# CSCI 1430 Final Project Report: STARVE: Style TrAnsfeR for VidEos

*Team name*: Yicheng Shi, Yuchen Zhou, Yue Wang, Zichuan Wang.
Brown University

## Abstract

*Style transfer is a technique that combines the style of one image and the content of the other image. In this project, instead of simply applying style transfer to images, we further apply this technique to short video clips. To stabilize the stylized video, we leverage the previous frame and optic flow as an initialization method. We also introduce short-term and long-term temporal loss. In the end, we optimize the stylized frames in a multi-pass fashion that makes the best of both forward and backward optic flows.*

## 1. Introduction

Handcrafting a sequence of stylized artistic pictures requires expert knowledge and enormous time. Leveraging the Convolutional Neural Networks (ConvNets) to stylize a video could potentially save repeated work for artistic pictures crafting by transferring the style from the target image. For a style transferring metric, it needs to consider the content from the original video, the style of the target image, and consistency with the frame nearby. To address these problems, we re-implement the Lua code[1] from Ruder et al. [8] to a TensorFlow version. Our code is available on GitHub[2].

## 2. Related Work

Our work is mainly based on Ruder et al. [8]. The original code is implemented by Lua Torch. In our project, we re-implement the model in TensorFlow. Following Ruder et al. [8], we use the ImageNet [9] pre-trained VGG19 [10] to optimize the stylized image.

Furthermore, optical flow techniques, such as DeepFlow [11] and EpicFlow [6], are applied to enhance the consistency between neighbor image frames. We also use DeepMatching [7] to generate point correspondence as a guidance for DeepFlow to calculate optic flow. We further use other optical flow techniques from OpenCV and other neural networks, such as sparse to dense flow, PCA flow, Dislow, Simpleflow, Farneback [1] and LiteFlowNet [3].

[1] https://github.com/manuelruder/artistic-videos
[2] https://github.com/zhou671/STARVE

## 3. Method

To stylize a video, we employed a pre-trained VGG19 on *ImageNet* and apply gradient descent directly on the stylized image. Afterward, the following techniques are applied.

### 3.1. Style, content and image variation Loss

We've tested both fast neural style transfer and neural style transfer. For neural style transfer, as denoted in Ruder et al.[8], $\Phi^l$ is the $l$th layer output of ConvNet model $\Phi$. For original image $\mathbf{p}$, target style image $\mathbf{a}$ and stylized image $\mathbf{x}$, their feature maps are denoted as $\mathbf{P}^l = \Phi^l(\mathbf{p})$, $\mathbf{S}^l = \Phi^l(\mathbf{a})$, and $\mathbf{F}^l = \Phi^l(\mathbf{x})$. The dimension of these feature maps are $N_l \times M_l$, where $N_l$ is the number of channels and $M_l$ is the spatial dimension of the feature maps.

Nikulin et al.[5] further define the content loss (Eq. 1) and style loss (Eq. 2) as following:

$$L_{\text{content}}(\mathbf{p}, \mathbf{x}) = \sum_{l \in L_{\text{content}}} \frac{1}{N_l M_l} \sum_{i,j} (F^l_{\text{ij}} - P^l_{\text{ij}})^2 \quad (1)$$

$$L_{\text{style}}(\mathbf{a}, \mathbf{x}) = \sum_{l \in L_{\text{style}}} \frac{1}{N_l M_l} \sum_{i,j} (G^l_{\text{ij}} - A^l_{\text{ij}})^2 \quad (2)$$

Where $A^l_{ij} = \sum_{k=1}^{M_l} S^l_{ik} S^l_{jk}$ and $G^l_{ij} = \sum_{k=1}^{M_l} F^l_{ik} F^l_{jk}$ are the gram matrix which represent the correlations of the filter responses for style image $\mathbf{a}$ and stylized image $\mathbf{x}$. The dimension of both $\mathbf{A}$ and $\mathbf{G}$ are $N_l \times N_l$.

In order to denoise the stylized image, we further apply image variation loss (Eq. 3):

$$L_{\text{imagevariation}}(\mathbf{x}) = \sum_{i,j} |x_{\text{i+1,j}} - x_{\text{i,j}}| + |x_{\text{i,j}} - x_{\text{i,j+1}}| \quad (3)$$

By weighting the all three loss components, a single image loss (Eq. 4) can be derived:

$$L_{\text{singleimage}}(\mathbf{p}, \mathbf{a}, \mathbf{x}) = \alpha L_{\text{content}}(\mathbf{p}, \mathbf{x}) + \beta L_{\text{style}}(\mathbf{a}, \mathbf{x}) + \gamma L_{\text{imagevariation}}(\mathbf{x}) \tag{4}$$

### 3.2. Optical flow

Optical flow is usually defined as the pattern of visible motion of objects, lines, and edges in a visual scene. Optical flow plays an important role in many computer vision applications that carry out the task of motion detection or action recognition. In this project, we have integrated different approaches to producing optical flow.

#### 3.2.1 DeepFlow[12]

In this paper[12], DeepFlow mixed a matching algorithm with a variational approach for optical flow. The matching algorithm was proposed by Revaud et al. [7] to solve the optical flow problem. By adopting DeepFlow, the performance on fast motions is improved. The matching algorithm is composed of a multi-stage six-layers architecture in which convolutions and max-pooling are interleaved. It has a similar architecture to deep convolution neural networks. It efficiently perceives quasi-dense correspondent points by utilizing dense sampling. DeepFlow is capable of detecting large displacements that occur in real videos.

#### 3.2.2 LiteFlowNet [3]

LiteFlowNet [3] is proposed by Hui et al. [3]. FlowNet2 [4] uses convolution neural network to do the optical flow estimation. It indeed achieves state-of-the-art performance on the task of the optical flow estimation. However, it needs more than 160 million parameters to achieve an accurate flow estimation. Thus, in this project, we choose the LiteFlowNet, which has a much smaller model size. In addition, it is 1.36 times faster than FlowNet in the running speed.

In this project, due to the time limitation of this project, we use the pre-trained model weights for this approach.

#### 3.2.3 Sparse to dense flow

Fast sparse to dense optical flow is based on PyrLK sparse matches interpolation. An advantage of using this method is its fast speed in producing the optical flow.

#### 3.2.4 Farneback [1, 2] optical flow

Dense Optical Flow by Gunnar FarneBack technique that was published in the paper [1, 2]. Farneback optical flow estimation uses the two-frame motion estimation algorithm. Firstly, it estimates each neighborhood of both frames by quadratic polynomials using the polynomial expansion transform. After that, it derived a method to estimate displacement fields from the polynomial expansion coefficients followed by some operations of refinements. We tried this method to generate the optical flow. However, it has a very poor performance than the previous methods. We will not describe more details on this.

#### 3.2.5 PCA optical flow [13]

We tried the PCA optical flow [13] as well. However, the optical flow produced by this methodology does not perform well. We will not discuss too many details on this one.

### 3.3. Temporal consistency loss

To enhance the consistency between neighbors, a temporal consistency loss is applied. This is mainly done by leveraging the optical flow as a prediction of the current frame from the previous frame. To check the consistency of optical flow between two frames, a forward flow wrapper (Eq. 5), a disocclusion inequality (Eq. 6) and a motion boundaries inequality (Eq. 7) are defined as:

$$\tilde{w}(x, y) = w((x, y) + \hat{w}(x, y)) \tag{5}$$

$$|\tilde{w} + \hat{w}|^2 > 0.01(|\tilde{w}|^2 + |\hat{w}|^2) + 0.5 \tag{6}$$

$$|\nabla\hat{u}|^2 + |\nabla\hat{v}|^2 > 0.01|\hat{w}|^2 + 0.002 \tag{7}$$

Where $w$ and $\hat{w}$ are optical flow in forward and backward direction respectively.

The temporal consistency penalizes where the wrapped image is not consistent but the optical flow is consistent.

$$L_{\text{temporal}}(\mathbf{x}, \mathbf{w}, \mathbf{c}) = \frac{1}{D}\sum_{k=1}^{D} c_k \cdot (x_k - \omega_k)^2 \tag{8}$$

Where $c \in [0, 1]^D$ is a mask for each pixel whether the pixel passes disocclusion and motion boundaries test. Note that $D = W \times H \times C$.

### 3.4. Short and long term consistency loss

A short term consistency loss (Eq. 9) only checks whether the wrapped image is consistent to the previous one frame, while a long term consistency loss (Eq. 10) could check for multiple previous frames, E.g. $J = \{1, 2, 4\}$ means check the consistency for frame $i$ with frame $i-1$, $i-2$, $i-4$.

$$L_{\text{shortterm}}(\mathbf{p}^{(i)}, \mathbf{a}, \mathbf{x}^{(i)}) = L_{\text{singleimage}}(\mathbf{p}^{(i)}, \mathbf{a}, \mathbf{x}^{(i)}) + \omega L_{\text{temporal}}(\mathbf{x}^{(i)}, w^i_{i-1}(\mathbf{x}^{(i-1)}), \mathbf{c}^{(i-1,i)}) \tag{9}$$

$$L_{\text{longterm}}(\mathbf{p}^{(i)}, \mathbf{a}, \mathbf{x}^{(i)}) = L_{\text{singleimage}}(\mathbf{p}^{(i)}, \mathbf{a}, \mathbf{x}^{(i)}) + \omega \sum_{j \in J: i-j \geq 1} L_{\text{temporal}}(\mathbf{x}^{(i)}, w_{i-j}^{i}(\mathbf{x}^{(i-j)}), \mathbf{c}_{long}^{(i-j,i)}) \tag{10}$$

where

$$\mathbf{c}_{long}^{(i-j,i)} = max(\mathbf{c}^{(i-j,i)} - \sum_{k \in J: i-k > i-j} \mathbf{c}^{(i-k,i)}, \mathbf{0}) \tag{11}$$

### 3.5. Multi-pass algorithm

In order to mitigate the inconsistency caused by strong camera motion, a multi-pass algorithm is applied. The initialization method is defined as follows:

$$\mathbf{x}'^{(i)(j)} = \begin{cases} \mathbf{x}'^{(i)(j-1)} & \text{if } i = 1, \\ \delta\mathbf{c}^{(i-1,i)} \cdot w_{i-1}^{i}(\mathbf{x}^{(i-1)(j)}) & \\ +(\bar{\delta}\mathbf{1} + \delta\bar{\mathbf{c}}^{(i-1,i)}) \cdot \mathbf{x}^{(i)(j-1)} & \text{else} \end{cases} \tag{12}$$

$$\mathbf{x}'^{(i)(j)} = \begin{cases} \mathbf{x}'^{(i)(j-1)} & \text{if } i = N_{frame}, \\ \delta\mathbf{c}^{(i+1,i)} \cdot w_{i+1}^{i}(\mathbf{x}^{(i+1)(j)}) & \\ +(\bar{\delta}\mathbf{1} + \delta\bar{\mathbf{c}}^{(i+1,i)}) \cdot \mathbf{x}^{(i)(j-1)} & \text{else} \end{cases} \tag{13}$$

Where $\bar{\mathbf{c}} = \mathbf{1}$ - $\mathbf{c}$ and $\bar{\delta} = 1 - \delta$. The multiple-pass algorithm initialize the stylized image by stylize each image independently. Then initializing the images as stated in Eq. 12 and Eq. 13 depends on the passing direction. The passing direction changes after several iterations of gradient descenting.

## 4. Results

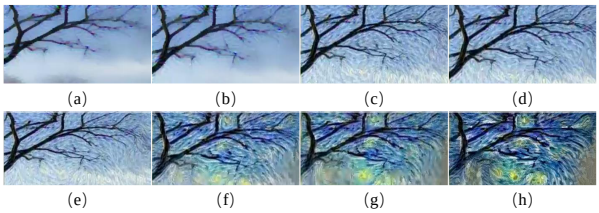


Figure 1. Comparison between different hyper-parameters.

The result of our style transferred video is illustrated in Fig. 1, and the corresponding hyper-parameters are listed in Tab. 1. The stylized videos are also displayed on YouTube [3].

[3]https://youtu.be/i5yk5Y3pp4g

Table 1. Hyper-parameters of Fig. 1.

| Name | $\alpha$ | $\beta$ | $\omega$ | #passes |
|---|---|---|---|---|
| (a) | 7e-5 | 5e-6 | 2e-1 | 1 |
| (b) | 7e-4 | 1e-5 | 2e-1 | 1 |
| (c) | 1 | 20 | 200 | 1 |
| (d) | 1 | 30 | 300 | 1 |
| (e) | 1 | 20 | 200 | 1 |
| (f) | 1 | 20 | 200 | 15 |
| (g) | 1 | 20 | 200 | 25 |
| (h) | 1 | 20 | 200 | 50 |

### 4.1. Technical Discussion

From Fig. 1 (a), (b) we can see that when the weights are very small, it's hard to optimize images in a relatively small number of iterations (50 in our experiment). The comparison between Fig. 1 (a) and (c), (b) and (d) shows that as we increase the ratio between $\beta$ and $\alpha$, the style in the output image will be more noticeable. Fig. 1 (e), (f), (g) and (h) demonstrate that the more passes we operate, the better the visual effect is. To sum up, to generate a good stylized image, we should increase the weight of style loss and increase the number of iterations.

### 4.2. Societal Discussion

#### 4.2.1 Socio-historical context

1. Since our work might have the potential to replace human work, we need to consider about what would happen if people in this industry would lose job. Our initial goal is to reduce the amount of work to produce image sequences with the same style. But it might lead to an increasing unemployment rate within the industries such as animation or art creation. Given that the job growth of animation creation is 4% [4] , we need to investigate what is the job growth after this work could impact the industries.

2. Anime is prevalent in all major countries with open internet access. Episodes and movies of it have fans from all over the world. People used to make hand drawings or learn Japanese to show their passion for anime, if not learning Ninjutsu Mudras from Naruto. And now, with our project, they would have a chance to turn their life into an anime scene.

3. In this era, a large portion of social interactions happens on online social platforms. The research of Li et al. [14] raises concerns about personal privacy due to the data shared online. However, if users can apply style filters to their real faces before posting private images online,

[4]https://study.com/articles/Cartoon_Animators_Job_Outlook_and_Requirements_for_a_Career_in_Cartoon_Animation.html

this style transfer technique can largely prevent privacy leakage. This is because some private information can be wiped out with the aid of new styles added to the image. For example, the size of eyes, the color of skin, and other personal details can be protected in this way. Also, since style images can be customized, it's very likely that users are willing to share these stylized images or video clips in this way. In summary, style transfer will not only protect users' privacy but also maintain their willingness to share their lives on social media.

#### 4.2.2 Stakeholders

Major stakeholders include video platforms and people who use their platforms, since we provide a new way to process the video. Sometimes, YouTubers or other KOLs need to pixelate the video for the purpose of protecting the user privacy. Our technique provides another way of pixelating the video while still maintaining some original information. By applying the style to the video, we could blur the video to complete this task. Meanwhile, companies such as ByteDance can use such an algorithm as a filter on video products like TikTok. However, we would imagine that some people using these techniques for malicious purposes such as modifying the copyrighted video without the permission of the original video owner. Additionally, content users of video platforms such as YouTube might use it to stylize harmful videos such as child pornography, and current censor algorithms might not be able to recognize the content in the stylized videos.

Moreover, animation producers and artists can benefit from this technique. Style transferring a real-person scene into an animated video. This could help to save money to re-draw complicated frames and come up with more realistic anime characters' behavior. Artists no longer need to spend tons of time redrawing frames with the same theme. Instead, they could use this technology to transfer a single frame into a video, thus helping them force on the theme of their product.

#### 4.2.3 Implications of existing research

Some style transfer techniques have been applied to short video production. Social apps such as TikTok have already provided the service of doing the style transfer for the streaming video. In terms of this, it may affect how we should frame our goal. When we designed our model, we are not much concerned about the running time of the model. But we have seen the live video style transfer. The underlying implication is that the running time also counts.

#### 4.2.4 Impacts on individuals and communities

As mentioned above, copyright issues might occur with the implementation of our project, affecting both style image artists and content video authors. Some artists may use the stylized image or video as his/her original work to cheat. And it might create new challenges for video platforms to censor the content of a video. Besides, people may do severe pranks or even bullies by using the style of thriller movies. For example, in high schools, some students may generate new images or videos of their teachers or classmates with terrifying styles. In this way, this technology may harm the mental health of those students being affected.

#### 4.2.5 Biases

We are not using data and no bias has been found in our experiment yet. We've adapted VGG model trained with ImageNet to extract features from style image and content image, but we don't believe biases from ImageNet are adapted in our model since we don't label the items in our videos.

## 5. Conclusion

We've implemented various techniques to accomplish artistic style transfer on videos and to stabilize the process and result of it, including warped image initialization, define loss functions for short-term consistency, long-term consistency, and multi-pass. These techniques are aimed at stabilizing frames in a video. With these methods, there will be less flickering in the stylized video. We hope this technique can be used in the animation industry and entertaining mobile applications. In this way, we may bridge the gap of dimensions and taste what it is like to live in a 2D world.

## References


[1] Gunnar Farneback. Fast and accurate motion estimation using orientation tensors and parametric motion models. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 1, pages 135–139. IEEE, 2000. 1, 2

[2] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003. 2

[3] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2

[4] 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. 2

[5] Yaroslav Nikulin and Roman Novak. Exploring the neural algorithm of artistic style. *arXiv preprint arXiv:1602.07188*, 2016. 1

[6] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow, 2015. 1

[7] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 120(3):300–323, 2016. 1, 2

[8] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German conference on pattern recognition*, pages 26–36. Springer, 2016. 1

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 1

[11] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *2013 IEEE International Conference on Computer Vision*, pages 1385–1392, 2013. 1

[12] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1385–1392, 2013. 2

[13] Jonas Wulff and Michael J Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 120–130, 2015. 2

[14] Li Yan, Li Yingjiu, Yan Qiang, and Deng Robert H. Privacy leakage analysis in online social networks. *Computers Security*, 49:239–254, 2015. 3

# Appendix

## Team contributions

**Yicheng Shi** Reviewed different types of optical flows and analyzed the mechanism and the effect of the optical flow on the model. Integrated FlowNet, LiteFlowNet into the project, experiment with different optical flows. Prepared and did the presentation.

**Yuchen Zhou** Implemented temporal, short term consistency loss, long term consistency loss and multi-pass algorithm. Ran the multi-pass algorithm experiment. Wrote the methodology, introduction and related work parts for the final report.

**Yue Wang** Implemented the baseline model of style transfer frame by frame. Prepared and designed the presentation. Experimented with Fast style transfer model for the purpose of comparison.

**Zichuan Wang** Team leader and the project architect. Built the basic code architecture. Read the Lua source code, modified, and integrated different branches and pull requests. Compiled Caffe and DeepMatching GPU version from scratch to realize a 6x speed-up for optic flow calculation. Implemented utility and visualization functions. Wrote tutorial jupyter notebooks for beginners.