APRML Project
Attribute Guided Facial Image Generation

Akshay Sethi, 2014133

Introduction : In this project we focus on realistic facial image generation while controlling the attributes present in the images. This problem is very important one as it can enable automatic photo retouching For E.g. to add desirable attributes like Smile and remove attributes like double-chin, bags under eyes from images. To achieve this we explore the use of Generative Adversarial Networks and using an encoder coupled with GANs achieve an accuracy of 79.4% on single attribute on the CelebA dataset and 83.8% on the IMDB-WIKI dataset.

Algorithms Used :

1.1st Interim Submission

The following work was done till the 1st Interim :

- **Generation of Facial Images using DCGAN**

  For the purpose of Generating Facial Images the DCGAN architecture was used. The architecture was trained for 200 epochs.

Figure 1 : Example Generated Images using DCGAN

The DCGAN paper gives guidelines on how to stabilize the training of Generative Adversarial Networks. To test the stability we tried searching over hyper-parameters and found that set of hyper-parameters given in paper are essential for the GAN training to stabilize.

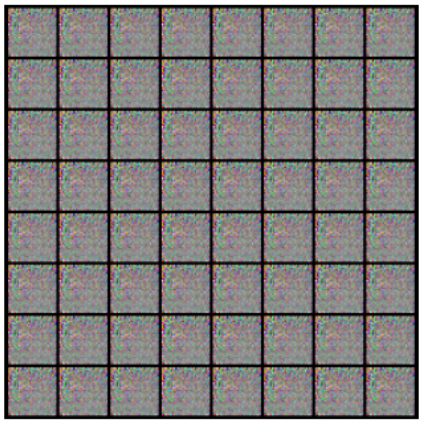


Figure 2 : Changing the optimizer from Adam to SGD the GAN does not converge

- **Generation of Facial Images using WGAN**

  Next we used another one of State of the art models known as WGAN(Wasserstein GAN) to generate Facial Images. Here the loss used is MSE instead of BCE (Binary Cross entropy) used in the above model.

Figure 3 : Example Generated Images using W-GAN

- **Generation of Facial Images with specific Attributes using C-WGAN**

  Next we used a Conditional GAN model conditioned on specific attributes like Smile and Gender.

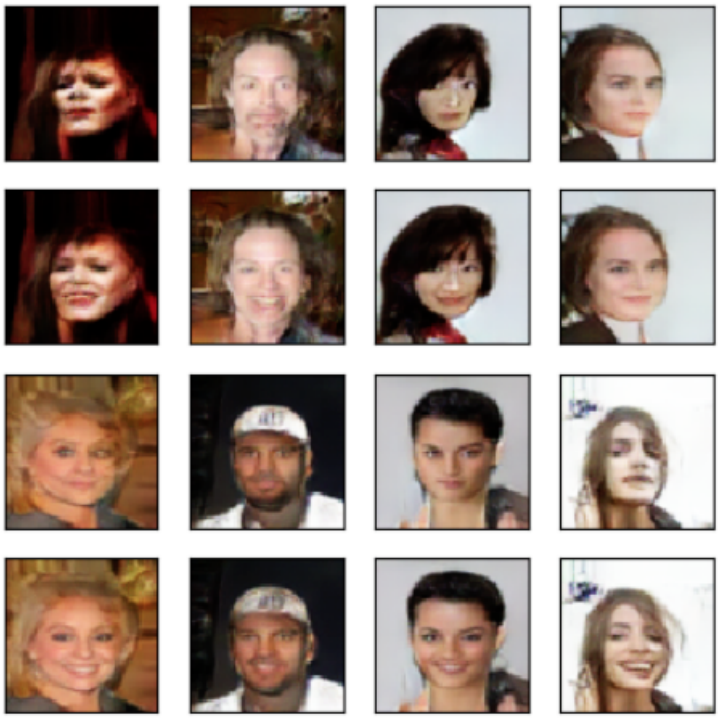


Figure 4 : Generating Images conditioned on the smile attribute. The first and 3rd row shows the absence of the attribute whereas the 2nd and 4th show presence.

2.2nd Interim Submission

The following was done till the 2nd Interim :

- **Baseline on Identity Preservation and Single Attribute Change**

  For Changing a single attribute, we use the **CycleGAN** model. The Cycle GAN is an unpaired image to image translation model. For training the model we take images which and have a certain attribute Absent and use the target image as those which have the attribute present.

- **Generation of Facial Images with Specific Attributes and Identity Preservation**

  1. We use the trained WGAN model in the 1st interim. We take the Discriminator and add a Fully Connected Layer at the end with 100 units to convert it to an encoder for inference in the latent space. Using the Discriminator weights are the Initialization we train it for 100 epochs. For training the Encoder we use MSE loss between the image input to the encoder and image generated by the Generator of the GAN.

$$L_2(i_1, i_2) = \sum_{j=1}^{n} (x_j - y_j)^2$$

Equation 1 : The loss between the input image i_1 encoder and the generated image i_2 from the Generator of the GAN.

2. Next for each attribute in the CelebA dataset we calculate the latent space vector(100-d) for each attribute. To do this, we subtract the average vector that does not have the specific attribute and from the average vector that has the attribute present. For example for the attribute Smile we calculate the attribute vector as **S = p – n** , where **p** is the average vector(encoder output) of all images which have the smile attribute present and **n** is the average vector(encoder output) of all images where the smile attribute is absent.

3.Final Submission

The following work was done till the Final Submission :

- **Varying the Encoder architecture for better reconstruction of Images**

  In the previous Interim, we had converted the Discriminator Architecture to Encoder. To improve the reconstruction we use Deeper Model for the Encoder.

  The models used are :

  Encoder Baseline

  FC - 1024
  FC - 512
  FC - 256
  FC - 100

  Encoder Basic Transfer – (Initializing the weights using the Discriminator)

  Key : Conv (n , n , f) n : size of filter
  f : number of filters

  Conv(3 , 3 , 128)
  BatchNorm()
  Conv (3 ,3 , 256)
  BatchNorm()

Conv (3 ,3 , 512)
BatchNorm()
Conv (3 ,3 , 1024)
Flatten()
FC – 100

Encoder Basic – Training from scratch

Key : Conv (n , n , f) n : size of filter
f : number of filters

Conv(3 , 3 , 128)
BatchNorm()
Conv (3 ,3 , 256)
BatchNorm()
Conv (3 ,3 , 512)
BatchNorm()
Conv (3 ,3 , 1024)
Flatten()
FC – 100

(Please note that Encoder basic and Encoder Basic Transfer are same models. Encoder Basic Transfer is initialized from the weights of the Discriminator whereas Encoder basic is initialized using Xavier Init)

Encoder Deep Transfer – (Initializing the weights using the Discriminator)

Key : Conv (n , n , f) n : size of filter
f : number of filters

Conv(3 , 3 , 128)
BatchNorm()
Conv (3 ,3 , 256)
BatchNorm()
Conv (3 ,3 , 512)
BatchNorm()
Conv (3 ,3 , 512)
BatchNorm()
Conv (3 ,3 , 1024)
BatchNorm()

Conv (3 ,3 , 1024)
Flatten()
FC – 100

- **Identity Preserving Optimization for better Reconstruction**

  While we do see that attribute can be treated as vectors as above, the reconstructed images still have some error. To improve the reconstruction error, instead of compute the loss in the image space we compute the loss after passing the input and generated image to a face recognition CNN like **FaceNet**. Now instead of computing the error between 64x64 images, we compute it between the 128-d embeddings of FaceNet.

$$z^*_{IP} = \underset{z}{\arg\min} \, ||FR(x) - FR(\bar{x})||_{L_2}$$

Equation 2 : L2 loss after Identity preserving optimization

- **Compute the efficacy of the model using discriminatively trained Convolutional Network Models**

  To show the efficacy of the Encoder-GAN we trained the VGG-Face network on the CelebA dataset for the purpose of attribute prediction.Next we take the trained CNN model and test it on the Generated Images to see whether the attribute we want present in the image is present or not.

  The Accuracy of the VGG-Face Network Adapted to Attribute Classification is 88.69%.

Datasets -

CelebA : This is a large scale celebrity faces dataset with 2,02,621 images. Each image is annotated with 40 binary attributes such as Smile , Gender , Double Chin etc. The images in this dataset cover large pose variations and background clutter. The dataset contains 10,000 identities each with 20 images. It is divided into 3 parts : Train : 160K images, Validation : 20K images and Test : 20K images.

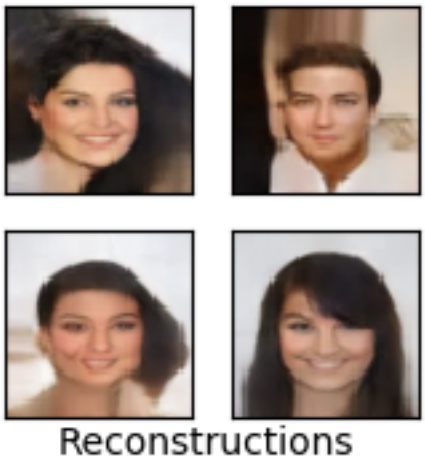
IMDB-Wiki : This Dataset contains 5,23,000 facial images scrapped from websites like IMDB and Wikipedia. Each image has annotations on Gender and Age of the image. The Dataset consists of the most popular 1,00,000 actors listed on the IMDB Website.

Results :

Image Reconstructions using Various Encoder Models
(Image Fed to encoder to get latent Code and then fed to Generator to output image. In this sense Generator Acts as a Decoder here.)

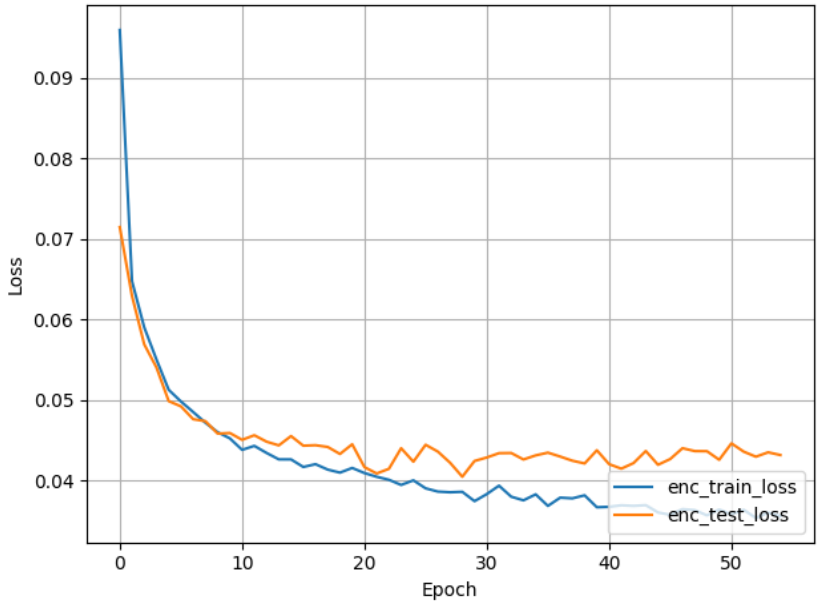
Using Encoder Baseline Model

Using Encoder Basic Transfer Model



Using Deep Encoder Transfer Model with Identity Preservation

Loss Curve of the Deep Encoder Transfer Model with Identity Preservation



Adding a Single Attribute to Images

For IMDB-WIKI Dataset: Age < 30 was considered young whereas Age > 40 was considered old

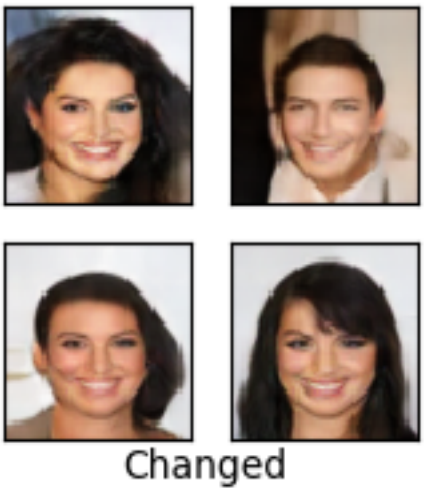
Evaluation Protocol : From the test set, all images which had a particular attribute absent were taken and that particular attribute was Added using Encoder GAN. Next the output images were tested using trained CNN model for the presence of the attribute. The reported accuracies are averaged accuracies over all the 40 attributes.

Model	CelebA	IMDB-WIKI
Encoder Baseline	56.2%	58.9%
Encoder Basic	65.8%	70.3%
Encoder Basic Transfer	68.1%	71.5%
Deep Encoder Transfer with Identity Optimization	79.4%	83.8%

Visual Results with Deep Encoder with Identity Preserving Optimization

No Specs to Specs

No-Smile to Smile



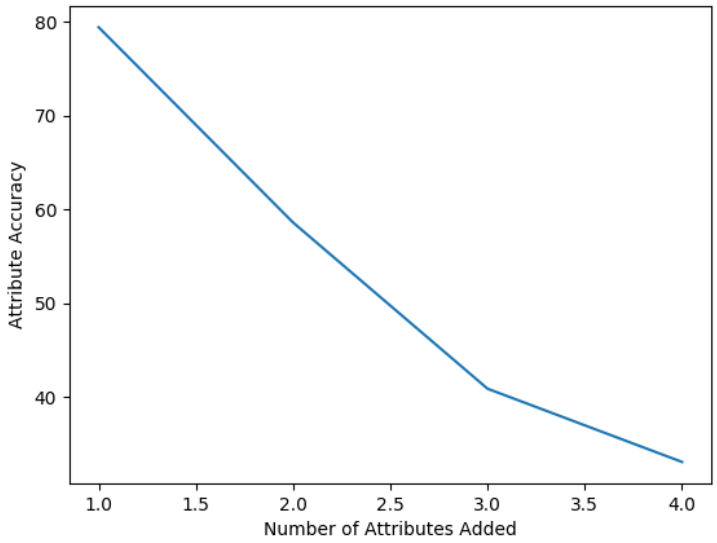
Old to Young

(Random Images taken off the Internet)

Visual Results with Multiple Attributes

Changed Image = Original Image + Eyeglasses + Male + Moustache

Performance with Multiple Changed Attributes

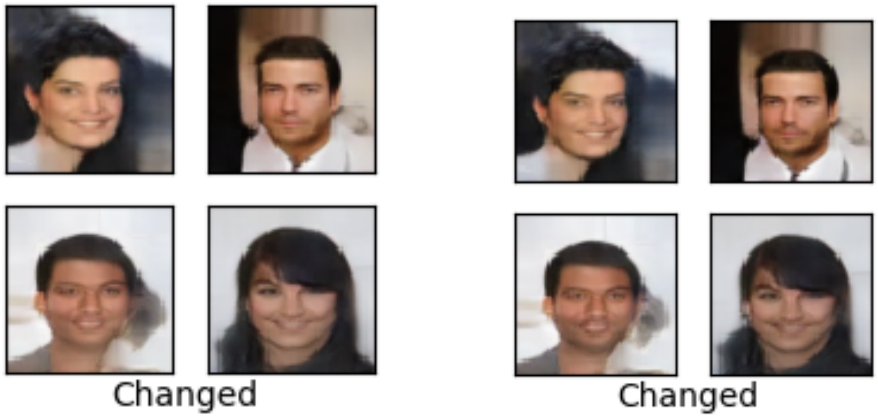


Visual Results with Intensity of Attributes

(Latent = Latent + I*Attribute , Here I is the intensity factor)
Varying the Intensity factor from 0.5 to 2.5

Moustache

I = 0.5 **I = 1.0**



l = 2.0

Changed

l = 3.0

Changed