E9 246 **ADVANCED IMAGE PROCESSING**
COURSE PROJECT

K.Kalyan Reddy
21361
M.Tech AI

G.Ramesh Babu
20950
M.Tech AI

Paper 1 : Convolution Sketch Inversion (https://arxiv.org/pdf/1606.03073.pdf)

Paper 2 : Controlling Deep Image Synthesis with Sketch and Color
(https://arxiv.org/pdf/1612.00835.pdf)

Paper 1 : ***Convolutional Sketch Inversion***

The goal here is to develop a deep learning model that can invert a hand-drawn sketch into a realistic image. This is achieved by using a convolutional neural network (CNN) that is trained on a dataset of sketches and their corresponding images. The project is inspired by the work of researchers who have previously used similar techniques to generate images from sketches.

Dataset:

The authors have used over 200k images for training(which comprises 3 datasets). However due to colab constraints we couldn't process that much data. Hence we used the below Dataset.

Dataset used: CUHK Face Sketch (CUFS) database
Training Images : 3600
Testing Images : 236

Preprocessing:

All the images have been resized to 240*240.

Implementation :

1) The entire code for this paper is written from scratch (however some inbuilt sklearn, pytorch libraries were used)
2) For the VGG pretrained model, the features after removing the last fc layer are used.
3) PSNR(Peak Signal to Noise ratio) and SSIM are calculated as mentioned in the paper.
4) PSNR is calculated using the below formula

$$\mathrm{PSNR} = \frac{1}{3}\sum_{k} 10\log_{10}\frac{\max \mathrm{DR}^2}{\frac{1}{m}\sum_{i,j}\left(t_{i,j,k}-y_{i,j,k}\right)^2}$$

5) We have also tried, predicting sketches from images, which surprisingly did pretty well on the same dataset.

Results:

Average PSNR(db) of all test images is given by: 42.721792715590745

Average SSIM of all test images is given by: 0.6459761773821339

Observations:

Some sample Images from dataset:

Deep Network Architecture:

Layer	Type	in_channels	out_channels	ksize	stride	pad	normalization	activation
1	con.	1 or 3	32	9	1	4	BN	ReLU
2	con.	32	64	3	2	1	BN	ReLU
3	con.	64	128	3	2	1	BN	ReLU
4	res.	128/128	128/128	3/3	1/1	1/1	BN/BN	ReLU
5	res.	128/128	128/128	3/3	1/1	1/1	BN/BN	ReLU/+x
6	res.	128/128	128/128	3/3	1/1	1/1	BN/BN	ReLU/+x
7	res.	128/128	128/128	3/3	1/1	1/1	BN/BN	ReLU/+x
8	res.	128/128	128/128	3/3	1/1	1/1	BN/BN	ReLU/+x
9	dec.	128	64	3	2	1	BN	ReLU
10	dec.	64	32	3	2	1	BN	ReLU
11	con.	32	3	9	1	4	BN	tanh

Loss used:

The first component is the standard Euclidean loss for the targets and the predictions, also called the pixel loss.(lp)

The second component is the Euclidean loss for the feature-transformed targets and the feature-transformed predictions, also known as the feature loss. The feature loss is calculated using the outputs of the last fc layer of a 16-layer deep neural network (DNN) (fc layer outputs of the **VGG-16** pretrained model), which is used to transform the targets and the predictions.

$$\ell_f = \frac{1}{n} \sum_{i,j,k} \left(\phi(t)_{i,j,k} - \phi(y)_{i,j,k} \right)^2$$

where n is the total number of features, $\Phi(t)_{i,j,k}$ is a feature of the targets and $\Phi(y)_{i,j,k}$ is a feature of the predictions.

The third component is the total variation loss for the predictions, which measures the smoothness of the output image.

$$\ell_{tv} = \sum_{i,j} \left((y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2 \right)^{0.5}$$

Where y(i,j) is the predicted pixel.

To calculate the overall loss, a weighted combination of these components is used. The authors set the weights for the pixel loss and feature loss to 1, and the weight for the total variation loss to 0.00001.

A weighted combination of these components resulted in the following loss function:

$$\ell = \lambda_p \ell_p + \lambda_f \ell_f + \lambda_{tv} \ell_{tv}$$

where we set $\lambda_p = \lambda_f = 1$ and $\lambda_{tv} = 0.00001$

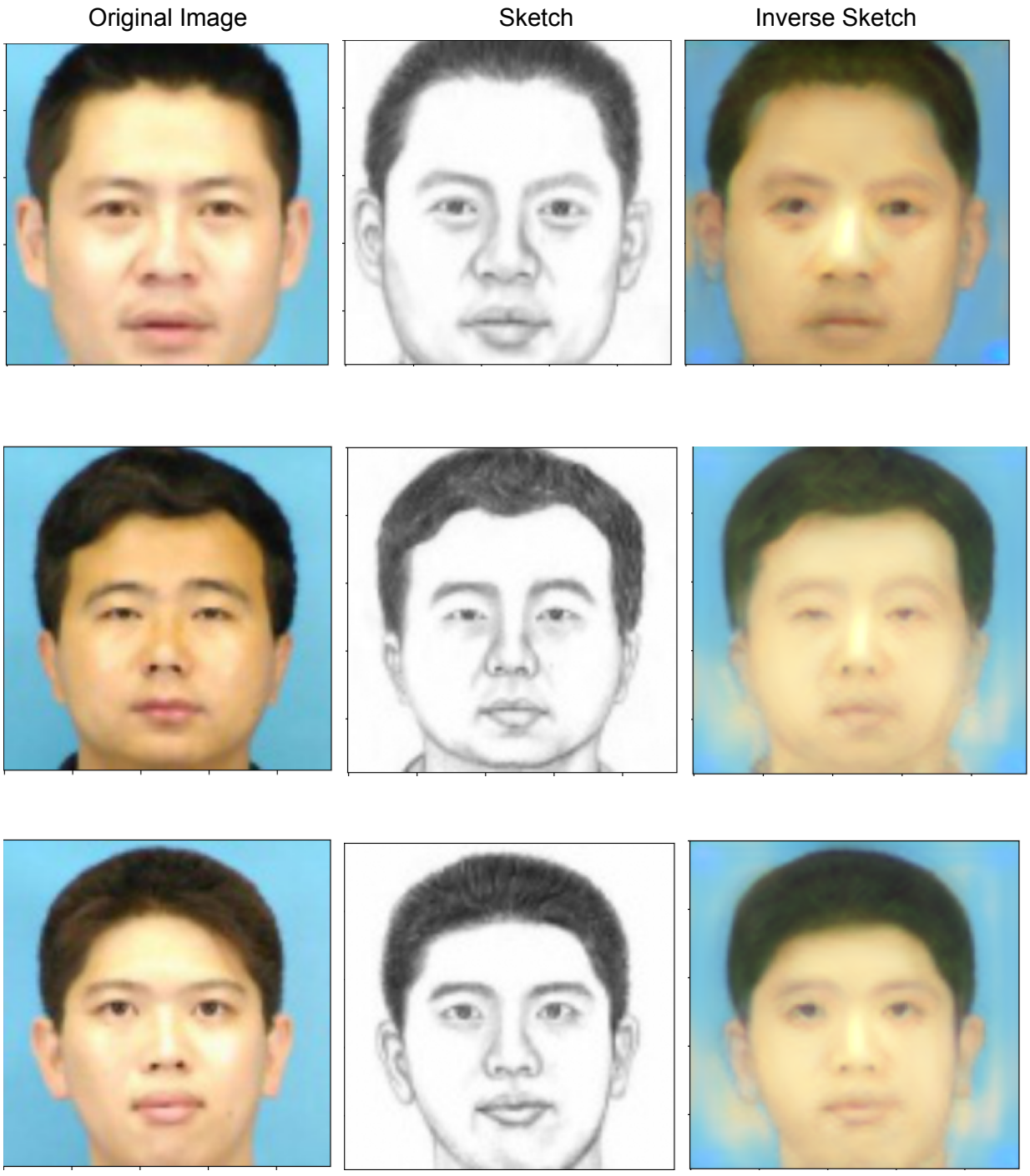
Optimizers, hyper parameters used:

The authors used the Adam optimizer with specific parameters, including a learning rate of 0.001, beta values of 0.9 and 0.999, epsilon of 10^-8, and a mini-batch size of 4. The model was trained for 200,000 iterations while minimizing a loss function consisting of three components.

We used the same Hyperparameters and optimizers, But we trained it for 50 epochs(3hrs) and the output images were decent considering less data.

Results obtained:

Some visualized outputs:



Results:

Average PSNR(db) of all test images is given by: 42.721792715590745

Average SSIM of all test images is given by: 0.6459761773821339

The reported average PSNR of 42.7 dB indicates that, on average, the difference between the original and the reconstructed images is relatively small. A higher PSNR value generally implies better reconstruction quality, but it is worth noting that PSNR is known to be less sensitive to perceptual differences than other metrics.

The reported average SSIM of 0.64 indicates that the reconstructed images preserve some of the structural similarities and luminance of the original images. SSIM is often considered a more perceptually meaningful metric than PSNR, as it takes into account the human visual system's sensitivity to changes in structure, contrast, and brightness.

References :

[1] X. Tang and X. Wang, “Face sketch synthesis and recognition,” in International Conference on Computer Vision, Institute of Electrical & Electronics Engineers (IEEE), 2003.

[2] Q. Liu, X. Tang, H. Jin, H. Lu, and S. Ma, “A nonlinear approach for face sketch synthesis and recognition,” in Conference on Computer Vision and Pattern Recognition, Institute of Electrical & Electronics Engineers (IEEE), 2005.

Experiments: We tried predicted sketches from color images, the below are the results

Original image	Sketch Image	Predicted sketch
		
		
		
		

Paper 2 : **Controlling Deep Image Synthesis with Sketch and Color**

Objective:

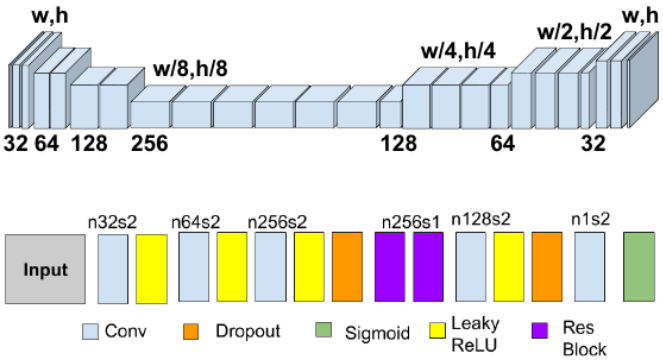
The objective of this research paper is to propose a deep adversarial image synthesis architecture that generates realistic images based on sketched boundaries and sparse color strokes. The authors aim to provide a user-friendly system that allows users to control the output by scribbling over the sketch to indicate preferred colors for objects. The network generates convincing images that satisfy both the color and the sketch constraints of the user in real-time. The paper compares their approach to recent works on sketch to image synthesis and demonstrates that their architecture generates more realistic, diverse, and controllable outputs. Additionally, the paper shows that the proposed method is effective in user-guided colorization of grayscale images.

Dataset:

Dataset used: CUHK Face Sketch (CUFS) database
Training Images : 3600
Testing Images : 236

Implementation Details:

Network Architecture:



The network architecture used in this case is an encoder-decoder type with residual connections. The input image is downsampled several times and then goes through a sequence of non-linear transformations before being upsampled to the desired output size. The generator uses three downsampling steps, seven residual blocks at the bottleneck resolution, and three upsampling steps. The discriminator is a fully convolutional network. Overall, the architecture has around 4.2 million learnable parameters.

Objective Function:

Given pairs of training images (input, ground-truth), where the input image is derived from the ground-truth photo (synthetically generated sketches and color strokes in our case), the following losses are used to train the network

1) Pixel loss (Lp)
2) Feature loss (Lf)
3) Total variation loss (Ltv)
4) Adversarial loss ($Ladv$)

With only pixel and feature losses, the network tends to average over all plausible solutions, due to the lack of a loss which pushes realism and diversity. Hence To encourage more variations and vividness in results, we experiment with adding an adversarial loss to the objective function. Total Objective function is the weighted combination of all these four losses:

$$\mathbf{L} = \mathbf{w_p L_p} + \mathbf{w_f L_f} + \mathbf{w_{adv} L_{adv}} + \mathbf{w_{tv} L_{tv}}$$

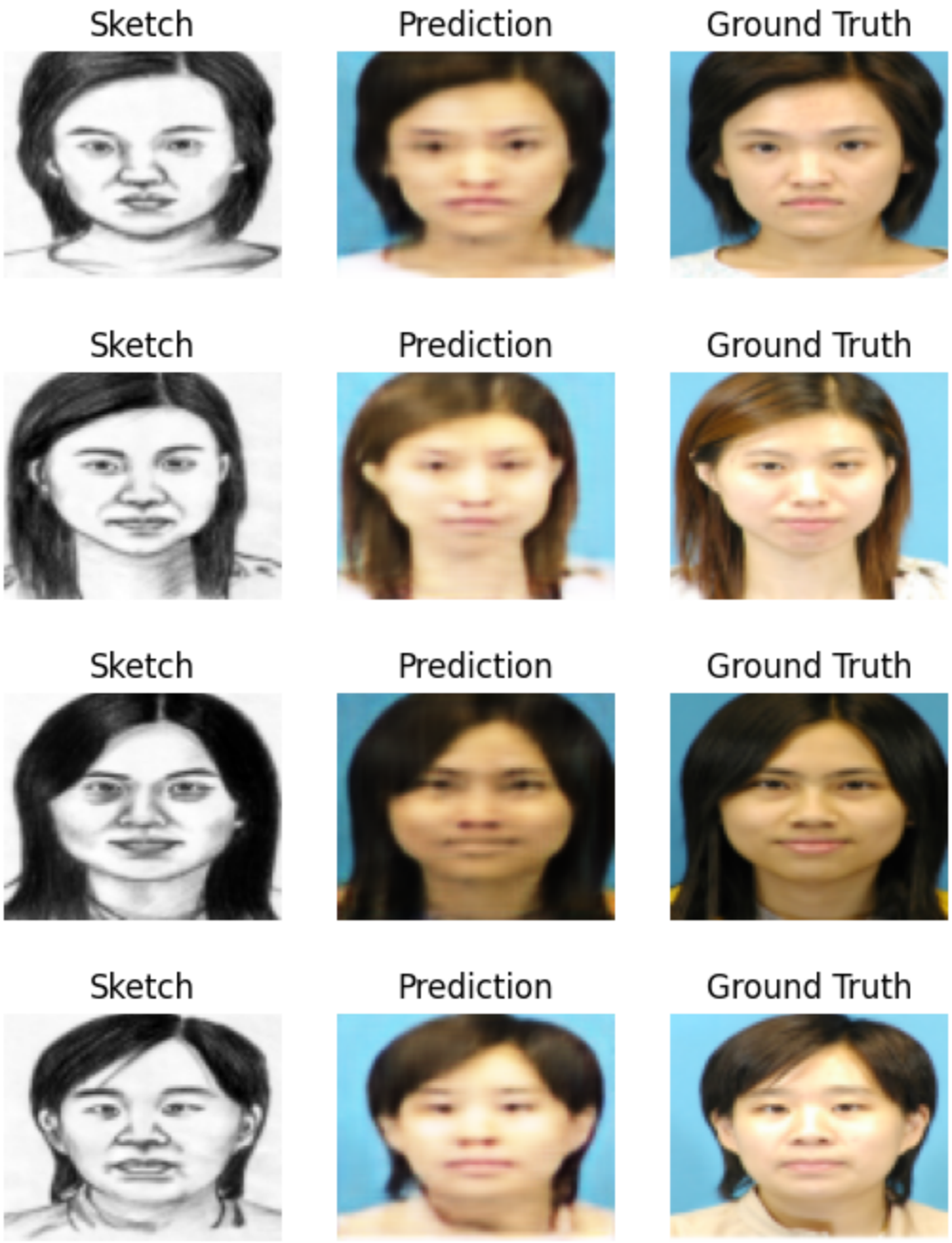
where,

$$\mathbf{L_{adv}} = -\sum log\mathbf{D}_{\phi}(\mathbf{G}_{\theta}(x_i))$$

Results:

```
Average psnr of all the test images is 53.58046812806303
Average ssim of all the test images is 0.9036602558601654
```

Model is initialized with pretrained weights and training is done on a very few image dataset and some of the predictions obtained are as follows:



Future Work:

- Network to be trained on a larger dataset containing various kinds of images
- The proposed paper claims that the network is a feed forward neural network which means the changes in the sketches result in a different Prediction in real time. This needs to be implemented
- This network is to be trained on input data containing Sketches combined with color strokes which results in predictions generated with those colors as claimed by the paper

References:

1. https://github.com/BKHMSI/Sketchback
2. https://github.com/Pingxia/ConvolutionalSketchInversion