

IMAGE PROCESSING (67829) – EXERCISE 5

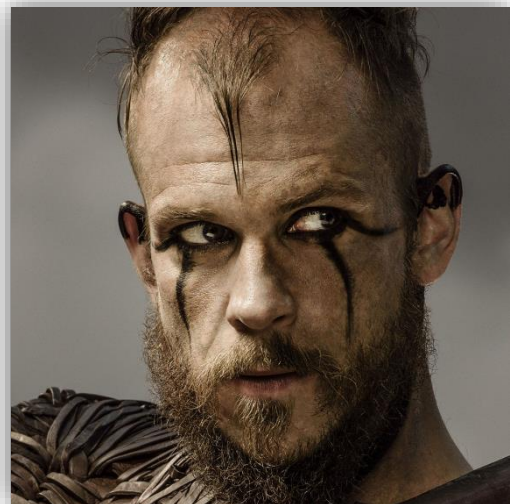
BY: AMIT HALBREICH, ID: 208917393

3.1 IMAGE ALIGNMENT

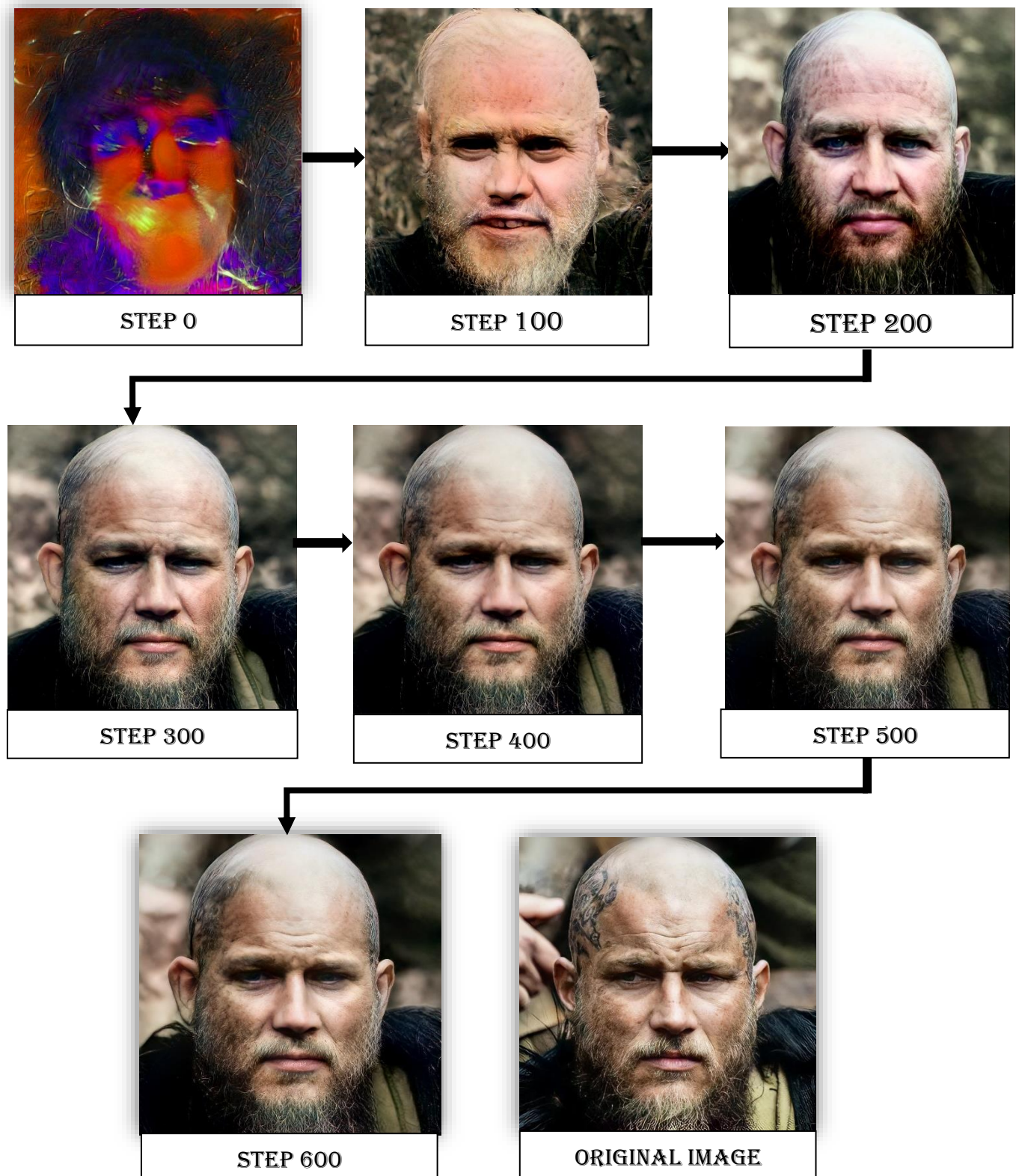
BEFORE ALIGNMENT



AFTER ALIGNMENT



3.2 GAN INVERSION – RAGNAR LOTHBROCK



The parameters I used for this run were:

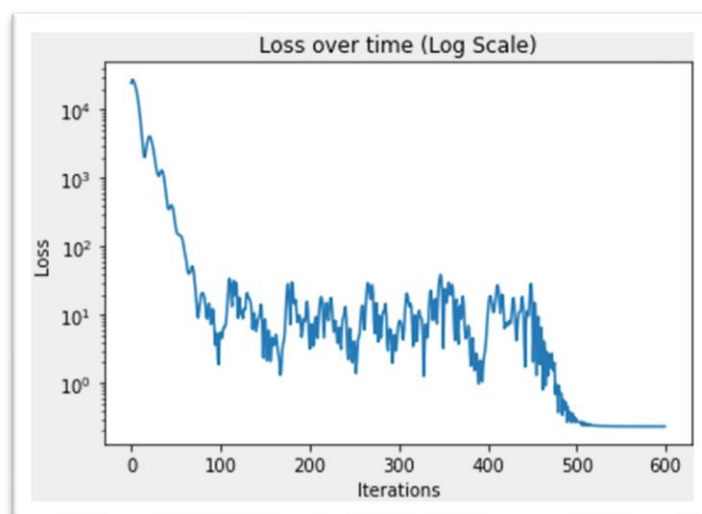
- $\text{num_steps} = 600$

The reason I used **num_steps** = 600 instead of the default 1000 is that the original image is ordinary, and we don't perform any special degradation, so the networks don't need a lot of iterations in order to produce a good reconstruction of the original image.

- $\text{latent_dist_reg_weight} = 0.001$

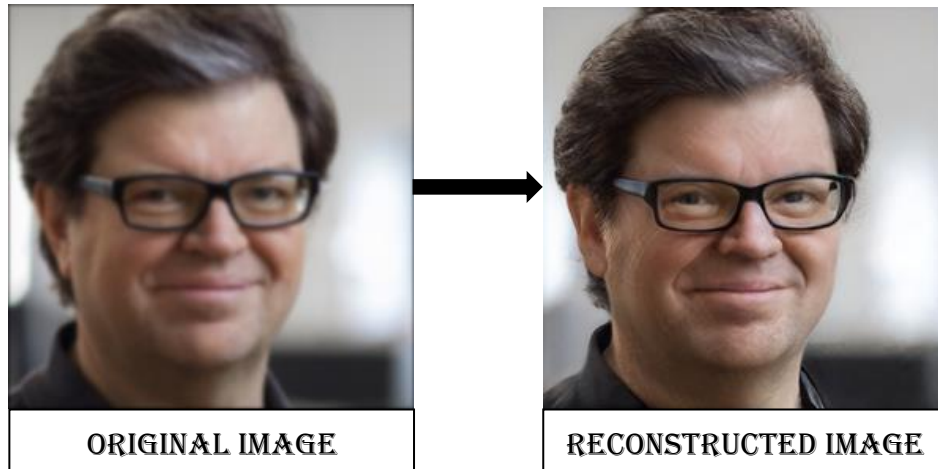
The loss function does not require a lot of iterations to converge. Similarly **latent_dist_reg_weight**, since the original image does not involve any noise or imperfection and overall resembles a real person – we can set this value to a lower bound.

LOSS OVER TIME (LOG SCALE) - GRAPH



3.3 IMAGE RECONSTRUCTION TASKS

3.3.1 IMAGE DEBLURRING – YANN LECUN



The parameters for this run were:

- `num_steps = 900`, `latent_dist_reg_weight = 0.25`

As can be seen in the loss per step graph, around 850 steps would have sufficed.

Since the deblurring process is the most complex process in this exercise because a lot of data is lost, I used a relatively large value for **latent_dist_reg_weight** to ensure better convergence and relatively high **num_steps**. With low values of **latent_dist_reg_weight** I found slightly worse deblurring results, so I kept it higher than usual.

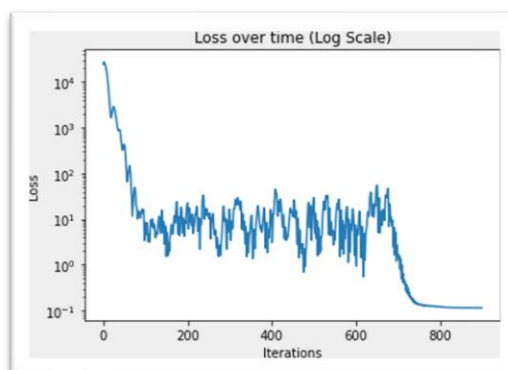
Blur parameters:

- Kernel size = 25, $\sigma^2 = 7$

The reason I used these blurring parameters is mostly from trial-and-error, I could see that the given image has strong blur effect, so I concluded it uses a large kernel with medium variance.

The variance value was interpreted from trial-and-error process completely and I used the parameters that produced best results.










LOSS OVER TIME (LOG SCALE) - GRAPH



3.3.1 –YANN LECUN – EFFECTS OF DIFFERENT BLUR PARAMETERS

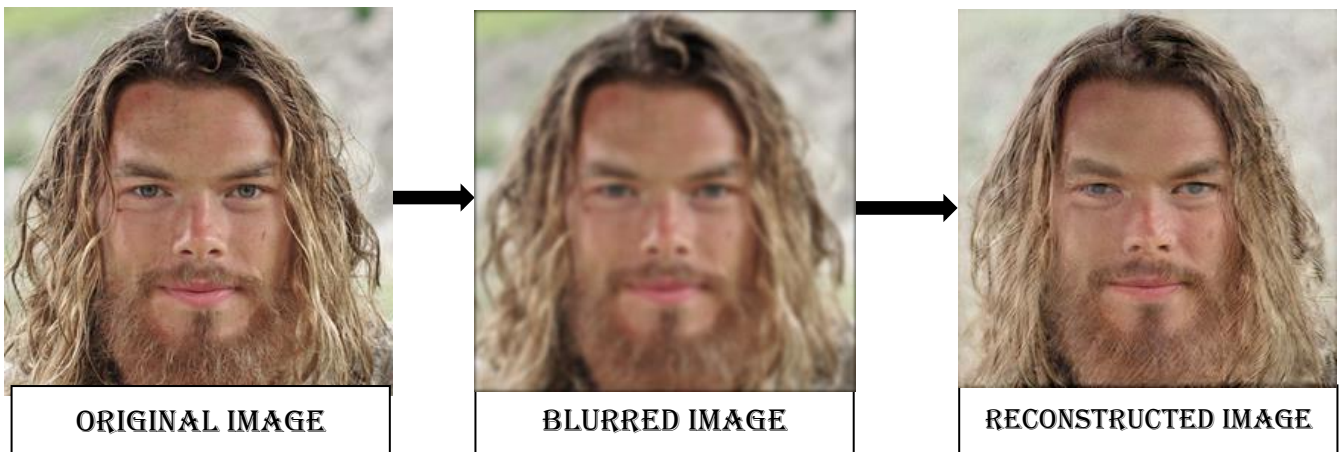
We can easily see there is a connection between kernel size and variance parameters on the reconstructed image output.

Below, a table of images reconstructed from the same image with the same **number of steps - 900** and the same **latent_dist_reg_weight - 0.25** but with different values for kernel size and variance:

	$\sigma^2 = 5$	$\sigma^2 = 7$	$\sigma^2 = 11$
KERNEL SIZE = 5			
KERNEL SIZE = 25			
KERNEL SIZE = 41			

As we can spot, the larger the kernel is, the more pixels are involved in the calculation of each pixel in the convolution, and the larger the variance value is, the more emphasis is given to spatial proximity in the convolution. This leads to the conclusion that for larger kernels and variance values, the output image will be sharper. We get good results usually for kernel size of 25 and variance of 7 so these are set to default.

3.3.1 IMAGE DEBLURRING – MY IMAGE – LEIF ERIKSON



The parameters for this run were:

- `num_steps = 900`, `latent_dist_reg_weight = 0.01`

As can be seen in the loss per step graph, around 800 steps would have sufficed.

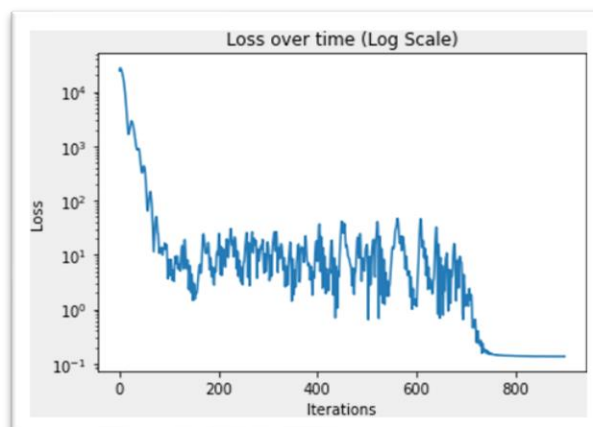
Since the blurring of the image is not very strong, the blurred original image, given as input for the loss calculation, is not very “far” (in terms of subtraction of the original from the blurred image) of from an average image that can be found in the dataset that the GAN was trained on, so we can use a pretty small **latent_dist_reg_weight** value.

Blurring parameters:

- $\sigma^2 = 7$, `kernel_size = 25`

The reason I used these blurring parameters is because we know what the blurring parameters were for the blurring of the original image, the ideal way to minimize the loss function, in theory, is to use the same parameters for kernel and variance to deblur the image and reconstruct it so it will be the closest to the original image as we can.

LOSS OVER TIME (LOG SCALE) - GRAPH



ABOUT DEBLURRING PROCESS ON THE EXERCISE:

The Image Deblurring part of the exercise was the most challenging part – I tried 2 different approaches:

The first was to create a kernel using only convolution of $[1,1]$ with itself like the filter we built in exercise 3.

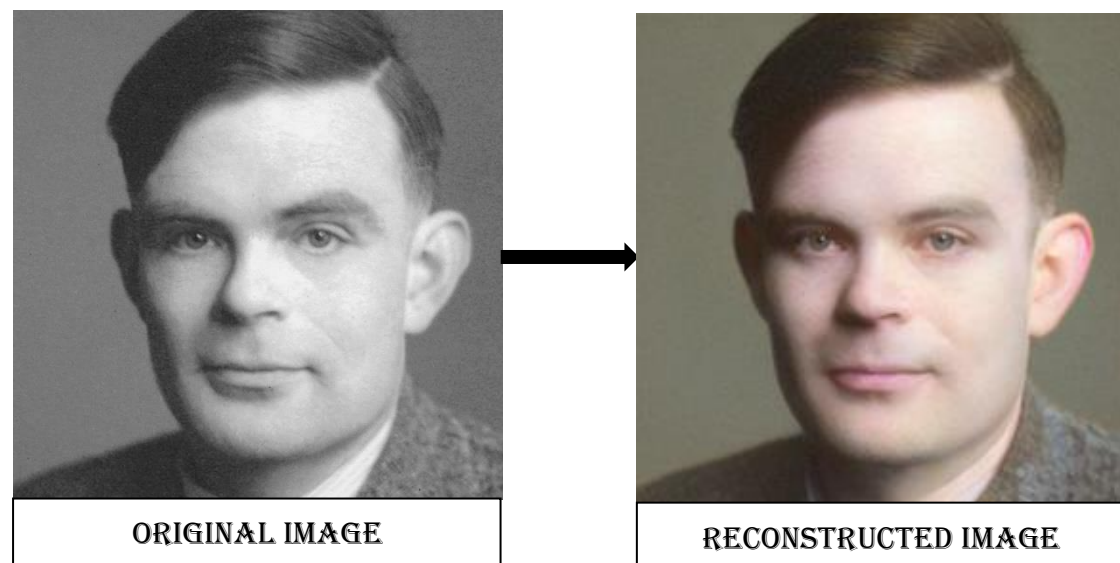
The problem was that I couldn't control parameters like variance and therefore the image sometimes would not reconstruct so easily and remained pretty blurry.

The second approach was to create a function that will construct a real gaussian like filter that will involve variance so the blurring effect could be controlled more easily.

Using kernel with varying variance that can be supervised helped me get better results – so it was a better option and image deblurring process went smoother.

In addition, I managed to portray the changes in variance and its effect on the final image reconstruction – I added this hyper parameter to the pool of parameters and tried it with different kernel size value to maximize the results.

3.3.2 IMAGE COLORIZATION – ALAN TRUING



The parameters I used for this run were:

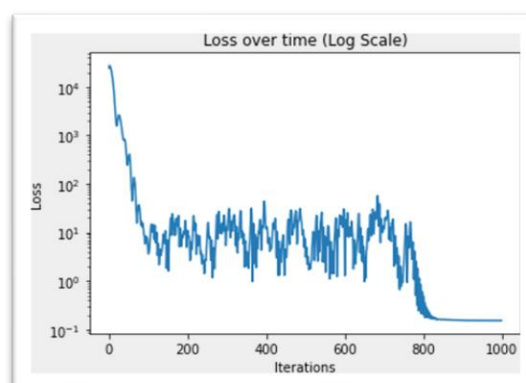
- $\text{num_steps} = 1000$,

As can be seen in the loss per step graph, around 820 steps would have sufficed but with 900 I didn't get good color results.

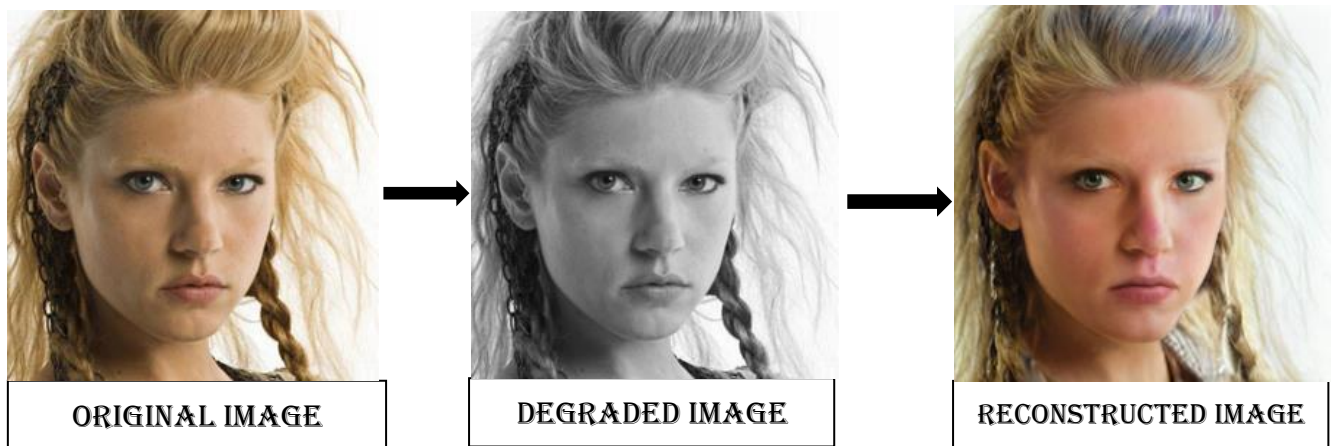
- $\text{latent_dist_reg_weight} = 0.27$

The reason for this relatively large value is that the grayscale degradation is very hard to reconstruct images with - we lose a lot of data and need to reconstruct the 3 RGB channels. Similarly, to the previous example, latent_dist was found mostly through trial-and-error, other values almost always resulted in the image having hues of pink, green, red weren't so realistic and does not resemble human skin color.

LOSS OVER TIME (LOG SCALE) – GRAPH



3.3.2 IMAGE COLORIZATION – MY IMAGE - LAGERTHA



The parameters I used for this run were:

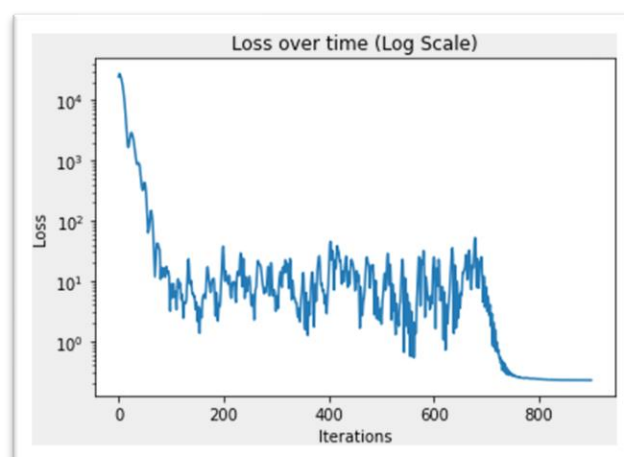
$\text{num_steps} = 900$

As can be seen in the loss per step graph, around 800 steps would have sufficed but less iterations than that wouldn't produce good enough colors.

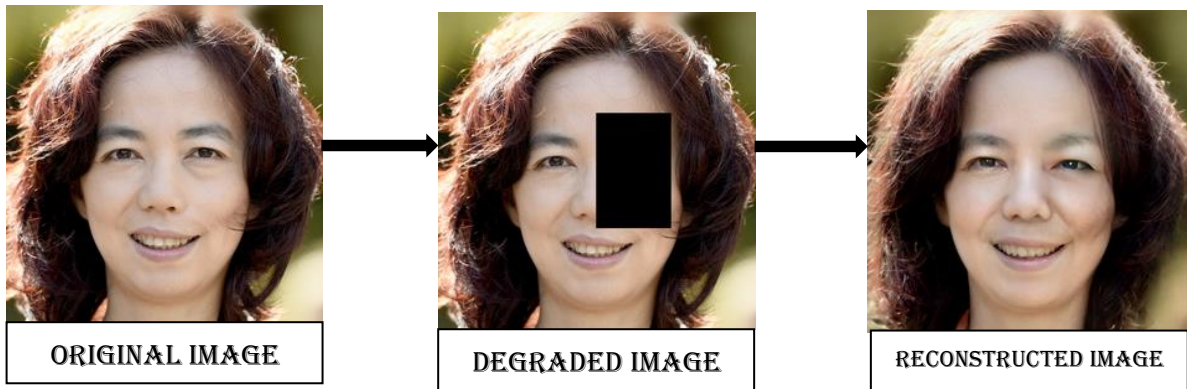
$\text{latent_dist_reg_weight} = 0.4$

This parameter was found mostly through trial-and-error, other values almost always resulted in the image having hues of pink, red, blue or green in areas that should have been human color-skinned. It was hard to find a good result with good and close enough colors since we lose a lot of data in this degradation – that's why I used the highest value in this exercise 0.4.

LOSS OVER TIME (LOG SCALE) - GRAPH



3.3.3 IMAGE INPAINTING – FEI FEI LI



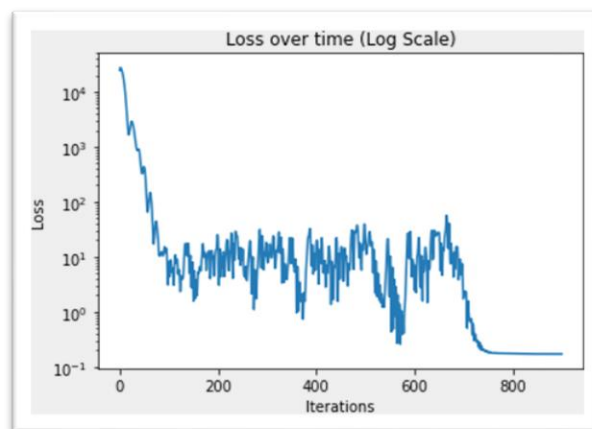
The parameters for this run were:

- `num_steps = 900`, `latent_dist_reg_weight = 0.01`

Exactly as before, the **num_steps** was chosen as 900 because we still need to run enough iterations such that the loss value will decrease and it is a pretty average choice that gives us good results as we can see above.

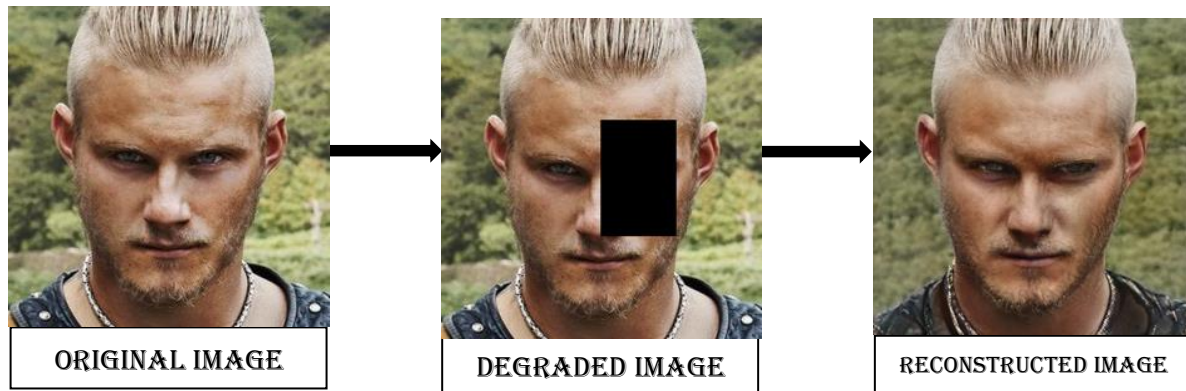
I used smaller **latent_dist_reg_weight** values. The reason for that is because the original image is very similar to face of a person as we know just with a missing part of the image. In comparison to the other two degradations – this is the degradation which leaves us with the most data - we lose only certain area in the image and the rest of it stays even after the degradation so it's easier to reconstruct.

LOSS OVER TIME (LOG SCALE) - GRAPH



As can be seen in the loss per step graph, around 800 steps would have sufficed.

3.3.3 IMAGE INPAINTING – MY IMAGE - BJÖRN IRNSIDE



The parameters I used for this run were:

- $\text{num_steps} = 900$

As can be seen in the loss per step graph, around 800 steps would have sufficed but I got better results using this number.

- $\text{latent_dist_reg_weight} = 0.05$

This parameter chosen, which is smaller relatively (in comparison to the other sections in the exercise) was chosen because the original image is very similar to face of a person as we know just with a missing piece, therefore a small latent distance should be enough to reconstruct the missing part efficiently. Also in this section, I changed the **latent_dist_reg_weight** and **num_steps** using trial-and-error process.

LOSS OVER TIME (LOG SCALE) - GRAPH

