Neural Style Transfer

Deep Learning

Group Number : 25

Group Members:

1. Aditya Somani - 2020A7PS2049H
2. Deep Chordia - 2020A7PS2073H
3. Kavyanjali Agnihotri - 2020A7PS0185H
4. Parth Tulsyan - 2020A7PS1883H

Literature Review:

Image Style Transfer Using Convolutional Neural Networks

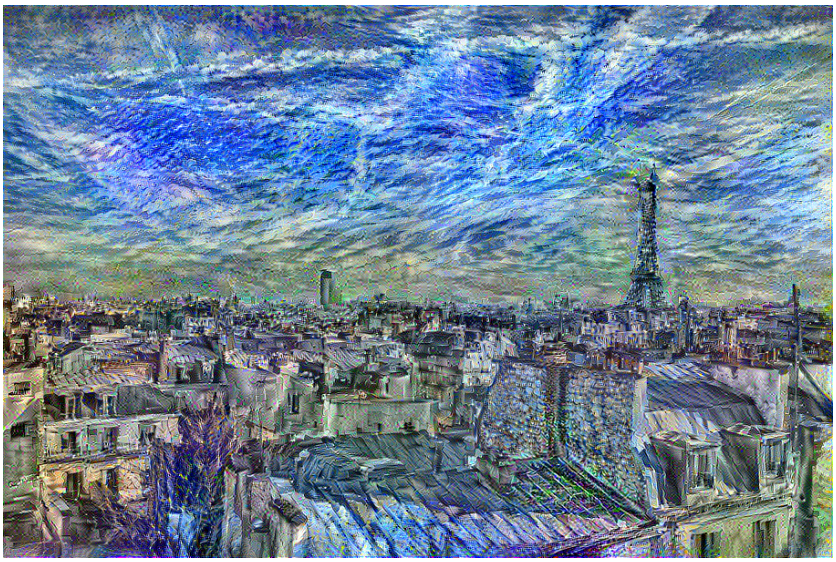
Despite the impressive results some of the other algorithms produce, they all share the same fundamental flaw: they can only transfer textures using the low-level picture attributes of the destination image. However, in a perfect world, a style transfer algorithm could extract the semantic image content from the target image (such as the objects and general scenery). Moreover, the paper uses this information to guide a texture transfer procedure to render the semantic content of the target image in the source image's style.

The paper proposes a texture transfer algorithm that limits the texture synthesizing technique by feature representations. The style transfer method elegantly simplifies an optimization issue inside a single neural network. New images are created by performing a pre-image search to match feature representations of example photos. Moreover, the paper used the feature space produced by a normalized version of the 19-layer VGG network. The squared-error loss between these two feature representations is then defined. Higher layers in the network capture high-level information about the objects and their placement in the input image. However, they do not significantly limit the reconstruction's actual pixel values. These layers are the content representation. Using a feature space that can collect texture data from numerous levels, the paper derives a representation of the style of an input image. By creating a matching image, the Gram matrix layers of the network provide these feature correlations. To visualize the data gathered by these style feature spaces, they minimize the mean-squared distance between the entries of the Gram matrices from the original image and the Gram matrices of the created image using gradient descent from a white noise image.

They create a new image that matches both the style and content representations by jointly minimizing the distance between the feature representations of a white noise image and the style representations of the painting.

Input Images:

Desired Output Image / Output Image by the original Model:



Our Attempts:

1. Initially we tried to implement the idea by utilizing the state-of-art GANs. We were unsuccessful. It was tough to figure out how GANs work and how to use them adequately for our use case.

2. Further we tried to implement the paper - A Learned Representation for Artistic Style, which is basically applying multiple styles at one go. While doing so, we ran into multiple problems with the long time of training as well as insignificant results after training. The results were insignificant as the image that lost the content from it and came out to be a gradient color based on which style was applied.
   Input Image:

The outputs we received after implementing the the same model locally, with pre trained and locally trained weights

Proposed Improvements:

- ### Trying ResNet instead of VGGNet.

  In VGGNet, each weight is slightly altered using the backpropagation method so as to reduce the model's loss. This value is multiplied by each local gradient as the gradient flows backward to the beginning layers. As a result, the gradient becomes smaller, which causes the updates to the initial layers to be very small and significantly increases training time. We tried to use ResNet in order to solve the vanishing gradient issue. Shortcut connections are used in ResNet architecture to address the issue.

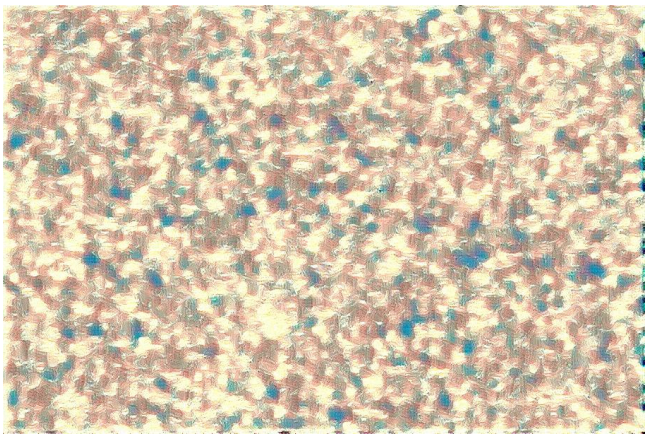
  Using the same input images as the original model, the output we got using ResNet.

Output using Resnet in a single iteration

- ## Trying EfficientNet instead of VGGNet:

  EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. The compound scaling method is justified by the intuition that if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image.

  Using the same input images as the original model and EfficientNet.

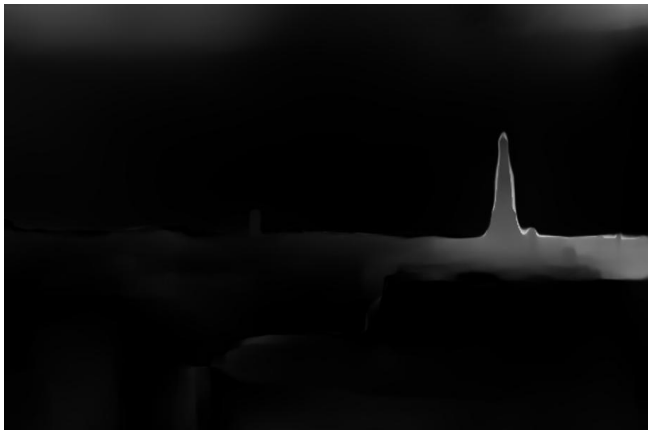


Best outputs achieved using EfficientNet

- ## Depth Image

  The high-level features on pre-trained networks are primarily intended for object recognition; thus, they concentrate on the primary target and ignore extraneous aspects. The distinction between the foreground and background and various objects is lost. The depth map of the semantic content is kept when a picture is styled, effectively representing the spatial distribution in the image. As a result, the semantic content of the image is preserved. While transferring the style, incorporating depth preservation as a second loss preserves the overall image layout.

  We added a plugin, where a depth map of the content and generated image is calculated. After taking the pixel wise difference between both of them and squaring it up, the sum is added to the loss function. Hence, making it richer. The output of the plugin can be seen below



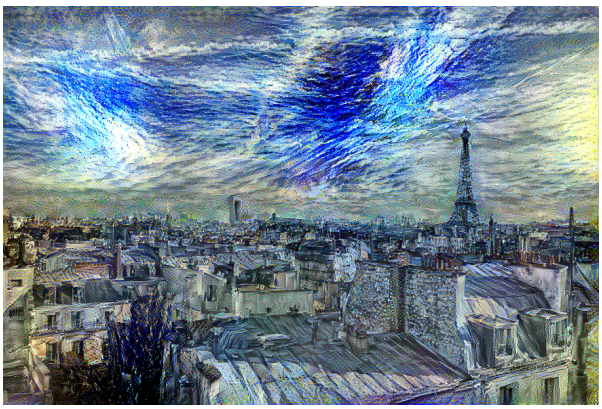
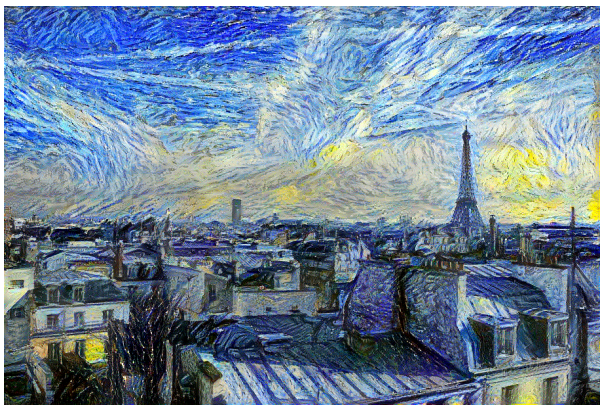
Depth difference between the generated
and content image after 200 iterations.

Results and Conclusion:

VGGNet with Depth Preservation

We broadly observed that increasing the complexity of the architecture wasn't giving better results. But the Depth Aware loss function was increasing the output considerably.

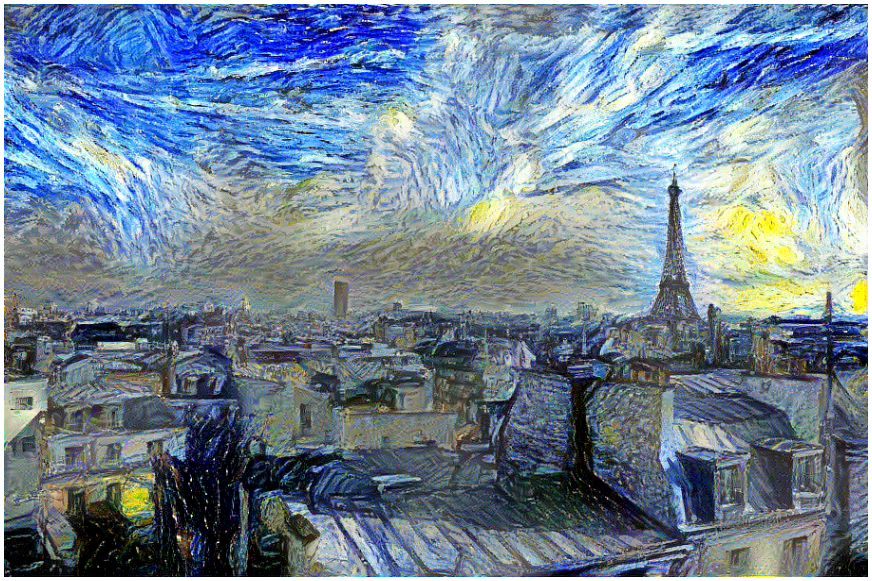
So, In order to preserve the details of the foreground as well as the background, we added a function for depth. We added it to the implementation of the chosen paper using VGGNet and got the following results.

Existing Model (Image Style Transfer)	**Our Model (with Depth aware loss function)**
	
Output after after 3000 iterations	Output after 1500 iterations

As we can clearly see the depth aware model has more content than the non-depth aware model and the colors are more prominent as well.

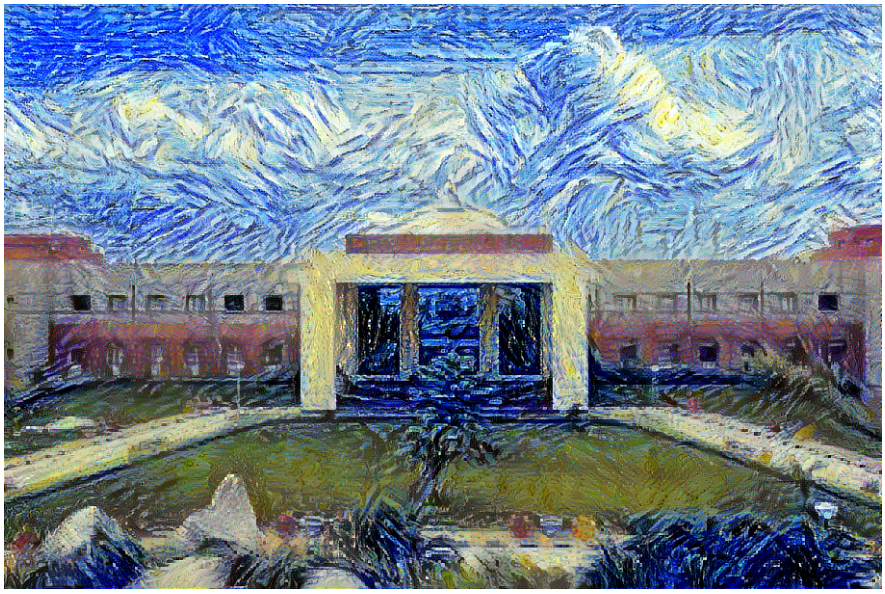
Along with this, we were able to reduce the training time considerably. This is indicated by the number of iterations needed by the models. In our implementation, the model provides a much better output at 1500 iterations, whereas the original model requires almost double the time.

Furthermore, the model had a completely apt amount of transfer from the two input images around 800 - 1500 iterations. After 5000 iterations, the model was over-training as we can see from the image below.



Output after 5000 iterations

Finally to give off the BITSIAN feeling



BITS Hyderabad Academic building with the same style after 500 iterations

Contribution of the Members:

Name	Contribution
Aditya Somani	Reading paper, Code for base model, Proposed improvement, Implementing Improvement, Report
Deep Chordia	Reading paper, Proposed Improvement, Implementing Improvement, Report
Kavyanjali Agnihotri	Initial papers selection, Reading paper, Base model selection, Trying Improvement, Report
Parth Tulsyan	Initial papers selection, Reading paper, Initial model selection, Proposed improvement

GitHub Repo

Latex Report - https://drive.google.com/file/d/1cS20BkLN6-HY6FhhDOtQWDxUsS0DsRGf/view