Efficient Super-Resolution Using MobileNetV3

Haicheng Wang(✉), Vineeth Bhaskara, Alex Levinshtein, Stavros Tsogkas, and Allan Jepson

Samsung AI Centre Toronto, Toronto, Canada
{h.wang1,s.bhaskara,alex.lev,stavros.t,allan.jepson}@samsung.com

Abstract. Deep learning methods for super-resolution (SR) have been dominating in terms of performance in recent years. Such methods can potentially improve the digital zoom capabilities of most modern mobile phones, but are not directly applicable on device, due to hardware constraints. In this work, we adapt MobileNetV3 blocks, shown to work well for classification, detection and segmentation, to the task of super-resolution. The proposed models with the modified MobileNetV3 block are shown to be efficient enough to run on modern mobile phones with an accuracy approaching that of the much heavier, state-of-the-art (SOTA) super-resolution approaches.

Keywords: Super-resolution · Efficient CNN · Mobile digital zoom

1 Introduction

Enhancing the quality of images taken with digital cameras is useful in a broad range of both professional and consumer applications, such as photo editing, or improving the quality of video and image content in TVs and mobile phones, respectively.

One particular form of image enhancement involves taking an image of relatively low starting resolution and "upscaling" it to create an image of higher resolution, allowing us to increase the raw size of the image, while producing fine details. In the context of computer vision and image processing, this task is called *super-resolution* (SR) and has traditionally been tackled in two different ways *Classical* super-resolution algorithms frame SR as an optimization problem, which is solved by minimizing the loss between the low-resolution (LR) input(s) and the projection of the predicted high-resolution (HR) image back to the low-resolution domain, combined with appropriate regularization [11,15,37]. In other works, the input is not a single, but *multiple*, slightly misaligned LR depictions of the same scene, defining a system of linear constraints that produces the underlying HR image, when solved [9,10,20,39]. The common feature of all these methods is that they do not require any training; however, this

H. Wang, V. Bhaskara, A. Levinshtein—Equal contribution.

A. Bartoli and A. Fusiello (Eds.): ECCV 2020 Workshops, LNCS 12537, pp. 87–102, 2020.
https://doi.org/10.1007/978-3-030-67070-2_5

advantage is usually offset by slow runtimes due to iterative optimization or kernel approximation schemes, or limited effectiveness for scale factors larger than $\times 2$.

Example-based super-resolution [3,12,13,23], on the other hand, relies on a dataset of corresponding low-resolution (LR) and high-resolution (HR) images (or patches). A model is trained to map a *single* low-resolution input to a high-resolution output, with a specified upscaling factor. Being fast and able to effectively handle higher scale factors (such as $\times 4$ or $\times 8$) are advantages example-based methods enjoy over classical approches; the caveat is that, inferring the ideal LR-to-HR mapping is an ill-posed problem, because there are multiple HR images that may correspond to a LR input[1].

Nevertheless, in the last few years, we have seen an explosion of example-based methods, especially with the proliferation of deep learning models such as CNNs [7,22,34] and GANs [27,41]. These models are quite effective in generating images of high perceptual quality, but they are often cumbersome, having a large number of parameters and increased time and memory footprint.

These requirements become especially limiting for mobile phone applications. Zoom, for instance, remains a challenge for mobile phones, since the manufacturer must satisfy two conflicting specifications: on one hand, high quality optical zoom requires a physically large lens; on the other hand, the phone device itself must remain compact to improve usability and aesthetics. Digital zoom can potentially be the answer, but the low-power CPUs and GPUs that fit inside a phone are not powerful enough to run the heavier, state of the art SR networks, that are designed for desktop computers[2].

Lifting the need for such compromises is the main motivation behind our submission to the AIM 2020 Efficient SR challenge [44]. Our goal is to propose a deep network that performs comparably to powerful, state of the art models, like ESRGAN [41] or RCAN [46], while being efficient enough to run on a mainstream mobile phone device.

There is already a significant body of research on improving the efficiency of deep CNNs for super-resolution. Some works attempt to design and use more lightweight architectures, without compromising performance [6,35,44], while others introduce new modules that improve performance in a cost-effective way [34]. However, it is still questionable whether these models are applicable to mobile applications, because their efficiency is evaluated on desktop GPUs rather than mobile devices, and the output size at test time is usually smaller than that of a typical photo taken with a mobile phone. In classification and segmentation tasks, MobileNet architectures [16,17,33] demonstrate promising performance-efficiency trade-offs. Besides attractive performance in the

[1] This is why this task is sometimes called *image hallucination*.

[2] ESRGAN [41] takes 2.69 s on a V100 GPU, and 10.46 GB of memory, to generate a 12MP (3000×4000) output – a standard photo size for a mobile camera. Obtaining the same output using mobile phone hardware would be prohibitively slow, or impossible, due to limited memory.

literature, the architectures are also widely adopted as the backbone model for many deep learning applications in computationally constrained environments.

In this work, we adapt the MobileNetV3 [16] architecture design for the SR task, and propose a deep learning method that can achieve good performance while running efficiently on a mobile phone, opening up the possibility for on-device deployment.

We evaluate our approach on 4x super-resolution and show that it approaches the performance of state-of-the-art (SOTA) methods such as ESRGAN and RCAN [41,46] *while being* 40–1000× *and* 30–1000× *more efficient in FLOPs and parameter size respectively.* In addition to achieving a better performance-efficiency trade-off compared to previous methods, we also showcase an "extremely lite" variant of our model. The runtime performance of the latter is fast enough to make direct deployment to mobile devices possible.

2 Related Work

2.1 Deep Learning Models Focusing on Image Quality

Since the seminal work of Dong *et al.* [7], there has been an steadily increasing interest in Single-image Super-Resolution (SISR) using deep neural networks. Network architectures of high representational power [22,27,36,38,41], including models of visual attention [46], have been proposed to improve fidelity. Researchers have also explored various loss objectives [28,32,41] to improve the sharpness of the generated image or make the output more visually pleasing, according to human perception, with tools such as GANs [41] and perceptual losses [21] that have been widely adopted. The caveat, however, is that these trained models have high requirements in terms of memory, processing power, or both, making their direct application to edge devices impossible.

2.2 Efficient Deep Learning Models

To achieve better trade-off between fidelity and efficiency, a separate line of SISR research has been focusing on more cost-aware architecture design. Dong *et al.* [8] speed up their model using an hourglass-shaped CNN architecture. Shi *et al.* [34] and Vu *et al.* [40] develop subpixel and de-subpixel convolution to approach efficient model inference. Ahn *et al.* [2] propose a lighter ResNet architecture with multi-scale cascading connections. Lai *et al.* [25] address the efficiency requirement by progressive reconstruction of the output, using a Laplacian pyramid in the feature branch. Hui *et al.* [19] introduce multiple information distillation blocks, which reduce the computation budget by progressively splitting features into one half that is merged with the output, and another half that is processed further.

Apart from manual architecture adaptation, Neural Architecture Search (NAS) has also been explored for efficient super-resolution. Chu *et al.* [6] employ evolutionary algorithms to get the proper parameters for convolution blocks and

connections among blocks. Song *et al.* [35] utilize a similar approach to determine the optimal location of non-linearities and upsampling layers, while also utilizing efficient Residual Dense blocks.

While these models are a step in the right direction, most of them are still not efficient enough to stay within the runtime and memory constraints on a mobile phone. First, the 12MP target output resolution on a typical mobile phone is much larger than the output size used in research benchmarks; and second, even flagship mobile phones possess hardware that is much less capable, compared to desktop GPUs, both in terms of memory capacity, and processing power. Thus, there is a strong incentive to make models that are even more efficient, and lightweight enough for mobile deployment. We achieve this by a network architecture that can bring better trade-off between FLOPs and fidelity performance, especially, when FLOPs are extremely low.

3 Proposed Method

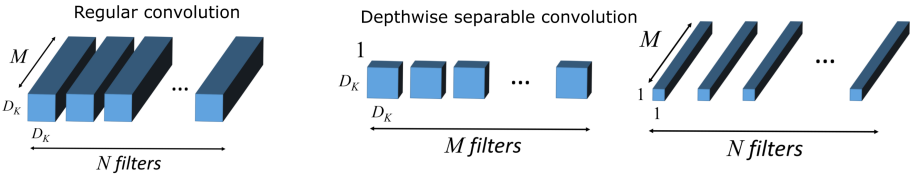
3.1 MobileNet Architecture Overview

MobileNet [16,17,33] is a family of CNN architectures that are specifically designed for deployment on lightweight devices. These architectures have shown promising performance-efficiency trade-offs in computer vision tasks such as classification, detection and segmentation.

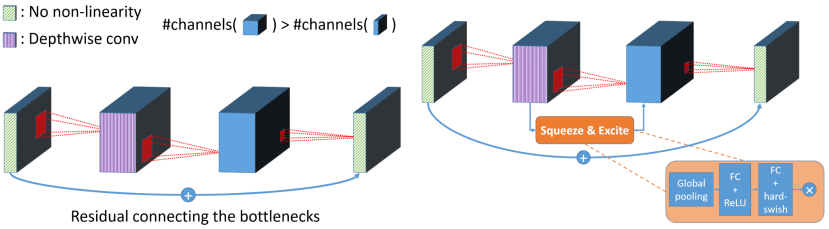
Each member of the MobileNet family builds on its predecessors, bringing new design changes that improve performance. MobileNet [17] introduces the depth separable convolution (Fig. 1**a**) that replaces the standard convolution operation, achieving comparable accuracy performance but with much fewer FLOPs. MobileNetV2 [33](Fig. 1**b**) proposes inverted residuals, expanding features inside a block so that the representation power of the model can be improved in a cost-effective way. MobileNetV3 [16](Fig. 1**c**) additionally introduces the squeeze-excitation (SE) attention module and a new activation function (hard-swish), combined with hardware-aware NAS for hyperparameter tuning.

3.2 Efficient Architecture Using Adapted MobileNetV3 Blocks

We use the most recent MobileNetV3 [16] blocks as the basis for our efficient super-resolution model. Our architecture takes a single LR image as input, and passes it through $N + 1$ modified MobileNetV3 blocks, with a skip connection (addition) from the output of the first block to the output of the last block. Contrary to the findings of Lim *et al.* [28], we found the BatchNorm layers within the MobileNetV3 blocks to be beneficial to performance; we discuss this in more detail in Sect. 4.4. The output of the last block is then upscaled using two pixel-shuffle operations [34] with non-linearities and convolutions. To yield three channels, post-processing consisting of a depth-wise separable convolution together with a 1×1 convolution layer, is applied to the output of the upscaling



(a) Regular vs depthwise separable convolution (MobileNet V1).



(b) Inverted residuals (MobileNet V2). (c) Squeeze excite + hard-swish non-linearity (MobileNet V3).

Fig. 1. Features introduced in different versions of the MobileNet architecture.

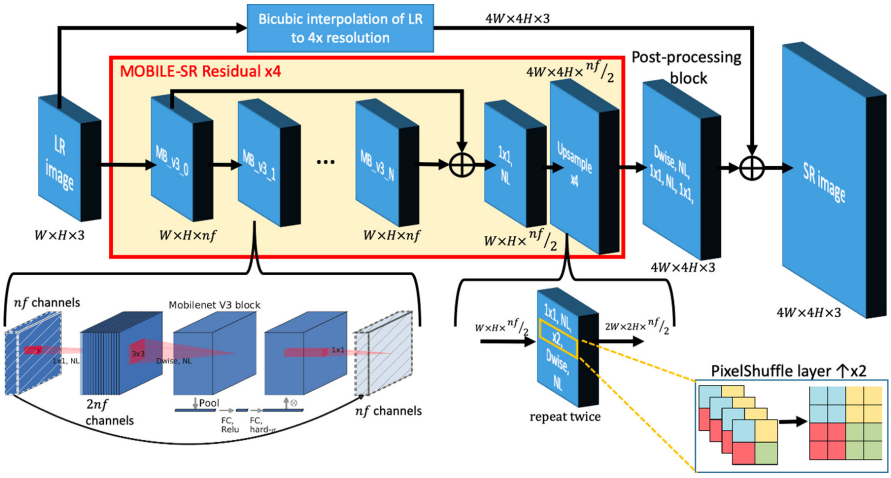


Fig. 2. Method architecture. NL stands for non-linearity, which we set to LeakyReLU with a slope of 0.2 for negative values.

result. Finally, the LR input is bicubically interpolated to the 4x higher resolution and added to the output of the post-processing block using a skip connection. As a result of this skip connection, the main body of the network is tasked with only computing an update to the bicubically interpolated image. Figure 2 provides a visual overview of our architecture.

We modify the MobileNetV3 blocks for SR as follows. Unlike the original MobileNetV3, which performs progressive downsampling to increase the

receptive field size, we operate in the original image resolution, to avoid loss of detail, as it is customary in modern SR architectures [8,41,46]. The expansion factor for the blocks, which determines the number of 1×1 convolution filters, is kept fixed to 2 throughout the layers. In Fig. 2, we are showing how a 1×1 convolution with an expansion factor of 2 expands a nf-channel input feature map to a $2nf$-channel intermediate feature map before the depth-wise convolution layer. We use LeakyReLU activations (with a slope of 0.2 for negative values) in place of the default h-swish non-linearity used in MobileNetV3. The remaining components are identical to the ones used in MobilenetV3, including batch normalization and the SE attention blocks, with a compression ratio of 4.

Other hyperparameters of our network include the number of blocks N and the number of features nf in each block. Although the target metric (PSNR) is a function of the mean squared error, we found training with an L1 loss is still better than an L2 loss. For our final submission, we set $N = 16$, $nf = 72$. We name this network SAM_SR_LITE.

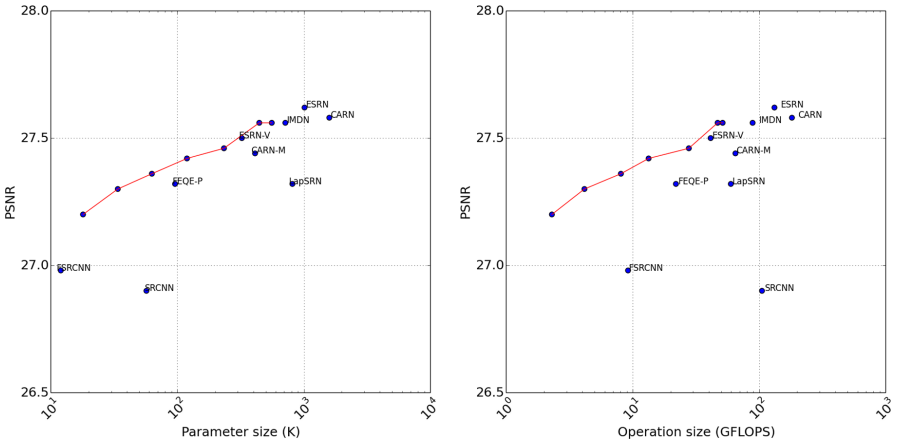


Fig. 3. Comparisons in PSNR-Y on BSD100 dataset w.r.t parameter size and operation count. Dots connected by red lines are our models with different N and nf configurations. From right to left are (nf, N) = (72, 16), (64, 16), (64, 8), (32, 16), (32, 8), (16, 16), (16, 8). Models closer to top left corner are better.

4 Experiments

4.1 Datasets and Implementation Details

Our submitted method (SAM_SR_LITE) was trained on the combined DIV2K (800 images) [1] and FLICKR2K (2680 images) [1] datasets. We use 100×100 low-res image crops, with a batch size of 128, training for a total of 350 K steps. For data augmentation, we use random cropping, flipping and 90° rotations. Learning rate is initialized to 0.005 and halved at steps [100K, 200K, 250K, 300K].

Training is done using the Adam optimizer [24] with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The model is implemented in PyTorch [30].

The results in Table 1 show that SAM_SR_LITE achieves a validation PSNR of 28.976 dB, which is on par with the baseline method for the AIM 2020 Efficient Super-Resolution challenge, namely, MSRResNet [41,45]. This is despite our model using just 18% of the FLOPs and 37% of the parameters (although, using more activations in total) compared to the baseline model. We measure FLOPs based on a LR image of size 256×256, and the memory reported is the peak memory allocated during inference on the DIV2K validation set. Activation size is the total number of elements among all convolutional layers' output tensors [31]. Note that by this measure, dpeth separable convolutions will have twice the number of activations as full convolutions. In this report, we measure runtime on the DIV2K validation set [1] (having an input size of 421×421, on average), with a NVIDIA RTX 2080 GPU (CUDA 10.1, no cuDNN, PyTorch 1.5.1), unless specified otherwise. We see from Table 1 that, for this environment, the runtime of SAM_SR_LITE is about half the baseline's. Samples of qualitative results can be found in Fig. 4.

Table 1. Comparison of our submitted model SAM_SR_LITE with the baseline method MSRResNet [45]. Our model achieves a performance on par with the baseline while requiring much fewer FLOPs, memory, and parameters.

Model	PSNR-RGB on DIV2K val dataset	PSNR-RGB on DIV2K test dataset	FLOPs	Parameters	Runtime	Activation	Memory
Baseline [45]	29.00	28.70	333.32 G	1517 K	0.336 s	292.55 M	4.37 GB
SAM_SR_LITE	28.98	28.71	58.6 G	558 K	0.169 s	576.45 M	2.56 GB

4.2 Trade-Off Between Quality and Efficiency

We have also experimented with lighter versions of our method, the results are summarized in Table 2 and Fig. 3. In particular, when setting $N = 8$ and $nf = 16$, which we call SAM_SR_XLITE (for eXtremely light), our method has 18K parameters (∼1% of the baseline), and achieves an inference time of $0.03s$ (∼8% of the baseline, using RTX2080), at 2.58G FLOPs (∼0.8% of the baseline), with a validation PSNR of 28.38 dB. Being 0.62 dB below the baseline performance, it was not submitted to the challenge. Nevertheless, we feel this model is of significant interest since it exhibits a large gain in computational efficiency (∼130× fewer FLOPs than the baseline) at the cost of a performance loss of 0.5 to 1 dB, where 0.5 dB difference is barely noticeable for casual human observers.

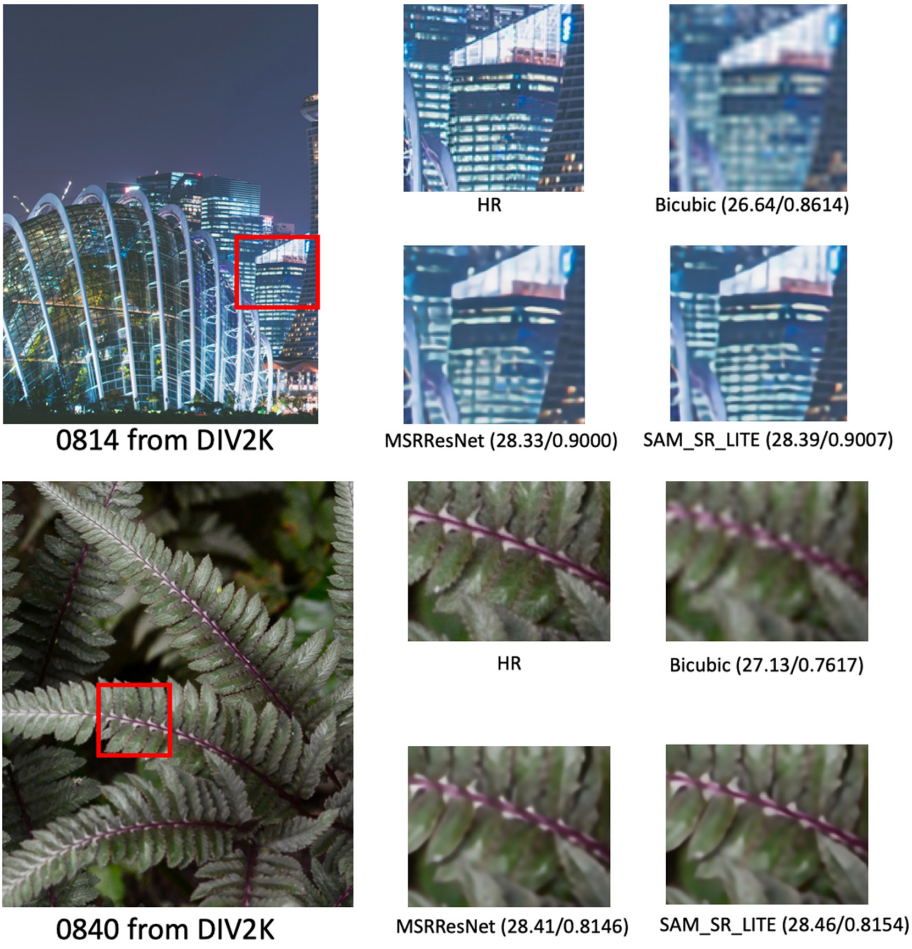


Fig. 4. Qualitative comparison. SAM_SR_LITE outperforms bicubic interpolation and is on-par with MSRResNet [45], while being significantly more efficient than the latter.

4.3 Comparison with Previous Methods

We compare our model with existing models that focus on both efficiency and image quality. We evaluate on Set5 [4], Set14 [43], BSD100 [29] and Urban100 [18], conventionally used as benchmarks for super-resolution, using the PSNR and SSIM [42] metrics computed on only the Y channel of the YCrCb color space. In order to align with test results in the literature, we retrained our models using only DIV2K (without observing any significant differences compared to our models trained on both DIV2K and FLICKR2K). Table 3 shows the results of the comparisons. The SAM_SR_LITE model achieves slightly higher PSNR-Y scores compared with previous methods with roughly similar, or smaller, operation

Table 2. Trade-off between generated image quality and efficiency for proposed model family. Note that the FLOPs and activations are calculated for the 256×256 input. The reported quantitative metrics are PSNR-Y and SSIM-Y, respectively, across different datasets.

nf	N	FLOPs	# Params	Runtime	Activation	Set5 [4]	Set14 [43]	BSD100 [29]	Urban100 [18]
72	16	58.6G	558.54K	0.169 s	576.45M	32.36/0.8968	28.71/0.7826	27.56/0.7363	26.18/0.7881
64	16	47.02G	444.9K	0.142 s	512.75M	32.24/0.8956	28.68/0.7817	27.56/0.7361	26.11/0.7864
64	8	27.84G	233.96K	0.093 s	344.98M	32.09/0.8938	28.54/0.7784	27.46/0.7325	25.86/0.7774
32	16	13.38G	119.28K	0.075 s	257.95M	32.00/0.8924	28.48/0.7768	27.42/0.7319	25.79/0.7758
32	8	8.08G	62.96K	0.051 s	174.06M	31.85/0.8903	28.39/0.7740	27.36/0.7290	25.54/0.7667
16	16	4.16G	33.85K	0.045 s	130.55M	31.75/0.8885	28.30/0.7721	27.30/0.7264	25.42/0.7625
16	8	2.58G	17.98K	0.029 s	88.60M	31.48/0.8843	28.12/0.7675	27.20/0.7226	25.18/0.7526

counts and parameter sizes. Specifically, consider all methods in Table 3 that have fewer than 1M parameters and require less than 100GFLOPs (i.e., the models in the first six rows, up to and including IMDN). We observe that our model has the highest PSNR (including one tie) among these seven methods, providing small PSNR increments of +0.15 dB (Set5), +0.13 dB (Set14), 0 dB (BSD100), and +0.14 dB (Urban100) over the previous best.

Moreover our second model, SAM_SR_XLITE, has less than 5% of the number of parameters and FLOPs as SAM_SR_LITE, and exhibits a loss of at most 1 dB in PSNR performance over the different test sets (specifically, the PSNR difference between the XLITE and LITE models are −0.88 dB (Set5), −0.59 dB (Set14), −0.36 dB (BSD100), and −1.00 dB (Urban100)). Recall that the difference between these two models is only in two of the hyperparameters, namely the number of blocks, N (16 versus 8), and the number of features per block, nf (72 versus 16). Indeed, we show in Fig. 3 that by selecting different values for these hyperparameters we obtain models that cover a wide range of trade-offs between PSNR performance and computational cost. For current mobile applications, where computational constraints are severe, we are interested in more lightweight models, such as SAM_SR_XLITE (Figs. 5 and 6).

4.4 Ablation Study

We conduct three sets of ablation experiments to investigate the impact of specific components in our architecture design. We first experiment with replacing the batch normalization (BN) and the squeeze and excite modules, each with the identity mapping. A third variation of our model removes the LR to HR bicubically interpolated skip connection.

We observe that batch normalization results in a significant performance improvement for our LITE model (i.e., +0.44 dB with BN), contrary to [28]. However, batch norm only brings tiny improvement to XLITE model (i.e., +0.02 dB). The skip connection also has a beneficial effect for both LITE (+0.27 dB) and XLITE model (+0.31 dB). This is likely because adding a bicubically interpolated image provides a good baseline for the low-frequency content of the

Table 3. Comparison between our model and efficient models in the literature. Note that the FLOPs are calculated for the 720P (1280 × 720) output. The reported quantitative metrics are PSNR-Y and SSIM-Y, respectively, across different datasets.

Model	FLOPs	# Params	Set5 [4]	Set14 [43]	BSD100 [29]	Urban100 [18]
FSRCNN [8]	9.2G	12K	30.71/0.8657	27.59/0.7535	26.98/0.7150	24.62/0.7280
FEQE-P [40]	11.0G	96K	31.53/0.8824	28.21/0.7714	27.32/0.7273	25.32/0.7583
ESRN-V [35]	41.4G	324K	31.99/0.8919	28.49/0.7779	27.50/0.7331	25.87/0.7782
LapSRN [25]	59.8G	813K	31.54/0.8850	28.19/0.7720	27.32/0.7280	25.21/0.7560
CARN-M [2]	65.0G	677K	31.92/0.8903	28.42/0.7762	27.44/0.7304	25.62/0.7694
IMDN [19]	88.9G	715K	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838
ESRN [35]	132.2G	1,014K	32.26/0.8957	28.63/0.7818	27.62/0.7378	26.24/0.7912
CARN [2]	181.8G	1,592K	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837
RCAN [46]	1839.86G	15.6M	32.63/0.9002	28.87/0.7889	27.77/0.7436	26.82/0.8087
ESRGAN [41]	2068.26G	16.7M	32.60/0.9002	28.88/0.7896	27.76/0.7432	26.73/0.8072
SAM_SR_XLITE	2.2G	18K	31.48/0.8843	28.12/0.7675	27.20/0.7226	25.18/0.7526
SAM_SR_LITE	51.5G	559K	32.36/0.8968	28.71/0.7826	27.56/0.7363	26.18/0.7881

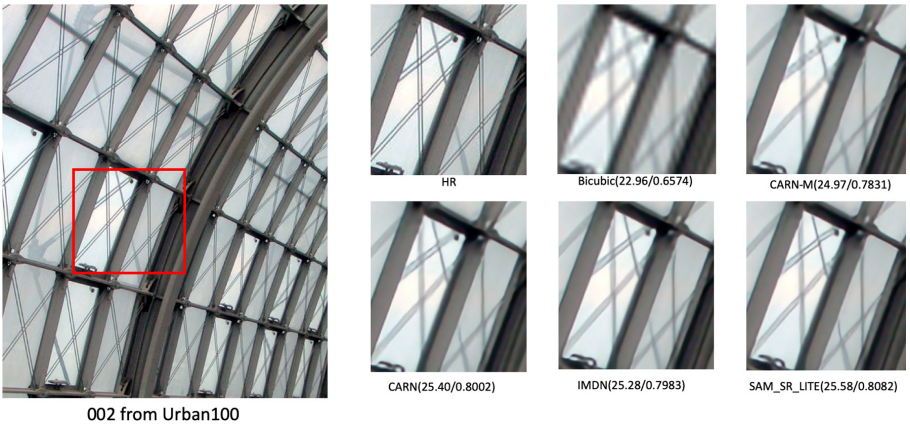


Fig. 5. Qualitative comparison on Urban100 [18]. SAM_SR_LITE outperforms other previous methods [2,19] with similar FLOPs and parameter size.

high-resolution image, thereby, allowing the model to focus on adding just the high-frequency details. The SE module is seen to provide a marginal improvement in image quality (+0.01 dB) for both LITE and XLITE model, which may be removed for more lightweight models. See Table 4 for more details.

Moreover, we fuse the batch norm layer to its preceding convolution layer to optimize the inference time without any performance change [26], see Table 4 under SAM_SR_LITE_FUSED. Although it does not improve FLOPs significantly, the fused model improves runtime by around 18%. Note that the fused model was explored after the challenge deadline and was not submitted to the challenge.

Fig. 6. Qualitative comparison on Set14 [43]. SAM_SR_XLITE outperforms bicubic intepolation, SRCNN [7] and FSRCNN [8] and on par with FEQE-P [40], while being $4-6\times$ smaller in operation size.

Table 4. Ablation study on SAM_SR_LITE and SAM_SR_XLITE. Runtime is measured in the same environment as Table 1. (*The SAM_SR_XLITE has a different validation performance here than the one in Table 2 as it was trained with a smaller input patch size of 64×64.)

Model	PSNR-RGB on DIV2K val dataset	FLOPs	Parameters	Runtime	Memory
SAM_SR_LITE	28.98	58.6G	558K	0.169 s	2.56 GB
w/o batch norm	28.55(−0.44)	57.0G(−1.6G)	546K(−12K)	0.138 s(−0.031 s)	2.50 GB(−0.06 GB)
w/o skip connection	28.71(−0.27)	58.6G(∼0.0G)	558K(0.0K)	0.166 s(−0.003 s)	2.56 GB(∼0.0 GB)
w/o SE	28.97(−0.01)	58.6G(∼0.0G)	393K(−165K)	0.155 s(−0.014 s)	2.56 GB(∼0.0 GB)
SAM_SR_LITE_FUSED	28.98(0.0)	57.0G(−1.6G)	546K(−12K)	0.139 s(−0.030 s)	2.50 GB(−0.06 GB)
SAM_SR_XLITE*	28.35	2.58G	17.98K	0.030 s	0.65 GB
w/o batch norm	28.33(−0.02)	2.4G(−0.18G)	16.54K(−1.44K)	0.027 s(−0.003 s)	0.65 GB(∼0.0 GB)
w/o skip connection	28.04(−0.31)	2.58G(∼0.0G)	17.98K(0.0K)	0.029 s(−0.001 s)	0.65 GB(∼0.0 GB)
w/o SE	28.34(−0.01)	2.56G(−0.02G)	13.88K(−4.1K)	0.028 s(−0.002 s)	0.65 GB(∼0.0 GB)

5 Inference Time

5.1 Disparity Between FLOPs and Runtime

Typically, a lower number of FLOPs implies a faster runtime, when measuring runtime efficiency. However, in our experiments we notice a palpable disparity between the expected runtime performance of our model and the one observed in practice. Results from AIM 2020 organizers indicate that in their environment, the SAM_SR_LITE is approximately two times slower than the baseline [44], while requiring only 18% of the FLOPs. We have two observations regarding this disparity.

Currently, optimized depth-wise convolution operations are not well supported by some deep learning frameworks, including PyTorch [30]. For example,

note the open issue in the PyTorch repository [14] showing that the inference runtime of a single depth-wise convolution layer `Conv2d(32,32,3,groups=32)` is *at best* only $\sim 3\times$ faster than a regular convolution layer `Conv2d(32,32,3)`, despite having a $32\times$ smaller operation count. A fair comparison would be to use depthwise separable convolutions; that is, depthwise convolutions `Conv2d(32,32,3, group=32)` followed by pointwise convolutions `Conv2d(32,32,1)`. This operation requires 14% FLOPs of a regular convolution `Conv2d(32,32,3)`, but takes 81% of the inference time for regular convolution (using PyTorch on a V100 GPU). Depthwise separable computations are a key design element of MobileNetV3 blocks, and subsequently, of our model as well. Therefore, any improvement in their efficiency would directly affect the runtime of our model.

In addition, we observed that our model does not fully take advantage of cuDNN [5]. Table 5 shows that cuDNN results in different speed-up factors for our model and for the MSRResNet baseline [45]. Much to our surprise, the challenge [44] organizers measured an even slower inference time using a *better* RTX 2080Ti GPU than our testing machine with a RTX 2080 GPU. In short, the inference time of our model varies from being 20% faster to being over twice slower than the baseline model as a function of software and hardware combination.

For these reasons, we argue that the number of operations (FLOPs) is a better measure for the computational efficiency of our model.

5.2 Activation Size as a Proxy for Runtime

The activation size is first analyzed as a proxy for performance in [31], with the authors reporting a strong correlation of the activation size of convolution layers with runtime. The AIM challenge organizers [44] also find that the activation size has a better ranking correlation with runtime compared to other complexity metrics (FLOPs, parameter size, memory).

We also investigate the correlation between runtime and activation size in our family of models by sampling models with various capacities. We find that for our family of models there is a strong linear correlation between runtime and **all** other complexity metrics (activation size, FLOPs, and parameter size); see Fig. 7. Thus, different from the finding in [31], for our models activation size is no more informative than the other metrics.

Evaluation of model complexity w.r.t. the activation size [44] puts our family of models at a disadvantage. A standard replacement of regular convolution with a depth separable convolution (a depthwise convolution followed by a pointwise convolution) will double the activation measurement, while blocks in MobileNetV3 would have an even larger activation size due to the internal expansion factor. As a result, despite the lower FLOP count, our models may be slower in runtime than models from different families with smaller activation size.

Finally, it is worth noting that, the strong correlation between activation size and runtime on memory-bound accelerators (desktop GPUs and TPUs) [31] may not translate to mobile phone accelerators, which have considerably

lower computational resources. Moreover, we believe that optimizing memory (e.g., pre-allocation of activations for a fixed input) for specific applications will reduce the impact of memory on runtime.

5.3 Runtime on Mobile Devices

Since the environment we are targetting for applying our models is the phone, we deploy a slightly modified SAM_SR_XLITE on a Samsung Note 10 to evaluate the runtime on an actual mobile device. We first perform the required conversion of the model from PyTorch to a Snapdragon Neural Processing Engine (SNPE) Deep Learning Container file. It takes our model 1.41 s to generate a 12MP image and 0.8 s for its 8-bit quantized version. Although the model cannot achieve real-time inference on the mobile devices, it is already deployable for many mobile applications such as zooming in mobile camera.

Table 5. Running time under different configurations for SAM_SR_LITE and baseline model. The running time is computed by averaging across 5 rounds per image (after 5 rounds warm-up on the same image).

GPU	CUDA	cuDNN	PyTorch	OS	Running Time-SAM_SR_LITE/ baseline	Note
RTX 2080	10.1	Disabled	1.5.1	Ubuntu 16.04	0.2137 s/−(OOM)	Our results
RTX 2080	10.1	7.6.3	1.5.1	Ubuntu 16.04	0.1668 s/0.1240 s	Our results
V100	10.1	Disabled	1.5.1	Ubuntu 16.04	0.1188 s/0.1488 s	Our results
V100	10.1	7.6.3	1.5.1	Ubuntu 16.04	0.0892 s/0.0680 s	Our results
RTX 2080Ti	10.2	7.6.5	1.5.1	Ubuntu	0.240 s/0.114 s	From challenge organizers

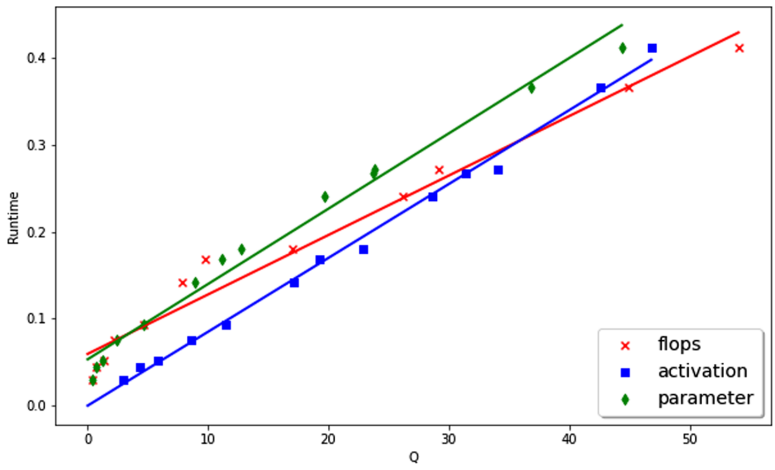


Fig. 7. Correlation between runtime and flops, activation and parameter in the proposed model family. The Q in the x-axis refer to $\frac{flops}{6G}$, $\frac{activation}{30M}$ and $\frac{parameter}{50M}$.

6 Conclusion and Future Work

Motivated by efficient super-resolution models for mobile applications, in this paper, we have proposed a set of efficient architectures that use adapted MobileNetV3 blocks. Our models achieve better trade-off between quality and efficiency than the current state-of-the-art in efficient super-resolution. The smallest of these, namely SAM_SR_XLITE, has an extremely small operation and parameter count, making it possible to be deployed on real mobile applications.

While being orders of magnitude more efficient in parameter size and operation count, the inference time of our models in PyTorch is, currently, less impressive. This is partly due to the limited optimization currently supported by the available software and hardware for deep learning and potentially large activation size. Considering the resource constrained devices as the final running environment of efficient SR models, the runtime on desktop GPUs may not fully reflect the runtime in actual applications. Still, optimizing the implementation of operators on both GPUs and edge devices, specifically depthwise convolutions, is a promising future direction for efficient super-resolution for real-time applications.

References

1. Agustsson, E., Timofte, R.: Ntire 2017 challenge on single image super-resolution: Dataset and study. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, July 2017
2. Ahn, N., Kang, B., Sohn, K.-A.: Fast, accurate, and lightweight super-resolution with cascading residual network. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11214, pp. 256–272. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01249-6_16
3. Baker, S., Kanade, T.: Hallucinating faces. In: Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580), pp. 83–88. IEEE (2000)
4. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding (2012)
5. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E.: cudnn: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759 (2014)
6. Chu, X., Zhang, B., Ma, H., Xu, R., Li, J., Li, Q.: Fast, accurate and lightweight super-resolution with neural architecture search. arXiv preprint arXiv:1901.07261 (2019)
7. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell. **38**(2), 295–307 (2015)
8. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 391–407. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_25
9. Elad, M., Feuer, A.: Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. IEEE Trans. Image Process. **6**(12), 1646–1658 (1997)

10. Farsiu, S., Elad, M., Milanfar, P.: Multiframe demosaicing and super-resolution of color images. IEEE Trans. Image Process. **15**(1), 141–159 (2005)
11. Farsiu, S., Robinson, M.D., Elad, M., Milanfar, P.: Fast and robust multiframe super resolution. IEEE Trans. Image Process. **13**(10), 1327–1344 (2004)
12. Freeman, W.T., Jones, T.R., Pasztor, E.C.: Example-based super-resolution. IEEE Comput. Graphics Appl. **22**(2), 56–65 (2002)
13. Freeman, W.T., Pasztor, E.C., Carmichael, O.T.: Learning low-level vision. Int. J. Comput. Vision **40**(1), 25–47 (2000)
14. Github: FP32 depthwise convolution is slow in GPU (2020). https://github.com/pytorch/pytorch/issues/18631. Accessed 16 July 2020
15. Gotoh, T., Okutomi, M.: Direct super-resolution and registration using raw CFA images. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004, vol. 2, p. II. IEEE (2004)
16. Howard, A., et al.: Searching for mobilenetv3. In: ICCV (2019)
17. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
18. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5197–5206 (2015)
19. Hui, Z., Gao, X., Yang, Y., Wang, X.: Lightweight image super-resolution with information multi-distillation network. In: Proceedings of the 27th ACM International Conference on Multimedia. pp. 2024–2032 (2019)
20. Irani, M., Peleg, S.: Improving resolution by image registration. CVGIP Graphical Models Image Process. **53**(3), 231–239 (1991)
21. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 694–711. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_43
22. Kim, J., Kwon Lee, J., Mu Lee, K.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1646–1654 (2016)
23. Kim, K., Kwon, Y.: Example-based learning for singleimage SR and jpeg artifact removal. MPI-TR, (173) 8 (2008)
24. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
25. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep laplacian pyramid networks for fast and accurate super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 624–632 (2017)
26. Learnml: Speeding up model with fusing batch normalization and convolution (2020). https://learnml.today/speeding-up-model-with-fusing-batch-normalization-and-convolution-3. Accessed 31 July 2020
27. Ledig, C., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4681–4690 (2017)
28. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: CVPRW (2017)
29. Martin, D., Fowlkes, C., Tal, D., Malik, J.: 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, vol. 2, pp. 416–423. IEEE (2001)

30. Paszke, A., et al.: Automatic differentiation in pytorch (2017)
31. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10428–10436 (2020)
32. Sajjadi, M.S., Scholkopf, B., Hirsch, M.: Enhancenet: single image super-resolution through automated texture synthesis. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4491–4500 (2017)
33. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
34. Shi, W., et al.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: CVPR (2016)
35. Song, D., Xu, C., Jia, X., Chen, Y., Xu, C., Wang, Y.: Efficient residual dense block search for image super-resolution. In: AAAI, pp. 12007–12014 (2020)
36. Tai, Y., Yang, J., Liu, X., Xu, C.: MemNet: a persistent memory network for image restoration. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4539–4547 (2017)
37. Takeda, H., Farsiu, S., Milanfar, P.: Robust kernel regression for restoration and reconstruction of images from sparse noisy data. In: 2006 International Conference on Image Processing, pp. 1257–1260. IEEE (2006)
38. Tong, T., Li, G., Liu, X., Gao, Q.: Image super-resolution using dense skip connections. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4799–4807 (2017)
39. Tsai, R.: Multiframe image restoration and registration. Adv. Comput. Vis. Image Process. **1**, 317–339 (1984)
40. Vu, T., Nguyen, C.V., Pham, T.X., Luu, T.M., Yoo, C.D.: Fast and efficient image quality enhancement via desubpixel convolutional neural networks. In: Leal-Taixé, L., Roth, S. (eds.) ECCV 2018. LNCS, vol. 11133, pp. 243–259. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11021-5_16
41. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Yu., Loy, C.C.: ESRGAN: enhanced super-resolution generative adversarial networks. In: Leal-Taixé, L., Roth, S. (eds.) ECCV 2018. LNCS, vol. 11133, pp. 63–79. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11021-5_5
42. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
43. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Boissonnat, J.-D., et al. (eds.) Curves and Surfaces 2010. LNCS, vol. 6920, pp. 711–730. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27413-8_47
44. Zhang, K., Danelljan, M., Li, Y., Timofte, R., et al.: AIM 2020 challenge on efficient super-resolution: methods and results. In: European Conference on Computer Vision Workshops (2020)
45. Zhang, K., et al.: Aim 2019 challenge on constrained super-resolution: methods and results. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 3565–3574. IEEE (2019)
46. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 294–310. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_18