CS7015: Deep Learning for Computer Vision

Project Report Submitted by:

Jose Moti (CE17B118) Sooryakiran (ME17B174)

Paper Title: "SinGAN: Learning a Generative Model from a Single Natural Image"
https://arxiv.org/pdf/1905.01164.pdf
Best Paper, ICCV 2019
Tamar Rott Shaham, Tali Dekel, Tomer Michaeli

Official Implementation: https://github.com/tamarott/SinGAN

Introduction

Through this paper on "Learning a Generative Model from a Single Natural Image", the authors present a novel unconditional generative model to learn the composition and the texture of a single natural image and generate similar images with the same global features and local textures. The network is trained in multiple scales, in which the lowest scale is completely generative and generates a low-resolution target image from Gaussian noise. Succeeding scales take the lower resolution outputs from the preceding scales mixed with Gaussian noise to generate a target image of higher resolution. The discriminator does not learn the entire image, but small patches of a fixed size from it. Since the generated images are of lower resolution in lower scales, the relative effective receptive field covered by the fixed patch size is larger in lower scales and decreases as we move up the scale, i.e. when resolution increases. This means that the lower scales learn the global structure of the image, and higher scales learn the image textures.

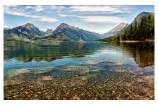
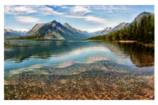
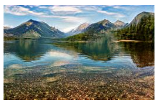
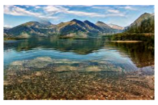
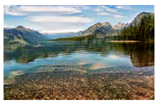
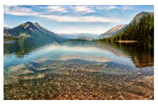
Since the original implementation was available, we carried out the following experiments and the results are given below.

Experiment 1: Effect of residual connections.

At each scale, the generator model takes a low-resolution image as the input and tries to generate a high-resolution image from it. When a normal generative network (without residual connections) is trained at a scale, the network output is random in the beginning. If we have residual connections in the generative model, the network can initiate learning from somewhere near the identity function (i.e. up-sampled low-resolution image). Hence, we think that a generator model with residual connections can learn faster.

Experiment details: Three models as presented below are trained on the given image.

Random generated samples from scale N.

Without Residual Connections 2000 steps/scale						
Without Residual Connections 1000 steps /scale						
With Residual Connections 2000 steps /scale						

Results: But the results we got are really different. We did not observe any difference in convergence speed. But we can see that the degree of variations in the generated samples with residual connections is marginally higher.

Possible Explanation: There are no much variations in the model without residual connections because, at each scale, the network has learned to ignore the input and has memorised the image, possibly because of the existence of L_{rec} (L_2) loss in the generator. The network cannot completely disregard the input in the case of the model with residual connections because, at each scale, the model will have to learn to add finer details to the coarser input.

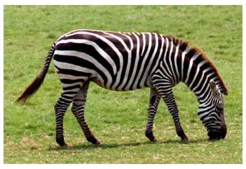
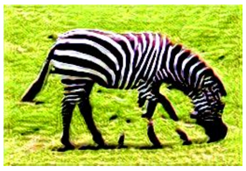
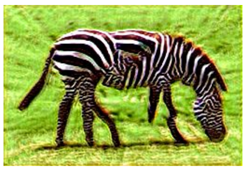
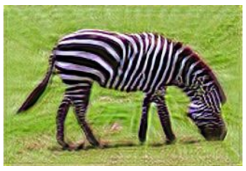
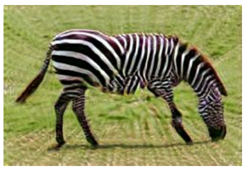
Experiment 2: Effect of L2 loss in the generator.

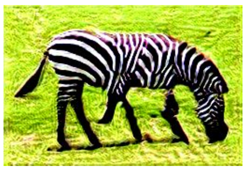
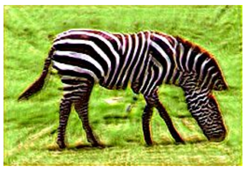
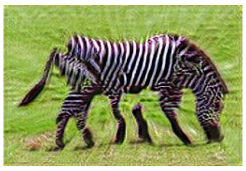
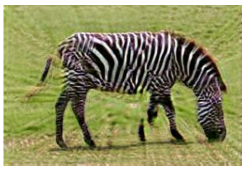
The original paper used a combination of L2 (*Lrec*) and GAN loss to train the generator. Here we investigate what added benefit the L2 loss yields.

$$Generator\ Loss = \min_{Gn} \max_{Dn} (L_{adv}(Gn, Dn)) + \boldsymbol{\alpha . L_{rec}(Gn)}.$$

In the paper, they say that the reconstruction loss ensures that there exists a specific set of noise maps that can produce the training sample.

In this experiment, we report the reconstruction quality for randomly generated samples after training for just 500 training steps per scale (as opposed to 2000 in the paper so that we can compare the performance before both of them starts producing better results) to compare the effectiveness of reconstruction loss.

Alpha α	0	0.1	1	10 (original)
Generated Samples from the following training image: 				

Results: We find that with the added L_2 reconstruction loss, the contrast of the generated images approaches to that of the training sample. Without the reconstruction loss, the generator was not able to generate images of the required contrast.

Possible Explanation: One explanation for this can be the discriminator which is trained to discriminate between image patches contains multiple batch normalization layers. So the contrast and brightness of the patches are not affecting the output of the discriminator.

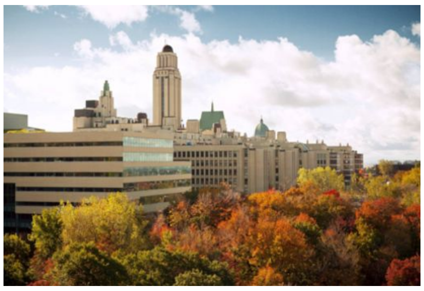
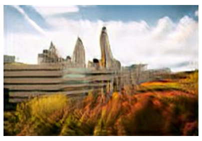
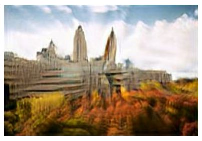
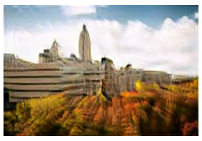
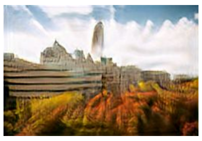
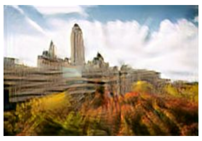
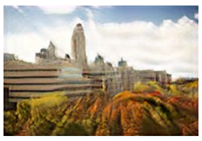
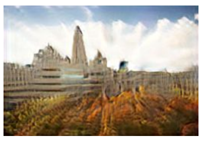
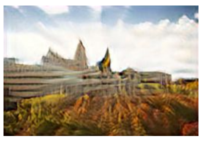
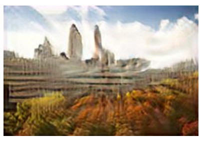
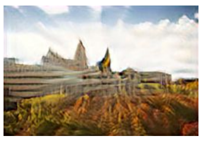
Experiment 3: Effect of texture loss.

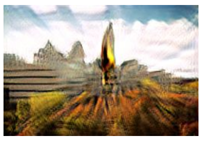
Since the network was doing a very good job of reconstruction of high-resolution textures from low-resolution coarse images, we tried to experiment by adding an extra texture loss term to the generator network. The new generator loss is,

$$Generator\ Loss = \min_{Gn} \max_{Dn} (L_{adv}(Gn, Dn)) + \alpha.L_{rec}(Gn) + \beta.L_{texture}(Gn)$$

$$Where,$$
$$L_{texture} = L_2(\,gram\_matrix(x_{gen}),\ gram\_matrix(x_{real}))$$

The gram matrix was calculated from intermediate activations of VGG16. A network was trained with the following settings with 500 steps per scale.

Relative Importance $R = \beta \times 10^{-6}$ R = 1 implies both losses are of the same order of magnitude	Generated samples from the training image : 				
0.00					
0.01					

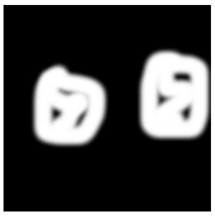
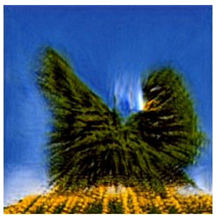
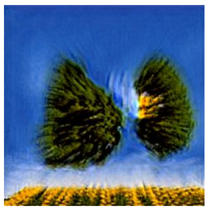
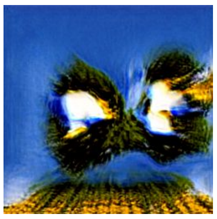
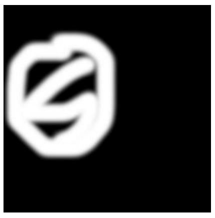
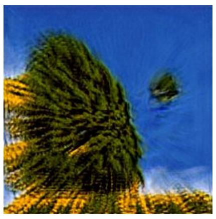
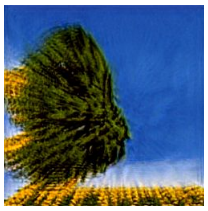
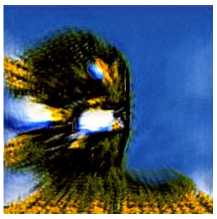
0.1					
1					
10					
100					

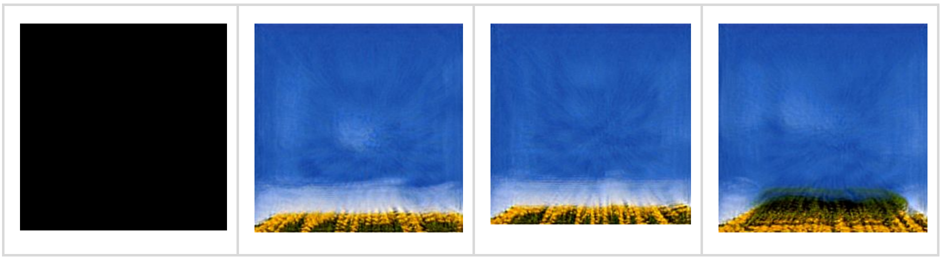
From the above results, adding texture loss does not improve the generated samples. The generated samples worsen when texture loss starts dominating. So it was a bad idea to include texture loss.

Experiment 4: Sketch to Paint.

Instead of training all the scales with the target real image patches, we tried to teach the network to draw sketches of images in the last 3 scales. The succeeding scales use these sketches as the input to paint the full image.
The results are not so convincing, but some of them are good.

Training Sample

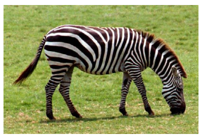
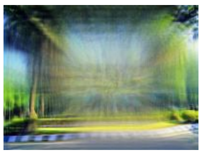
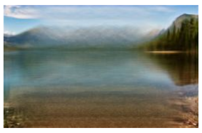
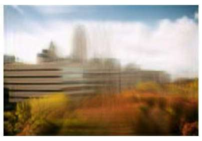
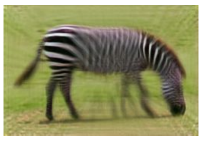
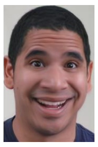
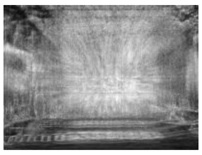
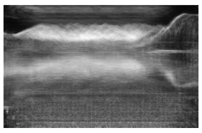
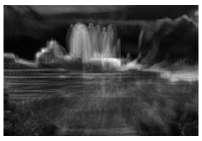
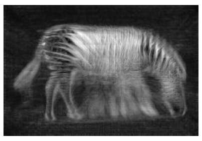
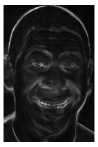
Input Sketch	Generation Scale N-1	Generation Scale N-1	Generation Scale N-1
			
			



We can see that at scale N-2, the results are slightly better, however not good enough.

Experiment 5: Spatial variations in generated samples.

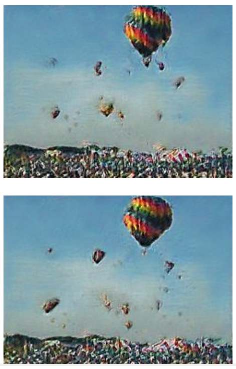
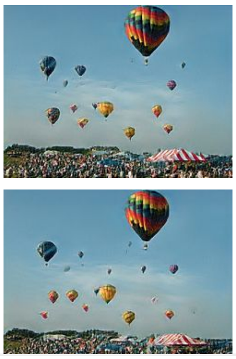
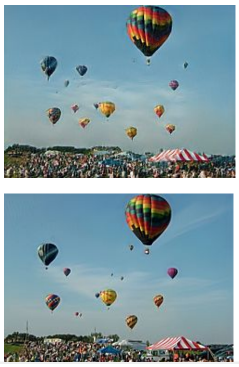
In this experiment, we explore the spatial variation of the generated samples.

Training Image					
Mean Generated Image					
Standard Deviation of Generations					

We can see that most of the variations lie on a particular region of the image with high semantic meaning. These regions the ones where a particular local texture has a spread distribution (like the zebra stripes) that are captured at the finer scales or can be many global structures (the mountain ranges) captured at coarser scales of generation. These regions with high variations correspond to regions of high 'information'. The generator memorizes other parts of the image as 'obvious'. More information is contained in the regions of high variances.

Experiment 6: Effect of scale factor/number of scales in the generation quality.

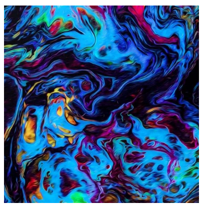
A network was trained on an image for different numbers of scales. The generated images are as given below.

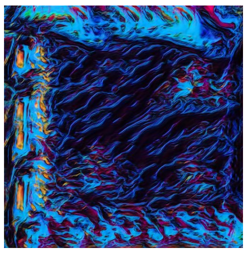
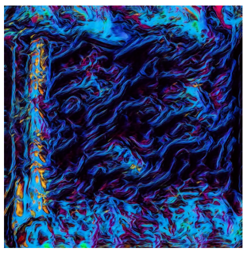
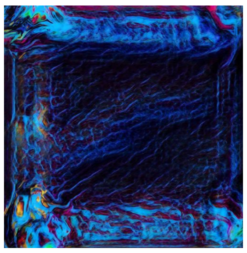
	Scale Factor 0.50		Scale Factor 0.75		Scale Factor 0.95

The quality of generation increases as we increase the scale factor because of the increase in the number of scales, further leading to a decrease in the amount of details to be generated per scale. An optimum value of 0.75 was chosen by the author to balance the training time and generation quality.

Experiment 7: Effect of changing the limits of the image pyramid for random generation.

In the generator pipeline, the given an input image is first resized to a min size. It is then scaled up by the scale factor until the max size is reached. In this task, random arbitrary samples of twice the sizes are generated by scaling to a factor of 2.
*i.e, Generated Output size = 2*max_size*


Training Image

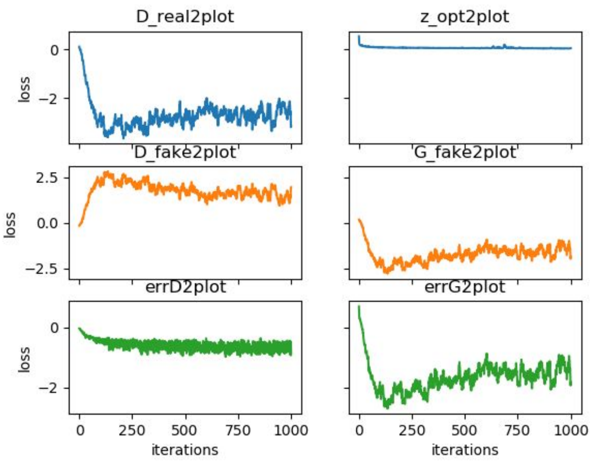
		
Min 25/ Max 250	Min 25/Max 520	Min 12/Max 250
10 Scales	12 Scales	13 scales

Generated Samples

Results: Given that the size of the image is 520 keeping its size as the max size helps in learning in more scales which helps in better output generation (25 to 520 by a factor of 0.75 total 12 scales). On the other hand, reducing min size nearly to the patch size of discriminator network makes it learn in more scales but towards the coarser region only. These features are not much helpful in generating random samples compared to those of higher scales. (12 to 250 by a factor of 0.75 total 13 scales). Useful in generating patterns of higher dimensions.

Inferences: Keeping max_size to be equal to image size always leads to better results but involves more scales and thus more training time. So a value of 250 is kept as max-scales to reduce training time. Reducing min size beyond a permissible value increases the training time but it leads to negative results as well. This is because when a high-resolution image is resized to a very small one a lot of features are lost and thus may be nowhere similar to original one and these features also will be learned.

Loss Analysis.



Legend

D_real2plot	Loss after passing the real image through the discriminator *{-D(x)}*
D_fake2plot	Passing generated image through discriminator *{-D(G(z))}*
Critic Loss:	*D(x) - D(G(z))* - Discriminator loss

z_opt2plot	Reconstruction loss
G_fake2plot	Passing generated image through discriminator *{D(G(z))}*
Generator loss	*D(G(z)) + RL*

G_fake2plot is *almost* negative to the *D_fake2plot*. Because the discriminator is updated before the second loss part. *D(x)* decreases with epoch which means that it learns real images more efficiently. *D(G(z))* has higher variations with epochs - It becomes difficult for the discriminator to identify real and fake with epochs. This shows that the generator is learning better compared to the discriminator.

Conclusion

Through this paper, the authors present a novel generative model that can be trained using a single image. A single model is capable of doing multiple tasks like random sample generation, paint to image synthesis, content-aware moving, etc. Results show that the generations of SinGAN are more realistic compared to other methods. This work presented has profound application in the image and video editing field. However, the training time for a single image on an NVIDIA Tesla K80 GPU is about 30 minutes. So, in the present form, SinGAN is not of any practical use unless we have extreme compute power available. However, the ideas present in SinGAN, including the use of a patchwise discriminator to learn the input image texture and structure distribution has profound implications for future studies.