# MESRGAN: Investigating a Lightweight Super-Resolution Alternative to (E)SRGAN

Jacob Eriksson
jaceri@student.chalmers.se

Sebastian Löf
slof@student.chalmers.se

***Abstract*—While super-resolution research has been largely focused on maximizing performance on various evaluation tests, we investigate whether it is possible to reduce computational cost while still maintaining the quality of similar architectures. In this report we evaluate SRGAN, ESRGAN and our own model: mobile ESRGAN, a variation of ESRGAN that incorporates residual bottleneck layers from MobilenetV2. This model roughly halves the theoretical computational cost of ESRGAN while maintaining its overall architectural design. Our model MESRGAN showed similar results to ESRGAN, both perceptually and with the evaluation methods used. However, due to limited low-level software support, the residual bottleneck layers actually resulted in slower performance, both in forward and backward passes.**

## I. Introduction

In conjunction with the present development and popularization of high-resolution displays, there grows a greater demand for high-resolution images that can fully utilize these displays. Consequently, an interest has grown for solving the problem of upscaling a given low-resolution (LR) image $\mathbf{I}_{\text{LR}}$ to its high-resolution (HR) $\mathbf{I}_{\text{HR}}$ counterpart. This problem is formally known as the single image super-resolution (SISR) problem, but is commonly abbreviated as super-resolution (SR).

A recent advancement in this field is the introduction of deep neural networks (DNN), and in particular deep convolutional neural networks (CNN), to solve the SISR problem. Two recently proposed solutions of this kind are super-resolution generative adversarial network (SRGAN) [1] and enhanced SRGAN (ESRGAN) [2], which both demonstrated state-of-the-art results when initially published.

In contrast to the development of heavy DNNs, there is also a desire to reduce training time, in order to speed up parameter tuning, and to increase runtime performance in order to make the technology suitable for mobile devices. One particular line of such effort is the MobileNet series of networks [3] [4] [5].

In this paper we investigate how core ideas from MobileNetV2 [4] can be integrated into the ESRGAN architecture in order to attain these desired performance properties. Our research question is thus whether these architecture modifications can be made without incurring any significant detriment to the visual quality of the generated SR images. Our main contributions in this work are:

- A thorough evaluation of performance and visual quality impacts of introducing the *bottleneck residual block* of the MobileNetV2 architecture into ESRGAN.
- PyTorch implementations of SRGAN, ESRGAN, and our proposed model: mobile ESRGAN (MESRGAN).

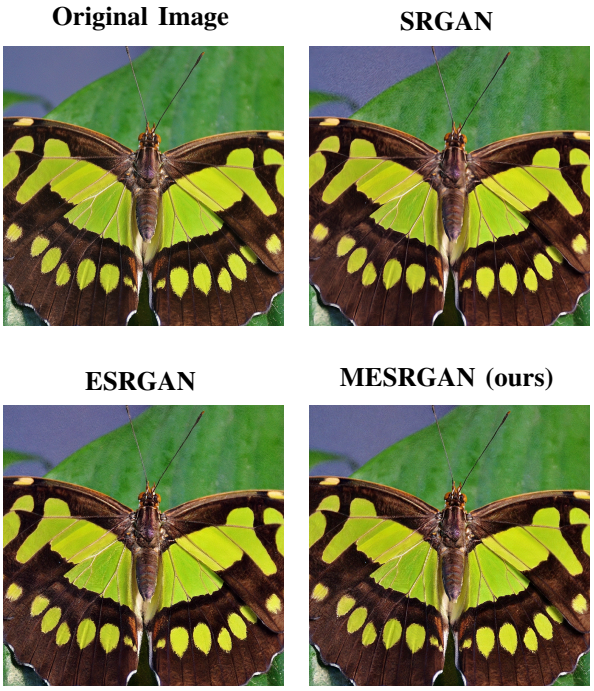



Fig. 1: Comparison of 4x upscaling between SRGAN, ESRGAN, and MESRGAN on a cropped image from DIV2K [6].

## II. Related Work

The current state-of-the-art works in SR all build upon the pioneering work done by Dong et al. [7] [8], who proposed the use of deep CNNs to solve SISR. Another highly influential work is the generative adversarial network (GAN) architecture proposed by I. Goodfellow et al. [9], which has been incorporated to SISR to improve training loss by discriminating generated SR images from ground truth HR images. Besides and prior to DNNs there also exist plenty of deterministic algorithms e.g. nearest-neighbor, Fourier-transforms, and bicubic interpolation, which are all still frequently used in the industry and as research baselines due to their consistency in performance and quality.

Traditionally, a large portion of research [10] has focused on optimizing visual quality with respect to the peak signal-to-noise ratio (PSNR) metric and the structural similarity index measure (SSIM). These metrics measure pixel reconstruction accuracy, but typically lead to overly blurred images when

solely optimized for [1] [2]. Consequently, there has been a recent strive toward metrics that better represent the way humans perceive image quality [1] [2] [11], e.g. perceptual index (PI) [11] and mean opinion score (MOS).

The (E)SRGAN networks have focused especially on perceptual quality, resulting in SR images optimized for human consumption. This policy, combined with the interesting technology involved with the (E)SRGAN networks, form the main reasons why we chose to investigate these networks in particular. MobileNetV2 was partially chosen with the hope of better adapting these heavy GANs to our modest hardware.

## III. Method

Our project method is structured into 4 parts:

1) Understanding and implementing SRGAN.
2) Extending the SRGAN implementation into ESRGAN.
3) Modifying ESRGAN into MESRGAN by introducing the *bottleneck residual block* from MobileNetV2.
4) Training all models and evaluating them in comparison.

### A. SRGAN

SRGAN [1] is composed of two components: a SISR generator $G_{\theta_G}$ and a SISR discriminator $D_{\theta_D}$, both of which are deep CNNs. $G_{\theta_G}$ outputs an SR image of the input LR, while $D_{\theta_D}$ outputs the probability of the input image being real (as opposed to generated/fake) and is only used during training. $G_{\theta_G}$ follows a 16-block-deep residual network architecture [12] [13] called SRResNet [1] and its training is split into 2 stages. For both stages, the training data is produced by taking a set of unlabeled HR images, applying multiple random $96 \times 96$ crops to them, and then downscaling the resulting patches by a factor of 4 (using bicubic interpolation) to obtain corresponding $24 \times 24$ LR images.

In the first training stage, the generator is optimized with respect to the pixel-wise MSE loss $\mathcal{L}_{\text{MSE}}$ until convergence:

$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{MSE}}(G_{\theta_G}(\mathbf{I}_{\text{LR}}^{(i)}), \mathbf{I}_{\text{HR}}^{(i)}) \tag{1}$$

This is similar to prior work, where PSNR and SSIM are optimized for. It also provides a good basis for the GAN phase.

In the second training stage, $G_{\theta_G}$ and $D_{\theta_D}$ are optimized in alternation as per the min-max problem formulation in [9]. The full generator loss is:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + 10^{-3}\mathcal{L}_{\text{Adv}} + 6 \cdot 10^{-3}\mathcal{L}_{\text{VGG}} \tag{2}$$

where $\mathcal{L}_{\text{Adv}}$ is the adversarial loss from the discriminator and $\mathcal{L}_{\text{VGG}}$ is the Euclidean norm between the features of a particular convolutional layer in a VGG19 classifier [14] that has been pre-trained on ImageNet. The intuition is that $\mathcal{L}_{\text{MSE}}$ maintains pixel-wise accuracy, $\mathcal{L}_{\text{Adv}}$ encourages plausible images with high-frequency details, and $\mathcal{L}_{\text{VGG}}$ helps ensure that object information (i.e. classification features) remain intact.

### B. ESRGAN

ESRGAN largely follows the same setup and training process as SRGAN, but with a few key differences that are suggested to improve it in various aspects.

*1) Generator:* The first and most notable difference is the modified generator architecture. They replaced what they refer to as the "basic block" of SRGAN, with a novel *Residual in Residual Dense Block* (RRDB), which in turn is made of three *Dense Blocks* (DB). Each DB is composed of five convolutional layers that employ dense connections [15] between every layer. This architectural modification was argued to increase the model's capacity while simultaneously making it easier to train [2].

Further, the ESRGAN authors removed the batch normalization layers of the SRResNet architecture. Empirically, they showed that these layers can introduce unpleasant artifacts that disrupt training stability during the second training stage[1] [2].

Lastly, the ESRGAN architecture also utilized residual scaling to further mitigate instability during training.

*2) Discriminator:* The second key difference is that the standard GAN framework [9] was replaced in favor of Relativistic average GAN (RaGAN) [16].

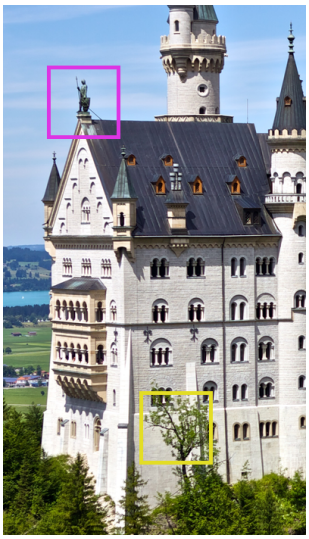
In this framework, the discriminator $D_{\theta_D}$ is trained to determine the probability whether an SR image is more realistic than its HR counterpart, rather than discriminating between whether its real or fake. This results in an adversarial loss where HR and SR images always contribute equally to the gradient, effectively stabilizing early training [16]. It is also argued that RaGAN helps $G_{\theta_G}$ produce sharper edges [2], but there is no reliable evidence for this observation.

### C. MESRGAN

MESRGAN uses the same setup as ESRGAN, but replaces every convolutional layer in each *Dense Block* with a *bottleneck residual block* (BRB) from MobileNetV2 [4], using an expansion factor of $t = 2$. The BRB consists of 3 layers, surrounded by a residual connection. The first layer is a pointwise (i.e. $1 \times 1$) convolutional layer that expands the input depth by the expansion factor $t$. The second layer performs a depth-wise convolution (i.e. one filter per input channel), and is then followed by the third layer which, in our case, shrinks the depth back by a factor of $t$.

The purpose of introducing BRB is to reduce the number of multiplications that normally has to be performed for each convolutional layer, which should result in faster training and runtime execution. For instance, considering the $128 \times 128$ patches that ESRGAN take as input, the total number of multiplications per *Dense Block* convolutional layer in ESRGAN is $h \cdot w \cdot d_{\text{in}} \cdot d_{\text{out}} \cdot k^2 = 128 \cdot 128 \cdot 64 \cdot 64 \cdot 3^2 \approx 6.04 \cdot 10^8$. In contrast, this number becomes $h \cdot w \cdot d_{\text{in}} \cdot t(d_{\text{in}} + k^2 + \cdot d_{\text{out}}) = 128 \cdot 128 \cdot 64 \cdot 2(64 + 3^2 + 64) \approx 2.83 \cdot 10^8$ (effectively half) when BRBs are used [4], as in MESRGAN. The hope is that this modification causes a boost in performance without significantly hurting the visual quality of the SR images.

[1] We also noticed these artifacts during the development and training of our SRGAN implementation.



A crop of image 887 of the DIV2K dataset. This image was a part of our test dataset DIV2K-90.


Fig. 2: Visual comparison between the SR images produced by each network after being fully trained. (PSNR, SSIM)

TABLE I: Forward pass runtime benchmark of all models, each measurement is an average of 100 runs. Everything was run on a NVIDIA GeForce GTX 1070 Ti GPU.

| | Time (ms) | | |
|---|---|---|---|
| Batch size | SRGAN | ESRGAN | MESRGAN |
| 1 | 26 | 33 | 155 |
| 2 | 50 | 62 | 299 |
| 4 | 134 | 126 | 596 |
| 8 | 279 | 266 | 1204 |
| 16 | 561 | 534 | 2402 |

TABLE II: The final models evaluated on DIV2K-90 and BSD100 with respect to PSNR, SSIM, PI, and RMSE.

| **DIV2K-90** | SRGAN | ESRGAN | MESRGAN |
|---|---|---|---|
| PSNR | **25.40** | 24.52 | 24.39 |
| SSIM | **0.96** | 0.95 | 0.95 |
| PI | 3.57 | **3.11** | 3.13 |
| RMSE | **13.61** | 15.82 | 15.86 |
| **BSD100** | | | |
| PSNR | 24.04 | **24.68** | 24.58 |
| SSIM | 0.92 | **0.93** | **0.93** |
| PI | **5.64** | 5.77 | 5.71 |
| RMSE | 15.38 | **14.52** | 14.56 |

## IV. Experiments

### A. Experimantal Details

All networks were trained in two stages using Adam [17]. During the first stage the networks were trained for 200k iterations using only MSE-loss for SRGAN, and L1-loss for ESRGAN and MESRGAN, as specified in [2]. The learning rate was initialized to $2 \times 10^{-4}$ and then halved after 100k iterations. For the second stage we trained for 200k iterations using the loss function (2). For ESRGAN and MESRGAN we used the relative average discriminator as described in section III-B. During the second stage the learning rate was initialized to $10^{-4}$ and then halved every 50k iterations. The remaining hyperparameters were chosen identically to that of ESRGAN [2]. All networks were trained using a batch size of 16. All models were implemented in PyTorch and trained using a Nvidia GTX 1070 Ti.

### B. Data

Following the initiative of ESRGAN [2], all networks were trained on a fusion of the DIV2K [6], Flickr2K [18], and OutdoorSceneTrain [19] datasets[2]. The DIV2k contains 800 training images, 80 of which were moved to a separate test set together with 10 images from the DIV2K validation dataset. This test set will be referred to as DIV2K-90 throughout this report. Flickr2K contained 2650 training images and OutdoorSceneTrain totaled to 10237 images. This fusion amounted to a total of 13607 training images.

### C. Architectural Changes

Due to the VRAM limitations of the Nvidia GTX 1070 Ti card, our implementation of ESRGAN had to be adjusted accordingly. The original paper [2] suggests the use of three DBs per RRDB, but we used one DB per RRDB. This variation was also used in MESRGAN, but with BRBs for the convolutional layers as described in section III-C.

[2]We omitted 87 images from OutdoorSceneTrain because these images had a dimension smaller than 128, which is not compatible with (M)ESRGAN.

### D. Results and Discussion

Fig. 1 and Fig. 2 compare the visual quality of all trained models: SRGAN, ESRGAN, and MESRGAN. It is clear that both ESRGAN and MESRGAN are more perceptually pleasant than SRGAN. Yet, SRGAN obtains better performance in both PSNR and SSIM. This metric limitation is a subject that has been discussed extensively in the super-resolution community [2] [20], and even changed the choice of evaluation tests in certain competitions [11]. Other methods such as MOS, where human participants rate the performance of the models, have also been used [1], but this is a slow and manual process. Thus, for better automatic evaluation, we chose to include PI scoring, based on the work of Y. Blau et al. [20].

In TABLE II we evaluate all models on both DIV2K-90 and BSD100. As expected, all models perform similarly, but SRGAN surprisingly achieved the best PI score on the BSD100 test set. This does not agree with our perceived quality of the images in Fig. 1 and Fig. 2, which contradicts the purpose of PI. However, for the DIV2K-90 test set, we observe expected results even for PI.

Our theory is that, while removal of batch normalization layers mitigated visual artifacts and stabilized training, it also made the ESRGAN and MESRGAN networks more susceptible to covariate shift. That is, while DIV2K-90 belongs to the same data distribution as the DIV2K training data (a subset of our training dataset), BSD100 seemingly follows a significantly different distribution. We also noticed a very slight color degradation in ESRGAN and MESRGAN that was not present in SRGAN. This could also lead to a reduced PI score, but it would not explain the difference in results between DIV2K-90 and BSD100 as observed in TABLE II.

In general, the performance metrics between ESRGAN and MESRGAN appears to be close to negligible. The difference is still visually noticeable, as seen in Fig. 2, but it does not appear to be necessarily worse. This shows that using BRB layers did not significantly degrade the network's output quality, while still theoretically using almost half of the calculations.

Regarding the computational performance of the networks, our theory did not prove to hold in practice, as can be seen in TABLE I. The average forward[3] pass had a significantly longer runtime on MESRGAN, compared to ESRGAN and SRGAN. After some research, we found that depth-wise convolution is significantly slower in PyTorch than standard convolution due to limited CUDA support [21]. The problem seems to have been mitigated in cuDNN 8.1.0 [22], but PyTorch does not currently support this version. TensorFlow supposedly does not have this performance issue [21], but we have not had the time to verify this ourselves.

## V. Conclusion

We have investigated the (E)SRGAN architectures and proposed an alternative called MESRGAN that is theoretically more computationally efficient. In practice, MESRGAN performs similar to ESRGAN, both perceptually and in the test evaluation methods used. Unfortunately, MESRGAN did not improve on ESRGAN in terms of computational efficiency according to our results, but it is likely that it might do so in future versions of PyTorch when further optimizations have been implemented.

In future work, it would be of interest to experiment with improving the low-level support for depth-wise convolutions and to investigate learned downscaling as outlined in [23].

[3] We noticed similar results with backward passes.

## References


[1] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," 2017.

[2] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, "Esrgan: Enhanced super-resolution generative adversarial networks," 2018.

[3] 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," 2017.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019.

[5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019.

[6] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1122–1131.

[7] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," 2014.

[8] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," 2015.

[9] Goodfellow et al., "Generative adversarial networks," 2014.

[10] S. M. A. Bashir, Y. Wang, M. Khan, and Y. Niu, "A comprehensive review of deep learning-based single image super-resolution," *PeerJ Computer Science*, vol. 7, p. e621, Jul 2021. [Online]. Available: http://dx.doi.org/10.7717/peerj-cs.621

[11] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, "The 2018 pirm challenge on perceptual image super-resolution," 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[15] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[16] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," 2018.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[18] Timofte et al., "Ntire 2017 challenge on single image super-resolution: Methods and results," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1110–1121.

[19] X. Wang, K. Yu, C. Dong, and C. C. Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," 2018.

[20] Y. Blau and T. Michaeli, "The perception-distortion tradeoff," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00652

[21] PyTorch contributors, "Fp32 depthwise convolution is slow in gpu," 2021. [Online]. Available: https://github.com/pytorch/pytorch/issues/18631

[22] Nvidia Corporation, "cudnn release 8.1.0," 2021. [Online]. Available: https://docs.nvidia.com/deeplearning/cudnn/release-notes/rel_8.html#rel-810

[23] W. Sun and Z. Chen, "Learned image downscaling for upscaling using content adaptive resampler," *IEEE Transactions on Image Processing*, vol. 29, p. 4027–4040, 2020. [Online]. Available: http://dx.doi.org/10.1109/TIP.2020.2970248