cSinGAN - Improving Generative Model Using Conditional GAN and Segmented Data

Yunke Zhao, Zhilin Guo, Xuejing Wang
Department of Computer Science
School of Engineering and Applied Science
Columbia University
New York, NY 10027
`yz3831/zg2358/xw2668@columbia.edu`

May 2, 2020

Abstract

Image generation combined with deep learning has become a hot topic these years. Inspired by the SinGAN developed by Shaham et al., we implemented a new optimized conditional stack-GAN model, named cSinGAN, in this project to explore some more potential solution on image generation, and compare them with current models. In the end, our model sees FID score improvement on 4 out of 5 image inputs.

1 Introduction and Related Works

Unconditional Generative Adversarial Nets (GAN) has seen great success on generating high fidelity natural images. Particularly, training GAN on large data set [1] and employing orthogonal regularization to the generator allows fine tweaking between image quality and image variety, such tasks remains a significant difficulty and very often asks for conditioning image formulation on some specific task or signal.

More recently, Shaham et al. developed a new GAN named SinGAN [2] which trains on a single natural image and uses unconditional generation on the internal statistics and structures of that single image to generate natural images. Without the need for a large database or labeled data, SinGAN employs a set of connected fully convolutional GANs that captures the internal structures of an image at each different locations, and produces state-of-the-art natural images as results.

However, one limitation of SinGAN is that it only receives an unsegmented image when generating new images, and can lead to failure cases that can only be resolved by manually fine-tweaking at which scale the generation starts. Hence, we propose to introduce an improvement on SinGAN that employs conditional GAN (cGAN) that generates high-fidelity natural images.

2 Problem Formulation

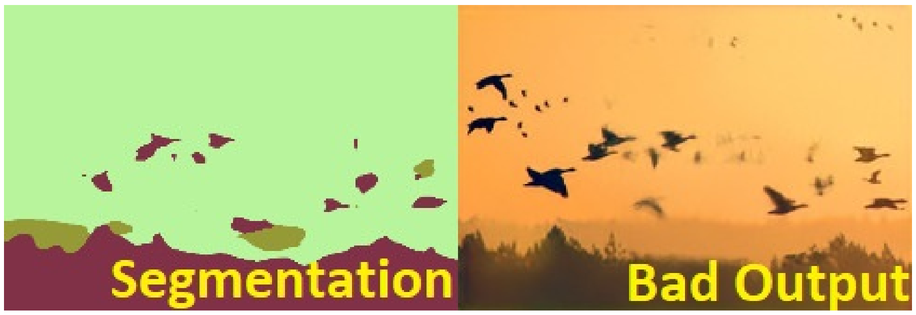
The goal of this project is to create and optimize a cGAN generative network model to learn the inner texture information of the given segmented image. SinGAN network gives us a good head-start on solving this problem. It uses single image to randomly generate training data by applying random image manipulations to the input without destroying the texture features. The model learns texture information and detect objects from the image and generate fake image from the giving input. Based on the SinGAN network, we will explore the possibility of combining a conditional GAN to the SinGAN network.

We propose to use the structure of the conditional GAN to refine the model. The current model contains similar structure of stack-GAN, which is start from generate small size of images and each time add more details to it by using a higher layer of GAN sturcture with the output of the past layer as the input of the current layer. For each layer, the input from the last layer performs as the "conditional" label, which limit the generation of the current layer. But the conditional GAN like structure is only applied on Generator, for the Discriminator, it still works as a normal GAN by deciding whether the output of the generator is sufficient. So we focus on changing the structure to refine the entire model.

3 Methods

3.1 Previous Method and it's shortcomings

As we described in the problem formation section, we focus on finding a we to convert the model into a "conditional" type of the GAN. We have focused on using image segmentation as the label before, but find out that it is a not good representation of a "label" we need. As we described in the milestone report, we found out that by using the segmentation model, we are limiting the dynamic structure of the image, especially when we trying to use the image segmentation as a labeled input of the Generator, we found out that it will conflict with the original input from the lower layer of the pyramid. The input from the lower layer is somehow different from the original image because it is generated, it supposed to have some dynamic difference from the original input and that is acceptable. But when we forcing the generator to limit the output from the image segmentation, it not only learned that it need to fix the position of the object, but also lose the ability to trying to generate real image that is different from the original image. We need to find a better way to implement a way to restrict the generated image to have the specific objects in it but not restrict the dynamic position and possibility.



Unsatisfying image generation with segmentation data

3.2 Our Method: A solution from a different view

To solve the problem above, we start from a different view: Trying to remain the structure of the Generator, but make Discriminator more "conditional" like. The original Discriminator of the SinGAN first train on real image and than only accept the output from the Generator as a input and decide whether it is real or not. We are trying to make it more complex to not only check if the input is real or not, but also can decide whether the input has the similar structure of the original real image. To achieve this proposal we implement two different discriminator.

Discriminator Type One remain the one input structure, but instead, pre-process the input to gain the ability to compare the generated image and the real image by concatenate the real image and the generate image to make it a 6 channel input.

Discriminator Type Two change the structure of the discriminator. Now, it first check through the image to decide whether it is real, then go through the real image to check the difference between the real and the generated one. The two output will be combined together as the final output of the Discriminator.

In the Github repo we used the type two discriminator since it has slightly better output.

4 Architecture

The GAN that we implemented is basically a pyramid of fully convolutional light-weight GANs, each is responsible for capturing the distribution of patches at a different scale. Both training and inference are done in a coarse-to-fine fashion. We start from the lowest scale with smallest output size, and each scale will adding more details to the output while increasing the output size. The effective patch size for the patch Discriminator decreases as we go up the pyramid.

In General, the general stack-GAN structure will be remained as shown above. And for each layer, will remain the same structure that is used in SinGAN:

$$\min_{G_n} \max_{D_n} \mathscr{L}_{adv}(G_n, D_n) + \alpha \mathscr{L}_{rec}(G_n)$$

But the D_n is modified to merge with condition on the input, where $G_n$is a special multi-scale generator that generate the output:

$$x_n = G(\bar{z_n}, (\bar{x}_{n+1}) \uparrow^r)$$

For the D_n, we will add another input s_n works as a label:

$$y_n = D(\bar{x_n}, s_n)$$

Where y_n and $\bar{x_n}$ are same with SinGAN, and s_n are the real image of the current scale.

5 Result

Images outputted by Generative Adversarial Networks can be difficult to evaluate. Different from regular deep learning networks, a GAN does not attempt to converge using a loss function but instead employs a generator and a discriminator together to create new images [4]. Hence, we cannot quantitatively evaluate our GAN using its loss function. Also, since our cSinGAN model attempts to create synthetic natural images from an input image, it is challenging to compare and evaluate the results using naked eye or other manual evaluation criteria. As a result, we decide to use Fréchet Inception Distance (FID) as a quantitative metric.

5.1 Fréchet Inception Distance (FID)

Fréchet Inception Distance compares feature vectors and the distance between them for natural and synthetic images [5]. In Fréchet Inception Distance, features are extracted using an Inception network from a middle later, and then modeled by data distribution as a Gaussian distribution [6]. Name the mean u and covariance , and Tr sums up Diagonal Elements, then the Fréchet Inception Distance between the input image x and synthetic image y can be calculated as

$$|| u_x - u_y ||_2^2 + \mathrm{Tr}\ (\Sigma_x + \Sigma_y - 2\ (\Sigma_x\ \Sigma_y)^{0.5})$$

And as the formula would suggest, the lower the FID score the better.

As a result, FID is very suitable for noisy images and can be a great measurement for diversity of model outputs, as Fréchet Inception Distance has comparatively small variance but large bias.

5.2 Our implementation

Fréchet Inception Distance aims to discover and compare how much real images and generated images differ. Out implementation employs the inception v3 model, whose coding layer captures the image's computer-vision-specific features

[5]. The activations of real and generated images are calculated as Gaussian model with mean and covariance like mentioned above, and the distance between the two images are then calculated using Fréchet Distance.

Our FID implementation iterates over all 50 images generated by our own GAN model, and outputs an average FID score.

5.3 Output evaluation

Our model sees improvement on 4 out of 5 input images. On birds.png, we observe an increase in FID score, which means that the model did not make better images. We suspect that the cause is that features in birds.png are too small and far-in-between, hence our model did not improve upon the previous SinGAN model.

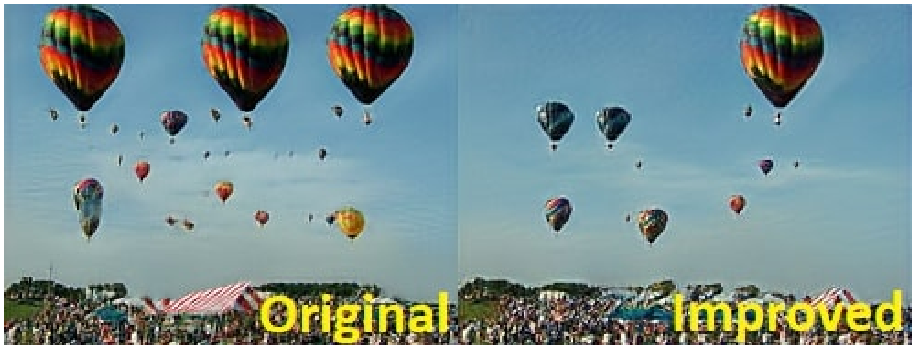
Here are our results:

Image name(.png)	original sinGAN FID	our cSinGAN FID
ballons	3.70213	3.46162
birds	0.23415	0.28225
zebra	9.60776	9.45568
colusseum	0.33532	0.29928
starry_night	0.09001	0.07597

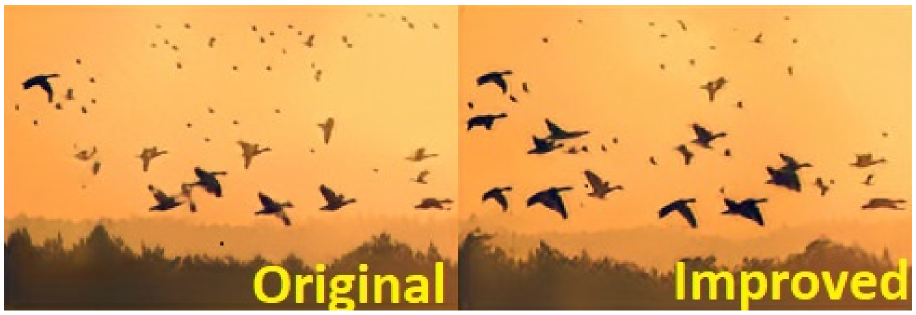
5.4 Noteworthy generated images

Here are some example images generated by original sinGAN model and our cSinGAN model.

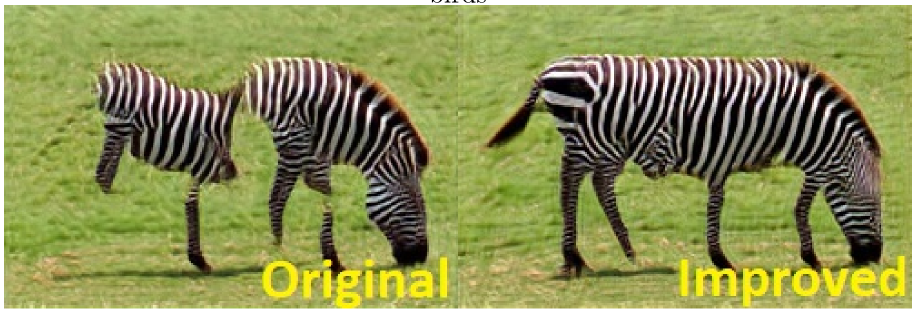
Image produced by original model is placed on the left, and image produced by our improved model is placed on the right.



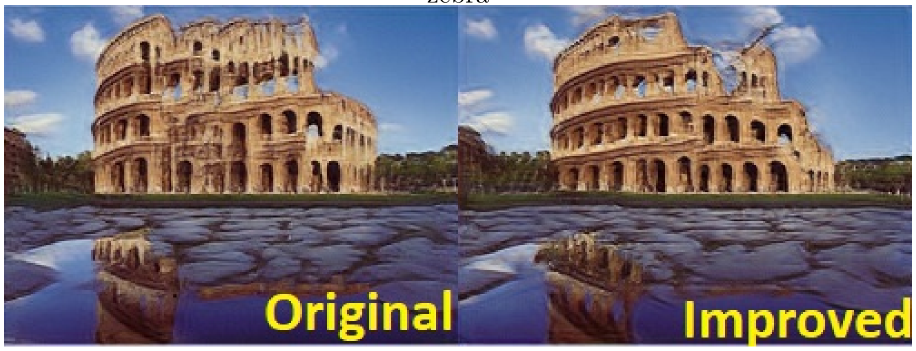
balloons



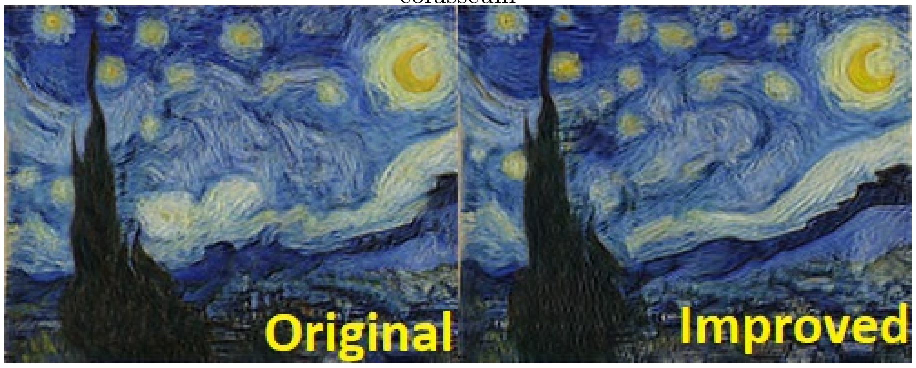
birds



zebra



colusseum



starry_night

6 Implementation

6.1 Implementation Details

Drawing inspiration from Hung-yi Lee's online lecture on conditional generation [7], we implemented our conditional SinGAN using PyTorch.

The Fréchet Inception Distance (FID) evaluation component is separately implemented using TensorFlow.

You can find our source code here:

Github repo: https://github.com/zg2358/cSinGAN

6.2 Training Time

Original SinGAN model has an estimated training time of 30 minutes using Nvidia 1080Ti Graphics card, and image generation time of less than 1 minute for each image.

We observed an estimated model training time of 2 hours using Nvidia GTX 1070 MaxQ, and comparable image generation time.

FID calculation takes about 1 hour for each input image and its 50 generated images using TensorFlow with CPU setup.

References

[1] Brock, A., Donahue J., & Simonyan K. *Large Scale GAN Training For High Fidelity Natural Image Synthesis.* OCLR 2019.

[2] Shaham, T.R., Dekel, T., & Michaeli, T, *SinGAN: Learning a Generative Model from a Single Natural Image.* ICCV 2019.

[3] Martin D., Fowlkes C., Tal D., & Malik J., *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics.* page 416. IEEE, 2001.

[4] Brownlee J., *How to Evaluate Generative Adversarial Networks.*, Machine Learning Mastery, August 26, 2019

[5] Brownlee J., *How to Implement the Frechet Inception Distance (FID) for Evaluating GANs.*, Machine Learning Mastery, August 30, 2019

[6] Hui J., *GAN — How to measure GAN performance?.*, Medium, Jun 18, 2018

[7] Lee H., *GAN Lecture 2 (2018): Conditional Generation.*, Youtube, May 11, 2018