QuickSRNet: Plain Single-Image Super-Resolution Architecture for Faster Inference on Mobile Platforms

Guillaume Berger* Manik Dhingra* Antoine Mercier Yashesh Savani Sunny Panchal Fatih Porikli
Qualcomm AI Research†
{guilberg, manidhin, amercier, ysavani, sunnpanc, fporikli}@qti.qualcomm.com

Abstract

In this work, we present QuickSRNet, an efficient super-resolution architecture for real-time applications on mobile platforms. Super-resolution clarifies, sharpens, and upscales an image to higher resolution. Applications such as gaming and video playback along with the ever-improving display capabilities of TVs, smartphones, and VR headsets are driving the need for efficient upscaling solutions. While existing deep learning-based super-resolution approaches achieve impressive results in terms of visual quality, enabling real-time DL-based super-resolution on mobile devices with compute, thermal, and power constraints is challenging. To address these challenges, we propose QuickSRNet, a simple yet effective architecture that provides better accuracy-to-latency trade-offs than existing neural architectures for single-image super-resolution. We present training tricks to speed up existing residual-based super-resolution architectures while maintaining robustness to quantization. Our proposed architecture produces 1080p outputs via 2× upscaling in 2.2 ms on a modern smartphone, making it ideal for high-fps real-time applications.

1. Introduction

Single-image super-resolution (SR) refers to a family of techniques that recover a high-resolution (HR) image I_{HR} from its low-resolution (LR) counterpart I_{LR}. In recent years, deep learning (DL) based approaches have become increasingly popular in the field [6, 10, 11, 20, 24, 27, 28, 34, 35], producing impressive results compared to interpolation-based techniques and hand-engineered heuristics (see Fig. 2). However, most existing DL-based super-resolution solutions are computationally intensive and not suitable for real-time applications requiring interactive frame rates, such as mobile gaming. While DL-based super-resolution has been successfully applied to gaming on high-end GPU desktops [7, 26], neural approaches are still impractical for mobile gaming due to their high latency and computational costs. For example, a DL-based architecture such as EDSR [24] takes 75 ms to upscale a 540p image to 1080p on a state-of-the-art mobile AI accelerator. This has driven the need for efficient DL-based super-resolution solutions [2, 5, 12, 13, 38] that can be used in real-time applications such as video gaming, where responsiveness and higher frame rates are essential.

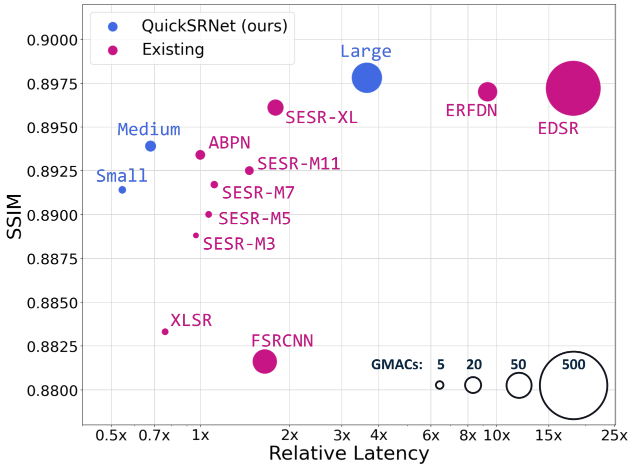
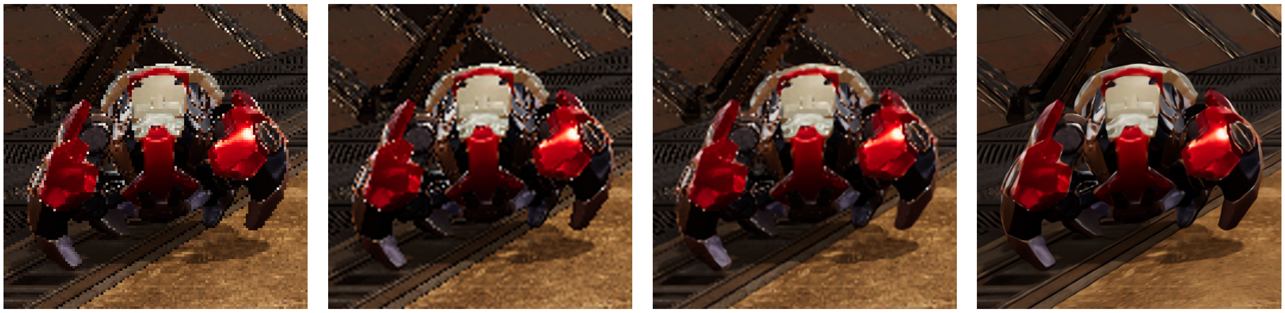


Figure 1. Accuracy-to-latency trade-offs of QuickSRNet (ours) against existing SISR architectures. We report accuracy after 8-bit quantization and measure latency on a state-of-the-art mobile AI accelerator.

In this work, rather than trying to achieve the state-of-the-art PSNR or SSIM scores on standard super-resolution benchmarks, we aim to develop efficient architectures that are suitable for high-fps real-time applications on mobile devices. To this end, we propose QuickSRNet, a simple single-image super-resolution neural network that obtains better accuracy-to-latency trade-offs than existing efficient SR architectures. In particular, we make the following key contributions:

[*] Contributed equally.

[†] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.



(a) LR (b) Interpolation (c) Algorithmic approach (d) ML-based approach

Figure 2. Side-by-side comparison of various upscaling approaches: (a) Low-resolution, (b) Bicubic interpolation, (c) Non-ML baseline: FSR1.0 [14], (d) ML baseline: EDSR [24].

- We streamline the network architecture, reduce the impact of residual connection removal, and ultimately demonstrate the effectiveness of simpler designs in achieving high levels of accuracy and on-device performance.
- We compare a wide variety of architectures in terms of on-device latency, measured on a device with Snapdragon® 8 Gen 1 Mobile Platform, instead of FLOPS count, which is not a reliable indicator of on-device performance [18].
- We measure accuracy *after* 8-bit quantization, a necessary step for better efficiency on mobile platforms, and describe architectural tricks that improve robustness to quantization.
- We apply our proposed architecture to a real-world use-case (video gaming) and compare its visual quality against that of a well-known industrial non-ML based approach (AMD's FidelityFX Super Resolution (FSR1.0) algorithm [14]).
- We describe an approach to perform $1.5\times$ upscaling, a setting that is occasionally used in gaming and XR use-cases but not trivially supported by SR architectures whose upscaling step is based on a sub-pixel convolution [35].

2. Related work

Several efficient SR architectures have been proposed recently. Overall, these architectures share many characteristics with the earlier work by [11] and [35] on FSRCNN and ESPCN respectively: they are usually fully convolutional, use a relatively small number of layers and channels, all layers run at the input resolution and the final output is mapped to higher resolution using a subpixel convolution[1]. Compared to these baselines, more recent approaches have incorporated the following changes:

XLSR [2] uses grouped convolutions to reduce the computational footprint of the architecture and "clipped" ReLU activations to improve robustness to quantization.

ABPN [13] employs a VGG-like convnet [37] (i.e. consisting of only 3×3 Conv-ReLU blocks) with an "anchor-based" input-to-output residual connection. This "anchor-based" connection adds a channel-wise nearest-neighbor upscaled version of the input to the output before the final depth-to-space operation. We confirmed that this channel-wise implementation runs faster on our profiling device than the more common approach of adding the spatially-upsampled input directly to the output. Thus, we follow the same strategy to implement input-to-output residual connections in all our experiments.

SESR [5] leverages linear over-parameterized residual modules which are collapsed into regular convolutions during inference for improved on-device performance. Other modifications include the use of long residual connections.

RepSR [38] investigates training VGG-like super-resolution architectures. Like ABPN, their convnet is equipped with a nearest-neighbor upsampling-based input-to-output connection. Similar to SESR, they find that using over-parameterized networks during training can boost accuracy. They propose a training scheme for using Batch Normalization (BN) layers [22] without introducing artifacts in flat regions of the image, a typical side effect of BN when employed for super-resolution. At test time, the over-parameterized, BN-equipped network is collapsed into a simpler, more efficient network.

[1] In the rest of the paper, we will use the term *depth-to-space* operation. In practice, a subpixel convolution amounts to performing a regular convolution producing $3 \times S^2$ low-resolution channels, where S is the scaling factor, followed by a *depth-to-space* operation to map to higher resolution.

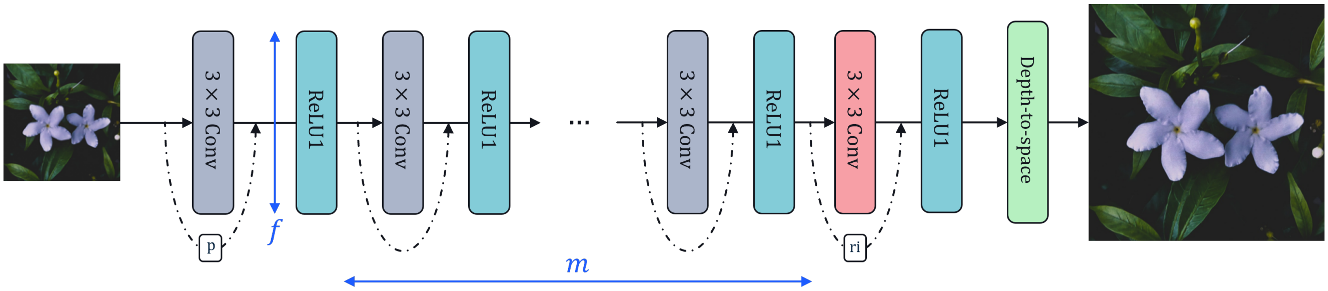


Figure 3. QuickSRNet architecture. We use the convention *QuickSRNet-fX-mY* to refer to the architecture variant that has Y intermediate conv layers and X feature channels. We use dotted lines to illustrate that the conv layers are initialized using an identity initialization scheme. In practice, these skip connections are incorporated into the weights of the corresponding conv module. p and ri stand for "partial" and "repeat-interleaving" respectively (see Sec. 3.2 for more details).

Architecture	PSNR (dB)		Latency (ms)
	FP16	INT8	
ABPN	31.84 (baseline)	31.80 (baseline)	2.17 (baseline)
Res.-free ABPN	31.75 (−0.09)	31.50 (−0.30)	1.42 (−35%)
QSRNet-Med	31.82 (−0.02)	31.77 (−0.03)	1.42 (−35%)

Table 1. On the accuracy and latency impacts of removing the input-to-output residual connection from the ABPN architecture. We report PSNR numbers obtained on BSD100 via 2× upscaling. Latency numbers were obtained on a device with Snapdragon 8 Gen 1 using an input resolution of 512 × 512.

Residual learning for super-resolution Many SR architectures utilize a long skip connection which adds an upscaled version of the input $U(I_{LR})$ directly to the output. Efficient architectures (like [13, 38]) will often implement U as nearest-neighbour interpolation. During training, SR architectures equipped with this technique are implicitly optimized to produce a residual $R = I_{HR} - U(I_{LR})$. One benefit is that the network produces reasonable outputs right after initialization which stabilizes training. Additionally, the input-to-output connection makes the architecture significantly more robust to quantization, as discussed in the next section.

3. Methodology

This section contains a detailed description of QuickSRNet as well as implementation details. The process for developing our proposed SR architecture began with preliminary experiments, which we present in the next paragraph.

3.1. On the impact of removing the input-to-output residual connection

VGG-style architectures such as ABPN [13] or RepSR [38] are already well-optimized, so it is unclear how much faster they can be made on mobile AI accelerators. Instinctively, reducing the number of layers and channels, or replacing 3 × 3 kernels with 1 × 1 kernels, can improve speed at the cost of accuracy. Instead, our experiments investigate how to effectively remove the input-to-output residual connection without affecting accuracy.

As observed by [2, 13], long residual connections can have a large impact on the efficiency of super-resolution architectures, particularly on memory-limited platforms such as smartphones or VR headsets. To confirm this, we trained and profiled a residual-free ABPN variant and found that removing the input-to-output residual connection improves latency by 35%. However, this modification resulted in a marginally lower accuracy and more importantly, reduced robustness to quantization, as can be seen in Tab. 1. A similar trend is evident in the results of the Mobile AI 2022 Challenge [21], where the fastest approaches do not use input-to-output residual connections at the cost of accuracy. To address this, we propose QuickSRNet, a residual-free architecture which is robust to quantization.

3.2. QuickSRNet

Our architecture, QuickSRNet, follows a VGG-like structure with no input-to-output residual connection (see Fig. 3). This architecture is denoted by m, the number of intermediate convolutional blocks, and f, the number of feature channels in those intermediate layers. To increase robustness to quantization, we use a residual learning-motivated initialization scheme along with clipped ReLU activations:

Identity initialization We utilize an intuitive initialization technique where each intermediate convolutional layer simulates a localized skip connection:

$$y = W \circledast x + x \tag{1}$$

where $\circledast$ is the discrete convolution operator and W refers to the kernel weights. In practice, we collapse the

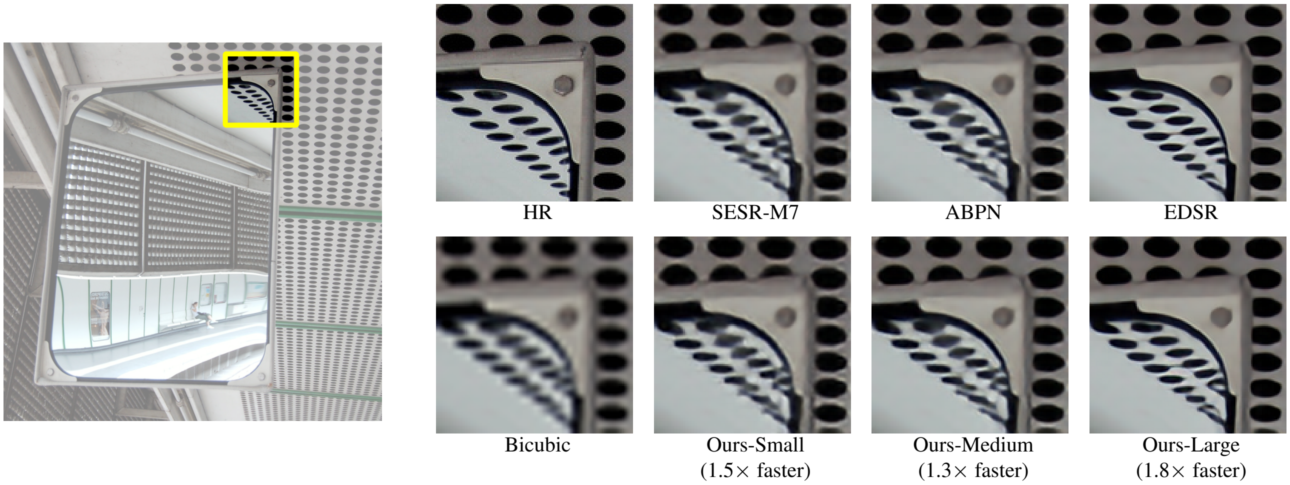


Figure 4. Visualization of 4× super-resolved images from Urban100 produced by our models and existing baselines. Our models match the quality of existing architectures while being significantly faster.

skip connection into the conv module: $y = (W + I) \circledast x = \hat{W} \circledast x$, where $\hat{W}$ are the modified weights after collapse and I is the identity of discrete convolution operators. In this case, collapsing amounts to adding a diagonal of ones to the randomly initialized kernel sliced at the spatial center: $\forall i, W[i, i, c_x, c_y] \mathrel{+}= 1$ (with $c_x = c_y = 1$ assuming a 3×3 kernel). This approach is akin to identity initialization [3, 16, 41, 42] and related to the over-parameterized networks used in [5, 9, 38, 39], except we collapse before training, during initialization.

Equation (1) only works if x and y have the same dimensions and is therefore not directly applicable to the first and last layer of the architecture, as these layers respectively change the number of channels from 3 to f and f to $3 \times S^2$, where S is the scaling factor. For these layers, we modify the initialization scheme as follows:

- **Partial identity initialization**: the 3-channel input to the first convolutional module are added to the first 3 output channels and the other $f - 3$ output channels are left unchanged.

$$y_i = \begin{cases} (W \circledast x)_i + x_i, & \text{if } 0 \leq i < 3 \\ (W \circledast x)_i, & \text{otherwise} \end{cases} \tag{2}$$

- **Repeat-interleaving identity initialization**: the first 3 input channels to the final convolutional module are repeat-interleaved S^2 times and added to the output.

$$y_i = (W \circledast x)_i + x_{round(\frac{i}{s^2})} \tag{3}$$

Similar to Eq. (1), the skip connections described in Eqs. (2) and (3) are incorporated into the corresponding convolutional module by adding ones to the kernel weights at the appropriate location. Intuitively, this initialization technique makes the input image propagate well throughout the entire network. The repeat-interleaving scheme used to initialize the final layer mimics the nearest-neighbour upscaling typically performed in the input-to-output connection of existing residual architectures.

ReLU1 In addition to identity initialization, we found that clipping ReLU activations between 0 and 1 improves robustness to quantization. Compared to XLSR [2], we use ReLU1s throughout the entire network as opposed to just the final layer. Note that for this approach to work well with our id-initialized architecture, it is important to scale input pixels between 0 and 1 (centering around 0 would cause roughly half the pixels propagated by the first id-initialized conv to be zeroed out).

Our experimental results (Sec. 4) show that combining identity initialization and ReLU1 activations significantly improve robustness to quantization.

3.3. Implementation details

Baselines We compare QuickSRNet against the following architectures: FSRCNN [11], ESPCN [35], XLSR [2], SESR [5], ABPN [13], ERFDN [27] and EDSR [24]. Note that, rather than reporting PSNR and SSIM scores from the original papers, we re-implemented, trained and quantized all existing baselines from scratch. As a result, all models shared most hyper-parameters (batch-size, losses, optimizer, etc.), including the data loading/augmentation pipeline. We did however tweak the learning rate for each architecture independently. In some cases, our re-

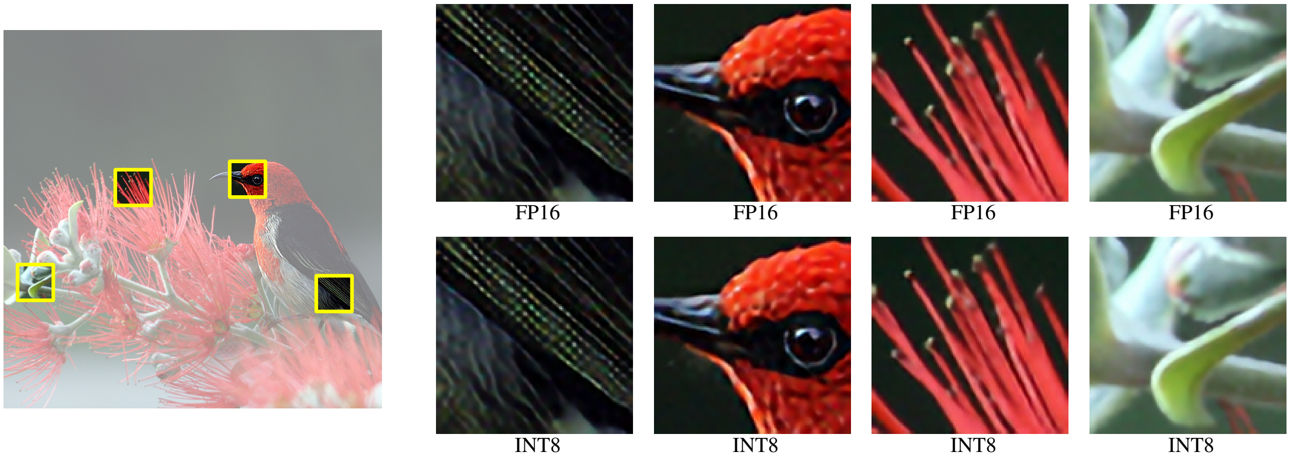


Figure 5. Visual comparison of 4× super-resolved images from DIV2K produced by QuickSRNet-Medium before and after quantization.

implementation deviates slightly from the original architecture when it includes operations that are not supported on the device used for profiling. For example, we replaced the parametric ReLUs [17] used in SESR and FSRCNN to regular ReLUs. Despite these minor modifications, we were usually able to reproduce PSNR and SSIM scores reported in the original papers.

Training details For most experiments, we train the models on the 800 training images from the DIV2K dataset [1] and evaluate them on standard SR testsets: Set5 [4], Set14 [40], BSD100 [29], and Urban100 [19]. We preprocess input and target images by scaling RGB values between 0 and 1. For data augmentation, we use random cropping, flipping and rotation. The models are trained for 1 million iterations with a batch size of 32. We use an L1 loss and the Adam optimizer [23] with hyper-parameters $\epsilon = 10^{-8}$ and $\beta = (0.9, 0.999)$. For the learning rate, we found that using an initial value of 5×10^{-4} and decaying it by a factor of 0.5 every 200K iterations is a strategy that works well for most architectures.

8-bit quantization We use the AIMET library [36] to perform model quantization [32] and compute post-quantization accuracy metrics[2]. Both weights and activations are quantized to 8-bit integers (W8A8 setup). We experimented with both Post-Training Quantization (PTQ) techniques and Quantization Aware Training (QAT). When we use QAT, we re-initialize the optimizer with a very small learning rate (usually 4×10^{-6}).

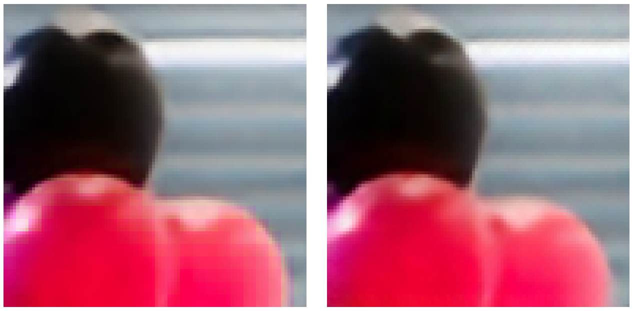


Figure 6. Visual artifacts by ABPN vs QuickSRNet-Medium on 4×-upscaled images from Urban100. More examples can be found in the supplementary material.

On-device profiling We profile the models on the Hexagon Processor of a device with Snapdragon 8 Gen 1 and report the average latency obtained on 100 inputs of spatial resolution 512 × 512. Before profiling, the model is converted from PyTorch [33] to ONNX. Please see the appendix for more details about the model conversion steps.

4. Experimental results

In this section, we compare QuickSRNet against existing SR architectures in terms of accuracy-to-latency trade-offs and demonstrate the effectiveness of our training tricks to improve robustness to quantization through ablation studies.

Scaling laws of QuickSRNet We experimented with several architecture specifications, varying the number of conv modules m and the number of feature channels f. PSNR and SSIM scores on the BSD100 dataset obtained with each specification and a scaling factor of 2 can be found

[2] Additionally, we confirmed accuracy numbers on target for a subset of the models and typically found that the simulated numbers produced by AIMET to be within a 0.02 range from the actual numbers obtained on target.

QuickSRNet specs	**2×**		**3×**		**4×**		**Latency**
	FP16	**INT8**	**FP16**	**INT8**	**FP16**	**INT8**	**Measurements (ms)**
f32 - m1	31.43	31.38	28.41	28.38	26.94	26.91	0.99 (−22%)
f32 - m2 (small)	31.61	31.58	28.57	28.55	27.07	27.06	1.14 (−35%)
f32 - m3	31.72	31.63	28.67	28.63	27.16	27.12	1.21 (−34%)
f32 - m5 (medium)	31.82	31.77	28.75	28.72	27.24	27.21	1.42 (−35%)
f32 - m7	31.88	31.81	28.81	28.76	27.30	27.27	1.74 (−30%)
f32 - m11	31.95	31.80	28.86	28.80	27.35	27.29	2.38 (−22%)
f64 - m11 (large)	32.08	31.97	28.98	28.93	27.47	27.43	7.63 (−7%)

Table 2. PSNRs (dB) and latencies (ms) of various QuickSRNet configurations (f : number of feature channels, m : number of convolutional blocks in the network). We report PSNR numbers obtained before and after quantization. We also report latency measurements on a 512×512 input, obtained on a device with Snapdragon 8 Gen 1, and gains introduced by not using an input-to-output residual connection.

Existing Models	**2×**		**3×**		**4×**		**Latency**
	FP16	**INT8**	**FP16**	**INT8**	**FP16**	**INT8**	**Measurements (ms)**
XLSR	31.62	31.32	28.59	28.31	27.09	26.82	1.59
ESPCN	31.37	30.19	28.35	27.86	26.87	26.44	1.83
SESR-M3	31.57	31.40	28.52	28.47	27.02	26.97	2.02
ABPN	31.80	31.74	28.73	28.70	27.22	27.20	2.09
SESR-M5	31.68	31.53	28.63	28.56	27.11	27.05	2.23
SESR-M7	31.76	31.67	28.68	28.61	27.16	27.07	2.33
SESR-M11	31.84	31.74	28.77	28.60	27.25	27.18	3.06
FSRCNN	31.50	31.22	28.49	28.32	26.99	26.89	3.45
SESR-XL	32.02	31.94	28.91	28.85	27.39	27.35	3.75
ERFDN	32.20	32.06	29.08	28.98	27.57	27.48	19.50
EDSR	32.21	32.08	29.04	28.89	27.61	27.53	37.95

Table 3. PSNRs (dB) and latencies (ms) of existing SISR solutions on BSD100. Please note that we re-implemented, trained, and quantized all architectures from scratch. Latency numbers were measured on a device with Snapdragon 8 Gen 1, using a 512×512 input.

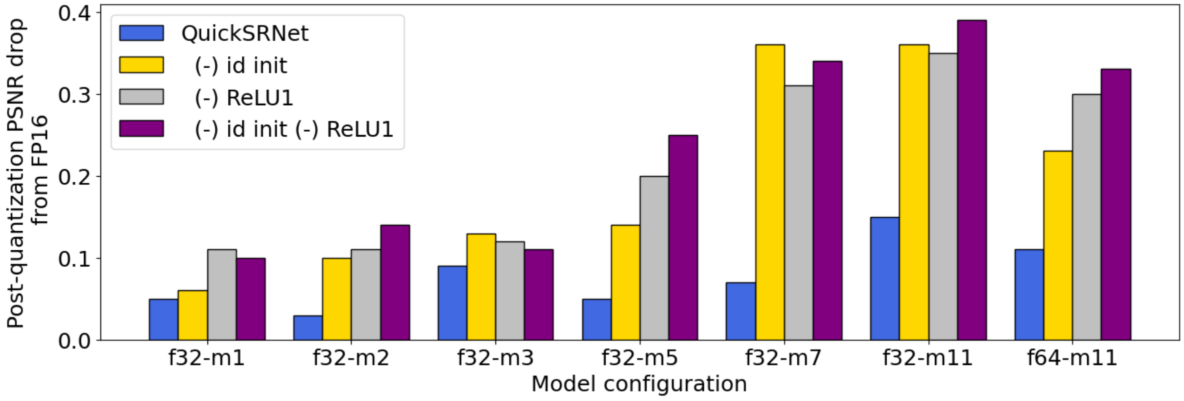
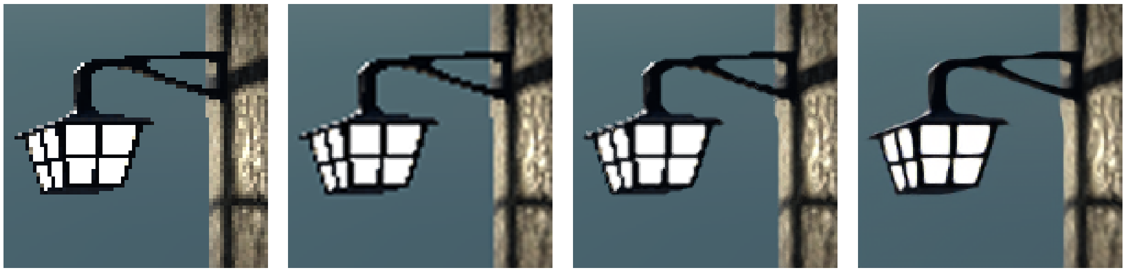


Figure 7. Ablation study comparing the post-quantization PSNR drop from FP16 when removing identity initialization and/or ReLU1 activations from the architecture design.

Specification	**Post-training FP16**	**No optimizations INT8**	**QAT INT8**	**Per-channel QAT INT8**	**Per-channel Adaround INT8**
QuickSRNet-Small	31.61	30.81 (−0.80)	31.34 (−0.27)	**31.57** (**−0.04**)	31.56 (−0.05)
QuickSRNet-Medium	31.82	30.74 (−1.08)	31.61 (−0.21)	31.75 (−0.07)	**31.77** (**−0.05**)
QuickSRNet-Large	32.07	31.37 (−0.70)	31.90 (−0.10)	31.97 (−0.10)	**31.99** (**−0.08**)

Table 4. Impact of various quantization techniques on accuracy. Activations are always quantized to 8-bit integers using per-tensor quantization. For weights, we tried both per-tensor and per-channel quantization and found the latter to work significantly better.



(a) LR (b) Bicubic (c) FSR1.0 (d) QuickSRNet-Small

Figure 8. SISR (2×) for Gaming: (a) Low-resolution, (b) Bicubic interpolation, (c) FSR1.0 [14], and (d) QuickSRNet-Small (ours)

Method	PSNR	SSIM
Bicubic	28.88	0.8683
FSR1.0	29.01	0.8707
QuickSRNet-Small	29.71	0.8806

Table 5. PSNR/SSIM scores for different 2× single-image super-resolution solutions for gaming.

in Tab. 2. As expected, larger/wider networks obtain higher PSNR/SSIM scores. The measured latency for each specification is reported on the last column and we indicate for each architecture the latency improvement introduced by removing the input-to-output connection. In the rest of the paper, we only use a subset of these model configurations: QuickSRNet-small (a.k.a. *QuickSRNet-f32-m2*), QuickSRNet-medium (a.k.a. *QuickSRNet-f32-m5*), and QuickSRNet-large (a.k.a. *QuickSRNet-f64-m11*). As can be seen in Fig. 1 and Tabs. 2 and 3, these 3 variants obtain similar accuracy scores in contrast to SESR-M7, ABPN and EDSR respectively while being significantly faster. Please refer to Fig. 4 for a comparison of QuickSRNet-{small, medium, large} versus SESR-M7, ABPN and EDSR in terms of image quality and latency improvement. More quantitative results and side-by-side comparisons for 2×, 3× and 4× upscaling are available in the supplementary materials.

W8A8 quantization For all our experiments, we quantize both, model weights and activations, to 8-bit integers. Without any optimizations, we observe a significant drop after quantization (see Tab. 4). While finetuning the per-tensor quantized weights via QAT can recover some of this drop, we found per-channel weight quantization to be important.

Furthermore, we experimented with several post-training quantization methods, including: cross-layer equalization (CLE) [31], bias correction (BC) [31], and adaptive rounding (AdaRound) [30], and found AdaRound to obtain comparable performance to per-channel QAT, outperforming the other PTQ approaches. CLE did not work well in our experiments, most likely because it skews the activation values outside the ReLU1 range. In an attempt to further improve post-quantization accuracy, we tried finetuning the per-channel Adarounded weights using QAT but this did not improve post-quantization accuracy.

Target resolution	*540p*	*720p*	*1080p*	*1440p*	*2160p*
Latency (ms)	0.69	0.95	2.24	4.25	8.15

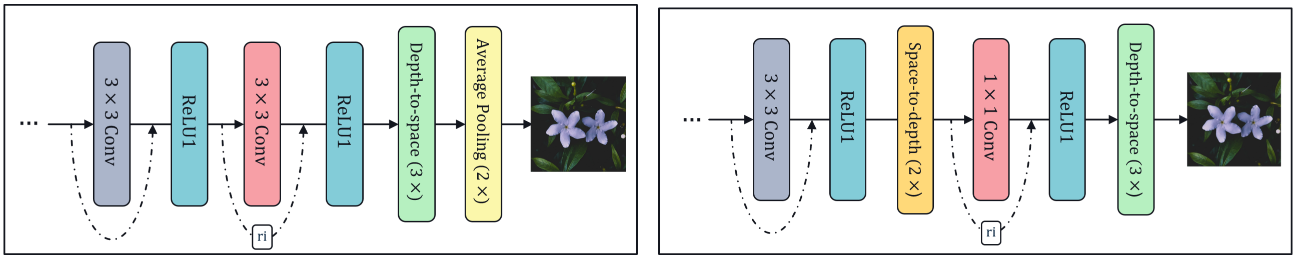
Table 6. QuickSRNet-Small latency (ms) running at different target resolutions on a device with Snapdragon 8 Gen 1.

Robustness to quantization Overall, QuickSRNet quantizes well to W8A8. As can be seen in Fig. 5, images produced by the quantized model are indistinguishable from their full-precision counterparts. In Fig. 7, we visualize the drop in PSNR post quantization and show the benefits of combining identity initialization and ReLU1 activations. Regardless of the model size, removing one or both of these ingredients from the model design results in a significantly worse accuracy after quantization.

Less prone to block artifacts Our experiments show that architectures with a nearest-neighbour upsampling skip connection tend to produce outputs with block-like artifacts of size $S \times S$. Interestingly, our residual-free architecture seems less prone to this issue and produces more perceptually pleasing results. A visual comparison of such artifacts can be seen in Fig. 6.

5. DL-based SISR for mobile gaming

A real-world application of efficient super-resolution is video gaming. While DL-based super-resolution (or supersampling) has already been commericalized on high-end gaming desktops [7, 26], these solutions are not supported on mobile platforms yet. One specificity of gaming content is that synthetically rendered images are significantly more aliased than natural images. Nevertheless, we find *QuickSRNet-Small* to work well on this domain, with no changes needed apart from re-training it on



(a) Naïve baseline

(b) Proposed approach

Figure 9. Two different architecture modifications to implement 1.5× upscaling: (a) Naïve approach, where we repurpose a 3× architecture by adding an average pooling layer on top, (b) Our approach, where we halve the resolution inside the network and map to target resolution using a 3× subpixel conv.

QuickSRNet Specification	Bicubic	Naïve Baseline	Proposed Approach
Small	32.47	34.71	34.89
Medium		34.87	35.13
Large		35.18	35.47

Table 7. PSNRs (dB) evaluated after quantization on BSD100 dataset via 1.5× upscaling

gaming data. Figure 8 shows some results obtained by *QuickSRNet-Small* when applied to gaming content. We compare our results against non-ML based single-frame upscaling approaches, including an FSR1.0 baseline [14] which was specifically designed for this use case. Overall, we find that *QuickSRNet-Small* produces better-looking images compared to the other baselines. The visual benefits also translate into PSNR and SSIM gains, as can be seen in Tab. 5. In terms of latency, Tab. 6 shows *QuickSRNet-Small* latency measurements at various target resolutions, from 540p to 4k. In the future, we would like to extend our architecture to the multi-frame case which has become the de facto standard for video gaming (*e.g*. FSR 2.0, [15], DLSS 2.0 [25], XeSS [8]).

5.1. QuickSRNet 1.5×

Standard super-resolution datasets are usually limited to 2×, 3× or 4× upscaling and non-integer scaling factors are rarely explored. On the other hand, 1.5× upscaling is often proposed in VR and gaming applications[3]. In this section, we describe an approach to perform 1.5× upscaling, a setting that is not trivially supported by most efficient SR architectures as non-integer scaling factors are not compatible with the final sub-pixel convolution.

3× upscaling followed by 2× downscaling baseline A naïve approach to 1.5× upscaling consists in downscaling by a factor 2 the output of a 3× SR model. This can be achieved by adding a 2×2 average pooling layer at the end of the architecture.

Proposed 1.5× upscaling approach Instead, we propose to halve the resolution inside the network using a *space-to-depth* operation with a block-size of 2 which we then map to target resolution using a 3× subpixel convolution. To compensate for the 4× increase of channels due to the *space-to-depth* operation, we implement the subpixel convolution using a 1×1 kernel.

Figure 9 shows the two considered 1.5× architecture heads. As can be seen in Tab. 7, the proposed approach significantly outperforms the naïve 3× upscaling followed by 2× downscaling baseline.

6. Conclusion

In this study, we propose QuickSRNet, an efficient super-resolution architecture for mobile platforms. We have thoroughly analyzed the performance of our models and existing ones, systematically checking accuracy after quantization and profiling latency on a mobile device. Our experiments have shown that QuickSRNet is well suited for real-time applications on mobile devices due to its high speed and good accuracy. We have also demonstrated the effectiveness of our solution on a real world use case (mobile gaming) and believe that our training tricks to improve robustness to quantization are applicable to other works. We have released the implementation and pretrained weights (including quantized weights) of QuickSRNet models as part of the AIMET model zoo[4]. We believe that QuickSRNet provides a practical solution for applications that require real-time super-resolution capabilities.

[3] Both DLSS and FSR support 1.5× via their “Quality” mode.

[4] For QuickSRNet-large, the released version of the model includes the input-to-output residual connection as this leads to slightly higher accuracy and the latency improvement (-7%) is minimal for larger architectures.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 126–135, 2017. 5

[2] Mustafa Ayazoglu. Extremely lightweight quantization robust real-time single-image super resolution for mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2472–2479, 2021. 1, 2, 3, 4

[3] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pages 1352–1361. PMLR, 2021. 4

[4] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012. 5

[5] Kartikeya Bhardwaj, Milos Milosavljevic, Liam O'Neil, Dibakar Gope, Ramon Matas, Alex Chalfin, Naveen Suda, Lingchuan Meng, and Danny Loh. Collapsible linear blocks for super-efficient super resolution. *Proceedings of Machine Learning and Systems*, 4:529–547, 2022. 1, 2, 4

[6] X Chen, X Wang, J Zhou, and C Dong. Activating more pixels in image super-resolution transformer. arxiv 2022. *arXiv preprint arXiv:2205.04437*. 1

[7] Hisham Chowdhury, Rense Robert Kawiak, Gabriel Ferreira de Boer, and Lucas Xavier. Intel xess-an ai based super sampling solution for real-time rendering.(2022). In *Game Developers Conference*, 2022. 1, 7

[8] Hisham Chowdhury, Rense Robert Kawiak, Gabriel Ferreira de Boer, and Lucas Xavier. Intel xess-an ai based super sampling solution for real-time rendering.(2022). In *Game Developers Conference*, 2022. 8

[9] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021. 4

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015. 1

[11] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 391–407. Springer, 2016. 1, 2, 4

[12] Tingxing Tim Dong, Hao Yan, Mayank Parasar, and Raun Krisch. Rendersr: A lightweight super-resolution model for mobile gaming upscaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3087–3095, 2022. 1

[13] Zongcai Du, Jie Liu, Jie Tang, and Gangshan Wu. Anchor-based plain net for mobile image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2494–2502, 2021. 1, 2, 3, 4

[14] GPUOpen-Effects. Fidelityfx-fsr2/readme.md at master · gpuopen-effects/fidelityfx-fsr2, Oct 2022. 2, 7, 8

[15] GPUOpen-Effects. Fidelityfx-fsr2/readme.md at master · gpuopen-effects/fidelityfx-fsr2, Oct 2022. 8

[16] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016. 4

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 5

[18] Lennart Heim, Andreas Biri, Zhongnan Qu, and Lothar Thiele. Measuring what really matters: Optimizing neural networks for tinyml. *arXiv preprint arXiv:2104.10645*, 2021. 2

[19] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5197–5206, 2015. 5

[20] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *Proceedings of the 27th acm international conference on multimedia*, pages 2024–2032, 2019. 1

[21] Andrey Ignatov, Radu Timofte, Maurizio Denna, Abdel Younes, Ganzorig Gankhuyag, Jingang Huh, Myeong Kyun Kim, Kihwan Yoon, Hyeon-Cheol Moon, Seungho Lee, et al. Efficient and accurate quantized image super-resolution on mobile npus, mobile ai & aim 2022 challenge: report. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 92–129. Springer, 2023. 3

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 2

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[24] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 1, 2, 4

[25] Edward Liu. Dlss 2.0-image reconstruction for real-time rendering with deep learning. In *GPU Technology Conference (GTC)*, 2020. 8

[26] Edward (Shiqiu) Liu, Robert Pottorff, Guilin Liu, Karan Sapra, Jon Barker, David Tarjan, Lei Yang, Kevin Shih, Marco Salvi, Andrew Tao, and Bryan Catanzaro. Dlss 2.0 – image reconstruction for real-time rendering with deep learning. In *Game Developers Conference 2020, GPU Technology Conference 2020*, 2020. 1, 7

[27] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In

Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, pages 41–55. Springer, 2020. 1, 4

[28] Zhi-Song Liu, Li-Wen Wang, Chu-Tak Li, and Wan-Chi Siu. Hierarchical back projection network for image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 1

[29] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001. 5

[30] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020. 7

[31] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019. 7

[32] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 5

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5

[34] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv:2104.07636*, 2021. 1

[35] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 1, 2, 4

[36] Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare. Neural network quantization with ai model efficiency toolkit (aimet). *arXiv preprint arXiv:2201.08442*, 2022. 5

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[38] Xintao Wang, Chao Dong, and Ying Shan. Repsr: Training efficient vgg-style super-resolution networks with structural re-parameterization and batch normalization. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2556–2564, 2022. 1, 2, 3, 4

[39] Sergey Zagoruyko and Nikos Komodakis. Diracnets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388*, 2017. 4

[40] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7*, pages 711–730. Springer, 2012. 5

[41] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019. 4

[42] Jiawei Zhao, Florian Schäfer, and Anima Anandkumar. Zero initialization: Initializing neural networks with only zeros and ones. *arXiv preprint arXiv:2110.12661*, 2021. 4