Fast Video Multi-Style Transfer

Wei Gao[1] Yijun Li[2] Yihang Yin[1] Ming-Hsuan Yang[3]
[1]Beihang University [2]Adobe Research [3]UC Merced

Abstract

Recent progress in video style transfer has shown promising results which contain less flickering effects. However, existing algorithms mainly trade off generality for efficiency, i.e., constructing one network per style example, and often work for short video clips only. In this work, we propose a video multi-style transfer (VMST) framework which enables fast and multi-style video transfer within one single network. Specifically, we design a multi-instance normalization block (MIN-Block) to learn different style examples and two ConvLSTM modules to encourage the temporal consistency. The proposed algorithm is demonstrated to be able to generate temporally-consistent video transfer results in different styles while keeping each stylized frame visually pleasing. Extensive experimental results show that the proposed method performs favorably against single-style models and some post-processing techniques that alleviate the flickering issue. We achieve as many as 120 stylization effects in a single model and show results on long-term videos that consist of thousands of frames.

1. Introduction

Video artistic style transfer aims to transfer the style of a reference image onto another input content video. For example, one can make the real video look like it is recorded under a fantasy world. So far, existing techniques of video style transfer are mainly limited in the number of styles to select. Those methods [14, 26, 2] only support single-style transfer without providing other style choices. In other words, users need to spend extra time to retrain model parameters when they want to acquire a new stylization effect. Other algorithms related with multi-style transfer [6, 10] or arbitrary style transfer [36, 22] only work for still images. When applied to videos in a frame-by-frame manner, they generate results with severe flickering effects which lack the temporal consistency.

In this work, we introduce a video multi-style transfer approach, which can handle multiple stylization effects while generating coherent video results. We extend image single style transfer to video multi-style transfer with the assistance of instance normalization. The instance normalization layer has demonstrated its power in artistic style transfer. In our approach, we improve the one-to-one instance normalization layer with one-to-many instance normalization layer. The one-to-many instance normalization layer can take a single layer feature as input and then produce multiple features to represent different style examples.

In addition to learning different styles, another challenge is to produce temporally consistent results. Previous works [26, 5, 2] mainly take advantage of optical flow to constrain neural network to produce coherent video results. However, we observe that by simply adding this flow-based regularization into our multi-style framework, it makes the network training harder when learning multiple styles at the same time, which is unable to generate temporally coherent stylization results. Therefore, we propose to learn a recurrent network with a convolutional long short term memory (ConvLSTM) [30] layer to keep the output videos temporally stable. We minimize a short-term and a long-term temporal loss between output frames and utilize perceptual loss from the pre-trained VGG [31] network to encourage the stylization effect on output frames. We demonstrate that the proposed method can be applied to videos with arbitrary lengths without computing the optical flow during inference time.

The main contributions of this work are summarized as follows:

- We utilize a multi-instance normalization block to learn 120 different style examples in one network.
- We embed two ConvLSTM modules to encourage the short- and long-term temporal consistency.
- We demonstrate that the proposed method performs favorably against existing models and especially show results on long-term videos.

2. Related Work

Artistic Image Style Transfer. The goal of image artistic transfer is to simulate the style of the reference image while maintaining the content of the source image. The seminal work by Gatys *et al*. [12, 11] demonstrated impressive visual styles by matching global feature statistics

in convolutional layers of VGG [31]. It is based on a slow optimization process that iteratively updates the image to minimize a content loss and a style loss computed by a loss network, which takes minutes to coverage even with modern GPUs. Several improvements have been made after that. The approach in [33] trains generative feed-forward models with complex and expressive loss functions to accelerate the speed of style transfer. Johnson *et al*. [17] proposed perceptual loss to achieve real-time artistic single style transfer models.

The work in [10] introduced the idea of conditional instance normalization. Ulyanov *et al*. [33] simply apply the instance normalization to transfer stylized effect to content images. Li *et al*. [21] design multi-style transfer model by the use of instance normalization. Huang and Belongie [36] utilize adaptive instance normalization to transfer arbitrary style with feed-forward networks to improve the efficiency. Our proposed approach inherits the instance normalization to achieve multi-style transfer effect. Li *et al*. [22] further change the gram statistics of intermediate feature to achieve stylized result by the aid of encoder-decoder structure and Lu *et al*. [24] preserve more content information to achieve visually pleasing results compared with [22]. The work of [7] proposed style swap to transfer arbitrary style to a specified content image. Sheng *et al*. [29] incorporates [22, 36, 7] to generate high-quality stylized images. The algorithm in [13] is an interesting technique to shuffle deep features of style image for arbitrary style transfer. However, when applying those methods to each frame of the input video, they generate temporally inconsistent results with obvious flickering effects.

Video Style Transfer. Generating a stylized video can be regarded as a conditional video generation problem [32] where one of the key tasks is to guarantee the temporal coherency in results. There are several approaches [28, 27] that make efforts to improve the temporal stability of CNN-based image style transfer. The work of [14, 26, 2] train feed-forward networks by jointly minimizing content, style and temporal warping losses. These methods, however, are limited to building one network for one specific style example. Chen *et al*. [5] need to make use of flow and mask networks to blend the intermediate features of the stylized network during inference stage. Although it extends single video style transfer into video multi-style transfer, it depends on the performance of flow network and can not achieve real-time efficiency.

Post-processing for temporal consistency. Recently, several post-processing methods have been proposed to improve temporal consistency for generated videos. Dong *et al*. [8] use optical flow to keep the coherence of output video. Bao *et al*. [3] propose a motion estimation and compensation driven neural network to enhance the video stability. Lai *et al*. [19] take the stylized output and content as input to maintain the temporal consistency of stylized result. However, these post-processing methods rely on the generated results and cannot process the video with severe flickering well.

Table 1. Differences between our approach and other example-based video stylization methods.

method	Style Number	Post Processing	Require Optical Flow (at test time)
[2]	1	×	×
[14]	1	×	×
[5]	32	×	✓
[19]	∞	✓	×
Ours	120	×	×

To summarize, we compare current different video style transfer methods with ours in Table 1. Our method outperforms the approach of [5] in real-time, memory requirements and the number of style selections. In comparison with the work in [19], our algorithm is not a post-processing technique to solve the flickering artifacts generated by other algorithms.

Video-to-Video translation. While we mainly focus on example-based video stylization methods, there is another line of research on domain-based video translation [34] which also aims at altering the style of a video (e.g., game videos to real videos). This builds upon the previous image-to-image translation work [16, 38, 35, 15, 18, 20, 39] and introduces flow-based temporal constraints to generate stable translation results. However, they require a training dataset of style images in the same domain and we target on translating the style of one specific style example.

3. Proposed Method

Figure 1 shows our proposed recurrent network architecture, which consists of several modules: encoder-decoder, multi-instance normalization block and ConvLSTM. Our model takes each frame and a certain kind of style specified by users as inputs, and then produces temporally consistent stylized output. We use a basic encoder-decoder architecture with a few residual blocks embedded in the middle. Compared with the heavy VGG-like autoencoder used in [22, 36], our network is more light-weighted on the memory usage and flops, and thus our approach achieves the real-time speed during the inference.

3.1. Multi-Layer Instance Normalization

In the existing single style video transfer framework, users have to retrain a new neural network for each newly added style. This process requires much more computation resources and time. For instance, in the work of [2], it needs

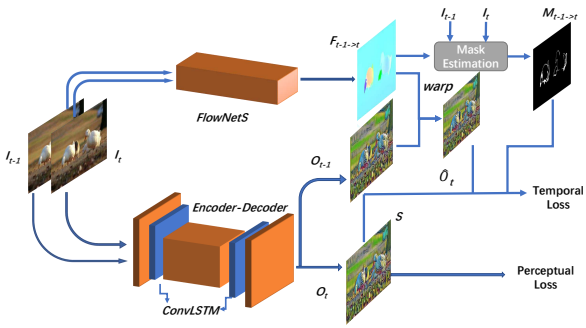

Figure 1. Overview of our framework. We embed two ConvLSTM modules into the encoder-decoder as our recurrent network. This network takes I_t and previous hidden state as input and generate output frame O_t. The output frame O_t keeps the temporal consistency with the previous output frame O_{t-1}. The ConvLSTM stores the current hidden state to predict the O_{t+1} in the next step. We optimize the entire network with the perceptual loss, long-term temporal loss, short-term temporal loss and total variation loss.

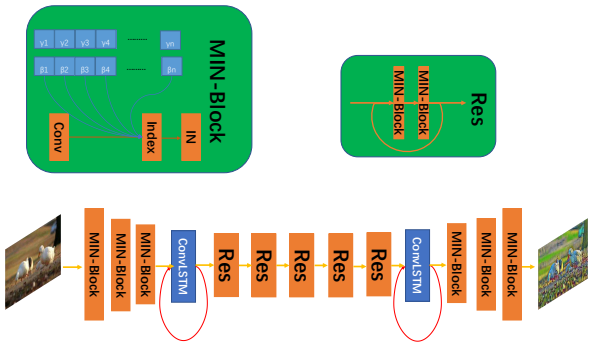

Figure 2. Architecture of the recurrent network. One ConvLSTM is embedded after the three MIN-Block(downsample) modules. After five residual blocks, we plug another ConvLSTM before three MIN-Block(upsample) modules. Illustration of MIN-Block: We insert a multi-IN layer after convolutional layer to transfer feature into a new stylized feature space. For the inference, users can specify the value of index to realize multi-style transfer.

at least 8 epoch to converge on COCO [23] dataset and 4 epoch computation time on Sintel [4] dataset. To solve this issue and enable the network to learn multiple styles, we use the instance normalization layer with multiple sets of parameters $\{\gamma_i, \beta_i\}$, as shown in Figure 2. Each style is associated with a certain $\{\gamma_i, \beta_i\}$ pair and selected by the index. Similar to [10], each learned parameter pair $\{\gamma_i, \beta_i\}$ in the instance normalization layer can be regarded as the embedding of a specific style. Hence, the index naturally represents a kind of style for users to control. By controlling stylization efffect with the parameters of instance normalization and sharing convolutional parameters with various style transfer network, we significantly save the memory and computation time for each style.

3.2. Recurrent Network

To fully take advantage of temporal information, we embed a recurrent module in our encoder-decoder network to generate output frames in a sequential fashion. The idea of recurrent network mainly combines all previous frames information and current frame information to infer current output.

To exploit the spatiotemporal information in videos, two ConvLSTM modules are inserted into the encoder-decoder network. The ConvLSTM module, commonly used in spatiotemporal sequence forecasting problem, is capable of compressing the whole previous input sequence into a hidden state tensor and then unfolds this hidden state to forecast the current state. We can easily utilize the ConvLSTM to compute the hidden state to represent spatiotemporal information of the current whole input content frames. Furthermore, the ConvLSTM module can extract the spatial-temporal information of video clips better with the help of temporal-related loss function. In addition, a more detailed discussion about the location and number of ConvLSTM modules is presented in Section 4.4.4.

3.3. Loss Function

Our algorithm aims to solve the temporal flickering problem in stylized frames while preserving the stylization effect. Hence, we train our network with the following three losses.

Perceptual loss. We compute the gram loss between O_t and style image S and L_2 loss between O_t and content frame C from a pretrained VGG [31] classification network, which is widely used in style transfer work to preserve the main structure in the content frame and capture the style statistics in the style image at the same time. The perceptual loss is defined as:

$$L_p = \sum_{n \in l_c} \|\phi^n(O_t) - \phi^n(C)\|_2^2 + \lambda \sum_{n \in l_s} \|Gram(\phi^n(O_t)) - Gram(\phi^n(S))\|_2^2, \quad (1)$$

where we compute L_2 loss on layers in l_c including relu2_1 layer and compute gram loss on layers in l_s including relu1_1, relu2_1, relu3_1, relu4_1 layer. $\phi^n(x)$ represents that the output of the layer n of VGG16 and $Gram(X)$ equals XX^T. λ is the parameter to balance two losses which is set as 10^5 here.

Short-term temporal loss. In our algorithm, the short-term temporal loss is defined as the warping error between two consecutive frames. The formulation is as follows:

$$L_{short} = \sum_{t=2}^{T} M_{t \to t-1} \|Warp(O_t, F_{t \to t-1}) - O_{t-1}\|_1, \quad (2)$$

$$M_{t \to t-1} = exp(-\alpha \|I_t - Warp(I_{t-1}, F_{t \to t-1})\|), \quad (3)$$

where O_t is the t-th frame result which represents the mutual visibility area of two frames. We use pretrained FlowNetS [9] to calculate the optical flow $F_{t\to t-1}$ during training process. In our experiment, we set α as 50 and use bilinear grid sample to warp frames.

Long-term temporal loss. Chen *et al*. [5] has shown that short-term temporal loss cannot guarantee the long-term coherence. To apply long-term temporal loss is an efficient way to keep the stability of video with many frames. Similar with the work of [5], we simply formulate long-term temporal loss as the warping error between two frames with long intervals. In our experiment, we calculate the temporal loss between the first frame and the t_{th} frame to maintain the long-term consistency:

$$L_{long} = \sum_{t=2}^{T} M_{t\to 1} \left\| Warp(O_t, F_{t\to 1}) - O_1 \right\|_1, \quad (4)$$

where T is set as 10 in our experiments.

Total variation loss. In addition, we also use the total variation regularizer to encourage the spatial smoothness of output. The total variation is defined as:

$$L_{tv} = \sum_{t=1}^{T} \frac{1}{HWC} \left\| O_t(x,y) - O_t(x-1,y) \right\|_2 + \frac{1}{HWC} \left\| O_t(x,y-1) - O_t(x,y) \right\|_2, \quad (5)$$

Overall, the network is trained to minimize the weighted sum of all previous mentioned losses:

$$L = \lambda_p L_p + \lambda_s L_{short} + \lambda_l L_{long} + \lambda_{tv} L_{tv}, \quad (6)$$

where $\lambda_p, \lambda_s, \lambda_l$ and λ_{tv} are set as 1, 100, 100, 0.001 respectively.

3.4. Progressive Network Training

We find that it is difficult to directly optimize the overall loss in (6), especially when learning multiple styles together. To effectively train our network, we adopt a two-stage progressive training strategy. First, we train the network without the temporal loss. This enables the network to first focus on learning style transferring. Unlike the case of learning one single style, multi-style learning in one network is much harder where the unique style information needs to be gradually distilled in different instance normalization layer so that we can select the desired style by using different normalization parameters. Then, we fine-tune the network with the temporal loss. The newly added short-term and long-term temporal constraints will encourage to generate stable transferred results in consecutive frames. Implementation experimental details are presented in Section 4.1.

4. Experimental Results

4.1. Implementation Details

First, we train a multi-style image transfer network without temporal loss. We use 80,000 images from COCO dataset [23] as the content images. Given a set of target style images, we train our multi-style transfer network with a batch of 16 content images for 40 epochs. Adam [25] optimizer is used with learning rate 0.001. To avoid the over-fitting problem, the weight decay is set as 0.0005. Note that our ConvLSTM modules are not optimized in this stage because we hope the encoder-decoder module can preserve the high level abstract content information of image when introducing stylized effect well instead of focusing on the temporal information. The parameters of optimized encoder-decoder are used as as our pretrained parameters for video style transfer.

Second, we use collected videos by [19] from [1] as our video training dataset. In addition, the test dataset in Sintel [4] is used in our user study experiment as well. During the training stage, all video frames are resized to 256×256, and we use the same style image dataset in the first stage. Parameters of the entire model are not frozen to prevent the under-fitting of model and exploit the model representation capability. We train our multi-style video transfer network with batch size 4 for 40 epochs. The input includes 10 video frames and 1 style image. We utilize the Adam [25] optimizer with the learning rate 1e-4. The long temporal weight and short temporal weight is set as 100. The content weight, style weight and total variation weight are kept consistent with the first stage. Training a video multi-style transfer network takes about 2 days with two NVIDIA Tesla M40 GPUs. All the source code and trained models will be made available to the public.

4.2. Qualitative Results

The results of different style transfer methods are shown in Figure 3. Compared with the single-style transfer methods, our approach can generate coherent video by introducing less color chunks and display more fine-grained style information such as texture. Combination of [37] and optical flow generates output with grey effect when transferring the style of a image with bright color. Our approach generates more visually-pleasing stylization result. In the $4th$ column of Figure 3, the work of [19] cannot resolve the serious flickering issue. But from close-ups in Figure 3 our recurrent network can efficiently enforce the temporal consistency. Moreover, we can use the combination of different parameters of IN to generate a new stylized video which is not seen during training process. Our recurrent model can preserve the temporal smoothness even in the unseen stylization effect. Figure 4 shows the combination of three different stylization effects. Furthermore, our method is also

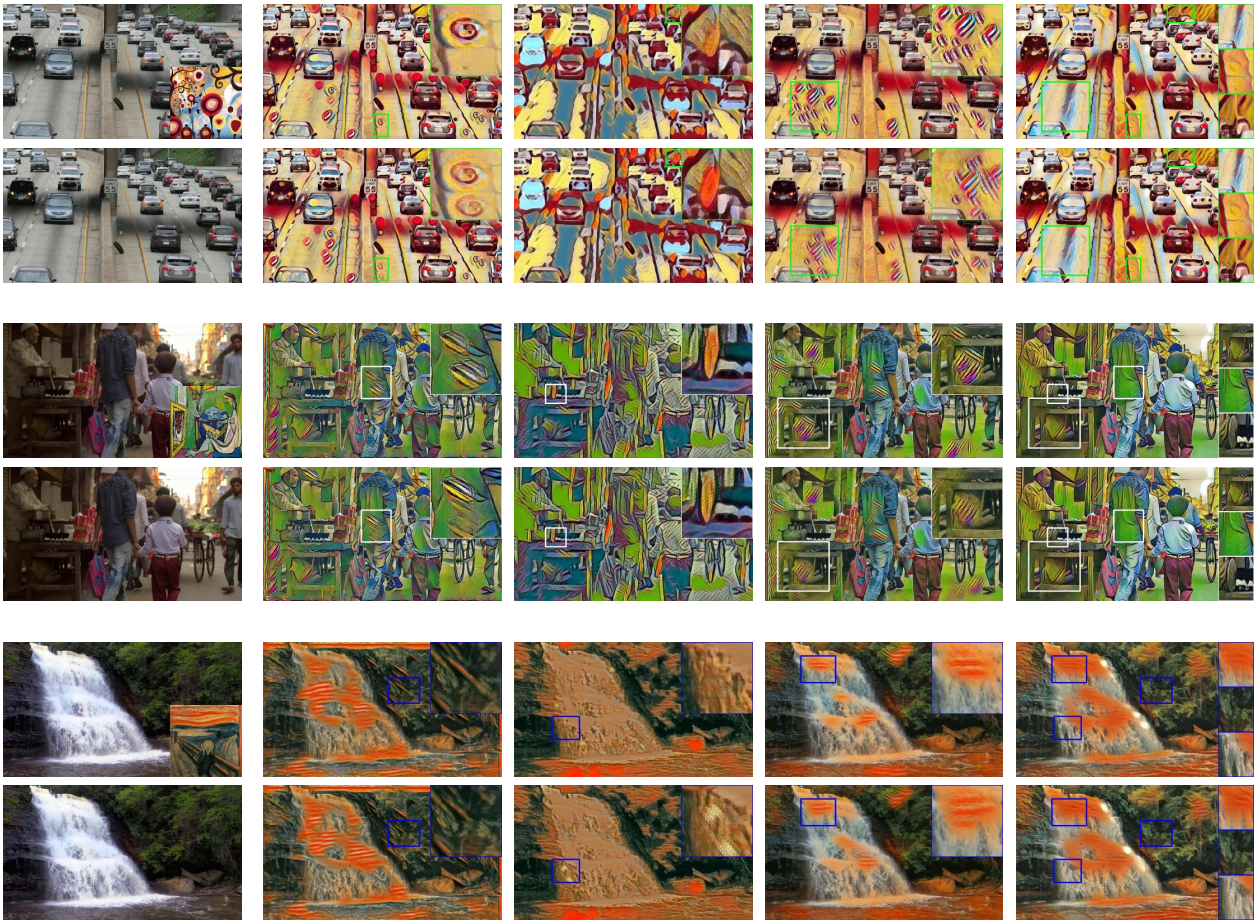

Figure 3. Qualitative comparison of different video stylization methods. Close-up regions in consecutive stylized frames are shown to demonstrate the better temporal coherency by our approach.

able to process long-term videos ($>$ 1 minute) and meanwhile maintain temporal coherence. More video results are shown in the supplementary materials.

4.3. Quantitative Results

4.3.1 User Study

While there exists no ground truth for the style transfer task, we compare our result with existing video style transfer methods to verify the effectiveness of our method by user study. We select three different works mentioned in the literature work: (i) single video style transfer (ii) multi-style transfer + temporal loss (iii) post-processing video stability. Compared with (i), we extend single style transfer to multi-style transfer in a novel way. Results of [37] shows that simple combination of the multi-style transfer and temporal loss is not practical. Moreover, post-processing video stability method strongly relies on the input stylized video. If the input video has severe flicker problems, the post-process does not work well.

Since the qualitative assessment is subjective, we conduct a user study to evaluate the aforementioned four methods. We use 5 content videos and 20 style images, and generate 100 results based on each content/style pair for each method. We randomly select results stylized by 12 style images for each subject to evaluate. We display stylized video results by four compared methods side-by-side on a webpage in random order. Each subject is asked to select the best one that is more faithful to the style and looks less flickering. We finally collect the feedback from 372 subjects of total 4464 votes and show the percentage of the votes each method received in Table 2 (second column). The study shows that our method receives the most votes for better and more stable stylized results.

4.3.2 Warping Error

We compare the warping error of different methods to validate the effectiveness of the proposed method. The warping error refers to the difference between a warped next frame

Table 2. Quantitative comparisons of different methods.

Method	Preference	Warping error(S)	Warping Error(C)
[2]	22%	0.0485	0.0486
[37] + flow	19%	0.0567	0.0564
[19]	10%	0.0302	0.0303
Ours	49%	0.0370	0.0369

Table 3. S represents FlowNetS and C represents FlowNetC.

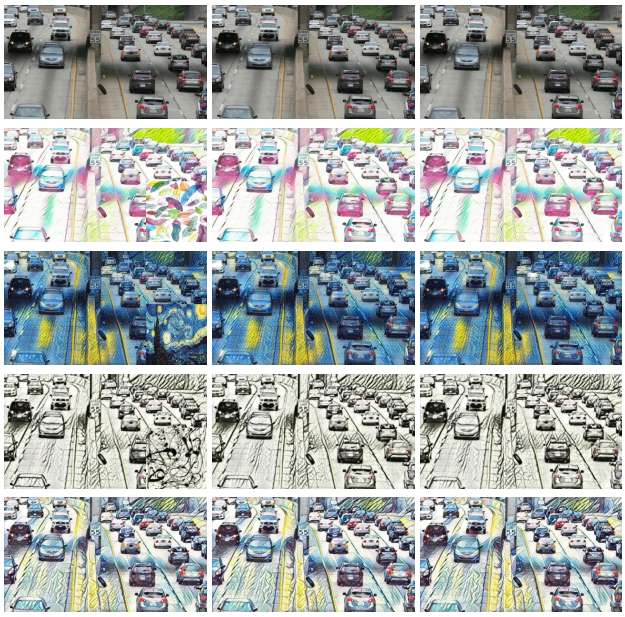

Figure 4. Combination of different style: we combine three different stylization effect into content image. Specifically, the combination weight of the three stylization effects is 0.3, 0.3 and 0.4 respectively.

and the true next frame. The value of warping error is a reasonable metric to evaluate the smoothness of video. In this experiment, we do not have the ground truth of optical flow between two consecutive frames, and we instead choose to the output of FlowNetS [9] as the ground truth of optical flow. Our proposed method use FlowNetS [9] in the training stage, hence we also utilize FlowNetC [9] to make a fair comparison. The results are shown in the third column of Table 2

From Table 2, our approach can achieve less warping error than [2] and [37]. This shows our method can achieve better video stability. In addition, we find that the results of [19] introduce more content information and erase stylization effect to maintain the temporal smoothness. Furthermore, the work of [19] cannot fix the flicker problem in local areas.

4.3.3 Run-time

While most single-style video stylization methods are also employing efficient feed-forward networks, here we mainly report the run-time of our proposed multi-style approach. Table 4 shows the run-time speed of our model under different input frame size. Our model can reach the real-time efficiency when the input size is less than 512×784 on GPU device. For the image resolution smaller than 128×256, our method can process about 6 frames per second on CPU. In the practical usage, our method can be applied in mobile devices by some existing model acceleration techniques.

Table 4. Run-time analysis under different input frame size.

Input Size	GPU(ms)	CPU(s)
128×256	7	0.16
256×384	19	0.43
512×784	84	1.54
784×960	153	2.55

4.4. Discussions

In this section, we carefully investigate some main factors in our proposed approach to provide a comprehensive understanding about its effectiveness and robustness.

4.4.1 Loss Weight

Content loss weight, style loss weight, temporal loss weight total variation loss weight collectively control the stylized effect and temporal stability of the optimized model. Generally speaking, total variation loss impacts on the spatial smoothness and we do not analyze the total variation loss weight deeply here. A temporally unstable video may have low warping error but poor stylized effect and in contrast the well-stylized video might be extremely blurred . Therefore, we need to strike a balance between temporal loss and perceptual loss to obtain satisfactory model parameters.

To probe into relationship between temporal loss and the perceptual loss(including style and content loss)we optimize models with the temporal loss weight ranging from 1 to 1000 and the style loss weight range from 10^4 to 10^6. Fro simplicity, we evaluate the warp error, content distance and style distance on the videos generated from all combinations of 5 kinds style and 5 kinds content, altogether 25 videos. The detailed results are seen in Table 5.

4.4.2 Dataset

In our experiemnts, we used two dataset to evaluate the temporal stability of our proposed method. In the dataset Videvo [1], about 25000 frames are used as the training dataset and 20 videos are used as the validation dataset. Previous work once utilize Sintel [4] Dataset to optimize their video style transfer network, however, in our experiments, we find that Sintel [4] is not a appropriate dataset to measure the video stability of a video style transfer work.

Table 5. Loss Weight Analysis. From the above table, the temporal loss weight 1000 easily leads the sudden increase of style loss. Overlarge temporal loss weight does harm to the stylized effect. In addition, Too small style loss weight(e.g.10^4) impedes on the performance of style transfer and the large gap between stylized output and style image validate this. Hence the bold rows are suitable combinations of different loss weight for our approacehs.

λ_t	λ_c	λ_s	Warping Error(S)	Content Loss	Style Loss
1	1	10^4	0.07950	5.850	0.000813
10	1	10^4	0.04087	5.9214	0.000879
100	1	10^4	0.0452	6.4453	0.000861
1000	1	10^4	0.0160	5.0387	0.000926
1	1	10^5	0.0785	8.0497	0.000686
10	1	10^5	0.0700	7.8189	0.000667
100	**1**	10^5	**0.0615**	**8.1802**	**0.000696**
1000	1	10^5	0.0483	8.8932	0.000857
1	1	10^6	0.0751	8.7171	0.000665
10	1	10^6	0.0690	8.6402	0.000673
100	**1**	10^6	**0.0591**	**8.6224**	**0.000690**
1000	1	10^6	0.0534	11.2800	0.001145

Content Videvo Sintel

Figure 5. Results of model optimized on different video dataset. The images in the second column are the results processed by model optimized on Videvo [1] dataset and that in the third column are stylized output of model trained on Sintel [4] dataset.

From Figure 5, we observe that the model trained on Sintel [4] dataset can not transfer well to Videvo [1] dataset, however the output of the model optimized on Videvo [1] dataset still keeps the temporal smoothness. Images in Sintel [4] dataset is created artificially in which it has massive local area similarity(even sameness), nevertheless the images in Videvo [1] dataset is more realistic which leads to a better generality of model train on Videvo [1] dataset.

4.4.3 Style Selection

The model trained on the different style images set behave different in the same style. We provide the results with the same style in Figure 6. The model is trained with the same settings except the number of style dataset. With the size of style set increasing, the generated images become lack of texture information especially at the background. We assume that to trade-off the style loss and temporal loss, the optimized model degrades the stylized effect of the background. But the generated output does not affect the visual experience from human eye if we do not take a close-up shot.

4.4.4 Location of ConvLSTM

We conduct an ablation study to explain the reason of location and number of ConvLSTM in our model. ConvLSTM module is inserted at the top of five residual block, middle of five residual block and bottom residual block. More specifically, **top** means that we put the ConvLSTM module before the last third MIN-Block, **middle** means that the ConvLSTM module is inserted after the second residual block, and bottom means the ConvLSTM module locates at the next layer of three MIN-Block. Our method is that two ConvLSTM modules are placed at the **top** and **bottom** location respectively.

Figure 7 shows the style transfer result of different recurrent networks. For the result of **top**, there exists the extremely bright part in local area. Because **top** has a significant effect on the result image, to minimize the overall loss, it tends to capture the simple image patch between current frame and next frame, e.g.bright white sleeve. For the results of **middle** and **bottom**, they look like dim in the whole. Recurrent module at these two locations receives the not-fully stylized feature, and to enforce the smoothness of output, the recurrent module removes stylization effect. Therefore, the output is prone to display the grey effect. Our proposed method displays visually pleasing effect compared with the other three method in Figure 7. In our method, recurrent module at the **bottom** location has less burden from temporal loss and can capture the overall spatiotemporal relation. Another recurrent module at the top location can tune the stylization feature space slightly to display more delicate result.

4.4.5 Backbone Selection

We make comparison experiments in this section to verify effectiveness of our used network architecture. Different from the classical network architecture used in [17], Zhang *et al*. [37] proposed another structure to transfer multi-style result. In this part, we will discuss why we do not adopt the network architecture in [37]. First, a simple extension of [37] is to apply the temporal loss to enforce the coherence of output. But the result of this method has been shown in the column 2 of Figure 3 and the row 2 of Table 2, which clearly shows that the generated result is not visual pleasing and temporally coherent. Second, we also

Figure 6. Results of model optimized on different style set.

Figure 7. Results of network with different locations of ConvLSTM.

Figure 8. Results of simply embedding ConvLSTM modules into the architecture proposed in the work of [37].

embed ConvLSTM modules into the network architecture of [37] but Figure 8 shows that the model still cannot transfer stylization effect well. One reason is that the shallow style extraction module in [37] cannot separate the texture, color in different stages well.

4.4.6 Limitations

Our approach can process thousands of frames and achieve up to 120 stylized effects but the number of stylized effects is limited. And adding one more stylized effect still requires computation resource and time. In addition, previous work [19] can obtain video arbitrary style transfer. However we need to point out that this approach strongly depends upon the pre-processing result which means that this method can not ensure the stability of extremely flickering videos after processing. Otherwise, the memory and speed of current arbitrary image style transfer limit its real-time capability. And from our supplementary materials, the method in [19] partly removes the stylized effect of input videos.

5. Conclusion

In this paper, we proposed a novel framework for real-time video multi-style transfer. This method is designed to train a feed-forward convolutional network to generate the smoothed stylized video frames. Furthermore, the network is capable of achieving real-time performance without relying on optical flow during evaluation. Extensive experimental results quantitatively and qualitatively demonstrate the efficiency and effectiveness of our method.

References

[1] Videvo: https://www.videvo.net/.

[2] G. Agrim, J. Justin, A. Alexandre, and F.-F. Li. Characterizing and improving stability in neural style transfer. In *ICCV*, 2017.

[3] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang. Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *PAMI*, 2018.

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[5] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua. Coherent online video style transfer. In *ICCV*, 2017.

[6] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017.

[7] T. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *PCoRR*, arXiv preprint: 1612.04337, 2016.

[8] X. Dong, B. Bonev, Y. Zhu, and A. L. Yuille. Region-based temporally consistent video post-processing. In *CVPR*, 2015.

[9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[10] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

[11] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NeurIPS*, 2015.

[12] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[13] S. Gu, C. Chen, J. Liao, and L. Yuan. Arbitrary style transfer with deep feature reshuffle. In *CVPR*, 2018.

[14] 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 *CVPR*, 2017.

[15] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[17] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[18] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017.

[19] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang. Learning blind video temporal consistency. In *ECCV*, 2018.

[20] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.

[21] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017.

[22] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *NeurIPS*, 2017.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014.

[24] M. Lu, H. Zhao, A. Yao, Y. Chen, F. Xu, and L. Zhang. A closed form solution to universal style transfer. 2019.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[26] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *GCPR*, 2016.

[27] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *GCPR*, 2016.

[28] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos and spherical images. *IJCV*, 2018.

[29] L. Sheng, Z. Lin, J. Shao, and X. Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *CVPR*, 2018.

[30] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *ICML*, 2015.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[32] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. MoCoGAN: Decomposing motion and content for video generation. In *CVPR*, 2018.

[33] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017.

[34] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.

[35] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[36] H. Xun and J. B. Serge. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[37] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. In *ECCV Workshops*, 2018.

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

[39] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017.