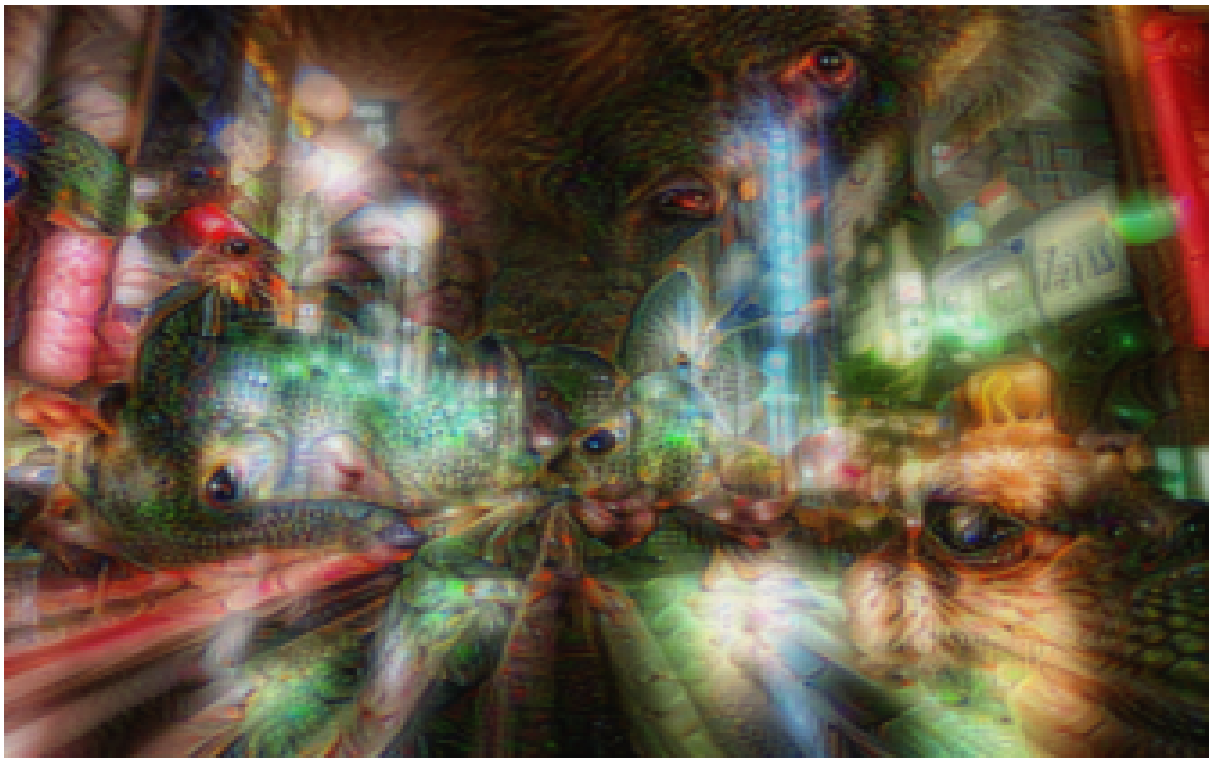
ART AND MACHINE LEARNING
CMU 2022 SPRING
PROJECT 2

DeepDream: Reinvent and Applied Research



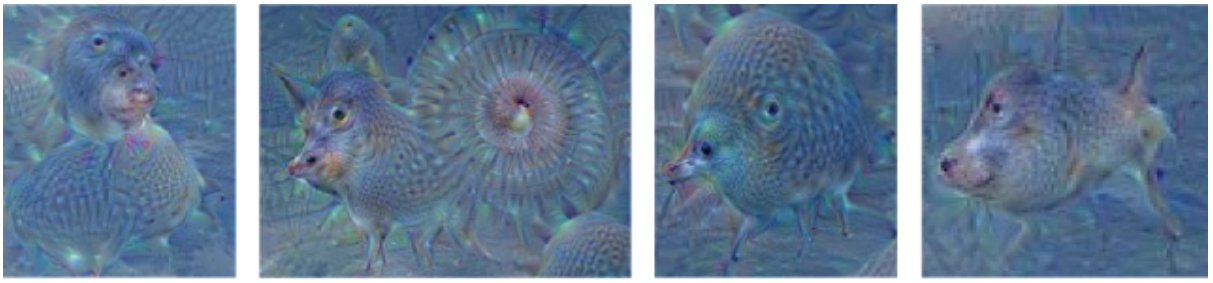
Shijun Liu
Haoran Lyu
Ningxin Xu

DESCRIPTION

In this project, we create a customized DeepDream Framework. Replacing some of the outdated frameworks used in the original paper with Pytorch and changing the network structure to see if we can discover some new ideas. We take three samples representing three kinds of images in our daily life and tune the parameters respectively in order to find what is the best way to deal with different raw materials. Generally speaking, we generate really good and unexpected results.

Concept

DeepDream is an interesting technique created by Google in 2015.In a pre-trained convolutional neural network, only a few parameters need to be set, then an image can be generated by this technique.



Typical images generated by DeepDream

These pictures are full of hallucinations and dreams, so that's why it is called DeepDream. This algorithm is very fascinating because it has an unexpected effect. We know that neural networks have made significant progress in image classification, but due to the fact that there are too many parameters in deep learning networks, this algorithm is a black box. Although it can achieve good results, people still know little about its internals. As a result, people want to be able to peek inside the network.

Reverse Neural Network

We know that a neural network reads an image as an input, passing through a multi-layer network, and finally outputs a classification result. However, only one result is not enough. One of the challenges of neural networks is to understand what is going on in each layer. We know that after training, each layer of the network gradually extracts more and more advanced image features until the last layer compares these features to make a classification result. For example, the first few layers may be looking for the features of edges and corners, and the middle layers analyze the overall outline features, so that more and more complex features can be developed by increasing the number of layers, and the last layers combine these feature elements to form a complete explanation, so that by the end the network will be able to react to very complex things, such as pictures of leaves, kittens, etc.

In order to understand how the neural network learns, we have to understand how the features are extracted and recognized. If we analyze the output of some specific layers, we can find that when it recognizes some specific patterns, it will highlight these features ground enhancement, and the higher the number of layers, the more complex the patterns identified. When we analyze these neurons, it is unrealistic for us to take a lot of pictures as inputs and then understand what features are detected by these neurons, because many features are difficult to recognize by human eyes. A better way is to reverse the neural network. Instead of inputting some pictures to test the features extracted by the neurons, we pick some neurons and see what it can simulate the most likely picture, and reverse this information. Passing back to the network, each neuron will show the pattern or feature it wants to enhance.

For example, in the above pictures, we can see that different neurons simulate different enhancement features and patterns, some are dogs, some are snails, and some are fish.

DeepDream

Through the above process of reversing the network, we will force the neural network to produce something that does not exist in the picture, which will produce similar dreams and hallucinations. In fact, these dreams emphasize what the network has learned. This technology provides us with a qualitative feeling of abstraction, although this has little to do with dreams in reality, this is the earliest inspiration of DeepDream.

If we repeatedly apply this algorithm to its own output, that is, iterate continuously, and apply some scaling after each iteration, so that we can continuously activate features to get endless new effects. For example, at the beginning, some neurons in the network simulated the outline of a dog in the picture. Through continuous iteration, the network will become more and more convinced that this is a dog. It will become more and more obvious.

Technique

1. DNN with gradient ascent

   For a normal deep neural network, given input samples and labels, we can calculate the loss by forward propagating the input data. Later, we back propagate the loss to calculate the gradient for parameters in the network. Then, update the parameters in the network and decrease the loss, to get a network that gives labels as similar as possible.
   The process can also be reversed. If we fix the parameters, then we need to optimize the input data. With a similar process, we can get the input data that meet the requirement. Different from gradient descent, the target is to maximize some certain index, so we just apply negative to the index and it's similar to the gradient descent.

2. Image Pyramid

   When operating on images, it is common to manipulate the size of source images to transfer to what is required. Enlarge or reduce the size of the image is scaling. One way is using the

Image Pyramid. There are several different kinds of image pyramids. In our experiments, we used the Gaussian Pyramid. The gaussian kernel is applied to the input image. Performing convolution operations on the image with the kernel, then subsampling the output image, we can get a higher-layer image. Repeat this process for multiple times, we can get the Gaussian Pyramid.
Appling Image Pyramid can generate textures in different sizes, making the image seem to be layering rather than monotonous.

3. Gaussian Blur

Gaussian Blur is used for reducing noise and details in images. From an mathematica point of view, Gaussian Blue is applying convolution operations on the image with normal distribution. Under 2 dimensional space, the normal distribution is:

$$G(u, v) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/(2\sigma^2)}$$

The value of each pixel is the weighted sum of its neighbor pixels. Applying Gaussian Blur can reduce some noisy pixels in the output images, to get more smooth images. In our experiments, we regard Gaussian Blur as an adjustable parameter. We applied Gaussian Blur in some of the following experiments. However, the result remains to be discussed.

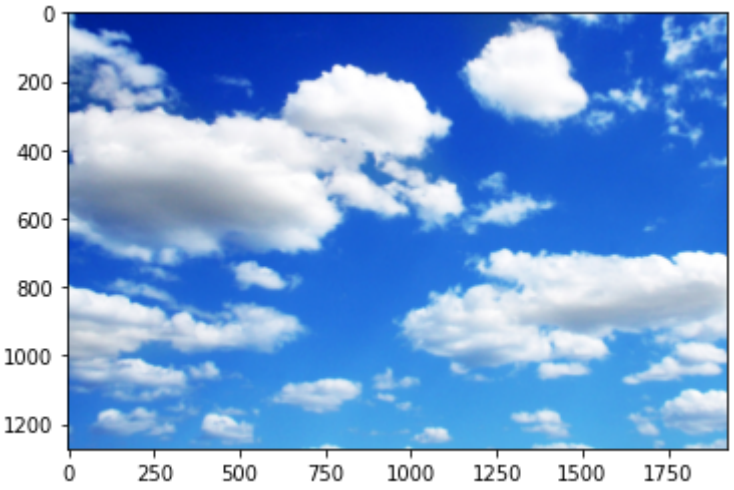
Process

We built our code from scratch according to the original paper of DeepDream presented by Google. Since Caffe, the framework used in that paper, is no longer popular in industry, we selected Pytorch as our tool and began to create our own DeepDream library.

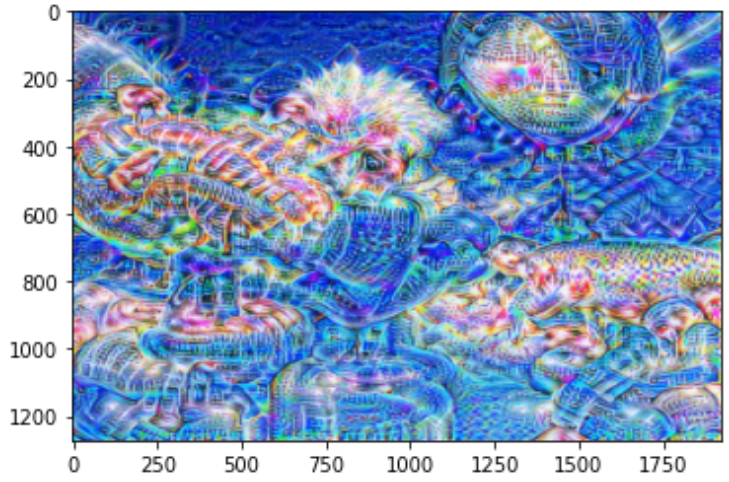
To simplify our model, we choose VGG-16 as our first experimental network, and we set the following parameters:

- epochs
- scale
- layers
- alpha
- times
- channels
- lr

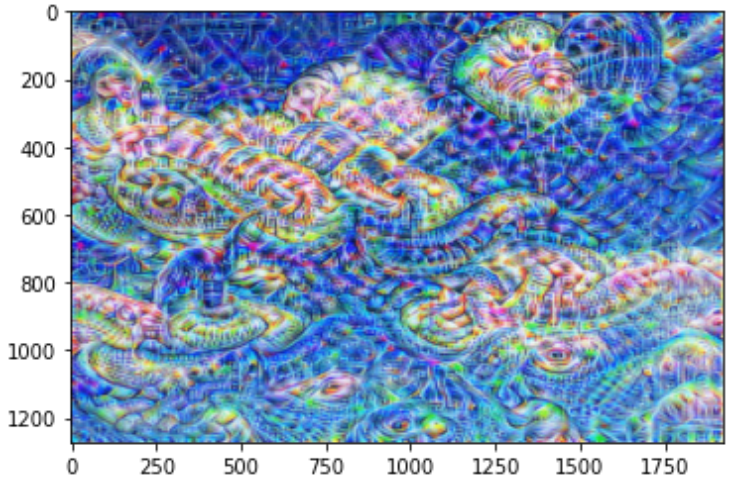
The highlight of our innovation lies in the fact that Google adjusts the original image based on a loss function summing up the outputs from the same layers or blocks in the neural network when they resize the image on different scales. We try outputs from different layers and blocks when we calculate the loss function for images resized by different scales.. The following images show some of the results we derived from the experiment on layer sizes and numbers.



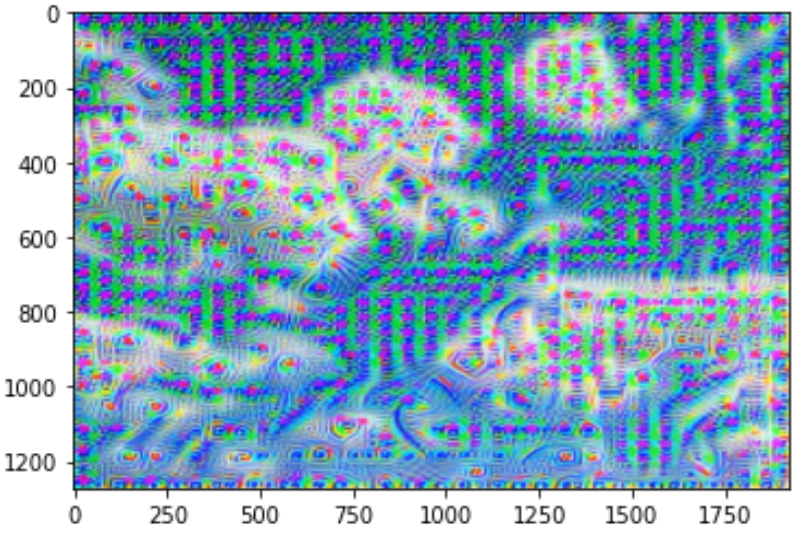
Original Image



Using only the outputs from the last convolution blocks



Using the outputs from the 6th, the 12th, the 18th, the 24th, the 30th, the 30th, the 30th, and the 30th blocks in the loss function when shrinking the image with scales of 1.8^8, 1.8^7, 1.8^6, 1.8^5, 1.8^4, 1.8^3, 1.8^2, and 1.8.

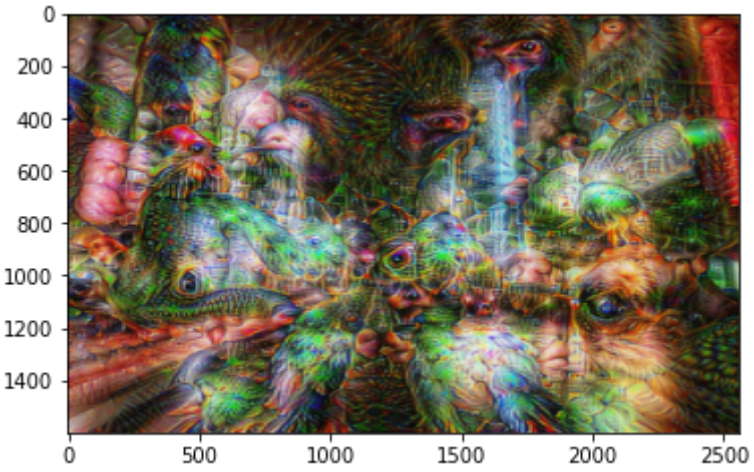


Using only the outputs from a block in the middle

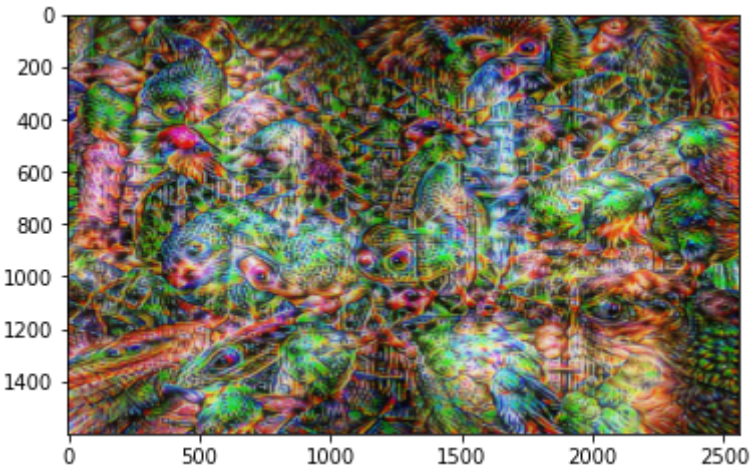
It's not difficult to choose our"Outputs from different layers & different scales" approach since it looks good and also provides us a chance to discover a different way of implementing DeepDream.

Then let us have a look at some other parameters. For epochs:

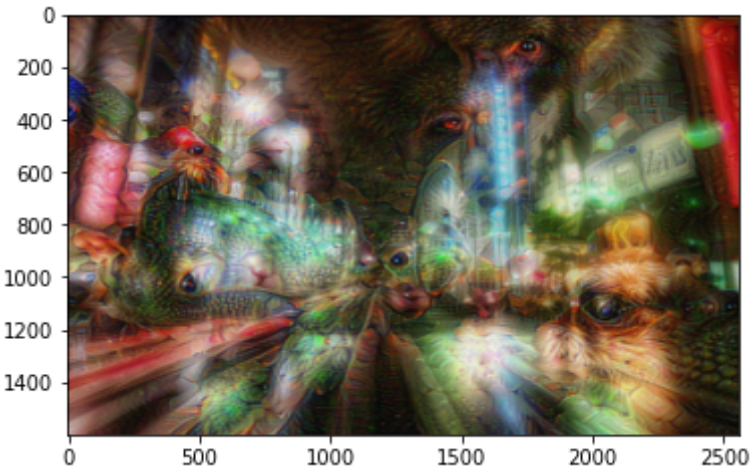
original image



epoch=10



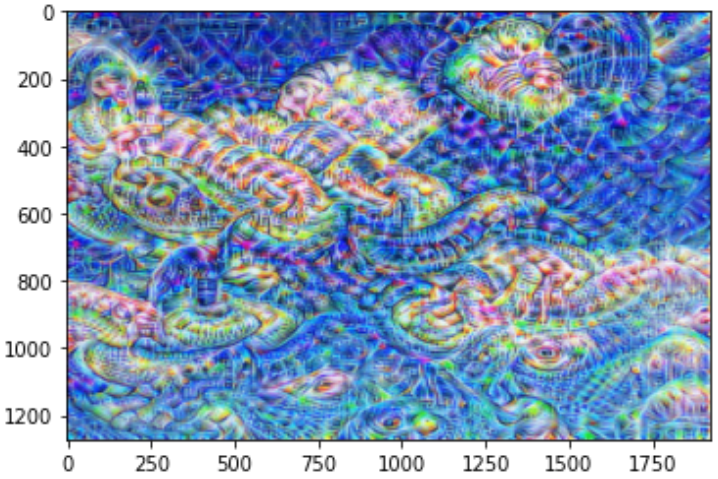
epoch=20



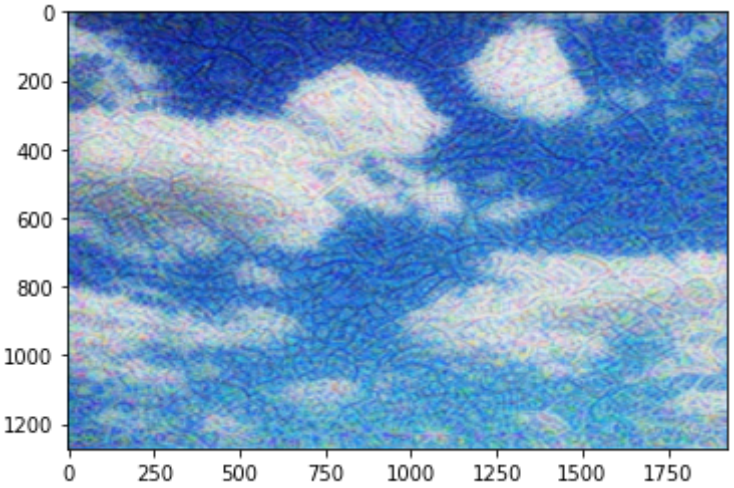
epoch=5

It's not hard to read these results. For complex images, less epochs will help if more epochs are used, too many patterns will break the overall structure of the image. But for simple images like the clouds in the first experiment,more epochs can be used to create more possibilities and randomness.

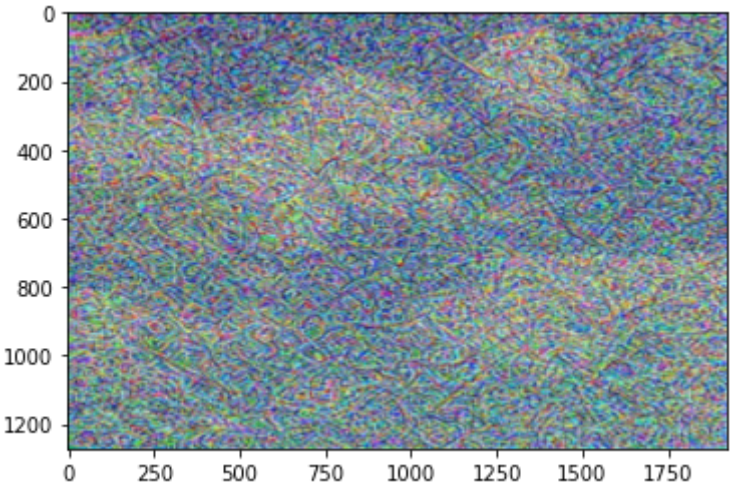
Next we did some experiments on the learning rate and optimizers.



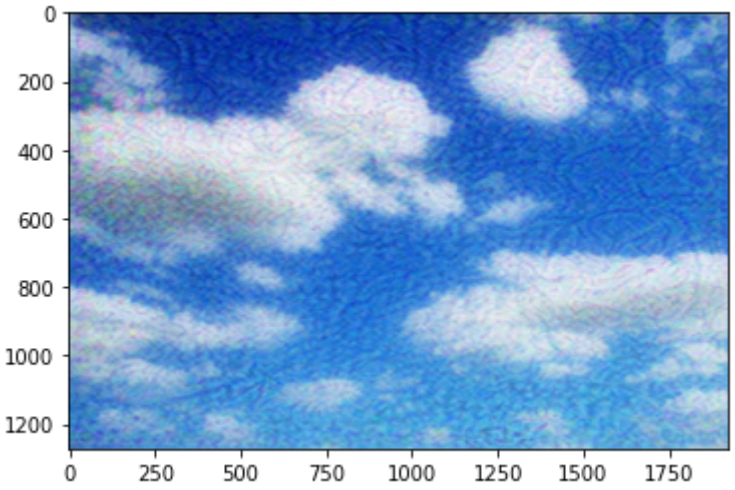
Naive Gradient Descent(lr=0.2), the original way



Adam(lr=0.2)



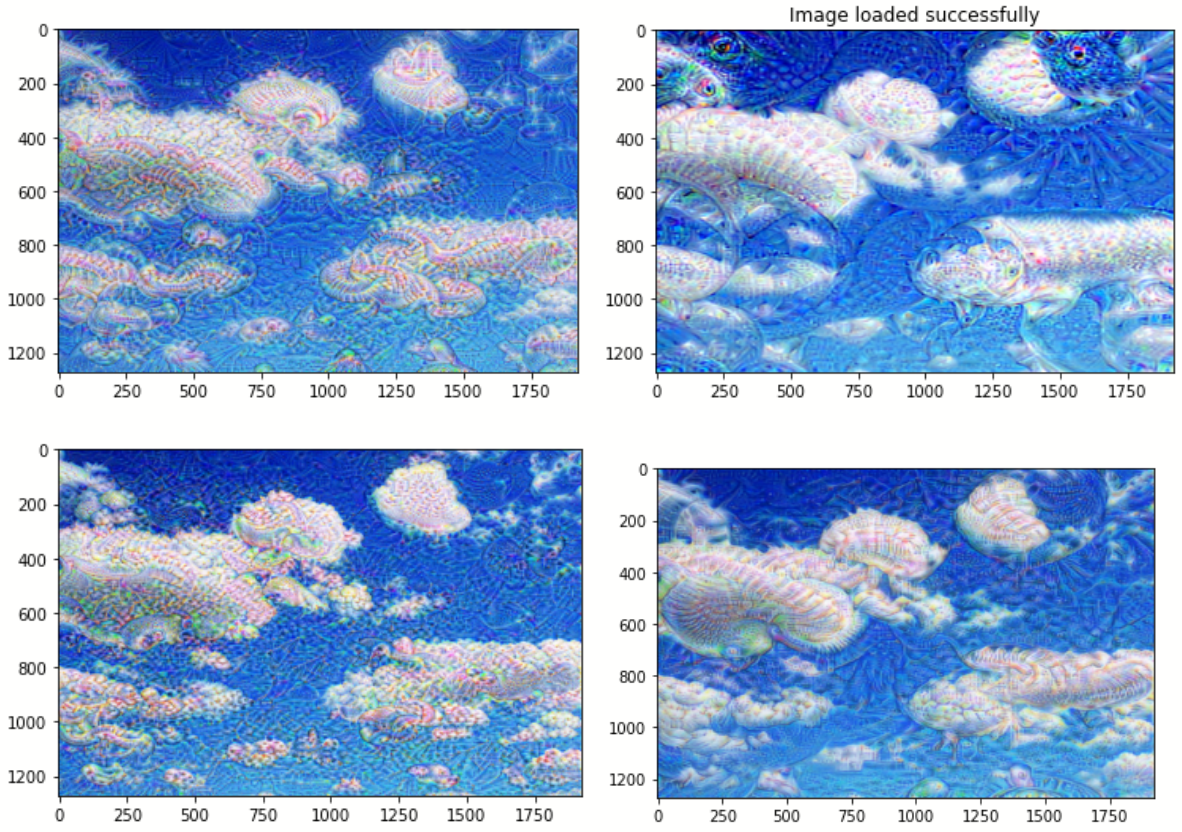
Adam(lr=2)



SGD(lr=2)

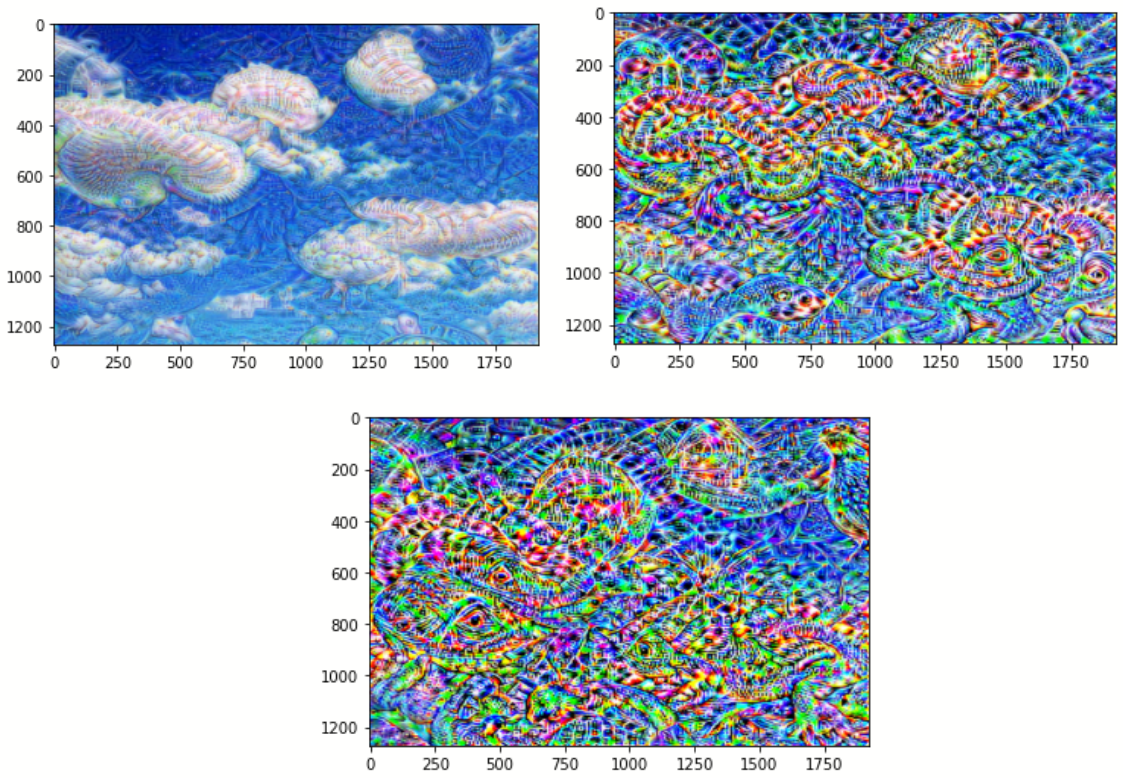
So what the learning rate and the optimizer do is approximately the same as the number of epochs we use- deciding the density of the dreamy patterns. And it seems that a concise optimizer will give a more unimaginable pattern.

Then we do some research on scale, which we think will affect the organization of the image.



Particularly, scales and how many times we resize an image have significant impacts on what the final image looks like, and such impacts vary significantly from image to image.

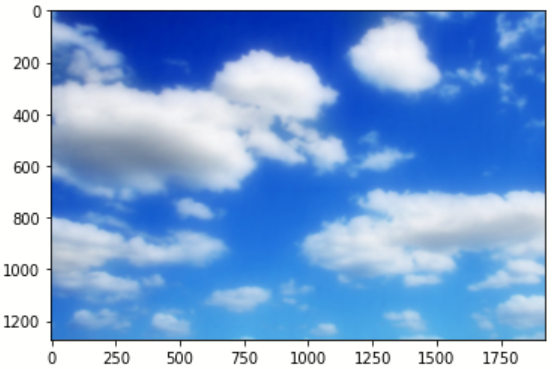
Finally we take a look at the alpha, which means the degree on how we merge the original image and the scaled image.



Seems that a big alpha also breaks the structure of the original image, which is not pleasant for viewing. We finally choose alpha=0.2.

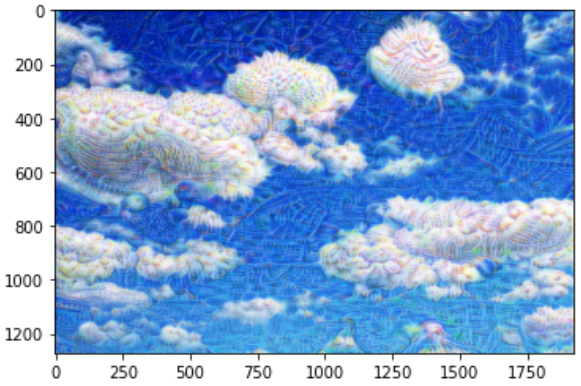
Reflection

1. Too many channels
   For each scale, we chose different channels to calculate the loss function. We randomly picked some channels and found that most channels are generating the same images without making any changes. One example is as follows:



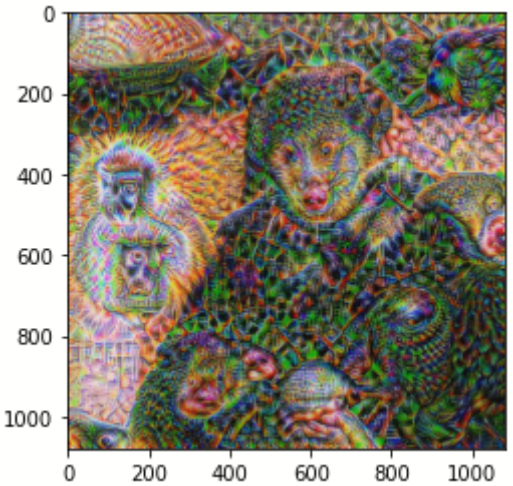
   It is hard to examine all the 398 channels and find some meaningful results. Thus, considering the limited time, we did not dive deep into it.

2. Gaussian Filter
   We tried to apply the gaussian filter on our result. The initial result is as follows:



The gaussian filter tends to blur the image and reduce the textures that we generated on the image. We also tried different parameters , and the result is not as good as what we expected. Unfortunately, it is different from the original paper of DeepDream. The most important effect on the result is, the artistry of the images is decreased.
If possible, we may examine more parameters and try to apply an effective filter on the image in the future. But this time, we did not apply the gaussian filter on our result.

3. Different scales and how many times the image are resized
   Trying different parameters is time consuming. The different combinations between scales, scaling time and alpha are numerous. Due to the limited time, we just tried limited parameters for the scale, the scaling time and the alpha. To be more specific, we adjust one parameter when the other two parameters are fixed. Since the result seems already similar to the expected effect, we did not try more. We choose scale = 1.8 and scaling time to be 8, as well as 0.2 for alpha. When applying the effect on human faces, the result is shown below:

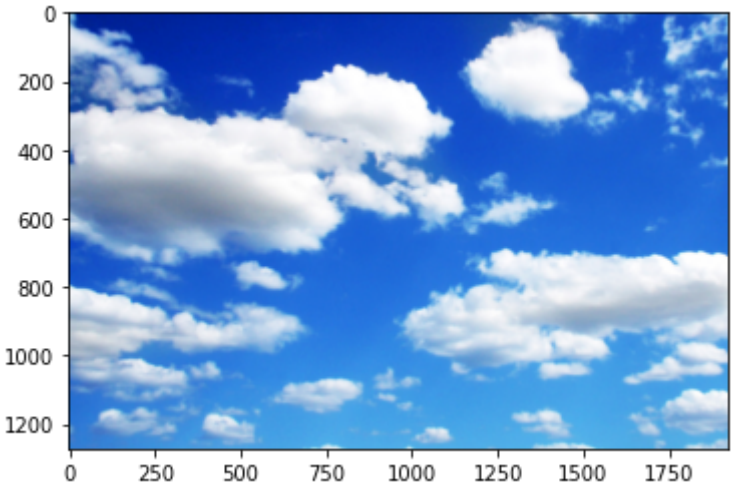


In the future, we may test different combinations of parameters for different subjects to find the best for each different subject.

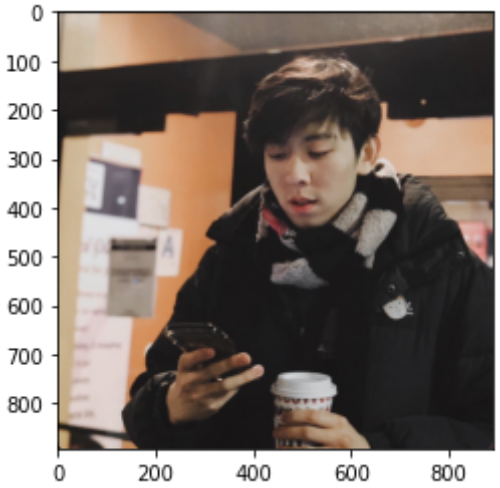
RESULT

We take different kinds of images as inputs for the deepdream network. Including:

City views(Complex)

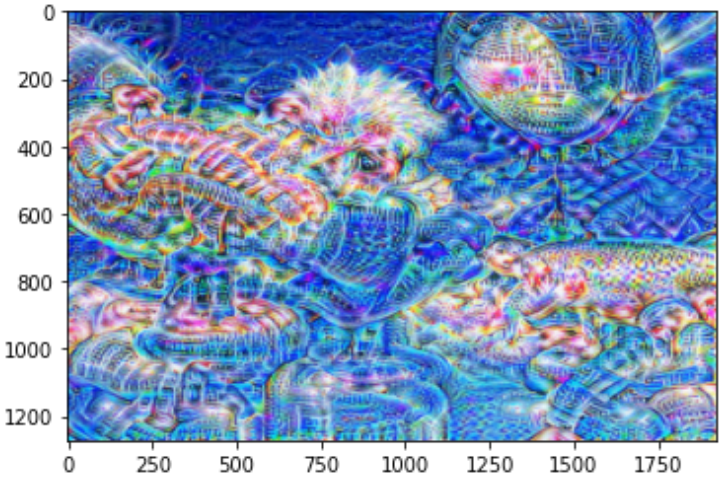


Nature views(Simple)



Casual life images(Neutral)

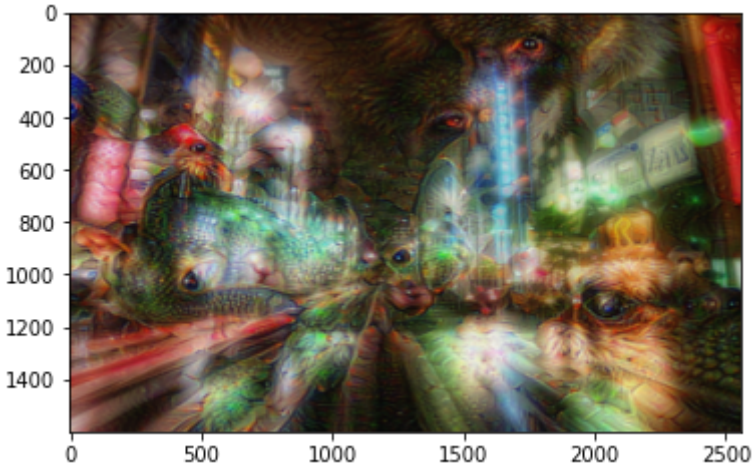
For the cloud image, obviously the deepdream can add some elements to the dull picture.



Scale 1.8, epoch 20, lr=0.2, Naive GD

We can transform the cloud into animal-like creatures, leaving imaginations for viewers.

Next we try some more complex pictures, this one is a typical Tokyo night scene. It is not difficult to come to the conclusion that a complicated process is not appropriate for creating authentic work. So what we do is to reduce the epoch and lr.



Scale 1.8,epoch 5, lr=0.1,Naive GD

This picture protects the colorful part(the shops on both sides) of the picture so it is still easy to recognize that it is originally a street view. The highlights are how deepdream transfers the night sky into some eyes, and the horizon end looks like a nose. As a whole the night sky looks like a wolf supervising the street. Also some of the shops are transformed into some small cute birds and cats. Such an interesting contrast makes this pic very successful.

Finally we take an avatar from our friend to see if daily human portraits are suitable for casual images.

Scale 1.8, epoch 20, lr=0.2, Naive GD

DeepDream adds some Cthulhu-styled elements for this picture. First, adding some twist features to some parts of the photo and then putting some eyes into the center of these twists. I think some people will definitely like it.

CODE

https://colab.research.google.com/drive/1_QinOik9k01A4_0T5RmkHa4SAyWi2ecv?usp=sharing

REFERENCE

[1] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015). [pdf] (Outstanding Work, most successful method currently)

[2] Mordvintsev, Alexander; Olah, Christopher; Tyka, Mike (2015). "Inceptionism: Going Deeper into Neural Networks". Google Research.

[3] Zhang, Richard, Phillip Isola, and Alexei A. Efros. "Colorful Image Colorization." arXiv preprint arXiv:1603.08511 (2016).