Real-Time Style Transfer – ENG 7111 Final Report

Neural Style Transfer:

Neural Style Transfer means to copy the style from one image (known as style image) onto another image (known as content image) while preserving the content. By style over here, we mean, the information regarding the patterns, brushstrokes etc present in an image.

Company Description:

This internship was offered to me by Australian Institute for Machine Learning (AIML). AIML is a research group based in Adelaide, Australia which is dedicated towards research in machine learning. It has around 160 members as of now and continues to grow. It was born out of Australian Center for Visual Technology (ACVT) in early 2018, funded by the South Australian government and the University of Adelaide. ACVT's long history of success and experience formed the core of what is now known as AIML. It is ranked among the top six institutions in the world for computer vision research and among the top three in Australasia for Artificial Intelligence research overall. AIML transforms research into impact through collaborating with world leading companies to develop products and solutions, conduct research, which is useful to society, raise Australia's profile as a land of innovation whilst also making sure to hire the bright and young minds of the state. Some of the ongoing projects at AIML include but not limited to the following domains:

- Computer Vision – Regular paper submissions at reputed computer vision publications like ECCV, CVPR, etc
- Robotics – Ongoing projects like an AI powered robotic dog.
- Medical Machine Learning – Projects involving commonly occurring ocular diseases like Glaucoma.
- Surveillance and Tracking: Real-time pose detection and object detection projects.
- Autonomous Systems
- Signal Processing
- Photogrammetry and 3D Modelling

AIML helps power all this research with its vast pool of resources like HPC clusters, vast number of GPUs lying around and while also providing an appropriate working atmosphere while catering to the needs of the people working over there.

What I did?

To begin with, I initially started with doing literature review for existing methods or techniques which were available for style transfer and to see if they were feasible for the task of style transfer on videos. Style transfer is something which has been going on for decades now. Neural style transfer is the same thing with the only difference of using neural networks to perform that task. I prepared a brief timeline for the style progress while researching various methods and summed up the most significant ones as follows:

- Early 2000s: Use of signal processing and filters to get stylized images (Gooch & Gooch, 2001)
    - Advantages: Easy and Fast to implement
    - Disadvantages: Failed to capture style in too much detail

- Early 2000s: Patch based methods like Image Analogies (Hertzmann et al., 2001)
    - Advantages: Produces decent results when compared to previous approaches
    - Disadvantages: Works in a supervised fashion and requires pairs of training data for training
- Neural Style Transfer: Gatys method of neural style transfer in which he uses convnets for the task of style transfer (Gatys et al., 2016)
    - Advantages: Outperforms all previous methods
    - Disadvantages: Based on an optimization-based approach and thus can be slow at times.

I started out with using Gatys method for style transfer by taking help from his paper which is primarily based on the finding that the 'decoupling' of the content and style representations is possible using a Convnet. The way they do so is to by using a linear weighted combination of two losses, a content loss, and a style loss which can be described below:

- <u>Content Loss:</u> The content loss tries to minimize the dissimilarity between the content image and the generated output image by penalizing the pointwise Euclidean distance between the two images.
- <u>Style Loss:</u> The style loss tries to minimize the dissimilarity between the style image and the generated output image in a similar fashion.

As good as this may sound, using this method for my project was not feasible simply because it worked on an optimization-based approach which could take some time to generate even a single example and thus was clearly not applicable for the task of real-time style transfer. Thus, after spending around one week playing around with Gatys method and trying to implement it I came to the following conclusion:

- Method: Gatys Optimization Method for Artistic Style Transfer
- Advantages: One of the first methods to explore ConvNets for style transfer and still considered as a gold standard in the industry.
- Disadvantages: Can be slow and not feasible for real-time based applications.

Moving forward, I started looking for other methods of neural style transfer and came across another novel approach of looking at Neural Style Transfer proposed by Johnson (Johnson et al., 2016) in which he uses a two-network architecture for training. The first network is a TransformNet which is simply an Encoder-Decoder network architecture. The second network is a VGG network (Simonyan & Zisserman, 2014) which they use for computing loss to better train the TransformNet. The only difference between this method and the previous one is that instead of computing the loss on the pixelwise values, they propose to use the VGG network to instead calculate the perceptual losses and then use that for training the TransformNet. The reason for this approach as specified by the authors is that they claim that by this way they are better able to capture the content and style features. I implemented this method too and came to the following conclusion:

- Method: Johnson's feed-forward method for Style Transfer
- Advantages: Fastest style transfer method till date
- Disadvantages: Does not consider temporal consistency in mind and thus not suitable for style transfer on videos where temporal consistency between subsequent frames is desired.

Having reached till this point, I began looking for methods which could be used for neural style transfer but keeping temporal consistency in mind and it is here where I came across a paper known as Reconet (Gao et al., 2019). Reconet builds upon where Johnson's work left and introduces two new temporal losses in addition to the three losses which were already being used by Johnson. Thus, in this model, a total of five losses were being used namely, feature temporal loss, output temporal loss, content loss, style loss and total variation loss. Each of these losses have been described in detail as follows:

- Content Loss: The content loss is same as explained earlier. The only difference being that the content loss utilizes feature maps at relu3_3 layer in this model.
- Style Loss: The style loss is again like the one explained before and uses feature maps at [relu1_2, relu2_2, relu3_3, relu4_3] layers, respectively.
- Total Variation Loss: Total variation loss works as a regularizer in our case and can be adjusted to control the level of smoothness in our stylized frames.
- Output Temporal Loss: All previous losses ignore the effects of luminosity while calculating loss. Keeping this in account, the relative luminance Y = 0.2126R + 0.7152G + 0.0722B, same as Y in the XYZ colour space, is added as warping constraint for all channels in the RGB colour space along with ground truth optical flow and the ground truth forward occlusion mask.
- Feature Temporal Loss: The feature temporal loss penalizes temporal inconsistencies on the encoded feature maps between two consecutive input image frames, thus minimizing the flickering effect between two frames which were earlier present in Gatys model.

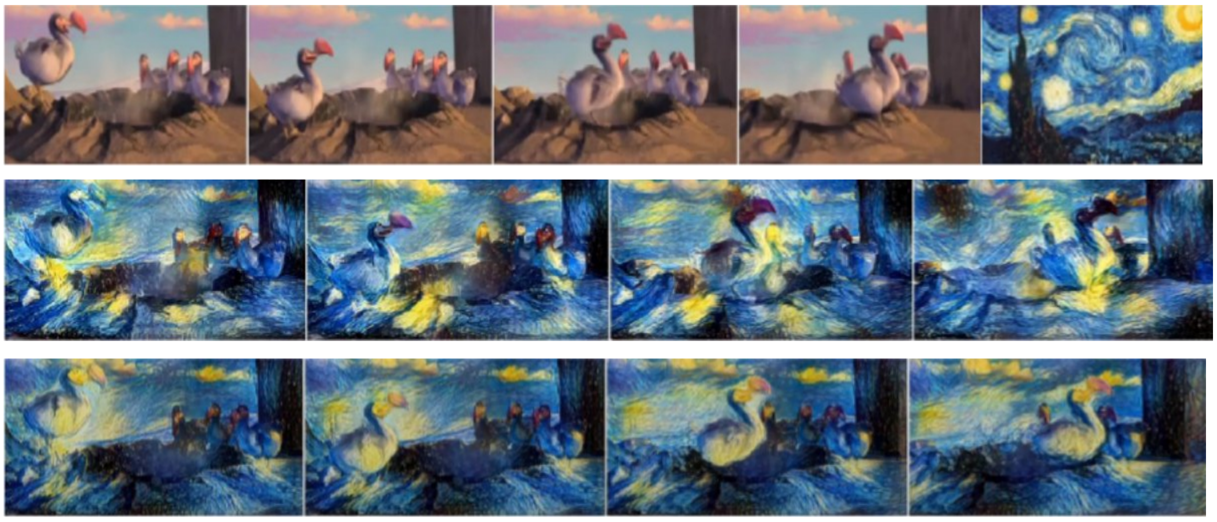
Fig. In the above figure, the first row represents the content images and the style image (Starry Night in this case). In the second row, we can observer how the frames look like if they are independently processed without keeping temporal consistency in mind whereas the last row shows the same input frames with the same style applied but with maintaining temporal consistency.

However, as I was implementing and working with this model I came across an unusual anomality which should not be happening otherwise. I noticed that the generated images had multiple white patches. This can be explained because of the use of Instance Normalization in the original Reconet model. Now, Instance Normalization has been proven to lead to white artifacts/bubbles in the formed image as described by (Karras et al., 2020). However, this did not work in my case. Either the style or the content loss did not converge, or some blurry artifacts appeared, and this is where my work comes in. Instead of using the ideal proposed solution, I worked with something known as Filter Response Normalization with a Threshold Linear Unit (Singh & Krishnan, 2020). It acts like Instance Normalization but preserves mean values in some sense. This normalization leads to the same results as original architecture but without the bubble artifacts. Using FRN + TLU combined with the reconet model I was able to get rid of the white bubble artifacts appearing in the style transferred samples.

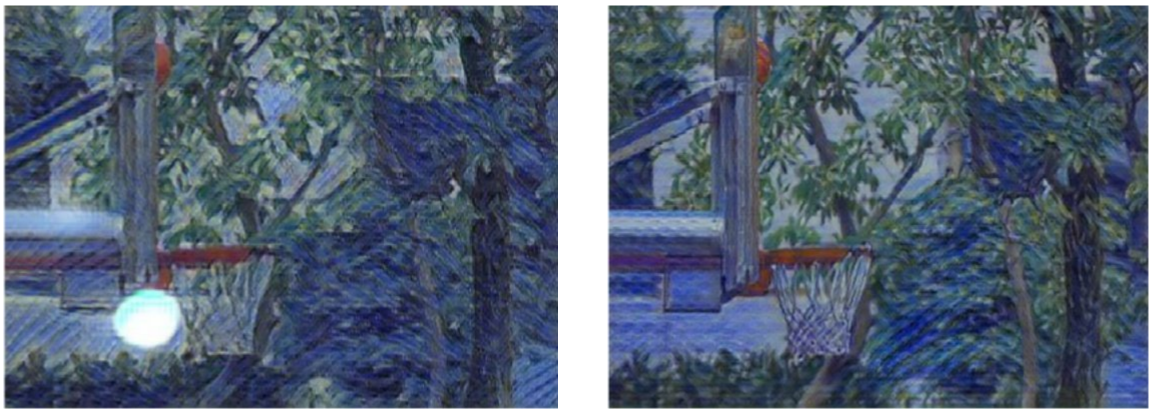

Fig. In the left figure white artifacts can be seen as it is generated using the original recoent model from the paper. However, in the figure on the right there is no such artifacts as it also uses the reconet model but with FRN + TLU applied.

After I was done training my model, my primary task was to get it to run in real time. I tried with different image resolutions and was able to conclude that if the images are under the dimensions of 500x500, the algorithm can run at real time. However, as we keep on increasing the size of the input image the algorithm tends to run slow.

After everything was done, I proceeded further to put everything together. I used a Logitech c922 camera and OpenCV to capture live feed and input those frames to my model. The camera had a latency of approximately 33ms whereas the model had a latency of around 65ms. However, note that this latency was on an Nvidia GTX 1060 GPU and this model can be made to run even faster and with more frames with some modular GPU which can reduce the GPU latency. Thus, combining these two and adding around 30ms for extra calculation delay we get an average frame rate of 8 FPS for 512x512 input image resolution.

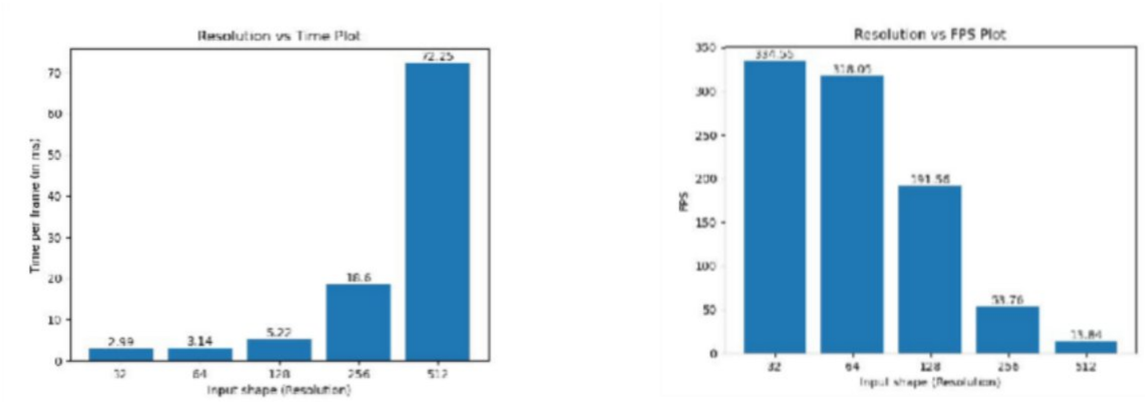

Fig. The figure on the left shows the inference time taken in ms for one sample image by our model and the figure on the right shows the average frame rate we get for different input resolutions. All these benchmarks were conducted on a randomly generated noisy image.

To sum up what I did, although initially I expected it to be a simple neural style transfer task and thought that it would not take me much time to get it done but as I progressed through my internship, I realised that it was completely different from my expectations which I had earlier. Although it is safe to assume that the project was still style transfer only at the end of the day. The only difference was that it was style transfer along with more added constraints like working on a higher resolution while also maintaining temporal consistency among consecutive frames and that too in real time.

What did I Learn?

During these 12 weeks, there were numerous things which I learnt along the way which helped me with my internship out of which some I have summarised below:

- Foundational Maths: I had to go through a lot of maths topics and had to revise them again to better understand the different research papers which I was following during my internship. Some of these topics were covered in my courses like Foundational Maths for Data Science and Introduction to Statistical Machine Learning which were of immense help to me.
- Optical Flow: Optical Flow was something which was completely new to me, and I had never heard of it before. However, for my project I had to learn about Optical Flow too like how it is calculated and its significance in helping maintain temporal consistency among consecutive frames.
- PyTorch: Although I had studied about PyTorch before and had also worked with a few examples, I had never used PyTorch in an actual project and always preferred working with Tensorflow. During this internship, I was not only able to go through PyTorch tutorials again but also get more hands-on experience with it.
- Data Logging: I learnt how to use TensorBoard and Weights and Biases and how to integrate them with PyTorch for better understanding of what my models were doing and to monitor them better. However, for this project, TensorBoard was sufficient for me, and I did not have to integrate Weights and Biases though.
- Bigger does not usually mean better: Prior to this internship, I had a common notion in mind that bigger models usually work better when dealing with Machine Learning, but it clearly was not the case in my project. I also tried to experiment with CycleGAN (Zhu et al., 2017) for Style Transfer and while it does indeed do an amazing job at it, it was not real time and hence not suitable for my project Thus, it clearly depends on what my task was and how one tries to trade-off between speed and accuracy. While CycleGAN did promise better results it was too slow and thus I had to look for more conventional ConvNet based approaches as compared to a GAN based approach.
- Run-Time Optimization: One problem which all big networks suffer from is run-time inference. Sometimes our models can achieve commendable results but are too slow to generate them and I faced a comparable situation when working with my model. I had to study about run-time optimization which involved techniques like Quantization, Mixed-precision training, and Knowledge Distillation. I also learnt about tools like PyTorch-JIT and TensorRT to help speed my inference time to achieve the goal of real time style transfer.

Apart from the technical skills that I gained from this internship, there were other valuable skills too which I would like to add to this list which I feel could help me further in my life whenever I am working at some other place. Some of these are:

- Effective Work Communication: Being an introverted guy and suffering from anxiety, I find it exceedingly difficult to converse with people. There were times when I even had some results but was not able to communicate them properly. Working at AIML under the supervision of Dr. Cuong and Dr. Aprit, I was able to overcome this fear of communication. We had regular meetings from time to time where I was asked to present my results to them. This not only helped me to better convey my results but also to effectively convey them in an efficient manner.
- Effective Time Management: While I was working on this internship, I was also simultaneously taking my course subjects at university and dealing with my casual job. Being a part of this internship taught me how to effectively manage my time and prioritize tasks in my life.

The abovementioned skills may not seem of much importance to some people from a report point of perspective, but for to me, these were equally important skills I gained from this internship apart from the technical knowledge which would help me in some other work environment as well as further on ahead in my life.

Advice for other interns:

There are a lot of things which I feel I could have done better and would certainly like to improve upon if given a chance to do so. Out of all things, some of these which I would like to convey to other interns are:

- Communication is the key: There will be instances when one will feel stuck and will not know where to go from. In my case, I was facing some technical difficulties with my internship which led to an unwanted delay. As the final date was approaching, I was quite panicked and so I told everything to my supervisor. I was pleased to know that if I open about my problems to others, there are people who will listen to it and try their best to help me out and I just wish I had done it a little bit sooner rather than constantly worrying about it for so many weeks.
- Do not wait for anyone: There will be times when one might face some difficulties due to some issues which may be caused by something which is not in their hands. In my case, it was some technical issues but my advice to all future interns would be to not wait for anyone and rather get started on their own hardware or rent hardware from cloud and try to stay on their planned schedule cause if they do not, at the end of the day when the last weeks of the internship will be approaching it will be too much to handle all at once.
- Networking: According to me, internships not only offer learning experience but also provide a platform for meeting new people and engaging with them both for personal and professional interests. There may be instances when you might meet someone and be able to land your next internship or job even before completing the ongoing one. Thus, to make the absolute best of your internship, make sure you talk to at least one new person every now and then and stay in touch with the people you have met already so far.
- Do not panic: One thing that every student needs to remember while interning is to not panic. Always remember that it is an internship, and no one expects you to know everything beforehand. Internships are more of a learning experience rather than an actual full-time job experience, so you are allowed to make mistakes if no one else suffers due to them.
- Log everything: Make sure to log everything and every result even if they might not seem of much significance at that time. This way, when you will be making your final report at the end of the internship you would not have to read the entire code again but would rather have everything you need already done for you.
- Stick to a schedule: Last but not the least, my advice to all future interns would be to make a schedule at the beginning of the internship and make sure to always stick to it no matter what happens. This way, it will help you at the end of the day when the assignment submission dates would be nearby.

References:

Gao, C. *et al.* (2019) “Reconet: Real-time Coherent Video Style Transfer Network,” *Computer Vision – ACCV 2018*, pp. 637–653.

Gatys, L.A., Ecker, A.S. and Bethge, M. (2016) “Image style transfer using Convolutional Neural Networks,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Gooch, B. and Gooch, A. (2001) “Non-photorealistic rendering.”

Hertzmann, A. *et al.* (2001) “Image analogies,” *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*

Johnson, J., Alahi, A. and Fei-Fei, L. (2016) “Perceptual losses for real-time style transfer and Super-Resolution,” *Computer Vision – ECCV 2016*, pp. 694–711.

Karras, T. *et al.* (2020) “Analyzing and improving the image quality of stylegan,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*

Singh, S. and Krishnan, S. (2020) “Filter response normalization layer: Eliminating batch dependence in the training of Deep Neural Networks,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*

Zhu, J.-Y. *et al.* (2017) “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *2017 IEEE International Conference on Computer Vision (ICCV)*