CS 4701 Final Report

Video Neural Style Transfer

Team Name: NStylist, **Members:** Roberto Halpin (rgh224), Shirley Thomas (smt244), Meiqi Wu (mw849)

Keywords: video, style transfer, convolutional neural network, deep learning, transfer learning

Neural style transfer (NST) is the technique of combining the artistic style of one image to another using deep learning networks. It is one of the most fun techniques in deep learning. It merges two images, namely the “content” image (Fig 1. A) and “style” image (Fig 1. C), to create a “generated” image (Fig 1. B).

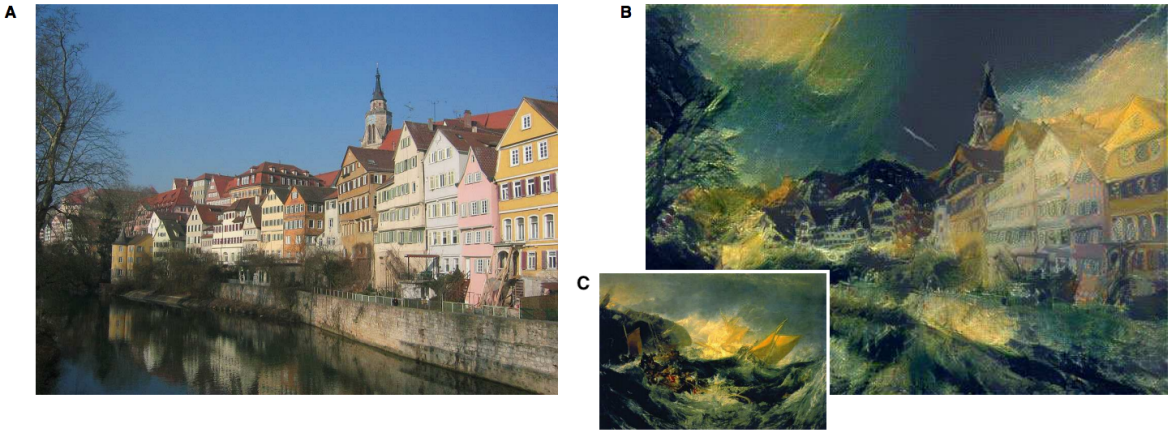


Fig 1. Example of art generation with neural style transfer.
(A) “Content” image: photograph depicting the Neckarfront in Tubingen, Germany (Photo: Andreas Praefcke).
(B) “Generated” image. (C) “Style” image: The Shipwreck of the Minotaur by J.M.W. Turner, 1805.

Goal

In this project, we planned to explore the neural style transfer on videos. Here, we would be taking the style from an image and applying it to a content video to generate a new stylized video. The simple strategy would be to implement the style framewise, but this would bring discontinuities among the frames, leading to an unappealing flickering effect.

To remedy this, when training the model, the plan was to add a temporal loss term to maintain the flow of the video, reducing the amount of discontinuity between consecutive frames. Along with that, it was decided to build a browser based application which allows users to upload a content video to view a generated stylized video.

Initially, we planned to allow the users to upload a style. However, when we switched to the implementation of another idea (will be discussed later), which requires the user to use only the models trained beforehand, it became clear that giving an option to upload would not be viable.

Current Status

After implementing the framewise model, we discovered that the architecture we based our model on was very slow, not functional for any demonstration. Thus, we did a literature search and found an interesting paper that addressed this problem on video style transfer. The paper claims real-time video style transfer, so we shifted our focus to implementing this paper from scratch.

This paper required us to train models conditioned on style images offline. Thus, when performing inference, we would load trained models conditioned on certain style images. The users must choose one of the predefined style images, but are still free to upload their own content videos.

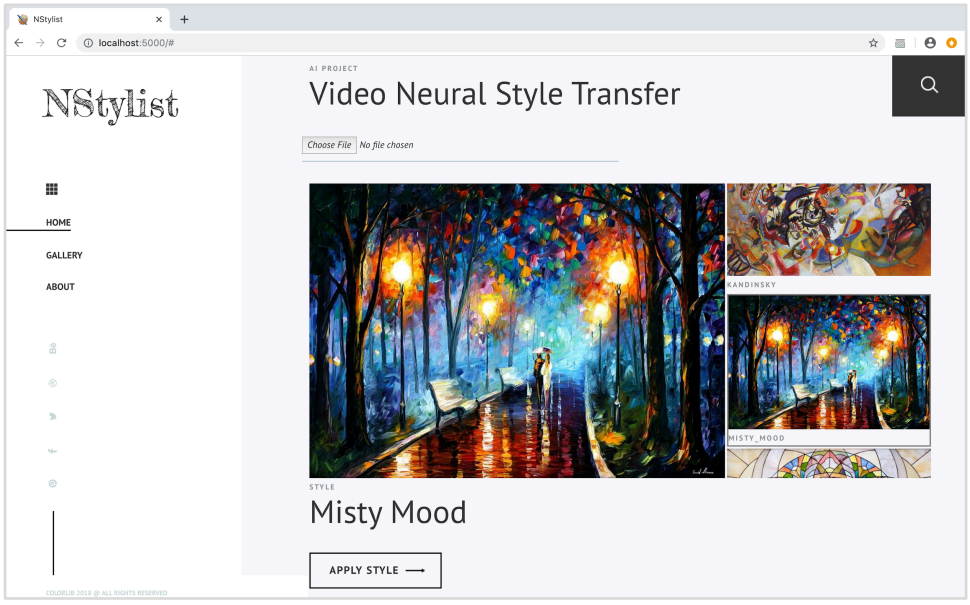
The frame-wise method transforms noise to a stylized image by repeating forward passing through a model and updating the noise with gradients from a context loss given the original image, a style loss given the style image, and a regularization loss. Applying this method to videos, requires this process to be performed on every individual frame, which means forward and backward passes through the model every frame. As you can imagine, this is infeasible to perform anywhere near real-time style transfer on video. The new method we are implemented takes a different approach. It aims to create style transfer models conditioned on specific style images by training a model offline through a training dataset. Like the paper we use videos from Videvo.net, transform the videos into frames, and create our own data loader, which outputs consecutive video frames from random videos without replacement as batches. The approach to train this network is the classic supervised deep learning training scheme. For every epoch we iterate through the entire dataset using batches of consecutive frames from random videos, feed the frames through a lightweight network, calculate the total combined loss on the stylized frames, and backpropagation through the network, updating all the learnable parameters. The loss for this training scheme consists of similar losses as before: content, style, and regularization loss, but also adds a temporal loss component. The goal of this approach is to train the model for around one or two epochs (through all the video frames) and then apply the lightweight trained model conditioned on a style image to any inputted video during inference.

As we mentioned before we wanted to introduce a temporal loss to reduce the amount of discontinuity between consecutive frames. This new method employs exactly such a loss, the main idea is that each forward pass through the model takes a batch of two consecutive frames, and we form a correspondence between the first stylized frame and the second stylized frame. This correspondence is modeled by optical flow, a calculated flow field representing the motion an images undergo from timestep to timestep. Here we calculate the forward and backward optical flow between every frame pair in all the videos, and save the flow matrices. The loss uses the precomputed optical flow field to warp the first stylized frame to give a representation of the second stylized frame, then we proceed to minimize the average $l2$ distance between the warped stylized frame and the second stylized frame. However, there are certain pixel regions that should not be included in the loss because they would provide misleading information: pixels in occluded regions and motion boundaries. Thus these pixels are given a weight of 0 to the loss, where all other pixels are given a weight of 1. To calculate the occluded regions and

motion boundaries pixels we used the inequalities mentioned in this paper. The overall idea of this temporal loss, is that we want the stylized images to be fairly similar when accounting for the motion between frames, so we want to avoid the stylizing process from producing discontinuities. In addition, one caveat of the paper and a large amount of deep learning model, is the need to utilize a graphics processing unit (GPU), to achieve real-time inference and manageable offline training runtimes. Due to the difference in hardware, it can be quite tedious to correctly port a CPU ready training routine to a GPU.

Unfortunately, near the end of our implementation we faced a couple of large roadblocks. For one, finding an easy to use optical flow algorithm was difficult because the cited algorithm used in the paper, required the Caffe framework, as well as old versions of python and GPU libraries, and other modern optical flow algorithms had intricate, required data preparation. We ended up using a dense optical flow algorithm from OpenCV, but even though we were able to compute forward and backward flow fields for the temporal loss, it was not simple to verify their correctness, especially due to the small amount of movement present in most consecutive pairs of video frames. In addition, we ran into problems transferring our CPU ready code to the GPU. We ran into many errors native to the GPU implementation that were not present in our CPU implementation. Struggling to overcome these roadblocks left us with little to no time to verify the complete correctness of our implementation. We planned to train the model on a training dataset of 90 videos. Unfortunately due to a lack of time, we had to cut short the training to 34 videos. After the model was trained, we planned to slightly fine tune each model's hyperparameters to work well with our chosen conditioned style images. Due to the time constraints, we did not fine tune the hyperparameters further, so we kept the values as mentioned in the paper. However, currently our models produced by our implementation of real-time video style transfer are producing grey videos, devoid of representations of both the original content video and the conditioned style image.

The desktop app was built as planned. The user would be able to select a style image and load a video. The video gets processed, which is displayed back to the user.



AI Components

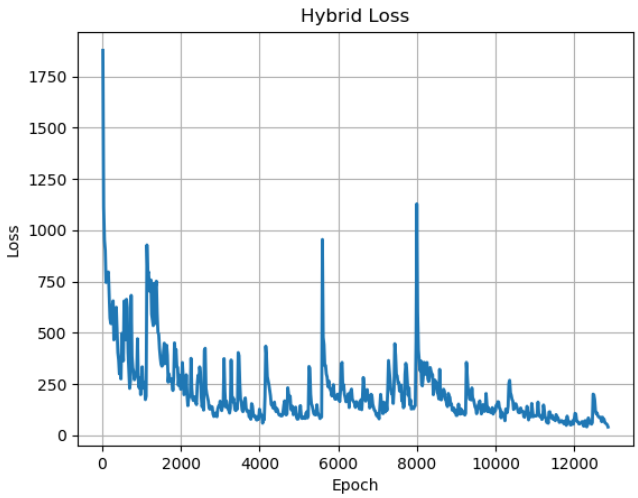
The key aspects of AI in our project are convolution neural network, deep learning, computer vision and transfer learning.

Evaluation

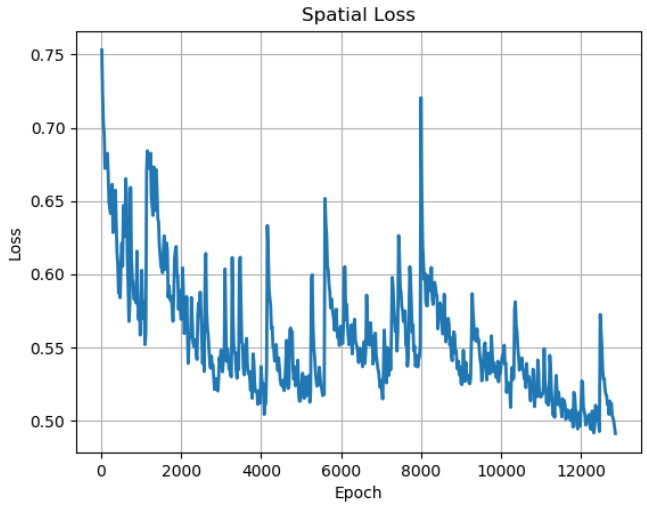
We wanted the video to look similar to the style of the image without making it difficult to see. However, this was not obtained and the styled videos are appearing grey. Also, we wanted it to not flicker as the frames changed. Another thing to keep in mind is that we want the inference time to be reasonably short, since we plan to build a browser based application, which will allow users to upload their own content videos and get the styled videos within a short time (say about 5 minutes, depending on the frames numbers). This was when we searched for an answer and we obtained the paper.

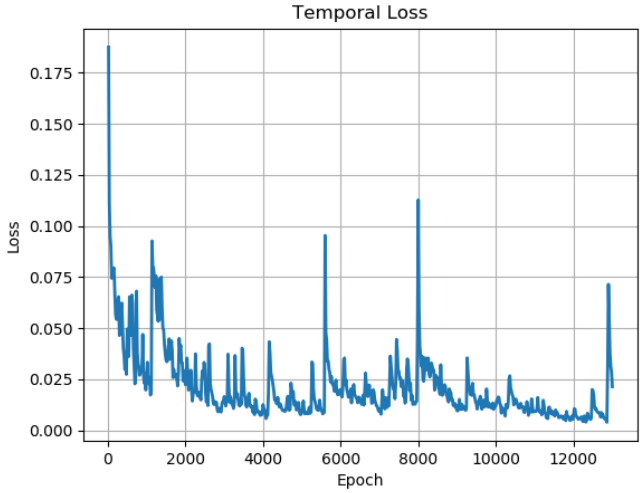
Quantitative Evaluation

The major chunk of the quantitative evaluation was to measure how similar the flow of the stylized video was compared to that of the original. Since we were not successful in implementing the idea in the paper, we could not evaluate the application properly.
Here are the losses obtained while training the models. This shows that the network was learning something.



,





Qualitative Evaluation

Due to the nature of our project, a main component of the qualitative evaluation condenses into how interesting, or “cool” the generated video appears. It is important that the produced video maintains the structural form of the content video, but also adapts the style from the designated style image. In addition, the generated video should look smooth, it should not introduce unpleasant jaggedness that makes the video unappealing. Clearly qualitatively measuring these characteristics has a great deal of subjectivity, thus in addition to evaluating them ourselves, we plan on showing the generated videos to others.
We planned on generating 2 videos for 3 styles, one in the Artisto app and the other in our app. And the idea was to add them to a Google Form and ask a selected group of people to pick the best from the 2 options.
Unfortunately, since our videos were graying out, we could not try out the Qualitative Evaluation.

References:

Haozhi Huang et al.; Real-Time Neural Style Transfer for Videos. In CVPR, 2017. http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Real-Time_Neural_Style_CVPR_2017_paper.pdf

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge; A Neural Algorithm of Artistic Style arXiv:1508.06576 [cs] (2015). https://arxiv.org/pdf/1508.06576 ArXiv: 1508.06576.

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge; Image Style Transfer Using Convolutional Neural Networks. In CVPR, 2016. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

Manuel Ruder, Alexey Dosovitskiy, Thomas Brox; Artistic Style Transfer for Videos arXiv:1604.08610 [cs] (2016). https://arxiv.org/abs/1604.08610 ArXiv: 1604.08610.

Videvo Team. Videvo free footage. http://www.videvo.net, 2016.

My.Com. Artisto. https://artisto.my.com/, 2016.

Prisma Labs inc. Prisma. http://prisma-ai.com/,2016.

Colorlib, https://colorlib.com, 2018