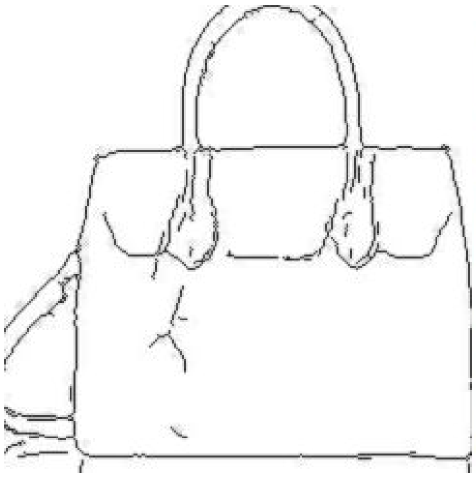
Machine Learning for the Arts
UCSD FALL 2019
**FINAL PROJECT**

# Edges to Stylized image



Chandhini Grandhi

**Description**

1. Concept
Artist Jean-Micheal Basquiat once quoted “ I want to make paintings that look as if they were made by a child”. This got me thinking on how to apply machine learning techniques we learnt in class to generate art that resemble children’s art, the one that involves lots of color. I decided to work on improving my project 3 - image to image translation using pix2pix mode. I combined image-to-image translation and style transfer to generate a stylised image from sketches.

The project takes in a dataset of faces of people obtained from CUHK dataset . It consists of two phases: The first model is a Pix2pix Generative Adversarial networks that takes in the image, does the processing required and generates the photos from this. Essentially, this step involves translating edges to faces. The second model is the Neural Style transfer whose content image is the image generated from pix2pix model and style image is chosen by the user. The final generated image is the stylized version of face image generated from edges. I first built the models and experimented with the dataset. Then, I used some of the images drawn by my friends and generated standalone faces from user sketches and performed style transfer on them.

2. Technique
The project involves two models

a. Pix2Pix GAN
   Pix2Pix is a generative adversarial network, a model design for image to image translation. I followed the Tensor flow implementation of [1] “Image to Image translation with Conditional Adversarial Networks” by affinelayer [2].

   Pix2Pix is a type of Conditional GAN where the generator is conditional of the input. It is comprised of generator model for outputting plausible images and a discriminator model that classifies images. The two models are trained simultaneously, where the generator wants to fool the discriminator and the discriminator wants to better identify fake images.

b. Neural Style Transfer

The key idea of Neural Style Transfer [3] is that it is possible to separate style representation and content representation in CNN. On a high level, the Neural Style Transfer uses a pretrained CNN, specifically of VGG16 architecture and defines two loss functions such that it minimizes the difference between the content image, style image and generated image. The content loss function makes sure that the correlation of activation of higher layers between the content image and generated image remain similar. The style loss function ensures that the correlations of activations between the style image and the generated image are similar.

3. Process
   1. Prepare the data

      I downloaded the photos dataset from CUHK dataset and followed the steps listed below:

      i. Resize the original images to 256x256 px

      ```
      python tools/process.py --input_dir data/original --operation resize
      --output_dir data/resized
      ```

      ii. detect edges from the photos

      ```
      python tools/edge-detection.py
      ```

      iii. Combine input and target images

      ```
      python tools/process.py --input_dir data/resized --b_dir data/edges --operation
      combine --output_dir data/combined
      ```

      iv. Split combined images into train and val directories

      ```
      python tools/split.py --dir data/combined
      ```

   2. Train the pix2pix model

      ```
      python pix2pix.py --mode train --output_dir faces_train --max_epochs 200
      --input_dir data/combined/train --which_direction BtoA --ngf 32 --ndf 32
      ```

I experimented with different epochs from 10 to 100 insteps of 10 and settled down with 100 as it produced plausible results.

3. Test the pix2pix model

```
python pix2pix.py --mode test --output_dir faces_test --input_dir
data/combined/val --checkpoint faces_train
```

4. Export the trained model

```
python pix2pix.py --mode export --output_dir single_test --checkpoint
faces_train
```

5. Generate standalone image
   The input images from users (my friends)  were stored in user_images folder.

```
python singleproduction.py --input_file user_images/resized/autodraw-1.png
--model_dir single_test --output_file user_images/output_autodraw-1.png
```

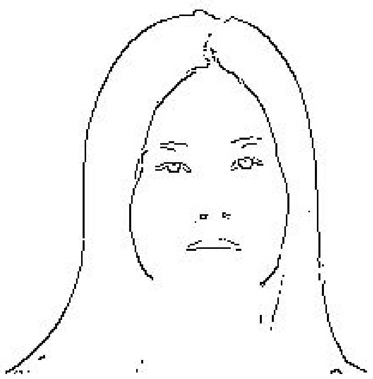
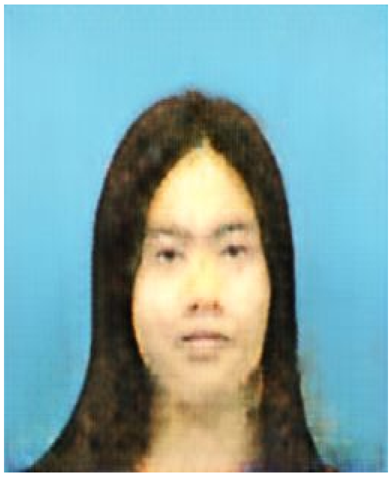
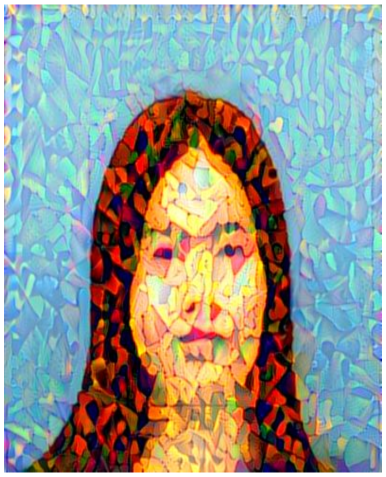
6. Generate Style Transfer for the input image
   The input image (content) and the style image were fed to  the style transfer model, run in jupyter notebook. The model was trained for 20 epochs to produce the stylized image.

**Result**

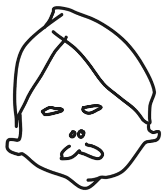
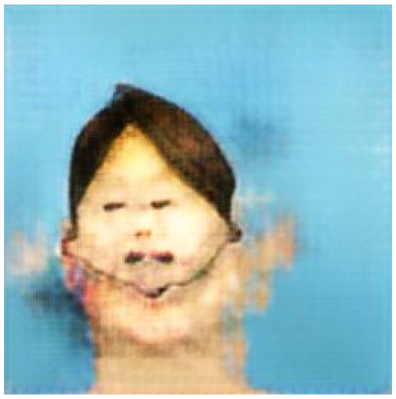
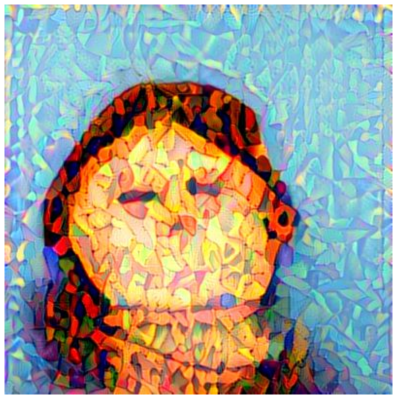
Two versions are results are presented below

1. Generated stylized faces from the validation dataset
2. Generated stylized faces from user's images without the true image.

Generated images from validation dataset

| *Input image* | *Edgestofaces* | *Stylized face* |
|---|---|---|
|  |  |  |

Generated images from user input

| *Input image* | *Edgestofaces* | *Stylized face* |
|---|---|---|
|  |  |  |

**Reflections**
The project produces decent results for images from the validation dataset. I arrived at this after tuning the learning parameters, but the model does not generalize well for unseen data, that is the user images. This can be because of the cleaning of data, where I tried to remove the background from the images but when the pix2pix mode learns the input images, it adds the background which leads to production of unclean images for user input.

Future work: Clean data to begin with and the project can be extended to detecting edges from videos and performing style transfer on them. The high level idea would be to divide the video into frames and fool the machine learning model to make it believe that it is working with images and put those generated images together.

**References**

[1] Philip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A.Efros (2016), “Image to Image translation with Conditional Adversarial Networks”, arvix: 1611.07004
[2] https://github.com/affinelayer/pix2pix-tensorflow
[3] Leon A. Gatys, Alexander S. Ecker, Matthias Betge (2015), “A Neural Algorithm of Artistic Style”, arvix: 1508.06576

**Code https://github.com/ucsd-ml-arts/ml-art-final-chandhini-g-1**