Across The Spider-Verse Style Transfer

Arjan Chakravarthy, Abhijith Chandran, George Chemmala, and Siddu Sitaraman

May 10, 2024

Abstract :

This research works to enhance image style transfer, applying artistic styles to content images using CNNs, with a focus on the movie Across the Spider-Verse. Inspired by prior works, the project optimizes style transfer for the content in the movie by using techniques like transfer learning on the CNN used as the base for style transfer. Results show effectiveness, especially when averaging styles. Challenges have included data collection and model training complexities.

1 Introduction

The primary objective of our project is to explore and enhance the capabilities of image style transfer, a powerful and creative application of convolutional neural networks (CNNs) . Style transfer involves applying the artistic style of one image to the content of another, allowing for the creation of unique, stylized images. In our case we chose to apply this to the movie Across the Spider-Verse. Our interest in this area was sparked by seminal works such as those by *Gatys et al.*, which introduced the neural algorithm of artistic style transfer, and subsequent improvements by *Dumoulin et al.* that proposed methods for more versatile style applications.

Our project seeks to re-implement and extend these foundational approaches. The reason for choosing this particular area of research stems from its blend of art and technology, presenting a fascinating challenge in computer vision and machine learning. Specifically, our focus is on optimizing the style transfer process for better speed and accuracy with the content we are working with. This project will primarily involve elements of structured prediction and unsupervised learning, as the task does not require labeled data in the traditional sense but rather leverages the intrinsic properties of the images themselves.

2 Background

We've drawn inspiration for our project from various sources (listed below). We are interested in utilizing style transfer which is quite well documented. The first two sources (*Gatsy et al.* and *Dumoulin et al.*) broadly introduce the concept of image style transfers using CNNs. Beyond these original reference papers, we have found other sources that show novel applications and extensions of style-transfer that may be helpful. For example, we have looked into many cutting-edge improvements to model speed and accuracy in papers such as Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization which uses adaptive instance normalization to quickly change styles based on learned feature patterns. We have also been looking at new approaches to quantifying stylized image loss as described in A Style-Aware Content Loss for Real-time HD Style Transfer. Finally, we have the goal of potentially extending our work to videos and even real-time transfer, as detailed in the prior and many other papers.

3 Methods

We used a combination of transfer learning and style transfer during this project. Transfer learning was used in order to improve the CNN we were using to identify features in order to create style and content representations for style transfer.

3.1 Transfer Learning

For the transfer learning, we attempted to train the VGG model on our own data that we had collected from the movie in order to improve the results on the style transfer. Our idea was that by training the model architecture on our own dataset, we would be able to make the layers that are later used for style transfer more accustomed to images from the movie. To achieve this, we set up a classification task where the aim of our model was to take in an image from the move (from one of the five universes) and output the universe from which the image was taken. We only used a training dataset since we were not too preoccupied with the accuracy of our classification model. This was because we wanted the CNN layers of the model to be tuned towards our dataset and the classification task was simply a means to achieve this aim. For this `TransferCNN`, we used the convolutional layers from the VGG model but constructed our own head of Dense layers (specifically three dense layers with dropout in between) so that our model would be able to train in a reasonable amount of time. We trained this CNN classification model for 10 epochs with a batch size of 16 and a learning rate of 10^{-5}. After training the model, we saved the weights of the model so that when we performed the style transfer, we could load the model weights and choose the certain layers that we would want to apply for the actual style transfer.

3.2 Style Transfer

The idea behind style transfer is to use a pre-trained CNN model to detect features and later decompose images into their style and content representations. We used the CNN we had developed from the previous section for this component.

We defined a class called `StyleContentModel` which selects layers from the CNN to use for style and content. Typically style layers are taken from throughout the entire model (reflecting a representation based on all types of features), while only a single content layer is taken towards the end (reflecting a representation of only big picture items). We then take a gram matrix of the features in the style layers in order to get a better idea of how features pair together. We also set up a call function for `StyleContentModel` to return style/content representations for any image we give it.

Next we define a class called `StyleTransfer` which uses `StyleContentModel` as an extractor of style/content representations.

We first extract the style/content representations using `StyleContentModel` for both the style image and the content image to get our style and content targets respectively. Then we

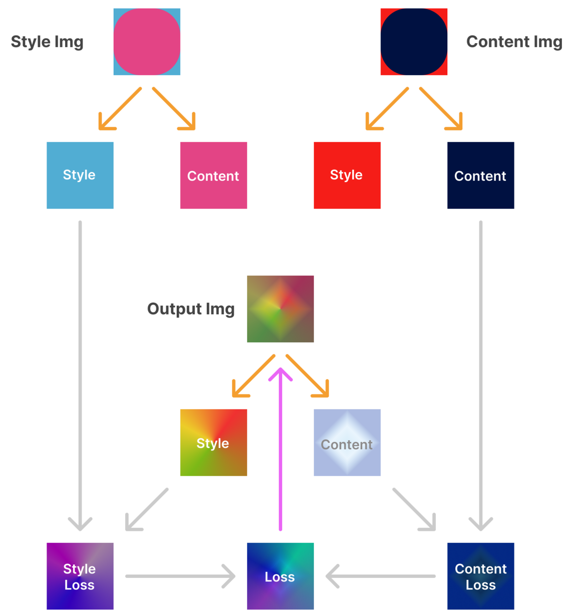

Figure 1. *Style Transfer Architecture*: Orange arrows represent extracting style/content representations, while the purple arrow represents gradient descent

extract the style/content representations for our output image (which is initialized as our content image) and compare the representations with our targets with MSE to get our style and content loss. Finally we combine the style and content loss with weightings to get the loss that we use to preform gradient descent on our output image.

4 Results

The images below depict some of the results that we received using our model for the style transfer.

Overall, we noticed that using the averaging of multiple styles 2 functionality made the style transfer arguably stronger, whereas with a single image 3, the style transfer naturally very strongly depicted the style of that single image. Moreover, with the averaging of multiple styles, sometimes certain areas of the image would have less chromatic artifacts (e.g., the green next to the nose on 3). For our results, we primarily used images from Gwen's universe as we found those images to have the most style (in relation to images from other universes in the movie) and would batter capture a good style transfer.

With our implementation, we also set up a loss graph 4 in order to analyse any issues with our model. We'd typically see plots reminiscent of an exponential decay, which would approach values around 10^6. Since we start off with the content image it is expected to initially see content loss start at 0 and then jump up after the model makes a correction for style loss 5 and then slowly decrease 6 7.

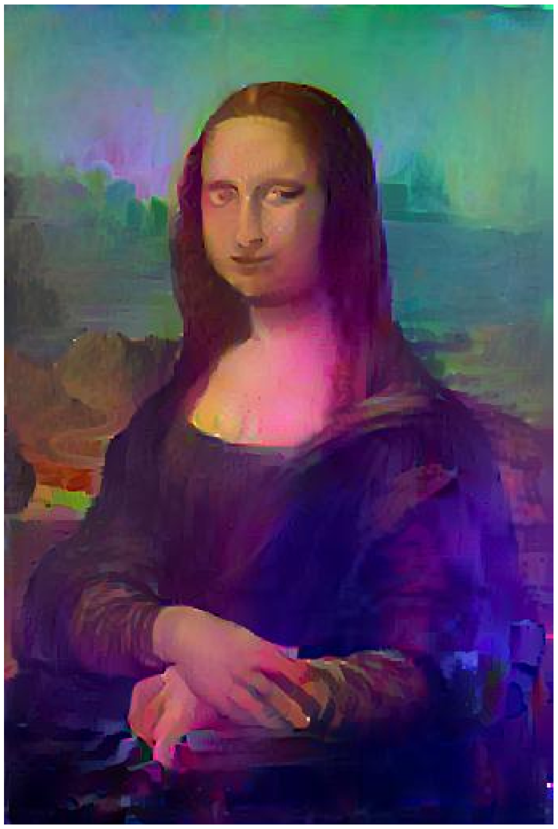

Figure 2. Style Transfer using Multiple Style Images

5 Challenges

One significant challenge that we faced was with the data collection part of the project. Since we wanted to collect data to train our `TransferCNN` (to solve a classification task), we needed images from each of the universes. To achieve this, we created a function that would sample images from each universe at regular intervals. The five universes from the movie that we collected data for were Miles' Universe, Gwen's Universe, Pavitr's Universe, the headquarters universe, and Universe 42 (the universe at the very end of the movie). Since we had more screen time for some of these universes, it was easier to collect data, but for some of the ones with less screen time (especially Universe 42), the images that we collected were very similar since we were sampling more than one image each second. This led the data in some universes (especially the ones that had more footage) to be more diversified while the images in one or two of the images were more homogenized. To tackle this challenge, we applied augmentations to increase the size of our dataset. Initially we collected 800 images for 3 universes and around 500 images for the other two images. By applying data augmentations, we tripled the size of our dataset. However, one problem we still faced was that the imbalances in the dataset size for each universe led our `TransferCNN` model to always predict one class (usually one of the classes with 2400 images). Since the data was imbalanced, the model could just predict this single class and receive a decent accuracy without really learning the features of each universe. To tackle this problem, we randomly sampled 1335 images from each universe which would end up compris-

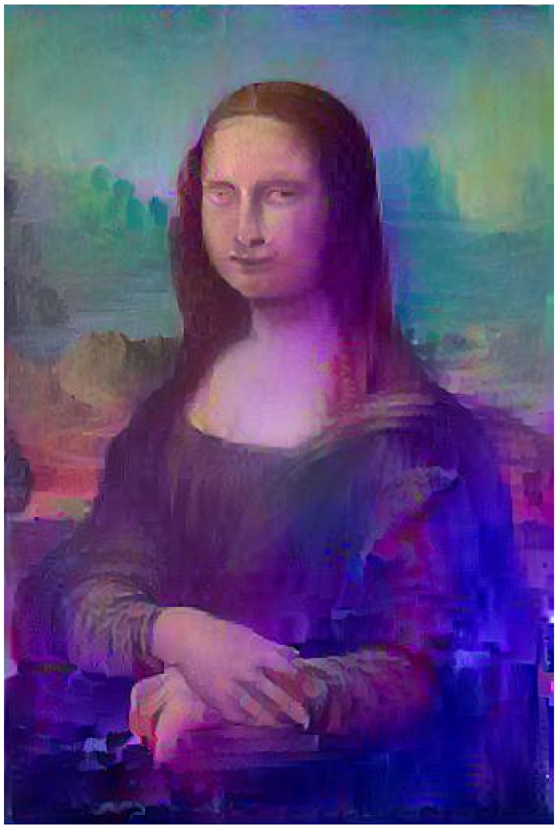

Figure 3. Style Transfer using Single Style Image

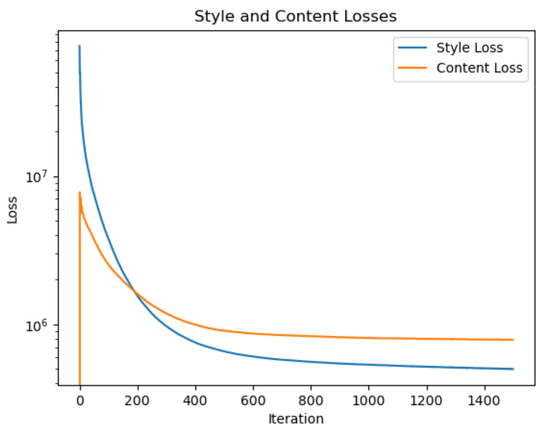

Figure 4. Style and Content Loss of 2

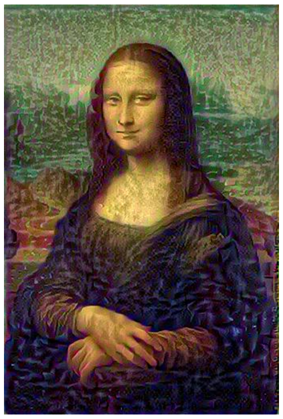

Figure 5. Initial Train Step

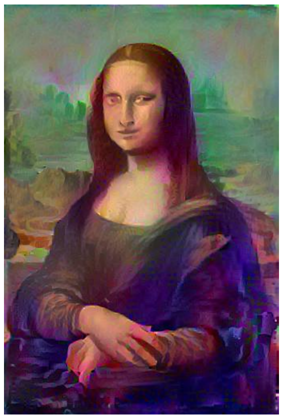

Figure 6. Mid Training

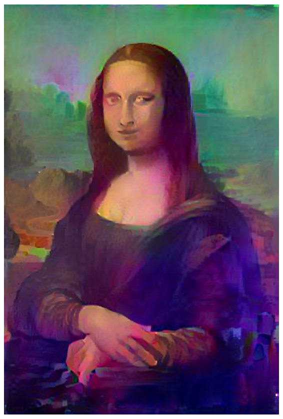

Figure 7. Almost Done

ing of our final dataset that we used to train the model. Since we had the same number of images for each model, this dataset now worked well for our `TransferCNN` model.

6 Reflection

In relation to our base, target, and stretch goals, we think we did pretty well. We achieved our base and target goals but did not have enough time to attempt the reach goal. We also really liked the results that we got from running our own `TransferCNN` model to apply the style transfer, and we found that averaging the styles across multiple style images generally made the output images more clear. Our reach goal was to work on implementing video style transfer, however, we did realize that this would be a bigger project given the time that we had and the other work we wanted to achieve with image style transfer itself.

Our model did work out the way we expected it to, though there were definitely challenges with building the `TransferCNN` model and getting it to produce reasonable results. There were several models that we trained that produced pretty bad results either losing most of the content image or applying the style incorrectly to the image. But eventually we were able to train a model that produced pretty good results, and we were happy with the outcome.

We had to change our approach quite a bit as we progressed through the project. We initially believed that implementing the `TransferCNN` model would be quite simple, but the whole pipeline from collecting the data to actually producing reasonable results was quite time-consuming. Initially, we thought that we could design our own CNN architecture to train our model on, however we quickly realized that the popular CNN architecture for style transfer was the VGG architecture, so we ultimately settled on this. Even with the VGG architecture, we tried training two different models, one where we loaded previous weights and one where we did not. Even though the accuracies on the classification part were quite similar, the results with the style transfer were generally better when using the previous weights, as expected. Initially, we also thought we may get better results by only training the few layers that actually do the style transfer, but we ultimately changed this approach to train all of the layers of our model as we noticed slightly better results.

If we had more time, we would like to add more functionality around averaging different style images. We currently achieve this functionality through the gram matrix (as explained earlier). Towards the end of the project, we set up some functionality where we could derive the variance of the styles to the average gram matrix. This would help in determining good images to combine when utilizing the average gram matrix functionality. However, we were not able to fully integrate this into our model, so if we had more time, we would likely spend time developing this further.

Our biggest takeaways from this project is learning the full model workflow. This project forced us to go through each stage of model pipeline (starting with data collection and preprocessing and ending with being able to easily generate results). We generally realized the importance of properly collecting and preprocessing the data for our model, which although may not seem like a critical step in the model deployment pipeline proves quite important in guaranteeing good results. Besides learning a lot about Style Transfer and CNNs in general, we also learned that breaking down our task into smaller, more manageable tasks not only allowed each of us to contribute to the final project, but also reach a working version of our model faster. Overall, we all really enjoyed working on this project.

Acknowledgements

Special thanks to my goat Aryan Singh