FPT Education

FPT UNIVERSITY

REPORT

SUPER-RESOLUTION USING GENERATIVE ADVERSARIAL NETWORKS(GANS)

Group 3:

Phạm Minh Tuấn - SE160561

tuanpmse160561@fpt.edu.vn

Nguyễn Quốc Hùng - SE162007

hungnqse162007@fpt.edu.vn

Abstract

To respond to the trend of the breakthroughs in Deep Learning and the series innovations in Computer Vision, more and more algorithms and models are published widely around the world on lots of science forums, research websites… However, almost all of these models focus on using mean squared loss to reconstruct the image and very few models have a photo-realistic natural image such as SRGAN, a generative adversarial network (GAN) for image super-resolution (SR) of Christian et al [1]. Moreover, that's the reason why our motivation is encouraged to discover what actually exists in this 'black-box', which makes the SRGAN so popular all over the world. Except for that, we also try to reuse or optimize that model for creating better super-resolution output images.

I. Introduction

As mentioned above, we decided to reimplement the method of Christian et al. [1] for upscaling to high-resolution image from low-resolution image rather than creating a completely new direction from scratch. Our aim is to know what is the solution, thinking's direction that are referred to in their research to build our complete model for super-resolution problems.

In this work, we have optimized the variant architecture of the generator with more widely-used sub-layers such as convolution, PeakyReLU, BatchNormalization, increasing the number residual blocks as well as replacing the traditional zero padding method with a new type of padding named "Symmetric Padding". We also decided to keep the original discriminator structure proposed by Christian et al. [1]. On top of that, we add some support minor tools to achieve a better super-resolution of the output image such as matching histogram to ensure the RGBs of the image after training and applying Fast Fourier Transform (FFT) to create the clearer pattern of the low-frequency details.

II. Related work

The successive work by Krizhevsky et al. [29] of designed CNN structure was shown that using the application of residual blocks [29] and skip-connections [30, 34] as well as batch normalization [32] led to the efficient assessment of accuracy and speed for image super-resolution problems. That's why lots of existing solutions for this one is published widely around the world and . For example, the deeply-recursive convolutional network(DRCN), Kim et al. [34] has increased recursion depth and can improve performance without introducing new parameters for additional convolutions. Besides, Dong et al. [9, 10] had proposed the model's name Super-Resolution Convolutional Neural Network (SRCNN) which used patch extraction and representation and non-linear mapping to reconstruct SR images.

The main core for the structure of Christian's work is employing generative adversarial networks (GANs) [22], which using the generative model is pitted against a proposed adversarial nets framework. Except for that, Christian Ledig also combined the Euclidean distances for loss function of Dosovitskiy and Brox [13] as well as the use of features extracted from a pretrained VGG network of Johnson et al. [33] and Bruna et al. [5] instead of low-level pixel-wise error measures with poor perceptual quality [42, 33, 13, 5] to create the better SR images.

III. Method

1. Framework :

1.1. The original generator network :

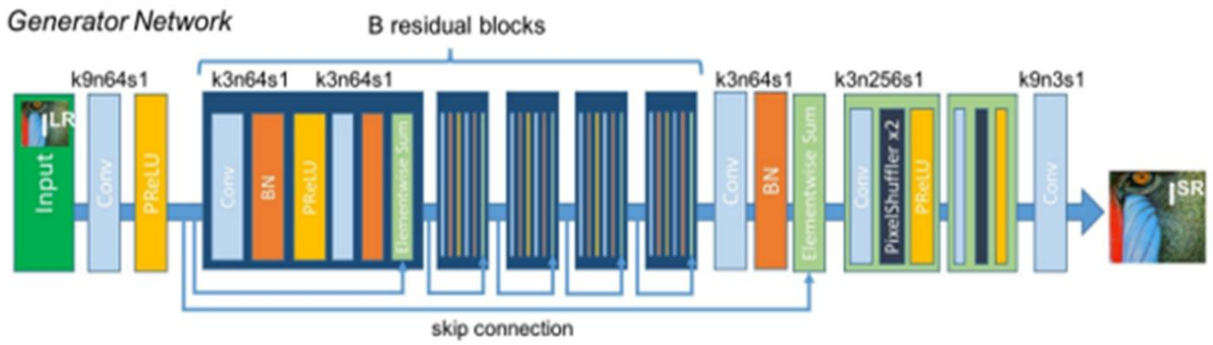


Figure 1. The proposed initial generator network by Christian et al. [1].

We mainly concentrate on the deep generator and our modified generator model after adjusting:

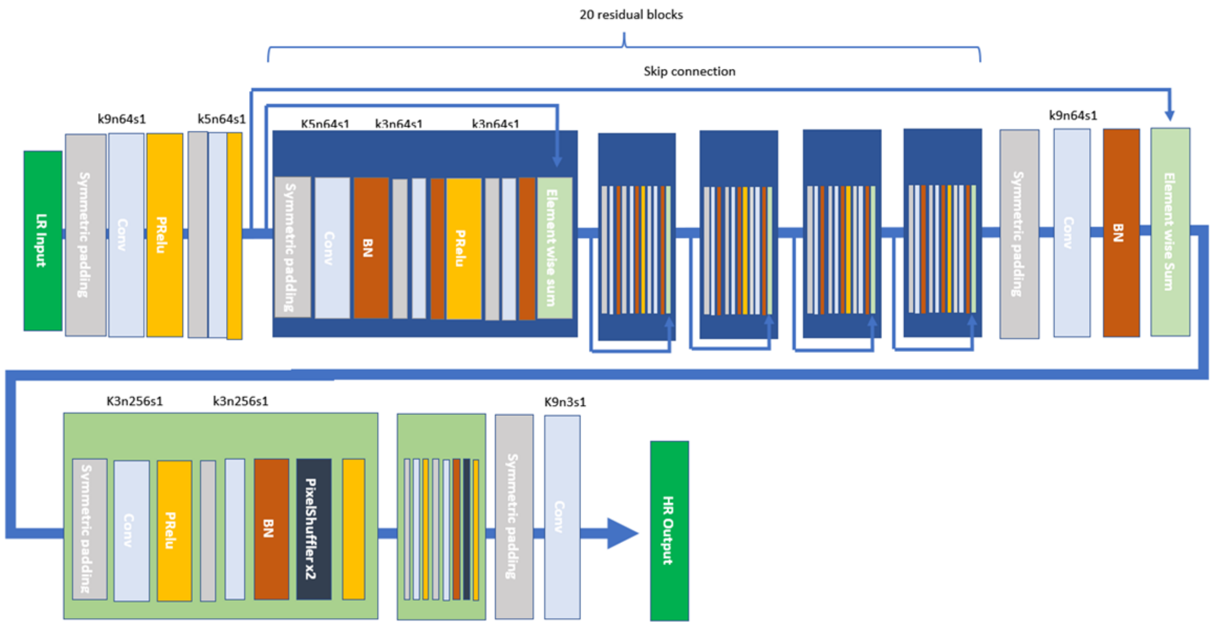


Figure 2. Our modified deep generator network.

By using symmetric padding of Rong Song et al. [2] instead of default zero padding , we confidently reserve details at the edge of the output image for a cleaner merged final image. We also increase the size of the kernel by 9 x 9 for the first input in the hope of accelerating the model's ability to learn larger structure. Specially, using two combined convolutional layers with these 9x9 kernels and 64 feature maps followed by batch-normalization layers [32] and ParametricReLU [28] as the activation function have increased the resolution, proposed by Shi et al. [48] We also increase the number of residual blocks to 20 blocks for accelerating the learning speed of the model and using Batch Normalization layer with momentum is 0.5 to avoid the instance of **vanishing gradient**.

1.2. A discriminator network:

We decided that we preserve the architecture proposed by Christian et al. [1]. The architecture is shown in Figure 3. After testing the demos using different hyperparameters, we agreed with each other to use the LeakyReLU activation which α = 0.3 and avoid max-pooling throughout the network. With eight convolutional layers and lots of filters, we believe that our discriminator has the ability to punish the generator to create a better super-resolution image.

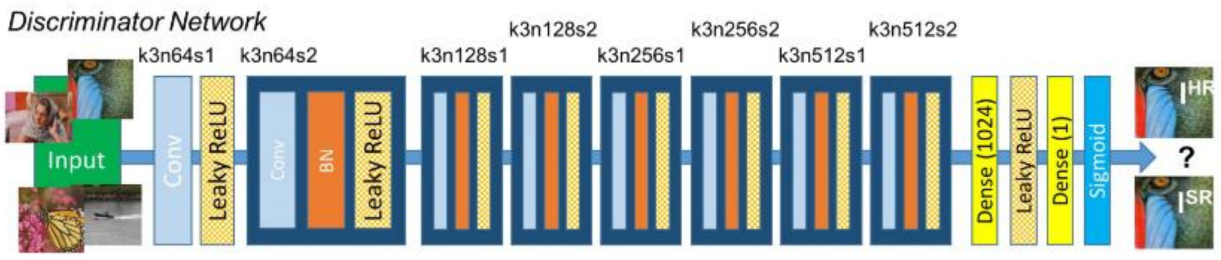


Figure 3. The proposed discriminator network by Christian et al. [1].

2. The adjustment of content loss (perceptual loss) proposed by Christian et al. [1]:

Instead of using just the tenth-layer of the pre-trained VGG19 network as referred and implemented by Christian et al. [1], we have decided to combine 2 more optional layers which is as close to the content representation as described in Figure 4 by Leon A. Gatys et al. [2]. This conclusion makes our model create more and more fake artistic feature patterns to fill in input low-resolution image, from this leading to decrease effectively the loss of VGG19 network for content loss of Christian et al. [1].

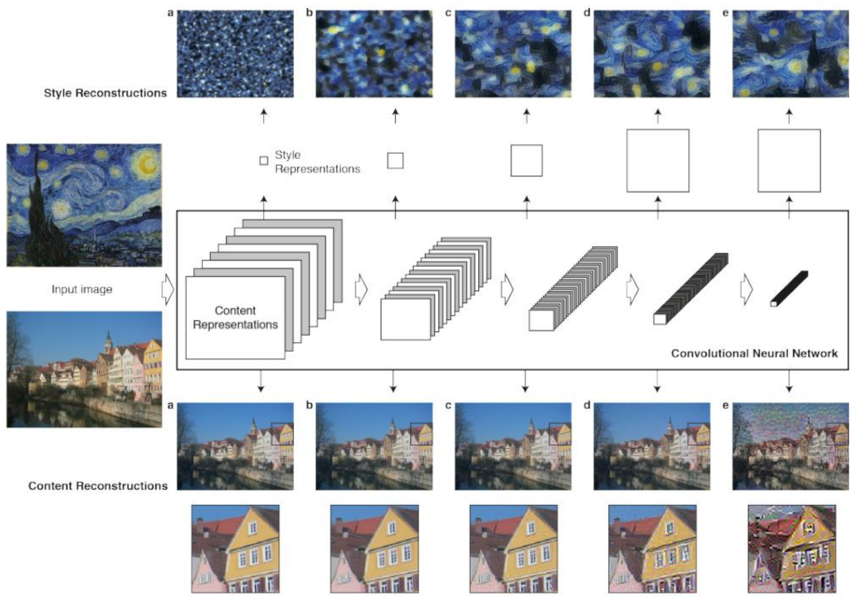


Figure 4: Convolutional Neural Network (CNN) represents the "artistic style" level of each corresponding layer. The closer content reconstruction means the better preservation for real details and is almost as similar as the initial input.

3. Symmetric padding:

To stabilize consistently about the size of the image through all of the layers in these residual blocks. However, when our model passes in each block, the neural network will add zero padding in the rear of the input in order for the consistent size throughout the model. As a result, this brings some useless information when reconstructing the image in the rear, which is fine if we only predict one image, however our goal is to merge multiple outputs to create a larger image, using zero padding will result in visible lines in the final merged image, described in Figure 5. . Luo et al. [19] invented a new way by calculating the value for the pad of the output result by using the original value of existing in the input image. With this solution, we can avoid the existence of useless information which leads to some visible borders on the image after reconstructing, Figure 6.

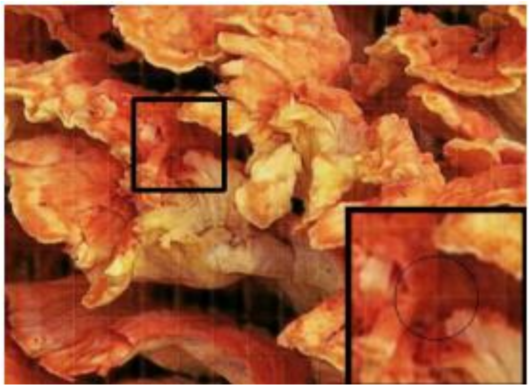


Figure 5. The image shows an existence of cutting lines in research of Rong Song et al. [2].

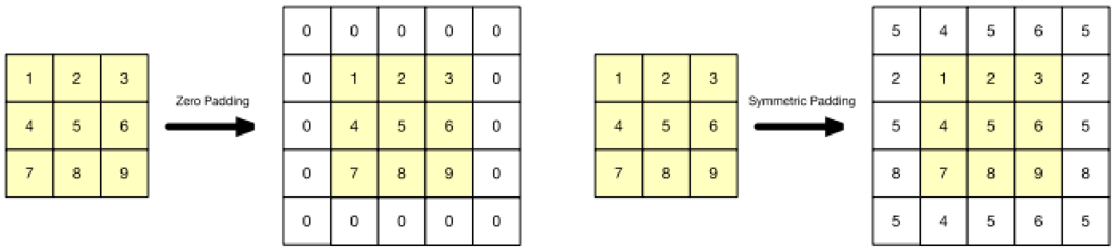


Figure 6. Left one is zero padding, right one is symmetric padding sample from the Rong Song et al papers [2].

4. Matching histogram

Since image features are easier to tell for the discriminator to determine the image is real or fake and help to guild the generator. It is the opposite for the color value, which if the generator produces a slight discoloration from the input image, it can be easily told by the human eye but might be quite a challenge for the discriminator. That is why our resulting epoch might have a better feature reconstruction down the line, color features are a slight vary with the generator having a slight problem determining the right color from the input image. That is why for the sake of consistency, we decided to use histogram matching technique with color reference from the original low-resolution input in order to match the color for the high-resolution output.

We follow the direction of Dori Shapira et al. [3] who proposed the method for multiple histogram matching. After that, we use ski-image library which has an available pre-implementation function named skiimage.exposure.matchingHistogram and start to match the color of target image and source image.

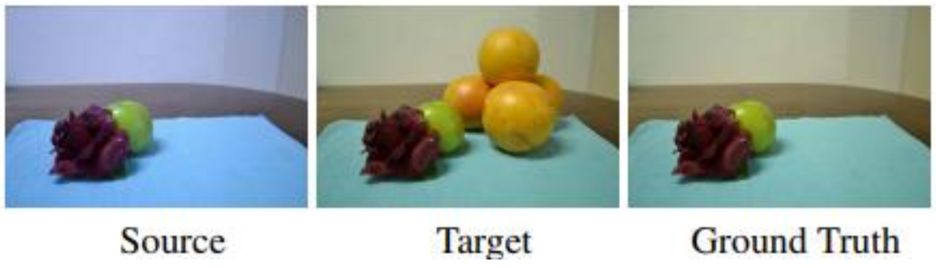


Figure 7. The general vision what mean matching histogram.

5. Application for Fast Fourier transform in image processing

Another problem occurs with our generator is the tendency to produce patterns where it can tell there are a lot of details that need to be filled in. The generator might be using it as it can tell the discriminator was consistently being fool with the pattern added in, and the pattern might be used as a means of minimizing the perceptual loss of the overall GAN network. That is why we find our way into digital signal processing via Fast Fourier transform (FFT) to create a low pass filter to create a more pleasing output image with the noise and the pattern generated from our generator being minimized.

Introduced by Ann Maria John et al. [4], we have an idea to use the Fast Fourier Transform (FFT) to convert the final output image to the spectrum of a frequency signal, after that apply our custom mask to the center of the spectrum consisting of low-pass signals. Then, our final image can decrease the applying disorderly pattern that the generator created generally to all of the images when training

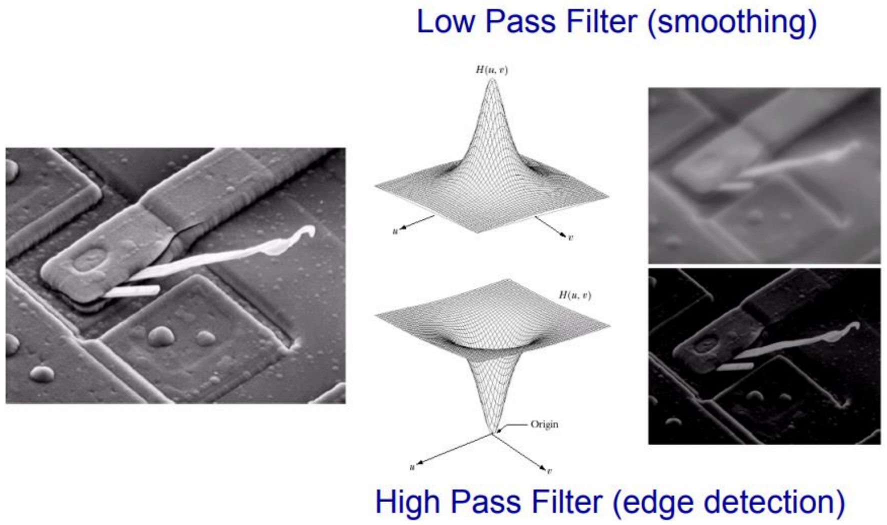

Low Pass Filter (smoothing)
H(u, v)
u
v
H(u, v)
Origin
u
v
High Pass Filter (edge detection)

Figure 8. The representation of low-pass filter and high-pass filter.

IV. Experiments

1. Preparation:

We decided to train all networks on NVIDIA GeForce 1650RTX GPU and total 25 thousand images from the mflickr dataset [5] since it is a problem of upgrading image resolution for general love (general high-end AI) that is not in domain-specific field but a variety of ones. Then, resize image from folder to 2 files with low resolution LR file with size 64x64 and resolution HR file with size 256x256, all of them has RGB color channels, represented in Figure 9. Since these data is random with little correlation in structure, it is difficult for the model to exactly access the data's evaluation, so we may not need to validate with the test.

To keep arbitrary size unchanged, we will calculate the ratio between the length and the width of low-input image, then compare the larger dimensional length of the image with the close number divisible by 64. After choosing the number that is divisible by 64 approximate with the largest dimension of the input image, we will pad both dimensions of the image with zero padding to that choosing number. After that, we divide the image into sub-batch images with 64x64 to put into the generator and this generator provide these upscaled sub-batch images. Finally, we combine them to corresponding with each other and remove the original padding by multiplying by the ratio, which was calculated in the original process.

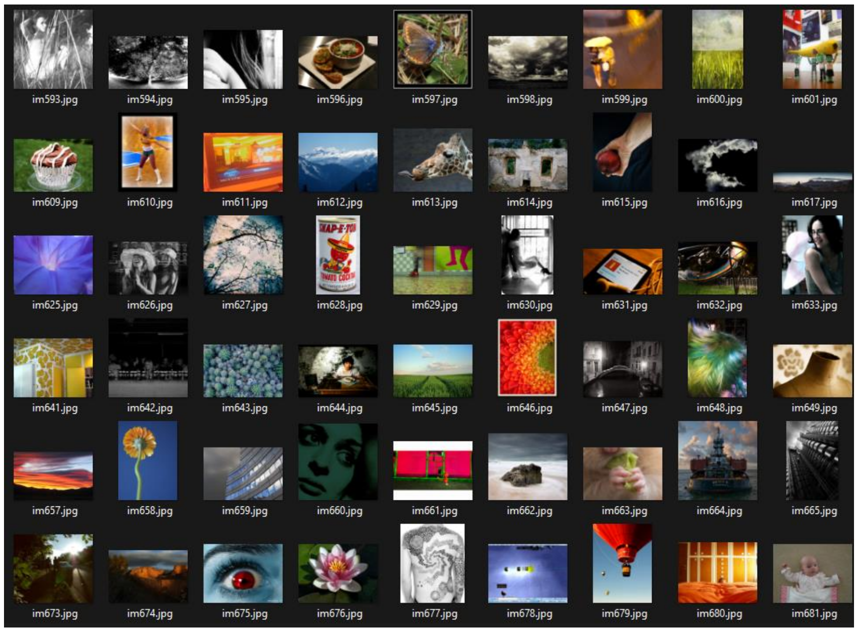

im593.jpg im594.jpg im595.jpg im596.jpg im597.jpg im598.jpg im599.jpg im600.jpg im601.jpg
im609.jpg im610.jpg im611.jpg im612.jpg im613.jpg im614.jpg im615.jpg im616.jpg im617.jpg
im625.jpg im626.jpg im627.jpg im628.jpg im629.jpg im630.jpg im631.jpg im632.jpg im633.jpg
im641.jpg im642.jpg im643.jpg im644.jpg im645.jpg im646.jpg im647.jpg im648.jpg im649.jpg
im657.jpg im658.jpg im659.jpg im660.jpg im661.jpg im662.jpg im663.jpg im664.jpg im665.jpg
im673.jpg im674.jpg im675.jpg im676.jpg im677.jpg im678.jpg im679.jpg im680.jpg im681.jpg

Figure 9. The 54 random images of the mflicker dataset. []

This whole mathematical basis is converted from unsupervised to supervised learning by having a resolvable distinction aka the discriminator to instruct the generator how to output the correct answer.

Since we train on a very limited hardware and pretrained model are not an option as it defeats the purpose of our self-build model, we decided to train on 5-10k images, a subset of our total 25k images in order for a fast conclusion to any adjustment of our generator model. We call this method the continuous update process.

After this phase and we are satisfied with the generator model, we used the total 25k images in our dataset with a longer training period. We train a total of 12 epochs with each epochs training time is approximate 4 hours and a half, which bringing our total training time of the entire 12 epochs roughly 51 hours.

While in the process of testing, we found out our final trained weights has many problems that still need to be solved, these problems can't simply be resolved by retraining the model but, but for the input and the result outside the process. That's why we choose to some external methods directly on the output image to enhance more pleasing result. As mentioned in the method, we use FFT for pattern filtering, match histogram for solving the uncertainty of the generator with color, symmetric padding for an even merged final image without border.

2. Result:

Symmetric padding:

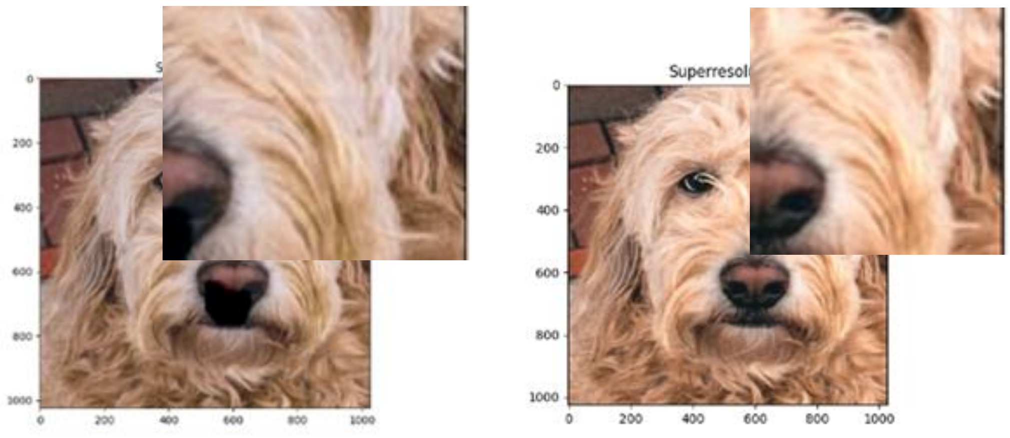


Figure 10. No longer exists the cutting lines when applying symmetric padding

With Fourier transform:

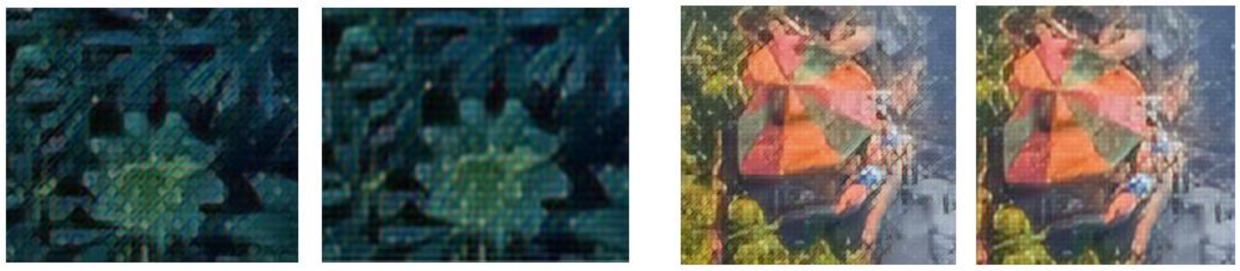
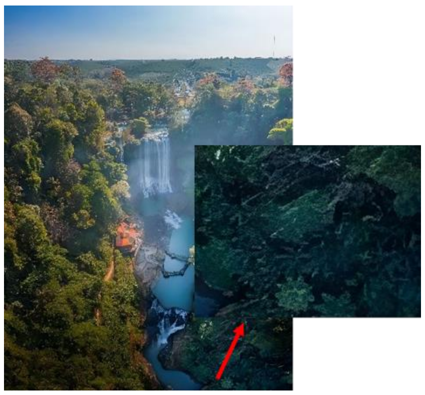
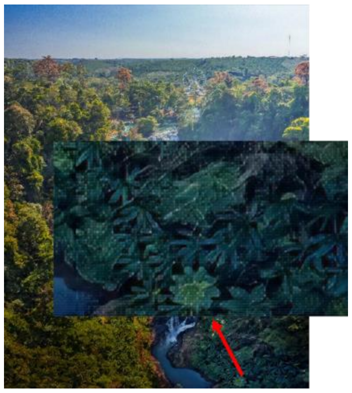


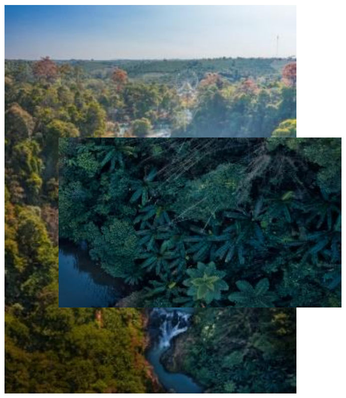
Figure 11. Applied the low-pass filter which resulting the square are lighter.

Comparisons to another AI model deploy on the web:

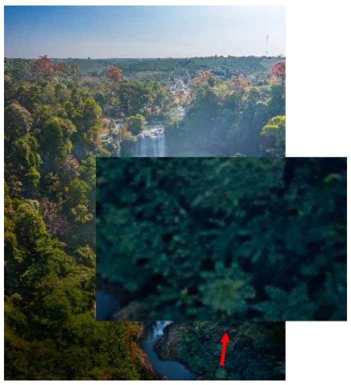
Upscale media (https://www.upscale.media) :

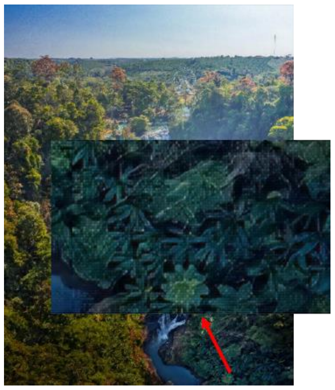

Upscale media image

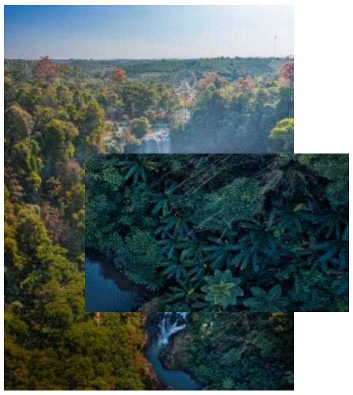

Ours


Origin

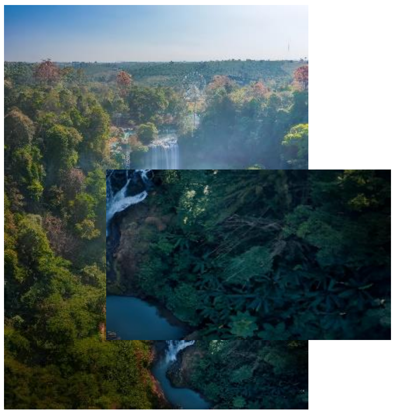
AnyMP4(https://www.anymp4.com/image-upscaler) :

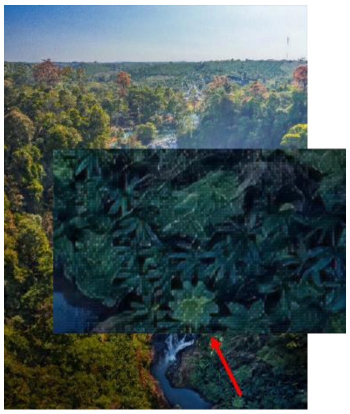

AnyMP4 Image

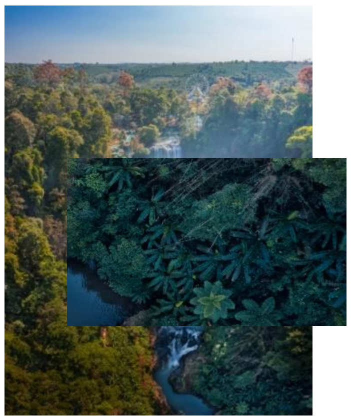

Ours


Origin

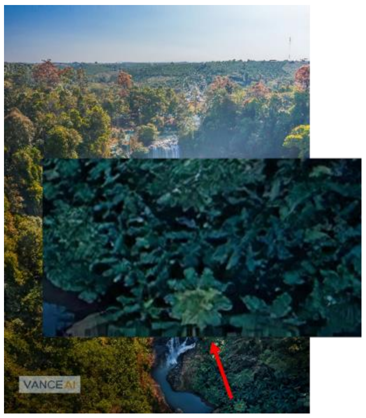
ACV.AI(https://avc.ai/#/upscale-image) :

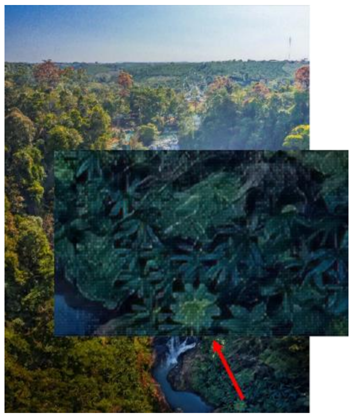

AVC.AI image

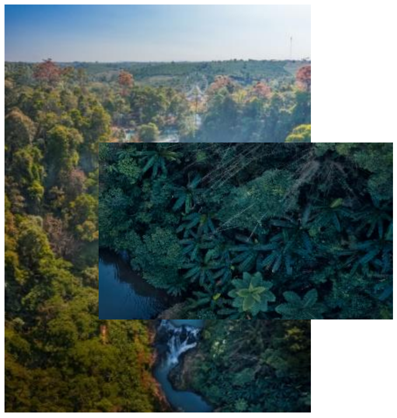

Ours


Origin

Vance AI (https://vanceai.com/image-enlarger/)

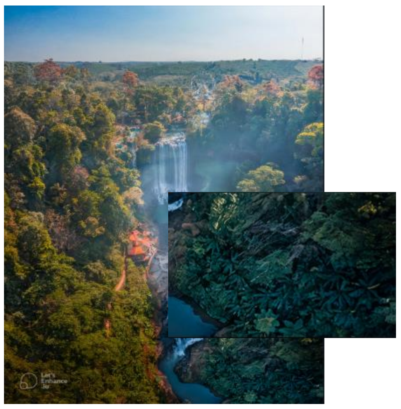


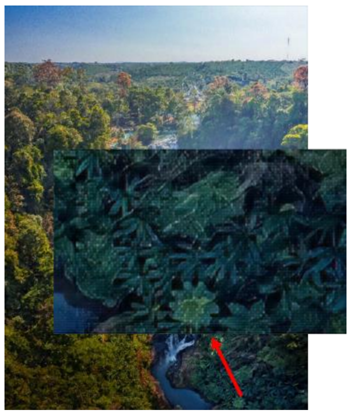


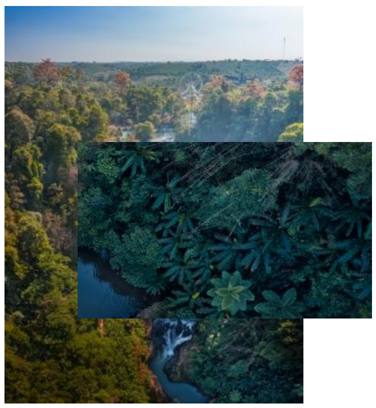


VanceAI image Ours Origin

Let Enhance (https://letsenhance.io/boost)

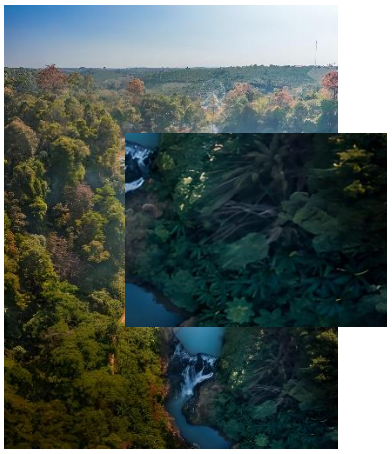


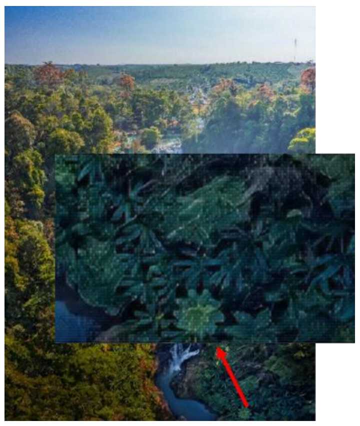


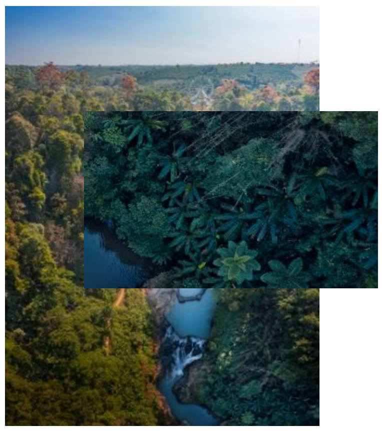


LetEnhance.io image Ours Origin

Zyro(https://zyro.com/vn/cong-cu/phan-mem-tang-chat-luong-anh) :







Zyro image Ours Origin

V. Future work

Due to the limitations of the data and hardware, our tendency in the future is having a preparation of better foundations such as 32GB - RAM or state-of-art GPU... to accelerate the speed better. Except for that, we will be improving the details more and more clearly by trying many denoising Fast Fourier Transform filters as well as apply the gradient to the Fast Fourier Transform method to the image. Moreover, we will plan to using the particular evaluation such as PSNR or other evaluated tool that give our more details how our model is good or bad.

VI. Conclusion

We have published a reimplemented SRGAN that has been optimized at the present as well as parallel limitation section. In the future, we will continue to improve these tendencies in section 5 so that our model will always be state-of-art status. We have confidentially confirmed our model by supporting lots of sub-evaluated methods giving better results and more photo-realistic than other algorithms and almost other models.

References

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 3, 4

[30] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In European Conference on Computer Vision (ECCV), pages 630–645. Springer, 2016. 3, 4

[34] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 3, 6, 8

[9] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In European Conference on Computer Vision (ECCV), pages 184–199. Springer, 2014. 3, 6, 8

[10] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(2):295–307, 2016. 3, 5, 7

[33] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super- resolution. In European Conference on Computer Vision (ECCV), pages 694–711. Springer, 2016. 2, 3, 4, 5, 7

[5] J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. In International Conference on Learning Representations (ICLR), 2016. 2, 3, 5

Rong Song et al papers [2].
Dori Shapira et al. [3]
by Ann Maria John et al. [4],
the mflickr dataset [5]

[13] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In Advances in Neural Information Processing Systems (NIPS), pages 658–666, 2016. 3

[32] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML), pages 448–456, 2015. 3, 4, 6

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014. 3, 4, 6

[42] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond means square error. In International Conference on Learning Representations (ICLR), 2016. 3

[48] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1874–1883, 2016. 3, 4, 5, 6, 7, 8

[28] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In IEEE International Conference on Computer Vision (ICCV), pages 1026–1034, 2015. 4