Weighted SRGAN and Reconstruction Loss Analysis for Accurate Image Super Resolution

Hangpu Cao1, a, † and Sicheng Mi2, b, †

1School of Information Science and Technology, Fudan University, Shanghai, 200433, China;
2College of Intelligence and Computing, Tianjin University, Tianjin, 300350, China;
*Corresponding author e-mail: a18307130061@fudan.edu.cn, bmisicheng@tju.edu.cn
†These authors contributed equally.

Abstract. Super resolution (SR) image generation is to generate a high-resolution image from a given low-resolution image, which can be used in numerous vision tasks, such as small object detection and specific image processing. Generative adversarial network for super resolution (SRGAN) is the mainstream framework in SR task, which utilizes two reconstruction losses, including MSE loss and VGG loss. However, the influence of these losses on model learning is not examined. In this paper, the importance of MSE and VGG losses is analyzed. The loss function of traditional SRGAN is improved, and the weights of MSE and VGG losses are set manually. The weights range from 0 to 1, and the sampling interval is 0.1. Furthermore, a learnable parameter to dynamically adjust the two weights is proposed. Experiments on the datasets, including Set5, Set14, BAS100, and Urban100, show that our method is capable of generating much better images than SRGAN, with higher values of PSNR and SSIM. It is found that the MSE loss makes a greater contribution to learning the discriminative model and the VGG loss plays a supplementary role. Our WSRGAN can apply to most SRGAN-based methods to improve their accuracy.

1. Introduction
Super resolution (SR) is a highly challenging task that generates a high-resolution (HR) image from a low-resolution (LR) image. The evaluation of its detailed information is very important. And the authenticity of generated images is also of vital importance. SR has been widely used in the field of computer vision such as small object detection [1, 2] and medical image processing [3, 4]. Related papers and work about Generative adversarial network for super resolution (SRGAN) network include detecting small objects [5], designing a novel 3D neural network in high-resolution (HR) magnetic resonance images (MRI) [6], employing Maximum a Posteriori (MAP) inference [7], using Wasserstein distance as the training objective [8], and addressing the super-resolution problem for fluid flow by using a temporally coherent generative model [9]. Although SRGAN plays an important role in the SR domain, there are several problems with SRGAN. In some cases, the results of SRGAN are not very satisfying. The loss function of SRGAN consists of two parts: VGG loss and MSE loss [10]. Though they are proposed without explanation, the analysis between the two losses needs to be conducted. The best result cannot be produced by simply putting them together. Since the relationship between VGG loss and MSE loss remains unclear in the reference [10], the relationship between these two loss functions is explored in this paper. In this process, which one of the two loss functions...
contributes more and what proportion between them can make the model have better results are figured out. In order to achieve the above goals, this paper uses the same architecture as SRGAN [10] and improves its loss function. Different weights to VGG loss function and MSE loss function are assigned and the performance of the model under different weights is observed. In addition, a dynamic weight that can find better weights more accurately is also learned. Finally, this paper uses the commonly used Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM) to evaluate the performance of all models and make a comparison between them. In this paper, the contribution of two loss functions, including MSE loss and VGG loss, is analyzed. It is found that the MSE loss makes a greater contribution to model learning and VGG loss is supplementary to build a more discriminative model. Furthermore, the Weighted Super-Resolution GAN (WSRGAN) which can dynamically learn the weight between the MSE and VGG losses is proposed. Our networks on the training dataset with 16700 images sampled from VOC2012 are trained. Our model on the testing dataset which is sampled from Set 5[11], Set 14[12], BSD 100, and Urban 100[13] is tested. When 0.9 to MSE loss and 0.1 to VGG loss are assigned manually, the best performance is obtained. For example, on the Set 5 dataset, the PSNR score is 26.8537 and the SSIM score is 0.7923. With the dynamic weight, our WSRGAN achieves the best performance with the PSNR score being 27.3371 and the SSIM score being 0.8036 on the Set 5 dataset.

2. Method

2.1. GAN

2.1.1. Construction
GAN (Generative Adversarial Networks) is one of the most popular deep learning models for unsupervised learning. It is firstly introduced by Ian J. Goodfellow [14] in 2014 and has been widely used for image generation. GAN model has two modules: G (Generator) and D (Discriminator). G is an image generating network. It receives a random noise $z$ and generates an image $G(z)$. D is a discriminating network. It receives an image $x$ and judges whether $x$ is a real image. The output $D(x)$ is the probability that $x$ is real. If $D(x)$ is 1, $x$ is a 100% real image. And if $D(x)$ is 0, $x$ cannot be a real image.

2.1.2. Training
In the training process, the goal of G is to generate “real” images to cheat D. And the goal of D is to discriminate images generated by G from real images. Therefore, G and D constitute a dynamic “game process”.

More specifically, the loss function is

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim P_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim P_z} [\log (1 - D(G(z)))]$$

(1)

After initializing G and D, we get some fake images from G. GAN training can be divided into two steps in a round of gradient back-propagation. Firstly, we fix G’s parameter and train D. We send fake images and real images to D, then D will judge them through Binary Cross-Entropy Loss (BCE loss). Its loss function is made up of the difference between D’s output and the label. Secondly, we fix D’s parameter and train G. We put the last fake image into D and give it the label 1(the goal). Its loss function consists of the difference between D’s output and 1.

2.1.3. Test
At last, the ideal result is that D cannot discriminate $G(z)$, in other words, $D(G(z))$ is 0.5. Hence, we have a good generator G which can generate images.

In the test part, after we train the model with a set of images in MNIST, the generator can produce a similar image that seems to be “real”. The input is just random noise, and the discriminator also gives high confidence. That image generated by G is the output we want.
2.2. GAN for super resolution
Our research aims to generate a corresponding higher-resolution image when a low-resolution image is given. The paper SRGAN[10], which is proposed by Christian Ledig et al. in 2017, is utilized as our baseline. The authors put forward two main revolutionary innovations. First, the image super-resolution network SRResNet [10] based on ResNet [15] is established. Second, the optimization goal is the perceptual loss that contains the content loss based on MSE loss and VGG loss. The two main innovations mentioned above are described in more detail in Section 2.2.1.

2.2.1. Model

2.2.1.1. SRGAN’s architecture
According to the structure of the general GAN [14], SRGAN is also divided into generator model G and discriminator model D.

According to the paper by Christian Ledig et al. [10], the generator of SRGAN was composed of a residual network and a sub-pixel convolutional network. As shown in Figure 1, the core of the generator was B residual blocks. Each residual block contained two convolutional layers with 3*3 kernels and 64 feature maps. And each convolutional layer was followed by a batch-normalization layer and activation function ParametricReLU. Finally, they used two trained sub-pixel convolution layers to increase the resolution of the input image.

It is worth noting that the skip connection is used in each residual block. And this is one of SRGAN’s great innovations. We know that deep convolutional neural networks play a very important role in the field of image processing because deeper layers mean better accuracy. However, with the increase of network depth, the accuracy reaches saturation and then degrades rapidly. That is to say when the network reaches a certain number of layers, the performance of the model will decline if someone continues to deepen the network. Fortunately, the deep residual network solves this problem. In the famous paper written by Kaiming He et al. [15] in 2015, they introduce identity mapping. And the “shortcut connection (skip connection)” implements it. Meanwhile, the identity shortcut connections don’t bring more parameters and computational complexity. And this feature is well-preserved in the residual module of SRGAN’s generator.

![Figure 1. The architecture of generator and discriminator. (“9*64*1” means that the corresponding convolutional layer’s kernel size is 9, the number of feature maps is 64, and stride is 1.)](image-url)
As for the discriminator D, we learn the architecture summarized by Radford et al [16] as Figure 1. They also choose LeakyReLU as the activation function ($\alpha = 0.2$). The network has 8 convolutional layers. Each layer contains several $3 \times 3$ filter kernels, and the number of channels is doubled every second layer from 64 to 512, just like the VGG network [17]. The final 512 feature maps go into two dense layers. And they use sigmoid as the last activation function to get the classification probability. It is worth mentioning that they use stridden convolutions to reduce the image resolution when the kernels are doubled.

2.2.1.2. SRGAN’s loss function

Designing a completely new loss function is another great innovation in the paper [10]. And the performance of SRGAN is significantly improved by their perceptual loss function:

$$l_{SR}^* = l_{SR}^* + 10^{-3} l_{gen}^{SR}$$

(2)

The perceptual loss function is composed of content loss and adversarial loss.

Previous studies based on convolutional neural networks have used Mean Squared Error (MSE) as a loss function. This method can get a good Signal-to-Noise Ratio (SNR) but lack high-frequency information. SRGAN pays more attention to the differences between the features of the reconstructed pictures and the original pictures, rather than the color brightness differences between pixels. Therefore, they do not simply use MSE loss to express content loss. SRGAN uses the VGG19 network to extract image features, and then calculates content loss:

$$l_{VGG}^{SR} = \frac{1}{W_i \cdot H_i} \sum_{x=1}^{W_i} \sum_{y=1}^{H_i} \left( \varphi_{i,j}(I_{HR})_{x,y} - \varphi_{i,j}(G_{\theta_{g}}(I_{LR}))_{x,y} \right)^2$$

(3)

$\varphi_{i,j}$ is the feature map from the jth convolution (after activation) before the ith max-pooling layer in the VGG19 network. $G_{\theta_{g}}(I_{LR})$ is the feature representations of a generated image. $W_{i,j}$ and $H_{i,j}$ are the dimensions of the respective feature maps in the VGG network.

The adversarial loss function helps SRGAN have a better performance on the manifold of natural images:

$$l_{gen}^{SR} = \sum_{n=1}^{N} - \log D_{\theta_{d}} \left( G_{\theta_{g}}(I_{LR}) \right)$$

(4)

$D_{\theta_{d}} \left( G_{\theta_{g}}(I_{LR}) \right)$ represents the probability that the generated image is a natural HR image.

2.2.2. Training

In the paper of Christian Ledig et al. [10], the generator uses a residual network consisting of 16 (B = 16) residual blocks with skip connections. When training the SRGAN model, 350 thousand randomly sampled images from the ImageNet database are used. They use a bicubic kernel to downsample the HR images and obtain the LR images (downsampling factor = 4). When starting training, they firstly input the LR image into the generator, and the generator output a reconstructed picture. Then the reconstructed picture and the original high-resolution picture are input into the discriminator, and the discriminator judges the reconstructed picture. They define the difference between the reconstructed image and the original image as loss and update the loss to train the model. In order not to struggle in the local optima, the generator is initialized by the trained SRResNet network that has a learning rate of $10^{-4}$ and $10^{-6}$ update iterations. And they use an NVIDIA Tesla M40 GPU to train all the models.

2.2.3. Test

In the paper of Christian Ledig et al. [10], they use the Mean Opinion Score (MOS) test. 26 raters score the super-resolution images from 1 (bad) to 5 (excellent). They use the images of three benchmark datasets Set5 [11], Set14 [12], BSD100 to test their model’s performance. And they find that SRGAN is significantly better than other methods, such as Nearest Neighbor(NN), bicubic, SRCNN [18], SelfExSR [13], DRCN [19], ESPCN [20], SRResNet.
2.3. Weighted super-resolution GAN

After carefully studying [10], we believe that their content loss function is a revolutionary improvement. Content loss consisted of MSE loss and VGG loss. However, their paper does not explain the impact of MSE loss and VGG loss on network training. Therefore, the aim of our study is to analyze the impact of MSE loss and VGG loss on training an SRGAN model and improve its performance.

The way we achieve our goal is to assign different weights to MSE loss (0:0.1:1) and VGG loss (1:0.1:0). Our training dataset has 16700 images which are sampled from VOC2012. And our testing datasets are sampled from Set 5, Set 14, BSD 100, Urban 100. When we train our network, we give weight \( w \) to MSE loss and weight \( (1-w) \) to VGG loss \( (0<\text{w}<1) \). We set the interval of \( w \) to 0.1. In this case, we train SRGAN according to [10]. We train 100 epochs for each weight. After many experiments, we evaluate the results of these experiments and then choose the optimal weight.

\[
I_{\text{SR}}^x = w \times I_{\text{MSE}}^x + (1 - w) \times 10^{-3} \times I_{\text{VGG}}^x \quad (5)
\]

In addition, we also use dynamic weight, which is a parameter that can be learned, to improve our network performance and name it WSRGAN.

Our method explores the relationship between two different loss functions, which could help improve the super-resolution image quality. In particular, WSRGAN could help the network perform well in different situations, such as the training dataset of photos, which contain different objects.

3. Results and Discussion

To find a better weight, we use two universal evaluation indexes in the image field to evaluate each weight: Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM).

PSNR is used to measure the quality of a reconstructed image and based on the error between the corresponding pixels. In our experiment, we use PSNR to evaluate the difference between the reconstructed image and the original high-resolution image. The more similar the two images are, the greater the PSNR value is.

Most image quality evaluation methods based on error sensitivity (such as MSE, PSNR) use linear transformation to decompose the image signal, which does not involve correlation. The basic idea of structural similarity is that pictures are highly structured and there is a strong correlation between adjacent pixels. And this correlation reflects on the structural information of the objects in images. People often focus on and extract such structural information when viewing pictures. Therefore, compared with PSNR, SSIM is more consistent with people's feelings of picture quality. The similarity measurement of SSIM consists of three parts: brightness, contrast, and structure. When the two images are exactly the same, the value of SSIM is equal to 1.

Our machine configuration is as follows: CPU: I9-9900K, GPU: RTX 2080ti, RAM: 64G, Pytorch.

Table 1~4 lists the weights that we test for MSE loss and VGG loss, and the PSNR and SSIM values corresponding to each weight. As we can see in Table 1~4, we use four datasets to test. And the weight changes from 1 to 0. Then the PSNR and SSIM change as well. The results of the experiment find clear support for our assumption that MSE loss and VGG loss have varying degrees of influence on SRGAN, and its performance can be improved by adjusting the weight. Here we compare the results of our method with those of the basic method SRGAN [10]. When we give 0.5 to MSE loss and 0.5 to VGG loss, we reproduce a basic SRGAN (baseline). And when we give 0.9 to MSE loss and 0.1 to VGG loss, we find that the values of PSNR and SSIM are better than the baseline result, and they are the best in the results of 11 weights we test.

Table 1. The impact of MSE loss and VGG loss under different weights on PSNR and SSIM, using Set5 dataset.

| \( w \) (MSE loss) | 1 - \( w \) (VGG loss) | PSNR | SSIM |
|-------------------|------------------------|------|------|
| 1                 | 0                      | 26.03| 0.7867|
| 0.9               | 0.1                    | 26.85| 0.7923|
It shows that both MSE loss and VGG loss play important roles in training. MSE loss mainly contributes to the learning processing. VGG loss plays a supplementary role, but it is necessary. With the guidance of VGG loss, we can obtain a more precise model for the super-resolution image reconstruction task. We also find that giving a larger weight to MSE loss will improve the quality of the reconstructed image.

The results given in Table 5 show the difference between the weight set manually and the weight learned when training the network. As we can see in Table 5, in our four datasets, weighted SRGAN performs better than SRGAN, and WSRGAN is better than weighted SRGAN. Compared with weighted SRGAN, WSRGAN can automatically give the appropriate weight. And we believe that in most cases, the weights given by WSRGAN are better than those given manually. That is because the number of weights given manually is limited and cannot be very accurate. Even if we assume that the weights given manually in a

| w (MSE loss) | 1 - w (VGG loss) | PSNR | SSIM |
|-------------|------------------|------|------|
| 1           | 0                | 23.97| 0.6965 |
| 0.9         | 0.1              | 24.13| 0.6971 |
| 0.8         | 0.2              | 24.10| 0.6980 |
| 0.7         | 0.3              | 23.72| 0.6885 |
| 0.6         | 0.4              | 23.73| 0.6895 |
| 0.5 (baseline) | 0.5 (baseline)   | 22.68| 0.6886 |
| 0.4         | 0.6              | 23.28| 0.6824 |
| 0.3         | 0.7              | 23.03| 0.6721 |
| 0.2         | 0.8              | 22.63| 0.6747 |
| 0.1         | 0.9              | 23.75| 0.6755 |
| 0           | 1                | 4.973| 0.008211 |

Table 3. The impact of MSE loss and VGG loss under different weights on PSNR and SSIM, using BSD100 as the dataset.
Table 4. The impact of MSE loss and VGG loss under different weights on PSNR and SSIM, using Urban100 as the dataset.

| w (MSE loss) | 1 - w (VGG loss) | PSNR | SSIM |
|--------------|------------------|------|------|
| 1            | 0                | 22.10| 0.6620|
| 0.9          | 0.1              | 22.12| 0.6601|
| 0.8          | 0.2              | 22.10| 0.6638|
| 0.7          | 0.3              | 21.95| 0.6519|
| 0.6          | 0.4              | 21.92| 0.6555|
| 0.5 (baseline)| 0.5 (baseline)   | 21.16| 0.6557|
| 0.4          | 0.6              | 21.63| 0.6462|
| 0.3          | 0.7              | 21.29| 0.6391|
| 0.2          | 0.8              | 21.27| 0.6412|
| 0.1          | 0.9              | 21.81| 0.6445|
| 0            | 1                | 5.274| 0.0166|

certain dataset are optimal, we cannot guarantee that accurate weights can be given to different datasets. It is unacceptable to spend a lot of time doing repetitive things each time. Therefore, in order to make our conclusions robust, we believe that WSRGAN is the best result we want.

Table 5. Different promotion of the model by 3 methods: SRGAN, weighted SRGAN, and dynamic weighted SRGAN.

| Method         | DataSet | PSNR | SSIM |
|----------------|---------|------|------|
| SRGAN          | Set5    | 24.38| 0.7864|
| Weighted SRGAN | Set5    | 26.85| 0.7923|
| WSRGAN         | Set5    | 27.34| 0.8036|
| SRGAN          | Set14   | 22.68| 0.6886|
| Weighted SRGAN | Set14   | 24.13| 0.6971|
| WSRGAN         | Set14   | 24.61| 0.7087|
| SRGAN          | BSD100  | 23.53| 0.6713|
| Weighted SRGAN | BSD100  | 24.67| 0.6791|
| WSRGAN         | BSD100  | 24.99| 0.6795|
| SRGAN          | Urban100| 21.16| 0.6557|
| Weighted SRGAN | Urban100| 22.12| 0.6601|
| WSRGAN         | Urban100| 22.47| 0.6785|

4. Conclusion
In the paper, it is found that the MSE loss plays a major role in the training of the SRGAN model. The VGG loss only plays a supporting role, but its existence is also necessary. When training our model, giving a larger weight to MSE loss will improve its performance. In addition, a WSRGAN that can dynamically learn the weight between the MSE and VGG loss is also proposed. In order to achieve our goal, different weights to MSE loss and VGG loss are assigned manually to observe the relationship between the two loss functions. The value of the weight is between 0 and 1, and its sampling interval is 0.1. Then, a dynamic weight to automatically change the weights between MSE and VGG losses is learned. Our training dataset has 16700 images sampled from VOC2012, and our testing dataset is sampled from Set 5, Set 14, BSD 100, and Urban 100. The optimal setting is that the weight of MSE loss is 0.9 and the weight VGG loss is 0.1. For example, on the Set 5 dataset, the PSNR score is 26.8537 and the SSIM score is 0.7923. It shows that our method finds the relationship between the two
loss functions and can improve the performance of SRGAN. In terms of dynamic weight, our WSRGAN gets the better performance, with the PSNR score being 27.3371 and the SSIM score being 0.8036 on the Set 5 dataset. Our WSRGAN can promote the performance of most SRGAN based methods in many research fields, including image compression, public security system, and medical imaging.

References
[1] Bell, S., Zitnick, C. L., Bala, K., Girshick, R. (2016). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2874-2883.
[2] Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A. (2014) Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531.
[3] Plenge, E., Poot, D. H., Bernsen, M., Kotek, G., Houston, G., Wielopolski, P., Weerd, L. V. D., Niessen, W. J., Meijering, E. (2012) Super-resolution methods in MRI: can they improve the trade-off between resolution, signal-to-noise ratio, and acquisition time? Magnetic Resonance in Medicine, 68(6), 1983-1993.
[4] Tanno, R., Worrall, D. E., Ghosh, A., Kaden, E., Sotiropoulos, S. N., Criminisi, A., Alexander, D. C. (2017) Bayesian image quality transfer with CNNs: exploring uncertainty in dMRI super-resolution. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 611-619.
[5] Li, J., Liang, X., Wei, Y., Xu, T., Feng, J., Yan, S. (2017) Perceptual generative adversarial networks for small object detection. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1222-1230.
[6] Chen, Y., Shi, F., Christodoulou, A. G., Xie, Y., Zhou, Z., Li, D. (2018) Efficient and accurate MRI super-resolution using a generative adversarial network and 3D multi-level densely connected network. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 91-99.
[7] Sønderby, C. K., Caballero, J., Theis, L., Shi, W., Huszár, F. (2016) Amortised map inference for image super-resolution. arXiv preprint arXiv:1610.04490.
[8] Chen, Z., Tong, Y. (2017) Face super-resolution through wasserstein gans. arXiv preprint arXiv:1705.02438.
[9] Xie, Y., Franz, E., Chu, M., Thuerey, N. (2018) tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. ACM Transactions on Graphics, 37(4): 1-15.
[10] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4681-4690.
[11] Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M. L. (2012) Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: British Machine Vision Conference.
[12] Zeyde, R., Elad, M., Proter, M. (2012) On single image scale-up using sparse-representations. In: International Conference on Curves and Surfaces. pp. 711-730.
[13] Huang, J. B., Singh, A., Ahuja, N. (2015) Single image super-resolution from transformed self-exemplars. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5197-5206.
[14] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014) Generative adversarial networks. arXiv preprint arXiv:1406.2661.
[15] He, K., Zhang, X., Ren, S., Sun, J. (2016) Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 770-778.
[16] Radford, A., Metz, L., Chintala, S. (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.
[17] Simonyan, K., Zisserman, A. (2014) Very deep convolutional networks for large-scale image
recognition. arXiv preprint arXiv:1409.1556.

[18] Dong, C., Loy, C. C., He, K., Tang, X. (2014) Learning a deep convolutional network for image super-resolution. In: European Conference on Computer Vision. pp. 184-199.

[19] Kim, J., Lee, J. K., Lee, K. M. (2016) Deeply-recursive convolutional network for image super-resolution. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1637-1645.

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