Generating Artistic Images Through Neural StyleTransfer Using Machine Learning and Image Manipulation Techniques

Author 1 :**Lajapathi Durai R A**, Post Graduate Student, Embedded Systems ,
Hindusthan College of Engineering and Technology,Coimbatore,

Co Author 2 :**Mr. Joshua Daniel** M.E., Assistant Professor, Department of Electrical and Electronics Engineering, Hindusthan College of Engineering and Technology, Coimbatore,

Co Author 3 : **Dr. K Mathan** M.E. Ph.D., Professor, Department of Electrical and Electronics Engineering, Hindusthan College of Engineering and Technology, Coimbatore.

ABSTRACT:

Neural Style Transfer (NST) is a computational technique that merges the content of one image with the stylistic characteristics of another, creating visually appealing compositions. Leveraging deep learning and image processing, this method separates andcombines the content and style of input images using convolutional neural networks (CNNs). The content representation of an image is captured by the high-level features extracted from a pre-trained CNN, while its style is characterized by the statistical correlations of features at multiple network layers. It explores the fusion of machine learning and image processing techniques in NST. Initially introduced by Gatys et al., NST has gained popularity due to its ability to generate artistic renditions and novel visualizations. The process involves extracting content and style features from separate images and optimizing a generated image to minimize differences in content and style representations. In this work, explored into the underlying concepts and technical aspectsof NST. This proceedings discussed the utilization of pre-trained CNN models like VGG19 or ResNet for feature extraction and how optimization algorithms, such as gradient descent, iteratively refine the generated image to achieve a fusion of content and style. Additionally, This development has been examined the impact of hyper parameters and layer choices on the quality of stylized outputs.Mainly this technology focuses on reduction of total variation loss, which we incur during the style transfer.These proceedings has carried on work with VGG19, XCEPTION, EfficientnetB7. These findings show how Convolutional Neural Networks can learn deep image representations and show how they may be used for sophisticated image synthesis andmodification.

CHAPTER 1: INTRODUTION

1.1. MOTIVATION OF THE PROJECT

The motivation behind Neural Style Transfer (NST) lies in its ability to merge the content of one image with the artistic style of another, creating aesthetically pleasing and unique visual compositions. This technique was primarily motivated by the desire to combine the capabilities of deep learning with artistic expression, enabling the generation of novel and creative images. Here are several key motivations behind the development and interest in Neural Style Transfer:

Artistic Expression and Creativity:

NST allows for the fusion of artistic styles, enabling the creation of visually appealing images that blend the content of one image with the textures, colors, and patterns characteristic of another. This intersection between machine learning and art motivates artists, designers, and researchers to explore new ways of visual expression.

Visual Content Transformation:

The technique offers a means to transform and enhance visual content. It allows for the generation of stylized images that can be used for various purposes, such as generating appealing visuals for advertising, content creation, or artistic projects.

Exploration of Deep Learning Capabilities:

NST provides an avenue to explore the capabilities of deep learning models, particularly convolutional neural networks (CNNs), in understanding and synthesizing artistic styles. It demonstrates the potential of CNNs beyond traditional classification and recognition tasks.

Bridging Art and Technology:

Bringing together art and technology, NST bridges the gap between machine learning and creative arts. It facilitates collaborations between artists, technologists, and researchers, fostering innovation in both domains.

Study of Neural Network Representations:

NST aids in understanding the representations learned by neural networks. It showcases how higher-level semantic content and lower-level style features are captured and manipulated within neural networks, contributing to insights in representation learning.

Educational and Research Interest:

The technique serves as an educational tool for understanding complex machine learning concepts. It also sparks research interest in improving the methodology, exploring different architectures, and developing more advanced stylization techniques.

1.2. OBJECTIVE OF THE PROJECT:

One issue with texture transfer is style transfer of style from one image to another. The objective of texture transfer is to create a texture from source image while limiting the texture synthesis to protect the semantic content of target image. There were many algorithms proposed in to achieve this task, they plan on reducing the content loss or style loss of original image. Many algorithms have shown different level of transferring andused low pass filters and utilized different mechanisms like the author Leon A. Gatys who also worked on photorealistic style transfer.

Separating content from style in natural images remains an extremely difficult problem. Deep Convolutional Neural Networks, on the other hand, have recently produced powerful computer vision systems that learn to extract high-level semantic information from natural images. It has been demonstrated that Convolutional Neural Networks trained on specific tasks such as object recognition learn to extract high-level image content in generic feature representations that generalize across datasets and even to other visual information processing tasks such as texture recognition and artistic style classification

.Generic feature representations learned by high-performing Convolutional NeuralNetworks can be used to process and manipulate the content and the data independently.

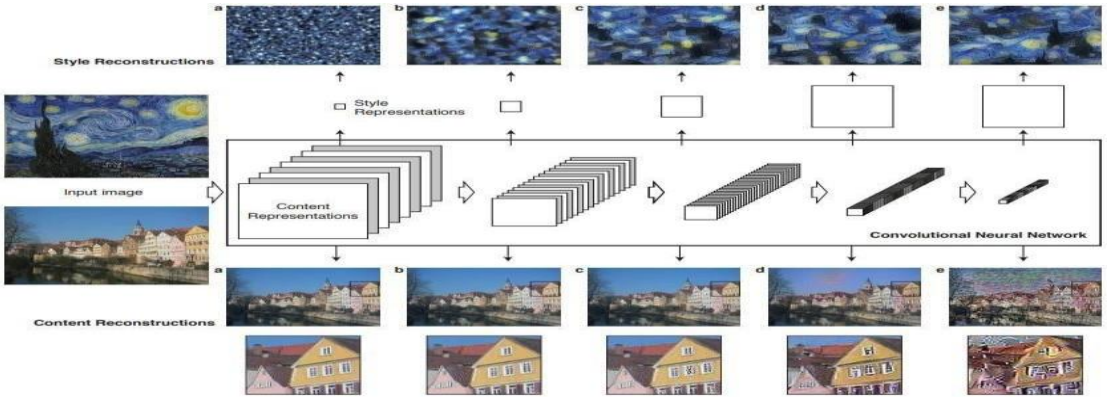

Fig.1.1 Working Of Style Transfer While Extracting Features

The style transfer view is illustrate in figure 1.1 while extracting the features and there are three different neural networks to extract feature information from given style image and paint it on content image without losing its absolute difference between the pixel values. Various works were proposed like stroke-based rendering, a method of placing virtual strokes upon digital canvas to render photograph with style.The goal of this methodology is to depict the prescribed style.

1.3. NEURAL STYLE TRANSFER

Neural style transfer is a technique used in artificial intelligence and computer vision that involves applying the artistic style of one image to another while preserving the content of the latter. This process uses deep neural networks to separate and recombine content and style representations from two different images. Neural Style Transfer (NST) is a fascinating technique in the realm of artificial intelligence andcomputer vision that merges the content of one image with the style of another, producing striking and artistic visuals. This innovative approach leverages deep neural networks to create captivating compositions by separating and merging the content and style elements from distinct images.At its core, NST relies on a delicate interplay between artistic aesthetics and computational prowess. By utilizing convolutional neural networks (CNNs) and leveraging their ability to extract features at various levels of abstraction, NST algorithms dissect and recombine images to produce captivating results.

Key Components of Neural Style Transfer:

- **Content Image:** This is the image whose overall content you want to preserve.
- **Style Image:** This is the image whose artistic style or visual characteristics you want to apply to the content image.
- **Neural Network:** Typically, a pre-trained convolutional neural network (CNN) like VGG or Res Net is used. This network helpsseparate the content and style representations from the images.
- **Loss Functions:** The network is used to define and minimize two main types of losses:
- **Content Loss:** Measures the difference in content between the content image and the generated image.
- **Style Loss:** Captures the difference in artistic style between the style image andthe generated image.

- **Optimization:** By minimizing both content and style losses, the algorithm generates an image that both resembles the content of the original image and incorporates the artistic style of the style image.

Working Principle:

- **Feature Extraction**:

The neural network processes the content and style images, extracting high-level features at various layers.

- **Loss Computation**:

Content loss quantifies the difference in content between the images, while style loss measures the variance in stylistic elements.

- **Image Generation**:

By minimizing these losses using optimization techniques (e.g., gradient descent), a new image is synthesized that balances both the content and style characteristics.

Implementation and Usage:

NST implementations are available across different deep learning libraries like TensorFlow and PyTorch, often accompanied by pre- trained models and functionalities for ease of use. Customizing parameters and choosing different layers for content and style representations can yield diverse artistic outcomes. Neural Style Transfer stands as an intriguing intersection of artistry and AI, showcasing the potential for AI to create aesthetically pleasing visual compositions while delving into the depths of understanding artistic styles and content representations. Several implementations and libraries exist for neural style transfer, including Tensor Flow, PyTorch, and various other deep learning frameworks. These implementations often provide pre-trained models and functions to easily apply style transfer to images.

1.4. DEEP LEARNING IN NEURO TRANSFER:

Deep learning, particularly in the context of image manipulation and artistic transformation, often employs neural style transfer as one of its applications. This technique utilizes neural networks to merge the content of one image with the stylistic elements of another, creating visually appealing and artistically inspired results.
The process of deep learning-based neural style transfer involves several key steps:

- **Data Preparation:** Collecting a dataset that includes content images (representing the subject matter) and style images (providing artistic styles or textures).
- **Choosing Neural Network Architecture:** Selecting a pre-trained deep neural network, such as VGG, ResNet, or others, that acts as a feature extractor.
- **Feature Extraction:** Utilizing the chosen network to extract high-level features from both the content and style images at different layers of the network.
- **Loss Computation**: Calculating content loss and style loss:
- **Content Loss**: Measures the difference between the content of the content imageand the generated image.
- **Style Loss:** Quantifies the discrepancy in stylistic features between the style image and the generated image.
- **Optimization**: Using optimization algorithms (e.g., gradient descent) to minimize the content and style losses simultaneously, generating an image that combines the content and style aspects.

Frameworks such as TensorFlow, PyTorch, and others offer tools and pre- implemented models to facilitate the implementation of neural style transfer and other deep learning techniques. These frameworks provide a convenient environment to experiment, fine-tune parameters, and develop custom neural network architectures for various applications.

1.5. DEEP LEARNING STRUCTURES:

Deep learning structures refer to the architectural configurations and design patterns used within neural networks, which are the fundamental building blocks of deep learning. These structures define the arrangement of layers, connections, and operationsthat enable neural networks to learn complex patterns and representations from data. Here are some key deep learning structures commonly used in neural networkarchitectures:

Feedforward Neural Networks (FNN):

FNNs are the simplest form of neural networks. They consist of multiple layers of neurons where information flows in one direction, from input to output. They commonly include input layers, hidden layers, and output layers. Examples include Multi-Layer Perceptrons (MLPs).

Convolutional Neural Networks (CNN):

CNNs are designed for processing structured grid-like data, such as images. They use convolutional layers that apply filters/kernels to extract features hierarchically, followed by pooling layers for downsampling. CNNs are widely used in image recognition, object detection, and computer vision tasks.

Recurrent Neural Networks (RNN):

RNNs are specialized for sequential data by using loops within the network to persist information across time. They excel in processing sequences and are commonlyused in natural language processing (NLP), speech recognition, and time series analysis.

Long Short-Term Memory Networks (LSTM) and Gated Recurrent Units (GRU):

These are specialized variants of RNNs designed to mitigate the vanishing gradient problem and capture long-range dependencies in sequences. LSTMs and GRUs have gated mechanisms to selectively retain or forget information over time.

Autoencoders:

Autoencoders consist of an encoder network that compresses the input data into a latent space representation and a decoder network that reconstructs the input from the latentspace. They are used for unsupervised learning, data denoising, and dimensionality reduction. **Generative Adversarial Networks (GANs):**

GANs comprise two neural networks—a generator and a discriminator—that compete against each other. The generator creates synthetic data samples, while the discriminator tries to distinguish between real and fake data. GANs are used for generating realistic images, data augmentation, and creative tasks.

Transformer Architecture:

Transformers are based on self-attention mechanisms and are prominent in natural language processing tasks. They leverage attention mechanisms to capture relationships between different elements in the input sequence efficiently.

Capsule Networks:Capsule Networks aim to overcome some limitations of CNNs by encoding properties of features such as pose, deformation, and spatial hierarchies. They work by routing information between capsules (groups of neurons).Deep learning structures

can be combined, modified, or extended to create more complex architectures to tackle various tasks in domains like computer vision, natural language processing, reinforcement learning, and more. The choice of architecture often depends on the nature of the data, the problem at hand, and the specific requirements of the task. Researchers and practitioners continually explore new architectures and adaptations to improve performance and address specific challenges in deep learning applications.

Convolutional Neural Networks (CNNs) are a class of deep neural networksprimarily used for processing and analyzing visual data such as images. Here, I'll describe and illustrate the typical components and structure of a CNN through diagrams.

Components of a CNN:

Convolutional Layer:

These layers apply filters or kernels to the input image, extracting local features through convolutions. Each filter detects specific patterns like edges, textures, or shapes.

Pooling Layer:

Pooling layers (commonly MaxPooling or AveragePooling) downsample the feature maps, reducing their spatial dimensions while retaining important information. Thishelps in reducing computational complexity and extracting dominant features.

Activation Function:

Activation functions (e.g., ReLU, Sigmoid, Tanh) introduce non-linearity, enabling the network to learn complex relationships within the data.

Fully Connected Layers:

These layers, often at the end of the network, process the extracted features and make predictions based on the learned representations.The Figure 1.2 represents the fundamental architecture of the deep learning model. The multiple deep dense layers which are the non-output layers of the deep learning models get trained as an auto-encoder and each of these dense layers learns effective features from the previous layers.

These auto encoders of the deep learning neural model get trained with the classic weight update algorithms to reproduce the presented inputs. Each of these deep hidden layers tends to be the best feature extractor extracting the prominent ones and all the dense layers get trained to be auto encoders. The final layer of the deep learning neural network model forms the decoder and this classifies or predicts the output from the earlier outputs of the deep layer and this makes the deep learning model possess better abstraction capability.

CHAPTER 2 LITERATURE SURVEY

2.1 Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville,and Y. Bengio, "*Generativeadversarial networks*'' 2014

Videos have consistently held a significant role in our lives, serving as a versatile tool for education, entertainment, and communication. Over time, creating videos has become more accessible, largely due to advancements in camera technology, especially in mobile devices. Today, the average person prefers using their smartphones to capture videos instead of relying on expensive and complex camera setups. This shift not only makes video creation more convenient but also democratizes the process, allowing a broader range of individuals to participate in and contribute to the dynamic world of video content. Conversely, professionals in the entertainment industry employ specialized hardware and advanced editing tools to craft visually stunning scenes,often leveraging Computer-Generated Imagery (CGI) software. These dedicated tools enable

entertainment producers to bring imagination to life, seamlessly integrating CGI elements into their productions. By utilizing sophisticated equipment and software, they have the capability to produce cinematic experiences that captivate audiences with breathtaking visuals and intricate details. This stark contrast between the average user's reliance on mobile devices and the entertainment industry's use of cutting- edge technology highlights the diverse landscape of video creation, showcasing both accessibility and sophistication in the evolving realm of multimedia production. Since the introduction of the first Generative Adversarial Network (GAN) by Goodfellow et al. [1], there has been a prolific evolution marked by numerous resources, diverse approaches, substantial improvements, and widespread implementations in the field. Researchers and practitioners alike have actively contributed to the expansion of GAN applications, exploring innovative methods and refining the initial concept.

2.2 H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu, "*Real-time neural style transfer for videos,*" in Proc. IEEE Conf. Comput. Vis. Pattern Recognit.

Moreover, the applications of video style transfer extend beyond conventional entertainment. There is a growing trend in utilizing this technology for recreational purposes, particularly in conjunction with Augmented Reality (AR). The work of Dudzik et al. [6] exemplifies this fusion of video style transfer and AR, presenting an innovative approach to creating a virtual world modeled after the real one. This synthesis allows users to immerse themselves in an augmented environment where the boundaries between the physical and virtual worlds are blurred, offering a novel and engaging recreational experience. By leveraging video style transfer within an AR framework, the potential for creating dynamic and interactive virtual landscapes becomesapparent. Users can interact with and experience a seamlessly blended combination ofreal and digitally enhanced elements, introducing a new dimension to recreational activities. This intersection of video style transfer, entertainment, and AR not only showcases the versatility of these technologies but also opens doors to innovative and immersive experiences that were once confined to the realms of imagination. Generative Adversarial Networks (GANs) have emerged as powerful tools in the field of machine learning and artificial intelligence since their conception by Goodfellow et al. [4]. GANs are particularly notable for their ability to generate realistic and novel data by learning from existing datasets. The fundamental idea behind GANs involves a dual-network structure—a generator and a discriminator—engaged in a continuous adversarial process. Gatys et al. [7] introduced a groundbreaking approach to style transfer using convolutional neural networks (CNNs), marking a significant advancement in the field. Their method, for the first time, distinguished between content and style in images by utilizing the feature maps derived from the network model. Specifically, content information in an image was represented through the feature map, while the style information was captured using the Gram matrix.By segregating content and style elements, Gatys and colleagues achieved notable improvements in the efficiency and effectiveness of style transfer.

2.3 W. Dudzik and D. Kosowski, "*KunsterAR art video makerReal time video neural style transfer on mobile devices*," 2020

Notably, the model incorporated a confrontation network to enhance stroke information,ensuring a more comprehensive representation of the character's visual elements.The significance of their model lies in its tailored design to address the specific intricacies of Chinese characters. By leveraging DenseNet, the proposed network demonstrated anability to retain the inherent structure of Chinese fonts, which is crucial for maintaining the integrity and readability of the characters during style transfer. The inclusion of the confrontation network further enriched the model's capacity to capture detailed stroke information, contributing to a more accurate and faithful rendering of the desired font style. Zhu et al. [10] introduced an innovative method centered around learning to transform images from a source domain to a target domain, all without the necessity of paired examples. This approach is designed to facilitate not only style transfer but also seasonal transfer within the images. The key distinction lies in the ability of the model to autonomously learn and apply transformations without relying on specific paired instances for training. The proposed method by Zhu and collaborators focuses on the flexibility of image transformation, allowing for the seamless adaptation of styles and even seasonal characteristics. By

eliminating the need for paired examples, the model demonstrates a capacity to generalize and apply learned transformations to a diverse range of images, enhancing its versatility.

2.4 Y. Chen, Y. Lai, and Y. Liu, "*Cartoongan: generative adversarial networks for photo cartoonization*,"‖ in Proceedings of the IEEE Conference on Computer Vision and PatternRecognition, pp. 9465– 9474, IEEE, San Juan, PR, USA, June 2018.

In the context of image style transfer, the foundational aspect involves preserving the essential content information of the input image while integrating stylistic elements from a separate style image using dedicated models and algorithms. In the exploration of the mapping relationship during image style transfer, the critical task is to extract the distinctive content information characteristics from the image. However, it is crucial to acknowledge a substantial gap between the representation of image features and human visual understanding [14–15]. During the process of image style transfer, the objective is to maintain the core content features of the original image while incorporating stylistic nuances from a reference image. Achieving this balance necessitates the extraction of meaningful content information from the input image.

CHAPTER 3 EXISTING SYSTEM

3.1 EXISTING SYSTEM METHODOLOGY

Utilizing the extracted content feature information and style feature information, the stylization of the input image is achieved. The fundamental process involves the amalgamation of the content image with the style image, establishing a mapping relationship between the input image and its stylized counterpart through a neural network. Gatys et al. [22] pioneered this approach by combining feature information fromboth images, minimizing the loss in content reconstruction and style reconstruction to derive a high-quality stylized image. While effective, this method requires significant computational resources. To address the computational complexity issue, some researchers have proposed faster image stylization methods based on feedforward networks. These approaches leverage pretrained network models to extract image feature information efficiently [23]. The key innovation lies in the use of feedforward networks, enabling rapid image stylization without the need for extensive calculations. By leveraging pretrained models, these methods capitalize on the ability of the network to quickly transform images, making the stylization process more practical and accessible for real-time applications. In summary, the stylization process involves combining content and style images, establishing a mapping relationship through a neural network. While Gatys et al.'s approach achieves high-quality stylized images, newer methodsbased on feedforward networks offer faster alternatives, making image stylization more computationally efficient for various applications.

The loss function in a machine learning model quantifies the inconsistency between predicted and actual values, serving as a critical factor in determining theoptimization objective for the entire model. By utilizing the loss function, networkparameters are optimized using the back propagation algorithm, enabling the adjustment of the model parameters to achieve the best possible outcome. Common loss functions, like square difference loss and cross-entropy loss, evaluate the model's quality by calculating the error between the generated image and the real image. However, these metrics fall short of measuring image stylization results at a perceptual level [24].In response to this limitation, the perceptual loss function emerges as a more nuanced approach. It extracts feature information from the image and assesses the error between the generated image and the real image across different levels of feature maps. Perceptual loss can capture semantic information from various levels, with higher feature levels extracting more abstract semantic features that closely align with the observational effect of the human eye [25]. Unlike the common L1 loss, which may struggle to generate clear high-frequency information, it excels at accurately capturing low-frequency information in the image. This nuanced perceptual loss function provides a more comprehensive and human-like evaluation of image stylization results, addressing the perceptual intricacies that traditional loss functions might overlook.With an increase in the depth of network layers, the style feature

information extracted from neural network models becomes progressively more abstract, showcasing high-level semantic expression effects. Zhao et al. [17] took a unique approach to address the challenge of extracting and understanding style feature information. They employed a deformable component-based model (DPM) to effectively extract the style features of an image. By utilizing DPM, the aim was to identify common features shared among imageswith the same style and distinguish the differences between images with distinct styles. This methodological choice allowed for a more nuanced exploration of style characteristics, considering both shared elements and unique variations within different styles. In doing so, Zhao and colleagues sought to enhance the understanding and representation of style information, acknowledging its inherently abstract nature in contrast to the more concrete nature of content features. In their work, Zhao et al. [18] employed a deformable component-based model (DPM) as a method to extract style feature information from images. Their approach focused on discerning both the shared characteristics among images of the same style and the distinctions between images with different styles.

CHAPTER 4 PROPOSED METHODOLOGY

4.1 NEURAL STYLE TRANSFER:

Neural networks are a class of machine learning models inspired by the structure and functionality of the human brain. They consist of interconnected nodes, known as neurons or units, arranged in layers that process and transform input data to produce output. These networks have gained significant attention due to their ability to learn complex patterns and make predictions or classifications across various domains.
Key Components of Neural Networks:

Neurons/Nodes: Neurons are the fundamental units that process information. They receive inputs, apply a transformation (often involving a weighted sum and an activation function), and pass the result to the next layer.
Layers: Neural networks are organized into layers: input layer, hidden layers, and output layer.

Input Layer: Receives initial data or features.

Hidden Layers: Intermediate layers between the input and output layers, responsible forlearning representations.

Output Layer: Produces the network's final output (e.g., predictions, classifications). **Weights and Bias:** Each connection between neurons has associated weights thatdetermine the strength of the connection. Bias terms allow the model to learn non-linearrelationships.
Activation Functions: These functions introduce non-linearity into the network, allowing it to learn complex patterns. Common activationfunctions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh.

Loss Function: Measures the difference between predicted and actual values. It guides the network in the right direction during training by adjusting weights and biases.

Optimization Algorithm: Techniques like Gradient Descent and its variants adjust weights and biases to minimize the loss function and improve the network's performance.

4.1.1 Types of Neural Networks:

Feed forward Neural Networks (FNN): The simplest type where information moves in one direction, from input to output, through layers without cycles or loops. Multi-layer Perceptrons (MLPs) are a type of FNN.
Recurrent Neural Networks (RNN): Designed to process sequential data by usingfeedback loops. They maintain an internal state/memory that allows them to retain information about previous inputs.
Convolutional Neural Networks (CNN): Primarily used for processing grid-like data, such as images, using convolutional layers to extractspatial hierarchies of features.
Generative Adversarial Networks (GAN): Comprising two networks—generator anddiscriminator—competing against each other to generate realistic synthetic data samples. **Long Short-Term Memory Networks (LSTM):**

A type of RNN that addresses the vanishing gradient problem, allowing better learning of long-term dependencies insequences.

Applications of Neural Networks:

- Image and pattern recognition
- Natural language processing (NLP)
- Speech recognition
- Time series forecasting
- Robotics and control systems
- Recommender systems
- Healthcare (diagnostics, drug discovery)
- Gaming and reinforcement learning

Neural networks have shown tremendous success in various domains, drivinginnovation and advancements in machine learning and artificial intelligence. Their flexibility, ability to learn complex relationships, and adaptability to different tasks make them a powerful tool for solving a wide range of problems.

The Figure 4.1 represents the images from the Neural style tranfer process, basically two images in Neural Style Transfer: style and content. To apply the style to the content image, we must duplicate it from the style image. When talk about style, mostly refer to the patterns, brushstrokes, etc. You must have observed that the lower-layer maps search for low-level characteristics like lines and blobs (gaborfilters).Our characteristics get more intricate as we move to higher tiers. We may conceptualize it intuitively in the following way: the bottom layers record low level features, such as lines and blobs, the layer above builds on these low level features and calculates somewhat more sophisticated features, and so on.

Now keep in mind that we are not training a neural network while performing style transfer. Instead, we optimize a cost function by altering the image's pixel values starting with a blank image made up of random pixel values. Simply said, we begin witha cost function and a white canvas. Then, we incrementally alter each pixel to reduce our cost function. To put it another way, when we train neural networks, we change the weights and biases, but when we transfer styles, we maintain the same weights and biaseswhile changing the image.

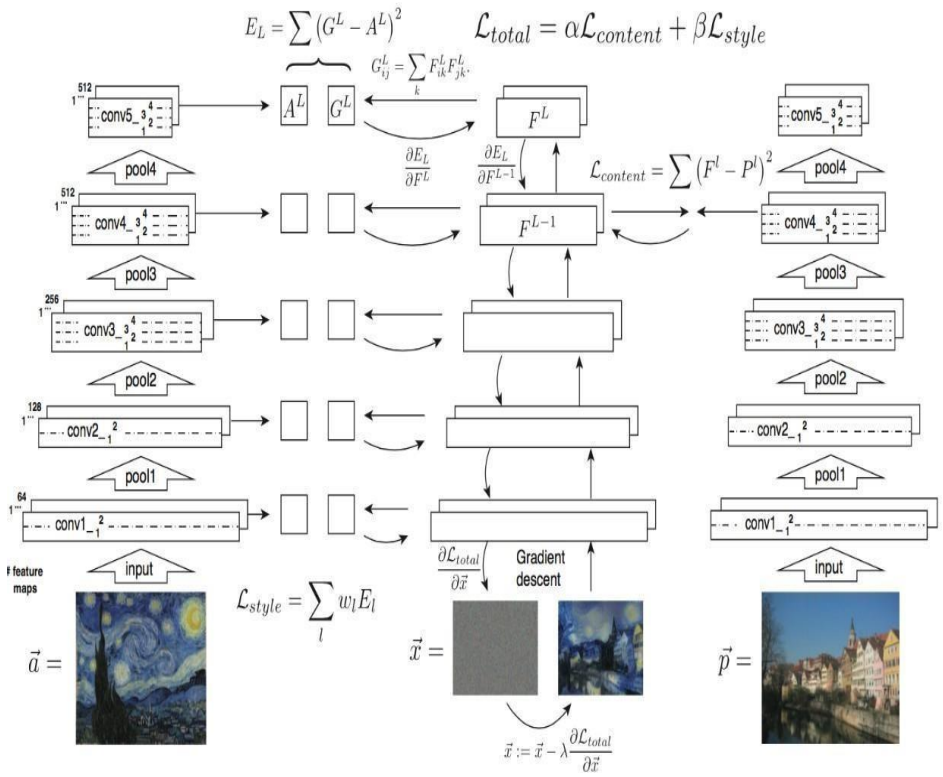

Fig 4.1 Neural Style Transfer

4.1.2 CODE IMPLEMENTATION IN TENSOR FLOW:

First, we import the necessary module. In this post, we use TensorFlow v2 with Keras. We will also import VGG-19 model from k eras

API.

4.1.2.1 IMPORTING LIBRARIES:

This script includes TensorFlow for deep learning, NumPy for numerical operations, Matplotlib for data visualization, and K eras- specific components for working with per-trained models and image processing.

```
# import numpy, tensorflow and matplotlib

import tensorflow as tf

import numpy as np
```

```
import matplotlib.pyplot as plt

import math

# import VGG 19 model and keras Model API

from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input

from tensorflow.keras.preprocessing.image import load_img, img_to_array

from tensorflow.keras.models import Model
```

4.1.2.2 IMPORT IMAGE DATA:

Now, we import the content and style images and save them into our working directory.

```
# Image Credits: Tensorflow Doc

content_path=

tf.keras.utils.get_file(
'content.jpg','https://storage.googleapis.com/download.tensorflow.org/example_images/YellowLabradorLooking_new.j pg')

style_path=

tf.keras.utils.get_file('style.jpg',
'https://storage.googleapis.com/download.tensorflow.org/example_images/Vassily_Kandinsky%2C_1913_-_Composition_7.jpg')
```

4.1.2.3 IMAGE PROCESSING:

Now, we load and process the image using Keras preprocess input in VGG 19. The expand_dims function adds a dimension to represent a number of images in the input. This preprocess_input function (used in VGG 19) converts the input RGB to BGR images and centre these values around 0 according to ImageNet data (no scaling).

```
# code to load and process image

def load_and_process_image(image_path):

img = load_img(image_path)

# convert image to array

img = img_to_array(img)

img = preprocess_input(img)

img = np.expand_dims(img, axis=0)

return img
```

Now, we define the deprocess function that takes the input image and perform the inverse of preprocess_input function that we imported above. To display the unprocessed image, we also define a display function.

```
def deprocess(img):

# perform the inverse of the pre processing step

img[:, :, 0] += 103.939

img[:, :, 1] += 116.779

img[:, :, 2] += 123.68

# convert RGB to BGR

img = img[:, :, ::-1]
```

```
img = np.clip(img, 0, 255).astype('uint8')

return img

def display_image(image):

# remove one dimension if image has 4 dimension

if len(image.shape) == 4:

img = np.squeeze(image, axis=0)

img = deprocess(img)

plt.grid(False)

plt.xticks([])

plt.yticks([])

plt.imshow(img)

return

# load content image

content_img = load_and_process_image(content_path)
display_image(content_img)
```

```
# load style image

style_img = load_and_process_image(style_path)

display_image(style_img)
```

4.1.2.4 MODEL INITIALIZATION:

Now, we initialize the VGG model with ImageNet weights, we will also remove the top layers and make it non-trainable.

```
# this function download the VGG model and initialise it

model = VGG19(

include_top=False,

weights='imagenet'

)

# set training to False

model.trainable = False

# Print details of different layers

model.summary()
```

4.1.2.5 CONTENT MODEL DEFINING:

Now, we define the content and style model using Keras.Model API. The content model takes the image as input and output the feature map from “block5_conv1” from the above VGG model.

```
# define content model

content_layer = 'block5_conv2'

content_model = Model(

inputs=model.input,

outputs=model.get_layer(content_layer).output

)

content_model.summary()
```

4.1.2.6 STYLE MODEL DEFINING:

Now, we define the content and style model using Keras.Model API. The style model takes an image as input and output the feature map from “block1_conv1, block3_conv1, and block5_conv2″ from the above VGG model.

```
# define style model

style_layers = [

'block1_conv1',

'block3_conv1',

'block5_conv1'
```

]

style_models = [Model(inputs=model.input,

outputs=model.get_layer(layer).output) for layer in style_layers]

4.1.3 CONTENT LOSS:

The produced picture's characteristics are contrasted with the content image in the content loss process. While the algorithm changes the style, it must be ensured that the output picture retains the same content. In this manner, the authenticity of the content image is preserved, and the style components are added from the style image. First, loss networks are constructed using two identical pre-trained CNNs for image categorization. A test picture (created) and a reference image (content) will be supplied to these networks, respectively. These two classifiers' outputs are sent into the loss function as inputs. You must first compute two things before you can calculate the content loss function. The characteristics of the produced picture's content as well as the contentimage created using pre-trained loss networks. The Mean Squared Error, also known as the L2 Norm, is then computed. We follow the name to do this computation. First, use element-wise subtraction to compute the error. Subtract the characteristics of the produced images from those of the content images. To obtain the squared errors, next square these mistakes elementally. The Mean Squared Error is obtained by averaging all the values and dividing the result by the number of features.

Now, we define the content loss function, it will take the feature map of generated and real images and calculate the mean square difference between them.

Content loss code

def content_loss(content, generated):

a_C = content_model(content)

a_G = content_model(generated) # Add this line to compute a_G

loss = tf.reduce_mean(tf.square(a_C - a_G))

return loss

The content loss between a style image and a generated image is determined by the function content_cost, which

is defined in this code. To make sure that the generated image preserves the original image's content, neural style transfer algorithms frequently employ content loss.

```
#content loss

def content_cost(style, generated):

J_content = 0

for style_model in style_models:

a_S = style_model(style)

a_G = style_model(generated)

GS = gram_matrix(a_S)

GG = gram_matrix(a_G)

content_cost = tf.reduce_mean(tf.square(GS - GG))

J_content += content_cost * weight_of_layer

return J_content
```

4.1.4 STYLE LOSS:

You must first compute two things before you can calculate the content loss function. the characteristics of the produced picture's content as well as the content imagecreated using pre-trained loss networks. The Mean Squared Error, also known as the L2 Norm, is then computed. We follow the name to do this computation. First, use element- wise subtraction to compute the error. Subtract the characteristics of the produced imagesfrom those of the content images. To obtain the squared errors, next square these mistakes elementally. The Mean Squared Error is obtained by averaging all the values and dividing the result by the number of features.

The function style_cost, defined by this code, determines the style loss between a generated image and a style image that is supplied. In neural style transfer algorithms, style loss is frequently employed to create an image that blends the content of two different images with their styles.

```
#style loss code

def style_cost(style, generated):

J_style = 0

for style_model in style_models:

a_S = style_model(style)

a_G = style_model(generated)

GS = gram_matrix(a_S)

GG = gram_matrix(a_G)

content_cost = tf.reduce_mean(tf.square(GS - GG))

J_style += content_cost * weight_of_layer


return J_style.
```

4.1.4.1 GRAM MATRIX:

Now, we define the gram matrix function. This function also takes the real and generated images as the input of the model and calculates gram matrices of them before calculate the style loss weighted to different layers.

```
# gram matrix
```

```
def gram_matrix(A):

channels = int(A.shape[-1])

a = tf.reshape(A, [-1, channels])

n = tf.shape(a)[0]

gram = tf.matmul(a, a, transpose_a=True)

return gram / tf.cast(n, tf.float32)

weight_of_layer = 1. / len(style_models)
```

4.1.5 TOTAL LOSS:

The overall loss must be determined after accounting for both style loss andsubstance loss. Simply said, total loss is the sum of content loss and style loss. By calculating the overall loss, you can comprehend how the optimizer locates an image that combines the aesthetics of one image with the information of another.

4.1.6 MODEL PREDICTION:

In the final step, we plot the final and intermediate results.

```
# code to display best generated image and last 10 intermediate results

plt.figure(figsize=(12, 12))

for i in range(10):

plt.subplot(4, 3, i + 1)
```

```
display_image(generated_images[i+39])

plt.show()

# plot best result

display_image(final_img)
```

4.4.3 Implementation:

The figure 4.5 represents the EfficientNetB7 architecture and its pre-trained models are available in deep learning frameworks like TensorFlow and PyTorch, enabling easy access to the architecture for training, fine- tuning, or transfer learning tasks. These models can be used as powerful feature extractors or directly employed for specific image-related tasks, providing a balance between accuracy and computational efficiency.
The researchers first designed a baseline network by performing the neural architecture search, a technique for automating the design of neural networks. It optimizes both the accuracy and efficiency as measured on the floating-point operations per second (FLOPS) basis. This developed architecture uses the mobile inverted bottleneck convolution (MBConv). The researchers then scaled up this baseline network to obtain a family of deep learning models, called EfficientNets. Its architecture is given in the below diagram

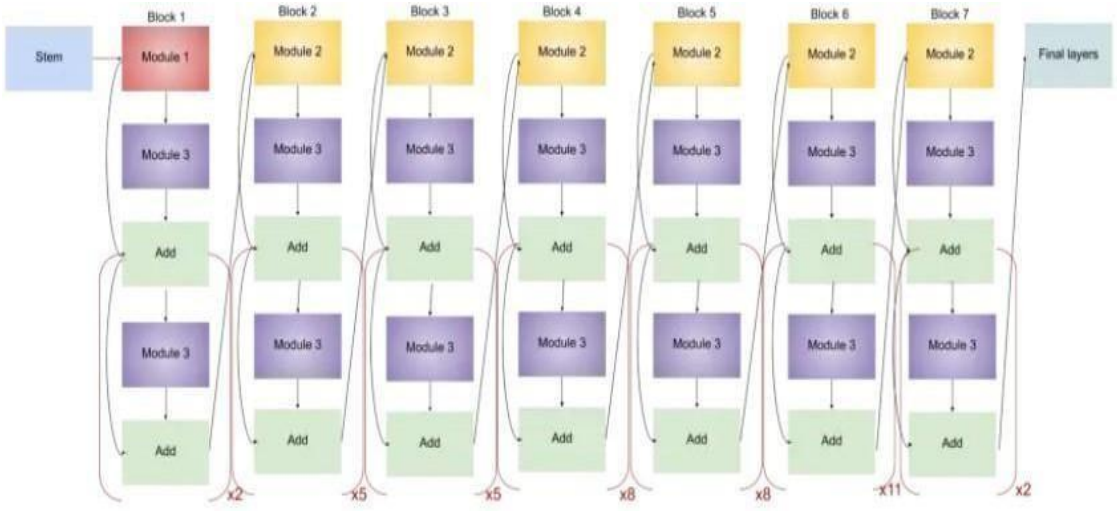

Fig.4.5 EfficientNetB7 Architecture

The work demonstrated the power of Convolutional Neural Networks (CNN) in creating artistic fantastic imagery by separating and recombine the image content and style. This process of using CNN to migrate the semantic content of one image to different styles is referred to as Neural Style Transfer.

The GAN based approach so far produces images that are sharper than any other generation method. Image

translation is related to style transfer, which constructs agenerated image with specific content and style by using a content image and a style image.

This Project introduces a neural style transfer model to conditionally generate a stylized image using only a set of examples describing the desired style. The proposed solution produces high-quality images even in the zero-shot setting and allows for greaterfreedomin changing the content geometry.

Proposed Gaussian optimal transport for Image style transfer in an Encoder/Decoder framework. Optimal transport for Gaussian measures has closed forms Monge mappings from source to target distributions. Moreover interpolates between a content and a style image can be seen as geodesics in the Wasserstein Geometry. Using this insight, we show how to mix different target styles , using Wasserstein barycenter of Gaussian measure.

The table 4.1 represents the statistical measurements of Time and var losses from the proposed CNN model in processing.

Table.4.1.Time of CNN Model

CNN MODEL	TIME TAKEN	T_VAR_LOSS
VGG19	25.2	51398.95
XCEPTION	804.3	99898.414
EFFICIENTNETB7	2574.4	34714.414

4.5 DATA SET:

A "data set" refers to a structured collection of data typically used for machine learning, statistical analysis, or research purposes. Datasets can come in various forms, sizes, and formats, and they serve as the foundation for training and evaluating machine learning models or conducting research in diverse fields.

4.5.1 Characteristics of Datasets:

Structure: Datasets can be structured, semi-structured, or unstructured. Structured datasets have well-defined formats like tables, while unstructured datasets contain free- form text, images, audio, etc.
Attributes or Features: Each data point in a dataset has specific attributes or featuresthat describe it. For instance, in an image dataset, the features could be pixel values, while in a tabular dataset, the features might be columns representing various parameters. **Labels or Targets**: In supervised learning, datasets often include labels or targets associated with each data point, indicating the desired output or class. Unlabeled data is common in unsupervised learning.
Size and Complexity: Datasets vary in size, ranging from small datasets suitable for experimentation to large-scale datasets with millions orbillions of records.

Types of Datasets:

Tabular Datasets: Structured datasets with rows and columns (e.g., CSV files, Excel spreadsheets, SQL databases) commonly used in various domains.
Image Datasets: Collections of images used for computer vision tasks, such as object detection, image classification, and segmentation. **Text Datasets**: Datasets containing text documents or corpora used in natural language processing (NLP) tasks, including sentiment analysis,text classification, and language modeling.
Time Series Datasets: Sequential data collected at regular intervals, commonly used in financial forecasting, weather prediction, and trend analysis.
Audio Datasets: Recordings of sound or speech used for speech recognition, audio classification, and signal processing tasks.Examples of Popular Datasets:
MNIST: A dataset of handwritten digits used for image classification tasks.

CIFAR-10 and CIFAR-100: Image datasets with small color images across variousobject classes.

ImageNet: A large-scale image dataset with millions of labeled images across thousandsof classes, commonly used for image classification challenges.
IMDb Reviews: A dataset containing movie reviews labeled with sentiment forsentiment analysis tasks.

UCI Machine Learning Repository: A collection of various datasets covering diversedomains, used for research and experimentation.

4.5.2 Importance of Datasets:

Model Training: Datasets are essential for training machine learning models, allowingthem to learn patterns and relationships within the data.
Model Evaluation: Datasets are used to assess the performance and generalizationcapabilities of models on unseen data. **Research and Analysis**: Datasets facilitate research in diverse domains by providingvaluable information for analysis and hypothesis testing.

Developing high-quality datasets that are representative, diverse, and well-labeled is crucial for achieving accurate and robust machine learning models across different applications and domains.
Here two different images are considered as input images and the content image and the style image whose style is going to be extracted using the gram matrix feature Mapping both images will be concatenated.
For Neural Style Transfer, a technique that merges the content of one image with the artistic style of another, datasets aren't used in the conventional sense as they are for typical machine learning tasks. Instead, Neural Style Transfer utilizes pre-trained Convolutional Neural Networks (CNNs), often trained on general image datasets like ImageNet, COCO, or other large collections of diverse images.These pre-trained CNNs serve as the basis for Neural Style Transfer, leveraging their learned representations of image features to separate content and style. Specifically, features extracted from different layers of these networks are used to represent content and style in the input images.When using Neural Style Transfer techniques, you typically need two types of images:

Content Image: The image whose content features you want to preserve in the final stylized image.

Style Image: The image whose style features you want to apply to the content image.

The Neural Style Transfer process uses the pre-trained CNN to extract features from both the content and style images. These features are then used to reconstruct a new image that combines the content of the content image with the stylistic elements of the style image.

So, there isn't a specific dataset dedicated to Neural Style Transfer itself. Instead, pre-trained models (like VGG, ResNet, etc.) that have been trained on large-scale datasets like ImageNet can be used . Then, by providing your chosen content and style images, you can apply the style of one image onto the content of another using Neural Style Transfer techniques based on these pre-trained models.

CHAPTER 5 RESULT AND DISCUSSIONS

The Neural Style Transfer process uses the pre-trained CNN to extract features from both the content and style images. These features are then used to reconstruct a new image that combines the content of the content image with the stylistic elements of the style image.

So, there isn't a specific dataset dedicated to Neural Style Transfer itself. Instead, pre-trained models (like VGG, ResNet, etc.) that have been trained on large-scale datasets like ImageNet can be used . Then, by providing your chosen content and style images, you can apply the style of one image onto the content of another using Neural Style Transfer techniques based on these pre-trained models.

The implementation of visualizing the VGG19 neural network pretrained on ImageNet yields several noteworthy observations and implications. VGG19, renowned for its simplicity and effectiveness, employs a stack of 3×3 convolutional layers, totaling 19 layers, which enables it to extract intricate features from images.

The choice of using ImageNet for pretraining is significant due to its vast scale, comprising approximately 1.2 million images across 1000 distinct object categories. This extensive dataset, famously employed in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), facilitates the learning of robust representations by the neural network.

This underscores the efficiency gains achieved through the more judicious utilization of model parameters, rather than merely increasing capacity.The figure 5.1 figure 5.2 and figure 5.3 illustrates the front end view of Visualize input images and the generated stylized images in which the original content image and style image are combined by the mentioned proposed techniques to create the stylized imgae.

Fig 5.1 Visualize Input Images And the Generated Stylized Images

Fig 5.2 Visualize Input Style Images

5.1 OUTPUT:

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 0s 0us/step
Model: "vgg19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, None, 3)]   0

 block1_conv1 (Conv2D)       (None, None, None, 64)    1792
```

block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_conv4 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_conv4 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_conv4 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

==========

Total params: 20024384 (76.39 MB)
Trainable params: 0 (0.00 Byte)

Non-trainable params: 20024384 (76.39 MB)

This underscores the efficiency gains achieved through the more judicious utilization of model parameters, rather than merely increasing capacity.The front end view of Visualize input images and the generated stylized images are shown in figure 5.3 in which the original content image and style image are combined by the mentioned proposed techniques to create the stylized imgae.

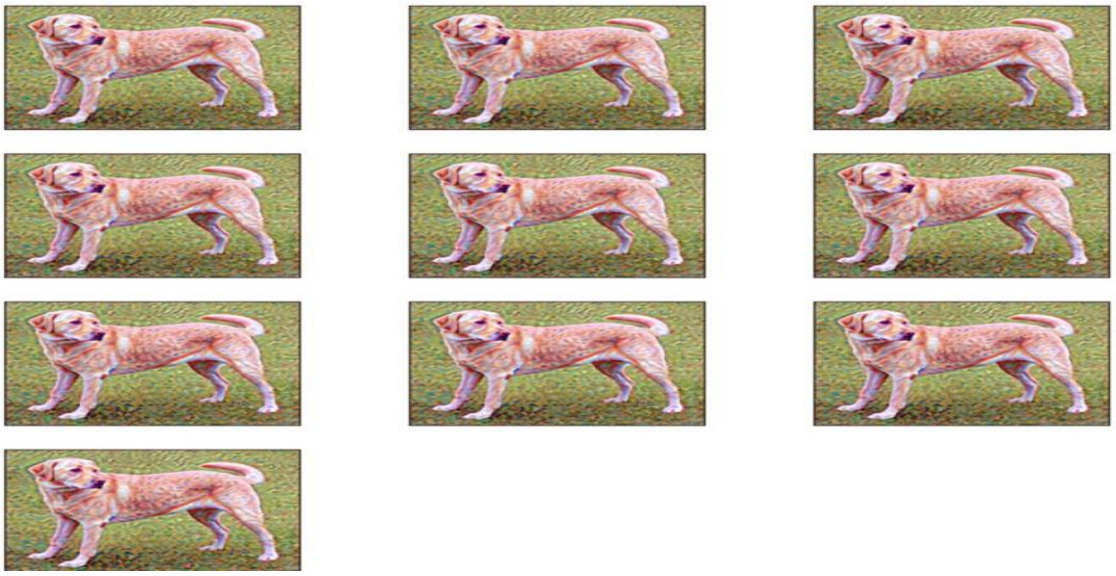

Fig 5.3 Generated Stylized Images

CHAPTER 6 CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In conclusion, the implementation and examination of both the VGG19 , Xception architectures and Neural style transfer and underscore the remarkable progress made in deep learning models for computer vision tasks.
The exploration of VGG19, with its straightforward yet powerful design consisting of stacked 3×3 convolutional layers, emphasizes the significance of leveraging pretrained networks like those trained on the ImageNet dataset. The vastness and diversity of ImageNet facilitate the learning of robust representations, enabling VGG19 to excel in various applications including style transfer, transfer learning, and fine-tuning. Its simplicity, combined with the effectiveness of transfer learning, renders VGG19 a valuable tool for practitioners seeking efficient model development and deployment across a myriad of image recognition tasks.
In contrast, the investigation into the Xception architecture reveals a paradigm shift in architectural design, exemplifying the pursuit of efficiency and performance optimization. By replacing standard Inception modules with depth-wise separable convolutions, Xception achieves superior performance while maintaining the same number of parameters as Inception V4. This highlights the importance of architectural innovation in maximizing model efficiency and effectiveness. Furthermore, the availability of Xception within popular deep learning libraries like TensorFlow and Keras enhances accessibility and empowers researchers and practitioners to achieve

state-of-the-art results in image-related tasks through transfer learning and fine-tuning.

6.2 FUTURE WORK

The neural style transfer image processing technique to video by generating smooth transitions between a sequence of reference style images across video frames. Video style transfer is a computational technique that applies the principles of neural style transfer, traditionally used for images, to video sequences. It involves altering the visual appearance of a video by infusing it with the artistic characteristics of a reference style image or a sequence of style images. Neural networks, often convolutional neural networks (CNNs), are employed to extract and combine the content of the original video frames with the artistic features of the chosen style, creating a stylizedoutput video. The goal is to generate visually appealing and artistically interesting videos that mimic the chosen style while maintaining coherence and consistency across consecutive frames. Advanced video style transfer methods aim to achieve not only spatial consistency but also temporal consistency, ensuring smooth transitions and fluid transformations between frames.

The generated output video is a highly altered, artistic representation of the input video consisting of constantly changing abstract patterns and colors that emulate the original content of the video. The user's choice of style reference images, style sequence order, and style sequence length allow for infinite user experimentation and the creation of an endless range of artistically interesting videos. In this work, I take one step further to explore the possibility of exploiting a feed-forward network to perform style transfer for video and simultaneously maintain temporal consistency among stylized video frames. Our feed-forward network is trained by enforcing the outputs of consecutive frames to be both well stylized and temporally consistent. More specifically, a hybrid loss is proposed to capitalize on the content information of input frames, the style information of a given style image, and the temporal information of consecutive frames.

To calculate the temporal loss during the training stage, a novel two-frame synergic training mechanism is proposed. Compared with directly applying an existing image style transfer method to videos, our proposed method will be employ the trained network to yield temporally consistent stylized videos which are much more visually pleasant.

Figure 6.1 Video Style Tranfer Result Of Six Styles

In contrast to the prior video style transfer method which relies on time-consuming optimization on the fly, our method runs in real time while generating competitive visual results. The front end view result of video style transfer showed in figure6.1. Video style transfer results of six styles. The high-level abstract content of each original inputvideo is kept, while the colors and textures are transferred from each style image.This technique finds applications in creative video production, artistic expression, and visual storytelling.

REFERENCES

[1] J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville,and Y. Bengio, ``*Generative adversarial networks*," 2014,arXiv:1406.2661. [Online]. Available: http://arxiv. org/abs/1406.2661.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge, ``*Image style transfer using convolutional neural networks*," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2414- 2424.

[3] Liu, P. N. Michelini, and D. Zhu, ``*Artsy-GAN: A style transfer system with improved quality, diversity and performance*," in Proc. 24th Int. Conf.Pattern Recognit. (ICPR), Aug. 2018,pp. 79-84.

[4] M.Ruder, A. Dosovitskiy, and T. Brox, ``*Artistic style transfer for videos*," 2016, arXiv:1604.08610. [Online]. Available: http://arxiv.org/ abs/1604.08610

[5] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu,``*Real-time neural style transfer for videos*," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 783791.

[6] W.Dudzik and D. Kosowski, " *KunsterAR art video makerReal time video neural style transfer on mobile devices*," 2020, arXiv:2005.03415. [Online]. Available: http://arxiv.org/abs/2005.03415

[7] Z.Guo, K. Yu, A. Jolfaei, A. K. Bashir, A. O. Almagrabi, and N. Kumar, –""*A fuzzy detection system for rumors through explainable adaptive learning*," IEEE Transactions on Fuzzy Systems, vol. PP, no. 99, 2021.

[8] Y.Chen, Y. Lai, and Y. Liu, -Cartoongan: "*generative adversarial networks for photo cartoonization,"* in Proceedings of the IEEE Conference on Computer Vision and PatternRecognition, pp. 9465–9474, IEEE, San Juan, PR, USA, June 2018.

[9] W. Lin, H.Yuan, and L. Lin, –""*Chinese typography transfer model based on generative adversarial network*," in Proceedings of the 2020 Chinese Automation Congress (CAC), pp. 7005–7010, IEEE, Shanghai, China, November 2020.

[10] Isola, J. Zhu, and T. Zhou, –"*Image-to-image translation with conditional adversarial networks*," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134, Honolulu, HI, USA, July 2017.

[11] H.Zhu, T. Park, and P. Isola, –""*Unpaired image-to-image translationusing cycle-consistent adversarial networks*," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2223–2232, Honolulu, HI, USA, July 2017.

[12] M. Rahaman, M. A. Ahsan, and M. Chen, –""*Data-mining techniques for image-based plant phenotypic traits identid"*- cation and classiDcation, Scientific Reports, vol. 9, Article ID 19526, 2019.

[13] L.Tan, K. Yu, F. Ming, X. Cheng, and G. Srivastava, –""*Secure and res artificial intelligence of things: a honey net approach for threat detection and situational awareness*," IEEE consumer Electronics magazine, Vol 99, no.1,2021.

[14] J.Zhang ,K.Yu, Z.Wen, X.Qi and A.Kumar paul ,‖3D "*Reconstruction motion blurred images using deep learning based intelligent system‖, computers , materials and continua*," vol 66,no. 2, pp.2087-2104,2021.

[15] W.Guo, A.K.Bashir , K..Yu,‖graphical *"Embedded based intelligence industrial decision for complex sewage treatment processes,"*International journal of intelligent systems, vol 40,2021.

[16] I.Siradjuddin, W. Wardana, and M. Sophan, -‴*Feature extraction using self-supervised convolutional autoencoder for content based image retrieval,"* in Proceedings of the 2019 3rdConference on Informatics and Computational Sciences (ICICoS), pp. 1–5, Semarang, Indonesia, October 2019.

[17] M..Zhao, Q. Miao, J. Song, Y. Qi, R. Liu, and D. Ge, -‴*Architectural style classification based on feature extraction module,"* IEEE Access, vol. 6, pp. 52598–52606, 2018.

[18] N.Wei, -Research on the *"Algorithm of painting image style feature extraction based on intelligent vision,"* Future Generation Computer Systems, vol. 123, pp. 196–200, 2021.

[19] W.T. Chu and Y.-L. Wu, *Image style classiDcation based on learnt deep correlation features*, IEEE Transactions on Multimedia, vol. 20, no. 9, pp. 2491–2502, 2018.

[20] L. Zhang, M. Peng, W. Wang, Z. Jin, Y. Su, and H. Chen,m-*Secure and eEcient data storage and sharing scheme for blockchain based mobile edge computing*, Transactions on Emerging Telecommunications Technologies, pp. 1–17, 2021.