CITS4404 Project Report

Angus Wylie (21962246), Lucie Cunningham (22260943), Jehan Sappideen (19523162)

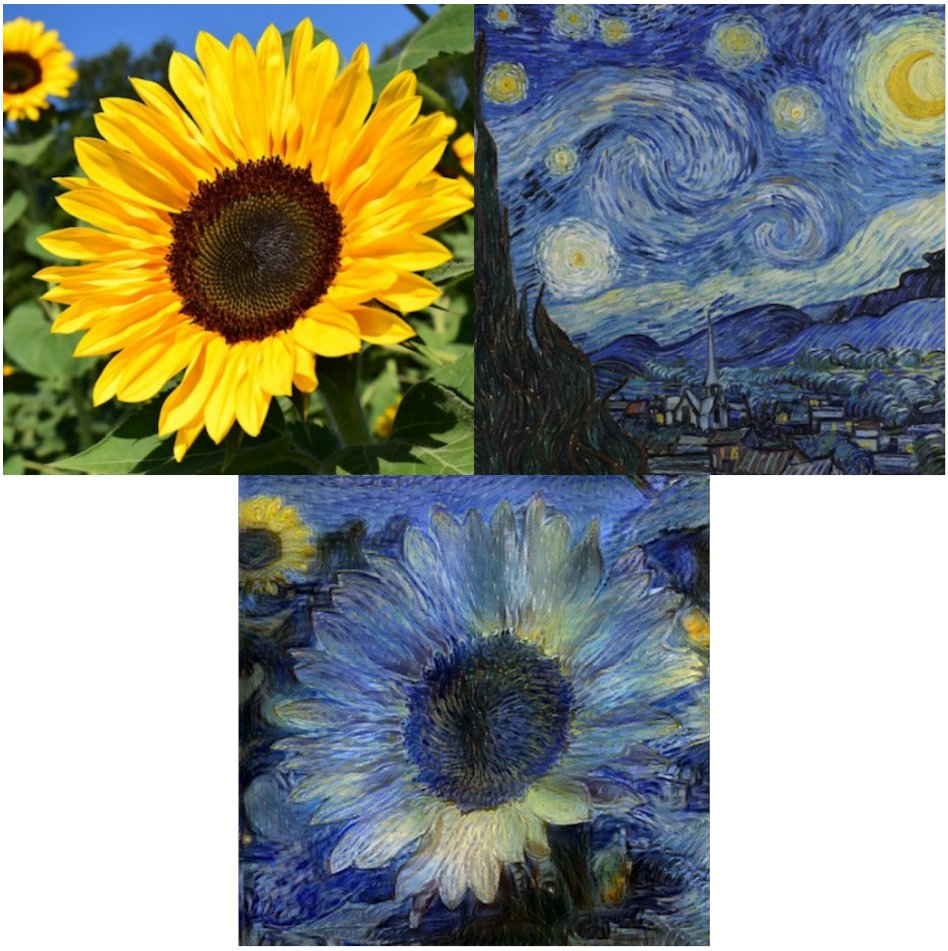

Figure 1: “Sunny Night”? An example of what can be achieved with neural style transfer.

1. Introduction

Style transfer is a relatively young topic of research within computer vision [1,2]. In style transfer the aim is to synthesise a new image containing both the content of one image and the style of another. Interest into this topic can largely be attributed to the works of Leon A. Gatys and his colleagues from their 2015 works "Texture synthesis using convolutional neural networks" and "Image style transfer using convolutional neural networks" [1,2].

Inspired by these works we chose the topic of neural style transfer for the CITS4404 group project. In this project we implement content reconstruction, texture reconstruction and style transfer as described in Leon A. Gatys' groundbreaking works. This is done as a series of IPython notebooks with theTensorFlow deep learning framework.

2. Literature Review

In this section the literature relevant to this project is presented.

Note that the works that this project is based upon are no longer the state of the art in neural style transfer, thus this review does not reflect the state of the art either.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNN) are a variant of neural networks developed for the processing of images and video, although they have found applications in other domains.

The design of CNNs is inspired by the human visual cortex. It has been observed that our visual system works by identifying specific patterns in small regions of our visual field known as receptive fields. These patterns are then progressively combined to form more complex patterns, until ultimately they are resolved into recognisable objects [3]. For example, multiple lines may form a square, then multiple squares may form a box, so on and so forth.

In CNNs this process of identifying and combining patterns in the image is performed by convolutional layers and the presence of patterns are represented in feature maps [3]. In the lower levels of the CNN the feature maps may represent basic patterns such as horizontal and vertical lines. In the higher layers of the CNN these feature maps may represent entire objects such as people.

VGGNet is a CNN architecture which came runner-up in the 2014 ILSVRC challenge [3]. It was developed by researchers at the Visual Geometry Group (VGG) lab at Oxford University. There are two variants; one using 16 convolutional layers and the other using 19. These are commonly referred to as VGG-16 and VGG-19 repectively. The VGG-19 trained on the imagenet dataset is the CNN forming the backbone of the works by Leon A. Gatys [1,2]. Figure 2 shows the architecture of the VGG-19.

2.2 Content Extraction and Reconstruction

Content reconstruction is the process of reconstructing an image from the high- level representation (feature maps) produced by a CNN. This process relies upon the fact that two images can be assumed to be approximatesly the same if when passed through the CNN they produce the same feature maps. Thus, to create a reconstruction for a given set of feature maps one must find an image which produces these feature maps when fed to the CNN.

This can be achieved by performing gradient descent upon the pixels of the image, using a loss based upon the difference between the feature maps the image produces and the feature maps that are desired. In the work by Leon A. Gatys, the sum of the euclidean distances between the corresponding feature maps is used as this metric [1].

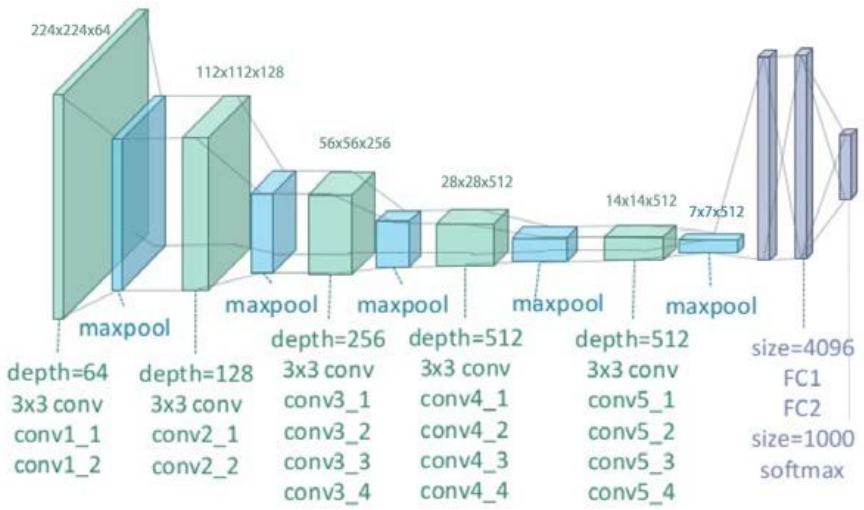

Figure 2: The VGG-19 architecture

Content reconstruction is performed as follows:

1. Initialize a random image of the required dimensions.
2. Pass the image through the CNN to get the feature maps.
3. Calculate the loss between the image's feature maps and those desired.
4. Perform backpropagation of the loss to get the pixel gradients.
5. Perform a gradient descent step on the image's pixels.
6. Repeat steps 2 to 5 for the desired number of epochs.

2.3 Texture Extraction and Reconstruction

Previous works in the region of texture synthesis have demonstrated that texture may be expressed using summary statistics over the spatial extent of an image. Any two images generating the same values for such summary statistics may be percieved as having the same texture.

Some of these past works used approaches where summary statistics were derived from feature maps on the image generated using human defined filters. It was Gatys however tho first made the switch to summary statistics derived from the feature maps of convolutional neural networks [1,2].

Texture reconstruction is very similar to content reconstruction. However, rather than comparing the raw feature maps of the image with those desired when performing gradient descent; the loss is calculated using the difference in summary statistics on these feature maps. Thus, rather that optimizing the reconstruction for content, it is optimized for texture.

In the work by Leon A. Gatys this summary statistic used to represent texture was the gram matrix. This is a matrix representing the correlations between all feature maps in the layer. The euclidean distance between the gram matrices of the layer give the loss.

Texture reconstruction is performed as follows:

1. Initialize a random image of the required dimensions.
2. Pass the image through the CNN to get the feature maps.
3. Calculate the gram matrices of the image's feature maps.
4. Calculate the loss between the image's gram matrices and those desired.
5. Perform backpropagation of the loss to get the image's pixel values.

6. Perform a gradient descent step on the images pixels.
7. Repeat steps 2 to 5 for the desired number of epochs.

2.4 Style Transfer

In style transfer the aim is to synthesise a new image containing both the content of one image and the style of another. This is achieved through the simultaneous application of content reconstruction and texture reconstruction. A new image is created from white noise and then gradient descent is applied to using a loss which is a weighted sum of the losses described in the above content and texture reconstruction sections. The content/texture ratio is the value which governs the relative weighting of the content and texture losses.

3. Implementation

This project was implemented as a series of notebooks, one for each of: content reconstruction, texture reconstruction and style transfer. These notebooks were developed and run using TensorFlow in the Google Colab environment, making use of the freely available GPU kernel to speed up processing time.

These notebook implementations are as described in the original works by Gatys et al. [1,2]. It should be noted, however, that the state of the art in style transfer methods has progressed significantly since.

4. Results and Discussion

In this section we present the results the from our implementation and discuss the effects of several hyper-parameters on the synthesized images. Note that there are no metrics for how "good" the resulting images look. Thus, the discussion of results presented here is somewhat opinionated.

4.1 Content/Texture Ratio

The content/texture ratio determines the relative weights of the content and texture losses in the overall loss function when performing gradient descent for style transfer. As such, this ratio has a significant impact on how the images look. A high content/texture ratio will result in images emphasising the preservation of the content over texture. The result is images that lean more towards the content image than the texture image. A low content/texture ratio has the opposite effect.

Figure 4 shows the effect of content/texture ratio by performing style transfer on the same pair of images for a variety of ratios. 4096 epochs are used for gradient descent. Notice how as the content/texture ratio decreases the synthesised images begin to resemble the texture image more.

4.2 Number of Epochs

The number of epochs used during gradient descent has a large impact upon the quality of the images produced. Increasing epochs generally results in the synthesised image looking more "natural". This comes at the expense of an increase of computation time, where the computation time scales linearly with the number of epochs.

For the generation of the results in Figure 3, 10,000 epochs were used during gradient descent. For content and texture images of dimensions 512 by 512 style transfer took approximately 20 minutes on the Google Colab GPU kernel. For some images reasonable results could be produced with fewer epochs (1024, 2048, 4096, etc.) taking less computation time. The number of epochs required for reaching a reasonable quality of synthesis is dependant upon the choice of images. If the content and texture images have similar content, colour, etc. then convergence occurs faster.

Figure 3: Results of our style transfer implementation. Content/texture ratio = 0.1, epochs = 10,000

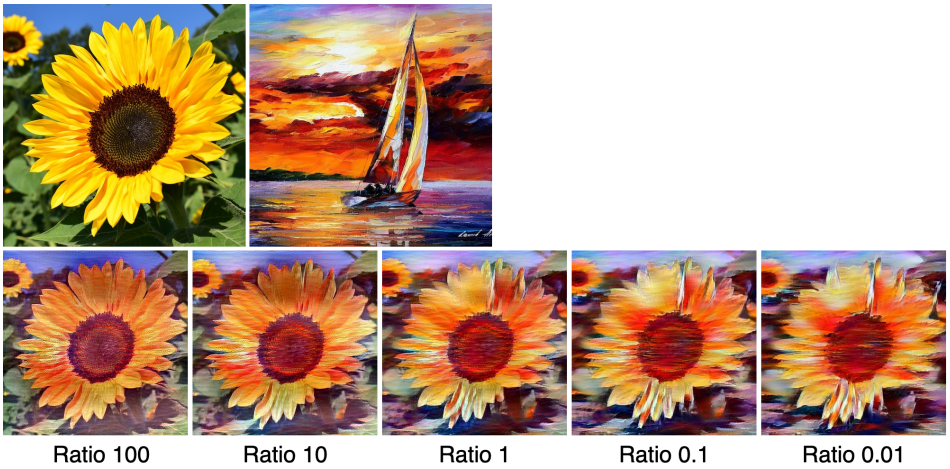

Figure 4: Style transfer for varying content/texture ratios

Figure 5 shows the effect of the number of epochs on the image quality by performing style transfer for the same pair of images for a variety of epochs in the gradient descent. The content/texture ratio used was 0.5. Notice how as the number of epochs is increased the amount of undesireable artifacts in the image decreased giving the image a more "natural" look.

Figure 5: Style transfer for varying numbers of epochs

References

[1] Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2414-2423).

[2] Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). Texture synthesis using convolutional neural networks. In Advances in neural information processing systems (pp. 262-270).

[3] Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.