SANET: A Deep Learning Approach for Style Fusion and Transformation of Arbitrary Images

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
SOFTWARE ENGINEERING

Submitted by:
Pratibha Rathi
(Roll No. 2K18/SWE/12)

Under the supervision of
Dr. Manoj Kumar
(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

JULY, 2020

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

CANDIDATE'S DECLARATION

I, Pratibha Rathi, Roll No. 2K18/SWE/12 student of M.Tech (Software Engineering), hereby declare that the project Dissertation titled **"SANET: A Deep Learning Approach for Style Fusion and Transformation of Arbitrary Images"** which is submitted by me to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Pratibha Rathi

Place: Delhi **Pratibha Rathi**

Date: 26/06/2020 (2K18/SWE/12)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

CERTIFICATE

I hereby certify that the Project Dissertation titled **"SANET: A Deep Learning Approach for Style Fusion and Transformation of Arbitrary Images"** which is submitted by Pratibha Rathi, 2K18/SWE/12 Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

[illegible]

Place: Delhi **Dr. Manoj Kumar**

Date: 27/06/2020 **SUPERVISOR**

(Associate Professor)

Department of Computer Science & Engineering

Delhi Technological University

ACKNOWLEDGEMENT

The success of the Major II project requires help and contribution from numerous individuals and the organization. Writing the report of this project work gives me an opportunity to express my gratitude to everyone who has helped in shaping up the outcome of the project.

I express my heartfelt gratitude to my project guide **Dr. Manoj Kumar** for giving me an opportunity to do my project work under his guidance. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success. His constant support and encouragement has made me realize that it is the process of learning which weighs more than the end result.

I would like to express my gratitude to the university and staff for providing us with the library, laboratories, infrastructure, testing facilities, right academic resources and environment which allowed us to work without any obstructions.

I also reveal my thanks to all my classmates and friends for constant support. They helped me throughout by giving new ideas, providing necessary information and pushing me forward to complete the work. I would also like to express sincere gratitude to my parents for constantly encouraging me during the completion of work.

Pratibha Rathi

Pratibha Rathi

Roll No – 2K18/SWE/12

M. Tech (Software Engineering)

Delhi Technological University

ABSTRACT

For real-time applications of arbitrary style transformation, there is a trade-off between the quality of results and the running time of existing algorithms. Hence, it is required to maintain the equilibrium of the quality of generated artwork with the speed of execution. It's complicated for the present arbitrary style-transformation procedures to preserve the structure of content-image while blending with the design and pattern of style-image. This project presents the implementation of a network using SANET models for generating impressive artworks. It is flexible in the fusion of new style characteristics while sustaining the semantic-structure of the content-image. The identity-loss function helps to minimize the overall loss and conserves the spatial-arrangement of content. The results demonstrate that this method is practically efficient, and therefore it can be employed for real-time fusion and transformation using arbitrary styles.

Keywords: image processing; deep learning; neural style transfer; computer vision; arbitrary image stylization; SANET; ASPM; convolutional neural networks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations and Nomenclature

1. CNN- Convolutional Neural Network
2. PSPM- Per Style Per Model
3. MSPM- Multiple Style Per Model
4. ASPM- Arbitrary Style Per Model
5. IBAR- Image Based Artistic Rendering
6. SBR- Stroke Based Rendering
7. RBT- Region Based Techniques
8. IPF- Image Processing and Filtering
9. EBR- Example Based Rendering
10. NST- Neural Style Transfer
11. MGAN- Markovian Generative Adversarial Networks
12. AdaIN- Adaptive Instance Normalization
13. WCT- Whitening and Coloring Transformation
14. SANET- Style Attentional Networks
15. MRF- Markov Random Field
16. CPU- Central Processing Unit
17. RAM- Random Access Memory
18. ReLu- Rectified Linear Unit
19. GPU- Graphics Processing Unit
20. IDE- Integrated Development Environment

CHAPTER 1

INTRODUCTION

1.1 Overview

The art of clicking photos involves the realization of the proper background, decent lighting, and editing to make them beautiful. Adequate editing for photographs can add life to our still postures or actions and deliver us a creative artwork to relive and cherish those moments. Nowadays, this art has been spread to all of the people with the smartphone revolution. Wherever we go, we can see people holding their phones and clicking random photographs to gather their memories. For improving the quality of those pictures, many picture editing tools are available. The editing of ordinary photographs can make them look extraordinary. The editing of the photographs can make our memories even more memorable. Most people edit their photos before posting them on social networking sites to look more attractive and vibrant.

Picture editing is also helpful for real-estate services firms, for increasing the generous attractiveness of properties by improving and intensifying images displayed on their websites. It will help in attracting numerous clients and also help in selling that property quicker. Photo editing can make any occasion look much lively and pleasant. We can also convert our old black and white photographs into colorful pictures by applying unlimited effects to simple ones. A newly developed mobile application called Prisma is an industrial application that applies the NST algorithms to edit photographs. This application has achieved great success and popularity around the world due to the much better quality of stylization. Few similar applications provide similar services like Ostagram is paid but offers faster stylization speed. With these automated applications, people can build their artwork and paintings to share with others on twitter, instagram, and facebook, which is a new method of social communication. These methods are much favorable for artists and widely used by them to impose their art works on digital photographs.

1.2 Scope of Project

In the recent few years, various research works contributed to improve and accelerate the development of deep learning methods for continuous reformations of style transformation procedures. Arbitrary style transformation is the process of fusing new artistic styles with the content of the given image to create impressive artworks. NST was first introduced and explained by Leon A. gatys in 2015. On the basis of this idea presented by Gatys, many NST applications like deepdream, prisma, and ostagram, etc. have been released to generate amazing artworks.

The method proposed by Gatys [1] [2] follows an iterative procedure for the optimization of the generated image, which makes it computationally expensive and slow. However, the quality of the generated output is quite impressive. After that, numerous models have been proposed based on the optimization of the employed model. Johnson et al. [3] and Ulyanov et al. [4] [5] proposed fast offline approaches for generating output in a single feed-forward-pass, but their model is limited in the number of styles, and the quality of the result is also reduced.

For fusing arbitrary styles using a single model, some ASPM approaches have been introduced. The style-swap model [6] was stated that employs the patch match procedure to substitute the features of content by the nearest match of characteristics of the style. Li et al. [7] specified multilevel stylization by recursively employing whitening-colour-transformation to enhance the quality of output and to protect the content-structure. AdaIN [8] model changes the mean-variance of the content-image to match with style-image for transforming features. Avatar-net [9] model contains a style-decoder model based on patch-match and an hourglass network for adapting styles on multiscale. To solve the trade-off between content and style losses, this project presents a new loss function identity-loss and implementation of a network consist of an integration of two SANET models for preserving both content-structures and style-characteristics.

1.3 Problem Statement

There is a trade-off between the quality of resulting output and the execution time of existing algorithms for real-time applications of arbitrary style transformation. After thoroughly examining the properties of various procedures for neural-style fusion and transformation, we can see that the online method algorithm based on optimization of image proposed by Gatys et al. takes a significant amount of time to calculate results compared with offline methods that apply the procedure of optimization of employed models. However, the quality of results generated by online methods is better in comparison to the quality of output produced by offline methods. Therefore, it is required to create a network capable of producing resulting artworks for any arbitrary image while maintaining a balance between the quality of output-images and speed of execution of the model.

1.4 Objective of work

The given research project work is presented with the following given objectives.

- To study all the existing significant works that have already been done in this field of style fusion and transformation, before and after the emergence of deep learning.
- Analysis of the quality of results and execution time of various NST approaches and to demonstrate the analysis graphically.
- To examine the categorization of NST algorithms based on their working.
- To implement a new model consisting of style-attentional networks for generating artwork using arbitrary style-images while sustaining both style-features as well as the spatial distribution of content-image.
- To introduce a new loss function named identity-loss function that helps to minimize the overall loss and conserves the spatial-arrangement of content.
- To analyze the proposed mechanism and compare its output and execution time with other ASPM approaches.
- The ultimate goal of arbitrary style-transformation is to simultaneously achieve and preserve generalization, quality, and efficiency.

1.5 Organization of Dissertation

The arrangement of the dissertation is as follows. In chapter 1, the need and importance of this project work are explained in detail. Below in chapter 2, background information of style transformation methods is given, which covers the explanation of some IBAR methods along with their drawbacks. In chapter 3, a detailed literature review has been done explaining all the major work that has already been done in this field of work. Section 3.1 and 3.2 provide information about the existing neural methods of style transformation and their categorization into online and offline divisions. Section 3.3 illustrates the analysis of optimization approaches, and section 3.4 gives expansion areas of NST by providing some detailed information about them. Chapter 4 describes the proposed method in which section 4.1 explains the architecture of the SANET model, and section 4.2 shows the calculation of loss-functions. Section 4.3 illustrates the hardware requirements, software requirements, and data set used in training, and the model's testing is listed along with the description of the detailed implementation procedure. In section 4.3.5, all the steps of implementation are described. Then in Chapter 5, in-depth analysis and review of the implemented method have been done. In section 5.1, the analysis of the execution time of various NST approaches is displayed with the help of graph plots. Section 5.2 demonstrates the implementation results, and section 5.3 describes the comparison of implemented SANET network with other existing ASPM models. Finally, section 5.4 summarizes the conclusion. Then Chapter 6 contains the references to all the resources that have been used to gather the information to work on this project.

CHAPTER 2

BACKGROUND

In this chapter, we will explain the background information of the approaches used in style transformation before the evolution of deep learning. The techniques in the taxonomy of image based artistic rendering (IBAR) were employed for the stylization of two-dimensional images. First, we describe the procedure of some IBAR [10] methods and then review their drawbacks.

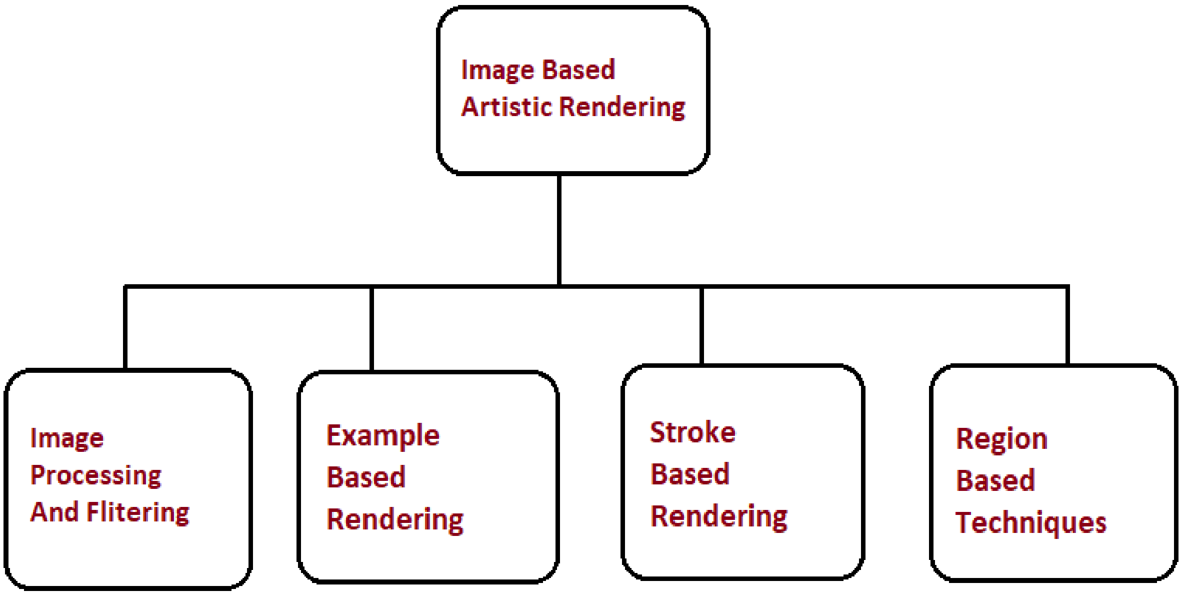

Fig.2.1: Categorization of IBAR techniques

2.1 Stroke-based rendering

It is the process of integrating portraits, paintings, and artworks by providing the strokes or marks on a digital canvas using brushes, lines, and tiles [11]. For converting a photograph into a particular style, virtually some brush strokes are superimposed at some positions on a digital canvas. It can create conventional arts based on the brush, for instance, an oil painting. This procedure generally starts from a source picture by sequentially adding strokes to modify that picture and ultimately composing a nonphotorealistic image that looks like the original photograph carrying an exquisite style. The limitation of SBR methods is that all SBR approaches are intended to create a single distinct style and therefore is not proficient for composing arbitrary styles. It illustrates the nonflexibility of this approach.

2.2 Region-based techniques

RBT is used to accommodate the effects on the basis of the content in regions. Previous RBT algorithms [12] deteriorate the shape of structures in managing the arrangement of strokes. The latter method introduces an IBAR procedure [13] based on regions to alter the semantic geometry of photographs for generating creative effects. This algorithm replaces parts of the picture by many canonical patterns for constructing uncomplicated shape rendering results. In this approach, a source image and target photograph is distributed into many areas using the method based on graphs. The semantic conformity is estimated between them on the basis of colours, patterns, and texture. Ultimately, the colours and textures are transferred for each area by employing a patch match method. Drawbacks of RBT methods is similar to SBR, all RBT approaches are intended to create a single distinct style and therefore is not proficient for composing arbitrary styles. It represents the nonflexibility of this approach.

2.3 Image-processing and filtering

Usually, the IPF approach for rendering is simple to implement and effective in application. IPF techniques [14] are applicable to real time rendering and accommodate multicore CPU and GPU. Bilateral and gaussian filters are employed to compose the effects of cartoons. Kuwahara filter is an exciting range of filters for preserving edges. It works relatively well on pictures of high contrast. These filters eliminate features in sharp regions and preserve shape outlines in areas of low contrast. Some other filters are morphological, diffusion, and shock filter. Gradient domain techniques are also utilized in which the result is interpreted by creating a gradient field. The drawback of IPF is that style variety, and diversity is inadequate.

2.4 Example based techniques

EBR [15] understands the mapping within the prototype pairs containing the source photograph and the corresponding styled artwork. The learned mapping is employed to style arbitrary pictures. EBR can work in two categories by performing texture transformation and colour transformation. The proposed framework named image analogy is a supervised method containing two stages, namely the design stage and application stage. In the design stage, it intends to study the analogous mapping between the pairs of source pictures and target outputs. The data used for training in image-analogy contains combinations of input-pictures along with the corresponding stylized output-pictures. In the application phase, learned mapping is applied to new target photographs to produce relevant analogous results. The main drawback of this method is the unavailability of training data. This method fails to capture the structure of images accurately and uses only low level features of the picture.

CHAPTER 3

LITERATURE REVIEW

Neural Style-Transfer is a machine learning technique that takes two images in input, one of which is content-input image and another one is called style-input image, for example, the artwork in a famous painting and blend them in order to transform the input images into an output image that looks like the picture called content-image but it will be painted in a similar fashion to the picture called style-image. Neural style transfer uses deep learning and its convolution neural networks to perform this amazing task. It is much propitious for artists and widely used by them to impose their art work on digital photographs.

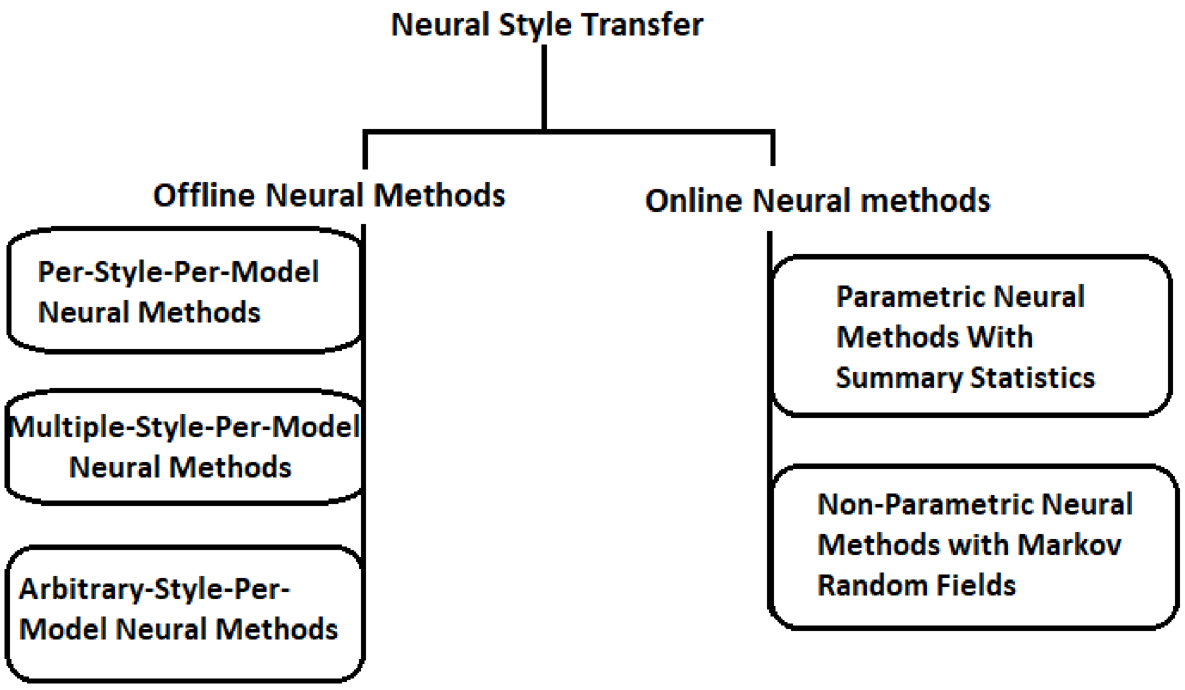
Neural Style Transfer
Offline Neural Methods
Online Neural methods
Per-Style-Per-Model Neural Methods
Multiple-Style-Per-Model Neural Methods
Arbitrary-Style-Per-Model Neural Methods
Parametric Neural Methods With Summary Statistics
Non-Parametric Neural Methods with Markov Random Fields

Fig.3.1: Categorization of NST techniques

It was first introduced and explained by Leon A. Gatys in 2015. After the Gayts, several endeavours are in progress related to the style transfer and its algorithms to perform training and learning faster as much as possible and to extend the procedure of style transfer techniques from static images to videos, audios, and other mediums. Neural methods of style transfer are broadly divided into two categories. One of which is the online method and the other one is the offline method.

3.1 Online neural methods

One of the famous applications in the field of style transfer is Deep-dream, which is based on online approach of NST. The online method focuses on the optimization of the generated output image but not on the improvement of the applied model. This approach employed several iterations for enhancing the quality of the output image. Due to the multiple iterations, this procedure is much time consuming and computationally expansive. Therefore it is also known as the slow neural method for style transformation. The process is to extract the content information and style representation from input images and then apply this extracted information to the CNN model for recombining both components to generate the artistic output. Gradient descent algorithm is used for the optimization of the result by minimizing the value of loss function. These online methods differ in the approach of applying the content and style representation to the CNN model so categorized into two classes.

3.1.1 Parametric online neural methods

These methods employed spatial representation summary and statistics to generate artworks. Gatys et al. [1] [2] in 2015 proposed an algorithm that applies deep CNN model VGG 19 for extracting structure, shape, and texture, etc. related information of content and style features from preprocessed input pictures. This information is spatially modelled in different layers of VGG 19 and used for the reconstruction of target artwork. VGG nineteen network is composed of sixteen layers of convolution and five layers for pooling, where higher layers are handled to represent the content characteristics. Style-image input is passed through the VGG network, and the gram matrix is employed to store style characteristics.

The overall loss is a linear combination of both content and style losses balanced by parameters calculated through the one thousand iterations of gradient descent. Content loss is computed in terms of mean squared difference. The α and β hyper parameters provide the appropriate weights to both components of loss for balancing the equation of total loss function.

$$L_{total} = \alpha L_{content} + \beta L_{style} \tag{3.1}$$

$$L_{content}(C, G, L) = 0.5 \sum_{ij} \left(a[L](C)_{ij} - a[L](G)_{ij} \right)^2 \tag{3.2}$$

$$L_{GM}(S, G, l) = \frac{1}{4{N_l}^2{M_l}^2} \sum_{ij} \left(GM[l](S)_{ij} - GM[l](G)_{ij} \right)^2 \tag{3.3}$$

$$L_{style}(S, G) = \sum_{l=0}^{L} w_l \times L_{GM}(S, G, l) \tag{3.4}$$

Here S, C, G, and GM are used to represent style-image, content-image, output generated-image, and gram-matrix, respectively. Activation of layer L is used for the calculation of content-loss. N_l is the number of channels in the feature map, and M_l is the product of the height and width of the feature map. After that demystifying approach [16] of NST was introduced that employs a different method for the statistical representation of style features. In this procedure, the gram matrix was replaced by MMD. It states that the minimization of maximum-mean-dispersion is comparable to the matching of gram-matrices. Several different methods for the representation of style were used, which include gaussian, linear, polynomial and batch-normalization, etc.

In 2017, Lapstyle NST was presented [17], which added a laplacian-loss function to calculate the overall loss. This laplacian-loss preserves the low-level fine details of the content characteristics and is aimed to reduce the distortion of edges, shapes, structure, contour, and colours of original input when converted to artistic output.

3.1.2 Non parametric online neural methods

In 2016, Li and wand [18] proposed a method of style transformation that employed a combination of markov-random-field models and deep CNN for synthesizing two dimensional artworks. This approach works well with both photo-realistic and non-photo-realistic transformation of style. For minimizing loss, it uses limited memory BFGS algorithm. This algorithm matches style on the level of neural-patches. Hence, it is suitable for preserving fine details of the structure. They formed a new function for calculating the style-loss and content-loss by calculating euclidean distance between patches using the energy function of MRF-neural-patch-matching given in (3.5).

$$E_s(\varphi(x), \varphi(x_s)) = \sum_{i=1}^{m} \left\| \psi_i(\varphi(x)) - \psi_{NN(i)}(\varphi(x_s)) \right\|^2 \tag{3.5}$$

The characteristic of this algorithm is that it produces considerably well resulting artwork for photo-realistic styles, especially if the shape, structure, and panorama of both content-image and style-image are alike. Though, it usually doesn't work well if there is a great difference in view and arrangement of structures of content-image and style-image because the patches in both pictures could not be paired accurately.

3.2 Offline neural methods

Earlier explained online method uses the iterative approach of gradient descent for improving the quality of generated output. Therefore it consumes a large amount of computation time when image size is considerably large. To overcome this limitation of speed, offline neural methods were introduced. This approach is much speedy compared to the previous procedure because it generates the output artwork in a single feed-forward-pass by optimizing the adopted model. As it mainly focuses on the optimization of the applied model, hence it is also known as the model-optimization-based offline method of NST. This offline approach is further divided into three categories based on the number of styles generated by the model in its output artwork.

3.2.1 Offline neural model generating single style-PSPM

For adopting an individual style, this approach optimizes the employed model. Johnson et al. [3] proposed a parametric method of training a feed-forward-network for the task of style transformation. This method replaces the perpixel losses by perceptual losses computed using high level features of images. The system contains two components of the network, which are image-style transformation network and loss-network. Image-style transformation network is a deep residual CNN that can convert input images to an artistic output by understanding the mapping. It is trained using stochastic-gradient-descent for minimizing the weighted combination of loss-functions, which is computed by employing loss-network. Perceptual losses measure the similarities and dissimilarities of images more precisely compared to perpixel losses.

Another similar approach proposed by Ulyanov et al. [4] uses multi-scale architectures that can be trained faster and result in a smaller loss in content and texture characteristics and better visual quality while working with comparatively few parameters. These are very lightweight models that are capable of generating artworks comparable to the quality of output produced by Gatys et al., and their speed is almost 100 times faster. However, these are also limited in considering the small details and depth information. After that, a new advance concept was introduced, which is known as instance-normalization [5]. It means normalization is applied to each single image in place of the batch of images. It converges faster, learns quickly, and significantly improves the visual quality of generated stylized artwork. This model can be trained rapidly, reduces the overall loss, and enhances the diversity of network output.

The non-parametric approach of single-style per model offline neural methods uses MGAN [19]. It is a markovian model applied to generative adversarial networks for learning the mapping between different representations of the same content-image. MGAN utilized the imagenet dataset for adversarial-training. It successfully preserved the coherent texture of compact content-images, and the speed of synthesis is remarkably fast. MGAN network consists of two subnetworks, namely discriminator and generator, for improving the model in iterations. This method lacks in transferring non-texture style, such as facial-features of two different face-pictures. Facial-features cannot be interpreted as texture because these require the understanding of expression, poses, and gender, etc. semantic characteristics.

3.2.2 Offline neural model generating multiple styles-MSPM

Above mentioned PSPM procedures can generate artistic output hundred times speedier than previous offline neural methods based on image-optimization. But it needs to train several network models individually to generate output for every distinct style-image. It exhibits inflexibility and excessive time consumption in training for each style. For reducing the irrelevant time consumption during the separate training of various models in PSPM, a new approach was introduced for consolidating multiple-styles into a single model, which is known as MSPM.

In 2017, Dumoulin et al. [20] proposed that it is possible to model different styles at the same time by using the same parameters in the convolutional network. For incorporating different styles, shifting and scaling of parameters is required in the layers of instance normalization, which is called conditional-instance-normalization. CIN is an alteration of style-transformation networks that is uncomplicated, scalable, flexible, and efficient for incorporating diversity in output artwork. By consolidating affine parameters of different styles, it can be applied to incorporate multiple-styles in one output artwork.

$$CIN(F(I_c), s) = \gamma^s \left(\frac{F(I_c) - \mu(F(I_c))}{\sigma\big(F(I_c)\big)} \right) + \beta^s \tag{3.6}$$

Here, β and γ are the parameters of CIN. After this style-bank method [21] was introduced, that is an aggregate of various convolution filter-banks. One style is expressed by every filter-bank explicitly in that combination. For converting a picture to a specific style, the corresponding filter-bank is turned over the intermediate feature-embedding generated by a single auto encoder. While keeping a fix auto encoder, it is possible to carry out incremental-learning for appending a new style by training a new filter-bank.

This style-bank fusion approach works in two ways, linear and region-based. In linear fusion, the style-bank layer is fed with a linear combination of multiple styles. While in region-based fusion, content-image is disintegrated into some disjoint regions using automatic k-means clustering, and then combined style transformation can be conducted using more than one style simultaneously. But the size of the model expands as the number of styles learned by it increases, that is a drawback.

To eliminate this limitation, Zhang and Dana [22] explored the capabilities of a single CNN model VGG 16 for consolidating both content and style in one network. The Multiple-style-generative network was introduced, which can control the size of brush strokes in real time transfer of style. In MSG-Net, the comatch layer was used to represent styles in two dimensional ways. For the comatch layer, it is possible to differentiate and train end to end, but tough to conserve the fine details of structures.

3.2.3 Offline neural model generating arbitrary styles-ASPM

The third type of offline neural method is ASPM that intends to build one model for all styles. It states that a single model can be trained to transfer any arbitrary style. ASPM is also divided into two types. Parametric-ASPM employed spatial representation summary and statistics to generate artworks while nonparametric-ASPM applies MRF for texture-modelling.

3.2.3.1 Style-Swap

Chen and Schmidt [6] proposed the first algorithm for nonparametric-ASPM in which they introduced the concept of style-swap for training an inverse network to generate the stylized output for any arbitrary style.

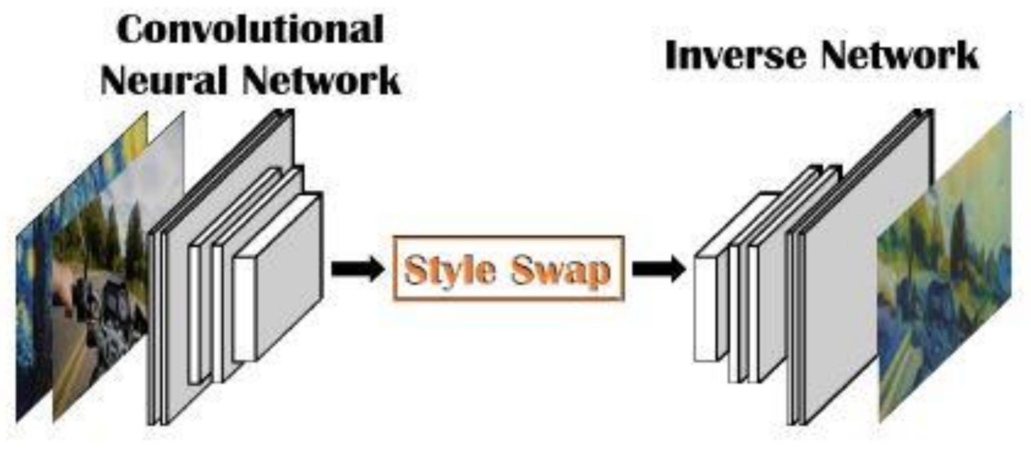

Fig.3.2: Representation of style-swap for training inverse network

At first, a set of patches having sufficient overlap for both content and style activations are extracted for computing in a previously trained VGG network. Each content patch is matched with style patches to find the closest and similar match for swapping. Then content activation is reconstructed by averaging areas of overlapping. This approach of one-model-for-any-style is more flexible than the previous methods but generates less appealing results in which the content is preserved thoroughly, whereas the style is not reflected adequately.

3.2.3.2 Adaptive-Instance Normalization (AdaIN)

In parametric-ASPM, Huang and Belongie [8] modified conditional-instance-normalization (CIN) to adaptive-instance-normalization (AdaIN) instead of training a network for parameter prediction. The above mentioned style swap method takes a considerable amount of time and occupies much memory, but AdaIN is a simplistic approach with the low cost of computation. Here, $I_c, I_s,$ and I_g represents content-image, style-image and generated output-image respectively.

$$AdaIN(F(I_c), F(I_s) = \sigma\big(F(I_s)\big)\left(\frac{F(I_c) - \mu\big(F(I_s)\big)}{\sigma\big(F(I_c)\big)}\right) + \mu\big(F(I_s)\big) \tag{3.7}$$

$$I_g = Dec\Big(AdaIN\big(F(I_c), F(I_s)\big)\Big) \tag{3.8}$$

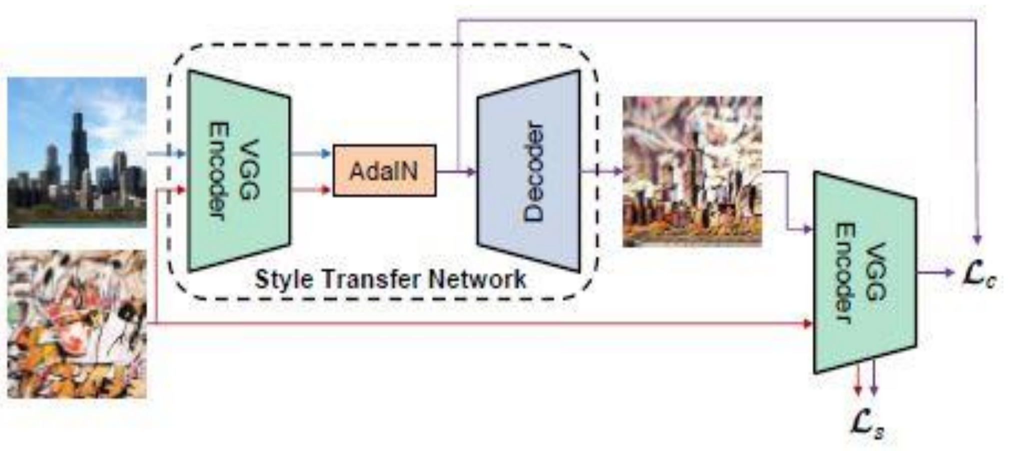

Fig.3.3: Architecture of AdaIN method of parametric-ASPM

For the given input pictures, AdaIN modifies the mean value and variance value of both content-image and style-image for matching. AdaIN layer efficiently transfers the feature-characteristic statistics for performing style transformation. After that, a decoder is used for inverting the output of AdaIN layer, and the VGG encoder is utilized for measuring loss. It is a good parametric-ASPM approach for performing stylization in real time, but it is insufficient in the generalization of new styles. Therefore, it is challenging to synthesize intricate style patterns consist of fine details and complex structures.

3.2.3.3 Whitening-Coloring-Transformation (WCT)

Li et al. [7] changed the AdaIN-layer of the above described model by a combination of the whitening-colour-transformation network. The objective is to protect the arrangement and composition of content. Whitening-transformation receives content-activations from the encoder and then produces a filtered representation of content-image.

After that, colouring-transformation is applied for consolidating the style-patterns into the filtered content design, and the decoder provides stylized output artwork. This method is good in generalizing distinct styles as learning is not required of every style, but not useful in delivering sharp-features and small strokes-details.

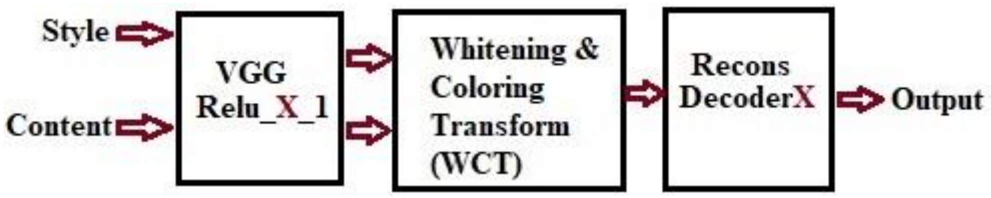

Fig.3.4: Architecture of WCT model

3.2.3.4 Avatar-net

It blends the images based on the semantic-arrangement of content and employs multi scale stylization in a single feed forward pass [9]. It is better than WCT that requires several recursive feed forward passes. Avatar-net uses a style-decorator and an hourglass-network. Patch based style decorator module is used to decorate the content features with style patterns while retaining the semantic properties of content. The hourglass network is employed for multi scale adaptation of style. But its output depends on the patch size, and therefore usually, it is unable to express the local and global styles simultaneously.

3.3 Comparison of Optimization Algorithms

Optimization algorithms are applied to optimize the output of NST by minimizing the value of loss-function. They work during training and learning of the model to update the weights and finding the relevant values for parameters to get an optimal solution. In Table 3.1, we have discussed some popular optimization methods with their merits and drawbacks.

Table 3.1: Analysis and Comparison of Optimization Algorithms

Type	Description	Merits	Drawbacks
Gradient Descent-optimizer	It is a traditional algorithm for the optimization of neural network models. It updates weights and tunes the parameters of the model iteratively to minimize the value of loss function.	Updates weights in neural network models iteratively.	It is very slow and takes ample computation time for huge datasets. It is difficult to regulate the learning rate as the algorithm sometimes stuck in local minima.
AdaGrad-optimizer	It applies a separate learning-rate for each parameter. It modifies the learning rate of the parameter at each step on the basis of the computation of the previous gradients of that parameter.	Very helpful for handling scattered data and we don't need to set the learning rate manually.	Its main limitation is the decrease and decay in its learning rate value.
AdaDelta-optimizer	It is an improvement of AdaGrad-optimizer. It entirely replaces the learning-rate parameter by the exponential moving-average of squared-deltas.	No need of the learning-rate parameter.	There is a necessity of two-state variables for saving the second-moments of gradients and the variation in parameters.
Adam-optimizer	It is an adaptive-method for defining separate learning-rates for every parameter using momentum and therefore termed as adaptive-moment-estimation.	It separately defines the learning-rate for every parameter.	Its gives better result in the starting of training, but performance diminishes as the time increases. Therefore, for some tasks, it does not converge to an optimum value.

L-BFGS-optimizer	It is a limited-memory-BFGS algorithm in the family of QUASI-Newton-methods because it takes a small amount of memory. It uses a line-search approach, which makes it much steady in training and more accessible to examine for convergence.	It is the most efficient, speedy, and popular approach of optimization compared to previous methods.	It does not scale well while updating and calculating gradients. Hence, there is a necessity of mini-batch-training that needs to estimate gradients on small subsets of data.

3.4 Expansion of Neural Style Transformation

Neural style transfer has an influence ahead art and entertainment. In medicine, the pattern-matching technique helps in the diagnostic domain and in the design of unique molecules and proteins. NST techniques are already becoming popular in entertainment and social communication. Some websites allow users to create their artwork using different photographs. There is a rising concern in style tools that enhance the field of digital art. However, the real strength of this procedure goes ahead of image creation. Similarly, video, audio/music style transformation has also made some progress.

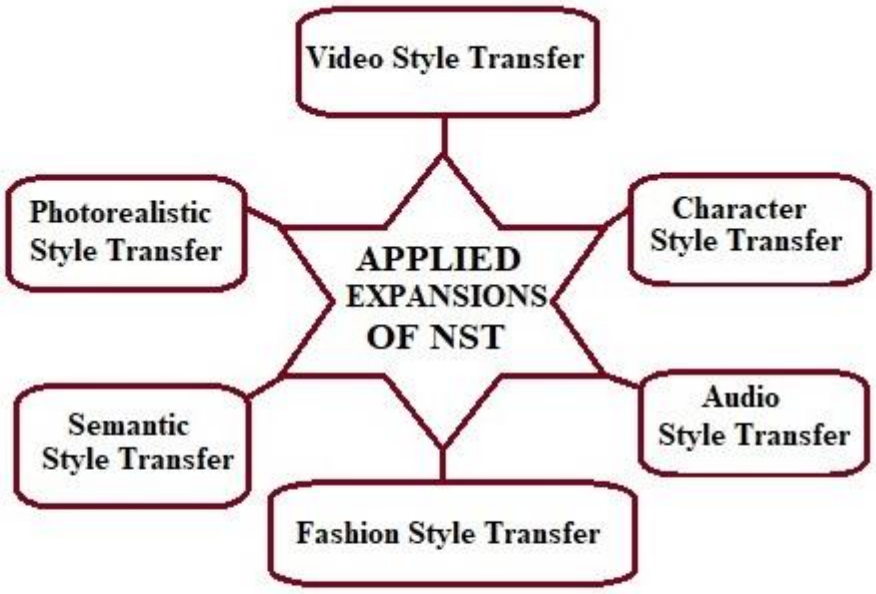

Fig.3.5: Expansion of NST in various applications

3.4.1 Video-Style Transfer

Several approaches of NST are expanded to style transformation of audio, video, and characters, etc. For altering the style of videos, the requirement is to achieve a smooth variation among contiguous frames of video. A method [25] based on iterative optimization of images uses temporal-consistency loss obtained by calculating optical-flow. It produces stylized videos by excluding temporal-artifacts but requires significant time for computing results. Another approach [26] was suggested that applies a flow-subnetwork to compose feature-flow and to consolidate the knowledge of optical-flow. This model is based on a network that contains a previously trained combination of encoder and decoder.

3.4.2 Audio-Style Transfer

Another expansion is to generate distinct sound effects by allotting the desired style from the target-audios [29]. It is called the style transformation of audio. It uses two approaches based on iterative optimization of audio and optimization of the employed model. One approach iteratively optimizes a noise signal applying back propagation. Another method develops efficiency by transforming audio in a feed-forward manner and can provide the output in real time.

3.4.3 Photorealistic-Style Transfer

Photorealistic transformation [30] refers to the transfer of style of arrangements of colors. Its primary intent is to conserve the fundamental structure of the content-image by utilizing a two stage optimization method, namely stylization and smoothing. Hence, for decreasing the deformation of content-image, it employs photorealism-regularization.

3.4.4 Character-Style Transfer

Style transformation is also operated on characters [28] for producing unique fonts and unusual text-effects. By collectively training the conditional-generative model and an ornamentation-network model, the style transformation of characters can be accomplished.

3.4.5 Semantic-Style Transfer

The semantic transformation [27] of style is an expansion of NST, which is applied to a pair of input images holding analogous content. The method is to obtain a semantic correlation between content-image and style-image. Hence, the style of every region of style-image is applied to the corresponding semantically analogous part of the content-image.

3.4.6 Fashion-Style Transfer

Another expansion of NST approaches is in the field of fashion [31] to create clothes containing the desired styles of fashion. In the synthesized output, the applied GAN-model should protect the basic design of cloth while blending with the given target style.

CHAPTER 4

PROPOSED APPROACH

As there exist a trade-off between the quality of results and the running time of existing algorithms. Hence, we need to maintain a balance between quality and speed. Therefore, in this chapter, we will explain the proposed approach used in arbitrary image style transformation using SANET models. The hardware requirements, software requirements and data set used in training and testing of model is listed along with the description of detailed procedure of implementation.

4.1 Style-Attentional Network Model (SANET)

It stands for the style-attentional network that can blend style-patterns in the content-image efficiently and flexibly. It uses a kernel that can learn similarities rather than fixed kernel. This neural-network model modifies the self-attention mechanism to understand the mapping between content and style features.

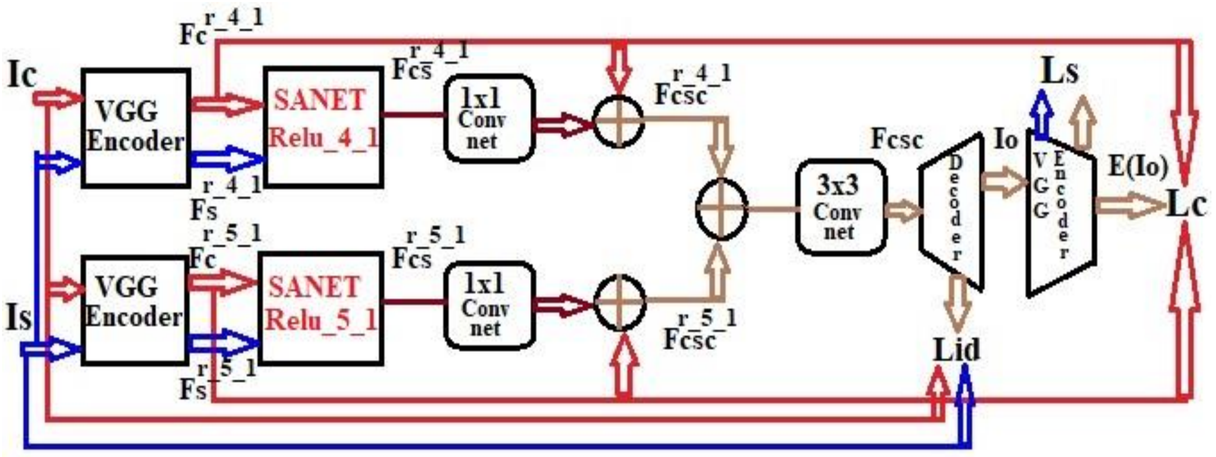

Fig.4.1: Network Architecture of SANET model

For synthesizing artworks, this model takes two images in input, which are content-image I_c and style-image I_s. By applying the encoder module, which is a previously trained VGG nineteen network, both the input-images are encoded into an intermediate representation. After encoding both input-images, we apply both the obtained feature-maps F_c and F_s, to two distinct style attentional networks to produce the combined feature-map F_{cs}.

$$F_c = E(I_c) \tag{4.1}$$

$$F_s = E(I_s) \tag{4.2}$$

$$F_{cs}^{r41} = SANET_{r41}(F_c, F_s) \tag{4.3}$$

$$F_{cs}^{r51} = SANET_{r51}(F_c, F_s) \tag{4.4}$$

$$F_{csc}^{r41} = F_c + W_{cs}F_{cs}^{r41} \tag{4.5}$$

$$F_{csc}^{r51} = F_c' + W_{cs}'F_{cs}^{r51} \tag{4.6}$$

$$F_{csc} = conv_{3\times3}\left(F_{csc}^{r41} + upsamle(F_{csc}^{r51})\right) \tag{4.7}$$

$$I_o = D(F_{csc}) \tag{4.8}$$

Here F_{csc}^{r41} and F_{csc}^{r51} are the synthesized feature-map values generated by applying 1x1 conv-net to the output feature-maps of SANETs. F_{csc} is the integrated output feature-map achieved by applying 3x3 conv-net to the resulting feature-maps obtained by relu_4_1 SANET and upsampled relu_5_1 SANET. The combined output of these SANETs is concatenated by passing through a symmetric-decoder, which produces the final output-image I_o.

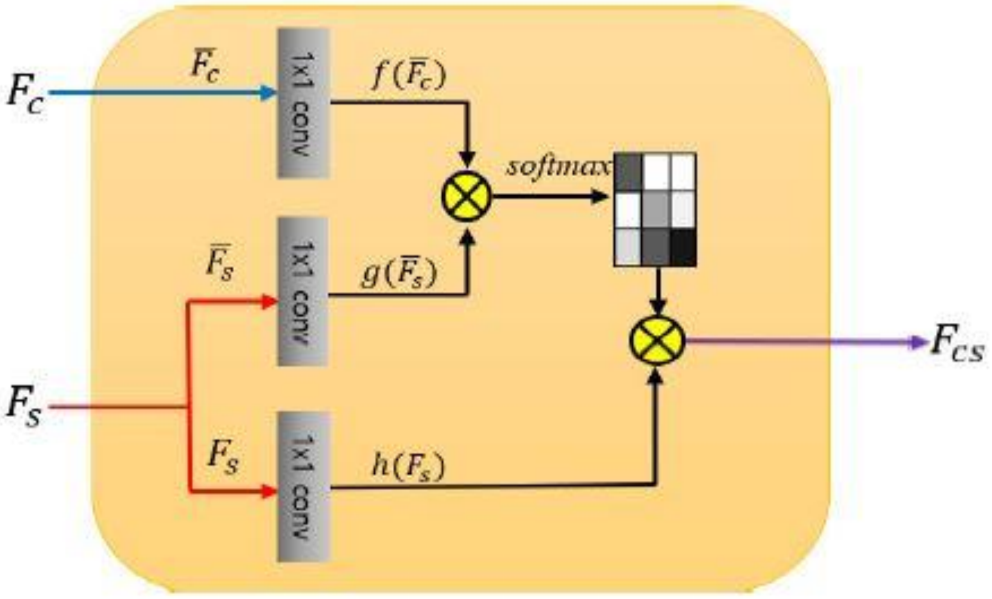

Fig.4.2: Internal Architecture of SANET model

4.2 Calculation of Loss Functions

This method is both effective and practically efficient. It flexibly performs style transformation using the loss function that unites traditional style-reconstruction-losses and the newly introduced identity-loss.

4.2.1 Content-Loss Function

Content-loss represents the decline of spatial-structure and arrangement of content during the formation of resulting output-image. It is calculated in terms of euclidean distance.

$$L_c = \left\| \overline{E(I_o)^{r41}} - \overline{F_c^{\,r41}} \right\|^2 + \left\| \overline{E(I_o)^{r51}} - \overline{F_c^{\,r51}} \right\|^2 \tag{4.9}$$

Here, $\overline{F_c^{\,r41}}$ and $\overline{F_c^{\,r51}}$ are channel-wise-normalized mean, variance feature-map values of content-image. $\overline{E(I_o)^{r41}}$ and $\overline{E(I_o)^{r51}}$ are channel-wise-normalized mean, variance feature-map values of output-image after passing through the encoder. The euclidean distance computed between these gives the value of content-loss.

4.2.2 Style-Loss Function

Style-loss represents the lack of style characteristics and features during the formation of the resulting output-image. It is defined as follows,

$$L_s = \sum_{i=1}^{L} \left\| \mu(\varphi_i(I_o)) - \mu(\varphi_i(I_s)) \right\|^2 + \left\| \sigma(\varphi_i(I_o)) - \sigma(\varphi_i(I_s)) \right\|^2 \tag{4.10}$$

Here, φ is the feature-map value of the layer of encoder denoted by i. The relu_1_1, relu_2_1, relu_3_1, relu_4_1, and relu_5_1 layers are employed with similar weights. The gram-matrix loss and the AdaIN style-loss are applied, in which AdaIN style loss is very satisfying. The SANET model is trained by examining only the global-statistics of the style loss L_s.

4.2.3 Identity-Loss Function

The new identity-loss function is defined for considering both the global-statistics and the semantically local-mapping between the content-features and the style characteristics.

$$L_{id} = w_{id1}(\|I_{cc} - I_c\|^2 + \|I_{ss} - I_s\|^2) + w_{id2} \sum_{i=1}^{L} \begin{pmatrix} \|\varphi_i(I_{cc}) - \varphi_i(I_c)\|^2 \\ +\|\varphi_i(I_{ss}) - \varphi_i(I_s)\|^2 \end{pmatrix} \quad (4.11)$$

The L_{id} is identity-loss and w_{id1}, w_{id2} are the weights assigned to identity-loss. This loss function helps to preserve the structure of content-image while simultaneously maintains the style-patterns.

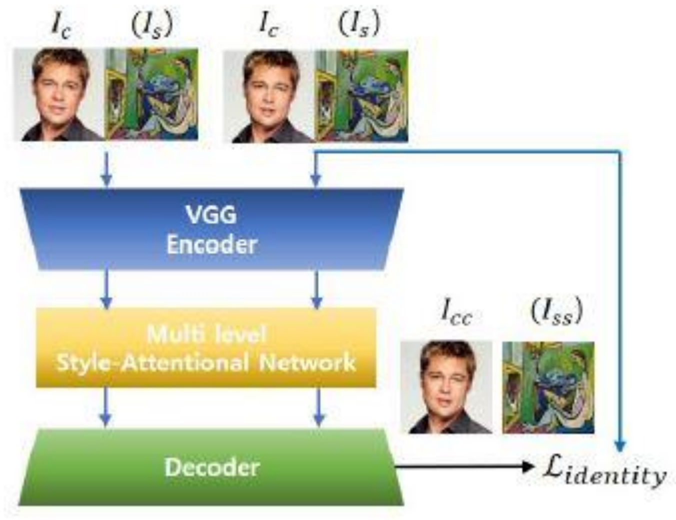

Fig.4.3: Calculation of Identity-Loss Function

4.2.4 Total-Loss Function

The total-loss function is a linear combination of content-loss, style-loss, and identity-loss functions. An encoder, which is a previously trained VGG 19 network, is employed in calculating the loss function during the training of SANET and decoder.

$$L = w_c L_c + w_s L_s + L_{id} \quad (4.12)$$

Here, L_c, L_s, and L_{id} represent the content-loss, style-loss, and identity-loss functions, respectively. w_c and w_s are the weights assigned to content-loss and style-loss respectively.

4.3 Implementation Details

Below in this section, the hardware requirements, software requirements and data set used in training and testing of model is listed along with the description of detailed procedure of implementation.

4.3.1 Hardware Requirements

The network of the SANET model is implemented with the following hardware requirements.

- A PC comprising Windows 10 operating system
- Intel core i5 eighth-generation quad-core CPU
- NVIDIA GeForce-GTX 1050 GPU
- A RAM of size 8 gigabytes
- It also demands an internet of extremely high-speed.

4.3.2 Software Requirements

The network of the SANET model is implemented with the following software requirements.

- We have used a jupyter-notebook under the Anaconda navigator. This IDE is the most suitable for machine learning programming as it provides many features to control codes that make it easy to write and debug. Finding the errors in code is very easy. It also helps in minimizing code redundancy.
- PyTorch 0.4.1 is a free and open-source library for python programming that helps to build projects in deep learning.
- CUDA 9.2 provides parallel programming for general computing using GPU. It facilitates programmers to increase the speed of complex computations by improving GPUs' power for the parallelization.
- Open CV 3.4.2 is a very optimized-library that focuses on real-life problem-solving. It is a python machine learning library and used for image processing problems.
- Programming Language used is python. It is easy to use and understand. Due to an extensive collection of inbuilt functions and a very easy coding environment, it has become prevalent among developers.

4.3.3 Data Set Used

The network of the SANET model is implemented with the following Data Set requirements. During the training of the model,

- For content-images VOC 2012 dataset is used, which is an extensive collection of classified images and also used by many other tasks of computer vision. This dataset can be found on pjreddie.com under the folder of projects which contain subfolder pascal-voc-dataset-mirror. It has many sets of images, which is demonstrated in the Table 4.1 given below.
- For the style-images WikiArt dataset is used, which is open source and can be found on the github repository. It also has a huge amount of data, which is composed of a lot of pictures with so many variations for the task of training and validation.

Table 4.1: VOC 2012 Data Set

	Train		Val		Train-Val		Test	
	image	object	image	object	image	object	image	object
Airplane	327	432	343	433	670	865	-	-
Bicycle	268	353	284	358	552	711	-	-
Bird	395	560	370	559	765	1119	-	-
Boat	260	426	248	424	508	850	-	-
Bottle	365	629	341	630	706	1259	-	-
Bus	213	292	208	301	421	593	-	-
Car	590	1013	571	1004	1161	2017	-	-
Cat	539	605	541	612	1080	1217	-	-
Chair	566	1178	553	1176	1119	2354	-	-
Cow	151	290	152	298	303	588	-	-
Dining-table	269	304	269	305	538	609	-	-
Dog	632	756	654	759	1286	1515	-	-
Horse	237	350	245	360	482	710	-	-
Motor-bike	265	357	261	356	526	713	-	-
Person	1994	4194	2093	4372	4087	8566	-	-
Potted-plant	269	484	258	489	527	973	-	-
sheep	171	400	154	413	325	813	-	-
Sofa	257	281	250	285	507	566	-	-
Train	273	313	271	315	544	628	-	-
TV monitor	290	392	285	392	575	784	-	-
Total	5717	13609	5823	13841	11540	27450	-	-

4.3.4 Description of Implementation

Below in this section, the description of implementation of network of SANET model, its training and testing are explained.

- We have taken input as two preprocessed images and resized them to 512×512.
- During training, the size of each batch is six, and 180000 default iterations are applied.
- The initial values assigned to weights while computing the loss are as follows, w_c=1.0, w_s=4.0, w_{id1}=50, and w_{id2}=1.
- We have perceived that if the weight w_c is increased in the absence of identity-loss function, then spatial-distribution and structure of content-image can be protected but with a lack of style-features. However, if the weights for identity-loss function are increased, then it's possible to preserve the semantic-arrangement of content along with sustaining the style-features.
- Consequently, we estimated the total loss by incorporating identity-loss along with content and style losses.
- We applied Adam optimizer to minimize the value of loss function.
- The encoder uses a previously trained VGG nineteen network.
- Two SANET models are jointly trained and integrated by taking the VGG feature-maps as inputs and encodes output feature-maps using distinct layers Relu_4_1 and Relu_5_1.
- Outputs of these SANETs are combined by using the symmetric decoder.
- After that, the trained network of SANET models is applied to the pictures taken by us in our college campus to examine the execution time and quality of the obtained results.

4.3.5 Steps of Implementation

Below in this section, the steps of implementation of network of SANET model, its training and testing are explained.

1. We are importing files, which will help to initialize values and unzip files of datasets.

```
from google.colab import files
            uploaded=files.upload()
            for fn in uploaded.keys():
            print('User uploaded file "{name}" with length {length} bytes'.format(
            name=fn, length=len(uploaded[fn])))
```

2. We are finding the standard mean, mean_variance_norm, and flatten mean standard values by using inbuilt standard deviation functions such as mean.expand() and std.expand(), etc. After that, we are finding the average value of a large set of numbers. A small value is also added to the variance to avoid exceptions during the calculation. Here we are using 3d feat for storing values inside array within channels and returning the results stored in feat_mean, feat_std, normalized_feat, feat_flatten, mean & std.

```
import torch
def calc_mean_std(feat, eps=1e-5):
size = feat.size()
assert (len(size) == 4)
N, C = size[:2]
feat_var = feat.view(N, C, -1).var(dim=2) + eps
feat_std = feat_var.sqrt().view(N, C, 1, 1)
feat_mean = feat.view(N, C, -1).mean(dim=2).view(N, C, 1, 1)
return feat_mean, feat_std

def mean_variance_norm(feat):
size = feat.size()
mean, std = calc_mean_std(feat)
normalized_feat = (feat - mean.expand(size)) / std.expand(size)
return normalized_feat

def _calc_feat_flatten_mean_std(feat):
assert (feat.size()[0] == 3)
assert (isinstance(feat, torch.FloatTensor))
feat_flatten = feat.view(3, -1)
mean = feat_flatten.mean(dim=-1, keepdim=True)
std = feat_flatten.std(dim=-1, keepdim=True)
return feat_flatten, mean, std
```

3. In this, we are defining the weights of decoder by importing the torch as nn and calculating relu() by converting it into 2d dimension followed by upsampling where scale_factor is used 2 and mode as nearest.

```
import torch.nn as nn
        decoder = nn.Sequential(
        nn.ReflectionPad2d((1, 1, 1, 1)),
        nn.Conv2d(512, 256, (3, 3)),
        nn.ReLU(),
        nn.Upsample(scale_factor=2, mode='nearest'),
        nn.ReflectionPad2d((1, 1, 1, 1)),
        nn.Conv2d(256, 256, (3, 3)),
        nn.ReLU()
```

4. In this section, vgg using three relu layer with each having two sub-layers are being calculated to perform dimension based on reflection. Dimension values are default, and max pool values with ceil_mode equal to true is used.

```
vgg = nn.Sequential(
            nn.Conv2d(3, 3, (1, 1)),
            nn.ReflectionPad2d((1, 1, 1, 1)),
            nn.Conv2d(3, 64, (3, 3)),
            nn.ReLU(),
            nn.ReflectionPad2d((1, 1, 1, 1)),
            nn.Conv2d(64, 64, (3, 3)),
            nn.ReLU(),
            nn.MaxPool2d((2, 2), (2, 2), (0, 0), ceil_mode=True),
            nn.ReflectionPad2d((1, 1, 1, 1)),
            nn.Conv2d(64, 128, (3, 3)),
```

5. We are defining a class called Sanet, which has two methods first, one is used to initialize the plane values based on image and passing arguments to the superclass to satisfy the dependency. The second method forward is used to calculate the mean and variance of content and style image based on a permutation of four side calculations, and in last, all values are used in calculation to make the initial phase object structure.

```
class Transform(nn.Module):
            def __init__(self, in_planes):
               super(Transform, self).__init__()
               self.sanet4_1 = SANet(in_planes = in_planes)
               self.sanet5_1 = SANet(in_planes = in_planes)
               self.upsample5_1 = nn.Upsample(scale_factor=2, mode='nearest')
               self.merge_conv_pad = nn.ReflectionPad2d((1, 1, 1, 1))
               self.merge_conv = nn.Conv2d(in_planes, in_planes, (3, 3))
               def forward(self, content4_1, style4_1, content5_1, style5_1):
               return self.merge_conv(self.merge_conv_pad
                     (self.sanet4_1(content4_1, style4_1) +
               self.upsample5_1(self.sanet5_1(content5_1, style5_1))))
class Net(nn.Module):
              def __init__(self, encoder, decoder, start_iter):
              super(Net, self).__init__()
              enc_layers = list(encoder.children())
              self.enc_1 = nn.Sequential(*enc_layers[:4])  # input -> relu1_1
              self.enc_2 = nn.Sequential(*enc_layers[4:11])  # relu1_1 -> relu2_1
              self.enc_3 = nn.Sequential(*enc_layers[11:18])  # relu2_1 -> relu3_1
              self.enc_4 = nn.Sequential(*enc_layers[18:31])  # relu3_1 -> relu4_1
              self.enc_5 = nn.Sequential(*enc_layers[31:44])  # relu4_1 -> relu5_1
              self.transform = Transform(in_planes = 512)
self.decoder = decoder
if(start_iter > 0):
self.transform.load_state_dict(torch.load('transformer_iter_' + str(start_iter)
+ '.pth'))
self.decoder.load_state_dict(torch.load('decoder_iter_' + str(start_iter) + '.pth'))
self.mse_loss = nn.MSELoss()
for name in ['enc_1', 'enc_2', 'enc_3', 'enc_4', 'enc_5']:
for param in getattr(self, name).parameters():
param.requires_grad = False
```

6. The encoder uses a previously trained VGG nineteen network. Two SANET models are jointly trained and integrated by taking the VGG feature-maps as inputs and encodes output feature-maps using distinct layers Relu_4_1 and Relu_5_1. Outputs of these SANETs are combined by using the symmetric decoder. After that, the trained network of SANET models is applied to the pictures. We are passing four arguments in initialization function, and upon that, we are performing the transformation of encoder and decoder.

```
def encode_with_intermediate(self, input):
                results = [input]
                for i in range(5):
                    func = getattr(self, 'enc_{:d}'.format(i + 1))
                    results.append(func(results[-1]))
                return results[1:]

def calc_content_loss(self, input, target, norm = False):
                if(norm == False):
                  return self.mse_loss(input, target)
                else:
                  return self.mse_loss(mean_variance_norm(input),
                           mean_variance_norm(target))

def calc_style_loss(self, input, target):
                input_mean, input_std = calc_mean_std(input)
                target_mean, target_std = calc_mean_std(target)
                return self.mse_loss(input_mean, target_mean) + \
                 self.mse_loss(input_std, target_std)
```

7. In the process of generating output, we need to calculate content loss & style loss by calling the mean_variance normalization method on input content-image and style-image after applying loss functions. Content picture loss ensures the activation of higher layers are same as the input picture in the output image. Style picture loss provides that the output image should adopt the style image functionality adequately.

```
class SANet(nn.Module):
super(SANet, self).__init__()
self.f = nn.Conv2d(in_planes, in_planes, (1, 1))
self.g = nn.Conv2d(in_planes, in_planes, (1, 1))
self.h = nn.Conv2d(in_planes, in_planes, (1, 1))
self.sm = nn.Softmax(dim = -1)
self.out_conv = nn.Conv2d(in_planes, in_planes, (1, 1))

def forward(self, content, style):
F = self.f(mean_variance_norm(content))
G = self.g(mean_variance_norm(style))
H = self.h(style)
b, c, h, w = F.size()
F = F.view(b, -1, w * h).permute(0, 2, 1)
b, c, h, w = G.size()
G = G.view(b, -1, w * h)
S = torch.bmm(F, G)
S = self.sm(S)
b, c, h, w = H.size()
H = H.view(b, -1, w * h)
O = torch.bmm(H, S.permute(0, 2, 1))
b, c, h, w = content.size()
O = O.view(b, c, h, w)
O = self.out_conv(O)
O += content
return O
```

8. We have taken input as two preprocessed images and resized them to 512×512. During training, the size of each batch is six, and 180000 default iterations are applied. The initial values assigned to weights while computing the loss are as follows, w_c=1.0, w_s=4.0, w_{id1}=50, and w_{id2}=1. After defining all dimensions upon the content-image and style-image to produce output-image, the parser method parameter is passed.

```
parser.add_argument('--lr', type=float, default=1e-4)
parser.add_argument('--lr_decay', type=float, default=5e-5)
parser.add_argument('--max_iter', type=int, default=180000)
parser.add_argument('--batch_size', type=int, default=5)
parser.add_argument('--style_weight', type=float, default=4.0)
parser.add_argument('--content_weight', type=float, default=1.0)
parser.add_argument('--n_threads', type=int, default=50)
parser.add_argument('--save_model_interval', type=int, default=1000)
parser.add_argument('--start_iter', type=float, default=0)
args = parser.parse_args('')
```

9. We estimated the total loss by incorporating identity-loss along with content and style losses.

```
if(args.start_iter > 0):
optimizer.load_state_dict(torch.load('optimizer_iter_' + str(args.start_iter) + '.pth'))

for i in tqdm(range(args.start_iter, args.max_iter)):
adjust_learning_rate(optimizer, iteration_count=i)
content_images = next(content_iter).to(device)
style_images = next(style_iter).to(device)
loss_c, loss_s, l_identity1, l_identity2 = network(content_images, style_images)
loss_c = args.content_weight * loss_c
loss_s = args.style_weight * loss_s
loss = loss_c + loss_s + l_identity1 * 50 + l_identity2 * 1

optimizer.zero_grad()
loss.backward()
optimizer.step()
```

10. We need to import all required python libraries which need to test the model.

```
import argparse
import os
import torch
import torch.nn as nn
from PIL import Image
from os.path import basename
from os.path import splitext
from torchvision import transforms
from torchvision.utils import save_image
```

11. We are calculating standard mean based on feat size, whereas the default dimension size is two for the variance. By default, we also add a small value to eps to avoid errors during the program's execution.

```
def calc_mean_std(feat, eps=1e-5):
size = feat.size()
assert (len(size) == 4)
N, C = size[:2]
feat_var = feat.view(N, C, -1).var(dim=2) + eps
feat_std = feat_var.sqrt().view(N, C, 1, 1)
feat_mean = feat.view(N, C, -1).mean(dim=2).view(N, C, 1, 1)
return feat_mean, feat_std
```

12. We are calculating variance normalization by using standard deviation of inbuilt functions.

```
def mean_variance_norm(feat):
size = feat.size()
mean, std = calc_mean_std(feat)
normalized_feat = (feat - mean.expand(size)) / std.expand(size)
return normalized_feat
```

13. The transform class defines an initialization method for sanet4_1 and 5_1, whereas dimensions are as scale_factor is 2, and mode is nearest, which provides module-based output values.

```
class Transform(nn.Module):
def __init__(self, in_planes):
super(Transform, self).__init__()
self.sanet4_1 = SANet(in_planes = in_planes)
self.sanet5_1 = SANet(in_planes = in_planes)
self.upsample5_1 = nn.Upsample(scale_factor=2, mode='nearest')
self.merge_conv_pad = nn.ReflectionPad2d((1, 1, 1, 1))
self.merge_conv = nn.Conv2d(in_planes, in_planes, (3, 3))
```

14. Content and style image for relu 4_1 and relu 5_1 is allocating arguments steps to produce final output image dimensions.

```
with torch.no_grad():
for x in range(args.steps):
print('iteration ' + str(x))
Content4_1 = enc_4(enc_3(enc_2(enc_1(content))))
Content5_1 = enc_5(Content4_1)
Style4_1 = enc_4(enc_3(enc_2(enc_1(style))))
Style5_1 = enc_5(Style4_1)
content = decoder(transform(Content4_1, Style4_1,
Content5_1, Style5_1))
content.clamp(0, 255)
content = content.cpu()
```

15. This gives the final output image after passing through many iterations on content and style image.

```
output_name = '{:s}/{:s}_stylized_{:s}{:s}'.format(
args.output, splitext(basename(args.content))[0],
splitext(basename(args.style))[0], args.save_ext)
save_image(content, output_name)
```

CHAPTER 5

RESULTS AND DISCUSSION

In this chapter, the run-time of some NST approaches on three distinct dimensional images are analyzed and compared. At first, the experimental results of SANET model are demonstrated clearly. After that we will compare the performance of SANET model with other ASPM approaches described earlier.

5.1 Analysis of various approaches of NST

We have thoroughly examined the characteristics of various distinct procedures for neural style fusion and transformation. The charts given below in Fig.5.1, Fig.5.2, and Fig.5.3 represent the comparison of running time performance on input-images of three distinct dimensions.

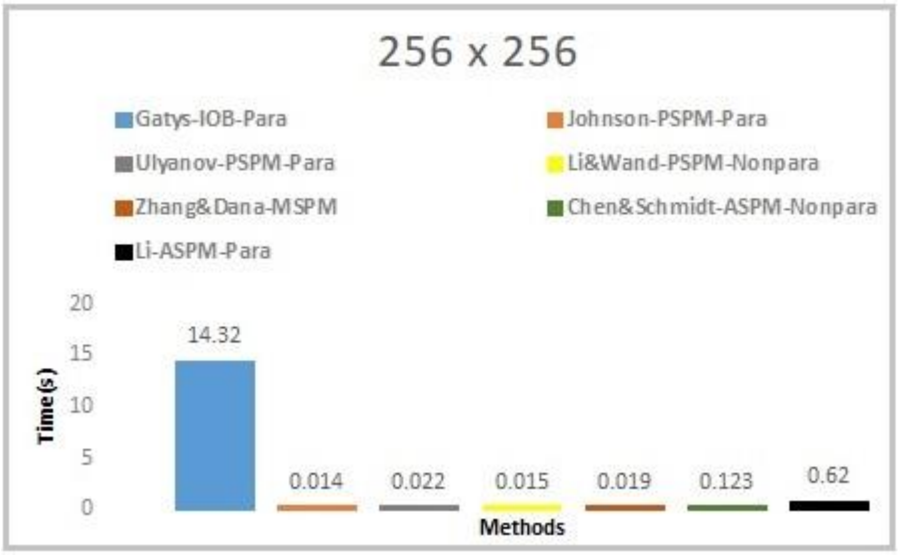

Fig.5.1: Comparison of run-time of NST approaches on 256×256 dimensional images

We can see that the algorithm based on optimization of image proposed by Gatys et al. [1] takes a considerable value of computation time when compared with offline methods based on optimization of employed models [3] [4] [5]. However, the quality of generated results is better when compared to others. For real-time applications of arbitrary style transformation, there is a trade-off between the quality of results and the running time of existing algorithms. Therefore, we need to maintain a balance between quality and speed.

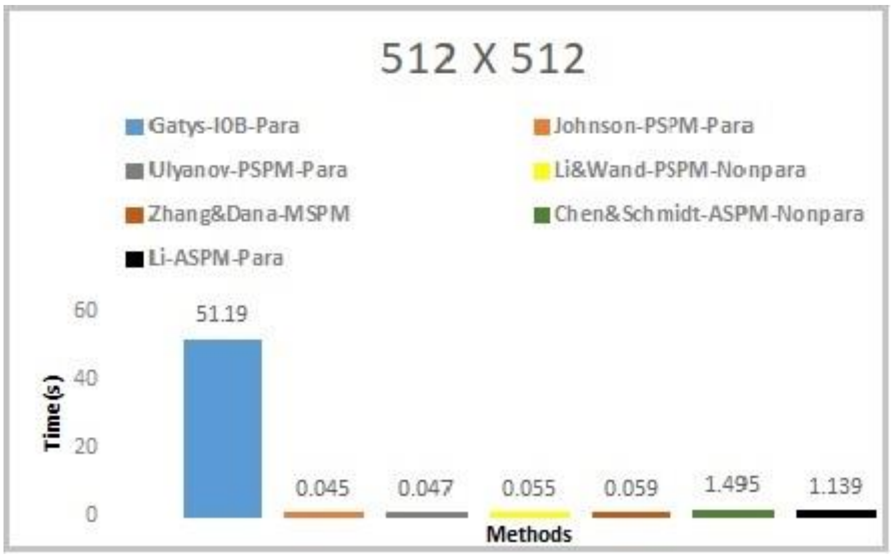

Fig.5.2: Comparison of run-time of NST approaches on 512×512 dimensional images

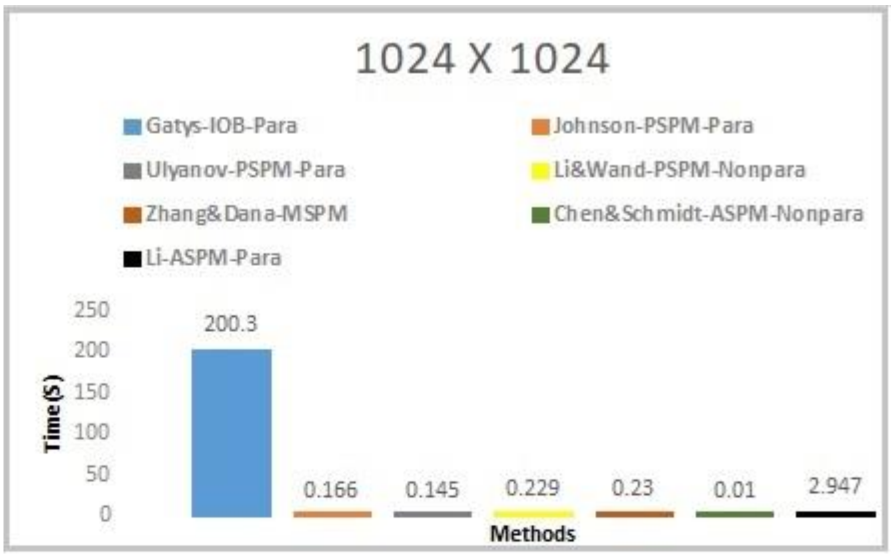

Fig.5.3: Comparison of run-time of NST approaches on 1024×1024 dimensional images

5.2 Implementation Results

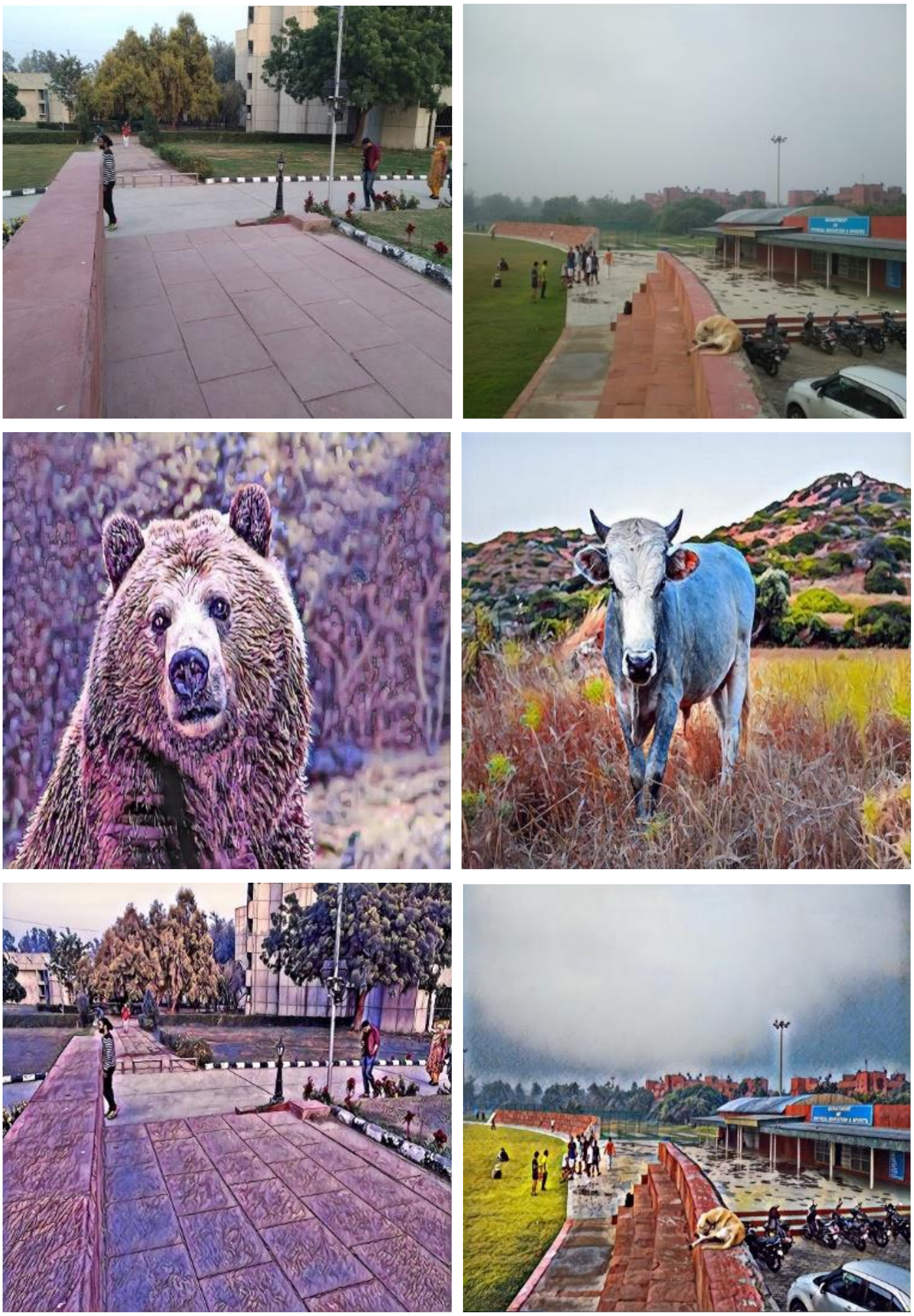

Fig.5.4: Experimental results 1 and 2 of style transformation using SANET model

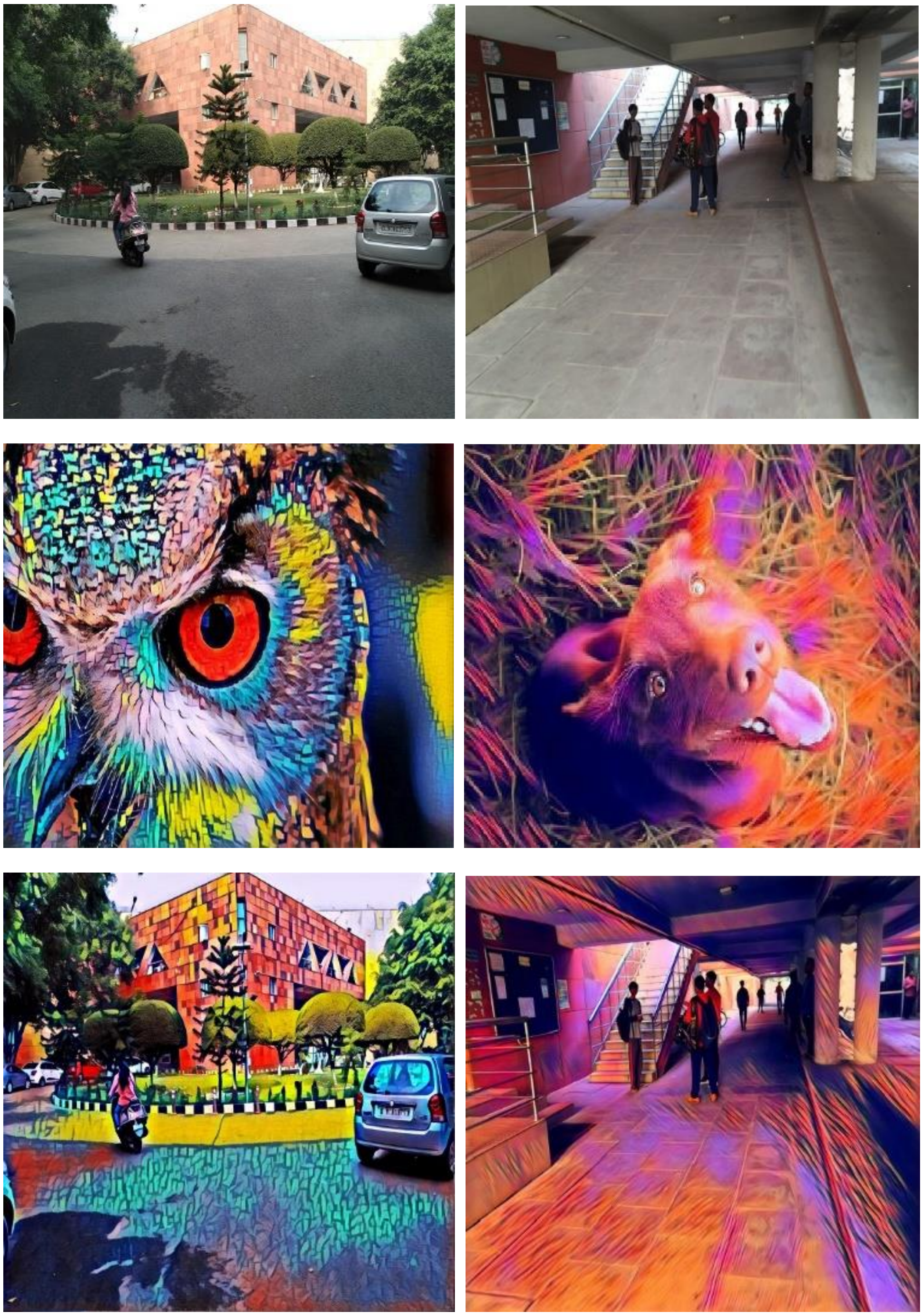

Fig.5.5: Experimental results 3 and 4 of style transformation using SANET model

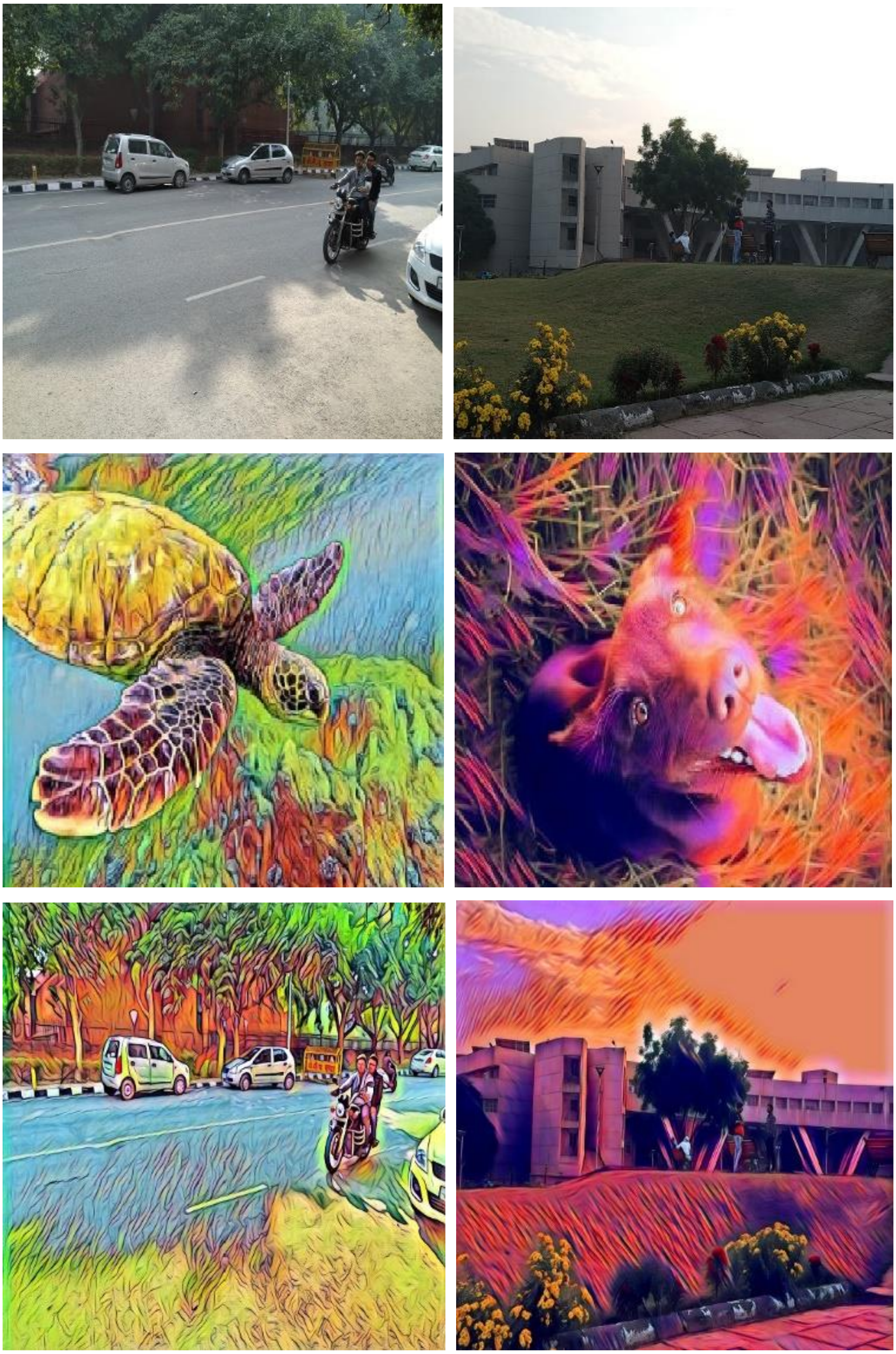

Fig.5.6: Experimental results 5 and 6 of style transformation using SANET model

5.3 Comparison of SANET with other ASPM approaches

For determining the quality of SANET network, we analysed its comparison with four types of models for fusing and transforming arbitrary-styles. We observed that AdaIN [8] is the fastest ASPM approach, but it renders a suboptimal result as it copies the channel wise mean-variance value, and the trained decoder usually attach frequently seen similar patterns and style-textures to all stylized outputs. Hence, its resulting output is less appealing and sometimes fails to retain the distribution of colours.

Despite matching the second-order-statistics optimally in WCT [7], sometimes, it creates unseen and collapsed-patterns. Avatar-Net [9] blends the images based on the semantic-arrangement of content and employs multi scale stylization. Still, because of its dependence on the patch size, it fails to realize the local and global style-patterns simultaneously. Seldom Avatar-Net and WCT render the blurred hair-texture and distorted vision of content due to the collapsed brush strokes. In contradiction, the network containing two integrated SANET models can generate various exquisite styles efficiently.

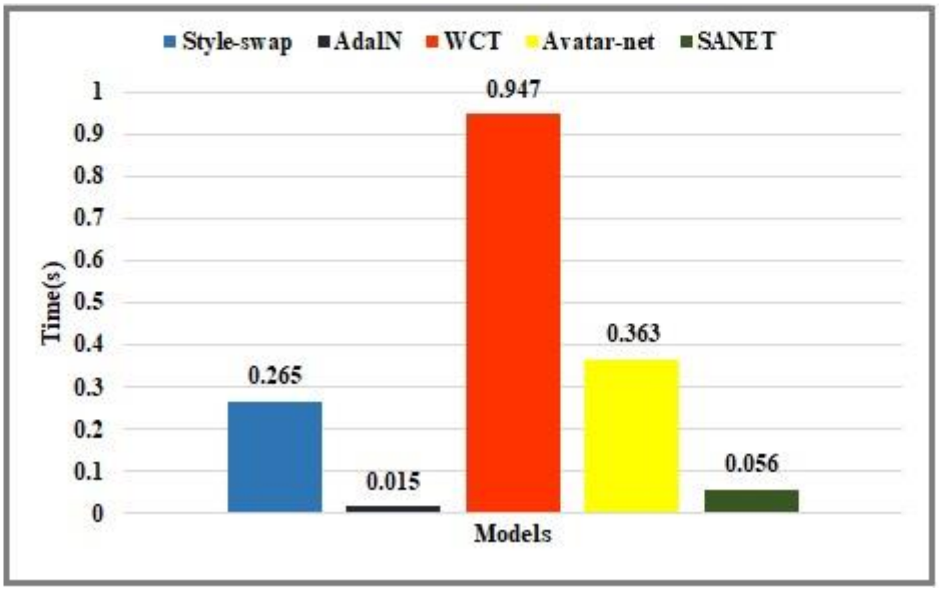

Fig.5.7: Comparison of execution time of SANET with other ASPM models

The graph in Fig.5.7 displays a comparison of the running time of SANET with other existing methods of arbitrary style transformation for images of 512×512 dimensions. SANET is much faster than style-swap, Avatar-net, and WCT models but a bit slower in comparison to AdaIN.

However, the quality of the result surpasses AdaIN and others. Therefore, this method is both effective and practically efficient.

5.4 Conclusion and Future Scope

This project presents an advance network containing a combination of two SANET Models for style fusion and transformation. This network ensures the semantic-arrangement and spatial-distribution of content-image while fusing with the characteristics of new arbitrary style-patterns by computing a new identity-loss component. We compared the execution time of different NST algorithms on input-images of distinct dimensions. We observed that online neural models are slow but deliver better results in comparison to fast offline neural models. In all existing approaches, there is a trade-off between the quality of results and the running time of existing algorithms. Our results demonstrate that this model is much faster than other ASPM models and produces results with better quality and flexibility, which makes it much effective and practically efficient. We are planning to expand this model for the style transformation of videos in the future.

CHAPTER 6
REFERENCES

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2414–2423, 2016.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” in ArXiv e-prints, Aug. 2015.

[3] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for realtime style transfer and super-resolution,” in European Conference on Computer Vision, pp. 694–711, 2016.

[4] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” in International Conference on Machine Learning, pp. 1349–1357, 2016.

[5] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6924–6932, 2017.

[6] T. Q. Chen and M. Schmidt, “Fast patch-based style transfer of arbitrary style,” in Proceedings of the NIPS Workshop on Constructive Machine Learning, 2016.

[7] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, “Universal style transfer via feature transforms,” in Advances in Neural Information Processing Systems, pp. 385–395, 2017.

[8] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” in Proceedings of the IEEE International Conference on Computer Vision, pp. 1501–1510, 2017.

[9] L. Sheng, Z. Lin, J. Shao, and X. Wang. Avatar-Net: Multiscale zero-shot style transfer by feature decoration. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8242–8250, 2018.

[10] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg, “State of the ‘art’: A taxonomy of artistic stylization techniques for images and video,” IEEE transactions on visualization and computer graphics, vol. 19, no. 5, pp. 866–885, 2013.

[11] A. Hertzmann, “Painterly rendering with curved brush strokes of multiple sizes,” in Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, pp. 453–460, 1998.

[12] B. Gooch, G. Coombe, and P. Shirley, “Artistic vision: painterly rendering using computer vision techniques,” in Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering, ACM pp. 83–ff, 2002.

[13] Y.-Z. Song, P. L. Rosin, P. M. Hall, and J. Collomosse, "Arty shapes," in Proceedings of the Fourth Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging, Eurographics Association, pp. 65–72, 2008.

[14] H. Winnemoller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," in ACM Transactions On Graphics (TOG), vol. 25, no. 3. ACM, pp. 1221–1226, 2006.

[15] M. Zhao and S.-C. Zhu, "Portrait painting using active templates," in Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering. ACM, pp. 117–124, 2011.

[16] Y. Li, N. Wang, J. Liu, and X. Hou, "Demystifying neural style transfer," in Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 2230–2236, 2017.

[17] S. Li, X. Xu, L. Nie, and T.-S. Chua, "Laplacian-steered neural style transfer," in Proceedings of the 2017 ACM on Multimedia Conference. ACM, pp. 1716–1724, 2017.

[18] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2479–2486, 2016.

[19] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in European Conference on Computer Vision, pp. 702–716, 2016.

[20] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in International Conference on Learning Representations, 2017.

[21] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1897–1906, 2017.

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

[23] D. Y. Park and K. H. Lee., "Arbitrary style transfer with style-attentional networks," arXiv preprint arXiv:1812.02342, 2018.

[24] S. Ruder, "An overview of gradient descent optimization algorithms" arXiv preprint arXiv:1609:04747, 2017.

[25] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," in German Conference on Pattern Recognition, pp. 26–36, 2016.

[26] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," in Proceedings of the IEEE International Conference on Computer Vision, pp. 1105–1114, 2017.

[27] A. J. Champandard, "Semantic style transfer and turning two-bit doodles into fine artworks," ArXiv e-prints, Mar. 2016.

[28] S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, and T. Darrell, “Multi-content gan for few-shot font style transfer,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[29] P. Verma and J. O. Smith, “Neural style transfer for audio spectograms,” in Proceedings of the NIPS Workshop on Machine Learning for Creativity and Design, 2017.

[30] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, “A closed-form solution to photorealistic image stylization,” in European Conference on Computer Vision, 2018.

[31] S. Jiang and Y. Fu, “Fashion style generator,” in Proceedings of the 26th International Joint Conference on Artificial Intelligence. AAAI Press, pp. 3721–3727, 2017.

[32] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, M. Song. "Neural Style Transfer: A Review", IEEE Transactions on Visualization and Computer Graphics, 2019.

CHAPTER 7

PUBLICATION

[1] Pratibha Rathi, Pranav Adarsh, Dr. Manoj Kumar. "Deep Learning Approach for Arbitrary Image Style Fusion and Transformation using SANET model", in Proceedings of the Fourth International Conference on Trends in Electronics and Informatics (ICOEI), IEEE, 2020.

DVD Part Number: CFP20J32-DVD; ISBN: 978-1-7281-5517-3; Page No. 1036-1043.

978-1-7281-5518-0/20/$31.00 ©2020 IEEE

For real-time applications of arbitrary style transformation, there is a trade-off between the quality of results and the running time of existing algorithms. Hence, it is required to maintain the equilibrium of the quality of generated artwork with the speed of execution. It's complicated for the present arbitrary style-transformation procedures to preserve the structure of content-image while blending with the design and pattern of style-image. This project presents the implementation of a network using SANET models for generating impressive artworks. It is flexible in the fusion of new style characteristics while sustaining the semantic-structure of the content-image. The identity-loss function helps to minimize the overall loss and conserves the spatial-arrangement of content. The results demonstrate that this method is practically efficient, and therefore it can be employed for real-time fusion and transformation using arbitrary styles.