CS 726 Project Report: Painting-Image Fusion using Deep CNNs

Rishabh Shah(150050006)
Huzefa Chasmai(15D170013)
Manas Bhargava(150050057)
Devansh Shah (150070004)

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

May 3, 2019

Abstract

Photo-realistic fusion of an image into a painting is a challenging task. Just merging image with the background of style yields to images that appear fake or look like collages. We need the image to retain its content while simultaneously adapt to the painting style. Most of the existing painterly stylization algorithms operate on a global painting level and perform poorly when applied to this local image. We address this issue by using local statistics that need to be transferred to the image to make the fusion seamless. Our algorithm captures the styles of the painting, including strokes, color and the flow of edges and transfers these to an arbitrary image so that one cannot figure out that this image was not a part of the painting in the fused painting. Consequently, our implementation produces significantly better results than photo compositing or global stylization.

The following is a detailed report of our project done as part of the course project for the course CS 726 - Advanced Machine Learning. The link to our code for the project is the following : Painting-Image-Fusion

Contents

(a) (b)

Figure 1: a : Original Naive Image, b : Our Result

1 Introduction

Image compositing is a one of the essential tools to create new visual content. There are many applications for this method. In some scenarios like photo collages, visible funky boundaries are desirable but many applications have the objective to make fusion seamless. For instance, to add an object into a photograph in a way that makes it look like the object was already present in the original scene. For fusion of images into paintings, the options are limited. One can manually paint the object in a particular style which would require both time and skill possessed by few. Hence automation of this task using Machine Learning principles seemed unequivocal. A lot of work is done in the domain of fusion of images into photographs, but these don't take into account the brush strokes, texture and abstraction typical of paintings. Since we seek a local blend of texture and color, existing painterly stylization algorithms

produce sub-par results for our application.

2 Related Literature

Image harmonization: Compositing is the process of adding the foreground image from one image and combined with the background image of another. The appearances of the foreground image is made compatible to the background appearance. There are many techniques which improve this realism by transferring statistics of handcrafted features between the foreground and the background. [Tsa+17] proposed an end to end deep CNN for image harmonization, which capture both the context and semantic information of the composite image during harmonization. These techniques are mostly designed for cases where the background is a photograph whereas our focus is on cases where the background are paintings.

Style Transfer using Neural Networks: Recent work on Neural Style transfer [GEB16] has shown impressive results on transferring styles by matching the correlations between feature activation extracted from a pretrained deep neural network on image classification (i.e., VGG network [SZ15]). Then reconstruction process is based on optimizing metrics like style and content loss. Much of work has been done to further improved upon the works of . Such advancements include making the transfer operation real-time, using other kinds of reconstruction losses like histogram loss and variation loss to make the style transfer photo-realistic. All these methods have in common that they change the style of entire images at once. Our work differs in that we focus on local statistics matching which in turn improves the aesthetics of a fused image.

3 Main Goals

Our main goal is perform seamless fusion of an arbitrary image into an existing painting so that someone looking at the fused painting cannot figure out which part of it is added. To achieve this goal we build upon the work of [GEB16]. We refer and extend the ideas from [Lua+18] to come up with techniques to get local statistics and transfer them to the input image.

4 Theory and Terminology

4.1 Content Representation

When CNNs are trained on object recognition, they develop a representation of the image that makes object information increasingly explicit along the processing hierarchy. Higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the reconstruction very much. In contrast, reconstructions from the lower layers simply reproduce the exact pixel values of the original image. We therefore refer to the feature responses in higher layers of the network as the content representation

4.2 Style Representation

The style representation can be built on top of the filter responses in any layer of the network. It consists of the correlations between the different filter responses. These correlations amongst all responses in a given layer represent a multi-scale representation of the input image, which capture its texture information but not the global arrangement. These feature correlations are given by the Gram matrix $G^l \in R^{N_l N_l}$, where G^l_{ij} is the inner product between the vectorised feature maps i and j in layer l:

$$G^l_{ij} = \sum_k F^l_{ik}.F^l_{ij}$$

4.3 Reconstruction Losses

A layer with N_l distinct filters has N_l feature maps each of size M_l , where M_l is the height times the width of the feature map. Let P and X be the original image and the image that is generated, and P^l and F^l their respective feature representation in layer l. A_l and G_l their respective style representation of P and X in layer l.

4.3.1 Content Loss :

There are a couple of metrics we used here :

- Mean squared error

$$L_{content}(P, X, l) = \frac{1}{2}\sum_{i,j}(F^l_{ij} - P^l_{ij})^2$$

- Normalized Cross Correlation Loss

$$L_{content}(P, X, l) = -1.0 * \sum_{i,j} \hat{F}^l_{ij}.\hat{P}^l_{ij}$$

where $\hat{F}$ and $\hat{P}$ are normalized

$$\hat{F}_{ij}^{l} = \frac{F_{ij}^{l} - mean(F_{ij}^{l})}{sd(F_{ij}^{l})} \quad \text{and} \quad \hat{P}_{ij}^{l} = \frac{P_{ij}^{l} - mean(P_{ij}^{l})}{sd(P_{ij}^{l})}$$

4.3.2 Style Loss :

Let the layers considered for the loss be L and $\{w_l\}$ be the weights assigned to the different layers l

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{ij} (G_{ij}^{l} - A_{ij}^{l})^2$$

$$L_{style}(P, X) = \sum_{l=0}^{L} w_l E_l$$

4.3.3 Total Variation Loss :

$$L_{total\_variation} = \sum_{x,y} ((X_{i,j} - X_{i-1,j})^2 + (X_{i,j} - X_{i-1,j})^2)$$

This loss is introduced to ensure that the generated image is smooth which is the case which most natural paintings.

4.4 Terminology

From now on, we refer to unmodified painting image as `style_image` and input image copy pasted onto painted as `content_image`. The image which we modify to get to the final desired fusion image is referred to as `fusion_image`.

5 Approaches

5.1 Global Neural Style Transfer

Initially, we implemented our baseline [GEB16] technique. In this approach, we pass `fusion_image`, `content_image` and `style_image` through a standard trained VGG 19. The activations obtained from `conv5_2` layer are used for calculation content loss and [conv1_1, conv2_1, conv3_1, conv4_1, conv5_1] for style loss. Then we optimize the loss calculated to update the `fusion_image` accordingly. The results obtained from this approach were not very good as we have used global style and content transfer and thus one can clearly identify the input image part in `fused_image`. Also, we could see some background part of *fusion_image* affected by the input image in terms of color spread and texture mismatches in background.

5.2 Local Neural Style Transfer

As a start we fix the flaws of previous approach by limiting the loss functions to masks. These are essentially binary images with 1s over pixels directly above the input part of `content_image` and 0s everywhere else. Now the content loss consists of L2 norm between masked features. Style loss now consists of using gram matrices constructed from masked activations. These improved the results by limiting the changes in fusion image only in mask part thus not affecting the rest of the background. But, since we are now using gram matrices that are limited to masked features, we are essentially trying to equate the style of input image to that of region exactly behind it. This may not capture the complete style of image and may not give good results in case of all painting styles. Also, using the mask creates some visible seams between input image and background which makes it stand out in the `fusion_image`

5.3 Patch based local statistics transfer

For the style representation issue in previous approach, we try apply ideas from [Lua+18] to capture most similar style to the input image. Basically, we try to map the nearest neighbour patch in `style_image` for every patch for every 3 X 3 patch in `input_image` for each of the layer in CNN. By doing this we try to ensure that the style of input_image is matched with closest style from style image. This would not only preserve the of input image's representation but would also be easier to optimize. Unfortunately, we were not able to completely implement this complicated approach due to underlying tensorflow framework used initially. To improve current result we tried to use both local and global gram matrices and adjust weights accordingly to get good results.

5.4 Seamless Fusion and Post Processing

For smoothening the transition between input image and background we introduce the variation loss that would minimize image gradients and ensure that fusion is seamless at boundaries. Also, instead of using masks only on the input image, we use dilated masks that cover 2-3 pixel layer more than original input image mask. This ensures that our algorithm tries to take into account nearby patches for content loss thus smoothening the transition. While calculating the losses we used another approach of using the normalized Cross Correlation Loss in addition to the MSE Loss. We experiment with both these losses and choose the best of them.

As a post processing step we use an additional spacial gaussian mask that further smoothens out the boundaries.

6 Implementation Details:

This section describes the implementation details of our approach.

6.1 Code Description :

Our project is present on code.

Our code spans around 600 lines. We have implemented the whole algorithm in the python environment using tensorflow and keras as our backend. We have used the tensorflow eager execution environment to debug the code.

6.2 Additional Tools used :

We use the publicly available VGG 19 Net architecture [SZ15]. We use the activations from the following layers of the above net both for image style and content representation. conv5_2 activations for content and [conv1_1, conv2_1, conv3_1, conv4_1, conv5_1] layer activations for the style representation. In reconstruction step, we used the standard scipi implementation of L-BFGS [Zhu+97] optimizer which takes CNN gradients and loss to update the *fusion_image*.

6.3 Code Evolution:

We started off with a keras tutorial on Style Transfer and modified the code to serve our painting fusion problem. Then onward we had implemented Approach1 i.e. the Global Neural Style Transfer before the Mid-Semester. We improved upon that code trying out the different approaches listed above.

We have built a framework which takes as an input the original painting image and the object image to be embedded as input producing the final fused image obtained from the optimizer. There are 3 hyper-parameters involved in our algorithm. These are the weights given to each of the loss functions, i.e. content loss, style loss and the total variation loss. We have tuned hyper-parameters manually for each test image.

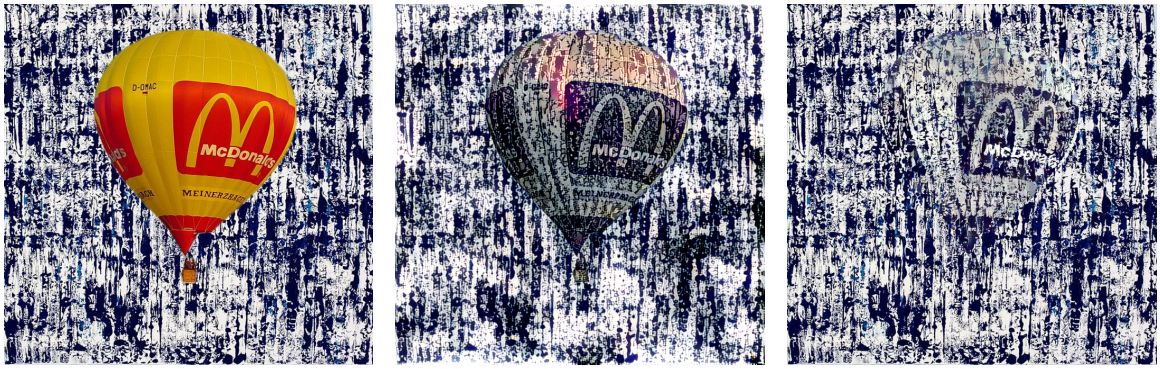
6.4 Experimental Platform

We have used a (GeForce GTX 1080 Ti GPU) enabled machine. Each of our experiment takes around 60 minutes to output the final optimized image when running for around 5000 iterations. We do not have any learning step involved but most of the processing goes in the optimization step.

7 Experiments and Results

We show our results across all the approaches tried on following 5 different images.

7.1 Results:



(a) (b) (c)

Figure 2: Results for naive (a), Baseline (b) and Final(c)

(a) (b) (c)

Figure 3: Results for naive (a), Baseline (b) and Final(c)

(a) (b) (c)

Figure 4: Results for naive (a), Baseline (b) and Final(c)

7.2 Observation:

As we can see from above results that even Global Neural Style Transfer techniques shows quite an improvement over naive copy paste composite method. Also, it can be clearly seen by comparing base and GNST image (Fig: 3) that during the reconstruction step background is affected in Einstein image. This effect has been nullified by introduction of mask in third approach and can be seen in all the final images. Since we are using local statistics for creation of fused image, it can be seen that those positions in the painting where style in masked region is either uniform or similar to input image, the results are visually more appealing.

8 Efforts

8.1 Time spent on different parts of the project

Roughly equal amount of time was spent on formulating, implementing and experimenting with all the approaches and then finally compiling them into this report. Initially a lot of time was spent on debugging the code as desirable results were not achieved. We tackled this by using the eager environment for debugging which itself took some time.

8.2 Challenges Faced

A lot of time was spent in debugging and getting to run the first approach. Through this we got familiar with the whole framework which in turn helped ease working on the other approaches. Coming up with other approaches that worked well was difficult. Experimentation with various ideas used in the papers took some time. Many of our approaches failed to provide good results. A major limitation was our inability to implement "for" loops for calculating the loss function. We tried to work around this problem by using multiple matrices for evaluating patch based approaches. But then again an issue of calculating argmax led to further complications.

8.3 Key Work Distribution

We equally distributed the research part for this project in terms of reading related literature and coming up with advancements and solutions to our problem statement. Coding and report making was done together as a team.

9 Future Work

We need to implement the technique of Patch based Local statistics transfer which seems promising but we couldn't implement because of complications in the coding framework. Another possible extension would be to the idea of real time style transfer where an image transformation network is trained to transform a input image into a desired output. This would result in faster optimization.

10 Acknowledgement

We would like to thank Prof. Sunita Sarawagi, CSE Department, Indian Institute of Technology Bombay for this opportunity and our mentor Rasna Goyal for their support.

References

[Zhu+97] Ciyou Zhu et al. "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization." In: *ACM Trans. Math. Softw.* 23.4 (1997), pp. 550–560. URL: http://dblp.uni-trier.de/db/journals/toms/toms23.html#ZhuBLN97.

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2015).

[GEB16] L. A. Gatys, A. S. Ecker, and M. Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 2414–2423. DOI: 10.1109/CVPR.2016.265.

[Tsa+17] Yi-Hsuan Tsai et al. "Deep Image Harmonization". In: *CoRR* abs/1703.00069 (2017). arXiv: 1703.00069. URL: http://arxiv.org/abs/1703.00069.

[Lua+18] Fujun Luan et al. "Deep Painterly Harmonization". In: *Comput. Graph. Forum* 37 (2018), pp. 95–106.