Using Generative Adversarial Networks for Single Image Super-Resolution

Marwan Farag[1(✉)] and Friedhelm Schwenker[2]

[1] German University in Cairo, New Cairo, Egypt
marwan.farag@student.guc.edu.eg
[2] Institute of Neural Information Processing, Ulm University, Ulm, Germany
friedhelm.schwenker@uni-ulm.de

Abstract. In digital image processing, retrieving a high-resolution image from its low-resolution version is considered to be a major topic. Super-resolution *SR* is a problem that has direct applications in numerous disciplines like medical diagnosis, satellite imagery, face recognition and surveillance. The choice of the optimization function has been a major factor in previous super-resolution approaches. Optimizing metrics that are determined based primarily on pixel-level variance is the most common objective for supervised super-resolution algorithms. Nevertheless, these approaches do not output perceptually satisfactory images. This paper adopts an idea in which depending on only pixel-space similarity is avoided. Instead, the major goal is to utilize a content loss based on perceptual resemblance using feature maps of the VGG network in conjunction with Generative Adversarial Networks *GAN*. This depends on training two networks: a generator and a discriminator. In an adversarial game, they compete to outperform each other with an ultimate objective of producing super-resolution images that are identical to the real high-resolution images that already exist in the dataset. This paper's main contribution is a comparison of the effects of taking the VGG-19 content loss from various layers. On public benchmarks, super-resolution *GAN* was successful in recovering detailed textures from highly downsampled images. *SRGAN* reveals large gains in perceptual quality in a mean opinion score *MOS* test.

Keywords: super-resolution · *SRGAN* · content loss · perceptual loss · *HR* · *LR* · *SRResNet*

1 Introduction

Single Image Super Resolution, also known as *SISR*, is a technique that is focused on enhancing the clarity of a poor quality image. In general, the expression *super-resolution* refers to extracting knowledge from an existing low-resolution signal and use it to reach a high-resolution signal. Depending on the application, the relationship between the high-resolution *HR* image and the low-resolution *LR* version may differ. This paper presents addresses case that the *LR* image is a

T. Girma Debelee et al. (Eds.): PanAfriCon AI 2022, CCIS 1800, pp. 154–168, 2023.
https://doi.org/10.1007/978-3-031-31327-1_9

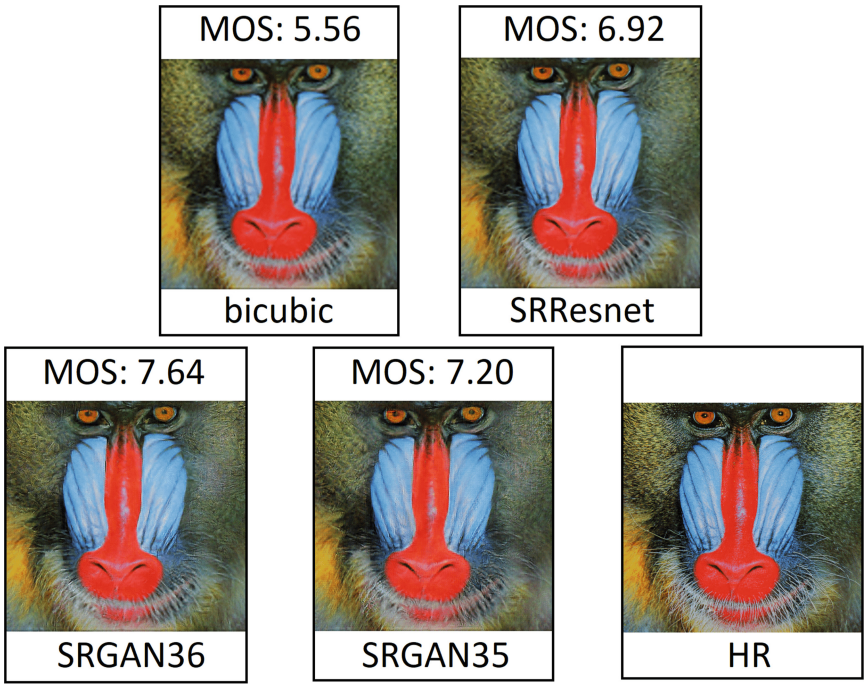

Fig. 1. Mean Opinion Scores *MOS* for *Set14* [16] using bicubic interpolation, *SRResNet*, *SRGAN36* (VGG loss taken before 36^{th} layer) and *SRGAN35* (VGG loss taken before 35^{th} layer) in comparison to the ground-truth *HR* image [×4 upscaling].

bicubic downscaled counterpart of the corresponding *HR* image is adopted. The method used to extract and use the data from the *LR* image influences how well the image is recreated. Due to the fact that a large number of *HR* photos can be downscaled into a single *LR* image, single image super resolution continues to be an unsolved and demanding challenge in the field of computer vision.

Super-resolution is involved in many applications. First, it can be used in surveillance systems for a better face recognition in the images obtained from surveillance cameras [4]. Second, it is beneficial for diagnostic imaging, particularly for magnetic resonance imaging *MRI* [11]. By reducing cost of scan time, spatial coverage, and signal-to-noise ratio, it becomes more convenient to use *SR* techniques to output super-resolved *MRI* scans by processing their corresponding low-resolution ones. Third, data transmission and storage can be a relevant application [9], as one may send a low-resolution signal, and upscale it on the fly, rather than sending the high-resolution one, reducing cost. Finally, using super-resolution satellite imagery can help in finding and determining the number of elephants in African environments [3].

1.1 Related Work

A very early solution was to interpolate the values of the missing pixels. This typically results in solutions with excessively smooth textures. Dong et al. [2] proposed the first preprocessing interpolation method that consisted of three layers: feature extraction layer, then feature mapping to high-dimensional feature vectors by using 1×1 convolutional filters that add some non-linearity, then a final reconstruction layer that constructs the final target high-resolution images. However, because this super-resolution convolutional network *SRCNN* is shallow and convolution kernels are small, image fine details are not obtained and the network is limited to a single scale. An improvement over the *SRCNN* was utilizing the very-deep-super-resolution *VDSR* [6]. It used a much deeper network in which a reduced size of convolutional kernels, higher learning rate and gradient clipping were used. This speeded up convergence and improved training stability. Nevertheless, both *SRCNN* [2] and *VDSR* [6] were pre-upsampling techniques that accomplished feature extraction in the high-resolution space. *FSRCNN*, on the other hand, does not use an interpolation method at the beginning but does feature extraction in the low-resolution space [11]. It uses multiple convolutional layers with reduced kernel size which in turn reduces the count of learnable weights. Deconvolutional filtering is used in the final step of upsampling.

Sub-pixel convolution, rather than a deconvolutional layer for upsampling, was first suggested by another proposed technology, the efficient sub-pixel convolutional neural network, or *ESPCN* [12]. Recursively connected units are used by the deeply-recursive convolutional network *DRCN* [7] to make the convolutional layer much deeper and enhance the fine details of the recounstructed image. The training difficulty of the deep network's parameters can be mitigated through weight sharing while also improving the model's capacity for generalization.

The main objective of *SR* optimization techniques is to reduce the mean squared error *MSE* between retrieved *HR* images and dataset original images. This is beneficial because it optimizes the peak signal-to-noise ratio, or *PSNR*, a typical measure for assessing *SR* approaches. There are severe limitations on *PSNR*'s capacity to detect perceptual dissimilarity because this metric is calculated depending on pixel-level numeric differences. The lowest *PSNR* may not always indicate a perceptually enhanced super-resolved image outcome [8].

Generative Adversarial Networks. As described by Goodfellow et al. who were the first to propose the concept of *GAN* [5], the final aim of *GAN* is to produce data that has the same distribution as an input dataset. The idea of *GAN* is based on a rivalry between these generator and discriminator networks which play an adversarial game to beat each other. In this case, the generator's target is to produce fake output that looks very similar to the original dataset examples, trying to fool the discriminator and make it unable to recognize the difference between the real and generated data. On the other side, the discriminator is to be trained to become a reliable judge that can accurately detect the fake output from the generator and tell them apart from the real data. This

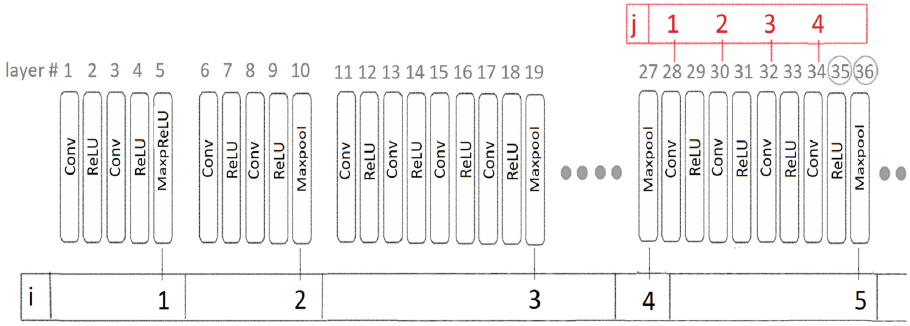

Fig. 2. These are the layers of the VGG19 architecture [13]. In [8], they said that taking an *MSE* loss between the feature maps at $i = 5$ and $j = 4$ after the ReLU activation gives the best results. In [15], however, they took the feature maps before the ReLU activation.

adversarial game will improve both the generator and the discriminator, resulting in the final goal which is to make the generator a well-founded one that can generate accurate images or any kind of data. The target function of *GAN* is:

$$V(\theta^{(D)}, \theta^{(G)}) = E_{x \sim p_{data}(x)} \log(D(x)) + E_{z \sim p_z(z)} \log(1 - D(G(z)) \quad (1)$$

in which the probability that an input example, x, is real or not is represented by (D(x)), while the generated samples are represented by (G(z)). [5].

The value function $V(\theta^{(D)}, \theta^{(G)})$ can be viewed as a payoff: the aim is to maximize its value with regard to the discriminator (D), while reducing its value with respect to the generator (G), that is, $\min_G \max_D (V(\theta^{(D)}, \theta^{(G)}))$.

GAN was used to solve the problem of *SISR* for the first time by Ledig et al. [8] in 2017. Their proposed solution made use of residual blocks to enhance the output. Wang et al. [15] improved the work of [8] by employing the fundamental unit known as the Residual-in-Residual Dense Block. In addition, they removed batch-normalization and altered the content loss by changing the VGG layer from which they compared the outputs.

Another method for *SR* reconstruction [11] has been presented using self-attention *GAN* or *SRAGAN*. To enhance the fine details of the recovered image, the generator network assigns higher weights and makes use of the attention mechanism model to create a more complex architecture. Another recently proposed image reconstruction technique called SwinIR [10] makes use of the well-known Swin Transformer network. Their architecture consists of layers for shallow feature extraction, deep feature extraction, and high-quality image reconstruction.

1.2 Contribution

To not rely on the *MSE* loss function, *GAN*s for image super-resolution *SRGAN* [8] have been used in this paper. A perceptual loss function that consists of an adversarial loss and a content loss is used to achieve this. The adversarial loss pushes the output images to look more like the original high-resolution images using a discriminator network. The discriminator has been trained to tell apart between generated super-resolved images and existing ground-truth in the dataset.

As presented in Fig. 2, an enhancement that [15] has done over the original *SRGAN* paper was changing the VGG layer from which the content loss is taken. Nevertheless, this was one of many changes thay have made over *GAN*'s implementation [8]. Evaluating the results of changing the content VGG loss alone has not been done. For this reason, this paper's main contribution is to evaluate the results of taking the content loss of the VGG19 network from different layers. It was discovered that taking the VGG-loss at the 4^{th} convolution (after activation) before the 5^{th} maxpooling layer in the VGG19 network yielded more perceptually fulfilling results compared to taking it before the activation. Figure 1 shows a sample of the *MOS* test result to show a comparison between taking the VGG loss from different layers. On public benchmarks, Super-Resolution *GAN* was able to restore fine texture details from $\times 4$ downscaled images. *SRGAN* reveals large gains in perceptual quality in a mean opinion score *MOS* test as shown in Fig. 1.

2 Methods

2.1 Generator and Discriminator Networks

This paper is adopting the same generator and discriminator structures described by Ledig et al. [8], who were the first to employ a perceptual-based objective function for a real-looking *SISR* output using the notion of *GAN*.

Generator Structure. As shown in Fig. 3, the generator network G is composed of identically designed N residual blocks. In the experiments of this paper, number of residual blocks $N = 16$ was used. Two layers of convolution are employed with small 3×3 filters and a count of 64. Since deep structures like these were found to be challenging to train, batch-normalization is employed to mitigate the internal co-variate shift in order to train these deeper network structures quickly. After that, parametric ReLU is used as the activation function. Finally, the input image's resolution has been increased using two layers of sub-pixel convolution.

Discriminator Structure. As previously mentioned, the discriminator is used as a judge to tell apart between produced images from high-resolution images in the dataset. LeakyReLU is used as an activation function with $\alpha = 0.2$ while

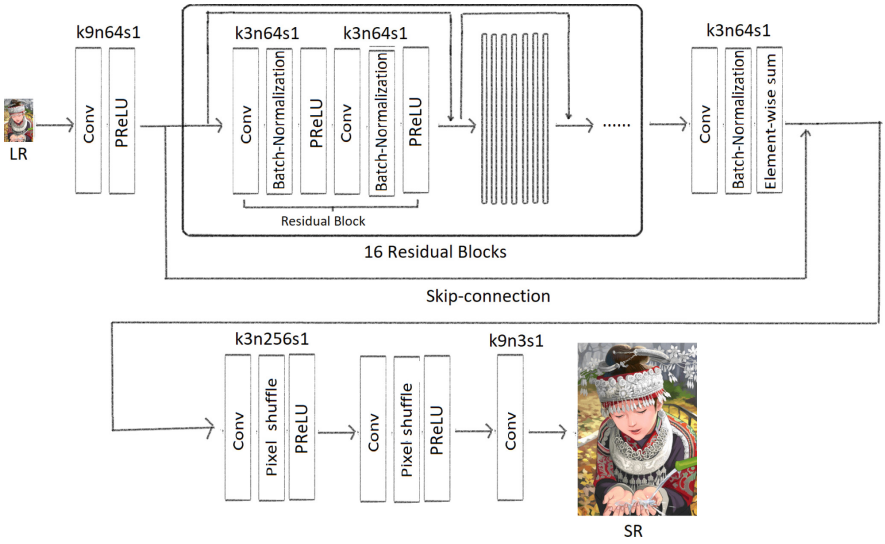

Fig. 3. The architecture of the generator is based on [8], with k denoting the filter size, n denoting representing the count of feature maps, and s denoting the stride for each layer. The generator uses 16 residual blocks, then sub-pixel convolution has been used to increase the image's resolution.

max-pooling is not utilized. The network architecture is composed of eight convolutional layers of a kernel size 3×3. The count of kernels in each layer are (64, 64, 128, 128, 256, 256, 512, 512) respectively. At the final stages, two dense layers and a sigmoid activation functions are employed that output a probability that represents classifying an input image as real or generated. The architecture layers details are depicted in Fig. 4.

2.2 Loss Function

Following the loss function that was suggested by Goodfellow et al. [5], to resolve the adversarial optimization problem that was mentioned in Eq. 1, the generator and discriminator networks are alternately optimized. This formulation enables the training of a generator to mislead a discriminator taught to differentiate between generated images and real ones. The final objective function to determine the perceptual quality of generated images is a summation of the content loss and the adversarial loss, each multiplied by some hyperparameter. This objective function can be represented as follows:

$$l^{SR} = l_X^{SR} + 10^{-3} l_{Gen}^{SR} \tag{2}$$

where l_X^{SR} represents the content loss and l_{Gen}^{SR} represents the adversarial loss and they both add up to the perceptual loss l^{SR} [8]. For the content loss,

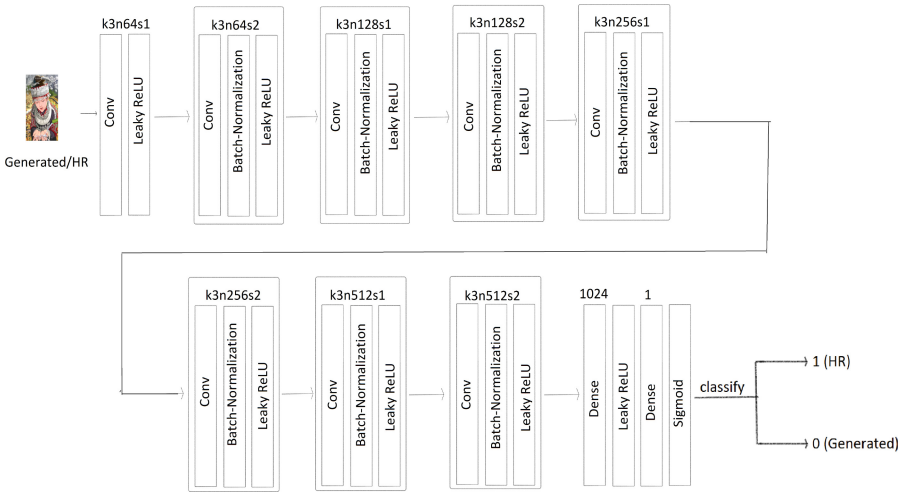

Fig. 4. The Discriminator network adopted from [8] is composed of eight convolutional filters proceeded by dense layers and a final sigomid activation layer to output probability of input being real.

the mean-squared-error *MSE* is the one that is conventionally used. The *MSE* loss in pixels is calculated as follows:

$$l^{SR}_{MSE} = \frac{1}{r^2WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I^{HR}_{x,y} - G(I^{LR}_{x,y}))^2 \tag{3}$$

such that r represents the factor with which the image is to be upscaled, W represents the image's width, H represents the image's height and (x, y) represent the indices of the pixels [8].

Many modern solutions depend on this objective as it is the most often used image *SR* optimization target. *MSE* optimization approaches mostly lack the fine details, which leads to perceptually unpleasant solutions, despite the fact that they often produce high *PSNR*.

The authors of *SRGAN* [8] use an objective function that is more related to perceptual relevance rather than pixel-wise losses. In their experiments, they use the ReLU activation layers of the pre-trained VGG network to calculate the VGG content loss. The feature map acquired by the j^{th} convolution prior to the i^{th} maxpooling layer within the VGG19 network, which is pre-trained, is indicated by $\phi_{i,j}$ (See Fig. 2). The VGG loss is then specified as the euclidean distance between the feature maps of the super-resolved image by the generator $G(LR)$ and the original HR image:

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G(I^{LR}))_{x,y})^2 \tag{4}$$

where $W_{i,j}$ and $H_{i,j}$, respectively, describe the widths and heights of different feature maps within the VGG network [8].

In [8], they claim that setting $i = 5$ and $j = 4$ gives the most visually compelling output, that they credit to the capacity of deeper architectures to capture elements of higher abstraction. They take the output after the activation (before layer 36 in the *VGG* network) in all of their experiments. However, in [15], they state that applying Eq. 4 before the activation (before layer 35 in the VGG network) is one of the steps to enhance the work the authors of [8] have done. One of the contributions of this paper is to compare the results of training the same network using varying layers of the VGG network to better understand the effect of relying on the VGG content loss.

For the adversarial loss, it depends on the output classification the discriminator $D(G(I^{LR}))$ on the training data, the generative loss l^{SR}_{Gen} is defined as:

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -\log D(G(I^{LR})) \tag{5}$$

2.3 Training Details

The two experiments have been conducted on a NVIDIA GeForce GTX 1650 GPU. Before training, the *LR* images were acquired by downsampling the *HR* counterparts in a bicubic kernel with a factor $r = 4$. A batch size of 16 images was used, where each high-resolution image is a 96×96 randomly cropped image from a distinct training image from the dataset. However, It is essential to keep in mind that the generator can accept images of any size since it is based on convolution layers.

Before training, low-resolution images were rescaled to the range $[0, 1]$ while the high-resolution images were rescaled to the range $[-1, 1]$. The same has been applied in the experiments of this paper. Adam optimizer was used with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ in all experiments. The *SRResNet* model that was trained with the *MSE* loss was utilised as a start for the generator while training the *GAN*-based generator to prevent undesirable local optima.

In each update iteration, both the adversarial models were optimized once in an alternative fashion, which means that $k = 1$ as defined by authors of *GAN* [5]. The residual blocks in the generator network are all identical, where the generator consists of 16 residual blocks in all the conducted experiments ($B = 16$). Pytorch was used to construct and train all the models in the experiments. Datasets were organized in a data loader, where each batch (16 examples) contained $24 \times 24 \times 3$ *LR* images and their corresponding $96 \times 96 \times 3$ *HR* images. Outdoor Scenes *OST* dataset [14] is the main dataset that was used for training in the experiments.

2.4 Training Experiments

Experiment 1: ***SRResNet* and *SRGAN36* training on *OST* dataset** [14]

- The generator has been trained alone (without training the discriminator) for 440 epochs (270,000 update iterations) using a learning rate $\eta = 1 \times 10^{-4}$. The loss that was used is an *MSE*-based loss between super-resolved images and HR ground truth images. This trained generator phase will be referenced as the *SRResNet* in the course of this paper.
- The *SRResNet* model (trained with *MSE* loss) was used as a start model for the generator in the next training phase, *SRGAN* [8]. The generator was trained using $\eta = 1 \times 10^{-5}$, while the discriminator was trained using a $\eta = 1 \times 10^{-6}$. Both networks were trained for 165 epochs (100,000 update iterations). The VGG loss has been taken with $i = 5$ and $j = 4$ **after the activation**. This takes feature maps of the VGG architecture before the 36^{th} layer. Hence, this phase will be referenced as *SRGAN36* in the course of this paper.
- Finally, the content loss Eq. 4 was multiplied by a factor of 0.006. This is to make the content loss scale comparable to the scale of the adversarial loss.

Experiment 2: *SRResNet* and *SRGAN35* training on OST dataset [14]

- The generator has been trained alone (without training the discriminator). This trained generator phase is exactly similar to the *SRResNet* training phase that was mentioned in Experiment 1.
- The *SRResNet* model (trained with *MSE* loss) was used as a start model for the generator in the next training phase, *SRGAN* [8]. The generator was trained using a $\eta = 1 \times 10^{-5}$, while the discriminator was trained using a $\eta = 1 \times 10^{-6}$. Both networks were trained for 165 epochs (100,000 update iterations). The VGG loss has been taken with $i = 5$ and $j = 4$ **before the activation**. This takes feature maps of the VGG architecture before the 35^{th} layer. Hence, this phase will be referenced as *SRGAN35* in the course of this paper.

2.5 Testing Details

Set5 [1] and *Set14* [16], the testing sets, are two commonly used benchmark datasets on which tests were conducted. As shown in Fig. 5, between low- and high-resolution images, a scaling factor of 4 is used in all tests. This translates to a $\times$16-pixel reduction in image size. Generated images from other *SR* techniques, including nearest neighbor interpolation, bicubic interpolation, *SRCNN* [2] were obtained from online supplementary materials to do a comparison with

Fig. 5. The left image is the low-resolution bicubic downsampled version that is input to the generator. The right image is the high-resolution real image with a scaling factor ×4. Both images are sample from the *Set14* dataset [16].

the generated *SRGAN36* and *SRGAN35* images.[1]. The following are all the 6 methods on which the tests have been done:

1. Nearest Neighbor Interpolation
2. Bicubic Interpolation
3. Super Resolution Using Deep Convolutional Networks or *SRCNN* [2]
4. Super Resolution Residual Network or *SRResNet* that was trained as initialization for experiments 1 and 2
5. Super Resolution *GAN* or *SRGAN36* of experiment 1
6. Super Resolution *GAN* or *SRGAN35* of experiment 2

The following has been done to test the above different super-resolution algorithms:

- For all the 6 methods, *PSNR* [dB] was calculated for all *Set5* [1] and *Set14* [16] for fair comparison.
- For all the 6 methods, *SSIM* was calculated for all *Set5* [1] and *Set14* [16] for fair comparison.
- For all the 6 methods, a Mean Opinion Score (*MOS*) has been calculated for 3 random images from the *Set5* [1] dataset and 6 random images from the *Set14* [16] dataset. This has been done by creating a google form. For each question, two images are presented beside each other, the first was a *Set14* [16] or *Set5* [1] generated output of one of the reference methods or one of the experiments of this paper, and the second was the *HR* ground-truth image. The quality of the output image has been rated by the participants with a number between 1 (bad quality) and 10 (excellent quality) compared to the *HR* original image from the dataset.
  The form consisted of 9 pages (each for a test image). Each page contained 6 comparison questions to compare the 6 corresponding methods to be compared. Consequently, each participant rated 6 versions of 9 images that were presented in a random order, summing up to 54 images to be rated.

[1] https://www.kaggle.com/datasets/ll01dm/set-5-14-super-resolution-dataset.

Fig. 6. Results for *Set14* [16] using bicubic interpolation, *SRResNet*, *SRGAN36* and *SRGAN35* in comparison to the ground-truth *HR* image [×4 upscaling].

Table 1. This table shows comparison of the 6 tested methods on *Set14* [16] dataset. Highest measures of average *PSNR*, *SSIM*, and Mean Opinion Score *MOS* are in bold.

Results on *Set14* [16] dataset						
	Nearest	Bicubic	SRCNN	SRResnet	SRGAN36	SRGAN35
PSNR	22.6	23.8	**24.72**	23.64	22.18	21.24
SSIM	0.64	0.68	**0.71**	0.70	0.62	0.58
MOS	4.68	5.29	6.36	6.23	**6.48**	6.07

3 Results and Analysis

This section will go over the *PSNR*, *SSIM*, and *MOS* numerical results that the trained generators of the two conducted experiments produced, beside showing the results of nearest neighbor interpolation, bicubic interpolation, and *SRCNN* [2] for comparison.

In the *MOS* test, 25 raters have participated in grading the *SR* techniques output images from ×4 downsampled images with a score between 1 (poor quality) and 10 (great quality). Of these images, 6 images were obtained from the *Set14* [16] dataset and 3 images were obtained from the *Set5* [1] dataset. The outcomes of the *MOS* test are presented in Tables 1 and 2.

Table 2. This table shows comparison of the 6 tested methods on *Set5* [1] dataset. Highest measures of *PSNR*, *SSIM*, and Mean Opinion Score *MOS* are in bold.

Results on *Set5* [1] dataset						
	Nearest	Bicubic	SRCNN	SRResnet	SRGAN36	SRGAN35
PSNR	24.37	26.45	**27.9**	25.87	23.57	22.42
SSIM	0.71	0.77	**0.81**	0.79	0.71	0.67
MOS	3.92	5.32	6.12	**6.21**	5.84	5.53

3.1 Investigating the Perceptual Loss

Basically, the goal of training an *SRGAN36* model in experiment 1 and an *SRGAN35* model in experiment 2, with the same *SRResNet* initialization, was to compare the effect of taking the VGG-loss before and after the activation. Tables 1 and 2 show the performance of both networks on the *PSNR*, *SSIM* and *MOS* metrics. It can be observed that *SRGAN36* outperformed *SRGAN35* on all metrics. This might make sense given the notion that taking VGG-loss deeper in the network yields the most convincing results [8]. It is also important to mention that the authors of [15] enhanced on [8] by taking the VGG-loss before the activation and achieved better results. However, this was accompanied by some changes in the structure of the generator itself. For example, they removed all

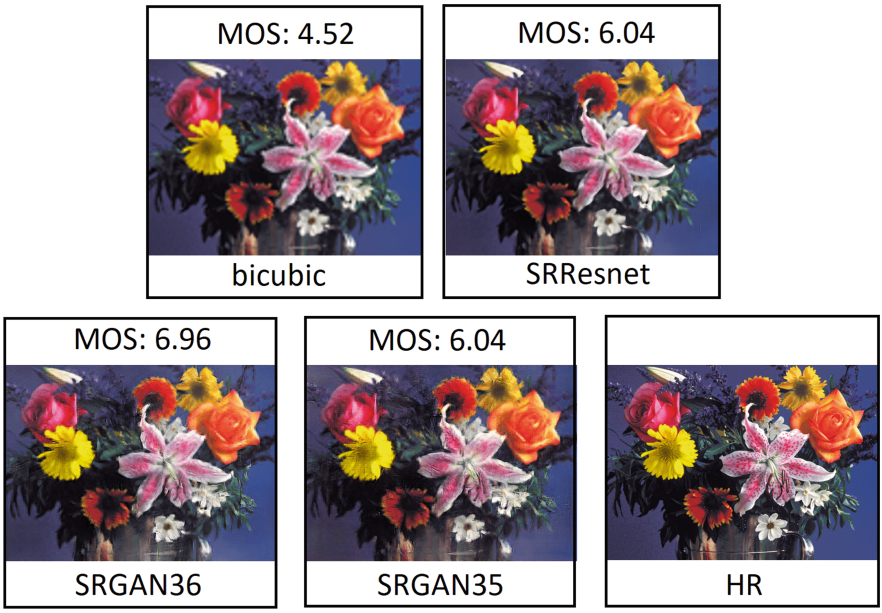

Fig. 7. Results for *Set14* [16] using bicubic interpolation, *SRResNet*, *SRGAN36* and *SRGAN35* in comparison to the *HR* image from the dataset [×4 upscaling].

the batch-normalization layers and increased the number of residual connections. These changes, however, were not employed in experiment 2 since the aim was to see the effect of only taking the VGG-loss before the activation but with the same generator structure that was used in the other experiments.

3.2 Investigating the Performance of Final Networks

SRResNet, *SRGAN36* and *SRGAN35* are compared to nearest neighbor interpolation, bicubic interpolation, and one of the modern algorithms, *SRCNN* [2]. Tables 1 and 2 summarize the quantitative results while Fig. 1, 6, 7, 8 summarize the qualitative results. These results show that *SRResNet* and *SRGAN36* establish a new state-of-the-art on *Set14* [16] and *Set5* [1] datasets. Compared to nearest neighbor interpolation and bicubic interpolation, *SRResNet* had higher *PSNR* and *SSIM* results on mostly all test images. *SRGAN36* and *SRGAN35*, on the other side, could not accomplish superior *PSNR* and *SSIM* results. Despite that, participants in the form gave *SRGAN36* and *SRGAN35* (and *SRResNet*) much higher scores than nearest neighbor and bicubic interpolations, on average.

Moreover, *SRCNN* [2] had the best *PSNR* and *SSIM* results. However, *SRGAN36* was able to do better than it on the average *MOS* on *Set14* [16] as shown in Table 1. This shows the limited potential of metrics like *PSNR* and *SSIM* to recover the image's perceptual quality, taking into account the fine texture details. This might be reasoned by the fact that these metrics are mostly based on pixel-level resemblance between two images. Hence, the weighted average of adversarial loss and content loss produced a new loss that has shown a

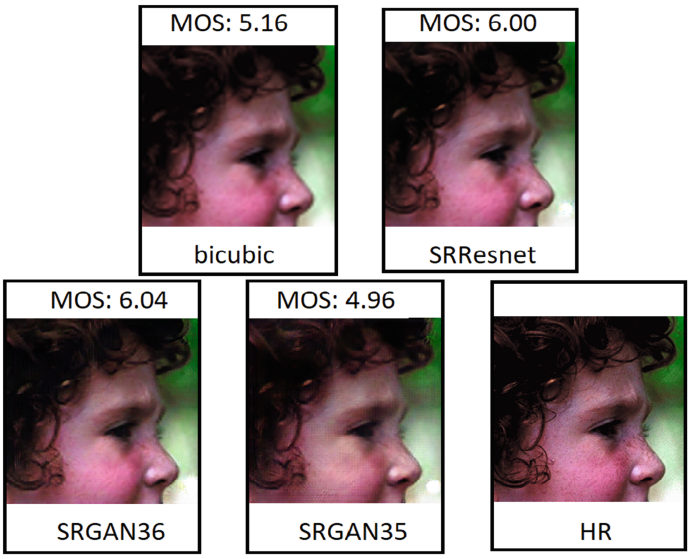

Fig. 8. Results for *Set5* [1] using bicubic interpolation, *SRResNet*, *SRGAN36* and *SRGAN35* in comparison to the*HR* image from the dataset [×4 upscaling].

great performance in capturing the fine details of images. This gave *SRGAN* [8] the ability to produce images that are relatively of higher quality.

4 Conclusion

After conducting two training experiments, an *SRResNet* model and two *SRGAN* models were trained and proven to be competitive with both conventional and cutting-edge super-resolution methods. With *MOS* testing, *SRGAN*'s satisfactory perceptual performance was confirmed. Images generated by the trained *SRResNet* and *SRGAN* models had the highest *MOS* scores on two benchmark datasets compared with interpolation techniques and one of the modern techniques, *SRCNN*. It was also demonstrated that typical performance measures like *PSNR* and *SSIM* do not always succeed in effectively judging image quality like how the perception of a human does. Despite the fact that traditional interpolation methods achieved higher *PSNR* and *SSIM* than *SRGAN* methods, *SRGAN* methods outperformed them in the average mean opinion score. However, It is essential to keep in mind that the *MOS* is quite subjective and the result can be different depending on the individuals who participated in the evaluation, the conditions in which the images are shown to the evaluators, and even the order of the images were shown. While attempting to output perceptually convincing result to solve the *SR* problem, the selection of perceptual loss is especially important. Taking the VGG-loss exactly before the $5th$ max pooling layer and after the activation, as opposed to taking it before the activation, yielded more perceptually appealing results for the participants in the *MOS* test on the *OST* dataset.

References

1. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding (2012)
2. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE PAMI **38**(2), 295–307 (2015)
3. Duporge, I., Isupova, O., Reece, S., Macdonald, D.W., Wang, T.: Using very-high-resolution satellite imagery and deep learning to detect and count African elephants in heterogeneous landscapes. Remote Sens. Ecol. Conserv. **7**(3), 369–381 (2021)
4. Gohshi, S.: Real-time super resolution algorithm for security cameras. In: ICETE, vol. 5, pp. 92–97. IEEE (2015)
5. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
6. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. In: CVPR (2016)
7. Kim, J., Lee, J.K., Lee, K.M.: Deeply-recursive convolutional network for image super-resolution. In: CVPR (2016)
8. Ledig, C., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR (2017)

9. Li, H., Zheng, Q., Yan, W., Tao, R., Qi, X., Wen, Z.: Image super-resolution reconstruction for secure data transmission in internet of things environment. Math. Biosci. Eng. **18**(5), 6652–6672 (2021)
10. Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., Timofte, R.: Swinir: image restoration using swin transformer. In: CVPR, pp. 1833–1844 (2021)
11. Liu, Y., Qiao, Y., Hao, Y., Wang, F., Rashid, S.F.: Single image super resolution techniques based on deep learning: status, applications and future directions. J. Image Graph. **9**(3) (2021)
12. Shi, W., et al.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: CVPR (2016)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
14. Wang, X., Yu, K., Dong, C., Loy, C.C.: Recovering realistic texture in image super-resolution by deep spatial feature transform. In: CVPR (2018)
15. Wang, X., et al.: ESRGAN: enhanced super-resolution generative adversarial networks. In: Leal-Taixé, L., Roth, S. (eds.) ECCV 2018. LNCS, vol. 11133, pp. 63–79. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11021-5_5
16. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Boissonnat, J.-D., et al. (eds.) Curves and Surfaces 2010. LNCS, vol. 6920, pp. 711–730. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27413-8_47