CSCE 5214 - SOFTWARE DEVELOPMENT FOR ARTIFICIAL INTELLIGENCE

Project 2 Report HW9 Team 12

NEURAL STYLE TRANSFER USING CONVOLUTIONAL NEURAL NETWORKS

Team Members : Sai Kaushik Surampudi
Saketh Gondela
Manvitha Kancharla
Pradyumna Poralla
Deepthi Padmasree Gollamudi

ABSTRACT

One of the most popular and widely used methods of neural networks that is now in style in the AI world is neural net image styling. This algorithm is used by all of the top social media sites to develop their applications, which allow users to apply different filters to their images and customize their picture styling. The exponential expansion of neural networks and its application to several real-world AI issues are clearly seen as a result of numerous breakthroughs in the AI sector. We would be dealing with the job of neural picture styling in this project. The basic objective is to combine a style[1] picture with content images to generate a style image. In order to do this, we employ the VGG CNN method. The two most common forms are both artistic and photorealistic. In this assignment, we would be transferring artistic style. For the normal content photos and the style images from Kaggle or Google, we intended to utilize the COCO image dataset from Kaggle. For this project, we also intended to create a front-end API that would allow users to contribute content and style images and view the results. During the project update phase of the coursework, we must examine, discuss, and provide an update on the model implementation in HW8. The process of neuronal style transmission is shown in the image below.

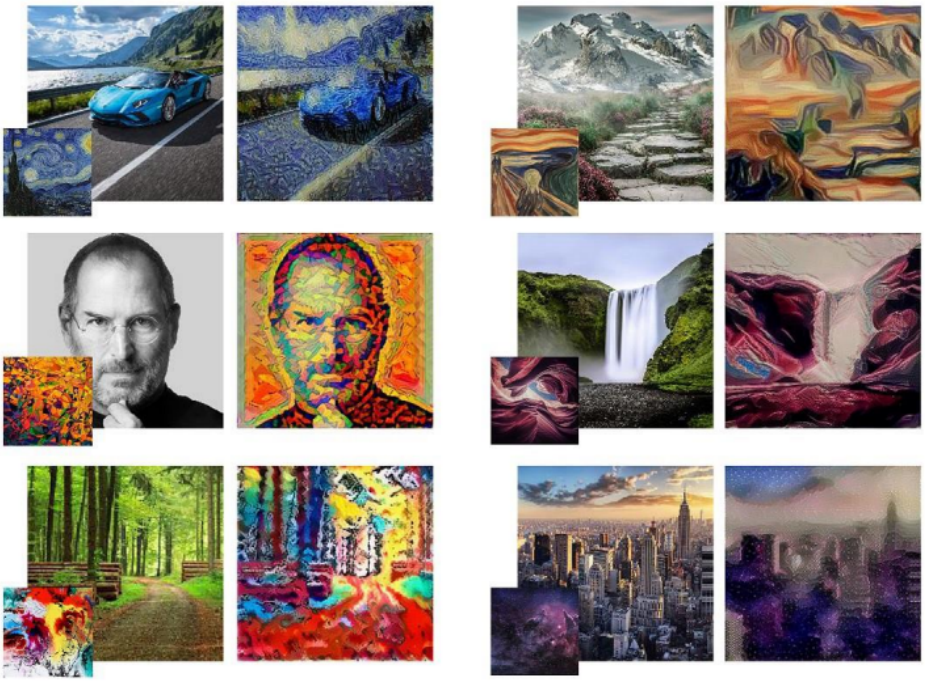


Fig. 1. Neural Style Transfer

INTRODUCTION

One of the most difficult problems in today's computational world is to transfer the style of one image to another image. In order to achieve automatic style transfer, many efforts have been made in recent years. In 2001, Hertzmann et al, Efros and Freeman proposed a work which is a semantic based approach for style transfer. In 2016, Gatys captured the style of artistic images and used it to transfer to other images using deep learning networks like CNN's. In this work, the problem has been dealt with by finding an image that can be used to match the content and style characteristics based on activation functions of each layer in the neural network. In 2016, Johnson, Ulyanov and Ruder formulated more impressive approaches. This work of these guys drew more attention in the artificial intelligence[2] community. The gram matrix in these approaches are not fully explained. How the gram matrix represents the artistic style of the image is still a project under development. In this project we will be dealing with neural style transfer in which we have two images - one content image and one style image. The main aim of the project is to copy the style of one image and apply this style on the content image. In this project applying style means by mimicking the patterns, brushstrokes and other features of an image. In this model, we are planning to use the VGG-16 pretrained model. In this model, we will be planning to demonstrate the process of neural style transfer and matching the gram matrices. Style transfer is one of the major applications of this project and in style transfer which is one of the active topic in the industry. In usual traditional processes we deal with the patch texture transfer and synthesis by which we tend to resample the pixel values of an image using the original image pixel values. In most of the earlier applications, Markov Random Field (MRV) is used in order to retain the entire global structures and the only limitation is that it completely utilizes the potent representation provided by Deep Convolutional Neural Networks (CNN). This technique represented the creative style of a picture using Gram vectors of the neural activation functions from various levels of a CNN. Then, it matched the neural activations with the content image as well as the Gram matrices with the style image to create a new image from white noise using an iterative optimization technique.on in using such methods would be that it suffers from low texture transfer. **Domain Adaptation** is the process which uses the process of transfer learning and using this transfer learning to model the target domain. In order to minimize the

difference of the source and target, we need to select the MMD which is the Maximum Mean Discrepancy - a factor that is used to calculate the gap between the sample mean. Maximum Mean Discrepancy is used by taking two different samples and defining the samples by calculating the mean square difference between the samples.

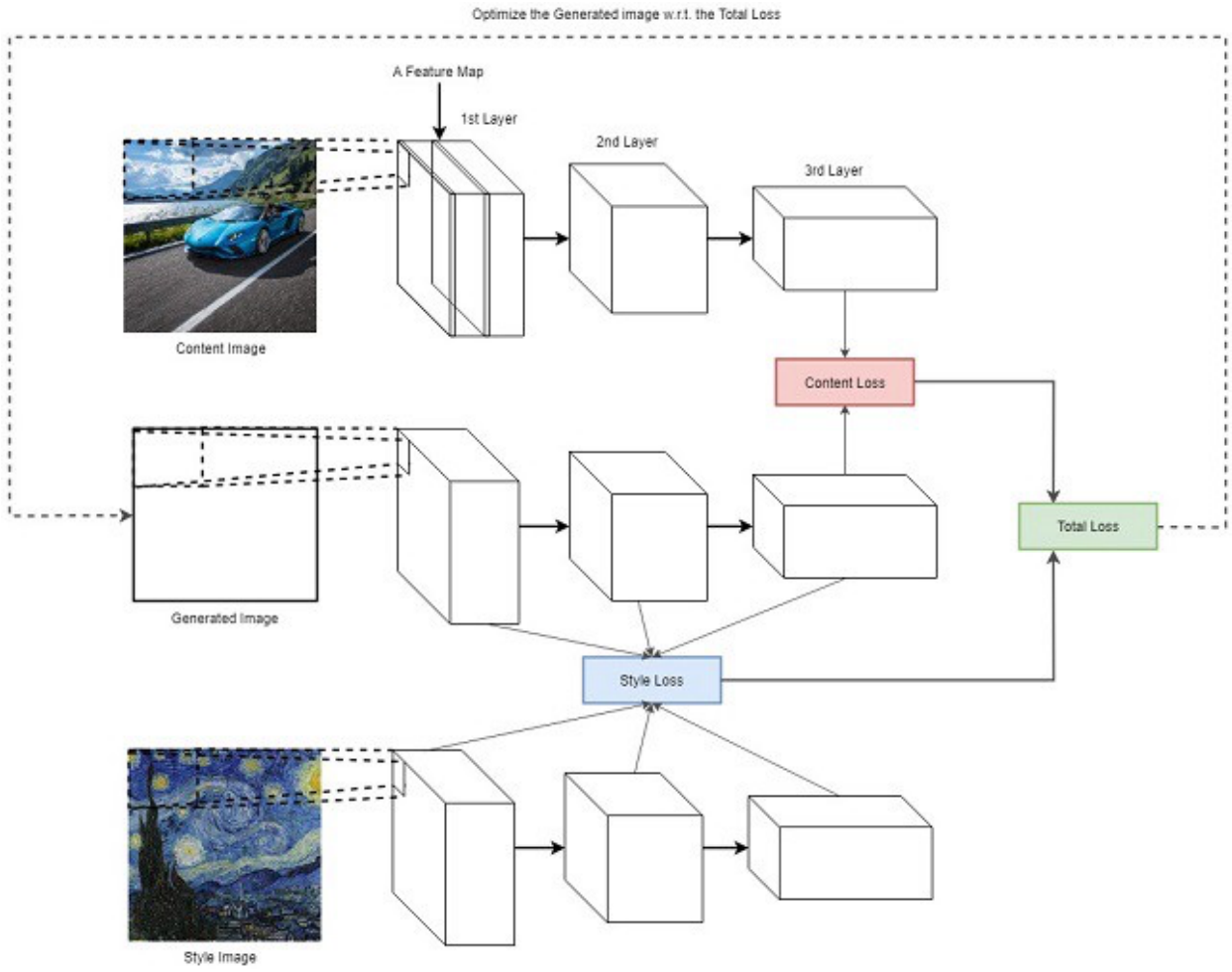


Fig. 2. The background process of neural style transfer

ARCHITECTURE

Fig 2 shows the basic architecture of how neural style transfer actually works. As we can see in the figure, we can see the content image is taken and a feature map[3] is taken from the image which can be designed by extracting the features of the image. This feature map is present in the first layer followed by the other two layers. From the third layer, we can calculate the content image loss. By using the style image, we can extract the features similarly by using the feature extraction techniques of the content image. The style loss is calculated by using all the three layers and the total loss is calculated by adding the content loss and the style loss together. From this total loss, we then get a generated target image which we can use to cross verify our loss and can be used to define the basic structure of the entire NST process.

VGG - 19

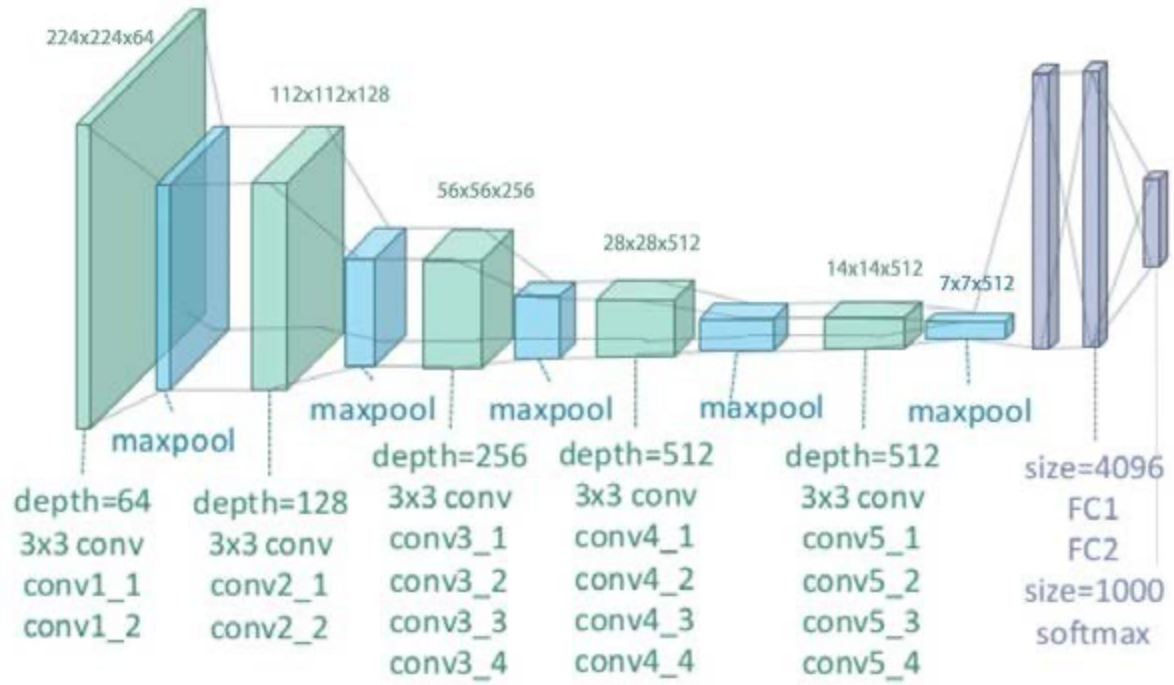


Fig. 3. VGG 19 Architecture

VGG19 was demonstrated to be the model with the best performance out of all the settings on the ImageNet dataset. Let's look at the actual architecture of this arrangement. Any network configuration may be thought of as having a fixed 224 by 224 image with R, G, and B channels as its input. The only pre-processing done is to normalize the RGB values of each pixel. To do this, the mean value is removed from each pixel. The picture is transmitted through the first stack of two convolution layers after ReLU [4] activations, with a minuscule 3 3 receptive area. Each of these two layers has 64 filters. The convolution stride is fixed at 1 pixel, whereas the padding is 1 pixel. In this configuration, the spatial resolution is preserved, and the dimensions of the output activation map coincide with those of the input image. Then, spatial max pooling is applied to the activation maps using a 2 pixel stride over a 2 x 2 pixel frame. As a result, the activations' size is reduced by half. At the bottom of the first stack, there are activations that are 112 112 x 64 in size. The second stack, which is identical to the first but contains 128 filters as opposed to 64 in the first stack, is then applied to the activations. As a consequence, the size is 56 x 56 x 128 after the second layer. The third stack, which consists of three convolutional layers and a max pool layer, is subsequently applied. Due to the 256 filters that were used in this instance, the output size of the stack is 28 x 28 x 256. Then, two stacks of three convolutional layers are built, each with 512 filters. The final outputs of both of these stacks will be 7 7 x 512.

The three fully connected layers that come after the convolutional layer stacks are separated by a flattening layer. One thousand neurons make up the output layer, the last fully connected layer, which represents the 1,000 potential classes in the ImageNet dataset. 4,096 neurons are found in each of the first two layers. Following the output layer is the Softmax activation layer, which is used for categorical categorization.

MODELLING

Dataset:

The dataset that we used for the project is taken from Kaggle. The link to the dataset is here. The dataset is a collection of best artworks done by 50 of the most influential artists of all time. In this dataset, we have three main files - artist.csv file which contains information about the artists and their works. This excel sheet basically gives entire information with regards to the information of each artist. We also have two main folders - image.zip and resized.zip. These two folders contain the images that are necessary in order to complete our project. The image.zip has all the images of all the 50 artists and the resized folder has the same collection of images but in this the images are resized and sequentially numbered. The figure below shows a visualization of the dataset excel file and also the images in the dataset.

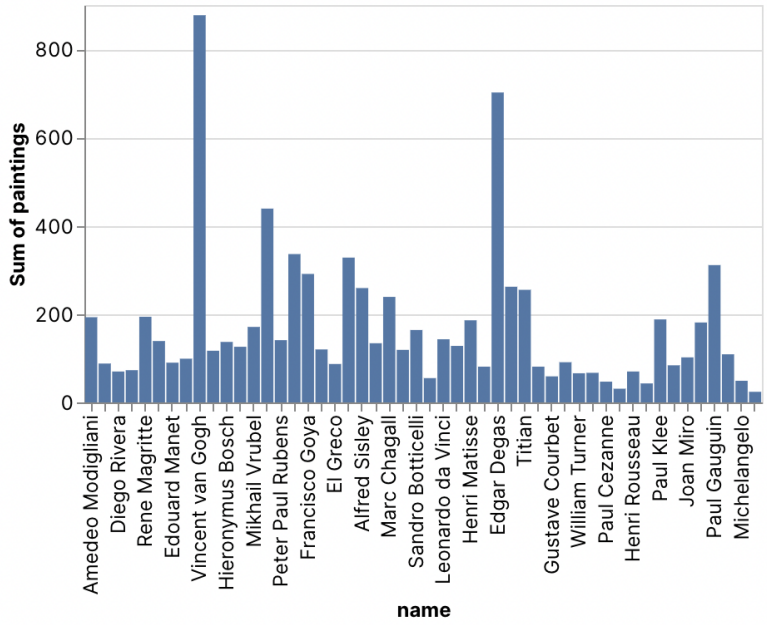


Fig. 4. Dataset Visualization

Convolutional Neural Networks:

Around the 1980s, CNNs were first created and put to use. At the time, a CNN [6] could only identify handwritten numbers to a certain extent. Reading postcodes, pin numbers, etc. was mostly done in the postal industry. The most crucial thing to keep in mind about any learning algorithm is that it needs a lot of computational power and data to train. Because of this significant disadvantage at the time, CNNs were restricted to the postal industry and were unable to penetrate the machine learning field.

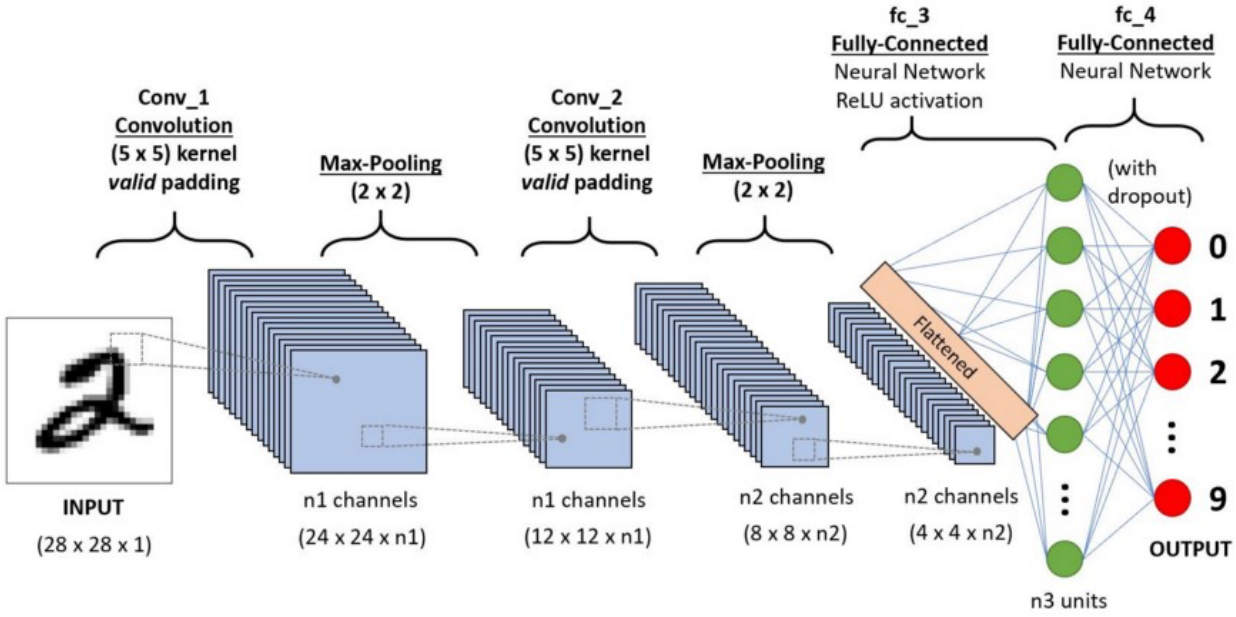


Fig. 5. Convolutional Neural Networks

Convolutional neural networks (CNN/ConvNet) are a kind of deep neural networks used most frequently to interpret visual data in deep learning. Normally, matrix multiplications come to mind when people think of a human brain, but that is not the situation with ConvNet. It makes use of a unique method called convolution. Convolution is a statistical procedure that takes two

functions and creates a quantifiable result that explains how the form of one is changed by the other in mathematics. Artificial neurons are arranged in numerous layers to form convolutional neural networks. Artificial neurons are computational models that compute the weighted sum of several inputs and output an activation value, roughly imitating their biological counterparts. Each layer of a ConvNet creates a number of kernel functions [7] that are sent on to the following layer when an image is entered. Typically, the first layer removes fundamental characteristics like edges that run horizontally or diagonally. The following layer receives this output and recognizes more intricate characteristics like corners or multiple edges. The network may recognize increasingly more complex elements, including objects, faces, etc., as we go further into it.

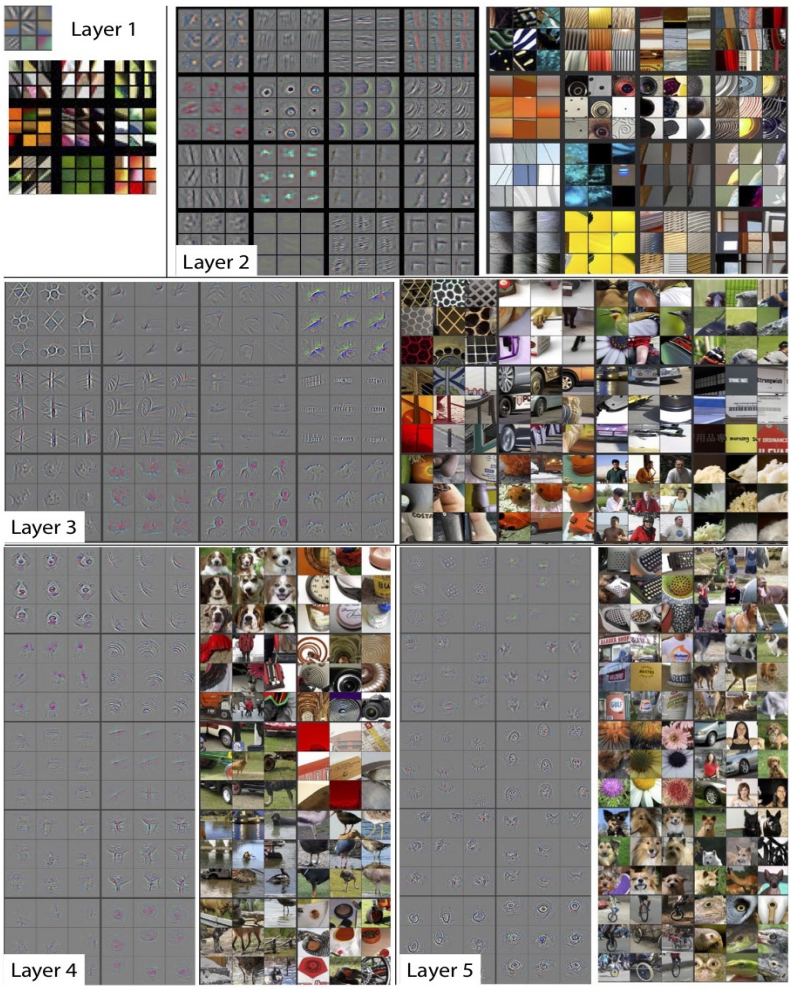


Fig. 6. Functioning of CNN on a image dataset

Hyperparameter Tuning:

It takes time to train deep neural networks to address a specific issue. Any task requiring NLP, whether it is picture classification, object identification, image segmentation [8], etc. We frequently work with huge models and datasets. A truly good model is also challenging to train. Both time and computing resources are needed. Additionally, the fact that there are several hyperparameters that must be properly tuned in order to obtain the optimum simulation does not make the task any simpler. To train the most effective learning algorithm we can for a particular job, the appropriate hyperparameter selection is essential. Because of this, deep learning hyperparameter tweaking is a hot topic both for researchers and developers. While developing a deep convolutional neural network, we may adjust a variety of hyperparameters. These below are some of them, in no particular order:

- The quantity of training epochs or iterations
- The speed of learning.
- Learning algorithm parameters itself.
- Total number of layers and neural units that make up a neural network.
- Size of the batch for tackling computer vision problems like object identification and picture categorization.

Neural Style Transfer:

In our project, we will be dealing with neural style transfer using python deep learning networks.[9] In order to proceed further with the process of style transfer, we deal with our dataset. In order to make the images more accessible to the machine, we resize the images and then continue with the process of building the model in order to apply the style of one image to another. As we start the process of neural style transfer, let's have a look at our dataset images. In our dataset, we have the following artworks of 50 major artists from all over the world. The two images below show the images.

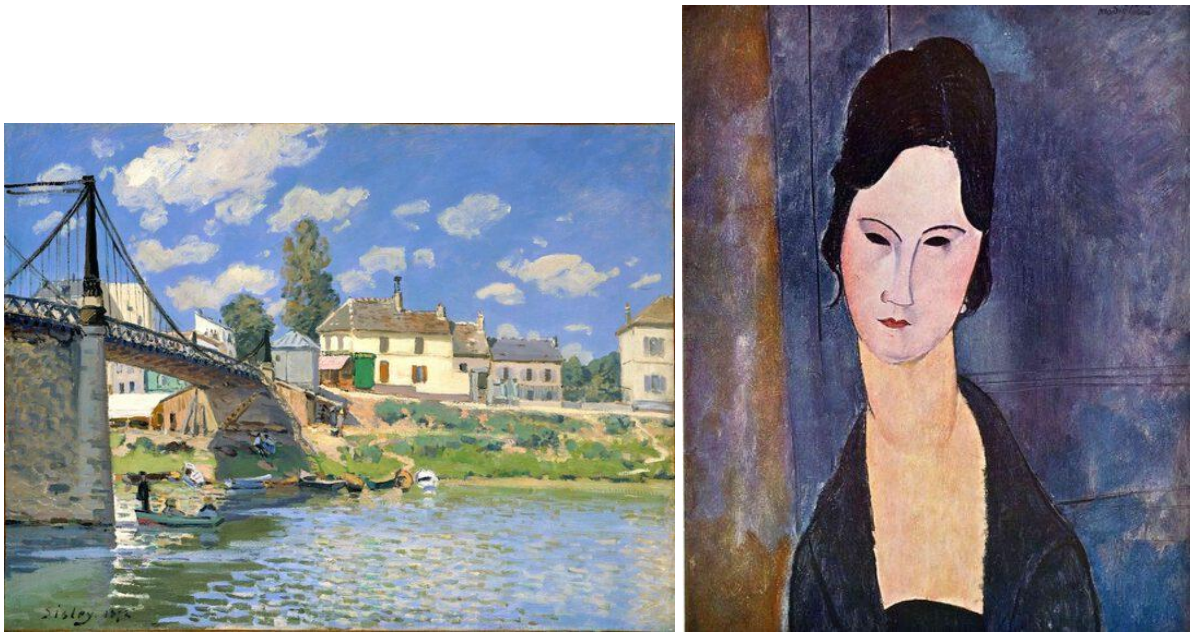


Fig. 7. Dataset Images

Neural style transfer is a technique for transferring a certain look from one picture to another while preserving the original image's information. The only modification is the way the image is styled to give it a more artistic feel. The layout or drawing is depicted in the content image, and the colors or painting used to represent the style are used. It is a computer vision application involving CNN model and image processing methods. Let's now examine how NST functions.

In order to enable the Deep Learning approach to distinguish between style depictions and content images, cognitive style transfer is used. To transport style from one picture to another and create a new picture with the features we wish to add, NST [10] uses a pre-trained deep neural network with additional loss functions. In order for style transfer to work, the neurons must be activated in a specific way, resulting in an output image and content image that are particularly similar in terms of content, while the design picture and the output values image should be similar in terms of texture and seize the very same ultimately present in the feature map. In a single loss formula, these two goals are merged, and we may choose how much we value both style reconstruction and content reconstruction.

The following data must be provided to the picture style transfer model as inputs:

- A content image is a picture that we wish to add style to.
- A Style Image: The look we want the content image to have.
- A produced input picture is the combined result of the content and style of the image.

A pre-trained feature selection method and a transfer network are required for developing a style transfer model. A pre-trained TensorFlow model called ImageNet-VGG is used by NST. The model cannot understand the images on its own. These must be transformed into raw pixels and provided to the model, which uses convolutional neural networks to turn the raw pixels into a set of features. As a result, the model functions as a complicated feature extractor halfway between the layers where the picture is supplied into the model and the layers that deliver the output. The model's intermediate layers are all we need to employ to characterize the content and aesthetic of the input photos.

Content Loss: Establishing commonalities between the produced picture and the content image is helpful. It seems to make sense that layers of the algorithm would concentrate more on the characteristics already existing in the image, or the image's overall content. The Euclidean distance between the intermediate higher-level feature representations of the input picture (x) and the content image (p) at layer l is used to determine content loss.

$$L^l_{content}(p, x) = \sum_{i,j}(F^l_{ij}(x) - P^l_{ij}(p))^2$$

Style Loss: Style loss and Content loss are fundamentally distinct. The style loss cannot be determined by just comparing the transitional aspects of the two photos. For this reason, we use the phrase "Gram matrices." Gram matrices, which display the normal distribution curve of characteristics in a certain layer, can be used to analyze the info in a picture. It is determined by how closely the feature mappings in a particular layer correlate with one another.

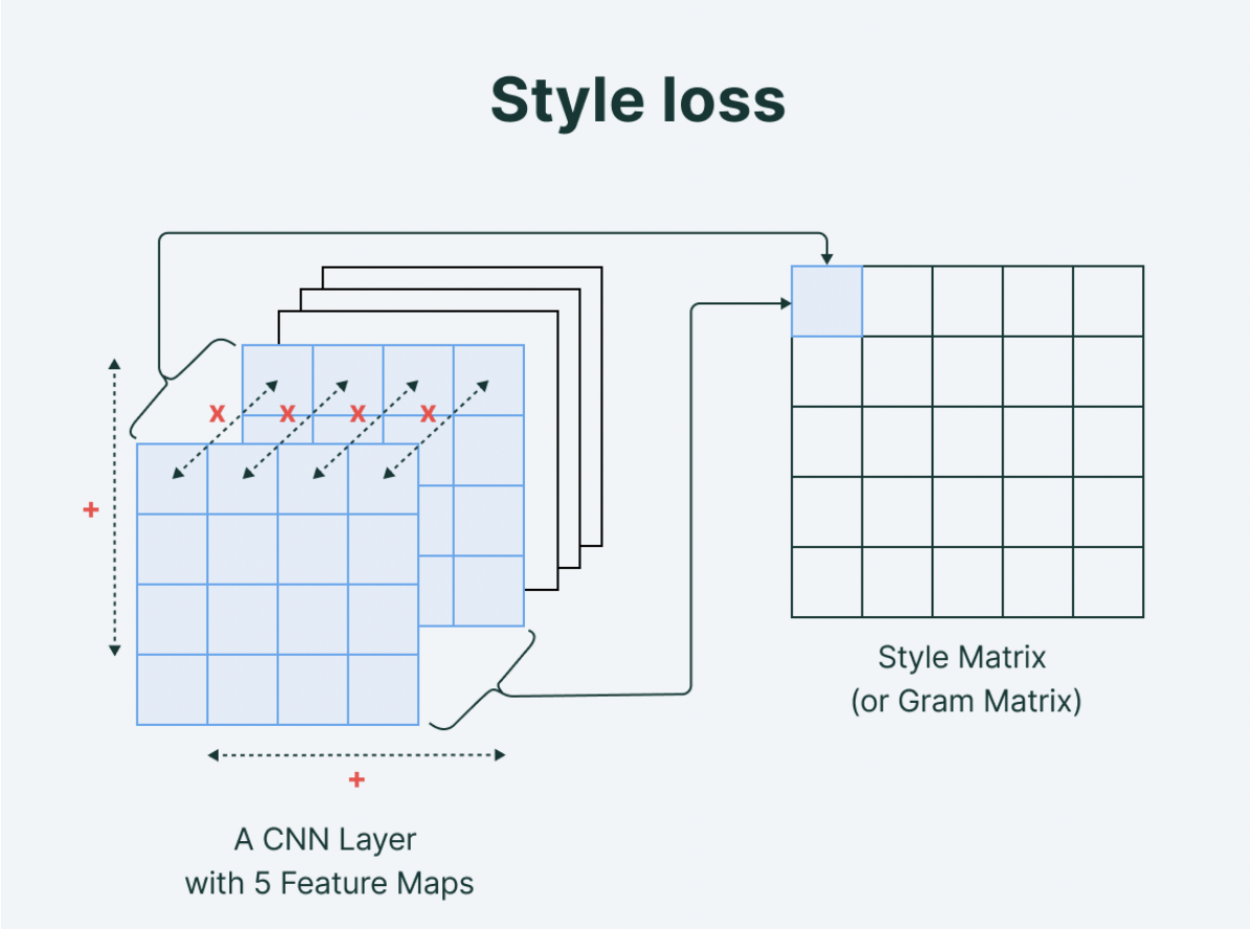


Fig. 8. Style Loss

RESULTS AND DISCUSSION

The below code snippet shows the experimental results and discussion with regards to the project. In this project we use the pre-trained Vgg-19 model in order to process the style transfer from one image to another.

```
from torchvision.ops.poolers import torchvision
def load_img(path, max_size = 400, shape = None):
  image = Image.open(path).convert('RGB')

  if max(image.size) > max_size:
    size = max_size
  else:
    size = max(image.size)


  if shape is not None:
    size = shape


  in_trans = tranforms.Compose([tranforms.Resize(size),
                                tranforms.ToTensor(),
                                tranforms.Normalize((0.485, 0.456, 0.406),
                                                    (0.229, 0.224, 0.225))])
  image = in_trans(image)[:3,:,:].unsqueeze(0)

  return image
```

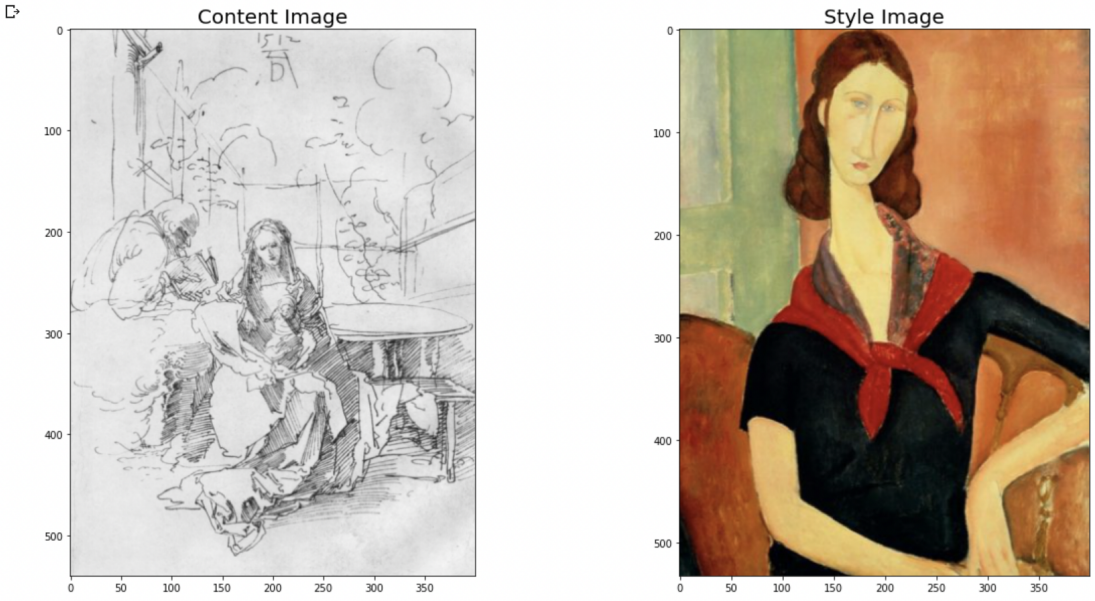
This code snippet shows how we use pytorch transforms in order to convert our image into a tensor and normalize the tensor.

```
def convert(tensor):
  image = tensor.to("cpu").clone().detach()
  image = image.numpy().squeeze()
  image = image.transpose(1,2,0)
  image = image * np.array((0.229, 0.224, 0.225)) + np.array((0.485, 0.456, 0.406))
  image = image.clip(0, 1)

  return image
```

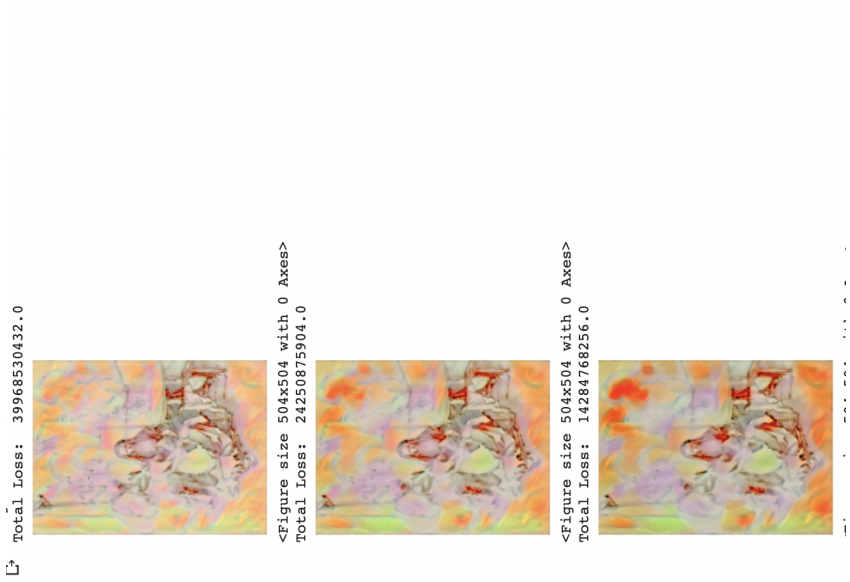
```
fig ,( ax1, ax2) = plt.subplots(1, 2, figsize = (20, 10))

ax1.imshow(convert(data))
ax1.set_title("Content Image",fontsize = 20)
ax2.imshow(convert(style_data))
ax2.set_title("Style Image", fontsize = 20)
plt.show()
```



The code snippet below shows the content image as well as the style image necessary for our process of style transfer.

Fig. 9. Feature Extraction



The image above shows how each iteration is carried out and the loss is minimized as the style gets applied to the image. Below image shows the final output of the project and shows how the style has been transferred to the content image.

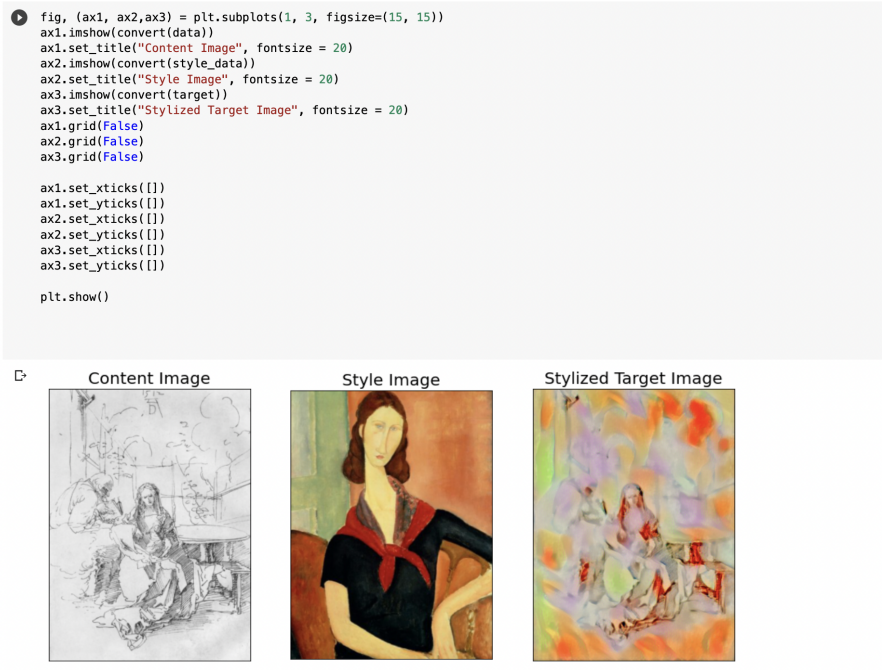


Fig. 10 Final Output

CONCLUSION AND FURTHER SCOPE

This project has a lot of scope and applications in the real world artificial intelligence industry. In this project, we deal with style transfer and how this works. We already see its application on our day - day activities such as our phone photos filters, Snapchat filters, Instagram filters, etc. Apart from these this can also be incorporated in the fields of:

- Commercial Art
- Gaming
- Virtual Reality

In this study, we present simple controls for neutral style transfer. In order to significantly enhance the quality and adaptability of the current technique, we propose strategies to accommodate the space, color, and scale aspects that we hypothesize are involved in image style. Combining styles in a way that is understandable is one way the control mechanisms we offer might be put to use. The alternate method of merging styles by linearly overlaying in the style representation, as is done, for instance, in the contemporaneous work of Dumoulin et al. [3], contrasts with this. A potential issue with that strategy is that it becomes challenging to produce visually appealing new styles if the directions in the design representation do not correlate to perceptual factors. Even with our approaches, it might be difficult to choose which variables to combine for visually acceptable outputs. Predicting which mixtures of styles will blend harmoniously to create new, perceptually pleasant styles is an intriguing open research subject. The ability of neural style transfer to generate new picture patterns based on the original images makes it particularly alluring. This adaptability results from the style being represented as geographic summary statistics rather than patch-based approaches [12, 20, 6]. However, it is challenging to exert significant parametric control over the stylization since it is unclear how the perceptual features of design are reflected in the summary statistics. To do this, it could be necessary to promote proper CNN factorizations.To learn representations that factor the visual information throughout spatial scales, for example, use representations during network training. In reality, this hits on a key machine vision research question: how to create interpretable but potent picture representations that separate images into the individual components.

REFERENCES

[1] L.Benedetti,Winnemo ̈ller,M.H.,Corsini,andR.Scopigno. Painting with bob: Assisted creativity for novices. In *Proc. UIST*, 2014.

[2] A. J. Champandard. Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks. *arXiv:1603.01768 [cs]*, Mar. 2016. arXiv: 1603.01768.

[3] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *Proc. ICLR*, 2017.

[4] H.S.Faridul,T.Pouli,C.Chamaret,J.Stauder,E.Reinhard, D. Kuzovkin, and A. Tremeau. Color mapping: A review of recent methods, extensions and applications. *Computer Graphics Forum*, 35(1):59–88, 2016.

[5] P. Wilmot, E. Risser, and C. Barnes. Stable and Control- lable Neural Texture Synthesis and Style Transfer Using Histogram Losses. *arXiv preprint arXiv:1701.08893*, 2017.

[6] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proc. CVPR*, 2016.

[7] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proc. ECCV*, 2016.

[8] A. Hertzmann. *Algorithms for Rendering in Artistic Styles*. PhD thesis, New York University, 2001.

[9] L.A.Gatys, A.S.Ecker, and M.Bethge Image Style Transfer Using Convolutional Neural Networks. In *Proc. CVPR*, 2016.

[10] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree- structured vector quantization. In *Proc. SIGGRAPH*, 2000.