Neural Style Transfer in Images using Deep Learning

T.H Feiroz Khan[1], Ameer Naseh[2], V. Naga Sumanth[3] , Rohan Hazra[4]

[1]Assistant Professor, [2]Students, [3]Students, [4]Students

Department of CSE ,SRMIST, Chennai ,India

Abstract*—Style Transfer aims to transfer the style of one or more image to the base image using Machine Learning. This gives users more editing powers. Neural style transfer is the process of re-imagining one image in the style of other. It is one of the coolest applications of image processing using convolution neural networks. Recent advances made in Neural Networks and Image processing has made techniques like Neural Style Transfer possible. Neural style transfer is different from many image filters that is seen on Instagram or other camera apps. Unlike most image filters modifying or enhancing an image by enhancing certain features or removing other features such as smoothing, sharpening edges, etc., Style transfer synthesis a new image from two source images called content image and style image.

Keywords: Style, Image, Editing, Network, Filters.

I INTRODUCTION

Exchanging the style from one picture onto another can be viewed as an issue of surface exchange. In surface transfer, the objective is to orchestrate a surface from a source picture while obliging the surface amalgamation keeping in mind the end goal to protect the semantic substance of an objective picture. For surface amalgamation here exist an expansive scope of intense non-parametric calculations that can integrate photorealistic normal surfaces by resampling the pixels of a given source surface. Most past surface exchange calculations depend on these non-parametric techniques for surface combination while utilizing diverse approaches to save the structure of the objective picture. For example, Efros and Freeman present a correspondence outline incorporates highlights of the objective picture, for example, picture force to compel the surface combination technique. Hertzman et al. utilize picture analogies to exchange the surface from an effectively stylised picture onto an objective picture. Ashikhmin centers around exchanging the high- recurrence surface data while saving the coarse size of the objective picture. Lee et al. enhance this calculation by also educating the surface exchange with edge introduction data.

In spite of the fact that these calculations accomplish amazing outcomes, they all experience the ill effects of a similar basic restriction: they utilize just low-level picture highlights of the objective picture to in-frame the surface exchange. In a perfect world, be that as it may, a style exchange calculation ought to have the capacity to remove the semantic picture content from the objective picture (e.g. the items and the general landscape) and afterward advise a surface exchange system to render the semantic substance of the objective picture in the style of the source picture. Subsequently, a major essential is to discover picture portrayals that autonomously display varieties in the semantic picture content and the style in which it is exhibited. Such factorized portrayals were beforehand accomplished just for controlled subsets of common pictures, for example, faces under various enlightenment conditions and characters in various text style styles or transcribed digits and house numbers. To for the most part, isolate content from style in normal pictures is as yet a to a great degree troublesome issue. Nonetheless, the re-penny advance of Deep Convolutional Neural Networks has created great PC vision frameworks that figure out how to extricate abnormal state semantic data from regular pictures

II PROPOSED METHOD

Both visual texture synthesis, whose goal is to synthesize a natural looking texture image from a given sample, and visual texture transfer, which aims at weaving the reference texture within a target photo, have been long studied in computer vision. Recently, within the era of convolutional neural networks (CNNs), these subjects have been revisited with great success. In their seminal work, Gatys exploit the deep features provided by a pre-trained object recognition CNN to transfer by optimization the “style” (mainly texture at various scales and color palette) of a painting to a photograph whose layout and structure are preserved [3]. The result is a pleasing painterly depiction of the original scene. This is a form of example-based non-photorealistic rendering, which is now available in many social applications. Since then, the work of Gatys et al [1]. has sparked a great deal of research, see recent review on neural style transfer. Moving toward audio, sound texture can be understood as the global temporal homogeneity of acoustic events, while the sound color palette may be seen as a set of most representative spectral shapes.

Note that model-based natural sound texture synthesis has also been extensively studied in the literature. Closer to audio style transfer, a cross-synthesis method that hybridizes two sounds by simply multiplying the short -term spectrum of one sound by a short term spectral envelop of another sound was proposed by Serra. More recently, Collins and Sturm proposed a dictionary-based approach for sound morphing, where sound textures can be represented by the dictionary’s atoms [2]. To date, thanks to the power of CNNs, data-driven approaches have gained a lot in performance allowing high quality synthesis of complicated audio such as speech and music. Beyond texture synthesis and in analogy to image style transfer, some attention has been recently given to audio style transfer, notably in the form of exploratory work presented in blog posts. These initial investigations showed that the original algorithm proposed in for image is not fully adapted to audio signals. The result sounds often as a mixture of the style and content sounds rather than as a stylized content. Another attempt was done to apply style transfer for prosodic speech. The authors concluded that their approach, while allowing the transfer of low-level textural features, had a difficulty in transferring high-level prosody such as emotion or accent

A. Neural algorithm for artistic Style

This paper based on artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images**.**

B. Deep Photo Style Transfer

Style transfer is a magical algorithm where we have one photograph with content and one with a interesting style and the output is a third image with 'these two photo fused together. This is achieved machine learning technique that we call a convolution neural network [2]. The more layer this network contains, The more powerful they are and the more capable they are in building an intuitive understanding of an

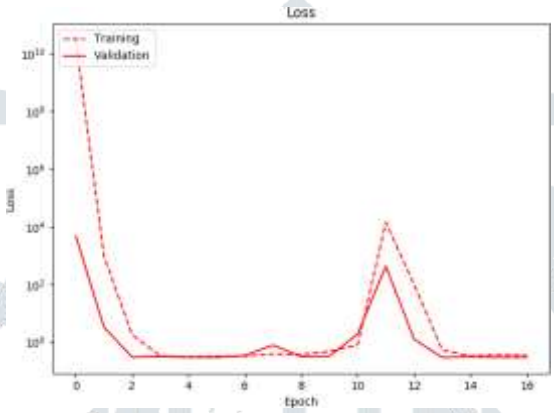

Fig. 1. Loss represents the loss of the content image and style image

C.Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram LossesNeural style transfer is an incredible technique where we have to input two photographs and output would be the combination of these two namely the content of one and artistic style of another fused together[5].

The paper contains a nice formal explanation of the weak points of the existing style transfer algorithms. The intuition behind the explanation is that the neural networks think in terms of neuron activations, which may not be proportional to the colour intensities in the source image styles, therefore their behaviour often becomes inconsistent or different than expected. The authors propose thinking in terms of histograms, which means the output image should rely on statistical similarities with the source images. And as we can see, the results look outstanding even when compared to the original method.

D. Universal Style Transfer via Feature Transforms The main goal is to transfer visual styles to content images. We take a photo with some content, and for example, a painting with the desired style, and the output is an image where the style is applied to our content[8]. If this is done well and with good taste, it really looks like magic. However, for pretty much all of the previous techniques, there are always some mysterious styles that results in failure cases. And the reason for this is the fact that these techniques are retained on a set of style images, and if they face a style that is wildly different from these training images, the results won't be very usable. This new algorithm is also based on neural networks, it doesn't need to be trained on these style images, but it can perform high-quality style transfer, and it works on arbitrary styles.

E. Image Style Transfer Using Convolutional Neural Networks

This paper introduces a deep-learning approach to photographic style transfer that handles a large variety of image content while faithfully transferring the reference style. Our approach builds upon the recent work on painterly transfer that separates style from the content of an image by considering different layers of a neural network

F. Summary of Proposed System

We consider image transformation problems, where an input image is transformed into an output image. Recent methods for such problems typically train feedforward convolutional neural networks using a per-pixel loss between the output and ground-truth images. Parallel work has shown that high- quality images can be generated by defining and optimizing perceptual loss functions based on high-level features extracted from pretrained networks [5]. We combine the benefits of both approaches, and propose the use of perceptual loss functions for training feedforward networks for image transformation tasks. We show results on image style transfer, where a feed-forward network is trained to solve the optimization problem proposed by Gatys in real- time. Compared to the optimization-based method, our network gives similar qualitative results but is three orders of magnitude faster

I. SYSTEM DESIGN

Rendering the semantic content of an image in different styles is a difficult image processing task. Arguably, a major limiting factor for previous approaches has been the lack of image representations that explicitly represent semantic information and, thus, allow to separate image content from style. Here we use image representations derived from Convolutional Neural Networks optimised for object recognition, which make high level image information explicit. We introduce A Neural Algorithm of Artistic Style that can separate and recombine the image content and style of natural images [7]. The algorithm allows us to produce new images of high perceptual quality that combine the content of an arbitrary photograph with the appearance of numerous well- known artworks. Our results provide new insights into the deep image

representations learned by Convolutional Neural Networks and demonstrate their potential for high level image synthesis and manipulation.

The results presented below were generated on the basis of the VGG network which was trained to perform object recognition and and is described in the original work . We used the feature space provided by a version of the 16 and 5 pooling layers of the 19-layer VGG network. We configure the network by scaling the weights such that the mean activation of each filter over images and positions is equal to one. Such re-scaling can be done for the VGG network without changing its output, because it contains only rectifying linear activation functions and no normalization or pooling over feature maps. We do not use any of the fully connected layers [3]. The model is publicly available and can be explored . For image synthesis we found that replacing the maximum pooling operation by average pooling yields slightly more appealing results, which is why the images shown were generated with average pooling.

A. Convolutional Neural Network Convolutional Neural Networks (CNNs) are a category of Neural Network that have proven very effective in areas such as image recognition and classification. CNNs have been successful in computer vision related problems like identifying faces, objects and traffic signs apart from
powering vision in robots and self-driving cars [11]. CNN is shown to be able to well replicate and optimize these key steps in a unified framework and learn hierarchical representations directly from raw images [7]. If we take a convolutional neural network that has already been trained to recognize objects within images then that network will have developed some internal independent representations of the content and style contained within a given image.

Convolutional neural networks are deep artificial neural networks that are used primarily to classify images (e.g. name what they see), cluster them by similarity (photo search), and perform object recognition within scenes. They are algorithms that can identify faces, individuals, street signs, tumors, platypuses and many other aspects of visual data.Convolutional networks perform optical character recognition (OCR) to digitize text and make natural- language processing possible on analog and hand-written documents, where the images are symbols to be transcribed. CNNs can also be applied to sound when it is represented visually as a spectrogram. More recently, convolutional networks have been applied directly to text analytics as well as graph data with graph convolutional networks.The efficacy of convolutional nets (ConvNets or CNNs) in image recognition is one of the main reasons why the world has woken up to the efficacy of deep learning. They are powering major advances in computer vision (CV), which has obvious applications for self-driving cars, robotics, drones, security, medical diagnoses, and treatments for the visually impaired.

B. Content representation and loss
We can construct images whose feature maps at a chosen convolution layer match the corresponding feature maps of a given content image. We expect the two images to contain the same content—but not necessarily the same texture and style.Given a chosen content layer l, the content loss is defined as the Mean Squared Error between the feature map F of our content image C and the feature map P of our generated image Y.

$$\mathcal{L}_{content} = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2$$

(1)

When this content-loss is minimized, it means that the mixed-image has feature activation in the given layers that are very similar to the activation of the content-image. Depending on which layers we select, this should transfer the contours from the content image to the mixed-image.

C. Style representation and loss
We will do something similar for the style-layers, but now
we want to measure which features in the style-layers activate simultaneously for the style-image, and then copy this activation-pattern to the mixed-image.One way of doing this, is to calculate the Gram-matrix(a matrix comprising of correlated features) for the tensors output by the style-layers [9]. The Grammatrix is essentially just a matrix of dot- products for the vectors of the feature activations of a style- layer.If an entry in the Gram-matrix has a value close to zero then it means the two features in the given layer do not activate simultaneously for the given style-image. And vice versa, if an entry in the Gram-matrix has a large value, then it means the two features do activate simultaneously for the given style-image [7]. We will then try and create a mixed- image that replicates this activation pattern of the style- image.If the feature map is a matrix F, then each entry in the Gram matrix G can be given by:

(2)

$$G_{ij} = \sum_{k} F_{ik}F_{jk}$$

The loss function for style is quite similar to out content loss, except that we calculate the Mean Squared Error for the Gram-matrices instead of the raw tensor-outputs from the layers.

$$\mathcal{L}_{style} = \frac{1}{2}\sum_{l=0}^{L}(G_{ij}^{l} - A_{ij}^{l})^2$$

(3)

As with the content representation, if we had two images whose feature maps at a given layer produced the same Gram matrix we would expect

both images to have the same style, but not necessarily the same content. Applying this to early layers in the network would capture some of the finer textures contained within the image whereas applying this to deeper layers would capture more higher-level elements of the image's style.

D. Optimizing loss function and styling the image Using a pre-trained neural network such as VGG-19, an input image (i.e. an image which provides the content), a style image (a painting with strong style elements) and a random image (output image), one could minimize the losses in the network such that the style loss (loss between the output image style and style of 'style image'), content loss (loss between the content image and the output image) and the total variation loss (which ensured pixel wise smoothness) were at a minimum. In such cases, the output image generated from such a network, resembled the input image and had the stylist attributes of the style image.The total loss can then be written as a weighted sum of the both the style and content losses.

(4)

we will minimize our total loss by Adam optimizer. As our loss go down we will go close to our goal of producing a style transfer image Y.

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

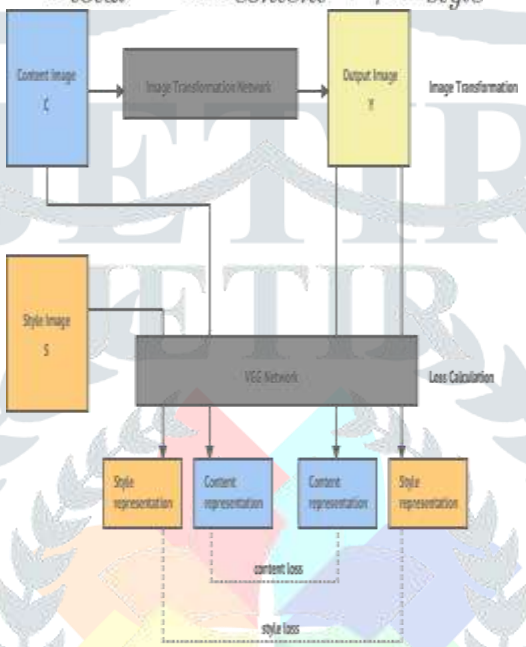

Fig 2. Convolutional Work flow

II. SYSTEM IMPLEMENTATION

We used the instance on Floyd-hub. We used it because we want to use GPU and it is the cheapest option that offers a GPU. We choose PyTorch as our deep learning framework because is enables us to use Python and it generates computational graph dynamically, making it easier to debug. We need to upload data using FileZilla. After finishing the project, we think that implementing published research papers from scratch is very tricky as research papers are not written in a way that is very friendly for people who are relatively new to deep learning to replicate. However, we are able to understand deep learning a lot better by understanding published research papers enough to implement it. In addition, by doing this project, we understand PyTorch a lot better [6].

For implementing this algorithm, it takes very little time to upload data . However, the implementation of this algorithm took us the longest since we don't understand the neural style transfer algorithm and we don't know PyTorch very well. PyTorch is still a relatively new framework and the documentation sometimes is not as comprehensive as we would like them to be. There are two points in the paper that confused us in the beginning. The first one is that if we are calculating the style and content loss only using a subset of layers in VGG19 net, do we need to pass the images through every layer of the VGG19 Net? After a more detailed reading of the paper and watching a talk given by the author on Youtube, we decide that we should pass the images through all layers of VGG19 net.

We also tried only passing the images through a subset of VGG layers(we tried using VGG19 net up to the 8th Convolutional layer), we did not obtain good results at all(many results resemble neither the content nor the style image). We are able to obtain results visually similar to the published paper by using all layers of the VGG19 net. At the beginning, we were also confused regarding how to optimize the initialized image. We were confused because we have only seen cases where we optimize a set of parameters in some neural network before. By understanding this, we gained a better understanding of how optimizer works in deep learning [15].

The first step in implementation is to get the images into the formats required by PyTorch. We did so by writing an ImageDataSet class, that turns the images in a certain folder into numpy arrays. We then.transforms class that turn the numpy arrays that represent images into Tensors, the datatype required by PyTorch framework. We then used Dataloader that enables us to iterate through the set of images we obtained, making it possible to train. The second step is to implement the process of training(optimizing the initialized image). We encountered some major challenges in this step. The first challenge is the implementation of Gram Matrix. We were very sure that our implementation is right. However, using my implementation, the style loss

decreases during the first one or two iterations, and then the style loss stays at a very high level [12]. The optimizer will optimize the content loss to 0. As a result, the output is always an image that is almost identical to the content image and has almost no resemblance to the style of style image. We spent two days trying to debug and had no progress.

The input parameter in the forward function of GramMatrix is a Variable(which is just a wrapper that contains the tensor). Their implementation kept it as a Variable, but our implementation pulled the tensor out of the Variable. We tested the Gram Matrix function independent of the rest of our system [3]. Our implementation and their implementation gave the same numeric results. However, if we use our implementation in the overall system, the output image is almost identical to content image(do not optimize style loss). If we use their implementation, we get very good results. We tried very hard to figure out why this is happening, but we still don't understand why We would really appreciate some likely explanations since this will help us if we were to do any projects involving deep learning in the future [12].

We then proceed to implement the training algorithm. One tricky thing is that we have to be aware of in-place operation in PyTorch.. If we overwrite some variables that are needed for gradient computations later via some in-place operations, we would mess up the computational graph. We had to fix a few bugs that are related to this issue during our implementation. Another tricky part during this implementation is to make PyTorch use GPU. PyTorch code uses CPU by default, and PyTorch's official tutorial for GPU is less than comprehensive. CPU is too slow for our purpose. We had to spend a while to figure this out by looking at Youtube tutorials and reading PyTorch Forums. It basically involves moving the input and the network in and out of GPU at the correct time. The final quirky part we found about PyTorch is that we had to normalize the pixel values of our input images to [0,1]. Otherwise, we get white noise or blank images as output.

III.RESULTS

Below we present our results for single-style transfer using different methods of up sampling, convolution transposes and nearest neighbor up samlping. Additionally, we use the same picture of Tubingen as our content image in these results unless otherwise noted.

Fig-3. Convolutional Image

Fig-4. Artistic Style image

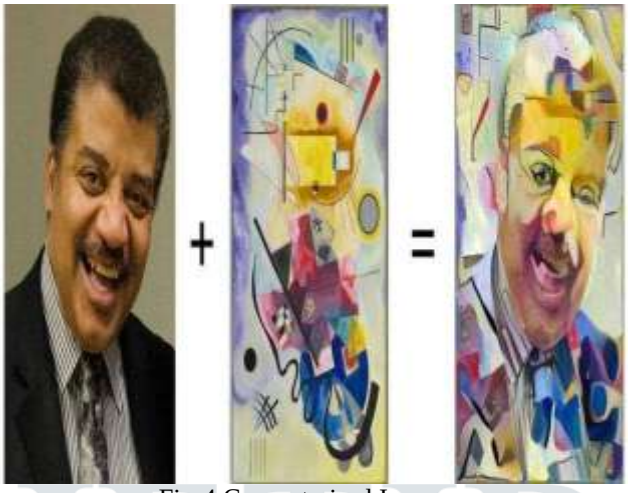

Fig-4 Computerised Image

After training the single-style transfer network, we can perform style transfer with one fast forward pass taking less than a second. We note the following results:

• Much faster than naive style transfer which iterates on the image

• Qualitatively the images are equally convincing, and quantitatively (based on values for style/feature loss) comparable in performance

Disadvantages: training time is much longer, and each model is restricted to only a limited set of styles.

IV. CONCLUSION AND FUTURE WORKS

While our algorithm for fast style transfer is sufficiently general and can handle any variety of input images, it is still limited to a fixed selection of style images to be trained on, followed by a very long training process of a few hours. We can certainly stop the algorithm earlier, but doing so leads to noticeably weaker results since the network wasn't able to run as many epochs and receive its fullest training. It is also restricted to a predetermined set of parameter weights on content and style losses, so we aren't able to alter these afterwards to see how the algorithm can transform the image to be highly stylized / more abstract on one end versus less styled / more realistic on another.

REFERENCES

[1] N. Ashikhmin. Fast texture transfer. IEEE Computer Graphics and Applications, 23(4):38–43, July 2003.

[2] M. Berning, K. M.Boergens, and M. Helmstaedter. SegEM: Efficient Image Analysis for High-Resolution Connectomics. Neuron, 87(6):1193–1206, Sept. 2015.

[3] C. F. Cadieu, H. Hong, D. L. K. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo. Deep Neural Networks Rival the Representation of Primate IT Cor
2421 tex for Core Visual Object Recognition. PLoS Comput Biol, 10(12):e1003963, Dec. 2014.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. arXiv:1412.7062 [cs], Dec. 2014. arXiv: 1412.7062.

[5] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3828–3836, 2015.

[6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. arXiv:1310.1531 [cs], Oct. 2013. arXiv: 1310.1531.

[7] A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, volume 2, pages 1033–1038. IEEE, 1999.

[8] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 341–346. ACM, 2001.

[9] D. Eigen and R. Fergus. Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture. pages 2650–2658, 2015.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture Synthesis Using Convolutional Neural Networks. In Advances in Neural Information Processing Systems 28, 2015.