Making yourself into a work of art

Kiah Hardcastle * **and Julie Makelberge** †

*Neurosciences Program, and †Graduate School of Business, Stanford University

Project Milestone for CS 230: Deep Learning, February 2019

Here we begin the implementation of a deep learning framework that will swap the face in an artistic painting with the face in a separate image. For example, given a headshot and a picture of the Mona Lisa, this framework would inpaint the headshot face into the face of the Mona Lisa, while retaining the artistic style of the Mona Lisa. While face swapping has been done before, many previous projects have simply focused on swapping faces in images that contain only a face (e.g. cropped images), or implanting faces without taking into account the pose of the face. Thus, our approach is novel in that we will replaces faces in an image, while retaining the style and pose of the original face.

Neural style transfer.. give some introduction here

Approach and Results

Successful completion of our task requires several steps. First, we must identify the region containing the face in both images (art image = image A with face A, other image = image B with face B). Note that we assume that images with only one face are considered. Once face A and face B are identified and cropped out of the image, we must then align the faces. Specifically, we require that face B be in the same pose, or alignment, as face A. Using the aligned faces, we then generate a new face C, using neural style transfer (or visual attribute transfer, see below). Here, face C contains the content of face B, in the style of face A. Next, we will replace face A with the new face C in the image. Finally, we will smooth the boundaries between the excised region surrounding face C and the rest of the image.

Data collection. Fortunately, as many of our methods will rely on pre-trained networks, we do not need large numbers of new images with which to train our network. However, there are large datasets of artwork and faces available on the internet, which we plan to take advantage of here in order to demonstrate our approach. To generate our dataset of artwork, we use the Web Gallery of Art. This dataset is free to use for educational purposes and contains the meta data and links to images for more than 45,000 works of art, 31,000 of which are paintings. In order to create the dataset, we wrote write a script to download these images (see github repo). To generate a dataset of faces, we have used our own headshots. However, we plan to use the celebA dataset if more face images are necessary.

Data pre-processing: face-cropping. The first step in our pipeline is to identify the face in both images (image A and image B), and generate new images A' and B', which are the cropped-out face of images A and B respectively. To accomplish this, we implemented an open-source face recognition from the OpenCV package. Following the procedure detailed in this blogpost: https://medium.com/analytics-vidhya/how-to-build-a-face-detection-model-in-python-8dc9cecadfe9, we used Haar Feature-based Cascade Classifiers to detect the bounding box of the face. In brief, Haar features are similar to the filters in a CNN, but different in that they are not learned during training - they are pre-determined. Each filter can be used in a classifier to determine whether or not a face is present in a bounding box of the image. While the classification based on one Haar filter is not great, combining the output of many filters (e.g. through boosting) will return a relatively good classification. The OpenCV implementation organizes the features into a cascade, so that a lot of filters are "tried" if the image within the bounding box seems promising (i.e. early filters classify the image as a face). The OpenCV implementation contains pre-trained filters for faces, which we can load and then use to detect face regions within an image (see code in the github). An example of an output of the model is given below in Figure 1:

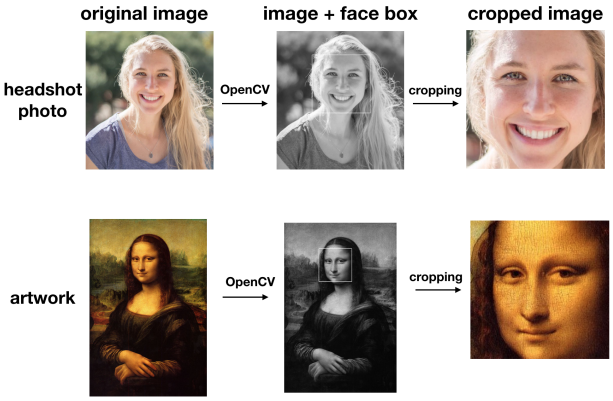

Fig. 1: Examples demonstrating face cropping pipeline.

Data pre-processing: face-alignment. The second step in our pipeline is to properly align the faces so that the pose of the face is similar between the two images. Specifically, we want face B to be in the same pose as face A - e.g. the major face landmarks (eyes, nose, mouth, chin) should be in similar locations. To accomplish this, we must first identify important landmark features in both images, and then apply a transformation to get face B in the same pose as face A. In order to identify salient landmarks, we used the dlib face detection and landmark identification package, and based our code off of this github repository: https://github.com/italojs/facial-landmarks-recognition-. An example of the output of this procedure is given below:

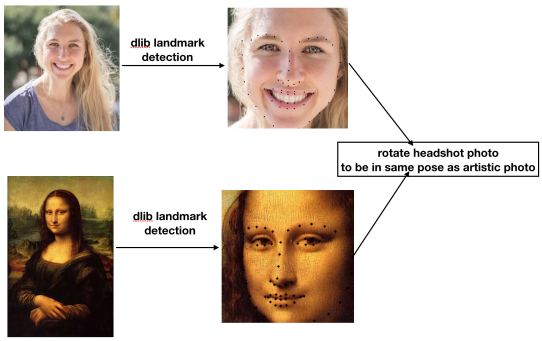

Fig. 2: Face alignment pipeline with example landmark detection.

We will then implement a function that will transform the headshot photo into the same pose as the artistic photo by applying a transformation that will align the landmark points. While we have not completed this yet, we plan on completing this step for our final project. There are several previous implementations of this task, including one found in this paper: Face Alignment in Full Pose Range: A 3D Total Solution by Xiangyu Zhu et al.

Style transfer between face images. The third step in our pipeline is to generate a third image that contains the content of face B (the non-artistic face) in the style of face A (the artistic face). There are several possible methods to accomplish this task. One simple method would be to use Neural Style Transfer in a manner similar to the Coursera CNN module homework, or using another implementation of Neural Style Transfer. While this method is not face specific, and thus may not perform very well, it is relatively straight-forward to implement and iterate on. A second method would be to follow the algorithm defined in the paper "Visual Attribute Transfer through Deep Image Analogy", by Jing Liao et al (CITE). This method differs from traditional Neural Style Transfer in that it focuses on matches patches rather than pixels, among other differences. Finally, a third method would be to follow the algorithm described in the paper "Painting Style Transfer for Head Portraits using Convolutional Neural Networks" by Selim et al. (CITE). In this method, they use network trained on faces specifically, and so the style transfer is targeted towards head portraits.

While ultimately we feel that the third method is the most promising, as it is focused on style transfer of portraits, thus far we have tried more traditional neural style transfer on the cropped images of face A and face B. We have tried two implementations of Neural Style Transfer. In the first, we altered the code that was used in the Coursera course to perform neural style transfer on our two images. Briefly, in this method, we generate a new image C with the content of face B and in the style of face A. To do this, we minimize the following loss function:

$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

where

$$J_{content}(C, G) = \frac{1}{4 n_H n_W n_H} \sum_{all} (a(C) - a(G))^2$$

and

$$J_{style}(C, G) = \sum_{l} \lambda_l \frac{1}{(2 n_H n_W n_H)^2} \sum_{ij} (G_{ij}(S)^{(l)} - G_{ij}(G)^{(l)})^2$$

where $a(C)$ are the activations of single layer of a pretrained VGG-19 network with the content image as input, $a(G)$ are the activations of the same layer of the VGG-19 network with the newly generated image put through, $G_{ij}(S)^{(l)}$ is the Gram matrix of activations of a single layer (l) of the same network when the style image is used as input, and finally $G_{ij}(G)^{(l)}$ is the Gram matrix of activations of a single layer of the same network when the newly generated image is used as input. Minimizing this loss function by altering the pixels in the generated image will result in a new image that contains the content of the content image, and the style of the style image.

While we were able to successfully perform neural style transfer using our code from the homework, we found another implementation of the same algorithm that gave us a much better output (drawn from here: https://colab.research.google.com/github/titu1994/Neural-Style-Transfer/blob/master/NeuralStyleTransfer.ipynb). The result of this implementation is shown below in Figure 3:

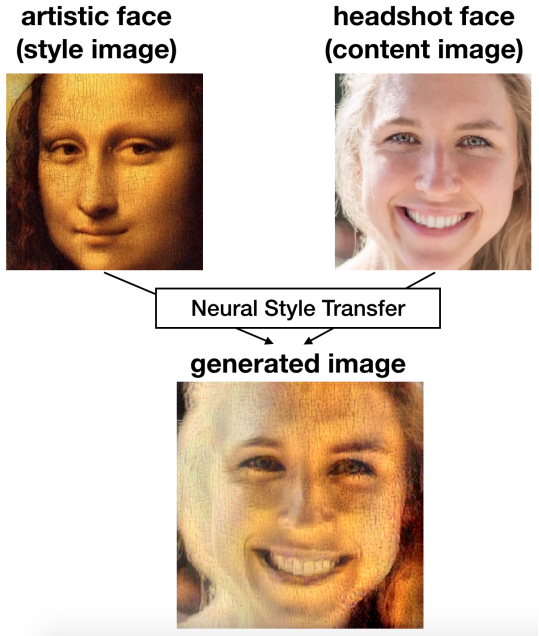

Fig. 3: Example output using Neural Style Transfer.

Face-replacement in artistic image. Finally, we must replace the original face A with the new, properly aligned face C. This task contains two components: first, we replace the pixels within the original bounding box in image A with the new image C. Second, we must then smooth the boundary between in the inpainted pixels and the border. The first step of this procedure is quite easy. To do the second step, we will remove the area around the bounding box of the face (i.e. the boundary between the new image and the old image), and then use methods of inpainting in order to fill in the boundary.

Summary

In summary, successful completion of our project will require identifying faces in two images, cropping the faces out, aligning the faces, performing style transfer to generate a new face image, replacing the artistic face with the newly generated

face, and smoothing the edges so the artistic image looks similar (except for the face) to the original. Thus far, we have been able to identify faces in images, crop them out, did the first step in aligning the faces, and performed neural style transfer. To finish the project, we plan on first working on aligning the faces, replacing the image with correct smoothing and inpainting if need be, and finally improving our neural style transfer by trying other implementations or by varying hyperparameters and training time in our current implementations.

Code

All code for the project, and in fact everything for the project, can be found in the following github repository: https://github.com/khardcastle/face-in-art.

ACKNOWLEDGMENTS. We thank Andrew Ng and Kian Katanforoosh for teaching us the techniques used in this project, and to our project TA Kaidi Cao for guiding us towards workable solutions for our problem and pointing us to relevant literature.