Real-Time Neural Style Transfer for Videos

Haozhi Huang[†‡*] Hao Wang[‡] Wenhan Luo[‡] Lin Ma[‡]
Wenhao Jiang[‡] Xiaolong Zhu[‡] Zhifeng Li[‡] Wei Liu[‡*]

[†]Tsinghua University [‡]Tencent AI Lab
[*]Correspondence: huanghz08@gmail.com wliu@ee.columbia.edu

Abstract

Recent research endeavors have shown the potential of using feed-forward convolutional neural networks to accomplish fast style transfer for images. In this work, we take one step further to explore the possibility of exploiting a feed-forward network to perform style transfer for videos and simultaneously maintain temporal consistency among stylized video frames. Our feed-forward network is trained by enforcing the outputs of consecutive frames to be both well stylized and temporally consistent. More specifically, a hybrid loss is proposed to capitalize on the content information of input frames, the style information of a given style image, and the temporal information of consecutive frames. To calculate the temporal loss during the training stage, a novel two-frame synergic training mechanism is proposed. Compared with directly applying an existing image style transfer method to videos, our proposed method employs the trained network to yield temporally consistent stylized videos which are much more visually pleasant. In contrast to the prior video style transfer method which relies on time-consuming optimization on the fly, our method runs in real time while generating competitive visual results.

1. Introduction

Recently, great progress has been achieved by applying deep convolutional neural networks (CNNs) to image transformation tasks, where a feed-forward CNN receives an input image, possibly equipped with some auxiliary information, and transforms it into a desired output image. This kind of tasks includes style transfer [12, 27], semantic segmentation [19], super-resolution [12, 7], colorization [11, 31], *etc*.

A natural way to extend image processing techniques to videos is to perform a certain image transformation frame by frame. However, this scheme inevitably brings temporal inconsistencies and thus causes severe flicker artifacts. The second row in Fig. 1 shows an example of directly applying the feed-forward network based image style transfer method of Johnson *et al*. [12] to videos. It can be observed that the zoom-in content marked by white rectangles is stylized into different appearances between two consecutive frames, therefore creating flicker artifacts. The reason is that slight variations between adjacent video frames may be amplified by the frame-based feed-forward network and thus result in obviously different stylized frames. In the literature, one solution to retain temporal coherence after video transformation is to explicitly consider temporal consistency during the frame generation or optimization process [18, 1, 14, 22]. While effective, they are case-specific methods and thus cannot be easily generalized to other problems. Among them, the method of Ruder *et al*. [22] is specifically designed for video style transfer. However, it relies on time-consuming optimization on the fly, and takes about three minutes to process a single frame even with pre-computed optical flows. Another solution to maintaining temporal consistency is to apply post-processing [15, 2]. A draw-

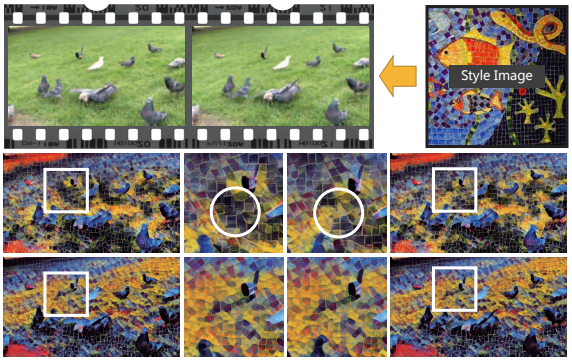


Figure 1: Video style transfer without and with temporal consistency. The first row displays two consecutive input frames and a given style image. The second row shows the stylized results generated by the method of Johnson *et al*. [12]. The zoom-in regions in the middle show that the stylized patterns are of different appearances between the consecutive frames, which creates flicker artifacts. The third row shows the stylized results of our method, where the stylized patterns maintain the same appearance.

back of post-processing is that it can only deal with image transformations whose edited results have pixel-wise correspondences to their inputs, which is not the case that style transfer obeys. Moreover, both solutions need to compute optical flows for new input video sequences, which prevents their usage for real-time video style transfer.

In view of the efficacy of feed-forward networks on image transformation tasks, a natural thinking will be whether a feed-forward network can be adapted to video transformation tasks by including temporal consistency. In this paper, we testify this idea on the problem of video style transfer. We demonstrate that a feed-forward network can not only capture content and style information in the spatial domain, but also encourage consistency in the temporal domain. We propose to use a hybrid loss in the training stage that combines the losses in the spatial and temporal domains together. Aided by the supervision of the spatial loss, our proposed video style transfer model can well preserve high-level abstract contents of the input frames and introduce new colors and patterns from a given style image. Meanwhile, the introduced temporal loss, guided by pre-computed optical flows, enables our feed-forward network to capture the temporal consistency property between consecutive video frames, therefore enforcing our model to yield temporally consistent outputs. To enable the calculation of the temporal loss during the training stage, a novel two-frame synergic training method is proposed. After training, no more optical flow computation is needed during the inference process. Our extensive experiments verify that our method generates much more temporally consistent stylized videos than the method of Johnson *et al*. [12]. An example result of our method is shown in the last row of Fig. 1, from which we can see that the stylized patterns incur no more flicker artifacts. The experiments also corroborate that our method is able to create stylized videos at a real-time frame rate, while the previous video style transfer method [22] needs about three minutes for processing a single frame. This makes us to believe that a well-posed feed-forward network technique has a great potential for avoiding large computational costs of traditional video transformation methods.

The main contributions of this paper are two-fold:

- A novel real-time style transfer method for videos is proposed, which is solely based on a feed-forward convolutional neural network and avoids computing optical flows on the fly.
- We demonstrate that a feed-forward convolutional neural network supervised by a hybrid loss can not only stylize each video frame well, but also maintain the temporal consistency. Our proposed novel two-frame synergic training method incorporates the temporal consistency into the network.

2. Related Work

Style transfer aims to transfer the style of a reference image/video to an input image/video. It is different from color transfer in the sense that it transfers not only colors but also strokes and textures of the reference. Image analogy is the first classic style transfer method for images [10], which learns a mapping between image patches. As an extension to image analogy, Lee *et al*. [16] further incorporated edge orientation to enforce gradient alignment. Recently, Gatys *et al*. [9] proposed to perform style transfer in an optimization manner by running back-propagation with a perceptual loss defined on high-level features of the pre-trained VGG-19 network [23]. Though impressive stylized results are achieved, Gatys *et al*.'s method takes quite a long time to infer the stylized image. Afterwards, Johnson *et al*. [12] proposed to train a feed-forward CNN using a similar perceptual loss defined on the VGG-16 network [23] to replace the time-consuming optimization process, which enables real-time style transfer for images. Some follow-up work was conducted to further improve the feed-forward CNN based image style transfer method. Li and Wand [17] proposed to use patches of neural feature maps to compute a style loss to transfer photo-realistic styles. Ulyanov *et al*. [28] suggested instance normalization in lieu of batch normalization, which gives more pleasant stylized results. Dumolin *et al*. [8] demonstrated that a feed-forward CNN can be trained to capture multiple different styles by introducing conditional instance normalization.

Simply treating each video frame as an independent image, the aforementioned image style transfer methods can be directly extended to videos. However, without considering temporal consistency, those methods will inevitably bring flicker artifacts to generated stylized videos. In order to suppress flicker artifacts and enforce temporal consistency, a number of approaches have been investigated and exploited for different tasks [18, 15, 1, 30, 14, 2, 22]. Specifically, Ruder *et al*. [22] used a temporal loss guided by optical flows for video style transfer. On one hand, Ruder *et al*.'s approach depends on an optimization process which is much slower than a forward pass through a feed-forward network. On the other hand, the on-the-fly computation of optical flows makes this approach even slower. In this paper, we show that temporal consistency and style transfer can be simultaneously learned by a feed-forward CNN, which avoids computing optical flows in the inference stage and thus enables real-time style transfer for videos.

3. Method

Our style transfer model consists of two parts: a stylizing network and a loss network, as shown in Fig. 2. The stylizing network takes one frame as input and produces its corresponding stylized output. The loss network, pre-trained on

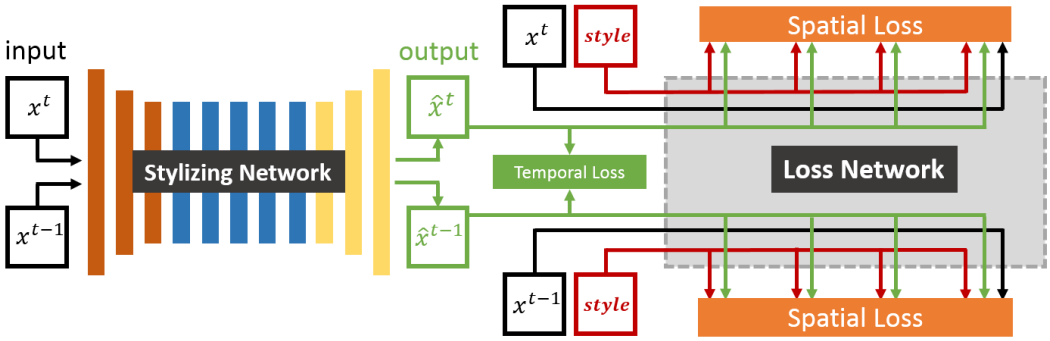


Figure 2: An overview of our proposed model. It consists of two parts: a stylizing network and a loss network. Black, green and red rectangles represent an input frame, an output frame and a given style image, respectively. A hybrid loss function including spatial and temporal components is defined on the loss network. Specifically, the spatial loss is computed separately for each of two consecutive frames and the temporal loss is computed based on both of them. This hybrid loss is used to train the stylizing network.

the ImageNet classification task [6], first extracts the features of the stylized output frames and then computes the losses, which are leveraged for training the stylizing network. On one hand, these features are used to compute a spatial loss in order to evaluate the style transfer quality in the spatial domain, which is a weighted sum of a content loss and a style loss. The content loss evaluates how close the high-level contents of the input and the stylized output are. The style loss measures how close the style features of the given style image and the stylized output are. On the other hand, a novel term, namely temporal loss, is introduced in our model to enforce temporal consistency between the stylized outputs. During the training process, two stylized output frames $\hat{\mathbf{x}}^{t-1}$ and $\hat{\mathbf{x}}^{t}$ of two consecutive input frames $\mathbf{x}^{t-1}$ and $\mathbf{x}^{t}$ are fed to the loss network to compute the temporal loss, which measures the Euclidian color distance between the corresponding pixels referring to precomputed optical flows.

The stylizing network and loss network are fully coupled during the training process. The spatio-temporal loss computed by the loss network is employed to train the stylizing network. With sufficient training, the stylizing network, although taking one single frame as input, has encoded the temporal coherence learned from a video dataset and can thus generate temporally consistent stylized video frames. Given a new input video sequence, the stylized frames are yielded by executing a feed-forward process through the stylizing network, so real-time style transfer performance is achieved.

3.1. Stylizing Network

The stylizing network accounts for transforming a single video frame to a stylized one. The architecture of the stylizing network is outlined in Table 1. After three convolutional blocks, the resolution of the feature map is reduced to a quarter of the input. Then five residual blocks are subsequently followed, leading to fast convergence. Finally, after two deconvolutional blocks and one more convolutional block, we obtain a stylized output frame which is of the same resolution as the input frame.

Table 1: The stylizing network architecture. *Conv* denotes the convolutional block (convolutional layer + instance normalization + activation); *Res* denotes the residual block; *Deconv* denotes the deconvolutional block (deconvolutional layer + instance normalization + activation).

	Layer	Size	Stride	Channel	Activation
Stylizing Network	Conv	3	1	16	ReLU
	Conv	3	2	32	ReLU
	Conv	3	2	48	ReLU
	Res × 5				
	Deconv	3	0.5	32	ReLU
	Deconv	3	0.5	16	ReLU
	Conv	3	1	3	Tanh
Res	Conv	3	1	48	ReLU
	Conv	3	1	48	

Compared to the existing feed-forward network for image style transfer [12], an important benefit of our network is that it uses a smaller number of channels to reduce the model size, which turns out to infer faster without a noticeable loss in the stylization quality. More discussions on the model size can be found in Sec. 4.5.2. Moreover, instance normalization [28] is adopted in our stylizing network in lieu of batch normalization for achieving better stylization quality. Although with a similar architecture, the most distinguishable difference of our network from [12] is that the temporal coherence among video frames has been encoded into our stylizing network. As such, the stylizing network can simultaneously perform style transfer and preserve temporal consistency. As will be demonstrated in Sec. 4, our stylizing network yields much more temporally consistent stylized video sequences.

3.2. Loss Network

For training the stylizing network, reliable and meaningful features of the original frame, the stylized frame, and the

style image need to be extracted for computing the spatial and temporal losses. In this paper, we use VGG-19 as our loss network, which has demonstrated its effectiveness on the image content and style representations [9]. Other loss network architectures for image style transfer have been investigated by Nikulin *et al*. [21], which is beyond the focus of this paper.

Different from the image style transfer approaches [9, 12, 28] which only employ the spatial loss for training, video style transfer is more complicated. Temporal consistency, as one of the most significant perceptual factors for videos, needs to be taken into consideration. Therefore, we define a hybrid loss as follows:

$$\mathcal{L}_{hybrid} = \underbrace{\sum_{i\in\{t,t-1\}} \mathcal{L}_{spatial}(\mathbf{x}^i, \hat{\mathbf{x}}^i, \mathbf{s})}_{\text{spatial loss}} + \underbrace{\lambda \mathcal{L}_{temporal}(\hat{\mathbf{x}}^t, \hat{\mathbf{x}}^{t-1})}_{\text{temporal loss}}, \tag{1}$$

where $\mathbf{x}^t$ is the input video frame at time t, $\hat{\mathbf{x}}^t$ is the corresponding output video frame, and $\mathbf{s}$ is the given style image. The spatial loss mainly resembles the definitions in [9, 12], which ensures that each input frame is transferred into the desired style. The newly introduced temporal loss is leveraged to enforce the adjacent stylized output frames to be temporally consistent. While the spatial loss is computed separately for two consecutive stylized frames $\hat{\mathbf{x}}^t$ and $\hat{\mathbf{x}}^{t-1}$, the temporal loss is computed based on both of them.

3.2.1 Spatial Loss

The spatial loss is tailored to evaluate the style transfer quality in the spatial domain for each frame. It is defined as the weighted sum of a content loss, a style loss, and a total variation regularizer:

$$\mathcal{L}_{spatial}(\mathbf{x}^t, \hat{\mathbf{x}}^t, \mathbf{s}) = \underbrace{\alpha \sum_l \mathcal{L}^l_{content}(\mathbf{x}^t, \hat{\mathbf{x}}^t)}_{\text{content loss}} + \underbrace{\beta \sum_l \mathcal{L}^l_{style}(\mathbf{s}, \hat{\mathbf{x}}^t)}_{\text{style loss}} + \underbrace{\gamma \mathcal{R}_{V^\eta}}_{\text{TV regularizer}}, \tag{2}$$

where l denotes a feature extraction layer in VGG-19. The content loss defined at layer l is the mean square error between the feature maps of an input frame $\mathbf{x}^t$ and its stylized output $\hat{\mathbf{x}}^t$:

$$\mathcal{L}^l_{content}(\mathbf{x}^t, \hat{\mathbf{x}}^t) = \frac{1}{C_l H_l W_l} \left\| \phi_l(\mathbf{x}^t) - \phi_l(\hat{\mathbf{x}}^t) \right\|_2^2, \tag{3}$$

where $\phi_l(\mathbf{x}^t)$ denotes the feature maps at level l, $C_l \times H_l \times W_l$ is the dimension of the level l feature maps. This content loss is motivated by the observation that high-level features learned by CNNs represent abstract contents, which are what we intend to preserve for the original input in the style transfer task. Therefore, by setting l as a high-level layer, the content loss ensures the abstract information of the input and stylized output to be as similar as possible. In this paper, we use the high-level layer *ReLU4_2* to calculate the content loss.

Besides high-level abstractions preserved from the original frame, for the purpose of style transfer, we also need to stylize the details according to a reference style image. Therefore, a style loss is incorporated to evaluate the style difference between the style image and the stylized frame. In order to well capture the style information, a Gram matrix $G^l \in \mathbb{R}^{C_l \times C_l}$ at layer l of the loss network is defined as:

$$G^l_{ij}(\mathbf{x}^t) = \frac{1}{H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} \phi_l(\mathbf{x}^t)_{h,w,i} \phi_l(\mathbf{x}^t)_{h,w,j}. \tag{4}$$

Here G^l_{ij} is the (i,j)-th element of Gram matrix G^l, which is equal to the normalized inner product between the vectorized feature maps of channel i and j at layer l. The Gram matrix G^l evaluates which channels tend to activate together, which is shown to be able to capture the style information [9, 12]. The style loss is thus defined as the mean square error between the Gram matrixes of the style image $\mathbf{s}$ and stylized output frame $\hat{\mathbf{x}}^t$:

$$\mathcal{L}^l_{style}(\mathbf{s}, \hat{\mathbf{x}}^t) = \frac{1}{C_l^2} \left\| G^l(\mathbf{s}) - G^l(\hat{\mathbf{x}}^t) \right\|_F^2. \tag{5}$$

In order to fully capture the style information at different scales, a set of Gram matrixes in different layers of the loss network are used to calculate the overall style loss. We choose *ReLU1_2, ReLU2_2, ReLU3_2, ReLU4_2* as the layers for computing the style loss. Other layers could also be used as style layers but will give stylized results of different flavors, which depend on personal tastes.

Additionally, to encourage spatial smoothness and suppress checkerboard artifacts in the stylized output frame, we also add a total variation regularizer:

$$\mathcal{R}_{V^\eta} = \sum_{i,j} \left(\left\| \hat{\mathbf{x}}^t_{i,j+1} - \hat{\mathbf{x}}^t_{i,j} \right\|^2 + \left\| \hat{\mathbf{x}}^t_{i+1,j} - \hat{\mathbf{x}}^t_{i,j} \right\|^2 \right)^{\frac{\eta}{2}}, \tag{6}$$

where $\hat{\mathbf{x}}^t_{i,j}$ represents the pixel of stylized frame $\hat{\mathbf{x}}^t$ at the spatial position (i,j), and η is set to be 1 empirically [20].

3.2.2 Temporal Loss

As aforementioned, by simply applying an image style transfer method to video frames, flicker artifacts will be inevitably introduced. Therefore, besides the spatial loss that leads to style transfer for each frame, we incorporate a temporal loss to enforce the temporal consistency between adjacent frames. As illustrated in Eq. (1), two consecutive frames are fed simultaneously into the network to measure the temporal consistency. The temporal loss is defined as the mean square error between the stylized output at time t

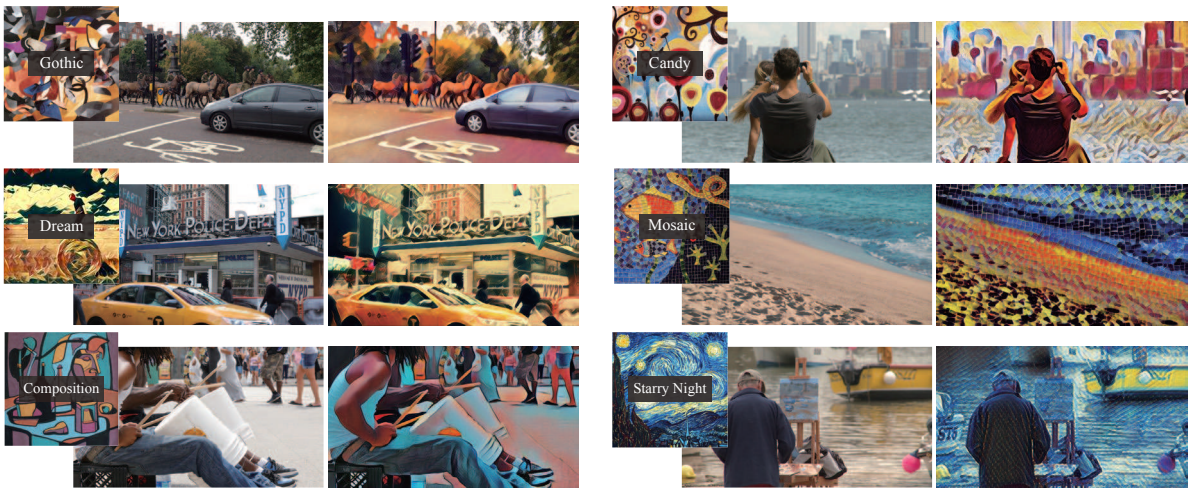


Figure 3: Video style transfer results of six styles. The high-level abstract content of each original input video is kept, while the colors and textures are transferred from each style image.

and the warped version of the stylized output at time $t-1$, namely the short-term temporal loss defined in [22]:

$$\mathcal{L}_{temporal}(\hat{\mathbf{x}}^t, \hat{\mathbf{x}}^{t-1}) = \frac{1}{D}\sum_{k=1}^{D} \mathbf{c}_k \big(\hat{\mathbf{x}}_k^t - f(\hat{\mathbf{x}}_k^{t-1})\big)^2, \quad (7)$$

where $\hat{\mathbf{x}}^t$ and $\hat{\mathbf{x}}^{t-1}$ are the stylized results of the current frame and previous one, respectively. $f(\hat{\mathbf{x}}_k^{t-1})$ is a function that warps the stylized output at time $t-1$ to time t according to a pre-computed optical flow. $D = H \times W \times C$ is the dimension of the output. $\mathbf{c} \in [0,1]^D$ denotes the per-pixel confidence of the optical flow: 0 in occluded regions and at motion boundaries, and 1 otherwise.

Unlike the calculations of the content loss and style loss, the output of the stylizing network is directly employed to compute the temporal loss other than the high-level features of the loss network. We have tried using the higher-level feature maps of the loss network to compute the temporal loss, but encounter terrible flicker artifacts. The main reason is that higher-level feature maps only capture abstract information. Hence, we insist on using the stylized output of the stylizing network to compute the temporal loss, which enforces the pixel-wise temporal consistency.

In our experiments, we use Deepflow [29] to compute the optical flows needed for the temporal loss computation. Alternative methods for optical flow calculation can also be used. Please note that the optical flows only need to be computed once for our training dataset. The temporal loss based on the optical flows can enforce our stylizing network to create temporally consistent stylized output frames, thus suppressing the flicker artifacts. In the testing (or inference) stage, the optical flow information is no longer required. As the feed-forward stylizing network has already incorporated the temporal consistency, we can apply the network on arbitrary input video sequences to generate temporally coherent stylized video sequences. Besides, we have also trained our stylizing network with non-consecutive frames to encourage long-term temporal consistency [22], which gives similar results with longer training time. For more discussions on the long-term temporal consistency, please refer to Sec. 4.5.1.

4. Experiments

4.1. Implementation Details

We download 100 videos of different scenes collected from Videvo.net [26]. 91 videos (about 40,000 frames) are used as the training dataset and 9 videos are used as the validation dataset. All video frames are resized to 640×360. Given a style image, we train a feed-forward stylizing network with a batch size of 2 for 80,000 iterations, reaching roughly two epoches over the training dataset. For each batch, the stylizing network simultaneously forwards two consecutive input frames $\mathbf{x}^{t-1}$ and $\mathbf{x}^t$, and outputs two stylized frames $\hat{\mathbf{x}}^{t-1}$ and $\hat{\mathbf{x}}^t$. Then $\hat{\mathbf{x}}^{t-1}$ and $\hat{\mathbf{x}}^t$ will be fed to the loss network to compute the hybrid loss. And a back-propagation process is performed to update the parameters of the stylizing network according to the gradients of the hybrid loss function. Note that the parameters of the loss network are fixed during the training process. The stochastic gradient descent technique we use in training is Adam [13], with a learning rate of 10^{-3}. The total variation strength γ is set to 10^{-3} to enforce spatial smoothness of the stylized frames. The default content strength α, the style strength β, and the temporal strength λ are set to 1, 10 and 10^4, respectively. All the hyperparameters are chosen based on the results of the validation set. We implement our video style transfer method using Torch [5] and cuDNN [4]. Training a

Table 2: Temporal errors of three different methods on five testing videos in the Sintel dataset.

Method	Alley_2	Ambush_5	Bandage_2	Market_6	Temple_2
Ruder *et al.* [22]	0.0252	0.0512	0.0195	0.0407	0.0361
Johnson *et al.* [12]	0.0770	0.0926	0.0517	0.0789	0.0872
Ours	0.0439	0.0675	0.0304	0.0553	0.0513

single stylizing network takes about 92 hours with a single NVIDIA Tesla K80 GPU.

In the following we present the experimental results from three perspectives. At first, the qualitative results of our method are given in Sec. 4.2. Then the quantitative comparison with existing methods in the literature is presented in Sec. 4.3. We also compare our method against some popular commercial Apps in Sec. 4.4. To explore the influence of long-term consistency and different model sizes, Sec. 4.5 discusses two variants of our method, and gives the quantitative results.

4.2. Qualitative Results

We verify our method on 30 testing videos, 10 from the Sintel dataset [3] and 20 from Videvo.net [26], using more than 20 styles. For each style, an individual stylizing network is trained. Fig. 3 shows the stylized results using 6 exemplar styles, named as *Gothic*, *Candy*, *Dream*, *Mosaic*, *Composition* and *Starry Night*. These results show that: (1) semantic contents of the original input videos are preserved; (2) colors and textures of the style images are transferred successfully. Due to the space limit, we cannot include all the stylized results here. Please refer to our supplementary material for more results.

4.3. Comparison to Methods in the Literature

We compare our method with two representative methods in the literature. One is the feed-forward CNN based method proposed by Johnson *et al.* [12], which is designed for image style transfer and runs in real time, but does not consider any temporal consistency. The other is the optimization-based method proposed by Ruder *et al.* [22], which is designed for video style transfer and maintains temporal consistency, but needs about 3 minutes for processing one frame. Since the Sintel dataset [3] provides ground truth optical flows, we use it to quantitatively compare the temporal consistencies of different methods. We defined a term *temporal error* $E_{temporal}$ over a video sequence to be the average pixel-wise Euclidean color difference between consecutive frames:

$$E_{temporal} = \sqrt{\frac{1}{(T-1)\times D}\sum_{t=1}^{T-1}\sum_{k=1}^{D}\mathbf{c}_k\left(\hat{\mathbf{x}}_k^t - f(\hat{\mathbf{x}}_k^{t+1})\right)^2}, \quad (8)$$

where T represents the total number of frames. This formulation is very similar to Eq. (7), except that we sum the temporal loss for all consecutive frame pairs in a video sequence. The function $f(\hat{\mathbf{x}}_k^{t+1})$ warps the stylized output at time $t+1$ back to t because the Sintel dataset only offers forward optical flows.

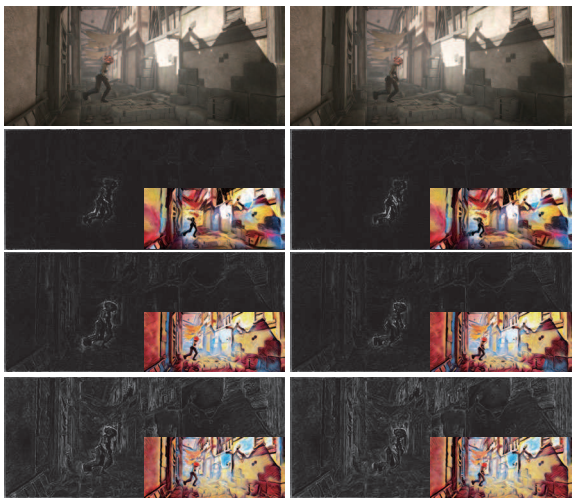


Figure 4: Comparison to two methods in the literature. The first row displays two consecutive input video frames. The following three rows show the error maps of Ruder *et al.*'s, our and Johnson *et al.*'s results. Ruder *et al.*'s method achieves the best temporal consistency, our method comes at the second place, and Johnson *et al.*'s method is the worst.

Table 2 lists the temporal errors of three different methods on five testing videos in the Sintel dataset when transferring the style *Candy*. It shows that Ruder *et al.*'s method achieves the smallest temporal error, our method follows, and Johnson *et al.*'s method turns out to be least temporally coherent. The example error maps of the three methods are displayed in Fig. 4. We can intuitively see that our method achieves smaller temporal errors than Johnson *et al.*'s method. Although our method does not outperform Ruder *et al.*'s method in terms of the temporal consistency, the latter needs about 3 minutes for generating one stylized frame even with the pre-computed optical flows. To summarize, our method generates more temporally consistent stylized frames than Johnson *et al.*'s method, while running at a real-time frame rate.

4.4. Comparison to Commercial Softwares

There are also some commercial Apps on smartphones, providing style transfer for videos. Artisto [24] and Prisma [25] are two representatives. Since we are not able to acquire the exact style images they are using, we select some of the styles that look most similar to ours, and generate results using their Apps directly. Both Artisto and Prisma support only square inputs, so we crop and resize all the input frames to 436×436. The examples of the stylized consecutive frames of different methods transferring the style *Gothic* are shown in Fig. 5. The results suggest that the temporal consistency of our results is better. The two columns

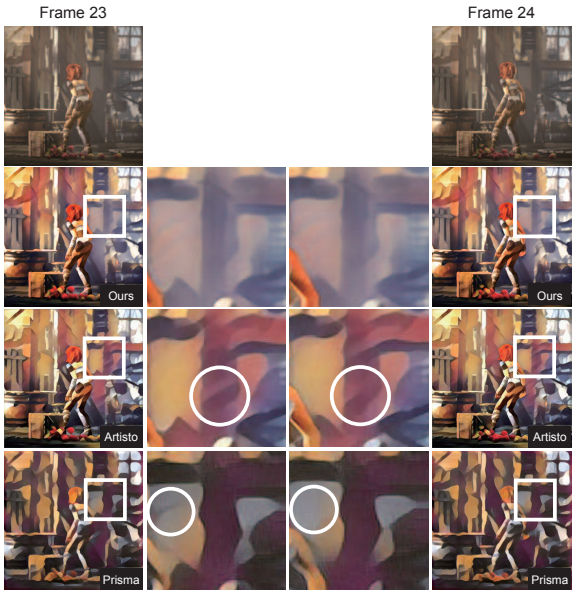


Figure 5: Comparison to commercial softwares. The first row shows two consecutive input frames. The second row shows our stylized results using style *Gothic*. The third row shows Artisto's results. The fourth row shows Prisma's results. The two columns in the middle show the zoom-in version of white rectangle regions.

in the middle show the zoom-in version of white rectangle regions. While the patterns change from frame to frame in Artisto and Prisma's results (see the regions marked by white circles), the patterns that our method creates maintain the same.

Both Artisto and Prisma create stylized videos at a different frame rate from that of an original input video, so we cannot use the ground truth optical flows of the Sintel dataset to compute *temporal error* as in Sec. 4.3. The best we can do is carrying out a user study. We invite 20 people, aged from 21 to 35, to participate in our study. 10 of our 30 testing videos are used. We select two styles (*Candy* and *Gothic*) which look like the styles provided by both Prisma and Artisto, and create $10 \times 2 = 20$ testing cases. In this user study, we compare our method with Artisto and Prisma separately. In each testing case, we simultaneously show the original input video, our result, and the result of either Artisto or Prisma on the same screen. The order of the two stylized videos is arranged randomly to avoid participants' laziness. To ensure that a user has enough time to distinguish the difference and make a careful judge, we loop all the videos for three times. In each testing case, we ask the participant which stylized video he/she prefers, or whether the two stylized videos are equally good, especially taking the flicker artifacts into account. The user study results after all testing cases are plotted in Fig. 6, which indicates that users prefer our method to both Artisto and Prisma.

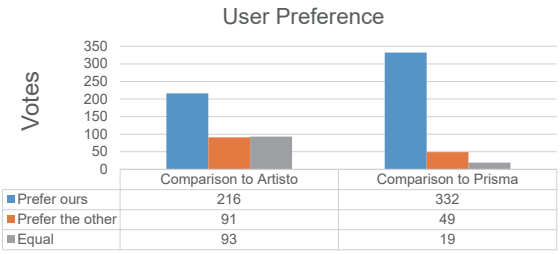


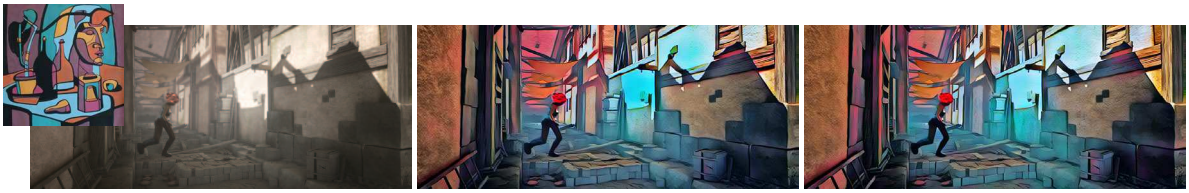
Figure 6: User study results. Comparing with Artisto, our method receives 216 votes while Artisto receives 91 votes. Comparing with Prisma, our method receives 332 votes while Prisma receives 49 votes.

4.5. Variants of Our Model

4.5.1 Long-Term Temporal Consistency

For the optimization-based method of Ruder *et al*. [22], only considering a temporal loss between consecutive frames may not guarantee long-term temporal consistency. See the first row of Fig. 8, which shows the stylized output frames of Ruder *et al*. s method according to style image *Starry Night* with a short-term temporal loss. At the top-right corner of the stylized frames, the color has changed from blue to yellow after the woman passed by. We would suspect that the optimization process of frame 30 is initialized with the previous frame 29 and ends up in a different local optimum from that of frame 1. This process makes the stylized results highly unstable. To address this issue, Ruder *et al*. [22] considered a long-term temporal loss between the current stylized frame $\hat{\mathbf{x}}^t$ and a set of previous stylized frames $\{\hat{\mathbf{x}}^{t-1}, \hat{\mathbf{x}}^{t-10}, \hat{\mathbf{x}}^{t-20}, \hat{\mathbf{x}}^{t-40}\}$, which brings a heavier computational burden and slows down the optimization process significantly. For our method, since the training process has already encouraged overall temporal consistencies over a video dataset, the stylized results are highly stable, meaning that similar contents will end up to be stylized similarly. This saves us the effort of including a long-term temporal loss in our proposed loss function. See the second row of Fig. 8, the content of the top-right corner remains the same even after the woman passed by, which reveals that our stylizing network has already included the long-term temporal consistency even trained with only a short-term temporal loss.

We also carry out an experiment to testify whether including a temporal loss of non-consecutive (long-term) frames will decrease the temporal errors defined in Sec. 4.3. During the training process, we not only consider the frame pairs of consecutive frames ($\mathbf{x}^{t-1}$ and $\mathbf{x}^t$), but also include the frame pairs of non-consecutive frames ($\mathbf{x}^{t-2}$ and $\mathbf{x}^t$). Both the consecutive pairs and non-consecutive pairs are gathered together and disrupted into a random order to formalize a training epoch. We compare our default model and



(a) Input (b) 353-Res48 (c) 353-Res128

Figure 7: Stylized results of two model sizes. Both results are stylized similarly.

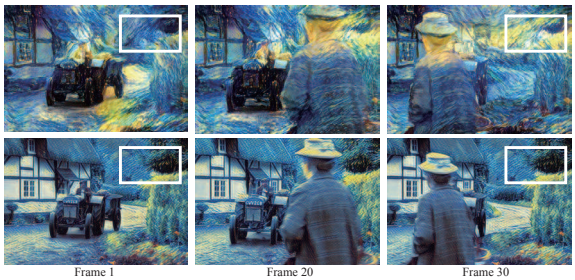


Figure 8: Long-term temporal consistency comparison. The first row gives Ruder *et al.*'s results without using a long-term temporal loss. The color of the top-right corner changes after the woman passed by. The second row gives our results. Without using a long-term temporal loss, our stylizing network can produce visually similar results as those of [22] with long-term consistency.

Table 3: Temporal errors without/with non-consecutive frame pairs.

Method	Alley_2	Ambush_5	Bandage_2	Market_6	Temple_2
$(\mathbf{x}^{t-1}, \mathbf{x}^{t})$	0.0250	0.0406	0.0182	0.0330	0.0297
$(\mathbf{x}^{t-2}, \mathbf{x}^{t}) + (\mathbf{x}^{t-1}, \mathbf{x}^{t})$	0.0241	0.0394	0.0180	0.0311	0.0283

the model trained including non-consecutive frame pairs on the videos from the Sintel dataset using style *Composition*. Table 3 shows the *temporal errors* of the two cases. There is only a slight drop of the temporal errors observed if we compare these two cases. This leads us to believe that including a long-term temporal loss during the training stage gives a very limited improvement on the temporal consistency, whereas at the cost of double or even more training time. Thus our default model for video style transfer does not include a long-term temporal loss.

4.5.2 Influence of Model Compression

The stylizing network proposed by Johnson *et al*. [12] has 5 residual blocks and each residual block produces 128 feature maps. In this paper, we find that using a smaller number of feature maps gives visually similar results while saving lots of space for storing models and shortening the inference time.

The design of our model has already been shown in Table 1, which has 5 residual blocks and each residual block produces 48 feature maps. This model is termed as *353-Res48*. We test another stylizing network that mimics the architecture of Johnson *et al*. [12], which is obtained by increasing the feature map number to 128 and called as *353-Res128*. The channel number of other layers is also adjusted accordingly, which is consistent with Johnson *et al*.'s network. Once again, we use the Sintel dataset [3] to calculate the *temporal errors* defined in Sec. 4.3. The results are collected in Table 4, where we can see that *353-Res48* presents a similar temporal error to *353-Res128*. The results in Table 4 illustrate that although the number of learnable parameters is reduced, *353-Res48* still creates visually similar results as *353-Res128*. By reducing the model size, we can accelerate the inference speed. To perform style transfer for one frame at the resolution of 1024×436, our model *353-Res48* takes about 0.041 seconds, while *353-Res128* takes about 0.098 seconds, both with a single NVIDIA Tesla K80 GPU. This enables our method to support higher frame rates or higher resolutions for real-time applications.

Table 4: Temporal errors of different model sizes.

Model	Alley_2	Ambush_5	Bandage_2	Market_6	Temple_2
353-Res128	0.0243	0.0408	0.0193	0.0330	0.0296
353-Res48	0.0244	0.0425	0.0195	0.0334	0.0302

5. Conclusions

In this paper, we proposed a novel neural method for real-time video style transfer. This method relies on training a feed-forward convolutional neural network to simultaneously preserve the high-level abstract contents of input video frames, introduce the colors and patterns from a given style image, and enforce the temporal consistency among the stylized video frames. Moreover, the trained feed-forward network, with the relief of on-the-fly optical flow computation, is capable of performing real-time video stylizing. The extensive experimental results clearly demonstrate the efficacy and superiority of our method.

Acknowledgements

When this work was conducted, the first author was an intern at Tencent AI Lab. This work is supported in part by Tsinghua University - Tencent Joint Lab.

References

[1] N. Bonneel, K. Sunkavalli, S. Paris, and H. Pfister. Example-based video color grading. *ACM Transactions on Graphics*, 32(4):39, 2013.

[2] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister. Blind video temporal consistency. *ACM Transactions on Graphics*, 34(6):196, 2015.

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

[4] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv:1410.0759*, 2014.

[5] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *Proc. NIPS 25 Workshop on BigLearn*, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on PAMI*, 38(2):295–307, 2016.

[8] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *arXiv:1610.07629*, 2016.

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

[10] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. ACM SIGGRAPH*, 2001.

[11] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*, 35(4):110, 2016.

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

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[14] N. Kong, P. V. Gehler, and M. J. Black. Intrinsic video. In *Proc. ECCV*, 2014.

[15] M. Lang, O. Wang, T. O. Aydin, A. Smolic, and M. H. Gross. Practical temporal consistency for image-based graphics applications. *ACM Transactions on Graphics*, 31(4):34, 2012.

[16] H. Lee, S. Seo, S. Ryoo, and K. Yoon. Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, 2010.

[17] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proc. CVPR*, 2016.

[18] P. Litwinowicz. Processing images and video for an impressionist effect. In *Proc. ACM SIGGRAPH*, 1997.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015.

[20] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proc. CVPR*, 2015.

[21] Y. Nikulin and R. Novak. Exploring the neural algorithm of artistic style. *arXiv:1602.07188*, 2016.

[22] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *Proceedings of German Conference on Pattern Recognition*, 2016.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[24] My.Com. Artisto. `https://artisto.my.com/`, 2016.

[25] Prisma Labs inc. Prisma. `http://prisma-ai.com/`, 2016.

[26] Videvo Team. Videvo free footage. `http://www.videvo.net`, 2016.

[27] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proc. ICML*, 2016.

[28] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

[29] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proc. ICCV*, 2013.

[30] G. Ye, E. Garces, Y. Liu, Q. Dai, and D. Gutierrez. Intrinsic video and applications. *ACM Transactions on Graphics*, 33(4):80, 2014.

[31] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *Proc. ECCV*, 2016.