Artistic Facial Replacement with Convolutional Neural Networks

Vlad Chilom, Denis Semenenko, and Eli Waalkes

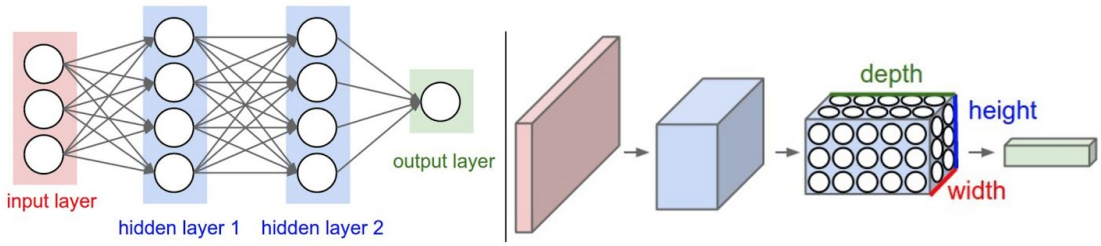
Abstract

In this project, we create a software program that allows a user to insert their face into a piece of artwork using convolutional neural networks. More specifically, through a user input image of a person and a piece of art, we successfully detect the faces within both images, extract the artistic style of the art face, impose the aforementioned style on the regular face, and overlay the stylized regular face onto the art face in the original piece of art in order to conduct artistic facial replacement. Another accomplishment of this project rests within the investigation of the face detection and style extraction methods. This includes an in-depth analysis on the architecture of the underlying VGG convolutional neural network—the network that powers both face detection and style extraction. Finally, we offer a set of results that allow for both further growth and greater inspection with regards to the strengths and limitations of the facial replacement program, the methods for face detection and style extraction, and the underlying convolutional neural network.

Introduction

Before delving into the face detection and style extraction methods, some important context must be provided for convolutional neural networks. Foremost, "[convolutional neural networks] are designed to process data that come in the form of multiple arrays, for example a color image composed of three 2D arrays containing pixel intensities in the three color channels (Lecun, 2015)". In other words, the functionality of convolutional neural networks best serve

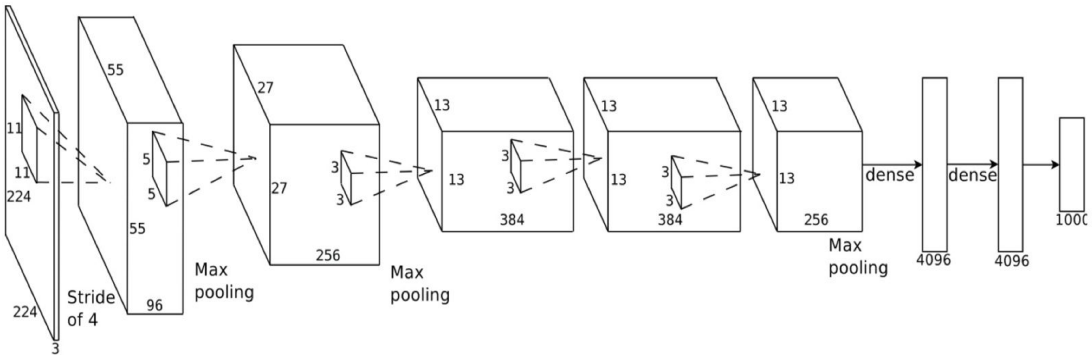
Figure 1: Contrast between neural network and convolutional neural network (image taken from cs231n.github.io).



computer vision related problems—a conclusion supported by the networks' wide adoption in the computer-vision community (Lecun, 2015).

While convolutional neural networks are still neural networks, their architecture differs from that of a regular neural network. Convolutional neural networks function through a series of convolving and pooling layers (Lecun, 2015). A filter of varying size iterates over the input in order to convolve the data. The resultant data is either convolved again or pooled through some operation—such as max or average pooling. Considering that the convolving works to resolve features within the input, the repetitive convolution and pooling allow for features to be comprised of other local features. For example, in the case of images: "local combinations of edges form motifs, motifs assemble into parts, and parts form objects (Lecun, 2015)." This localization of feature detection differs from the fully-connected nature of the regular neural network. A visualization of the process can be seen in figure 1 where the neural network on the left is fully-connected while the convolutional neural network on the right becomes a three-dimensional stack of convolved data that loses height and width but gains depth with each pooling—a sign of local features assembling into larger features. Figure 2 offers a more specific inspection of the convolving and pooling. With this basic depiction of a convolutional neural network, the VGG convolutional neural can now be analyzed.

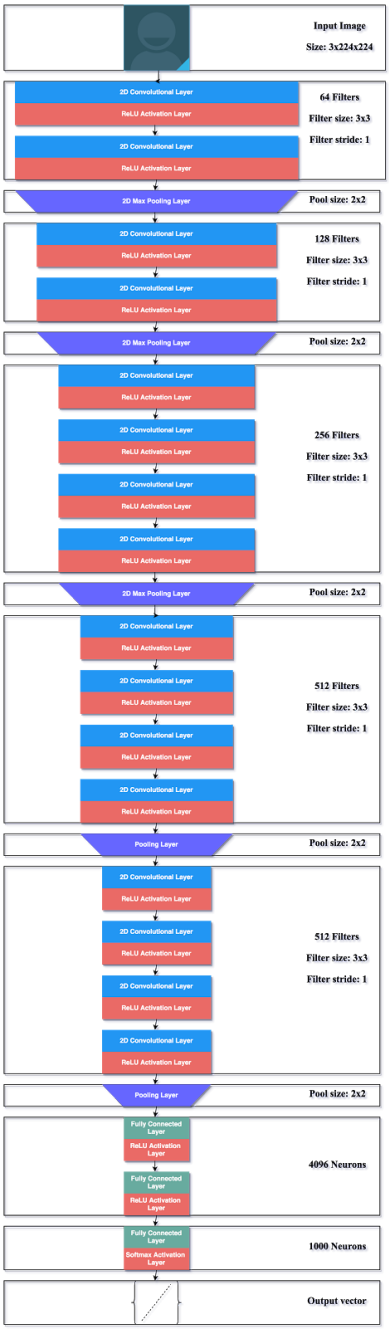
Figure 2: Convolutional neural network (image taken from medium.com).



The VGG neural network is one of multiple convolutional neural networks developed by the Visual Geometry Group at the University of Oxford (Simonyan, 2014). Their work revolved around investigating the "the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting (Simonyan, 2014)." The style extraction method used in this project utilized the 19-layer version of the neural network—formally named the VGG-E; however, the architecture of all versions of the VGG networks are essentially the same disregarding the number of layers, so we will continue to refer to the network as the VGG (Simonyan, 2014). The VGG network uses a relatively small receptive field as it has filters of 3x3 size (Simonyan, 2014). The convolution stride—stride being the distance in pixels that the filter moves over each iteration passing over the image—is 1 (Simonyan, 2014). On the other hand, the pooling size is 2x2 and the pooling stride is 2 (Simonyan, 2014). In total there are 5 pooling layers, 16 convolutional layers, and 3 fully connected layers that process the convolved data at the end in order to classify the picture into one of one thousand classes; note that these final fully connected layers are not used within the style extraction, and that average pooling is used instead of max pooling (Simonyan, 2014). Furthermore, all of the hidden convolutional layers use ReLU as the activation function—this being the function that signals feature detection

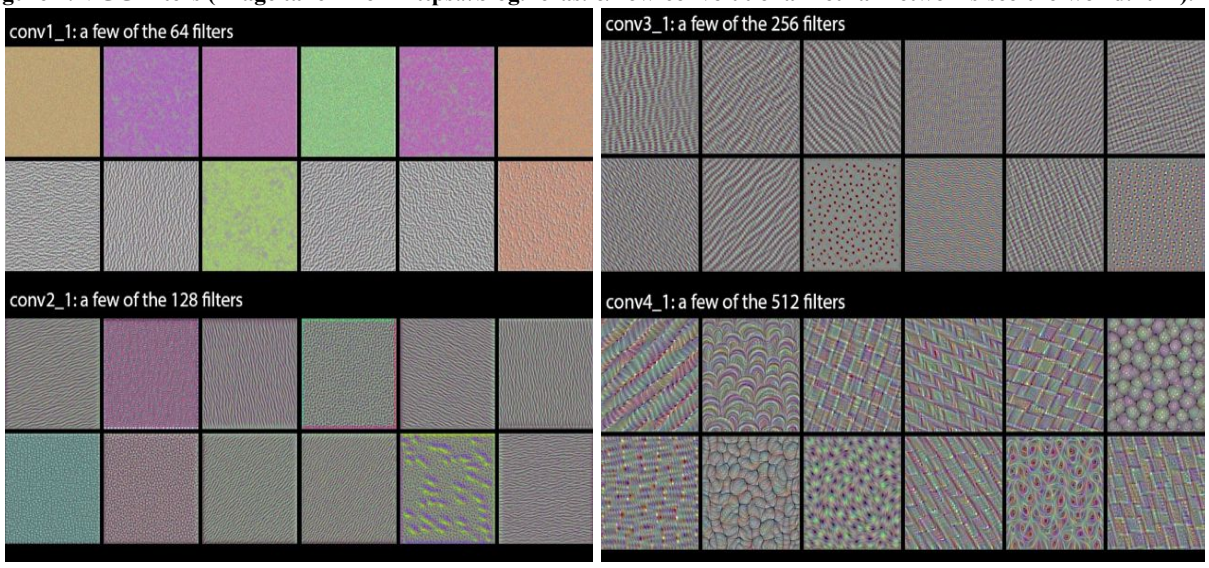
(Simonyan, 2014). A complete view of the VGG network can be seen in figure 3. Considering the aforementioned architecture of the network, the nature of the filters is the final pertinent portion of the VGG that must be discussed in order to inspect the facial replacement algorithm.

Figure 3: VGG convolutional neural network.



As mentioned before, the VGG uses 3x3 size filters in order to convolve the image. Note the image in figure 4 when considering the following discussion. The first convolutions (conv1_1) extract at most the color of the image, and some semblance of the texture. In following convolutions—those done after pooling when there are more filters, the filters seem to highlight basic patterns and shapes within the image (conv3_1). Finally, in the last set of filters, the filters highlight somewhat complex objects and textures such as feathers, fabric, or eye-like shapes. The size growth in this features directly correlates to the idea mentioned earlier that convolutional neural networks stack their features in order to progressively create larger features (Lecun, 2015). The range of objects and textures that these filters highlight provides the first inkling of intuition regarding how a convolutional neural network can successfully extract the style from a piece of art. At this point, the VGG has been sufficiently explained for the purpose of the facial replacement algorithm.

Figure 4: VGG filters (image taken from https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html).



Methods

In order to conduct our facial replacement, we used the VGG network as the basis for both our facial landmark extraction and artistic style extraction. Initially, we implemented the stylistic features as described by Gatys et al., before shifting our network to reflect the progress of Wand and Li by including Markov Random Fields for a multi-layer synthesis and a better layout on a more abstract level. The following illustrates the approach to our facial replacement algorithm.

A. Network Structure

Using the VGG as a baseline, we propagated the content image and the style image through different levels of the network to create "target" outputs. Each level reflected a different level of abstraction of the image. With each iteration the loss function accounted for error on each level of the network, minimizing the loss to create an output that conformed with the more specific pixel detail found at the deepest level of the network as well as the artistic style and structure founder at higher levels of abstraction. The input is traditionally random white noise that with each iteration decreases the distance between the style and content targets according to the loss function. See figure 8 for a depiction of the style and content extraction.

For facial extraction we initially used the VGG-D 16-layer neural network trained for facial recognition. However, its performance on painted faces was not sufficient for our purposes, since most of the time it failed to detect facial landmarks. In our search for a better model we found that the best result is produced by the network and algorithm described in HyperFace paper (Rajeev Ranjan et al, 2016). Their architecture has two main features: first, they train a multipurpose model to simultaneously detect faces, landmarks, poses, and gender. It

has been shown that training a network for multiple correlated tasks improves the performance of each individual task (Zhu 2012, Zhu 2015). In general terms, it happens because the model learns to extract more abstract information by its lower levels, that are used simultaneously for each purpose, therefore yielding better generalization. Second, they recognize that different tasks, for example, landmark estimation vs. gender recognition, require information about the picture that varies in its level of abstraction. Hence, they conclude that they need to use hyperfeatures from several layers of CNN at once. In order to implement this idea, they use AlexNet model pretrained on faces and merge the results of three different layers into a single layer, after reducing dimensionality of each layer to the deepest one by adding another convolutional layer with 1x1 filter windows. This concatenated layer is then followed by a fully connected layer with 3072. Finally, the network is branched into five layers, one for each individual task, with every branch represented by one fully connected layer with 512 neurons, and an output layer. ReLU activations are used in every part of the model.

Moreover, as described in [Rajeev Ranjan et al, 2016], in order to produce the final result the image is divided into regions, and each region is fed into the network separately in order to detect which ones contain a face. Then the regions with the high level of confidence are merged back together, and the final result is produced based on that region only, and not the whole picture. This improves accuracy of their approach.

The results are quite impressive (figure 5), HyperFace managed to outperform every state-of-the-art model in landmark estimation. As can be seen from figure 6, it was effective in our case too, achieving high accuracy with most of the paintings we tested it on.

Figure 5: Performance of different models for landmark estimation on AFLW dataset, taken from [Rajeev Ranjan et al, 2016]

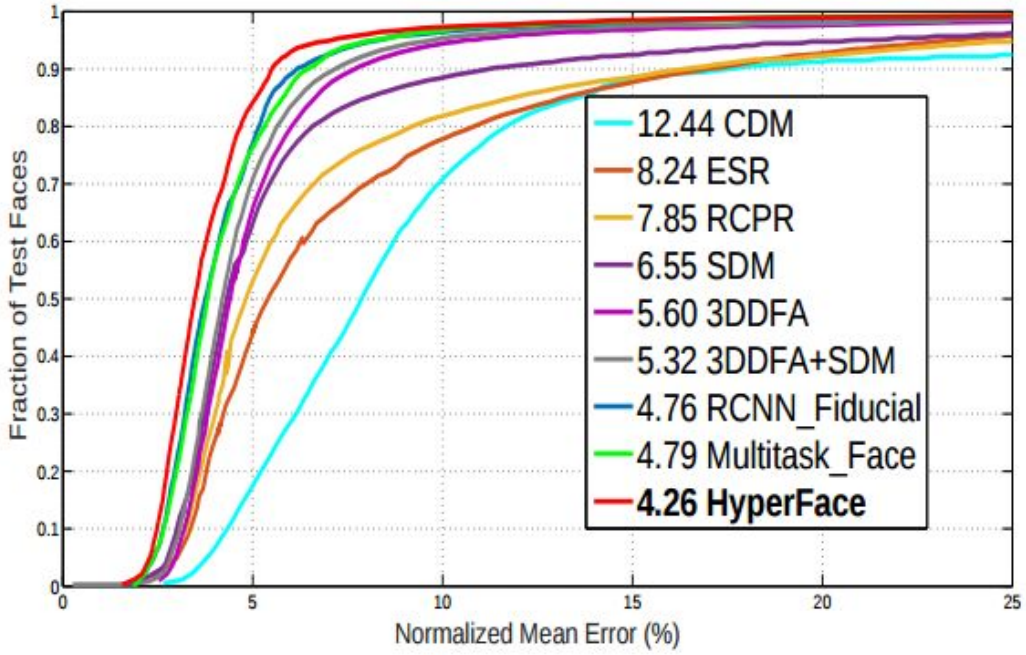


Figure 6: Landmark recognition in paintings with HyperFace

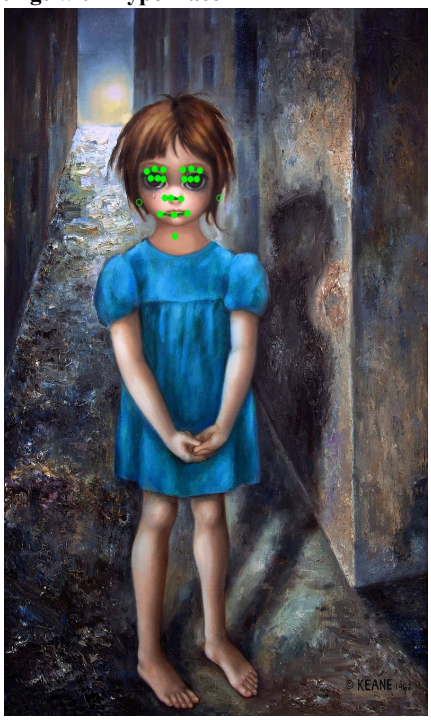


Figure 7: Illustration of the HyperFace architecture from [Rajeev Ranjan et al, 2016]

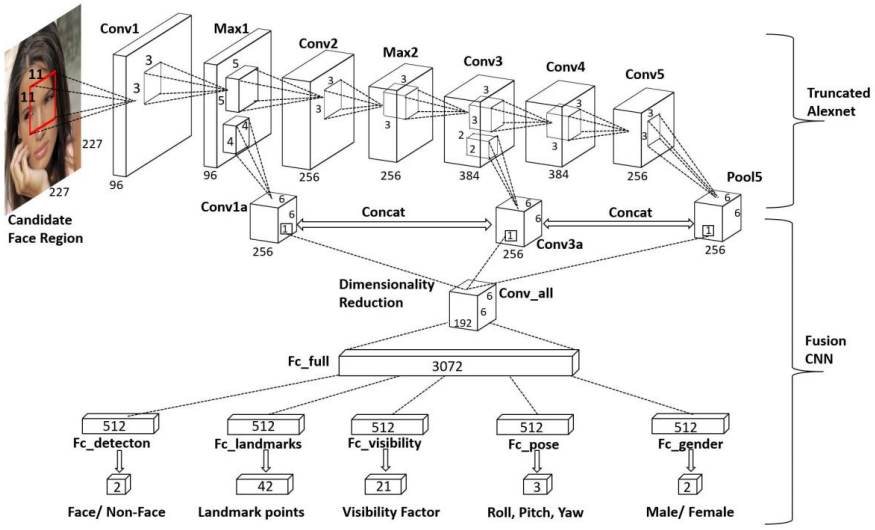
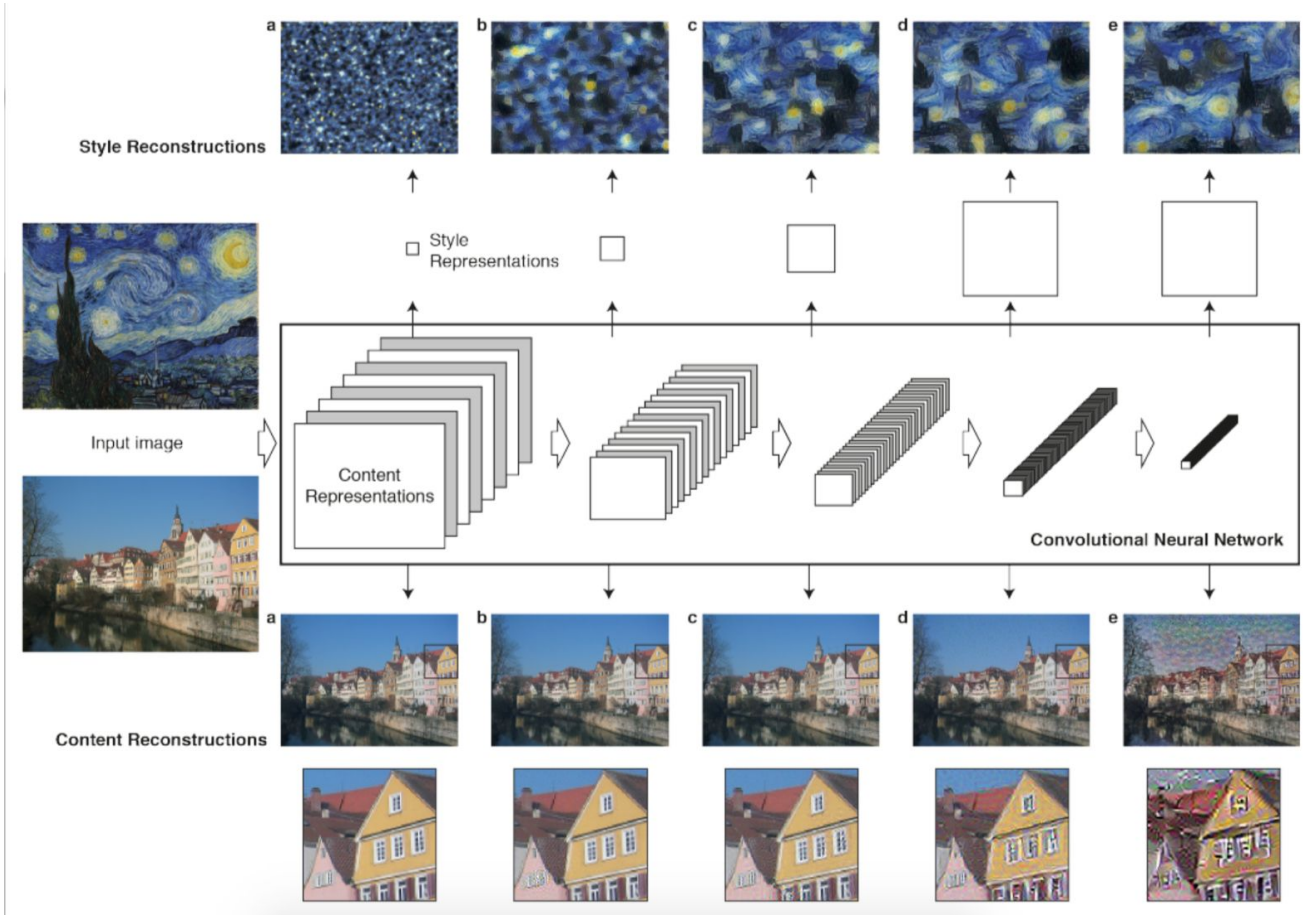


Figure 8: Style and content extraction (image taken from Gatys 2015).



B. Loss Function

The Loss Function was fundamental to the success of the process. We initially used the loss function as described by Gatys et al. before implementing the change in style loss function as described by Wand et al. Below are the descriptions of the two loss functions.

i. Gatys et al. utilized a joint loss function that combined a loss of style and a loss of content, each weighted according to user preference. The content loss was a construction of the filter outputs of each position of the original image (P_{ij}) and the generated image (F_{ij}) . The squared error loss is then taken for each layer *l.* The style loss was formulated on a layer by layer basis but minimized by taking all layers into account. E_l was the error at each level according to squared error between a Gram matrix of the original image and the image to be generated. W_l is a weight vector which allowed the style to be computed on a multi-layer basis. The loss functions can be seen in figure 9, *f* describes the content image, *a* describes the style image, and *p* is the output image. As can be seen from the results in the next section, implementation of Gatys does not perform as well on faces, as it does on landscapes. This is especially visible in figure 12. The reason for this is that the style loss functions is not sensitive to particular regions, the resulting style transformation is the collective style of the whole painting, rather than a local area. Intuitively, the style used to paint the eyes in a portrait must be different from the style used to paint the lips. It is especially so in terms of the color choice. Therefore, in order to produce good results we had to find another approach that defines style locally.

ii. Wand et al. describe such approach (Wand, 2016). The style loss function they use introduces a form of penalty to preserve style locality. Specifically, instead of using Gram matrices they utilize a Markov Random Field model with the Convolutional neural network

resulting in a product that more accurately to maintains local patterns of style by evaluating loss from different local "patches." By shifting the loss function to include this MRF structure we were able to recreate details on a more localized basis. The loss function calculates the loss for each "patch" and then uses the patch that yields the lowest loss. This shifts the style optimization from a process of pixel values to one of neural patches, which are blended linearly according to a texture term E_s. After calculating the results from each patch the algorithm optimizes the loss after applying a regularization term Y. The loss function is described in full by x in the formula below. The variables, x_s, x_c x represent the style, content and output images respectively.

Figure 9: Gatys loss functions.

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}\left(F_{ij}^{l} - P_{ij}^{l}\right)^2 \qquad \mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha\mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta\mathcal{L}_{style}(\vec{a}, \vec{x})$$

Figure 10: Wand loss functions.

$$E_s(\Phi(\mathbf{x}), \Phi(\mathbf{x}_s)) = \sum_{i=1}^{m} ||\Psi_i(\Phi(\mathbf{x})) - \Psi_{NN(i)}(\Phi(\mathbf{x}_s))||^2$$

$$\mathbf{x} = argmin E_s(\Phi(x), \Phi(x_s)) + \alpha_1 E_c(\Phi(x), \Phi(x_c)) + \alpha_2 \Upsilon(x)$$

C. Optimization

Regardless of the style loss implementation, we used L-BFGS optimization to calculate the loss function by minimizing the filter outputs of the style and content layers. L-BFGS is based on the BFGS but estimates the inverse Hessian Matrix to reduce memory needs. BFGS is a variant of Gradient descent that approximates the newton method, using the derivative of function and its Hessian Matrix. The Hessian matrix is a square matrix of second-order partial derivatives of a scalar valued functions. It describes the local curvature of a function with many variables. We implemented this optimization technique using a torch function built into the Neural Network library.

D. Implementation Details

Our product was built upon the Torch7 framework for deep neural networks. The VGG-19 was uploaded using the loadcaffe module for its use within torch. The process was ran using GPU processing with Nvidia CUDA-8.0 and the CuDNN deep learning libraries to increase computational efficiency. Without GPU-optimized computation the process could take between 20 minutes and many hours depending on the machine. On the machines in the Computer Science department with GPU-optimization the process took about 5 minutes to complete 1000 iterations.

E. Parameter Tuning

We made several parameters tunable, most notably: number of iterations, content weight, and style weight.

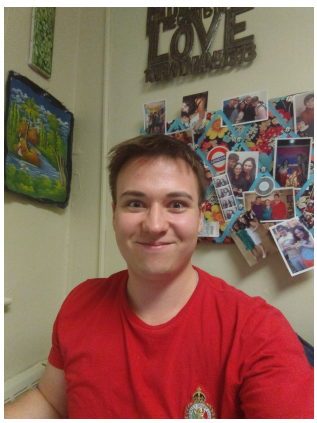
F. Product

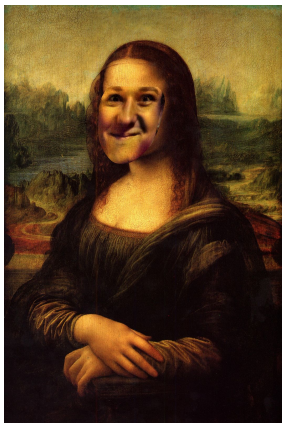
Combining everything our product would take in two images, a content photo of a face and a style image of a portrait painting. After extracting the facial landmarks, we used the artistic style transfer to transform the content image to reflect the style of face extracted from the painting. Following the artistic transfer, we re-impose the shifted face onto the painting, creating what hopefully would be a realistic output of an individual's face in the painting.

Results

After running the facial replacement program, using both the Gatys and Wand definitions of loss, on a few examples of faces and artwork, we produced a wide range of results. Most of these results assisted in crafting the ideas for the next version of the program; thus, the work became somewhat progressive. The first of these results can be seen in figure 11. The misplaced face on Mona Lisa hinted at a transformation issue regarding the face overlaying. The eyes match on the image; however, the overall feature sets of the faces do not align. This led to a different implementation in both the face detection, and in the pose used within the photo.

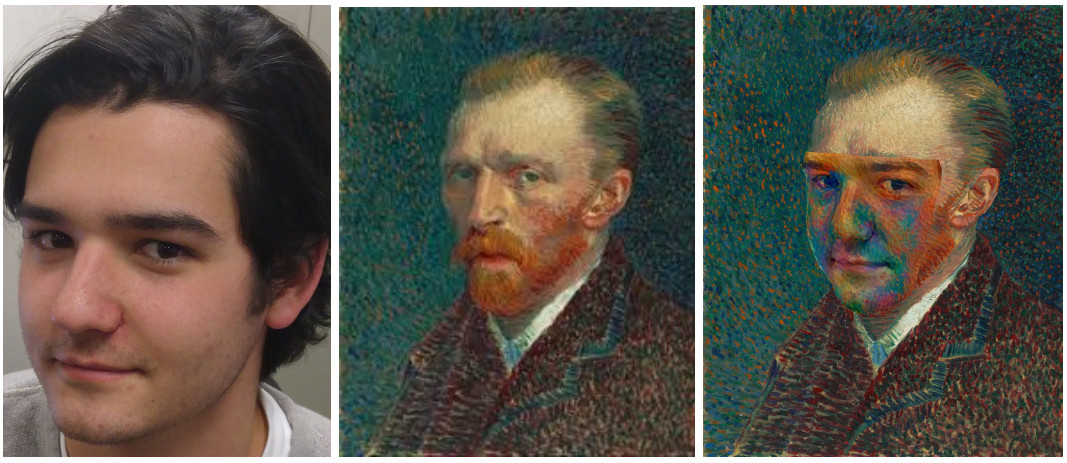
Figure 11: First results for facial replacement program using the Gatys loss function (Denis and Mona Lisa).





In our next approach, we rectified the issue with the pose and produced the result in figure 12.

Figure 12: Second results for facial replacement program using the Gatys loss function (Eli and Van Gogh).



In this case, the style from the artwork seems to have not transferred in a localized fashion. In other words, the style of Van Gogh's chin transferred to more than Eli's chin—the majority of his face is red. Considering this output used the Gatys loss function, we attempted to produce run the facial replacement program again with the Wand loss function instead. The results for that run can be seen in figure 13.

Figure 13: Final results for facial replacement program using the Wand loss function (Eli and Van Gogh).

With this output, it becomes clear that the Wand loss function transfers style in a substantially more localized manner. Thus, we can conclude that our best results stemmed from using the Wand loss function. The key to artistic style transfer is the loss function which maintains both the overall structure while being able to impart the style onto the result. With our Gatys et al. implementation it was clear that the style was drawing from the entire image. When dealing with unique and local features like eyes, beards and other facial features, we needed the algorithm to extract only style that pertained to the specific regions. We can see in the localized color of skin and beard, that the Wand loss function successfully localized the style. Using the neural patches and Markov Random fields described in Wand et al. we were able to create a loss function that impart the style of each specific section resulting in the best output.

Shifting to HyperFace from VGG enabled us to improve the final results of style extraction, and made the process of extracting the face from the painting and putting it back in completely automatic, by defining an oval region around the outer landmarks.

Conclusions

To conclude, we achieved our original aim to insert a person into a piece of art using facial replacement through convolutional neural networks. The process of completing the program, however, had very little dependence upon the architecture of the neural network. Rather, the application of the network—specifically with regard to the loss function—played the greatest role in shaping our results. This was rather surprising on first inspection; however, throughout the project it became clear that the highly trained VGG network was extremely flexible due to the abstract nature of its feature detection. Considering these observations, we would approach this project in a different manner. Foremost, we would discard the Gatys loss

function from the beginning as it does not fit our use case. Furthermore, we would retrain the facial landmark detector to consider non-human faces such as the Scream from Edvard Munch's famous painting. Overall, we are pleased with the quality of the results and hope that this project demonstrates the wide functionality of a well trained convolutional neural network.

References

Gatys, Leon, Alexander Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style." *Journal of Vision* 16.12 (2016): 326. Web. 8 Dec. 2016.

Lecun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning." *Nature* 521.7553 (2015): 436-44. Web. 9 Dec. 2016.

Ranjan, Rajeev , Vishal Patel, and Rama Chellappa. "HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition." *CoRR* 1603.01249 (2016). n pag. Web. 11 Dec. 2016.

Simonyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *Presented as a conference paper at ICLR, San Diego, 2015* (2014). Web. 8 Dec. 2016.

Wand, Michael, and Chuan Li. "Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis." *CoRR* 1601.04589 (2016). n. pag. Web. 11 Dec. 2016.

Zhu, Xiangxin, and Deva Ramanan. "Face detection, pose estimation, and landmark localization in the wild." *Presented as a conference paper at IEEE Conference on Computer Vision and Pattern Recognition* (2012). 2879–2886. Web. 11 Dec. 2016.

Zhu, Xiangxin, and Deva Ramanan. "FaceDPL: Detection, pose estimation, and landmark localization in the wild." *Dept. of Computer Science, University of California, Irvine* (2015). Web. 11 Dec. 2016.