University of BRISTOL

DEPARTMENT OF COMPUTER SCIENCE

Temporal Consistency in Low-Light Video Enhancement

[redacted]

Supervisor: Dr. Pui Anantrasirichai

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Thursday 4th May, 2023

Abstract

Improving video temporal consistency is a difficult topic in video processing. We detect flicker effects after enhancing video quality, and the output may reveal temporal inconsistencies, particularly when processing the video frame by frame because of memory limitation. The overall purpose of this research is to eliminate temporal inconsistencies in videos using deep learning approaches. We propose a style migration technique in this thesis that uses Transformer to improve overall brightness, and integrate LLVE and KinD's network architectures to provide time-domain consistency by controlling luminance changes between adjacent frames.

The previous attempts used a variety of complicated algorithms to improve temporal consistency. In this project, we first boost pixel luminance frame by frame for each frame of the video, and then perform pixel smoothing frame by frame. To evaluate the proposed method, we apply our approach to different computer vision tasks. Experiments show that the model we introduce reduces the effect of video flicker, resulting in smooth and fluid videos with visually improved image quality.

Contents

List of Figures

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Prof. Pui Anantrasirichai.

Supporting Technologies

A detailed summary of any third-party resources (e.g., hardware and software components) used during the project are listed as follows:

- I used the High Performance Computing (HPC) system BlueCrystal Phase 4 (BC4) provided by the University of Bristol.
- I used the Visual Studio Code, Python version 3.7, to write the code.
- I used the Torch.nn which is a sub library module of the PyTorch framework that can be used to build neural networks.

Notation and Acronyms

DL	:	Deep Learning
CNN	:	Convolutional Neural Network
DVP	:	Deep Video Prior
LLVE	:	Low Light Video Enhancement
NLP	:	Natural Language Processing
PE	:	Positional Encoding
CAPE	:	Content-Aware Positional Encoding
MSA	:	Multi-head Self-Attention
FFN	:	Feed-Forward Network
IAN	:	Illumination Adjustment Net
LSTM	:	Long Short-Term Memory
CMP	:	Conditional Motion Propagation
SID	:	See In the Dark
GPUs	:	Graphics Processing Units
DRV	:	Dark Raw Video
StyTr^2	:	Image Style Transfer with Transformer
KinD	:	Kindling the Darkness
ResNet	:	Residual Network
HPC	:	High Performance Computing
MABD	:	Mean Absolute Brightness Difference

Chapter 1

Introduction

In this project, we propose a method to improve temporal consistency of low-light video enhancement. This method is based on a two-branch transformer network architecture to extract the lighting information and content information of the video continuous frame images and constrain them separately. The video with enhanced luminance and consistent illumination is synthesized at the output to maintain temporal consistency to some extent. This chapter first describes the research background of temporal consistency, then explains the motivation, particular objectives, and significance of the work. Finally, the overall framework of this dissertation is described.

1.1 Temporal Consistency Overview

Temporal consistency is an important processing approach in computer vision for video enhancement. Many image processing algorithm approaches have demonstrated good performance in a single picture processing project[12, 17, 24, 38]. However, video frames are dealt with independently in video processing, which might result in variable visual quality, such as flickering or skipping. This issue is addressed by the temporal consistency technique, which makes the video appear smoother and more authentic by guaranteeing that each frame's visual quality is consistent with those of the previous frames. It can be used for a range of video enhancing tasks such as denoising[25], colour correction[22], and image stabilisation. Temporal consistency approaches, for example, can be employed in denoising to ensure that the noise elimination process does not produce flicker or additional aberrations into the video. Image stabilisation technologies[3] can be used in a similar way to maintain temporal consistency, so that the stabilised video seems smooth and fluent, with no abrupt jumps or shifts.

1.2 Motivation

In recent years, improving the temporal consistency of low-light video enhancement has necessitated its usage in a variety of applications, including surveillance, security monitoring systems, advertising, and cinema. Camera video aimed at night in low-light settings is of poor quality, with low light, low contrast, blurring, and colour departure. Many image processing algorithms perform well in individual picture processing, but when applied to video in frame-by-frame manner, they frequently exhibit temporal anomalies. Furthermore, over the years, researchers have researched and built broad frameworks for numerous video processing tasks, such as increasing video quality in low-light circumstances. Various general algorithms and approaches are used in these frameworks to improve and optimise video quality[3, 11, 39]. These frameworks are often designed to work on a per-frame basis, with each frame being processed separately to improve visual quality. A video's resolution determines how much memory it requires. Bonneel et al. [3] suggested a gradient domain technique for generating a temporally constant video sequence that is blind to the warping error between two subsequent output frames for a certain image processing algorithm from a series of processed frames impacted by flicker. Yao et al. [39] extended this approach by utilising additional information from the keyframe stack to track dynamic objects and handle obstructed regions. However, these two methods may not be applicable in practise because they presume that the output and processed videos are equivalent in the gradient domain (emphasises the variations between adjacent in pixel intensities). The reason for this is that some tasks may require significant changes in the gradient domain to achieve the desired results, and the gradient domain similarity assumption may not be flexible enough to handle a wide range of video processing tasks with different requirements. Besides, Eilertsen et al. [11] stated a CNN framework with pre-training, which can be used to fine-tune convolutional networks (CNNs) by imposing regularization on transform invariance.

Nowadays, the Deep Video Prior (DVP) method [23], the Image Style Transfer with Transformers (StyTr^2) method [10], and the Low Light Video Enhancement (LLVE) method [41] are widely used in image processing for achieving video's temporal consistency. Video priors refer to the inherent properties or features of a video, such as spatial and temporal redundancies, consistent motion patterns, and structural resemblances. However, DVP leverages the power of deep learning to capture and exploit video priors. The DVP method trains a convolutional network on video to utilize deep video priors. This allows the bulk of video information to be recovered before flickering artifacts are overfitted. Furthermore, the corresponding patches of convolutional network (CNN) outputs in video frames should be constant. The DVP approach is self-supervised learning, which does not need a training dataset and can only be trained on a single test video. However, it has the disadvantage that the method using maximum likelihood estimation has more iterations and is slower at generating time-consistent images for training. Besides, this strategy does not improve video brightness on its own and must be used in conjunction with other methods. StyTr^2, a transformer-based approach, considers the long-range dependencies in input images for image style transfer and generates images with artistic qualities inspired by a style reference while preserving the original content. To maintain temporal consistency, an extra loss term can be incorporated to promote consistent style changes between adjacent frames. This can be done by comparing the style representations of successive frames and reducing their disparities. The approach benefits from a self-attention mechanism [26] that enables the model to freely learn the global information of the input, facilitating a comprehensive understanding within each layer. Additionally, the transformer architecture captures the relationships between input shapes [29], with different layers extracting analogous structural information [31]. This method is easy to train and offers faster inference. The LLVE method maintains the time stability of low-light video improvements by utilizing static images. It learns and infers motion fields (optical flow) from a single image and synthesizes short-distance video sequences. However, the only use of static images for video enhancement ignores the movement information between the frames,which can lead to discrepancies between the enhanced frames and flashes or other visual artifacts when frames are combined into composite videos.This method indicates the motion parameters for improving single-image low-light video, while maintaining time consistency and is easily scaled to large data sets.

1.3 Aim and Objectives

This project aims to -

1. Demonstrate the referenced and improved StyTr^2 framework for generating images based on the corresponding patch of style images and content images using the Transformer style migration method.
2. Analyze quantitatively of the model by enforcing the feasibility of consistency on the distorted output and the results of the trained model in terms of enhanced quality and temporal stability.
3. Simulate an application scenario containing a moving object to demonstrate faster performance and execution than the traditional method of video boost tasks.

The objectives of this project is to -

1. Develop a transformer-based model trained on a substantial number of video datasets, which can enhance the brightness of low-light videos while preserving a certain degree of temporal consistency.
2. Use StyTr^2 as a framework reference, we replace the encoder module that obtains the style image based on a single image with one that obtains the optical flow variation characteristics based on adjacent frames.
3. Calculate the optical flow information from adjacent frames and finally transform into pixels by convolutional neural network (CNN). The network architecture of LLVE [41](image stream) and KinD [42] (single image) is also combined to achieve the extraction of optical stream and dark image enhancement by using two branches, and the long-term data memory of the training process is achieved by sharing parameters.
4. Train and test the model using the Dark Raw Video (DRV) dataset[6] to achieve temporal consistency by constraining the difference in brightness of adjacent frames.

1.4 Thesis Organisation

This thesis focuses on the generation of transformer models based on the training of a large number of video data sets and the application of these transformer models to dark videos, improving their brightness, ensuring a certain degree of temporal consistency and maintaining perceptual quality.

The remainder of dissertation is organised as follows.

- Chapter 2 describes background and related work in temporal consistency. It introduces the concept of encoder and decoder in style conversion transformer, the basic principle of CNN and how to translate into pixels, the current state of the art in improving temporal consistency and how to achieve temporal consistency by constraining the difference in brightness of adjacent frames.

- Chapter 3 presents methodology and the detailed implementation of the transformer to synthesize luminance enhanced and lighting consistent video at the output. Moreover, the LLVE method performs temporally consistent processing in low light video enhancement. It includes general frameworks, model implementation, the process of data preparation, model training and testing, and video preocessing.

- Chapter 4 presents the proposed method is evaluated by comparing the results of the study with those of other researchers.It also illustrates the limitations of the technique. The experiments demonstrate that the proposed method has better temporal consistency.

- Chapter 5 summarises the conclusions and speculates on the future work.

Chapter 2

Background and Related Work

2.1 Image Style Transfer with Transformers

The transformer architecture was first presented for machine translation jobs [35] rather than recurrent or convolutional neural networks, and it has since gained widespread adoption in numerous natural language processing (NLP) [30, 9] applications. Motivated by the transformer's success in NLP, numerous researchers have devised vision transformers to tackle a wide range of image and video-related tasks [37]. Compared with fully convolutional networks, transformer is a deep learning model that processes and generates data sequences using self-attention methods [35]. Encoder-decoder architecture is used in the majority of competitive neural sequence transduction models [8, 34]. This general architecture is followed by transformer, which employs stacked self-attention, point-wise, and completely linked layers for both encoder and decoder, as shown in the left and right portions of Figure 2.1.

StyTr^2 is a transformer-based solution for image style transfer that considers the long-range dependence of the input image. In Figure 2.2, StyTr^2 features two separate transformer encoders that generate domain-specific content and style sequences, as opposed to visual transformers for other vision tasks. At this point, the encoder extracts features from both the content and style images, and the decoder generates the stylized output. The multi-layer transformer decoder is employed after the encoders to stylize the content sequences based on the style sequences. Ultimately, a progressive upsampling decoder is employed to generate the final output. In the paper, We associate the illumination luminance as a special style, which is first applied to the enhancement of a single image frame. Then, it is necessary to extract the illumination information of the dark image frames through the network, and this architecture can be used for the final lighting information addition.

2.1.1 Image Patches and Content-Aware Positional Encoding

To harness the ability of transformers to capture long-range dependencies in picture characteristics for style transfer, the challenge is framed as a sequential patch generation assignment. Provided a content image $I_c \in \mathbb{R}^{H\times W\times 3}$ and a style image $I_s \in \mathbb{R}^{H\times W\times 3}$, both images are divided into patches which is analogous to tokens in NLP tasks. A linear projection layer is utilized to transform the input patches into an ordered pattern embedding ε, the dimension is defined as,

$$L \times C, \tag{2.1}$$

where L is the length of ε and C is the dimension of ε. L is formulated as,

$$L = \frac{H \times W}{m \times m}, \tag{2.2}$$

where $m = 8$ is the patch size. The linear projection layer efficiently captures the critical information from the input patches, which is suitable for subsequent processing stages.

To obtain structural details when utilising a transformer-based model, the positional encoding (PE) should be incorporated into the sequence of inputs. The attention value [35] for the i-th patch and the j-th patch is calculated as follows:

$$\begin{aligned} A_{i,j} =& ((\varepsilon_i + \mathcal{P}_i)W_q)^T((\varepsilon_j + \mathcal{P}_j)W_k) \\ =& W_q^T \varepsilon_i^T \varepsilon_j W_k + W_q^T \varepsilon_i^T \mathcal{P}_j W_k \\ &+ W_q^T \mathcal{P}_i^T \varepsilon_j W_k + W_q^T \mathcal{P}_i^T \mathcal{P}_j W_k, \end{aligned} \tag{2.3}$$

where P_i displays the i-th one-dimensional PE and W_q and W_k are parameter matrices for the derivation of the query and key. In 2D situations, the patch at pixel $(x_i,\ y_i)$ and the patch at pixel $(x_j,\ y_j)$ have the following positional relative relation:

$$\begin{aligned} &\mathcal{P}(x_i, y_i)^T \mathcal{P}(x_j, y_j) \\ &= \sum_{k=0}^{\frac{d}{4}-1} (cos(w_k(x_j - x_i)) + cos(w_k(y_j - y_i))), \end{aligned} \tag{2.4}$$

where $d = 512$ and $wk = 1/10000^{2k/128}$. The physical distance between two patches is the only factor determining their positional relative relationship.

Traditional PE is made for phrases that are logically arranged, whereas picture patches are arranged according to content. Content-Aware Positional Encoding (CAPE) depicted in Figure2.2 is scale-invariant and more appropriate for style transfer tasks, as opposed to sinusoidal PE, which only takes into account the relative distance within patches. The semantics of the visual content determines CAPE. It can be postulated that employing $n \times n$ positional encodings is sufficient for representing the image semantics . For an image $I \in \mathcal{R}^{H \times W \times 3}$, the fixed $n \times n$ positional encoding is rescaled to $\frac{H}{m} \times \frac{W}{m}$, as illustrated in Figure 3(b). This approach ensures that varying image scales do not impact the spatial relationship between two patches. The CAPE of the patch (x, y) is given as:

$$\begin{aligned} \mathcal{P}_{\mathcal{L}} &= F_{pos}(AvgPool_{n \times n}(\varepsilon)), \\ \mathcal{P}_{CA}(x, y) &= \sum_{k=0}^{s} \sum_{l=0}^{s} (a_{kl} \mathcal{P}_{\mathcal{L}}(x_k, y_1)), \end{aligned} \tag{2.5}$$

where$\mathcal{P}_{\mathcal{L}}$ is followed by the sequence ε which is accessible PE, F_{pos} is utilised as an accessible positional encoding function which is 1×1 convolution operation, $AvgPool_{n \times n}$ presents the average pool functions, n is set to 18, s is the amount of adjacent patches and a_{kl} presents the interpolation weight. Finally, the i-th patch is embed as the last feature at a pixel point (x, y) by adding $\mathcal{P}_{CA_i}$ to ε_i.

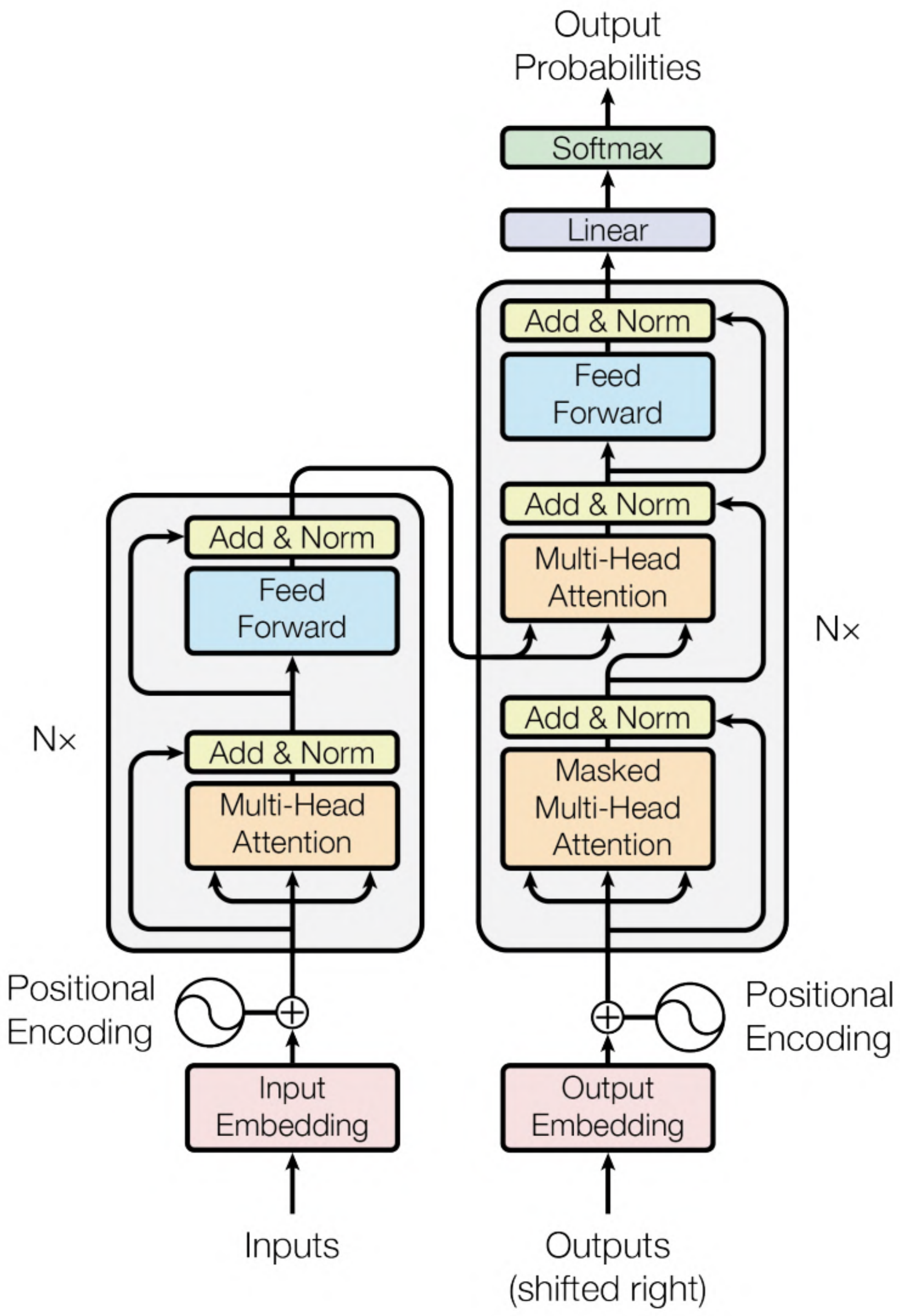


Figure 2.1: The Transformer model architecture [35]

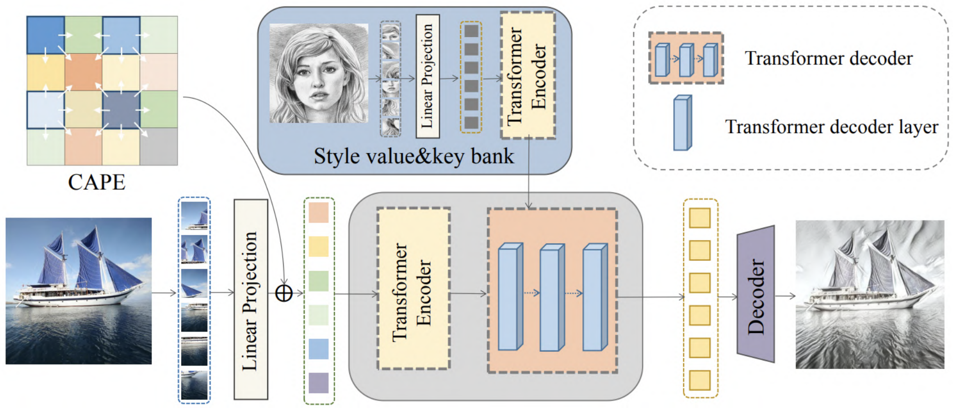


Figure 2.2: The StyTr^2 framework [10]

2.1.2 Transformer encoder

The encoder is made up of N = 6 identical layers. Each layer is divided into two sub-layers. One is a multi-head self-attention module (MSA), and the other is a simple fully linked feed-forward network (FFN) including positional information. Following layer normalisation [1], we use a residual connection [14] around both two sub-layers. The residual connection is the short-cut structure in the residual network (ResNet) shown in Figure 2.3, which is proposed to solve the network degradation problem. If there exists some k-layer network f that is the current optimal network, then a deeper network can be constructed whose last layers are only identity mapping of the output of the kth layer of the network f to achieve the same results. However, k is not necessarily the optimal number of layers, and then a deeper network can produce superior results. Identity mapping is not easy to fit for neural networks. Therefore, the residual structure requires the use of constant.

Following this, using a transformer-based structure to capture long-range interdependence of picture patches by learning successive visual representations. StyTr^2 has two transformer encoders for encoding domain-specific information, which are employed in the next stage to convert a sequence from a particular domain to another. The given sequence of input contents is initially sent into the converter encoder, which is:

$$Z_c = \varepsilon_{c_1} + \mathcal{P}_{C_{A_1}}, \varepsilon_{c_2} + \mathcal{P}_{C_{A_2}}, ..., \varepsilon_{c_L} + \mathcal{P}_{C_{A_L}}. \tag{2.6}$$

where ε represents the sequence and $\mathcal{P}_i$ presents the i-th one-dimensional positional encoding (PE). In the context of a transformer-based model, it is crucial to incorporate the PE within the input sequence to effectively capture the structural information.

The input sequence is represented as query (Q), key (K), and value (V):

$$Q = Z_c W_q, K = Z_c W_k, V = Z_c W_v, \tag{2.7}$$

where $W_q, W_k, W_v \in \mathbb{R}^{C \times d}$. The MSA is then calculated by

$$F_{MSA}(Q, K, V) = Concat(Attention_1(Q, K, V), ..., Attention_N(Q, K, V))W_o, \tag{2.8}$$

where N represents the amount of attention heads, $d = fracCN$, and $W_o \in \mathbb{R}^{C \times C}$ are learnable parameters.

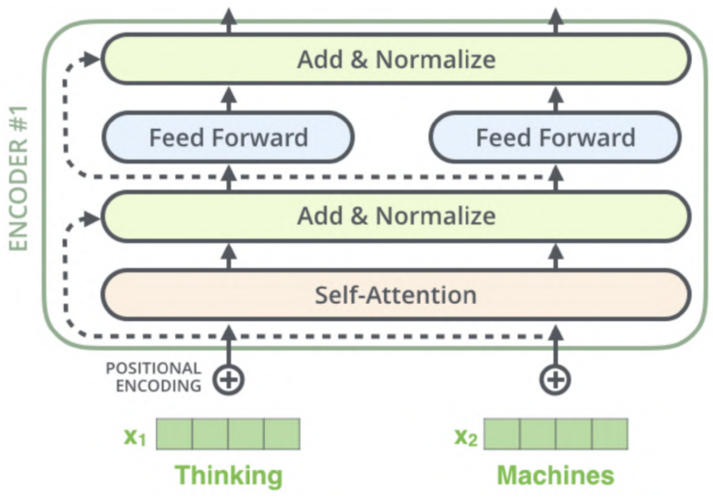


Figure 2.3: short-cut structure in ResNet with self-attention module [36]

2.1.3 Transformer decoder

The decoder is also made up of N=6 identical layers stacked on top of each other as shown in Figure 2.4. Two MSA layers and one FFN are present on each transformer decoder layer. In addition to two sub-layers in each layer, the decoder also inserts a third sub-layer which is called encoder-decoder attention. Similar to the encoder, a residual join is applied to each sub-layer, and then Layer normalization is performed. Each block additionally has layer normalisation in the end [35]. In encoder-decoder attention, Q comes from the previous output of the decoder, and K and V come from the output of the encoder. This masking, together with the positional offset of the output embedding, ensures that the prediction of position i can only depend on known outputs smaller than position i.

The transformer decoder is used to regressively transfer the encoded content sequence Y_c to the encoded style sequence Y_s. In contrast to the auto-regressive approach used in natural language processing (NLP) tasks, we use all of the sequential patches as input at once to forecast the output. The input to the transformer decoder consists of the $\hat{Y}_c = Y_{c_1} + \mathcal{P}_{CA_1}, Y_{c_2} + \mathcal{P}_{CA_2}, ..., Y_{c_L} + \mathcal{P}_{CA_l}$, and the $Y_s = Y_{s_1}, Y_{s_2}, ..., Y_{s_L}$. The query Q is generated by the content sequence, and the key K and value V are generated by the style sequence:

$$Q = \hat{Y}_c W_q, K = Y_s W_k, V = Y_s W_v. \tag{2.9}$$

Afterwards, the output sequence X of the transformer decoder is worked out:

$$\begin{aligned} X'' &= F_{MSA}(Q, K, V) + Q, \\ X' &= F_{MSA}(X'' + \mathcal{P}_{CA}, K, V) + X'', \\ X &= F_{FFN}(X') + X'. \end{aligned} \tag{2.10}$$

where F_{MSA} is a multi-head self-attention and F_{FFN} is a feed-forward network.

2.1.4 CNN decoder

A CNN decoder is a network architecture for reconstructing or generating output data from a set of learned features. Unlike the Transformer decoder, the CNN decoder relies

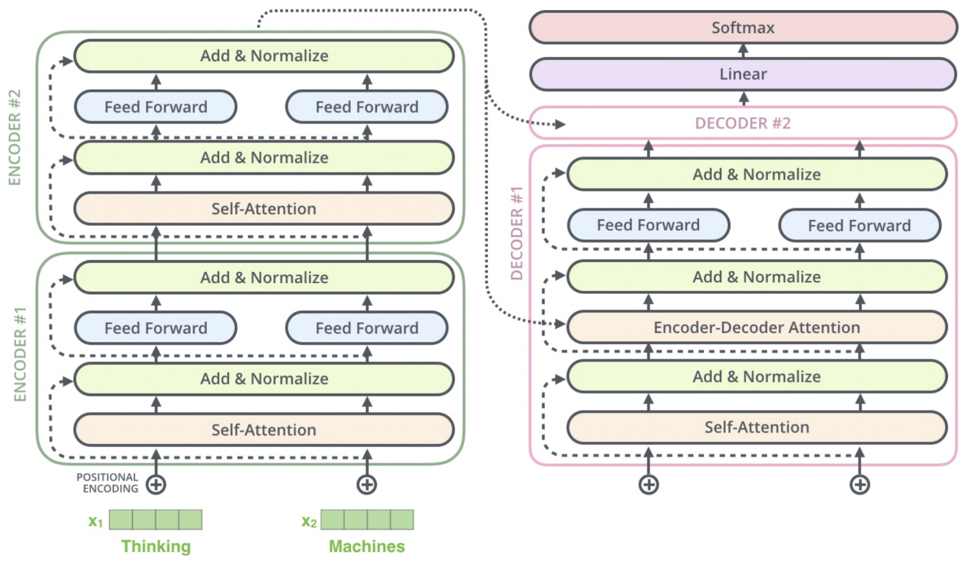


Figure 2.4: The decoder stacks [36]

mainly on convolutional layers for processing. A three-layer CNN decoder is generally used to refine the output of the Transformer decoder to produce a high-resolution output that is closely related to the target data[43].

In our work, we employ the CNN decoder as well. CNN decoder is used to convert the processing result of Decoder into image when using VGG. The CNN decoder is especially good at maintaining spatial details in pictures or videos. Furthermore, the CNN decoder has fewer parameters than the entire connected layer due to the shared weights and local connections in the convolutional layer, which reduces computational complexity. Additionally, additional mechanisms or loss functions must be introduced to perform consistent enhancement in consecutive frames, which may assist with reduce flicker or other temporal artifacts in the strengthened video.

2.1.5 Attention

An attention function can be characterized as a mapping that associates a query Q with a collection of key-value pairs K, V, ultimately producing an output. The output is calculated as a weighted total of the values, where the weight designated to every value is determined by a compatibility function relating the query to the appropriate key. The calculation is done in three steps, the first step is to calculate the similarity of the comparison Q and K, denoted by f,

$$f(Q, K_i), i = 1, 2, ..., m \tag{2.11}$$

Then, the obtained similarity f is normalized by Softmax operation,

$$\alpha_i = \frac{e^{f(Q,K_i)}}{\sum_{j=1}^{m} e^{f(Q,K_j)}}, i = 1, 2, ..., m \tag{2.12}$$

where the α_i obtained by softmax function are within (0, 1). Finally, for the computed weight α_i, all the values in V are weighted and summed to obtain the Attention vector of query $\sum_{i=1}^{m} \alpha_i V_i$.

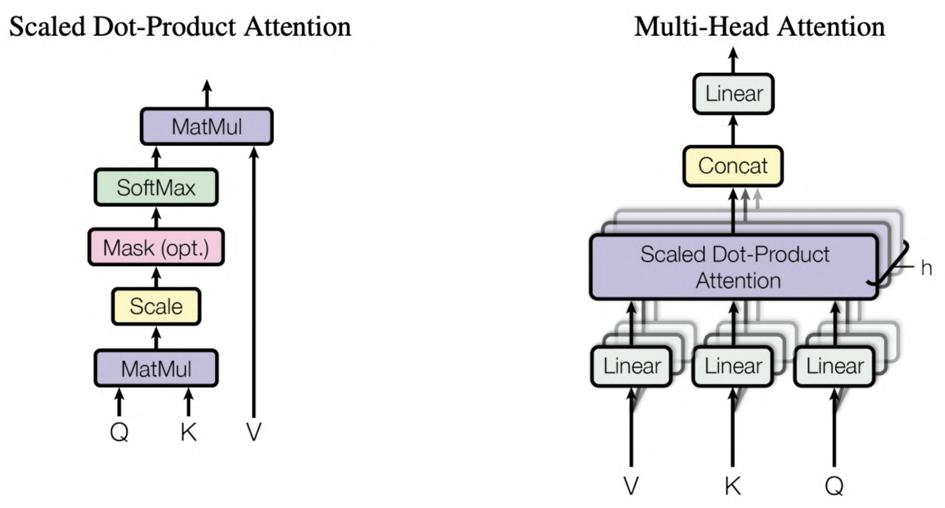


Figure 2.5: Scaled Dot-Product and Multi-Head Attention[35]

Additive attention and dot-product attention are the two most common prevalent functions. Additive attention employs a feedforward network to calculate the compatibility function. Although the theoretical complexity of both methods is comparable, dot-product attention proves to be significantly faster and more spatially efficient in practice. Scaled dot-product attention which is similar to the dot-product attention method but for the addition of a scaling factor, $\frac{1}{\sqrt{d_k}}$, is also explored. Besides, a set of queries is calculated by the attention function,the output matrix is:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.13}$$

It can be observed that both mechanisms exhibit similar performance when the value of d_k is small. However, when d_k is larger and not scaled, additive attention performs better than dot-product one[4]. We hypothesize that for larger d_k values, the dot product increases rapidly, leading the softmax function to enter a zone with an exceedingly minimal gradient. To mitigate this effect, we employ $\frac{1}{\sqrt{d_k}}$ to measure the dot product.

In contrast to employing a single attention function utilizing keys, values, and queries in the d_model dimension, it is advantageous to project queries, keys, and values h times into d_q, d_k, and d_v dimensions using distinct learned linear projection matrices. For each projected variation, the attention function is executed concurrently, generating output values in the d_v dimension. From Figure2.5, these outputs are subsequently concatenated and projected once more to derive the final values. Multi-head attention enables the model to concurrently concentrate on information stemming from various representation subspaces across distinct locations. By employing an averaging approach for a single attention head, this situation can be effectively mitigated. The formula of Multi-Head Attention is defined in Equation 2.8.

2.1.6 Perceptual loss

In StyTr^2[10], the generated results are required to preserve the initial content structures and the optical flow style features. Consequently, two distinct perceptual loss terms are established to evaluate the content disparity between the input content image I_c and the output image I_o, as well as the style disparity between I_o and the input style benchmark I_s. To create the content loss and the style loss, we utilize feature maps extracted by a pre-trained VGG model[15]. The following are the definitions of the content perceptual loss$\mathcal{L}_c$ and the style perceptual loss $\mathcal{L}_s$:

$$\mathcal{L}_c = \frac{1}{N_l} \sum_{i=o}^{N_l} \parallel \phi_i(I_o) - \phi_i(I_c) \parallel_2, \tag{2.14}$$

and,

$$\begin{aligned} \mathcal{L}_s = & \frac{1}{N_l} \sum_{i=o}^{N_l} \parallel \mu\big(\phi_i(I_o)\big) - \mu\big(\phi_i(I_s)\big) \parallel_2 \\ & + \parallel \sigma\big(\phi_i(I_o)\big) - \sigma\big(\phi_i(I_s)\big) \parallel_2, \end{aligned} \tag{2.15}$$

where $\phi_i(\cdot)$ represents characteristics taken from the i-th layer in a pretrained VGG19, N_l is the amount of layers, $\mu(\cdot)$ denotes the mean and $\sigma(\cdot)$ represents variance of extracted features.

2.2 Deep Video Prior

The Deep Video Prior (DVP) approach[23] is trained solely on a pair of the initial and modified videos, rather than relying on a large dataset. In contrast to many previous techniques[5, 13, 28] that ensure temporal consistency using optical flow, temporal consistency can be accomplished by training a CNN on a video, leveraging the Deep Video Prior. Let I_t represent the input video frame at time step t, and the related processed frame $P_t = f(I_t)$ can be generated by using the image processing method f (e.g., image colorization). The objective of blind video temporal consistency [3, 19] is to devise a function g that outputs a temporally constant video $\{O_t\}_{t=1}^T$ for $\{P_t\}_{t=1}^T$. Prior study[3, 20] typically employs correspondences to enhance temporal consistency, where correspondences between two frames using optical flow or PatchMatch[2] should exhibit comparable characteristics (e.g., color or luminance). L_{reg} is a regularisation loss specified the distance for minimum between relationships in the generated frames $\{O_t\}_{t=1}^T$. Additionally, a reconstruction loss, L_{data}, is utilized to minimize the spacing between $\{O_t\}_{t=1}^T$ and $\{P_t\}_{t=1}^T$. Consequently, for blind video temporal consistency, a loss function[20] or objective function[3] L, is usually used as:

$$L = L_{data} + L_{reg}, \tag{2.16}$$

$$L_{reg} = \sum_{t=2}^{T} \parallel O_t - W(O_{t-1}, F_{t\to(t-1)}) \parallel, \tag{2.17}$$

where W represents the warping function, $F_{t\to(t-1)}$ indicates the optical flow from I_t to $I_(t-1)$.

L_{reg}, can be implicitly obtained through DVP, as the outputs of a CNN on two identical patches are anticipated to be alike, and the same item appears similarly in different video frames. The DVP facilitates the recovery of most video information while mitigating flickering, ultimately preventing overfitting to all information and inconsistency artifacts.

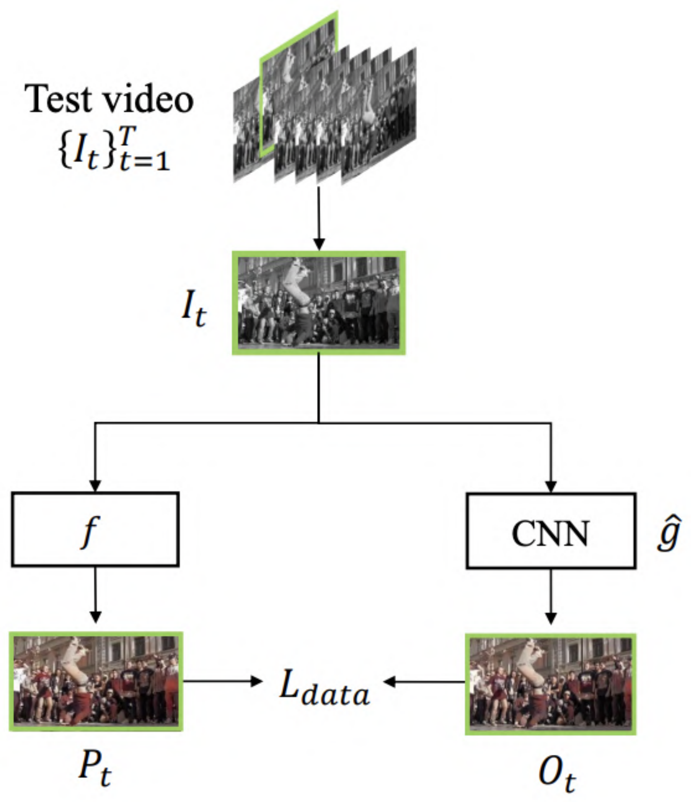


Figure 2.6: The DVP framework trains on the Test video directly[23]

As depicted in Figure2.6, DVP recommends employing a fully CCN, $\hat{g}(I_t;\theta)$, to emulate the original picture operator f while maintaining temporal consistency. Moreover, $\hat{g}$ is initialized at random and can be optimized using a single data term in each iteration, eliminating the need for explicit regularisation:

$$\arg\min_{\theta} L_{data}(\hat{g}(I_t;\theta), P_t), \tag{2.18}$$

where $L_(data)$ calculates the distance between $\hat{g}(I_t;\theta)$ and P_t. The training process should be halted when $\{O_t\}_{t=1}^T$ is proximate to $\{P_t\}_{t=1}^T$ and before artifacts (e.g., flickering) are overfitted. Employing this fundamental architecture, unimodal inconsistency can be mitigated. Within this framework, a neural network $\hat{g}$ and a data term $L_(data)$ are designed. In practice, U-Net[32] and perceptual loss[19] are utilized for all evaluated tasks. Furthermore, $\hat{g}$ is not limited to U-Net and other suitable CNN architectures can be applied as well.

2.3 Illumination Adjustment Net in KinD Network

KinD is a deep neural network, and Illumination Adjustment Net (IAN) is a sub-network in this network that focuses specifically on adjusting the illumination level of the input image[42]. First, IAN extracts the features of the low-light input image and the reference image using a network of independent encoders. To estimate the global illumination adjustment map, the combined features are processed through a series of decoder layers. The estimated illumination adjustment map is then multiplied element by element with the input low-light image to produce an adjusted image with improved brightness and contrast.

In Retinex theory[21], the image I can be considered as composed of two components which are the reflectance R and the luminosity L. The relationship between them is $I = R \circ L$, where $\circ$ specifies the wise product of the elements. After having paired illuminance maps, even without knowing the exact relationship between the paired illuminations, we can generally determine their intensity ratio, the mean of α is $\mu_\alpha = L_t/L_s$, which divided by element. This ratio can be utilised for training an adjustment algorithm from the source light L_s to the target light L_t as an indicator. When a lower light level is raised to a higher level, $\alpha > 1$, else $\alpha \leq 1$. Throughout the testing period, the user can specify a by themselves. The network is lightweight which is made up of three convolutional layers and one Sigmoid layer. As part of the network's input, the indicator α is visibly enlarged as a feature map. The loss of the IAN is shown as:

$$\mathcal{L}^{IA} :=\| \hat{L} - L_t \|_2^2 + \| | \nabla \hat{L} | - | \nabla L_t | \|_2^2, \tag{2.19}$$

where either L_h or L_l can be considered as L_t, and $\hat{L}$ is the modified illuminance map from L_s (L_h or L_l) to L_t.

2.4 LLVE

LLVE undergoes training an image-based model with image data and temporal consistency is implicitly incorporated into the optical flow produced by an individual image. Lai et al.[20] propose a general subsequent processing method to mitigate flicker, a deep network with a convolutional LSTM (ConvLSTM) module that explicitly employs FlowNet2[16] in the training phase to efficiently compute the flow rate during the training process, so that the estimated optical flow acquires temporal consistency from the sequence of videos.

The difference is that LLVE takes into account specific tasks, such as denoising, deblurring, colorization, or super-resolution. This method employs optical flow to represent the motion between video frames within an animated scene. It simulates this motion by warping adjacent image frames based on the corresponding optical flow. An ideal time-stabilized model should produce a transformed output with consistent transformations. Prior to training the network, it is essential to first predict a reliable optical flow using a pre-trained exemplar segmentation model, followed by an unsupervised model referred to as CMP [40] for optical flow prediction. Once the required optical flow is prepared, the image-based model is trained in a concatenated fashion. Prior research [6, 7, 18] has compiled low-light datasets and utilized a U-Net architecture[32] for training on these data. This basic and efficient model is well suited to evaluate the efficacy of the method and should adhere to the implementation outlined in SID [7]. All losses are employed by the l_1 loss, and the loss function utilized for network training can be described as a fusion of enhancement loss ($\mathcal{L}_e$) and consistency loss ($\mathcal{L}_c$):

$$\mathcal{L} = \mathcal{L}_e + \lambda \mathcal{L}_c, \tag{2.20}$$

where λ represents the weight that balances the contributions of the two loss components. More specifically, $\mathcal{L}_e$ and $\mathcal{L}_c$ are expressed as follows:

$$\mathcal{L}_e = \sum_{i=1,2} \| g(x_i) - y_i \|_1, \tag{2.21}$$

and

$$\mathcal{L}_c = \| W(g(x_1), f) - g(x_2) \|_1, \tag{2.22}$$

where $g(\cdot)$ denotes the network's forward operation. The variables x_i and y_i correspond to the input and ground truth, respectively, in the ith pass. The term f represents the optical flow generated for simulating motion.

Chapter 3

Methodology and Implementation

3.1 Overall Environment

The suggested model is generated with the PyTorch framework and trained them using GPUs. Torch.nn is a sub library module of the PyTorch framework that can be used to build neural networks and set up optimizers to train models. PyTorch is preferred over other deep learning systems such as TensorFlow. Because of its consistent versioning, easy debugging flexibility that allows it to experiment and iterate quickly, and more natural integration with Python systems. This makes the development experience smoother and more intuitive for researchers and is very popular among academics. Furthermore, the environment I used is PyTorch Version 1.13.1+GPU, Numpy version is 1.21.6, Opencv version is 4.5.1.48.

Graphics Processing Units (GPUs) are specialized processors designed to accelerate the rendering of visual effects. Their capacity for parallel processing enables the efficient handling of large datasets and models, resulting in significant reductions in training time for deep learning tasks. The GPU resources BlueCrystal Phase 4 (BC4) which is a high performance computing system provided by The University of Bristol is utilised during the training phase.With its ability to perform 600 trillion computations per second, BC4 is well-suited for activities requiring the utilisation of an Nvidia P100 GPU.

3.2 Framework Development

This study proposes a two-branch transformer network-based architecture for extracting lighting and content information from consecutive video frames separately. The approach aims to synthesize videos with enhanced luminance and temporal consistency at the output, while maintaining distinct constraints on each branch. In contrast to the single-frame image processing involved in the style migration algorithm, which primarily focuses on the individual frame transformation, the extraction of video-level features necessitates a more comprehensive approach. By sharing model parameters across multiple frames and processing temporal data, the algorithm can better understand the dynamic nature of videos and the relationships between consecutive frames. The overall structure of our framework is depicted as the Figure3.1, by adopting the Style Transfer Transformer in [10] ,LLVE [41]and KinD[42] network architecture.

The low light image is input to the left $frame_{t-1} \in R^{H \times W \times C}$ and the light image is input to the right $frame_t \in R^{H \times W \times C}$. Afterwards, the linear projection is used to create a succession of patches once the content and special style images have been partitioned into smaller, non-overlapping segments. By transforming the content and style images into a series of patches, the model can effectively focus on smaller regions of the image, allowing it to better capture the local features and intricacies of both the content and style. ε in patch sequence is defined as $\varepsilon_v \in R^c, i = 1, 2, ..., L$. Then, the content sequences, augmented with CAPE, are input into a content-converting Encoder 1 with the Equation2.5, while the specialized sequences corresponding to lighting brightness are input into a style-converting Encoder 2. Replace the encoder module in the StyTr framework, which originally obtains the style image based on a single picture, with an encoder module that obtains the light flow change characteristics based on adjacent frames. By considering nearby frames, the model can capture the temporal dynamics and connections between succeeding frames in the video. This preserves temporal consistency in the output, preventing flickering and other artifacts that may arise from processing individual frames independently. In the right part of Figure3.1, the multi-head attention is $Y_c{}' = Concat(A_1(Q, K, V), ..., A_N(Q, K, V))W_o$. Multi-head attention refers to the processing of input data by a number of concurrent attention mechanisms called heads. Each head individually calculates its own attention weights and values. The multi-head attention mechanism generates its final output by concatenating the attention values computed by each attention head, passing them via a linear transformation layer afterward. The model may capture a wider and more varied collection of contextual interactions in the input data by integrating the outputs of numerous attention heads, which enhances modelling performance. The Residual connections are employed to acquire the encoded content sequence Y_c, allowing for efficient information flow throughout the network:

$$\begin{aligned} Y_c &= FFN(Y_c{}') + Y_c{}' \\ \hat{Y}_c &= \{Y_{c_1} + \mathcal{P}_{CA_1}\}_{i=1,...,L}. \end{aligned} \tag{3.1}$$

In the left, the multi-head attention is $Y_f{}' = Concat(A_1(Q, K, V), ..., A_N(Q, K, V))W_o$ and the encoded flow sequence is $Y_f = FFN(Y_f{}') + Y_f{}'$.

After that, the encoded content sequence and the encoded flow sequence to the transformer decoder are combined and fed into the transformer decoder, which is charge of synthesizing the final output based on the information provided by both the content and flow sequences. By combining both content and flow information, the decoder can generate output that upholds the desired style and also ensures temporal consistency between consecutive frames in a video sequence. At the same time, the optical stream is added with the transformer decode. This allows the decoder to take into account the temporal dependencies between frames. The query Q, the key K and the value V are the input encoded sequences shown as:

$$K = Y_f W_k, V = Y_f W_v, Q = \hat{Y}_c W_q. \tag{3.2}$$

The output of the transformer decoder can be computed by Equation2.10.

3.3 Data Preparation

Obtaining dark video datasets can be challenging, as there is a limited number of publicly accessible datasets available for such purposes. In this research, the Dark Raw

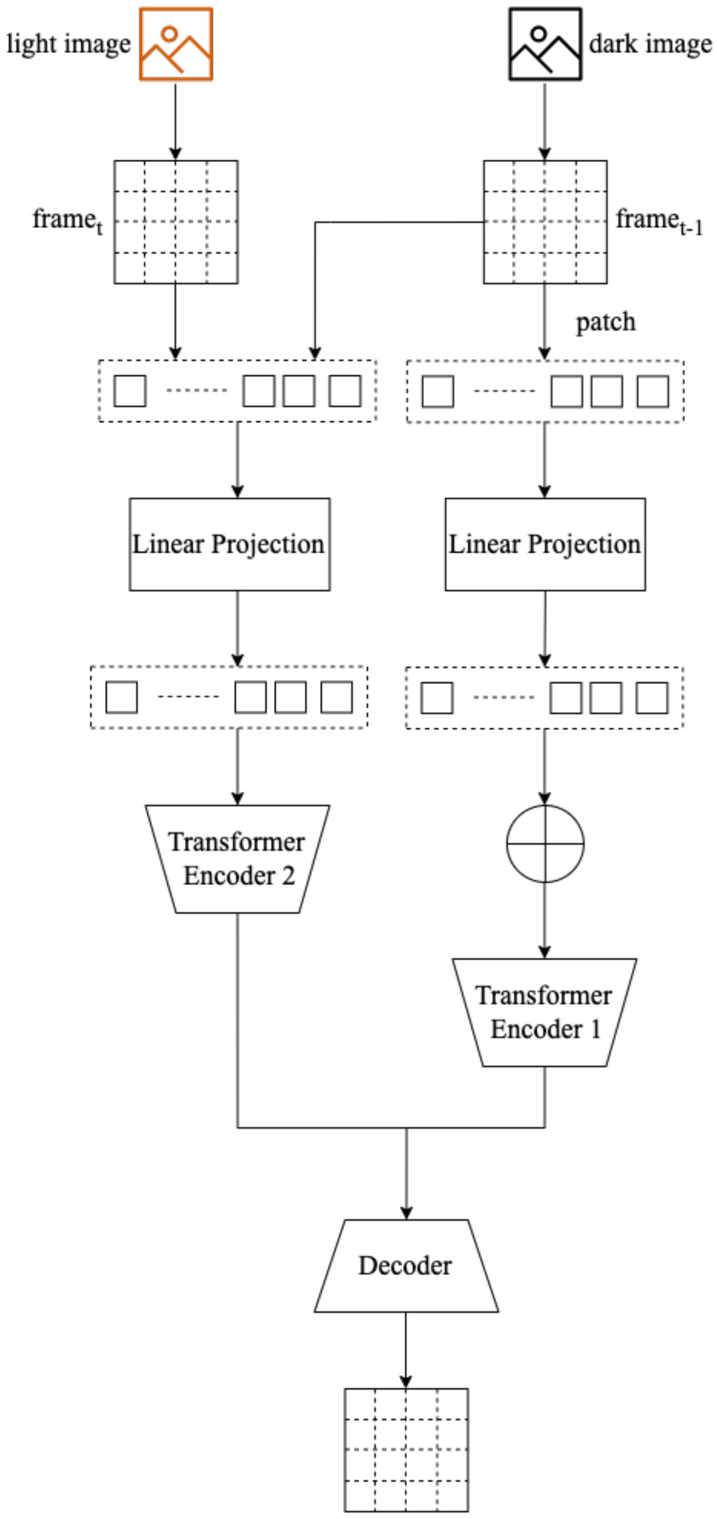


Figure 3.1: Framework Development

Video Dataset[6] is utilized, which comprises real-world footage captured using a Sony RX100 VI camera. In continuous shooting mode, the camera records raw photo sequences at a rate of 16 to 18 frames per second. The buffer can hold 110 frames in total, which is equal to 5.5 seconds of video at a frame rate of 20fps. The Bayer images have a resolution of 3672 x 5496 and the collection includes both indoor and outdoor settings. From this dataset, 200 sets of static videos in total were chosen for training and statistical analysis. These data were randomly partitioned into 130 groups for training purposes and the remaining 70 groups for testing. batch_size is the number of samples sent to the network at one time, when we process the video, the batch_size is 2, patch_size is the transformer applied to and is divided by 16x16.

3.4 Training

When training, there are both bright video and dark video as input, at this time, the upper frame in the Figure3.1 should replace the next $frame_t$ with the corresponding bright image frame $ground - truth_t$. Through the Equation2.14 and the Equation2.15, consistency loss $\mathcal{L}_c$ and optical flow loss $\mathcal{L}_f$ are illustrated as follows:

$$\mathcal{L}_c = \frac{1}{N_l} \sum_{i=o}^{N_l} \parallel VGG_i(X) - VGG_i(X_{gt}) \parallel_2, \tag{3.3}$$

and

$$\begin{aligned} \mathcal{L}_f = \frac{1}{n-1} \sum_{j=0}^{n-1} \Bigg(\frac{1}{N_l} \sum_{i=o}^{N_l} \parallel \mu\big(VGG_i(X_j)\big) - \mu\big(VGG_i(X_{j+1})\big) \parallel_2 \\ + \parallel \sigma\big(VGG_i(X_j)\big) - \sigma\big(VGG_i(X_{j+1})\big) \parallel_2 \Bigg) \end{aligned} \tag{3.4}$$

where N_l stands for the number of the layers, X_{gt} denotes the ground truth of dark image X and X_j, X_{j+1} are two adjacent frames.

In the Figure3.2, Light color is the loss curve calculated according to the number of iterations (related to the number of samples and training rounds) for each training, and the dark color is the average of several training experiment data. Besides, the training loss decrease over time, which indicates that the model is converging as it learns to minimise the error on the training dataset.

3.5 Overall Flowchart of the Training Algorithm

The flowchart as shown in Figure3.3 provides a comprehensive visual representation of the systematic process employed to train, validate, and test our model. In this section, the model is saved in a .pth file and is trained and tested on a dataset containing 200 video sets, with each set having approximately 100 frames. The dataset is split into two subsets: 130 sets are utilized for training, and the remaining 70 sets are employed for testing. During the training process, all the frames of the training videos are processed once per epoch. The loss is calculated and recorded every 20 frames throughout each epoch. After the training phase is completed, the best performing model is selected for the testing phase. The testing phase consists of evaluating the model on the 70 video

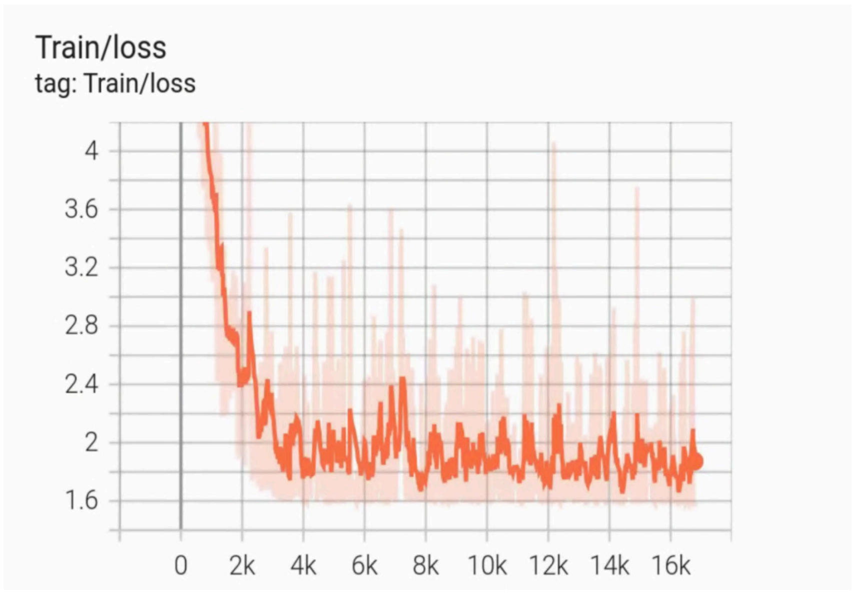


Figure 3.2: Training Loss Plot

sets reserved for testing. The performance of the model is assessed on each of these sets, and the entire process concludes once all 70 tests are completed. This comprehensive approach to training and testing ensures that the model is both robust and effective when applied to new data.

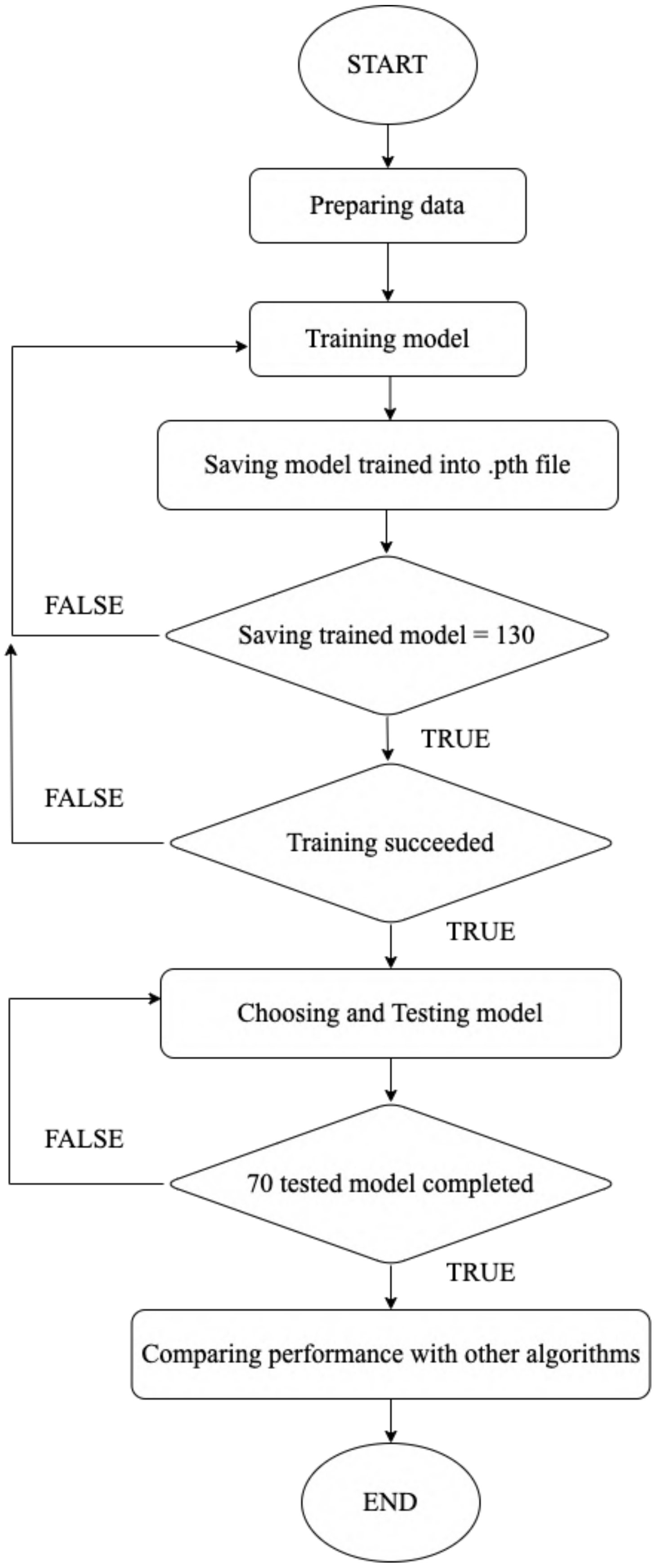

START
Preparing data
Training model
Saving model trained into .pth file
FALSE
Saving trained model = 130
TRUE
FALSE
Training succeeded
TRUE
Choosing and Testing model
FALSE
70 tested model completed
TRUE
Comparing performance with other algorithms
END

Figure 3.3: Overall Training algorithm decision-making flowchart

Chapter 4

Results and Evaluation

4.1 Evaluation metrics

4.1.1 Average Brightness(AB)

To evaluate the inter-frame consistency of the enhanced videos generated by our algorithm, we first calculate the difference in the AB values between each frame of the enhanced video and the corresponding frame of the original video.[27] After computing these differences for all frames, we then calculate the variance of these differences. A smaller variance indicates better inter-frame consistency in the results generated by our algorithm.

4.1.2 Temporal stability

The flow distortion error between two frames is utilized as a measurement.[20] The distinction of motion between two successive frames is measured by the flow distortion error. It is determined using an estimate of optical flow, which measures the obvious mobility of surfaces, borders, and items within a frame. The temporal stability of the video can be determined:

$$E_{warp}(V_t, V_{t+1}) = \frac{1}{\sum_{i=1}^{N} {M_t}^{(i)}} \sum_{i=1}^{N} {M_t}^{(i)} \parallel {V_t}^{(i)} - \hat{V}_{t+1}^{(i)} \parallel_2^2, \tag{4.1}$$

where $\hat{V}_{t+1}$ is the warped frame V_{t+1} and $M_t \in \{0, 1\}$ is a non-occlusion mask referring to non-occluded areas. To estimate the mask M_t, the occlusion detection technique[33] is employed. A video's warping error is determined as follows:

$$E_{warp}(V) = \frac{1}{T-1} \sum_{t=1}^{T-1} E_{warp}(V_t, V_{t+1}), \tag{4.2}$$

which represents the overall sequential average warping error.

4.1.3 Mean Absolute Brightness Differences(MABD)

A metric called MABD is employed to assess the temporal consistency of jobs involving video enhancement.[18] It calculates the average absolute difference in brightness values at each pixel point between two successive frames. Better temporal consistency is indicated by a lower MABD, which shows that the brightness of the video is more consistent across frames. It is determined by:

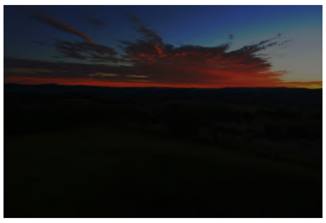
$$MABD(k) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} \mid br^{k+1}(i,j) - br^{k}(i,j) \mid, \tag{4.3}$$

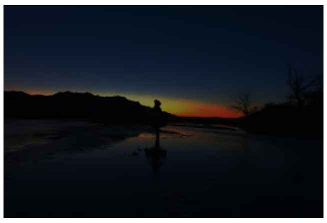
where M is height of the frame and N is width of the frame. The brightness of pixel (i,j) at frame k, with k being in the region of $1 \leq k < N_{frames}$, is known as $br^{k}(i,j)$.

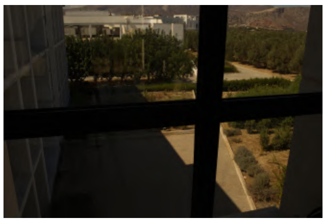
4.2 Qualitative evaluation

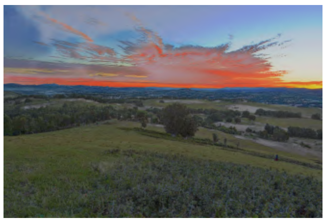
The test results for videos featuring intuitive still scenes are presented in Figure4.1. The input videos, denoted as(a),(b),(c), which are characterized by low lighting conditions and a blurred appearance. The enhanced video generated using KinD[42] method, labeled (d),(e),(f), exhibits less realistic colors and excessive exposure. The (g),(h),(i) video, produced using SMD[6] method which is very early, demonstrates that this approach is not effective for dark videos. Our proposed method, based on StyTr's transformer-based network[10], generates the (j),(k),(l) video. This network offers superior feature representation capabilities and effectively captures the long-distance dependencies within the input image. The resulting video exhibits overall improvements in color; however, it appears hazy and not as well-lit as desired. Despite these shortcomings, the video is subjectively acceptable to the human eye. In terms of video brightness enhancement, our method's performance closely aligns with the ground truth (m),(n),(o).

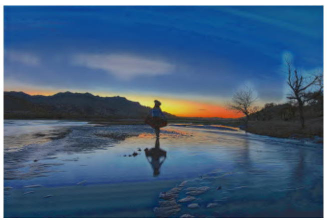
We gathered actual low-light movies to further confirm the reliability of the proposed method. The input image is captured in complete darkness, as seen in Figure4.2, and preserving temporal consistency between frames on dark video is essential for aesthetically smooth output. However, some enhancement algorithms may have difficulty maintaining consistency between frames, resulting in flicker, ghosting, halo artifacts, and colors may become distorted or unbalanced. Visually, KinD performs the best in contrast to other data in terms of lighting situations and shadows, and its colours appear realistic after enhancement without a sensation of overexposure. Although our output frames are not as colourful as KinD, it is obvious that the lighting is constant throughout, with no jarring changes, inconsistencies, or discontinuities.

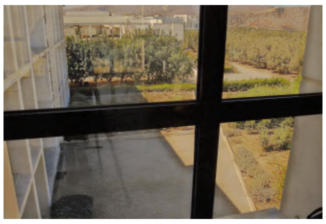

(a) Input

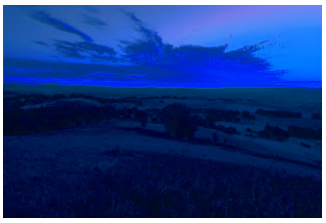

(b) Input

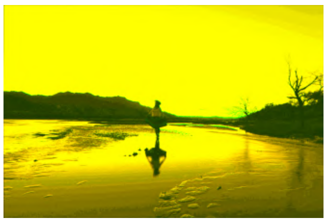

(c) Input

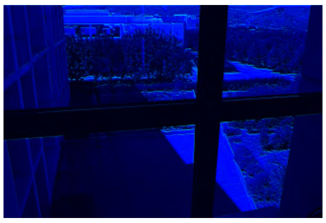

(d) KinD[42]

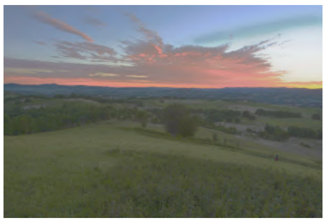

(e) KinD[42]

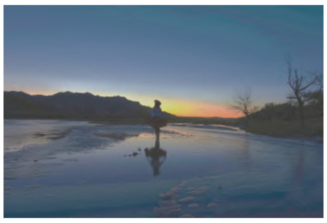

(f) KinD[42]

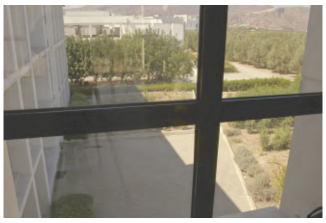

(g) SMD[6]

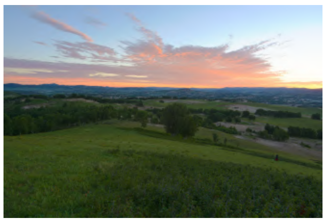

(h) SMD[6]

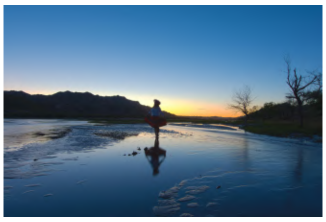

(i) SMD[6]


(j) Ours


(k) Ours


(l) Ours


(m) Ground Truth


(n) Ground Truth

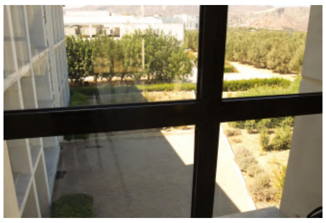
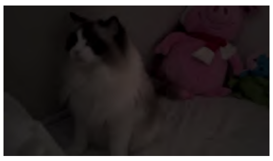
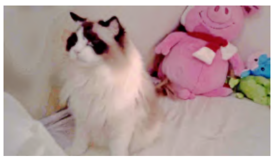
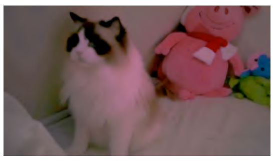
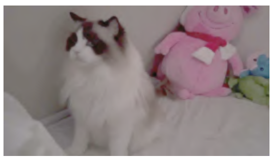
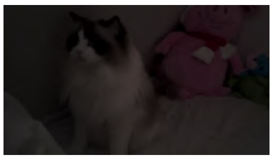
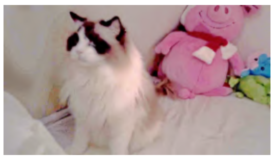
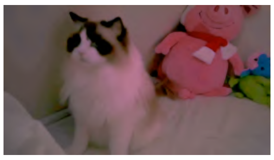
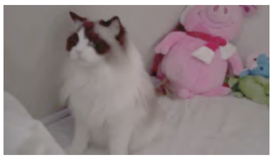
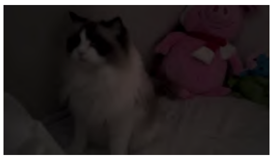
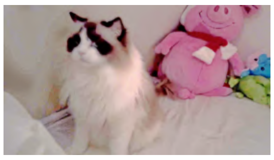
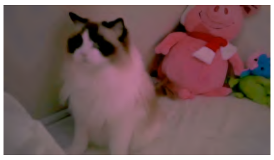
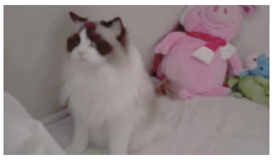
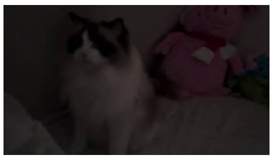
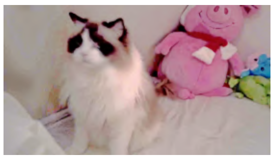
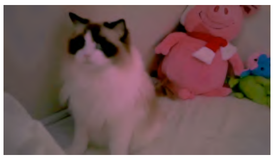
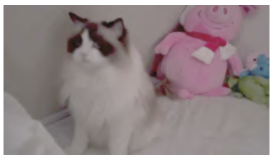

(o) Ground Truth

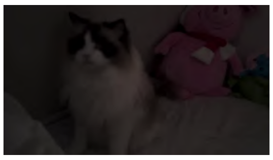
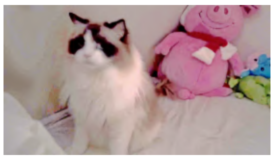
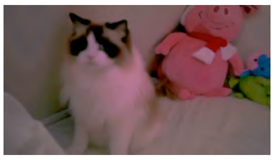
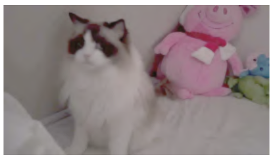
Figure 4.1: Contrast of the effect of increasing brightness


(a) Input

(b) KinD[42]

(c) SMD[18]

(d) Ours


(e) Input

(f) KinD[42]

(g) SMD[18]

(h) Ours


(i) Input

(j) KinD[42]

(k) SMD[18]

(l) Ours


(m) Input

(n) KinD[42]

(o) SMD[18]

(p) Ours


(q) Input

(r) KinD[42]

(s) SMD[18]

(t) Ours

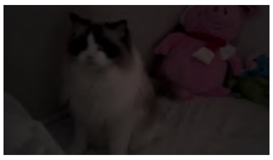
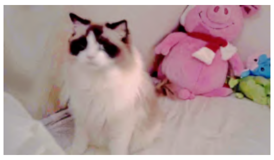
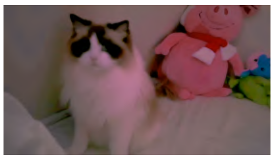
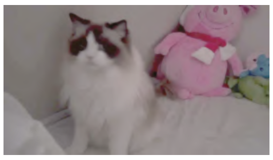

(u) Input

(v) KinD[42]

(w) SMD[18]

(x) Ours

Figure 4.2: Several frames for real data test. Six consecutive frames were selected and from top to bottom is Frame83 to Frame88.

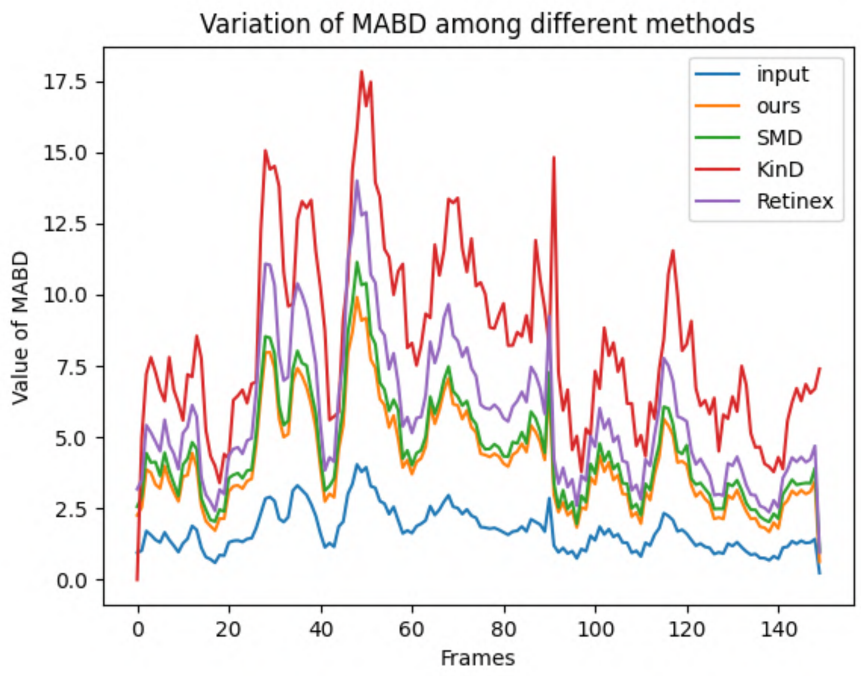


Figure 4.3: Caption

4.3 Quantitative evaluation

In this study, we aim to evaluate the effectiveness of our proposed approach using quantitative metrics, specifically, the Mean Absolute Brightness Difference (MABD). MABD is a metric that captures variations in consecutive video frames, effectively measuring changes in the video itself, such as abrupt alterations in lighting conditions. For a given video, MABD can be approximated as a temporal derivative across frames. The results of our MABD calculations are presented in the corresponding Figure4.3. Our algorithm demonstrates relatively minor changes in MABD This finding implies that our proposed method is effective in maintaining temporal consistency while enhancing low-light videos.

4.4 Failure cases

Initially, the decision was made to employ a Deep Video Prior (DVP)-based method [23] for this project. However, during the implementation process, several challenges were encountered as follows:

The implementation of the DVP method primarily involves smoothing the transition from input to output luminance. For cases where there is a substantial difference in lighting between the input and processed images, this method may result in blurry outputs, as illustrated in Figure4.5. Figure4.4 represents the input image, while Figure4.6 depicts the ground truth. Subsequently, an attempt was made to utilize the Transformer model exclusively, with dark images as input and brighter images as output4.7. Style transfer can be perceived as a generalized application of image processing in the realm of pure image processing. This led to further exploration of the use of Transformer-based network architecture for enhancing low-light videos.

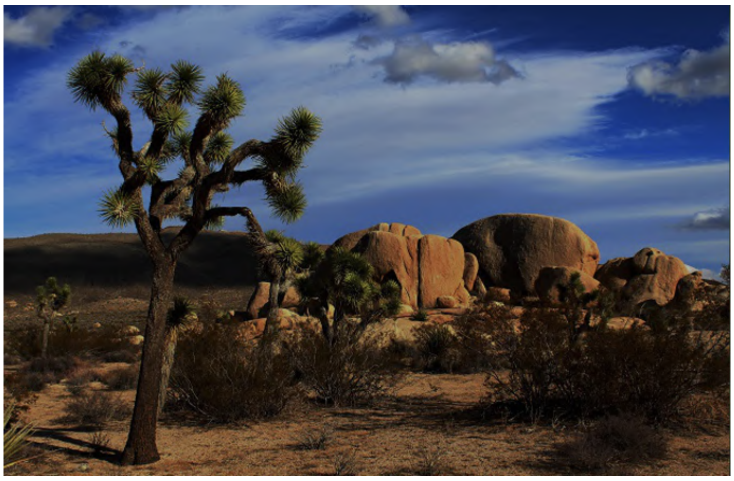


Figure 4.4: The input

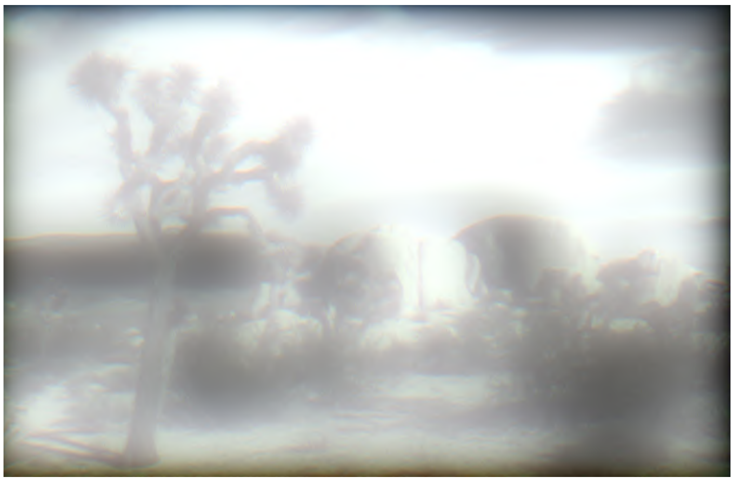


Figure 4.5: The output

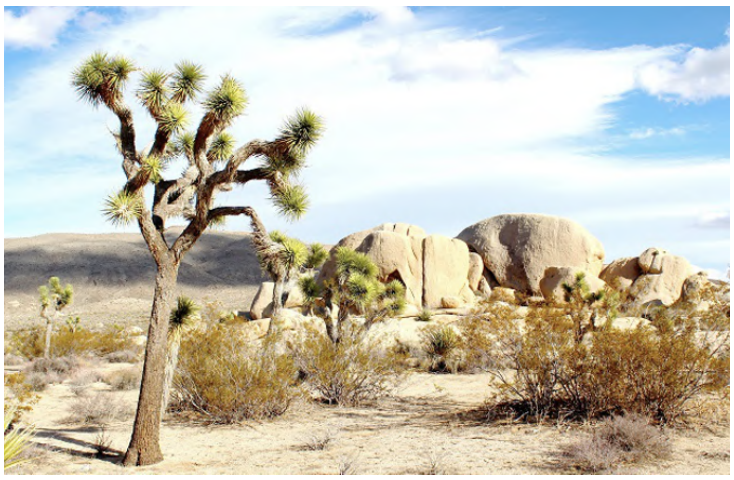


Figure 4.6: The ground Truth

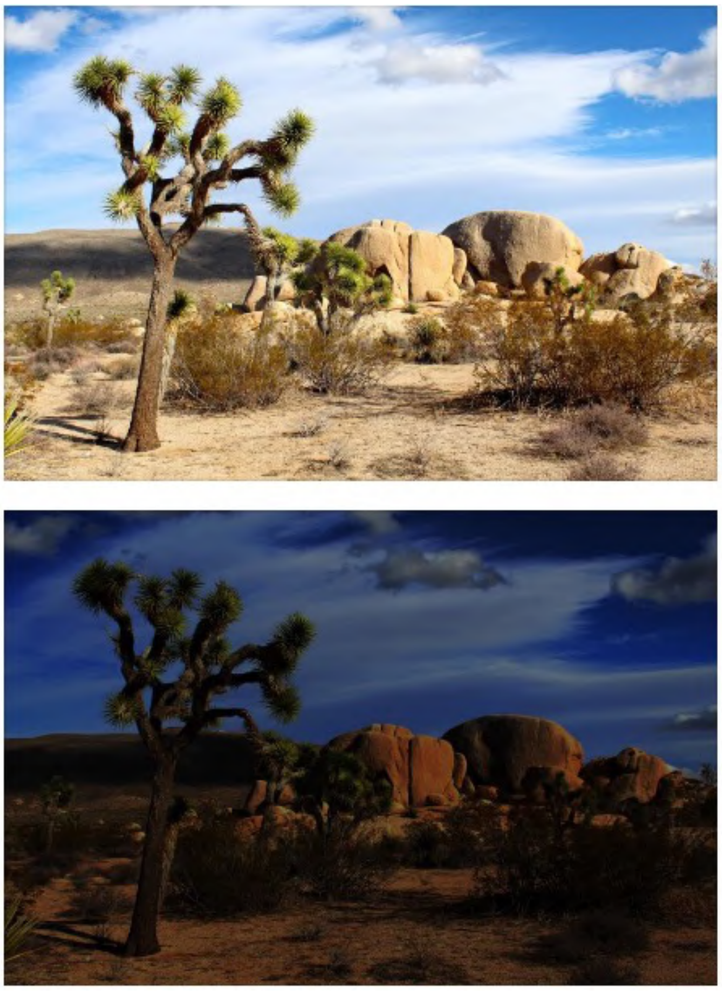


Figure 4.7: The first image above uses a dark image as the input, and the bright image as the style based on the output of the Transformer.

In this study, the Transformer model requires a substantial amount of data for pre-training; however, once trained, the model can be readily employed on untrained datasets, making it well-suited for unsupervised learning tasks. Additionally, the inference stage necessitates far fewer parameters compared to the DVP method. In contrast, the DVP approach requires obtaining a set of videos, processing them to generate brightness-enhanced versions, and using these enhanced versions as ground truth for training the model on specific datasets. Consequently, the DVP method must be retrained for each new set of dark videos, resulting in a considerably time-consuming training process. In summary, the Transformer model chosen for this study is more appropriate for generalizing to previously unseen data, offering a more efficient and adaptable solution for low-light video enhancement.

Chapter 5

Conclusion

5.1 Conclusion

This work presents a transformer-based model, trained on an extensive collection of video datasets, designed to enhance the luminance of low-light videos while preserving a degree of temporal consistency. Based on a two-branch transformer network architecture that effectively extracts and constrains the lighting information and content information of successive video frames independently. Moreover, considering the light brightness as a special style and applying it to the enhancement of a single image frame, it is then necessary to extract the brightness information of the dark image frame through the network. By employing this approach, the enhanced video output showcases improved luminance and consistent lighting, contributing to the overall quality of the video. For real data sets, both quantitative and qualitative results demonstrate that our model achieves an optimal balance between quality enhancement and temporal stability. The contribution of this research has been to confirm that taking inspiration from the style transfer approach provides new ideas for solving such problems. The findings of this study provide insights for the domain of temporal consistency in low-light video enhancement, offering substantial improvements in temporal consistency.

5.2 Future Work

Given the complexity of the study of temporal consistency in low-light video enhancement and the limited time available for this final project, there are plenty of aspects of my work that need improvement and correction. Specifically, I consider the following as shortcomings of this project and would like to do some future work if given the opportunity.

1. Little attention has been paid in this work to consider that the temporal inconsistency of the input video itself can have a large impact on the output if the dark video itself has temporal inconsistency. It is crucial to consider the impact of such inconsistencies within the input video, as they may significantly influence the quality and temporal coherence of the enhanced output. By examining the temporal inconsistencies inherent in the input video with separate comparisons of temporally consistent dark video and inconsistent dark video, we can better understand their impact on the enhancement process and develop more effective techniques to maintain temporal consistency in low-light video enhancement.

2. Bringing strength segmentation into my model would greatly improve time consistency but was not studied to try because of time constraints. By introducing strength segmentation for individual objects or entities within each frame, because instance segmentation provides more granular information about the scene, especially when dealing with low-light conditions, this information can be used to efficiently improve consistency between consecutive frames.

Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[3] Nicolas Bonneel, James Tompkin, Kalyan Sunkavalli, Deqing Sun, Sylvain Paris, and Hanspeter Pfister. Blind video temporal consistency. *ACM Transactions on Graphics (TOG)*, 34(6):1–9, 2015.

[4] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*, 2017.

[5] Albert YC Chen and Jason J Corso. Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 614–621. IEEE, 2011.

[6] Chen Chen, Qifeng Chen, Minh N Do, and Vladlen Koltun. Seeing motion in the dark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3185–3194, 2019.

[7] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3291–3300, 2018.

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[9] Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. Style transformer: Unpaired text style transfer without disentangled latent representation. *arXiv preprint arXiv:1905.05621*, 2019.

[10] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu. Stytr2: Image style transfer with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11326–11336, 2022.

[11] Gabriel Eilertsen, Rafal K Mantiuk, and Jonas Unger. Single-frame regularization for temporally stable cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11176–11185, 2019.

[12] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[13] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *2010 ieee computer society conference on computer vision and pattern recognition*, pages 2141–2148. IEEE, 2010.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[16] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[18] Haiyang Jiang and Yinqiang Zheng. Learning to see moving objects in the dark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7324–7333, 2019.

[19] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.

[20] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *Proceedings of the European conference on computer vision (ECCV)*, pages 170–185, 2018.

[21] Edwin H Land. The retinex theory of color vision. *Scientific american*, 237(6):108–129, 1977.

[22] Chenyang Lei and Qifeng Chen. Fully automatic video colorization with self-regularization and diversity. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3753–3761, 2019.

[23] Chenyang Lei, Yazhou Xing, and Qifeng Chen. Blind video temporal consistency via deep video prior. *Advances in Neural Information Processing Systems*, 33:1083–1093, 2020.

[24] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep joint image filtering. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 154–169. Springer, 2016.

[25] Ce Liu and William T Freeman. A high-quality video denoising algorithm based on reliable motion estimation. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III 11*, pages 706–719. Springer, 2010.

[26] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6649–6658, 2021.

[27] Feifan Lv, Feng Lu, Jianhua Wu, and Chongsoon Lim. Mbllen: Low-light image/video enhancement using cnns. In *BMVC*, volume 220, page 4, 2018.

[28] Ondrej Miksik, Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Efficient temporal consistency for streaming video scene analysis. In *2013 IEEE International Conference on Robotics and Automation*, pages 133–139. IEEE, 2013.

[29] Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2071–2081, 2022.

[30] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[31] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[33] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *Pattern Recognition: 38th German Conference, GCPR 2016, Hannover, Germany, September 12-15, 2016, Proceedings 38*, pages 26–36. Springer, 2016.

[34] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] A Waswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. Attention is all you need. In *NIPS*, 2017.

[37] Yifan Xu, Huapeng Wei, Minxuan Lin, Yingying Deng, Kekai Sheng, Mengdan Zhang, Fan Tang, Weiming Dong, Feiyue Huang, and Changsheng Xu. Transformers in computational visual media: A survey. *Computational Visual Media*, 8:33–62, 2022.

[38] Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (TOG)*, 35(2):1–15, 2016.

[39] Chun-Han Yao, Chia-Yang Chang, and Shao-Yi Chien. Occlusion-aware video temporal consistency. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 777–785, 2017.

[40] Xiaohang Zhan, Xingang Pan, Ziwei Liu, Dahua Lin, and Chen Change Loy. Self-supervised learning via conditional motion propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1881–1889, 2019.

[41] Fan Zhang, Yu Li, Shaodi You, and Ying Fu. Learning temporal consistency for low light video enhancement from single images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4967–4976, 2021.

[42] Yonghua Zhang, Jiawan Zhang, and Xiaojie Guo. Kindling the darkness: A practical low-light image enhancer. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1632–1640, 2019.

[43] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.