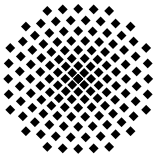

University of Stuttgart
Institute for Signal Processing and System Theory
Professor Dr.-Ing. B. Yang

Research Project S1408

Conditional GAN based Mask Guided Portrait Editing

Author: Prabhakar Kumar Panday

Date of work begin: 25-10-2021
Date of submission: 09-05-2022

Supervisors: Mr. Florian Strohm
Mr. George Basem Fouad Eskandar

Keywords: Portrait Editing, Digital Image Processing, Computer Vision

Abstract

This thesis investigates geometry-guided technique using semantic facial mask as a shape guide for high-level facial component editing. The framework built in this research work leverages conditional GANs directed by supplied face masks for learning individual facial feature embeddings and the facial style. The generated images display high diversity, quality and the framework provides very good controllability on the features and the style of the generated images. The framework is capable of generating new faces from a single fixed mask, transfer style form one face to other, edit individual facial components of the image, and copy facial features from one face to another. By changing the feature extracting technique from the facial masks for training, this framework gives more control over the generated image when compared to original research work ([1]).

Contents

1. Introduction

Portrait editing is a popular subject in photo manipulation, it is of great interest in the vision and graphics community due to its potential applications in movies, social media, professional photography and many others. Generative Adversarial Networks(GANs) [2] have made tremendous progress in synthesizing realistic faces [3], face aging [4], pose changing and attribute modification [5]. Three widely used approaches for portrait editing are label-conditioned method [6], reference-guided method [7], and geometry-guided method [4]. This research project will focus on the geometry-guided technique using semantic segmentation of facial mask as a shape guide for high-level component editing.

This project will continue work based on the paper by Gu et al. [1], a framework is built using a local embedding sub-network to learn five features of the reference facial image by using Autoencoders 2.1.2 and then combine the learned component feature embedding and target facial image mask using a mask guided generative sub-network to generate the foreground face image. The framework is capable of synthesizing diverse, high-quality, and controllable facial images from given masks. The face mask provides very good facial constraint and serve as a guide of face generation and on the other hand local embeddings provides accurate contour for each facial component (e.g., nose, eyes, mouth, hair etc). Previous research work on portrait stylization [8, 9] shows promising results, but they focus on style transfer rather than component editing.

Some of the GAN based works like pix2pix [10], pix2pixHD [11] and BicycleGAN [12] provide better image-to-image translation by making use of face masks for image editing and generation. However, these methods are not good at changing the emotions of the generated image from the source image. They also suffer from quality issue in the hair and background regions. This research work aims to build a framework capable of image-to-image translation which address problems like diversity, quality and controllability of the generated images. For image translation, diversity necessitates the learning of good correspondences between picture pairings that may differ in lighting, postures, colors, different ages, and gender. Fine face details, like further improvement in hair, and face background is related to improving the quality. Having more control over local face components is also an essential requirement of the research work.

In this research work, a conditional GAN based framework is built and trained for portrait editing with the help of face masks. The framework can be divided into three major components: *local embedding sub-network, mask-guided generative sub-network, and background fusion sub-network*. The *local embedding sub-network* consists of five auto-encoder networks that encode embedding information for five face components: "left eye," "right eye," "mouth," "skin nose," and "hair." The *mask-guided generative sub-network*

recombines the local embeddings from the local embedding sub-network and the target face mask to produce the foreground face picture. The target face mask gives accurate component shape for image reconstruction and component level correspondence (e.g., mouth-to-mouth, hair-to-hair, etc.). Finally the *background fusing sub-network* pastes the generated image over the original background of the target image.

By using the fask of the facial images, the framework can be used for various applications like mask-to-face, style transfer like change skin color, remove facial hair or add facial hair, change the emotion of the image, edit individual facial features like changing eye, hair, mouth etc. The framework is capable of generating high quality facial images.

1.1. Related Work

Generative Adversarial Networks:

There is a significant amount of research on applications of Generative Adversarial Networks (GANs) in various fields [2]. GANs belong to the realm of unsupervised learning, whose main task is to learn the underlying probability distribution of a given dataset, it forces the generated samples to be indistinguishable from the given target distribution. In the field of computer vision, GANs are extensively used for applications like image synthesis [13, 14], image translation [10, 11, 15, 12], and representation disentangling [16, 17, 5], among others. Another flavor of GANs which is conditional GAN models [18] where along with the training data the network is also fed with 'y', upon which both the generator and discriminator are conditioned. These can be conditionally trained on masks to generate images from that mask. This research work is an attempt to leverage the locally embedded information for individual facial components using their masks and to generate high quality portrait images with greater diversity and better controllability.

Deep Visual Manipulation:

Deep neural networks have ample applications in image editing and manipulation, including deep analogy [19], super-resolution [20], image completion [21], and sketch based portrait editing, viz. faceshop [22] and scribbler [23]. Mask-guided image editing methods like pix2pix [10], photographic image synthesis with cascaded refinement networks [24], and pix2pixHD [11] which demonstrated various approach to synthesize photographic graphical images using semantic label maps. This work is influenced by visual attribute transfer methods, such as style and color transfer [25, 26, 27]. The approach used in this work enhances on face swapping methods [28, 29] by assisting explicit face and hair swapping, owing to the structure that disentangles and re-combines facial instance embeddings with face masks.

Non-Parametric Visual Manipulation:

Splitting and stitching together existing patches from the original dataset to synthesize new images [30, 31, 32] and using neural networks to improve the quality of the stitched images [33] are the different non-parametric image synthesis methods. Unlike the previous techniques, the local embedding sub-network in this research work encodes facial instances as embeddings rather than image patches. Instead of warping and stitching image patches together, this method generates new face images using a mask-guided creative sub-network. By jointly training all sub-networks, this model produces higher-quality facial images than non-parametric methods.

1.2. Motivation

The motivation for this research project work is to use the facial mask of a reference image to guide image generation using conditional GANs [18]. A face mask provides a good geometric constraint for facial features like nose, eyes, lips, skin, and hair; which helps in synthesizing realistic faces. Works based on face masks [8, 9, 12] achieve promising results, but they do not synthesize different faces, suffer from quality issues, like lack of fine details in skin, difficulty in dealing with hair, background blurring, and their diversity is limited to color or illumination. The goal of this research project is to build a framework based on conditional GANs [18] for portrait editing. This framework will be used to edit the facial features of a target image driven by a reference facial mask.

2. Background

2.1. Deep learning

Deep learning is a subfield of machine learning that addresses the question of how to learn from prior experience. While the machine learning field has produced a variety of learning algorithms, deep learning is only concerned with neural network architecture. Because of their arbitrary complexity and large representation capability, neural networks have been widely and successfully applied in a variety of tasks, including machine translation, speech processing, and computer vision. Deep learning architectures have evolved into many forms over time to be suitable for a specific task, making it impractical to describe the best architecture for all tasks. As a result, we only describe the most fundamental blocks used in this thesis work.

2.1.1. Convolutional Neural Network Architecture

Convolutional Layer:

The central component of a CNN is the convolutional layer, it's where the majority of the computation occurs. The hyper-parameters of the convolutional layer are as follows: **F** denotes the field size, **K** denotes the number of filters, **S** denotes the stride, and **P** denotes the amount of padding. The input matrix is first padded with zeros in accordance with the parameter P. The dot product between the kernel elements and the input matrix elements is then computed by performing convolution on a window kernel within the input matrix. This procedure generates **K** two-dimensional feature maps. It should be noted that the number of trainable weights can be reduced by employing a parameter sharing scheme. This scheme requires all kernels within the same depth to use the same weights and bias. As a result, the number of weights becomes **K·F·F·D**, which is reduced by a factor of **W2·H2** (where **D** - the depth of the input matrix, **W2** and **H2** - the width and height of the resulting feature map).

Pooling layer:

Pooling layers are often placed between CNN layers and are used to reduce dimensionality of the input matrix, thereby controlling the size of the entire network and help to avoid overfitting, it also helps in reducing variance and extract low level features from nearby pixels(Only in the case of Average Pooling). The pooling layer, like the convolutional layer, uses a kernel window of size F. This kernel is convolved across the input matrix with S

- stride, but instead of the dot product, it outputs a single value depending on the type of pooling layer. It should be noted that a pooling layer has no learnable parameters. In this research work two pooling operations are implemented.

- **Max pooling** picks the maximum value from the span of the kernel and the operation is not differentiable, thus only directs the gradient of the largest element during backpropagation.
- **Average pooling** layer computes the average of all the pixels in the span of the kernel.

Fully-connected layer:

A single vector is the desired output of a forward pass through a neural network. This vector can represent a variety of meanings depending on the input data. The output of the convolutional and pooling layers, on the other hand, is a matrix. As a result, it is common practice to first flatten this matrix into a one-dimensional vector. This vector is then multiplied by a fully-connected layer weight vector to produce an output vector of the desired size.

Residual Networks:

Residual Networks, or ResNets, learn residual functions with reference to the layer inputs, instead of learning unreferenced functions [34]. In deep CNN network, the performance starts to degrade after certain depth due to vanishing gradients. Resnet introduce skip connections, allowing gradients to flow straight from later layers to initial filters. The number of residual blocks in the global generator is Nine and in the local enhancer network is three, and can be changed by passing it as an argument during training.

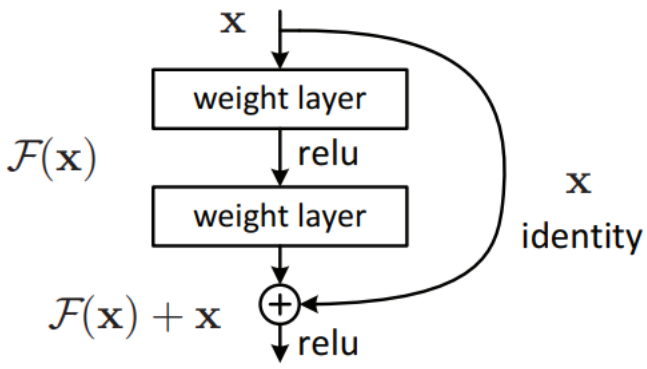

Figure 2.1.: Residual learning: a building block

2.1.2. Autoencoder

The Autoencoder(AE) is a type of self-supervised learning model which is primarily used in dimensionality reduction technique, it contains two major parts, namely an encoder and

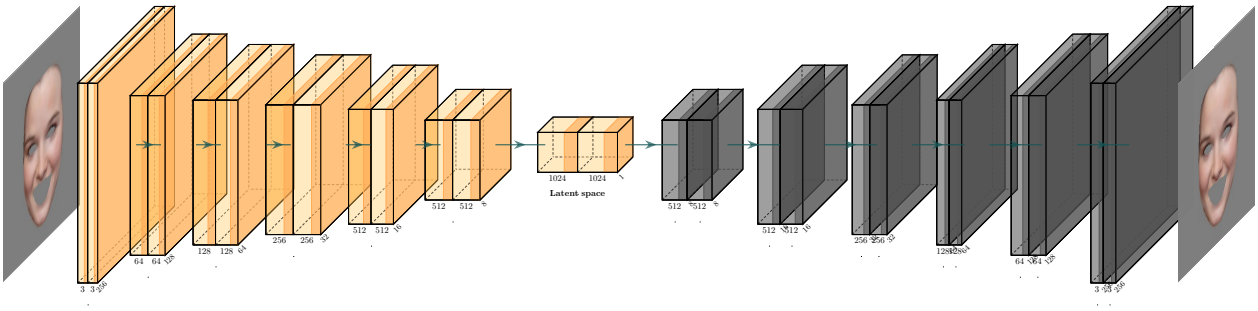

Figure 2.2.: Architecture of the autoencoder E_{local} (3.1.1), it has six CNN layers, a latent space and then another six layers of transpose convolutional networks. The autoencoder is trained on the skin of the source image x^s . The shape of the image tensor is (256,256,3), the latent vector is 1024 long.

a decoder (Figure 2.2). The encoder and decoder contains multiple CNN layers along with pooling layers. These CNN layers are capable of learning low level and high level features from the image data. The encoder is responsible to compress the input data x with an unknown distribution $p(x)$ to a specific latent dimension z with distribution $p(z)$. The latent dimension contains mapped information of encoded input data. In the figure 2.2, the latent dimension contains the compressed information on features of the image. In the second part, the decoder uses the same latent information z to reconstruct the image. During training, the autoencoder uses pixel wise loss function like MSE loss (2.1.6).

$$z = \sigma(W \cdot x + b) \tag{2.1}$$

$$\hat{x} = \hat{\sigma}(\hat{W} \cdot z + \hat{b}) \tag{2.2}$$

The encoder network is represented by the conventional neural network function σ after it has been processed through an activation function, where z is the latent dimension as shown in the equation 2.1. Similarly, the decoder is represented as equation 2.2 with different set of weights($\hat{W}$) and bias($\hat{b}$).

2.1.3. Variational Autoencoder

Variational Autoencoder (VAE) [35] is an unsupervised learning technique that works briefly on the architecture of Autoencoder, but with small variation.VAE is a generative model that uses a probabilistic approach in defining the latent space. It has an additional sampling approach, where instead of mapping a latent dimension to a vector, the latent dimension is mapped to a known distribution $p(z)$ with a specified mean and variance. In this way, each feature of the image is mapped to a certain normal distribution, which is represented by $p(z)$. The mapped latent distribution is further used by the Decoder to generate the new image $p(z)$ based on the distribution information.

The loss function for training the VAE are Kullback Leibler (KL) divergence and reconstruction loss [35]. Both losses are combined and used while training the model, as shown in equation 2.3.

$$E_{q(z|x)} \log p(x|z) - KL(q(z|x)||p(z)) \tag{2.3}$$

where x is input data, z is a latent variable, $p(x)$ probability distribution of the input data, which is unknown. $q(z|x)$ defines the distinction of mapping x to latent variable z. $p(z)$ is the probability distribution of latent variable and $p(x|z)$ is the distribution of generated data given latent variable.

2.1.4. Transpose Convolution

Convolutional Neural Network(CNN) 2.1.1 and Transpose Convolution or fractionally strided convolution along with pooling layers are the building blocks of semantic segmentation network architectures. The loss functions in a deep learning architecture with images which is trying to classify(what) and also group the pixels(where) demand that the final output of the model must be of the same dimensions as the input. CNNs are downsamplers by nature, and upsampling is accomplished by the use of transpose convolution with learnable parameters. The decoders in autoencoders 2.1.2 use transpose convolution to upsample the distribution in the latent vector $'z'$. Like in the case of CNNs, Transpose Convolution also has 4 parameters viz; **F** denotes the field size, **K** denotes the number of filters, **S** denotes the stride, and **P** denotes the amount of padding. For a given input image of size x, size of the output image is given by:

$$Outputsize = [\frac{x + 2 \cdot P - (F - 1) - 1}{S} + 1] \tag{2.4}$$

2.1.5. Conditional GAN(cGAN)

Generative Adversarial Network

A Generative Adversarial Nets [36] is a generative framework which estimates the distribution of the input and generate output which is indistinguishable from the input. A GAN (Figure 2.3 consists of two models, a generator G which constantly tries to learn the model to generate the input and a discriminator model D that estimates the difference between real and generated image. The training strategy of GAN is to make generator play the min-max game with the discriminator. In the Equation 2.5 for the GAN, $\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)]$ is the expectation of loss of the discriminator for real image ($Dloss_{real}$) and $\mathbb{E}_{z\sim p_z(z)} \log [1 - D(G(z))]$ is the expectation of loss of the discriminator for fake or generated images ($Dloss_{fake}$).

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)} \log [1 - D(G(z))] \tag{2.5}$$

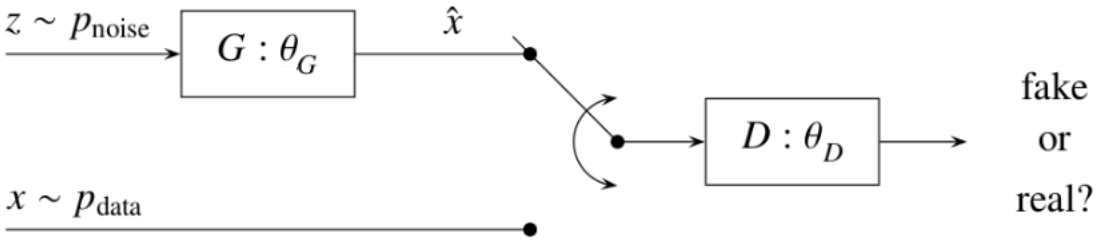

Figure 2.3.: Structure of a Generative Adversarial Nets(GAN), where:
z is the noise sample drawn from a known probability distribution function(PDF).
$P_{noise(z)}$, e.g. N(0, **I**) $\hat{=}$ latent variable.
G: generator with the parameter vector θ_G $\hat{=}$ decoder in VAE.
$\hat{x} = G(z) = G(z; \theta_G)$: generated fake sample.
x: real sample with the unknown PDF $P_{data(x)}$.
D: binary discriminator/classifier with the parameter vector θ_D). Its output $D(x)$ $= D(x; \theta_D) \in [0, 1]$ is the probability of x being real, i.e. class label 1 for real and 0 for fake. Hence, **D** uses a sigmoid activation function in the output layer.

Conditional GAN

The generator and the discriminator of a GAN can be conditionally trained on class labelled data such that we can manipulate the generated output of the GAN, this type of GAN is called conditional GAN [18]. The Figure 2.4 summarizes a CGAN model, both the discriminator and the generator are fed an additional set of values, $'y'$ which is the class label. In the Equation 2.6 for the conditional GAN, both the inputs to the generator and discriminator are conditioned on one additional piece of information $'y'$.

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)} \log [1 - D(G(z|y))] \tag{2.6}$$

2.1.6. Optimization

Loss functions: A loss function is used to measure the performance of a deep neural network. The feedback of the loss function is then used by the optimizer to update the learnable parameters of the network so that it can make better prediction in the subsequent trial. The choice of the objective function mainly depend upon the learning task, here we discuss the loss functions used in this thesis. Denoting N - the number of training samples, (x_i, y_i) - the i-th training sample, x - the input, y - the output, $L(\theta)$ - the loss, and θ - the parameter.

Mean squared error (MSE)

MSE approximates the average of the squares of the difference between ground truth $y(n)$ and predicted outputs $\hat{y}(n)$. MSE is usually employed in reconstruction tasks in generative models like GANs.

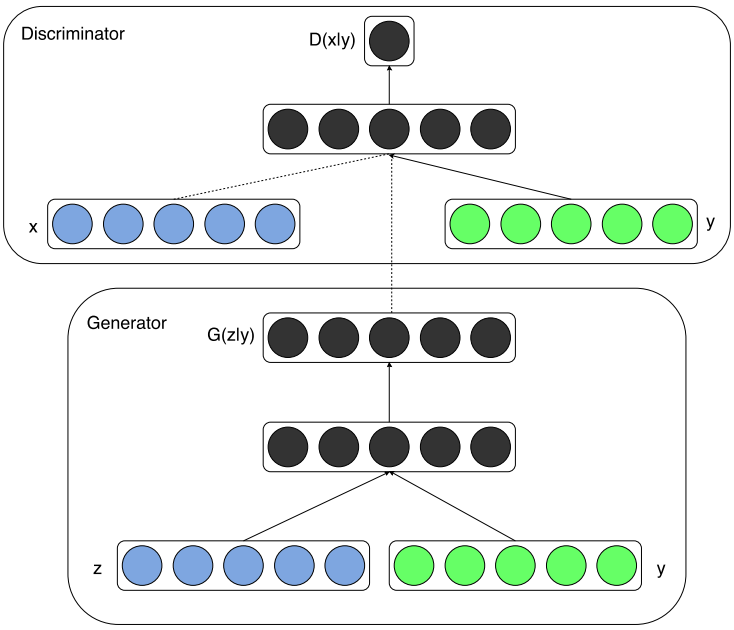

Figure 2.4.: Basic structure of a conditional GAN, where:
x: input, y: output and z: noise sample
$D(x|y)$: Discriminator with input x conditioned on class label y
$G(x|y)$: Generator with input x conditioned on class label y

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{2.7}$$

Mean Absolute Error(MAE)

MAE is the arithmetic average of the absolute difference between the actual and predicted values.

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i| \tag{2.8}$$

Cross-entropy loss

Cross-entropy loss minimizes the Kullback-Leibler divergence between the predicted distribution and the correct distribution. It minimizes the negative log-likelihood of the correct class.

$$KL(p||y) = -\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \tag{2.9}$$

where, M - number of classes, log - the natural log, y - binary indicator (0 or 1) if class label c is the correct classification for observation o, p - predicted probability observation o is of class c.

GAN Feature matching loss

GAN Feature matching loss [37] addresses GANs instability by defining a new objective for the generator that prevents it from overtraining on the current discriminator. Instead of directly maximizing the discriminator's output, the new objective requires the generator to generate data that matches the given distribution, with the discriminator only being used to specify the distribution. On an intermediate layer of the discriminator, we specifically train the generator to match the expected value of the features.

$$L(\theta) = \| \mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{z \sim p_z(z)} f(G(z)) \|_2^2 \tag{2.10}$$

where, $f(x)$ denote activations on an intermediate layer of the discriminator, $'\mathbb{E}'$ is the expectation or mean of $f(x)$, and $G(z)$ is the output of the Generator.

Gram matrix loss

Gram matrix loss [38] is the dot product between all the C feature vectors(flattened) from the output of a CNN layer with their transpose, the name of the matrix is **Gram matrix** which gives the information about correlation between all the feature vectors of the CNN at a depth C. The feature vectors of the CNN layers capture only the spatial information of the image, without any style information. Gram matrix loss gives us the information about the style in the image, and MSE loss between the gram matrix of the input and the style image is used to train the network.

$$L(\theta) = \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \tag{2.11}$$

where, G is the gram matrix, A is the Style image, and i, j are the indices for the rows and columns of the matrices.

VGG loss

VGG loss or perceptual loss [20] is an alternative to the pixel-wise loss; it strives for perceptual similarity. The VGG loss is calculated using the ReLU activation layers of a pre-trained 19-layer VGG network. The feature map obtained by the j^{th} convolution (after activation) prior to the i^{th} max pooling layer within the VGG19 network, denoted by, $\theta_{i,j}$ is considered given. The VGG loss is then defined as the euclidean distance between a reconstructed image's feature representations $G_{\theta G}(I^{LR})$ and the reference image I^{HR}.

$$L(\theta)_{VGG/i.j} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \tag{2.12}$$

where, $W_{i,j}$ and $H_{i,j}$ are the width and height of the respective feature maps within the VGG network

Backpropagation:

Backpropagation [39] is a powerful technique for distributing gradients from the output to the inputs. This can be illustrated using chain rule: For example, if $f = (x + y)z = qz$, implies $\frac{\partial f}{\partial x} = z$ and $\frac{\partial f}{\partial z} = q$. Also, we know that $\frac{\partial q}{\partial x} = \frac{\partial q}{\partial y} = 1$, we apply the chain rule and obtain $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} = z$ and $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y} = z$. The inputs (x, y, z) change in relation to the output f. In practice, abstracting $q = x + y$ is known as staged backpropagation. This enables us to deconstruct the complex f into simpler functions with local gradient rules. Without explicitly deriving the partial gradients of the function f, these local gradients are chained together using the chain rule to obtain the global gradient.

Optimization Algorithm:

An optimizer makes use of the loss values and updates the model parameters during backpropogation, this is called learning and the rate at which the model learns is called the learning rate. The optimizer used in this research work is **adam optimizer** [40]. Adam optimizer is a first-order gradient-based optimizer algorithm of stochastic objective functions, based on adaptive estimates of lower-order moments. Adam optimizer is ideally suited to applications of computer vision due to its simplicity, computational efficiency, invariance to diagonal rescaling of the gradients and perform better for large datasets. Adam optimizer (Equation 2.13) makes use of an exponentially decaying average of past squared gradients $\hat{v}_t$ and an exponentially decaying average of past gradients $\hat{m}_t$. The learning rate for the update rule, η can either be fixed or change during training.

$$\theta_{n+1} = \theta_n - \frac{\alpha}{\sqrt{\hat{v}_n + \epsilon}}\hat{m}_n \tag{2.13}$$

where, ϵ is a very small value for numerical stability (10^{-8} in our project) and $'n'$ is the current iteration number during the training.

2.1.7. Normalization

Before sending the pixel values of an image to the model, all the values must be normalized. Normalization is a data standardization method and used in data preprocessing. It is crucial to normalize the data before training to improve learning pace of the network and avoid any failures of network during training. In this project, batch normalization and layer normalization are used.

Batch Normalization

Batch Normalization [41] is a special technique of data normalization where data is normalized between the neural networks just before giving it to the activation function, this helps in keeping the values fed to the activation function within its range. Batch normalization is applied on mini batches such that all the values are standardized using mean

and standard deviation of the neuron.

$$z^N = \frac{z - m_z}{s_z} \tag{2.14}$$

where, z is the sample data, m_z the mean of the neurons' output and s_z is the standard deviation of the neurons' output.

Instance Normalization

Unlike batch normalization where we require running averages of the summed input, in instance normalization [42] the normalization statistics are directly estimated from the summed inputs to the neurons inside a hidden layer, hence normalization does not create any new dependencies across training examples(equation 2.15).

$$z^N_{i,k} = \frac{z_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \cdot \gamma + \beta \tag{2.15}$$

where, μ_i and σ_i are the mean and the variance respectively for each minibatch, ϵ is a very small value for numerical stability, γ and β are learnable affine transform parameters.

2.1.8. Activation Function

Activation functions are critical for adding nonlinearity to the linear Neural Network(NN) models; they determine whether a neuron should be activated based on its weights and bias. Because the gradients are supplied along with the error to update the weights and biases, activation functions enable the back-propagation. The activation function applies a non-linear transformation to the input, allowing it to learn and perform more complex tasks.

Following specific properties of the activation functions are considered advantageous.

1. Nonlinear: To add nonlinearity to the model.
2. Smooth and monotonic: To mirror input output relation and for convex optimization.
3. Mathematically differentiable: This aids the process of backpropagation

The activation function largely influence the performance of the NN, thus carefully considered while designing NN model. In our implementation, we used four activation functions. Denoting x - input to the activation function.

Sigmoid function

Sigmoid function (Figure 2.5) is a non-linear function where the output values are very steep and continuous in the range (0,1). In this project, sigmoid activations are implemented for multiscale discriminator.

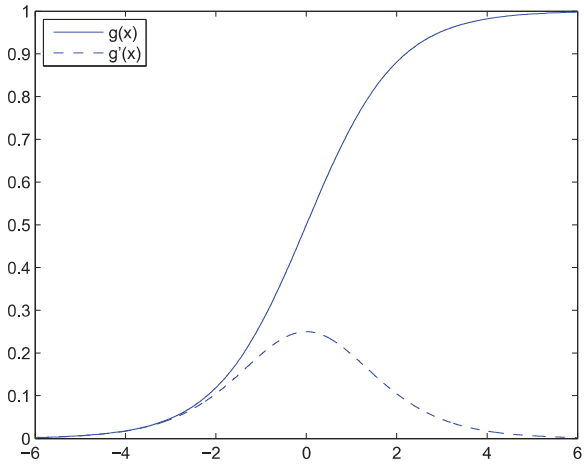

Figure 2.5.: The graphical representation of the Sigmoid function and its derivative

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.16}$$

Hyperbolic tangent (*Tanh*)

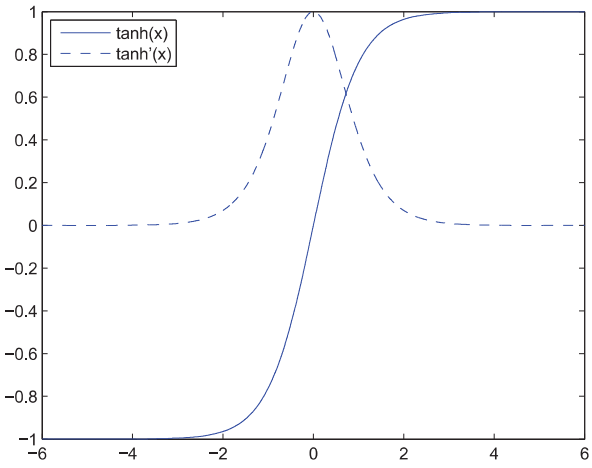

Figure 2.6.: The graphical representation of the Hyperbolic Tangent function and its derivative

Hyperbolic tangent (Figure 2.6) is a mathematically shifted version of the sigmoid function and widely used activation function due to its faster convergence property and is comparatively less computationally expensive. The range of values for *tanh* s from −1 to +1. In this project, hyperbolic tangent activations are implemented in CNN layers.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.17}$$

Rectified linear units (*ReLU*)

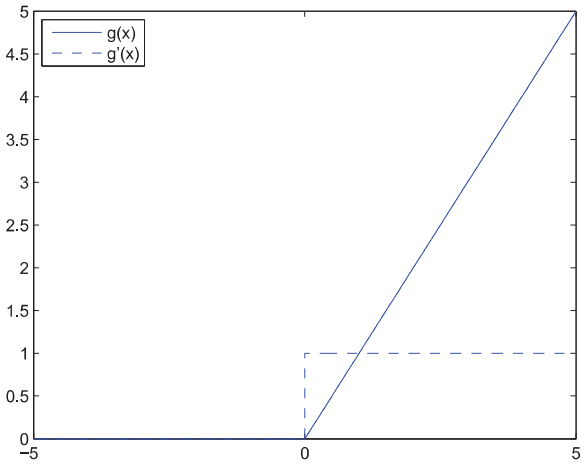

Figure 2.7.: The graphical representation of the RelU function and its derivative

Rectified linear units or piecewise linear function (Figure 2.7) is another type of nonlinear activation function, which depicts a diode functionality in which the output follows the input when the value is positive and returns zero when the value is negative. ReLU is primarily used because it has the property of making the system sparse by forcing negative values to zero.

$$ReLU(x) = \Phi_i(x) = max(x, 0) \tag{2.18}$$

Softmax

Softmax is a type of activation function that is commonly used in the neural network's output layer in multi-class classification tasks, it squeezes the output for each class between 0 and 1, hence an ideal choice for binary classification task.

$$\Phi_i(x) = \frac{e^{xi}}{\sum_{i=1}^{K} e^{xi}} \in (0, 1) \quad for\ i = 1, 2, \ldots, K\ classes \tag{2.19}$$

$$\sum_{i=1}^{K} \Phi_i(a) \tag{2.20}$$

This function maps the logits or numbers into respective probabilities of the possible outcome. This function thus returns the vector representing the possibility of class outcome in probabilities.

2.2. Semantic Segmentation

Semantic segmentation is the task of clustering parts of images together, which belong to the same object class [43]. Semantic segmentation is an important field of application in computer vision; it is utilized in the automobile industry, automation, health care, and most other areas where a machine must classify the parts of the image according to the objects in it. Based on the amount of information encoded, there are different types of semantic segmentation.

1. Grayscale/colored segmentation encodes what object is present in the image and where is it.
2. Depth encoded segmentation is used in robotics and autonomous driving where along with image channels an additional channel is used to encode information about depth.
3. 3D segmentation is used in X-ray and CT scan images, where our interest is in encoding the volume of the object using voxels [44].

In this research work, grayscale facial masks of the image is used as shape guide for the generator to generate images.

2.2.1. U-Net

The U-Net [45] is a U-shaped symmetric structure of CNN layers which is very good at semantic segmentation of images. The input image is passed through a contracting path (encoder) which is a series of 2D convolutional layers, this project consists of six 2D-CNN layers with kernel size of 4x4, at the end of the encoder, there is a sequential layer. The output of the sequential layer is passed through an expansive path (decoder), which consists of transpose convolutional layers with the parameters identical to the CNN layers of the encoder. In this project, the U-Net (Figure 2.8) is used as an image parsing network for generating the semantic segmentation of the output image from the generator G_b as described in the section 3.1.3. The output of the parsing network and the input label are used to compute cross entropy loss, which helps the generator G_b in generating better images. In the paper [1], the author argues that using a face parsing network improves the average pixel accuracy of the image, and it was found to be true during the training of the framework.

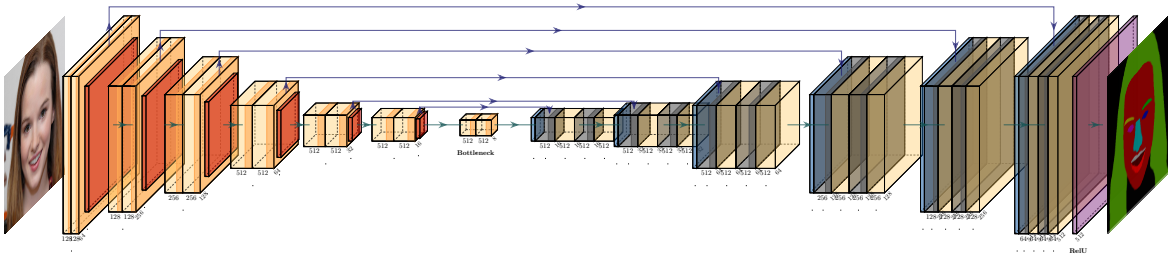

Figure 2.8.: Structure of the Unet model, the input to the UNet model is (1,256,256) mask image, it is passed through a series of CNN layers. The bottleneck layer tensor has dimensions of (512,8,8). The output of the Unet which is of the same shape as the input mask image, is passed through a RelU activation layer as shown in the figure above.

3. Mask-Guided Portrait Editing Framework

This project is based on the work by Gu et al.[1]. The framework, as shown in the Figure 3.1 consists of three major components, namely *local embedding sub − network*, *mask guided generative sub − network* and *background fusing sub − network*. The framework requires four inputs, a source image x^s, the mask of source image m^s, a target image x^t, and the mask of target image m^t. The source and target mask can either be provided or generated using a face parsing network. The target mask m^t is an eleven channel tensor where each channel encodes one facial feature, and it can be manually edited. It is possible to retrieve the appearance of each facial component, such as "left eye" "right eye", "mouth", "skin", and "hair" from the source picture x^s using the source mask m^s. The background of an image can be extracted from the target picture x^t using the target mask m^t. The framework combines the appearance of each component from the source image(x^s) and the target mask(m^t) to produce the foreground face, which is then fused with the background image from x^t to get the final result $G(x^s, m^s, x^t, m^t)$, where the letter G denotes the general generative framework.

3.1. Framework Architecture

3.1.1. Local Embedding Sub-Network

A face mask provides good geometric constraint for facial features, but when training an image using an autoencoder 2.1.2 there is no robust mechanism to extract a specific facial feature recursively. In order to have individual facial component level controllability, a local embedding network was used by the authors of the paper, Gu et al.[1]. As can be seen in the figure 3.1 the source image x^s is segmented into five individual facial components namely $x_i^s, i \in \{0, 1, 2, 3, 4\}$, i.e., "left eye", "right eye", "mouth", "skin", and "hair" using the source mask m^s. The tensor for the facial components "right eye","left eye", and "mouth" are of different shapes, in order to verify that the desired features are within the cropped block, we need to find the center location $\{c_i\}_{i=0,1,2}$ of each component image, more on this will be discussed later in the section 3.2.1.

The facial masks are then used to train a network consisting of five Encoders and ten decoders. For each of the facial features, auto-encoders networks $\{E^i_{local}, D^i_{local}\}, i \in \{0, 1, 2, 3, 4\}$, are used to learn the feature embeddings. The other set of five decoders, $\{D^i_{sub-network}\}, i \in \{0, 1, 2, 3, 4\}$, are attached to E^i_{local} and feed generator G_m. The

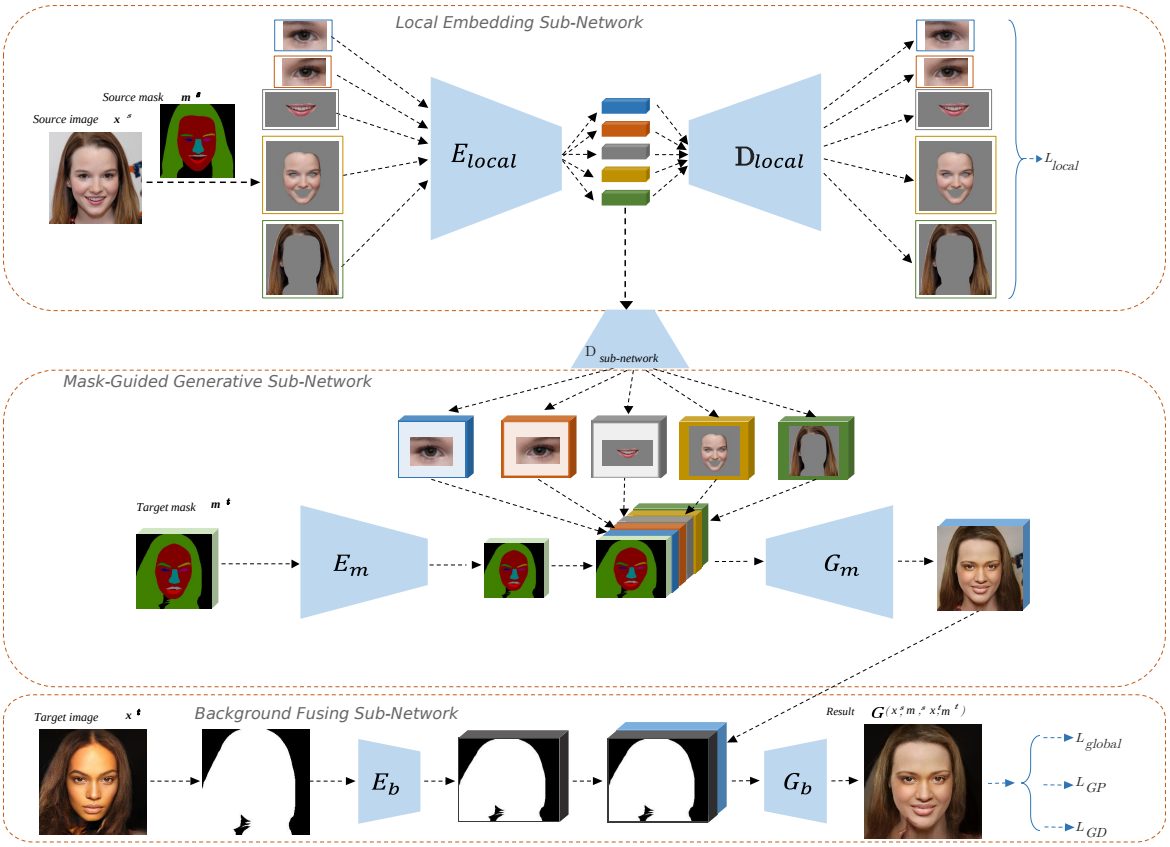

Figure 3.1.: The project architecture for GAN based Mask Guided Portrait Editing framework ([1]). It consists of three parts: *local embedding sub − network*, *mask guided generative sub − network*, *and background fusing sub − network*. Local *embedding sub − network* learns, the feature embedding of the local components of the source image. *Mask guided sub − network* combines the learned component feature embeddings and mask to generate the foreground face image. *Background fusing sub − network* generates the final result from the foreground face and the background. The loss functions are drawn with the blue dashed lines.

decoder D^i_{local} uses a separate local reconstruction loss during training. Adding a local reconstruction loss increases the ability of the model to learn local details of each instance.

3.1.2. Mask-Guided Generative Sub-Network

To use the target mask m^t as a guide for mask equivariant face creation, the five component feature tensors from $D^i_{sub-network}, i \in \{0, 1, 2, 3, 4\}$ and eleven mask feature tensors from $E^i_m, i \in \{0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11\}$ are used as shown in the Figure 3.1. The five component features, which are the outputs of $D_{sub-network}$, have the same number of channels but different shape. At start, the center position $\{c_i\}$ i=1...5 of each component is extracted from the target mask x^t. Then five tensors are created that are all filled with 0, i.e., $\{f_i\}$

$i \in \{1, 2, 3, 4, 5\}$. Every tensor has the same height, width, and channel number as that of the mask feature tensors which is the output of E_m. Then, using the target mask, we replicate each of the five learned component feature tensors to an all-zero tensor f_i centered at c_i (e.g., mouth-to-mouth, eye-to-eye etc.). Following that, we concatenate all the local embedding tensors from $D_{sub-network}$ and mask feature tensor from the encoder E_m to create a fused feature tensor as shown in the Figure 3.1.

3.1.3. Background Fusing Sub-Network

The output of the generator G_m now has the face structure of the target mask m^t and facial features of the source image x^s. To combine it with the background of the target image, another set of generative network (Figure 3.1) is used by the authors [1]. If a straightforward approach of cropping and pasting the output foreground image over the background of the target image is performed, it causes noticeable boundary artifacts in the resulting image.
The solution to this is to use another pair of encoder-decoder network where the background of the target image is encoded by the encoder E_b, then its output is concatenated with the output of G_m, the resulting tensor is then fed to the final generator G_b. This method of using a generator greatly reduces the artifacts' problem, as the segmentation mask for the hair part will not be perfect always. The final result of the network $G(x^s, m^s, x^t, m^t)$ is used to optimize the network using a set of loss functions.

3.2. Loss Functions

The complete project architecture, Figure 3.1 consists of seven encoders, ten decoders, two generators and a set of two multiscale discriminators. Training of a complicated model like this requires use of various learning techniques and loss functions. In this section, I will explain these methods and the corresponding loss functions.

3.2.1. Local Reconstruction Loss

As discussed in the section 3.1.1 the input image mask m^s is used to crop the input image x^s into individual facial features. To begin with, five tensors are created of different sizes pertaining to each facial feature, as given in the Figure 3.2, and they are all filled with 0, i.e., $f_i = 0, ß \in \{1, 2, 3, 4, 5\}$. The center position of eyes and mouth component is extracted from the source image mask, m^s and it is verified if the source image mask has distinct facial features by checking the position of the center pixel of the facial feature of interest in the entire image. Now, using the center position of each facial component, a rectangular patch is cropped away from the source image x^s and pasted onto the previously created tensor f_i for each of the facial components i ∈ {1,2,3} i.e., 'left eye', 'right Eye' and 'mouth'. For the facial feature of hair and skin, the above step is not required as their size is equal to that of

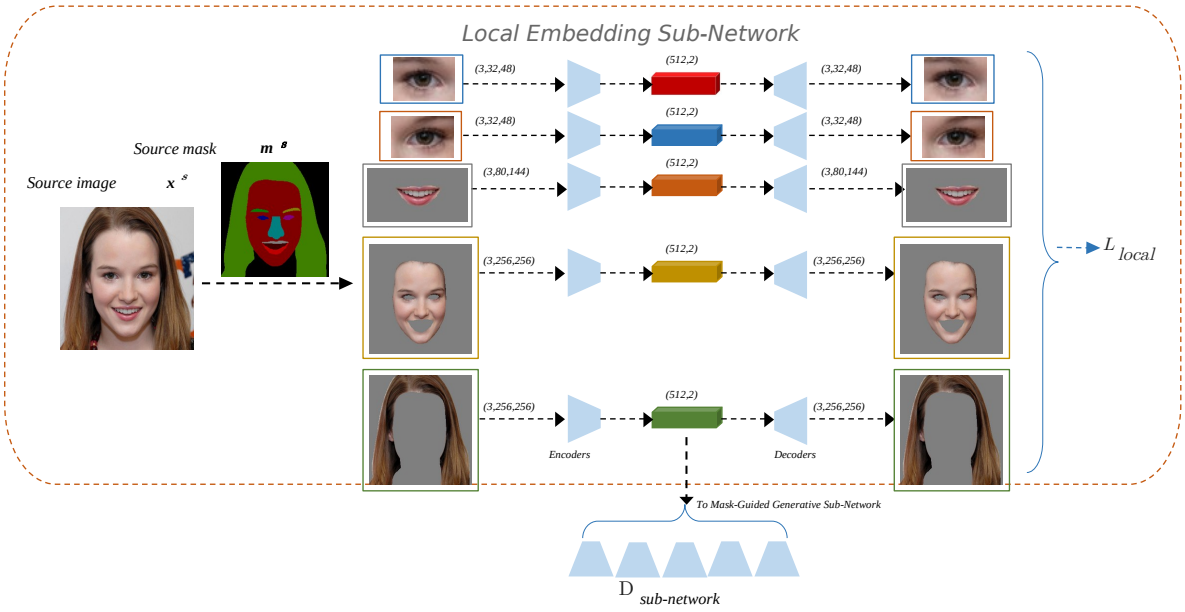

Figure 3.2.: Local Embedding Sub-Network, source mask m^s is used to extract facial components from the source image x^s and it is fed to a network of five Autoencoders(AE) 2.1.2. The loss function used for the training of the AE is a pixel wise MSE loss L_{local}. $D_{sub-network}$ is another set of five decoders which take the latent space or the output from the encoders as its inputs

the input image, and it is directly extracted from the image using the mask.

All five facial instances are fed to five different encoders E^i_{local}, i ∈ {1,2,3,4,5} of varying input size and network structure based upon the size of the tensor f_i, i={1,2,3,4,5}. Each layer of the encoder is made up of CNN layers, batch normalization and ReLU activation function as shown in the figure 2.2. The output of each of the encoders used as the inputs for another set of five decoders D^i_{local}, i ∈ {1,2,3,4,5} which reconstructs each facial features. To learn the feature embedding of each instance, a pixel wise MSE loss is used between the input examples and the rebuilt instances [1].

$$L_{local} = \frac{1}{2} \parallel x^s_i - D^i_{local}(\mathbb{E}^i_{local}(x^s_i)) \parallel^2_2 \tag{3.1}$$

where, $x^s_i, i \in \{1,2,3,4,5\}$ represents “left eye”, “right eye”, “mouth”, “skin, and “hair”, **Encoders** and **Decoders** are the local decoders and encoders respectively.

3.2.2. Global Reconstruction Loss

Mask-Guided Generative Sub-Network

The $D_{sub-network}$ consists of five decoders of different shape based on the feature to be encoded. The input size to each of the five decoders is the same $(512, 2)$ and the output size

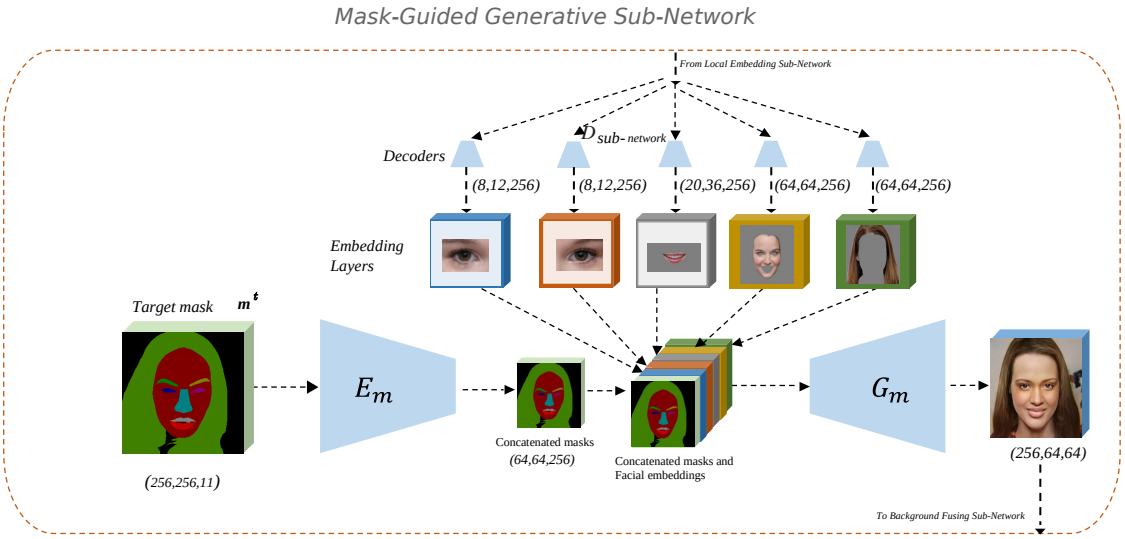

Figure 3.3.: Figure explaining the structure of the Mask-Guided Generative Sub-Network, E_m is used to encode the target mask m^t and the output of the $D_{sub-network}$ is the local embedding of each of the facial features for the source image x^s. All the tensors are concatenated and fed to the generator G_m which generates the foreground image as shown in the figure above.

is different as seen in the Figure 3.3. The target mask m^t is a tensor with eleven channels, and the shape of the tensor is $(256, 256, 11)$. Each channel of the tensor encodes one feature of the image in the form of the labels, as given in the Table 3.1. The labels for "upper lip", "lower lip", and "mouth" are concatenated to make the mouth mask, similarly "nose", "right eyebrow", "left eyebrow", and "skin" are the part of the skin mask. The target mask m^t is encoded by the encoder E_m, the shape of the output tensor is (256,64,64). As explained in the Section 3.1.2 the five output tensors of the $D_{sub-network}$ are transformed such that their shape is equal to the shape of the output tensor of the encoder E_m. All the six tensors are now concatenated and fed to the generator, G_m which learns to generate images such that it retains the style of the target image x^t but the facial features are changed to that of the source image x^s.

Background Fusing Sub-Network

As discussed in the section 3.1.3 the output of the generator G_m and the output of the encoder E_b are concatenated and fed to the final generator G_b. The target image x^t is fed to the encoder E_b (Figure 3.4) and the resulting tensor is of the shape (256,256,64). As shown in the subsection 3.2.2 and the table 3.1, the zeroth channel of the target image

Labels	Feature
label **0**	Background Mask
label **1**	Skin Mask
label **2**	Left Eyebrow Mask
label **3**	Right Eyebrow Mask
label **4**	Left Eye Mask
label **5**	Right Eye Mask
label **6**	Nose Mask
label **7**	Upper Lip Mask
label **8**	Mouth Mask
label **9**	Lower Lip Mask
label **10**	Hair Mask

Table 3.1.: The table above shows the unique label for individual facial features, the values in the label column are the pixel values corresponding to that feature in the second column as explained in the section 4.2.

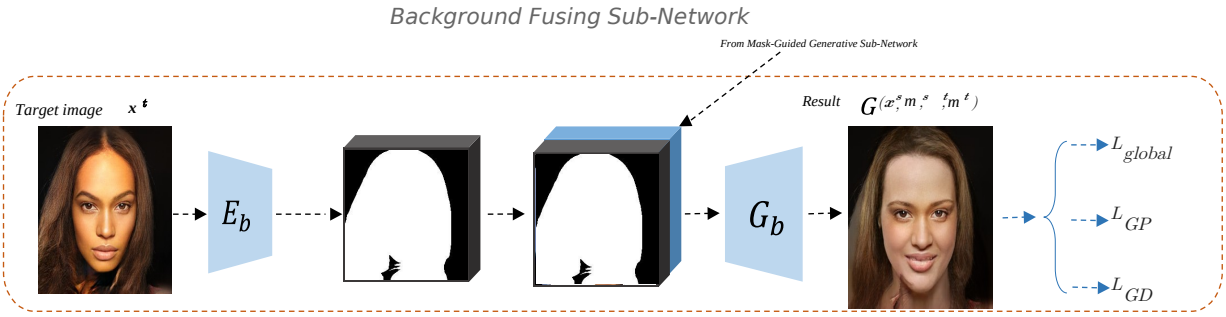

Figure 3.4.: Diagram of the Background Fusing Sub-Network. the encoder E_b extracts the background features from the target image x^t which is then concatenated with the foreground image tensor coming from the generator G_m and fed to the generator G_b to generate the final output image tensor of the size (256,256,3).

mask m^t encodes background information of the image. This channel is used to extract the background image from the target image x^t, and it is used to remove the face data from the output tensor of the encoder E_b. This tensor is then concatenated with the output tensor from G_m and fed to the final generator G_b. The output of the generator G_b is an RGB image tensor, which is used to compute the losses in generators G_b and G_m and the encoders $D_{sub-network}$, E_m, and E_b.

$$L_{global} = \frac{1}{2} \parallel G(x^s, m^s, x^t, m^t) - x^s) \parallel_2^2 \tag{3.2}$$

The global reconstruction loss L_{global} is a pixel wise MSE loss. During training, when the input source image X^s is the same as the input target image x^t, the resultant image generated by the generator G_b which is a function of all four inputs $G(x^s, m^s, x^t, m^t)$ (section 3), must be the same as x^s.

3.2.3. Adversarial Loss

The output of the background fusing sub-network framework is used by the loss functions to train the complete model. Since the output of the generator G_b is a (256,256,3) image, it is not enough to use a simple discriminator for image synthesis due to the instability of the GANs during training and its sensitivity to hyperparameters [46]. To overcome this issue, the authors of the paper [1] used a multiscale discriminator similar to work in the paper [11]. Two identical set of discriminators d_i, $i \in \{1, 2\}$ operating at different image scales, are used for learning. The first discriminator is fed with the full scale images, whereas for the other discriminator, both the generated and real images are scaled down by a factor of two using an average pooling layer. The loss function for the discriminators d_i, $i \in \{1, 2\}$ is given as:

$$L_{D_i} = \mathbb{E}_{x^t \sim p_r}[\log D_i(x^t, m^t)] - \mathbb{E}_{x^t \sim p_r}[\log (1 - D_i(G(x^s, m^s, x^t, m^t), m^t))] \tag{3.3}$$

and, the loss function for the framework G is given by:

$$L_{sigmoid} = -\mathbb{E}_{x^t \sim p_r}[\log (D_i(G(x^s, m^s, x^t, m^t), m^t))] \tag{3.4}$$

Pairwise Feature Matching

To overcome the unstable gradient problem of the GAN (Equation 3.4) and generate realistic facial images, the features of the discriminator network D are matched pairwise for real and generated images as expleined in the equation 3.5.

$$L_{FM} = \frac{1}{2} \parallel f_{D_i}(G(x^s, m^s, x^t, m^t), m^t) - f_{D_i}(x^t, m^t) \parallel_2^2 \tag{3.5}$$

where, L_{FM} stands for pairwise feature matching loss, and it is defined as the Euclidean distance between feature representations, $f_{D_i}(x^t, m^t)$ is the features on an intermediate layer of the discriminator, and f_{D_i} is the last output layer of the network D_i.

Combining Equation 3.4 and 3.5 gives us the overall loss from the discriminator networks to the framework G as given in the equation 3.6:

$$L_{GD} = L_{sigmoid} + \lambda_{FM} \cdot L_{FM} \tag{3.6}$$

where, λ_{FM} is a parameter to control the weightage of L_{FM}.

3.2.4. Face Parsing Loss

Each pair of the input images are supplied with their facial masks, these facial masks are in turn used to train a face parsing network P_F whose network structure is derived from the Unet [45]. The generated sample from the framework are passed through the face parsing

network and the output of the network is matched with the mask of the input images, this helps the whole framework in generating mask equivariant images. The loss function, L_P as shown in the Equation 3.7 [1] used to train the face parsing network, P_F is a pixel-wise cross entropy loss.

$$L_P = -\mathbb{E}_{x\sim p_r}[\sum_{i,j} \log P(p_{i,j}|P_F(x)_{i,j})] \tag{3.7}$$

where, (i, j) is the location of the pixel in the image.

The fully trained face parsing loss P_F is then utilized to stimulate the produced samples of the framework to have the same mask as the target mask, therefore the generative network has the loss function as given in the Equation 3.8:

$$L_{GP} = -\mathbb{E}_{x\sim p_r}[\sum_{i,j} \log P(m^t_{i,j}|P_F(G(x^s, m^s, x^t, m^t))_{i,j})] \tag{3.8}$$

where, $m^t_{i,j}$ is the input mask label for x^t located at (i, j) and $P_F(G(x^s, m^s, x^t, m^t))_{i,j}$ is the predicted pixels of the face parsing network at the location (i, j).

3.2.5. Overall Loss Functions

The total loss of the framework is the sum of the losses in the Equations 3.1, 3.2, 3.6, 3.8.

$$L_G = \lambda_{local} \cdot L_{local} + \lambda_{global} \cdot L_{global} + \lambda_{GD} \cdot L_{GD} + \lambda_{GP} \cdot L_{GP}, \tag{3.9}$$

where, λ_{local}, λ_{global}, λ_{GD}, and λ_{GP} are the parameters to balance the importance of different losses. In this experiment, the values of the constants are $\{10, 1, 1, 1\}$ respectively [1].

3.3. Training Strategy

During training, the input masks m^s and m^t can be provided or extracted from the parsing results of the source and target images, x^s and x^t respectively. There are two scenarios during training: 1) x^s and x^t are the same, which is referred to as paired data; 2) x^s and x^t are different, which is referred to as unpaired data, one phase of the training for the paired data training and the other for the unpaired data [16]. The training loss functions for these two values, however, should be different. For paired data, all the losses in the Equation 3.9 is used, but for unpaired data, λ_{global} is set to zero in Equation 3.9 and λ_{FM} is set to zero in Equation 3.5. The new overall loss function for unpaired case is given inas Equation 3.10

$$L_G = \lambda_{local} \cdot L_{local} + L_{sigmoid} + \lambda_{GP} \cdot L_{GP} \tag{3.10}$$

4. Implementation

4.1. Dataset

The dataset used in this research work is CelebAMask-HQ [47] dataset. It is a large-scale face image dataset with 30,000 high-resolution face images of size (1024,1024,3) 4.1. Each of the image has up to 19 segmentation black and white masks of the facial attributes corresponding to the celeb image with the size of each mask as (512,512,1). The classes of facial components are skin, nose, eyes, eyebrows, ears, mouth, lip, hair, hat, eyeglass, earring, necklace, neck, and cloth as shown in the Figure 4.2

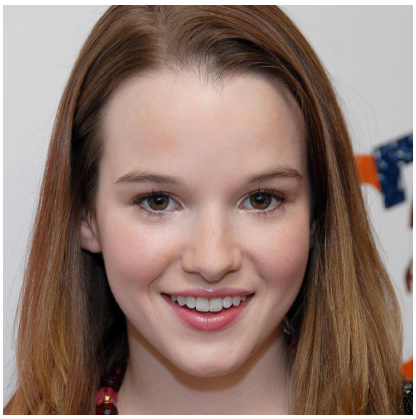

Figure 4.1.: A sample image from the CelebAMask-HQ dataset, the shape of the image is (1024,1024,3

4.2. Image Preprocessing

As discussed in section the section 3.1.1, the framework is fed with an image and the corresponding facial mask. This facial mask is then used to extract individual facial features from the image (3.1.1 and 3.2.1). All the masks corresponding to each image in the CelebAMask-HQ dataset (Figure 4.2) are concatenated into one single mask image such that each facial feature is encoded with a unique pixel value as per the table 3.1. Figure 4.3 shows the pixel encoding corresponding to individual facial features for the image 4.1.

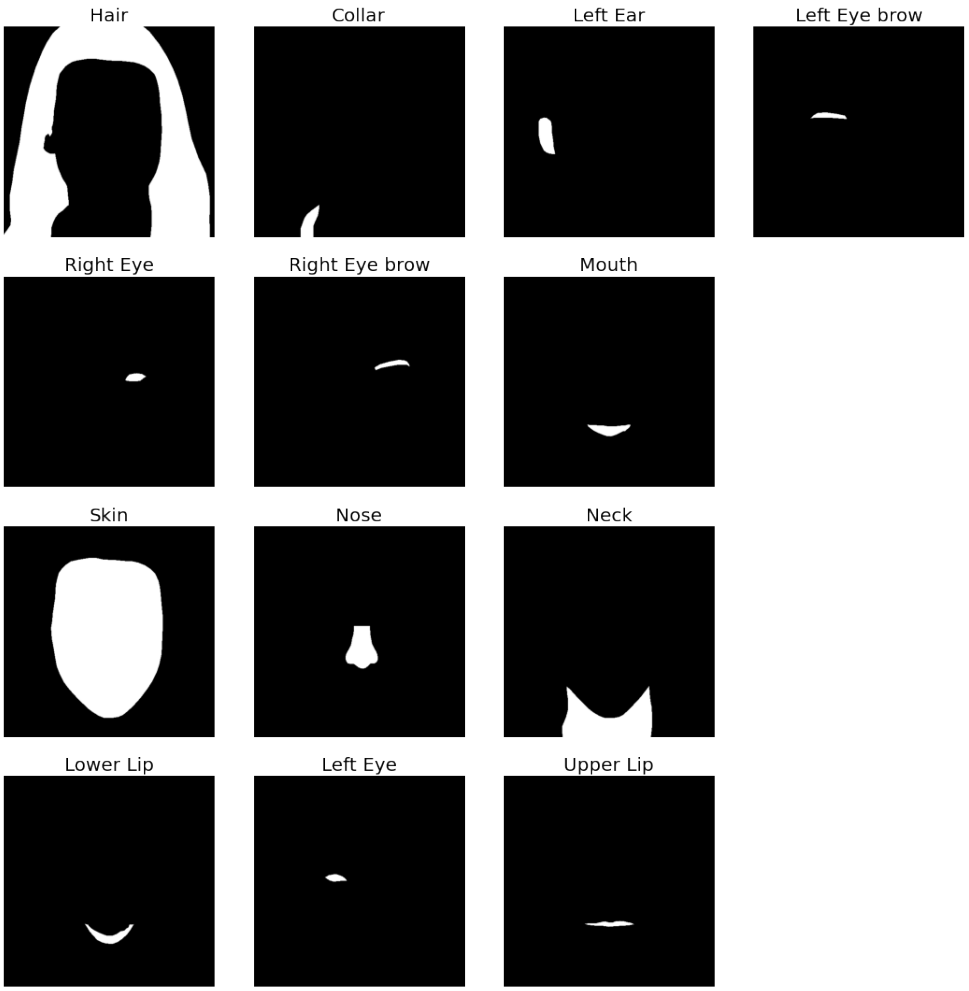

Figure 4.2.: Grayscale masks for the sample image in the Figure 4.1). Each of the mask image is highlighting one feature on a black background, the size of each image is (512x512)

4.2.1. Aligned Dataset - Feature Extraction from the Image using its Mask

A dataset is created with different applications in mind, and it is tried to make it as generic as possible. Yet no dataset is free of outlier sample images, which can hamper the learning of the Generative Adversarial Networks(GAN) [36] which are inherently very unstable.

To manage faulty images, the mask of each individual image is used to find out the position of the facial features like "left eye", "right eye", "mouth", and "nose", and it is verified if the image is legitimate and can be used for the training of the GAN or not. For instance, the mouth to be cropped from the source image x^s (section 3.1.1) is of the size (3,80,144), during preprocessing as discussed in the section 3.2.1 the center position of the feature(mouth in this case) is calculated by drawing a square around the feature of the size (3,80,144) and computing its center position. But there can be an image where the mouth is in the left or the right most corner, and it is not possible to draw a square of the size

Figure 4.3.: Pixel encoding for the image shown in the Figure 4.1 as per the table 3.1. It must be noted that the values for the pixel encoding are very low, and the picture cannot be distinguished easily, hence 70 units of brightness is added to each pixel of the mask image to improve the visibility of the image in this figure only.

(3,80,144) around the center of the mouth without crossing the boundary of the image. In this case, the center position of the square is adjusted such that the square does not go beyond the edges of the image. The same set of steps are carried for both the eye masks whose feature dimensions are (3,32,48). The hair and skin masks do not need to be prechecked as their size is equal to that of the image (3,256,256).

4.2.2. Append region

As discussed in the section 3.2.1, each of the facial feature is trained by a separate set of encoder-decoder network, and it is then merged using a generator. For the facial feature like eyes and mouth, the border positions where the mask region of the feature ends, and the skin region begins, there may exist blind spots because of the flawed masks annotations which can produce artifacts in the final generated image. To overcome this issue, the masks of the mouth and both the eyes are expanded by 10% relative to their size given in the original mask. This new mask is later used to calculate the cross entropy loss between the new mask and the mask generated by the parsing network for the fake images generated by the generator G_b (Figure 3.4).

4.2.3. Custom Data loader and Input Encoder

As explained in the section 3 the framework need four inputs, x^s, m^s, x^t, m^t, a custom data loader is used to generate the dataset with pairs of images and facial masks respectively. The custom data loader with the help of the aligned dataset 4.2.1 generates the training dataset. During training, a special encoding function uses the training data and converts the pixel encoding (Figure 4.3) into a one-hot encoded target vector which is used to extract facial features from the training image as shown in the section 3.2.1 and 3.1.1. The same special encoding function also creates the set of paired and unpaired data (section 3.3 for the

training of the framework.

4.3. Training

A custom data loader provides the images in batches for the training of the framework. As discussed in the section 3.3 the trainable parameters are chosen based on the images in each batch i.e., paired data or unpaired data. The hyperparameters like number of down sampling layers in the generator, number of Resnet blocks, normalization type, data type of the weights, learning rate at the beginning, number of epochs, image size, Decay rate for learning, dropout, GPU IDs to use and others must be provided during the beginning of the training.The framework was trained for 35 hours using three of the 32GB Tesla V100 GPU manufactured by nvidia for 55 epochs.

4.3.1. Training Parameters

- Batch size : **12**.
- Image size : **(256,256,3)**.
- Learning rate: Starting learning rate is **0.0002**, and it decreases to zero by the end of last epoch, changing after every 10 epochs.
- Data type of the weights : **float32**.
- Training dataset size: **28,000** images with their masks.
- Number of residual blocks in the global generator network: **9**
- Number of residual blocks in the local enhancer network: **3**
- Number of downsampling layers in the encoders: **4**
- Number of downsampling layers in netG: **3**
- Normalization: Batch Normalization
- Optimizer: **Adam Optimizer**

4.3.2. Training Loss

Figure 4.4 is the plot of the local reconstruction loss for the Local Embedding Sub-Network as per in the equation 3.1 during training.

Figure 4.5 compares the perceptual similarity in between the generated images and the reference image as per the equation 2.12.

Figure 4.6 is the plot of the discriminator losses for discriminator D1 and D2 as per the equations 3.6, 3.4, and 3.5.

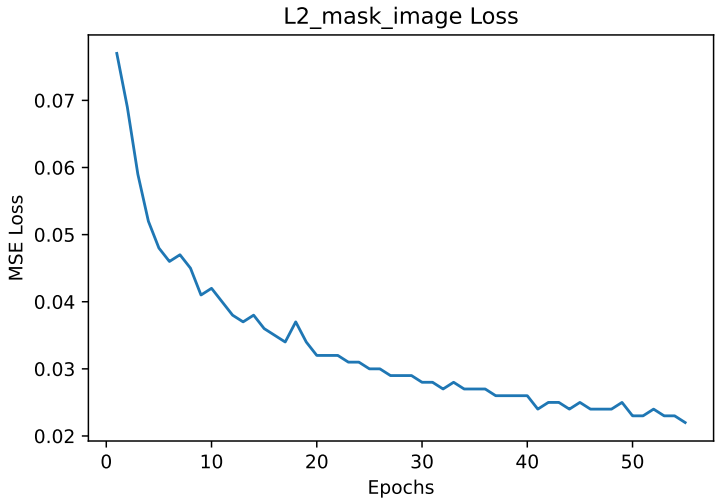

Figure 4.4.: Mean squared error(MSE) loss, also called as L2 loss between the generated facial embeddings for each of the encoders in Local Embedding Sub-Network(3.1.1) and the cropped ground truth images

4.3.3. Fréchet Inception Distance

Fréchet Inception Distance score (FID) is a metric to evaluate the performance of GAN network by measuring the distance between feature vectors calculated for real and generated images. FID score is said to be better than Inception score [48] because FID takes into account the mean and variance of both the distributions (generated and real-world images) and the distance between then is the Frechet distance also known as Wasserstein-2 distance [49]. Unlike the inception score which uses the output of the last layer of the Inception V3 model [50], the FID score method makes use of the second-to-last layer of the same Inception V3 model using the Inception V3 model as a feature extractor with 2048 activations (feature vector) in the output for each image. The 2048 features vectors for the collection of both real and synthetic images are used to compute the FID score as given in the equation 4.1. Table 4.1 shows the FID scores of different methods.

$$d^2((m, C), (m_w, C_w)) = \| m - m_w \|_2^2 + Tr(C + C_w - 2(C \cdot C_w)^{1/2} \tag{4.1}$$

where, $d^2((m, C), (m_w, C_w))$ is the MSE distance between both the distributions, m and m_w are the feature-wise mean of the real and generated images and C and C_w are the covariance matrices and Tr is the trace of the square covariance matrix.

Experiment Method	FID scores
Original Paper [1]	26.54
Changing the Eye mask 4.12	26.04
Multi Target Mask Encoder 4.15	28.48

Table 4.1.: The table above shows the FID scores of different methods

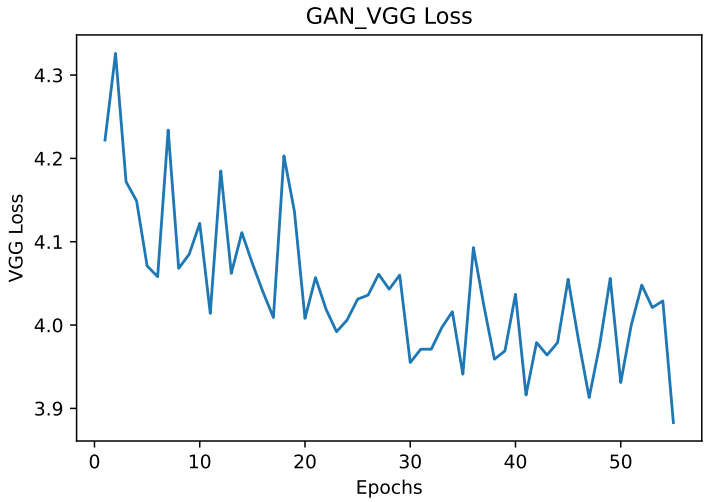

Figure 4.5.: VGG Loss 2.1.6 for the generated images by the final generator G_b during training

4.4. Analysis of the Framework

The main objective of the mask-guided portrait editing framework (3) is to be able to produce diverse high images with control. The first module of the framework, an autoencoder network 3.1.1, learns the global embedding of the source image x^s, then all five global encoding are concatenated along with the mask of the target image m^t and fed to the generator G_m. The generator G_m 3.1.2 with the help of losses L_{global}, L_{GP} and L_{GD} 3.2.5 learns to generate a foreground image such that the generated image uses the style of the global embeddings and the shape of the target mask m^t. Then finally, the background fusing sub-network 3.1.3 crops the background from the target image x^t and directly paste it to the generated foreground face to get the final result.

4.4.1. Mask-to-face Synthesis

This section demonstrates the ability of the GAN to produce realistic and diverse images from a fixed target mask and randomly choosing source images. The Local embeddings from each of the source images are concatenated with a fixed target mask and, as explained in the section 4.4, the generator G_m extracts the style for the generated image from the Local embeddings and the components are derived from the target mask. It can be seen in the Figure 4.7 that all the generated images have similar facial features, specially the skin with different style. In the last row of the Figure 4.7 the image of the lady is changed such that the eyes are replaced with glasses it is accomplished using the same method as the information about glass is stored in the local embeddings. It must also be noted that the background remains the same in the last row, demonstrating the ability of the framework to control the generated image.

Previous methods were also able to do the mask-to-face synthesis like BicyleGAN [12] which used image-to-image translation model to generate output images. In the paper

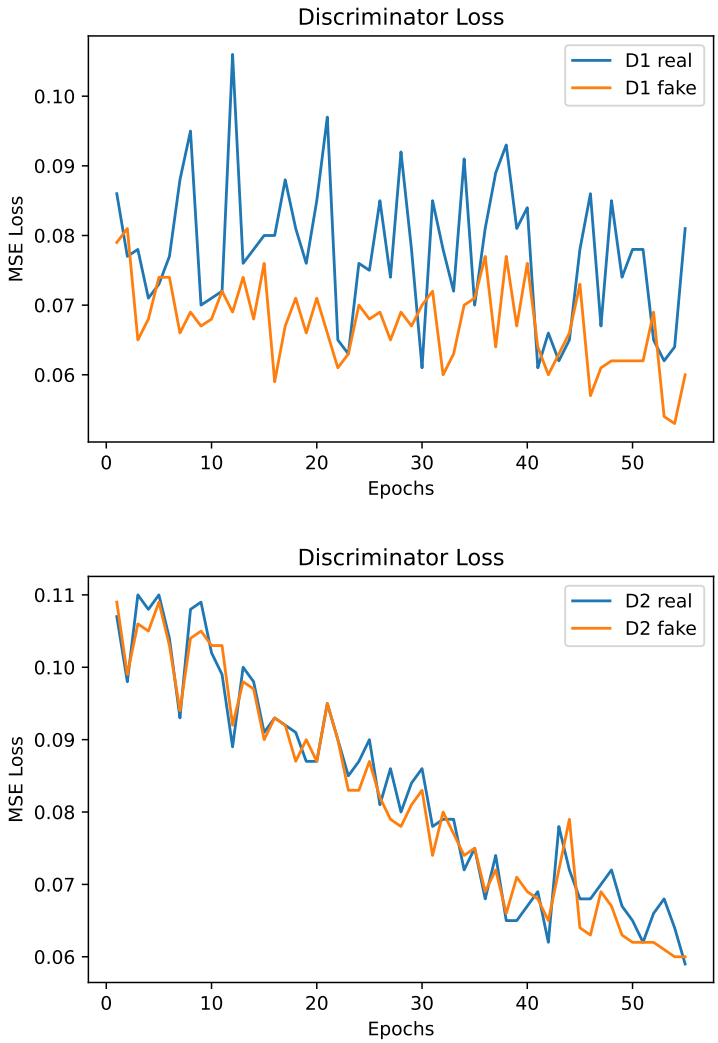

Figure 4.6.: Comparison of the loss of the Discriminator D1 and D2 during training for real and fake generated images, the MSE loss in both the figures are within 0.11 and decreasing

Figure 4.7.: The framework synthesizes realistic, diverse and mask equivariant faces from one target mask. In the above figure, the first and scond columns of each row are the target image and target mask respectively. All the remaining images are the generated images by the framework with random source faces.

pix2pixHD [11] the authors were able to generate high resolution image-to-image translation. It is observed that both BicycleGAN and pix2pixHD show poor results in terms of diversity of the generated images, skin color, and style [1]. This framework achieves better results owing to the fact that it makes use of multiple encoders in the form of local embedding sub-network and mask guided generative subnetwork.

There is a great demand for changing the style of the image by keeping the facial features fixed. As an extension of the application of mask to face synthesis, the style of one image can be transferred to another without copying the facial features.

4.4.2. Face Component Editing

Another application of the framework is face component editing, by leveraging the fact that the generated image of the framework depends on the mask of the target image for facial components we can control the facial components of the generated image. This can be achieved by two ways:

1. By editing individual components of the target mask from other's faces to explicitly replace the corresponding component of the generated image.
2. By swapping the target mask with the mask of another image.

The figure 4.8 and 4.9 shows the generated images by changing individual facial components from the mask of the target image. The framework allows the users to manipulate the facial expression of the generated images and change the emotions conveyed by them.

Figure 4.8.: The framework is capable of generating images with one or two components of the target image changed. In this figure, the images in the first row are input and the second row images are the generated images. The generated image of the first column is a consequence of using a target mask with more hair. In the second column, the eyes and skin of the target mask are edited. In the final column, the generated image has bangs which are not present in the source image.

Changing the target mask lets the user manipulate facial features of the generated image at an individual component level.

Figure 4.10 shows the sample of generated images when swapping the complete target mask, it must be noted that the new target mask chosen must resemble the mask of the image in the pose and size to get better results. It is observed that using a broader skin mask as the target mask than that of the original source image makes the output image wider, the skin also helps in generating images with no hair on the forehead. Besides this, using a target mask with smaller eye size forces the generated image to have smaller eyes, thus the model is capable of editing more than one facial feature.

4.4.3. Style transfer

There is a great demand for changing the style of the image by keeping the facial features fixed. Similar to the section 4.4.1, here the mask of the target image was fixed, but the same target image was used as the source image such that the local feature embedding of one facial feature was copied from a different source image. Figure 4.11 shows the output of this technique, the generated image keeps the overall facial structure intact and only change the style of the facial feature. This technique is capable of changing the color of the lip, the hair color without changing the hairstyle, grow beard on a clean-shaven face and change the color of the eyes.It can be inferred from the output images, that the model is able to successfully transfer the style from one image to another.

Figure 4.9.: The figures demonstrate the ability of the framework to manipulate expressions of the generated image. In the first column, the expression of the source image is changed from smile to giggle and in the second column it is changed from grin to stern face by changing the mouth component of the target image mask. The generated image of the last column shows the ability of the framework to remove bangs from the face.

4.5. Experiments

The framework of this project makes use of a number of different loss functions and components, this opens the opportunity to experiment by changing the components and the loss functions of the framework. In this section, we will discuss the experiments performed on the framework and their results.

4.5.1. Changing the Eye mask

In the original work 3.1, the author Gu et al. [1], used a window of the size (3,32,48) to crop the eye from the original source image. Since the eye of the image is smaller than this window, it also contains parts of the skin which is fed to the eye encoder E_{local}. This is problematic during generating style images to transfer skin from one image to another using the framework. As explained in the section 4.4.3, for style transfer of the skin, we have to encode both the images using the local encoder E_{local} and the transfer the skin of one image to another. But since the encoding of the eye also has part of the skin, the generator gets two inputs for the skin style, one from the local skin encoding of the source image and another from the encoding of the eyes. The generated image gets new skin style, but the area surrounding the eyes remain the same as the original image. To overcome this issue, the style of the eyes and skin must be transferred together, as can be seen in the figure 4.13.

Figure 4.10.: The framework is capable of generating new images by replacing the target facial mask of the source image completely, with good amount of control. In the above figure, first row images are the input and second row images are the images generated by the framework. In the first column, the generated image has broader face, smaller eyes and receded hairline when compared to the source image. The output of the second column has reduced smile, slightly wider face and the in the last column, the generated image has broader face, smaller eyebrow and larger forehead. All the generated images maintain the original style and only the shape of the facial features are changed.

To overcome this issue, the original architecture is edited such that only the eye section is cropped from the source image and trained in the local encoder E_{local}.The figure 4.12 shows the new architecture of the local embedding sub-network. The section of the skin around the eye is completely cropped away from the source image with the help of the segmentation mask x^s. The local encoders for the eyes now learn only the style features of the eye, whereas only the skin encoder encoders the information about the skin. The new method provides better control over the generated images, as shown in the figure 4.14.

4.5.2. Multi Target Mask Encoder

As discussed in the section 3.2.2 the shape of the target mask m^t is (11,256,256) wherein each channel is encoding one facial feature (3.1), and it is provided as input to the encoder E_m. In this experiment, the facial features of the image m^t are split into individual components and then fed to a set of five encoders "$E_{Left\ Eye}$", "$E_{RightEye}$", "E_{Mouth}", "E_{Skin}", and "E_{Hair}". The outputs of these five encoders are then concatenated with the facial embeddings coming from local embedding sub-network 3.1.1 as shown in the Figure 4.15 and fed into the generator G_m. The framework is then trained with all the model parameters same as the original implementation.

Outputs

The model was successfully trained on CelebAMask-HQ [47] dataset and was able to produce output images with fair controllability. A FID score of 28.48 was achieved using the model which is not better than the original model, the L_{local} loss was never below 0.029 which is higher than the original implementation. Figure 4.17 shows a sample output of the model.

4.5.3. VAE based local Embedding Sub-network

The local embedding sub-network (section 3.1.1) uses a set of five autoencoders for learning the style parameters for each of the facial embeddings. In this experiment, the autoencoders are replaced with variational autoencoders 2.1.3 in a bid to make the local embedding sub-network, a generative model. Figure 4.18 shows the training loss for the first epoch for the framework with VAE instead of autoencoders, the learning of the framework stops before the end of the first epoch. Since the latent space of the VAE is also connected to the generator G_m through $D_{dub-network}$ (section 3.1), the loss functions-KL embed loss and L2 loss is not enough.

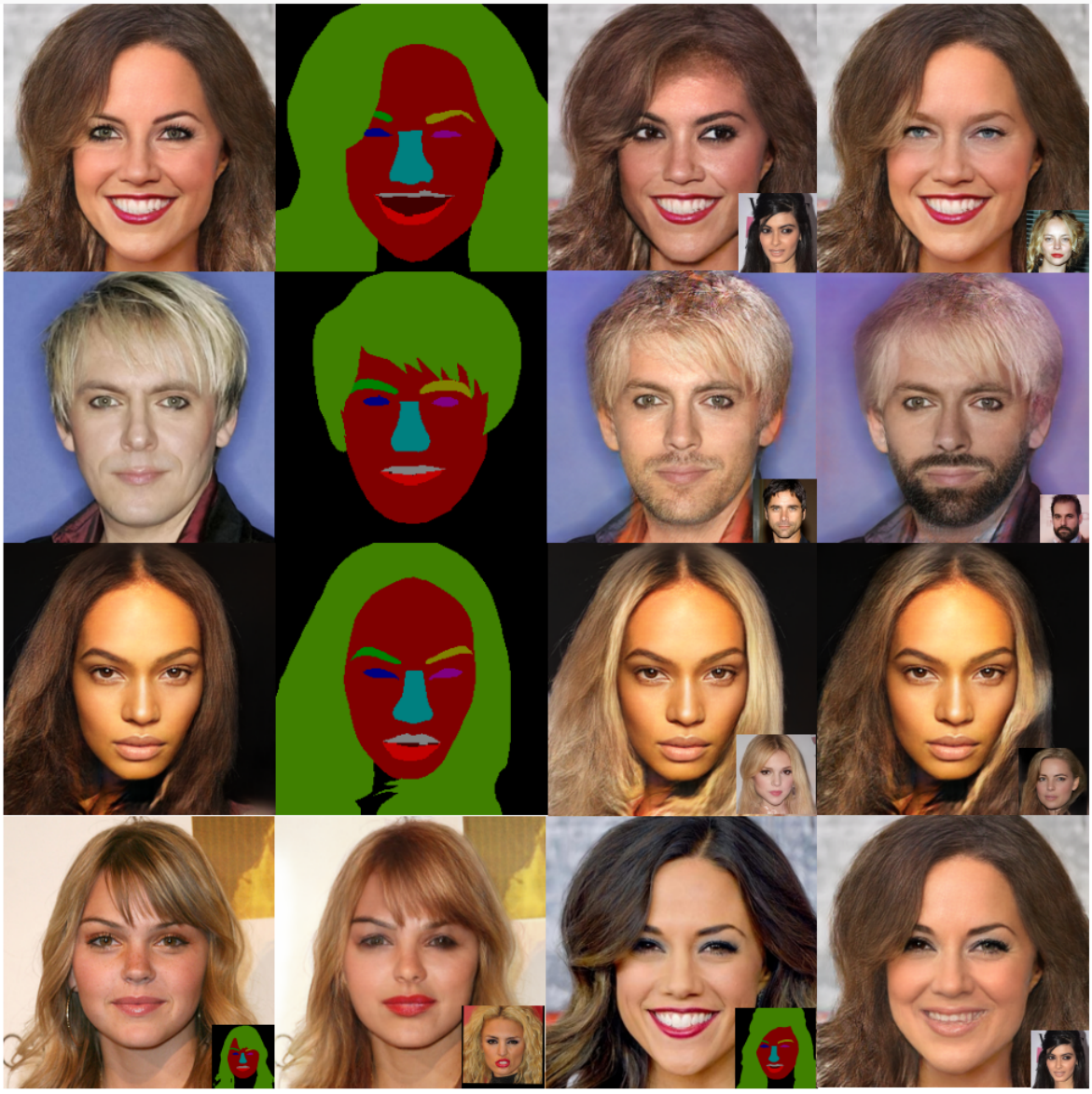

Figure 4.11.: The above figure demonstrates the ability of the model to transfer the style of an image without copying the facial features. In the first row, the eye color of source image changed, in the second row, the model is able to generate beard from the input image which do not have any beard. The third row demonstrates the ability of the image to change hair color and in the final row the model is able to change the color of the lips of the source images, here the first and the third images are the source image, and the second and fourth images are the outputs respectively.

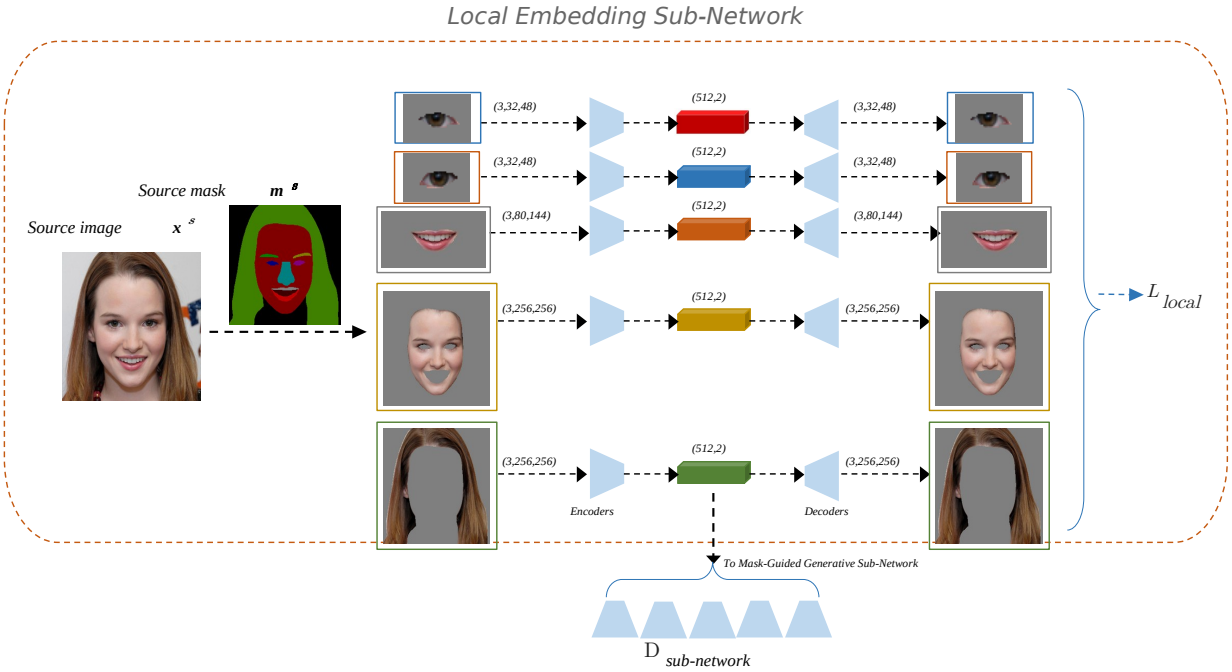

Figure 4.12.: The cropped image for the eyes from the source image x^s is free from the skin encoding, it is fed to the autoencoder which learns the style encoding of the source image.

Figure 4.13.: Output of the original framework for the application of style transfer. The images in the first column are the images to be edited, and the second column consists of the images whose style is to be copied onto the first image. The images of the third column is the output image of the framework having the skin style of the second column images with the style of eyes unchanged. Since only the eyes of the source image is used by the local encoder during training, the generated images are free of any sudden skin tone change.

Figure 4.14.: Output of the new framework for the application of style transfer. The images in the first column are the images to be edited, and the second column consists of the images whose style is to be copied onto the first image. The third column contains the output of the experiment where eye masks are changed 4.5.1 and the final column consists of the output from the original method.

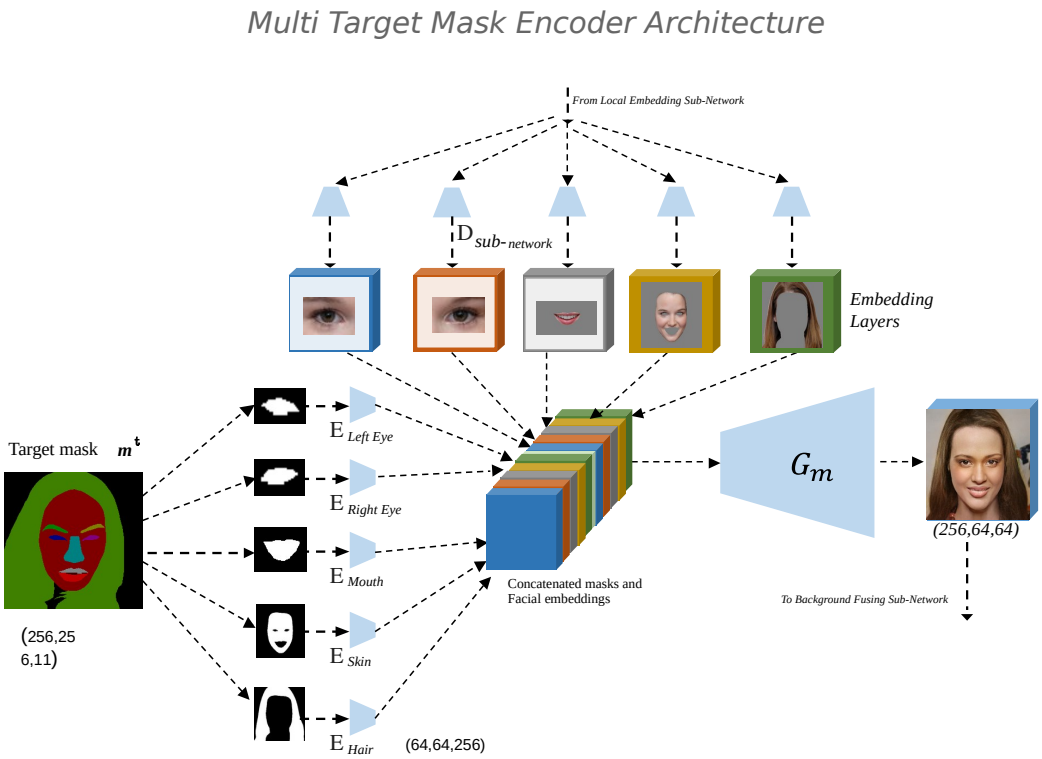

Figure 4.15.: Multi Target Mask Encoder: "$E_{Left\ Eye}$", "$E_{RightEye}$", "E_{Mouth}", "E_{Skin}", and "E_{Hair}" are the five local encoders. The shape of the output of each of the local encoders is (64,64,256)

Figure 4.16.: Real input image to the multi target mask encoder model

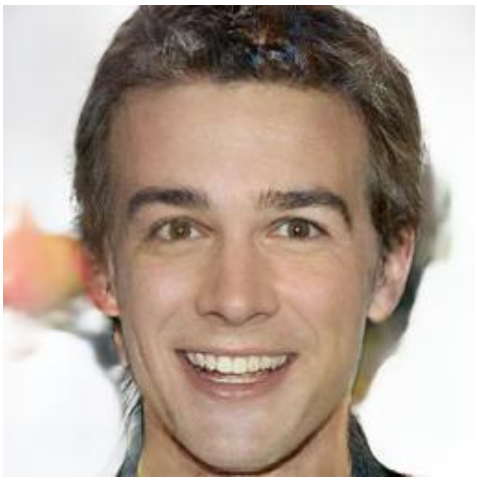

Figure 4.17.: Reconstructed image by the multi target mask encoder model

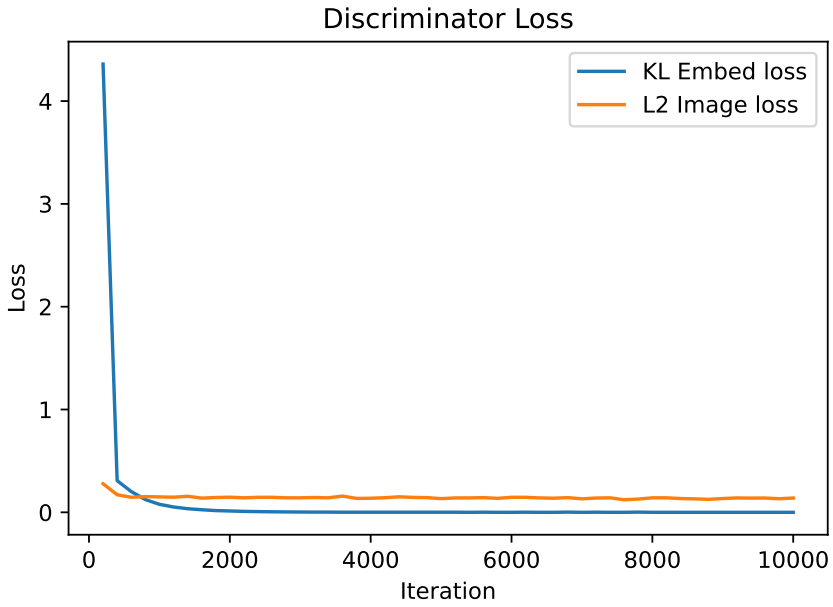

Figure 4.18.: The KL embed loss goes to zero within 2000 iterations of the first epoch and L2 mask image loss stays constant at 0.13 and learning of the model stops.

5. Conclusion

From the series of experiments conducted on the mask-guided conditional GAN based framework, it is demonstrated that the model can synthesize a diverse variety of high quality facial images with very good controllability using the segmentation mask of the face. The framework is capable of explicitly editing facial components, style transfer from one image to another, generate new faces using a fixed target mask.

A series of image processing techniques were implemented to extract facial features from the image for the training of the framework. The feature extraction from the source image was accomplished with the helped of the aligned dataset module, by identifying the position of the features of all the images during training as explained in the section 4.2.1. The training images and their masks (x^s, x^t, m^s, m^t) are grouped as paired and unpaired data with the help of a custom data loader and a special encoding function as explained in the section 4.2.3. The FID score 4.1 is used to evaluate the quality of the generated images as per the training parameters explained in the section 4.3.1.

It is observed that the local embedding sub-network (section 3.1.1) is able to learn the local embeddings and the style of the source image with the help of a set of five autoencoders and the loss L_{local} (equation 3.1). The Mask-Guided Generative Sub-Network is able to generate high quality images by extracting the style information from the local embeddings and the boundaries of the facial features from the mask of the target image, thus giving complete control over the style and features of the generated image. The background fusing sub-network is successfully able to stitch the generated image over the background of the target image.

The section 4.4.1, demonstrates the ability of the framework to generate images by keeping the target mask fixed, the framework is able to generate realistic faces by keeping the components of the face fixed. The resulting images have different skin colors and hair colors by keeping the face structure and hairstyle the same. The framework is also capable of adding glasses to the face, change eyebrow, and eyeball color. In the section 4.4.2 it is shown that the framework is able to edit individual components of the face like adding or removing bangs, increasing hair, changing hair line by changing the skin encoding, change the expression of the face from smile to giggle or stern. The section 4.4.3 elaborates that the framework is capable of changing the style of the image by keeping the facial component same, like growing beard on the face, changing lip, eye, and hair colors.

A series of experiments were conducted by altering the architecture of the framework, which was helpful in understanding the capabilities and limitation of the framework and various deep learning techniques. Increasing the learning rate helped the framework in learning facial embeddings faster, but the style transferability of the output image was not good. Changing the normalization from batch to instance normalization for the encoder network and adding dropout to the CNN layers was not very helpful in generating good images, in fact the quality of the generated images degraded. In the section 4.5.2 the target mask encoder

was split into individual masks and the model was trained with four additional encoders, the output images with this method were as good as the original images if not more which can be inferred from the table 4.1, but there was no significant gain in the controllability of the framework.

In the section 4.5.1 the window used to crop the eye from the source image is altered such that only the eye is used for the training of the encoder and not the skin surrounding it. Training the model with this change helped in achieving better control over the generated images when compared to original technique, and the generated images were marginally better than the images generated with original technique (Table 4.1). In another experiment where VAE was used instead of autoencoders in local embedding sun-network (section 4.5.3), the learning of the framework stopped before the end of the first epoch due to the interlinked architecture of the framework where the local embedding network is connected with the generator G_m and G_b. A different loss function other than KL divergence and L2 pixel wise loss is needed to increase the performance of the VAE based network, which is outside the scope of this research work.

A. Acronyms

Adam Optimizer: Adaptive moment estimation optimizer

AE : Autoencoder

cGAN : Conditional generative adversarial network

CNN : Convolutional neural network

E_b : Encoder to encode background of the target image

E_{local} : Local encoder for embedded facial features

E_m : Target mask Encoder

GAN : Generative adversarial network

G_b : Generator to concatenate foreground image with the target background

G_{local} : Local decoder for embedded facial features

G_m : Generator to concatenate target mask with facial components

MAE: Mean absolute error

MSE : Mean squared error

ResNets : Residual networks

VGG : Visual geometry group

List of Figures

List of Tables

Bibliography

[1] S. Gu, J. Bao, H. Yang, D. Chen, F. Wen and L. Yuan, "Mask-guided portrait editing with conditional gans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3436–3445.

[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*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.

[3] G. Antipov, M. Baccouche and J. Dugelay, "Face aging with conditional generative adversarial networks," *CoRR*, vol. abs/1702.01983, 2017.

[4] H. Yang, D. Huang, Y. Wang and A. K. Jain, "Learning face age progression: A pyramid architecture of gans," *CoRR*, vol. abs/1711.10352, 2017.

[5] L. Tran, X. Yin and X. Liu, "Disentangled representation learning gan for pose-invariant face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1415–1424.

[6] Z. He, W. Zuo, M. Kan, S. Shan and X. Chen, "Attgan: Facial attribute editing by only changing what you want," *IEEE transactions on image processing*, vol. 28, no. 11, pp. 5464–5478, 2019.

[7] Q. Deng, J. Cao, Y. Liu, Z. Chai, Q. Li and Z. Sun, "Reference-guided face component editing," *arXiv preprint arXiv:2006.02051*, 2020.

[8] Y. Shih, S. Paris, C. Barnes, W. T. Freeman and F. Durand, "Style transfer for headshot portraits," 2014.

[9] J. Fišer, O. Jamriška, D. Simons, E. Shechtman, J. Lu, P. Asente, M. Lukáč and D. Sỳkora, "Example-based synthesis of stylized facial animations," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[10] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[11] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8798–8807.

[12] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang and E. Shechtman, "Toward multimodal image-to-image translation," *Advances in neural information processing systems*, vol. 30, 2017.

[13] J. Adler and S. Lunz, "Banach wasserstein gan," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[14] G.-J. Qi, "Loss-sensitive generative adversarial networks on lipschitz densities," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1118–1140, 2020.

[15] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[16] J. Bao, D. Chen, F. Wen, H. Li and G. Hua, "Towards open-set identity preserving face synthesis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6713–6722.

[17] X. Huang, M.-Y. Liu, S. Belongie and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 172–189.

[18] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[19] J. Liao, Y. Yao, L. Yuan, G. Hua and S. B. Kang, "Visual attribute transfer through deep image analogy," *arXiv preprint arXiv:1705.01088*, 2017.

[20] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[21] S. Iizuka, E. Simo-Serra and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–14, 2017.

[22] T. Portenier, Q. Hu, A. Szabo, S. A. Bigdeli, P. Favaro and M. Zwicker, "Faceshop: Deep sketch-based face image editing," *arXiv preprint arXiv:1804.08972*, 2018.

[23] P. Sangkloy, J. Lu, C. Fang, F. Yu and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5400–5409.

[24] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1511–1520.

[25] S. Gu, C. Chen, J. Liao and L. Yuan, "Arbitrary style transfer with deep feature reshuffle," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8222–8231.

[26] L. A. Gatys, A. S. Ecker and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.

[27] M. He, J. Liao, L. Yuan and P. V. Sander, "Neural color transfer between images," *arXiv preprint arXiv:1710.00756*, vol. 2, 2017.

[28] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur and S. K. Nayar, "Face swapping: automatically replacing faces in photographs," in *ACM SIGGRAPH 2008 papers*, 2008, pp. 1–8.

[29] I. Korshunova, W. Shi, J. Dambre and L. Theis, "Fast face-swap using convolutional neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3677–3685.

[30] P. P. Busto, C. Eisenacher, S. Lefebvre, M. Stamminger *et al.*, "Instant texture synthesis by numbers." in *VMV*, 2010, pp. 81–85.

[31] J. Hays and A. A. Efros, "Scene completion using millions of photographs," *ACM Transactions on Graphics (ToG)*, vol. 26, no. 3, pp. 4–es, 2007.

[32] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn and A. Criminisi, "Photo clip art," *ACM transactions on graphics (TOG)*, vol. 26, no. 3, pp. 3–es, 2007.

[33] X. Qi, Q. Chen, J. Jia and V. Koltun, "Semi-parametric image synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8808–8816.

[34] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *arXiv preprint arXiv:1906.02691*, 2019.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[37] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.

[38] Y. Li, N. Wang, J. Liu and X. Hou, "Demystifying neural style transfer," *arXiv preprint arXiv:1701.01036*, 2017.

[39] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[42] D. Ulyanov, A. Vedaldi and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[43] M. Thoma, "A survey of semantic segmentation," *arXiv preprint arXiv:1602.06541*, 2016.

[44] S. Hu, E. A. Hoffman and J. M. Reinhardt, "Automatic lung segmentation for accurate quantitation of volumetric x-ray ct images," *IEEE transactions on medical imaging*, vol. 20, no. 6, pp. 490–498, 2001.

[45] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[46] A. Karnewar and O. Wang, "Msg-gan: Multi-scale gradients for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7799–7808.

[47] C.-H. Lee, Z. Liu, L. Wu and P. Luo, "Maskgan: Towards diverse and interactive facial image manipulation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[48] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 2018.

[49] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[50] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

Declaration

Herewith, I declare that I have developed and written the enclosed thesis entirely by myself and that I have not used sources or means except those declared.

This thesis has not been submitted to any other authority to achieve an academic grading and has not been published elsewhere.

Stuttgart, 09-05-2022 Prabhakar Kumar Panday