Single Image Super-Resolution: Efficiency and Real-World Performance

Project Report

Mohammad Ali Jauhar
ENS Paris-Saclay
4 Avenue des Sciences, France 91190
mohammad.jauhar@ens-paris-saclay.fr

Abstract

Single image super-resolution is a challenging task. Current solutions use deep neural networks to achieve excellent performance. However, deep neural networks are inherently compute and memory intensive and present high latency. As such, developing architectures that are efficient and suitable for real-world deployment on, for example, mobile phones, is even more challenging. However, we are seeing great interest in the field with regular challenges organised in top computer vision conferences. In the same spirit, in this project, we evaluate two lightweight architecture, LESRCNN and Omni-SR, built around a vanilla CNN framework and an attention framework, respectively and assess their efficiency in terms of FLOPs, model parameters, and CPU & GPU runtime. We also evaluate both models on real-world dataset, namely RealSR and DRealSR. Finally, we present the difficulties faced in performing the experiments.

Codes and notebooks used for the results could be accessed here[1].

1. Introduction

Image Super Resolution refers to the task of restoring high-resolution (HR) images from one or more low-resolution (LR) observation of the same scene. Based on the number of inputs, Image super resolution could be of two types, single image super resolution (SISR) or multi-image super resolution (MISR). In general use, SISR is more popular than MISR because of its high efficiency. However, SISR in itself is a very challenging problem because a single low-resolution image could correspond to many high-resolution images where having many input images could help in narrowing down the high-resolution image space. Therefore, the problem is intractable (NP-hard) and thus tradition methods perform poorly.

[1] https://github.com/majauhar/DL_MVA

In a typical SISR framework, the low resolution image y is modelled as follows:

$$y = (x \otimes k) + n \tag{1}$$

where $x \otimes k$ is the convolution between the blurry kernel k and the unknown high-resolution image x, and n is the independent noise term.

Current SISR techniques could be divided into three categories: interpolation-based methods, reconstruction-based methods, and learning-based methods. Interpolation-based methods, such as bicubic interpolation and Lanczos resampling are fast and straightforward. But the suffer from accuracy issues. Reconstruction-based methods often adopt sophisticated prior knowledge to restrict the possible solution space with an advantage of generating flexible and sharp details. However, the performance of many reconstruction-based methods degraded rapidly when the scale factor increases. Furthermore, reconstruction-based methods are very slow.

Learning-based methods utilise machine learning (ML) algorithms and are relatively fast and perform very well. Unlike traditional methods, these methods tries learning the statistical relationships between low-resolution and corresponding high-resolution counterpart from a large number of training examples. Traditional ML techniques such as Markov random field, neighbour embedding, sparse coding, random forest have been used in the past for SISR, often in conjugation with other techniques such as reconstruction-methods. While traditional ML techniques worked well in many scenarios, true breakthrough in SISR has been achieved with the advent of deep learning.

As with the innovations in deep learning architectures, so has the implementation of SISR evolved. Convolutional neural networks (CNNs) are traditionally the go-to architecture for vision tasks and thus have formed the basis for successful works. CNN based approaches are usually linear networks, such as SRCNN [6], or residual networks such as FormResNet [8]. Subsequently, Generative adversarial networks (GANs), such as ESRGAN [17], were also shown to

perform very well, albeit with huge training cost. Current state-of-the-art methods, however, are based on transformer architecture.

2. Problem Definition

Even with the outstanding performance of current methods, a number of challenges still persist. Most state-of-the-art network employ overparameterised deep neural networks such as CNNs, Transformers, or a combination of both, which results in networks that are compute and memory intensive and infer with high latency, thus making them unsuitable for deployment on edge devices such as smartphones.

The efficiency of a deep neural network can be measured using different metrics, including runtime, number of parameters, computational complexity (FLOPs), activations, and memory consumption. These metrics affect the deployment of deep neural networks in various ways. Among them, runtime is the most direct indicator of a network's efficiency. Increased computational complexity is associated with higher energy consumption, which could shorten the battery life of edge devices. Furthermore, the number of parameters is related to AI chip design, with more parameters resulting in larger chip areas and increased costs for the designed AI devices.

Furthermore, most of the performance results are shown on curated dataset where low-resolution images are generated algorithmically, for example, by using bicubic downsampling. However, real-world low-resolution images aren't generated this way and therefore, models published in literature don't often work very well in real-world scenarios.

Performance on real-world images and efficiency are jointly important for a assessing real-world performance. In this context, we look at two SISR architecture, one based purely on CNN (LESRCNN[15]) and another which has transformers (Omni-SR [16]) at its core. We analyse them on their compute and memory efficiency and assess their performance on real-world datasets.

2.1. Models

2.1.1 LESRCNN

Lightweight image super resolution with enhanced CNN is a convolutional neural network that is built with three successive sub-blocks, with an information extraction and enhancement block (IEEB), a reconstruction block (RB) and an information refinement block (IRB). Specifically, the IEEB extracts hierarchical low-resolution (LR) features and aggregates the obtained features step-by-step to increase the memory ability of the shallow layers on deep layers for SISR. To remove redundant information obtained, a heterogeneous architecture is adopted in the IEEB. After that, the RB converts low-frequency features into high-frequency features by fusing global and local features, which is complementary with the IEEB in tracking the long-term dependency problem. Finally, the IRB uses coarse high-frequency features from the RB to learn more accurate SR features and construct a SR image. The LESCRNN can obtain high-quality images by a model for different scales.

Architecturally, it is a 23-layer architecture consisting of 3 blocks, an IEEB, a RB, and an IRB. The 17-layer IEEB extracts and enhances the low-frequency features, and then refines the extracted low-frequency features to reduce computation. Then, the 1-layer RB converts these low-frequency features to high-frequency features. Finally, the 5-layer IRB refines the coarse high-frequency features extracted by the RB to derive more accurate SR features. The SR process with LESRCNN can be formulated as follows:

$$SR = f_{IRB}(f_{RB}(f_{IEEB}(LR))) \tag{2}$$
$$= f_{LESRCNN}(LR), \tag{3}$$

where LR and SR are low resolution and reconstructed images, respectively. The model uses an $L2$ loss objective.

2.1.2 Omni-SR

Omni aggregation networks for lightweight image super-resolution is a hybrid network transformers and convolutional neural networks with a self-attention mechanism termed omni-self attention (OSA) at its core. It is based on ligthweight vision transformers (ViT) that tackles the problem of not modelling spatial and intra-channel correlation together. The proposed OSA blocks model the pixel-interaction on both spatial and channel perspectives, and thus provides for a better effective receptive field (ERF). On a high-level architecture perspective, Omni-SR first employs CNNs for extracting shallow features. After that, it employs a series of omni-scale aggregation groups (OSAG). Final reconstruction and upsampling is again delegated to convolutional layer.

Each OSAG contains 4 blocks. Firstly, a CNN block, followed by meso-OSA (for intermediary features) and global-OSA (for global features) block, and then an enhanced spatial attention (ESA) block.

The LR to HR process could be described as following:

$$z_{sf} = z_0 = f_{SF}(LR), \tag{4}$$

Where z_{sf} are the shallow features. This extracted shallow features are processed by a series of OSAGs as follows:

$$z_i = OSAG_i(z_{i-1}), \;\; i = 1, ..., N \tag{5}$$

The output from the last OSAG is then passed through a convolution layer to get the deep features as follows:

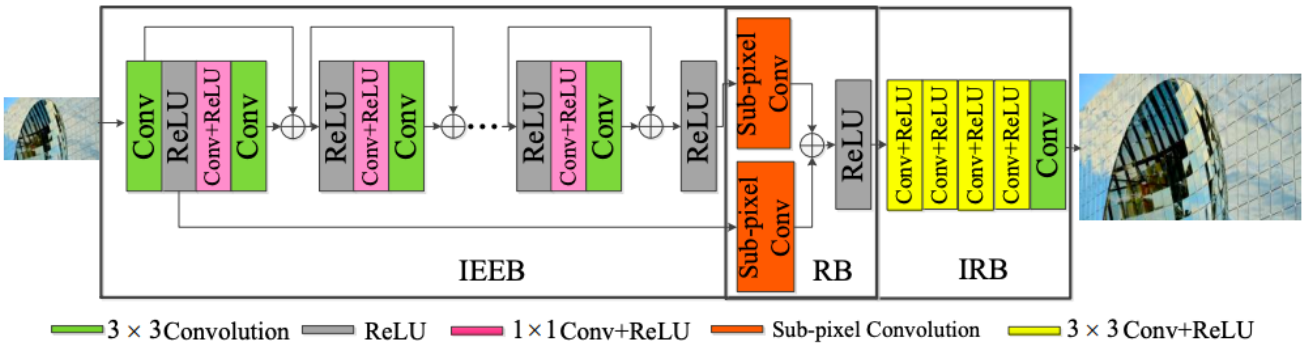

Figure 1. LESRCNN Architecture. Taken from [15].

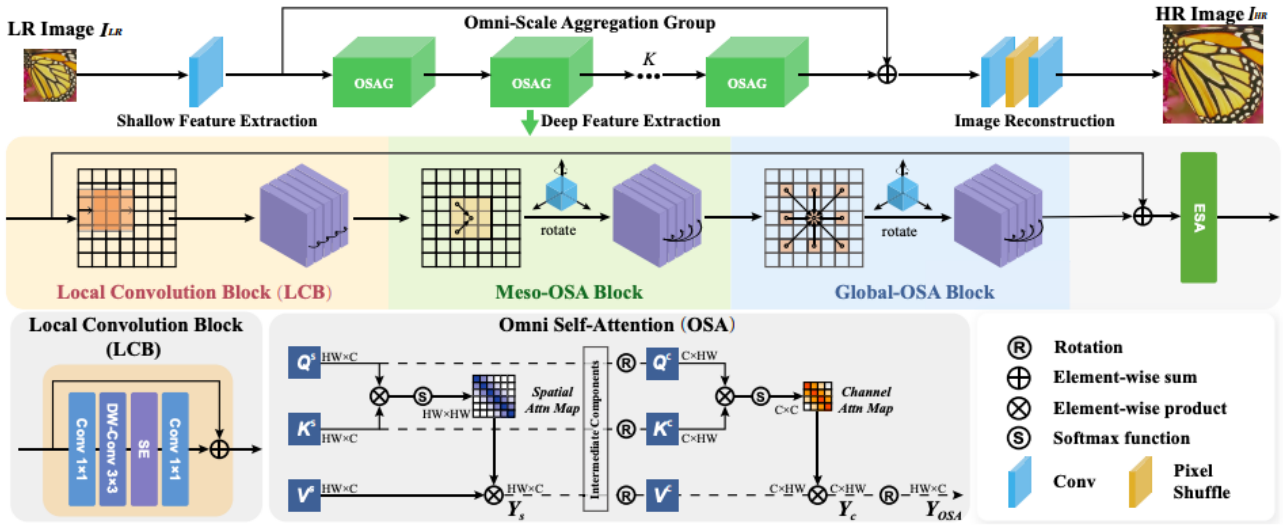

Figure 2. Omni-SR Architecture. Taken from [16].

$$z_{df} = f_{conv}(z_N) \tag{6}$$

Finally, the shallow and deep features pass through a reconstruction layer (a CNN) to generate the reconstructed image (SR) as follows:

$$SR = f_{recon}(z_{sf} + z_{df}) \tag{7}$$

Omni-SR uses a $L1$ loss objective.

2.2. Dataset

RealSR [3]: We use version 3 of the dataset which contains 459 scenes for training and 100 scenes for testing. The testing set contains 50 scenes shot by Nikon and Canon digital cameras each. LR and HR pairs are created by adjusting the focal length of the camera.

DRealSR [19]: This dataset contains 93 image pairs for each upscaling factors of x2, x3, and x4. The images are captured by 5 different digital (DSLR) cameras: Canon, Sony, Nikon, Olympus, and Panasonic. LR and HR pairs are created by zooming and then using SIFT [11] to crop them into matching pairs.

2.3. Metrics

PSNR: Peak signal-to-noise ratio is used to calculate the ratio between the maximum possible signal power and the power of the distorting noise which affects the quality of its representation. Thus, in the context of SISR, this is the ratio between the original HR image and the reconstructed (SR) image. The unit for PSNR is decibels (dB). Mathematically,

$$PSNR = -log(\frac{MSE}{S^2})\ dB, \tag{8}$$

where S is the maximum pixel value and MSE is the mean-squared error between the HR and SR images.

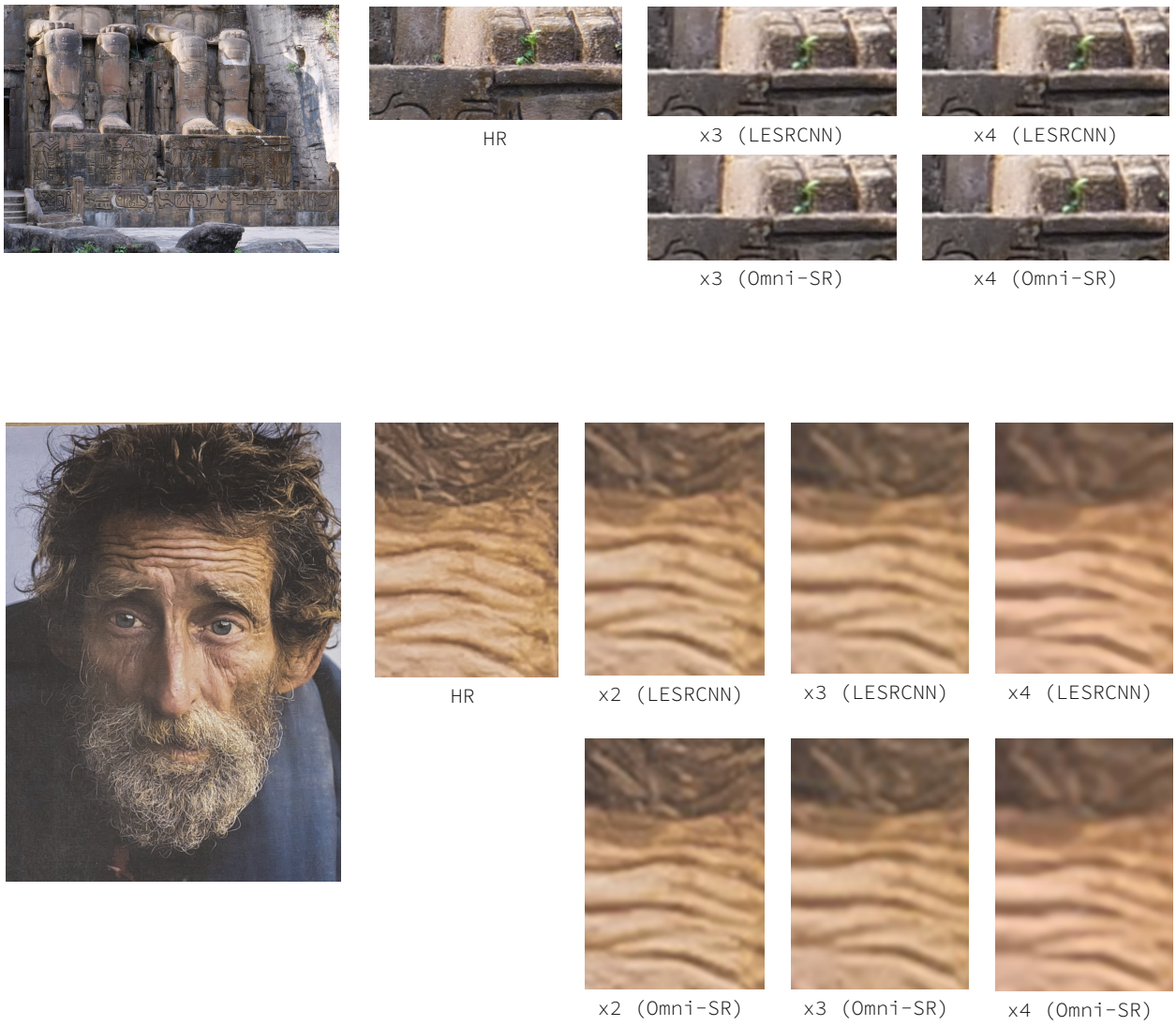

Figure 3. Visualisation on DRealSR (up) and RealSR(down) dataset sample. Zoom in for best visualisation

SSIM: Structural similarity index provides a metric of how human perceive an image and is a factor of luminance, contrast, and structural information of the reconstructed (SR) and ground truth (HR) image. Mathematically, at (i, j) pixel,

$$SSIM(i,j) = L(i,j) \times C(i,j) \times S(i,j), \qquad (9)$$

where $L(i, j)$ is the luminous similarity at location (i, j) of the image pairs. $C(i, j)$ is the contrast similarity at location (i, j) of the image pairs. And $S(i, j)$ is the structural similarity at location (i, j) of the image pairs. The final value of the image is the average of all pixels.

Runtime: Runtime is one of the most important efficiency criteria. A computing machine is considered real-time if the latency is $\sim$ 30 ms.

Model Complexity: These would include parameters such as total floating point operations (FLOPs), number of model parameters, number of activations of convolutional layers, number of convolutional layers, and memory consumption (CPU and GPU).

3. Related Work

Efficient deep neural networks has been an active research topic for a long time. Methods such as network pruning, low-rank filter decomposition, network quantisation, neural architecture search, and knowledge distillation

Table 1. Results of Efficiency of LESRCNN and Omni-SR, along with results of NTIRE 2023 top performers. All models have at least a PSNR of 29.00 dB. FLOPs in this table are calculated using NTIRE 2023 method, # param denote the total number of parameters, # Acts denote the total number of convolutional layer activations outputs, # Conv denote the total number of convolutional layers. Inference time is the time for inference averaged over 100 inputs of size 256x256.

Architecture		FLOPs [G]	# Params [M]	# Acts [M]	# Conv	Inference time (CPU) [ms]	Set5 (PSNR/SSIM)
LESRCNN	x2	80.18	0.63	173.74	26	612.50	-
	x3	153.33	0.81	301.53	28	1208.35	-
	x4	275.66	0.77	513.99	28	2135.21	31.64/0.886
Omni-SR	x2	134.89	2.83	3166.81	453	5909.67	-
	x3	135.46	2.84	3167.80	453	6461.19	-
	x4	136.25	2.85	3169.17	453	6023.81	**32.59/0.899**
MegSR [21]	x4	14.90	0.24	72.97	39	306.2	32.20/0.895
Zapdos [18]	x4	21.97	0.35	63.01	26	306.9	32.15/0.894
DFCDN [5]	x4	15.49	0.25	82.76	39	307.4	32.20/0.895

has been developed for deep neural networks and thereafter successfully transferred to SISR architectures.

SRCNN [6] is the original lightweight image super-resolution architecture and it uses a deep convolutional network. To reduce the number of parameters in classic CNN-based models, DRCN [9] uses a deep recursive network together with skip connections. *Fully quantised image super-resolution network* (FQSR) uses quantisation to improve efficiency by replacing expensive floating-point operations with efficient fixed-point or bitwise arithmetics. MegSR [21] uses iterative pruning to improve efficiency. RFDN [10] and DFCDN [5] employ feature distillation using shallow residual blocks, while MemSR [20] employs model distillation using a teacher-student training regime. Recently, *attention* based lightweight methods have also received some ..attention. ESRT [12] employs efficient transformer based attention while FERN employs channel attention in a feature enhancement residual network. Among more recent methods, NGswin [4] employs swin transformers with window self-attention and N-Gram locality context for achieving efficiency and performance.

4. Methodology

We first verify the the results of the two architectures, LESRCNN and OMNI-SR, on datasets Set5 [2], Set14 [22], B100 [13], Urban100 [7], and Manga109 [14] as mentioned in the respective papers as a sanity check. Thereafter, we evaluate their efficiency on the following metrics, CPU and GPU runtime, number of parameters, FLOPs, and activations. We do this for upscaling factors of x2, x3, and x4, for both networks. For standardisation and comparison with results in literatures, we follow the NTIRE 2023 [23] convention of using a (3x256x256) input LR image for all our experiments. Furthermore, the model running statistics are average over 100 runs to reduce the effect of randomness associated with computing processes' statistics.

Computing model complexity, particularly, in terms of FLOPs, is challenging for deep learning models. There are no hardware specific tools available and the methodology for computing such numbers are not standardised across different Python libraries. To highlight this fundamental issue, we also compute the number of FLOPs using two different methods. The first implementation used the methodology adopted by NTIRE 2023 and the second method used the Facebook research's fvcore[2]library.

Furthermore, to evaluate the models real world super-resolution performance, we evaluate both networks on real world datasets, RealSR and DRealSR, both of which contains 100 test images. Both LESRCNN and Omni-SR are trained on the DIV2K [1] dataset.

5. Evaluation

Analysis of Model Complexity: Model complexity includes parameters such as FLOPs, number of model parameters, number of convolutional layers, number of activations in layer outputs, CPU and GPU memory footprint, etc. From table 1. we see that the FLOPs remain around the same for Omni-SR for all three upscaling factors (x2, x3, and x4) at around 135 G. However, LESRCNN scales negatively with an increase in scaling factor, and even with considerably smaller number of parameters when compared with Omni-SR, its FLOPs go beyond Omni-SR on a upscaling factor of x3. There is a similar story in the number of activations and inference time where LESRCNN scales poorly with an increase in scaling factor while Omni-SR, albeit slow, is consistent. The increase in FLOPs could be attributed to the increase in the number of activations and this drives the inference time significantly. In table 3. the results for x2 on DRealSR is missing for LESRCNN. This could also be due to the same limitations of LESRCNN mentioned above. For some images in the dataset, LESR-

[2] https://github.com/facebookresearch/fvcore

Architecture	Scale Factor	PSNR (dB)	SSIM
LESRCNN	x2	31.61	0.882
	x3	28.63	0.807
	x4	27.40	0.767
Omni-SR	x2	31.66	0.884
	x3	28.67	0.809
	x4	27.47	0.770
Bicubic	x2	32.61	0.907
	x3	29.34	0.841
	x4	27.99	0.806
LK-KPN [3]	x2	33.64	0.917
	x3	30.14	0.856
	x4	28.63	0.821
ESRT [12]	x2	-	-
	x3	30.38	0.857
	x4	28.63	0.815

Table 2. Performance on RealSR dataset. LESRCNN is from 2020. Omni-SR (CVPR 23) is the most recent method. RealSR (2019) is the original algorithm published with the dataset. ESRT (CVPR 2022)

Architecture	Scale Factor	PSNR (dB)	SSIM
LESRCNN	x2	-	-
	x3	31.44	0.847
	x4	29.13	0.812
Omni-SR	x2	32.82	0.909
	x3	31.61	0.873
	x4	30.60	0.862
CDC[19]	x2	32.80	0.867
	x3	31.65	0.847
	x4	30.41	0.827

Table 3. Performance on DRealSR dataset. CDC (ECCV 2020) is the original method proposed with DRealSR dataset.

Architecture	Scale factor	Inference Time (GPU) [ms]
LESRCNN	x2	38.66
	x3	70.89
	x4	128.41
MegSR	x2/x3/x4	18.30
Zapdos	x2/x3/x4	18.59
DFCDN	x2/x3/x4	18.71

Table 4. Runtime on GPU

CNN would ask for over 64 GigaBytes of memory and it led to the OS killing the processes. We replicated this behaviour on multiple machine including a local OSX x-86 machine and a cloud Ubuntu 20.04 VM. It should be noted that the images sizes in DRealSR datasets are very big and around 5000x3000. When compared to 256x256 used in the analysis of table 1, we think that LESRCNN performance degrades exponentially with an increase in the input image size.

Architecture	Scale factor	NTIRE	fvcore
LESRCNN	x2	80.18	80.02
	x3	153.33	153.07
	x4	275.07	274.66
Omni-SR	x2	134.89	205.03
	x3	135.45	205.94
	x4	136.25	206.73

Table 5. Difference in the calculations of FLOPs by different methods. NTIRE 2023 method is from the NTIRE challenges at CVPR and fvcore is a Python library from Facebook AI Research. Values are Giga FLOPs.

Table 4 shows the results for GPU inference time. We again observe the trends with LESRCNN as mentioned above. However, we failed to provide the results for Omni-SR as it proved to be very memory intensive and would cross the maximum GPU memory available (16 GigaBytes) in the Colab environment, even for a batch size of 1 input image.

We compare LESRCNN and Omni-SR against the winning entries of the NTIRE 2023 challenge on efficient super-resolution. The winning entries perform consistently for all scaling factors. Even with a larger number of convolutional layer, they beat LESRCNN comfortably on all parameters, and we see no competition with Omni-SR at all. However, setting the performance on Set5, we see that Omni-SR has a PSNR advantage of 0.39 dB, 0.44 dB, and 0.39 dB against MegSR, Zapdos, and DFCDN, respectively. However, the SSIM advantage is very minute.

Performance on RealSR dataset: Both LESRCNN and Omni-SR perform poorly on RealSR dataset. In the table 2, all the results are for models trained on datasets other than RealSR training data. While the original architecture that came along with RealSR dataset, LK-KPN, isn't lightweight by construction, ESRT is a newer method and is a lightweight architecture by design. Similar to Omni-SR, it is also based on attention. However, Omni-SR (2023), even though being a more recent method, fails to outperform ESRT (2022). It should also be noted that both LESRCNN and Omni-SR fail to beat the results of Bicubic interpolation either.

Performance on DRealSR dataset: Here both LESRCNN and Omni-SR perform relatively better across the different scaling factors. Omni-SR even outperforms the original method, CDC, proposed with the DRealSR paper, albeit very marginally.

Complexity Calculation: Table 5 shows the difference between the FLOP count when employing two different methods. While the count remains almost the same for LESRCNN, it shows big dfference in the case of Omni-SR. This could be due to the fact that CNNs are older architec-

ture and therefore, are more standardised than Omni-SR's attention architecture. However, for the sake of comparison, in table 1, we show the results obtained using NTIRE method as it is convention in current research.

6. Conclusions

In this work, we evaluate the efficiency and real-world performance of two models, LESRCNN and Omni-SR. While LESRCNN is an older model (2020), it has seen adoption in many real products. However, we observed its fatal limitation when presented with a large image input. This is a real-world scenario since most cameras these days shoot pictures of larger size of 10s of order of mega-pixels. Omni-SR, however, performs better. However, it is not suitable for real-world application either because of it's very high latency.

The statistics for computing processes (runtime, memory usage) are system and operating system dependent. Therefore, having a standardised testbench is essential but we couldn't find any such testbench. Furthermore, the software tools used for calculating model parameters are also ill-developed with lots of sharp edges. Therefore, we need robust tools for making such calculations.

Besides these observations, we think that we need to conduct more experiments involving a larger list of recent work on lightweight image super-resolution to present further remarks on the usefulness of the models. Future work would first establish a standardised testbench. Lightweight models could also benefit from training on more realistic image dataset such as RealSR and DRealSR to perform better on such datasets.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 5

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

[3] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward real-world single image super-resolution: A new benchmark and a new model. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 3, 6

[4] Haram Choi, Jeongmin Lee, and Jihoon Yang. N-gram in swin transformers for efficient lightweight image super-resolution, 2023. 5

[5] Marcos V Conde, Eduard Zamfir, Radu Timofte, Daniel Motilla, Cen Liu, Zexin Zhang, Yunbo Peng, Yue Lin, Jiaming Guo, Xueyi Zou, et al. Efficient deep models for real-time 4k image super-resolution. ntire 2023 benchmark and report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1495–1521, 2023. 5

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2015. 1, 5

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

[8] Jianbo Jiao, Wei-Chih Tu, Shengfeng He, and Rynson W. H. Lau. Formresnet: Formatted residual learning for image restoration. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1034–1042, 2017. 1

[9] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution, 2016. 5

[10] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution, 2020. 5

[11] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004. 3

[12] Zhisheng Lu, Juncheng Li, Hong Liu, Chaoyan Huang, Linlin Zhang, and Tieyong Zeng. Transformer for single image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 457–466, 2022. 5, 6

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

[14] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76:21811–21838, 2017. 5

[15] Chunwei Tian, Ruibin Zhuge, Zhihao Wu, Yong Xu, Wangmeng Zuo, Chen Chen, and Chia-Wen Lin. Lightweight image super-resolution with enhanced cnn. *Knowledge-Based Systems*, 205:106235, 2020. 2, 3

[16] Hang Wang, Xuanhong Chen, Bingbing Ni, Yutian Liu, and Jinfan Liu. Omni aggregation networks for lightweight image super-resolution, 2023. 2, 3

[17] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018. 1

[18] Yucong Wang and Minjie Cai. A single residual network with esa modules and distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1970–1980, 2023. 5

[19] Pengxu Wei, Ziwei Xie, Hannan Lu, Zongyuan Zhan, Qixiang Ye, Wangmeng Zuo, and Liang Lin. Component divide-and-conquer for real-world image super-resolution, 2020. 3, 6

[20] Kailu Wu, Chung-Kuei Lee, and Kaisheng Ma. MemSR: Training memory-efficient lightweight model for image super-resolution. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 24076–24092. PMLR, 17–23 Jul 2022. 5

[21] Lei Yu, Xinpeng Li, Youwei Li, Ting Jiang, Qi Wu, Haoqiang Fan, and Shuaicheng Liu. Dipnet: Efficiency distillation and iterative pruning for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1692–1701, 2023. 5

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

[23] Yulun Zhang, Kai Zhang, Zheng Chen, Yawei Li, Radu Timofte, Junpei Zhang, Kexin Zhang, Rui Peng, Yanbiao Ma, Licheng Jia, Huaibo Huang, Xiaoqiang Zhou, Yuang Ai, Ran He, Yajun Qiu, Qiang Zhu, Pengfei Li, Qianhui Li, Shuyuan Zhu, Dafeng Zhang, Jia Li, Fan Wang, Chunmiao Li, TaeHyung Kim, Jungkeong Kil, Eon Kim, Yeonseung Yu, Beomyeol Lee, Subin Lee, Seokjae Lim, Somi Chae, Heungjun Choi, ZhiKai Huang, YiChung Chen, YuanChun Chiang, HaoHsiang Yang, WeiTing Chen, HuaEn Chang, I-Hsiang Chen, ChiaHsuan Hsieh, SyYen Kuo, Ui-Jin Choi, Marcos V. Conde, Sunder Ali Khowaja, Jiseok Yoon, Ik Hyun Lee, Garas Gendy, Nabil Sabor, Jingchao Hou, Guanghui He, Zhao Zhang, Baiang Li, Huan Zheng, Suiyi Zhao, Yangcheng Gao, Yanyan Wei, Jiahuan Ren, Jiayu Wei, Yanfeng Li, Jia Sun, Zhanyi Cheng, Zhiyuan Li, Xu Yao, Xinyi Wang, Danxu Li, Xuan Cui, Jun Cao, Cheng Li, Jianbin Zheng, Anjali Sarvaiya, Kalpesh Prajapati, Ratnadeep Patra, Pragnesh Barik, Chaitanya Rathod, Kishor Upla, Kiran Raja, Raghavendra Ramachandra, and Christoph Busch. Ntire 2023 challenge on image super-resolution (x4): Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1865–1884, June 2023. 5

Appendices

A. Layerwise Model Complexity for Omni-SR

module	#parameters or shape	#flops
model	2.854M	0.207T
residual_layer	2.788M	0.202T
residual_layer.0	0.558M	40.479G
residual_layer.0.residual_layer	0.551M	40.282G
residual_layer.0.esa	7.04K	0.196G
residual_layer.1	0.558M	40.479G
residual_layer.1.residual_layer	0.551M	40.282G
residual_layer.1.esa	7.04K	0.196G
residual_layer.2	0.558M	40.479G
residual_layer.2.residual_layer	0.551M	40.282G
residual_layer.2.esa	7.04K	0.196G
residual_layer.3	0.558M	40.479G
residual_layer.3.residual_layer	0.551M	40.282G
residual_layer.3.esa	7.04K	0.196G
residual_layer.4	0.558M	40.479G
residual_layer.4.residual_layer	0.551M	40.282G
residual_layer.4.esa	7.04K	0.196G
input	1.792K	0.113G
input.weight	(64, 3, 3, 3)	
input.bias	(64,)	
output	36.928K	2.416G
output.weight	(64, 64, 3, 3)	
output.bias	(64,)	
up.0	27.696K	1.812G
up.0.weight	(48, 64, 3, 3)	
up.0.bias	(48,)	

B. Layerwise Model Complexity for LESRCNN

module	#parameters or shape	#flops
model	0.774M	0.275T
sub_mean.shifter	12	0.59M
sub_mean.shifter.weight	(3, 3, 1, 1)	
sub_mean.shifter.bias	(3,)	
add_mean.shifter	12	9.437M
add_mean.shifter.weight	(3, 3, 1, 1)	
add_mean.shifter.bias	(3,)	
conv1.0	1.728K	0.113G
conv1.0.weight	(64, 3, 3, 3)	
conv2.0	36.864K	2.416G
conv2.0.weight	(64, 64, 3, 3)	
conv3.0	4.096K	0.268G
conv3.0.weight	(64, 64, 1, 1)	
conv4.0	36.864K	2.416G
conv4.0.weight	(64, 64, 3, 3)	
conv5.0	4.096K	0.268G
conv5.0.weight	(64, 64, 1, 1)	
conv6.0	36.864K	2.416G
conv6.0.weight	(64, 64, 3, 3)	
conv7.0	4.096K	0.268G
conv7.0.weight	(64, 64, 1, 1)	
conv8.0	36.864K	2.416G
conv8.0.weight	(64, 64, 3, 3)	
conv9.0	4.096K	0.268G
conv9.0.weight	(64, 64, 1, 1)	
conv10.0	36.864K	2.416G
conv10.0.weight	(64, 64, 3, 3)	
conv11.0	4.096K	0.268G
conv11.0.weight	(64, 64, 1, 1)	
conv12.0	36.864K	2.416G
conv12.0.weight	(64, 64, 3, 3)	
conv13.0	4.096K	0.268G
conv13.0.weight	(64, 64, 1, 1)	
conv14.0	36.864K	2.416G
conv14.0.weight	(64, 64, 3, 3)	
conv15.0	4.096K	0.268G
conv15.0.weight	(64, 64, 1, 1)	
conv16.0	36.864K	2.416G
conv16.0.weight	(64, 64, 3, 3)	
conv17.0	4.096K	0.268G
conv17.0.weight	(64, 64, 1, 1)	
conv17_1.0	36.864K	38.655G
conv17_1.0.weight	(64, 64, 3, 3)	
conv17_2.0	36.864K	38.655G
conv17_2.0.weight	(64, 64, 3, 3)	
conv17_3.0	36.864K	38.655G
conv17_3.0.weight	(64, 64, 3, 3)	
conv17_4.0	36.864K	38.655G
conv17_4.0.weight	(64, 64, 3, 3)	
conv18.0	1.728K	1.812G
conv18.0.weight	(3, 64, 3, 3)	
upsample.up.body	0.295M	96.637G
upsample.up.body.0	0.148M	19.327G
upsample.up.body.0.weight	(256, 64, 3, 3)	
upsample.up.body.0.bias	(256,)	
upsample.up.body.2	0.148M	77.309G
upsample.up.body.2.weight	(256, 64, 3, 3)	
upsample.up.body.2.bias	(256,)	