Theme based colorization

Nehal Singh Puja Kumari Prabha Kumari
Akshat Solanki
IIT2018119 IIT2018191 IIT2018195 IEC2018055
Indian Institute of Information Technology, Allahabad

Abstract

We consider image transformation problems, where an input image is transformed into such output image which is a combination of the content images and style images. Recent approaches to such problems typically train neural feed-forward convolutional networks using the loss of each pixel between output and low-resolution images. Concurrent performance has shown that high-quality images can be created by defining and enhancing vision loss functions based on high-quality features extracted from pre-trained networks. So In this project we are going to use convolutional neural networks for image transformation tasks and train using perceptual loss function with VGG19 model.

1. Introduction

1.1. Problem Statement

We will try to build the enhancement method using two images, a content image and a reference style (like a famous artist's work), and blend them together to make the resulting image look like a content image, but a "painted" style reference image style.

1.2. Solution Proposal

In other important areas of visual acuity such as visual acuity and facial recognition have recently demonstrated a class of biologically inspired visual models called Deep Neural Networks. Here we will use a Deep Neural Network based system that creates high quality visual images.

2. Motivation

In fine art, especially painting, people have learned the art of creating unique visuals by combining complex images.

Therefore, we will create such a model that can produce new high-quality imaginative images that incorporate absurd image content and the appearance of many well-known works of art.

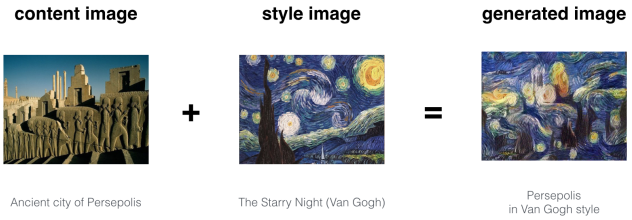


Figure 1. **Using two famous artworks and performing theme based colorization on them.**

3. Literature Survey

We referred to a number of reports and papers to understand the extent of the problem areas.

3.1. Paper Title:Perceptual Losses for Real-Time Style Transfer and Super-Resolution

[4] **Objective** : Objective of this paper is to transform the input image into an output image which is a combination of the content image and style image.

Methodology : Approach for solving image transformation tasks is to train a feed-forward convolutional neural network in a supervised manner, using a per-pixel loss function to measure the difference between output and ground-truth images. Such approaches are efficient at test-time, requiring only a forward pass through the trained network.

Results : They achieve comparable performance and drastically improved speed compared to existing methods, and to single image super-resolution where training with a perceptual loss allows the model to better reconstruct fine details and ed

3.2. Paper Title : A Neural Algorithm of Artistic Style

[3] **Objective** : A key finding of this paper is that content and style presentations on the Convolutional Neural Network vary. That is, we can manipulate both presentations independently to produce new, more realistic images. The

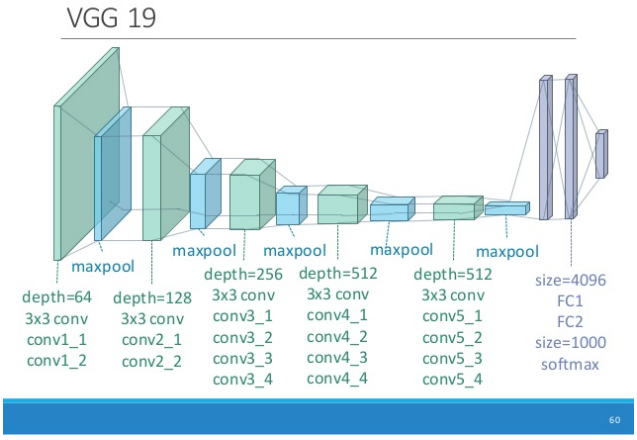


Figure 2. **CNN Model**

program uses neural representations to isolate and reassemble the content and style of in appropriate images, providing a neural algorithm for artistic image creation.

Methodology :In this paper they present a performance system based on the Deep Neural Network that creates high-quality imaginative art images. The pre-submitted results in the main text were developed on the basis of the VGG-Network, a Convolutional Neural Network that works in line with the visual function of the visual object and is presented and explained in detail. Use the feature space provided by 16 convolutional and 5 integration layers of 19 layer VGG Network.

Results : They generate mixed images of the content and style representation from two different source images.

4. Summary of Proposed Plan

Our main aim will be to produce a mixed image out of two images provided, i.e a content image and a style image. Mixed image contains style (such as texture, colour) from style image and content from content image.

- Style Transfer : Technique of re-composing images in the style of other images [1]
- CNN : Also called as Convolutional Neural Networks and are the class of Deep Neural Networks that are most powerful and widely used in image processing tasks.
- We would like to present it with web-UI to make it user friendly.

5. Methodology

CNN i.e Convolutional Neural Network consists layers that process in feed-forward manner having small computational units. Each layer is a collection of image filters which extracts a certain feature from the image. Output of each given layer consists of feature-maps : differently filtered versions of the input image.

We found content and style representations of the image using the middle layers of the model. Starting from the input layer of the network, first few layers were activated which represents low-level features,for example: edges and textures. Moving ahead last few layers represent high-level features,for example:parts of eyes. We used VGG19 network architecture which is a pre-trained image classification network for this.So in short intermediate layers were used to define content and style representation from the images. Now for the input image we tried matching the corresponding content and style representations at these layers.When we come to a high level,to perform image classification a network must understand the image. It should take input pixels of the raw image and build a internal representation which will convert the pixels within the image into a complex understanding of features.

This is also one of the main reason that CNNs are able to generalize so well because they are able to capture the defining features and invariances within classes(for example:cats against dogs).Thus,this model serves like a complex feature extractor between the raw image that was fed to the system and classification levels as output.We can describe style and content from the images by accessing intermediate layers.

- **Extraction of content from the input image**: The input image needs to be transformed into representations that will care more about the actual content of the image than detailed pixel values.

  We refer to feature responses for high-level network coverage as content representation. The 4_2,5_2 convolution layer of the pre-trained VGG-19 network is used as a content extractor.

- **Extraction of style from the image**: To get style input image representation, we use correlations between different filter responses for spatial extent of the feature map . Using that we get a stationary and multi-scale representation of the input image which therefore captures its texture information but not the global arrangement.

  This correlation between feature maps is called as gram matrices. The layers used for calculation of the gram matrix are 1_1, 2_1, 3_1, 4_1, 5_1 with varied style weight constant for each layer. This constant can be seen as a hyper-parameter used for changing style levels

- We have used VGG-19 network where the feature space is provided by the 16 convolutional and 5 pooling layers of the 19 layers.

- Input image as $\overrightarrow{x}$ is coded on each CNN layer with filter responses for that image. A layer with N_L distinct filters has N_L feature maps each of size M_L , where M_L is the width times the height of the feature map.

- For a layer L in a matrix the responses can be stored as: $F^l \epsilon R^{N_l \times M_l}$ where F^l_{ij} is the activation of the i^{th} filter at position j in layer L.

- **Content Loss** : Let P, the original image and X, the generated image and P and F represent their different features in the layer. We then define a square error error between two feature presentations as content loss.

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

- **Style Loss** : We can define style as means and correlations across the different feature maps. We can define A, as the original image and X,as the Image that is generated and A_i and G_i their respective style representations in the layer i.So we calculate the contribution of that layer to the total loss as

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

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

- **Gram Matrix** :

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

is called as Gram matrix ,where G_{ij}^l is defined as the inner product between the vectorized feature map i and j in layer l .

- In order to generate the images which are a mixture of the content and style,we jointly minimised the distance of a white noise image from the one layer of the network for the content representation and in a number of layers for the style.

- Considering p,as the photograph and a, the artwork. The loss function was minimised and calculated by:

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

α and β are hyper-parameters α is Content weight β is Style weight

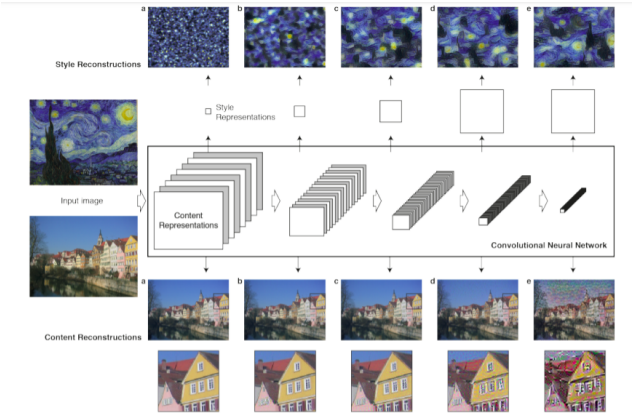


Figure 3. **Reconstructions of Style and Content image.**

6. Implementation

1. First step was to read content and style images.
2. If multiple style images are used, defining weight for each style image, else by default equal weights are assigned.
3. Defining:
   CONTENT_LAYERS = ('relu4_2', 'relu5_2')
   STYLE_LAYERS = ('relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', 'relu5_1').
4. Neural_style.py is called with parameters "content image","style image", "checkpoints-output format","checkpoint iterations number"
5. The stylize() function from Stylize.py is called which yields tuples(iteration,image,loss_vals) at every iteration.
6. Normalization of style layer weights is done
7. Compute content features and Style features in feed forward mode
8. Make stylized image using backpropagation.
9. Calculating the content and style loss using formulas mentioned above.
10. Calculating total variation denoising and then total loss. Total variation denoising is decreasing high frequency artifacts by using an explicit regularization term on the high frequency components of the image.
11. Setup optimizer as adam's optimizer and start optimization for each iteration. To optimize it we will use a weighted combination of the two losses to get the total loss as mentioned above.
    train_step = tf.train.AdamOptimizer(learning_rate, beta1, beta2, epsilon).minimize(loss)

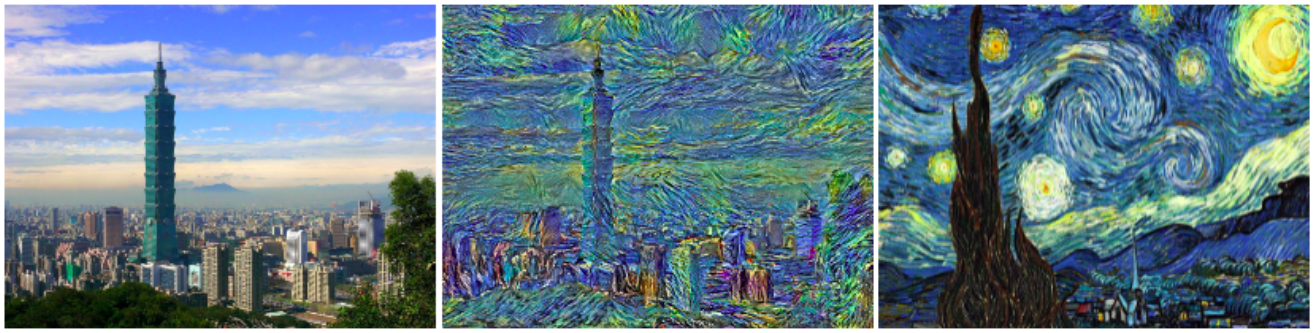
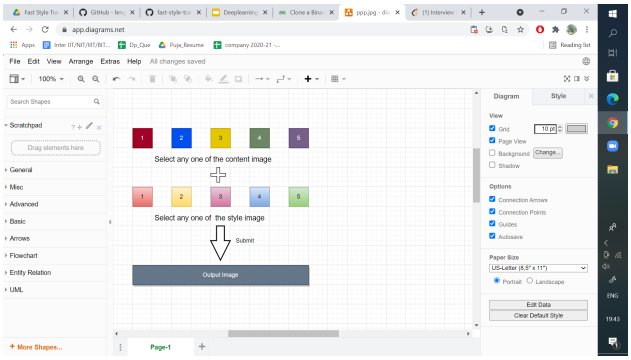


Figure 4. Results



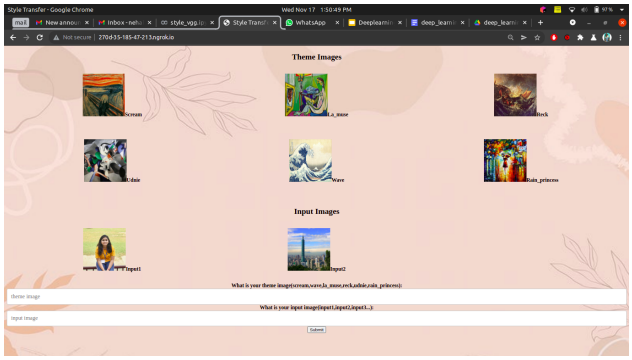


Figure 5. Initial and final design of website

12. Luminosity transfer steps:
    - Convert stylized RGB→ grayscale according to Rec.601 luma (0.299, 0.587, 0.114)
    - Convert stylized grayscale into YUV (YCbCr)
    - Convert original image into YUV (YCbCr)
    - Recombine (stylizedYUV.Y, originalYUV.U, originalYUV.V)
    - Convert recombined image from YUV back to RGB
13. Save image for specific iteration in numpy array ,and then save the image in path with : *def imsave(path, img): img = np.clip(img, 0, 255).astype(np.uint8) Image.fromarray(img).save(path, quality=95)* [2]

7. Results

Code was run for 1000 iterations which took approximately 6 hours.We can get more good and accurate results after doing more iterations over there. See Figure 7 for accuracy and iterations view and Figure 4 for results. Also to have a look at images generated at each 100 iterations Figure 6 Also we made a web application for better interaction of user where user can simply select Content and Style image .Initial design and final design of website is shown at Figure 5

8. Limitations of proposed work

Since convolutional neural networks are often used to classify images, we are often dealing with high-resolution data (images). We usually need a large amount of data for the convolutional neural network to function properly. It takes long time to train the convolutional neural network, especially with large databases. To speed up the process we need some kind of hardware like a GPU. CNNs are generally not good at managing rotation and scale fluctuations without explicit data additions. It take high calculation costs.

9. Conclusions

In this project we generate a mixed image out of two images provided, i.e a content image and a style image using CNN with VGG19 model. Here, a mixed image contains

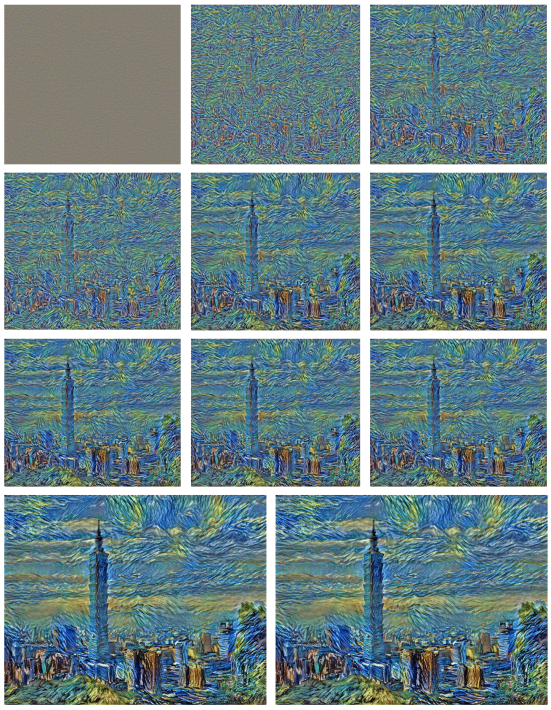


Figure 6. **Images generated at each 100 iterations(Numbered from 0 to 1000)**

style (such as texture, colour) from style image and content from content image. For showing this project we built a web application which would allow users to select themes and input images and see their output. It was made using Flask, CSS and HTML. We also calculated the content and style loss for computing the final loss that is 1.77528e+06.

References

[1] Neural style transfer : Tensorflow core. https://www.tensorflow.org/tutorials/generative/style_transfer. 2

[2] Logan Engstrom. Fast style transfer. https://github.com/lengstrom/fast-style-transfer/, 2016. 4

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015. 1, 6

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016. 1, 6

[5] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017. 6

Supplementary Material

Project Planning

Task	Status	Date
Requirement Verify	Done	5th September 2021
Project Planning	Done	8th September 2021
System Design	Done	20th September 2021
Details Design	Done	30th September 2021
Coding	Done	15th October 2021
Debugging and coding	Done	30th October 2021
Testing	Done	10th November 2021
Documentation and Final	Done	20th November 2021

Table 1. **Work done in C1, C2 and C3 respectively.**

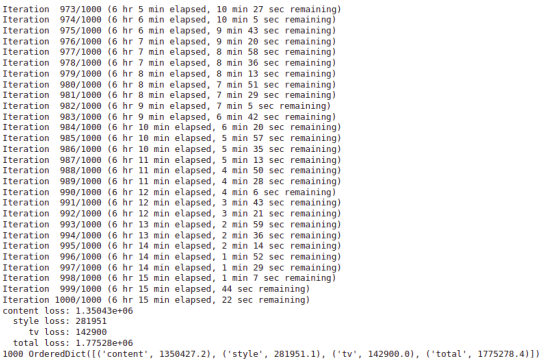


Figure 7. **Screenshots of alogrithm run.**

A. Timeline

See Table 1.

B. Dataset description

Our dataset consists of content image and style images . For content images we took different images from net. For style images we took famous paintings. Ex.,Udnie , Rain princess,The Starry Night, Reck etc. We used pretrained VGG model : http://www.vlfeat.org/matconvnet/models/beta16/imagenet-vgg-verydeep-19.mat

C. Technologies Used

- python (Anaconda) [V 3.5.4]
- tensorflow [V 1.4.0]
- numpy [V 1.13.3]
- PIL [V 4.2.1]
- matplotlib [V 2.0.2]

D. Description of code availability

There are research papers and codes available which resemble our problem statement. However most of them need a high GPU to run and cannot be run on our system. So we have be tried to implement the Gatys' Research paper. There are some research papers which we use in order to understand algorithms. Research paper : [4] [3] [5]

E. Contribution of group

We took help from a research paper named "A Neural Algorithm of Artistic Style" (A Neural Algorithm of Artistic Style). We are going to show this project through a web application for ease of use. There users can select content image image and style image and can see their output. There are following tasks which did in this project :

- Training for Style Image
- Testing for Style Image
- Evaluation of checkpoint created using content image
- Creation of web application