CS 545: Machine Learning **Fall Term 2018**

Project Topic: Super Resolution Generative Adversarial Network

Due: December 7, 2018, 5:00PM PT

Student Names: Archit Tatwawadi, Mohak Patel

1 Introduction

The challenging task of estimating a high-resolution(HR) image for a low resolution(LR) image counterpart is called as the super-resolution(SR). The existing methods for a SR image generation do the job of super resolving properly but fail to retain the finer details in the image. In this project, we implemented the Super Resolution Generative Adversarial Network as explained in the original paper "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network"[1]. In this project, we trained the SRGAN on some randomly fetched images from Image-net data set. There were total Four different models implemented in this project using Keras library. The implemented models are: Generator Network, Discriminator Network, combined model of Generator and Discriminator,pretrained VGG network. The SRGAN was trained and tested for different learning rates: 10^{-4} and 10^{-5} using Adam optimizer. The VGG network was not trained and it was used to calculate the content loss of the generated image. One more loss was calculated called as Adversarial loss.The adversarial loss basically is the measure of how realistic the generated image is. This encourages our network to favor solutions that reside on the manifold of natural images, by trying to fool the discriminator network. In this project we also experimented with $2\times$ and $4\times$ magnification and exploited how the magnification level affects the generator output.

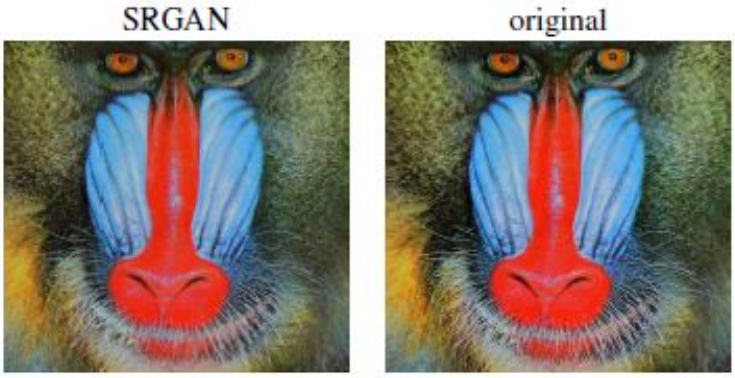


Figure 1: Example of output from the original paper.

2 Method

This project aims to estimate a high resolution, super resolved image(I^{SR}) from a single low resolution(I^{LR}) image. The low resolution image is generated from its high resolution(I^{HR}) counter part. For training purpose, the I^{LR} is generated by down sampling the original I^{HR} image by a factor r such that the final I^{LR} image has dimensions $rW \times rH \times C$ (where, W: Width, H: Height, C: Number of channels in the image).

2.1 Adversarial network architecture

As discussed in the original "Generative adversarial nets" paper[2], the authors implemented a generative network and a discriminator network. The generative network is implemented by using a feed-forward CNN network which trains a generative function(G_{θ_G}). The author also defined a discriminator network with function D_{θ_D} which is trained along with the generator network by solving the adversarial min-max problem:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR}}[log D_{\theta_D}(I^{HR})] + \mathbb{E}_{I^{LR}}[log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR})))] \tag{1}$$

The architecture of generator and discriminator network is as shown in Fig2.

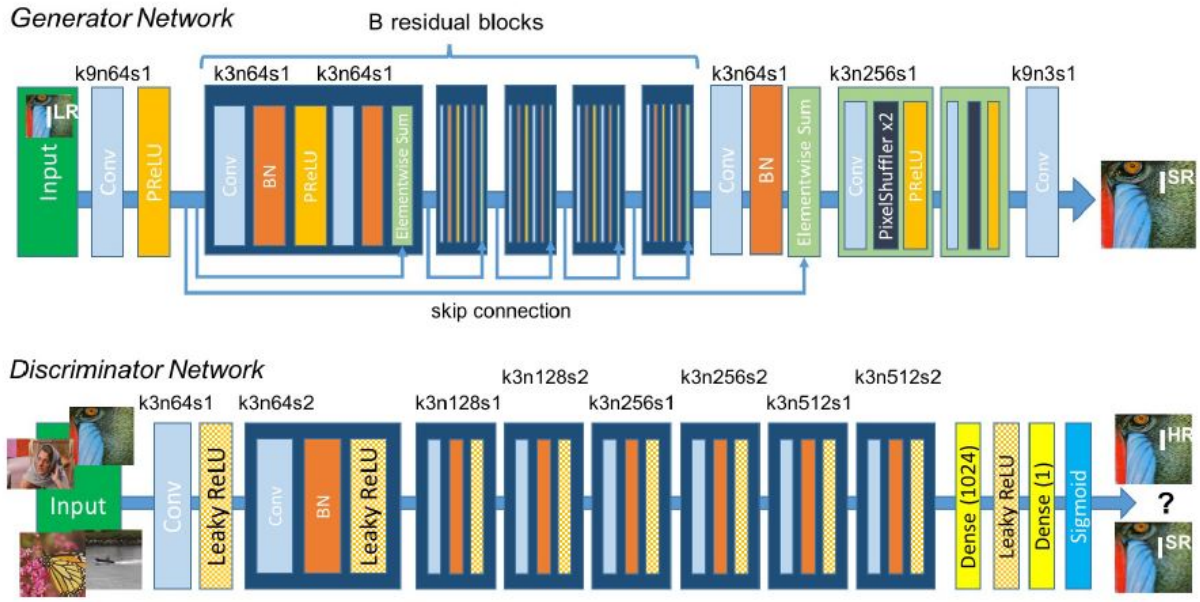


Figure 2: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer from the original paper.

The main block of the generator network is the residual block consisting of five layers:Convolutional layer,batch normalization,PReLU, convolutional layer, batch normalization, element-wise sum. The kernel size and number of strides in the convolutional layer is shown in the architecture Fig 2. The number of residual blocks(B) to use to get better results is an hyperparameter. After the residual blocks, there is a pixel shuffler(sub-pixel convolutional block). It upscales the generated output to increase the resolution of the convolved output.

The discriminator blocks are implemented exactly as shown in Fig 2. The main block of discriminator network has three layers: Convolutional layer, batch normalization, Leaky ReLU. The Leaky ReLU layer has activation $\alpha = 0.2$ and it is made sure that max-pooling layer is not used throughout the network. The discriminator is used to solve the maximization problem as described in Equation (1). There are two fully connected layers at the end along with sigmoid activation to get a probability for sample classification.The two losses, content loss and the adversarial loss is calculated while training of the SRGAN. The adversarial loss is extracted from the pretrained VGG network. While training, either of Generator or discriminator networks are back propagated and the values are updated accordingly.

2.2 Experiments

In this project, we did some experiments regarding the upscaling factor and the learning rates of the networks. To implement this project, we used Google Colab GPU runtime. We tried to use Google TPU, but context switching between generator and discriminator models caused a lot of delay rendering the improvement in training time useless. The Image-net data set URL's were downloaded from the official site and divided into 10 different training and testing data sets. We implemented a parallel threading script to download the images in bulk and handled all the exceptions. The script also cleaned the downloaded data by removing the unavailable images. We trained SRGAN for 2000 images and for 90000 epochs for both 10^{-4} and 10^{-5} learning rates. The scaling factor for training was fixed at $4\times$. For implementing the SRGAN, we defined a class which can be initialized with different configurations like scaling factor, learning rate of each network, load or save weights of all the networks, etc. As is explained by the authors in [1], we have chosen 16 residual blocks for the generator network as it gives sharper results. The Adam optimizers for all the networks were initialized with $\beta_1 = 0.9$. As the project was implemented using Keras library, there were some other changes that were made while implementing. The PReLU was instantiated with zeros and the axes of the data were shared as the layer previous to the PReLU layer is a convolutional layer. The pixel shuffler was replaced by a UpSampling2D layer. It serves the same purpose of upscaling the image to increase the resolution but since it is not the same function, it may affect the result.

3 Results

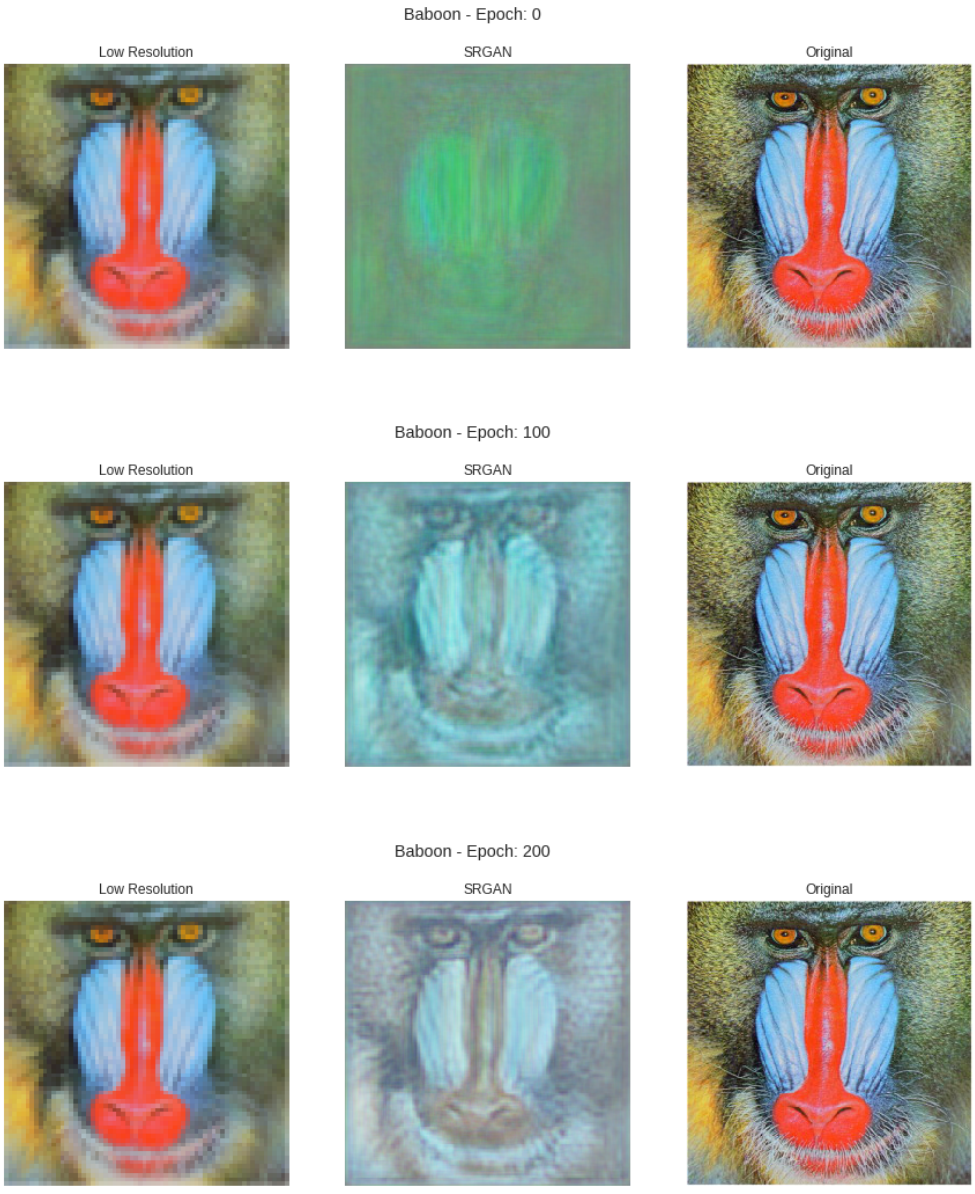


Figure 3: Test image results during training(Set1)

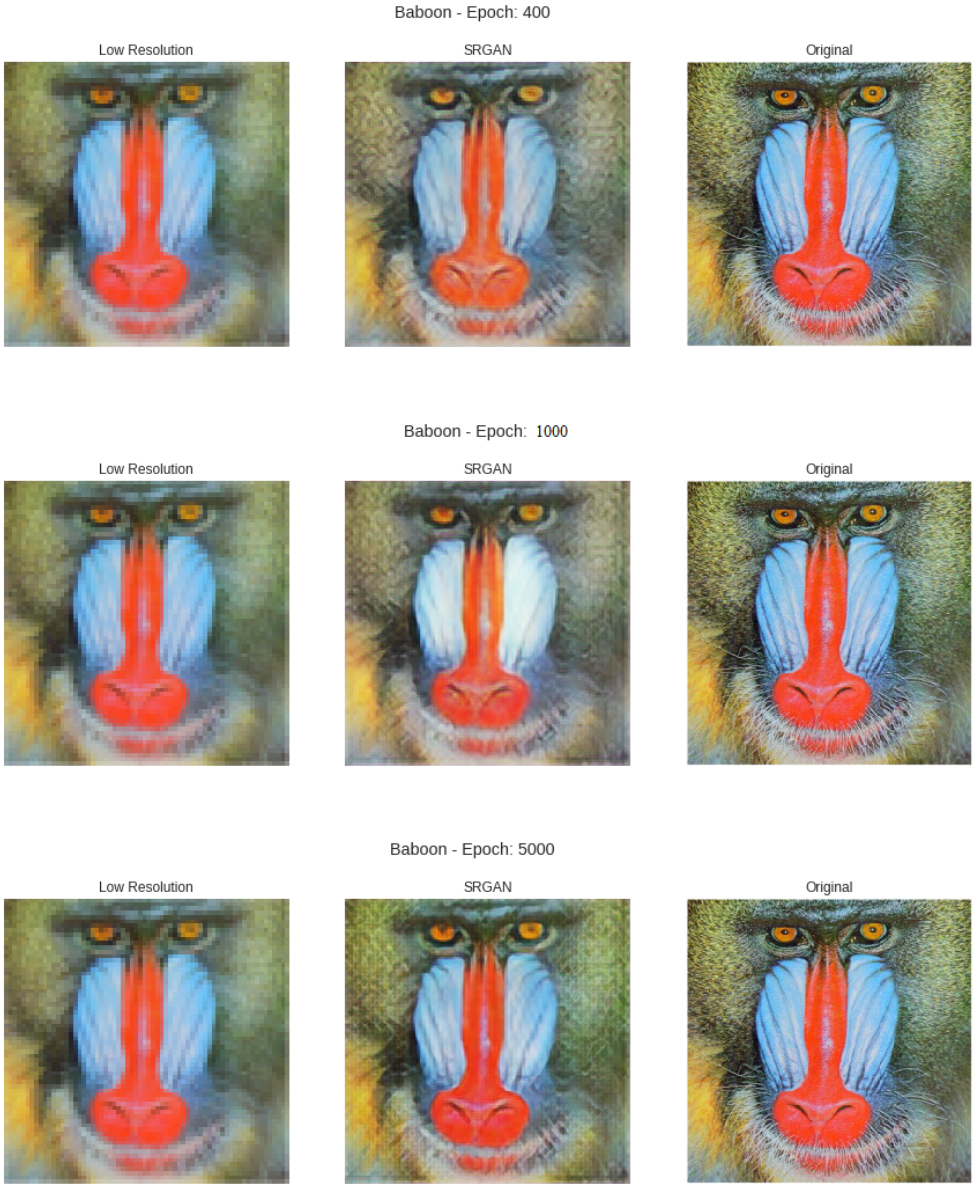


Figure 4: Test image results during training (Set2)

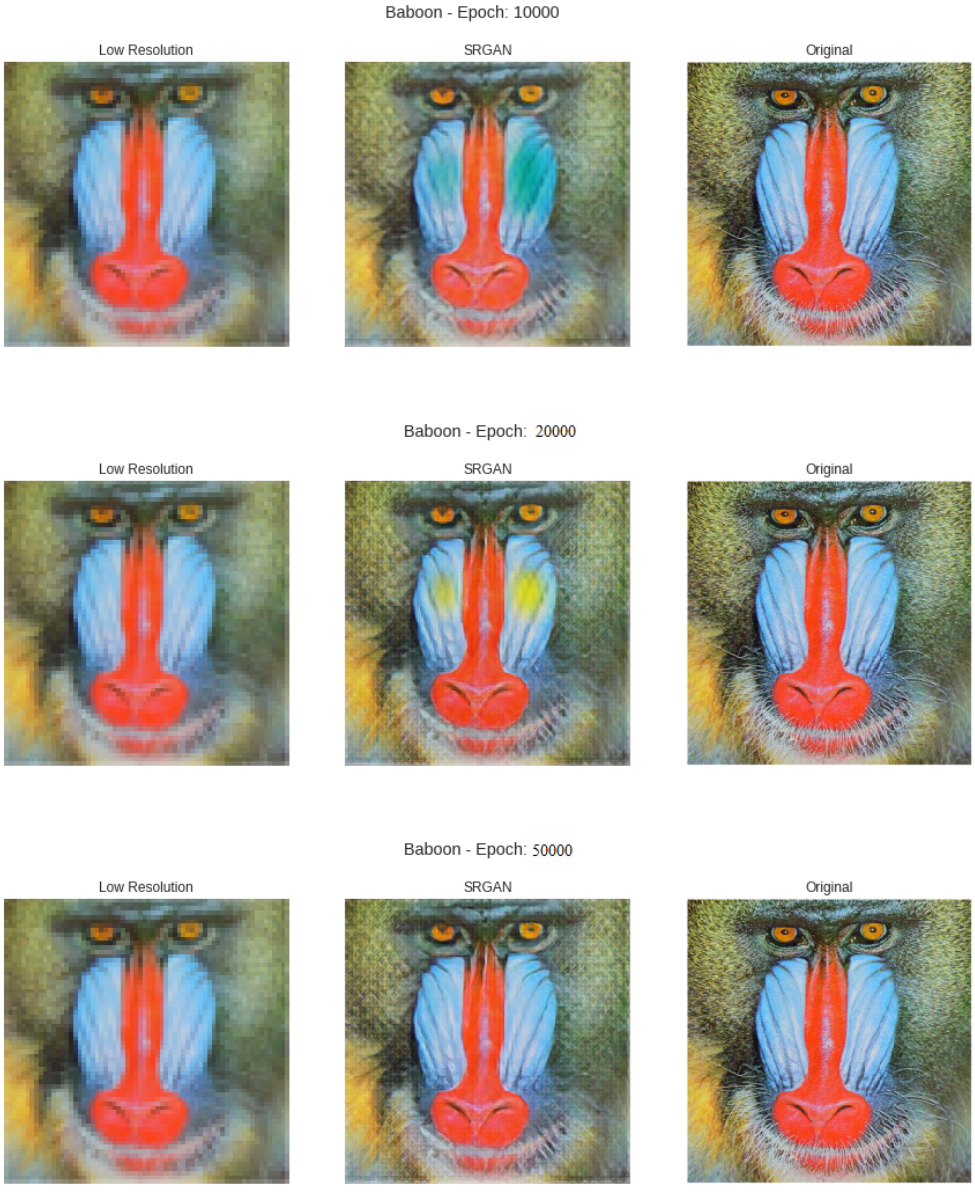
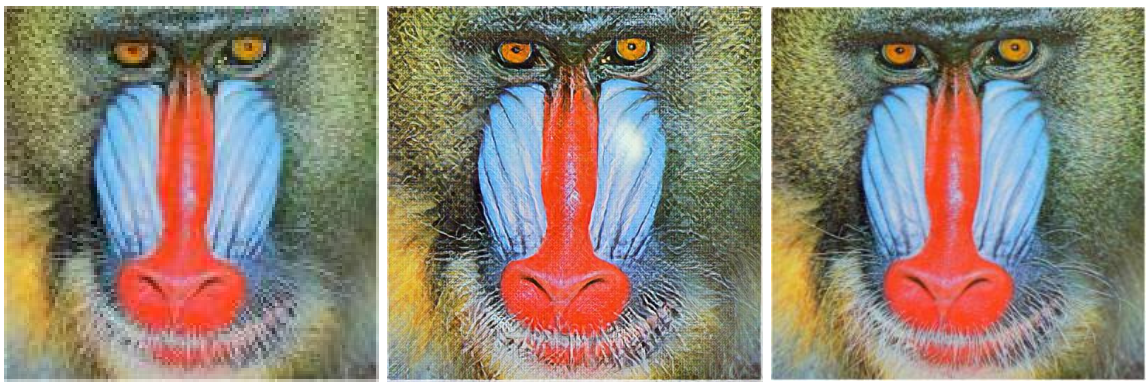


Figure 5: Test image results during training (Set3)

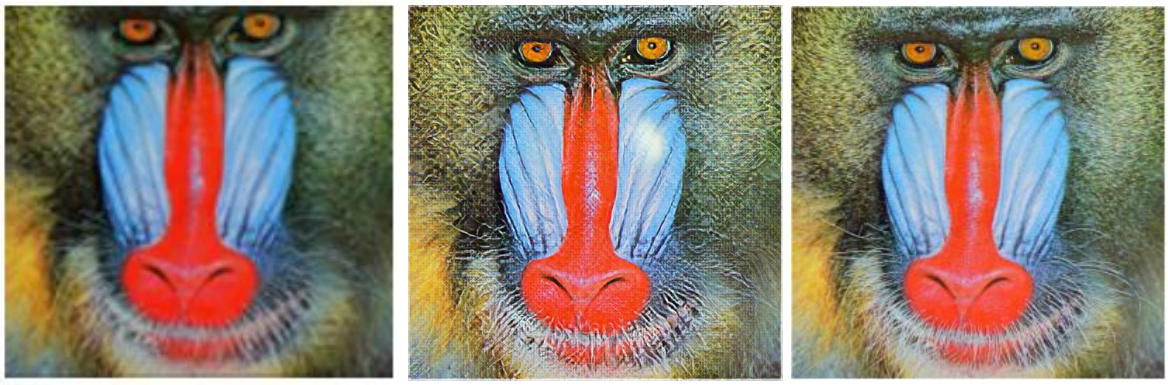


(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 6: Test images 2x (Set 1)

(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 7: Test images 4x (Set 1)



(a) Bicubic output image (b) SRGAN image (c) Original image

Figure 8: Test images 2x (Comparison)

(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 9: Test images 2x (Set 2)

(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 10: Test images 4x (Set 2)

(a) Bicubic output image (b) SRGAN image (c) Original image

Figure 11: Test images 2x (Comparison 2)

4 Discussion

As we can see in the section 3, the quality of the output generated by the SRGAN increased significantly as the number of epochs increased. After 1000 epochs the SRGAN learnt to regenerate main structure in the image. It still failed to regenerate the finer details or closely situated objects(Ex. eyes of the baboon). As we were using a single training set for the training purpose, the SRGAN saturated the blue color, causing it to increase the blue color brightness. This is evident from the Epoch 10000 image. We trained the SRGAN for 90000 epochs for learning rate 10^{-4} then we trained the same SRGAN with learning rate 10^{-5}. When we reduced the learning rate, the SRGAN was able to learn more finer details in the image. The SRGAN learnt to recreate finer objects like hair, whiskers in the super resolved images. This is evident in the $2\times$ magnification results. The trained SRGAN was also tested on SET 14 dataset. Some of the results are attached in the appendix A. The SRGAN failed to recreate the $4\times$ magnification of images properly. It was generating images with too much variance. We believe this can be resolved by training the images in small patches of original image rather than whole image.

5 Conclusion and Future Work

We confirmed that SRGAN has superior performance than bicubic interpolation method. It retains the finer texture of the original image. The project works very well for 2× magnification but needs some improvement for the 4× magnification. The texture quality of currently trained SRGAN can be improved by training it more on different set of images. The generator is now able to understand different textures and color variations in the input image. The reconstruction of the image is better than most of other image resizing or magnification methods. In future, this method can be trained for different set of images to learn a better understanding of textures in the low resolution input image.

References

[1] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network.," in *CVPR*, vol. 2, p. 4, 2017.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

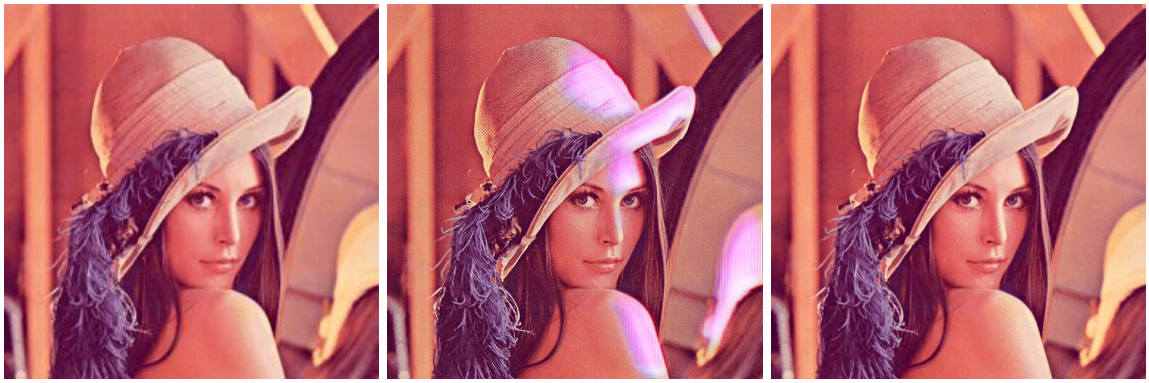
A Appendix

(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 12: Test images 2x (Set 3)

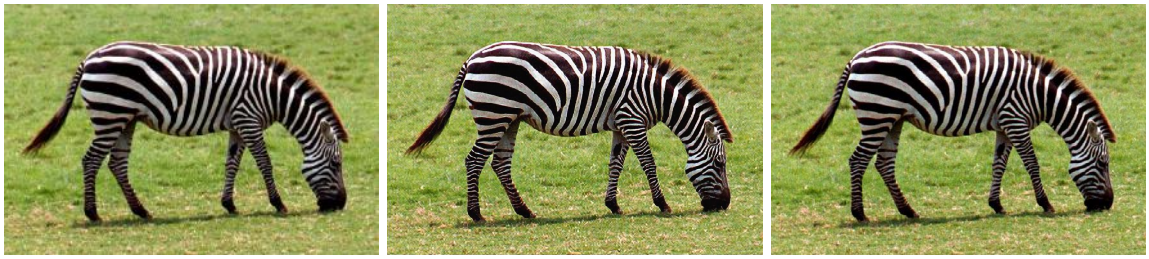
(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 13: Test images 2x (Set 4)



(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 14: Test images 2x (Set 5)



(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 15: Test images 2x (Set 6)

(a) Low resolution image (b) Super resolution image (c) High resolution image

Figure 16: Test images 2x (Set 7)