CSCI 1430 Final Project Report: Artistic Style Transfer for Videos and Images

Team name: Sequence Stylizers
TA name: Ria Rajesh.
Brown University

Abstract

This project is a culmination of modern deep learning and computer vision technology. By leveraging VGG19, a state of the art Image Net deep learning network, we were able to extract and repurpose previously trained dense layer weights to aid in the restyle of both images and videos in the context of an art piece. Additionally, we went a step further and provided a tidy interface for users to easily use and interact with our program.

1. Introduction

We sought to solve the problem of how to apply an artistic style to an entire video. While the single frame implementation is fairly straightforward due to a large amount of resources online concerning the topic, generalizing the technique to a video is quite difficult. Using a neural network for style transfer is not deterministic and therefore slightly different frames can have vastly different appearances. By leveraging a temporal loss function, we managed to reduce frame to frame flickering and produce a much smoother stylized video. Our general approach was to first implement single frame stylization which uses gradient descent to optimize an image based on a loss function that combines metrics for how close the optimization is to the original content image and the style image. We then expanded this loss function by adding a temporal term that made loss function also consider how similar the current output was to the previous frame. Video stylization is an area with a lot of current research, with new papers coming out on the topic every year. Each paper uses slightly different training techniques to try and make the temporal cohesion of the output a little better. If this problem were solved already, we would have an ideal temporal loss function which does not yet exist. We implemented this paper to understand and improve on a single type of temporal consistency, one based on temporal flow, but there are many out there with all different pros and cons.

2. Related Work

The application side of this project was bootstrapped with Tailwind CSS and NextJS. These tools provided the basic structure of our application. We then use Python to run the style transfer, using the VGG19 Convolutional Neural Network and Tensorflow to optimize the stylized image. For optical flow and image processing we used OpenCV and Pillow. For numerical operations, we used the Numpy library. Lastly, to stitch all the individually stylized photos back into a video we used the FFMPEG tool.

The field of style transfer is very vibrant and many papers have been published on the topic. The paper we partially implemented (*Ruder et al.* [4]) extends the work of single frame optimization pioneered by this *Glatys et al.* [1] paper. These two papers Texler et al [2] and Jamriska et al. [3] get incredible results of extending style of one keyframe to the rest of a video. They use many of the same temporal techniques we implemented to keep their video consistent.

3. Method

3.1. Single frame stylization:

To implement single frame stylization, we wrote up a loss function that prioritized maintaining the content of the original image provided, while also emulating the style of the style sample image given. For both of these functions, we used intermediate layers of the VGG19 network to extract the features of our content image and our style image and then compare them to each other. The total loss for a single frame can be written as

$$\mathcal{L}_{singleimage}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha \mathcal{L}_{content}(\boldsymbol{p}, \boldsymbol{x}) \; + \; \beta \mathcal{L}_{style}(\boldsymbol{a}, \boldsymbol{x})$$

Where α and β are experimentally derived constants, p is the original image, a is the style image, and x is the current stylized image.

3.1.1 Content loss:

To calculate content loss (how similar the current state of the optimization is to the input image), we simply took the mean squared error between the extracted feature maps of the original image and the stylized image, as the paper directed. Thus, a greater difference in features extracted between the original and stylized image produces a greater overall content loss.

$$\mathcal{L}_{content}(\boldsymbol{p}, \boldsymbol{x}) = \sum_{l \in L_{content}} \frac{1}{N_l M_l} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2.$$

Here, F^l represents the extracted feature maps from the stylized image from layer l in the VGG19 network. P^l represents the same feature maps extracted from the original content image. N^l and M^l are simply the dimensions of the feature maps.

3.1.2 Style loss:

To calculate style loss, we perform a similar mean squared error operation between the "Gram matrices" of our stylized image and our style sample image. The gram matrix of an image's feature maps is essentially a giant matrix of different combinations of inner products between the image's features, and is frequently used to extract/represent the style of an image.

$$\mathcal{L}_{style}(\boldsymbol{a}, \boldsymbol{x}) = \sum_{l \in L_{style}} \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

3.2. Video stylization:

To expand single frame stylization to videos, we used two main methods proposed by the paper: the first was to initialize each new frame with a warped version of the last frame, transformed according to the perceived optical flow between the last two frames. The second method was to add an additional loss factor to our function, temporal loss.

A naive approach to writing a temporal loss function could be simply calculating the squared difference between the stylized frame and the previous frame, warped forward using optical flow. However, this naive approach would unjustly penalize any regions that are at the edges of a moving component, or ones that become disoccluded through motion, which witness a great deal of deviation from their predicted warped image, yet are necessary for describing motion. Thus, we weight the squared difference between a region on the current frame and its optical-flow-predicted counterpart with a 0 if the region is thought to contain a disocclusion or motion boundary, and a 1 if otherwise. The inequalities found for finding these disocclusions come from the *Sundaram et al.* [5] paper on motion tracking.

Disocclusions are calculated using the following inequality:

$$|\widetilde{\mathbf{w}} + \hat{\mathbf{w}}|^2 > 0.01(|\widetilde{\mathbf{w}}|^2 + |\hat{\mathbf{w}}|^2) + 0.5$$

Where $\tilde{w}$ represents the forward optical flow between the current and previous frames warped to the second frame and $\hat{w}$ is the backwards flow from the current frame to the previous frame.

Motion boundaries are calculated using the following inequality:

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

Where ∇u is the gradient of the forward flow's x component and ∇v is the gradient of the forward flow's y component.

We then multiply these weights (1 or 0) by the mean squared error between each pixel and its warped counterpart and take the mean across all pixels in the image.

$$\mathcal{L}_{temporal}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{c}) = \frac{1}{D} \sum_{k=1}^{D} c_k \cdot (x_k - \omega_k)^2 \ .$$

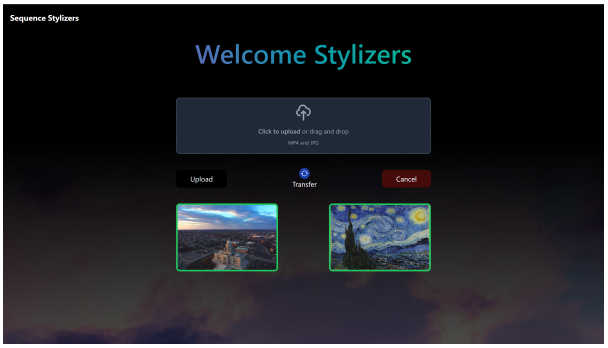
This brings the total loss of a stylized frame to be

$$\mathcal{L}_{shortterm}(\boldsymbol{p}^{(i)}, \boldsymbol{a}, \boldsymbol{x}^{(i)}) = \alpha \mathcal{L}_{content}(\boldsymbol{p}^{(i)}, \boldsymbol{x}^{(i)}) + \beta \mathcal{L}_{style}(\boldsymbol{a}, \boldsymbol{x}^{(i)}) + \gamma \mathcal{L}_{temporal}(\boldsymbol{x}^{(i)}, \omega_{i-1}^{i}(\boldsymbol{x}^{(i-1)}), \boldsymbol{c}^{(i-1,i)})$$

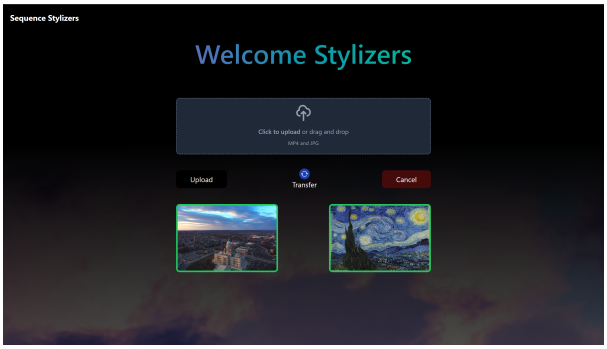
Where γ is another experimentally derived constant.

3.3. GUI Visualizer:

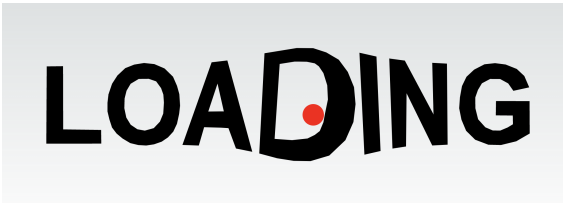
Finally, we created a tidy GUI visualizer for our video stylizer that launches in a user's browser, lets them interactively upload media into the program, and displays the results of the stylization process.



The app is bootstrapped with Tailwind CSS and NextJS, which encompasses a NodeJS backend. The initial file upload makes an API call to the backend which uploads the images to the server and, subsequently, turns the media wrappers green. This is also integrated with intelligent file validation to ensure the media uploaded is compatible with the process. Once a base image/video and a style image are uploaded successfully, the user is able to make a stylize request via the transfer button which will generate a styled version of the base media and provide the user an easy way to download it.



Since a one second clip can still take up to 30 minutes to render, we also created four different unique loading screens for the user to interact with while our program synthesizes new frames.



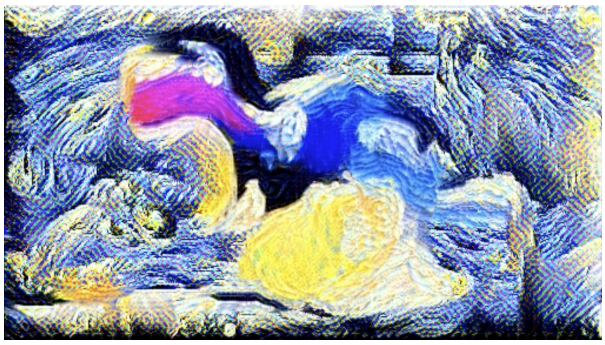
The first loading screen uses a hyperbolic secant function to take points close to the cursor and displace them away from it using simple vector math. The second loading screen is a 3D projection of multiple layers of letters that oscillate and change color, which the user can rotate in 3D by using their mouse. The third loading screen mimics the behavior of a clock, where each vertex of the letters rotates around 12 times before returning to its initial position- it uses a combination of a step function and a damped sine wave to make the motion advance in stuttered increments that vibrate ever so slightly. The final loading screen tracks the mouse position and mimics the behavior of shadows created by a moving light source. The effect is quite simple, we simply scale each line segment by a factor of 1000 away from the cursor, then form a black shaded polygon with the resulting points, and repeat this for every single line segment that makes up the points of the letters.

4. Results

Although there is always room for improvement, we were generally quite impressed with our results. Our best results can be found in the *results* folder submitted with the code for this project. After a great deal of hyperparameter tuning, we were able to make the algorithm produce nicely styled videos with significantly less noise compared to previous videos created without the temporal loss factor or the warping of the initial optimization. Check out *lions_base.mp4* and *lions_temporal.mp4* to see the difference between the two. There is a fair amount of noise present in the background of *lions_base.mp4*, yet in *lions_temporal.mp4* the background remains much more stable between frames. There is a bit of noise but we mostly think that is because we did not let the frames optimize for long enough during training. Despite the general success of these results, we still noticed some flaws with our approach. Even with the the three degrees of freedom given by our different loss components (α, β, and γ) we found that the algorithm focuses a bit too heavily on the temporal factor, which can lead to a large resistance to motion. This often produces smudging and strange artifacts from previous frames. An example of this can be found in a frame from *dino.mp4*:

As you can see, under the feet, there is a lot of noise and smudging because that is an area with a lot of movement. In the full video you can also see moments where it appears there is a duplicate leg next to the currently moving leg.

However, the smudging can be reduced by tweaking the parameters. Here is an image from an early render of the dinosaur video with bad parameters:



As you can see, almost all detail the dinosaur once had, especially around its feet has vanished. This was due to the temporal and style components of the loss function being weighted far too heavily. It took many hours of trial and error to find parameters that were even remotely good, as it takes several minutes to render enough frames to get a sense for what the movement will look like.

The nicest result videos we got from our testing were *dino.mp4*, *lions_temporal.mp4* and *dunk.mp4*.

4.1. Technical Discussion

Unlike the paper which used NN-based optical flow methods, we used the Gunnar-Farneback Optical Flow algorithm to calculate the flow between frames. We wonder if perhaps this choice, combined with our use of low-res images for efficiency's sake, led to so much smudging at points of a lot of motion. We found that the optimal hyperparameter settings for our model tended towards the following values:

$$\alpha = 1 * 10^0$$

$$\beta = 1 * 10^4$$

$$\gamma = 4 * 10^2$$

These parameters vary wildly from the proposed parameters in the paper which leads us to believe that we implemented our loss function differently from the paper. However, we still remain satisfied with our results, and believe that determining success in terms of whether or not an image is stylized correctly still remains fairly subjective. This raises some very interesting questions about how to best infer how an image is changing from frame to frame. This project takes an interesting approach and uses differences between the forwards and backwards optical flows to find regions of stagnation and regions of disocclusion. We wonder if this is the best way to quantify how a video is changing. Flow seems to be a bit of a finicky thing to approximate, and so we wonder if having something like depth data as well as just the images would help when trying to find areas with consistent motion as opposed to disocclusion.

4.2. Socially-responsible Computing Discussion via Proposal Swap

The first point in the critique references our failure to establish a potential set of policies that our users would need to follow in order to prevent copyright violation. We agree with this criticism and acknowledge that, as described in the critique, it may be difficult to actually scan for IP we can, and should, establish guidelines that are clearly presented to users when they visit our application. Although it may be very difficult to actually prevent users from uploading IP, most people are not malicious by nature (at least we hope so) and very visible guidelines that establish rules with free use of our site may go a long way in reducing the amount of IP infringement.

The second point in the critique refers to the potential of our application to blur the line between human-created and AI-generated content. Our group acknowledges that this is a very valid point for advanced technology, and one that will definitely need to be considered down the road. However, we do not believe this is an issue we need to currently address. We strongly believe this because our application does not produce style transfer at a level nearly sophisticated enough to be misidentified as human made. Instead, we view our product as a fun site which users can use to alter their favorite videos and images in new ways. The video results have some buggy components that we feel obviously distinguish it from something that could have been compiled from frames of individual art pieces. Whilst transfer of style to images is a bit more foggy, we believe that it is still pretty clear it was not made by a person. However, we would like to briefly suggest an idea to be implemented in the future where we augment the file upload system to include the metadata stack. This would provide us the opportunity to augment and return the updated metadata, and in this way ensure that the new image has a written history of AI editing.

The third point in the critique refers to the possibility

of our application's contributing towards the spread of misinformation. We acknowledge that this may certainly be a possibility with respect to things like lying about the origin of an image, and we believe that this ties back into the previous two responses. To solve this problem, we looked back to our previous two responses where we described a need to physically manipulate the output - to ensure its metadata encapsulates a history of AI editing - and also establish clear social guidelines on the site to dissuade users from pursuing such activity. With that said, however, we disagreed with the third point with respect to its description of spreading false news through the disguise of trusted News sources. This application is not a powerful deep fake editor, and is instead merely a way to transfer the style of one media source onto the content of another. The actual content of the base image will remain constant, and the model has no ability to train an image to something completely different - such as a CNN morning show. As a result, we believe this technology could not be leveraged for such malicious activity do not believe we need to consider this as a serious possibility.

5. Conclusion

In this project we created an app that lets anyone upload and stylize any video. By doing this we hope to make neural stylization more approachable and fun to use. We hope to encourage artists and creators to use this tool and make fun, new art. Moving forwards, we still have a lot to improve, such as implementing a long-term temporal loss function that considers more than two frames, and using different temporal metrics than just optical flow. However, we are very happy with the results our script gives currently and we hope people can have fun playing around with it! We learned a lot from this project. Specifically, we are all much more comfortable with the complex logic surrounding the complicated topic of style transfer. Additionally, we learned a lot about app development from the complicated process of spawning python scripts within a conda environment from a node backend and validating files during server upload. This was an excellent opportunity to utilize some of the knowledge we gained from the semester, and we are so thankful to the CS1430 course staff for providing us these opportunities!

References

[1] Matthias Bethge Leon A. Gatys, Alexander S. Ecker. A neural algorithm of artistic style. 2015. https://arxiv.org/pdf/1508.06576.pdf. 1

[2] Ondrej Texler Michal Lukac Jakub Fiser Jingwan Lu Eli Schectman Ondrej Jamriska, Sarka Sochorova and Daniel Sykora. Stylizing video by example. 2019. https://dcgi.fel.cvut.cz/home/sykorad/Jamriska19-SIG.pdf. 1

[3] Michal Kucera Ondrej Jamriska Ondrej Texler, David Futschik and Sarka Sochorova. Interactive video stylization using few-shot patch-based training. 2020. https://arxiv.org/pdf/2004.14489.pdf. 1

[4] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. 2016. https://arxiv.org/pdf/1604.08610.pdf. 1

[5] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. 2010. https://lmb.informatik.uni-freiburg.de/Publications/2010/Bro10e/. 2

Appendix

Team contributions

Please describe in one paragraph per team member what each of you contributed to the project.

Person 1 All three group members have been in the same room together, working on things and understanding each other's parts. It was a very nice and collaborative environment throughout the entirety of the project. I have helped lead the charge on the Neural Style Transfer aspects, reading papers and writing some pything script. I also have the fastest computer and so I led the charge on tuning a lot of the parameters with respect to the loss functions and learning rates which helped gradually get us better results once we got an MVP.

Person 2 Aside from aiding person 1 in implementing VGG network architecture and a basic implementation of the style transfer code during the first phase of single image style transfer, I was mainly focused on the application side of this project. I bootstrapped a ReactJS app with nextJS, and successfully implemented a NodeJS server with multiple REST endpoints. The first one successfully validates and handles file upload, while the second executes our style transfer script within our conda environment. On the frontend, I created a welcome page with a form and validation for file uploading and a download page once the style is complete. I was able to nicely integrate this frontend with the loading pages designed by Person 3. Overall, this application is extremely solid, and if I had the funds to run the server I would absolutely deploy it.

Person 3 The project takes quite a long time to run. To address this and keep our app fun to use, I created a comprehensive array of creative and engaging loading screen animations for the website. I started by drafting the screens in desmos before transferring everything into typescript functions by using d3.js. I then helped work on the temporal loss function, sitting with person 1 and helping write the function.