Comparison of cGANs vs CycleGANs for Image Augmentation by Sketching

Rodrigo Ledesma
Department of Computer Science, University of Ottawa
rlede046@uottawa.ca

Abstract

In this report, we will compare the performance of Conditional Adversarial Networks (cGANs) against that of CycleGANs for Data Augmentation (DA) tasks using Frechet's Inception Score (FID) and human validation. Our approach will consist of Image-to-Image translation (i2iT), giving as input a sketch and turning it into a Monet painting. DA's purpose will be to create a complementary set of images by using sketches to improve the quality of the original translation models.

1 Introduction

This project will focus on i2iT, which consists of translating one possible scene representation into another [1], for example, synthesizing photos from label maps, reconstructing objects from edge maps, and colourizing images. Initially, this task was tackled using Convolutional Neural Networks (CNN)[2,3,4]. Unfortunately, each problem had to be solved with a specific architecture and a personalized loss function, which required specialized machinery and experts. This was not only time-consuming but also economically inefficient. This issue began to see the light sometime after Goodfellow introduced Generative Adversarial Networks (GANs)[5].

Pix2Pix was an excellent public introduction to image translation technology. This software was developed using cGANs. The problem with this approach is that it is too general, and the images it produces from the sketches look unrealistic because sketches are unrealistic too. Please click this LINK1 [6] to view an example. [1] In [1] Isola and his team demonstrated that cGANs are an excellent approach to tackle different i2iT problems without using different architectures and loss functions. Simultaneously, CycleGANs were tested and showed a good and measurable performance on i2iT tasks. We will discuss this in section 2.

This report will explore and compare both architectures for Data Augmentation purposes. Figure 1 is the result of training a cGAN and asking it to transform a sketch containing a castle into a painting. As it can be appreciated, the castle's definition is very poor compared to the original. This behaviour is justified as there is not enough training data containing castles. In section 3, we will explain how we used both architectures to produce an extensive and new dataset containing images of castles painted with an Impressionist technique. We will compare which model creates the best-augmented image dataset from only a tiny sample of 50 impressionist castles (model=m2). Moreover, use both datasets and the original Monet paintings dataset to evaluate which one produces the best Impressionist representation of a sketch (model=m3).

The motivation behind this project is to venture into an alternative methodology to traditional data augmentation, which consists of cropping, rotating, and jittering images. Our approach wants to take advantage of the fact that it is easier to obtain sketches of pictures. Either by downloading an existing

[1] Due to limitations of space, images and graphs will be stored in a cloud drive, and the reader will be able to access them by clicking the hyperlink indicated by the word LINK in each part

Figure 1: Result of generating a paining containing a castle using m1. The generated image is at the far right, input sketch for conditioning is in the far left and in the center is the real image.

dataset or by using an AI-based technology to transform a dataset into sketches. And now, use those sketches and translate them into useful and accurate images with the help of the created models.

2 Related Work

In 2018 Philip Isola et al. [1] contributed to this field by evaluating the efficiency of conditional Adversarial Networks as a general-purpose architecture to tackle several i2iT problems. Their work showed that cGANs could efficiently solve all the given tasks without training different models with independent loss functions for each task.

One problem with this architecture is that pairs of images are needed to train the model. Input and output must be concatenated in a single file to feed the training algorithm. As this technique proved its efficiency, another type of architecture was tested. In 2017 CycleGANs were introduced, and their original concept worked as well as cGANs for i2iT tasks[7]. Instead of training a single pair of models consisting of a generator and a discriminator to transform image X into image Y, CycleGANs train four models consisting of two independent generators and two discriminators. The first pair will tackle the translation of an image X to Y, while the second pair will be simultaneously trained to translate the image in the opposite direction from Y to X. CycleGAN's architecture comes with another perk; for training purposes, users can use unpaired images, which means that you can feed the algorithm with independent and scrambled datasets.

Another exciting effort was made by Lin J et al. in 2018 [8]. In this paper, they use the conditioning of paired images in processing "edged to handbags." This technique translates an image made only by edges (the equivalent of our sketches) in black and white into an image of an authentic purse. However, the authors discuss in their paper that this methodology presents a problem, and it is that a fixed image usually leads to a deterministic translation. Therefore, the researchers use another "conditional Dual GAN" technique to avoid this issue. This architecture consists of four parts, two encoders that will extract two different types of features: domain-independent and domain-specific. They will all be fed into two different decoders, which will be used as generators which take as inputs the domain-independent features from the image in the source domain and the domain-specific features from the image in the target domain and output a generated image in the target domain.

3 Research Questions

RQ1: Comparison of image quality cGAN vs CycleGAN *Are images generated by a CycleGAN better than those generated by cGAN?*

RQ2: Training time comparison *Based on the time it took to train both models, it is worth the time invested in training CycleGANs related to the quality of their predictions?*

RQ3: Comparison of Data Augmentation technique cGAN vs CycleGAN it Which architecture produced the best impressionist-like images of castles?

RQ4: Testing images' realisticness with humans it Are the generated images' quality of both architectures good enough to fool users?

RQ5: Will this DA technique improve models' performance it Will our new Data Augmentation technique improve the FID score or the way generated images look?

4 Technical Approach

The GAN's work is to generate realistic images by learning how to map random noise into an image, they are divided into two different parts. The first neural network will be in charge of encoding the input information, creating and improving the latent space and then generating new images based only on a vector of random noise z. This network will be assisted by a discriminator whose job will be to classify their inputs into real or fake (created by the generator). Eq 1 represents the loss function of an unconditional GAN. The loss function will be updating both parameters at the same time, but looking for maximizing the loss for the discriminator and minimizing the loss for the generator.

Let's now also analyze the loss function used in conditional GANs in eq. 2 the generator uses as inputs both the original images tensor x and the random noise tensor z. As well as the Discriminator. This conditioning forces the model to follow a specific design and to nudge the predictions towards a specific result.

$$\mathbb{L}GAN(G,D) = min\theta_g max\theta_d[\mathbb{E}_x(logD_{\theta d}(x) + \mathbb{E}_z log(1 - D_{\theta d}(G_{\theta g}(z))] \quad (1)$$

$$\mathbb{L}_c GAN(G,D) = min\theta_g max\theta_d[\mathbb{E}_{xy}(logD_{\theta d}(x,y) + \mathbb{E}_{xz} log(1 - D_{\theta d}(G_{\theta g}(x,z))] \quad (2)$$

In this report we have decided to use the same architecture for both the cGAN and CycleGAN, thanks to the python library "pix2pix". Let's take a closer look to the architecture's details.

4.1 Generator's architecture: U-Net / Generator with skips

When the generator performs i2iT task, it tries to map a high-resolution input grid to a high-resolution output grid. Generally speaking, this process can be performed using auto-encoders, but they present an issue. The encoder down-samples the image's information until it reaches a bottleneck, then up-samples the same information. Due to the bottleneck, the probability of losing important information while creating the latent space for reconstruction is high. U-Net architecture is a variation of a standard encoder which links the encoder's layers with the decoder's layers and allows the information to pass directly from one to another [9]. In terms of application, this technique multiplies the direct results from the down-sampled layers with the ones in the up-sampled layers. For a closer look, please click this LINK2 to see a graphical representation f the U-Link architecture.

4.2 Discriminator's architecture: PatchGANs

PatchGANs use Convolutional Neural Networks (CNN). The discriminator's job is to analyze the image and decide whether it was created by the generator or is original. PatchGANs logic divides the analyzed image into patches or chunks and analyzes each patch individually, determining if each portion of the image is fake or original. In [2], the authors fine-tune the patch size, which we can relate to the concept of "filter" used in a typical CNN. Please click this LINK3 to open an example of PatchGAN's analysis, and as seen in the image, most of the patches are blue or white which means the discriminator is classifying them as fake, as the model is untrained.

4.3 Evaluation Metrics

Evaluating a generated image's quality can be subjective and difficult to perform. In [1], the authors use a technique called FCN-8s [10]. Another commonly used method for evaluating images is the AMT perceptual studies [1]. It refers to Amazon Mechanical Turks, a platform Amazon provides to link the human workforce with people in need of services. In our study, we will be using Frechet's Inception Inception Distance (FID) which has been proven to give an objective score of how similar a generated set of images is to the original set [11]. FID is calculated using the inception v3 function in python, which generates the distributions using features from both sets of images using the mean and covariance. Then calculate the distance between both distributions using Frechet's distance. This method was proposed as an improvement to the Inception Score. The smaller the number, the more similar both sets of images are. The second metric we will use will be a survey. We will ask 11 random people to identify which image (in a pair) is the original painting. For our research, we will show each pair of images for 1 second and use 40 pairs. 20 for m2 and 20 for m3.

5 Methodology

In this paper, we will divide the pipeline into 5 semi-independent steps: 1) Dataset description and pre-processing. 2) Training m1 Initial GANs for sketch-to-Monet Translation. 3) Training m2 Castles' dataset preparation and model training for Data Augmentation. 4) Training m3 Final GANs model for improved sketch-to-Monet Translation with castles.

5.1 Step 1: Dataset

The base dataset used in the project is a group of Monet's paintings cropped into a format of 256x256 pixels. This dataset was obtained from a repository [12] which also was used in the work of Isola et al. for cycleGANs [10]. There are 1073 training images and 122 for testing. This dataset can be found ready to use in folder 01MonetRealDataset inside the "dataset" folder in the provided drive [13].

As we want to use sketches as the condition of our project, each of Monet's paintings will be turned into a sketch. Therefore, we used an online tool that gives excellent results. The web page's name is BeFunky [14]. This platform allows the user to upload an image and create a sketch of it online.

The last step to prepare this first dataset was to create pairs of images to feed the cGAN. We had to concatenate both the sketch and the painting in a single 512x256 pixel with the painting first followed by the sketch (click this LINK4 to look at the concatenated image.) Please look at both datasets by opening 02 Monet sketch dataset, and 03 MonetMerged. To make the concatenation, we used a simple python script written in a jupyter notebook. This notebook's name is 01 Image concatenation.ipynb. Please open link [19] that contains all jupyter notebooks used in the project .JUPYTER NOTEBOOKS

5.2 Step 2: Sketch-to-Monet

As mentioned before, this project will compare the use of cGANs against CycleGANs for image data augmentation. So first, let us describe the cGAN's pipeline.

The dataset was uploaded, split, and turned into tensors for future processing (please open the notebook entitled 02 cGAN Monet to appreciate the process step by step). We built the generator according to [1]Isola's work and pix2pix methodology by down-sampling the images from 256x256 to 1x1, creating the U-Net links and then starting re-sampling the image from 1x1 to 256x256. In the executed cell 29, please appreciate an example of the generator without any training using a test image, or click the following LINK5

The next step was to create the discriminator, which uses convolutional neural networks in 2D, and a fixed patch size. Please see an example in the executed cell 35 or click this LINK6 to appreciate how the untrained discriminator decides which patches are authentic and which ones are not. Finally, losses and gradients are defined, and we train this algorithm for 40 epochs as this was the default number given in the paper.

Talking about CycleGAN's pipeline, the general process of ingestion, data processing, and architecture are the same. We were lucky enough to find a library that allows users to download these architectures. For example, when going into the executed cell no 11 (on 03 cycleGAN sketch to monet.ipynb) you will be able to appreciate how by using the pix2pix library, the user can download the U-Net generator and the patchGAN discriminator. For this architecture, we trained it for 20 epochs.

5.3 Step 3: Impressionist Castles Generation

This is the most crucial part of the project. This section will describe how we gathered a small dataset and produced more data using different GANs architectures.

First, we did a quick search on Google, looking for two different datasets: 1) Pictures of castles painted with an impressionist technique. 2) A dataset of actual real castle images.

For dataset 1, we only downloaded 50 different impressionist-like paintings of castles. This dataset was also turned into sketches using the BeFunky platform, and they both can be found under the names: 07 Castles Impressionism and 08 Sketch castles impressionism. Click this LINK7 to see an example of these images. Dataset 2 was more straightforward, as we could find a complete dataset

which already contained more than 1000 different pictures of real castles on the webpage "images.cv" [15]. Click this LINK8 to look at examples of real castle dataset. We had to refine the images, as some contained text or banners. Once debugged, this dataset was resized, turned into .jpg, and finally turned into sketches. Click this LINK9 to see an example of the sketch. Please find these datasets inside the provided folder: 09 real castle train, 10 real castle test, 11 real castles sketches.

Let us first describe the cGAN. Loading the data and creating the model has the same procedure described before. It is essential to mention that as we are training an algorithm with only 50 pairs of pictures, we decided to have some overfit in the model and train this specific model for 50 epochs instead of only 40. This notebook can be found under the name 04 cGAN for castles.ipynb. Moving on to the CycleGAN, the process was recycled from the last implementation, changing only the algorithm's inputs. This model was intentionally overfitted and trained for 100 epochs instead of only 20. Please find more detail on the process opening notebook 05 cycleGAN castles.ipynb.

The purpose of overfitting both models is to ensure the accurate generation of castles with an Impressionist technique. Once both models were trained, we used each model (trained on the 50 impressionist castles dataset) to generate 1200 new images of impressionist castles using the sketches converted from real pictures of castles. The results of generating new impressionist castles from both models can be found in the provided folder under 13 cGANCastlePredictions and 14 CycleGAN Castle Predictions.

5.4 Step 4: Sketch-to-Monet/Castles

The final step was to create a dataset that contains two different types of paintings: 1) The original Monet's artwork with 1000 different paintings and 2) The augmented data provided by the GANs, which contains 1200 generated pictures of impressionist castles. For this purpose, we first combined all pictures of Monet's original paintings and the generated castles into a single folder. Subsequently, we split them into train and test.

Once the dataset was ready, the final models were now trained; starting with cGAN, we used the standard 20 epochs to train the model with the training dataset. If desired, you can validate the process and results by opening the notebook called 06 Final cGAN Monet + Castles.ipynb. The final cGAN model's results can be found in the folder titled 15 Final Predictions cGAN Monet+Castles.

Our last step in the data augmentation pipeline was to train the CycleGAN with the new dataset. This model trained for 20 epochs and as you can see when opening the notebook 07 Final cycleGAN castles-monet.ipynb. To validate the model's predictions, please open the folder entitled 16 Final Predictions CycleGAN monet+castles.

6 Evaluation of Results

Starting with the first m1 model trained by sketches-to-monet dataset using cGAN architecture. Training time per epoch was, on average 570 secs, and we trained this model for 40 epochs. It took a total of 6.3 hours to train, and by clicking this LINK10 you will be able to see some generated predictions. Some sketches are still visible from the original input, and also, the sky presents blur in patterns. Talking about the FID score, this model got 169.24, which we will use as a baseline. Moving on with the same dataset but changing the architecture to CycleGAN, there is a considerable increase in the training time. This one took 3400 seconds per epoch; being trained with 20 epochs, the total training time was 19 hours. And its FID score was 220. Click this LINK11 to see an example of the cycleGAN's predictions. If you open the folder "06 Predictions Sketch2Monet CycleGAN" you will appreciate how this architecture does not perform optimally, images are very saturated, and red colours are always present, the sky presents blurry spots.

Second, we will evaluate how accurate the generation of impressionist castles was by both architectures. First, cGAN, the average training time per epoch was 540 seconds, and training for 40 epochs, the total training time was 6 hours, and the generated images got an FID score of 131.81. Compared with CycleGAN, the second architecture using the same data took 200 seconds per epoch, and training for 100 epochs, the total training time was 5.6 hours. With an FID score of 237.31. Please take a look at this LINK12 and appreciate how this architecture generated very yellow-saturated images.

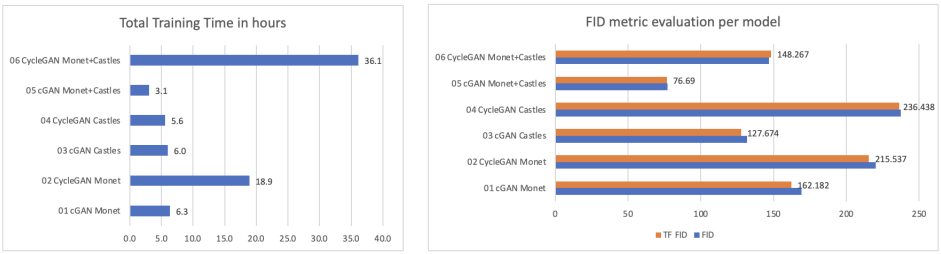

Figure 2: Left graph shows the number of hours it took to train each model, while the right image represents the FID score obtained by each model by comparing their generated pictures with the originals

Finally, when we combined the original sketch2Monet dataset with the generated castles dataset obtained by DA, we appreciate that in cGANs, each epoch took an average time of 550 secs, with 20 epochs of training, the total time was 3 hours. Obtaining a remarkable FID score of 77.25. And now, compared with CycleGANs, the training time rocketed; each epoch took 6500 secs and trained for 20 epochs, the total time was 36 hours. Obtaining a final FID score of 36.1.

Moving on to the human inspection of the images. We presented 11 volunteers with 40 pairs of images (one generated and one real) for 1 sec per pair. This task was divided into 4 groups [cGAN castles, CycleGAN castles, cGAN monet+castles and CycleGAN monet+castles]. Please click this LINK13 to view the results chart. It can be appreciated how the amount of cGAN-produced paintings fooled the volunteers more times compared to the CycleGAN-generated pictures. 42 percent of the pictures generated by cGAN fooled the volunteers, and only 33 percent of the pictures generated by CycleGAN fooled the volunteers.

7 Conclusions

When analyzing Figure 2, we can appreciate that CycleGAN has problems generating similar paintings, and it also consumes a lot of resources and time to train. So it is not worth using this architecture to produce Data Augmentation. Regarding our Research Questions. In RQ1, we will conclude from the FID analysis that the images generated by cGAN are better; this means predictions made by the model are also more similar to the original inputs. Also, by analyzing the human analysis, we are aware that cGANs fooled volunteers more times than CycleGANs. This is also the answer for RQ2, it is not worth investing more time into training a cycleGAN, as we are technically forcing the model to perform 2 separate tasks in training 4 different models, which is counterproductive.

For RQ3, based on the FID score from Figure 2, we can also conclude that cGANs are better for Data Augmentation; their FID score metric is higher when comparing models 3 and 4. For RQ4, let's talk about the survey methodology. We interviewed 11 random volunteers and showed 40 pairs of images for 1 second per pair and asked them to classify the paintings as real or generated. As you can appreciate in this LINK14 the graph shows that cGANs fooled volunteers more times than CycleGANs, but in general, their results are not good enough as to conclude the models are effective in generating high-quality Impressionist paintings. For answering RQ5, let's take a look at figure 2 in the FID graph, as it can be appreciated when we compare model 01 and 05's FID; we can see it was reduced from 162 to 76, also comparing 02 with 06, it goes from 215 to 168. This means that using our Data Augmentation technique helped improve the FID score, which means that generated images are more similar to the original images.

References

[1] Isola, P. Zhou, T. & Efros A. A (2018) Image-to-image translation with conditional adversarial networks *In Proceedings of the IEEE conference on computer vision and pattern recognition* , pp. 1125-1134.

[2] L. A. Gatys A. S. Ecker & M. Bethge. (2015) Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*

[3] L. A. Gatys A. S. Ecker & M. Bethge. (2016) Image style transfer using convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 2414-2423

[4] J. Gauthier Conditional generative adversarial nets for convolutional face generation. Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, (5):2, 2014. 2

[5] Goodfellow, I. J. Mirza, M. Xu, B. Ozair, S. Courville, A. & Bengio, Y. (2014). Generative Adversarial Networks. arXiv. *https://doi.org/10.48550/arXiv.1406.2661*

[6] Reisinger, Don (2017) What is Pix2Pix, and How Do you use it. *https://www.tomsguide.com/us/pix2pix-faq,news-25334.html*

[7] Zhu, J. Park, T. Isola, P. & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv. https://doi.org/10.48550/arXiv.1703.10593*

[8] Lin, J. Xia, Y., Qin T., Chen Z. Liu T. Y. (2018). Conditional image-to-image translation. *In Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 5524-5532

[9] Ronneberger, O. Fischer, P. & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *In International Conference on Medical image computing and computer-assisted intervention* pp. 234-241

[10] J. Long, E. Shelhamer & T. Darrell (2015) Fully convolutional networks for semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 3431-3440

[11] Obukhov, A. & Krasnyanskiy, M. 2020, October. Quality assessment method for GAN based on modified metrics inception score and Fréchet inception distance. it In Proceedings of the Computational Methods in Systems and Software pp. 102-114

[12] Jun-Yan Zhu Taesung Park Phillip Isola &Alexei A. Efros (August 24, 2020) Monet2Photo CycleGAN Dataset http://efrosgans.eecs.berkeley.edu/cyclegan/datasets/

[13] Ledesma, Rodrigo (2022) Images repository on One Drive *https://1drv.ms/u/s!Ar9hEjcPB2vDhewD8PDajtOKXqdFtg?e=IJDbYU*

[14] BeFunky photo art editor *https://www.befunky.com/create/photo-to-art/*

[15] Images.cv Download 2K Castle labeled image dataset *https://images.cv/dataset/castle-image-classification-dataset*

[16] House, J. (2004). *Impressionism: Paint and politics.* Yale University Press.

[17] Goodfellow, I. Pouget-Abadie, J. Mirza, M., Xu B., Warde-Farley, D. Ozair, S. & Bengio, Y. (2020) Generative adversarial networks. *Communications of the ACM* pp. 139-144.

[18] Pathak, D. Krahenbuhl, P. Donahue, J. Darrell, T. & Efros, A. A. (2016). Context encoders: Feature learning by inpainting. *In Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 2536-2544

[19] Ledesma, Rodrigo (2022) Jupyter Notebooks repository on One Drive *https://1drv.ms/u/s!Ar9hEjcPB2vDhtoaZMYCLMIkvyJuMg?e=2CpH6m*

[20] Ledesma, Rodrigo (2022) Drive folder with images stored into LINKS *https://1drv.ms/u/s!Ar9hEjcPB2vDhet0ZtVaRzh4LYzbPw?e=isLJG6*[2]

[2]If there is any problem opening the images by clicking the hyperlink within the word "LINK" this folder contains all images in order of reference