Generating Photo-Realistic Super Resolution Images With Generative Adversarial Network

Shoushun Li
sl138@illinois.edu

Rohan Patel
rohanp7@illinois.edu

Haiming Zhang
haiming2@illinois.edu

Jiayang Zheng
zheng68@illinois.edu

Abstract

Super Resolution has been one of the most well researched topics in the field of Computer Vision for some years. Despite the groundbreaking accuracy and speed of recently proposed models, most of modern approaches seem to focus on the aspect of tweaking object loss functions to better aid deep learning models to perform better single-image super resolution. Most of these models focus on minimizing reconstruction error that consists of mean squared error. Although such approach often results in high peak signal-to-noise ratios, it is easy to observe that the generated images lack certain high frequency details that are prevalent in natural high resolution images. As an effort to solve this problem, SRGAN[9] was proposed in 2017 to utilize generative adversarial network to generate high frequency details from low resolution images. Although SRGAN[9] reports good metrics and image qualities in generating of 4× up-scaling factor, we found it struggling at 8× up-scaling factor. As a result, we perform further modifications upon the original SRGAN architecture: add additional loss to objective loss functions to help the model train with specific emphasis on high frequency details, changing Generator Architecture from Residual Blocks to Residual in Residual Dense Blocks [11] and adding High Frequency features to Discriminator to make Discriminator more powerful.

1 Introduction

Although image resolution has improved for smartphone cameras, there still exists a large amount of images that were taken using lower resolution cameras. Additionally, images are often down-sampled to meet constraints such as internet speed and limited memory storage. Using super resolution, we hoped to improve the quality and resolutions of low resolution images to produce high resolution images. Generative Adversarial Networks can be utilized to generate synthetic images from learned distributions, and we felt it would be promising to attempt to utilize the Generative Adversarial Network's ability to generate artifacts to infer details that are missing when up-sampling a low resolution image. The goal of our project was to implement a Generative Adversarial Network (GAN) trained on the Div2K dataset to perform single image super resolution at 8× up-scaling factor. GAN, or generative adversarial network, consists of two components: discriminator and generator. In terms of Super Resolution, the generator is given low resolution images and produces high resolution images. The high resolution images are passed to the discriminator which will try to identify whether the given image is ground truth or an image synthesized by the generator. The loss function's output from the discriminator will be passed to the generator. The Generative Adversarial Network gets its name because the discriminator competes with the generator when it attempts to distinguish synthesized images from ground truth images while the generator attempts to fool the discriminator by generating realistic up-sampled images. Our desired final output is an image that contain the same high resolution as the image before down-scaling. We hoped to at the minimum produce up-sampled images that are of higher resolution than the down-sampled images fed as input to the GAN, and our ideal goal is to produce the original images from the down-scaled images using our model.

After conducting research on the various Generative Adversarial Networks utilized by researchers to perform super resolution, we decided to use the architecture laid out in SRGAN as a baseline for improvement [9]. As mentioned in the SRGAN paper, most super resolution methods currently employed are built upon the idea of enforcing pixel similarity by using MSE to maximize peak signal-to-noise ratio (PSNR). However, super-resolved images enforcing pixel similarity does not seem to be perceptually realistic to natural high resolution images. Although SRGAN has made improvement upon previously existed works, taking a closer inspection into the images it generated shows that SRGAN can still be improved. Although SRGAN did well on 4× up-scaling factor, we observe its performance drop significantly when 8× up-scaling images is involved. We wanted to address this drop in performance by exploring possible modifications we can make to SRGAN such that we can transform a low-resolution image to a high-resolution image with as little perceptual difference as possible. Bearing this goal in mind, we researched possible methods for improving on SRGAN and found several options.

2 Related Work

SRGAN. SRGAN [9] is a model based on generative adversarial networks aiming to perform single image super resolution. The model consists of two parts, a generator and

a discriminator. The generator performs the super-resolution on the input images, while the discriminator attempts to distinguish between the ground truth high resolution image and the high resolution image generated by the generator. The discriminator encourages the generator to create high resolution images that are indistinguishable from the ground truth images. What sets SRGAN apart from other Super Resolution models is that Ledig et. al. [9] proposed a perceptual loss using high level feature maps extraction of the VGG network. The main idea behind the perceptual loss is that deep pre-trained neural networks encode some perceptual information that is essential to classification. Forcing the generated image to be perceptually similar to the real image would benefit the model by producing images beyond pixel similarity [9]. The generator learns to create images that fool the discriminator while the discriminator learns to distinguish generated images from the ground truth. Architecturally, the generator consists of residual blocks utilizing convolution layers, batch normalization and Parametric Leaky ReLU units. The discriminator contains convolution blocks, each of which contains convolution layers, batch normalization and Leaky ReLU units. Ledig et. al. [9] also states that they follow inspiration from another paper and refrain from using max pooling layers. The perceptual loss function used in the paper for the generator network is an improvement on a previous work by Johnson et al. [7] and Bruna et al. [2]. Perceptual loss is formulated as a weighted sum of content loss and adversarial loss, where the content loss is the L2 distance between the feature representation of the ground truth and the generated image to capture perceptual similarity and the adversarial loss is utilized to encourage the generator to create realistic super resolution images to fool the discriminator.

3 Proposed Approach

In this part, we discuss the modifications that we make to the SRGAN [9] baseline model. In general, our changes can be divided into changes made to the generator, the discriminator, and the object training loss. In the following paragraphs, we will discuss in detail what we have achieved in terms of these three aspects.

3.1 SRGAN

We start our implementation with what is congruent with the original SRGAN as Fig1 [9]. Low-resolution (LR) images are simulated by down-sampling the original high-resolution(HR) images and then applying bi-cubic interpolation to re-upscale. Pairs of HR and LR images are fed into the network. Generator uses the LR images as the input and outputs the super-resolution(SR) images with the goal to fool the discriminator that is trained to distinguish the real HR images from the fake SR images. Following the GAN procedure [3], we train the generator and discriminator in an alternating way to solve the adversarial min-max problem:

$$min_G max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \quad (1)$$

In addition to solving the min-max problem of the generator and discriminator, we also have additional losses mentioned in the Related Work section that aim to improve the quality of the generated images.

3.2 Total Variation Loss

As evident in SRGAN[9], the loss function used to train generator consists of perceptual loss function, content loss and adversarial loss. However, we observe that SRGAN model may learn certain pattern and texture from one training example and then broadcast it into another SR image. Thus, the generator produces texture and artifacts that are unrealistic and repetitive. On this basis, we introduce the total variation Loss, which is often used in Style Transfer, to alleviate this broadcasting of texture and artifacts problem.

$$V(y) = \sum_n |y_{n+1} - y_n| \quad (2)$$

In particular, total variation loss is the sum of the absolute differences for neighboring pixel-values in the input images. It can be used to denoise the image by minimizing the gradients of adjacent pixels, hereby helping reduce boxy artifacts and unrealistic textures. With the help of total variation loss, the generated SR images are smoother and have less fake textures or artifacts.

3.3 Residual in Residual Dense Block

The original SRGAN generator consisted of residual blocks from ResNet and Upsample module that consisted of Pixel Shuffle, which increased the feature map dimension at the cost of reducing channels. In both of such modules, batch normalization is heavily utilized in the hope of facilitating training and providing better results. However, as pointed out by [11], removing batch normalization has already been proven that it can not only improve performance of the generator but also reduce computational complexity as batch normalization is quite expensive in terms of computation. Moreover, batch normalization keeps a running mean and variance of the training dataset, which might not be a good idea when the testing dataset is drastically different from the training, especially when batch size is relatively large. In terms of the generator, such phenomenon often manifests itself as artifacts and poor generalization ability [11]. Now, we introduce our main changes to the SRGAN architecture in our implementation. Instead of using regular Residual Blocks from ResNet, we replace them with Residual in Residual Dense Block (RRDB) [11], where each block is totally free of batch normalization and only consists of Convolution layers and Leaky ReLU units. Inside each RRDB block, the

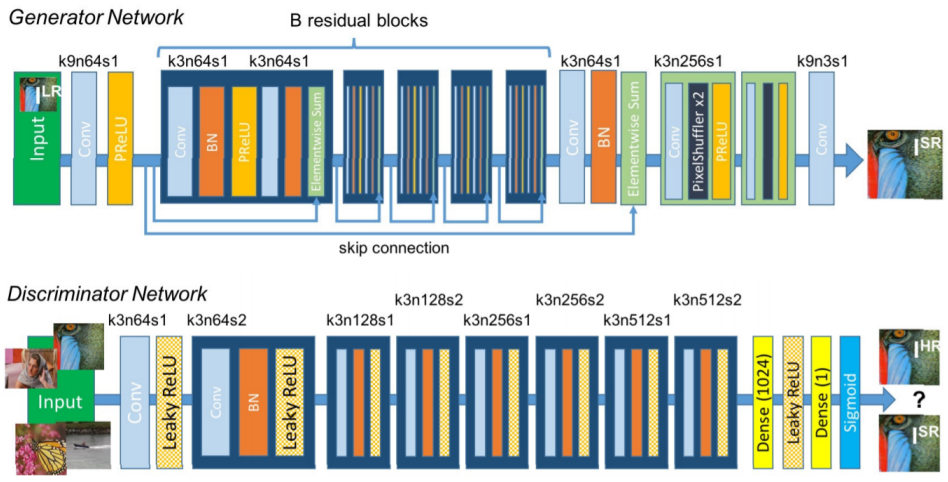

Figure 1. Architecture of Generator and Discriminator Network in SRGAN

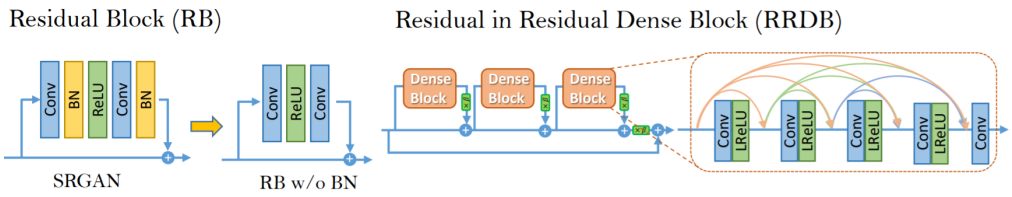

Figure 2. Left: Structure of Residual Blocks contains convolution layers, batch normalization (which we remove), and nonlinearity; **Right:** residual in RRDB are densely connected to the subsequent blocks for better information propagation and is finally added to the model output with residual scaling.

outputs are concatenated channel-wise and residual is added at the very end of RRDB blocks with residual scaling, which aims to scale down residual to prevent training instability in GANs. Empirically, RRDB blocks could potentially outperform regular Residual Blocks due to the fact that each block in RRDB will receive all the information processed by all of its predecessors. In this way, the RRDB more or less resembles the architecture of DenseNet [5], which has proven to outperform ResNet of relative depth on Image Classification Tasks. In addition to changing the Residual Block architecture, we also replaced the Pixel Shuffle in SRGAN with bi-linear interpolation. Empirically, Pixel Shuffle performs up-sampling just as well as bi-linear interpolation. However, Pixel Shuffle will reduce the amount of channels in the resulting feature map. We fear that this reduction in channels may have caused the generator to lose some of the vital spatial information regarding generating high frequency details. Last but not least, we scaled down the initial weights after initializing it with Kaiming initialization [4] as suggested in [11]. We have observed that scaling down the initialization facilitates the generator's training process. The author in [11] claims that this is due to the initial parameter's variance becoming smaller with the scaling down.

3.4 Relativistic Discriminator

As introduced in SRGAN [9], the discriminator largely takes its architecture from VGG [10] only replacing the normal ReLU layer with Leaky ReLU and eliminating all the max pooling layers in the network. As the discriminator is a vital component of the GAN network and is directly connected to the final result of our model, we developed another discriminator of similar architecture but with more depth and more parameters. Our discriminator contains about two times the parameters contained in the original SRGAN discriminator implementation. As a result, our discriminator is more sophisticated and could potentially help the generator to generate more detail-oriented images. Besides the increased number of parameters in the discriminator, we also adapted a relativistic approach to train the discriminator during the training process [8]. During the normal training process for the discriminator, the discriminator is trained with Binary Cross Entropy loss to make a prediction on whether the given image is real or generated. However, relativistic discriminators are trained to predict that a real image is more realistic than the generated fake image and vice versa. We can then write the adversarial loss using relativistic discriminator as the following:

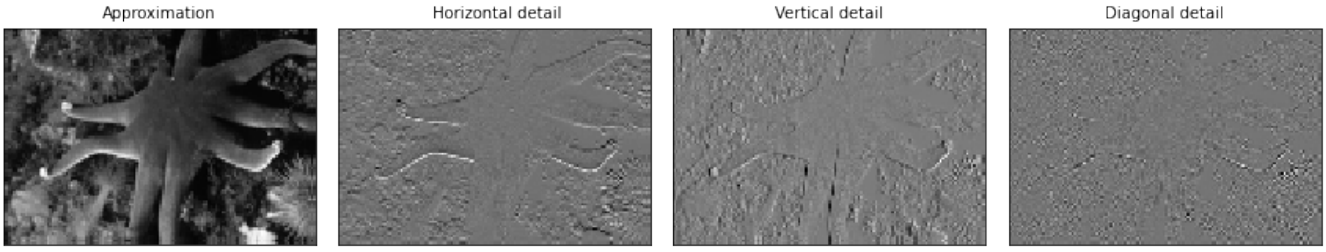

Figure 3. discrete wavelet transform

$$L_D^{RA} = -\mathbb{E}_{(x_r,x_f)\sim(\mathbb{P},\mathbb{Q})}[\log(\sigma(C(x_r) - C(x_f)))] \quad (3)$$

$$L_G^{RA} = -\mathbb{E}_{(x_r,x_f)\sim(\mathbb{P},\mathbb{Q})}[\log(\sigma(C(x_f) - C(x_r)))] \quad (4)$$

Where x_r denotes the real image, x_f denotes the fake image, $C(\cdot)$ denotes the raw output from the discriminator after taking in the input image, and σ denotes the Sigmoid function that transforms a raw output into the range $[0, 1]$.

As can be observed from the equation above, one of the reasons for using relativistic loss to train the discriminator and generator is that gradient back-propagation will be coming from both the real image and synthesized data produced by the generator. Empirically, we observed that this relativistic loss urges the generator to generate more high frequency details such as edges and sharp boundaries that are lacking in SRGAN generated images.

3.5 Dual Discriminators

Because we want our model to generate high frequency information when producing super-resolved images, we think it would be a good idea if our discriminator could utilize high frequency information to guide the generator. As a result, we introduced an additional discriminator focusing on the extracted high-frequency information of images, calling it as HF-Discriminator. We apply discrete wavelet transform to reconstruct the image into four channels [6]. Three of such channels represent the high frequency features of the original image including horizontal detail, vertical detail, and diagonal detail respectively, see Figure 3. HF-Discriminator is trained to distinguish the high frequency features of HR images from the high frequency features of SR images. The generator model architecture is not changed and we take the combination of the original Discriminator loss and the HF-Discriminator loss as its adversarial loss.

3.6 Refinement Network

One of the problems with SRGAN is that it struggles to define the high frequency details that are prevalent in natural high resolution image. Since most of these high frequency features manifest themselves as edges, boundaries, and corners alike, we built an additional network appended to the end of the generator to help capture these details that might get left out. The refinement network takes as input the generated images and is trained with L1 loss to enforce pixel similarity between the generated image and the ground truth image. Since the main rationale for adding this additional refinement network is to help generate high frequency details, we utilizes Gaussian kernel to achieve this goal:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Due to the fact that Gaussian is a low-pass filter, if we continuously apply Gaussian filter to the same image with increasing σ, we can essentially obtain a Gaussian stack that records a gradually smoothed hierarchy of the images. Then we can extract a Laplacian stack from this constructed Gaussian stack by subtracting each level of the Gaussian stack by its predecessor. For L_i indicate the Laplacian image at level i we have that,

$$L_i = G_i - G_{i+1}$$

By obtaining a Laplacian stack, we were able to extract high-frequency details from the low resolution image that might not be able to be extracted otherwise. Then using the Laplacian stack as an input, we utilized several convolution layers to learn the high frequency details presented in the stack and projected all of them into the same channel dimension as the input low resolution image. Empirically, introducing high frequency details from the generated image should make the final image looked sharpened, hereby producing a more photo-realistic image.

4 Experimental Results

In this part, we will report our experimental results on the dataset DIV2K [1] along with our experiment details. Besides the baseline, which is SRGAN as described in paper [9], we have six variations combining different modifications as our ablation study in trying to find the model with the most improvement over the original baseline. We will discuss their training details and settings in the this section as well as analyze some of the problems that can be observed in the image. For fair comparison, we report two metrics for each variation, PSNR and SSIM [12]. We will also report their generated images and final validation metrics in Figure 4 and Table 1.

Metric	SRGAN	TVL	RN	HFTV	RRDB-Simple	RRDB-HF	RRDB-Complex
PSNR	20.3542	**22.6035**	19.6722	**22.5914**	19.4312	**22.1678**	20.2790
SSIM	0.3984	**0.6381**	0.3902	**0.6354**	0.4761	**0.6145**	0.5039

Table 1. Validation result with different variation of the model. The three highest scores for both metrics are bold. Higher score indicates better quality images; the name of the variations can be found in each subsection in Section 4; Note that both of these metrics compares pixel level similarity between the ground truth and the generated image, they can only reflect to an extend human perceptual quality.

As can be seen in the Figure 4, SRGAN baseline is not performing very well in our experiment despite the very photo-realistic images reported in the paper [9]. We contribute this to the difficulties of the task. In the SRGAN paper [9], the authors perform super resolution on 4× up-scaling factor while we are using 8× up-scaling factor. Given that 8× up-scaling factor could potentially destroy a lot of high frequency details, it is not surprising that SRGAN does not perform well in our experiment compared to the experiment conducted in [9].

4.1 SRGAN + Total Variation Loss

Since the original SRGAN only contains content loss and adversarial loss, we observe that the generated images often contains boxy artifacts and repetitive hallucinated texture that should not be there. To eliminate this phenomenon, we introduced total variation loss into the object loss function. This variation of the SRGAN is trained with 2 Adam optimizers both with learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Both optimizers have no weight decay and no learning rate scheduler. The model is trained with batch size = 4, a crop size of 256×256 and an up-scaling factor of 8. The validation result can be seen in Table 1 and the corresponding generated image can be seen in Figure 4. For the sake of simplicity, we abbreviate this variation as TVL.

As can be seen in Figure 4, introducing total variation loss really had a positive effect on the generated images. The boxy artifacts present in SRGAN's generated image disappeared and the color distortion is also corrected. We hypothesize that total variation loss enforced pixels to be similar to its neighbor by minimizing the absolute value of the difference. In this way, the boxy artifacts cannot present in smoother areas since they will cause a large variation error.

4.2 SRGAN + Refinement Network

As another attempt to solve SRGAN's inability to generate high frequency details, we appended an additional refinement network right after generator in hopes that the additional network can learn to generate high frequency details when the generator fails to do. This variation of the SRGAN is trained with 2 Adam optimizers on the generator and discriminator both with learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a third Adam optimizer on the refinement network with learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All three optimizers have no weight decay and no learning rate scheduler. The model is trained with batch size = 4, a crop size of 256×256 and an up-scaling factor of 8. The validation result can be seen in Table 1 and the corresponding generated image can be seen in Figure 7. For the sake of simplicity, we abbreviated this variation as RN.

As can be seen in Figure 7, the Refinement Network does seem to smooth the SRGAN generated image a bit. However, it is introducing some color distortion at certain area of the image. We suspect that these phenomenon could be caused by the fact that the Refinement Network simply has too few learnable parameters. Compared to the generator and discriminator, whose parameter size lie in millions, the Refinement Network only possesses about 40,000 learnable parameters. This insufficient amount of parameters results in an insufficient learning capacity.

4.3 SRGAN + Total Variation Loss + HF Discriminator

After solving the problem mentioned above, we introduced the additional high frequency discriminator to further improve performance of the generated edge structure. This variation of the SRGAN is trained with 3 Adam optimizers both with learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the generator and two discriminators. The weights in the adversarial loss function for the two discriminators are equal. All of them have no weight decay and no learning rate scheduler. The model is trained with batch size = 4, a crop size of 256×256 and an up-scaling factor of 8. The validation result can be seen in Table 1 and the corresponding generated image can be seen in Figure 4. For the sake of simplicity, we abbreviated this variation as HFTV. As can be seen in the Figure 4, using the high frequency discriminator, the model is able to fix the color distortion of the original SRGAN generated Images as well as producing more details. We hypothesize that since high frequency details are introduced to the discriminator, the discriminator will favorite images with highly congruent details to the ground truth. Thus, this forces the generator to take extra effort producing details to fool the discriminator, causing the final generated images to be prevalent in details.

4.4 Residual in Residual Dense Block + Simple Relativistic Discriminator + Total Variation Loss

As introduced in the Proposed Method section, we change the architecture of SRGAN from Residual Blocks into Residual in Residual Dense Block. This model consists of 16 RRDB modules and 3 Upsample modules consists of bilinear interpolation and convolution layers. This variation has a simple discriminator that contains about 5 million learnable parameters. This variation of the SRGAN is trained with 2 Adam optimizers both with learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Both optimizers have a learning rate scheduler that would decrease the learning rate by 0.5 at epoch number $\frac{TotalEpochs}{8}$, $\frac{TotalEpochs}{4}$, $\frac{TotalEpochs}{2}$, $\frac{3\times TotalEpochs}{4}$. The model is trained with batch size = 8, a crop size of 256×256 and an up-scaling factor of 8. The validation result can be seen in Table 1 and the corresponding generated image can be seen in Figure 4. For the sake of simplicity, we abbreviated this variation as RRDB-Simple.

As can be seen in Figure 4, even when the discriminator is simple and easy to fool, we can already see that the generated image contains more high frequency details than the original SRGAN generated image. We think the relativistic losses coupled with the absence of batch normalization improves the training process and results in more focus from the generator on the high frequency details.

4.5 Residual in Residual Dense Block + Simple Relativistic Discriminator + Total Variation Loss + HF Discriminator

At the basis of the above improvement, we further add the HF Discriminator while the generator structure stays the same as last subsection. The HF Discriminator has the same structure as the simple discriminator mentioned above but is used on the high frequency domain of the images. Hyperparameter settings stays the same with added optimizer and learning rate scheduler for the HF Discriminator. The weights in the adversarial loss function for the two discriminators are equal. The model is trained with batch size = 4, a crop size of 256×256 and an up-scaling factor of 8. The validation result can be seen in Table 1 and the corresponding generated image can be seen in Figure 4. For the sake of simplicity, we abbreviated this variation as RRDB-HF. Judging from both resulting images and metrics, we can say that this variation of our model is the most successful one so far. Not only does it contain high frequency details, the effect of transparent boundary around object in the RRDB models are also reduced to the point of almost invisible. We think that it is possible that adding high frequency details to the discriminator actually makes it more powerful and forcing the generator to produce better output images in order to fool the discriminator.

4.6 Residual in Residual Dense Block + Complex Relativistic Discriminator + Total Variation Loss

Similar to the 4.4, this variation has no structural difference in the generator. However, instead of having a simple discriminator that only contains 5 millions learnable parameters, this variation contains 11 million learnable parameters. Similarly, this variation of the SRGAN is trained with 2 Adam optimizers both with learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Both optimizers have a learning rate scheduler that would decrease the learning rate by 0.5 at epoch number $\frac{TotalEpochs}{8}$, $\frac{TotalEpochs}{4}$, $\frac{TotalEpochs}{2}$, $\frac{3\times TotalEpochs}{4}$. The model is trained with batch size = 8, a crop size of 256×256 and an up-scaling factor of 8. The validation result can be seen in Table 1 and the corresponding generated image can be seen in Figure 4. For the sake of simplicity, we abbreviate this variation as RRDB-Complex. In this RRDB-Simple variation, the discriminator is not as powerful as the one in RRDB-Complex.

As shown in Figure 4, the output of these two variations are about the same, they both concentrate on the high frequency details presented in the image. One drawback we observed in the generated image is that there is this transparent edge around the object if you zoom in. We think this has something to do with the Residual in Residual Dense Block. We hypothesize that since the very first block's output will be propagated to all the subsequent blocks, a lot of the low level details extracted by the very first block would get computed over and over again. This may cause the block to rely heavily on the low level details such as edges and lines. As a result, the boundaries of objects, which consists of multiple line segments, would get emphasized creating this transparent boundary around objects.

5 Conclusion

In this paper, we explore generating photo-realistic super resolution images using generative adversarial network. We observe that despite being the first model to infer 4× photo-realistic super resolution image [9], SRGAN struggles to generate good quality images for 8× up-scaling factor. Building from SRGAN [9], we introduce total variation loss function, replace Residual Blocks, and utilize high frequency features in discriminator to aid training. We are able to report that most of our methods appear to produce better images and metrics than our baseline.

6 Member Contribution

During the whole process of this project, all team member participated equally in the conceiving, realization, experimenting, and paper writing stages. We had regular weekly meetings and discuss our ideas and possible modification as a team. Each member was responsible for conducting research on the best improvements to make to the model and

presenting those research findings to the rest of the team during the weekly team meetings. Haiming was responsible for improvements to the discriminator, while Rohan and Jiayang were responsible for improvements to the loss functions and the generator. Rohan and Jiayang were also responsible for running the models and evaluating the results. Haiming was also responsible for implementing the baseline SRGAN as a starting point for our team's improvements. Each member equally contributed towards the writing paper and presentation as well. Our fourth team member Shoushun did not contribute to the paper or the presentation, because he was attempting to drop the class during that time.

7 External Sources

Our work is largely based on GitHub repositories that implement GANs and we draw most of our inspirations from our references:

https://github.com/leftthomas/SRGAN

https://github.com/Lornatang/ESRGAN-PyTorch

The implementation of SSIM metric is from:

https://github.com/Po-Hsun-Su/pytorch-ssim

Project Video: https://youtu.be/DRAIyPKewB8

References

[1] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017.

[2] J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics, 2016.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[6] J.-H. Huang, H.-K. Wang, and Z.-W. Liao. Hfd-srgan: Super-resolution generative adversarial network with high-frequency discriminator. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3148–3153. IEEE, 2020.

[7] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing.

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

[9] 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. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

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

[11] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[12] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

8 Appendix

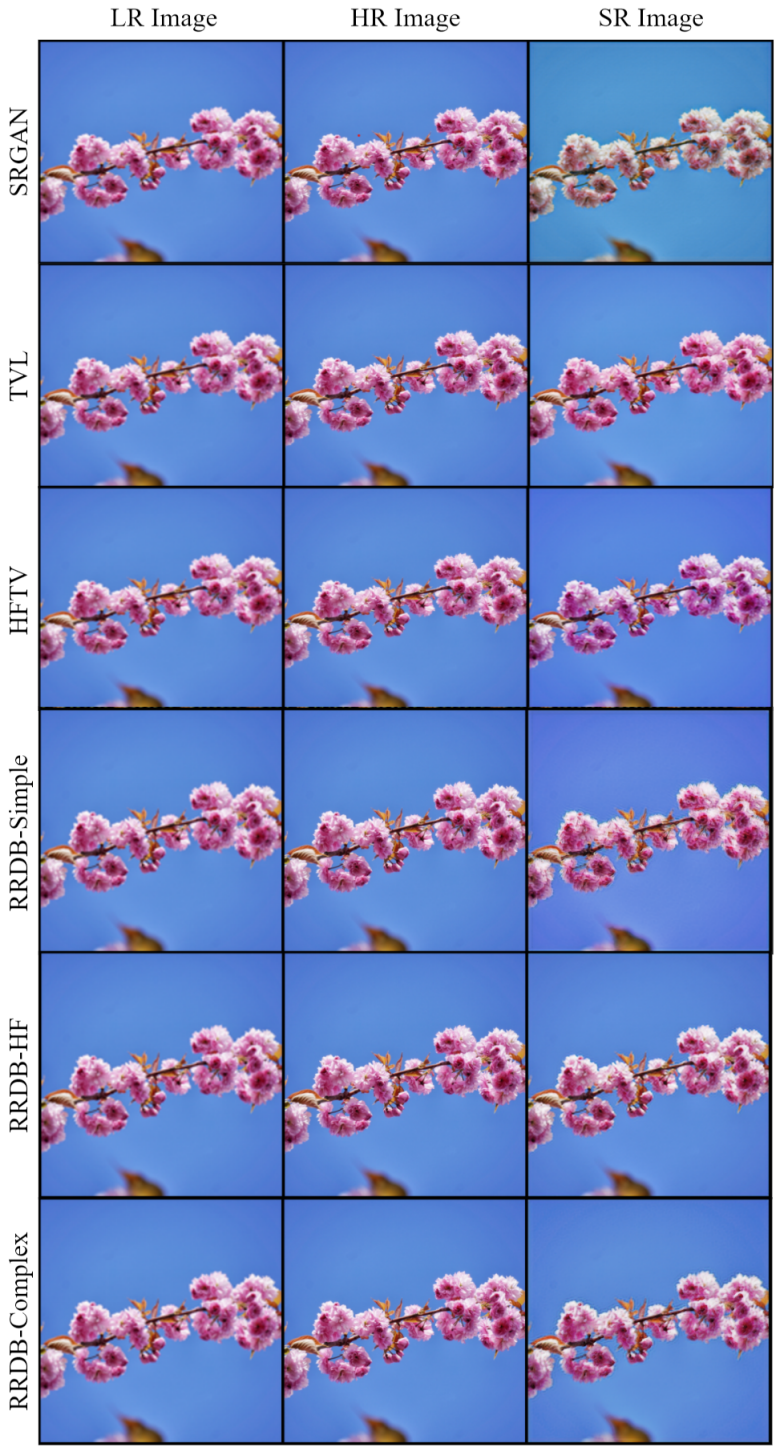

Figure 4. Final Result Containing all the variation model except for the refinement network. As can be seen the metrics of PSNR and SSIM do not faithfully reflect human perceptual difference as the one with highest PSNR and SSIM (TVL) does not seem to lack a lot of high frequency details.

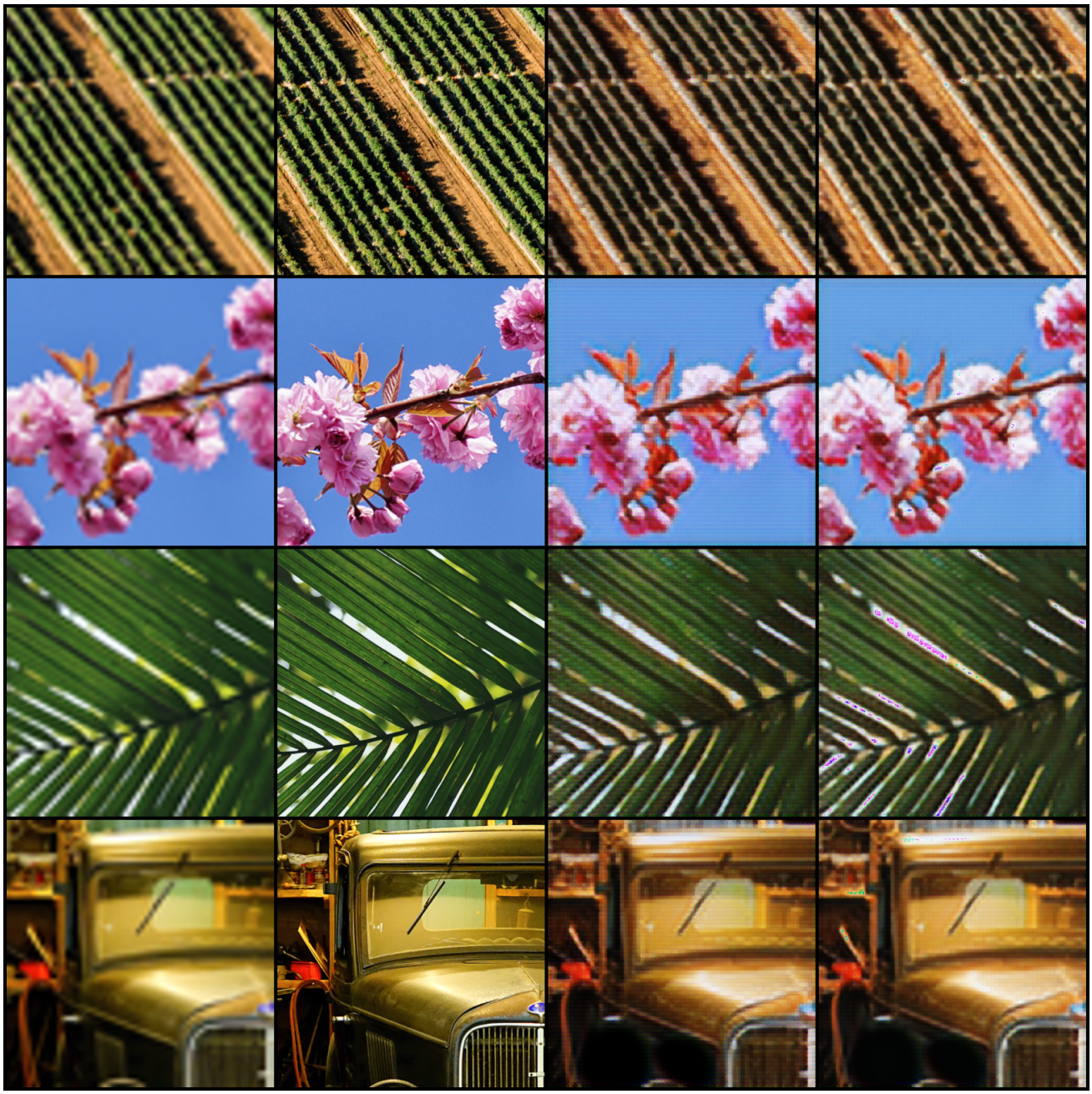

Figure 5. From Left to Right, the images correspond to low resolution image, high resolution ground truth, SRGAN baseline output, and Refinement Net output. As can be seen, despite the effort to minimize the image difference between Refinement Network output and ground truth, the Refinement Network output still does poorly.

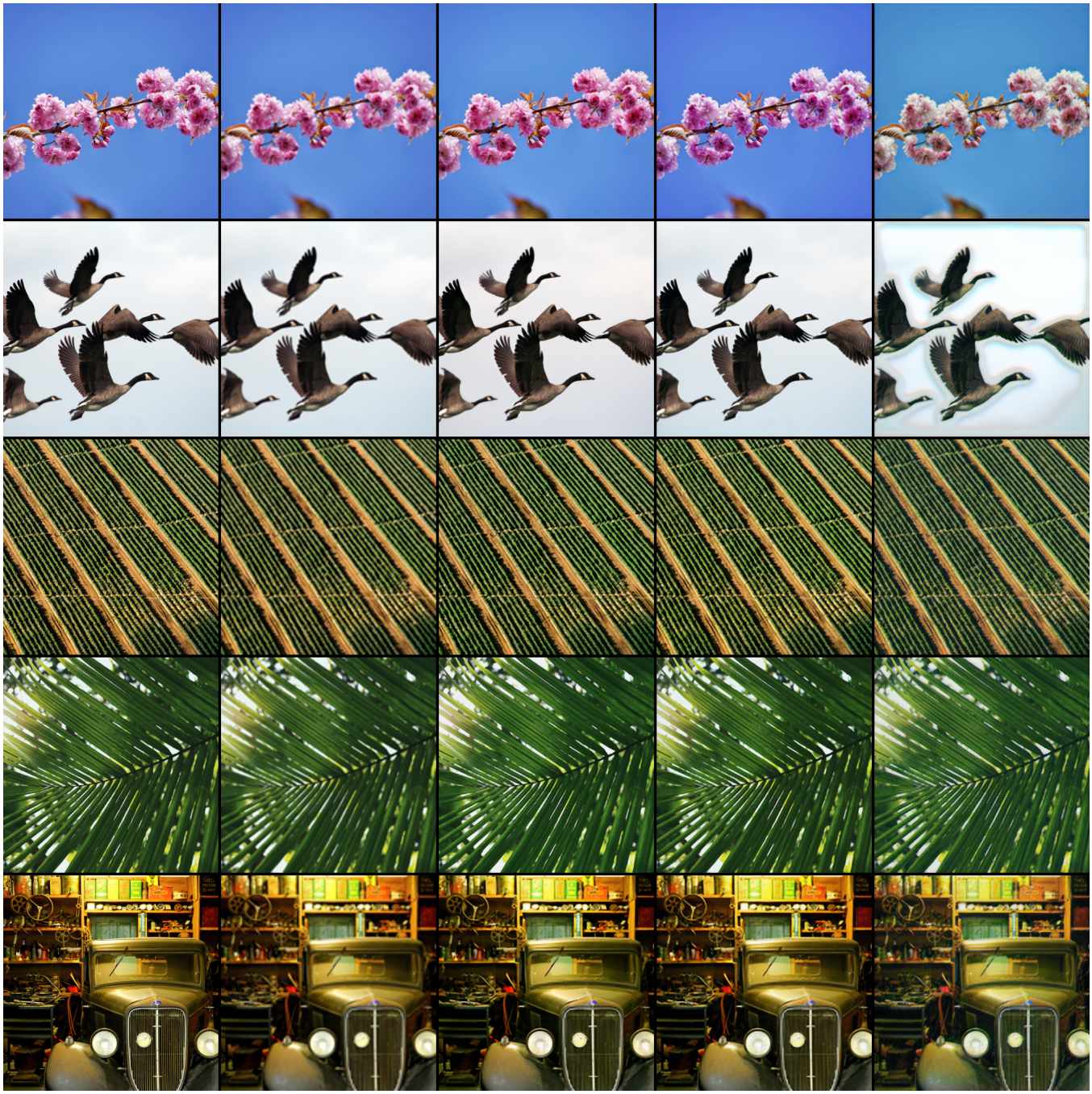

Figure 6. From Left to Right, the images correspond to high resolution image ground truth, low resolution image, SRGAN baseline output, HFTV output and Refinement Net output.

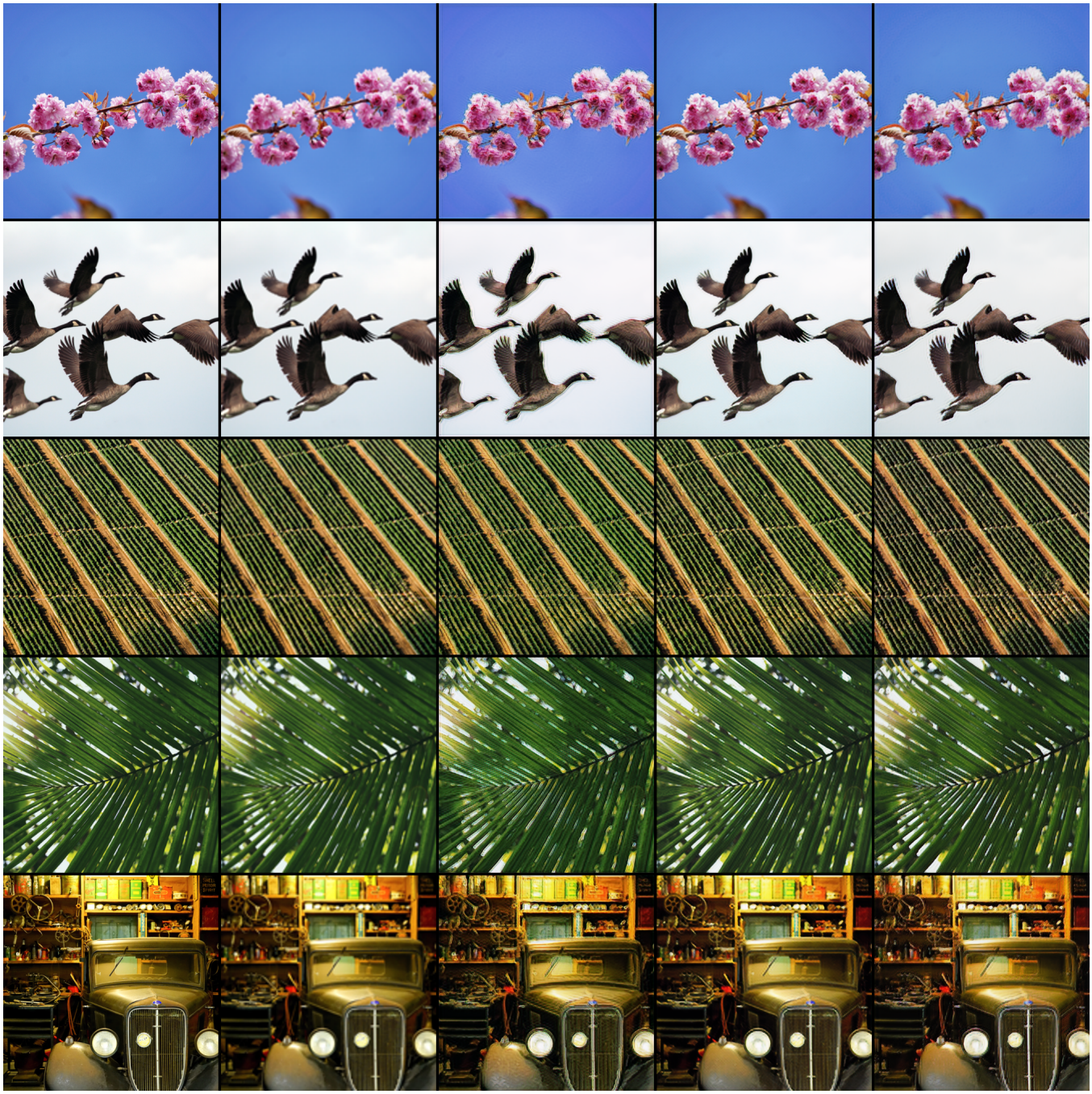

Figure 7. From Left to Right, the images correspond to high resolution image ground truth, low resolution image, RRDB-simple output, RRDB-HF output and RRDB-Complex output.