Image style transfer using convolutional neural networks based on transfer learning

Varun Gupta*, Rajat Sadana and Swastikaa Moudgil

Department of Computer Science and Engineering,
Chandigarh College of Engineering and Technology,
Chandigarh, India
Email: varun3dec@gmail.com
Email: rajatsadana@hotmail.com
Email: swastikaa15@gmail.com
*Corresponding author

Abstract: The purpose of an image style transfer system is to extract the semantic image content from the target image and then using a texture transfer procedure display the semantic content of target image in the style of the source image. The uphill task in this context is to render the semantic content of an image but with the advent of convolutional neural networks, image representations have been made much more explicit. In this work, we explore the method for image style transfer using transfer learning from pre-trained models of convolutional neural networks (CNN). Use of these models gives us the power to produce images of a high perceptual quality that are a union of the content of an arbitrary image and the appearance of renowned artworks. Further, this paper compares pre-trained CNN models for image style transfer task and highlights the potential of CNN to deliver appealing images using modern manipulation techniques.

Keywords: image style transfer; convolutional neural networks; CNN; transfer learning; deep learning; machine learning; artificial intelligence.

Reference to this paper should be made as follows: Gupta, V., Sadana, R. and Moudgil, S. (2019) 'Image style transfer using convolutional neural networks based on transfer learning', *Int. J. Computational Systems Engineering*, Vol. 5, No. 1, pp.53–60.

Biographical notes: Varun Gupta received his PhD in Computer Engineering from the National Institute of Technology, Kurukshetra (Haryana), India. He is currently an Associate Professor in the Department of Computer Science and Engineering, Chandigarh College of Engineering and Technology (Degree Wing), Chandigarh, India. Before joining Chandigarh College of Engineering and Technology, he worked in the power sector as an Information Technology Engineer. He has published several technical research papers in leading journals and conferences from IEEE, ACM, Elsevier, Springer, etc. His research interests include artificial intelligence, deep learning, machine learning, big data analytics, smart grid and software engineering.

Rajat Sadana is currently pursuing Bachelor of Engineering in Computer Science and Engineering from the Chandigarh College of Engineering and Technology, Chandigarh. His main areas of research interest are pattern discovery using data mining and feature extraction using neural networks.

Swastikaa Moudgil is an undergraduate student in the Department of Computer Science and Engineering at the Chandigarh College of Engineering and Technology, Chandigarh, India. Her research areas include web analytics and artificial intelligence.

1 Introduction

In the field of fine art and painting, humans have proved their mettle by creating extraordinary visual experiences by creating an intrinsic interplay between content and style of image. Every time technology has been created it has been used by artists to be used as a creative tool. For instance, the camera was initially seen only as a tool to capture reality until it gave birth to animation. Recent advances in machine learning have allowed creating beautiful masterpieces with few lines of code. If we can extend these biologically learned patterns with machine learned patterns it will become much clearer that it is not just that the machines are our artistic competitors it is that we have upgraded our own creativity. The aim of this paper is to implement and compare systems that are able to transform an image into the style of the artist that we choose and then that style can be transferred to any other image.

One of the first attempts at computational artistry was by a British artist named Harold Cohen in 1973.Cohen hand coded base structures into it so that we were able to

arrive at a form (https://www.youtube.com/watch?v=Oex0eWoU7AQ). It was a tree-like structure and using heuristics the program was able to take encoded rules and generate new combinations of what it knew. The paintings thus generated were displayed at museums across the world. Japanese artist Hiromi Ito too created a program called Aaron (https://www.youtube.com/watch?v=Oex0eWoU7AQ) which created abstract drawings. Later in 2015 Google released DeepDream that trained a convolutional net to classify images, using an optimisation technique enhanced patterns in the input image rather than its own weights based on what it had learnt.

Finally, in 2016 Gatys et al. (2016a) used convolutional neural networks (CNN) to transfer the style of a given painting to any image and then the website called DeepArt was introduced in this respect too. Since then efforts have been made by the AI community to discover artistry using machine learning. Biologically inspired vision models called deep neural networks have demonstrated near-human performance in the field of visual perception, i.e., object and face recognition. Due to the bright prospects that deep learning promises, CNN-based models have been used it to create a platform that gives creative interesting images as a result (Gatys et al., 2015). Agglomerating the effects of biological vision via performance oriented artificial neural networks; these models offer the future of how humans perceive artistic imagery.

2 Background and related works

The concept of Image-style transfer gained much popularity with the Deep Dream project launched by Google although the term was first introduced way back in the 1900s. The applications of artificial intelligence were successfully implemented in a mobile app PRISMA that was released in 2016 and garnered immense popularity. There has been a wide array of researchers who have worked on and published their works related to this field.

Among the various applications of CNN with images as mentioned in Gatys et al. (2015) include object detection, object segmentation, pose estimation, image captioning, dense image captioning, visual question answering, image super-resolution, etc. Generating art is a significant one. It vividly explains that the concept of style transfer is based on feature inversion and texture synthesis. Once a content/base image and a style image is available then new image follows two conditions: matches the CNN features of the content image (feature reconstruction) and matches the Gram matrices of the style image (texture synthesis).

The style transfer can be applied to videos too and it offers a wide variation in terms of mixing two styles and creating a new image or the style/content trade-off. According to Gatys et al. (2015), deep image representations are made of two components style and content representations. Gradient descent on a white noise image is used to find the image information encoded at different layers. The mathematical functions and equations of content loss, style loss and total variation loss were also illustrated in the paper.

For style representations, the concept of Gram matrices has been used. The feature spaces are defined by correlations between filter responses and these feature correlations are given by Gram matrix.

Afterwards, Gatys et al. (2016b) helped to address a potential shortcoming of the original method. The algorithm used earlier transferred the colours of the original painting too that altered the appearance of the scene in undesirable ways. A solution has been devised and simple linear methods for transferring style while preserving colours have been defined. The methods that help in colour preservation are colour histogram matching and luminance-only transfer. The first method holds a general restriction on the colour transfer from content image to style image. In most cases, it has been observed that the transfer has not been very successful. The exact colour palette cannot be matched so it leads to differences between the two images. However, the method 'luminance-only transfer' performs much better as the colours of the content image are preserved completely although there is a loss of dependencies between luminance and colour channels. Another advantage offered by luminance-based method is that it leads to the reduction of the dimensionality of the optimisation problem.

The technique of full style transfer and colour matching gives the output images consisting of strokes similar to blotches of paint and not just variations of light and dark. Both these models can be expanded in different ways and it would be interesting to see in future how these can be explored for more sophisticated colour transfer and adjustment procedures.

Nikulin and Novak (2016) explained the partial and spatial style transfer. It was justified through mathematical functions how the transfer and other related processes like convolution can be applied. Proper implementation of the algorithm would lead to better applications with features like season transfer, exquisite artistic styles, super-resolution and illumination transfer.

Ruder et al. (2016) proposed artistic style transfer for videos. They introduced concepts of multi-pass algorithm and temporal consistency loss. There were a series of techniques for style transfer in videos which includes appropriate initialisation, a loss function to enforce short-term temporal consistency of stylised video, a loss function for long-term consistency, and a multi-pass approach. It was possible to produce stable and visually appealing stylised videos even in the presence of fast motion and strong occlusion. The implementation was based on the torch implementation called neural-style. The energy function was minimised using L-BFGS. For precise evaluation, the strict stopping criterion was used, i.e., the optimisation was considered converged if the loss did not change by more than 0.01% during 50 iterations.

Champandard (2016) introduced a novel concept to augment the generative architectures of CNN's with semantic annotations by stating two options manually authoring pixel labels or using existing solutions for

semantic segmentation. The result gave a content-aware generative algorithm that offered meaningful control over the outcome. The paper concluded that avoiding common glitches enhances the quality of images generated and helps to extend the functional range of these algorithms, i.e., whether for portraits or landscapes, etc. The analysis results were displayed under various categories weight sensitivity, blending segments, semantic map values and content accuracy v/s style quality, etc.

Zhao et al. (2017) presented an automatic image synthesis method to transfer the style of an example image to a content image. While implementing the standard neural style transfer approaches, the textures and colours of the different semantic regions of the style image were often applied inappropriately to the content image completely ignoring its semantic layout and ruining the transfer result. To mitigate such effects a novel method was devised based on automatically segmenting the objects and extracting them soft semantic masks from the style and content images to preserve the structure of the content image while having the style transferred. Each soft mask of the style image represented a specific part of the style image, corresponding to the soft mask of the content image with the same semantics. An augmented deep CNN framework incorporating a generative Markov random field (MRF) took both the soft masks and source images as multichannel input. The benefits of automatic semantic mask extraction by the combination of state-of-the-art methods for both semantic segmentation and facial features were demonstrated. The correctness and accuracy of the semantic masks were critical. Their work concluded with a certain scope to improve semantic segmentation, or to develop methods dedicated to generating soft semantic masks.

Most existing methods either used Gram matrices which treated images globally or for methods based on local patch matching region wise matching per object was done for instance match regions of one object in the style image to regions of a different object in the content image. This proved very critical for human faces as subtle mismatches can be detrimental to the quality of synthesised images. To address this, existing methods used manual segmentation to improve style transfer. However, manual segmentation was time-consuming and laborious. In contrast, the method proposed by them automatically performed a partial soft semantic segmentation of the content and style images.

3 Implementation and comparative analysis

This paper implements various pre-trained models based on CNN available in Keras (https://keras.io/) using Theano as Backend and performs the comparative analysis of these models in the field of image style transfer. Keras is a high-level neural networks API, written in Python and capable of running on top of various backend setups including TensorFlow, CNTK, or Theano. It has been developed with a focus on transfer learning for enabling fast implementation. The models provided in Keras are pre-trained on very large datasets and their learning can easily be transferred to a new given problem using new dataset. The list of image models of Keras includes:

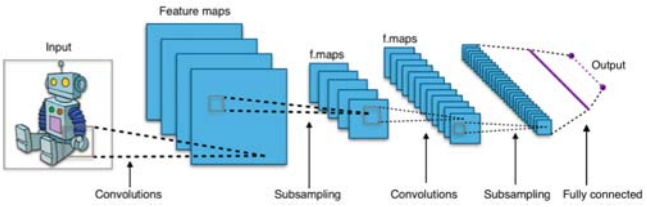
- VGG16
- VGG19
- ResNet50
- InceptionV3
- InceptionResNetV2
- MobileNet
- Xception.

Among these models, MobileNet and Xception are only available for TensorFlow (https://www.tensorflow.org/) backend while the remaining can adapt to both types of the backend. The proposed work is based on pre-trained models of CNN available in Keras library which are discussed as follows.

3.1 Convolutional neural networks

CNN are the most powerful category of deep neural networks that consist of layers of small computational units and process information in the feed-forward manner. Each layer has a collection of image filters each of which extracts a certain feature from the input image. The output of a given layer consists of feature maps that are defined as differently filtered versions of the input image. The biggest advantage of a CNN is that the input image gets transformed into representations that reflect the actual content instead of the pixel values only. Figure 1 shows the basic architecture of CNN.

Figure 1 Architecture of CNN (see online version for colours)



Source: https://en.wikipedia.org/wiki/Convolutional_neural_network

The feature spaces are used to capture texture information by obtaining the details about the style of the image. The feature space is built on top of the filter responses in each layer and consists of correlations between different filter responses over the spatial extent of feature maps. By including the feature correlations of multiple layers a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement is obtained.

3.2 Keras and Theano

Keras is a model-level library, providing high-level building blocks for developing deep learning models but is incapable

of handling itself low-level operations such as tensor products, convolutions, etc. It relies on a specialised, well-optimised tensor manipulation library to do so, serving as the 'backend engine' of Keras. It handles the problem in a modular way, and several different backend engines can be plugged seamlessly into Keras.

Theano is a Python library used to define, optimise, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. The striking features of Theano include tight integration with numpy and transparent use of GPU which increases computational speed over a CPU.

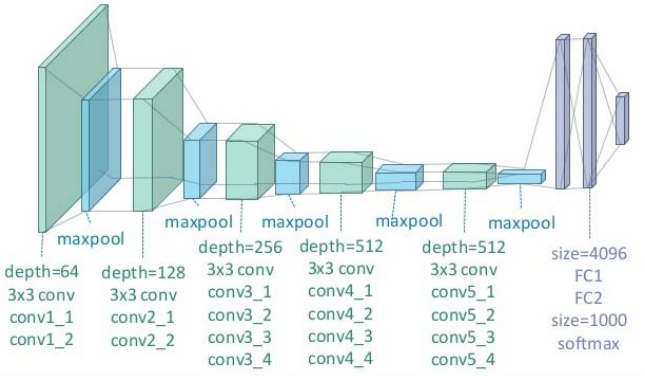
3.3 Pre-trained models (VGG 16 and VGG 19)

VGGNet provides the simplest methods to design network architecture by using only 3 × 3 convolution and 2 × 2 for pooling. VGG Network tells us that the depth of the network plays a very important role. The VGG network is very big and has about 160 M parameters. VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014. To this day is it still considered to be an excellent vision model, although it has been somewhat outperformed by more recent advances such as Inception and ResNet.

Feature space used in the project is provided by the 16 convolutional and five pooling layers of the 19-layer VGG network. Average pooling is preferred over max pooling as it improves gradient flow and it gives more appealing results. Generally, each layer in the network defines a nonlinear filter bank whose complexity increases with the position of the layer in the network. In Figure 2, the architecture of CNN-based VGG19 pre-trained model is depicted.

VGG networks perform much better at style transfer due to their architecture. Image models like AlexNet and GoogLeNet also exist but are not available in Keras. Such models strongly compress the input at the first convolutional layer using large kernels and stride and thus, a lot of fine detail is lost (Nikulin and Novak, 2016). VGG networks use 3 * 3 kernels with stride 1 at all convolutional layers and thus capture much more information.

Figure 2 VGG19 architecture (see online version for colours)



Source: https://www.slideshare.net/ckmarkohchang/applied-deep-learning-1103-convolutional-neural-networks

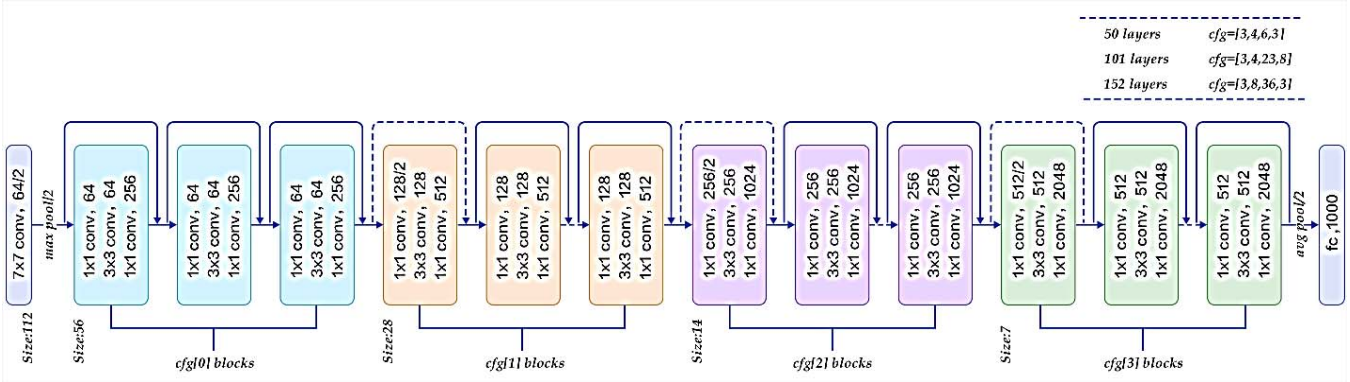
3.4 ResNet50

This deep learning neural network is short for residual network which is based on residual learning. Residual learning is based on the concept of reformulation of the layers as learning residual functions with respect to the layer inputs instead of learning unreferenced functions. The ResNet generally exists with 50 or 101 or 152 layers. The architecture of ResNet pre-trained model is shown in Figure 3.

4 Methodology

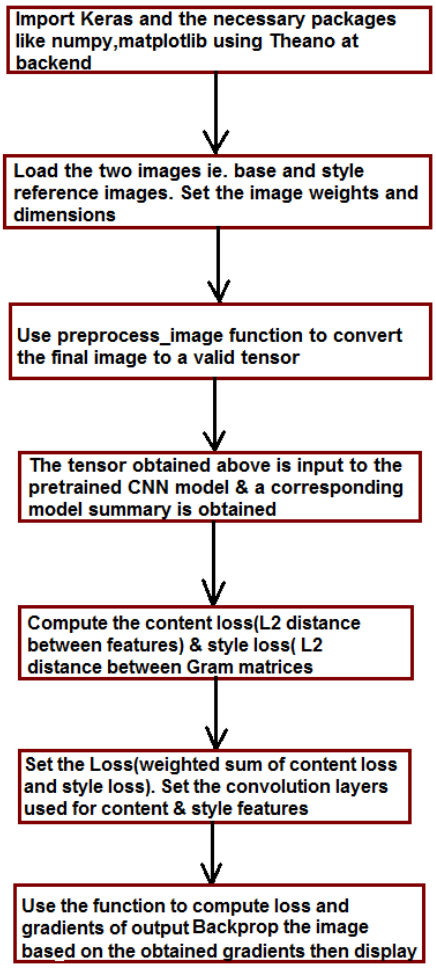
The methodology used for image style transfer using pre-trained models available in Keras is shown in Figure 4.

Figure 3 ResNet architecture (see online version for colours)



Source: http://paddlepaddle.org/docs/develop/book/03.image_classification/index.html

Figure 4 Methodology used (see online version for colours)



The style transfer process has been performed by implementing the Python script in Keras with Theano as backend. As Theano has been used at the backend different image models have been implemented using the same source code. Both the base and reference images have been stored in the same directory as the source code file and loaded from their respective locations. The performance of the given models has been analysed under the same conditions, i.e., only five pooling layers have been considered and the weights have also been kept the same. The models taken for comparison are VGG16, VGG19, ResNet50 and InceptionV3. As illustrated in above diagram the two primary functions used are content loss and style loss. The Gram matrices are used to compute the style loss. p and x vectors are the original image and the image that is generated respectively and P^l and F^l be the feature representations at layer l. The content loss is given by:

$$\Psi_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}\left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

The style loss is given by equation given below where **a** and **x** vectors are the original and generated image, respectively.

$$\Psi_{\text{style}}(\vec{a}, \vec{x}) = \sum_{I=0}^{L} w_l E_l$$

where E_l is given by

$$\frac{1}{4N_l^2 M_l^2}\sum_{i,j}\left(G_{ij}^{l} - A_{ij}^{l}\right)^2$$

The total loss is

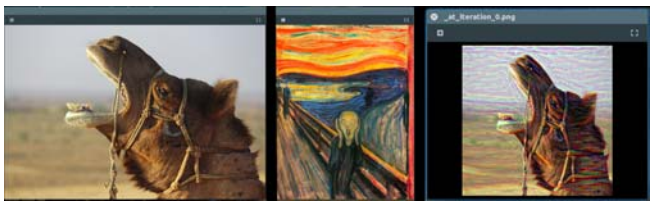
$$\Psi_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha.\Psi_{\text{content}}(\vec{p}, \vec{x}) + \beta.\Psi_{\text{style}}(\vec{a}, \vec{x})$$

In the optimisation step, the pixel values of x are updated to minimise the total loss function. The gradient with respect to the pixel values, i.e., derivative of L total w.r.t. x is computed using standard error back-propagation. Thus, gradient descent algorithm such as L-BFGS, Adam, and SGD can also be applied. Optionally, the total variation loss is added to the loss. This loss works as regulariser for smoothing of the generated image.

5 Results and discussions

The standard art style transfer looks as given in Figure 5. However, using the Keras application on different image models the variation of loss value is obtained that gives different outputs. We observe that as the current loss value increases the clarity and quality of the output enhances. Among the given models the best style transfer has been given by VGG19 with the highest loss value too. When the number of pooling layers is increased from 5 to 12 in case of ResNet50, there is a decrease in the loss value. But the downside of this model is that the images produced are blurred and with each optimisation iteration, the loss function decreases and the blurring exaggerates. The InceptionV3 gives the black screen as output in all cases showing that the weights are not compatible with the model. The L-BFGS algorithm with each iteration minimises the loss function further and gives a better output in case of VGG16 and VGG19. Limited-memory BFGS (L-BFGS or LM-BFGS) is an optimisation algorithm in the family of quasi-Newton methods that approximate the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm using a limited amount of computer memory. It is a popular algorithm for parameter estimation in machine learning. The algorithm's target problem is to minimise f{x} over unconstrained values of the real vector {x} where f is a differentiable scalar function (https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm).

Figure 5 Art style transfer (obtained using VGG19 model) (see online version for colours)



Like the original BFGS, L-BFGS uses estimation to the inverse Hessian matrix to steer its search through variable

space, but where BFGS stores a dense n × n approximation to the inverse Hessian (n being the number of variables in the problem), L-BFGS stores only a few vectors that represent the approximation implicitly. Also, the higher layers capture the exact intense content of objects and their arrangements. On the other hand, the reconstructions from the lower layers just reproduce the exact pixel values of the original image. The feature responses in higher layers are much more detailed as is observed in the project output.

Figure 6 InceptionV3 model output

```
Start of Iteration
Current loss value: 369.4181213378906
Image saved as _at_iteration_0.png
Iteration 0 completed in 47s
```

```
pylab.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f49ffbf6128>
```

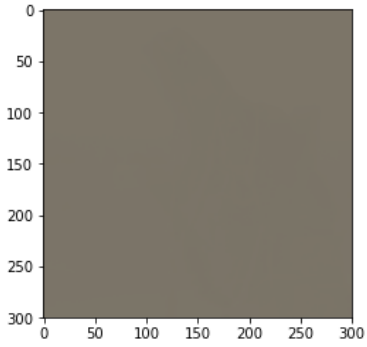
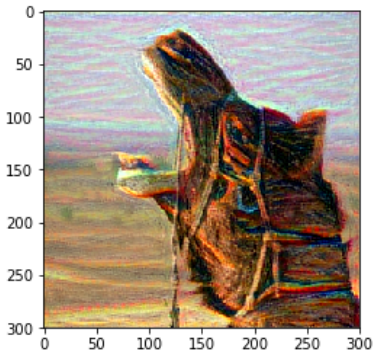


Figure 7 VGG16 model output (see online version for colours)

```
Start of Iteration
Current loss value: 2885143808.0
Image saved as _at_iteration_0.png
Iteration 0 completed in 100s
```

```
pylab.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fa6fb9a6780>
```



There is a general trade-off observed between the number of layers and the computation speed. Although a higher number of layers are preferred to extract maximum style content from the image but it leads to a compromise with the computation time. These results were obtained with the following weight values, i.e., content weight = 0.025 and the style weight = 5. The total variation weight considered was 0.1. The results obtained by using content weight and style weight were noisy and to improve upon this, we used the total variation weight which resulted in a smoothening effect. The motive was to set up an optimisation problem that aims to solve for a combination image that contains the content of the content image while having the style of the style image. Now that we have our input images massaged and our loss function calculators in place, all we have left to do is define gradients of the total loss relative to the combined image, and use these gradients to iteratively improve upon our combination image to minimise the loss.

Figure 8 ResNet50 model output (see online version for colours)

```
Start of Iteration
Current loss value: 43073176.0
Image saved as _at_iteration_0.png
Iteration 0 completed in 64s
```

```
pylab.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fdc8d3300f0>
```

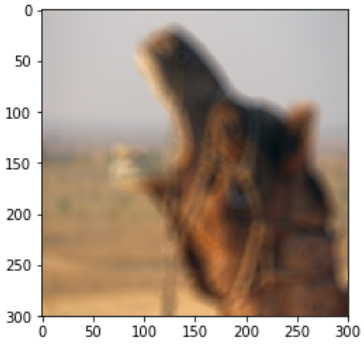


Figure 9 VGG19 model output (see online version for colours)

```
Start of Iteration
Current loss value: 8079787520.0
Image saved as _at_iteration_0.png
Iteration 0 completed in 115s
```

```
pylab.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f0a2da95eb8>
```

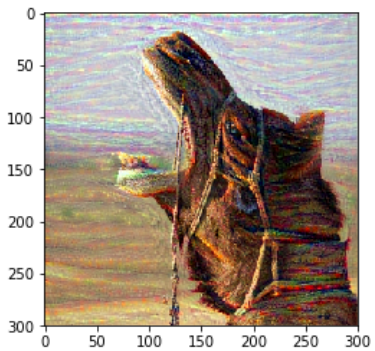


Figure 10 ResNet50: loss function (see online version for colours)

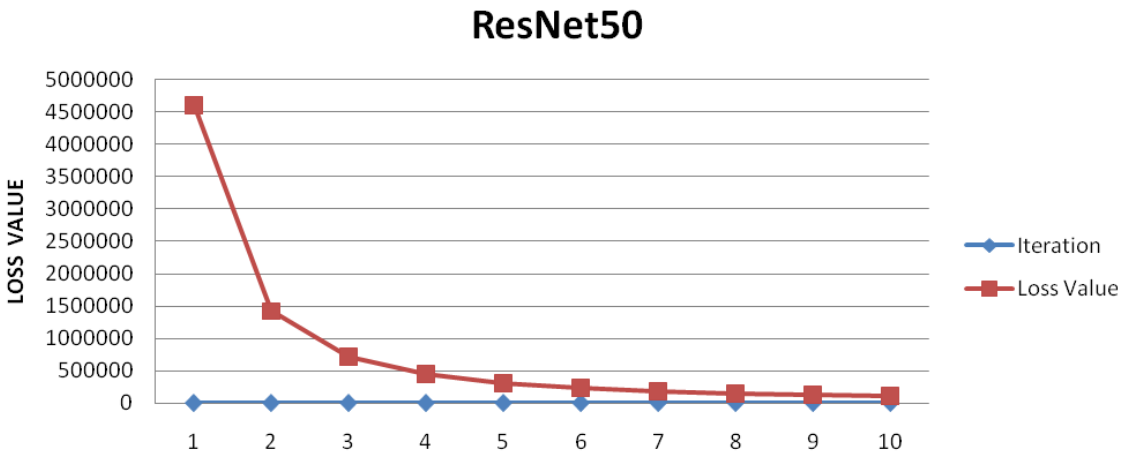


Figure 11 VGG16 loss function (see online version for colours)

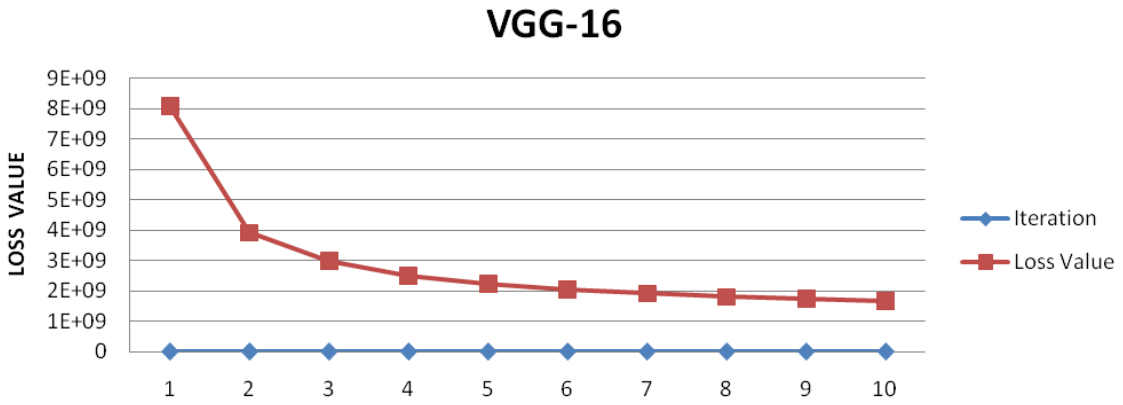
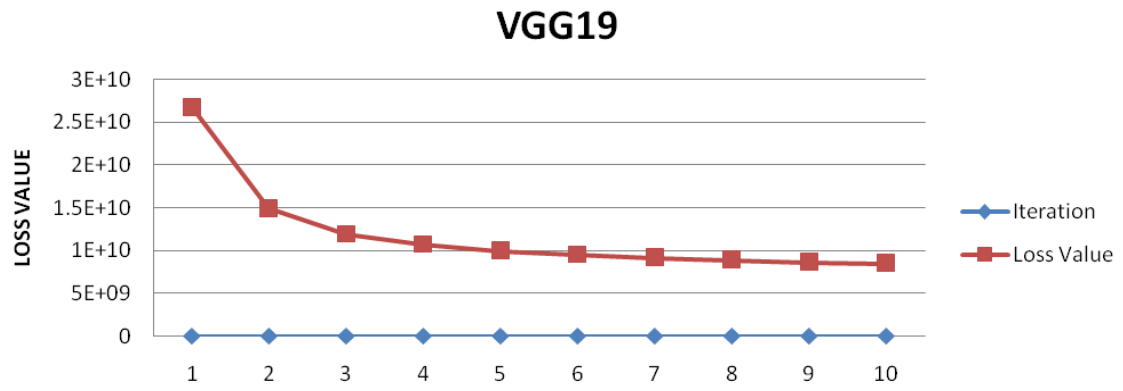


Figure 12 VGG19 loss function (see online version for colours)



The combination images Figures 6, 7, 8 and 9 begin as a random collection of valid pixels; the L-BFGS algorithm (a quasi-Newton algorithm that's significantly quicker to converge than standard gradient descent) is used to iteratively improve upon it. The result after the first iteration of each model is shown in the figures. This is an optimisation process where the loss function gets minimised with each iteration. We stop after ten iterations because we get a satisfactory output and the loss stops reducing significantly. Figures 10, 11 and 12 show the changes in the loss value for ResNet, VGG-16 and VGG-19, respectively, for ten iterations. Although the loss for ResNet is very low as compared to VGG-16 and VGG-19, the layers in ResNet fail to retain the structure of the original content image.

VGG-19 starts with a lesser loss value as compared to VGG-16, but as we proceed, there is a noticeable trend: the loss value curve starts to be similar for both the models. As a matter of fact, there is not much difference in VGG-16 and VGG-19. We just gain a bit of speed in VGG-16 due to the lesser number of layers. The art style transfer obtained using VGG19 model is shown in Figure 5.

6 Conclusions and future scope

In this paper, we have used various pre-trained models based on CNN using Keras for image style transfer problem using transfer learning. These models mainly InceptionV3, Resnet50, VGG16 and VGG19 have been compared and analysed for image style transfer work. Out of these, it has been found that InceptionV3 failed in the image style transfer and ResNet50 is also not suitable for the task at hand even though these models have got a higher number of layers than that of other models used in the study. Further, it has been observed that VGG16 and VGG19 are suitable for image style transfer task. Further, VGG19 performs better than VGG16 for image style transfer task which can be due to fact that both share similar architecture and VGG19 is deeper than VGG16. However, it has also been found that VGG19 is much slower than the VGG16 model in image style transfer task. In the future perspective, this work can be extended by using wide residual networks based on CNN for image style transfer task.

References

Applied Deep Learning 11/03 Convolutional Neural Networks [online] https://www.slideshare.net/ckmarkohchang/applied-deep-learning-1103-convolutional-neural-networks (accessed 25 Decembert 2017).

Champandard, A.J. (2016) *Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks*, arXiv preprint arXiv: 1603 [online] (accessed 22 November 2017).

Champandard, A.J. (2016) *Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks*, arXiv preprint arXiv: 1603.01768.

Convolutional Neural Network [online] https://en.wikipedia.org/wiki/Convolutional_neural_network (accessed 25 December 2017).

Gatys, L.A., Bethge, M., Hertzmann, A. and Shechtman, E. (2016a) *Preserving Colour in Neural Artistic Style Transfer*, arXiv preprint arXiv: 1606.05897.

Gatys, L.A., Ecker, A.S. and Bethge, M. (2016b) 'Image style transfer using convolutional neural networks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.2414–2423.

Gatys, L.A., Ecker, A.S. and Bethge, M. (2015) *A Neural Algorithm of Artistic Style*, arXiv preprint arXiv: 1508.06576.

How To Generate Art-Intro To Deep Learning [online] https://www.youtube.com/watch?v=Oex0eWoU7AQ (accessed 10 November 2017).

Image Classification [online] http://paddlepaddle.org/docs/develop/book/03.image_classification/index.html (accessed 20 December 2017).

Nikulin, Y. and Novak, R. (2016) *Exploring the Neural Algorithm of Artistic Style*, arXiv preprint arXiv: 1602.07188 [online] (accessed 30 November 2017).

Ruder, M., Dosovitskiy, A. and Brox, T. (2016) 'Artistic style transfer for videos', in *German Conference on Pattern Recognition*, September, pp.26–36.

Zhao, H., Rosin, P.L. and Lai, Y.K. (2017) *Automatic Semantic Style Transfer using Deep Convolutional Neural Networks and Soft Masks*, arXiv preprint arXiv: 1708.09641.