# CIS 680 Project

## Generating Super-Resolution Images using GANs

Nanthini Balasubramanian, Apurba Sengupta
University of Pennsylvania
{nanthini,aseng}@seas.upenn.edu

## 1 Introduction

Realistic super-resolution of images is an area that receives a lot of attention. We want to get the finer details of a low resolution image by upscaling it. The aim of the project is to be able to produce high resolution images of corresponding low-resolution photos. We try to achieve this by using Generative Adversarial Networks (GANs). The datasets we have used for this project are CIFAR10, CIFAR100, BSD200, General100, Urban100, T91, Manga109, Set5 and Set14, all of which are benchmark images used in Single-Image Super Resolution (SISR).

## 2 Related work

SISR has been attempted using CNNs, ResNets[8], etc. Most recently, SRGAN[1] has been successful in achieving the same. The authors made use of a perceptual loss for the generator, which is a combination of content loss and adversarial loss, for training the network. The content loss consists of two components – a mean-squared error between the original high-resolution image and the generated fake super-resolved image, and a mean-squared error between the two in the VGG19 feature space. The adversarial loss is as in DCGAN[3] implementation. The authors used a Mean-Opinion-Score (MOS) metric to evaluate the quality of generated images.

An extension of this was seen in ESRGAN[2], which makes use of a Residual-in-Residual Dense Block as the building block for the network. Further, the authors made use of a Relativistic GAN (RaGAN)[5], which helped in predicting the relative realness of the generated and the ground-truth images, instead of fake and real images. The content loss component of the generator uses pre-activation values in the VGG19[4] feature space. The adversarial loss component of the generator is as per the RaGAN objective. Perceptual Index (PI) was used for generated image evaluation.

## 3 Model details

We try to combine model details from SRGAN and ESRGAN in building our model. We feed in images downscaled by a factor of 2x. We represent these low resolution images by $I_{LR}$. The corresponding true high-resolution images of these low-resolution are represented by $I_{HR}$. We feed these low-resolution images into the generator network $G$ of the Generative Adversarial Network. The $G$ then gives a fake super-resolved image for each $I_{LR}$, which we represent by $I_{SR}$. This super-resolved image and the original high-resolution image are then fed into the discriminator network $D$ which then tries to evaluate the realness of the $I_{SR}$ with respect to $I_{HR}$. Training this GAN-based model with images taken from different benchmark datasets for SISR (like BSD100, Urban100, Set5, Set14, etc.) enables it to generate good super-resolved outputs for given low-resolution images. We use the Structural Similarity Metric (SSIM)[6] and Visual Information Fidelity Measure (VFIP)[7] metrics to measure the quality of the generated super-resolved images as compared to the original high-resolution images.

### 3.1 Model architecture

We tried to combine architectural details from SRGAN and ESRGAN into the architecture of our model. The specifics are as follows:

**Generator** $G$

The low-resolution image $I_{LR}$ of size 50x50x3 are passed through a convolutional layer, then through 16 residual blocks (ResNet)[9], another convolution and then through an upsample block which upsamples the low-resolution image by a factor of 2x to 100x100x3, which is the super-resolved image $I_{SR}$. This is the generated "super-resolved" image. We use Parametric ReLU activation in the $G$ network. We test with Batch Normalization (BN)[10] layers before each activation, with partially removing few BN layers and removing all BN layers. The architecture is shown below in Figure 1.

**Discriminator** $D$

The super-resolved image $I_{SR}$ generated by $G$ and the corresponding original high-resolution image $I_{HR}$ are passed through a series of convolutions to finally generate a 512-dimensional vector representation for each. This feature space representation is then used to predict the relative realness between the original and the generated images. We use Leaky ReLU activation in the $D$ network with negative slope coefficient $\alpha = 0.2$. We test with both BN and Spectral Normalization (SN)[11] layers before each activation. The architecture is shown below in Figure 2.

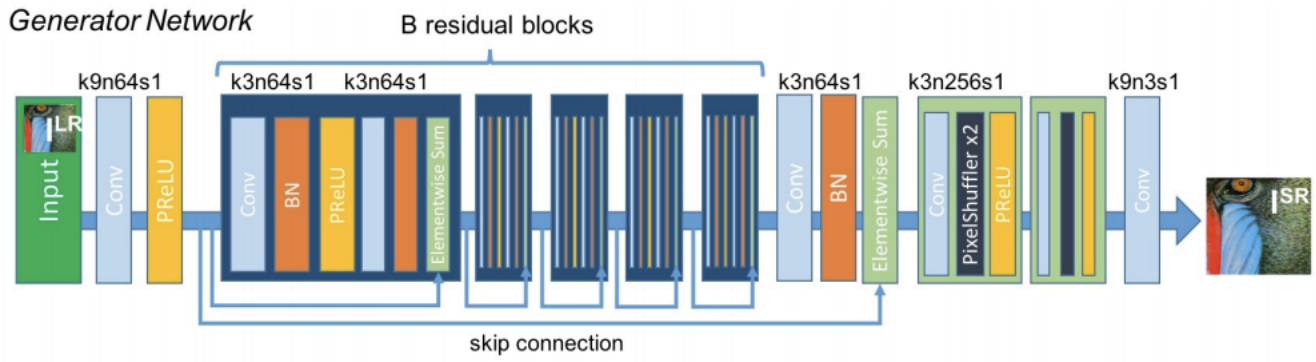



Figure 1: Generator $G$ with $B = 16$ residual blocks & BN layers

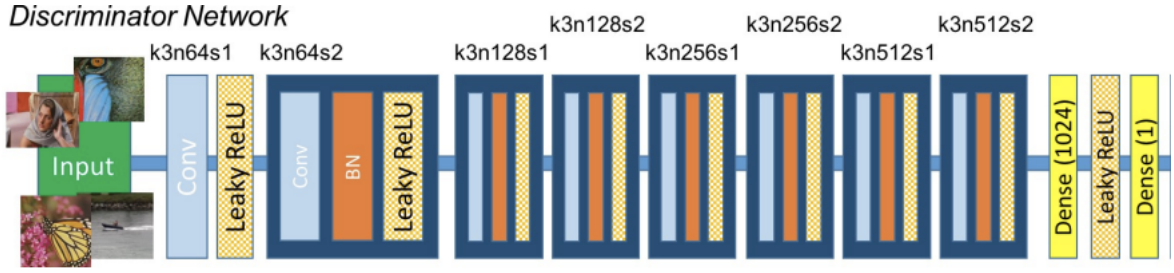



Figure 2: Discriminator $D$ with SN blocks and 512-dimensional output feature space

## 3.2 Loss functions

We tried to combine the loss functions from both SRGAN and ESRGAN into the loss function of our model. The specifics are as follows:

**Generator loss**

The $G$ loss function $L_G$ consists of two main parts: (1) content loss $L_G^c$, and (2) adversarial loss $L_G^a$. Further, the content loss is a weighted combination of two losses – the pixel-wise Mean Squared Error $l_{MSE}$ between the generated super-resolution image and the corresponding high-resolution image, and the same loss between the two, but represented in the VGG19 feature space, written as $l_{VGG}$. The adversarial loss tries to minimize the DCGAN generator loss objective, i.e., tries to fool $D$ into believing that the

generated image is real. In practice, it is just the Binary Cross Entropy Loss between the 512-dimensional Sigmoid activation of the generated image after it is passed through $D$, and a probability of all ones. Hence,

$$L_G^c = l_{MSE} + (0.006)l_{VGG} = \texttt{MSE}(G(I_{LR}), I_{HR}) + (0.006)\texttt{MSE}(G(I_{LR})^{VGG}, I_{HR}^{VGG})$$

$$L_G^a = \texttt{BCE}(\sigma(D(G(I_{LR}))), 1)$$

$$L_G = L_G^c + 10^{-3}L_G^a$$

**Discriminator loss**

The $D$ loss function is taken from the RaGAN discriminator loss objective and is represented by $L_D$. It tries to measure the relative realness of the generated super-resolution and the actual high-resolution images (see Figure 3). In particular, it consists of two parts. The first part compares the discriminator $D$ 512-dimensional representation of the original high-resolution image to that of the batch average of the generated super-resolution images to see if the high-resolution image is more realistic than the fake generated image. The second part compares the discriminator $D$ 512-dimensional representation of the generated super-resolution image to that of the batch average of the original high-resolution images to see if the super-resolution image is less realistic than the true original image. In practice, these losses are just the Binary Cross Entropy Losses between the difference of the above representations (high-resolution & batch average of super-resolution, and super-resolution & batch average of high-resolution) and corresponding high and low probabilities respectively. Hence,

$$L_D = \texttt{BCE}(\sigma(D(I_{HR}) - E[D(G(I_{LR}))]), \sim 1) + \texttt{BCE}(\sigma(D(G(I_{LR}))) - E[D(I_{HR}]), \sim 0)$$

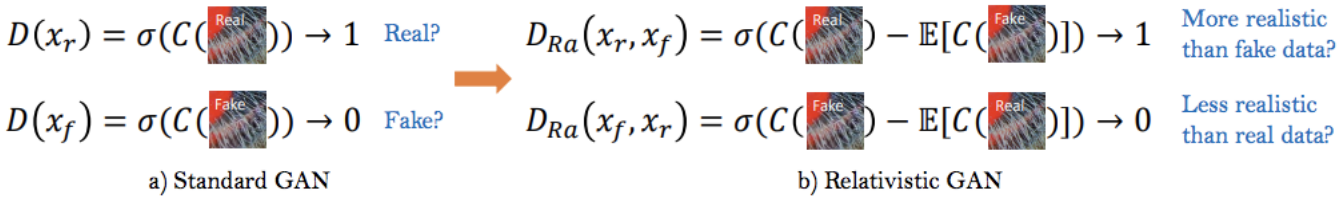



Figure 3: Comparison between discriminator outputs of vanilla GAN and RaGAN.

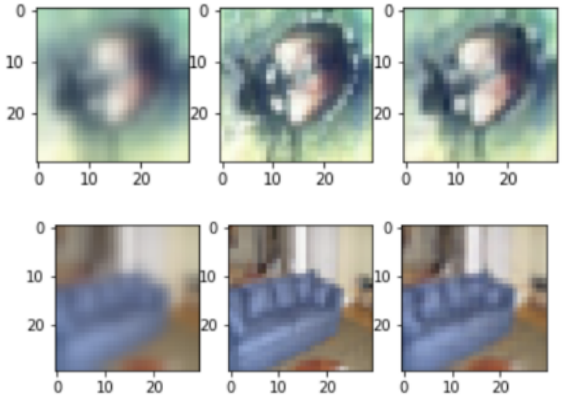
We used an Adam optimizer with a learning rate of 0.0001 to minimize the both generator and discriminator losses.

## 4 Experiments & Results

We used both Google Colaboratory and a NVIDIA GeForce GTX 1070 GPU machine for training our model(s). However, most of the development was carried out on the GPU.

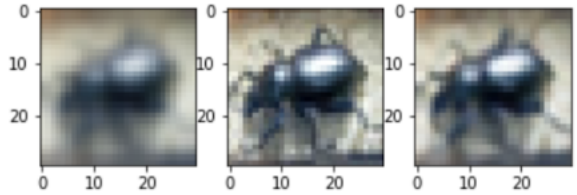
In an initial effort to try out an out-of-the-box model, we utilized an implementation of SRGAN on CIFAR10 & CIFAR100 as a starting point[13]. The activation function used throughout in this model was the Swish[14] activation. The sizing factor was 2x. In addition to using the VGG19 features of the generated and original images in the generator loss function, we also tried using AlexNet[12] features for the same. Training the network for 100 epochs with AlexNet features gave lower losses for both the discriminator and the generator as compared to training the network with VGG19 features for the same number of epochs. Figures 4(a) and 4(b) below shows the low-resolution, original high-resolution and generated super-resolution images for VGG19 and AlexNet features over the 100 epochs respectively. The loss values at the end of training the model on VGG19 and AlexNet features are listed in Table 1.

[59/100][200/500] Discriminator_Loss: 0.5771 Generator_Loss (Content/Advers/Total): 0.0536/1.8814/0.0555



[100/100][499/500] Discriminator_Loss: 0.6226 Generator_Loss (Content/Advers/Total): 0.0448/1.8730/0.0466

Figure 4(a): Training with VGG19 features – results at 59th and 100th epochs



[88/100][400/500] Discriminator_Loss: 0.6434 Generator_Loss (Content/Advers/Total): 0.0301/1.8770/0.0320

[97/100][400/500] Discriminator_Loss: 0.5882 Generator_Loss (Content/Advers/Total): 0.0306/1.9290/0.0325

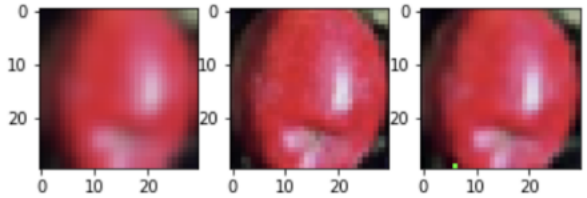
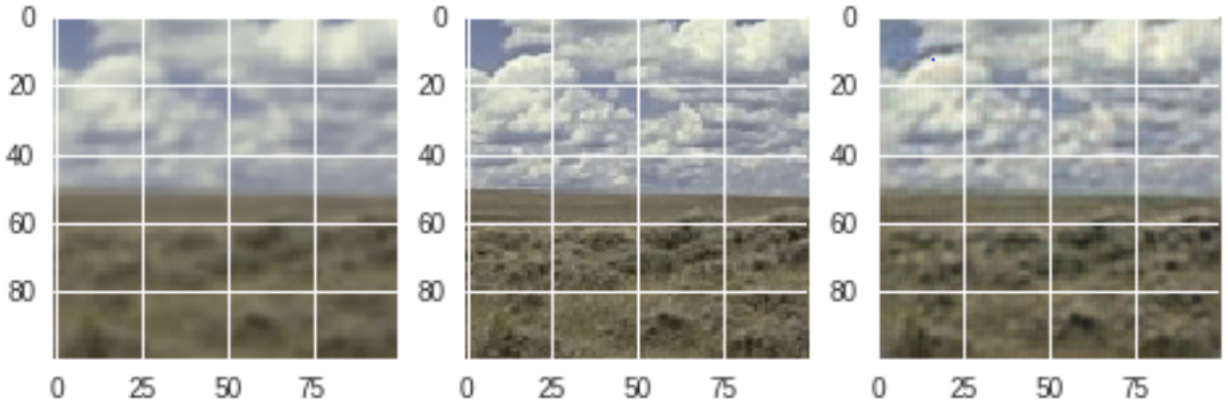


Figure 4(b): Training with AlexNet features – results at 88th and 97th epochs

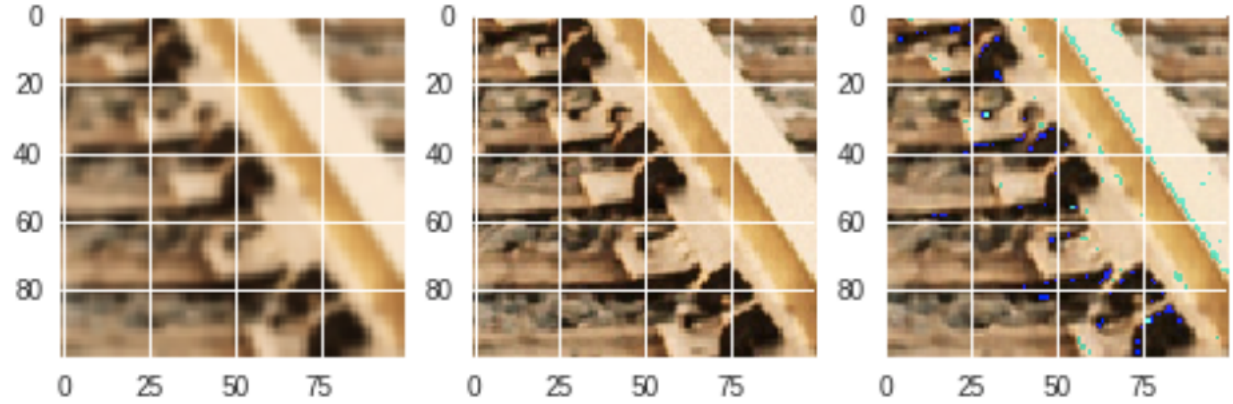
| Feature Extractor | Discriminator Loss | Generator Loss | | |
|---|---|---|---|---|
| | | Content | Adversarial | Total |
| VGG19 (100 Epochs) | 0.6226 | 0.0448 | 1.8730 | 0.0466 |
| AlexNet (100 Epochs) | 0.6034 | 0.0306 | 1.8413 | 0.0325 |

Table 1: Loss comparison of VGG19 v/s AlexNet feature extractors

We wanted to try out the task with larger images (100x100) and so we utilized around 600 images from BSD200[15], General100[16], Urban100[17], Manga109[18] and T91[19] to train the network we described in 3 (Model details) above. We tried a bunch of different architectural, loss function and optimizer modifications till we finalized the architecture in 3 as the superior performing one. Few of the intermediate models that we trained gave results as depicted below in Figure 5. The accompanying specification indicates what difference we had from the final model in 3. The order is again low-resolution, original high-resolution and generated super-resolution images.



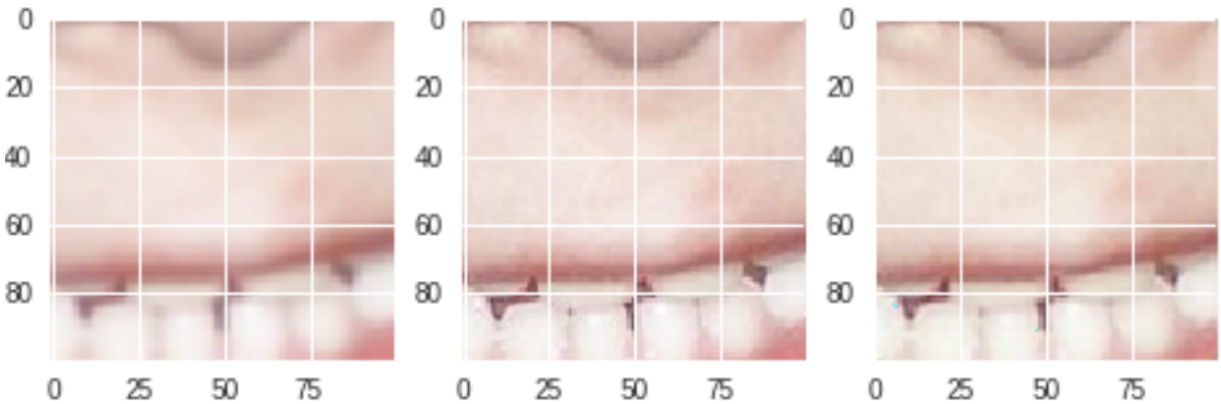
Generator optimizer - SGD with learning rate = 0.01 and momentum = 0.8, Discriminator optimizer - Adam with learning rate = 0.001



Generator and Discriminator both with RaGAN objective just as in the ESRGAN paper

Figure 5: Experiments with intermediate models

The model we described in 3 gave the following results during training on the 600 images we described above (see Figure 6 below). We tested the generator model on Set5 and Set14[17] images to visualize the quality of generated super-resolved images (see Figure 7 below).



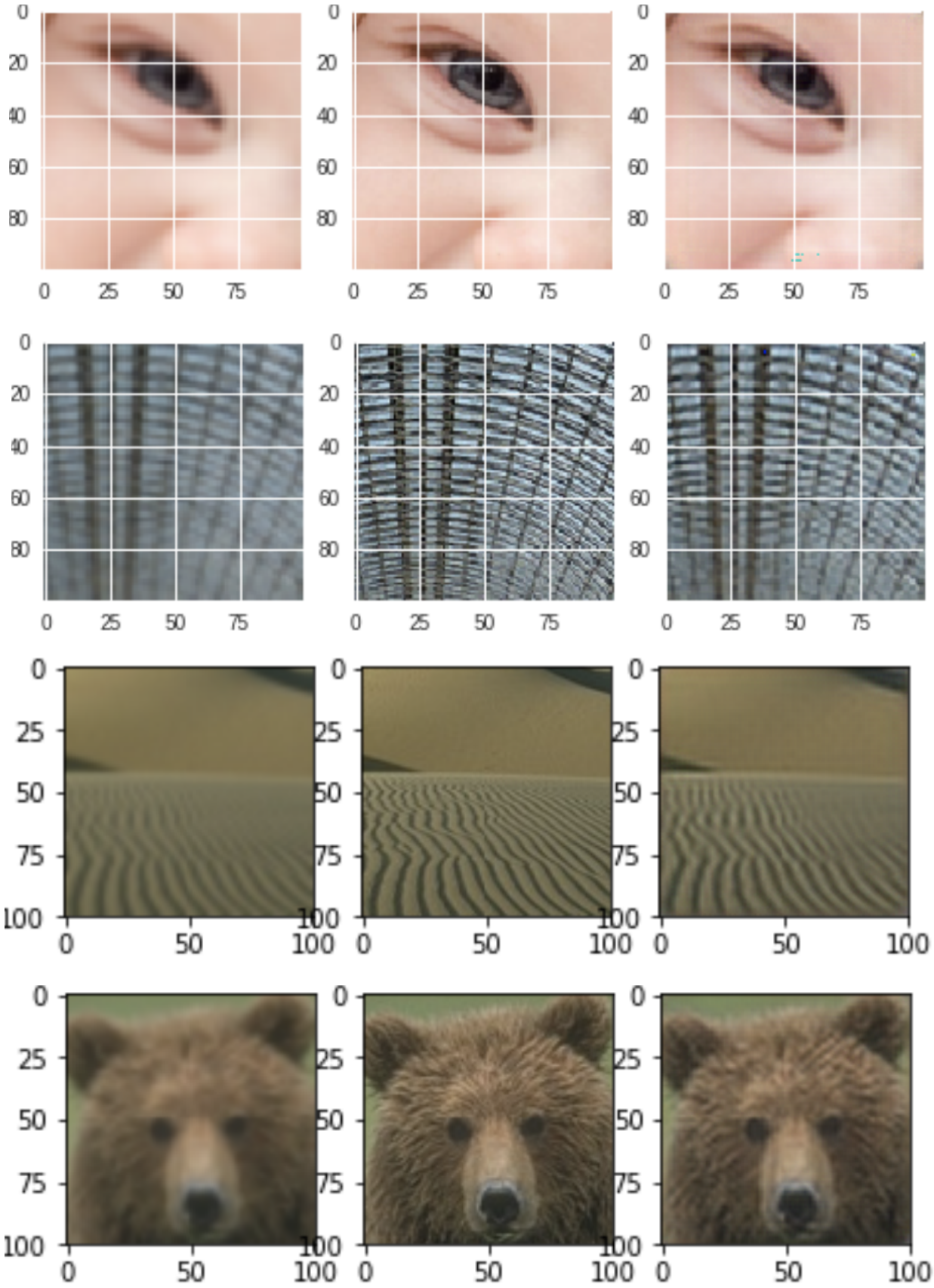
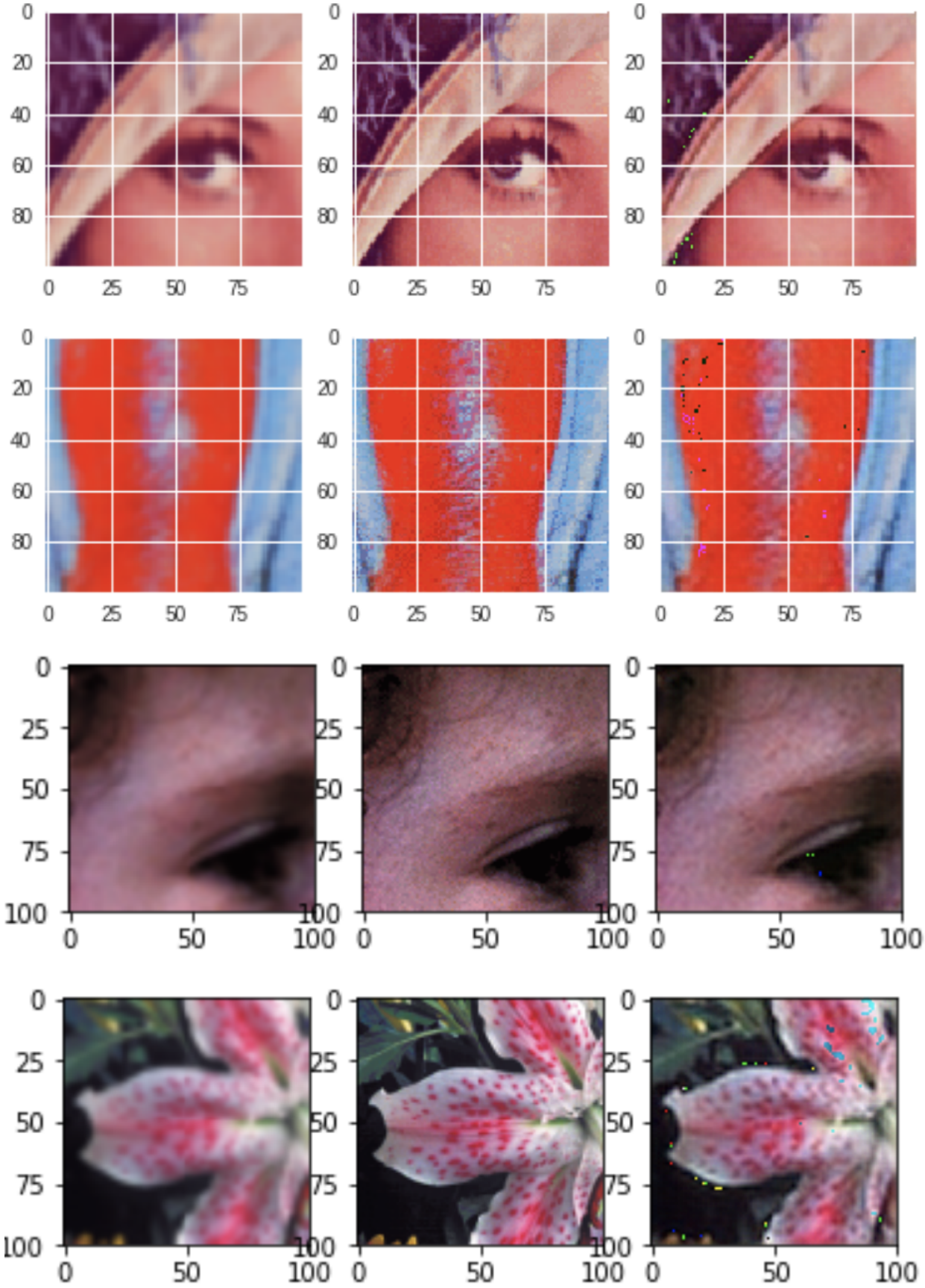


Figure 6: Training results on BSD200, General100, Urban100, Manga109 and T91


0
20
40
60
80
0 25 50 75
0
25
50
75
100
0 50 100

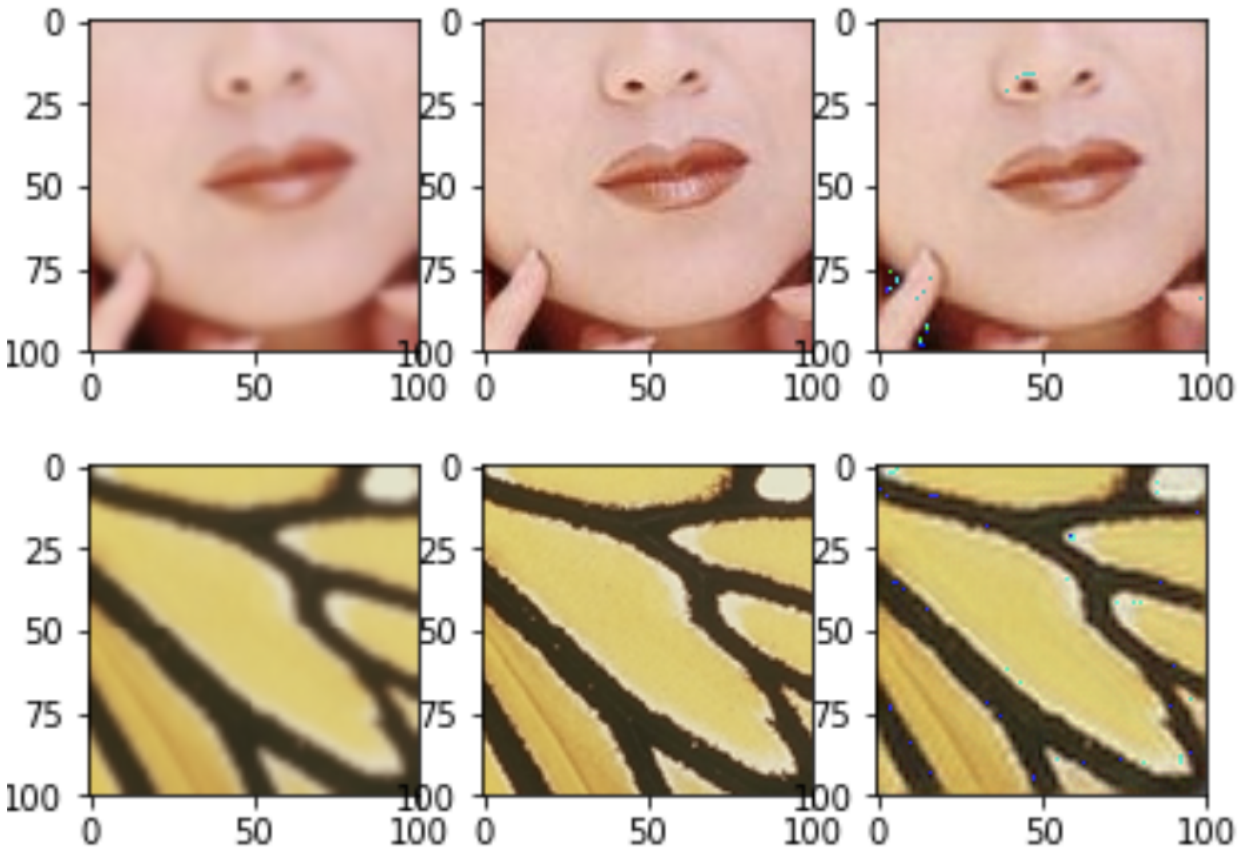


Figure 7: Test results on Set5 and Set14

While we see that some of the generated super-resolved images are rich in contents and very similar to the original high-resolution images, we do notice some RGB artifacts that appear in the generated images in the form of colored spots or patches[2]. This behavior is partially due to the presence and the location of BN layers in the $G$ network. Removal and changing the location of these BN layers gives different results. One such result is shown below in Figure 8.

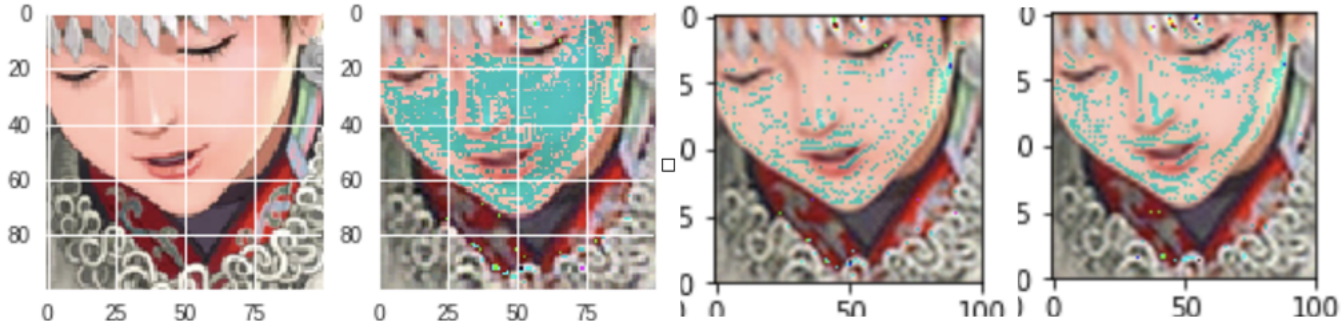


Figure 8: (From left to right) Original HR image; SR image from generator with BN layers; SR image from generator with BN layers removed from the residual blocks; SR image from generator with all BN layers removed

The mean SSIM score for the images generated during training increases from nearly zero to around 0.8, and the mean VIFP score for the same increases from nearly zero to around 0.94, both indicative that the generated image quality improves over training iterations and is gradually perceived to be a true super-resolved image. The mean SSIM and VIFP scores for the images generated for the test datasets are found to be around 0.8 and 0.88 respectively, which are comparable to the ones obtained during training. Thus, the model generalizes for unseen data as well.

We tried other changes to the model too, like incorporating Residual-in-Residual Dense Block (RRDB) architecture as in the ESRGAN paper[2], using features extracted from deeper architectures like DenseNet[20] instead of VGG19, and increasing the input image size & the upscaling factor to more than 2. But these modifications were too computation heavy for a single GPU machine and hence, we could not incorporate them in the final model.

## 5 Conclusion

The GAN-based model we developed for super-resolution of given low-resolution images achieves good visual quality scores as measured by SSIM and VIFP metrics. The key architectural components of our model are – 16 residual blocks, few or no BN layers, Parametric ReLU activations and DCGAN objective for the generator network $G$, and SN layers, Leaky RelU ($\alpha = 0.2$) activations, 512-dimensional output layer and RaGAN objective for the discriminator network $D$. We also tried to see the affect of removal of BN layers in $G$ and noticed the change in RGB artifacts that are generated in some of the super-resolved images. We can definitely try enlarging the architecture to incorporate RRDB or taking features from deeper CNN architectures or increasing the upsampling factor to more than 2. But we would need greater computational prowess to train such a large model.

## Acknowledgements

We would like to thank Prof. Jianbo Shi for covering an excellent and highly resourceful class on Deep Learning, and for always being available to address our queries. We would also like to thank the Teaching Assistants – Oleh Rybkin, Nikolaos Kolotouros and Bowen Ke – for their support and commitment towards addressing our queries, pointing to useful resources and making this class a highly enjoyable experience. Additionally, we would like to thank Mr. Yash Vardhan (M.S.E. in CIS '19) for lending us his personal GPU machine for computational purposes of this project. Last but certainly not the least, we would like to thank members of the GRASP Laboratory for providing us an environment for intellectual and cultural exchange.

We would once again like to thank the aforementioned people. This project could not have been completed without their support.

## References


[1] Ledig, C. et al., 2017, *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*, https://arxiv.org/pdf/1609.04802.pdf

[2] Wang, X. et al., 2018, *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks*, https://arxiv.org/pdf/1809.00219.pdf

[3] Radford, A. et al., 2016, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, https://arxiv.org/pdf/1511.06434

[4] Simonyan, K. et al., 2014, *Very Deep Convolutional Networks for Large-scale Image Recognition*, https://arxiv.org/pdf/1409.1556.pdf

[5] Jolicoeur-Martineau, A., 2018, *The relativistic discriminator: a key element missing from standard GAN*, https://arxiv.org/abs/1807.00734

[6] Wang, Z. et al., 2004, *Image Quality Assessment: From Error Visibility to Structural Similarity*, http://www.cns.nyu.edu/pub/lcv/wang03-preprint.pdf

[7] Sheikh, H. et al., 2006, *Image Information and Visual Quality*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.5187rep=rep1type=pdf

[8] Kim, J. et al., 2016, *Deeply-Recursive Convolutional Network for Image Super-Resolution*, https://arxiv.org/abs/1511.04491

[9] He, K. et al., 2015, *Deep Residual Learning for Image Recognition*, https://arxiv.org/pdf/1512.03385.pdf

[10] Ioffe, S. et al., 2015, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, https://arxiv.org/abs/1502.03167

[11] Miyato, T. et al., 2018, *Spectral Normalization for Generative Adversarial Networks*, https://arxiv.org/abs/1802.05957

[12] Krizhevsky, A. et al., 2012, *ImageNet Classification with Deep Convolutional Neural Networks*, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[13] Ruano, A., *PyTorch-SRGAN*, https://github.com/aitorzip/PyTorch-SRGAN

[14] Ramachandran, P. et al., 2017, *Searching for Activation Functions*, https://arxiv.org/pdf/1710.05941.pdf

[15] Arbelaez, P. et al., *The Berkeley Segmentation Dataset and Benchmark*, https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[16] Dong, C. et al, 2016, *Accelerating the Super-Resolution Convolutional Neural Network*, http://personal.fie.cuhk.edu.hk/ ccloy/files/eccv$_2$016$_a$*ccelerating.pd*

[17] Huang, J. et al., 2015, *Single Image Super-Resolution from Transformed Self-Exemplars*, https://github.com/jbhuang0604/SelfExSR

[18] Akamatsu, K. et al., *Manga109*, http://www.manga109.org/en/

[19] Lai, W. et al., 2016, *Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks*, http://vllab.ucmerced.edu/wlai24/LapSRN/

[20] Huang, G. et al., 2017, *Densely Connected Convolutional Networks*, https://arxiv.org/pdf/1608.06993.pdf