# An Artificial system to create Artistic Style images based on Deep Neural Networks

Kun Chen
Columbia University
New York, NY
Email: kc2980@columbia.edu

Jingyu Qian
Columbia University
New York, NY
Email: jq2250@columbia.edu

Hengzhi Ding
Columbia University
New York, NY
Email:hd2358@columbia.edu

***Abstract*—In the realm of art, humans seem to have special skills and talent in creating fine paintings and masterpieces. However, with the help of the biologically inspired vision models called Deep Neural Networks, computer is also capable of building artificial systems to create artistic images of high perceptual quality. The key concept is to separate and recombine content and style representations of two images. A famous architecture of Convolutional Neural Networks called VGG is used in this paper to implement the system and the result turns out to be good.**

***Keywords*—Artistic Style images, Deep Neural Networks, Convolutional Neural Networks, VGG**

## I. Introduction

### A. Motivation and Background

In the realm of art, humans seem to have special skills and talent in creating fine paintings and masterpieces. [1] For a long time, computers seem to have no capability to compete with humans in this field . With the development of deep neuron networks, however, people have achieved great success in computer vision and similar fields. Now people are wondering if an artificial system based on deep neuron networks technique can be able to create images of high perceptual quality. Few articles have tried to use different methods to achieve this[2][3][4] and the results have been proven to be quite good.

The main technique used to solve this problem is Convolutional Neuron Network(CNN) which is a very good architecture used for image processing. Similar to other type of neuron networks, CNN is a feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex.[5] It uses the idea of convolution in signal processing in each layer to make it very useful in edge detection in image processing. It also enjoys the properties of sparse interactions, parameter sharing and equivariant representations. So far, CNN is very popular in doing classification in image recognition, video analysis, natural language processing, drug discovery and many other areas.

CNN is also used here in our problem because of the following properties it shares[4]:

1) CNNs used for classification can be re-purposed to extract style features (e.g. textures, grain, strokes), thus helping us to get the style representation of each figure.
2) Higher-level layers achieved by CNN contain the details of the image, thus helping us to get the content representation of each figure.

By combining the content and style representation of two different images, the artificial neural network system can actually create a meaningful image as an output.

## II. Summary of the Original Paper

- Paper Review I

### A. Methodology of the Original Paper I

In this section, we are doing a breif review of the paper "A Neural Algorithm of Artistic Style"[2]. In this paper, the author discussed about the creation of artitstic style image and used convolutional neuron network to implement the articifial system.

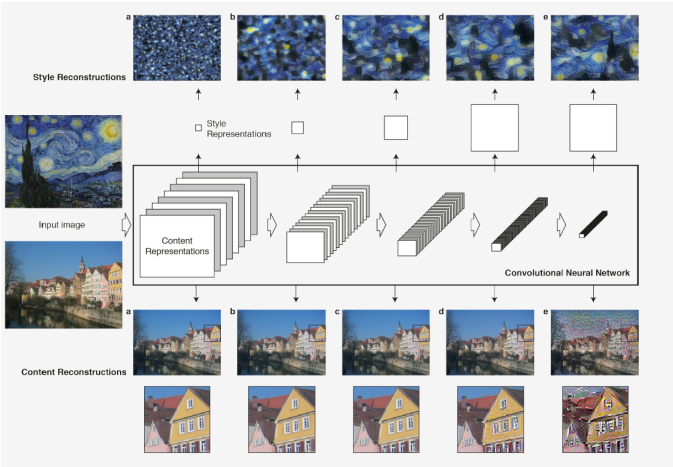


Fig. 1. Convolutional Neural Network

For an arbitrary image, we can view it from two different perspectives, its content and its style. One of the key findings of this paper is that the representations of content and style of an image are separable in CNN.

If an image is given as an input of a CNN, the outputs of the CNN contains both content and style representation of the image. Along the processing hierarchy of the network, CNN develops an increasingly explicit representation of the image, which means the input image is transformed into

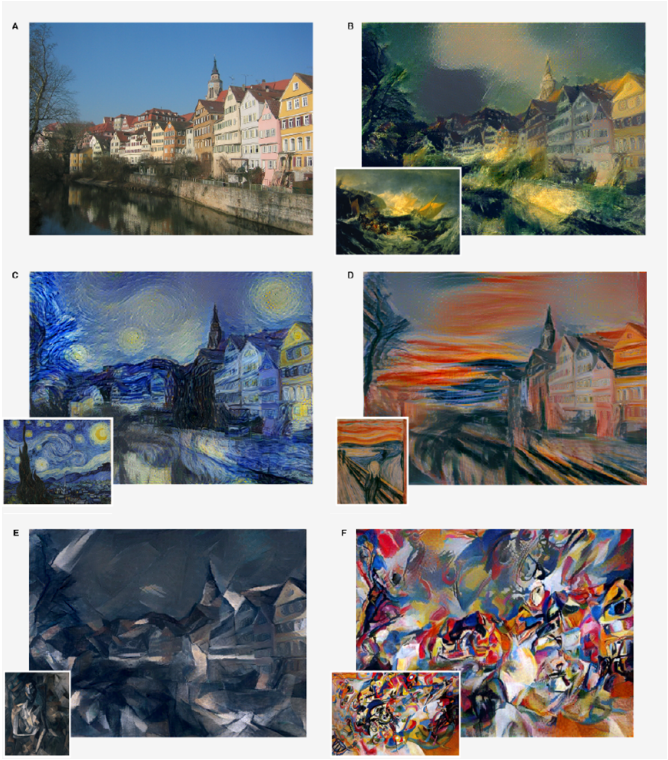

Fig. 2. Images that combine the content of a photograph with the style of several well-known artworks

representations that increasingly cares about the actual content of the image. Thus, the higher layers in the network captures the content representation of the image while the lower layers capture the style representation of the image. This can be shown in Fig.1.

By combining the content representation of one image with the style representation of another, we can create a new image of artistic style. Several examples can be seen to show this idea in Fig.2. The content image is a photo took by a river. The style image is chosen to be some famous paintings which we believe have good style.

In practice, it is usually not possible that we can simply combine the style representation and artistic representation to make a good picture. The procedure is usually done by defining two loss functions for both content and style representation. By giving them different weights and minimizing the total loss function, we can get the final output.

### B. Key Results of the Original Paper I

Result of the paper can be shown in Fig.3 . Fig 3 shows results for different relative weightings $\alpha/\beta$ of the content and style reconstruction loss and for matching the style representations only on layer conv1_1 (A), conv1_1 and conv2_1 (B), conv1_1, conv2_1 and conv3_1 (C), conv1_1, conv2_1, conv3_1 and conv4_1 (D), conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1 (E). Each column corresponds to a ratio of $\alpha/\beta$ and each row corresponds to a choice of representation layer.

Fig. 3. Key Results of the Original Paper I

- Paper Review II

### C. Methodology of the Original Paper II

In this section we provide a brief review of the paper "Semantic Style Transfer and Turning Two-Bit Doodle into Fine Artwork"[2], which gives us the core idea of our work. The paper introduces a new concept of neural network augmentation in order to let it incorporate semantic annotations. Such augmentation will let the network more content aware, and therefore can be used to automatically generate amazing artwork from mere doodles.

The work is based on a patch-based implementation in 2016[4]. Current style transfer neural networks lacks content awareness, and is likely to produce unexpected outcome when style and content are not mixed as well as the users want. The author address this shortcoming by pointing out 2 causes:

1) Convolutional neural networks are not specifically designed for image synthesis;
2) Backpropagation is not enough for high-low information transfer, and therefore needs augmentation of the network;

The pixel labeling neural network could produce labels of each part of an image, which can possibly be exploited by image synthesis neural network, and including semantic information of the image could prevent unwanted components in the output.

### D. Key Results of the Original Paper II

The proposed semantic information augmented network can be used both for image synthesis as well as style transfer. The following output were excerpted from the original paper.

In Fig. 4, the style image is a real painting, with manually

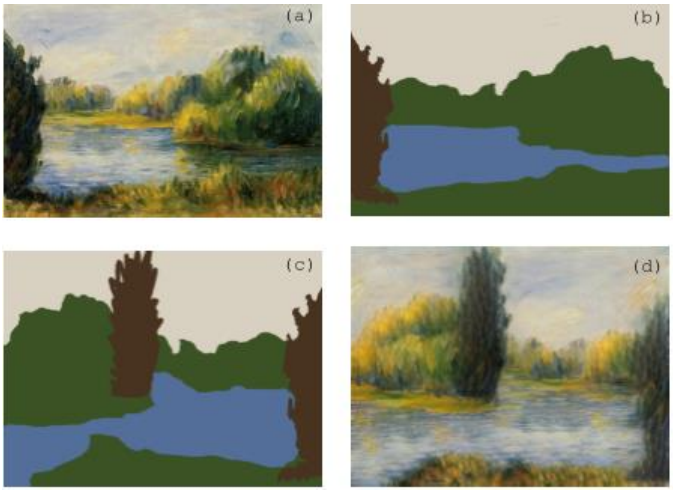


Fig. 4. Trained network for image synthesis(Alex. 2016). Top row is the real painting with its semantic annotation. Bottom row is the doodle as semantic annotation, with the network's output

made semantic annotation. The network succeeded in generating another semantic annotation into fine art work of the same style. In Fig. 5, the network managed to do the style transfer

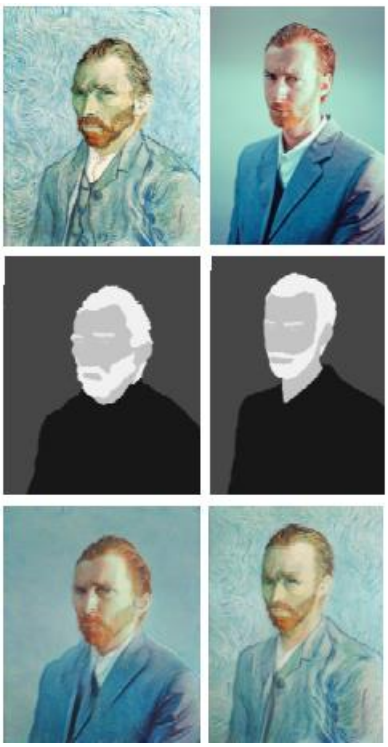

Fig. 5. Trained network for image style transfer(Alex. 2016). A painting of Van Gogh and a man's photo were used as input together with both semantic annotations. The annotations were carefully generated in the case two images are similar. The network turned the painting into a photo and the reverse for the photo.

work of two similar image but different styles. To be specific, the painting was transferred into photo-like picture, and the man's photo was turned into a picture with its style adopted from the input painting. This result could be very useful in terms of image blending and other editing fields.

## III. Methodology

- Methodology of Paper I

In this section, we discuss the details of the model and its implementation. The result here uses VGG-Network. We use the feature space provided by the 16 convolutional and 5 pooling layers of the 19 layer VGG-Network.

To obtain a representation of the style of an input image, we use a feature space originally designed to capture texture information [6]. This feature space is built on top of the filter responses in each layer of the network. It consists of the correlations between the different filter responses over the spatial extent of the feature maps.

Here we want to create a new image that matches the style and content of a given style image and a given content image. We use a white noise image and then update its value to get the target image by minimizing the loss function of the style and content.

We denote $\vec{x}$ to be the input image of CNN. Layer $l$ has $N_l$ distinct filters, each feature map has size $M_l$, where $M_l$ is the height times the width of the feature map. The responses in layer $l$ is stored in matrix $F^l \in R^{N_l \times M_l}$ where $F^l_{ij}$ is the activation of the $i$th filter at position $j$ in layer $l$.

Let $\vec{p}$ and $\vec{x}$ denote the original content image and the image generated, $P^l$ and $F^l$ their respective feature representation in layer $l$. We define the square-error loss of the content

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

The derivative of $L_{content}$ is

$$\frac{\partial L_{content}}{\partial F^l_{ij}} = \begin{cases} (F^l - P^l)_{ij} & if\ F^l_{ij} > 0 \\ 0 & if\ F^l_{ij} < 0 \end{cases}$$

On the top of the CNN responses in each layer of the network we build a style representation that computes the correlations between the different filter responses. These feature correlations are given by the Gram marix $G^l \in R^{N_l \times N_l}$, where $G^l_{ij}$ is the inner product between vectorized feature map $i$ and $j$ in layer $l$:

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

Let $\vec{a}$ and $\vec{x}$ denote the original style image and the image generated, $A^l$ and $G^l$ their respective feature representation in layer $l$ The contribution of the layer to the total loss is

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - A^l_{ij})^2$$

the total style loss is

$$L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

The derivative of $L_{style}$ is

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} (\frac{1}{N_l^2 M_l^2}((F^l)^T(G^l - A^l))_{ji} & if\ F_{ij}^l > 0 \\ 0 & if\ F_{ij}^l < 0 \end{cases}$$

The total loss function is

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

Where $\alpha$ and $\beta$ are the weighting factors for content and style reconstruction respectively.

- Methodology of Paper II

The augmented architecture uses the VGG19 CNN framework and concatenates additional semantic channels to some layers. The semantic information can be manually authored or generated by other neural networks, downsampled to fit the target layer. A hyper-parameter $\gamma$ is used to tune the "amount" of semantic information. For each concatenated layer we give it a new semantic label:

$$s^l = x^l || \gamma m^l$$

Where $s$ denotes the semantic label, $x$ is the outcome of a convolutional layer, and $m$ is the semantic channels. The number of semantic channels varies depending on needs: it can be RGBA formatted or greyscale or other as long as it labels the image correctly. Note that for style images we denote it as $s_s^l$.

The training target of this network is the same as proposed by the paper "Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis"[4]. The error function is an energy function that measures how style patterns locally resemble target style and how content patterns resemble original content:

$$E = E_s(\Phi(x), \Phi(x_s)) + \alpha_1 E_c(\Phi(x), \Phi(x_c)) + \alpha_2 \Gamma(x)$$

Here, $\Phi$ denotes the feature map of a layer, and $E_s$ is the MRF style loss function:

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

$$NN_i = \arg\min \frac{\Psi_i(\Phi(x))\Psi_j(\Phi(x_s))}{|\Psi_i(\Phi(x))| \cdot |\Psi_{NN_i}\Phi(x_s)|}$$

For each patch of the outcome features $\Psi(\Phi(x))$, we first find the "nearest neighbour" in the semantic patches by the above cross-correlation function. $E_s$ then is the Euclidean distance between the total output patches and target patches.

$E_c$ is the content loss function simply defined as content Euclidean distance:

$$E_c(\Phi(x), \Phi(x_s)) = ||\Phi(x) - \Phi(x_c)||^2$$

$\Gamma$ is a regularization item to prevent noisy outcome, because low-level information is not fully utilized. The gradient-based $\Gamma$ could introduce smoothness of the synthesized pictures:

$$\Gamma(x) = \sum_{i,j}((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2)$$

It can be seen that the semantic map is not part of the gradient when optimizing, but it indeed takes part in the loss calculation. The optimization step takes L-BFGS algorithm.

Parameters and value control is another key aspect for succeeding in find output, which can be summarized as follows:

1) As semantic channels are manually added or derived from other neural networks, it is often desirable to tune the values of semantic channels to a similar magnitude of the image;
2) $\gamma$ controls the semantic information amount. When it increases, the image will degenerate into "a patch of style content". As it decreases, the image tends to be the outcome of non-semantic-augmented network. An appropriate value would be, as the paper suggests, 50.
3) When the semantic patches are not enough, repetitive patches will be used, yielding imperfect results. In such case, diminishing the value of $\gamma$ and let go of style quality a little bit would ease the problem;

## IV. Implementation

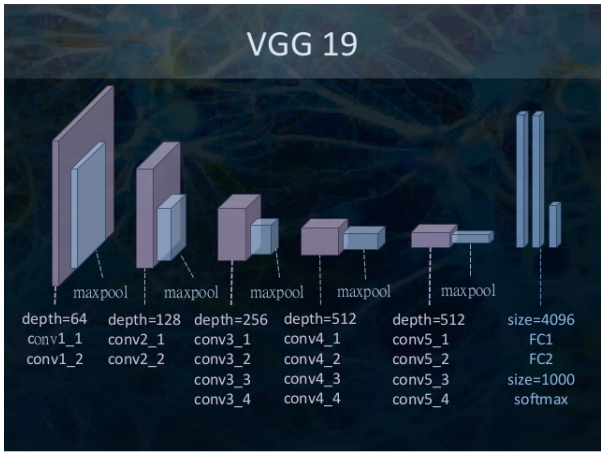


Fig. 6. VGG 19 Network Architecture

- Implementation of Paper I

### A. Deep Learning Network Architecture

In this problem, we use VGG 19 Network to Implement the artificial system. VGG 19 is currently the state of the art in visual recognition, which gets its name because it has 16-19 weight layers. The architecture of VGG 19 is shown in Fig.6

### B. Software Design

*1) top level flow chart:* Fig. 7 shows the top level flow chart of the software design.

In this implementation, we first write a function to build a Convolution Layer. We then build a VGG19 Network using the Convolution Layer we build and follow the architecture shown in part A.

We then generate a white noise image as one of our input. We pick a content image, a style image and the generated image as our input to VGG19 Network.

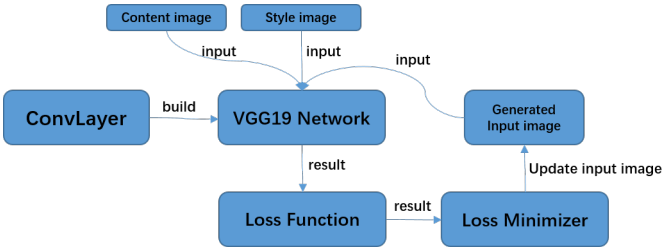


Fig. 7. Top level flow chart of software design

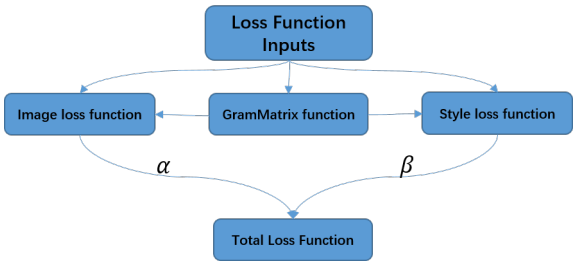


Fig. 9. Loss Function Design

We then define loss functions according to the functions we defined in the methodology part to get loss functions. We then calculate the weighted sum of the loss functions and minimize the total loss.

We use our result to update the generated input image and repeat this process until we arrive at a good result.

*2) Convolution Layer:* Fig. 8 shows the design of the Convolution Layer.

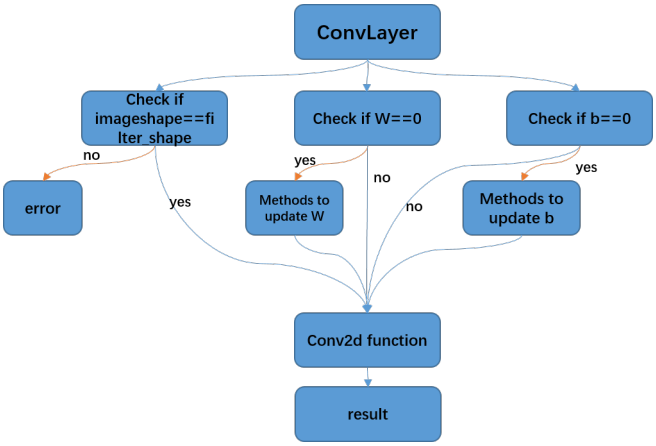


Fig. 8. Convolution Layer design

The design of the convolution layer is similar with the one we implement in our homeworks. However, since the weights we are using here is already trained in advance. We need to do several checks for parameter $w$ and $b$. We also need to keep in mind that there is no maxpooling function here and the activation function here is relu.

*3) VGG19 Network:* We have shown in the Fig. 6 about the architecture of VGG19 Network. We can see that we first go through two convLayers of depth 64 before we do the first maxpooling. Then we have another two convLayers conv2_1,conv2_2 before we do the second maxpooling. Then we have another 4 convLayers before maxpooling the third time. Next we have 4 convLayers before we do the 4th maxpooling.After that , we have conv5_1,conv5_2,conv5_3,conv5_4 and conv5_5.Then we do the last maxpooling. The final layers are fully connected.

*4) Loss Function Design:* We have shown in the Fig. 9 about the design of Loss Function.

We have shown in part iii about the loss functions of the content and style. By taking the weighted sum of the two functions, we can get our total loss function.

*5) Loss Minimizer:* For local minimizer, we use the "L-BFGS" algorithm to do optimization as recommended by the paper.For convience, we use the optimization function scipy.optimize.minimize() in Scipy to implement. By repeating the procedure, we can gradually optimize our result.

- Implementation of Paper I

### C. Deep Learning Network

*1) Architectural block diagram:* We used a VGG19 network as our training network, additionally, certain layers (conv3_1 and conv4_1) of the VGG19 is concatenated to map image Fig.10. And the output of conv4_2 layer is used for calculating the content loss (will be discussed later).

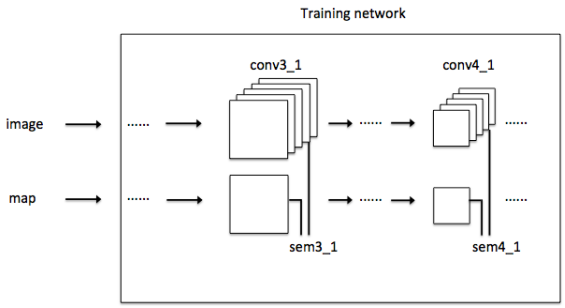


Fig. 10. Architecture of our network. The concatenated layers are called sem3_1 and sem4_1. The input map does not pass through a VGG19 network, it just passes through a pool layer such that the size of the pool layers output is kept the same as that of corresponding convolutional layers (conv3_1 and conv4_1).

*2) Training algorithm details:* Before training, we need to load the weights of an already-trained VGG19 network to our network, and the weights of network will not change during training. An output image will be randomly initialized and put into the network, it will be updated based on the output of its sem3_1, sem4_1 and conv4_2 layers. Besides, style image, style map, content image and content map will also be put into the network, their output of certain layers will also be used to update output image.

*3) Data used:* We used the same images as the article[3] did. There are four inputs. Content image is the image we want to inherit its content. Content map is the semantic part

of our desired image, with which we can change the some properties (e.g. layout) of output image. Style image is the image we want to inherit its style. Style map is semantic part of style image.

### D. Software Design

Top level flow chart is shown as Fig.11.

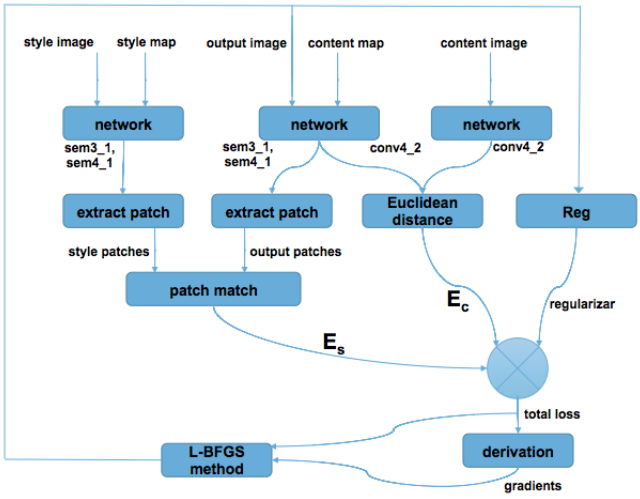


Fig. 11. Top-level flow chart of algorithm

In Fig.11 , the network block is our training network discussed above, certain layers has been used. The extract patch block realizes extract function, it has an extract size of $3 \times 3 \times C$, where C is the number of channels. It generates all patches of the input. The Euclidean distance block calculates Euclidean distance between to inputs, the Reg block calculate regularize which will be discussed later. The derivation block calculates the gradient of each element of output image and the L-BFGS block uses total loss, its gradients and older value of output image to update output image. For two inputs of the patch match block, every element in the input of output patches needs to go through the set of style patches to find matching patch (with maximal normalized cross-correlation).

As shown in Fig.11, style image and style map, output image and content map and content image pass through the network respectively. Certain layers of network has been used for further calculation. Total loss, which is used to update output image, consists of three parts, $E_c$, $E_s$ and regularize.

To calculate $E_s$, first use sem3_1 layer and sem4_1 layer of style image and style map, extract patch on them such that every patch has a size of $3 \times 3 \times C$, the set of patches is denoted as style_pathches Then do the same patch on sem3_1 layer and sem4_1 layer of output image and content map, the set of patches is denoted as output_patches. Next for every patch in output_patches, find the matching patch in style_pathces with the maximal value of normalized cross-correlation. Finally, sum the Euclidean distance between every patch in style_pathces and its matching patch up, $E_s$ is obtained through the sum timed by a weight denoted as style-weight. To calculate $E_c$, use conv4_2 layer of output image and content image, calculate the Euclidean distance between both layers. $E_c$ is obtained through the Euclidean distance timed by a weight denoted as content-weight. The regularize is calculated as:

$$\Upsilon(x) = reg\_weight * \sum_{i,j}((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2)$$

Where $x_{ij}$ is pixel in output image in every channel, $reg\_weight$ is the weight of this term.

After obtaining the total loss, calculate its gradient on each element of output image then iteratively use L-BFGS method to update output image. After the iteration is complete, show the output image.

## V. Results

- Results of Paper I

### A. Project Results

The results we get from the system is shown below: We

Fig. 12. Artisic Style image we get from our system

can see from our result that our system actually do quite well in creating artistic style images, being able to capture the style of the style image while preserving most details in the content image.

### B. Comparison of Results

We first compare Fig.12 with Fig.14. We can see that our picture has more details in style than the result obtained by the original paper. The color used by the original paper is also not as good because it uses black to the tower. Our result also preserve more details while the result from the paper loses some of the detials of the photo(buildings in the left hand side).

Fig.13 and Fig.15 are both not ”perfect” because they both pay too much attention on the ”style” of the sea which makes the picture too vague.

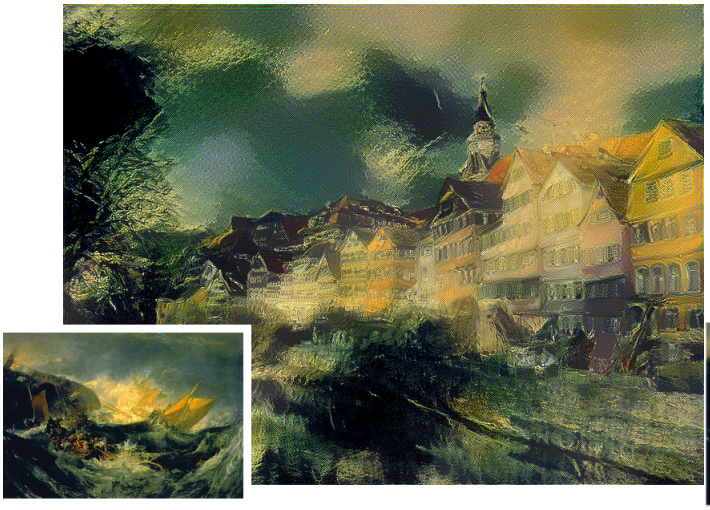

Fig. 13. Artisic Style image we get from our system


Fig. 14. Result obtained by the original paper

### C. Discussion of Insights Gained

In general, we think our system is giving better result than the original paper. However, this is partly due to tremendous times of paramter modification. During the modification, we find that the style layer output conv1_1 seem to has an too large effect while conv3_1 and higher convolution layers seem to preserve more delicate details of style image. So it is wise to give more weights to conv3_1,conv5_1 and less weights to conv1_1.

Also, since the parameters are modifiable by user, the system lack the ability to define the performance of result independently. Parameters good for one input may not work well for another. In order to get good results, experience and human involvement is still needed.

- Results of Paper II


Fig. 15. Result obtained by the original paper

### D. Project Results 2

We exerted three groups of experiments, in first two groups, we firstly set two pictures as style and content respectively then change the content image to style image and vice visa, in the third group, we set content image and style image as the same and change the layout of content map, wanting to obtain an image with the new layout.

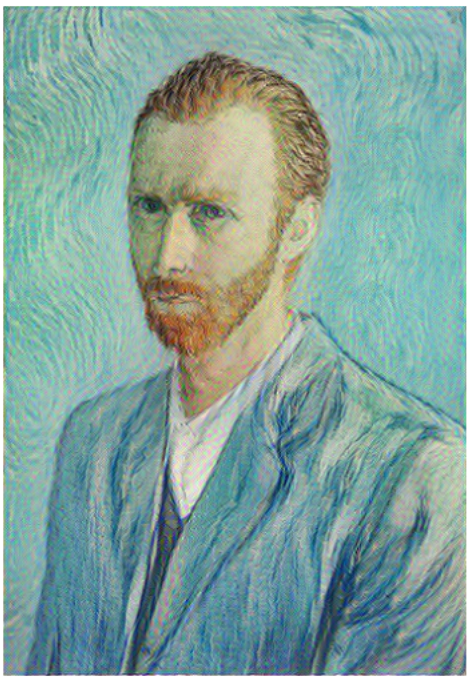

Fig. 16. Artisic Style image we get from our system

Three our synthesized images are shown in Fig.16,17,18 , corresponding images of article[3]: is shown in Fig.19,20,21 .

Fig. 17. Artisic Style image we get from our system

Fig. 18. Artisic Style image we get from our system

### E. Comparison of Results

In the first group, compared to our result, article[3]'s result inherited the style better. However, our result better reconstruct some detailed contents. For example, the eye in our result is closer to the content (theirs are blurred), and in part of the collar, our result recovers the content, theirs inherited the content of the style image more, which did not meet the requirement.

In the second group, our result shows some inappropriate folds on clothes and there should not be a color lump on brow, except those, other parts recovers well. The overall quality is worse than theirs.

In the third group, both two results represented the new layout we want, compared to their results, ours is slightly worse in style reconstruction, but the left-up part of the image better recovers content than theirs did.

### F. Discussion of Insights Gained

During our experiment, we find that we have to use plenty of time to adjust the weights (semantic weight, style loss weight, etc.), indicating that the ideal weights may be different for different specified input images. Make these weights to be trained processing may be a good way to solve the time-consuming problem.

Fig. 19. Result obtained by the original paper


Fig. 20. Result obtained by the original paper

## VI. Conclusion

We can create artistic style using semantic layer, compared to the article, several parts of our results is better while others are worse. The project familiarizes us with VGG network, deepen our understanding of CNN and theano. The conception of "style and content" and "semantic layer" is creative and is able to create artistic style images.

## References


[1] https://bitbucket.org/e_4040_ta/e4040_project_qcda

[2] Leon Gatys, Alexander Ecker,Matthias Bethge. *A Neural Algorithm of Artistic Style*. 2 Sep 2015

[3] Alex J. Champandard,nucl.ai Conference 2016 *Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artwork*. 5 Mar 2016

[4] Chuan Li,Michael Wand *Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis*. 5 Mar 2016

[5] https://en.wikipedia.org/wiki/Convolutional_neural_network

[6] Gatys, L. A., Ecker, A. S. & Bethge, M. *Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks*.

Fig. 21. Result obtained by the original paper

| | kc2980 | jq2250 | hd2358 |
|---|---|---|---|
| Last Name | Chen | Qian | Ding |
| Fraction of (useful) total contribution | 40% | 30% | 30% |
| What I did 1 | Coding of article1 | Coding of article 2 | Coding of article 2 |
| What I did 2 | Report writing | Report writing | Report wrting |
| What I did 3 | | | |

Fig. 22. Table of contribution