

IMAGE SUPER-RESOLUTION USING GENERATIVE ADVERSARIAL NETWORKS

RASHMI PHADNIS [RPHADNIS@SEAS], ADAM ALAVI [ADAM2318@SEAS], WILLIAM C. FRANCIS [WILLCF@SEAS],

ABSTRACT. Image Super-Resolution refers to the conversion of a low-resolution image to a high-resolution image. In this project, the implementation of ESRGAN is chosen as the baseline to generate a super-resolved image. Multiple experiments are then performed to improve upon the results from the baseline and also improve the stability of the GAN while training. It is found that when the baseline is modified to have a different set of losses and other normalizations to stabilize the training, the perceptual quality increases.

1. INTRODUCTION

Image super-resolution (SR) is the process of estimating a high resolution image from its low resolution counterpart by attempting to recover the missing information when upsampling the image. Super-resolution has uses in a wide-ranging field of applications, right from improving the quality of images in medical imaging systems to improving the quality of satellite images in astronomy. Traditionally, this has been done by finding explicit mappings between the low and high resolution images using sparse-coding. There have been greater advances in the field with CNNs finding more accurate and complex mappings[1]. More recently, high resolution images were recovered with the utilization of Generative Adversarial Networks. The experiments conducted in the project focus on improving the perceptual quality of the images obtained from the baseline model of ESRGAN[2]. We follow a two-pronged approach. On one hand, we improve upon the fundamental stability of the GAN by introducing tricks that streamline the training process. On the other hand, we improve the visual and perceptual quality of the images by changing the architecture of the Generator and Discriminator along with the losses used for both.

1.1. Contributions.

- **Improving Stability:**
 - We employ Spectral Normalization within the Discriminator to increase the overall stability[3].
 - We implement weight norm for every CNN within a Residual Block in the Generator as they have been proven to improve reconstruction quality and training stability[4].
 - We avoid sparse gradients which come with activation layers like ReLU by implementing Swish as the activation layer.
- **Improving perceptual quality:**
 - We improve upon the content loss by incorporating the Multi-Scale Discriminative Feature (MDF) loss function[5].
 - The baseline model of ESRGAN utilized only bicubic interpolation downsampling as its degradation method to generate low resolution images for training. We propose adding Gaussian Blur along with bicubic downsampling as a more realistic degeneration technique.
 - The baseline model again implemented a normal bicubic interpolation method for upsampling within the Generator. We implement PyTorch’s PixelShuffle which is essentially a sub-pixel convolution layer which learns an array of upscaling filters to upscale the final LR feature maps into the HR output[6].
 - We improve upon the architecture of the Generator by increasing the number of ”Residual In Residual Dense Blocks” in the Generator CNN.
 - We improve upon the architecture of the Discriminator by increasing the number of linear layers.

2. BACKGROUND

We define here the baseline model that we implemented so that our approach and the changes we made are easier to follow. In terms of the architecture, ESRGAN builds upon its predecessor SRGAN’s Generator with a more complex CNN model integrated which has ’Residual in Residual Dense Block’ (RRDB) as its basic block. The Discriminator is improved compared to SRGAN by employing a relativistic discriminator which tries to predict if the real image is

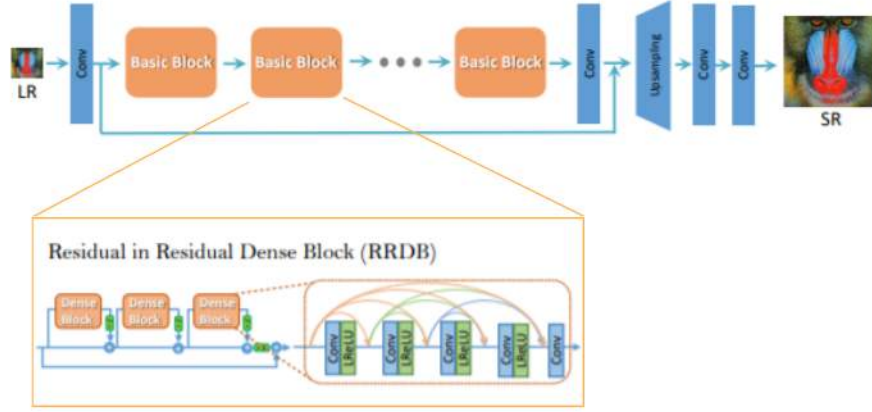


FIGURE 1. Architecture of Baseline Model

relatively more realistic than the fake image. If x_r and x_f are the real and fake images and $C(x)$ is the non-transformed Discriminator output, then the Discriminator output can be defined as:

$$D(x_r, x_f) = \sigma \left(C(x_r) - \mathbb{E}_{x_f} [C(x_f)] \right)$$

where \mathbb{E}_{x_f} is the average of the fake images for the mini-batch.

The three main losses which are used over the training process are pixel loss, adversarial loss and content loss. The pixel loss is the L1 loss between the Generator output and the real images. In the baseline model, the content loss is calculated by finding the L1 loss between the generated image and the features of a pre-trained VGG model. The discriminator loss is defined as follows:

$$-\mathbb{E}_{x_r} [\log (D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f} [\log (1 - D_{Ra}(x_f, x_r))]$$

The adversarial loss for the generator is similarly given as:

$$-\mathbb{E}_{x_r} [\log (1 - D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f} [\log (D_{Ra}(x_f, x_r))]$$

The final generator loss is a weighted sum of pixel, content and adversarial loss of the generator.

For the training process, the generator is trained first for a few epochs with only the content loss to give it a head-start and improve the PSNR. Post this, the adversarial training begins with the Discriminator training on the Discriminator loss mentioned above and the Generator on the final generator loss.

3. RELATED WORK

The field of super resolution has witnessed a variety of improvements since its inception. Many works such as [7] have used convolutional neural networks to perform super resolution. A pioneering work by Dong et al. [1] includes SRCNN which learns the mapping from low resolution image to a high resolution image in an end-to-end manner. Although a huge contribution, this results in unwanted artifacts in the super resolved images.

Recently, the field has seen some prominent changes in network architecture such as the introduction of residual dense network [8], deep back projection [9], densely connected network [10], recursive learning [11], residual blocks [12], and Laplacian pyramid structure [13]. The most notable improvements were put forth by more recent works that harness the ability of GANs to fill in the missing pixels [14].

Zhao et. al. [15] studied the visual quality of images produced by the image super-resolution using L2, L1, SSIM and MS-SSIM losses. We have adopted L1 loss in our generator as their results show sharper images using L1 loss. Our initial trials take inspiration from works such as [16] that used L2 norm between the features of the reference and test images extracted from the VGG network as a loss function to train a super-resolution GAN. Using such feature-based losses have drawbacks: they are computationally expensive, require regularization, and use a large network trained on an unrelated task. Hence, we replaced it with Multi-Scale Discriminative Feature (MDF) loss [5] which consists of discriminators that are trained to detect and penalize the presence of task-specific artifacts. Miyato et al. showed in their work [3] that Spectral Normalization generated images of better image quality as compared to previous stabilization techniques. We implement spectral norm in our discriminator.

The most notable works that boasts cutting edge performance in super-resolution using GANs include SRGAN [17] and ESRGAN [2]. SRGAN incorporated a perceptual loss function which consists of an adversarial loss and a content loss to build was the first framework capable of inferring photo-realistic natural images for 4 \times upscaling factors. ESRGAN introduced the Residual-in-Residual Dense Block, outperforming SRGAN in sharpness and details. They also borrowed the idea from relativistic GAN [18] to let the discriminator predict relative realness instead of the absolute value. The generated pixels, however, come with unpleasant artifacts that impair the visual quality of the images. In this work, we propose a higher order degradation model with an improved discriminator and generator architecture that are capable of generating more visually appealing super-resolved images as compared to the existing GANs.

4. APPROACH

To set up a baseline for our project, we decided to implement ESRGAN in PyTorch using the paper and a few other repositories. We tested the performance of the ESRGAN model on a set of test images that consists of images with different resolutions, orientations and scenes. We tried to keep this small testing set as diverse as possible in order to have a better understanding of the improvements that we make on our baseline. We obtained decent results from the baseline, but we feel that there was some scope for improvement as the images were very sharp and sometimes, choppy as well. Our earlier strategies included experimentation with hyper-parameters in order to get a better trained model and increased perceptual similarity to the ground truth high-definition images. We did not make a lot of progress with this but it gave us some ideas about what works and what does not. In the next phase of our changes, we replaced all our activation functions with Sigmoid Linear Units and placed the Batch normalization layers after the activation functions. We applied weight normalization to all the layers in the generator and spectral normalization to the layers in the discriminator in order to stabilize the training. We also added more RRDB blocks in the generator in order to improve the performance of our generator.

After making these changes to the structure of our models, we wanted to improve the performance by tweaking the losses and the training loop. We started experimenting with the losses as there are multiple losses that are being used in the ESRGAN implementation with different weights. One of the most important loss out of these is the Content Loss which is also referred to as Perceptual loss at different points in the paper. This loss is calculated by passing the real and the generated image through a pretrained VGG and the loss is calculated on the features generated after the 35th layer. In the paper, an L1 loss is used to compare these features but we found out that using an L2 loss is better for perceptual similarity. The Peak Signal to Noise ratio (PSNR) slightly goes down when the L2 loss is introduced but the image quality was definitely better and it looked more realistic and less sharp which is definitely a plus as ESRGAN authors pride themselves on the superior perceptual similarity that is obtained from their algorithm. Also, according to [19], Distortion measuring methods are always at odds with perceptual loss so this result was justified. We also tried a combination of L1 and L2 loss but we eventually replaced the Content Loss with MDF loss from [5] that follows a beautiful idea of using a generator to generate the perceptual loss and a series of discriminators that are used to penalize the predictions. This led to much better smoother super-resolution images and more realistic images.

One more major change that we made to the architecture for generator is the replacement of the bicubic interpolation function with PyTorch's pixel shuffle that is used to upsample the images. This also led to better performance perceptually and stabilized the training as well. In addition to that, we added a Gaussian Blur to the input images in order to make the generator generalize well to unseen images as well. We also added dropouts to our generator models in order to make it generalize better but that did not go well and the results were noisy and grainy. We planned to implement other changes as well but that was not possible as the process of training ESRGAN is computationally very expensive. Even on a p3 AWS instance, it took us an average of 30 hours to train every experiment that we conducted. We conducted 13 such experiments and we plan on conducting more experiments in the future to make the network even better.

5. EXPERIMENTAL RESULTS

Setup for the experiment: For our training dataset, we use DIV2K[20]. This has 800 high definition high resolution images collected from the Internet. The low-resolution images are generated using bicubic interpolation with downsampling factor of 4 and Gaussian blur.

Final Architecture: The final architecture builds on the baseline architecture shown in Figure 2.

- **Generator:** The Generator has 20 RRDB blocks as its basic blocks instead of 15. We employ weight-norm instead of batch normalization because whitening the inputs which are our low-resolution images does not work. We also use Swish as our activation function within every Residual Block.
- **Discriminator:** The Discriminator for ESRGAN is changed architecturally by adding more linear layers, adding spectral normalization for increased stability.
- **Upsampling:** The upsampling method seen in Fig. 2 is PixelShuffle instead of the baseline's bicubic interpolation.
- **Loss:** Content Loss and Adversarial Loss for the discriminator, generator stay the same as before. Perceptual Loss becomes the MDF Loss instead of the L1 loss between generated image and the features of a pre-trained VGG model.

Metric: Peak Signal to Noise Ratio was calculated for the super resolved images to evaluate their quality apart from visual cues. Apart from this, we plotted Adversarial Loss (for Generator), Content Loss, D Loss (discriminator loss), pixel loss, G Loss (final generator loss). New features were sequentially introduced to the baseline model and the results were studied to improve the GAN. New features were either kept or discarded based on their effect on the perceptual quality and not the metrics mentioned above.

Here, we have attached a table that explains 4 of our 13 experiments with the relevant features of the model in every experiment. We have also attached the average value of the PSNR for the validation set for every model. Please refer to the Appendix to view some more results from our improved ESRGAN model.





Experiment Name	Features	Results	Observations	PSNR
Baseline	Perceptual loss using VGG, perceptual and L1		Patchy artifacts are present with unwanted sharpening	24.47
Exp006	Used Spectral and Batch Norm, More linear layers in discriminator (8192, 100 -> 8192, 2048, 512, 100)		Color tones are different and evident square artifacts can be observed	24.14
Exp009	Introduced MDF loss in Generator. PixelShuffle used for upsampling. Dropout introduced. SGD in discriminator.		Grainy noise added due to dropout. Square artifacts are absent.	25.31
Exp013	Removed dropout.		Smoother output as compared to the baseline. The unwanted sharpness decreased and the perceptual quality increased.	23.25



FIGURE 2. Losses for Exp013 (Final)

6. DISCUSSION

We will divide this section into what we tried but discarded and what we'd have liked to implement further.

- **Changes Discarded:**

- Before we implemented the MDF loss for the Perceptual Loss, we tried improving the perceptual quality using L2 loss between the VGG-16 features and the Generator's output as compared to L1 loss. We observed that L2 improved the perceptual quality slightly but we discarded it in favour of MDF.
- A typical GAN network takes in "noise" as its input sampled from some distribution such as Gaussian. One way to stabilize the GAN is applying dropout on some layers of the Generator which would then add the noise themselves[21]. An important point to note here is that dropout should run while testing as well. We tried with p of 0.5 and 0.2. This did not work for our models - we think this is because the input "noise" in our Generator is the LR images, the dropout might have interacted with the LR images to simulate a distorted version of what real-world low-resolution images look like.

- **Improvements to Explore:**

- The input to the Generator is currently a low-resolution image which is created by using bicubic interpolation and Gaussian blur on our dataset of high resolution images. We believe that simulating LR images which are closer to real-world low-resolution images using better, higher order degradation methods might generate better results when we validate on our real-world low-resolution images.
- Novel GAN architectures like ProGAN by NVIDIA[22] grow from smaller sized layers for Generator and Discriminator that look at features in the low-resolution space to progressively adding more layers which look at the finer details as training progresses. A GAN architecture for Super-Resolution is an exciting space to explore with the GAN looking at lower resolutions progressively recovering the higher resolution image.

REFERENCES

- [1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In ECCV, 2014.
- [2] 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 ECCVW, 2018
- [3] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. CoRR, abs/1802.05957, 2018. 1
- [4] Sitao Xiang, Hao Li : On the Effects of Batch and Weight Normalization in Generative Adversarial Networks. In: stat.ML (2017)
- [5] Aamir Mustafa, Aliaksei Mikhailiuk, Dan Andrei Iliescu, Varun Babbar, Rafał K. Mantiuk.: Training a Task-Specific Image Reconstruction Loss, University of Cambridge.
- [6] Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network
- [7] Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. TPAMI 38(2) (2016) 295–307
- [8] Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y.: Residual dense network for image super-resolution. In: CVPR. (2018)
- [9] Haris, M., Shakhnarovich, G., Ukita, N.: Deep backprojection networks for superresolution. In: CVPR. (2018)
- [10] Tai, Y., Yang, J., Liu, X., Xu, C.: Memnet: A persistent memory network for image restoration. In: ICCV. (2017)
- [11] Kim, J., Kwon Lee, J., Mu Lee, K.: Deeply-recursive convolutional network for image super-resolution. In: CVPR. (2016)
- [12] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image superresolution using a generative adversarial network. In: CVPR. (2017)
- [13] Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep laplacian pyramid networks for fast and accurate super-resolution. In: CVPR. (2017)
- [14] Kelvin C.K. Chan, Xintao Wang, Xiangyu Xu, Jinwei Gu, and Chen Change Loy. Glean: Generative latent bank for large-factor image super-resolution. In CVPR, 2021.
- [15] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz, Loss Functions for Image Restoration with Neural. InL CVPR. (2018) Networks
- [16] Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV. (2016)
- [17] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint arXiv:1609.04802, 2016.
- [18] Jolicœur-Martineau, A. The relativistic discriminator: a key element missing from standard GAN. In ICLR, 2019.
- [19] Blau, Y., Michaeli, T.: The perception-distortion tradeoff. In: CVPR. (2017)
- [20] NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study
- [21] Image-to-Image Translation with Conditional Adversarial Networks
- [22] Progressive growing gfg GANs for improved quality, stability, and variation.

APPENDIX

The following images show the comparison of the Low resolution images, images generated using our baseline model ESRGAN and our final model. The LR images are on the left, ESRGAN outputs are in the middle and our model outputs are on the right. The major difference between the performance of our model and the baseline is that it generates excessively sharp images which is undesirable as it decreases the perceptual quality of the image. Instead, our model produces smoother images which are more appealing to the human eye and do not look very cartoon-like. The images generated from our model resemble the real life images more closely than the baseline.

Low Res image vs ESRGAN vs. Our model



