Area-Specific Convolutional Neural Networks for Single Image Super-Resolution

HONNANG ALAO¹, TAE SUNG KIM¹, AND KYUJOONG LEE²

¹Department of Computer and Electronic Engineering, Sun Moon University, Asan 31460, South Korea
²School of AI Convergence, Sungshin Women’s University, Seoul 02844, South Korea

Corresponding author: Kyujoong Lee (kyujoonglee@sungshin.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) under Grant 2021R1G1A1094961.

ABSTRACT The implementation of deep convolutional neural networks (CNN) in single image super-resolution (SISR) has been successful at improving restoration quality. However, due to analysis made in previous works, it is observed that missing details in low-resolution (LR) images mostly exist in high-frequency regions. Since CNN operates all regions of the low-resolution (LR) image equally, the computation redundancy is observed in the low-frequency area. We generate a gradient-based binary mask (decision mask) to discriminate the high-frequency areas from the low-frequency areas and apply two kinds of convolution to them separately. We propose an Area-Specific CNN (ASCNN) for super-resolution. It consists of high parameter convolutions and low parameter convolutions to process the high-frequency areas and low-frequency areas separately, which efficiently reduces the FLOPs (floating-point operation) while maintaining restoration quality. The settings for reduction are configurable and experimental results show that ASCNN achieves state-of-the-art performance with FLOPs reduction up to 40.1% / 37.0% / 34.0% for ×2/×3/×4 scale factors.

INDEX TERMS Area-specific convolutions, convolutional neural network, decision mask, sparse convolution, super-resolution.

I. INTRODUCTION Single Image Super-Resolution (SISR) is a computer vision process that aims at recovering high-resolution (HR) images from low-resolution (LR) images. It is implemented in cases where more image details are required. They include medical imaging [1], [2], surveillance systems [3], [4], and more. Moreover, trending technologies that improve display resolutions require images of their corresponding resolutions.

SISR is an ill-posed computer vision problem as there are several ways to restore an LR image. Traditional methods use 2-dimensional interpolation algorithms [5] to restore images, but the deep learning-based convolutional neural network (CNN) produces more accurate results, which are usually measured with the peak-to-peak signal ratio [6] (PSNR) metric and structural similarity index measure metric [7] (SSIM).
redundant computation. They also show that smooth areas in images require less computation compared to areas with higher frequencies. As a result, a tile-based selective super-resolution (TSSR) [15] was proposed to divide input images into numerous tiles. Each tile is processed with either a DNN (deep neural network) or a shallow CNN, which is given as “TSSRN-I” and “TSSRN-s” respectively in their paper. In essence, they process certain tiles of an input image differently, using more computation on the tiles with higher frequency. Nevertheless, high and low frequencies exist in each tile, which means that computation redundancy is still present in every tile.

To overcome this downside, we introduce an algorithm to process each area of an image separately at the pixel level. In this paper, we show that area-specific convolution applied to the low-frequency areas and the high-frequency areas separately can reduce computational costs. Firstly, we analyze the areas of an image that require more computation and how to efficiently extract those high-frequency areas. We then use separate convolutional filters on the high-frequency areas and the low-frequency areas, where filters of fewer parameters are used on the low-frequency areas. As shown in Fig. 1., when compared with \( \times 2 \) upscaling factor, our method achieves state-of-the-art (SOTA) PSNR performance with reduced FLOPs compared to other SOTA networks.

Our main contributions are 1.) Creating a decision mask that indicates high-frequency and low-frequency areas. 2.) We propose an Area-specific convolutional neural network (ASCNN) for the SISR algorithm. 3.) With experimental results, we show that the computational cost can be reduced efficiently without a significant drop in performance.

Further sections of this paper are organized as follows. Related works provide an insight into super-resolution past solutions while stating out their underlying problem. Section III shows a detailed analysis of the problem and proposes ASCNN as a novel solution. Experimental results and benchmarking are shown in chapter IV while chapter V concludes the paper.

II. RELATED WORKS

In recent years, convolutional neural networks for image generation are explored. These include image inpainting [16], image deblurring [17] and GAN (generative adversarial networks)-based algorithms [18], [19], [20]. SISR also falls in this category, as the input domain and the output domain are both images.

The SRCNN [8] model upscales LR images via bicubic interpolation to match the HR size and proposes a simple 3-layer model to generate more accurate SR outputs. Kim et al. introduced a very deep SR (VDSR) [21] network which has a depth of 20 layers. It utilized a residual network algorithm [22] and significantly improved the restoration accuracy in SISR.

FSRCNN (fast super-resolution convolutional neural network) [23] and ESPCN (efficient sub-pixel convolutional neural network) [24] propose methods to upsample LR images at the end of the network. This reduces the number of FLOPs as the convolution operations are performed in the LR space. The results were also more accurate because they are learning-based upsampling techniques. FSRCNN has been widely used by SR hardware accelerators [25], [26] because of its high performance with relatively small FLOPs.

EDSR [9] and RCAN [10] are SOTA models that can handle large amounts of parameters to produce relatively higher performance. Nevertheless, networks with large amounts of parameters and high computational costs are impractical for implementation on mobile devices [27]. Lightweight SOTA models [11], [28], [29] modify their network structures to efficiently improve their performance with reduced parameters and fewer computations. However, this does not change the computational cost of a convolutional operation.

Aiming at reducing computational costs, Wang et al. proposed a sparse mask SR (SMSR) [14] technique that implements sparse convolution [30] in SISR. Through experimental analysis, they discover that redundant computations exist in SISR models, and utilize sparse convolution to skip unnecessary operations.

TSSR [15] also shows that not every area of an image needs to be operated with deep convolutional layers. Therefore, they divide the input images into several overlapping tiles to be processed differently. They then, train a policy network to classify the high-frequency tiles from low-frequency ones, where the low-frequency tiles are operated with a shallower network. This denotes the need for two or more networks that work independently of each other.

The combination of a classification network and an SR network (ClassSR) [31] was proposed by Kong et al. and its implementation is similar to TSSR [15]. They divide the input image into tiles classified into three categories, which are simple, medium, and hard. Depending on its category, each tile is passed into one of three separate networks.

TSSR [15], SMSR [14] and ClassSR [31] show that high-frequency areas of the images are more difficult to restore compared to smooth regions. With this analysis, we realize the need for a network that operates on high-frequency regions and low-frequency regions differently at a pixel level.

![Comparison between PSNR and FLOPs](https://example.com/comparison.png)
Each pixel from a low-frequency area needs to be restored with less computation. However, operating on a single-pixel independently is impractical as SISR convolutional layers use $3 \times 3$ filters that are dependent on surrounding pixels.

We propose an algorithm that fuses double convolutional filters to a single layer, where one of them possesses fewer parameters and fewer FLOPs than the other. It is an area-specific convolutional neural network. Experiments denote that by modifying the network settings, we can get the most efficient trade-off between computational costs and performance for a certain image restoration task.

III. PROPOSED METHOD
As mentioned in section II, analysis in previous works explains that high-frequency areas of an LR image are harder to restore, thus, need more computation compared to the low-frequency areas. We extract the high frequencies of LR images and use this information to efficiently implement our ASCNN algorithm to generate the SR images.

Our model operates on the grayscale channel only. Therefore, our decision mask is generated with the LR grayscale image, making it a single channel mask.

A. GENERATING DECISION MASKS
For simplicity and efficiency, we use a gradient-based [32] method to extract high-frequency information. We take advantage of the average pooling layer by setting the stride to 1, which produces the same effect as an average blurring. The overhead FLOPs are insignificant, because, an average pooling layer has a lesser computational burden and it is applied on a single (grayscale) channeled image.

The average blurring is applied to the LR image. Then the absolute difference between the LR image and its blurred version is computed, which results in a high-frequency map. It can be represented as shown in (1).

$$|LR_{ij} - f^{\text{blur}}(LR)| = HFM_{ij}$$  \hspace{1cm} (1)

where $f^{\text{blur}}$ represents average blurring on the LR image. $ij$ is used to indicate the pixel position of each element. $HFM$ is the high-frequency map. A threshold value is then used to separate the areas of high-frequency from low-frequency regions. It is expressed in (2).

$$M_{ij} = \begin{cases} 1, & \text{if } HFM_{ij} \geq \theta \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (2)

where $\theta$ is the threshold value. Hence, $M$ represents the decision mask. 1 denotes a pixel from the high-frequency area and 0 represents a pixel from the low-frequency area. This can be represented as $P^{HF}$ (pixel of high-frequency) and $P^{LF}$ (pixel of low-frequency) respectively. $\theta$ decides which pixels are $P^{HF}$ and which are $P^{LF}$. This is achieved by setting all values in the high-frequency map that are equal to or above it to 1, while the others are set to 0.

A certain pixel of $M_{ij}$ with the value 0 in the decision mask indicates less computation on pixel $LR_{ij}$ during convolution. Accordingly, more 0 values found in the decision masks mean less computation. When applied to ASCNN, an increase in the mask’s sparsity promotes a decrease in overall model FLOPs. In this paper, mask sparsity refers to the ratio of zeros in the decision mask. Therefore, high sparsity means more zero values, which implies marking more pixels as $P^{LF}$ to reduce its computation.

Fig. 2 presents a visual representation of the decision masks generated with the bird image (from the set 5 [33] dataset). We can observe in Fig. 2(d) that smooth areas of the LR image in Fig. 2(a) are sparse compared to other areas. We can also see the increase in mask sparsity as $\theta$ rises.

However, decision masks can produce different sparsity results on different images, even when $\theta$ is constant. The mask will not be sparse if all areas of an LR image are of high-frequency. On the other hand, images that contain many low-frequency areas require very few numbers of FLOPs to restore. Hence, the computational burden depends on the nature of each LR image and $\theta$ is utilized to manually adjust the reduction rate. In Fig. 2, Threshold values are carefully selected to visualize masks of various sparsity values. We set $\theta$ to 0.002 and increase it to 0.016, 0.03, and 0.044, with a 0.014 interval.

The decision network in TSSR [15] classifies high-frequency tiles from low-frequency ones by training. This might be a good solution to discriminate them. Nevertheless, the number of FLOPs cannot be carefully adjusted to satisfy certain use cases. Moreover, due to additional operations in the decision network, overhead FLOPs reduce the impact of the computational reduction.

B. AREA-SPECIFIC CNN
The decision mask algorithm identifies and discriminates $P^{HF}$ from $P^{LF}$. We aim to induce less computation on the $P^{LF}$ while letting the $P^{HF}$ utilize the convolutional filters to the fullest. This is done by applying sparse convolution for the $P^{HF}$ and $P^{LF}$ separately.
However, achieving reduction is not an easy task due to the following issue: Fig. 3 describes a solution for operating on $P^{LF}$ and $P^{HF}$ by using separate CNN models. Bicubic and bilinear operations [5] reference surrounding areas to generate output pixels. Similarly, to predict an output, a convolutional spatial kernel is used to operate the middle and surrounding pixels. A spatial kernel is defined as a convolutional filter kernel with sizes of $3 \times 3$ or larger. This implies that the surrounding pixels are also operated in the previous layer. When this process is repeated, we realize that predicting a single output pixel is determined by more surrounding pixels. Thus, not only $P^{HF}$, but surrounding pixels require the same convolutional operation. The number of required surrounding pixels is equivalent to the receptive field size. For the same reason, TSSR [15] could not reduce its tile size, as it impacted the reduction rate negatively.

Our ASCNN core algorithm is shown in Fig. 4. The convolutional operation for the $P^{HF}$ is defined as high parameter convolution (HPC). “High” in this case implies that the convolutional operation is done only on the high-frequency areas. On the other hand, the low parameter convolution (LPC) operation is done on the $P^{LF}$. It has fewer parameters compared to the HPC because it is used only on low-frequency areas. Fig. 4. (a) shows an input image with $P^{HF}$ marked in red and the $P^{LF}$ marked in blue. The light red and light blue denotes the HPC operation and the LPC operation respectively. The $P^{HF}$ and $P^{LF}$ has been marked by the decision mask. We utilize the HPC on the $P^{HF}$ while skipping the convolutional operation for $P^{LF}$. On the other hand, the $P^{LF}$ are operated with the LPC and skip the convolutional operations for the $P^{HF}$. It can also be expressed as a double sparse convolution that is fused at the end. As a result, all pixels are generated and are present for operations in the next layer.

With our method, surrounding pixels are present at all times. Surrounding pixels that are generated either by LPC or HPC are utilized for the next convolutional operation to produce the output (Fig. 4 (a)). In a hidden layer, surrounding pixel data are outputs of the previous convolutional operation. In our method, even when the surrounding pixels are generated by LPC like in Fig. 4 (a), HPC uses them for its operation.

Although HPC and LPC have different focus areas, they depend on each other and work together to generate an SR output image. A detailed explanation of this work is shown in Fig. 4 (b) and Fig. 4 (c).

Use cases for the first layer and hidden layers are shown in Fig. 4 (b) and it can also be represented as shown in (3).

$$f^{HPC}(I^{1}_{n-1}) = I^{1}_{n}$$  

(3)

where $I^{1}_{n-1}$ stand for areas of an image marked as $P^{HF}$ while $f^{HPC}$ represents the HPC convolution. $f^{HPC}$ is the standard convolutional operation but skips the $P^{LF}$. LPC uses the same kernel size as HPC which is represented as (4).

$$f^{LPC-CE}[f^{LPC-S}(I^{0}_{n-1}, r)] = I^{0}_{n}$$  

(4)

where $I^{0}_{n-1}$ stands for the areas of an input image marked as $P^{LF}$ while $f^{LPC-S}$ represents the LPC with a spatial kernel (LPC-S). However, the output channels of LPC-S are reduced by a factor represented as $r$, which stands for the reduction scale. Input and output channels in Fig. 4 are represented as...
c-out and c-in respectively. \( f^{LPC-CE} \) is the LPC for channel-expansion (LPC-CE) operation. As a part of LPC, they have a \( 1 \times 1 \) kernel size used to recover the channels that were lost by \( f^{LPC-S} \). They also skip the convolutional operation for \( P^{HF} \) as they are independent of surrounding pixels. The output results from the LPC and HPC are fused together to form the final output as shown in (5).

\[
f^{Fusion}(I_n^1, I_n^0) = I_n
\]

where \( I_n^1 \) and \( I_n^0 \) are the outputs of \( I_{n-1}^1 \) and \( I_{n-1}^0 \), respectively. \( f^{Fusion} \) represents the fusion operation of \( I_n^1 \) and \( I_n^0 \) and \( I_n \) is the output. The last layer of a SISR is usually the upsampling layer. Transposed convolution and sub-pixel convolution are typical examples. For transposed convolution, the decision mask is enlarged by the training scale factor with the nearest neighboring interpolation operation. This is because high-frequency areas are extracted from the LR image and their positions do not change but only expand when the image is upsampled.

As shown in Fig. 4 (c), the ASCNN settings are different for the last layer. The final output channel number is 1 (grayscale image) or scale-factor\(^2 \) in the case of sub-pixel convolution. Hence, implementing the Fig. 4 (b) settings is ineffective for reduction. Instead, a \( 1 \times 1 \) kernel convolution is applied to reduce the input channels. This is shown in (6).

\[
f^{LPC-S}[f^{LPC-CR}(I_{n-1}^0, r)] = I_n^0
\]

where \( f^{LPC-CR} \) means LPC for channel reduction (LPC-CR). It represents the \( 1 \times 1 \) kernel convolution used to reduce the input channel before the \( f^{LPC-S} \) operation. Consequently, a skip in convolutional operation cannot be applied to the point-wise convolution due to the problem expressed in Fig. 3. The next operation (spatial convolution) depends on surrounding pixels, which cannot be generated if a convolution skip is applied in \( f^{LPC-CR} \).

Each channel in a CNN feature map possesses certain information about the LR image. Hence, a pixel within a feature map holds information about its corresponding input pixel of the LR image. Since \( P^{LF} \) are easy to restore, quality degradation is less significant when we reduce the number of extracted channels for those pixels only. Additionally, ASCNN does not change the network’s receptive field since the LPC-S uses the same kernel sizes as the HPC.

### C. MORPHOLOGICAL DILATION OF DECISION MASKS

To improve the effectiveness of the decision mask, we apply morphological dilation. Morphological dilation is a technique in image morphology [34], used to increase or dilate the positive (1) areas of a binary image. Dilation is the first step used in morphological closing. Areas filled with positive values depend on the dilation kernel. It is the process of sliding a window kernel across the binary image. If any value under the kernel is 1, the output results in 1 for that pixel. Due to its simplicity, its computational burden is insignificant.

As shown in Fig. 4 (a), when applying an HPC on a feature map, surrounding pixels under its filter kernel have to be calculated. However, these surrounding pixels might be \( P^{HF} \) which had been generated by LPCs in the previous layer. This affects the accuracy of the HPC in the current layer operation. To solve this issue, we exploit morphological dilation. The idea is to not only mark the \( P^{HF} \) generated by (1) and (2), but also mark its surrounding pixels as a part of \( P^{HF} \) which is shown in (7).

\[
f^{dil}(M, k3) = DM
\]

where \( f^{dil} \) denotes the dilation operation on the decision mask \( M \) to output a dilated mask \( DM \). It is done with a \( 3 \times 3 \) kernel size represented as \( k3 \).

Fig. 5 shows the decision mask of an image from the Set 5 [33] dataset. Fig. 5 (b) and Fig. 5 (c) illustrate the mask without dilation and the mask generated by \( k3 \), respectively. To properly visualize their differences, we adjust \( \theta \) to produce the same sparsity, which is 0.4. We observe that an increased threshold value is needed to obtain the same sparsity as the non-dilated mask. This implies that we reference \( P^{HF} \) along with its surrounding pixels during an HPC operation to reduce the number of \( P^{LF} \) in the image. Therefore, by operating the surrounding pixels of \( P^{HF} \) with HPC, dilation helps ASCNN to rigidly operate on high-frequency areas.

Fig. 6 shows a rise in mask sparsity as \( \theta \) increases. This is tested with the LR images for the Set 14 [35] dataset. The dilated masks show less increase because higher \( \theta \) is needed compared to the non-dilated masks. We observe less curvature as kernel size increases. This means that with increased \( \theta \) and dilution, the high-frequency areas are easier to predict, hence making the threshold-sparsity relationship more linear.

### IV. EXPERIMENTS

Nvidia 2080Ti GPU is used for training. To rigorously compare the ASCNN performance with TSSR [15] and FSR-CNN [23], we utilize the same training dataset which is the
H. Alao et al.: Area-Specific Convolutional Neural Networks for Single Image Super-Resolution

The grayscale channel of the 91-images dataset by [8]. Data augmentation techniques such as rotations and left-right image flip are also employed. The training dataset images are divided and downsampled to produce LR image patches of size $32 \times 32$. We use the MSE (mean squared error) loss function and the ADAM optimizer [36] with a minibatch size of 32 during training. Every experiment is trained for $8 \times 10^5$ iterations. The learning rate is initially set to $10^{-4}$ and decays to 0 by the cosine annealing scheduler [37].

Unlike ClassSR [31], experiments on various tile sizes and overlapping tiles are performed in TSSR [15]. However, ClassSR [31] show results for only a single tile size. Moreover, their targets are images of high resolution. Thus, only results of 8K, 4K, and 2K resolution images were given. They claim that the commonly used SR test sets such as Set5 [33] and Set14 [35] are too small for their implementation. Therefore, we compare our work with TSSR [15] which provides results with the commonly used SR datasets.

A. EFFECTS OF DECISION MASKS ON ASCNN

As shown in Fig. 7, our baseline model is the same as TSSRN-l from TSSR [15]. It is a replication of FSRCNN [23] but with different input and output channel numbers in its layers.

Although FLOPs decrease with ASCNN, parameters are increased due to the additional filters (LPC). In Table 1, the baseline model’s parameters and ASCNN’s parameters are indicated. The overall parameter numbers depend on the reduction scale ($r$). In this case, we set $r$ to 4.

As shown in Table 1, we run experiments with various $\theta$ on the Set 14 [35] dataset with scale factor $\times 2$. We notice a performance drop as $\theta$ increases, which can be visualized in Fig. 8 as well. Although there is a drop in performance, the PSNR difference between the baseline model and model 4 is 0.11dB. However, the FLOPs are reduced by 58%.

Due to increased parameters, our baseline model performs better than the FSRCNN [23]. As mentioned earlier our baseline model architecture is identical to TSSRN-l from TSSR [15], which is the model without any reduction. The TSSR [15] model successfully reduces the computational burden, but at the expense of performance drop. According to our experiments, model 3 reduces the FLOPs by 52%, and its performance is the same as TSSR [15].

B. MORPHOLOGICAL DILATION

As shown in Fig. 9, there is an increase in performance with the implementation of dilation in decision masks. Results are also presented in Table 2. For an accurate comparison, we adjust the threshold ranges of the masks generated by dilation to roughly cover various sparsity levels. Due to additional parameters (LPC) and the effectiveness of morphological dilation, when sparsity is less than 0.26, the performance is better than the baseline model. However, with dilation, performance rapidly drops when the sparsity closely approaches 1. Although dilation focuses on the high-frequency areas more, when the $\theta$ is high, most information extracted by the high-frequency map is lost. Consequently, the decision mask
will mark fewer points as $P_{HF}$ with its surrounding areas only. On the other hand, as illustrated in Fig. 10, the non-dilated mask with the same sparsity can operate on spread areas, because its $\theta$ is relatively smaller. Dilation with the kernel size $3 \times 3$ is the optimal option because when sparsity is lower than 0.73 (51% FLOPs reduction), it always improves the performance of ASCNN compared to the model without dilation. Performances of models with dilation kernel size larger than $3 \times 3$ drop more rapidly. We also observe better performance and reduction trade-off, hence, we employ dilation with kernel size $3 \times 3$ in further experiments.

C. GENERALIZATION OF ASCNN
In this section, we apply the ASCNN method to each layer of the ESPCN [24] and the SRCNN [8] models. These models are also trained with the 91-image dataset. Compared to the ASCNN baseline model, these models have more channels in their hidden layer feature maps (64 and 32). As a result, when $r$ is 4, degradation in model performance is hardly observed. Nevertheless, we aim to discover the reduction limit for performance to degrade so we set $r$ to 8. As shown in Fig. 11, performance drops predictably as mask sparsity increases. We notice that the performance degradation is less significant in these cases. Based on the results achieved by applying the ASCNN method on SRCNN [8] and ESPCN [24], we show that the ASCNN method on models with sufficient feature-map such as 32 or 64 is more effective at reducing the FLOPs while maintaining its performance.

D. COMPARISON WITH OTHER SR METHODS
We compare our ASCNN method with other lightweight state-of-the-art SR methods. Just like TSSR [15], testing
### TABLE 3. Quantitative results of deep learning-based SR algorithms.

| scale | Model          | Params. | FLOPs  | Set5 PSNR/SSIM | Set14 PSNR/SSIM | BSD100 PSNR/SSIM | DIV2K val. PSNR/SSIM |
|-------|----------------|---------|--------|----------------|----------------|------------------|------------------|
|       |                |         |        |                |                |                  |                  |
| ×2    | SRCNN [8]      | 57.2K   | 52.7G  | 36.66/0.9542   | 32.42/0.9063   | 31.36/0.8879     | 33.05/0.9581     |
|       | ESPCN [24]     | 21.2K   | 4.9G   | 36.80/0.9549   | 32.43/0.9059   | 31.31/0.8874     | 34.65/0.9329     |
|       | FSRCNN [23]    | 12.5K   | 6.0G   | 37.00/0.9558   | 32.63/0.9088   | 31.58/0.8908     | 34.98/0.9362     |
|       | FSRCNN-s [23]  | 3.9K    | 2.7G   | 36.57/0.9532   | 32.28/0.9052   | 31.23/0.8866     | 34.48/0.9316     |
|       | TSSR [15]      | > 21.3K | 3.2G   | 37.06/0.9565   | 32.62/0.9099   | 31.47/0.8893     | 34.87/0.9351     |
|       | ASCNN (ours)   | 18.1K   | 2.9G   | 37.12/0.9560   | 32.71/0.9081   | 31.54/0.8898     | 34.94/0.9350     |
| ×3    | SRCNN [8]      | 57.2K   | 52.7G  | 32.75/0.9090   | 29.28/0.8209   | 28.41/0.7863     | 29.64/0.9138     |
|       | ESPCN [24]     | 22.6K   | 2.3G   | 32.82/0.9096   | 29.26/0.8189   | 28.37/0.7851     | 31.17/0.8646     |
|       | FSRCNN [23]    | 12.5K   | 5.0G   | 33.16/0.9140   | 29.43/0.8242   | 28.52/0.7900     | 31.06/0.8623     |
|       | FSRCNN-s [23]  | 3.9K    | 2.5G   | 32.54/0.9055   | 29.08/0.8167   | 28.33/0.7815     | 30.88/0.8857     |
|       | TSSR [15]      | > 21.3K | 2.5G   | 33.59/0.9258   | 29.74/0.8399   | 28.92/0.8013     | 31.20/0.8658     |
|       | ASCNN (ours)   | 18.1K   | 2.2G   | 33.24/0.9149   | 29.50/0.8237   | 28.55/0.7895     | 31.41/0.8691     |
| ×4    | SRCNN [8]      | 57.2K   | 52.7G  | 30.48/0.8628   | 27.49/0.7503   | 26.90/0.7101     | 27.78/0.8753     |
|       | ESPCN [24]     | 24.6K   | 1.4G   | 30.58/0.8632   | 27.50/0.7493   | 26.91/0.7109     | 29.36/0.8089     |
|       | FSRCNN [23]    | 12.5K   | 4.6G   | 30.71/0.8657   | 27.59/0.7535   | 26.96/0.7139     | 29.33/0.8056     |
|       | FSRCNN-s [23]  | 3.9K    | 2.5G   | 30.11/0.8499   | 27.19/0.7423   | 26.84/0.7085     | 29.01/0.7977     |
|       | TSSR [15]      | > 21.3K | 1.6G   | 30.41/0.8578   | 27.37/0.7534   | 26.86/0.7079     | 29.29/0.8049     |
|       | ASCNN (ours)   | 18.1K   | 2.0G   | 30.92/0.8721   | 27.71/0.7571   | 27.05/0.7158     | 29.53/0.8137     |

**FIGURE 12.** Visual comparison of SR (×2) results generated with different threshold. The bird image from Set 5 and 102061 image from the Div2K dataset are employed.

**FIGURE 13.** Visual comparison among SR (×2) images generated with other models. The 223061 image from BSD 100 and 0823 image from the Div2K dataset are used for the visualization.

Datasets are Set 5 [33], Set 14 [35], BSD100 [38], and the DIV2K validation [39] (DIV2K Val.) dataset. We show performance in PSNR and SSIM on the testing datasets with different scale factors and compare our work to models with FLOPs less than 60G. Quantitative results are shown in Table 3. The FLOPs and the number of parameters (Params.)
are also shown. The FLOPs are calculated assuming the HR image size is 720p.

For benchmarking with ASCNN, we use $\theta = 0.04$ with a decision mask dilated by kernel size $3 \times 3$. The FLOPs for TSSR [15] are achieved by calculating the average reduction for all the testing datasets. Its average reduction is 34.8%, 28.5%, and 47.8% for $\times 2$, $\times 3$, and $\times 4$ SR respectively. ASCNN average reduction is 40.1%, 37.0%, and 34.0% for $\times 2$, $\times 3$, and $\times 4$ SR respectively.

Table 3 shows that some TSSR [15] results significantly outperform ASCNN with differences above 0.3dB in $\times 3$. A similar observation is made when comparing it to FSRCNN [23]. Although the authors did not explain the reason performance is better in $\times 3$, it might be an effect of their decision network. Nevertheless, ASCNN shows the best performance among all compared networks for $\times 2$ and $\times 4$.

The total parameters of the TSSR [15] model wasn’t stated in their paper, but the parameters excluding the fully connected layers are 21.3K. Thus, the total parameters are more than that ($>21.3K$). Additionally, experimental results for BSD100 [38] and DIV2K val. [39] are not given in the ESPCN [24] paper. Therefore, we trained the ESPCN [24] network to achieve the required results.

As shown in Table 3, for $\times 2$ and $\times 3$ SR scale factors, ASCNN maintains similar performance with TSSR [15] but its FLOPs reduction is better. For $\times 4$ SR scale factors, although TSSR [15] has fewer FLOPs, its performance is worse than ASCNN.

E. VISUALIZATION OF RESULTS

In Fig. 12, we visualize the results that are achieved with different $\theta$ by our ASCNN model. We also show a visual comparison of SR results among other SOTA models in Fig. 13.

F. ASCNN WITH A SINGLE MODEL

For each ASCNN model, $\theta$ is decided before training and a different $\theta$ cannot be used during inference. Consequently, training always has to be performed for a specific $\theta$. To tackle this issue, we set a range of thresholds ($\theta = [0.01 \sim 0.09]$) to be trained on. During training, the $\theta$ from the range is randomly selected for the forward and backward propagation of each training batch.

Results are visualized in Fig. 14 and they are compared to the original models trained separately. Although performance degradation is observed, the difference is not significant (below 0.02dB). Moreover, an inference can be applied for any $\theta$ within its range.

V. CONCLUSION

ASCNN aims to solve the problems of redundancy computation in unnecessary areas. According to our analysis, low-frequency areas of LR images do not need the same computational burden as the high-frequency areas. We discriminate the high-frequency areas from the low-frequency by marking them as $p_{HF}$ and $p_{LF}$ respectively. The HPC and LPC work jointly to satisfy operations of high-frequency and low-frequency areas. As a result, ASCNN reduces the computational complexity up to 40.1% / 37.0% / 34.0% for $\times 2 / \times 3 / \times 4$ scale factors while maintaining image restoration performance which is comparable to other SOTA models.

However, since LPC had to be added to solve for $P_{LF}$ and reduce the FLOPs in those areas, the overall parameters are increased. Consequently, the limitation of ASCNN is its need for additional trainable parameters. Further work can be done by converting the HPC to LPC for $P_{LF}$ instead of adding more parameters. Additionally, this work focuses on discriminating frequency data with just a binary mask. In future works, the decision mask can be generated to represent a variety of frequency levels and operate in them distinctively in order to improve the model’s performance or effectively decrease its FLOPs.

REFERENCES

[1] W. Shi, J. Caballero, C. Ledig, X. Zhuang, W. Bai, K. Bhatia, A. M. Marvao, T. Dawes, D. O’Regan, and D. Rueckert, “Cardiac image super-resolution with global correspondence using multi-atlas patchmatch,” in Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. Berlin, Germany: Springer, 2013, pp. 9–16.
[2] Y. Li, B. Sixou, and F. Peyrin, “A review of the deep learning methods for medical images super resolution problems,” IRBM, vol. 42, no. 2, pp. 120–133, Apr. 2021.
[3] P. Rasti, T. Uiboopin, S. Escalera, and G. Anbarjafari, “Convolutional neural network super resolution for face recognition in surveillance monitoring,” in Proc. Int. Conf. Articulated Motion Deformable Objects. Cham, Switzerland: Springer, 2016, pp. 175–184.
[4] H. Seibel, S. Goldenstein, and A. Rocha, “Eyes on the target: Super-resolution and license-plate recognition in low-quality surveillance videos,” IEEE access, vol. 5, pp. 20020–20035, 2017.
[5] S. Fadnavis, “Image interpolation techniques in digital image processing: An overview,” Int. J. Eng. Res. Appl., vol. 4, no. 10, pp. 70–73, 2014.
[6] W. Yuanji, L. Jianhua, L. Yi, F. Yao, and J. Qinzhong, “Image quality evaluation based on image weighted separating block peak signal to noise ratio,” in Proc. Int. Conf. Neural Netw. Signal Process., vol. 2, 2003, pp. 994–997.
[7] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, Apr. 2004.
[8] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 2, pp. 295–307, Feb. 2015.
H. Alao et al.: Area-Specific Convolutional Neural Networks for Single Image Super-Resolution

[9] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jul. 2017, pp. 136–144.

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

[11] N. Ahn, B. Kang, and K. A. Sohn, “Fast, accurate, and lightweight super-resolution with cascading residual network,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 252–268.

[12] J. Kim, J. K. Lee, and K. M. Lee, “Deeply-recursive convolutional network for image super-resolution,” in Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 1637–1645.

[13] Y. Tai, J. Yang, and X. Liu, “Image super-resolution via deep recursive residual network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 3147–3155.

[14] L. Wang, X. Dong, Y. Wang, X. Ying, Z. Lin, W. An, and Y. Guo, “Exploring sparsity in image super-resolution for efficient inference,” in Proc. IEEE/ACM Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 4917–4926.

[15] J. Lee, J. Lee, and H.-J. Yoo, “SRNPU: An energy-efficient CNN-based super-resolution processor with tile-based selective super-resolution in mobile devices,” IEEE J. Emerg. Sel. Topics Circuits Syst., vol. 10, no. 3, pp. 320–334, Sep. 2020.

[16] Z. Qin, Q. Zeng, Y. Zong, and F. Xu, “Image inpainting based on deep learning: A review,” Displays, vol. 69, Sep. 2021, Art. no. 102028.

[17] K. Zhang, W. Luo, Y. Zhong, L. Ma, B. Stenger, W. Liu, and H. Li, “Deblurring by realistic blurring,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 2737–2746.

[18] J.-V. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image translation using cycle-consistent adversarial networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2223–2232.

[19] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “StarGAN: Unified generative adversarial networks for multi-domain Image-to-Image translation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8769–8777.

[20] C. Ledig, L. Theis, F. Huszár, 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.

[21] 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. 1646–1654.

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[23] 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. 391–407.

[24] H. 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.

[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] Z. He, H. Huang, M. Jiang, Y. Bai, and G. Luo, “FPGA-based real-time super-resolution system for ultra high definition videos,” in Proc. IEEE 26th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM), Apr. 2018, pp. 181–188.

[27] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, arXiv:1704.04861.

[28] X. Chu, B. Zhang, H. Ma, R. Xu, and Q. Li, “Fast, accurate and lightweight super-resolution with neural architecture search,” in Proc. 25th Int. Conf. Pattern Recognit. (ICPR), Jan. 2021, pp. 59–64.

[29] H. Zhao, X. Kong, J. He, Y. Qiao, and C. Dong, “Efficient image super-resolution using pixel attention,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 56–72.

[30] C. Liu and Y. Furukawa, “MASC: Multi-scale affinity with sparse convolution for 3D instance segmentation,” 2019, arXiv:1902.04478.

[31] X. Kong, H. Zhao, Y. Qiao, and C. Dong, “ClassSR: A general framework to accelerate super-resolution networks by data characteristic,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 12016–12025.

[32] J. A. Saif, M. H. Hammad, and I. A. Alqabati, “Gradient based image edge detection,” Int. J. Eng. Technol., vol. 8, no. 3, p. 153, 2016.

[33] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, “Low-complexity single-image super-resolution based on nonnegative neighbor embedding,” in Proc. 23rd Brit. Mach. Vis. Conf. (BMVC), 2012, pp. 135.1–135.10.

[34] M. Goyal, “Morphological image processing,” IJCSIT vol. 2, no. 4, p. 59, 2011.

[35] I. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” IEEE Trans. Image Process., vol. 19, no. 11, pp. 2861–2873, Nov. 2010.

[36] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[37] I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts,” 2016, arXiv:1608.03983.

[38] 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. 8th IEEE Int. Conf. Comput. Vis. (ICCV), vol. 2, Jul. 2001, pp. 416–423.

[39] 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.

HONNANG ALAO received the B.S. degree in electronic engineering from Sun Moon University, in 2020, where he is currently pursuing the M.S. degree with the Department of Computer and Electronic Engineering. His research interests include deep learning, image processing, and multimedia.

TAE SUNG KIM received the B.S. degree in electrical engineering from Pusan National University, Busan, South Korea, in 2010, and the M.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2013 and 2017, respectively. From 2018 to 2021, he was a Senior Researcher at Samsung SLSI. Since 2021, he has been an Assistant Professor with the Department of Electronic Engineering, Sun Moon University. His research interest includes algorithm and architecture design of image/video coding systems.

KYUJOONG LEE received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2002, the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2008, and the Ph.D. degree in electrical engineering and computer science from Seoul National University, in 2013. From 2013 to 2017, he worked with the SLSI Division, Samsung Electronics Corporation. In 2017, he was appointed as an Assistant Professor with the Department of Electronic Engineering, Sun Moon University, Asan, South Korea. From 2002 to 2005, he was with Com2us Corporation, Seoul, as a Developer. In 2022, he moved as an Associate Professor with the School of AI Convergence, Sungshin Women’s University, Seoul. His major research interests include algorithms and architectures of deep learning and image/video processing.