Unpaired image-to-image translation with GANs: short study

Benjamin LAZARD
Ecole Polytechnique
Route de Saclay, 91128 Palaiseau
benjamin.lazard@polytechnique.edu

Abstract

In this report, I investigate some factors of performance of the CycleGan algorithm [4]. More specifically, topics like the number of images required to obtain a satisfying performance, the quality of the database, memory usage, and the design of the loss function were all explored at varying degrees of depth.

1. Reminder on GANs

Generative Adversarial Networks are a novel technique aimed at translating high-level properties from an image to another one. Famous exemples involve automatic coloring of black and white sketches, converting horses into zebras, apples into oranges, or transferring facial expressions from some animals onto others [3].

1.1. Neural Networks components

- a generator: a component which learns how to produce a realistic image embodying some key characteristics from the original one based on noise.
- a discriminator: a component which learns how to tell whether an image is real or a fake generated by the generator.

The generator is a special type of neural network called an encoder-decoder. It learns some key features from the set of images it is trained with (by a reduction of dimension using convolutions and activation functions), in order to be able to restitute realistic images when random input noise is fed in.

The discriminator is similar to the generator, but the output is a small patch of likelihoods for regions of the generated image to be a fake one.

1.2. Loss function

As often in complex deep learning structures, the rationale which explains the functioning of the global algorithm is hidden in the loss function. Basically:

- The discriminator must tell fake from real images, thus minimize $\mathcal{L}(D) = \mathbb{E}_{y\sim p_Y}\left[log(D_Y(y)\right]$
- The generator must fool the discriminator, thus minimize
  $\mathcal{L}(G) = \mathbb{E}_{x\sim p_X,\, z\sim\mathcal{N}(0,1)}\left[1 - log(D_Y(G(x,z))\right]$
- Cycle-consistency: to correct the fact that the discriminator is not trained on a known equivalent y_i for a given x_i but on the full set X, Y, we simultaneously train 2 generators and discriminators in order to go from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$ and back, to guarantee

  $$G_{\mathcal{Y}\to\mathcal{X}}\circ G_{\mathcal{X}\to\mathcal{Y}}(x_i) = G_{\mathcal{X}\to\mathcal{Y}}\circ G_{\mathcal{Y}\to\mathcal{X}}(y_i) = Identity$$

  that is

  $$\mathcal{L}_{cycle} = \mathbb{E}\|G_{\mathcal{X}\to\mathcal{Y}}\circ G_{\mathcal{Y}\to\mathcal{X}}(y) - y\|_1 + \|G_{\mathcal{Y}\to\mathcal{X}}\circ G_{\mathcal{X}\to\mathcal{Y}}(x) - x\|_1$$

The total loss being the sum of aforementionned losses.

2. Impact of the dataset

There were several implementations of the original paper [4]. I decided to use the tensorflow implementation of Vanhuiz [2] as the baseline for my experiments. The main motivation was the use of tensorboard, and the existence of a pre-built module enabling the use of locally trained models for inference.

After ensuring the algorithm worked on the dataset "apple2orange" as you can see on 3, 3, and 3,I decided to train the algorithm on a more challenging dataset.

Indeed, VanHuiz trained the network on classic GAN datasets such as horse2zebra and apple2orange. Both datasets involve clear-cut objects (from the background) with a relatively simple geometry, and change of color. It is explicitly said in the original paper [4] that GANs are not good so far at transferring geometry changes, but rather good at ctaching textures and colors. I wanted to witness

for myself what the effect would be on a more complex dataset, for which I didn't know if a good result was even theoretically possible (as the result shown were always impressive). I crawled the web for pictures of people and superman comics. I collected about 1000 images for each (the original paper implying good results good be obtained on simple objects with as little as 400 images). Because of the automatic collection, it is worth noticing there was a significant proportion of outliers for the people dataset (I kept it below 5%), and a significant heterogenity in the comicbook drawing style 6. Results were really disappointing. Basically as you can see on figure 7, the GAN barely smoothed some contours out, and picked out the blue/red color dominance. Training time was nonetheless optimal as the loss function clearly stopped decreasing (see figure 8). Maybe a cleaner dataset, or just a bigger dataset could have led to better results.

Rather than exploring potential solutions, I decided first to check the outcome of the algorithm on a dataset I was confident was not too complex.

3. The importance of model complexity

I crawled the web again for a simpler dataset: pictures of forests in the winter and in the summer. You can see a sample with a few outliers on figures 9 and 10. I expected the algorithm to be able at least to grasp the dominant green vs. white dominance in color, and then to deem its effectiveness by the ability to add lumps of snow where needed, or to generate leaves on trees for winter2summer conversion, etc.

I was however really disappointed to get very bad results with default parameters: just like superman, I got some sort of negative image in green for the summer and in white for the winter. 11.It is only true that the loss could potentially have decreased more as you can see on 12, yet I decided to impose a half-day limit for training each algorithm and assessing quality (this is about 10k steps or between 10 and 20 epochs)

This time, I went further and took it as a challenge to make it work. The best solution I found was not to change the mathematical model, but rather to add complexity to the model, more specifically, to add more layers in the residual blocks (convolutional layers in the discriminator), which was too easily fooled by the generator. I strayed away from the 9 described in the paper and with 36 I obtained a satisfying output (with leaves and flowers generation) 13, 16. It is however worth noticing that although total loss kept decreasing as desired, one of the generator's loss started diverging at the end of the training. Also, when facing objects not met in training, the color of these objects may be changed in unwanted ways 15. Finally some white patches with a strong contrast appeared on some pictures. I did not investigate their appearance, it could be just an attempt at creating a sun on pictures without it at first, but it may also be a side-effect of my modification of the structure of the network.

I pushed the number of residual blocks up to 64 with no significant improvements 17.

4. L1 vs L2 loss

Once I had verifyed the algorithm worked on my new dataset, I explored more mathematically the impact of a few changes on the loss function. An important choice mentionned in the original paper was the L1 loss used in the cycleGAN loss. Indeed, they argued L2 loss would lead to blurry images, whereas L1 loss would guarantee sparsity and thus sharpness and realism. I tested (that was before the resnet changes though), and I can only confirm as you can see on figure 18.

5. Regularization of Cycle consitency loss.

Another important choice was the choice of a λ parameter for regularization of the total loss:

$$\mathcal{L}(G_1, G_2, D_1, D_2) = \mathcal{L}(G_1)+\mathcal{L}(G_2)+\mathcal{L}(D_1)+\mathcal{L}(D_2)+\lambda\mathcal{L}_{cycleGAN}$$

In order to understand the impact of the cycle-consistency loss, I modified this parameter λ in order to reduce its weight by 10 or to multiply it by 5.

5.1. cycle loss x0.1

In this scenario, the images produced by the generator are no longer very consistent with the input (figures 19 and 20). I suppose the discriminator is no longer hard to fool, as it is harder for it to recognize the output distribution without the loop. White images are accepted as satisfying for both the generator and the discriminator in the direction summer2winter.

5.2. cycle loss x5

Conversely, in this scenario, the images produced are rather satisfying, but tend to lack sharp contrasts. Also I suspect training time might be longer in this scenario (loss not converging as fast) 21, 21.

6. Hardware limitations

GAN computations are very resource-consuming. The average training time is 4h, and take up to half a day. I learned how to use a remote computer for calculations: the machines researchers use in my university with Tesla V100 GPUs (16Go, but limited to one per user), but time was still a constraint. Although the original paper claim it is possible to use GANs on higher resolution images, I found I could not go above 512x512 images with this implementation of the algorithm and a single GPU, even with a batch-size set

to 1. However, I also tried other implementations of variants, such as the UNIT algorithm [1], which seemed less memory-consuming.

7. Pixelisation of output

It is noteworthy that most of my results are somewhat pixelised. I did not correct this because this phenomenon appears in the original paper and does not seem related to the quality of the algorithm. This is directly related to the patch size of the GAN (which is not explicit in the implementation of the code I used: layers of the disciminator increasingly decrease in size until they reach the final output size). Van Huiz's implementation actually used the worst case scenario mentionned in the paper: 16x16 patches for the discriminator, resulting in sharp images, but with a tiling effect around the indecision boundaries (border of the patch). It would be nice to remove this by increasing the output patch size to 70x70 (see original paper result 1).

Figure 1. Impact of the discriminator patch size on pixelization

8. Further improvements

It would be nice to explore other ways to make the neural network more complex: what is the ideal way to dimension the network depending on image size and dataset ? Also, it would be interesting to come back to the people2superman project with all these improvements. Working on several GPUs to get larger resolution results would be fantastic.

9. Conclusion

Please find a summary of the most interesting experiments undertaken at figure 2. This corresponds to all the models trained on the entire dataset, and their effect on the 2 same input images (to visually compare them on the same basis).

Working on GANs was very interesting, and I was surprised to discover how much some degree of hyperparameter tuning can drastically impact the quality of the result. Because of this, I believe GANs still requires some improvements before they can be easily generalized: how to dimension best the network, and choose the ideal complexity seem like an interesting research project to me.

References

[1] Ming-Yu Liu and Thomas Breuel, Jan Kautz. github repository for a unit gan implementation, 2017.

[2] Van Huyz. github repository for a cyclegan-tensorflow implementation, 2017.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial nets, 2017.

[4] J.-Y. Zhu, Taesung, P. P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.

Appendix 1

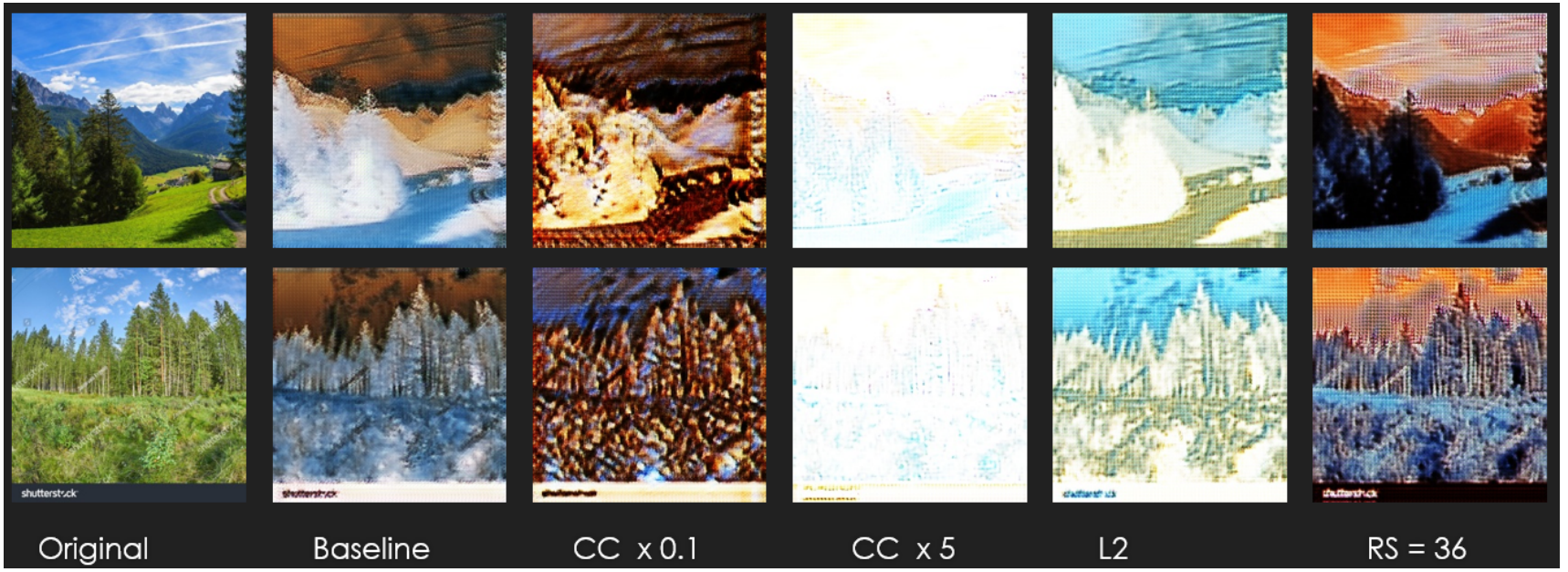

Figure 2. Summary of experiments

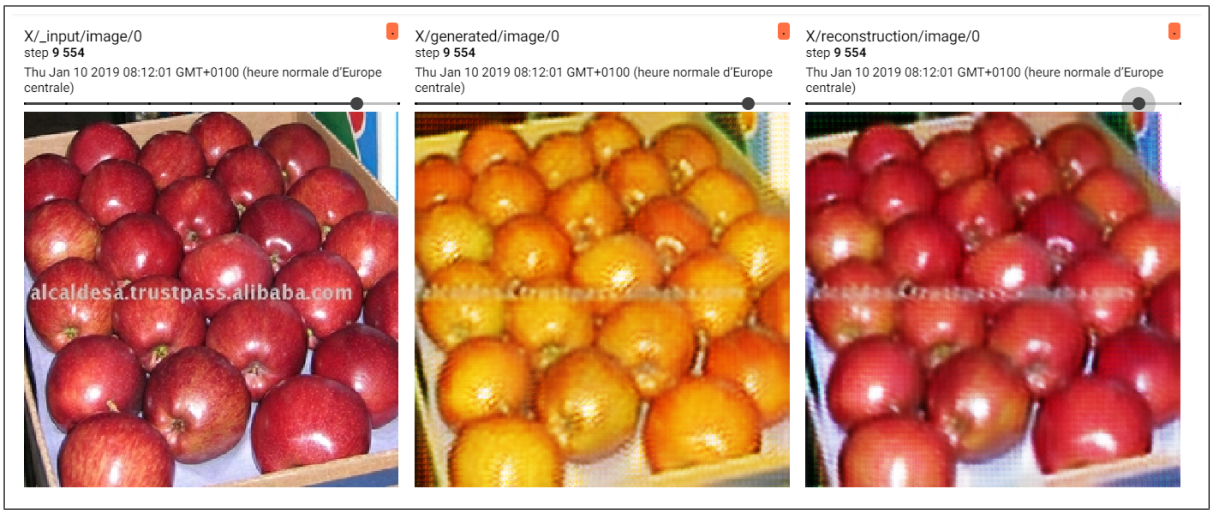

Figure 3. Baseline apple2orange success

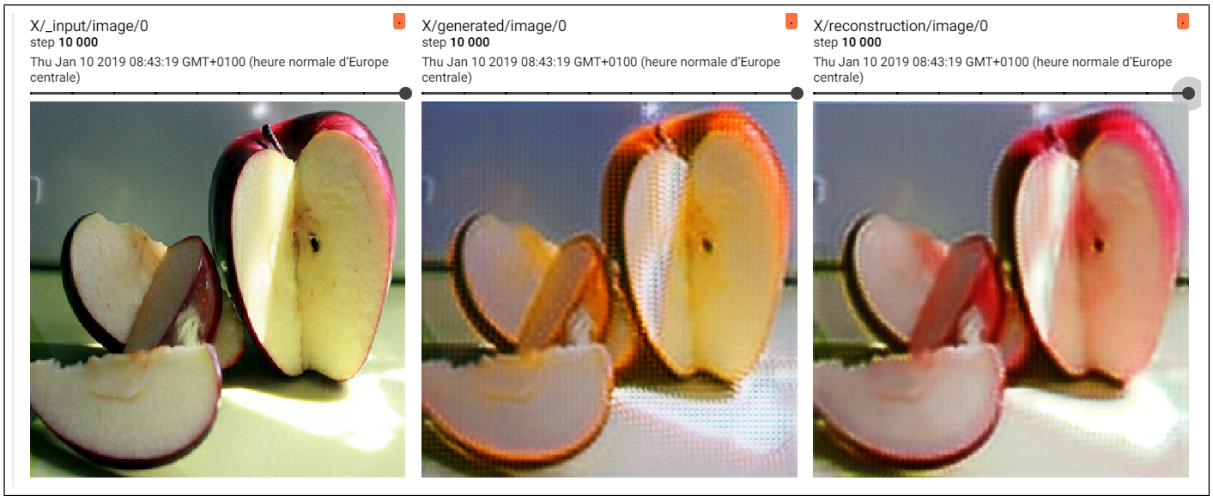

Figure 4. Baseline apple2orange failure

Appendix 2

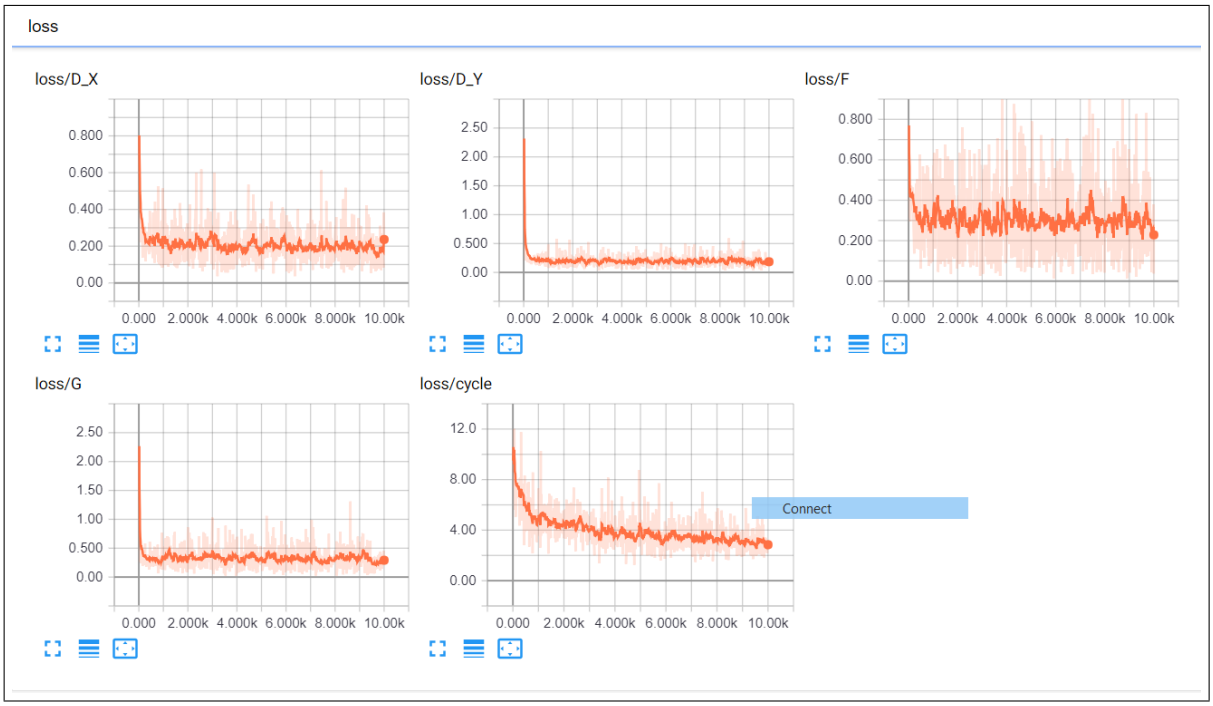

Figure 5. Baseline apple2orange evolution of the loss function.

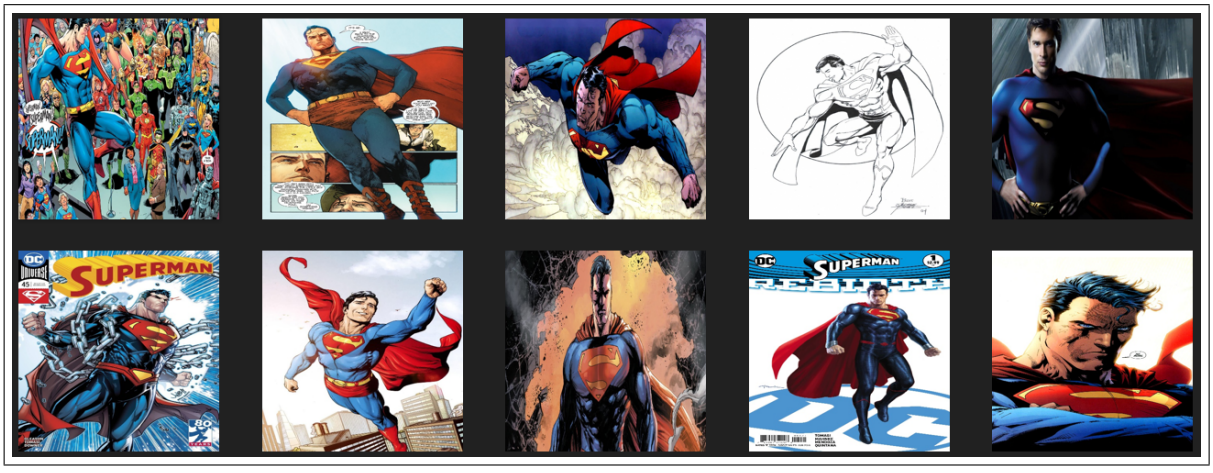

Figure 6. superman comics dataset sample

Figure 7. Baseline people2superman result

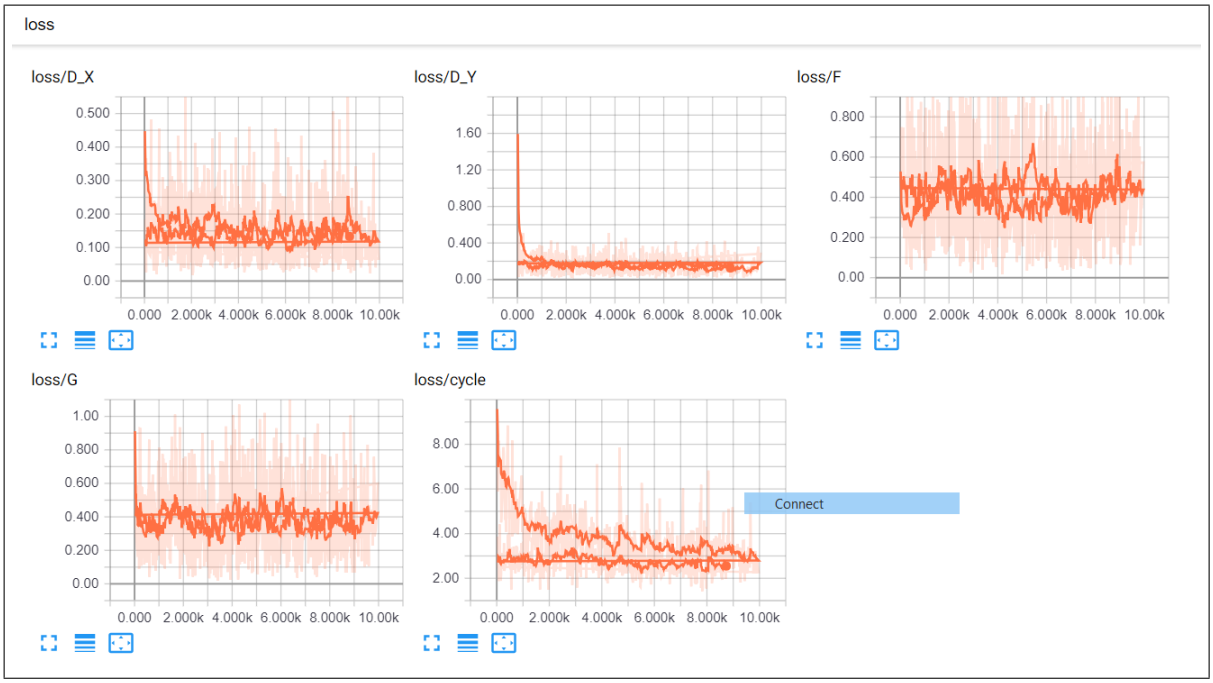

Figure 8. Baseline people2superman loss evolution (1 day of training)

Appendix 4

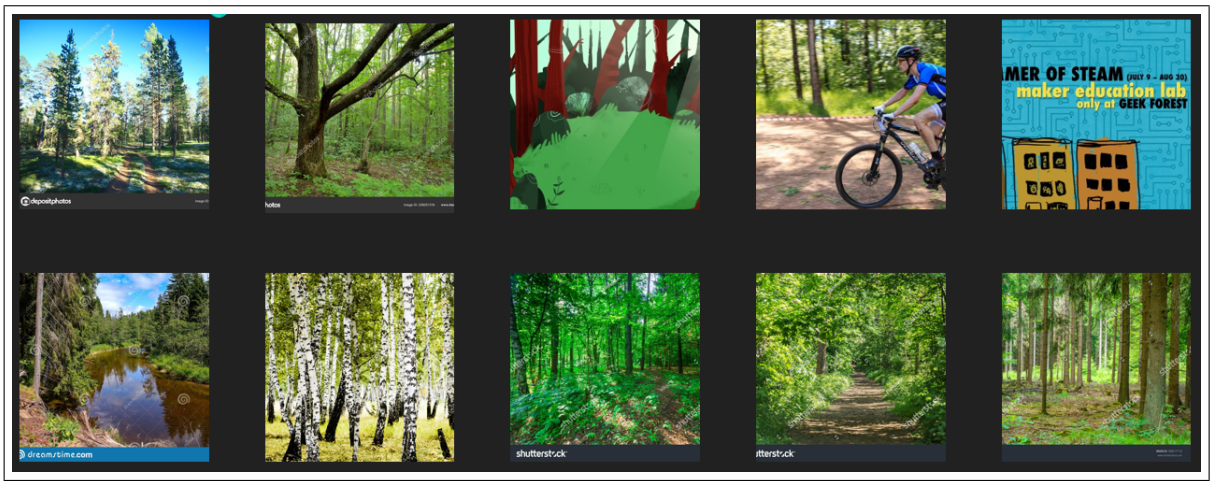

Figure 9. summer forest dataset

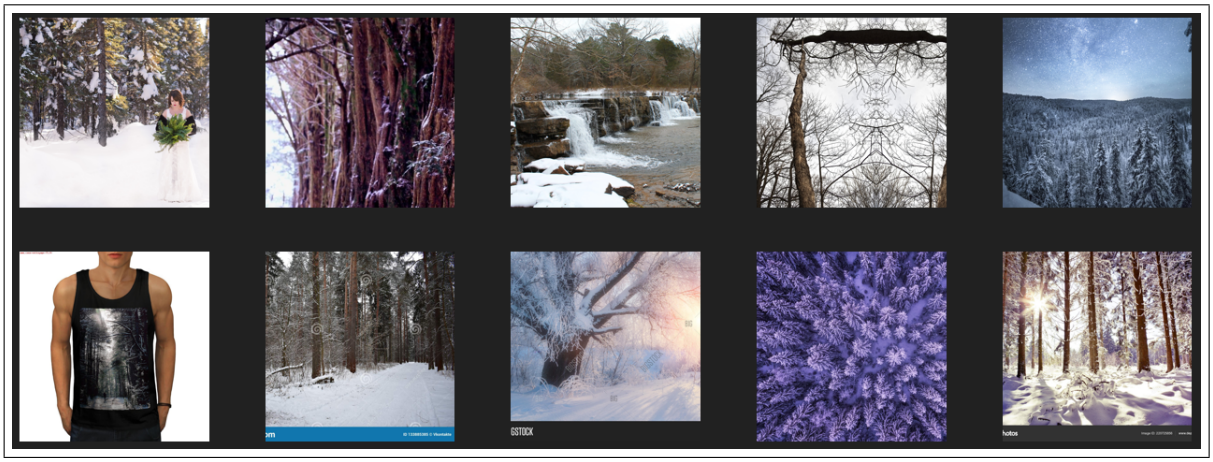

Figure 10. winter forest dataset

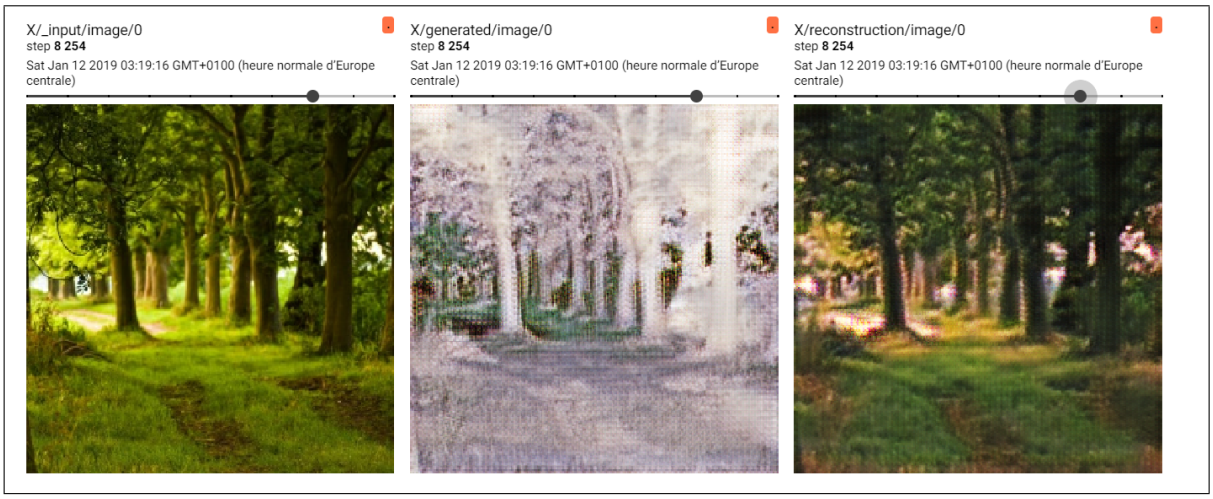

Figure 11. baseline summer2winter conversion example

Appendix 5

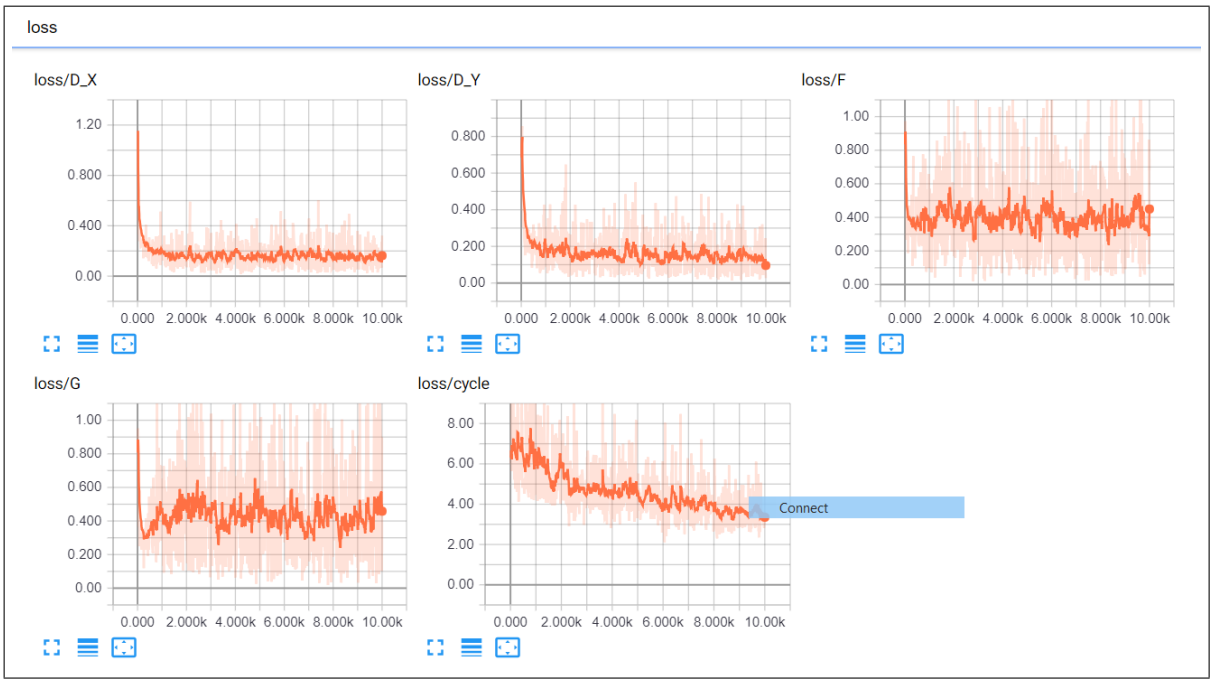

Figure 12. baseline summer2winter conversion loss decrease

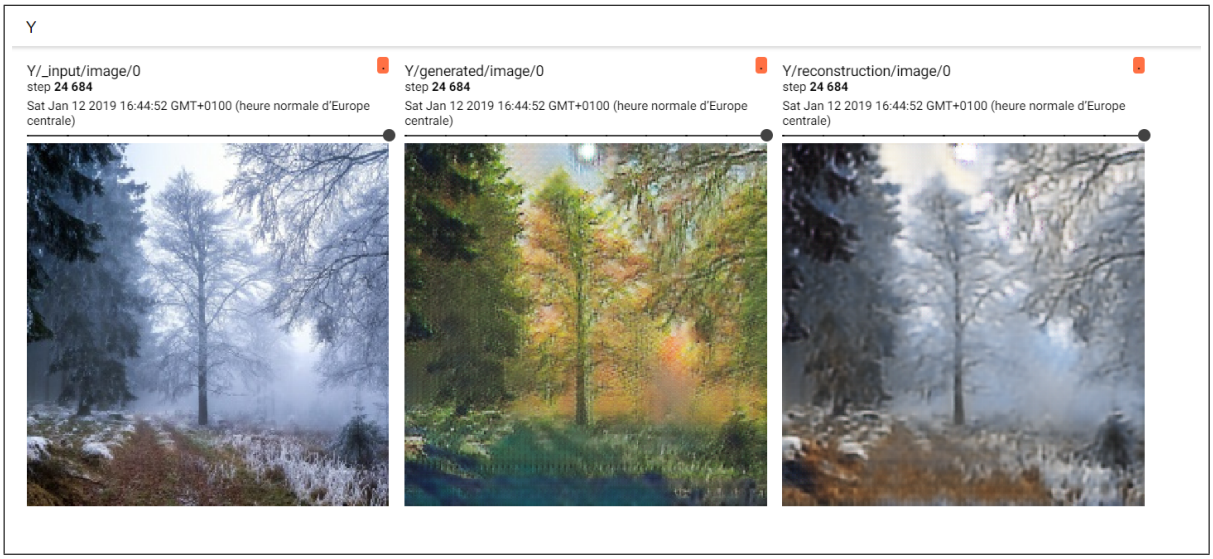

Figure 13. summer2winter 36 residual blocks satisfying output

Appendix 6

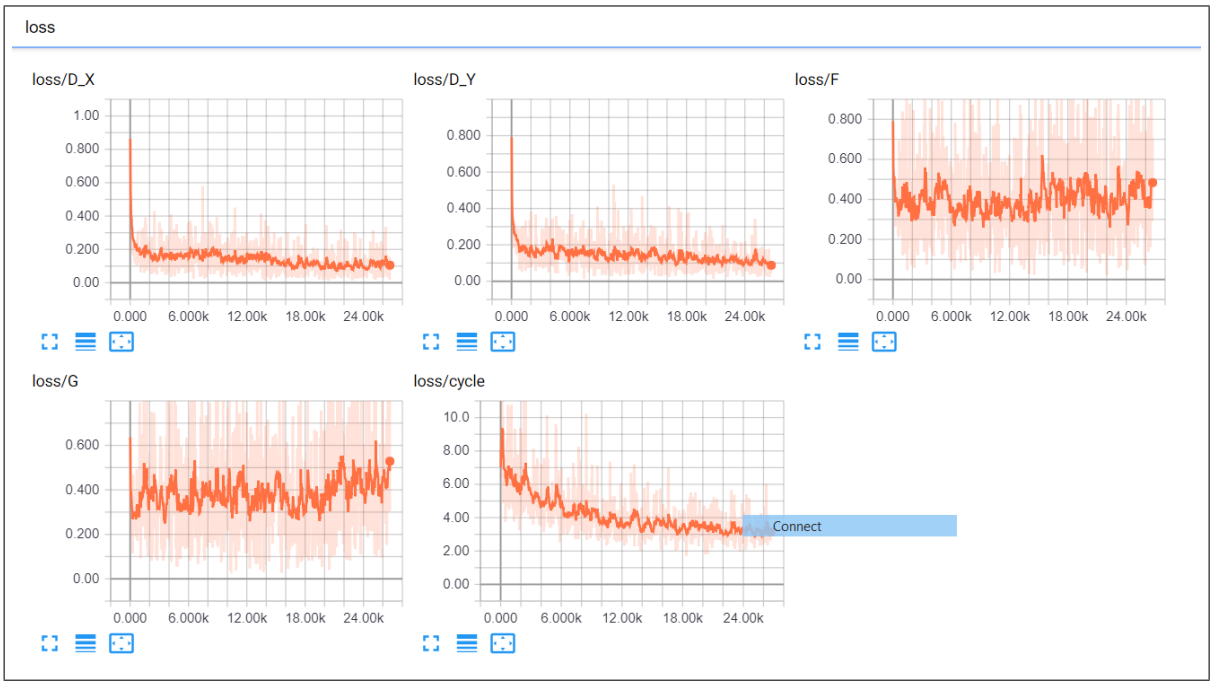

Figure 14. summer2winter 36 residual blocks loss

Figure 15. summer2winter 36 residual blocks unwanted color changes

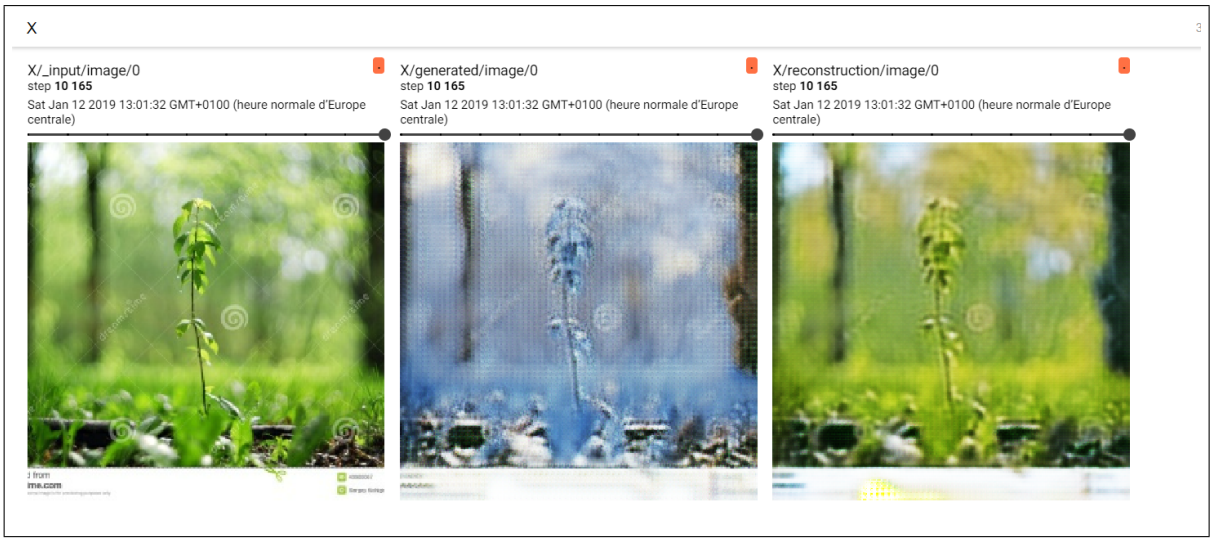

Figure 16. summer2winter 36 residual blocks satisfying output

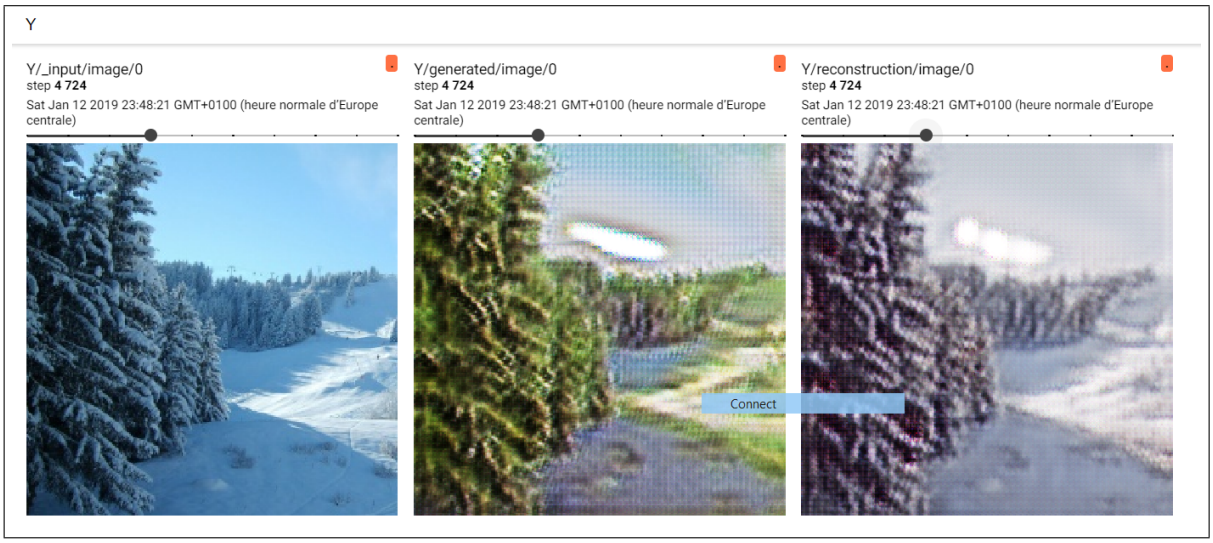

Figure 17. summer2winter 64 residual blocks satisfying output

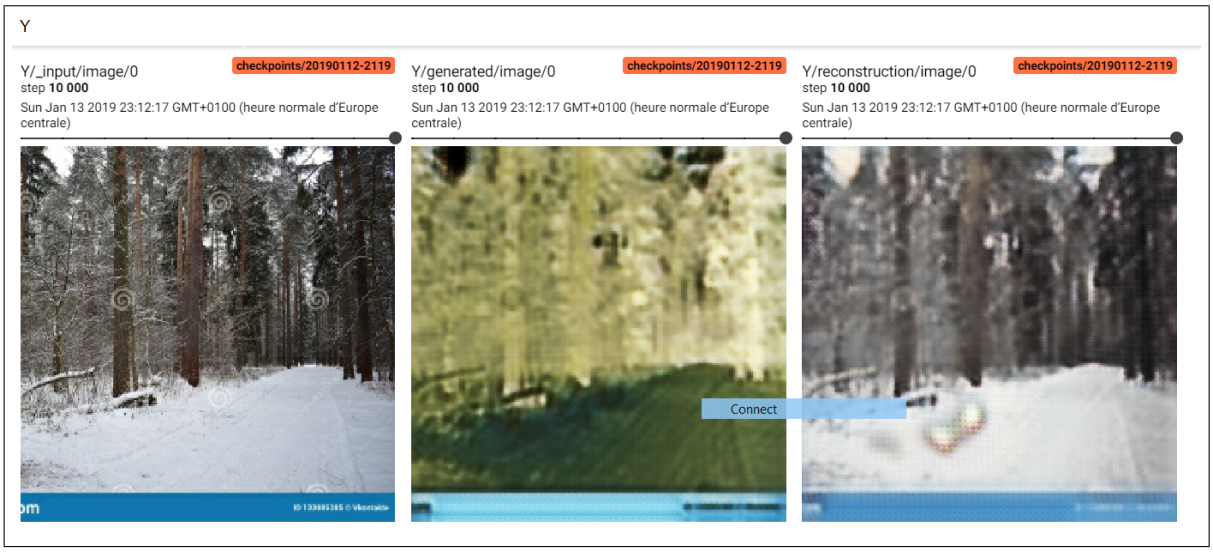

Figure 18. L2 loss summer2winter output

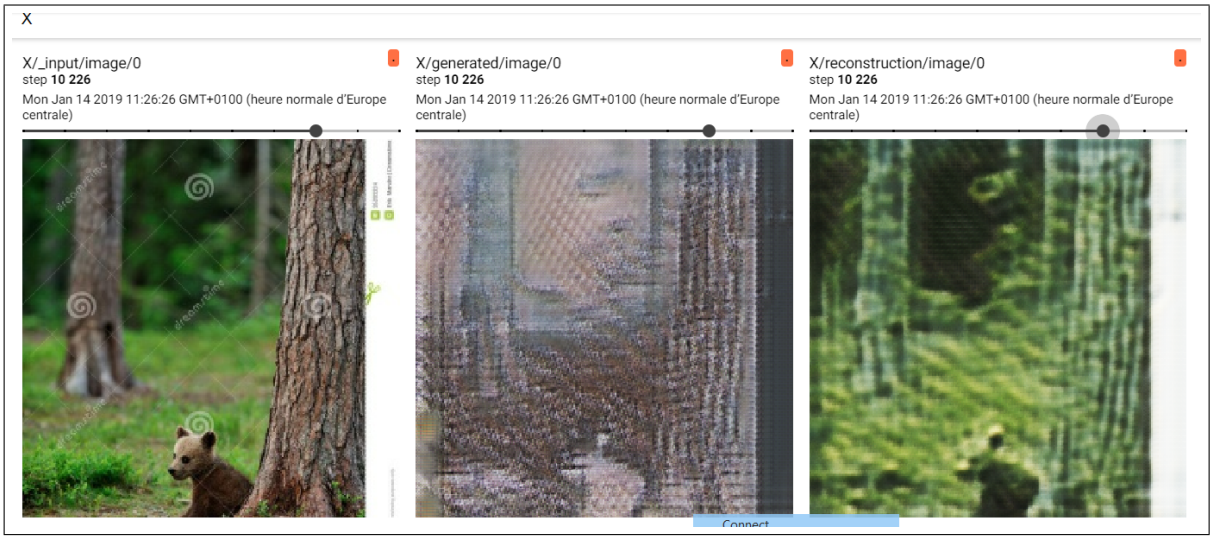

Figure 19. Cycle Loss x0.1 summer2winter output

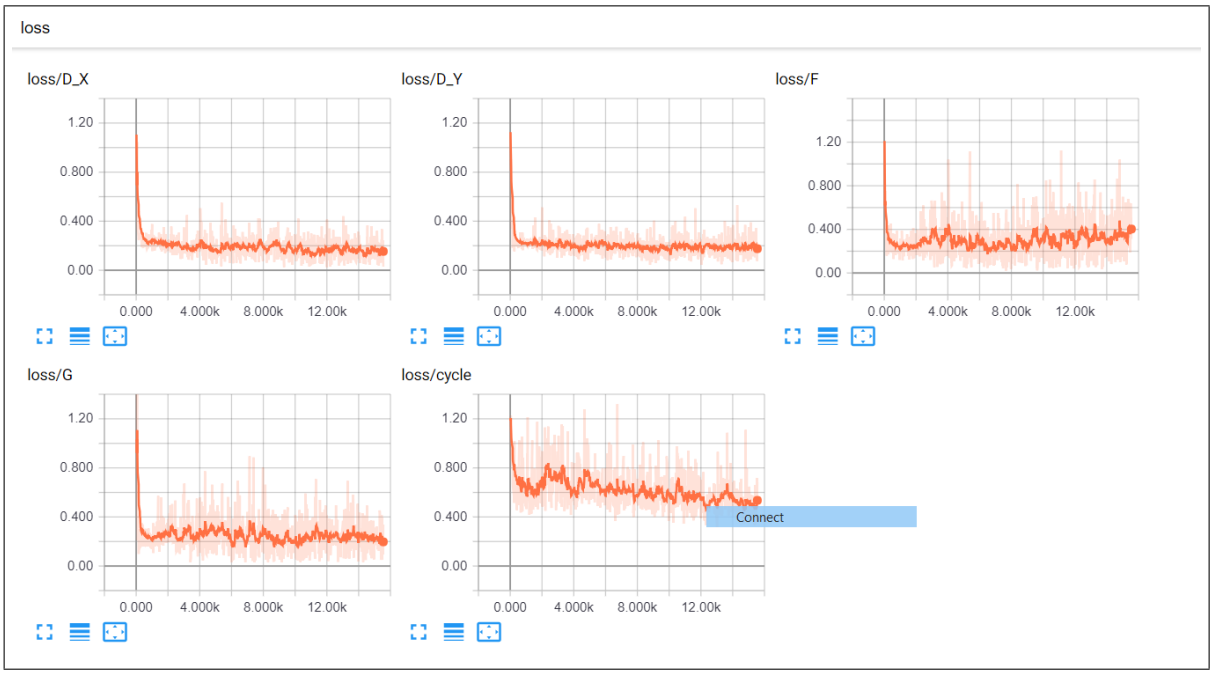

Figure 20. Cycle Loss x0.1 summer2winter loss

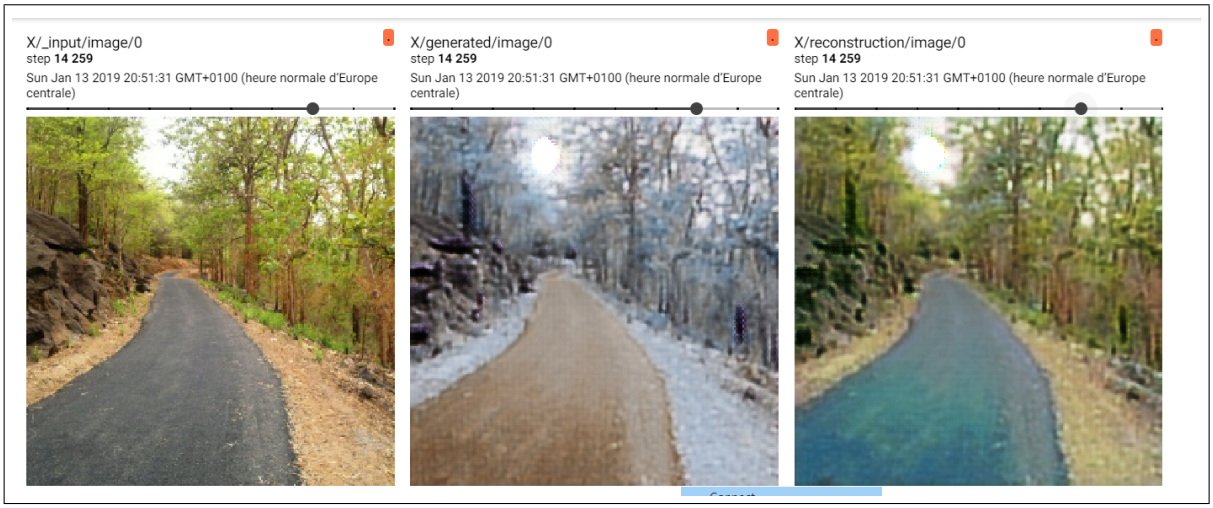

Figure 21. Cycle Loss x5 summer2winter output

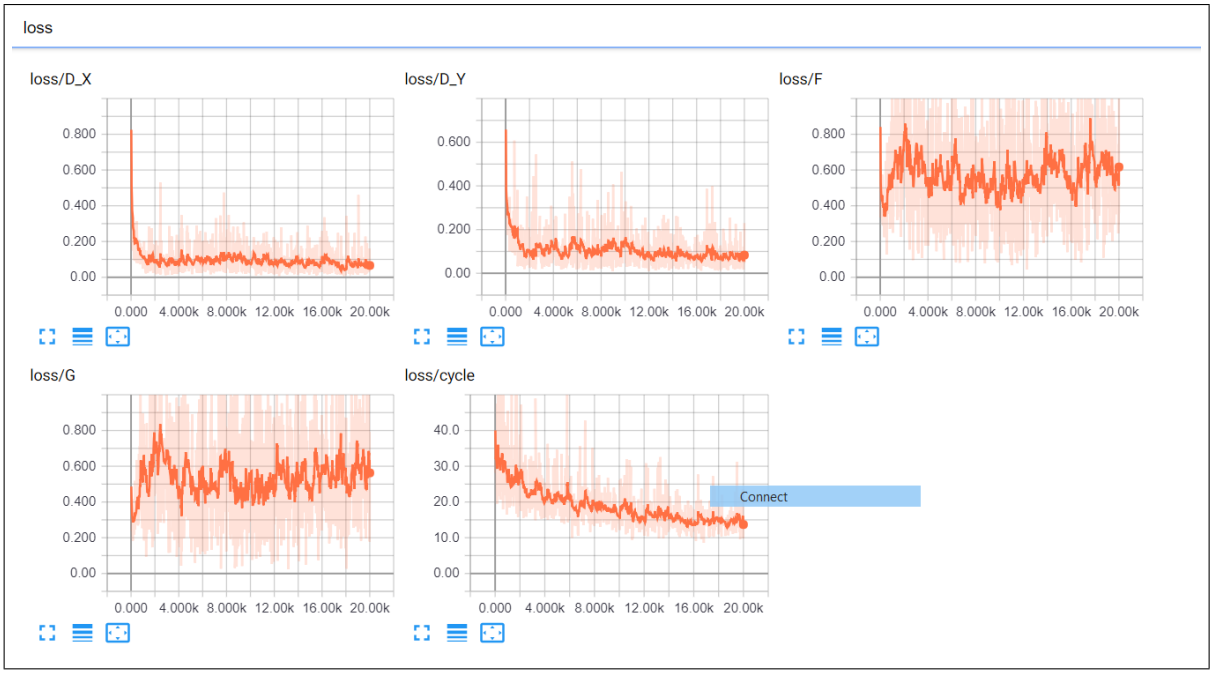

Figure 22. Cycle Loss x5 summer2winter loss