Style Transfer for Videos

Taranvir Singh

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Intelligent Systems)

Supervisor: Dr. John Dingliana

August 2020

Declaration

I, Taranvir Singh, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Taranvir Singh

September 7, 2020

Permission to Lend and/or Copy

I, Taranvir Singh, agree that Trinity College Library may lend or copy this thesis upon request.

Taranvir Singh

September 7, 2020

Acknowledgments

I would like to express my gratitude to Dr. John Dingliana for his constant support and guidance during the whole process of this research. Without his valuable inputs, the completion of this research wouldn't have been possible.

Taranvir Singh

University of Dublin, Trinity College
August 2020

Style Transfer for Videos

Taranvir Singh, Master of Science in Computer Science

University of Dublin, Trinity College, 2020

Supervisor: Dr. John Dingliana

Recent advances in the field of Deep Learning have allowed the use of machines to generate beautiful looking artistic images. This study introduces the techniques to extend methods of unpaired image-image translation to video-video translation. A new temporal loss function is introduced which uses optical flow information from the original video to guide the model to learn temporal relationships among the video input. Generative Adversarial Network is used to transfer style from a set of images to a full video sequence while maintaining the consistency of the output video. The method is evaluated by calculating the temporal smoothness of the generated videos using an evaluation metric proposed in this study and comparing the results against prior methods.

Summary

This dissertation's main aim is to design the Machine Learning model which can convert real time videos to artistic style videos while maintaining the temporal coherence in the final output. The image processing tasks have been long in the limelight now, however, the video counterparts have additional challenges when compared to the image processing tasks. Artistic style transfer from one image to another has been attempted in many ways, in the past painters and artists created paintings by hand to look like the real world photographs. This kind of approach is very labor-intensive for image style transfer let alone handling hundreds of frames of a video. However, with the resurgence of Artificial Intelligence in the past decade, many powerful computing techniques have paved the way to attempt traditional image processing problems in a whole new way. Recent advances in the field of Artificial Intelligence have provided some path-breaking methodologies for style transfer in images. However, these approaches don't naturally extend to video applications as when considering videos we have to take into account an additional dimension i.e. temporal dimension. This dissertation purposes a way to extend the recent techniques applied for image style transfer to the domain of Video stylization. By using, optical flow estimation between adjacent frames of the original video a new loss function is formulated which guides the algorithm to learn how to maintain temporal coherency in the final output allowing the final video to be smooth and consistent.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays, a variety of image editing software provides a way to convert real-time photographs to artistic style paintings. These tools provide different choices of style application to the input images. However, many of these tools are limited to the application of images. In modern-day most of these techniques are inspired by approach followed in Gatys et al.[1], which used the neural networks to combine content from input image and style from another style image.

The process of style application on videos should not only process special features of the input videos but due to the presence of third dimension i.e. time, while generating videos the temporal features of the input video also need to be taken into account, so that a smooth and coherent video can be produced. To make a video temporally smooth and coherent, the adjacent frames should not be drastically different from each other. However, applying image processing algorithms independently on video frames can result in different outputs for fairly similar content, therefore giving rise to flickering effect in the output videos.

Ruder et al.[2] built upon the style transfer approach provided in Gatys et al.[1] to generate coherent artistic style videos. However, the basic drawback of this approach is that using this technique the style can be captured from only a single image, and also the final product seems like the contents of the two images are overlaid on each other.

Zhu et al.[3] explain an approach to use Generative Adversarial Networks for learning unpaired image-image translation, this approach uses a dataset of images for target mapping instead of a single image.
The basic motivation behind this research was to modify Generative Adversarial Network architecture explained in Zhu et al.[3] to generate styled videos from the provided input video, not just by using a single style image but a complete dataset of artistic images, so that the resultant output videos' style doesn't look like a single painting but an aggregation of the general style of the different paintings.

1.2 Research Objective

The objective of this thesis is to transfer style from a set of artistic images to the input video while maintaining the frame to frame coherency in the generated video. The style application on the input videos should be temporally coherent throughout the video so that no noticeable flickering effect can be observed in the resultant video. To achieve these objectives we build upon the architecture discussed in Zhu et al.[3].

Cycle GAN[3] can be used to generate stylized images from the Real-World photo-

Figure 1.1: Input Real-world photograph styled in Monet, Van Gogh, Cezanne and Ukioyo-e style

graph (see fig 1.1). Our objective is to extend this application of Cycle GAN from images to videos. The goal is to generate artistic style videos while maintaining the temporal coherency amongst adjacent frames of the resultant video.

To achieve temporal coherency this research focuses on utilizing optical flow information between adjacent frames of the input video. So, the basic idea is to use optical flow information to guide Cycle GAN to learn temporal coherency amongst the frames of output video.

1.3 Structure of Thesis

Chapter 2 reviews the state of the art techniques related to style transfer using an image as an input or video as an input. This chapter will also review other techniques used to apply an artistic style to videos and Images.
Chapter 3 will present the design and architecture of our method used to generate artistic style videos. This chapter will also detail how the use of optical flow helps in achieving temporal coherence in the output video. Training data and videos used to train our model will also be described in chapter 3.
Chapter 4 will provide details about the evaluation and results, and also compare those with the original Cycle GAN[3]. This chapter will detail the metrics used to quantify the temporal coherence in videos.
In the end Chapter 5 will provide the main contributions and discuss the future work to further this research.

Chapter 2

Related Works

2.1 Generative Adversarial Networks

Generative Adversarial Networks(GAN)[4] presented a novel way of generating content using two neural networks getting trained parallelly. The GAN consist of a generative model, which performs the task of generation, and discriminator which performs the task of estimating whether the sample came from the generator or the training data. These two networks are generally implemented using Neural Networks. The GANs work on the principle of adversarial learning, which is a form of minimax learning. The protector, who constructs the classifier we want to function properly, looks into the parameter space to find parameters that reduce the classifier cost as much as possible. At the same time, the attacker is looking to find parameters that maximize the cost. The two networks compete against each other where discriminator tries to distinguish fake data generated by the generator and on the other hand the generator tries to generate data in such a way so that it can fool the discriminator.
Since its conception GANs are modified to use for a variety of tasks. Denton et al.[5] provided an approach to use GANs for high-quality image generation. Aidan et al.[6] explains the technique to use GANs for synthetic video generation by training GAN on Kinetics-600 dataset, the resulting GAN model was able to generate 256*256 samples up to the length of 48 frames. Most of these kinds of GANs use adversarial loss that guides the model to generate content that is indistinguishable from the original content. This adversarial loss is analogous to negative log-likelihood loss which is used

to train the discriminator network by minimizing this loss however the generator tries to maximize this loss. This kind of min-max setup allows GANs to produce content that looks close enough to the real data.
This adversarial loss is adopted in our technique which allows the model to generate video frames that look close enough to the artistic paintings.

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

InfoGan[14] proposes a new technique of decomposing the noise vector into two different parts rather than using a single unstructured noise vector. The first part consist of incompressible noise and the second part is called the latent code which will target the significant features of the real data distribution. The object function of the Info GAN looks like:

$$min_G max_D V_1(D,G) = V(D,G) - \lambda(c; G(z,c))$$

Conditional GAN introduced a new objective function for GAN by conditioning both Generator and Discriminator on some new information.

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x|y)] + E_{z\sim p_z(z)}[log(1 - D(G(z|y)))]$$

Conditional GANs opened a way for a lot of different possibilities, using CGANs samples can be generated by conditioning on some other data like class labels, text, bounding box etc. [7] use text as the conditioning data to generate photo realistic images based on the text. CGANs paved a way for research [8] to study image translation using Conditional GANs. Image to Image translation proved to be the stepping stone for using GANs for artistic style transfer. Isola eta al.[9] built upon the concept of Conditional GAN to build an architecture which they called pix2pix network. Generally in GANs the generator is fed a random noise as the initialization however in pix2pix network an image is fed as iintialization data to the generator and the generator is tasked translate the input image to the desired output image.

$$L_{cGAN}(D,G) = E_{x,y}[logD(x,y)] + E_y[log(1 - D(y, G(y)))]$$

Conditional Cycle GANs presented by Zhu et al.[3] extended the approach of Con-

Figure 2.1: Image to Image translation using pix2pix network

ditional GAN with cycle consistency and used it for artistic style transfer and other unpaired image translation tasks.

2.1.1 Training of GANs

Due to the minimax approach to learning, the training of Generative Adversarial Networks is often unstable. As the objective of GANs is to find a balance between Generator and Discriminator's performance where the Generator produces good enough results to fool Discriminator, there is no actual convergence to a minimum. GANs are generative networks and therefore many problems can occur while training GANs. Following are some common challenges:

Mode Collapse

Mode collapse is said to happen when the Generator starts generating only one kind of image every time to fool the Discriminator. For example, if a GAN is tasked to generate MNIST like handwritten digit images, the Generator may start generating images of a single digit that is called mode collapse for Generative Adversarial Networks.

Nash Equilibrium

The objective of GAN is to find the Nash Equilibrium rather than finding a minimum convergence point. The Generator and Discriminator are tasked to work against each other and the GAN is believed to converge when both generator and Discriminator find a Nash equilibrium, an optimal point for the minimax objective equation. The Nash equilibrium is a point where Generator and Discriminator stop changing actions regardless of the action of the opponent. It is very hard to find Nash Equilibrium using gradient descent, so generally GANs never converge and the loss values can keep on oscillating.

Hard to evaluate

As the overall loss function of the GAN measures how well Generator and Discriminator are performing against each other, many times Discriminator may be overpowering Generator so the loss for Generator and the overall loss will increase however the quality of generated output may keep on improving, therefore it is essential to manually evaluate the quality of generator at every epoch, which makes it harder to decide that where should the training be stopped.

2.2 Cycle-Consistent Generative Adversarial Network

Zhu et al.[3] provides a more elaborated structure of Generative Adversarial Network, named Cycle GAN. The authors present 4 neural networks setup to perform unpaired Image to Image translation. Instead of one Generator and one Discriminator, this setup uses two Generators and two Discriminators. The major aim of Cycle GAN is to translate an input image from domain X to domain Y without any supervision. This kind of setup can be successfully used to transfer style from a set of artistic paintings to real-world photographs.
The major goal of style transfer can be divided into two main objectives 1) Changing the input image characteristics so that it looks similar to the target domain paintings. 2) Maintaining the contents of the input image so that the output image doesn't look

completely different from the input image. To fulfill the second objective, Cycle GAN uses a concept of cycle consistency, which constrains the model to change the style of an input image along with maintaining the overall content of the image.

$$G1 = f\left(X\right) \rightarrow Y$$

$$G2 = f^{'}\left(Y\right) \rightarrow X$$

$$f^{'}\left(f\left(X\right)\right) \approx X$$

The equations above depict the idea of Cycle Consistency, which ensures that if the first Generator translates the image X from domain A to an image Y then the second Generator which translates the image from domain B to A should translate the image back to X.
Incorporating this idea with the idea of Generative Adversarial Networks, Cycle GAN fulfills the objective of image-to-mage translation without any supervision.This architecture can be used effectively to translate images from one domain to another, however, to handle videos as input, we built upon the idea of this architecture.

2.3 Neural Style Transfer

Gatys et al.[1] provided an approach to use Neural networks to perform style transfer from one image to another image. The authors provided a novel way of distinguishing the style and content of the image. To extract the content of the image, the authors used VGG network to build a feature map. The image is passed through the VGG network and higher layers of the model are used to build the feature map encoding the content of the image. To represent the style of the style image, authors generate Gram Matrix containing correlation of activations across different channels of a lower hidden layer from VGG. The optimization problem is formulated that generates an image having similar feature maps to the content image's feature map and similar correlation of features across different channels as that of style image.
VGG is an image classification model pre-trained to predict the class of an image. To

classify an image, Deep Convolution Nets performs a series of transformations that allow the network to build complex features from the raw pixel values. According to the authors, these features at intermediate layers of the network can be used to represent the content and style of the image.
Two different loss functions are used to converge the optimization problem, Content Loss function monitors the similarity of content to the content of base image and Style Loss Function penalizes the divergence of style from the Style Image.

$$L_{Content}\left(I_c, I\right) = \sum_{0}^{i,j}\left(f_{i,j}\left(I_c\right) - f_{i,j}\left(I\right)\right)^2$$

Where f(I_c) = VGG(base image) , f(I) = VGG(output image)

$$L_{Style}\left(I_S, I\right) = \sum_{0}^{i,j}\left(G\left(f_{i,j}\left(I_S\right)\right) - G\left(f_{i,j}\left(I\right)\right)\right)^2$$

Where $G\left(f_{i,j}\left(I_c\right)\right.$ = Gram matrix constructed from correlations of neurons across different channels of a hidden layer of VGG(style image)
$G\left(f_{i,j}\left(I\right)\right.$ = Gram matrix constructed from correlations of neurons across different channels of a hidden layer of VGG(output image)

$$L_{total} = \alpha_{Content}\left(I_c, I\right) + \beta_{Style}\left(I_S, I\right)$$

Style Loss is calculated across multiple different layers of VGG for better stylizing results, however, the content loss is calculated across a single hidden layer. Depending on the choice of hidden layer for content loss different results can be achieved, selecting lower layers will conserve the artifacts like edges and colors of the content image which may not be a good choice if we want stylization from style image, however, selecting higher layers will allow altering edges and color maps of the content image to generate a better-styled image. But this also means that fine details and structures of the content image will be lost. The work of Gatys et al.[1] was extended by a lot of researchers down the line. Li et al.[10] used a domain adaptation for style representation rather

than using Gram Matrix. Domain adaptation can be achieved by minimizing the discrepancy between the source domain and the target domain. Li et al.[10] proved that a linear kernel, Gaussian kernel, and polynomial kernel can also be used to minimize discrepancy.
Risser et al.[11] pointed out the instability of Gram Matrix based approach to minimse style loss, therefore provided an additional histogram loss that guides the algorithm to match histogram of feature activations. This approach provided a stable optimistaion for style transfer.
Berger et al.[12] pointed out that Gram Matrix approach captures the global content however fails to capture spatial arrangements, to resolve this, authors proposed to shift feature maps horizontally and vertically by a small value delta. This technique allows to capture spatial arrangements hence more effective for modeling symmetric textures.
To preserve the fine structure and details in the styled image Li et al.[13] proposed a new loss called Laplacian Loss, the loss is calculated as Euclidean distance between Laplacian filter response for content image and the styled image. Laplacian filters are widely used to detect edges in the image hence Laplacian loss helps to preserve fine structure and details of the content image.
The major drawback of the aforementioned approaches is the fact that these techniques learn style representation from a single image, however, these approaches are not suitable to capture a general style from a collection of images. This dissertation focuses on learning style representation from a set of images rather than a single image which will allow us to capture patterns of famous artists not just a single painting.

2.4 Style transfer for videos

Unlike images, videos have an additional time dimension which has to be taken into account when applying video processing techniques. Hence to apply style transfer on videos, in addition to preserving the content of the input frames and adding the style of the style image, the output frames also need to have smooth transition amongst them. Applying style transfer on individual frames of the video without accounting the temporal relation amongst them can lead to an unwanted flickering effect in the final video output. To avoid this flickering effect temporal relations between the adjacent video frames need to be taken into account.

Ruder et al.[2] for the first time proposed a technique of Neural Style transfer for videos. The authors built upon the work of Gatys et al.[1] A new loss called temporal loss was added to the model to guide the model to learn temporal coherence. This loss was used to penalize abrupt changes between two consecutive frames. The idea behind this loss function is that the consecutive frames should look the same in all regions excluding the occluded regions, hence deviations in a frame from the previous frame's content are penalized using this loss function which guides the algorithm to generate consistently styled frames across the length of the video.

To achieve this task the authors use optical flow estimation between two adjacent frames and use that to generate styled video frame from the previous video frame and optical flow estimation between the two frames. Optical flow estimation is performed using Deepmatch[14] and Epicflow[15]. However, as this method was built upon the method in Gatys et al.[1], the processing time for even a short video clip was very long.

Huang et al.[16] proposed first real-time processing for artistic style transfer for videos using a feed-forward neural network. As this approach used a feed-forward neural network, processing can be done by a single forward pass through the feed-forward neural network, therefore, providing much faster-processing speed as compared to the previous approach. The approach builds upon the model-based optimisation techniques which allow faster processing time however limits the model to a single style. To overcome this problem the authors used modified Instance normalization which they call AdaIn. This Instance Normalization technique is used to alter the mean and variance of the input to match those of the style image. The instance Normalization is used as the encoder network and a separate decoder network is trained to map the encoded representation of the input back to Image Space generating final stylized output. To calculate temporal loss optical flow estimation is used. Instead of optimising at the time of inference this type of architecture provided a way to infer on the fly with real-time speed.

Chen et al.[17] also proposed a feed-forward network for real-time inference of style transfer for videos. In a addition to short-term temporal coherency, the authors also trained for long term temporal coherency to maintain consistency over a longer period. Instead of using temporal loss using optical flow estimation, the authors used a flow subnetwork and use this network to produce feature flow and hence representing optical flow information in feature space. The other half of the network consists of style

subnetwork which follows the same encoder-decoder type architecture as in Huang et al.[16] The feature maps of style subnetwork and flow subnetwork are combined linearly to produce final feature map which is fed to the decoder that learns to map the feature space to image space to produce the final styled output. The mask subnetwork is also employed to encourage new initialization in occluded areas and only condition frames from the previous frame in the disoccluded regions. The major difference here is that the mask subnetwork is learned at training time instead of pre estimation.

2.5 Optical Flow Estimation

The apparent motion of individual pixels on the image plane is known as optical flow. Often it can serve as a good approximation to the actual physical motion projected onto the plane of the image. Optical flow gives a succinct overview of both the regions of the moving picture and the direction of travel. Ideally, the optical flow field is a dense field of displacement vector which maps all points of the first image in the second image to their respective locations. In this study, we have used the optical flow estimation to guide the model to learn temporal coherence.

Horn and Schunck[18] first provided a way to compute the optical flow information, it assumed that the brightness of pixels remains constant for a short interval of time, and using this assumption the authors calculated the optical flow between two frames. There are a lot of techniques used to estimate optical flow but for this study, we only want to look at CNN based optical flow estimation techniques.

Dosovitskiy et al.[19] for the first time provided a CNN architecture to estimate the optical flow amongst video frames, they called it Flownet. CNN provided a huge leap to many computer vision tasks, in case of optical flow estimation also this technique was proved to be magnitudes accurate than other traditional variational techniques to estimate optical flow. The authors used a supervised way of estimating optical flow using FlyingChair dataset constructed by the authors. Authors used two CNN networks, FlowNetS and FlowNetC, designed based on the denoising U-Net autoencoder. This architecture paved a way for many CNN based techniques for optical flow estimation.

Ranjan and Black[20] developed the Spatial Pyramid Network following the coarse-to-fine concept of how to manage massive displacements. A convolution network is taught at each pyramid level to measure optical flow, which is then up-sampled to the next

level to direct the second image's warping towards the first. This technique gave higher accuracy than Flownet and consisted of considerably fewer parameters.
Flownet 2.0[21] extended the architecture of Flownet by stacking up multiple encoder-decoder networks. Flownet 2.0 improved the accuracy by nearly 50 percent however the model became even larger than Flownet and therefore considerably slower.
Sun et al.[22] provided an architecture named PWC-Net which gave a comparable performance to Flownet 2.0, however, is many folds smaller than Flownet 2.0. Like Spatial Pyramid Networks, PWC-Net also uses pyramidal processing. Due to its compact size and fast processing time PWC-Net is favorable to use for real-time applications. For this study, we used PWC-Net for performing optical flow estimation.

2.6 Occlusion

Occlusion happens when an object of interest is either blocked by another object in the camera vision or either out of the scope of the camera. When guiding a model to calculate temporal loss it is essential to have information about occluded and disoclcluded regions. When estimating the next frame of a video using optical flow information, it is important to have information about occluded regions as the occluded regions won't be present in the previous frame hence it is not possible to estimate their position in the next frame using optical flow information.

2.6.1 Occlusion Detection

Alvarez et al.[23] evaluated the validity of forward and backward flow with a threshold to detect occlusions, and then ignored the occlusions by resetting the identified pixels to zero in post-processing. This technique is used in this study to generate occlusion masks.

Chapter 3

Design and Implementation

This chapter will give a detailed description of the model architecture used for this study. Given the input video as a sequence

$$I_t|t = 1....n$$

the objective of the model is to output the stylized video as a sequence

$$O_t|t = 1....n$$

This chapter will detail the process of translating the input video to stylized output video.

3.1 Data set

There are two types of datasets used for the training of the model.
1) Video Dataset
2) Style Images Dataset
The video data set consists of 10 small length video clips downloaded from videvo.net, an open-source stock footage resource. The videos are of varied content ranging from landscape shots to city shots. Average video length ranges from 60-70 seconds, for processing the videos each video is broken into frames. The network is trained using frames of 256*256 image size, therefore each frame of the video is resized to dimensions

of 256*256.

Figure 3.1: Examples of video frames used for training

The style image dataset consists of images of a particular style that will be used as a target domain for image translation. These images are used to learn style representation. For our study, we used 3 collections of images, Van Gogh paintings, Monet paintings, and Ukiyo-e paintings. Figure 3.2 shows some examples of the paintings used for this dataset.

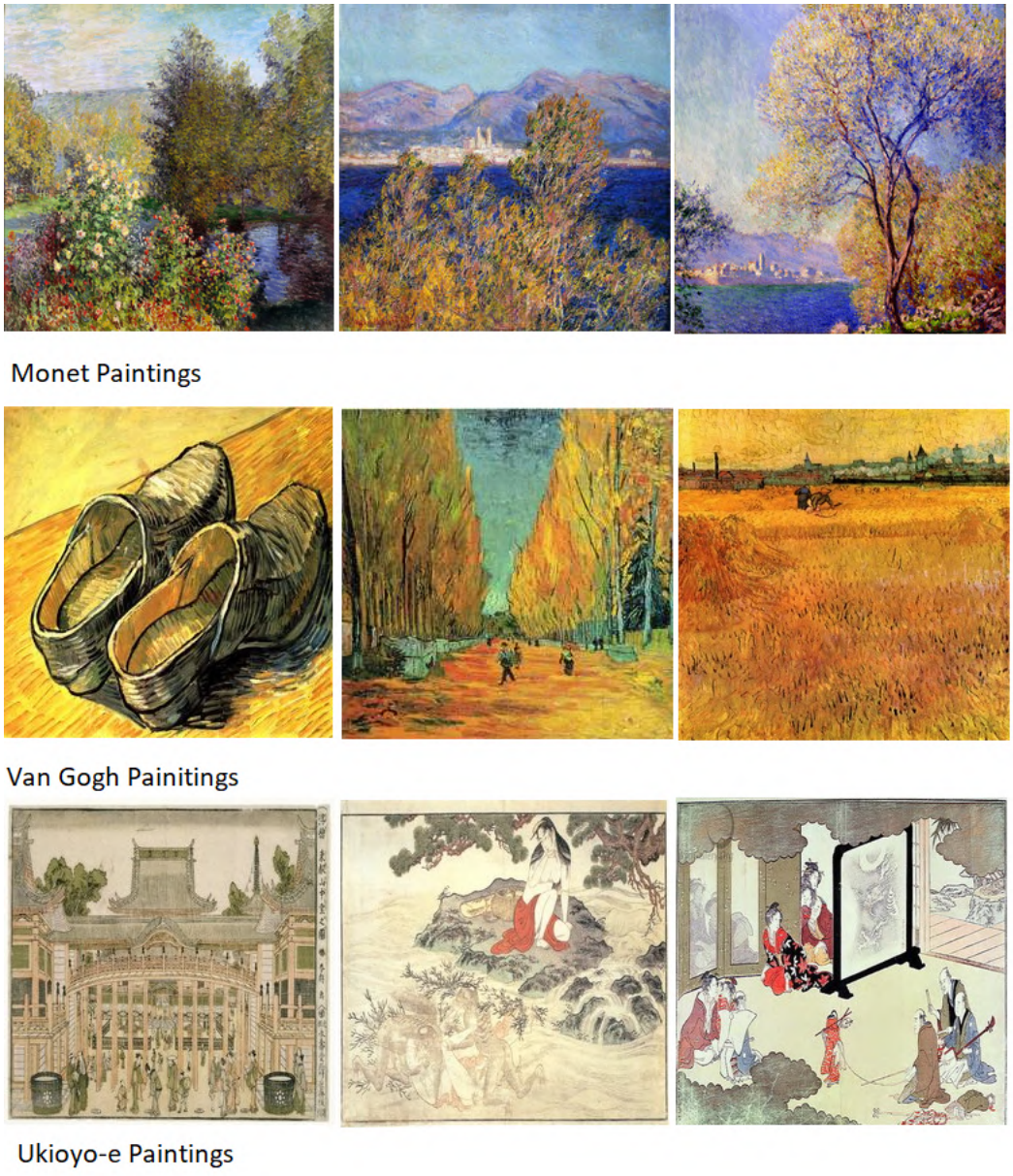

Figure 3.2: Examples of artistic paintings used for training

The artistic paintings are used as the target domain from which the model learns the style representation. Each paintings dataset contains around 500 paintings.

3.2 Data Preprocessing and Augmentation

3.2.1 Video Frames

As the input of our generator and discriminator takes 256*256*3 tensor as the input data, all the input frames of the video are resized to the size of 256*256*3. As the video frames need to be warped to compute the temporal error we didn't use any kind of cropping and jittering for the input video frames. However, the pixel values of the input frames are normalized before feeding into the generator.

3.2.2 Target Images(Paintings)

Target Images are drawn at random from the dataset of paintings(Van Gogh/Monet/Ukioyo-e). Before feeding the images to the model, the images are randomly cropped with the crop size of 256*256. The input images are all of size 256*256 therefore before cropping the images are resized to the image of 286*286 and then cropped. Also to augment the input data, random flipping and color jittering are also applied to the painting images. The data augmentation helps the model to perform better, to avoid losing the original color use of the paintings the color jittering is only applied randomly for 15 percent of the time. Also, the tensor values are normalized before feeding it to the model.

3.3 Network Architecture

3.3.1 Generator

A Convolution Neural Network with skip residual connections is used as the generator however there are some key changes to the general ResNet architecture used for image classification tasks. We trained the model using 256*256 images therefore for the generator network we used 10 residual blocks, however, the number of residual blocks can be increased depending on the size of the input.
Inspired from the [24], Instead of max pooling or average pooling in the CNN, strided convolutions are used in the network for the downsampling purposes so that the network can learn the way to downsample instead of using the pooling functions. For downsampling, the network uses stride-2 convolution and half for upsampling. The

process of downsampling and then upsampling the input provides computational conservation and allows deeper networks to train more efficiently. Therefore the input is downsampled by using strided convolution and then used for further convolutional layers which significantly reduces the computational costs. Also, the network doesn't use any fully connected layers on top of the convolutional layers.
All convolutional layers are followed by Instance Normalization, which increases the stability of the training process by normalizing the activation of each neuron in the convolutional layer. The normalization at hidden layers also helps in better gradient flow when the networks become deeper. Instance normalization also helps the generator to avoid mode collapse. All the convolutional layers use Relu as the non-linearity function except the last layer which uses tanh.

3.3.2 Discriminator

We used 70X70 PatchGan as the discriminator architecture, which means that the output of the model maps to a 70X70 square of the input image. In other words, a 70X70 PatchGAN will classify 70X70 patches of the input image as real or fake.
The PatchGAN is constructed according to the receptive field size, often referred to as the effective receptive field. The receptive field is the relationship between the model's one output activation to a region on the input image.
The network consists of a four-layered neural network, consisting of 4 convolution layer each followed by Batch Normalization and Leaky Relu, with slope set to 0.2 Each convolutions layer consist of fixed 4X4 kernel and stride of 2X2 is used on all layers except last 2 layers. For the last layer, the sigmoid activation function is used to give the output. The weights of the model are initialized using Gaussian Distribution with mean 0 and a standard deviation of 0.2. Adam is used as the optimizer choice.

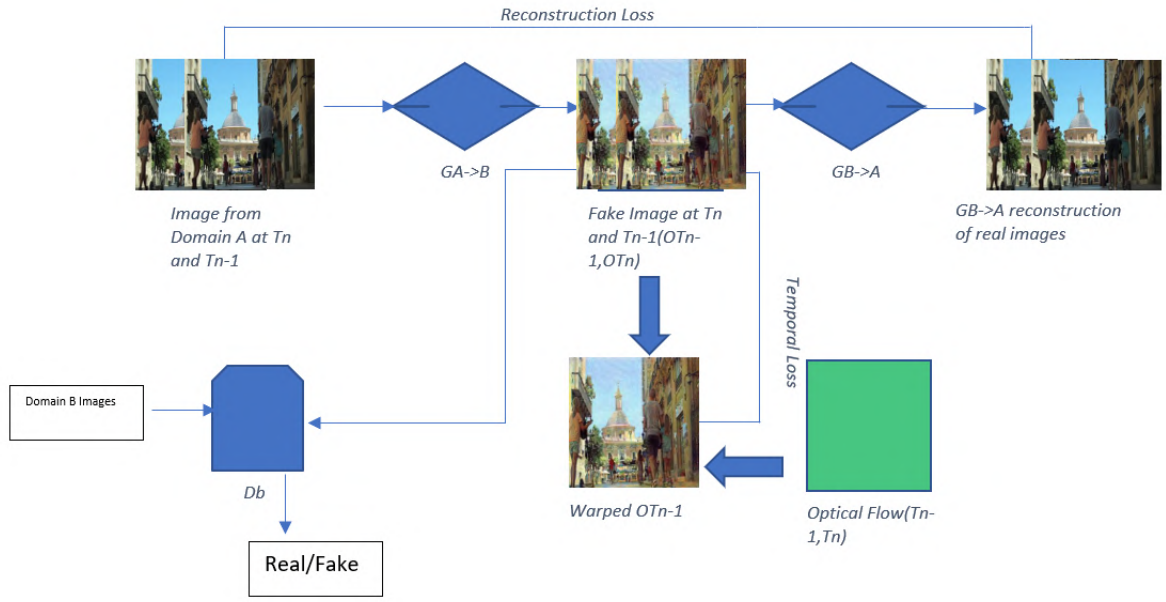

Figure 3.3: Architecture of the overall model

Figure 3.3 shows the overall architecture of the model, that contains 2 Generators and 1 discriminator. The input to the model consists of the two adjacent frames from the input video. Following are the steps performed by the model:
1) Both the input frames are translated to the styled images using the first generator.
2) The output of the first generator is fed to the Discriminator, which tries to predict whether the input image is real or fake
3) Also, the output of the first Generator is fed to the second generator which translates the image from the target domain back to the source domain.
4) The output of the frame at time stamp Tn-1 from the first generator is fed to the warping function along with the optical flow estimation between the two frames.

3.4 Implementation

The overall goal of the model is to map the input video from domain A to target domain B while maintaining the temporal coherency of the video. To achieve this, the two generators and a discriminator is used, this section gives details about the loss function.

$$Ł = \alpha * L_{adv} + \beta * L_{recon} + \gamma * L_{temp} + \theta * L_{identity}$$

3.4.1 Adversarial Loss

$$L_{adv}(D, G) = E_{y \sim p_{data}(y)}[logD(y)] + E_{x \sim p_{data}(x)}[log(1 - D(G(x)))]$$

The equation above gives the adversarial loss, the goal of this loss function is to find the distance between the distribution of the real data, and the distribution of data generated by the Generator.
D(y) gives the probability estimate of the Discriminator of y being the instance of real data.
G(x) is the output of the Generator given the input image x.
D(G(x)) is the Discriminator's estimate of generated data being real.
Overall the formula outputs the cross-entropy between the real data and the generated data.
In this loss the generator plays a role only in the second half of the loss function, so for the generator to minimize the loss, it needs to minimize the second half only because it has no control over the first half term of the loss.
As we have two frames of the video as input this loss is calculated independently for the two frames and the final loss is the average of two.

3.4.2 Reconstruction Loss

The goal of the model is to not only to translate the input image to the styled image but also to maintain the contents of the input image. So it is necessary to condition the model to maintain the contents of the input image. However, with only adversarial loss the GAN can come up with many different solutions that can be used to fool the discriminator. So the input image can be translated to any kind of image that matches the distribution of the target domain images and it is very much likey that the final output may lose all the content of the input image. However, we want to preserve the content of the input image and only change its style to match the target distribution. To achieve this we used Reconstruction Loss. It works on the concept of encoder-decoder, the first generator encodes the input and the second generator decodes the output of the first generator. In other words, the first generator is tasked to map the source domain images to the target domain and the second generator is tasked to map the target domain images to the source domain.

$$G_2\left(G_1\left(x\right)\right) \approx x$$

This loss penalizes the model if the reconstructed image from the second generator is too far from the input image. Therefore it limits the model to make too many changes to the input image but still making it look like the target images, this limitation, in turn, helps in preserving the contents of the input image.

$$L_{recon}\left(G_1, G_2\right) = E_{x \sim p_{data}(x)[||G_2(G_1(x))-x||]} + E_{y \sim p_{data}(y)[||G_1(G_2(y))-y||]}$$

Figure 3.4: Training Examples of Reconstruction objective

Figure 3.4 shows some of the intermediate training examples with real input and the reconstructed output of the model. The reconstructed image is translated from the fake styled image.

3.4.3 Identity Loss

Identity loss is used to discourage the generator from mapping too many unnecessary changes to the input frames and to preserve the details of the input frames.

$$L_{identity}\left(G_1, G_2\right) = E_{y \sim p_{data}(y)}[||G_1(y) - y||] + E_{x \sim p_{data}(x)}[||G_2(x) - x||]$$

The motivation behind this loss is to train identity mapping for both the generators. As the first generator is tasked to map images from source domain to target domain, so the goal of identity loss is to minimize the difference between the input and output of the first generator when image from target domain is fed to it.

$$G_1(y) \approx y$$

The goal of the second generator is to map the image from target domain to source domain so this loss will try to minimize the difference between input and output when image from source domain is fed to the second generator

$$G_2(x) \approx x$$

The weightage of this loss can control how aggressively the model will try to stylize the input image, more the weightage of this loss more conservative the model will be and avoid making too many unnecessary changes to the input image.

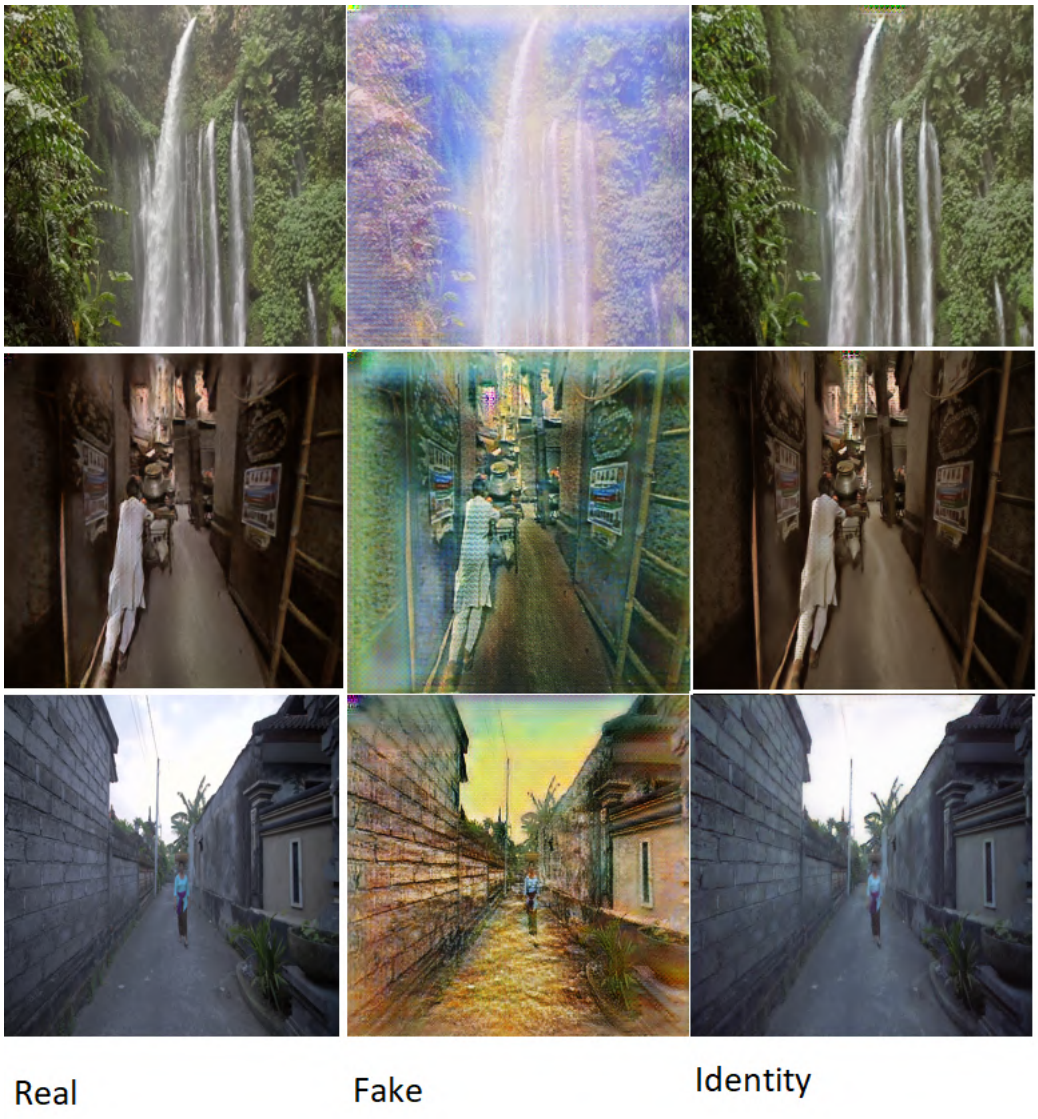

Figure 3.5: Training Examples of Identity mapping

3.4.4 Temporal Loss

Temporal loss is used to encourage temporal coherency in the final video output. In the input the model gets two adjacent frames of the video. Without using the temporal loss to guide the model, adjacent frames will result in similar styling yet the styling elements may change in the frames even with very similar contents, this inconsistency in styling adjacent inputs brings a flickering effect in the final video output.

The flickering effects can be minimized by keeping the styling consistent for the elements in the adjacent frames that don't change. Ideally, when two adjacent frames of the video are taken into account most of the content between two frames remains consistent, so the styling of the content which doesn't change from one frame to another should also remain unchanged. Only the areas that were not present in the previous frame should be styled differently than the previous frame. Following this approach, the smoothness and coherence of the video will greatly improve.

The naïve way to tackle this problem can be styling the input frames of the video and reducing the difference between these styled frames, however, the videos involve the motion of objects and by blindly minimizing the distance between the styled adjacent frames the effects of the motion will not be taken into account, therefore for better results, the motion between adjacent frames needs to be estimated.

There are many techniques used to estimate the optical flow between two frames of the video, in this study we have used PWC Net to compute the forward and backward optical flow between the two frames. The forward optical flow can now be used to estimate how the frame at time step t might look like by warping the frame at time step t-1.

Warping is the process of changing the placement of pixels of the input frame by using the information provided by the forward optical flow estimation. The optical flow estimation provides information regarding the motion in the frames, so it will tell in which direction the pixel will be moving and how strong the movement will be, using this information the warping function adjusts the pixel placements in the input frame to estimate how the next frame might look like.

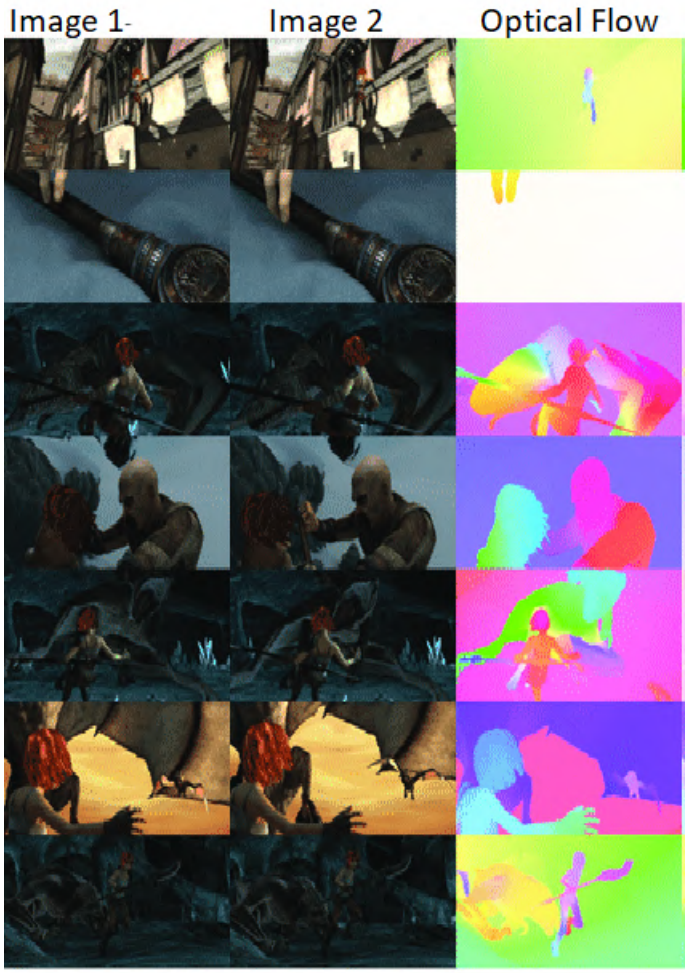

Figure 3.6: Optical Flow Examples

Figure 3.6 provides examples of optical flow estimation between two adjacent frames. This optical flow estimation is fed to the warping function along with the styled frame at timestamp t-1, the warping function uses the optical flow to warp the styled frame at time step t-1, so this warped version of the styled frame at times step t-1 gives an estimation of how the frame at time step t should look like.

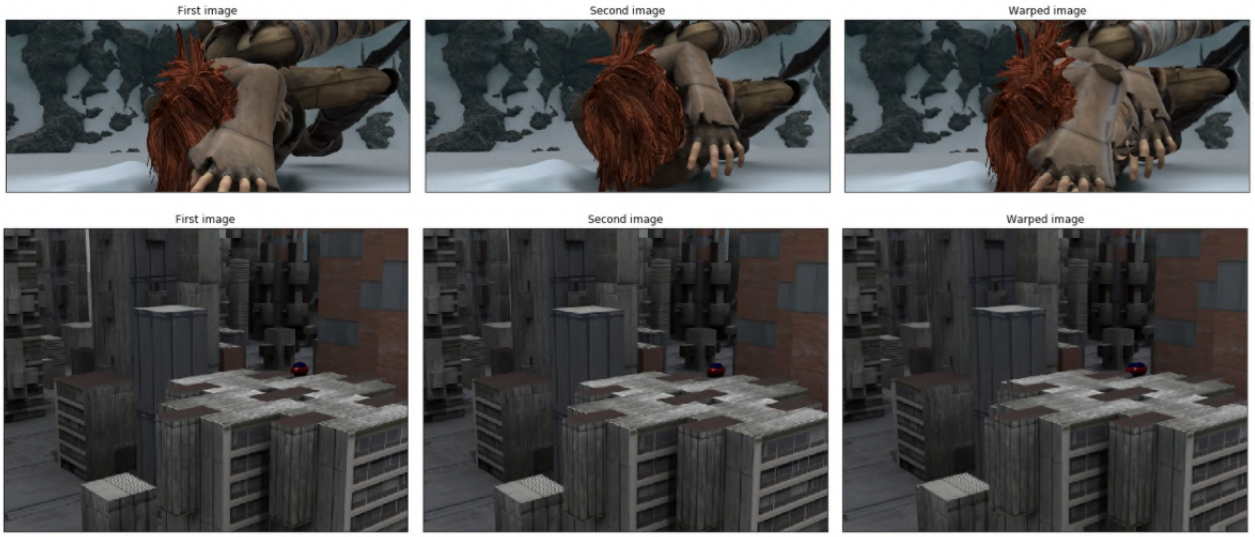

Figure 3.7: Image warping using Optical Flow

Figure 3.7 shows examples of warping images from the second image to the first image using optical flow estimation. It can be observed that this technique provides results very similar to ground truth image.
Using the warping function the styled frame at t-1 is warped using the optical flow estimation between frames at times step t and t-1. This warped frame now includes the motion information so we can use this to compare with the styled frame at time step t to compute the temporal error.

3.4.5 Occlusion Mask

Occlusion occurs when an element in the video is not visible on the frame is either blocked by another object from the point of view or is a result of some motion discontinuities. An element is said to be occluded when it is not visible in one frame but becomes visible in the next frame. It is necessary to detect the occluded areas because we want our model to encourage fresh style initializations in the disocclusion areas as there is no reference point in the previous frame to compare these elements with, so we want to exclude these areas when calculating the temporal loss. In this way, the model will not be penalized for fresh style elements for the areas getting disoccluded in the present frame.

To perform this task we first need to detect the areas which are getting disoccluded. To perform this task we used optical flow consistency check[25]. The consistency of optical flow in the forward direction and the backward direction can be checked to construct an occlusion mask.

$$w = (u, v) => Forward\,Optical\,Flow$$

$$\acute{w} = (\acute{u}, \acute{v}) => Backward\,Optical\,Flow$$

Warped Flow in the forward direction is given by

$$\hat{w}(x, y) = w((x, y) + \acute{w}(x, y))$$

Ideally, in the areas of disocclusions the backward flow should be almost opposite to this warped flow. Therefore occlusion mask can be constructed using the following inequality:

$$|\hat{w} + \acute{w}|^2 > 0.001(|\hat{w}|^2| + \acute{w}|^2) + 0.5$$

Also the motion boundaries can be calculated using

$$|\nabla\acute{u}|^2 + |\nabla\acute{v}|^2 > 0.01|\hat{w}|^2 + 0.002$$

The cooeffiecients of the above equations are taken from [25] To calculate the temporal loss the following equation is used:

$$L_{temp}(V_t, \acute{V_{t-1}}, c) = \frac{1}{D}\sum_{i=1}^{D} c_i.||V_{t_i} - \acute{V_{t-1_i}}||^2$$

$$D = Width * Height * Channels$$

Where c is the occlusion mask computed using optical flow consistency.

3.4.6 Training

Following are some key features of the training procedure followed in the study.
1) Adam is used as an optimizer with batch size 4.
2) The least-square error loss[26] is used as the overall training objective of the GAN.
3) The image pool is maintained from previously generated images using Generators to train the discriminator. The size of the image pool is maintained as 50.
4) The network is trained for 170 epochs.
5) The learning rate is initialized as 0.0002 and after 100 epochs the linear decay is applied to the learning rate.

Chapter 4

Results and Evaluation

4.1 Experiments

For this study, we trained 3 different models with 3 different style images dataset, Van Gogh paintings, Monet paintings, and Ukiyo-e paintings. Van Gogh and Monet models are trained from the scratch using our architecture with temporal loss, for Ukiyo-e model we fine tuned the already trained model of cycle Gan using the new loss function.
We trained the model with a different number of Resnet blocks for the generator, for 256*256 images we tried 7,9 and 10 blocks however 10 blocks were giving the better visual quality of images, therefore, trained each model with 10 blocks.
For the normalization part, we tried Batch Normalization and Instance Normalization. With a batch size of 1 both Batch Normalization and Instance Normalization worked well, however with a batch size greater than 1 Instance Normalization worked better. The following figures show the results of different models.

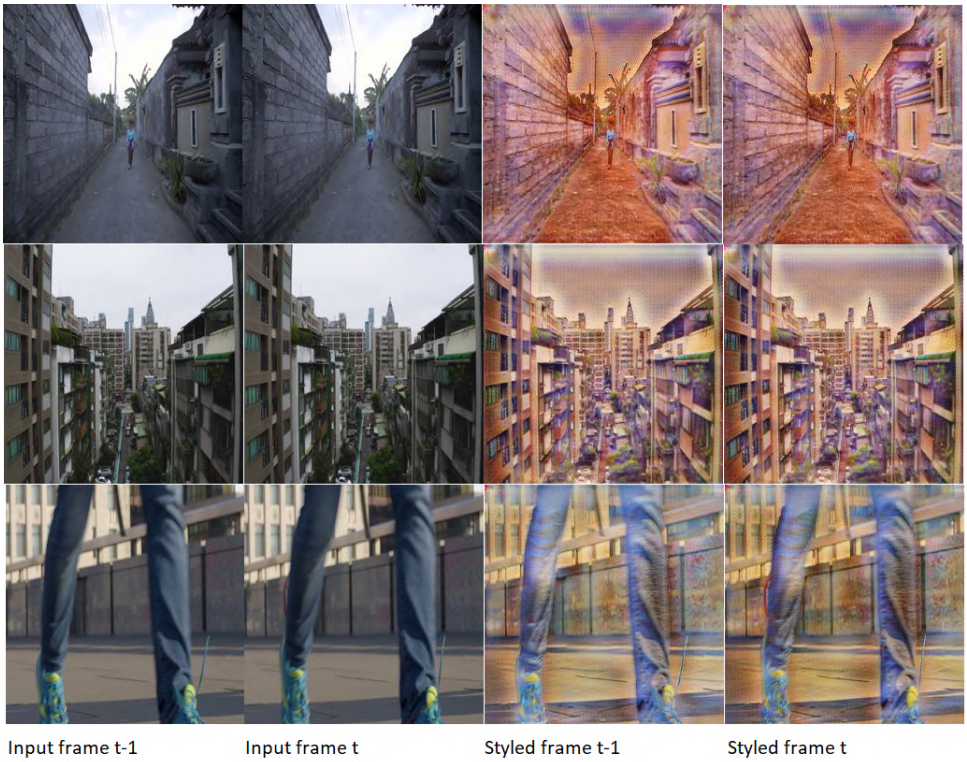

Figure 4.1: Monet Examples

Figure 4.2: Van Gogh Examples

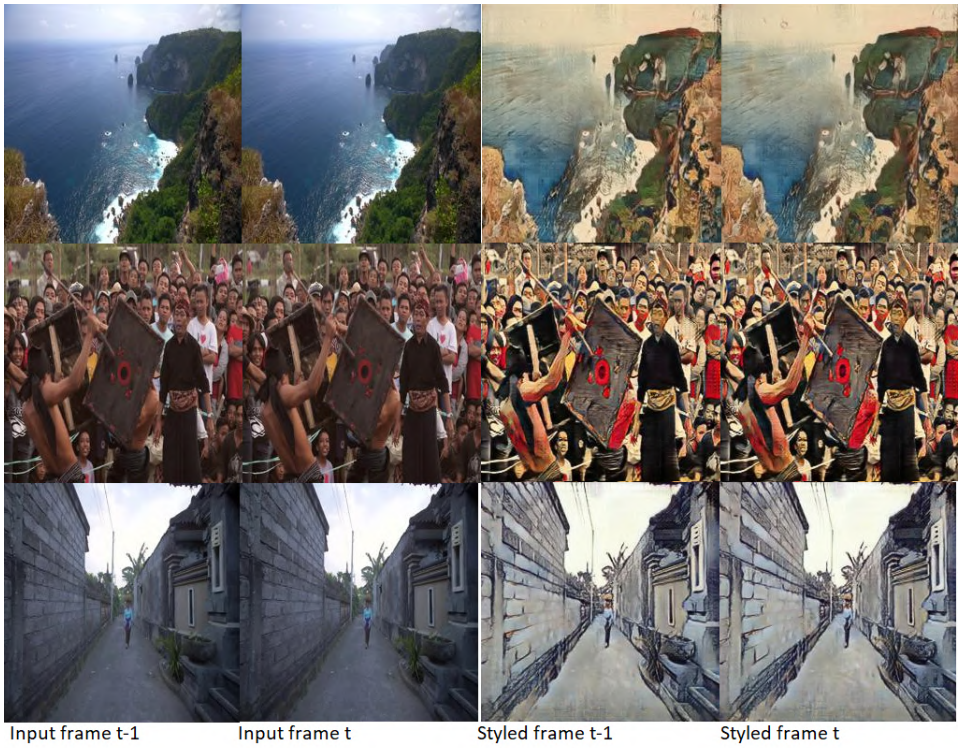

Figure 4.3: Ukiyo-e Examples

4.2 Temporal Consistency Results

The major goal of this study is to see if Optical Flow can be used to guide the model to output temporally consistent videos. To check this we produced styled videos with original Cycle GAN and compared those to the outputs of our model. Figure 4.4 shows examples of styled video using the same input video, first and the third row shows the sample output frames of the styled video using original Cycle GAN, second and fourth rows show the examples of the same video styled using our model. The black bounding boxes in the figures shows inconsistency in the consecutive frames, the area in the bounding box changes appearance from one frame to another in the case of original Cycle GAN, this kind of inconsistency gives rise to flickering effects in the output videos.

In the first row, the region in the bounding box has a lighter shade in the first frame however it becomes darker in the second frame. Similarly in the third row, the area in the bounding box shows that a part of the sky has a darker smudge in the first frame and it is considerably lighter in the next frame. These changes can seem to be very small and insignificant, however, when a lot of this kind of small inconsistencies combine they become more prominent to the human eye in the videos with a higher number of frames per second. Even a small change in RGB values of some pixels which are hardly noticeable if two frames are observed individually, will amplify greatly when these frames are stacked together to form a video.

However, the outputs of our model show that there is consistency among adjacent frames which results in smoother and temporally coherent videos.

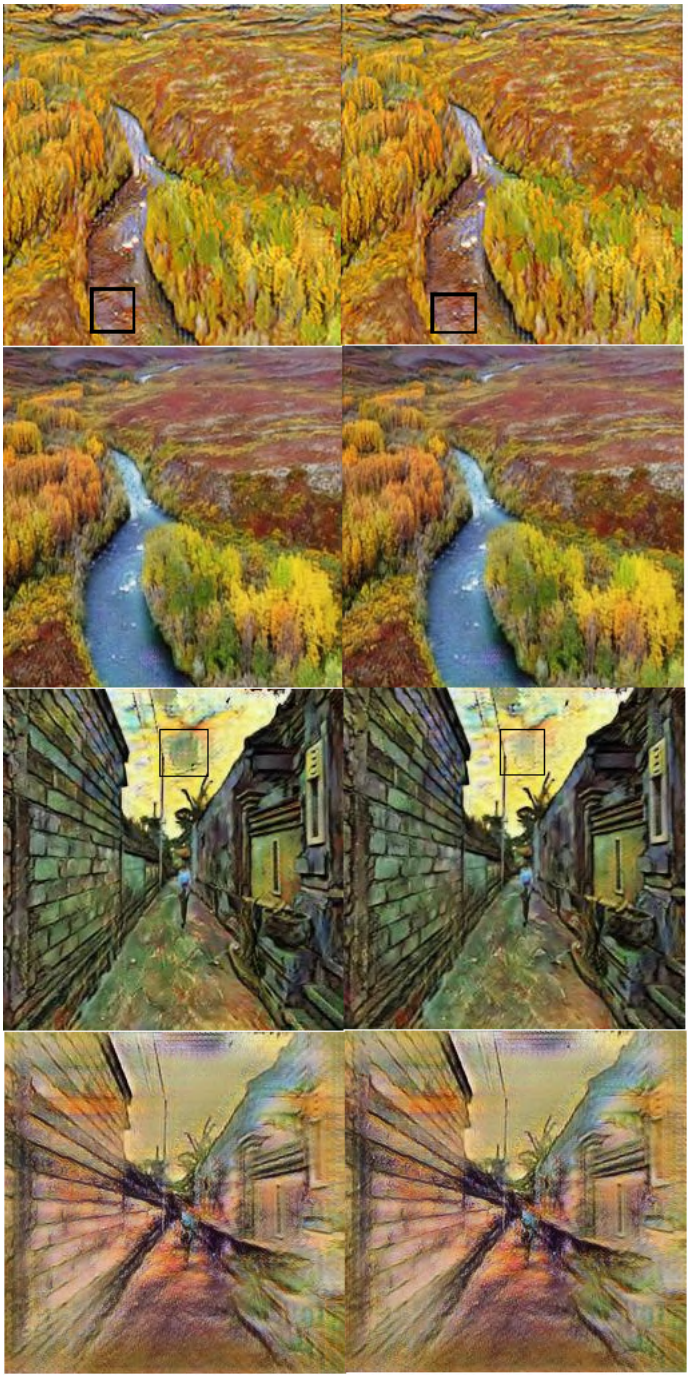

Figure 4.4: Temporal Consistency Examples

The results shown in figure 4.4 are produced using the same Van Gogh paintings as the style images dataset, however it can be observed that the results of our model and

the original Cycle GAN are visually different. The following are the reasons behind this discrepancy:
1) For the original Cycle GAN results we used a pre-trained model of Cycle GAN. The Cycle GAN model training uses a various number of hyperparameters like weights in the weighted loss function, different values for these parameters can result in different visual results. Therefore as we trained our model from scratch it is very hard to try different hyperparameters to produce the same results as that of the pre-trained Cycle GAN model.
2) The convergence of GANs is not a definite point, due to the mini-max training objective of GANs, the same model with the same hyperparameters may also converge to different saddle points generating visually different results.

4.3 Temporal Evaluation Metrics

For quantitative evaluation, we calculated the temporal loss of the output video using the following formula.

$$E(V) = \frac{1}{T-1} \sum_{t=1}^{t-1} \frac{1}{D} \sum_{i=1}^{N} O_t^i ||V_t^i - W(V_{t-1}^i)||^2$$

Where

$$V = Styled\,Video$$

$$T = Number\,of\,frames$$

$$D = Width * Height * Channels$$

$$W = Warping\,Function$$

$$O = Oclussion\,Mask$$

The temporal error is calculated for some output videos and the result of our model is compared with that of the Original Cycle GAN.

Video	Temporal Error	
	Cycle GAN(Ukiyo-e)	Ours(Ukiyo-e)
Video1	0.0152	0.0096
Video2	0.0045	0.0032
	Cycle GAN(Van Gogh)	Ours(Van Gogh)
Video1	0.0124	0.0084
Video2	0.0054	0.0031
	Cycle GAN(Monet)	Ours(Monet)
Video1	0.0115	0.0078
Video2	0.0038	0.0031

Table 4.1: Temporal Consistency comparison of Original Cyle GAN and Our model

Table 4.1 shows the results of the evaluation metric applied to the results of Cycle GAN and our model. Two videos downloaded from videvo.net(open source media) are styled using three different models of Cycle GAN and Our version. For all the three styles the comparison of the result is given in the table. It is observable from the comparison that the use of temporal loss using optical flow helps the model to achieve better temporal coherency.

Chapter 5

Conclusion

5.1 Main Contribution

The initial hypothesis of our study was to use optical flow information and formulate temporal loss to guide the model in achieving temporal coherency in the final output video. From the visual and quantitative results, it can be observed that using temporal loss helps the model to achieve better temporal coherency as compared to Cycle GAN. Therefore this model can be used to style real world videos to artistic style. We used three data sets as style reference data set which includes, Van Gogh painting, Monet paintings, and Ukiyo-e paintings, however, any data set of style images can be used to train the model provided there is some pattern followed in the paintings included in the data set. For example, a dataset of sketches can be used as style reference images.

We also provided an evaluation metrics to check the quality of the temporal coherence in any video. There are many other applications where the video generation tasks are performed and temporal coherency is an essential objective for any type of video generation model to achieve. The evaluation metrics proposed in this study uses optical flow estimation to calculate a loss value which indicates the degree of temporal smoothness in any type of generated video.

The general Neural style transfer models, discussed in chapter 2, use a single image as a style reference, so the final output video is generated using the style of a single style image. However, in this study, we proposed a way to use a collection of style images instead of a single image, so the style application is learned from a distribution of data rather than a single image.

5.2 Limitations

It took 65-70 hours for each model to train from scratch on 2 Nvidia RTX 2080Ti GPUs, therefore the longer training period only allowed to train using a relatively small dataset containing 10 video sequences. In some scenarios the model may not make enough changes to the input video, this can be resolved using a varied and larger data set.

The model is trained using the temporal loss between adjacent frames only which takes care of the short term temporal coherency however, in a video, many elements become occluded for few frames and reappear in later frames. In this scenario, ideally, the model should style this kind of reappearing element in the same manner, however, as the model compares the adjacent frames only there is no constraint on the model to handle this kind of long term temporal coherency, so the model can style the same element in different ways if the element is getting disoccluded after few frames.

5.3 Future Work

Larger and more varied data set can be used to increase the performance of the model. Using a larger data set the model can learn to handle the more varied type of input videos that will make the model more robust. The problem of long term temporal coherency is discussed in the Limitations section, to solve this issue a new loss value calculating temporal error between frames further apart from each can encourage long term temporal coherency, making the videos consistent over longer periods.

To further improve the temporal coherency, a new flow cycle loss can be introduced which calculates the flow between reconstructed frames using the second generator and

compare that with the original flow between input frames. This will be similar to the cycle loss, however instead of using the original frames and reconstructed frames, the optical flow map will be compared to calculate the loss.

Bibliography

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, 2016.

[2] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," *Pattern Recognition*, p. 26–36, 2016.

[3] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[5] E. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," 2015.

[6] A. Clark, J. Donahue, and K. Simonyan, "Adversarial video generation on complex datasets," 2019.

[7] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," 2016.

[8] H. Tang, D. Xu, H. Liu, and N. Sebe, "Asymmetric generative adversarial networks for image-to-image translation," 12 2019.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2016.

[10] Y. Li, N. Wang, J. Liu, and X. Hou, "Demystifying neural style transfer," 2017.

[11] E. Risser, P. Wilmot, and C. Barnes, "Stable and controllable neural texture synthesis and style transfer using histogram losses," 2017.

[12] G. Berger and R. Memisevic, "Incorporating long-range consistency in cnn-based texture generation," 2016.

[13] S. Li, X. Xu, L. Nie, and T.-S. Chua, "Laplacian-steered neural style transfer," *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, 2017.

[14] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *2013 IEEE International Conference on Computer Vision*, pp. 1385–1392, 2013.

[15] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," 2015.

[16] 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 *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7044–7052, 2017.

[17] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," 2017.

[18] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185 – 203, 1981.

[19] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," 2015.

[20] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," 2016.

[21] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," 2016.

[22] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," 2017.

[23] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez, "Symmetrical dense optical flow estimation with occlusions detection," 05 2002.

[24] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014.

[25] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," in *ECCV*, 2010.

[26] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," 2016.