A Cost-Effective Interpolation for Multi-Magnification Super-Resolution

KUAN-YU HUANG, SURAJ PRAMANIK, AND PEI-YIN CHEN, (Senior Member, IEEE)

1Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan
2Gogoro Inc., Taoyuan 33308, Taiwan
 Corresponding author: Pei-Yin Chen (pychen@mail.ncku.edu.tw)

This work was supported in part by the Ministry of Science and Technology at Taiwan under Grant MOST 108-2221-E-006-096-MY3, and in part by Qualcomm through a Taiwan University Research Collaboration Project.

ABSTRACT Super-Resolution (SR) was an important research topic, and SR methods based on Convolutional Neural Network (CNN) confirmed its groundbreaking performance. However, notably implementing the CNN model into resource-limited hardware devices is a great challenge. Therefore, we present a hardware-friendly and low-cost interpolation for Multi-Magnification SR image reconstruction. We follow our previous work, which is a learning-based interpolation (LCDI) with a self-defined classifier of image texture, and extend its original ×2 architecture to ×3 and ×4 architecture. Besides, the required pre-trained weights are reduced by the fusion scheme. Experimentally, the proposed method has only 75% lower pre-trained weights than LCDI. Compared to the related work OLM-SI (One linear learning mapping-SI), the run-time and quantity of pre-trained weights of the ×2 proposed method are at least 90% lower. Compared to CNN-based SR methods, the proposed method loses a little lower performance, but the evaluation of computational cost is much lower. In conclusion, the proposed method is cost-effective and a practical solution for resource-limited hardware and device.

INDEX TERMS Super-resolution, interpolation, multi-magnification, hardware-friendly, machine learning.

I. INTRODUCTION

Single image Super-Resolution (SISR) indicates generating the High-resolution image (HR) from a single Low-resolution image (LR), and we abbreviate SISR to SR in this paper. The conventional SR is based on image interpolation, such as Bilinear [1] or Bicubic [2], and many interpolation methods [3], [4], [5], [6], [7], [8], [9] are developed successfully. Winscale [3] is one of them, which computes the cover area of the pixels to determine interpolated weights instead of the distance of the pixels. Another is an extended linear interpolation [4], which is a low-complexity method and is regarded as a separable filter as well. Based on [5], edge-preserving interpolation (EPI) [6], [7] has an additional error compensation function to improve the quality of the scaled image. Likewise, to solve the aliasing effects; instead, Chen et al. [8], [9] employ pre-filters before Bilinear interpolation.

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

Conventional SR methods indicate image scaling or interpolation, which employ polynomial function, so they are limited on the quality of reconstructed image until a convolutional neural network (CNN) is applied for SR. SRCNN [10] is the first work, which applies CNN to the SR method, and improves the SR performance and quality of the reconstructed image to a brand-new stage. Later almost SR methods [11], [12], [13], [14], [15], [16], [17] were based on CNN. They continue to confirm better performance through many advanced CNN implementation skills, such as the Fast SRCNN (FSRCNN) [11], the first employ of Residual Net for SR [12], the design of Sub-Pixel Layer [13], Generative Adversarial Network for SR [14], Residual Dense Net for SR [15], Residual Channel Attention for SR [16], and Meta-SR [17] et al. Furthermore, Wang et al. [18] introduce and summarize almost classic and thrilling SR methods.

Since the hardware resource is limited, some studies have developed novel interpolation-based SR methods [19], [20], [21], [22], [23], [24], combining machine learning instead...
of the CNN-based SR method. They employ interpolation architecture and exploit pre-learned weights stored in a look-up table to calculate interpolated pixels instead of the polynomial function. They are also known as learning-based interpolations, and their critical advantages include low computational complexity and competitive quality of the reconstructed image compared to the CNN-based SR methods.

Jo et al. [19] proposed a practical and fast learning-based interpolation and called it SR-LUT. In the training phase, they train a deep CNN-based SR network, whose output values are transferred to the look-up table. They quickly get the pre-computed HR output values in the test phase by querying the LR input values.

Another well-known learning-based interpolation is Super Interpolation (SI) [20], as shown in Fig.1. One of the critical novelties of SI is a designated classifier, which discriminates the feature of each patch of the image, and another is that the new pixels are calculated by the pre-learned weights. After each patch of the image is classified, the pre-trained weights based on its feature are taken from the weight in the table. The weights in the table are pre-trained by numerous image data and regarded as the optimal interpolated coefficient based on a different feature of the patch. Since SI is the interpolation-based SR method, it can only require a low computational cost. In addition, its designated classifier and pre-learned weights bring competitive quality to the reconstructed image. OLM-SI [21], which is SI’s extended work, employs a more efficient classifier, and its performance is much better than SI and is similar to SRCNN.

Since SI is introduced as a hardware-friendly method, the hardware implementation of SI (HSI) is provided in [22] as well, confirming that SI only costs very few hardware resources to achieve competitive performance. Based on [22], Bae et al. [23] developed a more simplified classifier to achieve similar performance to [22]. Since the local binary pattern classification is used in [23], we call it LBPC-SI in this paper.

Likewise inspired by SI, we observe that it is still ameliorable; we proposed a modified method of SI in our previous work [24], called Learning-based Cross-directional Interpolation (LCDI). The main difference between SI and LCDI is the dimension of the referenced patches. As shown in Fig.2, the referenced patches of SI and LCDI are presented in (a) and (b), respectively. SI uses the 2-D interpolation, and LCDI applies the 1-D interpolation. The main advantage of 1-D interpolation is lower computational complexity than 2-D. Moreover, we proposed an efficient classifier for 1-D patches to improve the quality of the reconstructed image. To confirm our method is suitable for hardware, we implement it on ASIC and FPGA. From the experimental results, compared to [22], the required hardware resource of our LCDI-3 is lower, and the evaluated metric of quality of the reconstructed image is similar. Compared to the hardware implementation of the CNN-based method [25], [26], [27], [28], our LCDI-7+ only required much fewer hardware resources without much loss of the quality of the reconstructed image. Note that the comparison results with [27] are not provided in [24], but we get the same conclusion mentioned above.

Here, we briefly introduce the studies [25], [26], [27], [28] on implementing CNN-based SR methods on hardware, most proposed modified and lightweight models. Based on FSRCNN-s, the deconvolution accelerators of SR methods are presented in [25] and [26]. Lee et al. [27] replace the Deconvolution Layer of FSRCNN-s with the Sub-Pixel Layer so that the cost of their weight is about 60% of that of FSRCNN-s and achieve almost similar performance. Kim et al. [28] designed a novel CNN-based SR model and its hardware: Real-time SRCNN (RTSRCNN).

In [24], we proposed an efficient SR method and its hardware implementation, but we only developed ×2 SR architecture, and there is a problem that the quantity of pre-learned weights is too much. Therefore, in this work, we develop ×3 and ×4 SR architecture based on the LCDI-7+ and attempt to reduce the required pre-trained weights. The main contributions of the proposed method are listed as follows:

1) Based on LCDI-7+, we apply the fusion scheme to reduce the required pre-learned weights. The quantity of pre-trained weights of the proposed ×2 SR architecture is a quarter of that of LCDI-7+ without noticeable compromising performance.

2) The original classifier of LCDI-7+ is improved. Since the number of its classes is descended from 2025 to 625, the quantity of pre-learned weights is reduced by around two-thirds. Although the performance decrease with the reduction of the number of classes, it isn’t affected notably. Besides, the new
classifier’s smaller number of comparison operations brings less run-time. After weighing carefully, we think the proposed new classifier is more cost-effective.

3) We modify LCDI-7+ to implement the ×3 and ×4 SR architecture. This method is universal and wholly compared with the current methods, which provide multi magnification results, such as OLM-SI, SR-LUT, and FSRCNN-s.

Here, we briefly present the comparison results with three main competitors, OLM-SI, SR-LUT, and FSRCNN-s, and comment on the proposed method. Among them, OLM-SI and SR-LUT are the kinds of learning-based interpolation. The proposed ×2, ×3, and ×4 SR method requires much less run-time and quantity of pre-trained weights than the OLM-SI without noticeable loss of performance. Compared to the SR-LUT, the computational cost of the proposed SR method is slightly higher. However, in terms of performance, we get better results regardless of magnification. For CNN-based SR methods, we take lightweight models as the competitors. The lightweight models, even such as FSRCNN-s or more simplified [27], still require considerable computational complexity. Therefore, the proposed method has a notable advantage in terms of computational cost.

In summary, we follow our previous work LCDI-7+, consider the universality and balance between performance and cost, and develop the multi-magnification and cost-effective SR method. Experimental results confirm the proposed method achieves competitive performance with relatively few costs and run-time compared to those recent SR methods.

The rest of this paper is organized as follows. The related work is introduced in Section II. The proposed method is elaborated in Section III, including the fusion of pre-learned weights, ×3 and ×4 SR architecture, 1-D classifier, and the pre-training and interpolation process. The experimental results are presented in Section IV. Finally, the discussion and conclusions of this study are stated in Section V and Section VI, respectively.

II. RELATED WORK

As mentioned in the introduction above, SI [20] and LCDI [24] are the base related works in this paper, and they can be regarded as 2-D and 1-D interpolations, respectively. The main advantage of 1-D interpolation architecture is the reduction of computational complexity compared to the 2-D interpolation architecture. As shown in Fig.2(b), 1-D interpolation is regarded as a two-stage process, which includes vertical and horizontal interpolation. The process from gray spot to yellow spot is a vertical interpolation, and the horizontal interpolation indicates the one from yellow spot to orange spot. The general SR method is the process that generates the HR image directly from the LR image, and the two-stage SR process requires an extra Intermediate Resolution (IR) image between LR and HR image. As shown in Fig.3, we employ vertical interpolation to get an IR image from an LR image and exploit horizontal interpolation to generate an HR image from an IR image. Among both interpolations, the designated classifier is applied first, and then the corresponding pre-learned weights are obtained and used to compute the value of pixels of IR and HR images.

Based on a detailed statement in [24], the critical reason, why the computational complexity of 1-D interpolation is lower, is the reusable pixels of the IR image. Back to Fig.2(b), the gray spots and orange spots mean pixels of LR and HR images, respectively, and the yellow spots are regarded as pixels of IR images. Since the pixels of the IR image are reused to generate the pixels of the HR image, the total number of operations is reduced. Based on the results in [24], the computational-complexity comparison of 2-D and 1-D interpolation is shown in TABLE.1, while the 7 × 7 block is referenced. For 2-D interpolation, each pixel of an HR image costs 49 multiplications and 48 additions. For 1-D interpolation, each pixel of IR and HR image requires only seven multiplications and six additions.

Take ×2 architecture as an example. While LR and HR images are size M×N and 2M×2N, respectively, there are 6×M×N times operation of 1-D interpolation. In sum, there are 10.5 (= 7 × (6/4)) multiplications and 9 (= 7 × (6/4)) additions for the average computation of each pixel of the final HR image. Note that the practical number of operations may be slightly different based on the different sizes of LR images.

LCDI-7+ has confirmed its low computational complexity, but only ×2 SR architecture is provided. Therefore, we follow it and develop this architecture for ×3 and ×4 SR methods. Moreover, we think the quantity of pre-learn weights is too much, so we employ a fusion of pre-learn weights. Experimentally, our method has a 75% lower quantity of pre-learn weights and is more cost-effective than LCDI-7+.

III. PROPOSED METHOD

In this Section, we elaborate on our method, including the fusion of pre-learned weights and extended ×3 and ×4 SR architecture. Moreover, we state how to tackle the fusion of weights in the pre-training and interpolation process and the simplified classifier of 1-D interpolation.

A. THE FUSION OF PRE-LEARNED WEIGHTS

Since the quantity of pre-learned weights is too much in LCDI, we attempt to merge its four groups of pre-learned weights into one. An example of ×2 SR architecture is shown in Fig.4, including vertical and horizontal interpolation. The number label #1, #2, #3, and #4 indicate four groups of pre-learned weights in LCDI-7+ [24], but we attempt to fuse

| Method       | SI [20] | LCDI-7+ [24] |
|--------------|---------|--------------|
| Addition / Multiplication | 48 / 49 | 9 / 10.5     |
them. The fusion method is shown in Fig.4 as well, and we observe that #1 and #2 groups of weights are invertible, so we only require a single group of weights in vertical interpolation. Besides, for horizontal interpolation, the following property of invertibility in vertical interpolation, and we reuse a single group of weights in #3 and #4. Especially, we find the pre-learned weights of vertical and horizontal interpolation are even shared, which indicates the only single group of weights \( \{W_0, W_1, W_2, W_3, W_4, W_5, W_6\}^{2\times} \) is required for \( \times 2 \) SR architecture.

Taking LCDI-7+ with 2025 classes as a comparator, TABLE 2 presents a brief performance comparison, including the number of classes, the quantity of pre-learned weights, and the objective evaluated metric for average peak signal-to-noise ratio (PSNR) in testing dataset Set5. Experimentally, the quality of the reconstructed image through the fusion of weights only slightly decreases than the one through four groups of weights but has a 75% reduction in the quantity of weight. Therefore, we conclude that the method through the fusion of weights is more cost-effective, and we develop later \( \times 3 \) and \( \times 4 \) SR methods through a fusion scheme.

### TABLE 2. The brief comparison of our previous and current work for evaluated items, including number of classes, quantity of weights, and PSNR in testing dataset Set5.

| Method          | LCDI-7+ [24] | Proposed |
|-----------------|-------------|----------|
| Number of Classes | 2,025      | 2,025    |
| Quantity of Weights | 56,700   | 14,175   |
| PSNR            | 36.32      | 36.29    |

**B. THE \( \times 3 \) AND \( \times 4 \) SR ARCHITECTURE**

For the \( \times 2 \) SR method based on 1-D interpolation, one pixel of LR image can be generated to two pixels of IR image, and one pixel of IR image brings two pixels of HR image. Following the \( \times 2 \) SR architecture, we develop the \( \times 3 \) and \( \times 4 \) SR architecture. As shown in Fig.5, there are two stages in the proposed \( \times 3 \) and \( \times 4 \) architecture likewise. For \( \times 3 \) SR architecture, whatever vertical or horizontal interpolation, both are imaging processes from one pixel to three pixels. For \( \times 4 \) SR architecture, it is an imaging process from one pixel to four pixels in vertical and horizontal interpolation.

The proposed fusion of pre-learned weights is employed for \( \times 3 \) and \( \times 4 \) architecture as well. Since \( \times 3 \) and \( \times 4 \) SR architecture is not provided in LCDI-7+, we assume the condition that we use separate groups of pre-learned weights in \( \times 3 \) and \( \times 4 \) SR architecture. As shown in Fig.6, there would be six \#1~#6 and eight \#1~#8 groups of weights for \( \times 3 \) and \( \times 4 \) SR architecture, respectively.

However, if we employ the fusion of weights to \( \times 3 \) architecture, we need two groups of weights, which are labeled as \( \{W_{0,1}, W_2, W_3, W_4, W_5, W_6\}^{3\times} \) and \( \{W_0, W_1, W_2, W_3, W_4, W_5, W_6\}^{3\times} \) in the vertical and horizontal interpolation of Fig.6 (a) and (b). Cases \#1 and \#4 can share the first group of weights, and its inversion is reused in \#3 and \#6 cases. Then the second group of weights is reused in cases \#2 and \#5, which indicate the middle pixel. For the fusion of weights in \( \times 4 \) architecture, the cases \#1 and \#5 use the first group \( \{W_0, W_1, W_2, W_3, W_4, W_5, W_6\}^{4\times} \), and the cases \#4 and \#8 use its inversion. The cases \#2, \#3, \#6, and \#7 share the second group of weights \( \{W_0, W_1, W_2, W_3, W_4, W_5, W_6\}^{4\times} \) in the same way, as shown in Fig.6 (c) and (d).
We summarize the results with the fusion of weights in TABLE 3, the fusion of weights is obviously practical, although it may result in a slight loss of performance, as shown in TABLE 2. Note that there is a different reduction rate of pre-learned weights for different magnifications.

C. THE PRE-TRAINING AND INTERPOLATION PROCESS
To tackle the fusion of pre-trained weights, we re-design the pre-training process, which differs from LCDI-7+. Taking \( \times 4 \) architecture as an example in Fig.7, we regard the pixels of LR and HR images as the pairs of training data and labels since it is the process from one pixel to four pixels. In the first of the process, the images in the training dataset are extracted into vertical and horizontal pairs. Then, they are split to forward (green line) and backward (blue line) sub-pairs, which include seven pixels of LR image and two pixels of HR image, respectively.

Moreover, four groups of sub-pairs are regarded as the same group, then fed into the block of “Classifier” to discriminate to the corresponding category based on its property. At last, the sub-pairs in each category are trained separately through the block of “Machine Learning” to get the optimal weights of each category. Note that a linear regression problem with seven variables is solved in the block of “Machine Learning”, since there are seven inputs and two outputs, as (1), shown at the bottom of the next page, each input multiplies the corresponding coefficient, which is known as the weight, to generate two outputs. Therefore, 14 weights, which belong to two groups, are trained. The number of the categories is \( N \), and the total \( 14 \times N \) weights are pre-trained.

Finally, we briefly introduce the interpolation process. Following LCDI-7+, we employ two stages to up-scale the images, including vertical and horizontal interpolation, as shown in Fig.3. The main difference is that we didn’t require two different tables of weights and only accessed the same table in both stages. Likewise, we take \( \times 4 \) architecture as an example, which contains two-stage computation. The vertical interpolation is (2), as shown at the bottom of the next page, which includes computations of four IR pixels \( \text{Out}_{1R}, \text{Out}_{2R}, \text{Out}_{3R}, \text{Out}_{4R} \) through seven LR pixels \( \text{In}_{1L}, \text{In}_{2L}, \text{In}_{3L}, \text{In}_{4L}, \text{In}_{5L}, \text{In}_{6L} \), also referenced in Fig.6 (c). Next, (3), shown at the bottom of the next page, indicates the computation in horizontal interpolation, where we get four HR pixels \( \text{Out}_{1H}, \text{Out}_{2H}, \text{Out}_{3H}, \text{Out}_{4H} \) by seven IR pixels \( \text{In}_{1R}, \text{In}_{2R}, \text{In}_{3R}, \text{In}_{4R}, \text{In}_{5R}, \text{In}_{6R}, \text{In}_{7R} \), as shown in Fig.6 (d).

D. THE CLASSIFIER OF A 1-D INTERPOLATION
In this Section, we discuss the classifier of 1-D interpolation. In LCDI-7+, we adopt the granularity of the gradient of the contiguous pixels to discriminate the property of the 1-D patches. As shown in Fig.8 (a), we compute the four gradients \( G_0, G_1, G_2, G_3 \) of the contiguous pixels of the 1-D patches, then granularize the gradient value based on different grades. Besides, the required operations of this classifier are only four subtractions and some comparisons, which make it more hardware-friendly.

For LCDI-7+, the \( G_1, G_2 \) employ the rule of nine grades, as shown in Fig.8 (b), and the \( G_0, G_3 \) employ the rule of five grades, as shown in Fig.8 (c). The total number of classes is 2,025 (= \( 5 \times 9 \times 9 \times 5 \)). However, we think the quantity is still too much, and we would attempt to adopt the rule of five grades to \( G_0, G_1, G_2, G_3 \). The total number of classes can be reduced to 625 (= \( 5 \times 5 \times 5 \times 5 \)).

Table 4. presents the brief experimental results. The comparison of Bicubic, regarded as the Baseline, and the proposed method with 2025 and 625 classes are listed and called Proposed A and B, respectively. The evaluated item contains the number of classes, the quantity of pre-learned weights, and the objective evaluated metric for PSNR in testing dataset Set5. From this result, we think the proposed B with 625 classes is more cost-effective.
In this paper, we follow our previous work LCDI-7+, which is a low-complexity learning-based interpolation with a self-defined 1-D feature classifier of image texture, extending its \( \times 2 \) architecture to \( \times 3 \) and \( \times 4 \) architecture and reducing the required pre-trained weights by fusion scheme. Besides, two versions of the method with different 1-D classifiers are provided, called the Proposed A and B with 2025 classes and a more simplified 625 classes.

**IV. EXPERIMENTAL RESULT**

This Section reveals the experimental results and comparisons with other related studies. The investigations of the dataset are explained in Section IV.A, and then we describe the details of experimental results in Section IV.B.

**TABLE 5.** The brief comparison of the Proposed A version with used different training datasets for PSNR in testing dataset Set5.

| \( x \times 2 \) method | Proposed A | Proposed A |
|------------------------|------------|------------|
| Training Datasets      | T91, G100, and B200 | T91, G100, B200, and DIV2K |
| PSNR                   | 36.29      | 36.32      |

**A. SURVEY OF THE DATASET**

There are several datasets available for SR, such as T91, which contains 91 training images used in SRCNN [10]. General-100 (G100) dataset that consists of images introduced by FSRCNN [11]. Another well-known BSD500
dataset [29] is the JPEG-format images, and some studies take 200 images from it as BSD200 (B200). Recently, the widely used datasets in computer vision technology are ImageNet [30] and high-resolution images named DIV2K [31].

As observed in FSRCNN [11], the image quality is slightly better when trained by ImageNet than by T91 images and G100. Therefore, T91 and G100 image datasets contained enough variability of natural images, which can achieve the

### TABLE 6. The comparison of the various ×2 SR methods for some evaluated objective items, including storage cost, computational cost, and performance.

| Method   | Cost | Performance |
|----------|------|-------------|
|          | Storage | Computation | Set5 | Set14 | B100 |
|          | Classes | Weights | Multiplication | PSNR | SSIM | Time | PSNR | SSIM | Time | PSNR | SSIM | Time |
| Baseline | Bicubic | No | No | 6 | 33.66 | 0.9296 | - | 30.34 | 0.8690 | - | 29.54 | 0.8434 | - |
| SI       | SI [20] | 625 | 122K | 49 | 35.99 | 0.9511 | 0.4 | 31.93 | 0.9099 | 0.7 | 30.88 | 0.8806 | 0.5 |
|          | OLM-SI [21] | 1024 | 200K | 49 | 36.37 | 0.9526 | 0.5 | 32.17 | 0.9035 | 0.9 | 31.09 | 0.8836 | 0.6 |
|          | HSI [22] | 625 | 22.5K | 9 | 34.78 | - | - | 31.63 | - | - | 30.34 | - | - |
|          | LBPC-SI [23] | 256 | 9.2K | 9 | 34.96 | - | - | 31.35 | - | - | 30.39 | - | - |
| SR-LUT [19] | Version F | 4913 | 19.65K | 4 | 35.64 | 0.9475 | - | 31.88 | 0.8991 | - | 30.77 | 0.8787 | - |
|          | Version S | 83521 | 334K | 5 | 35.46 | 0.9466 | - | 31.73 | 0.8958 | - | 30.64 | 0.8750 | - |
| CNN-based | FSRCNN-s [11] | Single | 4K | 14K | 36.58 | 0.9532 | - | 32.28 | 0.9049 | - | 31.23 | 0.8866 | - |
|          | HED-SR [27] | Single | 2.56K | 0.6K | 36.49 | 0.9538 | - | 32.29 | 0.9053 | - | 31.18 | 0.8862 | - |
| Proposed | Version B | 625 | 4.3K | 15 | 36.24 | 0.9511 | 0.019 | 32.10 | 0.9020 | 0.043 | 31.01 | 0.8835 | 0.025 |
|          | Version A | 2025 | 14.7K | 15 | 36.32 | 0.9514 | 0.021 | 32.14 | 0.9024 | 0.047 | 31.06 | 0.8840 | 0.026 |

### TABLE 7. The comparison of the various ×3 SR methods for some evaluated objective items, including storage cost, computational cost, and performance.

| Method   | Cost | Performance |
|----------|------|-------------|
|          | Storage | Operation | Set5 | Set14 | B100 |
|          | Classes | Weights | Multiplication | PSNR | SSIM | Time | PSNR | SSIM | Time | PSNR | SSIM | Time |
| Baseline | Bicubic | No | No | 5.33 | 30.40 | 0.8686 | - | 27.55 | 0.7741 | - | 27.21 | 0.7389 | - |
| SI       | OLM-SI [21] | 1024 | 451K | 49 | 32.43 | 0.9050 | 0.3 | 29.01 | 0.8156 | 0.5 | 28.21 | 0.7804 | 0.4 |
| SR-LUT [19] | Version F | 4913 | 44.22K | 4 | 31.88 | 0.8947 | - | 28.72 | 0.8088 | - | 27.97 | 0.7734 | - |
|          | Version S | 83521 | 751.7K | 5 | 31.95 | 0.8969 | - | 28.73 | 0.8057 | - | 27.92 | 0.7690 | - |
| CNN-based | FSRCNN-s [11] | Single | 4K | 0.44K | 32.54 | 0.9055 | - | 29.08 | 0.8167 | - | 28.33 | 0.7815 | - |
| Proposed | Version B | 625 | 8.75K | 12 | 32.42 | 0.9048 | 0.013 | 28.94 | 0.8151 | 0.030 | 28.12 | 0.7810 | 0.018 |
|          | Version A | 2025 | 28.35K | 12 | 32.50 | 0.9058 | 0.015 | 29.01 | 0.8159 | 0.033 | 28.15 | 0.7817 | 0.020 |

### TABLE 8. The comparison of the various ×4 SR methods for some evaluated objective items, including storage cost, computational cost, and performance.

| Method   | Cost | Performance |
|----------|------|-------------|
|          | Storage | Operation | Set5 | Set14 | B100 |
|          | Classes | Weights | Multiplication | PSNR | SSIM | Time | PSNR | SSIM | Time | PSNR | SSIM | Time |
| Baseline | Bicubic | No | No | 5 | 28.43 | 0.8109 | - | 26.01 | 0.7023 | - | 25.96 | 0.6678 | - |
| SI       | OLM-SI [21] | 1024 | 802K | 49 | 30.20 | 0.8557 | 0.2 | 27.25 | 0.7457 | 0.4 | 26.77 | 0.7057 | 0.3 |
| SR-LUT [19] | Version F | 4913 | 78.6K | 4 | 29.77 | 0.8429 | - | 26.99 | 0.7372 | - | 26.57 | 0.6990 | - |
|          | Version S | 83521 | 1336K | 5 | 29.82 | 0.8478 | - | 27.01 | 0.7355 | - | 26.53 | 0.6953 | - |
| CNN-based | FSRCNN-s [11] | Single | 4K | 0.25K | 30.11 | 0.8499 | - | 27.19 | 0.7423 | - | 26.84 | 0.7089 | - |
| Proposed | Version B | 625 | 8.75K | 10.625 | 30.19 | 0.8571 | 0.011 | 27.13 | 0.7447 | 0.023 | 26.66 | 0.7065 | 0.014 |
|          | Version A | 2025 | 28.35K | 10.625 | 30.29 | 0.8590 | 0.012 | 27.19 | 0.7457 | 0.026 | 26.69 | 0.7073 | 0.015 |

As observed in FSRCNN [11], the image quality is slightly better when trained by ImageNet than by T91 images and G100. Therefore, T91 and G100 image datasets contained enough variability of natural images, which can achieve the
same results as other larger datasets. In recent years, the CNN-based SR method has chosen to use DIV2K dataset, which provides high-resolution images.

After surveying many research papers, the T91, G100, B200, and DIV2K image sets (100 validation images) were carefully chosen as training datasets in this paper. Unlike LCDI [24], we add 100 images from DIV2K as training datasets and get the light improvement of the objective evaluated metric. In TABLE 5., we compare the Proposed A version with used different training datasets for PSNR.

B. EXPERIMENTAL RESULTS

Most of the research papers evaluate the up-scaled image quality by two metrics such as the PSNR and the structural similarity index measure (SSIM), so we compute and evaluate our experimental result by these two metrics as well. And Set5, Set14, and B100 are used as testing datasets. Every image in the testing dataset is down-scaled by Bicubic depending on magnification factors 2, 3, 4, and subsequently up-scaled by our proposed method as well as other methods and computed numerical value of PSNR and SSIM. The evaluated process was implemented on MATLAB R2021.b. Furthermore, the Python and Keras library was used to train datasets in the pre-learning process.

These experimental results of ×2, ×3, and ×4 SR methods are shown in TABLE 6, 7, and 8, which contain storage cost, computational cost, and performance. We regarded the quantity of pre-learned weights as the storage cost and roughly evaluated the operation of the multiplication as the main computational cost. The detail about the calculation of the operation number of the proposed methods can be referenced in [24]. Next, the performance contains the quality of image and run-time, which is tested by MATLAB R2021.b on an Intel i7-11700K CPU 3.6 GHz and RAM 32GB without any GPU.

Besides, some descriptions of experimental results are stated here. Because recent SR studies only evaluate the Y channel on objective metrics, we follow this process, which first transfers the input image from the RGB color domain to the YCbCr domain and only implements the proposed method on the Y channel. Besides, the run-time only includes the processing time of the Y channel. Note that the proposed method requires time to load pre-learned weights, which is excluded from the evaluation of run-time. A special condition for evaluation of run-time on MATLAB is discussed in Section V. The last concern is that the generated images by the proposed method lose a few pixels of the border; the generated ×2, ×3, and ×4 images are required to be cut 6, 8, 10 pixels of the border respectively.

In TABLE 6, 7, and 8, the proposed methods are compared with four kinds of SR techniques studies 1) conventional interpolation, 2) SI-based methods, 3) SR-LUT [19], and 4) CNN-based SR methods. The Bicubic method is the conventional interpolation and is regarded as the comparative baseline of almost SR studies. Study SI [20], OLM-SI [21], HSI [22], and LBPC-SI [23], which are considered the SI-based SR methods, and the primary competitors by us. Two versions F and S of SR-LUT, also learning-based interpolation, are considered our competitors. Compared with CNN-based SR methods, we choose the lightweight model FSRCNN-s [11] and [27]. Based on the title of the article [27], we abbreviate it HED-SR in this Section. Proposed A and B represent two versions with identical interpolation architecture but different 1-D classifiers and pre-learned weights.

Since many versions of improved Bicubic are proposed, we assume a lower bound of the computational cost of it straightly. Considering that 16 pixels must be referenced in Bicubic, we use 1-D interpolation with four referenced pixels to evaluate its computational cost. From the results, the required computational cost of the proposed methods is close to that of Bicubic, but ours achieves a noticeable improvement in the quality of the SR image.

For SI-based methods, which is a 2-D interpolation, OLM-SI presents the results of the ×2, ×3, and ×4 SR, but SI, HSI, and LBPC-SI only provide ×2 SR method. For the evaluated cost, the class numbers of the four methods are given in their original papers. The weights number of SI, OLM-SI, and LBPC-SI and the computational cost of the four methods may not be provided clearly. We speculate these values in TABLE 6, 7, and 8 from the description in...
their original paper. Besides, these values of computational cost only include the operation units of interpolation; if the operating units of the classifier are considered, this cost may increase.

Compared with the OLM-SI, our main competitor, the proposed methods obtain close to PSNR and SSIM with obvious reduction of storage and computational cost. Besides, our method is at least ten times faster on average than OLM-SI for different magnifications, and the run-time is approximately in line with the computational cost. Another benefit of ours is the reduction of pre-learned weights, which is confirmed by the experimental results. SI is the previous version of OLM-SI; it applied a smaller number of classes and got worse performance. Based on SI, HSI applies an identical classifier but only references the $3 \times 3$ patches instead of the $7 \times 7$ patches to calculate the interpolated pixels. Due to the employ of a smaller patch size, it reduces the computational cost, resulting in degradation of performance. LBPC-SI, which employs $3 \times 3$ patches likewise, simplifies the classifier of HSI and achieves similar performance with HSI. Compared with HSI and LBPC-SI, the proposed methods require a bit more storage and computational cost but get much better performance.

Next, SR-LUT is denoted as learning-based interpolation, but its architecture is different from SI widely. SR-LUT requires even less multiplication than Bicubic. However, considering the size of the look-up table, version F seems to be much less than version S. In comparison with SR-LUT, they require a bit less operation than the proposed methods. However, regardless of magnification, Proposed A and B have notable advantages considering the performance.

Compared with CNN-based SR methods, we take FSRCNN-s and HED-SR, lightweight CNN-based SR methods, as the primary competitors. Implementing CNN-based SR methods on hardware is challenging, so most hardware studies of CNN-based SR methods [25], [26], [27], [28] implement lightweight models. Although we don’t provide hardware implementation in this paper, we compare the evaluated number of operations as computational costs. From this result, we think the proposed methods have much less computational cost than FSRCNN-s or HED-SR and only lose a little performance in PSNR. Besides, our previous work [24] has confirmed this architecture is hardware-friendly and cost-effective compared to others.
Next, the comparison for the actual image, which is regarded as the subjective term, is presented in Fig. 9, Fig. 10, and Fig. 11. The detail of the actual image is stated; we transfer the input image from the RGB color domain to the YCbCr domain first and only implement the proposed method on the Y channel. The Cb and Cr channels are up-scaled by using Bicubic, and the YCbCr domain is converted to an RGB color domain. In Fig. 9, the “zebra” image from the Set14 dataset is taken as a sample to compare various ×2 SR methods, including Bicubic, SR-LUT, CNN-based FSRCNN-s, and the proposed methods. The comparison of ×3 SR methods is displayed in Fig.10., where the “monarch” image from the Set5 dataset is chosen. Finally, the “baby” image from the Set5 dataset is used to compare ×4 SR methods in Fig.11. From the subjective comparison, the SR images generated by the proposed method are marginally sharper and brighter than those obtained with Bicubic and FSRCNN-s. However, we observe the phenomenon that there is exactly a little distortion in the diagonal area of the image, which is generated by the proposed methods. This is discussed in Section V thoroughly.

V. DISCUSSION
In this Section, first, we discuss the special issue of the proposed method and take the “monarch” image from the Set14 dataset as an example. Fig. 12 presents that there is exactly a little distortion in the diagonal area of the image, which is generated by the proposed ×2 SR method, compared to the CNN-based methods. However, we found the down-scaled image by using Bicubic resulted in a slight distortion of the edge. Therefore, we experiment with employing the proposed method to up-scale the image from the original image instead of the down-scaled image by using Bicubic. There is a negligible distortion in the diagonal area of the image, which is shown in Fig. 13. In conclusion, we speculated that the slight distortion of the image of the proposed method is attributed to the down-scaled image by using Bicubic. Besides, we consider the verification flow of up-scaling the down-scaled image is just for fair and objective comparison, but not for the real situation. Moreover, the actual image in Fig. 13 showed that the edge of our result was sharper than that of FSRCNN-s [11]. Therefore, we thought our method had practical benefits when the proposed method was applied to the real application of image up-scaling or SR.

Another issue about the calculation of run-time is discussed here. We measure the run-time by using “tic” and “toc” functions on MATLAB. After a program is compiled on MATLAB, the run-time in its first testing is higher than in its later testing, and TABLE 9 presents the first, second, third, and fourth tested values of run-time for ×2 SR in testing dataset Set5. From our observation, the run-time becomes a stable value from the third test, so we adopt the third tested run-time as an experimental result in TABLE 6, TABLE 7 and TABLE 8.

VI. CONCLUSION
In this paper, we propose a hardware-friendly and cost-effective interpolation for Multi-Magnification ×2, ×3, and ×4 SR image reconstruction. The main benefit of our method is low computational complexity and the achievement of competitive quality of the image. The proposed method exploits 1-D interpolation, in which operations are lower than 2-D interpolation. Besides, to achieve the finer quality of the image, we design an efficient 1-D classifier and use machine learning to get more optimal interpolated weights. Finally, we use the fusion concept to reduce the number of pre-training weights. Experimentally, compared to the related work OLM-SI [21], which uses 2-D interpolation, the run-time and quantity of pre-trained weights of the proposed method reach an average of 90% lower for different magnifications. Compared with the recent work SR-LUT [19], there are pros and cons. The computational cost of the proposed SR method is slightly higher than theirs, but our performance is notably better. Compared to lightweight CNN-based SR methods, the proposed method has even less evaluation computational cost but only results in a little loss of performance. Notably, implementing the CNN model into resource-limited hardware devices is a great challenge, but the proposed method is more hardware-friendly. In conclusion, the proposed method is cost-effective and a practical solution for resource-limited hardware and device.
ACKNOWLEDGMENT
The authors would like to thank Dr. Yunqiang (Charles) Chen from Qualcomm, anonymous reviewers, and editors for their valuable comments and suggestions to improve the quality of the paper.

REFERENCES
[1] J. A. Parker, R. V. Kenyon, and D. E. Troxel, “Comparison of interpolation methods for image resampling,” IEEE Trans. Med. Imag., vol. MI-2, no. 3, pp. 31–39, Sep. 1983.
[2] H. S. Hou and H. C. Andrews, “Cubic splines for image interpolation and digital filtering,” IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-26, no. 6, pp. 508–517, Dec. 1978.
[3] C.-H. Kim, S.-M. Seong, J.-A. Lee, and L.-S. Kim, “Winscale: An image-scaling algorithm using an area pixel model,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 6, pp. 549–553, Jun. 2003.
[4] C.-C. Lin, M.-H. Sheu, H.-K. Chiang, C. Liaw, and Z.-C. Wu, “The efficient VLSI design of BI-CUBIC bicubic interpolation for digital image processing,” in Proc. IEEE Int. Symp. Circuits Syst., May 2008, pp. 480–483.
[5] C. C. Lin, M. H. Sheu, H. K. Chiang, C. Liaw, Z. C. Wu, and W. K. Tsai, “An efficient architecture of extended linear interpolation for image processing,” J. Inf. Sci. Eng., vol. 26, no. 2, pp. 631–648, 2010.
[6] C.-C. Huang, P.-Y. Chen, and C.-H. Ma, “A novel interpolation chip for real-time multimedia applications,” IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 10, pp. 1512–1525, Oct. 2012.
[7] P. Chen, S. Lin, and P. Chen, “Hardware implementation of an image interpolation method with controllable sharpness,” J. Inf. Sci. Eng., vol. 34, no. 1, pp. 51–64, Jan. 2018.
[8] S.-L. Chen, H.-Y. Huang, and C.-H. Luo, “A cost-low high-quality adaptive scalar for real-time multimedia applications,” IEEE Trans. Circuits Syst. Video Technol., vol. 21, no. 11, pp. 1600–1611, Nov. 2011.
[9] S.-L. Chen, “VLSI implementation of an adaptive edge-enhanced image scalar for real-time multimedia applications,” IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 9, pp. 1510–1522, Sep. 2013.
[10] C. Dong, “Learning a deep convolutional network for image super-resolution,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2014, pp. 1–10.
[11] C. Dong, C. C. Loy, and X. Tang, “Accelerating the super-resolution convolutional neural network,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 1–10.
[12] J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 1–5.
[13] W. Shi, J. Caballero, F. Huszar, 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. 1–4.
[14] C. Ledig, “Photo-realistic single image super-resolution using a generative adversarial network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2017, pp. 1–10.
[15] Y. Zhang, “Residual dense network for image super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 2472–2481.
[16] Y. Zhang, “Image super-resolution using very deep residual channel attention networks,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 286–301.
[17] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, “META-SR: A magnification-arbitrary network for super-resolution,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 1–6.
[18] Z. Wang, J. Chen, and S. C. H. Hoi, “Deep learning for image super-resolution: A survey,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 10, pp. 3365–3387, Oct. 2021.
[19] Y. Jiao and S. J. Kim, “Practical single-image super-resolution using look-up table,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 691–700.
[20] J.-S. Choi and M. Kim, “Super-interpolation with edge-orienation-based mapping kernels for low complex upsaling,” IEEE Trans. Image Process., vol. 25, no. 1, pp. 469–483, Jan. 2016.
[21] J.-S. Choi and M. Kim, “Single image super-resolution using global regression based on multiple local linear mappings,” IEEE Trans. Image Process., vol. 26, no. 3, pp. 1300–1314, Mar. 2017.
[22] Y. Kim, J.-S. Choi, and M. Kim, “2X super-resolution hardware using edge-orientation-based linear mapping for real-time 4K UHD 60 fps video applications,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 65, no. 9, pp. 1274–1278, Sep. 2018.
[23] S.-H. Bae, J.-H. Bae, A. Maqueet, M. S. Monira, and L. Kim, “Cost-efficient super-resolution hardware using local binary pattern classification and linear mapping for real-time 4K conversion,” IEEE Access, vol. 8, pp. 224383–224393, 2020.
[24] Y.-H. Shiau, K.-Y. Huang, P.-Y. Chen, and C.-Y. Kuo, “A low-cost hardware design of learning-based one-dimensional interpolation for real-time video applications at the edge,” IEEE J. Emerg. Sel. Topics Circuits Syst., vol. 11, no. 4, pp. 677–689, Dec. 2021.
[25] J.-W. Chang, K.-W. Kang, and S.-J. Kang, “An energy-efficient FPGA-based deconvolutional neural networks accelerator for single image super-resolution,” IEEE Trans. Circuits Syst. Video Technol., vol. 30, no. 1, pp. 281–295, Jan. 2020.
[26] W. Mao, J. Lin, and Z. Wang, “FDNA: Fast convolution architecture for deconvolutional network acceleration,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 8, pp. 1867–1880, Aug. 2020.
[27] S. Lee, S. Joo, H. K. Ahn, and S.-O. Jung, “CNN acceleration with hardware-efficient dataflow for super-resolution,” IEEE Access, vol. 8, pp. 187754–187765, 2020.
[28] Y. Kim, J.-S. Choi, and M. Kim, “A real-time convolutional neural network for super-resolution on FPGA with applications to 4K UHD 60 fps video services,” IEEE Trans. Circuits Syst. Video Technol., vol. 29, no. 8, pp. 2521–2534, Aug. 2019.
[29] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in Proc. ICCV, vol. 2, Jul. 2001, pp. 416–423.
[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015.
[31] 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. 126–135.

KUAN-YU HUANG received the B.S. degree in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan, in 2011, where he is currently pursuing the Ph.D. degree in computer science and information engineering since 2020. His research interests include image processing, computer vision, machine learning, and embedded systems.

SURAJ PRAMANIK received the B.S. degree in computer science and engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 2019, and the M.S. degree in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan, in 2021. His research interests include image processing and embedded systems.

PEI-YIN CHEN (Senior Member, IEEE) received the B.S. degree in electrical engineering from the National Cheng Kung University, Tainan, Taiwan, in 1986, the M.S. degree in electrical engineering from The Pennsylvania State University, University Park, PA, USA, in 1990, and the Ph.D. degree in electrical engineering from the National Cheng Kung University, in 1999. He is currently a Distinguished Professor with the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include very large-scale integration chip design, video compression, fuzzy logic control, and gray prediction.