KAIST

Paper 1

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Student : David Albert

Contents

1 Abstract

I chose to study the paper *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* [1] because I was interested to know more about Generative model and more specifically Generative Adversarial Networks. This is a relatively recent paper which uses GANs to learn to translate from a source domain X to a target domain Y. The authors apply their model on datasets of images. In this report we will explain what the authors of the paper did to achieve interesting results in this area. We will briefly see related works to this paper which could be seen as a baseline. Then, we will present our implementation and results. Finally we will have some discussion about how to improve the model.

2 Paper presentation and related works

One topic related to this paper because this paper use some of these tricks is the Generative Adversarial Networks (GAN). Ian Goodfellow introduced GAN in 2014 and since, this model have been strongly studied. In fact, generative adversarial networks demonstrates to be able to achieve impressive results in image generation. Recent methods apply the same idea for conditional image generation, such as text2image [3] or StackGAN [4] which use a text as a condition for the output. We can also consider an image as a condition for the output of the generative network. The $pix2pix$ approach [2] uses conditional generative adversarial network to learn a mapping from input to output images and thus achieves image-to-image translation task.

Pix2pix and CycleGAN approach

The main problem of the $pix2pix$ approach is that it requires to have access to paired data. It is not always easy to find such paired data. For exemple, it is easy to get images of horse and to get images of zebra but it is difficult and even impossible to get images of horse and paired images of zebra, which means at the same location with the same orientation. $CycleGAN$ paper introduce a way to train two generative adversarial networks to perform this task. $CycleGAN$ is closely related to the $pix2pix$ framework of Isalo et al. [2]. However, there is two major differences between the two models.

The first difference is that the $pix2pix$ paper employs an architecture for the generator called U-Net whereas $CycleGAN$ uses an architecture composed of two stride-2 convolutions, several residuals blocks and and two fractionally-strided convolution with stride $\frac{1}{2}$.

The second and main difference concerns the loss function. In $pix2pix$, authors use two losses for the generator. One is an adversarial loss to make the generated examples similar to the true distribution and the other is a $\mathcal{L}_1$ loss to evaluate the difference between the image generated and the one from the true data distribution. In $CycleGAN$ we cannot use the second loss because we do not have paired data. It occurs that the model is sub-constraint. In theory, an adversary training can be sufficient to translate from a domain X to a domain Y. In this case, we could consider train the model for learn to translate both from X to Y and from Y to X separately but it doesn't guarantee that an input image and an output image are paired in a meaningful way. To face this problem authors make an assumption. They consider that if G is the learned mapping from X to Y and F is the learned mapping from Y to X, then we should have a bijection between the domains. Mathematically, it means that we should have $\forall x \in X,\ F(G(x)) = x$ and $\forall y \in Y,\ G(F(y)) = y$.

This previous assumpiton lead to a *cycle consistent loss* given by :

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} \left(\|F(G(x)) - x\|_1 \right) + \mathbb{E}_{y \sim p_{data}(y)} \left(\|G(F(y)) - y\|_1 \right)$$

Thus, in addition to the two adversarial losses given by:

$$\mathcal{L}_{adv}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)} \left(log(D_Y(y)) \right) + \mathbb{E}_{x \sim p_{data}(x)} \left(log(1 - D_Y(G(x))) \right)$$

We can write the full objective as :

$$\mathcal{L}\left(G, F, D_X, D_Y\right) = \mathcal{L}_{adv}(G, D_Y, X, Y) + \mathcal{L}_{adv}(F, D_Y, Y, X) + \lambda \mathcal{L}_{cyc}(G, F)$$

3 Implementation details and results

The authors of the paper provide a Pytorch implementation of the model. I personnaly chose to do my own implementation using tensorflow 1 (version 1.15) using my coding style. The model wasn't very difficult to implement because the authors give detailed explanation of every part of the network architecture and losses used. The main difficulty was to accuratly choose the hyper-parameters because I quickly noticed that GANs are difficult to train and that they are very sensitive to hyper-parameters. In fact, even if the authors provide their own parameters, it is important to understand the role of each of them to be able to adapt in case where results are not decent. Therefore, I focused my attention to find good parameters. I kept the same neural network architecture consisting of two 70x70 PatchGAN and two generative models composed of residual blocks.

3.1 Download datasets

The first step before being able to train our model was to download the necessaries datasets. To this end I followed authors' instructions and downloaded art images from Wikiart.org. The way to download these images was not clearly specified so I used the two following UNIX commands to be able to get the filenames of all the images from the website (example in the case of images of Monet from Wikiart.org website) :

```
# Get page of all Van Gogh works
wget -q "https://www.wikiart.org/en/monet/all-works" -O page.html
# Get the filenames of the images
grep "https://www.wikiart.org/en/monet/" page.html \\
    | cut -d \" -f 2 > links.txt
```

Thus, thanks to all these filenames I was able to write a short code in Bash to download these data.

Concerning the other datasets I used (as mentionned in the paper) the Flickr API to download some of them (ie landscapes images) and directly find the other in public webpage.

3.2 Regularization

The original paper does not mention any regularization procedure. However, regularization like data augmentation proved to be essential in order to improve generalization. In fact, one problem that I noticed is that it is difficult to get a large number of data for some application. For instance, getting more than 500 images of Van Gogh paintings is not an easy task. To face this problem, I applied some data augmentation tricks to avoid the model to overfit due to the lack of data. At each step, after getting the next batch of data, I applied some randomness to the data. Particularly I used the following tricks.

Cropping data randomly

By cropping the original image we allow the network to not overfit and enforce the generator network to learn translation invariance. In order to not change the content of the image, I cropped it only a few (usually 286×286 to 256×256 pixel or 145×145 to 128×128 pixels).

Blurring images randomly

The second change over data that I did was to make them more or less blurred. We do not want to change a lot the original image by blurrying to much. I then blurred the image thanks to a filter of size x sample from uniform $\mathcal{U}_{\{0,..,n\}}$ where n is a samble from $\mathcal{U}_{\{0,..,6\}}$.

Changing brightness

I then changed lightly the brightness of the image by adding to each pixel of the image a random noise following $\mathcal{N}(0, 0.02)$.

Mirror changes

Finally I applied with probability 0.5 a mirror effect to the image.

4 Tests and Results

I tried to mofidy several parameters to make the model performing better but it wasn't an easy task.

4.1 Testing by changing parameter λ_{cyc} on zebra/horse dataset

The parameter that I more focused on was λ_{cyc} which is a factor of importance to the cycle loss. In the original paper they advise to use $\lambda_{cyc} = 10$ for each dataset.

I tried three values of λ_{cyc} to best understand the influence of this parameter on the model. I got some unexpected results that I tried to understand as well as possible. In each of these case, the other parameters were fixed. I used batch normalization, a batch size of 1 and a learning rate of 0.0002. The dimension of the images was 128px$\times 128$px.

Ghost-horse $(\lambda_{cyc} = 10)$

First, this is some results that I get after training the model on horse2zebra dataset with the parameter λ_{cyc} advised in the paper. It is quite fun because at first sight it seems that the generator learnt to change the color of horses in white which could be seen as a good start to learn the translation to the color of the skin of the zebra.

Figure 1 – Results for $\lambda_{cyc} = 10$
Right: original image - Center: generated image - Left: reconstructed image

Compute identity function $(\lambda_{cyc} = 8)$

I then tried with $\lambda_{cyc} = 8$ to see the difference. I was quite unhappy to see that it produced no results. Basically, the generators learned to produce the identity function.

Figure 2 – Results for $\lambda_{cyc} = 8$
Eight images of generated horse from zebra

One way to interpret this result is to consider that at this stage of training process, the discriminators weren't still well trained and so wasn't able to extract the specific features of the two distributions. It is not easy to know if it is the discriminator which failed to distinguish samples or the generator which failed to produce realistic samples. To best understand this result, I used the tools Tensorboard to visualize both loss function.

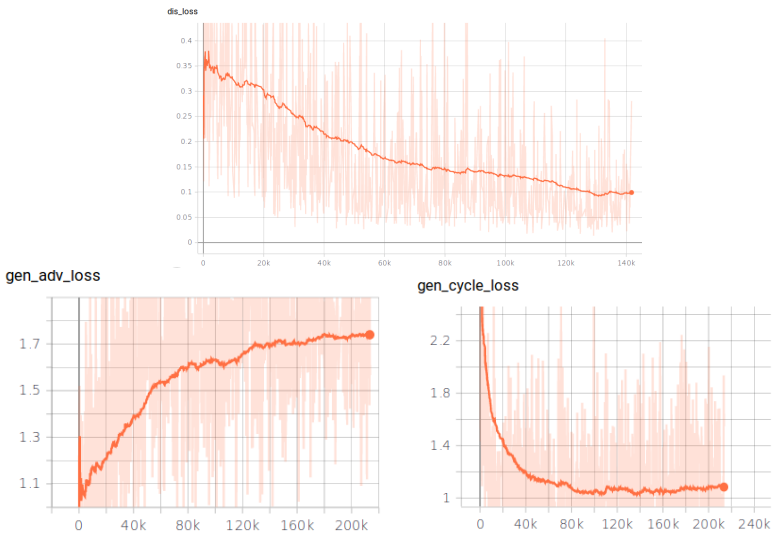

Figure 3 – Discriminatice loss (Up)- Adversarial loss (Left) - Cycle consistent loss (Right)

The loss of the discriminator seems to decrease seriously whereas the adversarial loss of the generator increase. In this case we can conclude that the discriminator have learnt the difference between the distributions. In the other hand, the generator is not able to generate realistic sample in order to fool the discriminator. The setting ($\lambda_{cyc} = 8$) is the best of the three because in our case there is two possibilities. Either the adversarial loss of the generator will decrease later or we have to slow down the learning of the discriminator.

Fall colors ($\lambda_{cyc} = 15$)

Finally, I tried as a last setting $\lambda_{cyc} = 15$. I quickly understood that it wasn't a good setting choice. In fact, in this case the generator get too high penalties for not being able to make a good cycle consistency. As a result, the generator will be less focus on minimizing the adversarial loss. We could first thing that in this case the generators should compute the function identity as in the previous example but it is not necessary. Typically, they will only transfer the color of each part of the image to another color (green to red for instance).

Figure 4 – Results for $\lambda_{cyc} = 15$
Right: original image - Center: generated image - Left: reconstructed image

4.2 Other tests

I tried different other settings like changing batch size and learning rate, using color regularization as mentioned in the paper for paintings data or use different form of the adversarial loss but it didn't make significantly better results on this dataset. I also trained differents configuration on other datasets such as maps images (fig.5) or landscape to Ukiyoe paintings (fig 6).

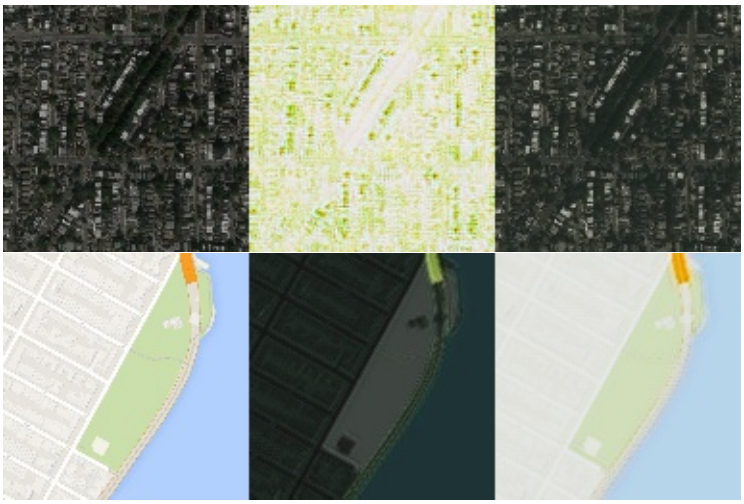

Figure 5 – Right: original image - Center: generated image - Left: reconstructed image

Figure 6 – Results for landscape → Ukiyoe

5 Conclusion

I didn't get as good results as I was expected. In fact, I discovered the difficulty to train generative adversarial networks which are very unstable models. I couldn't achieve more experiments, find the main reason why I couldn't get as good results as in the paper. It is obvious that the time taken to train the model was not the same. I didn't trained the model longer than 200 ephocs and I didn't used a lot of data augmentation.

6 Code

Code - My implementation of Cycle GAN - *Github* → `https://github.com/blavad/CycleGAN`

Bibliography

[1] J-Y Zhu, T. Park, P Isola, and A. Efros. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. Mar 2017. https://arxiv.org/abs/1703.10593

[2] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros. *Image-to-image translation with conditional adversarial networks*. Nov 2016. https://arxiv.org/abs/1611.07004.

[3] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. *Generative adversarial text to image synthesis*. May 2016. https://arxiv.org/abs/1605.05396.

[4] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. *StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks*. Dec 2016. https://arxiv.org/abs/1612.03242