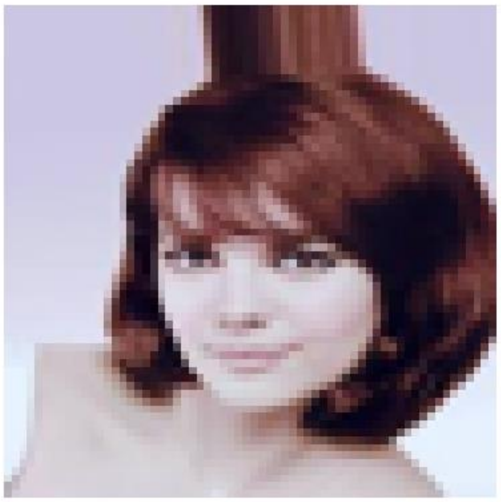
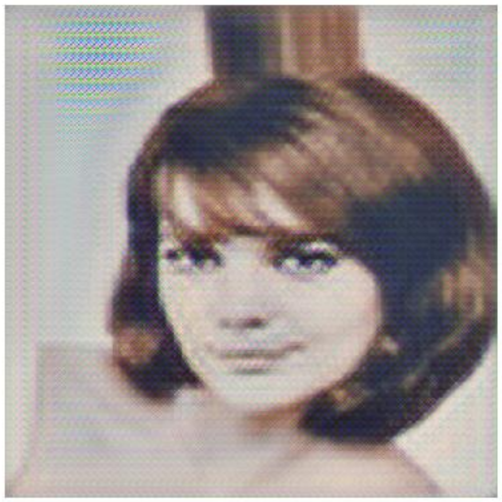
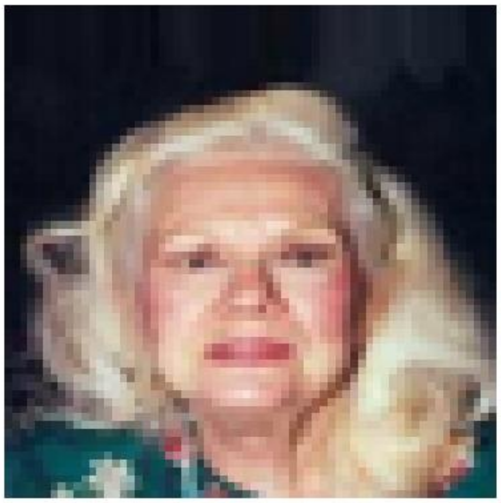
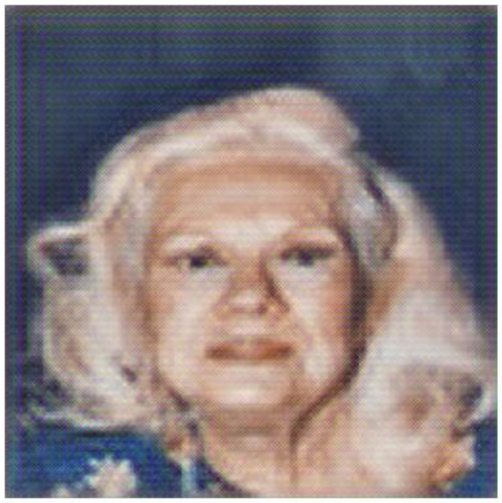
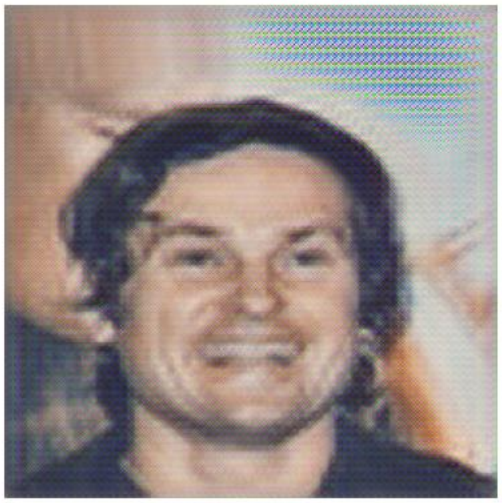
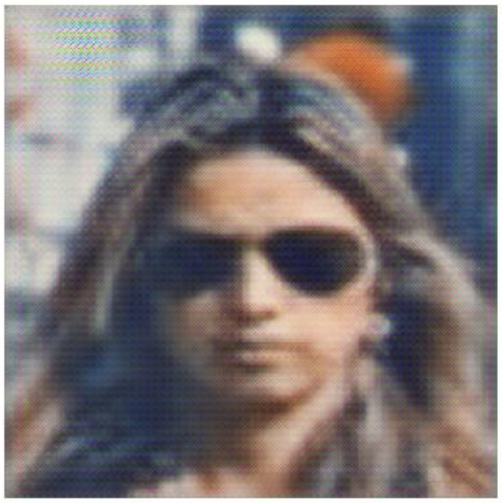
2. Upscaling image GAN in PyTorch

   I implemented the given code and used the same dataset from the paper. Due to computational restrictions (not having a dedicated GPU), I had to use only 100 images from the dataset for 10. Epochs but maintaining the 256x256 high resolution image size.
   Running the basic GAN training got the following metrics: D_loss=0.00609, G_loss=0.777

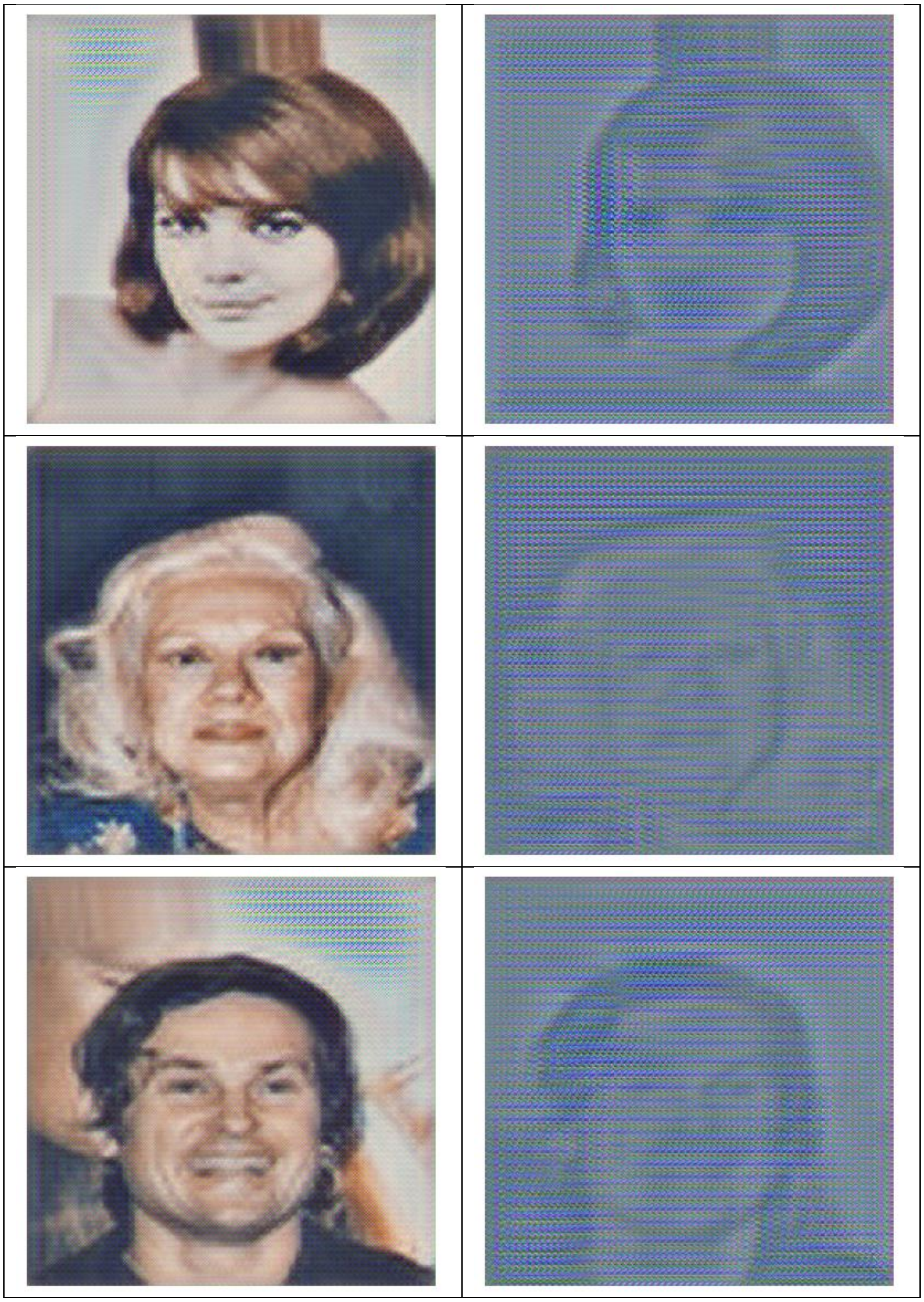
   input vs output comparison using the baseline trained model

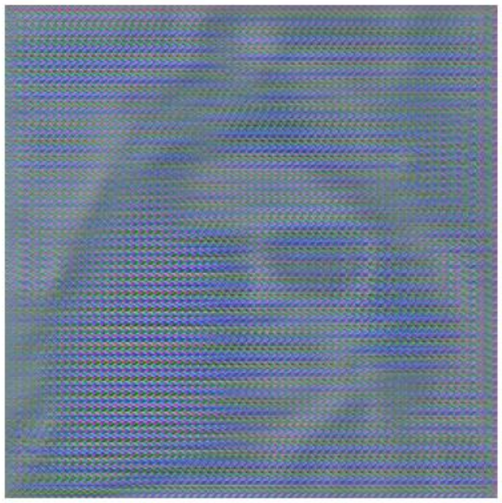
Input 64x64	Output
	
	

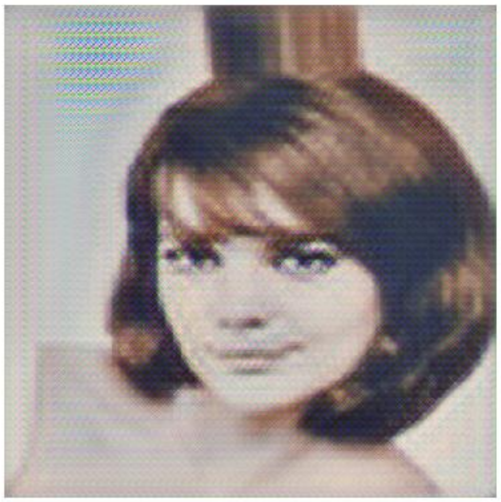
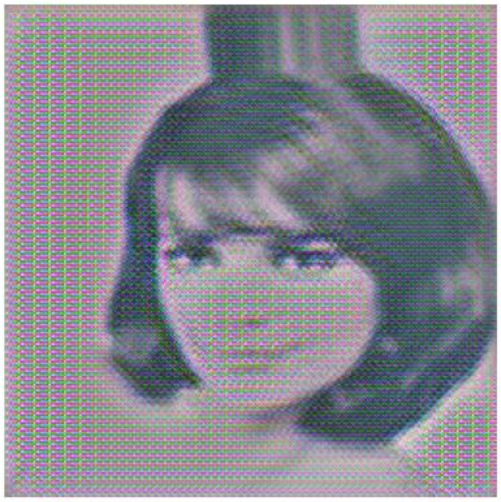
a. For the Generator, I chose to add another convolutional layer because it will allow the generator to better refine features or get additional details before unsampling. The output metrics: D_loss=0.00532, G_loss=0.911.
The output images of this modified generator model are kind of pattern images with a watermark of the original image. It clearly made the outputs worse than expected. The additional layer enhanced the wrong details and thanks to the watermarked original image, the discriminator was completely fooled.

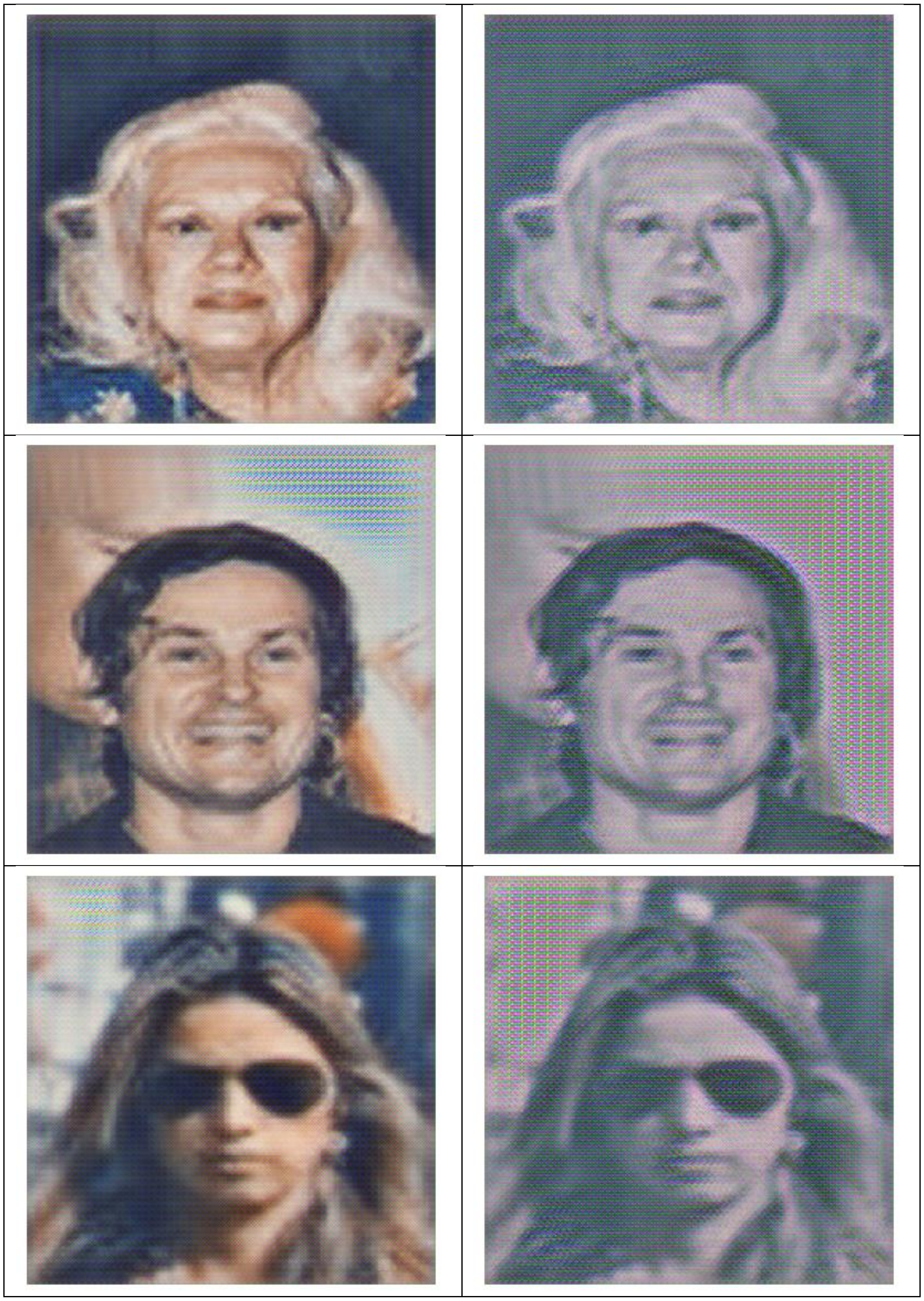
Baseline output	Modified generator output



b. For the Discriminator I also chose to add another convolutional layer since it will allow it to capture finer details in the image, making it harder for the generator to fool it. The output metrics: D_loss=0.0088, G_loss=0.899.
The output images of the modified discriminator model similar to the baseline output but with a sepia or vintage filter above. Apparently the discriminator put more attention to a different details rather than the color of the image.

Baseline output	Modified discriminator output
	



c. To obtain higher resolution images, I added another pixel-shuffle layer which will allow the generator to upscale the image to a 512x512 resolution.