution," 2021, *arXiv:2111.13905*.

[32] D. Becking, M. Dreyer, W. Samek, K. Müller, and S. Lapuschkin, "ECQ^X: Explainability-driven quantization for low-bit and sparse DNNs," in *Proc. Int. Workshop Extending Explainable AI Beyond Deep Models Classifiers*. Springer, 2020, pp. 271–296.
[33] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1122–1131.
[34] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. Curves Surf.* Springer, 2010, pp. 711–730.
[35] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2012, pp. 135.1–135.10.
[36] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5197–5206.
[37] A. Fujimoto, T. Ogawa, K. Yamamoto, Y. Matsui, T. Yamasaki, and K. Aizawa, "Manga109 dataset and creation of metadata," in *Proc. 1st Int. Workshop Comics Anal., Process. Understand.*, Dec. 2016, pp. 1–5.
[38] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.
[39] J. B. Diederik and P. Kingma, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 126–135.
[40] K. T. Chitty-Venkata, M. Emani, V. Vishwanath, and A. K. Somani, "Neural architecture search benchmarks: Insights and survey," *IEEE Access*, vol. 11, pp. 25217–25236, 2023.
[41] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–9.
[42] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. ECCV*, 2014, pp. 184–199.

EGOR SHVETSOV received the B.S. and M.S. degrees in applied physics and material science from Tomsk Polytechnic University, in 2008.

Since 2020, he has been a Research Engineer with the Applied AI Center, Skolkovo Institute of Science and Technology. His recent research interests include neural architecture search and computationally efficient deep learning.

DMITRY OSIN received the B.S. degree in applied mathematics and informatics from Lomonosov Moscow State University, Moscow, Russia, in 2021.

Since 2020, he has been a Research Engineer with the Skolkovo Institute of Science and Technology. His main research interests include neural architecture search, quantization, model compression, representation learning, super resolution, and sequential data.

ALEXEY ZAYTSEV was born in Kharkiv, Ukraine. He graduated from MIPT, in 2012. He received the Ph.D. degree in mathematics from IITP, RAS, in 2017. He is currently an Assistant Professor with the Skolkovo Institute of Science and Technology. His research interests include development of new methods for sequential data, Bayesian optimization, and embeddings for weakly structured data. In his master's thesis, he proposed a modification of Bayesian approach for linear regression that allows an automated feature selection.

IVAN KORYAKOVSKIY received the M.Sc. degree in computer science from the Computer Vision Laboratory, Seoul National University, Seoul, Republic of Korea, and the Ph.D. degree from Delft Biorobotics Laboratory, Faculty of Mechanical Engineering, Delft University of Technology, researching machine learning techniques for the control of the walking robots. He is currently a Research Scientist with Toloka AI. His research interests include on-device machine learning, reinforcement learning, computer vision, and autonomous agents.

VALENTIN BUCHNEV received the M.Sc. degree in applied mathematics and informatics from Moscow Institute of Physics and Technology, in 2023. From 2021 to 2023, he was a Research Engineer with Huawei Technologies Company Ltd. Since 2023, he has been a Research and Development Engineer with Yandex Self Driving Group LLC. His research interests include neural network quantization and computationally efficient deep learning.

ILYA TROFIMOV received the M.Sc. degree in theoretical physics from Moscow State University, in 2006, and the Ph.D. degree in computer science from the Federal Research Center "Computer Science and Control," RAS, in 2018.

He is currently a Research Scientist with the Skolkovo Institute of Science and Technology. His main research interests include large-scale machine learning, AutoML, neural architecture search, and generative adversarial networks.

EVGENY BURNAEV received the M.Sc. degree in applied physics and mathematics from Moscow Institute of Physics and Technology, in 2006, and the Ph.D. degree in foundations of computer science from the Institute for Information Transmission Problem, RAS, in 2008.

He is currently a Full Professor with the Skolkovo Institute of Science and Technology and the Director of the Skoltech Applied AI Center. His current research interests include regression based on Gaussian processes and kernel methods for multi-fidelity surrogate modeling and optimization, deep learning for 3D data analysis and manifold learning, online sequence learning for prediction, and non-parametric anomaly detection.