Exploration of conditional generative adversarial networks

Nicolas Hirab, Omar Talbi, Imrane Belhadia,Rami Shukr
Artificial intelligence: probabilistic methods and learning techniques, TP4

Abstract

Recently, in deep learning, there has been a rise in popularity in the use of generative models, but more precisely, the Generative Adversarial Networks models (GAN) , as it is the first to demonstrate realistic outputs. Being a rather new subject, we have given ourselves the objective to deepen our knowledge with this type of neural networks. Therefore, using the PyTorch library, we implemented the *Pix2Pix* model from [Isola, Zhu, Zhou, and Efros, Isola et al.2016], which is actually a cGAN model (conditional GAN), and it allowed us to execute image to image translation. The performances we obtained with our implementation of *Pix2Pix* with different data sets and tasks were compared with each other using the FID metric (*Frechet Inception Distance*). We used tested 3 different tasks : map to aerial photo, human face to comic face and car gradients to car. We noticed that the model had more ease with the comic faces task because it had the best FID metric out of the two other tasks. For the remaining tasks, the model returned a better FID for the car gradient to car when compared to the map to aerial task. All in all, we could now say that our knowledge of the GAN (and cGAN) domain is much more enriched.

1 Introduction

1.1 GAN

The general idea of GAN models (*Generative Adversarial Network*) is to generate new images thanks to a generator and to distinguish false images from true ones with the help of a discriminator. The training of the discriminator is done through a set of predefined samples. While, the generator is trained by a random sample set, and is evaluated by its ability to deceive the discriminator. Usually, the generator is a DNN (Deconvolutional Neural Network) and the discriminator is a CNN. Both parts of the model depend on back-propagation, the generator needs it to minimize errors in image production and the discriminator uses it to improve its discriminatory abilities. Generative adversarial networks brought a new paradigm to represent loss functions. We no longer need to hand engineer our loss but instead, we can represent the loss by a zero sum game between two concurrent networks, which play against each other. The problem in the basic version of GAN, is that it is not possible to control or guide the generation of an image with some information. The loss can be written as :

$$\min_G \max_D V(D,G) = E_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + E_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (1)$$

1.2 Conditional GAN

In the cGAN (*conditional* GAN) variant, there is a label layer in the network that will help guide the image generation to control the model's outputs. The generator model tries to output images as realistic as possible, while the discriminator try to penalize the latter if the images are not realistic enough. Jointly trained, the two models can achieve optimistic results in their respective tasks.

1.3 Pix2Pix

In the *Pix2Pix* model of [Isola, Zhu, Zhou, and Efros, Isola et al.2016], their version of GAN differs greatly from the basic one. Instead of using a mlp for the generator, they use U-Net architecture and for their discriminator a PatchGAN (*convolutional PatchGAN classifier*). This last one works by scanning the generated image and classifying the N by N pieces of the image, once all the pieces that form the image have been classified, the average of their answers will determine the output of the discriminant for the whole image. This type of prediction results of the assumption of an image as a Markov random field with of pixel being considered as independent from other pixel that are positioned outside of an N x N window. Using path gans allows to control the scale in the image that is penalized and helps better capturing local statisitcs [Isola, Zhu, Zhou, and Efros, Isola et al.2016].

We will provide, in this report, our implementation of the Pix2Pix [Isola, Zhu, Zhou, and Efros, Isola et al.2016], a review existing solution. We will then report the experiment we made and provide an analysis of the results.

2 Theoretical approach

Since we have reproduced the results of [Isola, Zhu, Zhou, and Efros, Isola et al.2016], our theoretical approach is

founded on this work. They highlighted the specificity of conditional GANs relative to GANs as taking some information in the input of the Generator and the Discriminator. The generator of cGans takes some random noise vector as input as well as some observed data in order to generate examples in the output space. Since cGans generators takes input from a certain distribution and output data sampled from another one, they can be thought of as performing "translation". The additional information allow to perform "mode selection"[3] on the learned probability distribution therefore resulting in conditioned generation of samples.

The learning of cGans is, as for GANs, performed by playing a minimax game between a discriminator and a Generator. The difference is that the generator will be inputted some information in addition to the random noise z used in traditional GANs and the discriminator will take as input both the data and a target to predict the correctness of the translation between input and target. Both information are generally concatenated in the channels. The evaluation function for the training can be rewritten as :

$$\begin{aligned}\mathcal{L}_{cGAN}(G,D) =& E_{x,y}[\log D(x,y)]+ \\ & E_{x,z}[\log(1-D(x,G(x,z))]\end{aligned} \tag{2}$$

We can compute this loss in Pytorch with the BCE with logistic loss given by :

$$\begin{aligned}&\text{(x, y)=L}=\{l_1,\dots,l_N\}^\top, \\ &l_n = -w_n\left[y_n \cdot \log \sigma(x_n) + (1-y_n)\cdot \log\left(1-\sigma(x_n)\right)\right]\end{aligned} \tag{3}$$

Where setting y to 0 or 1 gives us the terms we're interested in for computing each loss. Notice that in our implementation, we adjust G's objective function to maximize $log(D(G(x,z))$ instead of minimizing $log(1 - D(x,G(x,z)))$. As cited in [7] , this makes it easier for the generator to learn, because there is a much higher gradient signal in the beginning of training when samples are bad.

The quantitative evaluation of gans is a known challenge, and, specifically, with the conditional gans, the goal is to find a way to measure the ability of the generator to use it's input in order to generate realistic "translations". For the evaluation of gans, classification algorithms (trained on large dataset e.g. ImageNet) are generally used in order to measure the performance of the generator. Inception or FCN score was not a viable option due to the nature of our data set for the comic faces and the maps, since we trained our models on generating images that were not in ImageNet [Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, and Fei-Fei, Russakovsky et al.2014]. We the FID score, which is based on statistics on both labeled data and generated data in order to evaluate our models.

2.1 Architecture

In order to ensure the capacity of the model to map image of high resolution to image of high resolution of another probability distribution, an adapted version of U-net was used for the Generator, which consists of a fully CNN with with skip connections between layers i and $n-i$ where n is the number the layers.

layer down	initial	1	2	3	4	5	6	btlneck
module	CL	CBL	CBL	CBL	CBL	CBL	CBL	CONV
size	3*64	64*128	128*256	256*512	512*512	512*512	512*512	512*512
layer up	1	2	3	4	5	6	7	finale
module	DBR	DBR	DBR	DBR	DBR	DBR	DBR	deconv
size	512*512	1024*512	1024*512	1024*512	1024*512	1024*256	512*128	128*3

Table 1: architecture of the generator used for the experiments. Cl is for convolutions, CBL is for convolution-BatchNorm-leakyRelu, and DBR is for deconvolution-BatchNorm-Relu. We used dropout on the bottleneck with convolutions. layers on a same column have a skip connection from down to up layers.

In order to avoid blurry results, we used a $L1$ loss. With this loss, we used patchGAN discriminator [Isola, Zhu, Zhou, and Efros, Isola et al.2016], in order to regularize the model. The discriminator classifies patches of the image as real or fake, and all outputs for the patches are used to compute the global output. The use of patches instead of the whole image tends to speed up the computation for the discriminator.

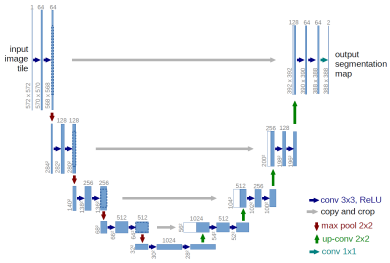

Figure 1: illustration of U-Net[6]

For the generator the first part of the U-net is made of modules with convolution-BatchNorm-leakyRelu (CBL) with 6 layers and the second part after the bottleneck is made of 7 layers with module deconvolution-BatchNorm-Relu(DBR) as shown in table 1. The discriminator is a CNN with the same CBL modules as the generator.The discriminator has 4 modules in a row with no fully connected layers at the end since predictions are made for patches on the image. Full details of the implementation can be found in the GitHub.

3 Experiments and results

We trained our implementation of conditional GANS on three datasets:

- map to aerial photo, trained on data obtained from Google Maps
- human face to comic face, where the faces were generated by another GAN.
- car gradients to car

The maps data set was used in Isola and al. [Isola, Zhu, Zhou, and Efros, Isola et al.2016] and was used to test our implementation of the pix-to-pix architecture on similar tasks. We trained pix-to-pix on face to comics translation as a second task. The natural faces were generated by GANs. For the third task we used the Stanford cars data set [??, sta].

We applied an edge detector on the cars images and trained the model to generate image of the cars with the edges as input. We then evaluated this model on sketches of cars. https://www.overleaf.com/project/60822a91215ca5a789f57f12

3.1 Qualitative Results

The visual results obtained on the comics dataset were very concluent. At epoch 40, we were able to obtain visually conclusive results. The performance of our implementation on the comics data set outperformed the other data sets.

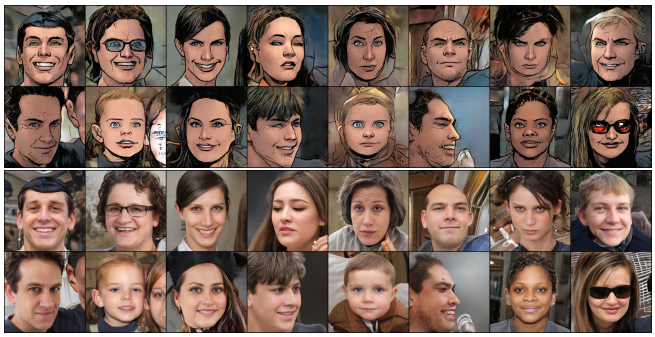

Figure 2: Results on the comics dataset on epoch 40

As for the Stanford car data set, we were able to obtain acceptable results. After 190 epochs of training, the model was able to give a recognizable picture of the car. We also conditioned on car sketches and were able to obtain similar results as the ones obtained with the cars data set.

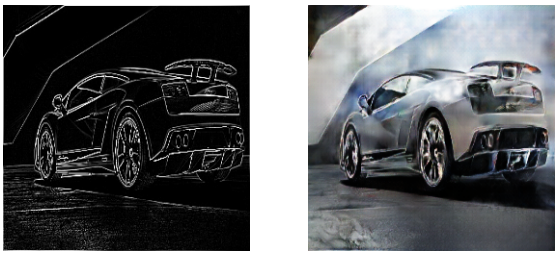

Figure 3: Results on cars data set epoch 190

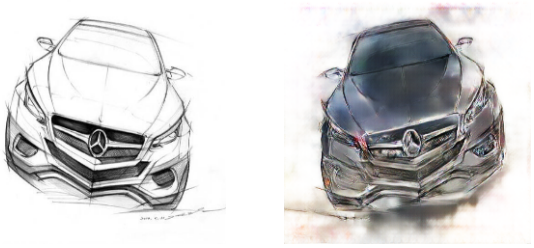

Figure 4: Results on a car sketch

On the Aerial-Maps data set, the model performed fairly well on some cases. However, it has proven to be unstable as we applied modifications to the input condition. In fact, while a given input would generate a good approximation of the target image, inverting along the x-axis of the input condition would generate a completely different approximation.

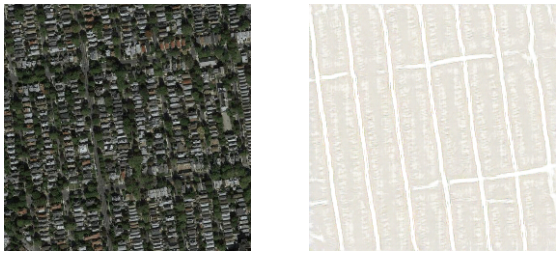

Figure 5: Results on the aerial Google Maps

3.2 Quantitative results

We evaluated several samples from the 3 data sets we explored. Unfortunately, there exists no globally accepted benchmark metrics for evaluating the performance of a GANs. However, there exists some metrics that can evaluate certain characteristics indirectly, enabling a meta measurement for evaluating and comparing GANs. The first one we used was the the Frechet Inception Distance, which embed a set of images in a feature space, given by a specific layer of an Inception Net, or CNN. It then uses this layer to calculate its mean and standard deviation, both for the real data and the model distribution and calculate the Wasserstein metric between them. The equation is given by :

$$\text{FID(r, g)}=\|\mu_r-\mu_g\|^2+T_r\left(\sum_r+\sum_g+2\left(\sum_r\sum_g\right)^{\frac{1}{2}}\right) \quad (4)$$

where $(\mu_r;\sum_r)$ and $(\mu_g;\sum_g)$ are the mean and covariance of real and generated data respectively.

The scores we calculated are correlated with the qualitative evaluation of the images. The FID score obtained

FID Score			
Normalization	Face2Comics	EdgeToCar	MapsToAerial
Batch Norm	**62.17**	**96.33**	**292.66**
InstanceNorm	**84.22**	-	-

Table 2: FID Scores calculated on samples generated by the model

4 Critical analysis of the learning approach

As a reminder, our goal was to gain experience in the implementation of models, and in particular CGANS with Pytorch. In order to understand more in depth the subject of CGANS, our first intuition was to re-read the articles presented in class related with the subject. These include U-nets, deep convolutional GANS, and the original paper about GANS. Coding the network presented in [Isola, Zhu, Zhou, and Efros, Isola et al.2016] allowed us to comprehend better how the different parts of the network interact with each other. We then tried to adapt our implementation so that we could train more

effectively and generate better results. Once the model implemented, the next step was to establish evaluation metrics so that we could quantitatively assess the quality of the generated samples and the performance of our generator. In order to do so, we instinctively tried applying the evaluation metrics cited in the original paper. However, those metrics were not applicable to our data sets as they were using the FCN score and Amazon Mechanical Turks(AMT).The score is based on the FCN-8 [Shelhamer, Long, and Darrell, Shelhamer et al.2016] architectures which is used for semantic segmentation. This quantitative metric is generated through the classification accuracy of the synthesized images over the labels. Nonetheless, our data sets didn't provide a semantic segmentation of the labels, which meant we wouldn't be able to train the FCN network to classify the synthesized images.

This lead us to explore alternative metrics such as the Frechet Inception Distance (FID). This score enable us to compare the distribution of the generated image to the distribution of the real image.The bigger the score, the bigger the contrast between the two distribution. Despite that, it would have been interesting to try implement the FCN score as it would have added a new informative metric. We could also have tried a hyperparameter tuning in order to generate better performances.

References

[??, sta]

[Isola, Zhu, Zhou, and Efros, Isola et al.2016] Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros (2016). Image-to-image translation with conditional adversarial networks.

[Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, and Fei-Fei, Russakovsky et al.2014] Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2014). Imagenet large scale visual recognition challenge.

[Shelhamer, Long, and Darrell, Shelhamer et al.2016] Shelhamer, E., J. Long, and T. Darrell (2016). Fully convolutional networks for semantic segmentation.