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

Kiah Hardcastle

Neural style transfer is the technique of using deep neural networks to transfer the style of a given reference image to the content of another. Traditionally, such problems have been adressed with classic image processing techniques such as histogram matching [8]. However, in 2015 Gatys et all. [2] introduced a novel technique that leverages the power of Convolutional Neural Networks to emulate famous painting styles in natural images. In their seminal paper, they proposed to use the encoding from different layers of a pre-trained CNN, to capture the style elements and content elements of images. They then used an interative approach to optimise an image with the objective of minimizing the distance between the style elements of the image and the reference work of art, while keeping changes to the content as small as possible.

Their work has inspired a number of new neural transfer algorithms, ranging from general models such as the work of Li and Wand [5] that combined generative Markov random field models with deep convolutional neural networks, to highly specialised domain-specific models such as Jiang anf Fu's Fashion style generator [4]. While general models have shown great potential in a large number of applications, they have generally introduced visual artifacts which are especially striking in faces, given human's sensitivity for facial irregularities. The work of Selim et all. [10] introduced the first approach for single-example based head portrait painting not constrained to a specific style. Their approach was successful by introducing additional constraints that exploit human facial geometry through the notion of gains maps. However, their methodology focused on front-facing cropped facial images and does not yet generalise to different facial positions or introduction within the wider context of a full painting.

In this work, we aim to expand on the work of Selim et all by introducing a framework that combines different Neural Transfer techniques to tackle the challenge of introducing faces of slightly varing positions in a work of art. Our goal is to create a resulting image that looks similar to our reference work of art, but with the exception of having a different person in mind at the time of painting. This technique would be valuable in the world of generative art as it alows different and personalised renderings of the same types of art work.

Approach and Results

We have broken down our framework in a number of 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, or in images with multiple faces we would only consider the most prominent one. Once faces A and B are identified and cropped out of the images, we must then align the faces to be in similar positions. Specifically, we require that face B be in the same pose, or alignment, as face A in our reference work of art. Using the aligned faces, we then generate a new face B', using neural style transfer (or visual attribute transfer, see below). Here, face B' contains the content of face B, in the style of face A. Next, we will replace face A with the new face B' in our reference work of art. Finally, we will smooth the boundaries between the excised region surrounding face B' 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 open-source datasets of artwork and faces available on the internet, which we plan to take advantage of in order to demonstrate our approach. To generate our dataset of artwork, we use the Web Gallery of Art [9]. 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 initially plan to use own headshots. However, we plan to use the celebA dataset [7] if more face images are required.

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 the Medium blogpost [11], 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:

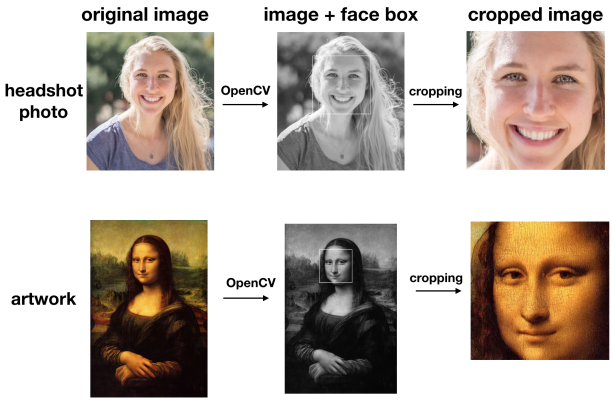

Fig. 1: Examples demonstrating face cropping pipeline.

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 a github repository [3]. An example of the output of this procedure is given below:

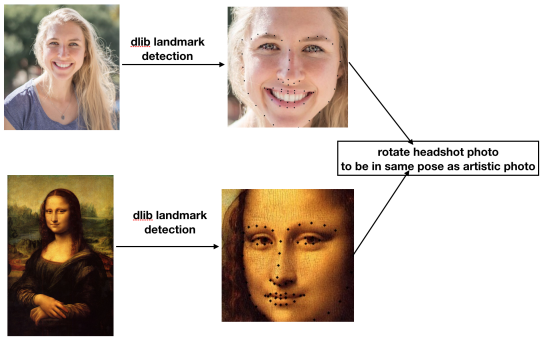

Fig. 2: Face alignment pipeline with example landmark detection.

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 deliverable. There are some previous implementations of this task, including one found in the Face Alignment in Full Pose Range paper by Xiangyu Zhu et al. [12]

Style transfer between face images. The third step in our pipeline is to generate a new image B' 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 a good straightforward approach to implement and further 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 [6]. This method differs from traditional Neural Style Transfer in that it focuses on creating a pixel by pixel mapping based on higher level features between the images, 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. [] 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 focussed on more traditional neural style transfer on the cropped images of face A and face B given our time constraints and the availability of pre-trained CNN weights. We are currently working with 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. In this method, we generate a new image B' with the content of face B and in the style of face A. To do this, we minimize the following loss function:

$$J(G) = \alpha J_{content}(B, B') + \beta J_{style}(A, B')$$

where

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

and

$$J_{style}(B, B') = \sum_{l} \lambda_l \frac{1}{(2n_H n_W n_H)^2} \sum_{ij} (B'_{ij}(A)^{(l)} - B'_{ij}(B')^{(l)})^2$$

where $a(A)$ are the activations of single layer of a pre-trained VGG-19 network with the content image as input, $a(B')$ are the activations of the same layer of the VGG-19 network with the newly generated image put through, $B'_{ij}(S)^{(l)}$ 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 $B'_{ij}(G)^{(l)}$ 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 [1]. The result of this implementation is shown below in Figure 3:

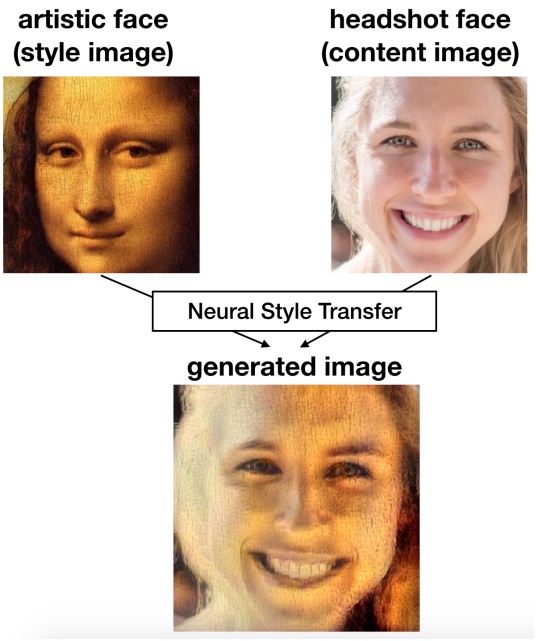

Fig. 3: Example output using Neural Style Transfer.

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 B'. 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.

1. Dlib. Neural style transfer implementation, 2018.
2. Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576, 2015.
3. github. Dlib face detection and landmark identification, 2018.
4. Shuhui Jiang and Yun Fu. Fashion style generator. In IJCAI, pages 3721–3727, 2017.
5. Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2479–2486, 2016.
6. Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. arXiv preprint arXiv:1705.01088, 2017.
7. Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. Retrieved August, 15:2018, 2018.
8. László Neumann and Attila Neumann. Color style transfer techniques using hue, lightness and saturation histogram matching. In Computational Aesthetics, pages 111–122. Citeseer, 2005.
9. Web Gallery of Art. Web gallery of art searchable database of european fine arts and architecture, 1996.
10. Ahmed Selim, Mohamed Elgharib, and Linda Doyle. Painting style transfer for head portraits using convolutional neural networks. ACM Transactions on Graphics (ToG), 35(4):129, 2016.
11. Vidhya. Face detection model in python, 2018.
12. Xiangyu Zhu, Xiaoming Liu, Zhen Lei, and Stan Z Li. Face alignment in full pose range: A 3d total solution. IEEE transactions on pattern analysis and machine intelligence, 41(1):78–92, 2019.