Superresolution of Images using SRGAN

Shreyas M.S. [S20190010161], Raghav Garg [S20190010148], Bhanu Teja [S20190010200]

***Abstract*—It has been almost two centuries since the first image was captured. We have still not perfected imaging technology to get a resolution that is higher than our eyes [1] and fast. To obtain higher resolutions of images that have already been taken, we turn to deep-learning techniques, specifically Super Resolution Generative Adversarial Networks (SRGAN for short). SRGANS allow us to train models that know how to "guess" more information about the image. The models are optimized on what is defined as content loss [2] and adversarial loss [2].**

I. Introduction

Super Resolution is the task of generating a higher resolution image from a given image. In reality it isn't possible to check whether the image generated is accurate or not, since we can't predict what the image would've looked like, we take an alternate approach. We take an image, down-sample to $highres/scale\_factor$ and then let the model upscale the image back to the original resolution.

The scale factor mentioned above is the number of times we want to increase the size of the image by. The commonly used cost metric for evaluating and optimizing the SR models is MSE (Mean Squared Error) and PSNR (Peak Signal to Noise Ratio)[2]. These metrics aren't a good measure however in certain cases. For example, if the generated image happens to be one shade darker or lighter from the expected image we would see a high MSE. To solve this, we use a modified version of content and adversarial loss.

II. Related Work

- Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial [2].
  This is the main paper that we looked at. Our implementation is a modified version of this same paper.
- Spectral Bandwidth Recovery of Optical Coherence Tomography Images using Deep Learning [3].
  This paper uses a mixed attention GAN to generate images.
- ADE20k Outdoors Dataset from the Kaggle Dataset[4] [5]
- The unreasonable effectiveness of deep features as a perceptual metric [6]

III. Problem Statement and Contribution

Given an image and a desired resolution, we are required to scale the image up to the desired resolution. Most commonly the ratio of the desired resolution to the original resolution is 2 or 4. We need to upscale in such a way that the image doesn't look blurry and does not have unwanted artifacts.

Contribution: Generative Adversarial Networks are great for generating data that is realistic. In this paper we have implemented a super resolution generative adversarial network with a few modifications to the architecture and loss function. Our main contributions are:

- We have improved the SSIM score of the generated models with the original images by using a modified content loss and adversarial loss that incorporates Structural similarity index measure.

- As a consequence of the above point, we have also seen an improvement in the Learned Perceptual Image Patch Similarity measure as compared to the paper's hyperparameters and architecture.

IV. Data

We used the ADE20K dataset [4][5] which consists of ten thousand images of varying sizes.

Half of the images (5000) are of type JPG and the other half of the images are of type PNG. The dataset consists mostly outdoor scenes with ample lighting and clearly defined textures and borders. While the dataset on its own isn't very big, when we load it into pytorch as tensors, the size increases significantly due to the fact that each pixel now takes up 8 bits * 3 channels and no compression is used. This for an average of 500px by 500px image still only adds up to around $10000 * 500 * 500 * 8 * 3 = 60000000000bits = 55.9Gb = 7GB$. The data size issues only show up when we use torch's auto-grad [7] during the training period.

V. Proposed Methodology

At first we have images of size H * W * 3 channels (RGB). We crop this into an image of size 256 * 2566 * 3 channels. In cases where one of the image dimensions is lower than the required 256, we simply pad the columns or rows. Once this is done, we have a standardized set of images, each in a HR*HR*3 size where HR=256. This is what we call the HR image, this image is then down-sampled to either HR/2 * HR/2 * 3 or HR/4 * HR/4 * 3 based on the required scale. This new downsampled image is called the 'Low Resolution' image.

The Low resolution image is now passed into the Super Resolution Generative Adversarial Network's Generator to obtain a HR * HR * 3 image. This 'fake' image that is created by the generator is what we want to perfect. The closer this image is to the original image, the lower the loss. Once we have this image, we compare it with the original image using SSIM content loss or MSE content loss.

Now that we have our 'fake' image, we pass it into our discriminator. The discriminator part of the network is simply a VGG19 model. The discriminator's job is to output a zero if the image is real and 1 if the image is fake. The farther away it is from the truth, the higher it's penalty. The discriminator can simply use a binary cross entropy loss since it only outputs one real number. The losses are then back-propagated from the discriminators last layer up to the generator's first layer. This means that the generator tries its best to fool the discriminator and the discriminator tries its best to flag the image if the image was generated by the generator.

The model architecture 1 we used is the same as what is used in the SRGAN [2] paper with a few modifications to the hyperparameters. The base paper used a learning rate of 10^{-4}, 10^6 update iterations, a scaling factor of 4, Leaky ReLU with $\alpha = 8$ and 16 Residual blocks.

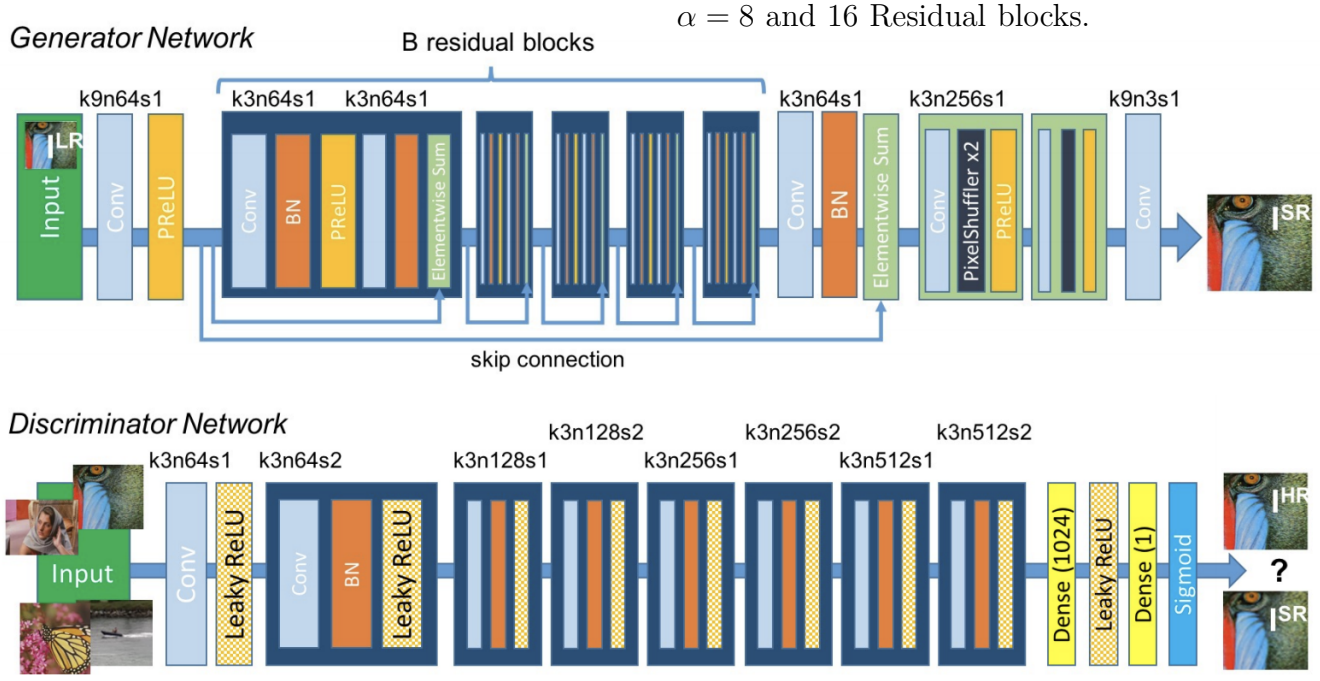

Fig. 1. SRGAN Architecture

For content loss, we use

$$(\text{loss}^{SR}_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rW}(I^{HR}_{x,y} - G_{\theta G}(I^{LR})_{x,y})^2$$

The adversarial loss is simply min-max loss which is defined as

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

Instead of using MSE (Mean Squared Error) in the content-loss, in our model, we have used SSIM (Structural Similarity Index Measure)[8] . This is because Structural Similarity Index Measure is often better at showing how different two images are visually (or more accurately, structurally). We also changed the number of Resnet blocks from 16 used in the original paper to 20 to improve the learning ability of the model. We found that these changes significantly improved our Structural Similarity Index Measure score as compared to the base paper.

VI. Performance Analysis

TABLE I
Model lpips and ssim comparision; OP=Original Paper, M=Modified

	OP (4x)	M (4x)	OP (2x)	M (2x)
SSIM	0.864	0.872	0.871	0.942
LPIPS	0.328	0.32	0.302	0.142

From Table I we can see that our 4x upscaling model performs slightly better in both Structural Similarity Index Measure and also Learned Perceptual Image Patch Similarity scores. As Expected our implementation of the 2x SRGAN beats the 4x paper implementation by a big margin. We can also see that the modified implementation of the 2x upscaler is significantly better than the 2x upscaler as defined by the paper's hyperparameters.

We attribute these improvements to the changes in the loss function and the increased number of ResNet blocks.

Let us look at a few examples of how our model is performing for some samples. Figure 2 shows a building that is scaled up by 4x, visually it is much clearer than the low resolution image.

Figure 3 shows us another example where we try to upscale the image of a forest, our model has been trained on images where there is usually a lot of light visible behind leaves. This bias shows up in this example, where the model is generating data that shouldn't exist.

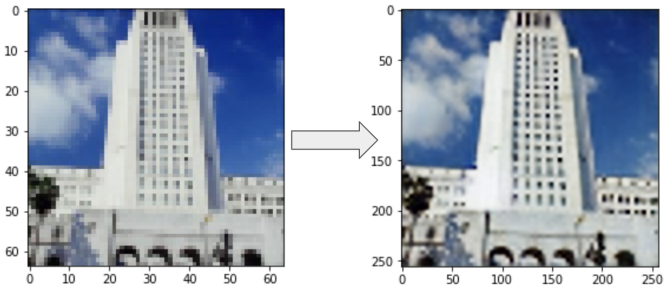

Fig. 2. Example 1

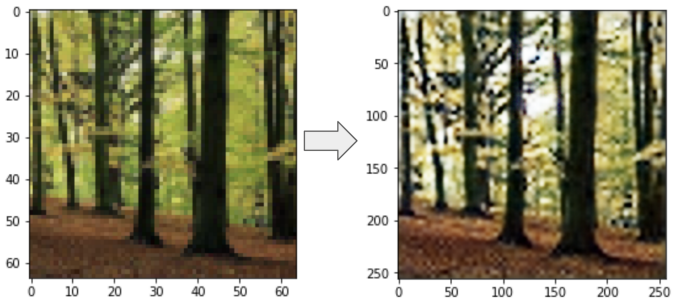

Fig. 3. Example 2

VII. Conclusion Future Work

We were able to improve on the original paper's implementation of the SRGAN. However the magnitude of improvement decreases as we increase the scaling factor. This is clear when we see that the 2x scaling factor results are significantly better than the paper's implementation; however the improvements in results are much lower for 4x scaling factor.

While working on this project we found out a few drawbacks:

- Generators have a lot of bias depending on the data they are trained on.
- SRGANS sometimes produce "soft" or "smooth" looking images.
- SRGANS can sometimes introduce a colour shift or even non-existent details in the images.
- Generators are notorious for being slow and hard to train. This is especially true when

using computationally heavy calculations like LPIPS.

In the future we plan to do the following:

- Try this model in a domain such as vehicle dashcam video upscaling.
- Test out other loss metrics that might get us better results
- Try to improve the efficiency of GANs

References

[1] S. Kreis, "Megapixels and human recognition of resolution," *The Physics Teacher*, vol. 46, no. 5, pp. 304–305, 2008.

[2] C. Ledig, L. Theis, F. Huszár, 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," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[3] C. Tian, J. Yang, P. Li, S. Zhang, and S. Mi, "Retinal fundus image superresolution generated by optical coherence tomography based on a realistic mixed attention gan," *Medical Physics*, vol. 49, no. 5, pp. 3185–3198, 2022.

[4] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[5] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, no. 3, pp. 302–321, 2019.

[6] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[8] J. Nilsson and T. Akenine-Möller, "Understanding ssim," *arXiv preprint arXiv:2006.13846*, 2020.