

# Single Image Super Resolution for Histopathological Images

## Project Report

Manoj Maddineni

B20CS034

Mentor - Angshuman Paul

Project partner - Aman Bhansali (B20ME010)

Deliverables - A software for super resolution of histopathological images

### **Abstract:**

*This paper reports my experience of building a software for Single Image Super resolution of medical images. We have used the kaggle BreakHis dataset which consists of histopathological slides of breast tumor tissue and has both malignant and benign samples. We first generate degraded images(LR) from original high resolution(HR) images available and then perform super resolution on them(SR). The results are evaluated based on the MSE and PSNR values.*

### **Acknowledgements:**

I would like to express my thanks to the people who have helped me most throughout my project. I am grateful to my mentor ,Angshuman Paul Sir, for non-stop support throughout the project. A special thanks of mine goes to my partner,who helped me out in completing the project by sharing interesting ideas, thoughts and made it possible to complete my project with all accurate information.

## Introduction-

Single image super-resolution (SR) aims at reconstructing a high resolution image from its low-resolution counterpart.

The earlier approaches include interpolation methods(like bicubic)[1], SRCNN[2], SRGAN[3][4][5], ESRGAN[6][7] among others. However, most of them don't work well in real world cases where the degradations are far too complex and unknown in nature.

The approach we have used in our project is the Real-ESRGAN [8]which is an extension of the powerful ESRGAN to restore real-world LR images by synthesizing training pairs with a more practical degradation process. The real complex degradations usually come from complicated combinations of different degradation processes, such as imaging system of cameras, image editing, and Internet transmission.

Hence, here the classical "first-order" degradation model is extended to the "second-order" degradation model for real world degradations and also to achieve a balance between simplicity and effectiveness.

## Methodology-

### 1. Degradation process

The below diagram gives an overview of the degradation model :

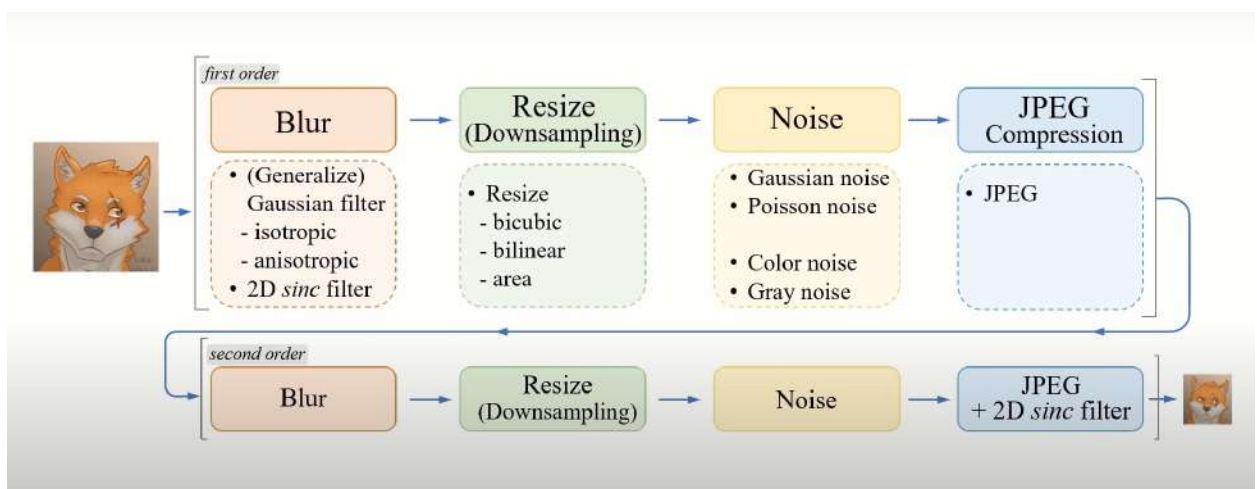


Figure-1.1 [9]

## 2. Network Architecture

Generator - Similar to that of ESRGAN

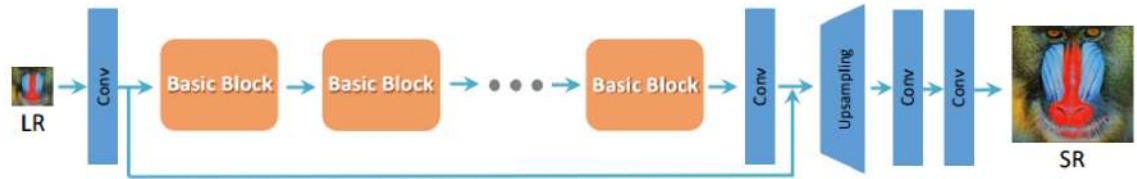


Figure 1.2 [9]

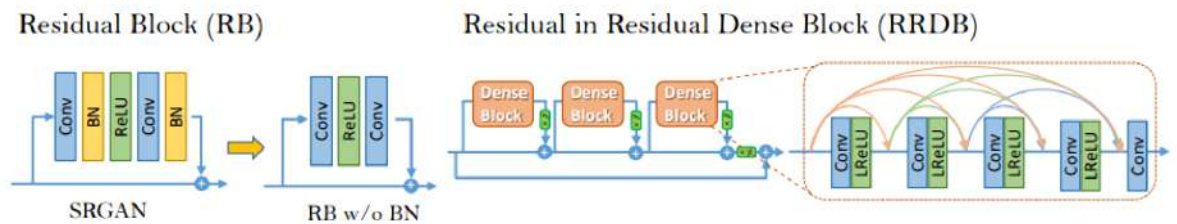


Figure 1.3 [9]

Discriminator -

As Real-ESRGAN aims to address a much larger degradation space than ESRGAN, the original design of the discriminator in ESRGAN is no longer suitable. Specifically, the discriminator in Real-ESRGAN requires a greater discriminative power for complex training outputs. Instead of discriminating global styles, it also needs to produce accurate gradient feedback for local textures. So a U-Net[10][11] discriminator is used. The U-Net outputs realness values for each pixel, and can provide detailed per-pixel feedback to the generator.

In the meanwhile, the U-Net structure and complicated degradations also increase the training instability. We employ the spectral normalization regularization to stabilize the training dynamic

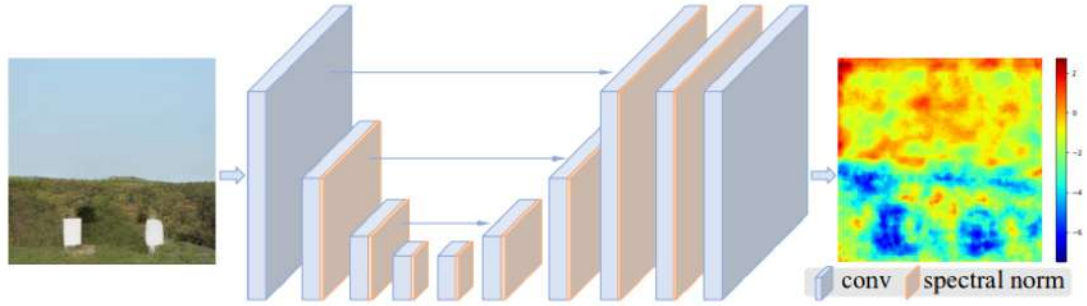


Figure 1.4 [3]

Architecture of the U-Net discriminator along with spectral normalization

#### - Training process

The training process is divided into two stages. First, we train a PSNR-oriented model with the L1 loss. The obtained model is named by Real-ESRNet. We then use the trained PSNR-oriented model as an initialization of the generator, and train the Real-ESRGAN with a combination of L1 loss, perceptual loss and GAN loss .

$$L1LossFunction = \sum_{i=1}^n |y_{true} - y_{predicted}|$$

$$l^{SR} = l_X^{SR} + 10^{-3} l_{Gen}^{SR}$$

Where,

$10^{-3} l_{Gen}^{SR} \Rightarrow$  adversarial loss

$l_X^{SR} \Rightarrow$  content loss

$l^{SR} \Rightarrow$  Perceptual loss(for VGG based content losses)

## Results-

### Evaluation metrics:

#### 1. Mean squared error(MSE):

Individual pixels of HR and SR are compared and the average of three channels(RGB) is taken as 'mse'.

$$(mse)_r = \frac{\sum_{i=0}^{rows} \sum_{j=0}^{columns} ((r_o)_{ij} - (r_s)_{ij})^2}{3}$$

$$(mse)_g = \frac{\sum_{i=0}^{rows} \sum_{j=0}^{columns} ((g_o)_{ij} - (g_s)_{ij})^2}{3}$$

$$(mse)_b = \frac{\sum_{i=0}^{rows} \sum_{j=0}^{columns} ((b_o)_{ij} - (b_s)_{ij})^2}{3}$$

$$mse = \frac{(mse)_r + (mse)_g + (mse)_b}{3}$$

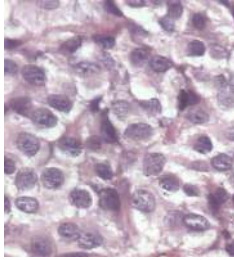
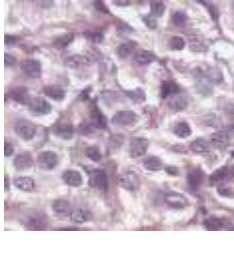
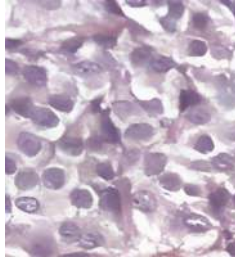
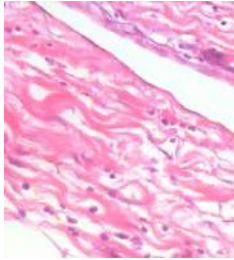
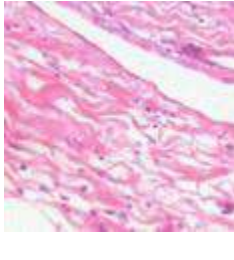
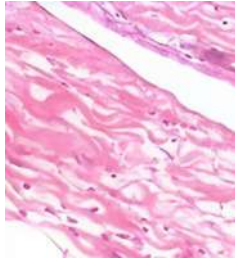
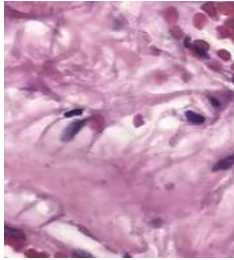
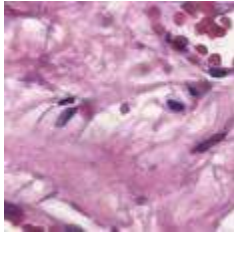
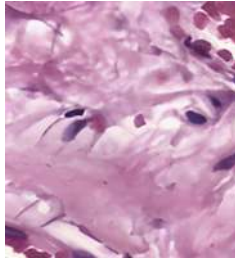
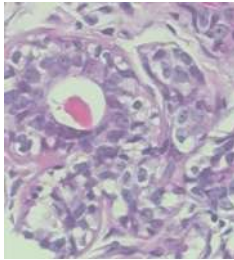
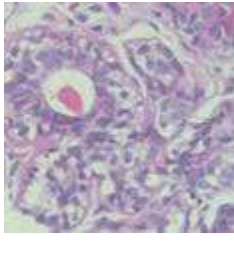
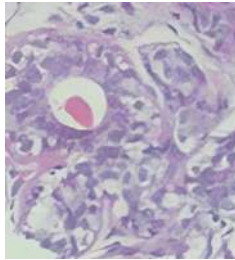
#### 2. Peak Signal to Noise Ratio(PSNR):

$$max_{pixel} = 255.0$$

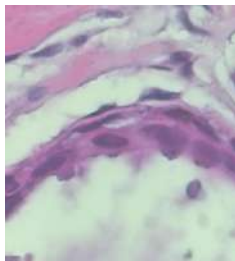
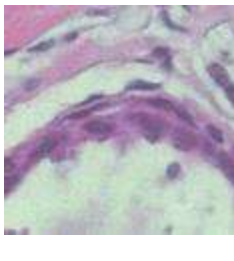
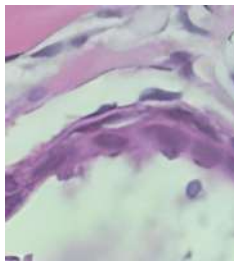
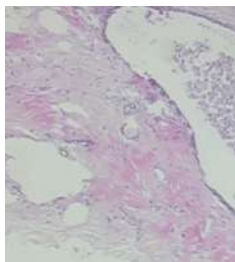
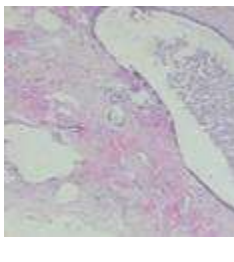
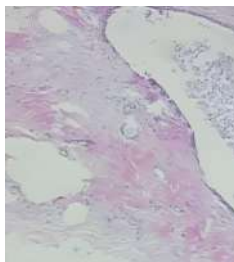
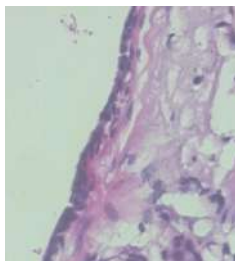
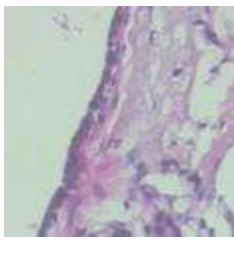
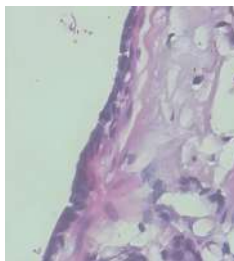
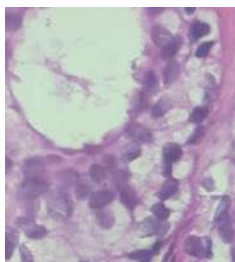
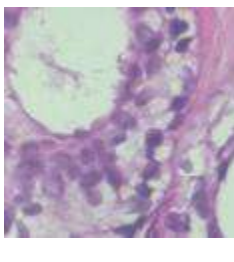
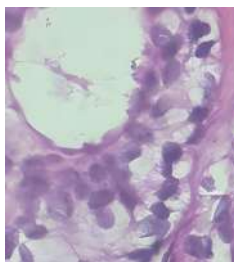
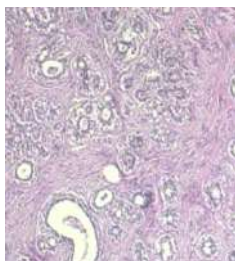
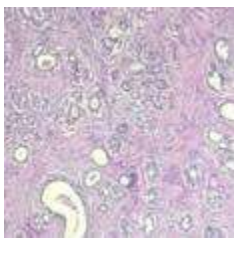
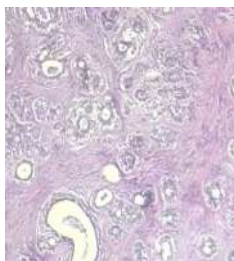
$$psnr = 20 \log_{10} \left( \frac{max_{pixel}}{\sqrt{mse}} \right)$$

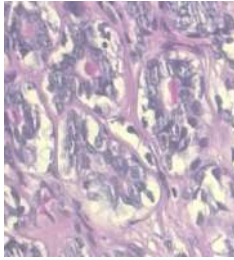
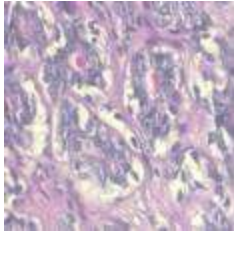
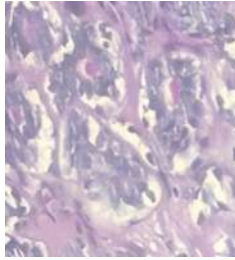
Comparison of HR(original image), LR(low resolution image) and super resolution image (SR)generated by using real-esrgan model -

Table-1.1

HR(Original)	LR	SR(Using real-esrgan model)	MSE(between the original HR and the generated HR)	PSNR
			34.36	32.77
			35.38	32.64
			24.028	34.32
			46.12	31.49



			18.07	35.56
			25.02	34.15
			15.78	36.14
			33.46	32.88
			85.57	28.81

			31.34	33.17
---	---	---	-------	-------

Note - Here all the images are shown on a similar scale but the LR can also be a small image, we can upscale the image(4X) with minimal loss of information.

The average value of MSE calculated for 1000 images = 47.78

The average value of PSNR calculated for 1000 images = 31.68

## Colab notebook -

<https://colab.research.google.com/drive/1GAmfPDrl17DD6wlwo2h5x1WHWo7C4Z8s#scrollTo=LK-A624Pqn2d>

## Link to more images and their super resolution versions -

[Real-Esrgan Results](#)

## References -

- [1] <https://github.com/rootpine/Bicubic-interpolation>
- [2] <https://github.com/tegg89/SRCNN-Tensorflow>
- [3] <https://github.com/tensorlayer/srgan>
- [4] <https://github.com/deepak112/Keras-SRGAN>
- [5] <https://github.com/leftthomas/SRGAN>
- [6] <https://github.com/xinntao/ESRGAN>
- [7] <https://github.com/peteryuX/esrgan-tf2>
- [8] <https://github.com/xinntao/Real-ESRGAN>
- [9] <https://arxiv.org/pdf/2107.10833v2.pdf>
- [10] <https://arxiv.org/abs/2002.12655>
- [11] <https://paperswithcode.com/method/u-net-gan>



