SecleGAN: Segementation of CycleGAN

Jin Wu
UC San Diego
9500 Gilman Dr, La Jolla, CA 92093
gew013@ucsd.edu

Vince Chen
UC San Diego
9500 Gilman Dr, La Jolla, CA 92093
bic015@ucsd.edu

Abstract

In the field of Deep Learning, Generative Adversarial Networks, or GANs, has gained wide success over the pass few years, and one of the notable extension of GANs model is Cycle-Consistent Adversarial Networks, or CycleGAN. which the algorithm has been practical in solving image-to-image translation problem. However, despite the fact that CycleGAN algorithm gained a lot success of in transforming images, it also has some limitations and failed some cases. In this paper, we would like to propose a new upgrade from the orginal CycleGAN that we combine segmentation with the CycleGAN to potentially extend and improve the performance of the CycleGAN algorithm. We call our upgrade as SecleGAN algorithm, which is a combination of segmentation and CycleGAN.

1. Introduction

Nowadays, CycleGAN algorithm has been broadly applied to many different fields. For example, the algorithm has been used in the Medical diagnosis field to help with the recognition of Lesion and tumor because of its excellent performance augmentation, which it could recognize and pick the lesion part in a CT scan image. Moreover, image-to-image translation is another useful application of CycleGAN, which it could be used into image reconstruction. For example, Using CycleGAN, Jack Clark was able to convert ancient maps of Babylon and London into modern Google Map satellite views. What's more, by transforming the domain into target domain, CycleGAN could also render real word sceneries into computer graphics style. Therefore, CycleGAN has much potential that is worthwhile to further optimize and improve. Even Yann LeCun, the God father of the modern deep learning, once said that "GANs is the most interesting idea in the past 10 years in machine learning". Therefore, we think diving deeper to the field of GANs, especially CycleGAN, is a meaningful try.

1.1. Problem

However, even the finest jewelry will not be perfect, just like the CycleGAN algorithm. Although it has much potential and proved practical in real world applications. It could sometimes generate failed results. For example, when doing horses-to-zebras transformations, if a person is riding the horse, there is a chance that the person will also be transformed into zebra, thus resulting a failure case.

Figure 1: A misclassified example by CycleGAN

The above result is definitely not the one that we expect from CycleGAN. Therefore, we would like to rectify this mistake generated by CycleGAN algorithm and hope to successfully transform the horse into zebra without changing the person at the same time.

1.2. Proposed Solution

Therefore, in our paper, we propose a new method that combines semantic segmentation and CycleGAN, which we call SecleGAN algorithm, that could perfectly solve the above problem that CycleGAN misclassify the background content. Specifically, we would first perform semantic segmentation and pull out the object instance out of the background content, and we then use the traditional CycleGAN algorithm to transform our object instance into the target instance. Finally, we would use a mask to put the transformed

instance back to the original background content. In this way, we would only transform the instance solely instead of the whole image so that we could avoid having the background content transformed as well.

2. Related Work

2.1. GAN

A generative adversarial network, also known as GAN[1], has two key components, the first component is what usually referred as generator network, which takes a random input and produce corresponding output sample, then together with the real input or sample label, they are fed into the second component of GAN model, a discriminator. Discriminator network would take the sample generated by the generator and the real sample to produce a binary classification result of either true or false, or real or fake, standing for whether the generator produce result that is similar to the real sample or not. If not, through backpropagation process, the discriminator would provide weights that the generator could use accordingly to generate samples that are more similar to the real sample.

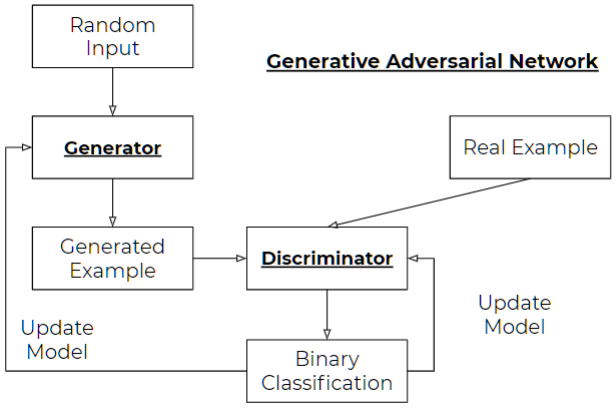

Figure 2: GAN architecture

2.2. CycleGAN

As described above in Introduction, CycleGAN[5] is a extension of GAN architecture.

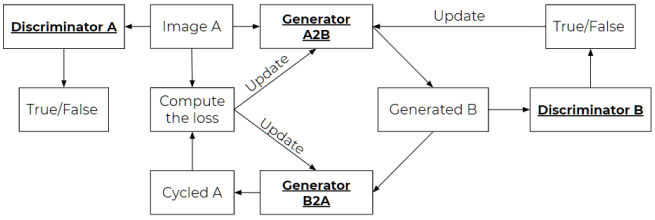

Figure 3: CycleGAN network architecture in forward cycle

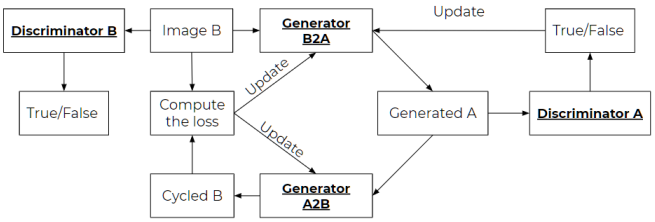

Figure 4: CycleGAN network architecture in backward cycle

The CycleGAN architecture contains two separate GANs running in cycles. The two cycles are denoted as forward cycle and backward cycle. In forward cycle, the input image from domain A will be used to train the discriminator A, then it's transformed into domain B, the target domain, by the generator A2B, which the result we denote as transformed B image. Unlike the tradition GAN model which constantly update the generator using weights, the transformed B image will then be transformed back to domain A by another generator B2A. And the result that comes out of generator B2A is denoted as reconstructed A image. Lastly, the CycleGAN would compare the reconstructed A image with the original A image to compute the loss and update the model.

In backward cycle, same logic is applied for the image in domain B. Just like to forward cycle, backward process transform the B image into domain A, then transformed again into domain B to acquire reconstructed B image. By comparing reconstructed B image with original B image, the algorithm again compute the loss and update itself using the loss information.

2.3. Pix2Pix

Pix2Pix[2] is another popular technique that is used commonly in image2image transformation. However, there is a limitation about it. This technique requires parallel data. For example, if we want train this generative model to transform a landscape picture from winter to summer. We need to have images of the same landscape in both summer and winter. It seems not a big deal in this case. However, if we want to train anther model to transform a horse to zebra. We won't be able to obtain this parallel data, since it is impossible to get the zebra pattern to show on a horse in real life. Despite the requirement on training data. Pix2Pix is still a powerful generative model. This model indeed still have the same problem as Cycle GAN, which it also cannot differentiate between the instance and the model itself.

3. SecleGAN

3.1. Structure

SecleGAN is our proposed method which combines the semantic segmentation together with the traditional CycleGAN algorithm. The SecleGAN architecture is shown below.

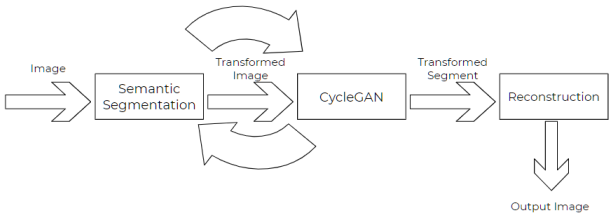

Figure 5: SecleGAN architecture

Note that there are two different ways to implement our SecleGAN algorithm. First, we could feed the input image into the semantic segmentation block, which in our paper we used U-net to perform the segmentation part. Then after the segmentation process, we would have the segmented instance, we then feed the instance into the CycleGAN algorithm to acquire the transformed target segmented instance. Finally, with the help of the mask, we are able to put the transformed target segmented instance back into the original background as the output image.

The other way would be that we could first feed the input image into the CycleGAN algorithm, and use the CycleGAN to transform the entire input image into the other domain. Often, even though the instance is successfully transformed into the target instance, the background content would be transformed as well. Therefore, in order to avoid this, we then use semantic segmentation technique to pull out the target instance, and then apply mask to move the instance back to the original background content. In this way we could get a reconstructed image that has successful instance transformation and , at the same time, with the original background content. SecleGAN technique would prevent background information loss and could potentially have a huge impact because it greatly improve the overall performance of the traditional CycleGAN algorithm.

3.2. Formulation for Cycle GAN

Again, our goal is to develop two generators for mapping between each domain. In order to explain the algorithm well, All the variables and functions name that will be used is shown in the table below.

Name	Meaning
G_{A2B}	Generator A to B
G_{B2A}	Generator B to A
D_A	Discriminator A
D_B	Discriminator B
a_i	i^{th} sample from set a
b_i	i^{th} sample from set b
x	random input vector for G_{A2B}
y	random input vector for G_{B2A}

Table 1: Variable names

For each iteration, the four models are updated, G_{A2B}, G_{B2A},D_A, D_B. We will be updating those model by minimizing the losses. The loss for Cycle GAN is consisted of two part. First, the the loss of the GAN, and we call it L_{GAN}. Since in the Cycle GAN algorithm there are two GANs, we will have two L_{GAN}. Second, the loss of the cycled image, L_{cyc}. They are defined as:

$$L_{GAN}(G_{A2B}, D_B, x, b) = E[log(D_B(b)] + E[log(1 - D_B(G_{A2B}(x)))] \tag{1}$$

$$L_{GAN}(G_{B2A}, D_A, y, a) = E[log(D_A(a)] + E[log(1 - D_A(G_{B2A}(y))] \tag{2}$$

$$L_{cyc}(G_{A2B}, G_{B2A}) = E[||G_{B2A}(G_{A2B}(x)) - x||_1] + E[||G_{A2B}(G_{B2A}(y)) - y||_1] \tag{3}$$

Note that the losses are depends on multiple parameters. Again, in Cycle GAN we are combining the losses above. So the total loss is $Eq.(1) + Eq.(2) + Eq.(3)$:

$$L(G_{A2B}, G_{B2A}, D_A, D_B) = L_{GAN}(G_{A2B}, D_B, x, b) + L_{GAN}(G_{B2A}, D_A, y, a) + \lambda L_{cyc}(G_{A2B}, G_{B2A}) \tag{4}$$

λ is the weight factor to balance between losses. Now we have defined our loss. Next, we will use the following equation to update our model.

$$G^*_{A2B} = arg \max_{G_{A2B}} L(G_{A2B}, G_{B2A}, D_A, D_B) \quad (5)$$

$$G^*_{B2A} = arg \max_{G_{B2A}} L(G_{A2B}, G_{B2A}, D_A, D_B) \quad (6)$$

$$D^*_A = arg \min_{D_A} L(G_{A2B}, G_{B2A}, D_A, D_B) \quad (7)$$

$$D^*_B = arg \min_{D_B} L(G_{A2B}, G_{B2A}, D_A, D_B) \quad (8)$$

See that the discriminator and the generator are working against each other. The generator is trying to minimize the loss function, and the discriminator is trying to maximize the loss function. The generator is minimizing the difference between the generated image and the real image, while is the discriminator is maximize that difference so that it can distinguish better.

The overall algorithm is defined and stated below:

Algorithm 1 Cycle GAN's algorithm

1: **for** `Not Converge` **do**
2: `Update` G_{A2B}
3: `Update` G_{B2A}
4: `Update` D_A
5: `Update` D_B
6: **return** G_{A2B}, G_{B2A}

A more detailed description of the above formulation can be learned from reference [5].

3.3. Formulation for Secle GAN

In our proposed Secle GAN algorithm, there is a slight difference in defining and training the discriminator and the generator. It will be trained only for learning the specific target domain instance, excluding the background and other no-related information. Now instead of calling them domain A and B, we will defined the domains with instance only be $A^{'}$ and $B^{'}$. We are using semantic segmentation(U-net)[3] to select the instances and differentiate them with the background. The updated algorithm procedure is shown here.

Algorithm 2 Secle GAN's algorithm

1: **for** `Not Converge` **do**
2: `Update` $G_{A'2B'}$
3: `Update` $G_{B'2A'}$
4: `Update` $D_{A'}$
5: `Update` $D_{B'}$
6: **return** $G_{A'2B'}, G_{B'2A'}$

Note the dataset is pre-processed by U-net[3] here. However, we can train the original Cycle GAN as defined in previous section, and use U-net to segment the transformed image. The two approaches are equivalent. In both approaches, we can reconstruct the image using the transformed instance and the original image.

4. Experiment

4.1. Butterfly Dataset

To test out our proposed improvement, Secle GAN. We are using the dataset found on Kaggle[4]. There are image dataset and segmentation masks labeled for ten different species of butterfly. The two kind of butterfly that we used in this experiment is "Danaus plexippus" and "Nymphalis antiopa", containing 65 and 100 images respectively. Note the difference between number of training data actually make a difference in our resulted generator models.

(a) Danaus plexippus

(b) Nymphalis antiopa

Figure 6: Butterflies images from the dataset

4.2. Models and Parameters

The two models we used is Cycle GAN[5] and U-net[3].

In this experiment, we are implementing U-net before Cycle GAN. Similar result can be achieved with the re-

(a) Result of A to B

(b) Result of B to A

Figure 7: Cycle GAN result

versed order. The images and segmentation masks are first used to train the U-net for it to learn how to segment butterfly out of the image. Next, the images are put into Cycle GAN to train. Then, we use the predict mask to segment out the background and the butterfly. Last, we put everything together and reconstruct the image.

The parameters for Cycle GAN training involves λ, which is the weights factor that balance between L_{GAN} and L_{cyc} as defined previously. However, in order to achieve better performance and explore, we have introduced an identity loss and weighted the three losses, instead of two. The three losses are

Notation	Meaning
L_{GAN}	GAN loss(L_2)
$L_{identity}$	Identity loss(L_1)
L_{cyc}	Cycle loss(L_1)

Table 2: Three kinds of losses been weighted

After trails and evaluation with cross-validation, the weightsused in this experiment is $[1, 5, 10]$, respectively. So the overall loss is defined as:

$$L = L_{GAN} + 5 * L_{identity} + 10 * L_{cyc} \tag{9}$$

Identity loss is computed by compare the original image from A and transformed images from A to A, using $G_{B2A}(A)$.

In addition to weight factors, the learning rate is also important. We found out it is very easy to miss the local minimum/maximum if a larger step is taken. The loss function will bouncing around, instead of "monotonically" decreasing. After trails with

$$\begin{aligned}\eta \in &(0.0001, 0.0002, 0.0005, 0.001,\\ &0.002, 0.005, 0.01, 0.02, 0.05, 0.1)\end{aligned} \tag{10}$$

we set the learning rate to $\eta = 0.0002$ eventually. Note that the learning rate is small because the loss is very sensitive to learning rate.

4.3. Performance

As we mentioned above, U-net was trained the first. The end accuracy was around 0.97 after 20 epoches. The accuracy and loss curve is shown.

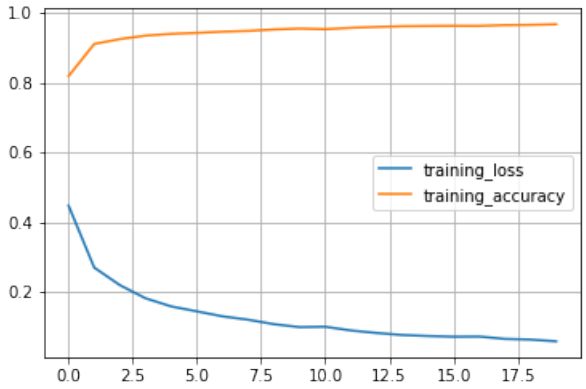

Figure 8: U-net accuracy/loss Plot

The performance of our Secle GAN algorithm is shown in the next page. Note there the background was not transformed, only the butterfly. The result shown is at 200

(a) Original Image

(b) Cycle GAN result

(c) Secle GAN result

Figure 9: Result of A to B

(a) Original Image

(b) Cycle GAN result

(c) Secle GAN result

Figure 10: Result of B to A

epoches. The loss was not minimized yet. The result could be better, but we are limited in time and computational constraint. However, the result is optimising. We can see the orange butterfly is turning black, while the black butterfly is turning orange. In our dataset, there are more black butterfly data than orange butterfly, as mentioned above. Therefore the discriminator is trained better, hence the generator. We see this in our result. The overall performance of orange to black is better than the performance of black to orange. We see the effect of generator and discriminator compete with other, causing both to learn better.

4.4. Baselines comparisons

The baseline that we used for compare and evaluate our solution is the original Cycle GAN. The result of Cycle GAN is shown in Figure 5 and Figure 6. We see the problem of the Cycle GAN, which the background got transformed. However, in our Secle GAN result, the background information is maintained. The images is more vivid, and the improvement is obvious. Therefore, Secle GAN is definitely a improvement upon Cycle GAN.

5. Reflection and Future Improvement

We made one mistake in this experiment. The input size for Cycle GAN and U-net was not consistent. Input size was 128x128 for Cycle GAN, and U-net 256x256. This caused us troubles during reconstruction and made the resulting images blurry as we can see in Figure 5 and Figure 6.

There are some places that can still be improved or application to be build upon.

- Use consistent input size for two models.
- Train till the losses converge.
- Video to Video transfer.
- Use instance segmentation, instead of semantic segmentation.

In this experiment, we used U-net for semantic segmentation. Because we are transforming one single butterfly in our image, so there is no difference between instance and semantic segmentation. However, when the image contains more than one butterfly, they will make a difference. If we use semantic segmentation, the pattern might be continue in the overlapping area of different butterflies. To avoid this,

we need to transform each butterfly separately, and reconstruct them at the very end. This can be achieved by instance segmentation, but not semantic segmentation.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[4] Josiah Wang, Katja Markert, and Mark Everingham. Learning models for object recognition from natural language descriptions. In *Proceedings of the British Machine Vision Conference*, 2009.

[5] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.