Stylish Selfie Generator Using Adaptive Instance Normalization CNNs, Color Clustering, and Color Space Manipulation

*Note: Sub-titles are not captured in Xplore and should not be used

Po-Chen Lin
Electrical Engineering dept.
National Tsing Hua University
Hsinchu, Taiwan
pochenlin0513@gmail.com

Chia-Wei Liu
Electrical Engineering dept.
National Tsing Hua University
Hsinchu, Taiwan
gaojibot@gmail.com

***Abstract*—In modern days, people are extremely concerned with their outfit due to the widespread of social application. As a result, having a stylish mugshot is indeed a big thing in concern. Due to this fact, we try to implement a user interface that enables the users to generate stylish mugshots with several styles by different algorithms and method.**

Keywords—style transfer, NPR, selfie, image transformation, sketch

I. Introduction

Graphic manipulation software such as Adobe Photoshop, provide built-in function to apply changes and stylish transformation on images. Some of these features are not easily customizable, and the choices of styles are limited. Our project aims at building the "Multiple Style Choices" application that enables the users to generate their unique mugshot in arbitrary style they want using CNNs and Digital Image Processing techniques. Also, for the ease of users, we build a system that allows arbitrary image size input and will also work.

The system of our application can be divided into three main parts: Generate NPR (Non-Photorealistic Rendering), Generate pencil sketch, and style transfer based on deep neural networks. Finally, we build a GUI (Graphical User interface) to let the users play with their stylish selfies at ease and generate result in real time as well as downloading it.

II. Dataset

A. Wikiart

Wikiart dataset contains more then twenty thousands of artworks which we implemented the style transfer neural network model with. To fit with the input of the model, we abandon those grey-scale images so that the remaining dataset has 23,585 images to feed in the model.

B. MSCOCO

MSCOCO dataset contains various common objects in our daily life. Although our task is to generate stylish selfies, we found that if the training data are consisted of purely human portraits, the model could only learn limited features from content images, resulting in poor performance on results. Thus, we introduced MSCOCO dataset as our content images in training.

C. CelebFaces Attributes dataset (CelebA)

CelebA is a large-scale dataset with more than 200K celebrity images. Initially, our model was trained on this dataset, but resulted in limited performance. Afterwards, we used this dataset simply testing performance and for example input in demonstration.

III. Previous Work

In our application, there are three main functions and corresponding technical details.

A. NPR drawing generation

The first function of our application aims to generate a non-photorealistic rendering drawing, representing what an artist would do: sketching the outline (edges) and details colors of the drawing.

*1) **Outline generation based on Canny edge detection***

Gradient-based edge detection offers a simple method to extract a line drawing from a photograph. We simply select the Canny edge detector. This algorithm produces binary representations of the lines in the photographs. Also, changing the two detection thresholds (strong threshold, weak threshold), the amount of detected edges can be tuned. Effectively this changes the conspicuousness of the line-sketch. Finally, the contour and internal line sketch is generated.

*2) **Color generation based on RGB-space k-means color clustering***

RGB-space k-means color clustering can generate the style of artwork drawing by reduce the dimension of colors in RGB-space of the picture. k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. For the k-means clustering applied on "RGB-space image", the distance now is seemed as the Euclidean distance of RGB value of the two pixel values.

$$\text{distance} = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}$$

After N centroids are decided, all pixels' distance between the centroids are calculated. Finally, as the ordinal k-means method: clusters those point into their nearest centroid and then changes their pixel value (color) to its cluster. Finally, the color detail of the drawing is generated.

*3) **Blending***

To get the final drawing output, we applied the line sketch on the color clustering image. The pixels of the latter lying on the line sketch are assigned to a designated pixel values (which means the color of the line sketch). The color of the line sketch can be tuned by changing the pixel RGB values, and the default is (128,128,128), which represent the color "gray".

B. *photo.Pencil sketch generation*

To generate the style of pencil sketch, we first converted the original RGB image into gray scaled image since there's only black and white color appears in the pencil sketched image. Then, we are going to invert the gray scaled image also known as "getting the image negative", which will be our "inverted gray scale image". This Inversion can be used to "enhance details". The operation produces its photographic negative, i.e., dark areas in the input image become light and light areas become dark. The following is the equation of the "inversion" operation on gray scaled image.

$$Q(i,j) = 255 - P(i,j)$$

Next, we use a Gaussian function to blur the image. In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. For this function, we use a Gaussian filter to make the picture smooth to simulate a imagery of shadow.

Finally, we are going generate the pencil sketched image by blending the "gray scale image" with the "inverted blurred image". This can be done by dividing the "gray scale image" by the "inverted blurred image".

C. *Style Transfer*

Our model structure is basically referred to the works from *Xun Huang* and *Serge Belongie* who issued the method of "Adaptive Instance Normalization" and the works form *Google Brain* to avoid checkerboard artifacts [1] [2].

1) Adaptive Instance Normalization

First, we take the output from several layers of VGG19 to encode the input photo and style photo as their content features and style features. The encoded content features of both are then calculated by an "AdaIN" layer to obtain the normalized content feature.

$$\mathrm{AdaIn}(x,y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \mu(y)$$

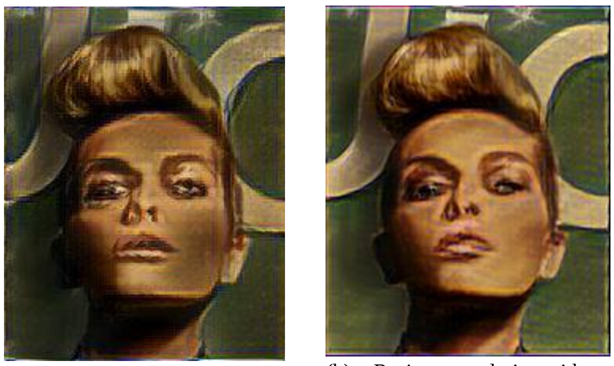
(a) Resize-convolution with nearest-neighbor (b) Resize-convolution with bilinear interpolation.

Fig. 1: Different interpolation method when upsampling.

With the normalized content feature, we train a decoder to transform it back to style transferred image. Finally, this output photo is encoded again by VGG19 to obtain content and style features so that we can calculate loss from encoded original photos and encoded output photo to optimize the decoder. Matching these statistics produces similar results with that using traditional Gram matrix, but is more conceptually cleaner and also results in generating outputs faster.

2) Resize-convolution

When using neural network to generate images, it is often that we see checkerboard patterns on the output image. This artifact is resulted from uneven overlap when doing deconvolution if the kernel size is not divisible by the stride. Moreover, even if we choose the kernel size and stride carefully, for example, stride 1 deconvolution, these artifacts can still leak through.

The alternative is to use resize-convolution approaches, that is, to apply nearest-neighbor interpolation or bilinear interpolation and then do normal convolution. The default upsampling layer in tensorflow is done using NN interpolation. After changing it to bilinear interpolation, we can see in Fig. 1 that there is significant reduction of checkerboard artifacts on the output images and it becomes smoother.

IV. Application Contents

A. *NPR (See Fig. 2)*

- Select a photo.
- Apply k-means clustering on RGB value.
- Canny edge detection.
- Generate output.

B. *Pencil Sketch (See Fig. 3)*

- Select a photo.
- Convert to grey scale.
- Invert the grey-scale image.
- Apply Gaussian filter
- Invert to get temporary result.
- Divide original image by temporary result
- Generate output.

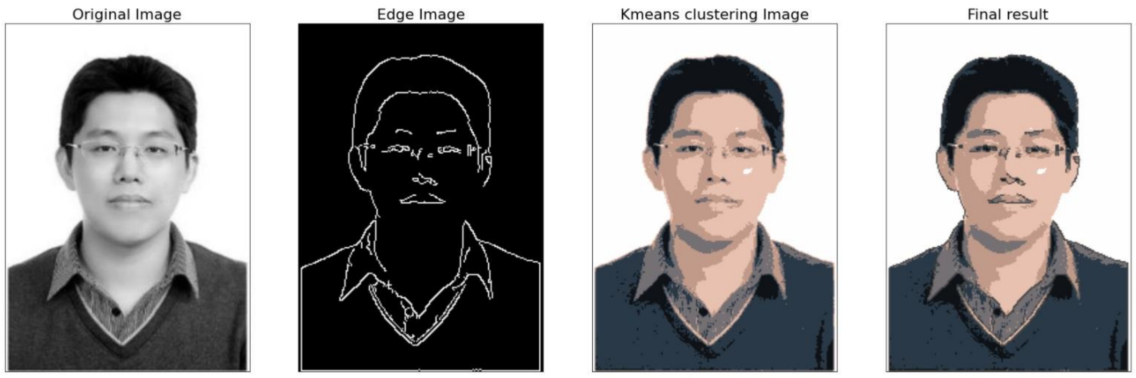

Fig. 2: The process of generating NPR drawing.

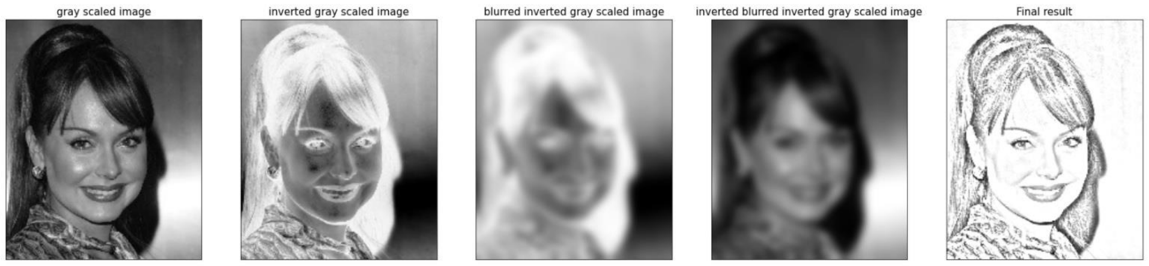
gray scaled image
inverted gray scaled image
blurred inverted gray scaled image
inverted blurred inverted gray scaled image
Final result

Fig. 3: The process of generating pencil sketch drawing.

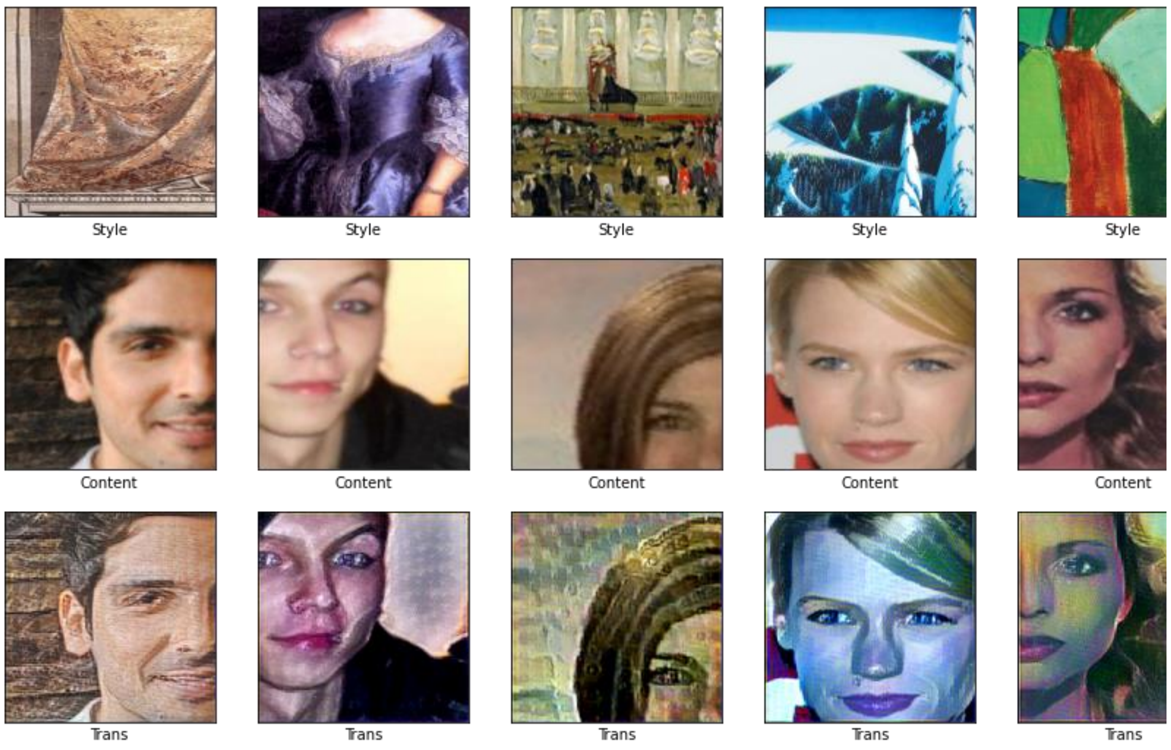

Fig. 4: Align style features on content photo.

C. *Style Transfer (See Fig. 4)*

- Select a photo and a style.
- VGG19 to encode.
- Normalize the encoded features by AdaIn layer.
- Decode the normalized feature.
- Generate output.

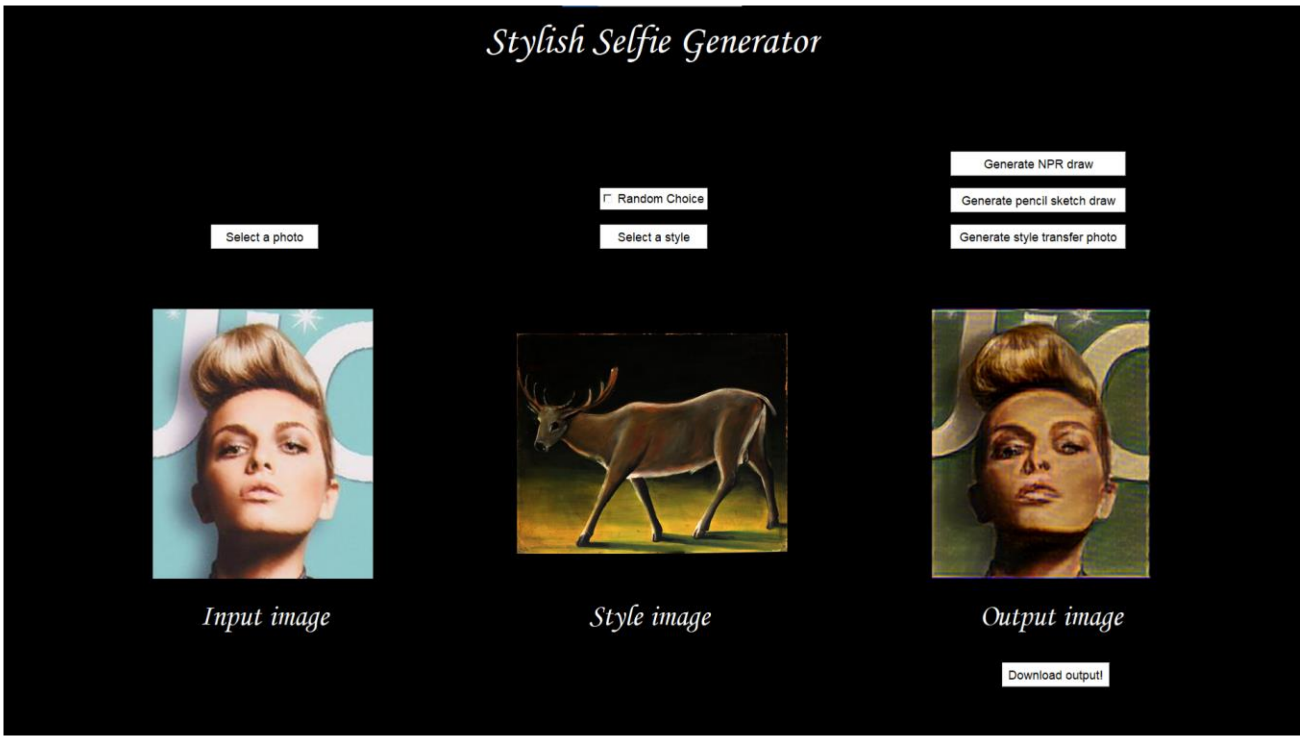

Fig. 5: Eventual works for user interface

V. User Interface

Our user interface is implemented with python package TKinter. See Fig. 5. First we select a selfie as input to generate stylish image. Select a mode to do transformation. Specifically, if we want to do style transfer, we can choose to use some selected fine styles or enable the random choice option (from 20K+ artworks) to have a surprise on unexpected result. Finally, you can download the result to your device and enjoy your stylish selfie.

VI. References

[1] Xun Huang and Serge Belongie (2017, March), Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization [v2], *arXiv*, Retrieved January, 1, 2021, from https://arxiv.org/abs/1703.06868

[2] Augustus Odena, Vincent Dumoulin, and Chris Olah, (2016, October), Deconvolution and Checkerboard Artifacts, *Distill*, Retrieved January, 1, 2021, from https://distill.pub/2016/deconv-checkerboard/?source=post_page