Lightweight Modules for Efficient Deep Learning based Image Restoration

Avisek Lahiri[†], Sourav Bairagya[†], Sutanu Bera, Siddhant Haldar and Prabir Kumar Biswas, *Senior Member, IEEE*

***Abstract*—Low level image restoration is an integral component of modern artificial intelligence (AI) driven camera pipelines. Most of these frameworks are based on deep neural networks which present a massive computational overhead on resource constrained platform like a mobile phone. In this paper, we propose several lightweight low-level modules which can be used to create a computationally low cost variant of a given baseline model. Recent works for efficient neural networks design have mainly focused on classification. However, low-level image processing falls under the *'image-to-image'* translation genre which requires some additional computational modules not present in classification. This paper seeks to bridge this gap by designing generic efficient modules which can replace essential components used in contemporary deep learning based image restoration networks. We also present and analyse our results highlighting the drawbacks of applying depthwise separable convolutional kernel (a popular method for efficient classification network) for sub-pixel convolution based upsampling (a popular upsampling strategy for low-level vision applications). This shows that concepts from domain of classification cannot always be seamlessly integrated into *'image-to-image'* translation tasks. We extensively validate our findings on three popular tasks of image inpainting, denoising and super-resolution. Our results show that proposed networks consistently output visually similar reconstructions compared to full capacity baselines with significant reduction of parameters, memory footprint and execution speeds on contemporary mobile devices. Codes are made available at https://github.com/avisekiit/TCSVT-LightWeight-CNNs**

I. Introduction

IMAGE restoration refers to recovery of clean signal from an observed noisy input. Following the ground-breaking work of Krizhevsky *et al.* [36] on ImageNet classification with deep neural networks, CNNs have superseded traditional methods across a variety of tasks such as object recognition [26], [65], [66], detection [14], [15], [56] and tracking [4], [23], action recognition [7], [22], segmentation [24], [51] to list a few. Image restoration frameworks also improved from the data driven hierarchical feature learning capability of deep neural networks with state-of-the-art performances on inpainting [30], [37], [38], [40], [74], [76], denoising [69], [78], [81], super-resolution [39], [67], de-hazing [57], de-occlusion [85], 3D surface reconstruction [32], [63] etc. Though these deep learning based restoration frameworks yield photo-realistic outputs, the models are computationally expensive with millions of parameters. Inference through such complex networks requires billions of floating point operations (FLOPs). This might not be seen as a problem while executing over a GPU enabled workstation; however such networks are practically not scalable to run on resource-constrained platforms such as a commodity CPU or a mobile device. However, with the proliferation of multimedia enabled mobile devices, there is an increased demand of on-device multimedia manipulations. For example, image denoising is a crucial component of imaging setup in any contemporary smartphone. Super-resolution is also an inevitable component because online multimedia hosting sites often prefer to transmit low resolution images and videos with super-resolution performed on device so that the end user enjoys high resolution multimedia experience even on low bandwidth channel. Similarly, inpainting plays a crucial role in many downstream applications such as image editing, Augmented Reality [59], 'dis-occlusion' inpainting [42], [45] for novel view synthesis in a multi camera video capture setting [8] to be integrated with mobile Head Mounted Displays (HMD).

Executing billions of FLOPs on mobile devices leads to fast reduction of battery life with potential heating up of the device. Also, the lag encountered while executing such large models on constrained platform tends to disrupt the engagement of the user. To address the above two issues, in this paper we propose several lightweight computing units which dramatically reduce the computational cost of a given deep neural network without any visual degradation of reconstructed outputs.

Recently, there has been a surge of interest for designing efficient neural networks mainly for object classification and detection. However, there is a dearth of literature for efficient processing of networks concerned with low-level image restoration. Fundamentally, restoration requires the spatial resolution of input and output signal to be same and the general practice [30], [76], [81] is to follow encoder-decoder based architectures to first down-sample and later on up-sample the intermediate feature maps of the network. On contrary, classification frameworks are mainly concerned with progressive downsampling and thus efficient strategies to up-sample in a network are not discussed. Also, dense prediction tasks such as inpainting requires long range spatial information and often deploys dilated/atrous convolutions [75] to increase the receptive field of processing. However dilated convolutions are rarely used in classification frameworks and thus recent advancements such separable convolution [29] and

[†] First two authors share equal contribution.

A. Lahiri, S. Bera, S. Haldar and P.K. Biswas are affiliated to Indian Institute of Technology Kharagpur. S. Bairagya is currently affiliated to Mathworks, India. The work was done during his tenure as a M.Tech student at IIT Kharagpur. All correspondences to avisek@ece.iitkgp.ac.in

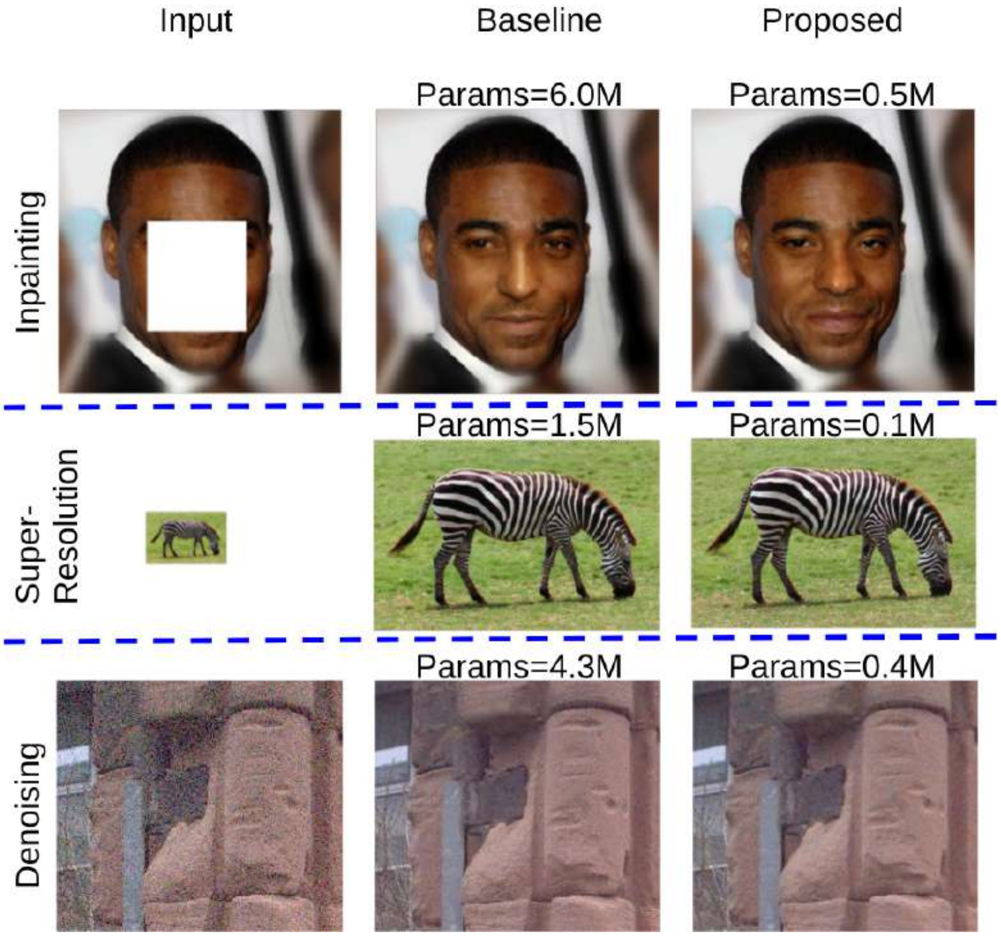


Fig. 1: Visual comparison of outputs from our computationally efficient variants on three common image restoration applications. The full-scale baselines are GLCIC [30] for inpainting, SRGAN [39] for super-resolution and CBDNet [17] for denoising. Number on top of a figure denotes the total number of parameters (in millions) of a particular model. Best viewed zoomed in.

group convolutions [82] cannot be directly applied for dilated convolution operations.

In this paper we have mainly focused on design principles for components to be used in low-level restoration tasks. Since 3×3 kernel [1] is the most commonly used kernel in contemporary low level vision applications [30], [39], [76], [81], we introduce *‘**LI**ght **S**patial **T**ransition’* layer, (*LIST*), which simultaneously benefits from local feature aggregation [41] and multi-scale spatial processing [65] and uses upto 24× fewer parameters than a similar 3 ×3 convolution layer. Next, we introduce *‘**G**rouped **S**huffled **A**trous **T**ransition’* layer, *GSAT*, which is an efficient atrous/dilated convolution layer by leveraging recent concepts of group convolution [36] and channel shuffling [82] and each layer uses approximately 7× fewer parameters compared to an usual dilated convolution layer. While designing efficient upsampling module, we show that separable convolution kernels are inept at sub-pixel convolution [60] based upsampling and we provide an analytical justification for the same. Instead we show that deterministic upsampling such as bilinear upsampling followed by our *LIST* module provides an efficient upsampling framework. Combination of these modules enable us to run image restoration models on mobiles with milli-seconds level execution speed compared to several seconds by contemporary full-scale models without any visual degradation of outputs. One of the major advantages of our proposed modules is that these can seamlessly replace commonly used computational blocks such as 3×3 convolution, dilated convolution, differentiable upsampling within a given network. Thus, in this paper we refrain from proposing new end-to-end architectures; instead we select recent state-of-the-art networks and reduce the computational footprints of those networks using our lightweight layers. In summary, our key technical contributions in this paper are:

-We present *LIST* layer as a computationally cheaper alternative to a regular 3×3 convolution layer. Each instance of *LIST* can save 12× - 24× parameters. Repeated use of *LIST* in a deep network leads to significant reduction of parameters and FLOPs

-We present *GSAT* layer which implements dilated convolution on separate sparse group of channels to reduce FLOPs followed by feature mixing for enhanced representation capability. Each instance of the proposed module utilizes approximately 7× fewer parameters than a regular dilated convolution layer

[1] Ideally, it should be $3{\times}3{\times}C_{in}$- where C_{in} is number of input channels. For brevity of notation, henceforth, we will drop the channel dimension.

- We present our findings on drawbacks of applying separable convolution for feature upsampling with sub-pixel convolution and provide a detailed insight for possible reason of failure. Instead, we show that deterministic upsampling followed by *LIST* layer based convolution is an efficient yet accurate alternative
- We perform extensive study of our network components on tasks of image inpainting, denoising and super- resolution. On all tasks we achieve significant reduction in parameters and FLOPs and massive execution speed-ups on resource constrained platforms without any compromise in visual quality. Such exhaustive experiments manifest the generalizability of our processing components across a variety low-level restoration tasks.

II. Related Works

In recent years deep neural networks have achieved overwhelming success on a variety of computer vision tasks in which network design plays a crucial role. Executing these large models on resource constrained platforms requires efficient design strategies [25]. Recently there has been a surge of interest in either compressing existing pre-trained big networks or designing small networks from scratch.

A. Kernel Factorization

For training small networks from scratch, factorization of kernels have been a preferred choice. The most common realization is depthwise separable convolution initially presented in [62] and then popularized in Inception module [65]. Following that, it has become the backbone of many popular architectures such as MobileNet [29] and MobileNet-V2 [28]. Xception network [10] showed how to scale up depthwise separable convolutions to outperform Inception-V3 [66]. Another popular concept of group convolution was introduced in [36] to distribute model parameters over multiple GPUs. Currently, it is utilized in several recent efficient networks [48], [64], [72], [83]. The idea is to convert dense convolutions across all feature channels to be sparse by channel grouping and performing convolution only on grouped set of channels.

B. Model Compression

Model compression is another genre of approach for efficient inferencing by lossy compression of a pre-trained network while maintaining similar accuracy. Compression can be achieved either by pruning some of the intermediate synaptic connections in the network or by quantizing pre-trained kernels to be represented as integers or booleans. Denton *et al.* [12] applied Singular Value Decomposition (SVD) to approximate a pre-trained network to achieve 2× inference speedup. Han *et al.* [21] pruned and fine-tuned a pre-trained network to identity important network connections to create a smaller network. The work was extended in *Deep Compression* [20] to combine network pruning with quantization. Later, *'Quantized CNN'* [71] was proposed which aimed at directly quantizing network weights during training. Chen *et al.* proposed *'HashedNet'* [9] to compress networks with hashing.

C. Task Specific Efficient Architectures

Some recent works have focused on smarter network designs for efficient low-level vision applications. Zhang and Tao [80] proposed a light-weight multi-scale network for single image dehazing. In [1] Ahm *et al.* proposed a cascaded residual network coupled with group convolution for efficient single image super-resolution. In [68], Tan *et al.* presented a low-cost network for unmanned aerial vehicle (UAV) noise reduction at low signal-to-noise (SNR) level. In [84], Zhang *et al.* present a *'mixed-convolution'* layer by merging normal and dilated convolution for image super-resolution. Kim *et al.* [34] presented dilated-Winograd transformation for a faster realization of dilated convolution. In RHNet [77] the authors present a dilated special pyramid pooling framework for dense object counting.

In this paper we mainly focus on constructing lightweight modules for training efficient networks from scratch for low-level image restoration tasks. However, the building blocks of modern efficient networks are mainly concerned with classification tasks in which essential components such as upsampling, sub-pixel convolution and dilated convolution are usually not involved. Hence, those methods are not self-sufficient for low-level computer vision applications.

In recent years deep learning based methods have produced phenomenal performances on a variety of low-level image restoration tasks. However majority of research has been focused on improving the visual quality without worrying much about the computational burden. In this paper we aim to realize lightweight versions of these networks which can be run on mobile devices with milli-seconds level execution time instead of multiple seconds required by full-scale baselines.

III. Proposed Network Modules

A. *'LIght Spatial Transition' layer: (LIST)*

This section elaborates on the architectural details of *LIST* layer. Pictorial representation of a *LIST* layer is shown in Fig. 2b. We will first discuss the driving intuitions and principles behind *LIST* followed by calculating computation savings achieved by using *LIST* instead of regular 3×3 convolution layer. Presence of a 'sub-network' capable of universal functional approximation such as multi-layer perceptron (MLP) in between two consecutive layers boosts the feature extraction capability in a CNN [41]. In *LIST*, we realize this functionality by having one parallel branch of two successive layers of 1×1 convolution with ReLu non-linearity in between to promote sparsity of features. Such cascades of 1×1 convolution promotes parametric cross-channel pooling and enables a network to learn non-trivial transformations.

Starting from the Inception [65] module of GoogleNet (see Fig. 2a), multi-path branched module has become the de facto choice for multi-scale processing of features in deep neural networks [66]. Following that, we incorporate a branch for 3×3 convolution in parallel with the 1×1 branch. In this case, the initial (top) 1×1 layer acts an embedding layer by projecting incoming feature volume to a lower dimension and thereby reducing the FLOPs requirement for performing 3×3 convolution. We further reduce the FLOPs count for 3×3 convolution branch by factoring it with depthwise separable kernels. However, we deviate from the design principles of Inception by restricting the number of parallel branches inside the *LIST* layer. This is motivated by the 'network fragmentation' issue pointed out in [48]. Parallel branches in a network creates overhead of kernel launching and synchronization resulting in reduction of execution speed. So, unlike that in Inception, we refrain from using two additional parallel branches of 5×5 convolution and max-pool layer inside our *LIST* layer. Apart from 'network fragmentation' issue, avoiding parallel branches also benefits from reduced number of final feature channels which need to be processed by next layer- this further helps in decreasing FLOPs.

1) Architecture Details: A *LIST*$^{M\rightarrow N}$ layer is meant for replacing a normal 3×3$^{M\rightarrow N}$ convolution layer with M input and N output feature channels. Input to a *LIST*$^{M\rightarrow N}$ module is a feature volume of shape (H, W, M) (height, width, channels). In the first step, the input volume is pointwise convolved with $\frac{M}{k}$ number of 1×1 kernels; k is the compression ratio. In the second stage, these $\frac{M}{k}$ feature maps are passed to two parallel streams of 1×1 and 3×3 convolution. In the 1×1 branch, we perform another set of pointwise 1×1 convolution and output $\frac{N}{n_b}$ channels; n_b is the branching factor. The 3×3 branch is realized with depthwise separable kernels and outputs $N - \frac{N}{n_b}$ channels. Outputs from 1×1 and 3×3 streams are concatenated (to form total N channels) and passed on to the next layer.

2) Computational Savings: [2]

Comparison to 3×3 convolution: In this section we elaborate on the savings of parameters and FLOPs achieved by our *LIST*$^{M\rightarrow N}$ layer over the usual 3×3$^{M\rightarrow N}$ layer. We assume the spatial resolution of incoming and outgoing features to be H×W. Number of trainable parameters for a 3×3$^{M\rightarrow N}$ is,

$$P_{3\times3} = 9MN, \tag{1}$$

while the total FLOPs is,

$$F_{3\times3} = 9MNHW \ . \tag{2}$$

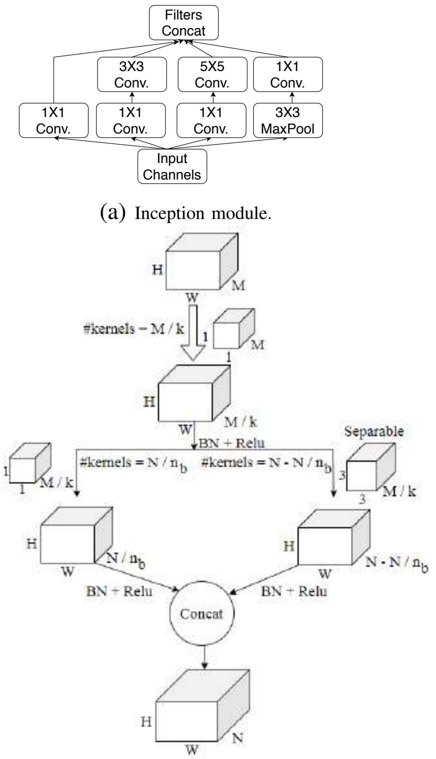
Computations for a *LIST*$^{M\rightarrow N}$ module will consist of three components- (a) 1×1 convolution in Stage-1; (b) 1×1 convolution in Stage-2 parallel stream; (c) separable 3×3 convolution in Stage-2 parallel stream. Assuming n_b = 2 (see Sec. IV-A1) number of parameters for a LIST$^{M\rightarrow N}$ layer is,

$$P_{LIST} = \underbrace{\frac{M^2}{k}}_{\text{1X1 Stage-1}} + \underbrace{\frac{MN}{2k}}_{\text{1X1 Stage-2}} + \underbrace{3^2 \times \frac{M}{k} + \frac{MN}{2k}}_{\text{3X3 Stage-2}} \ , \tag{3}$$

while total FLOPs is,

$$F_{LIST} = \underbrace{\frac{HWM^2}{k}}_{\text{1X1 Stage-1}} + \underbrace{\frac{HWMN}{2k}}_{\text{1X1 Stage-2}}$$

$$\underbrace{+ \, 3^2 \times \frac{HWM}{k} + \frac{HWMN}{2k}}_{\text{3X3 Stage-2}} . \tag{4}$$



(a) Inception module.

(b) LIST module

Fig. 2: (a) An usual Inception module [65]. (b) Proposed *LIST*$^{M\rightarrow N}$ module as a replacement of normal 3×3 kernel operating on M incoming channels and yielding N output channels. **BN:** Batch-normalization [31] layer; **Relu(x):** $max(0, x)$.

Ratio of parameters of 3×3$^{M\rightarrow N}$ to that of *LIST*$^{M\rightarrow N}$ is given by,

$$R^{3\times3|LIST}_{params} = \frac{9Nk}{M+N+9}, \tag{5}$$

$$\approx \frac{9Nk}{M+N}. \tag{6}$$

Since, k, is the compression ratio of incoming and outgoing channels to the first 1×1 layer, $k > 1$. Thus, we have,

$$R^{3\times3|LIST}_{params} > \frac{9N}{M+N}. \tag{7}$$

From Eq. 7 we get the lower bound of parameters saving by using proposed *LIST* layer instead of a 3×3 convolution layer. Some of the usual settings in a network are $M = N$, $M = 2N$ or $N = 2M$. After a brief hyper-parameters search (see Sec. IV-A1) we set k = 4 and thus we achieve 18×, 12× and 24× parameters saving at $M = N$, $M = 2N$ and $N = 2M$. Thus a single instance of our *LIST* layer is significantly cheaper than a normal 3×3 convolution layer. On a similar note, we can show that the ratio of FLOPS of 3×3$^{M\rightarrow N}$ to that of LIST$^{M\rightarrow N}$ is given by,

$$R^{3\times3|LIST}_{Flops} = \frac{9Nk}{M+N+9}. \tag{8}$$

Since the ratio is same as what we got for parameters savings, following the approximation done in Eq. 6 and lower bound logic of Eq. 7, we get similar scales of FLOPs savings as we showed for the parameters. Stacking several layers of *LIST* layer thereby helps in significant reduction of memory footprint (fewer parameters) and faster execution speed (fewer

[2] All throughout the paper, we consider 'valid' convolution by padding zeros at the border and stride of 1 pixel; this preserves the image resolution .

FLOPs) compared to a network realized with 3×3 convolution layers.

Comparison to depthwise separable 3×3 convolution: In this section we first find the condition under which proposed *LIST* layer is even cheaper than the widely used depthwise separable convolution layer. We again assume 3×3 convolution over a feature volume of M incoming and N outgoing channels and spatial resolution of H×W. Number of trainable parameters for a separable 3×3 convolution layer is,

$$P_{3\times3|sep} = 9M + MN, \tag{9}$$

while total FLOPS is,

$$F_{3\times3|sep} = 9HWM + HWMN \tag{10}$$

Ratio of parameters for a separable 3×3 convolution layer to that of *LIST* is,

$$R^{sep-3\times3|LIST}_{params} = \frac{k(N+9)}{M+N+9} \approx \frac{Nk}{M+N} \tag{11}$$

If we want $R^{sep-3\times3|LIST}_{params} > 1$ then we need to satisfy the following condition:

$$R^{sep-3\times3|LIST}_{params} > 1 \implies k > \frac{M}{N} + 1. \tag{12}$$

So, we have the following criteria for k at different ratios of $\frac{M}{N}$:

$$k > \begin{cases} 2 & if \;\; M = N. \\ 3 & if \;\; M = 2N. \\ 1.5 & if \;\; N = 2M. \end{cases} \tag{13}$$

To satisfy all the conditions of Eq. 13 we need $k > 3$ which gives the lower bound of parameters savings. Since we set $k = 4$ for all our experiments, the conditions of Eq. 13 are satisfied. With $k = 4$, from Eq. 11 we have $R^{sep-3\times3|LIST}_{params}$ = 2, 2.6 and 1.3 at M=N, N=2M and M=2N respectively. Similarly we can show that ratio of FLOPS of a depthwise separable 3×3 layer to that of *LIST* is,

$$R^{sep-3\times3|LIST}_{Flops} \approx \frac{Nk}{M+N} \tag{14}$$

With $M = N$, $N = 2M$ or $M = 2N$ we would approximately save 2×, 2.6× and 1.3× FLOPs respectively. Our *LIST* layer's design has appreciably fewer parameters and FLOPs compared to even a depthwise separable realization of 3×3 convolution and thus can be used as an off-the-self replacement for separable convolution layer.

B. 'Grouped Shuffled Atrous Transition' layer, (GSAT)

In this section we elaborate on the design of our proposed *GSAT* layer which is an efficient replacement for an usual atrous/dilated convolution layer found in numerous contemporary low-level vision applications [19], [30], [70]. Realizing a 3×3 dilated convolution is not trivially possible by our *LIST* module because of the 1×1 convolution in the first stage. For this we propose *GSAT* layer. We mainly consider a 3×3 dilated convolution with same number of incoming and outgoing channels. This is the most popular configuration in contemporary architectures. Illustration of a *GSAT* layer is shown in Fig. 3b.

Input to the layer is a feature volume of shape H×W×M. Based on group convolution [36], we divide the incoming M channels into g non-overlapping groups. Then each of the groups is individually processed by an usual dilated 3×3 convolution. The initial group partitioning helps in reduction of incoming channels to individual 3×3 dilated convolution layers and thereby saves on parameters and FLOPs. However, each of the g groups are processed independently on a subgroup of channels without any cross-group interaction. This property weakens the representation capability of the model. Thus for cross channel interaction we perform a channel shuffling operation [82] to periodically sample and stack features from each of g groups. This results in an intermediate volume of shape H×W×M. So features from a particular group are stacked every alternate $\frac{M}{g}$ channels apart. Thus a group of $\frac{M}{g}$ channels inside the intermediate volume has features from each of the g groups. Next, to perform a cross channel interaction [41] of features we include a 1×1 convolution layer. However to reduce FLOPS, we perform grouped 1×1 convolution partitioned over g groups. Since the channel shuffling operation already populated each of the sub-groups with features from all the 3×3 dilated convolution layers, the grouped 1×1 layer can now learn a non-linear transformation conditioned on all the dilated convolution layers. Thus we avoid any further channel shuffling operation. Lastly, inspired by residual connection [26], we add the input with the 1×1 group convolution's output. To our best knowledge, this is the first realization of dilated convolution layer with grouped convolution and channel shuffling.

1) Computational Savings: In this section we numerically illustrate the computational benefits of using our *GSAT* layer instead of usual dilated convolution layer. Number of trainable parameters for a normal 3×3 dilated convolution layer is given by,

$$P_{3\times3|dil} = 3^2 \times M^2, \tag{15}$$

where M is the number of incoming and outgoing channels. For *GSAT* layer, number of parameters for the first stage of grouped convolution is $\frac{3^2M^2}{g^2} \times g = \frac{3^2M^2}{g}$ while for the second stage of 1×1 grouped convolution is $\frac{M^2}{g^2} \times g = \frac{M^2}{g}$. So, total parameters for *GSAT* layer is,

$$P_{GSAT} = \frac{10 \times M^2}{g} \;. \tag{16}$$

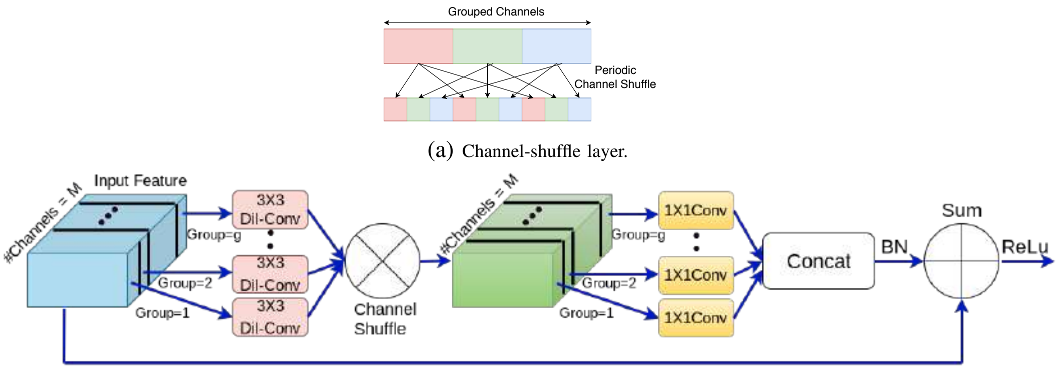
Ratio of parameters used in regular dilated convolution and that used by proposed *GSAT* layer is,

$$R^{3\times3|GSAT}_{params} = \frac{9g}{10} \;. \tag{17}$$

So, we can save parameters if $R^{3\times3|GSAT}_{params} > 1$, which requires $g \geq 2$. In fact, after hyper-parameters search (see Sec. IV-A1) we used $g = 8$ and thus *GSAT* module requires almost 7× fewer parameters compared to normal dilated convolution layer.

C. Efficient Upsampling Strategies

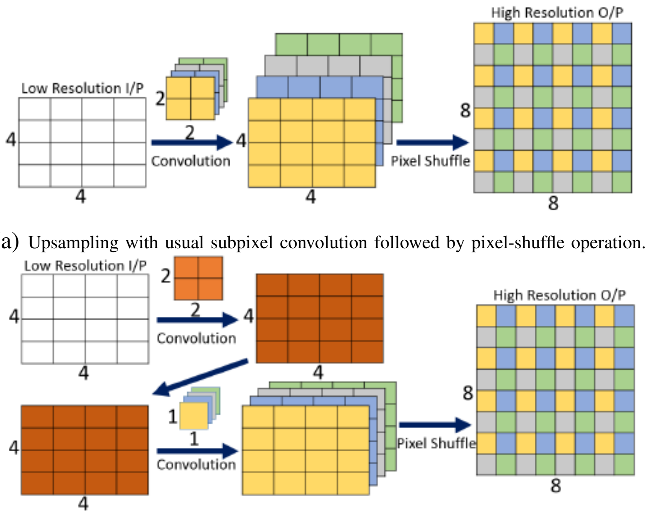
Upsampling of intermediate feature maps in a network is an essential component for low-level vision tasks. However, recent frameworks for efficient network design do not discuss upsampling strategies because it is rarely required in classification frameworks. We thus devote this section for discussing possible solutions for efficient upsampling. In recent literature transposed convolution (popular as deconvolution) [52] has



(a) Channel-shuffle layer.

(b) GSAT module

Fig. 3: (a) Channel-shuffle layer as presented in [82] (b) Proposed *GSAT* layer for lightweight realization of dilated convolution. Channel Shuffle enables periodic mixing of features coming from each of the preceeding dilated convolution layers. Concat block concatenates feature volumes (along channel dimension) output from the 1×1 convolution layers. **Dil-Conv:** Dilated convolution; **BN:** Batch-normalization.



(a) Upsampling with usual subpixel convolution followed by pixel-shuffle operation.

(b) Upsampling with separable sub-pixel convolution followed by pixel-shuffle.

Fig. 4: Approaches for upsampling with sub-pixel convolution and pixel-shuffle layer. Colored activation grids indicate the corresponding kernel responsible for activating that grid point on feature map.

become the de facto choice for upsampling. However, from an image generation perspective, transposed convolution is known to render ‘checkboard’ effects [2], [53] on the final synthesized image. Thus, even though there are efforts towards making transposed convolution computationally faster [10], [66] we explore other avenues for efficient upsampling.

1) Failure of Separable Kernels for Sub-Pixel Convolution: Sub-pixel convolution based upsampling is a preferred paradigm of upsampling specifically for image generation tasks because of its demonstrated ability to get rid of ‘checkboard’ artifacts introduced by transposed convolution layer. In this section we elaborate on our initial failed attempt of applying (see Fig. 5 for failed inpainting results) separable kernels for sub-pixel convolution based upsampling and provide justifications for the same.

It can be shown that, for an upscale factor of, r, a sub-pixel convolution with kernel shape $(k, k, o \times r^2, i)$ (height, width, # of output channels, # of input channels) is equivalent to a that of a transposed convolution by a kernel of shape (k, k, o, i). After sub-pixel convolution, the $o \times r^2$ channel elements are periodically shuffled to upscale feature maps by factor of r along height and width. See Fig. 4a for visualization. Refer to [60], [61] for more detailed derivation.

From the theory of sub-pixel convolution we know that with an upscale factor of 2, sub-pixel convolution can learn to represent n feature maps in LR (low resolution space) which are equivalent to $\frac{n}{4}$ feature maps in HR (high resolution space). We will show that this essentially means both networks have same run time complexity but sub-pixel convolution has more parameters. Let us consider a general case where shape of input volume at layer $l-1$ is (height, width, depth) = $(\frac{H}{2}, \frac{W}{2}, c_{l-1})$. The target is to upscale this to spatial resolution of $H \times W$, for next layer, l. Let, for sub-pixel convolution we choose kernels of shape (k, k, c_l, c_{l-1}). Then for counterpart of HR model (which first does deterministic up-scaling followed by convolution in HR space itself), kernel sizes will be $(k, k, \frac{c_l}{4}, c_{l-1})$. Total FLOPs for sub-pixel convolution is,

$$F_{LR} = k^2 \times \frac{H}{2} \times \frac{W}{2} \times c_{l-1} \times c_l \ . \tag{18}$$

The number of trainable parameters for LR model is,

$$|\theta|_{LR} = k^2 \times c_{l-1} \times c_l \ . \tag{19}$$

For convolution in HR, total FLOPs,

$$F_{HR} = k^2 \times H \times W \times c_{l-1} \times \frac{c_l}{4} \ , \tag{20}$$

and number of parameters,

$$|\theta|_{HR} = k^2 \times c_{l-1} \times \frac{c_l}{4} \ . \tag{21}$$

So, important observation is that FLOPs for both LR and HR models are same but LR model has more parameters and thus greater representation capability.

Let us now examine what will happen when we try to realize separable sub-pixel convolution. See Fig. 4b for a visualization. In the first stage, we need kernels of shape $(k, k, c_{l-1}, 1)$ (height, width, output channels, input channels). In this stage, total FLOPs,

$$F_{LR|sep_1} = k^2 \times \frac{H}{2} \times \frac{W}{2} \times c_{l-1} \ , \tag{22}$$

and number of parameters,

$$|\theta|_{LR|sep_1} = k^2 \times c_{l-1} \ . \tag{23}$$

In the next stage we need kernels of shape $(1, 1, c_l, c_{l-1})$. Total FLOPs in this stage,

$$F_{LR|sep_2} = c_{l-1} \times \frac{H}{2} \times \frac{W}{2} \times c_l \ , \tag{24}$$

and number of trainable parameters,

$$|\theta|_{LR|sep_2} = c_{l-1} \times c_l \ . \tag{25}$$

So, total FLOPs for separable LR model, $F_{LR|sep} = F_{LR|sep_1} + F_{LR|sep_2}$ and total number of parameters, $|\theta|_{LR|sep} = |\theta|_{LR|sep_1} + |\theta|_{LR|sep_2}$. Now consider the ratio,

$$\frac{|\theta|_{LR|sep}}{|\theta|_{HR}} = 4\left[\frac{1}{k^2} + \frac{1}{c_l}\right] \tag{26}$$

$\frac{|\theta|_{LR|sep}}{|\theta|_{HR}} < 1$ always and thus we see that converting a sub-pixel convolution to a separable paradigm reduces its representation prowess with respect to a convolution in HR. Similarly, if we compare the FLOPs by $\frac{F_{LR|sep}}{F_{HR}} = 4\left[\frac{1}{k^2} + \frac{1}{c_l}\right]$, separable sub-pixel convolution is computationally cheaper. But because of its reduced representation capability, it is not recommended for practical applications.

2) Deterministic Upsampling + Convolution: One way to mitigate 'checkboard' effect is to disentangle upsampling and convolution operations [53]. An usual procedure is to use some deterministic upscaling followed by convolution in the high resolution space. This has worked well in applications such as super resolution [13] and inpainting [76]. But, when implemented in naive version, this increases the computational cost. For example, if we do a bilinear upscaling by 4× followed by convolution, there is a quadratic increase of feature size but 'same information content' (if we count the number of floats). This makes bilinear upsampling + convolution almost 4× costlier than transposed convolution. We optimize this concept by first upsampling with bilinear interpolation followed by an efficient convolution block realized by the proposed *LIST* layer. This is our preferred method for efficient upsampling.

D. Downsampling in Network

To maintain the homogeneity in network design we prefer to realize spatial downsampling with *LIST* layer. However, strided convolution is not trivially possible by *LIST* module because of initial 1×1 convolution stream. So, we first downsample feature maps with bilinear interpolation and follow up with *LIST* based efficient convolution.

IV. Experiments and Results

We organize our results as follows. In Sec. IV-A, we initially perform extensive studies to select the hyper parameters governing the design choices for various proposed modules based on image inpainting. We systematically investigate the role of individual components towards reduction of parameters and FLOPs. This is followed by comparison with recent full capacity inpainting baselines and compressed models realized with MobileNet [29], ShuffleNet [82] and ShuffleNetV2 [48].

Next, with our understanding of best network configurations we compare applicability of our proposed layers on image denoising (Sec. IV-B) and image super-resolution (Sec. IV-C). It is encouraging to note that the proposed layers are quite insensitive to hyper parameters across different tasks which allows us to reuse the same set of hyper parameters across all the three above mentioned applications without degradation of visual quality.

A. Image Inpainting

We select the globally and locally consistent image inpainting model, GLCIC [30] as our baseline for image inpainting. Currently, GLCIC serves as a strong Generative Adversarial Network (GAN) [16] based contemporary baseline for inpainting and we aim at realizing a lightweight version of GLCIC using our proposed layers. A GAN framework consists of two deep neural nets, generator, G_{θ_G}, and discriminator, D_{θ_D}. The task of the generator is to generate an image, $x \in \mathcal{R}^{H\times W\times 3}$ with a latent noise prior vector, $z \in \mathcal{R}^d$, as input. z is sampled from a known distribution, $p_z(z)$. A common choice [16] is, $z \sim \mathcal{U}[-1, 1]^d$. The discriminator has to distinguish real samples (sampled from p_{data}) from generated samples. Discriminator and generator play the following two-player min-max game on $V(D_{\theta_D}, G_{\theta_G})$:

$$\underset{G_{\theta_G}}{min} \ \underset{D_{\theta_D}}{max} \ V(D_{\theta_D}, G_{\theta_G}) = \mathbb{E}_{x\sim p_{data}(x)}[\log D_{\theta_D}(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - D_{\theta_D}(G_{\theta_G}(z))]. \tag{27}$$

At the core, GLCIC comprises of repeated applications of 3×3 convolution, 3×3 dilated convolution and transposed convolution layers. Please refer to [30] for details of the architecture. We replaced the corresponding layers with proposed *LIST*, *GSAT* and *LIST* based upsampling layers.

Automated Visual Quality Metric: Manually analyzing the perceptual quality of reconstruction by different models is not feasible. Recent works [39], [76] have shown that PSNR and MS-SSIM metrics are not suitable for evaluating quality of adversarial loss guided reconstructions. Analyzing the quality and diversity of GAN samples is still an open research topic. Recently Fréchet Inception Distance (FID) [27] was proposed for quantifying quality and diversity of GAN samples. Lower FID value indicates overall better quality and diversity of generated samples. For automated screening of models, we use FID as the base metric.

Datasets: We experimented on CelebA (128×128) [43], CelebA-HQ (256×256) [33], Places2 (256×256) [87] and DTD (256×256) [11]. For CelebA, hole sizes greater than 48×48 occludes almost entire face and thus maximum training hole size is 48×48 at random location. For comparing FID during evaluation, a randomly positioned hole (but same for all models for a given image) of 48×48 is considered. At 256×256 image resolution, the maximum hole size of 96×96 is considered during training and FID is reported at hole size of 96×96. From CelebA, CelebA-HQ, Places2, and DTD we kept 20000, 10000, 20000, and 1000 (converted to 4000 with horizontal and vertical flip) samples for testing.

Training Details: In practice, we follow the stagewise training procedure as presented in [30]. In Stage-1, we pre-train the inpainting (generator) network alone with MSE (Mean Squared Error) loss for T_1 iterations. In Stage-2, we freeze the parameters of inpainting network and pre-train the

TABLE I: FID scores on CelebA validation set by networks controlled by different settings of bottleneck ratio, k and branching factor, n_b of proposed *LIST* module.

				$\frac{1}{k}$ = 0.25				
$\frac{1}{n_b}$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
FID	6.93	6.95	6.98	7.10	7.11	8.92	10.23	14.25
				$\frac{1}{n_b}$ = 0.5				
$\frac{1}{k}$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
FID	8.11	7.31	6.92	6.85	6.83	6.80	6.78	6.76

TABLE II: FID scores on CelebA validation set with different variants of our models controlled by number of groups (g) in the proposed *GSAT* module.

$g=1$	$g=2$	$g=4$	$g=8$	$g=16$	$g=32$
6.72	6.75	6.80	7.09	10.23	20.31

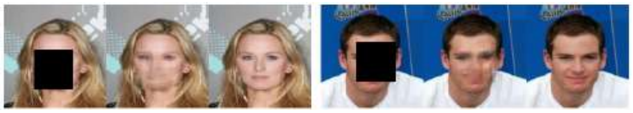


Fig. 5: Failure of separable sub-pixel convolution based upsampling (proposed model, M_2). For each triad, Left: Masked Image, Middle: Output with model M_2 guided decoder. Right: Output with decoder having regular sub-pixel convolution (proposed model M_1) based upsampling. Best viewed zoomed in.

TABLE III: Different variants of proposed light-weight inpainting models. Variations of models are achieved by different strategies to realize 3×3 convolution layers, dilated/atrous convolution layers and upsampling in decoder sections. *LIST*: 3×3 convolution realized with proposed *LIST* layer; *GSAT*: Proposed Grouped-Shuffled convolution based dilated convolution instead of normal dilated convolution; DS: Depthwise separable 3×3 convolution; BiL: Bilinear upsampling.

Model	3X3	Upsampling	Dilation	Params (10^6)	FLOPs (10^9)
M_1	DS	Pixel Shuffle (Normal Conv.)	Normal	3.42	33.1
M_2	DS	Pixel Shuffle (Separable Conv.)	Normal	2.93	27.1
M_3	DS	BiL + DS	Normal	2.81	26.9
M_4	*LIST*	BiL + DS	Normal	2.63	24.8
M_5	*LIST*	BiL + *LIST*	Normal	2.61	24.0
M_6	*LIST*	BiL + *LIST*	*GSAT*	**0.54**	**7.4**

critic (discriminator) network to distinguish between real and inpainted samples for T_2 iterations using cross-entropy loss. In Stage-3, both completion and critic networks are iteratively updated under the min-max GAN game formulation [16] for T_3 iterations.

Implementation Details We first discuss how we select design hyper parameters of our network modules such as *LIST* and *GSAT*. For a given parameter setting, we train on CelebA dataset and evaluate the FID on CelebA validation set (10000 samples). Due to lack of massive computational resources, we run parameter search sweep only on CelebA and adopted our understanding on other datasets. It is encouraging to see that lessons learned from CelebA generalize well to other datasets also. We set T_1, T_2 and T_3 to 10^6, 10^5 and 10^6 iterations. Mini batch gradient descent based optimization is performed with ADAM [35] optimizer with batch size = 64. Following [39], we perform paired two-sided Wilcoxon signed-rank tests and significance level set to 10^{-4}.

1) Hyper-parameters Search:

Design parameters for *LIST* module: A *LIST* module is characterized by the two hyper parameters, k and n_b. Firstly, we study the effect of reducing 3×3 kernels in the network by varying $\frac{1}{n_b} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$. To keep things constant, dilation layer for each case was realized with normal dilated convolution and $\frac{1}{k}$ fixed at 0.25. In Table I we report FID metrics on CelebA validation set at a hole size of 48×48. Decreasing $\frac{1}{n_b}$ (pushing more computations to 3×3 stream) less than 0.5 does not improve FID appreciably but increases the model parameters while FID deteriorates briskly with increase of $\frac{1}{n_b}$ (pushing more computations to 1×1 stream). We thus keep n_b = 2 in our further experiments. Such a balance of channels along two parallel processing streams is also recommended in [44], [54]. Next, we sweep over different settings of $\frac{1}{k}$ at a fixed $\frac{1}{n_b}$ = 0.5. Increasing $\frac{1}{k}$ improves the representation efficacy of the Stage-1 1×1 layer and thus aids in FID improvement but at a cost of higher parameters. With $\frac{1}{k} \geq$ 0.35, FID improvement almost saturates.

Finally, to find a suitable threshold of FID aligned with human perception, we showed 100 inpainted images of five models with FID $\in$ $[6.5, 7.5]$ (model with FID $\geq$ 8 are perceptually not acceptable) to five independent raters who were asked to rate a given image in $[1, 5]$; 5: excellent and 1: bad quality. The difference of mean scores of models with FID $\leq$ 7.0 were statistically insignificant. With $\frac{1}{n_b}$ = 0.5, from Table I we see that $\frac{1}{k}$ = 0.30 yields model in the regime of FID $\approx$ 7.0. Since channel counts in deep nets are usually of the form of 2^r, $r \in \mathbb{N}$, we proceed in the remaining paper with k = 4 ($\frac{1}{k}$ = 0.25) and n_b = 2 ($\frac{1}{n_b}$ = 0.5).

Number of Groups in *GSAT* layer: Our proposed *GSAT* module is characterized by number of groups, g, for the group convolution layers. For simplicity of parameter sweep, we keep the group numbers same for dilated 3×3 and 1×1 stages. In Table II we report FID scores on CelebA validation set for different values of g. For other layers, all models used *LIST* with k = 4 and n_b = 2 as discussed in previous section. A smaller value of g indicates more computational load on the initial 3×3 layers and subsequent better FIDs. However, at g = 8, we get FID $\approx$ 7.0, which is perceptually acceptable. On the contrary, increasing g creates many independent feature volumes and the combined channel shuffle and 1×1 group convolution is not able to properly amalgamate the groups leading to higher FID. So, for future experiments we set g = 8 for *GSAT* layers.

2) Different Speedup Variants: In Table III we define the proposed architecture variants and compare the associated parameters and FLOPs. Such analysis gives a foundation to appreciate the effect of a given speedup technique. In Table V we compare the FID scores of different proposed models with full-scale baseline models. Some of the key lessons from Tables III and V:

– Comparing M_3 and M_4: Proposed *LIST* layer is a much more efficient alternative to depthwise separable 3×3 convolution layer, but both models have similar reconstruction performances.

– Comparing M_5 and M_6: Proposed *GSAT* layer used in M_6 as an alternative for normal dilated convolution layer significantly helps in reduction of parameters without

TABLE IV: Comparing FLOPs, number of trainable parameters and model sizes of inpainting models. Full capacity baseline models of GLCIC [30], GIP [76] and Shift [74] are compared against variants of our proposed efficient models, M_1, M_2, M_3, and M_6 derived from GLCIC.

	GLCIC	GIP	Shift	M_1	M_2	M_3	M_6
FLOPs (10^9)	65.0	41.2	70.1	33.1	27.1	26.9	**7.4**
Params (10^6)	6.02	2.98	6.24	3.42	2.93	2.81	**0.54**

TABLE V: FID (lower is better) of full-scale baselines of GIP [76], Shift[74] and GLCIC [30] and proposed efficient variants, M_1-M_6 derived from GLCIC.

Dataset	Full-Scale Baselines					
			GIP	Shift	GLCIC	
CelebA			6.98	6.95	7.00	
CelebA-HQ			8.12	8.00	8.05	
Places2			13.10	13.00	13.25	
DTD			6.00	6.01	6.04	
	Proposed Efficient Variants					
	M_1	M_2	M_3	M_4	M_5	M_6
CelebA	7.12	23.41	7.11	7.09	7.11	7.03
CelebA-HQ	8.09	27.21	8.16	8.14	8.10	8.09
Places2	13.27	30.41	13.27	13.29	13.30	13.39
DTD	6.04	18.21	6.84	6.06	6.06	6.04

hampering visual quality. Since, M_6 combines both *LIST* and *GSAT* layers, it is our preferred proposed model unless otherwise stated.

– As per our theoretical justification, a network with separable sub-pixel convolution (model M_2) performs worse than a network with normal convolution based sub-pixel convolution (model M_1) as reflected by higher FID scores of M_2. Also, see Fig. 5 for visualizing such failures.

– Comparing M_1 and M_3: Model, M_3, with bilinear upsampling + separable convolution has fewer FLOPs than M_1 (upsampled with sub-pixel convolution) while having similar FID. Thus it is prudent to have efficient bilinear upsampling which we improve further with proposed $LIST$ based upsampling in M_5.

3) Comparing with full-scale baselines: Since we design all our smaller models based on the architecture of GLCIC [30], it is fair to compare performances only with GLCIC as baseline. However, for initial benchmarking of our model designs we also compared against recent state-of-the-art deep learning based models of GIP [76] and Shift [74].

Reduction in Computation In Table IV we report the parameters count, FLOPs and mobile memory size. Our preferred model, M_6 achieves almost 91% (= $\frac{6.2-0.54}{6.2} \times 100\%$) relative parameters savings compared to the parent framework of GLCIC with 88.6% and 93.5% relative savings in FLOPs.

Comparison of Reconstruction In Table V we report FID metrics of the comparing methods @256×256 on CelebA-HQ, Places2, and DTD datasets. We did not find any significant difference of FID between any of our models (except M_2) and the full-scale baselines. In Fig. 6 we provide some inpainting examples by GLCIC, GIP, Shift and our preferred proposed model, M_6. Clearly, the reconstruction qualities of our proposed smaller model are indistinguishable from full-scale baselines.

Mean Opinion Score Testing (MOS): To further bolster our findings, we conducted MOS testing to visually quantify the quality of inpainting by different models. Raters were asked to rate an inpainted image in the scale of 1 (bad quality) to 5 (excellent quality). Total of 20 raters were selected for the study. From each dataset, each rater was shown 50 inpainted images by GIP, Shift, GLCIC and proposed models M_3, M_6 models. Original images were also rated. So, each rater rated 1200 samples (4 datasets × 6 models × 50 images). We used two random positioned holes (but same across all model for an image) of 64×64. In Table VI we report the MOS for each dataset. Encouragingly MOS also follows the trend of FID scores. Similar to our FID findings, the difference of MOS scores between our models and any of the full-scale baselines are not significant.

TABLE VI: Mean Opinion Score (MOS) by different full-scale baselines of GIP [76], Shift [74], GLCIC [30] and our cheaper variants, M_3 and M_6 derived from GLCIC. Last column shows MOS on original images.

Dataset	GIP	Shift	GLCIC	M_3	M_6	Original
CelebA	4.24	4.30	4.25	4.18	4.22	**4.42**
CelebA-HQ	4.17	4.20	4.14	4.13	4.18	**4.72**
Places2	4.00	3.97	3.95	3.93	3.98	**4.60**
DTD	4.36	4.38	4.41	4.32	4.40	**4.55**

TABLE VII: Comparing average run time (in seconds) on different mobile devices (first three rows) and a commodity CPU (last row). Full-scale baseline models of GLCIC [30], GIP [76] and Shift [74] are compared against our proposed efficient variants M_1, M_2, M_3, and M_6 derived from GLCIC.

Device	GLCIC	GIP	Shift	M_1	M_2	M_3	M_6
Mi A1	8.2	5.5	9.2	1.9	1.6	1.4	**0.8**
Motorola	8.0	5.4	9.1	1.7	1.5	1.2	**0.7**
Asus	5.8	3.1	6.2	1.0	0.8	0.6	**0.35**
CPU	2.1	0.8	1.4	0.49	0.42	0.38	**0.30**

4) Execution Time on Mobile and CPU: For comparison on mobile we select two low-end mobile device namely, Mi A1 and Motorola G5 S-Plus and one high-end Asus Zenfone 5Z all running on Android operating system. Mi and Motorola has 1.9GHz Qualcomm Snapdragon 625 processor while Asus has 2.8 GHz snapdragon 845 processor. TensorFlow Lite [46] was used for mobile execution and the framework was executed on a single thread. In Table VII we report the execution times on 256×256 resolution images. Our preferred model, M_6 consistently runs at milli-seconds interval compared to multiple seconds by the full-scale baselines. It is also evident that newer generation processor present in Asus mobile helps in faster execution compared to the lower-end models of Mi and Motorola. We also profiled the execution times of the models on CPU of a regular commodity laptop with Intel i5 processor and 8GB RAM @ 2.2GHz without any GPU acceleration. It is encouraging to see that even without GPU, model M_6 is able to inpaint approximately 3.3 second compared to 0.9, 1.25, and 0.7 second by [30], [74], [76] respectively. Another encouraging observation is that sub-pixel convolution based upsampling (models M_1 and M_2) is slower on resource constrained mobile platform than proposed bilinear upsampling followed by efficient convolution. This is attributed to the computationally heavy pixel-shuffle operation in M_1 and M_2. However, on a more resourceful platform such as CPU, this difference is nullified. This observation further strengthens the pragmatism of using bilinear upsampling based efficient upsampling instead of pixel-shuffle based upsampling.

5) Comparison with MobileNet and ShuffleNet: We also designed cheaper variants of GLCIC baseline using efficient

TABLE VIII: Comparing computational requirements of various efficient models derived from the full-scale baseline of GLCIC [30] for inpainting. Cheaper variants using MobileNet, ShuffleNet and ShuffleNetV2 modules are indexed by '$MobNet'$', '$ShNet'$' and '$ShNetV2'$' superscripts.

Method	FLOPs (10^9)	Params (10^6)	Mobile (s)	CPU (s)	Memory (MB)
GLCIC	65.0	6.02	5.8	2.1	40.1
GLCIC^{MobNet}	9.8	0.68	0.52	1.1	4.3
GLCIC^{ShNet}	11.4	0.79	0.86	1.3	5.7
GLCIC^{ShNetV2}	10.6	0.70	0.68	1.2	4.9
GLCIC^{M_6}(**Proposed**)	**7.4**	**0.54**	**0.35**	**0.3**	**2.6**

convolution units from MobileNet [29] and ShuffleNet [82] and ShuffleNetV2 [83]. However, as discussed earlier, these frameworks were targeted for classification tasks and lack any efficient designs for dilated convolution and upsampling operations. For example, both ShuffleNet and ShuffleNetV2 units are invalid on layers in which the number of input and output channels are not same. This is a common design for any upsampling layer. We could have used usual full-scale dilated and transposed convolution for these three frameworks, but for fair comparison with our compressed networks, we add two modifications to these competing frameworks. Firstly, for dilated convolution, we initially perform a dilated 3×3 depthwise convolution followed by 1×1 pointwise convolution. This, itself can be seen as a novel cheaper way of designing dilated convolution layer. Next, for upsampling, we perform bilinear upsampling followed by separable convolution. With these modified settings, we did not find any marked difference of visual quality between the cheaper models and baselines (samples provided in supplementary material for space constraints). From Table VIII we see that our recommended model, M_6 is much more computationally efficient than MobileNet and ShuffleNet variants and, more importantly, M_6 has all the necessary components to be seamlessly used in 'image-to-image' translation tasks.

B. Image Denoising

In this section we show the applicability of our modules to reduce the computational costs of recent state-of-the-art image denoising networks. Henceforth in all experiments we will be using the design strategy and components from our variant, M_6, to realize a cheaper version of a given baseline. We initially experimented with *'DnCNN'* framework of Zhang *et al.* [81] for synthetic All White Gaussian Noise (AWGN) removal. We term our proposed smaller variant as DnCNN^{M_6}. We also experimented to compress the more recent model of CBDNet [18] which showed appreciable performance on real-world unknown noise removal and has immediate applications in today's AI-enable cameras. We term the smaller model as CBDNet^{M_6}.

1) Datasets and Training Details:

Synthetic Dataset: We initially compared the performance of our cheaper realization of DnCNN on synthetic AWGN on the widely used BSD68 [58] dataset consisting of 68 test images. We experimented on four different noise levels of $\sigma = 10, 15, 25, 50$ and zero mean. We followed DnCNN to use 400 images with size 180×180 for training the network. Random patches of 40×40 were sampled for training.

Real World Dataset: We also experimented with datasets perturbed with noise from real life unknown noise distributions usually encountered while capturing pictures with contemporary cameras. For this, we followed the procedures in CBDNet for training our models. A combination of synthetic noise images and real noisy images (120 from RENOIR[3], 400 images from BSD500[50], 1600 images from Waterloo[47], and 1600 images from MIT-Adobe FIve[6]) were used for training.

2) Denoising Performance:

Since all the models are trained to minimize reconstruction loss (instead of adversarial loss), it is pragmatic to compare the models directly in terms of PSNR and SSIM (Structural Similarity Index $\in 0, 1)$ instead of FID. Also, FID calculation requires at least a few thousand samples. However, our test set has a few hundred samples and thus FID metric would not have been a faithful representation of performance.

Denoising on Synthetic Dataset: In Table IX we report the denoising performances of baseline DnCNN and our proposed DnCNN^{M_6} in terms of PSNR and SSIM for AWGN noise removal. SSIM $\in 0, 1$ is the acronym for Structural Similarity Index. It is used a metric for comparing similarity between two images. SSIM = 1 means perfect match between two images. Across all noise levels, our model has comparable performance to that of DnCNN baseline.

Denoising on Real Dataset For quantitative evaluation we used the publicly available PolyU dataset [73] containing pairs of real-world noisy and ground truth images. The average PSNR and SSIM for full-scale CBDNet net is 37.95dB and 0.951 while for proposed CBDNet^{M_6} is 37.29dB and 0.948 Again, the differences are not significant. It is encouraging to see that even on real-world noise removal, our compressed variant performs at par with the full-scale CBDNet. Some visual comparisons are provided in Fig. 7. Additionally, for qualitative evaluations, we used the high-resolution DND [55] dataset in which the ground truths are not publicly available. Due to size limitations we include DND results in this Google Drive link.

Human Rating: In Table XI we report the MOS on different datasets. For each dataset, each subject was shown 20 random pairs of noisy and denoised (either from baseline or from our our compressed variant). Total 10 humans participated in the study. The grading strategy (between 0-5) was kept same as that we used during inpainting. We did not find any statistically significant difference (significance set to 10^{-4}) between the MOS of baselines and our variant on any of the datasets.

3) Reduction in Computation:

We report the total number of parameters for the full-scale baseline models of DnCNN and CBDNet and our proposed compressed versions in Table X. On DnCNN we achieve 87.27% and on CBDNet we achieve 90.2% relative savings of parameters. Since the models are fully convolutional, any arbitrary resolution of image can be processed. Thus reporting a specific count of FLOPs is not possible. However, for reference, in Table X we report the FLOPs for processing input image of resolution

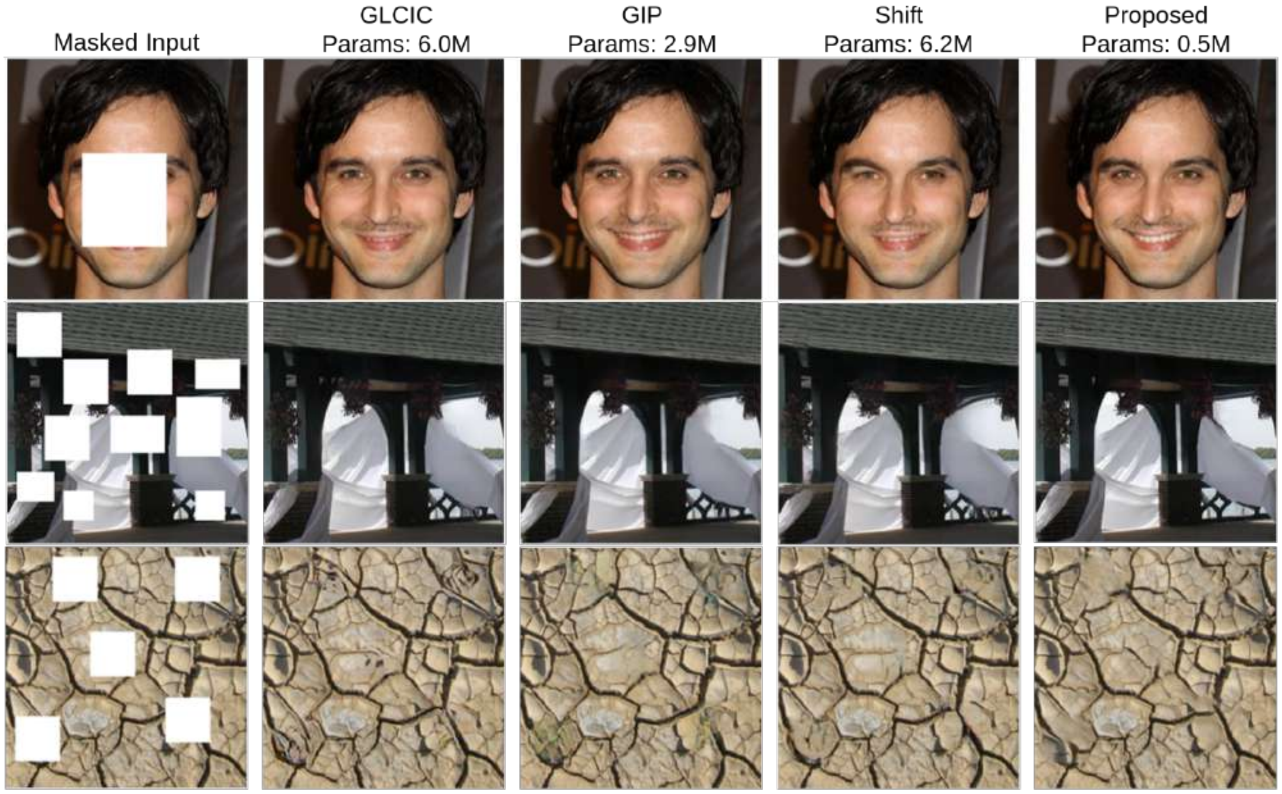


Fig. 6: Visual comparison on inpainting on CelebA-HQ (Row 1), Places2 (Row 2) and DTD (Row 3). GLCIC [30], GIP [76] and Shift [74] are full-scale baselines. Our proposed model is significantly cheaper in terms on parameters yet generates similar quality reconstructions. Best viewed when zoomed in.

TABLE IX: Comparison of PSNR and SSIM for AWGN denoising on BSD68 dataset by full capacity baseline of DnCNN and our proposed cheaper variant DnCNN^{M_6}.

	Noise Level (σ) →	10		15		25		50	
		PSNR(dB)	SSIM	PSNR(dB)	SSIM	PSNR(dB)	SSIM	PSNR(dB)	SSIM
Methods	DnCNN	33.78	0.92	31.75	0.89	29.23	0.83	26.29	0.72
	DnCNN^{M_6} (**Proposed**)	33.66	0.92	31.50	0.88	29.11	0.82	26.10	0.71

TABLE X: Computational requirements of different denoising networks. DnCNN [81] and CBDNet [17] are two different full-scale denoising baselines. Cheaper variants using MobileNet, ShuffleNet and ShuffleNetV2 modules are indexed by '$MobNet$', '$ShNet$' and '$ShNetV2$' superscripts. Interesting to note that CBDNet mainly operates on down-sampled feature space unlike DnCNN which operates on full-resolution. So CBDNet baseline has lower FLOPs compared to DnCNN even though the former has more parameters.

Method	FLOPs (10^9)	Params (10^6)	Mobile (s)	CPU (s)	Memory (MB)
DnCNN	36.73	0.55	3.43	0.58	7.8
CBDNet	36.09	4.34	3.30	0.49	29.4
DnCNN^{MobNet}	5.73	0.08	0.34	0.20	0.46
DnCNN^{ShNet}	7.44	0.18	0.41	0.24	0.6
DnCNN^{ShNetV2}	7.30	0.14	0.37	0.22	0.5
CBDNet^{MobNet}	6.23	0.6	0.40	0.27	3.1
CBDNet^{ShNet}	8.09	0.72	0.52	0.30	4.3
CBDNet^{ShNetV2}	7.60	0.69	0.48	0.28	4.1
DnCNN^{M_6} (**Proposed**)	2.97	0.04	0.16	0.11	0.21
CBDNet^{M_6} (**Proposed**)	4.12	0.41	0.25	0.14	1.93

TABLE XI: MOS of full-scale baseline of DnCNN [81] and proposed cheaper variant, DnCNN^{M_6}, for AWGN denoising on Set68. For real-world denoising on PolyU dataset we compare baseline of CBDNet [17] and our cheaper variant, CBDNet^{M_6}. Last column shows MOS on original images.

Dataset	DnCNN	CBDNet	Proposed	Original
Set68 (σ =10)	4.58	-	4.60	4.72
Set68 (σ =15)	4.34	-	4.32	4.72
Set68 (σ =25)	4.10	-	4.11	4.72
Real (PolyU)	-	4.50	4.49	4.83

256×256. Proposed CBDNet^{M_6} achieves 89.4% relative savings in FLOPS compared to CBDNet. We also compare against corresponding compressed variants of DnCNN and CBDNet with MobileNet, ShuffleNet and ShuffleNetV2 modules. Our proposed variant is more efficient in terms of memory requirement and FLOPs compared to both MobileNet and ShuffleNet variants.

4) Performance on Mobile and CPU:: In Table X we compare the execution times (@ 256×256) on mobile (Asus) and CPU and also the model sizes for mobile deployment. Both of our proposed variants are computationally more economic compared to full-scale baselines as well as MobileNet and ShuffleNet variants.

Image denoising is an essential component in majority of contemporary AI-enabled smartphones and the above presented results make our compressed variant a natural substitute for the full-scale models on mobile platforms.

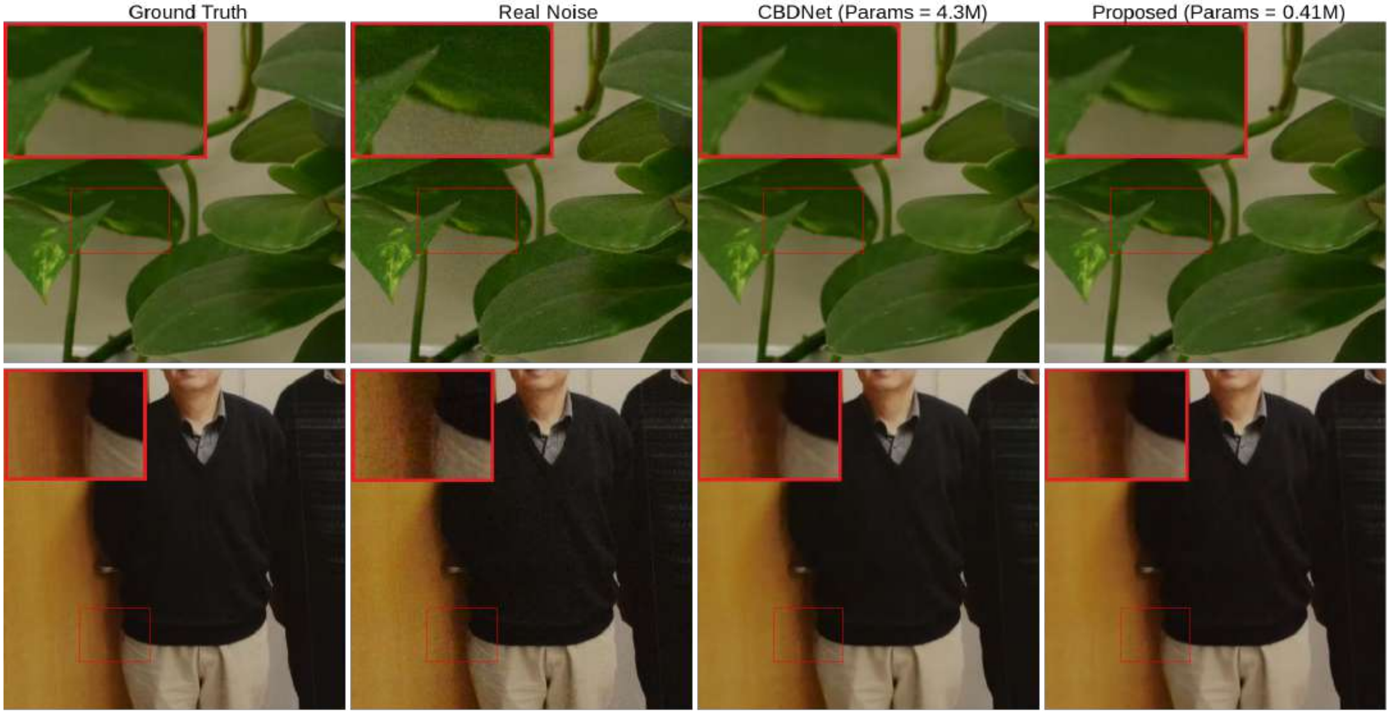


Fig. 7: Image denoising on real-world PolyU dataset by full-scale baseline of CBDNet and proposed cheaper variant, CBDNet^{M_6}. More examples provided in supplementary material.

C. Application in Image Super-Resolution

In this section we showcase the efficacy of our modules for single image super-resolution. For this, we consider the benchmark SRGAN model [39] as the baseline for 4× up-scaling. The baseline network consists of series of residual blocks (realized with 3×3 convolution) and upsampling is achieved with sub-pixel convolution with pixel-shuffle operation. We again follow the design principles of our model, M_6 to realize cheaper variant of SRGAN.

1) Datasets and Training Details: We used the training partition of Places2 dataset [86] to train baseline and proposed models. Similar to SRGAN we tested the models on the Set5 [5], Set14 [79], and BSD 100 (testing set of BSD300 [49]) dataset. Following [39], we randomly cropped 96×96 patch from a given image as HR (high resolution) target and down-sample with bicubic interpolation by 4× to create the corresponding LR (low resolution) input.

We follow the exact same protocol of stagewise training as done in [39]. Initially, we train the network with only ℓ_2 reconstruction loss. The authors term this network as SRResNet. For our smaller model, we term this network as SRResNet^{M_6}. Next, we fine-tune the network with VGG-54 content loss and an adversarial loss. Network at this stage is termed at SRGAN for baseline network and SRAGN^{M_6} for our proposed smaller network.

2) Super-resolution Performance:
Quantitative Comparison: In Table XII we first compare the PSNR (in dB) of SRResNet and SRResNet^{M_6}. Since, both of these models are trained on MSE loss, we compare the PSNR metric. Based on the average PSNR, we could not find any significant difference (significance level set to 10^{-4}) between the two models.

TABLE XII: PSNR (in dB) for 4× super-resolution by baseline SRResNet [39] and our proposed cheaper variant SRResNet^{M_6} on Set5, Set14, and BSD100 datasets.

Dataset	SRResNet	SRResNet^{M_6} **(Proposed)**
Set5	31.85	31.72
Set14	27.90	27.74
BSD100	27.01	26.90

Qualitative Comparison Next, we conducted a MOS test for the 2 models with 10 independent raters. Each rater was shown the original HR image and the super resolved versions by SRGAN and SRGAN^{M_6} networks. In Table XIV we report the MOS on the three datasets. Again, we could not find any significant difference between the scores received by the models. In Fig. 8 we visualize some super-resolved images by the two models. It is visually challenging to distinguish samples from the full-scale SRGAN baseline and our cheaper variant. More examples provided in supplementary material.

3) Reduction in Computation: In Table XIII we report the total number of parameters and FLOPs of different models. FLOPs were calculated on BSD100 dataset in which the original images are usually of dimension 480×320. or 320×480. So, for 4× super-resolution, input resolution is either 80×120 or 120×80. Compared to the baseline of SRGAN, our proposed cheaper variant, SRGAN^{M_6} achieves relative parameters and FLOPs savings of 88.4% and 99%. Proposed model is also appreciably cheaper compared MobileNet and ShuffleNet variants.

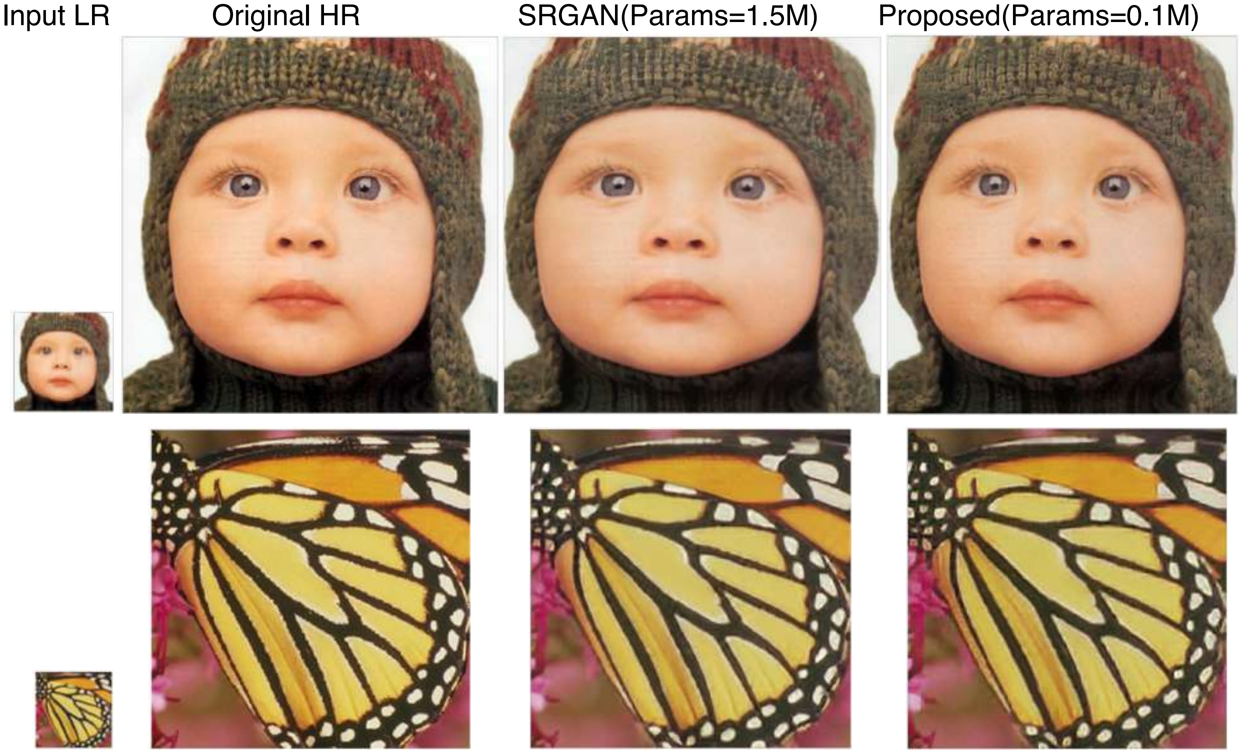


Fig. 8: 4× super-resolution by full-scale baseline of $SRGAN$ and our proposed cheaper variant, $SRAGN^{M_6}$.

TABLE XIII: Computational details of different super-resolution networks. SRGAN is the full-scale baseline. Cheaper variants using MobileNet, ShuffleNet and ShuffleNetV2 modules are indexed by '$MobNet$', '$ShNet$' and '$ShNetV2$' superscripts.

Method	FLOPs	Params	Mobile	CPU	Memory
	(10^9)	(10^6)	(s)	(s)	(MB)
SRGAN	38.4	1.55	3.48	0.65	12.8
SRGAN^{MobNet}	0.42	0.19	0.05	0.03	1.1
SRGAN^{ShNet}	1.08	0.42	0.11	0.07	2.2
SRGAN^{ShNetV2}	1.05	0.40	0.09	0.05	1.7
SRGAN^{M_6} **(Proposed)**	**0.27**	**0.10**	**0.02**	**0.01**	**0.5**

TABLE XIV: Mean Opinion Score for 4× super-resolution on outputs of baseline SRGAN-54 [39], our proposed cheaper variant SRGAN-54^{M_6} and original high-resolution images.

Dataset	SRGAN	SRGAN^{M_6} **(Proposed)**	Original
Set5	3.62	3.64	4.45
Set14	3.69	3.72	4.41
BSD100	3.50	3.49	4.23

4) Performance on Mobiles: In Table XIII we report the mobile model sizes and execution times on the Asus mobile and the commodity CPU. Proposed variant saves 92.1%, 47.1%, 76.1% and 75.0% on mobile memory compared to SRGAN-54 baseline, MobileNet, ShuffleNet and ShuffleNetV2 variants respectively. Execution speeds are reported on BSD100 dataset. Our proposed variant achieves significant speedup and reduction of FLOPs compared to full-scale baseline and even MobileNet and ShuffleNet versions.

V. Conclusion

In this paper we introduced several convolutional building blocks for low-level restoration tasks. Our proposed modules, *LIST* and *GSAT* were shown to be task agnostic and generalized to variety of restoration tasks. We showed that with specific design consideration, *LIST* layer can be made low cost computationally than contemporary de facto choices of depthwise separable and group convolution based 3×3 layer. We analytically and empirically analyzed the shortcoming of using depthwise separable kernels to realize sub-pixel convolution based upsampling in an encoder-decoder network configuration. Instead of we showed that homogeneity of network structure can be maintained by deterministic upsampling (instead of transposed convolution or pixel-shuffle based upsampling) followed by efficient convolution with *LIST* layer. Extensive evaluations on resource constrained platforms revealed the effectiveness of our modules in designing computationally efficient yet visually accurate models.

Acknowledgment

The work is funded by a Google PhD Fellowship and Qualcomm Innovation Fellowship awarded to Avisek.

Avisek is a Ph.D. candidate at the Indian Institute of Technology Kharagpur where he is focusing on image/video reconstruction tasks such as inpainting, super-resolution. His other research interests include data-efficient training of deep neural networks. He is recipient of Google PhD Fellowship and twice recipient of Qualcomm Innovation Fellowship. Avisek was selected as a Young Researcher by the Heidelberg Laureate Forum, 2019. Prior to his Ph.D, Avisek completed his M.S (by research) from IIT Kharagpur with focus on statistical machine learning.

Sourav received his M.tech degree from the department of Electronics and Electrical Communication Engineering, IIT Kharagpur, Kharagpur, India. He is currently working as an advanced deep learning engineer at Mathworks India Pvt. Ltd., Hyderabad. He received the ”Institute Silver Medal” for his academic performance at IIT Kharagpur. He received the ”Best Student Award” with a gold medal for academic performance during his B.Tech at Kalyani Government Engineering College, Kalyani, India. He has also received many scholarships and awards from the Government of West Bengal for securing 3^{rd} rank at state level in Class X Board examination (Madhyamik) and 8^{th} rank at state level in Class XII Board examination (Higher Secondary). His current research interest lies in deep learning, machine vision and generative adversarial models.

Sutanu received BS degree in Electronics and Communication Engineering from the West Bengal University of Technology in 2015 and the MS degree in Medical Imaging and Informatics from the Indian Institute of Technology Kharagpur in 2018. He was awarded the Institute Silver Medal at the time of M.Tech. He is currently a Ph.D. student in the Department of Electrical and Electronics Communication Engineering, Indian Institute of Technology Kharagpur. His current research interest is deep learning for image restoration, medical image restoration, low-level image processing.

Siddhant is a final year integrated master’s student at Indian Institute of Technology Kharagpur. He is majoring in Electrical Engineering with a minor in Computer Science. He has been doing research in the domain of computer vision, natural language processing, adversarial attacks and causal inference.

Prof. Prabir Kumar Biswas (M93SM03) received the B.Tech. (Hons.), M.Tech., and Ph.D. degrees from IIT Kharagpur, Kharagpur, India, in 1985, 1989, and 1991, respectively. He was a Visiting Fellow with the University of Kaiserslautern, Kaiserslautern, Germany, under the Alexander von Humboldt Research Fellowship from 2002 to 2003. Since 1991, he has been a Faculty Member with the Department of Electronics and Electrical Communication Engineering, IIT Kharagpur, where he is currently a Professor, and is the Head of the Department. He has authored over 100 research publications in international and national journals and conferences, and has filed seven international patents. His current research interests include image processing, pattern recognition, computer vision, video compression, parallel and distributed processing, and computer networks.

References

[1] N. Ahn, B. Kang, and K. Sohn. Efficient deep neural network for photo-realistic image super-resolution. *arXiv preprint arXiv:1903.02240*, 2019.

[2] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*, 2017.

[3] J. Anaya and A. Barbu. Renoir–a dataset for real low-light image noise reduction. *Journal of Visual Communication and Image Representation*, 51:144–154, 2018.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[5] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012.

[6] V. Bychkovsky, S. Paris, E. Chan, and F. Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR 2011*, pages 97–104. IEEE, 2011.

[7] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[8] B. Ceulemans, S.-P. Lu, G. Lafruit, P. Schelkens, and A. Munteanu. Efficient mrf-based disocclusion inpainting in multiview video. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2016.

[9] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

[10] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[11] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014.

[12] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[13] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[14] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[17] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang. Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1712–1722, 2019.

[18] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang. Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1712–1722, 2019.

[19] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka. Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. In *2018 IEEE Winter Conference*

on Applications of Computer Vision (WACV), pages 1442–1450. IEEE, 2018.

[20] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[21] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[22] K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3154–3160, 2017.

[23] A. He, C. Luo, X. Tian, and W. Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[25] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360, 2015.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[27] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, pages 6626–6637, 2017.

[28] A. Howard, A. Zhmoginov, L.-C. Chen, M. Sandler, and M. Zhu. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. In *CVPR*, 2018.

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

[30] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017.

[31] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[32] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos. Large pose 3d face reconstruction from a single image via direct volumetric cnn regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1031–1039, 2017.

[33] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018.

[34] M. Kim, C. Park, S. Kim, T. Hong, and W. W. Ro. Efficient dilated-winograd convolutional neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2711–2715. IEEE, 2019.

[35] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012.

[37] A. Lahiri, A. K. Jain, S. Agrawal, P. Mitra, and P. K. Biswas. Prior guided gan based semantic inpainting. In *CVPR*, pages 13696–13705, 2020.

[38] A. Lahiri, A. K. Jain, D. Nadendla, and P. K. Biswas. Faster unsupervised semantic inpainting: A gan based approach. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2706–2710. IEEE, 2019.

[39] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[40] Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative face completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.

[41] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[42] T.-L. Lin, C.-J. Wang, T.-L. Ding, G.-X. Huang, W.-L. Tsai, T.-E. Chang, and N.-C. Yang. Recovery of lost color and depth frames in multiview videos. *IEEE Transactions on Image Processing (T-IP)*, 27(11):5449–5463, 2018.

[43] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[44] P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic segmentation using adversarial networks. In *NeurIPS Workshop on Adversarial Training*, 2016.

[45] G. Luo, Y. Zhu, Z. Weng, and Z. Li. A disocclusion inpainting framework for depth-based view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2019.

[46] I. Ltkebohle. Tensorflow: a system for large-scale machine learning. https://www.tensorflow.org/lite//, 2016.

[47] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang. Waterloo exploration database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 26(2):1004–1016, 2016.

[48] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[49] D. Martin, C. Fowlkes, D. Tal, J. Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[50] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. In *Advances in Neural Information Processing Systems*, pages 1279–1286, 2003.

[51] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[52] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[53] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[54] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, pages 75–91. Springer, 2016.

[55] T. Plotz and S. Roth. Benchmarking denoising algorithms with real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1586–1595, 2017.

[56] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[57] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang. Single image dehazing via multi-scale convolutional neural networks. In *European conference on computer vision*, pages 154–169. Springer, 2016.

[58] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205, 2009.

[59] T. Schöps, M. R. Oswald, P. Speciale, S. Yang, and M. Pollefeys. Real-time view correction for mobile devices. *IEEE transactions on visualization and computer graphics*, 23(11):2455–2462, 2017.

[60] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[61] W. Shi, J. Caballero, L. Theis, F. Huszar, A. Aitken, C. Ledig, and Z. Wang. Is the deconvolution layer the same as a convolutional layer? *arXiv preprint arXiv:1609.07009*, 2016.

[62] L. Sifre and S. Mallat. Rigid-motion scattering for image classification. *Ph. D. dissertation*, 2014.

[63] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016.

[64] K. Sun, M. Li, D. Liu, and J. Wang. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. In *British Machine Vision Conference (BMVC)*, 2018.

[65] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[66] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[67] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3147–3155, 2017.

[68] Z.-W. Tan, A. H. Nguyen, and A. W. Khong. An efficient dilated convolutional neural network for uav noise reduction at low input snr. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1885–1892. IEEE, 2019.

[69] C. Tian, Y. Xu, and W. Zuo. Image denoising using deep cnn with batch renormalization. *Neural Networks*, 121:461–473, 2020.

[70] Y. Wei, H. Xiao, H. Shi, Z. Jie, J. Feng, and T. S. Huang. Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation. In *Proceedings of the IEEE Conference on*

Computer Vision and Pattern Recognition, pages 7268–7277, 2018.

[71] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[72] G. Xie, J. Wang, T. Zhang, J. Lai, R. Hong, and G.-J. Qi. Interleaved structured sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8847–8856, 2018.

[73] J. Xu, H. Li, Z. Liang, D. Zhang, and L. Zhang. Real-world noisy image denoising: A new benchmark. *arXiv preprint arXiv:1804.02603*, 2018.

[74] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan. Shift-net: Image inpainting via deep feature rearrangement. In *ECCV*, pages 1–17, 2018.

[75] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[76] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *CVPR*, 2018.

[77] R. Yu, X. Xu, and Y. Shen. Rhnet: Lightweight dilated convolutional networks for dense objects counting. In *2019 Chinese Control Conference (CCC)*, pages 8455–8459. IEEE, 2019.

[78] Q. Yuan, Q. Zhang, J. Li, H. Shen, and L. Zhang. Hyperspectral image denoising employing a spatial–spectral deep residual convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing*, 57(2):1205–1218, 2018.

[79] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, pages 711–730. Springer, 2010.

[80] J. Zhang and D. Tao. Famed-net: a fast and accurate multi-scale end-to-end dehazing network. *IEEE Transactions on Image Processing*, 29:72–84, 2019.

[81] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[82] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018.

[83] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[84] Z. Zhang, X. Wang, and C. Jung. Dcsr: Dilated convolutions for single image super-resolution. *IEEE Transactions on Image Processing*, 28(4):1625–1635, 2018.

[85] F. Zhao, J. Feng, J. Zhao, W. Yang, and S. Yan. Robust lstm-autoencoders for face de-occlusion in the wild. *IEEE Transactions on Image Processing*, 27(2):778–790, 2017.

[86] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016.

[87] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2018.