COMP6248 REPRODUCABILITY CHALLENGE A LEARNED REPRESENTATION FOR ARTISTIC STYLE

Joseph Early, Jamie Sian & Alexander Stonard
School of Electronics and Computer Science
University of Southampton
{je5g15,js17g15,ads1g15}@soton.ac.uk

1 INTRODUCTION

The aim of this work is to re-implement and reproduce a previously published ICLR conference paper. Our chosen paper came from ICLR 2017, and was titled 'A Learned Representation for Artistic Style' (Dumoulin et al., 2017). It focuses on style transfer in images - the process of extracting a style from a source image and applying it to a content image such that the content image is recreated with the extracted style yet retains its content. The major contribution of this paper was to extend previous work in style transfer to allow a network to learn several representations at once. These so called n-style networks are more efficient than single style networks as they capitalise on the shared features between different styles to improve the rate of training and reduce the space required to learn multiple styles (learning n styles previously would require n distinct networks, but a single n-styles network can encapsulate the same learning).

Our study focused on three different reproducability topics: comparing n-style and single style networks, style reconstruction, and style interpolation. The next section of the report gives a brief implementation overview, following by three sections detailing with each reproducability area in turn. Finally the the overall success of the reproducability investigation is summarised in the conclusion. The source code for this work is available online[1], and additional style transfer results can be found here[2].

2 IMPLEMENTATION OVERVIEW

The main aim of style transfer is to produce a transfer network that learns a specific style or set of styles, and then, given a content image, transforms the image into a new image with similar content but with the new style(s) applied. To achieve this, the transfer network is trained against a feature extractor (a deep pre-trained network); by comparing the difference in features between the transfer network's output image and the original content and style images, a notion of the content and style loss can be computed. If the transfer network is able to minimise these losses simultaneously, it will be able to transfer its learnt style to any input content image.

Our progression for developing n-style networks began with training single style networks. Based on the work of Johnson et al. (2016), we were able to develop single style networks trained using a pre-trained VGG-16 network as a feature extractor and the COCO dataset as source of images. This created a solid foundation for moving onto the n-style networks developed by Dumoulin et al. (2017).

One element of the learning process that was unclear from the literature was how to use the VGG-16 network as a feature extractor to compute content and style losses. The work of Johnson et al. (2016) and Dumoulin et al. (2017) gave conflicting results on which layers of the network should be used as extracted features. We used both sets of layers and perceived little difference, but to ensure we were confident that we were using the VGG-16 network correctly, we performed style reconstruction as per Gatys et al. (2015) - see §4.

[1] https://github.com/COMP6248-Reproducability-Challenge/A-Deep-Fried-Approach-To-Artistic-Style

[2] https://github.com/COMP6248-Reproducability-Challenge/A-Deep-Fried-Approach-To-Artistic-Style/blob/master/stylised_images.pdf

3 SINGLE STYLE VS n-STYLE NETWORKS

There exists a natural trade-off for single style and n-style networks - while n-style networks provide a space and time advantage (only one network is required rather than n), they must find a common representation between the set of styles, which can potentially lead to reduced efficacy of the style transfer. This was investigated by Dumoulin et al. (2017), and we aim to reproduce their findings as part of this work.

First, the training process of an n-style network was investigated. It was trained against 10 styles for 40,000 parameter updates as per the original paper. This was performed on a Google Compute Engine instance using an Nvidia T4 16GB GPU and took approximately four hours at 2-3 batches per second - comparable to the time reported by Johnson et al. (2016). As shown in Figure 1, the network is successfully able to learn all ten styles simultaneously. The original paper did not specify how to select the 'active' style during training, so our implementation used a round-robin system to constantly expose the transfer network to each style. Part of the work by Dumoulin et al. (2017) investigated fine-tuning networks (adding an $n + 1^{th}$ style to an existing n-style network), which proved to be successful, so there is further potential to try other active style selection methods, for example learning each style sequentially rather than shuffling them as we did.

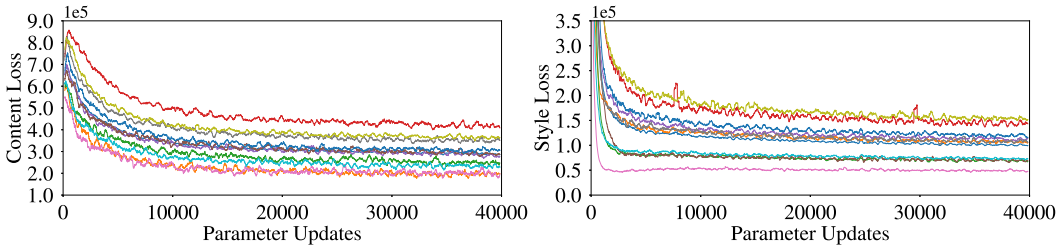


Figure 1: The content and style loss when training an n-style network against ten styles.

A further insight from Figure 1 is that the network does not learn each style equally - some have greater losses than others. This leads to the question of whether the network is learning some styles better than others, or whether some styles are just have fundamentally higher loss functions. To establish if this is the case, we compared the loss of an n-style network against n single style networks. The n-style network was trained for 10,000 parameter updates against five styles, and the single style networks were also trained for 10,000 parameter updates. This is a similar experiment to one performed by Dumoulin et al. (2017) to compare the efficacy of n-style networks against single style networks.

As can be seen in Figure 2, the single style networks losses follow a similar trend to the n-style network losses, suggesting that some styles are indeed harder to learn than others. Overall, the n-style network is slightly worse at learning individual styles, which confirms that using an n-style network does indeed sacrifice some of the style transfer efficacy.

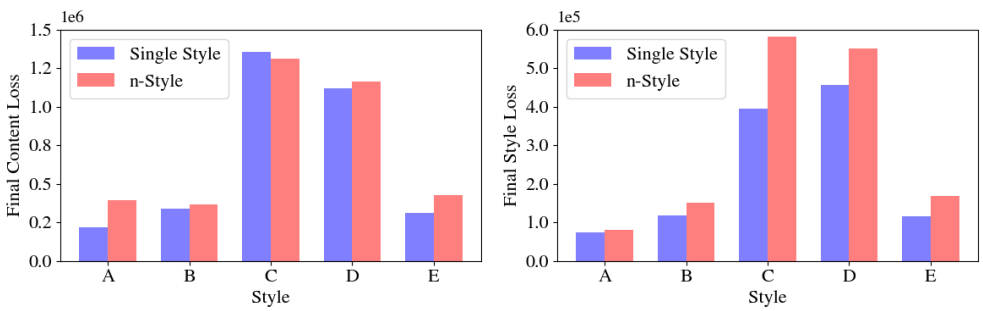
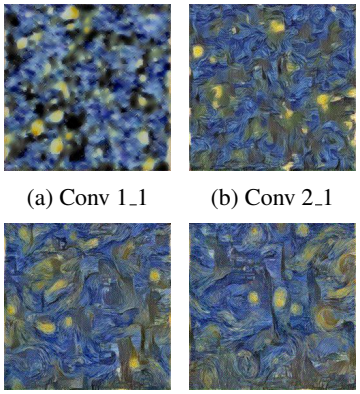


Figure 2: A comparison of the performance of an n-style network against five single style networks.

4 STYLE RECONSTRUCTION

Using Van Goghs 'Starry Night' as a base style, we aimed to analyse the effectiveness of VGG-16 as a style extraction tool. To achieve this we used gradient descent starting from a noisy image of size 256 x 256px, with RGB values sampled from the distribution $X \sim \mathcal{N}\left(\frac{255}{2}, \frac{255^2}{4}\right)$. Following image updates via backpropagation we were able to visualise images minimising stylistic loss.

To run gradient descent we used the ADAM optimiser with a learning rate of 0.001 and early stopping enabled. To model style reconstruction we adopted the approach of Gatys et al. (2015), calculating the cumulative loss τ at the first convolutional layer of each convolution block n. To achieve this we used the following formula: $\tau_n = \sum_{i=1}^{n} \phi_i(x_i)$, where x_i represents the activation entering block i and ϕ_i represents the function calculating the stylistic loss for the first layer of block i only. Upon training, τ_n would decrease on each parameter update. The resulting images that minimised τ_n are displayed in Figure 3.



(a) Conv 1_1 (b) Conv 2_1

(c) Conv 3_1 (d) Conv 4_1

Figure 3: Style reconstructions for convolution blocks 1-4.

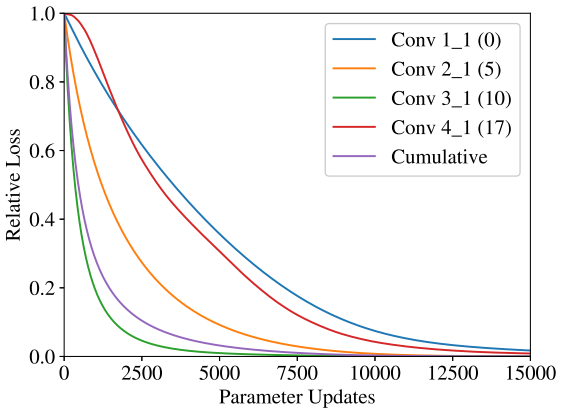


Figure 4: Constructing an image to minimise cumulative loss (τ_4 normalised) in convolution block 4.

As displayed in Figure 3, for 'Starry Night' we have produced similar results to that of Gatys et al. (2015), with higher level features being extracted by layers deeper in the VGG network. The corresponding losses for each of these images can be found in Table 1.

Table 1: Layer information for the style reconstruction loss

Layer	**Initial ϕ for τ_4** (x 10^7)	τ
Conv 1_1	0.0294	0.909
Conv 2_1	2.5988	7.336
Conv 3_1	8.4401	150.189
Conv 4_1	0.0758	15906.612

To determine which features posed the greatest importance for minimising style loss, we plotted $\phi_i(x_i)\ \forall i \in [1, 4]$, normalised by the initial ϕ when generating the image that minimised τ_4. This plot can be seen in Figure 4.

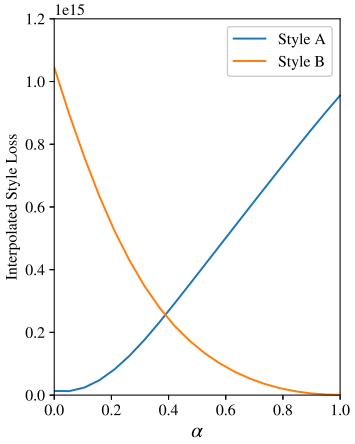
An interesting feature of Figure 4 is that the convolutional layer that minimised τ_4 the fastest was Conv3_1. Whilst deeper style layers are increasingly difficult to train, as shown in the τ column of Table 1, in this case the high initial loss posed the highest potential to minimise τ_4. A contradiction to this is the interaction between Conv1_1 and Conv4_1. Intially, Conv1_1 is learnt at a quicker rate than Conv4_1, until about 2000 parameter updates when Conv4_1 starts being learnt quicker. This implies that whilst it is initially easier to optimise Conv1_1, it has a relatively low ϕ; therefore the real (rather than relative) loss function optimisation in Conv4_1 presents a greater potential to decrease τ_4.

5 STYLE INTERPOLATION

When using an n-style network, the majority of the network parameters are shared between styles; only $\sim 0.2\%$ are used for individual style representation (Dumoulin et al., 2017). Due to this shared representation, it is possible to interpolate between two or more styles, viz. produce a blend of different styles and apply them to a content image. The original paper provides details how to interpolate between two styles, but lacked a more general description for more than two styles, although it did provide evidence that interpolation between more than two styles was possible. Therefore part of our work involved developing a generalised solution for n-style interpolation: by producing a normalised weighted vector $\boldsymbol{w}$, where $|\boldsymbol{w}| = n$, the interpolated style $\boldsymbol{s}$ is the weighted sum of the n base styles: $\boldsymbol{s} = \sum_{i=0}^{n} w_i \boldsymbol{s}_i$, where $\boldsymbol{s}_i$ is the set of network parameters that represent the i^{th} style.

A demonstration of the style interpolation that is possible is shown in Figure 5a - interesting new styles can be created by varying the weighting of four distinct styles. This means the style loss increases for each individual style (as shown in Figure 5b), however with an even weighting the interpolation is able to merge both styles very well.

(a) Four way interpolation.



(b) Style loss due to interpolation.

Figure 5: n-style networks allow for interpolation of multiple styles. (a) Each square of the grid uses a style weighting based on the distance from each style's corner. (b) α governs the weighting of style A ($w_A = 1 - \alpha$) vs style B ($w_B = \alpha$). As α increases, the interpolation shifts from style A towards style B, causing the loss of style A to increase and the loss of style B to decrease.

6 CONCLUSION

The main focus of this work was to try and reproduce the results of Dumoulin et al. (2017). This was relatively easy once the surrounding literature was considered as well, as the original paper left of some key details that were covered in the work that it built upon. One fairly major element that was missing was hyper-parameter tuning: the paper introduced style and content weights but didn't give specific values. Overall, our results match the original paper's very well, suggesting the paper and surrounding body of work does provide sufficient information for reproduction and the techniques described do indeed work.

REFERENCES

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *Proc. of ICLR*, 2, 2017.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.