UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

A report on the

Neuron Segmentation using Deep Style Transfer

Final Project

Heidelberg Collaboratory
for Image Processing

November 4, 2018

Submitted to:

PD Dr. Ulrich Köthe & M. Sc. Jakob Kruse

Submitted by:

Ahmad Neishabouri
(AN) Student ID: 3436580
M.Sc. Scientific Computing

Contents

1 Introduction

Back in the days, I had a friend who was really particular in his clothing taste, and was insisting on wearing a specific shirt the whole school year. However, one time I was outside the school in a restaurant with my family and he approached me to say hey, and I barely recognized him! Yes you guessed it, because he was wearing another shirt!

You might be thinking now what this has to do with Neuron Segmentation, but bare with me, I'll explain. When I was reading about Deep Style Transfer, and the ability to generate images with different *styles*, I draw this analogy that these styles can represent our clothing styles, and since we are most certainly the same person in each type/style of cloth, our friends will recognize us regardless of our clothing. However in my case, I was over-fitted with the image data of my friend with that lousy shirt, and when he changed his cloths, I couldn't classify him as one of the people that I know. This made me thinking that maybe if we are dealing with a problem of classification of any kind, maybe using different styles of image can act as a data augmentation method, which result in a bigger dataset and a more **robust** classifier.

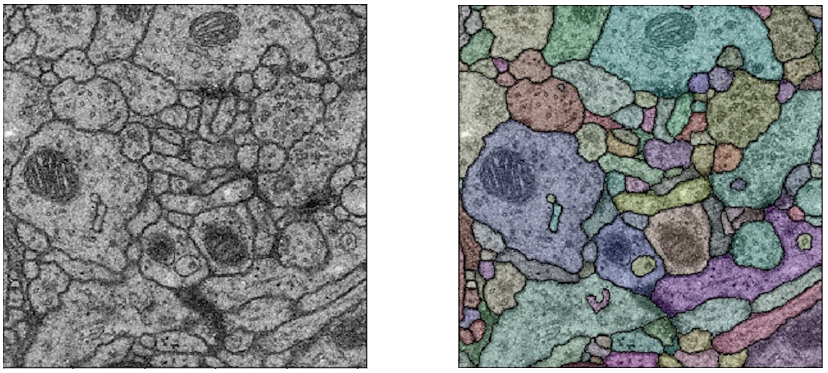
Style Transfer. As it first has been introduced by (1), it was pretty astonishing what these neural networks are capable to produce, however Style Transfer has been fully introduced ever since and there exist many repositories which have ready to execute code that you can download and try it your self. You can even do it yourself online (2) (based on paper (3))with out any computation required, and get a Picasso-styled profile picture within few minutes. Moreover, as Prof. Köthe has mentioned before, a group of students in this course has fully described this method in a well prepared report couple of years ago. So I didn't want to do the trivial work of repeating all of these and I came up with the idea of using this method for data augmentation.

This idea first came to me when I saw the style transfer example shown in figure 1. Here we can see clearly how edges, shapes, and the whole sketch of the image has been changed to a new style, which is as it'll be mentioned in section 2, an unfaithful style transfer. This made me think if we apply an unfaithful style transfer to the images of a data set to both the raw image, and the ground truth of the image, we will yield a more divergent data set to train, which would yield in a more robust style transfer. However, this wasn't the result and as I was afraid, the algorithm for segmentation, the random forest classifier, didn't see this images as a new, *informative* images and the output did not change that much comparing to the original, unaugmented data set.

The Problem. The problem of classification that I chose to work on is the ISBI 2012 challenge(5) for segmenting the neurons in the Drosophila larva ventral nerve cord images. The dataset contains 30 training images and 30 testing images. I have worked on this data set previously for another course's final project. The work has been done before on a followup to Beier's (6) work on this challenge and we used their implementation as a reference for applying the auto-context algorithm (7). In this work however, I tweaked the code in a way that it reads our result from the style transfer code and augment the data set accordingly.

Figure 1: An example of style transfer. Image Credit (4)

Figure 2 shows a sample of the image to be segmented and it's corresponding ground truth segmentation.



(a) Raw Image (b) Ground Truth

Figure 2: An example of ISBI challenge Neuron Segmentation Problem

Hardware. Fortunately I could lay my hand on a rusty, however relatively powerfull GTX970, and it did the work for me. However, I should add that as it can be seen in my code, my implementation is using a pretrained VGG network and this is for the following two reasons: First, in most of publications in Style Transfer, VGG network has sort of served as a benchmark so that the research community in this field could compare their result to each other. Second, training a network, requires

a descent hardware, setting aside all the hyper parameter tuning that can appear in a practical implementation(which actually did appear for me because I first started with training my own implementation and realized this can't be done in the time frame of a mini project).

Structure. The structure of this report is as follows: Section 2 will have a short introduction to Deep Style Transfer and how does it generates the artistic style images along with some beautiful results from our beloved Heidelberg city, then I'll shortly introduce the Multicut method to solve Neuron Segmentation problem. Section 3 will show the results of style transfer on the training set and Finally we'll have the result of inference on the augmented dataset and compare that with the original dateset inference result.

2 Theoretical Background

Machine Learning methods, and specifically Convolutional Neural Networks (CNNs) are showing promising results in a wide range of computer vision tasks, and are being used in different sectors. However, these methods works efficiently and with their best performance when they are fed with large-scale annotated data, a requirement that is not always easy to obtain. This is because preparing and labeling datasets are costly and there are ongoing challenges(8) which are meant to solve and automate this problem. On the other hand, there is also the data privacy which makes this task even more complicated, in which in sectors such Medical Image Analysis, normally patient data are secured by medical privacy laws which prevent any usage of the patients data in any open scientific tasks, and preparing dataset is pretty expensive and requires a collaboration of researchers and scientists(9).

Data Augmentation methods are the solution for making synthetic data out of a small data set. There are different basic methods in augmenting data such as, flipping, rotating, scaling, translation, and so on; even adding random noise is considered as a form of data augmentation which usually result in reducing overfitting and improving the robustness of the system(10). In this project however, it's intended to augment the data set with *styled* images of the data set.

2.1 Artistic Style Transfer

As it was mentioned earlier in the introduction section, there have recently been a lot of interest in the development of style transfer with the help of deep neural network. Based upon the innovative idea of Gatys et al, there have been two major stream lines of the developments, first being the development addressing the inefficiency of the computational process(11), and the capability of the algorithm to perform these artistic style transfers on videos. There were progress with utilizing the method of *learning equivalent feed-forward generator networks* which can generate the stylized image in a single pass(12; 13). The second stream of the development was working on the so called *photorealism* of the content image. This line of work concerns transferring the style of a reference photo to a content photo while maintaining faithfulness to the content image as much as possible(14). For instance one can turn a photo of a city with skyscrapers full of windows in the daylight, to the exact same photo(faithfulness) but this time late in the night(figure 3).

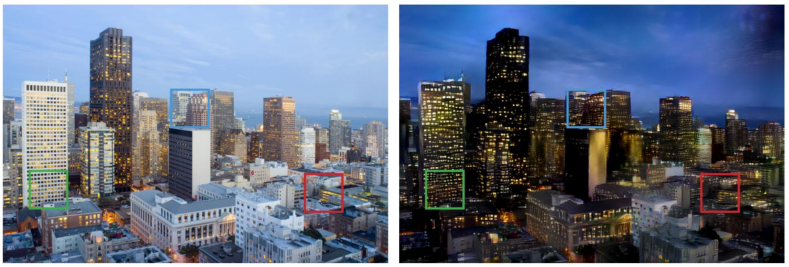


Figure 3: A photorealistic style transfer example. Image Credit (15)

In this work however, it is not intended to have a faithful style transfer, rather it's motivated to have unfaithful one, just like what Gatys et al. has published, paintings like with straight edges turns to wiggly and regular textures wavy(15). This is because I thought the more unfaithful the transfer is, the more "informative" the augmented data would be. This is why the implemented style transfer network is based on a Pytorch implementation of Gatys et al. work.

The aim of their work was to generate a third image x which has the content of image p and the style of image a. This has been done by an energy minimization problem in which encoded the content loss and the style loss in it in the same time. The innovation in their work was to figure out that features extracted by a convolutional network carry information about the content of the image, while the correlations of these features encode the style(16). Denoting P^l, S^l, and F^l as the feature maps extracted by the network from the original image p, the style image a, and the stylized image x we can formulate the content loss as equation 1, in which $N_l \times M_l$ denotes the dimensionality of these feature maps, N_l being the number of filters used, and M_l being the spatial dimensionality of these feature maps, i.e the product of its width and height.

$$\mathcal{L}_{\text{content}}(\boldsymbol{p}, \boldsymbol{x}) = \sum_{l \in L_{\text{content}}} \frac{1}{N_l M_l} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2 \tag{1}$$

This content loss, being the mean squared error between P^l and F^l doesn't need to be restricted to one layer, and as the equation 1 shows, it's the sum of the mean squared error in all layers(set L_{content}) that is being used for content representation.

Coming to style loss, as we mentioned earlier, it's the mean squared error between *correlation* of the filter responses, namely Gram matrices $A^l \in \mathbb{R}^{N_l \times N_l}$ for the style image a, and $G^l \in \mathbb{R}^{N_l \times N_l}$ for stylized image x in which

$$A_{ij}^l = \sum_{k=1}^{M_l} S_{ik}^l S_{jk}^l \tag{2}$$

$$G_{ij}^l = \sum_{k=1}^{M_l} F_{ik}^l F_{jk}^l \tag{3}$$

and by having as in equation 1 set L_{style} as the set of layers used to represent the style, the style loss would be

$$\mathcal{L}_{\text{style}}(\boldsymbol{a}, \boldsymbol{x}) = \sum_{l \in L_{\text{style}}} \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2 \tag{4}$$

and having them all together the loss function will be:

$$\mathcal{L}_{\text{ singleimage }}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha \mathcal{L}_{\text{ content }}(\boldsymbol{p}, \boldsymbol{x}) + \beta \mathcal{L}_{\text{style}}(\boldsymbol{a}, \boldsymbol{x}) \tag{5}$$

in which α and β are measures of importance for each component.

The image x is generated by minimizing equation 5 by a gradient-based optimization method. This is a straight forward description of Gatys method used in this project. The implementation can be found in the Github repository. Below you can see the output of this implementation with some beautiful photos of Heidelberg city, along with their style, and content itself.

Figure 4: Stylized images using the VGG16 implementation
Left: Content Image, Center: Style Image, Right: Stylized Image

Figure 5: Stylized images using the VGG16 implementation
Left: Content Image, Center: Style Image, Right: Stylized Image

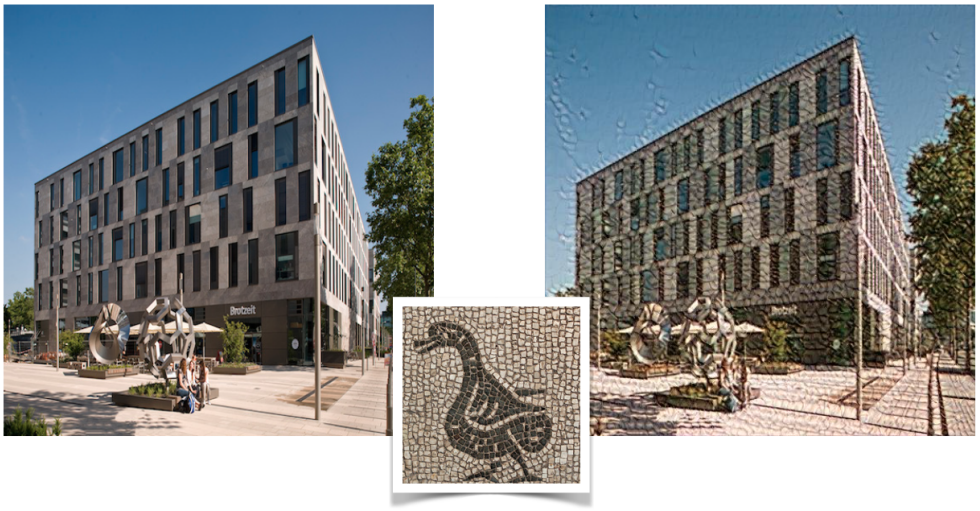


Figure 6: Stylized images using the VGG16 implementation
Left: Content Image, Center: Style Image, Right: Stylized Image

2.2 Minimum Cost Multicut Problem

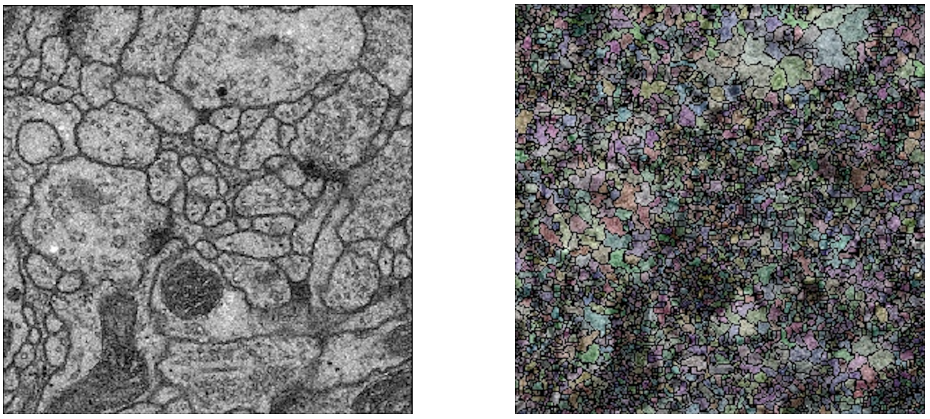
Beier et al. (6) has discussed the possibility of formulating an image segmentation problem as a minimum cost multicut problem on a designated graph. A short description of the method along with some definitions will be introduced here, for a detailed description refer to the original paper or to (17; 18). Applying a basic segmentation technique (e.g. watershed segmentation) on an image will result in a segmented image in which each segmentation contains a group of pixels with a very similar neighbouring characteristics. However as it is usually the case, applying these methods on our raw image of the training set(figure 2a) will result in the so called over segmentation, which means the actual segmentation that we are looking for will be segmented in many other small segmentation. In a minimum cost multicut problem, we try to assign each of these super pixels to a *node* in our graph, and connect each of these nodes to the neighbouring nodes(super pixels) with an *edge*. This way we will wind up with a graph designated to the raw image and we'll try to apply our learning problem on this graphs in a sense that a probability of how probable each edge to be "true" is calculated and learned, and whenever this probabilty is lower than a specific threshhold, the edge will be "cut" and the two super pixels would merge in one bigger super pixel.

The final/optimal solution to the problem is the solution to the optimization problem whose feasible solution is the proper decomposition of the graph. In a mathematical formulation, defining each graph as $G = (V, E)$ in which V denotes the set of nodes, E denotes the set of edges, and every $c : E \to \mathbb{R}$, the optimization problem designated for the minimum cost multicut problem would be

$$\begin{aligned} &\min_{x\in\{0,1\}^E} && \sum_{e\in E} c_e x_e \\ &\text{subject to} && \forall Y \in \text{cycles}(G) \forall e \in Y : x_e \leq \sum_{e'\in Y\setminus\{e\}} x_{e'} \end{aligned} \tag{6}$$

3 Experiments & Discussion

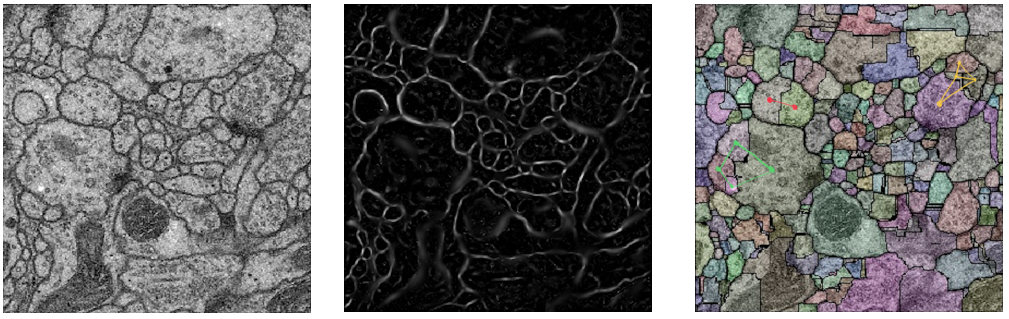
As it is discussed in section 2.2, first step to convert an image to a graph is to apply some sort of basic segmentation method. In order to do that, the implementation uses the NIFTY software (19) in which it first uses the watershed segmentation method and turns the raw image to an over-segmented image:



(a) Raw Image (b) Oversegmented Image

Figure 7: Watershed Segmentation Result

However, this results in a really big graph with thousands of edges, and applying this to our whole data set would result in a really huge training set of millions of instances. In order to avoid that, a *Frangi* filter (20) has been applied on the images in advance, so that it results in a bigger segmented super pixels.



(a) Raw Image (b) Frangi Filtered Image (c) Oversegmented Image

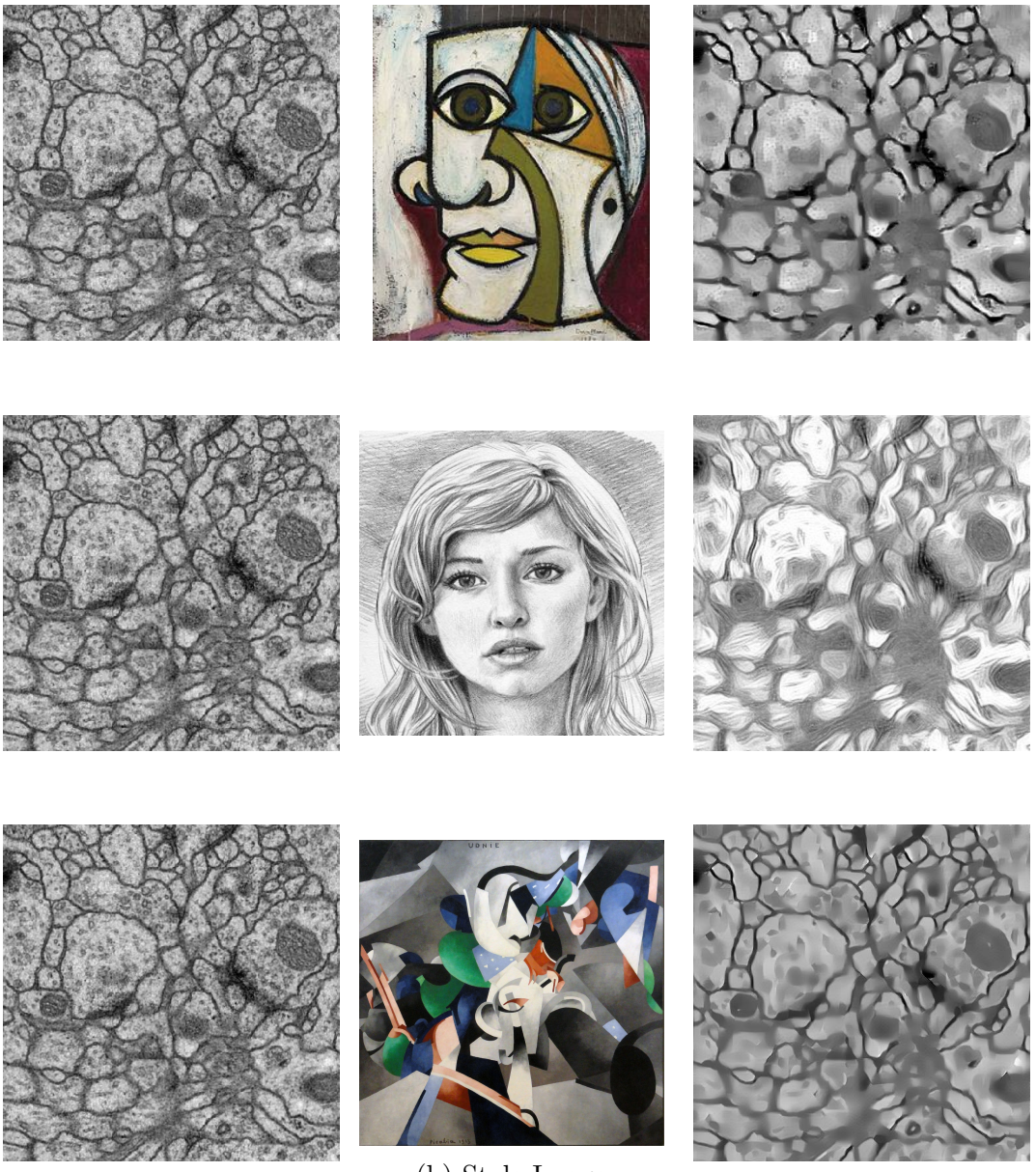
Figure 8: Watershed segmentation result with Frangi filter applied

Keeping in mind figure 2b for the ground truth image, the objective is learning to *cut* the edges in the over-segmented image(figure 8c) which does not have a high probability of being an edge. Some of this edges have been drawn on figure 8c with blue, green and yellow hand sketches.

3.1 Style Transfer on Raw Images

Right out of the bat, after I've seen the results of style transfer on the neuron images, I sort of realized this wouldn't introduce *informative* data to the training set. Reason being is the neuron edges did not move that much that I was expecting, therefore

the whole style transfer acted as a filter to the image. For this reason, I tried to apply many different style images on the raw data, with different optimization step numbers, hoping that I could end up with more informative images, however with no avail. Below you can see some of the results of these style transfers:



(a) Raw Image (b) Style Image (c) Stylized Image

Figure 11: Artistic Style Transfer on neuron images

The whole dataset has went through style transfer on 24 style images (21) with two different optimization number of steps (75, 150) and can be seen in the project Github repository. The smoothness achieved by transferring the style of artistic paintings to the raw images made me think that transferring images with continuous textures would act as a sort of filter to the image, this way we might use these styled transferred images with out applying Frangi filter on them. However the result wasn't satisfying either and can be seen for different styles (22) on the Github repository.

3.2 Result of Data Augmentation

In this section, various scenarios has been tested in an attempt to seek better results than the original inference. However, first we'll see how the result of the segmentation before augmenting the dataset is. As being mentioned earlier, the result with oversegmented(without applying Frangi filter) is not satisfying,

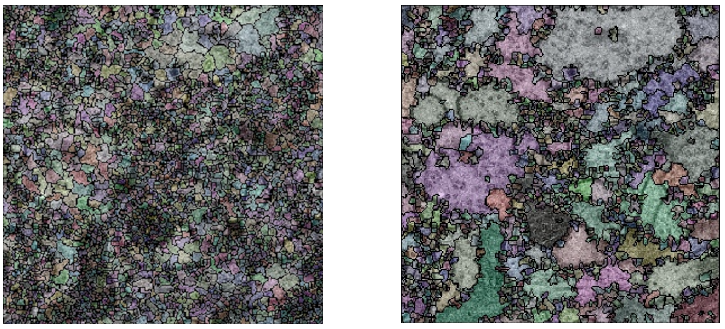
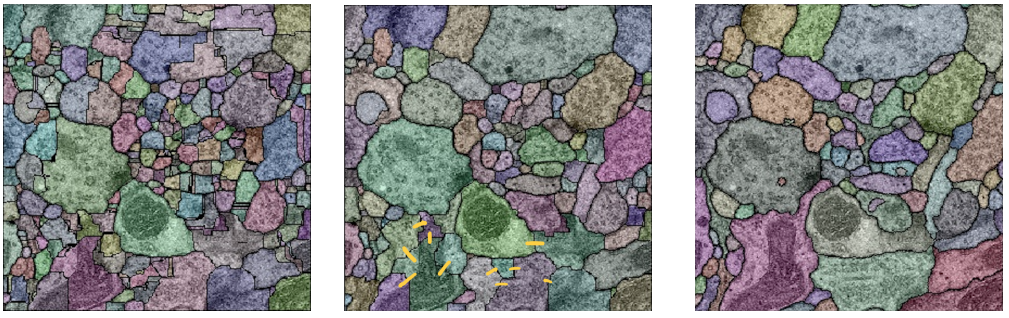


Figure 12: Inference on oversegmented(without Frangi) image

and we applied Frangi filter to obtain the oversegmented image and applying the inference with 24 images as training set we will obtain:

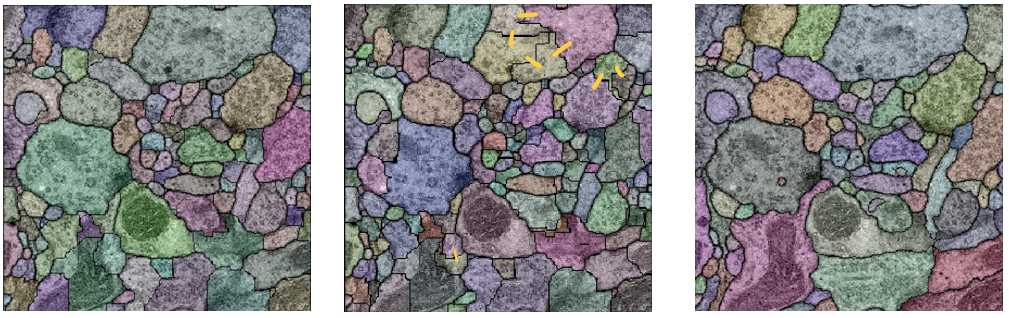


(a) Oversegmented Image (b) Inference Image (c) Ground Truth Image

Figure 13: Inference result with the original dataset

As it is shown in figure 13b, the inference is doing a pretty good job in general, however there are some edges(e.g. edges marked in yellow on figure 13b) that needs to be cut and I was hoping I could improve the inference with the data augmentation.

My first attempt was of course augmenting all the stylized images to the main data set, this way we obtain 600 images from 24 images (including the original image itself). Below you can see the result of this inference comparing to the original inference.

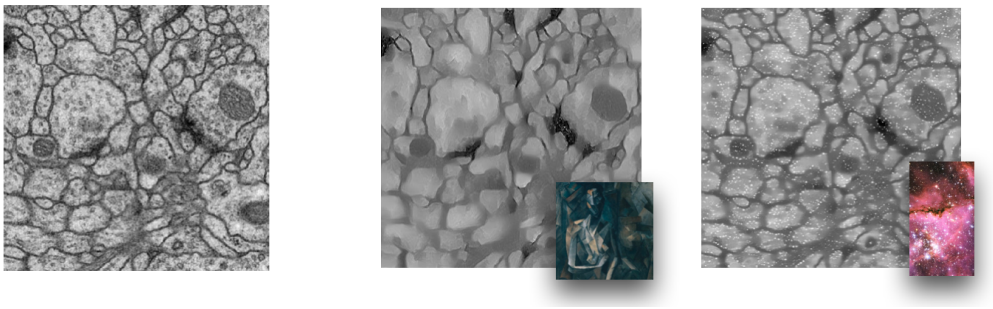


(a) Original Inference (b) Augmented inference (c) Ground Truth Image

Figure 14: Inference result with the augmented dataset

As it can be seen in figure 14b, not only the inference did not improve in the lower areas of the image comparing to the original inference, but also there are other edges that need to be cut in the upper part of the image(marked with yellow marker on figure 14b).

As it can be seen from the stylized neuron images in Github repository, some of the styles aren't that much good and result in a washed out neuron images, examples of these are brought here with their style image along:

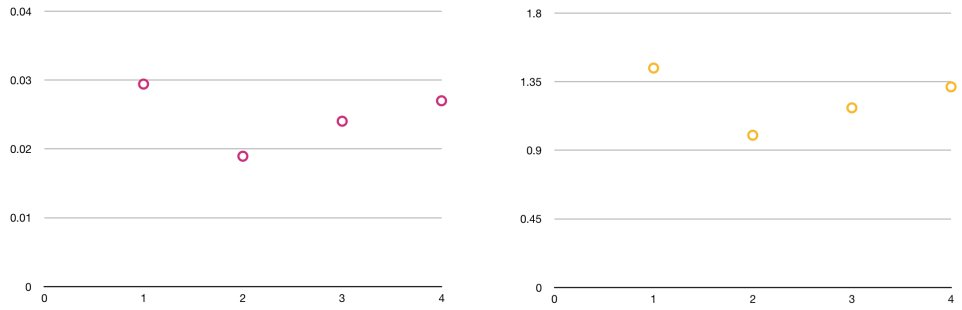


(a) Raw image (b) Stylized images

Figure 15: Examples of stylized images that went wrong.

So this made me think to apply the inference with selected stylized neuron images, and also with a specific stylized image, and combination of original image and a specific stylized image. Results wasn't satisfying in either cases.

Up until now we showed the result with visualizing it and seeing how the inference was working, however, in order to have a complete evaluation of the results, two testing criteria are being utilized, Random Error and Variation of Information Error which both are implemented in NIFTY software library. The above mentioned test scenario's along with the error measurements are plotted in figure 16. Unfortunately, as the figure shows, I couldn't get any improvement and using the original dataset has the lowest error comparing to other implemented scenarios. Among the implemented scenarios, using the original dataset plus one specific stylized image works the best which makes sense because the random forest probably still using the features from the original dataset. As it was expected, using a selected stylized images work better that using all the dataset together.



(a) Random Error (b) Variation Of Information Error

Figure 16: Different Scenarios Error Measurements; 1. Full Augmented Dataset, 2. Original Dataset, 3. Original Dataset + Specific Stylized Image, 4. Selected Stylized Images

4 Summary

An attempt to improve the inference problem on the ISBI Challenge has been done using Deep Style Transfer. The focus was some how to improve the results by augmenting the dataset with other styles images. Unfortunately no clear improvement has been obtained. The reason behind this might be mostly because the style transfer does not really introduce *informative* images to the Random Forest Classifier. Moreover, some undetectable changes might occured in specific areas of the data set when the style transfer implemented and this might resulted in miss alignment with the ground truth images.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," tech. rep., 2015.

[2] D. UG. `https://deepart.io`.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[4] L. Engstrom, "Fast style transfer." `https://github.com/lengstrom/fast-style-transfer/`, 2016.

[5] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamentsky, R. Burget, V. Uher, X. Tan, C. Sun, T. D. Pham, E. Bas, M. G. Uzunbas, A. Cardona, J. Schindelin, and H. S. Seung, "Crowdsourcing the creation of image segmentation algorithms for connectomics," *Frontiers in Neuroanatomy*, vol. 9, p. 142, 2015.

[6] *An Efficient Fusion Move Algorithm for the Minimum Cost Lifted Multicut Problem*, vol. LNCS 9906, Springer, 2016.

[7] M. R. Ahmad Neishabouri, "Mlcv2017." https://github.com/ahmadnish/MLCV2017/, 2017.

[8] "Imagenet object detection challenge," 2018.

[9] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Gan-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *CoRR*, vol. abs/1803.01229, 2018.

[10] A. Asperti and C. Mastronardo, "The effectiveness of data augmentation for detection of gastrointestinal diseases from endoscopical images," *CoRR*, vol. abs/1712.03689, 2017.

[11] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016.

[12] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016.

[13] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," *CoRR*, vol. abs/1603.03417, 2016.

[14] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A Closed-form Solution to Photorealistic Image Stylization," tech. rep.

[15] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," tech. rep., 2017.

[16] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," tech. rep.

[17] M. Grötschel and Y. Wakabayashi, "A cutting plane algorithm for a clustering problem," *Mathematical Programming*, vol. 45, pp. 59–96, Aug 1989.

[18] S. Chopra and M. R. Rao, "The partition problem," *Mathematical Programming*, vol. 59, pp. 87–115, Mar 1993.

[19] "Nifty 0.19.0 documentation." http://derthorsten.github.io/nifty/docs/python/html/index.html, 2017. [Online; accessed 15-August-2017].

[20] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. a. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, June 2014.

[21] H. Zhang and K. Dana, "Multi-style generative network for real-time transfer," *arXiv preprint arXiv:1703.06953*, 2017.

[22] "Unsplash - By different photographers." https://unsplash.com/search/photos/texture, 2018.