Implementing the
Real-Enhanced Super-Resolution Generative Adversarial Network (Real-ESRGAN)

Georgios Ioannou
Department of Computer Science, The City College Of New York
gioanno000@citymail.cuny.edu
ID: 23927106
https://github.com/GeorgiosIoannouCoder/realesrgan

Abstract. In the pursuit of enhancing blind super-resolution and upscaling techniques for the restoration of low-resolution images suffered by unknown and intricate degradations, many attempts have been made. However, these attempts have yet to sufficiently address the challenges posed by general real-world degraded images. The paper "Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data" **[1]** published on 17, August 2021 introduced an extension of the already existing and successful ESRGAN **[2]**, tailored for practical restoration applications. The training of the Real-ESRGAN involved the utilization of exclusively synthetic data, allowing for a more controlled learning environment. To better simulate the complexities of real-world degradations, a high-order degradation modeling process was incorporated. Furthermore, attention was given to mitigating common artifacts such as ringing and overshoot during the synthesis process. The integration of the U-Net as the discriminator with spectral normalization was a breakthrough as it enhanced the capability of the discriminator used during the training of the Real-ESRGAN and stabilized the training dynamics. Extensive comparisons across various real datasets demonstrated the superiority of the Real-ESRGAN over previous attempts. Therefore, considering the remarkable accomplishments and astonishing outcomes presented in the Real-ESRGAN paper **[1]**, it is highly advisable to allocate time and resources to study this paper more in-depth. Undertaking an in-depth study of the Real-ESRGAN paper **[1]** is not only merited but also stands as a valuable endeavor, given the substantial impact it has made in the field. Furthermore, the investment of time and effort into the detailed study of Real-ESRGAN is accompanied by the worthwhile pursuit of implementing the code to replicate and reproduce the documented results. The practical engagement in coding to recreate the outcomes outlined in the Real-ESRGAN paper **[1]** serves as a hands-on approach to understanding the intricacies of the proposed techniques. This active involvement not only solidifies comprehension but also provides an avenue for researchers and practitioners to gain firsthand experience in applying the concepts elucidated in the Real-ESRGAN paper **[1]**. Real-ESRGAN has a wide range of applications, particularly in the field of image and video processing. Applications include image restoration, image super-resolution, artifact removal, degradation simulation, resize and upscale operations, and JPEG compression.

1 Introduction

We will start by defining what super-resolution (SR) is. Single-image super-resolution (SR) constitutes an active research domain with the primary objective of reconstructing a high-resolution (HR) image from its low-resolution (LR) counterpart. The initial strides in this field were marked by the introduction of Super-Resolution Convolutional Neural Network (SRCNN), showcasing the strength of deep convolutional neural network (CNN) approaches and thus leading to significant advancements in SR. Previous blind super-resolution attempts to restore LR images suffered by unknown and intricate degradations. Degradations refer to the processes that transform a high-resolution (HR) image into a low-resolution (LR) version. These degradations are typically used to create training data for SR models. However, the degradations used in training may not perfectly match the degradations experienced in real-world scenarios, which can lead to poor performance of the SR model. Some common degradations used in training SR models include bicubic downsampling and convolution with blur kernels.

Existing strategies can be broadly classified according to the underlying degradation process. The first strategy is called explicit modeling and the second strategy is called implicit modeling. Explicit modeling found in classical degradation models utilizes blur, downsampling, noise, and JPEG compression. Implicit modeling leverages data distribution learning with Generative Adversarial Network (GAN) to discern the degradation model. While effective within training datasets, these methods encounter limitations in generalizing to out-of-distribution images.

In this project, we want to improve the effectiveness of ESRGAN **[2]** in enhancing regular low-resolution images from the real world. To do this, we will create training examples that mimic common ways images get worse in real life. Actual image problems often arise from a mix of different issues, like how cameras work, changes made to images, and sending pictures online. For instance, when we take photos with our phones, the images can have various issues like blurriness, sensor noise, artificial sharpening effects, and compression due to saving as JPEG files. After that, if we edit the photos and share them on a social media app, more compression and unexpected problems may occur.

These complexities require making the basic “first-level” breakdown model more advanced by considering “higher-level” and more precisely“second-level” breakdowns in real-world situations. This means looking at multiple instances of breakdowns happening and each instance follows the typical breakdown pattern. Moreover, sinc filters are used in the creation process to imitate the usual ringing and overshooting issues. Since Real-ESRGAN is an extension of the ESRGAN the degradation space is much larger which makes the training process difficult and the code more complex. The discriminator needs a more powerful capability to distinguish between real and complex training outputs. The gradient feedback from the discriminator needs to be more accurate for enhancing local details. Therefore, the VGG-style discriminator used in ESRGAN needs to be advanced to a U-Net design. However, the U-Net design structure and complex degradations increase the training instability. Thus, to stabilize the training dynamics, spectral normalization (SN) regularization comes to the rescue. Let us now

see how the new high-order degradation process to model practical degradations, the utilization of sinc filters to model common ringing and overshoot artifacts, and the introduction of the U-Net design create a breakthrough in the field of super-resolution.

2 Related Work

One of the first deep learning techniques that has been successfully used for super-resolution was the Super-Resolution Convolutional Neural Network (SRCNN), proposed by Dong and others **[3,4]**. This method was influential in solving problems such as face hallucination and depth map super-resolution. The SRCNN directly learns a mapping between low-resolution (LR) and high-resolution (HR) images, which makes it faster and more accurate. This is different from other learning-based methods, which need to process the bicubic-upscaled LR images. Wang and others **[5]** built on this work by replacing the mapping layer with a set of sparse coding sub-networks, creating a sparse coding-based network (SCN). This network outperforms SRCNN with a smaller model size, but it is hard to shrink the sparse coding sub-network without losing mapping accuracy. The Real-ESRGAN does not just work on the original LR image but also contains a simpler but more efficient mapping layer. This network only requires a different deconvolution layer to upscale an image to different sizes, making it faster than other methods that require a completely different network for each upscaling factor.

Generative Adversarial Networks (GANs) are often used to create visually appealing results. They guide the solutions to be closer to the natural image. However, already existing methods use a bicubic downsampling kernel, which often fails when applied to real images. There are two main existing categories of methods for blind super-resolution which both have limitations. The first category uses explicit degradation representations, which typically consist of degradation prediction and conditional restoration. These components can be performed separately or jointly. These methods rely on predefined degradation representations, such as the type and level of degradation, and usually consider simple synthetic degradations. However, if the degradation estimation is not accurate, it can result in artifacts. The second category involves generating training pairs that are as close to real data as possible, and then training a unified network to address blind SR. However, the training pairs are only constrained to degradations associated with specific cameras, and thus may not generalize well to other real images. Moreover, learning fine-grained degradations with unpaired data can be challenging, and the results are often unsatisfactory.

3 Second-order Degradation Model

By applying the already existing classical degradation model to generate training pairs, the trained model can handle some real-world samples. However, it still struggles to manage certain complex degradations, particularly unidentified noises and complex artifacts. This is due to the significant difference between synthetic low-resolution images and realistic degraded images. As a result, there is an immediate need to expand the classical degradation model to a high-order model and more specifically to a second-order model to better represent practical degradations. The classical degradation model is essentially a first-order modeling, as it includes a fixed number of basic degradations. However, real-world degradation processes are diverse and often involve a sequence of procedures such as camera imaging systems, image editing, and Internet transmission. For example, restoring a low-quality image downloaded from the Internet involves a complex combination of various degradation processes. The original image, taken with a cellphone many years ago, may contain degradations such as camera blur, sensor noise, low resolution, and JPEG compression. The image is then edited with sharpening and resizing operations, however, this introduces overshoot and blur artifacts. Digital transmission also introduces artifacts, making the process more complicated when the image is shared multiple times on the Internet.

Overshoot artifacts in super-resolution are typically combined with ringing artifacts and are usually produced by sharpening algorithms or JPEG compression. They manifest as an increased jump at the edge transition in an image, causing the image to appear sharper than it actually is.

Blur artifacts in super-resolution can be produced by various resize operations such as nearest-neighbor interpolation, area resize, bilinear interpolation, and bicubic interpolation. These artifacts can result in blurry images when the image is downsampled and then upsampled to its original size.

Therefore, to eliminate overshoot artifacts and blur artifacts there is an urgent need for a second-order degradation model as shown in Equation 1 where each D represents the classical degradation model, y is the input, and x is the output. To further assist the process of eliminating overshoot artifacts and blur artifacts sinc filters are used in the creation process to imitate the usual overshooting and blur issues. Figure 1 shows the overall process of the second-order degradation model used in the Real-EASRGAN training.

$$x = D^2(y)=(D_2 \circ D_1)(y)$$

Equation 1. Second-order Degradation Model Equation

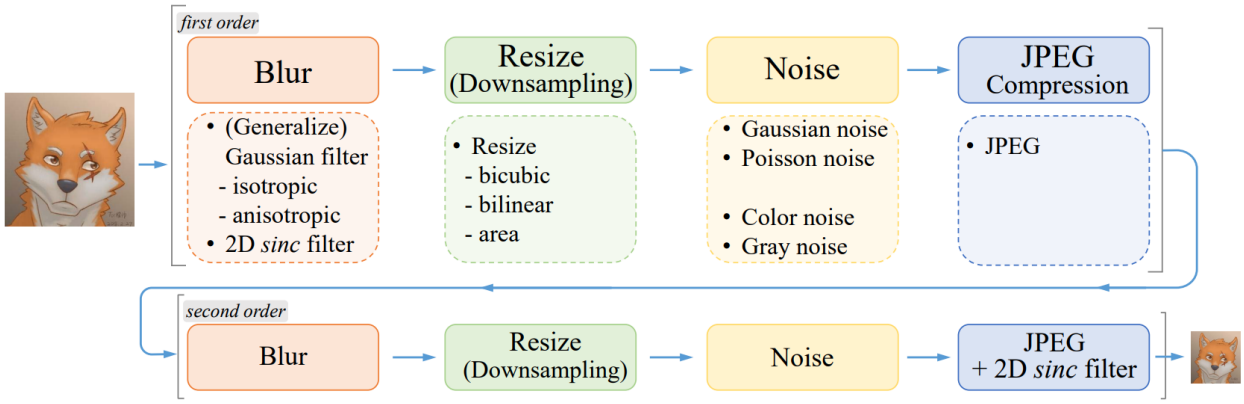

Figure 1. Second-order Degradation Model Process (2-step process)

4. Real-ESRGAN Model Architecture

In the context of super-resolution, the generator and discriminator are key components of a Generative Adversarial Network (GAN) **[6]**.

The generator's role is to produce plausible data, in this case, high-resolution images. The generated instances are used as negative training examples for the discriminator. The generator's goal is to produce output that can fool the discriminator, making it harder for the discriminator to distinguish between real and fake data. As the generator's training progresses, it gets closer to producing output that can fool the discriminator.

On the other hand, the discriminator is a classifier that tries to distinguish real data from fake data created by the generator. It uses real data instances as positive examples and the fake data instances created by the generator as negative examples during training. The discriminator penalizes the generator for producing implausible results, i.e., for producing fake data that it can easily distinguish from real data. The discriminator's goal is to improve its ability to distinguish between real and fake data, which in turn makes it harder for the generator to fool it.

Both the generator and the discriminator are neural networks and require separate training. The generator's output is connected directly to the discriminator's input. Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights. In our case, Real-ESRNET is the generator, and U-Net inspired by this paper **[7]** is the discriminator.

For this paper, we did not train the ESRGAN generator because this will require a total of three trainings instead of two as we will see after. Instead, for the Real-ESRGAN training, we used the ESRGAN model from this paper **[7]** as the generator. This helped in making the Real-EASRGAN more robust since the ESRGAN is already an effective and tested model. The ESRGAN model architecture as illustrated in Figure 2 is a deep network with several residual-in-residual dense blocks (RRDB). It is worth noting the pixel unshuffle step to reduce the spatial size and enlarge the channel size before feeding inputs into the ESRGAN model.

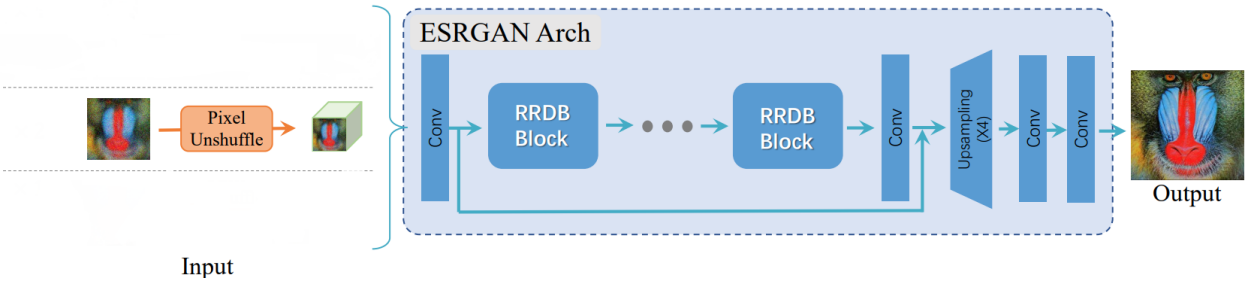

Figure 2. ESRGAN ModelArchitecture

As mentioned before, the training process for the Real-ESRGAN happens in two steps. First, a PSNR-oriented model named Real-ESRNET is trained with the L1 loss using the RRDBNET as the generator and ESRGAN as the base model. In other words, Real-ESRNET is finetuned from ESRGAN. Next, Real-ESRNET is used as an initialization of the generator to train the Real-ESRGAN with a combination of L1 loss, perceptual loss, and GAN loss. If we did not use the ESRGAN model from the paper **[7]**, then we would have to do additional training to get the ESRGAN model.

5. Experiments

To train the Real-ESRGAN model, we used three datasets. The first dataset is called DIV2K and contains 800 high-resolution (HR) images. The second dataset is called Flickr2K and contains 2,650 high-resolution (HR) images. Both of these datasets have been combined into a single dataset called DF2K and these 3,450 images consist of people, animals, buildings, and scenes. The third dataset is called OutdoorSceneTraining (OST) and is divided into 2,187 high-resolution (HR) images of animals and 2,285 high-resolution (HR) images of buildings. Therefore, the Real-ESRGAN model was trained on a total of 5,735 high-resolution (HR) images.

To train the Real-ESRNET and the Real-ESRGAN four NVIDIA A100 80GB PCIe GPUs were utilized through NERSC's exclusive GPU nodes. The training for the Real-ESRNET took three days and the training for the Real-ESRGAN took two days. Figure 3 and Figure 4 demonstrate the training for the Real-ESRNET and the Real-ESRGAN respectively. The figures clearly show the loss per pixel for each and every iteration of the training.

For both training sessions, the image size for the ground truth images was set to 256 during the training. Batch size was set to 48 and Adam was used as the optimizer. The L1 loss was used for the Real-ESRNET training. A combination of L1 loss, perceptual loss, and GAN loss was used for the Real-ESRGAN training. That is the reason why Figure 4, shows more loss per pixel values for each iteration in the training. LeakyReLU was used as the activation function. Real-ESRNET was trained with 1,000,000 iterations and a learning rate of 2×10^{-4}. Real-ESRGAN was trained with 400,000 iterations and a learning rate of 1×10^{-4}. All the

training parameters can be found in the .yml files inside the training_parameters directory of this project's GitHub repository.

Furthermore, Gaussian kernels, generalized Gaussian kernels, and plateau-shaped kernels were utilized, each assigned with a probability of 0.7, 0.15, and 0.15 respectively. The size of the blur kernel is chosen randomly from (7, 9, 11, 13, 15, 17, 19, 21). The blur standard deviation, σ, is sampled from the range [0.2, 3] for the first degradation process and from the range [0.2, 1.5 for the second degradation process. The shape parameter β is sampled from [0.5, 4] and [1, 2] for generalized Gaussian and plateau-shaped kernels, respectively. Additionally, a sinc kernel is introduced with a 0.1 probability, and there is a 0.2 probability of skipping the second blur degradation. For noise, Gaussian and Poisson noises are used with probabilities of {0.5, 0.5}. The noise sigma range and Poisson noise scale are specified as [1, 30] and [0.05, 3], respectively, and [1, 25] and [0.05, 2.5] for the second degradation process. The probability of gray noise is set to 0.4. JPEG compression quality factor ranges from 30 to 95. Finally, the application of the sinc filter is determined with a probability of 0.8.

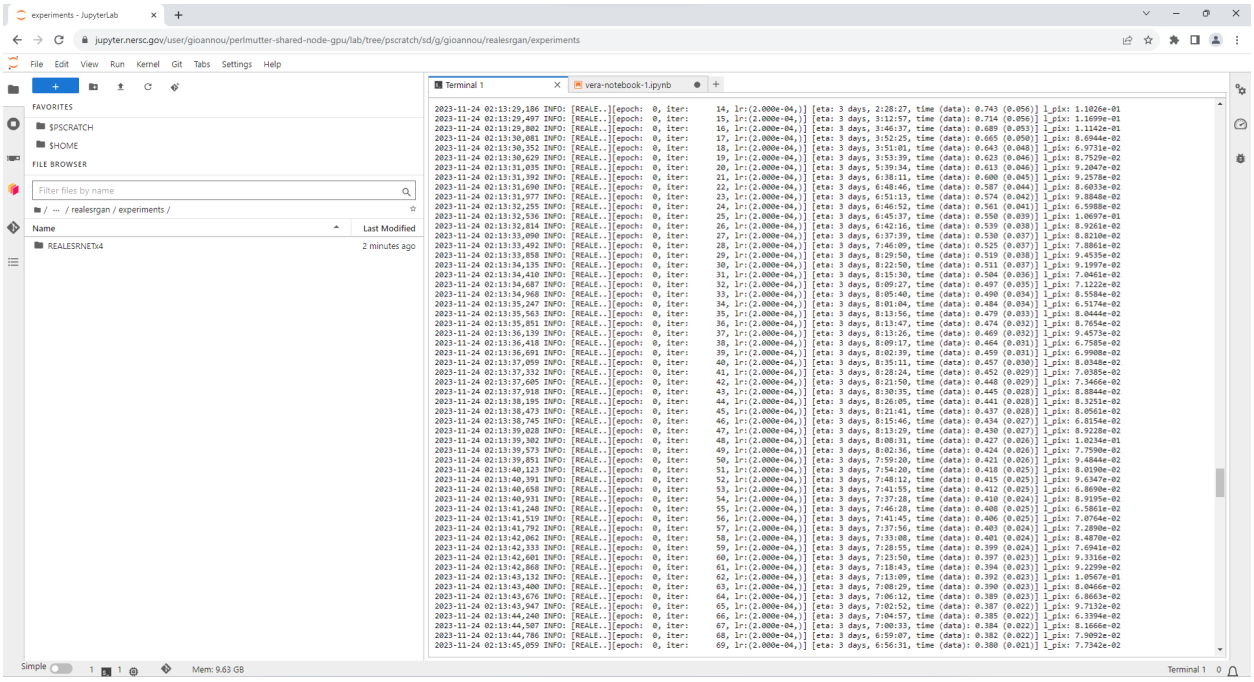

Figure 3. Real-ESRNET Training

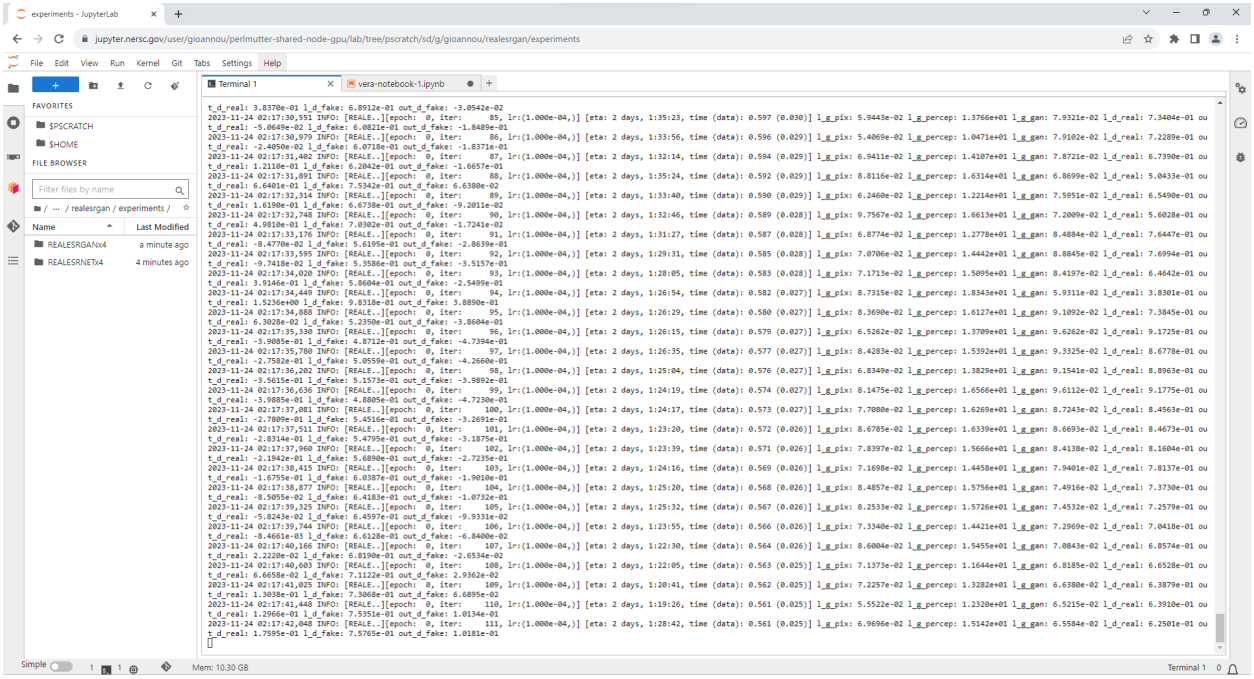

Figure 4. Real-ESRGAN Training

6. Coding Files

All the coding files (highly documented), steps on how to run them, steps on how to structure this project, and detailed video demos can all be found in this project's GitHub repository README.md file. Let us mention the purpose of each file.

The file unet_discriminator_sn.py defines the architecture of the U-Net discriminator used for training the Real-ESRGAN. The file real_esrnet_model.py defines The architecture of the generator Real-ESRNET used for training the Real-ESRGAN. The file image_crop.py crops the images of the DF2K dataset to sub-images for faster IO. The file image_path.py generates all the ground-truth image paths to be used during training. The file multiscale_df2k.py scales up and down images. The scales that are used are 0.75, 0.5, and ⅓. The files inference.py and real_esrgan.py serve as an inference for the Real-ESRGAN model. The file real_esrgan_model.py defines the architecture of the Real-ESRGAN model as described in this project and this paper **[1]**. The file data_used_during_training.py augments the datasets so that the Real-ESRGAN model can be trained on only pure synthetic data. The file training.py is used to train both the Real-ESRNET and the Real-ESRGAN models.

7. Differences From The Original Paper

This project differs slightly from the original paper **[1]** in two ways. The first difference is that this project only trained the Real-ESRGAN for a scale x4, but the original paper also trained the Real-ESRGAN model for the scales x1 and x2. This project chose to train the Real-ESRGAN model for the scale x4 because based on a previous paper **[2]**, the best tradeoff between performance and parameters is the scale factor x4. The second difference is that the original paper **[1]**, compared their Real-ESRGAN model with ESRGAN, DAN, CDC, RealSR, and BSRGAN using the testing datasets RealSR, DRealSR, OST300, DPED, and ADE20K. This project could not do that due to the difficulty of finding these datasets and the models from previous works.

8. Compare Results To Original Paper

The results from this project can be found in this project's GitHub repository README.md file as images and video demos. Since this project followed the same methodologies as the original paper **[1]**, the results were expected to be the same and as accurate. Figure 5 and Figure 6, illustrate the results of this project's model on images that are mentioned in the original paper **[1]**. Figure 7 shows this project's model performance on a building image.

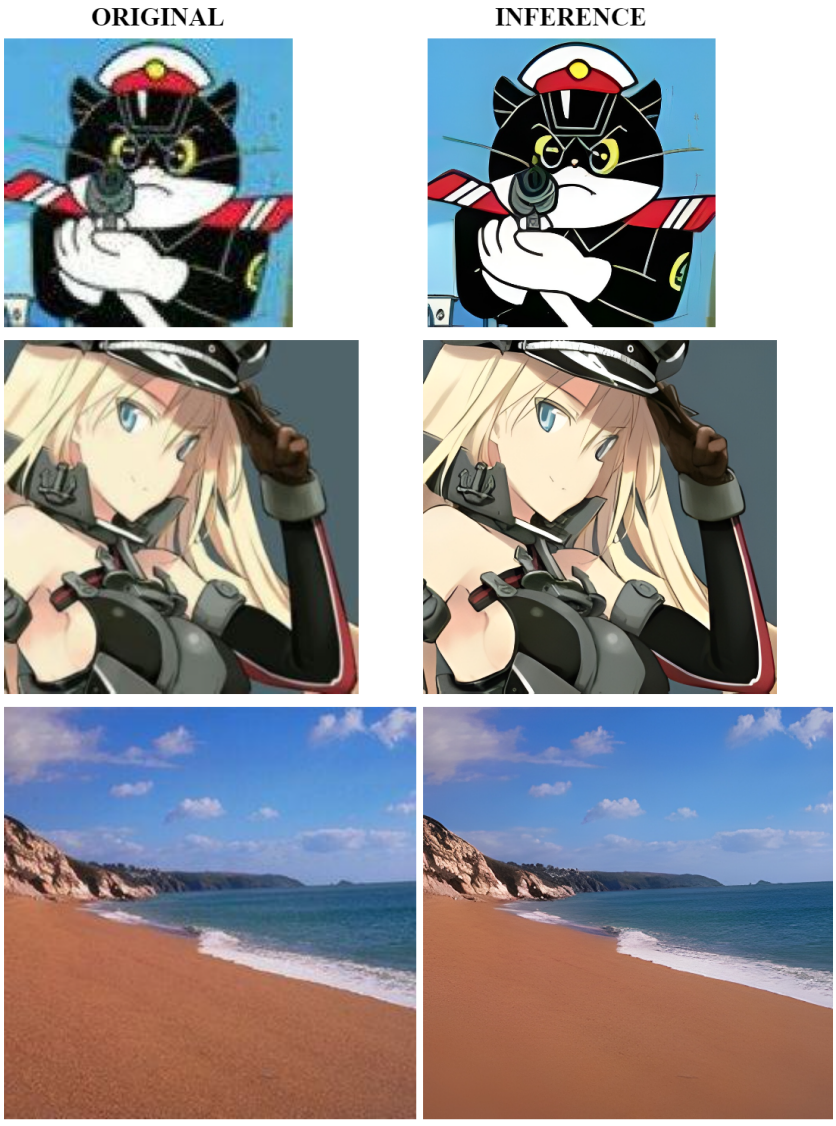

Figure 5. Inference Model

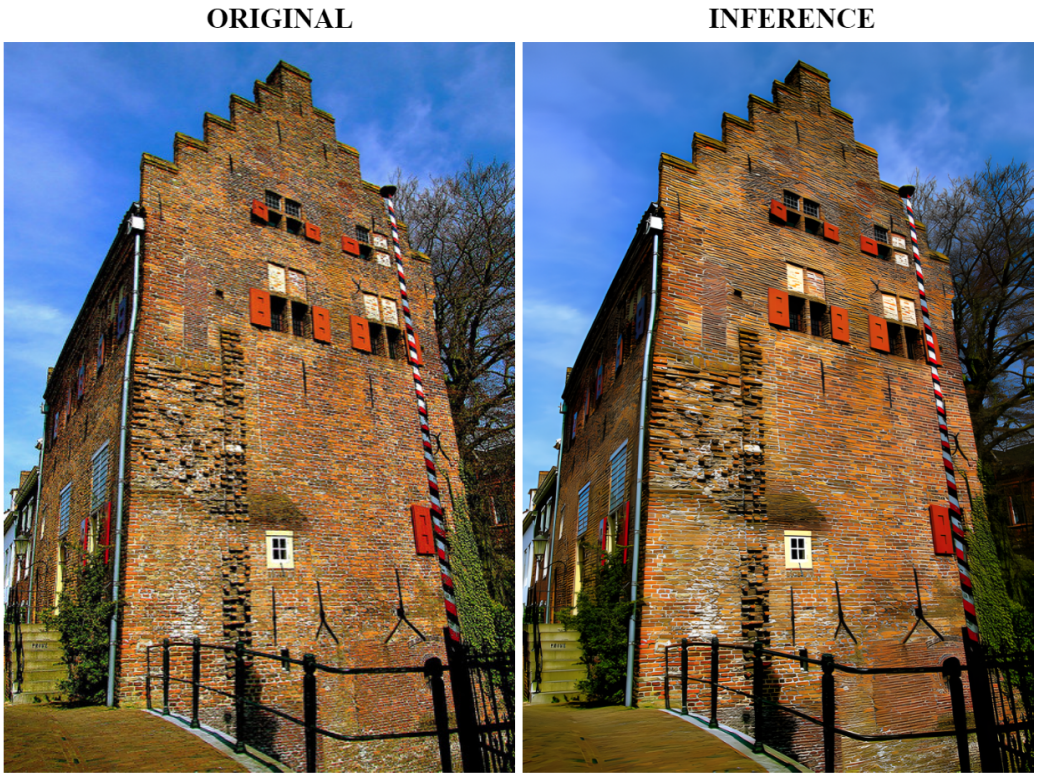

Figure 6. Model Inference

Figure 7. Inference Model

9. Conclusion

Overall, this project was an implementation of the paper "Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data" **[1]**. Through this project, we explained and coded a second-order degradation process to model practical degradations and utilize sinc filters to model common ringing and overshoot artifacts. The classical first-order degradation process is not sufficient for the state-of-the-art super-resolution models. U-Net discriminator with spectral normalization to increase discriminator capability and stabilize the training dynamics was also introduced and coded. Finally, Real-ESRGAN was trained from scratch with pure synthetic data (images generated/created and not from real-world images) coming from the three datasets of DIV2K, Flickr2K, and OutdoorSceneTraining (OST). ESRGAN is a combination of three concepts. "Enhanced" refers to the improvement in the quality of images. "Super-resolution" is a technique used to increase the resolution of an image. "Generative Adversarial Network" (GAN) is a type of AI model used for generating new data instances. The results of the Real-ESRGAN are very impressive and hard to believe. I am glad that I was able to reproduce the results and be able to work with cutting-edge technologies and have a model that I can use to enhance my images. In the future, I will retrain the Real-ESRGAN model for the x1 and x2 scale factors. Moreover, I would love to see how higher-order degradation processes such as third-class will affect the results.

References

[1] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data. In ECCV, 2021.

[2] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the Super-Resolution Convolutional Neural Network. In ECCV, 2016.

[3] Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV, 2014.

[4] Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. In TPAMI, 2015.

[5] Wang, Z., Liu, D., Yang, J., Han, W., Huang, T.: Deeply improved sparse coding for image super-resolution. In ICCV, 2015.

[6] Zihao Wei, Yidong Huang, Yuang Chen, Chenhao Zheng, and Jinnan Gao. A-ESRGAN: Training Real-World Blind Super-Resolution with Attention U-Net Discriminators. 2021.

[7] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In ECCV, 2018.