Towards Real-Time 4K Image Super-Resolution

Eduard Zamfir, Marcos V. Conde, Radu Timofte
Computer Vision Lab, CAIDAS, IFI, University of Würzburg, Germany
{eduard-sebastian.zamfir, marcos.conde, radu.timofte}@uni-wuerzburg.de

Abstract

Over the past few years, high-definition videos and images in 720p (HD), 1080p (FHD), and 4K (UHD) resolution have become standard. While higher resolutions offer improved visual quality for users, they pose a significant challenge for super-resolution networks to achieve real-time performance on commercial GPUs. This paper presents a comprehensive analysis of super-resolution model designs and techniques aimed at efficiently upscaling images from 720p and 1080p resolutions to 4K. We begin with a simple, effective baseline architecture and gradually modify its design by focusing on extracting important high-frequency details efficiently. This allows us to subsequently downscale the resolution of deep feature maps, reducing the overall computational footprint, while maintaining high reconstruction fidelity. We enhance our method by incorporating pixel-unshuffling, a simplified and speed-up reinterpretation of the basic block proposed by NAFNet, along with structural re-parameterization. We assess the performance of the fastest version of our method in the new NTIRE 2023 Real-Time 4K SR challenge and demonstrate its potential in comparison with state-of-the-art efficient super-resolution models when scaled up. Our method was tested successfully on high-quality content from photography, digital art, and gaming content.

1. Introduction

In image super-resolution (SR), one deals with the ill-posed problem of recovering the high-resolution (HR) counterpart of a previously down-sampled and possibly further degraded low-resolution (LR) source image. Previous research has introduced several classical methods for single image SR, as documented in [5, 6, 17, 42, 46–48]. Nonetheless, with the emergence of Deep Learning, research on single image SR rapidly shifted towards deep learning-based approaches [7, 13, 30, 35, 36, 51, 58, 62]. While much progress in restoration performance has been achieved through larger and deeper networks, these improvements have incurred a higher demand for time and computational resources, necessitating more efficient and lightweight solutions. As media streaming platforms have achieved overwhelming success, and the amount of image and video content created and shared online has become practically inexhaustible, the need for stable and high-bandwidth internet connections has significantly increased. However, the media industry has adopted the practice of compressing its content before transmission and reconstructing it to its full resolution at the consumer’s end. Efficient and lightweight super-resolution methods have become increasingly important as a result.

To further the development of efficient and fast SR methods, in conjunction with the NTIRE 2023 Real-Time 4K SR challenge [10] we investigate previous SR concepts in the context of upscaling diverse types of content, including digital art and photography, to ultra-high resolution. Following
this study, we propose a fast and lightweight model tailored for 4K image SR from large input resolutions (720p, 1080p → 4K). Moreover, our method showcases its scalability and achieves comparable results on established benchmarks while surpassing them in terms of runtime and efficiency. Additionally, we explore the significance of training data when presented with the challenge of enhancing computer-generated visuals along with photorealistic content.

Starting with the basic blueprint shown in Fig. 2 for learning-based super-resolution approaches, we gradually modify a simple and shallow network architecture to improve its performance in terms of runtime and reconstruction fidelity. Drawing concepts from the image compression research [53], the key aspect of our approach is downsizing the deep features to accelerate the computation, while simultaneously retaining valuable high-frequency (HF) information from the LR input. Therefore, we efficiently extract high-frequency details first before downsampling the feature. To ensure effective computation of deep features, we utilize the potent NAFNet [7] block. Additionally, we simplify the design of its basic components and apply structural re-parameterization [12] at inference time to further reduce the total runtime of our method. After extracting high-frequency details, we refine them through a dedicated parallel branch before reintroducing them to the deep features. This compensates for the previously performed downsampling operation. We provide exhaustive ablation studies using the novel 4K RTSR [10] benchmark as a reference.

2. Related Work

Efficient Architectures. In recent years, achieving near real-time SR on resource-constrained platforms has gained popularity [24, 33, 34, 57]. As a result, researchers have proposed optimized neural architectures [23], network compression methods, and training strategies to address the need for efficient solutions [2, 14, 29, 44].

IMDN [22] introduces a lightweight information multi-distillation network that employs cascaded blocks to extract hierarchical features using an information distillation mechanism (IDM). RFDN [37] refines the architecture of IMDN [22] by proposing the residual feature distillation network, which replaces IDM with feature distillation connections. ECBSR [61] introduces an edge-oriented convolutional block that utilizes structural re-parameterization [12] to enhance the learning capability of the model without impacting the inference time. While accessing preceding network layers can be compute-intensive, sequential operations can minimize memory consumption and runtime overhead. RLW [28] leverages this idea to achieve high reconstruction accuracy through the use of simple $3 \times 3$ convolutions instead of concatenation and feature distillation layers, as well as a multi-stage training strategy. Similarly, FMEN [16] employs a lightweight backbone by stacking multiple optimized convolutions and reducing compute through re-parameterization at inference time. ESRT [38] combines a lightweight CNN to dynamically adjust the feature map size, allowing for the extraction of deep features with low computational cost, with a lightweight Transformer [15, 49] to capture long-term dependencies between similar patches. VapSR [65] introduces large receptive field design with depth-wise convolutions into the attention mechanism and presents a novel pixel normalization approach for improved training stability. Furthermore, the Mobile AI workshop in 2022 [24] highlighted the challenge of achieving efficient and accurate quantization for image super-resolution on edge devices. To address this issue, most of the methods proposed at the workshop utilized a shallow CNN architecture and re-parameterization techniques to reduce inference time while maintaining competitive restoration performance. NAFNet [7] presents a highly efficient approach for image restoration by simplifying commonly used architectural components, i.e., removing nonlinearities, outperforming previous techniques across a wide range of image restoration problems.

Upscaling to Ultra-High Definition. The field of super-resolving images or videos to achieve ultra-high resolutions, such as 4K and 8K, remains relatively unexplored in the research community. While modern display technologies can handle ultra-high definition (UHD) content, effective broadcasting and streaming require significant bandwidth. As a result, the industry standard involves downsizing prior to data transfer and upsampling back to full resolution on the consumer’s end. This process demands highly efficient super-resolution (SR) approaches [8, 11, 26]. Moreover, cloud-based gaming experiences a large gain in popularity, where upsampling digital content presents additional challenges, e.g., aliasing, consequently requiring tailored approaches [52, 54]. The upsampling of images to 4K resolution in real-time remains a relatively unexplored topic within the broad research community of SR. The NTIRE 2023 4K RTSR challenge [10] addresses this open question by demanding lightweight yet effective SR solutions from its participants. It also provides them with a competitive benchmark for 4K image SR. Moreover, to the best of our
knowledge, Zhang et al. [59] offer a comprehensive dataset for evaluating recent SR techniques on upsampling to 4K and 8K resolution. However, this dataset does not consider the increased model runtime and has limited content.

3. Method

In this section, we first revisit the blueprint approach for SR based on deep models and introduce our proposed architecture in Sec. 3.1. Next, in Sec. 3.2 we describe the training schedule to enhance the performance of our method in the NTIRE 2023 4K RTSR challenge [10].

3.1. Model Architecture

Over the past few years, the SR research community has developed a common structure for CNN-based neural architectures. The majority of methods follow a three-part blueprint shown in Fig. 2 that includes an initial extraction of shallow features, a computationally intensive refinement in deep feature space, and a final upsampling stage to achieve the target resolution. Moreover, previous works [20,36,62,63] have shown the tremendous benefit of adding local and global residual connections. The methods proposed in [24] effectively utilize this blueprint to achieve strong reconstruction capabilities while maintaining a low computational footprint. In our work we adopt this aforementioned structure and prioritize real-time inference with a focus on a shallow and lightweight design. We begin with a simple stack of \( 3 \times 3 \) convolutions, each followed by a GeLU [19] non-linearity visualized in Fig. 5a. Contrary to prior work [24,31,56], we address SR from large-scale inputs, which poses an additional layer of complexity for real-time processing. In Deep Learning, a common approach is to reduce the spatial resolution of feature maps to keep the computational burden low. However, it has been demonstrated that decreasing spatial resolution within the network can negatively impact the reconstruction performance of SR methods, since high-frequency (HF) details are already scarce in the LR input image. Yet, the atypical large size of LR inputs in our use-case allows us to effectively process the HF information differently.

An overview of our final architecture design is presented in Fig. 3. First, we efficiently extract the HF components from the LR input. Subsequently, the LR input and extracted HF maps undergo processing via a shared convolution for shallow feature extraction. Next, we enhance the HF features in a dedicated high frequency branch (HFB) while simultaneously compressing the features in the deep feature extraction branch (DFEB) of the network. Lastly, we inject the enhanced HF components back into the deep features. We add a LayerNorm [3] and another convolution before upsampling to the desired output resolution using PixelShuffle [44]. In particular LayerNorm provides consistent improvement and stable training, becoming a standard in image restoration [7,9]. Next, we gradually modify the basic structure to enhance the network capabilities while aiming at keeping computational costs low.

**Enhancing High Frequencies.** Inspired by [38, 40], we aim at efficiently extracting and enhancing the remaining HF details in the LR input. To achieve this goal, we explore two straightforward approaches that are both rapid and do not introduce additional complexity that could impede the processing speed of our method. (i) We reduce the size of the LR input through average pooling, immediately followed by upscaling it back to the original resolution using Nearest Neighbor interpolation. (ii) We use an inexpensive Gaussian blur operation on the input to obtain its blurred version. Both approaches yield an image that represents the signal’s uniformity, which we then subtract from the initial LR input to obtain the HF components.

In Figure 4, we conduct a visual comparison between discussed approaches where the Down-and-Up operation falls short in extracting fine-grained details, while utilizing the Gaussian Blur aids in extracting circular contours. Tab. 2a quantifies the impact of both approaches on our method. The HF details are then further refined by a shallow parallel branch using a \( 3 \times 3 \) convolution and GeLU activation before being injected back into the deep features. Tab. 1 presents a direct comparison between the baseline and its variant with the HFB. Although both modifications depicted in Figs. 5a and 5b exhibit improved PSNR and SSIM, they also entail longer runtime.
Compressing Deep Features. In practical applications, the runtime of SR methods can be decreased by reducing the network’s depth or width. Nonetheless, this often results in inferior restoration performance. In addition to achieving the optimal balance between depth and width, downsizing the spatial dimensions provides another means of lowering the runtime. However, this usually results in the loss of valuable details, which is, in fact, crucial for SR tasks that aim to recover previously lost information. Earlier studies [37, 38] employed strided convolutions or pooling operations to attain the required spatial resolution. When we apply the Down-and-Up scheme to the DFEB of our method, as shown in Tab. 1, we observe a significant loss of reconstruction fidelity due to pooling. Intriguingly, the improvements obtained by incorporating the HFB into our method are not as prominent when the deep features are downscaled. To address this issue, we explore the use of the PixelUnshuffle [44] operation for feature downsampling, in addition to the initial extraction of HF components. PixelUnshuffle [44] reduces the spatial dimensions by a factor of s, while increasing the channel dimension by a factor of s^2. Although incorporating PixelUnshuffle naively into our architecture significantly reduces its efficiency, it does improve the reconstruction accuracy, as shown in Tab. 1 and the visualization in Fig. 5c. An architecture that employs the unshuffling may benefit from a larger channel dimension. However, to compensate for the loss of inference efficiency, we investigate the possibility of squeezing the channel dimensions after the unshuffling process and then mapping them back after the output of the DFEB. While this approach can reduce the runtime, the model is unable to recover the loss of information resulting from channel reduction, see Tab. 1. To address this issue, we restructure the 3 × 3 convolution used for extracting shallow features so that it occurs after the LR input has been unshuffled, see Fig. 5d. Both HF and LR features undergo additional processing through the DFEB and HFB modules. Prior to the upsampling stage, we merge the refined high-frequency components with the deep features, and then increase the output resolution by ×4 (for ×2 SR). Next, we detail the enhancements made to both the DFEB and HFB modules in order to improve their modeling capacity leading to the final design of our model.

Increasing Block Complexity. The architecture design at this point has limitations in terms of the expressiveness of its features as the basic component of the DFEB is a standard 3 × 3 convolution and GeLU activation repeated N times, see Fig. 5a. Recently, NAFNet [7] has shown strong reconstruction capabilities with a more complex block design. We enhance our plain block by incorporating components from the basic block of NAFNet visualized in Fig. 5e. Specifically, we replace the 3 × 3 convolution with an inverted depthwise separable convolution, which is followed by GeLU non-linearity to increase the feature dimensions from C to 2C. The basic block also includes Channel Attention [7, 50, 62], LayerNorm [3], and a local skip connection. To expedite the performance of this block design, we substitute the standard Channel Attention with its efficient version [50]. A final 1 × 1 convolution maps the feature dimensions back to C. As anticipated, incorporating the NAFNet-inspired block in Fig. 5f enhances the modeling capacity of our method, see Tab. 1. However, this also results in a substantial increase in runtime. Although we can address this issue by downsampling the deep features, we encounter challenges in maintaining the accuracy gains achieved by the more intricate block design, see Tab. 1. Unfortunately, this renders the modified architecture impractical for our use case. In the following section, we will outline our approach to streamline the network design while preserving both high inference speeds and reconstruction fidelity.

Model reparameterization. First introduced in [12], structural reparameterization has rapidly gained traction within the research community as a means of reducing model runtime during inference. Many participants of the NTIRE and AIM efficiency challenges [24, 33] have adopted various forms of reparameterization for their architectures. We closely follow [16] and replace the depthwise separable convolutions within the DFEB with a reparameterizable residual block (RRB), cf. Fig. 3b. The RRB expands the channel dimension C by a factor of \( f_{exp} = 2 \) with a 1 × 1 convolution. Next, a 3 × 3 convolution enhances the learned features in a higher dimensional space, followed by a final 1 × 1 convolution that compresses the features.

![LR Down-and-Up Gaussian Blur](Image 50x552 to 127x629)

Figure 4. Extracting high frequency (HF) information. Applying the Down-and-Up generates speckled information with higher intensities, whereas employing Gaussian Blur yields more intricate details, particularly in circular regions.

![LR Down-and-Up Gaussian Blur](Image 129x632 to 206x709)

![LR Down-and-Up Gaussian Blur](Image 208x552 to 284x629)

![LR Down-and-Up Gaussian Blur](Image 208x632 to 284x709)
channels back to $C$, retaining only the most discriminative features. Short and long residual connections facilitate feature propagation. During inference, we can summarize the RRB using a single $3 \times 3$ convolution. This reduces processing time while still preserving the expressive power of higher feature channels. Additionally, we eliminate the efficient Channel Attention module, final $1 \times 1$ convolution, and local residual, retaining only the normalization operation preceding the RRB. Our new reparameterizable basic block is visualized in Fig. 5g. Furthermore, we enhance the model capacity of the HFB by substituting the $3 \times 3$ convolution with the RRB, see Fig. 5h.

3.2. Towards Learning the High Frequency Details

Besides traditional pixel-wise reconstruction loss functions, the Computer Vision community proposed several perceptual losses [18, 25, 40, 60] to improve SR models and impose meaningful priors during model training. Explicitly modeling the high frequencies is a key concept of our model. Therefore, we extract high frequency information, e.g. edges and contours, from the SR output and HR target image using the same Gaussian blur operation as within our model. As an auxiliary optimization task, we minimize the L1 distance between obtained HF maps. The loss is formulated as follows:

$$L_{HF} = \| (y - (y \ast b)) - (\hat{y} - (\hat{y} \ast b)) \|_1$$  

We incorporate this auxiliary loss solely to enhance the performance of our model in the NTIRE 4K RTSR challenge [10]. Typically, participants develop increasingly complex training strategies to improve the performance of their methods in this challenge.

4. Experiments

4.1. Experimental Setup

Datasets and Metrics. Our training dataset is a combination of 800 images from DIV2K [1], 2650 images from Flickr2K, and 1000 images from LSDIR [32]. Following standard practice, we report PSNR and SSIM metrics on RGB. To explore the real-time performance of our method on large-scale inputs, we conduct most of our experiments on the new benchmark proposed in the NTIRE 2023 4K RTSR challenge [10]. Additionally, we evaluate our approach on canonical SR benchmarks, namely Set5 [4], Set14 [55], Urban100 [21] and BSD100 [41], when comparing our results to previously published work.

Training Details. We extract random crops of size $128 \times 128$ from the RGB training set and further augment the crops by random rotation, horizontal and vertical flipping. LR images are generated online using bicubic downsampling of the original HR images. We use ADAM [27] optimizer to minimize the $L1$ loss between the SR output and HR target for 100 epochs with the batch size set to 64 and an initial learning rate of $1e-3$, along with a step scheduler with step size 20 and decay factor 0.5.

Runtime evaluation. Unlike other studies [33], we assess the runtime of our proposed architectures by repetitively...
Table 1. Results on the NTIRE 2023 RTSR4K Benchmark. The runtimes are computed using Nvidia RTX 3090. For better comparison we color-code the runtime using \(<24\) FPS, \(30>x>24\) FPS, \(60>x>30\) FPS, \(120>x>60\) FPS and \(>120\) FPS, respectively.

<table>
<thead>
<tr>
<th>Scale Method</th>
<th># Params</th>
<th>FLOPs (G)</th>
<th>PSNR (dB,↑)</th>
<th>SSIM (↑)</th>
<th>Runtime (ms,↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicubic</td>
<td>-</td>
<td>-</td>
<td>33.92</td>
<td>0.8829</td>
<td>0.46</td>
</tr>
<tr>
<td>(a) Baseline</td>
<td>35.2K</td>
<td>429.98</td>
<td>34.11</td>
<td>0.8834</td>
<td>11.70</td>
</tr>
<tr>
<td>(b) Baseline + Down-and-Up</td>
<td>35.2K</td>
<td>200.66</td>
<td>34.01</td>
<td>0.8830</td>
<td>07.79</td>
</tr>
<tr>
<td>(c) Baseline + HFB + Down-and-Up</td>
<td>35.2K</td>
<td>286.65</td>
<td>34.00</td>
<td>0.8827</td>
<td>10.48</td>
</tr>
<tr>
<td>(d) Baseline + PixelUnshuffle</td>
<td>40.0K</td>
<td>217.65</td>
<td>34.01</td>
<td>0.8829</td>
<td>08.42</td>
</tr>
<tr>
<td>(e) Baseline + PixelUnshuffle + Squeeze-and-Excite Channels</td>
<td>346.6K</td>
<td>1347.28</td>
<td>34.15</td>
<td>0.8835</td>
<td>16.46</td>
</tr>
<tr>
<td>(f) NAFNet Basic Block</td>
<td>30.0K</td>
<td>353.54</td>
<td>34.17</td>
<td>0.8844</td>
<td>40.71</td>
</tr>
<tr>
<td>(g) NAFNet Basic Block + HFB</td>
<td>30.6K</td>
<td>439.54</td>
<td>34.17</td>
<td>0.8846</td>
<td>43.40</td>
</tr>
<tr>
<td>(h) NAFNet Basic Block + HFB + Down-and-Up</td>
<td>30.0K</td>
<td>181.55</td>
<td>34.00</td>
<td>0.8829</td>
<td>14.92</td>
</tr>
<tr>
<td>(i) NAFNet Basic Block + PixelUnshuffle + Down-and-Up</td>
<td>30.6K</td>
<td>267.54</td>
<td>34.01</td>
<td>0.8828</td>
<td>17.61</td>
</tr>
</tbody>
</table>

In this section, we analyze the correlation between fidelity improvement and runtime consumption for different network aspects. Our studies are conducted for \(\times2\) SR on the 4K RTSR [10] benchmark.

Non-learnable HF extraction. As mentioned in Sec. 3.1, we explore two widely-used and efficient approaches to extract HF details from images. While learning-based techniques have shown significant advancements over traditional hand-crafted methods, the overall efficiency of the model is crucial for our use case. This ablation study aims to determine which approach is the most effective in extracting valuable HF information to compensate for the feature downscaling inside the DFEB. The findings are showcased in Tab. 2a, indicating that the application of Gaussian blur not only results in more visually meaningful information but also improves the quantitative performance. As a result, we incorporate the Gaussian blur approach for extracting HF components into our final method.

Simplifying NAFNet’s basic block. In Sec. 3.1, we explained our decision to use the basic block proposed by NAFNet [7] as a starting point and detailed the modifications we made to arrive at our final version. In this experiment, we aim to investigate the impact of each modification on both the total runtime and the reconstruction performance of our approach. Our findings, presented in Tab. 2b using the 4K RTSR [10] benchmark, reveal that including normalization results in a significant increase in

Table 2. Model architecture. We present the PSNR and SSIM results of the S-variant of our method on the full RGB test samples of the 4K RTSR benchmark [10].

<table>
<thead>
<tr>
<th>Scale Method</th>
<th>PSNR (dB,↑)</th>
<th>SSIM (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\times2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT4KSR-S + Down-and-Up</td>
<td>34.17</td>
<td>0.8844</td>
</tr>
<tr>
<td>RT4KSR-S + Gaussian Blur</td>
<td>34.20</td>
<td>0.8848</td>
</tr>
<tr>
<td>(\times3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT4KSR-S + Down-and-Up</td>
<td>31.70</td>
<td>0.8295</td>
</tr>
<tr>
<td>RT4KSR-S + Gaussian Blur</td>
<td>31.72</td>
<td>0.8297</td>
</tr>
</tbody>
</table>

(b) Ablation on the basic block.

<table>
<thead>
<tr>
<th>Scale Method</th>
<th>Runtime (ms,↓)</th>
<th>Score (↑)</th>
<th>PSNR (dB,↑)</th>
<th>SSIM (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\times2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plain</td>
<td>05.19</td>
<td>34.16</td>
<td>0.8842</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>05.54</td>
<td>34.15</td>
<td>0.8841</td>
<td></td>
</tr>
<tr>
<td>LayerNorm [3]</td>
<td>07.09</td>
<td>34.20</td>
<td>0.8848</td>
<td></td>
</tr>
<tr>
<td>(\times3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plain</td>
<td>02.83</td>
<td>31.66</td>
<td>0.8285</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>02.98</td>
<td>31.65</td>
<td>0.8283</td>
<td></td>
</tr>
<tr>
<td>LayerNorm [3]</td>
<td>03.74</td>
<td>31.72</td>
<td>0.8297</td>
<td></td>
</tr>
</tbody>
</table>

(c) \(\times2\) and \(\times3\) results in the NTIRE 2023 4K RTSR challenge [10].
runtime, but yields the best reconstruction accuracy in terms of PSNR and SSIM. This result holds consistently across both upscaling scenarios from 720p (×3) and 1080p (×2) to 4K resolution. Our approach offers a practical trade-off between runtime and accuracy, besides adjusting the network’s depth or width. For our final contribution, we decide to incorporate the LayerNorm [3] operation for improved reconstruction fidelity.

**Training on diverse contents.** The novel 4K RTSR [10] features HR high-quality images from a variety of sources. Consequently, we enhance our training data with various sources of content, such as GTA5 [43] and LSDIR [32], in addition to the conventional DIV2K [1] and Flickr2K [45] datasets. To keep the dataset size reasonable, we include not more than 2500 images from GTA5 [43] and 1000 images from LSDIR [32]. In Tab. 3a, we present the performance results of our S-variant model trained on various dataset configurations. We find that by including a subset of LS- DIR [32] in our training data, we observe a marginal improvement in performance compared to training solely on DIF2K [1, 45]. We experiment the same behaviour with the inclusion of a random subset of 1000 images of gaming content from the GTA5 [43]. We attribute this to (i) the fact that unlike photorealistic datasets, GTA5 [43] does not offer high-resolution images exceeding [1914 × 1052], (ii) the constrained model complexity -which acts as self-regularization- does not allow to exploit the variety and abundance of data [32] during training.

**Increasing the model complexity.** This ablation study aims to explore the scalability of our method by increasing the model complexity, with the trade-off of longer runtime for improved model performance. We enhance our model by increasing the number of blocks B and channels C. During inference, we report the runtime and number of channels of the reparameterized model. However, at training time, the number of channels within the RRB doubles. As illustrated in Tab. 3b, our findings reveal that by considering reduced runtime, we can significantly enhance the reconstruction performance of our method in terms of PSNR and SSIM, with the XL-variant delivering the best results. Nonetheless, our current shallow architecture demonstrates limitations in simply increasing its size, indicating that more sophisticated approaches must be employed to effectively benefit from a larger model complexity.

**NTIRE 2023 4K RTSR challenge.** In Tab. 2c, we present the results of our S-sized variant for both tracks of the challenge [10]. In addition to the standard SR metrics and the runtime per image, the participating teams are evaluated and ranked by the score function described in [10]. Unlike other proposed solutions, we do not employ multiple training stages and extensive hyperparameter search. Our primary objective in this study is to provide a detailed account of how to develop a competitive baseline for 4K real-time SR while examining various architectural design choices.

**Visual comparison.** In Fig. 7, we show extracted crops from the 4K RTSR benchmark [10]. Also in Fig. 6 we provide SR results on a real 60MP image. Our model shows strong performance in reconstructing missing HF components from the LR input. Although our results still have room for improvement in dealing with shiny areas primarily found in computer-generated content, they produce sharper and visually more appealing outputs despite the presence of checkerboard artifacts.

### 4.3. Comparison to State of the Art

To ensure a fair comparison with published work, we exclusively train the XL and XXXL variants of our method for ×2 and ×4 SR on the DIV2K and Flickr2K datasets. Additionally, for ×4 SR, we trained 64 × 64 crops, following the widely accepted training schedule in the SR literature. Attending to Tab. 4 our models are, on average, **755% smaller** than the approaches we compare them to, even those considered “lightweight,” our performance is still impressive. While there is still a measurable gap, we are able to close it significantly in cases such as ×4 SR on Set14 [55].

---

<table>
<thead>
<tr>
<th>Scale</th>
<th>Dataset</th>
<th>PSNR (dB)</th>
<th>SSIM (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>×2</td>
<td>DIV2K + Flickr2K (DIF2K)</td>
<td>34.18</td>
<td>0.8844</td>
</tr>
<tr>
<td></td>
<td>DIF2K + LSDIR</td>
<td>34.20</td>
<td>0.8848</td>
</tr>
<tr>
<td></td>
<td>DIF2K + GTA5</td>
<td>34.20</td>
<td>0.8850</td>
</tr>
<tr>
<td></td>
<td>DIF2K + LSDIR + GTA5</td>
<td>34.21</td>
<td>0.8857</td>
</tr>
<tr>
<td>×3</td>
<td>DIV2K + Flickr2K (DIF2K)</td>
<td>31.71</td>
<td>0.8293</td>
</tr>
<tr>
<td></td>
<td>DIF2K + LSDIR</td>
<td>31.74</td>
<td>0.8297</td>
</tr>
<tr>
<td></td>
<td>DIF2K + GTA5</td>
<td>31.73</td>
<td>0.8297</td>
</tr>
<tr>
<td></td>
<td>DIF2K + LSDIR + GTA5</td>
<td>31.75</td>
<td>0.8300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th># B</th>
<th># C</th>
<th>Runtime (ms,↓)</th>
<th>PSNR (dB,↑)</th>
<th>SSIM (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT4KSR-XXXS</td>
<td>2</td>
<td>24</td>
<td>05.18</td>
<td>34.13</td>
<td>0.8837</td>
</tr>
<tr>
<td>RT4KSR-XS</td>
<td>34</td>
<td></td>
<td>06.21</td>
<td>34.17</td>
<td>0.8842</td>
</tr>
<tr>
<td>RT4KSR-S</td>
<td>4</td>
<td>24</td>
<td>07.09</td>
<td>34.20</td>
<td>0.8848</td>
</tr>
<tr>
<td>RT4KSR-M</td>
<td>34</td>
<td></td>
<td>08.71</td>
<td>34.20</td>
<td>0.8849</td>
</tr>
<tr>
<td>RT4KSR-L</td>
<td>24</td>
<td>6</td>
<td>09.01</td>
<td>34.21</td>
<td>0.8851</td>
</tr>
<tr>
<td>RT4KSR-XL</td>
<td>32</td>
<td>24</td>
<td>11.22</td>
<td>34.26</td>
<td>0.8857</td>
</tr>
<tr>
<td>RT4KSR-XXL</td>
<td>8</td>
<td>24</td>
<td>10.92</td>
<td>34.23</td>
<td>0.8857</td>
</tr>
<tr>
<td>RT4KSR-XXXL</td>
<td>32</td>
<td>32</td>
<td>13.71</td>
<td>34.25</td>
<td>0.8857</td>
</tr>
</tbody>
</table>
Table 4. Quantitative comparison with state-of-the-art. We compare RT4KSR-XL and RT4KSR-XXXL to published lightweight image SR methods and report SSIM and PSNR (Y) for $\times 2$ and $\times 4$ on standard benchmarks. Model sizes are compared w.r.t RT4KSR-XXXL.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Params</th>
<th>Set5 [4] PSNR (dB) ↑</th>
<th>SSIM ↑</th>
<th>Set14 [55] PSNR (dB) ↑</th>
<th>SSIM ↑</th>
<th>BSD100 [41] PSNR (dB) ↑</th>
<th>SSIM ↑</th>
<th>Urban100 [21] PSNR (dB) ↑</th>
<th>SSIM ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>LapSRN [29]</td>
<td>251K (+228%)</td>
<td>37.52 31.54 .9591 .8850 32.99 29.19 .9124 .7720 31.80 27.32 .8952 .7280 30.41 25.21 .9103 .7560</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARN [2]</td>
<td>1.992K (+1442%)</td>
<td>37.76 32.13 .9590 .8937 33.52 28.60 .9166 .7806 32.09 27.58 .8978 .7349 31.92 26.07 .9256 .7837</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMDN [22]</td>
<td>694K (+629%)</td>
<td>38.00 32.21 .9605 .8948 33.63 28.58 .9177 .7811 32.19 27.56 .8996 .7353 32.17 26.04 .9283 .7838</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LatticeNet [39]</td>
<td>756K (+685%)</td>
<td>38.15 32.30 .9610 .8962 33.78 28.68 .9193 .7830 32.25 27.62 .9005 .7367 32.43 26.25 .9302 .7873</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SwinIR [35]</td>
<td>878K (+795%)</td>
<td>38.14 32.44 .9611 .8976 33.86 28.77 .9206 .7858 32.31 27.69 .9012 .7406 32.76 26.47 .9340 .7980</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT4KSR-XL</td>
<td>91.8K (-17%)</td>
<td>36.83 30.43 .9545 .8600 33.46 28.02 .9197 .7806 31.76 27.09 .8935 .7213 30.75 25.83 .8955 .7208</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT4KSR-XXXL</td>
<td>110.4K</td>
<td>36.92 30.45 .9550 .8610 33.51 28.04 .9202 .7814 31.82 27.11 .8943 .7222 30.85 25.86 .8971 .7221</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6. Qualitative samples. Super-Resolution results on a real 60MP photography. Our method can recover structural elements and textures while being extremely efficient.

Figure 7. Rendered samples from 4K RTSR [10] benchmark.

5. Conclusion

In this paper, we provide a comprehensive analysis of super-resolution techniques for efficiently upscaling images to 4K resolution from 720p and 1080p. To address this, we started with a simple, yet effective baseline architecture and derived a competitive design by focusing on extracting important high-frequency details and downsizing feature maps for efficiency. Over-parameterization during training allowed us to learn more expressive features and transfer knowledge into inexpensive $3 \times 3$ convolutions at inference time using structural re-parameterization. Our proposed method reduces significantly the overall computational footprint in comparison to previous approaches and achieves high reconstruction fidelity on the new 4K RTSR benchmark and other standard SR test sets.

Acknowledgements. This work was supported by the Humboldt Foundation and Sony Interactive Entertainment.
References


[50] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Ec-net: Efficient channel at-


