Deep Learning

Generative Adversarial Networks (GANs)

Paul GRIGNON - Benjamin JOLY

12 Mai 2020

Generative Adversarial Networks (GANs), developed by I. Goodfellow (2014), were presented by Yann LeCun, vice-president and chief AI scientist at Facebook, as "the most interesting idea in the last 10 years in Machine Learning". This architecture allows to build generative models, i.e. models able to create new data by themselves. It puts at stake two networks, named Generator and Discriminator, which are in competition. In fact, the generator aims at generating designs as close as possible to the real ones, while the discriminator's role is to distinguish true from fake designs that are submitted to him. In this setting, the discriminator has to be more and more acurate in discriminating true and false designs, while the generator's goal is to mislead the discriminator by generating images more and more real. After this "double optimisation" is done, only the generator is used to generate new realistic images. After the initial papers presenting GANs, they have been the subject of many recent developments in Deep Learning research, giving rise to many different new models (for example StyleGAN, StarGAN, SRGAN, DC-GAN, Wasserstein GAN, CycleGAN).

Among others, GANs have been used to develop unsupervised techniques in computer vision, which have not been extensively developed as compared to supervised techniques (A. Radford, 2015). However, given the scarcity of situations where images are labeled, unsupervised algorithms are key in order to take advantage of the large unlabelled datasets. To do so, A. Radford (2015) propose to use GANs in order to obtain a good intermediate representations of images. More precisely, parts of the generator and discriminator could act as feature extractors for supervised tasks. In this perspective, they introduce Deep Convolutional GANs (DCGANs). Given the objective at the heart of unsupervised learning to estimate an underlying probability distribution, M. Arjovsky (2017) propose a GAN aiming at finding such a result when classical approaches based on the maximum likelihood can't be used. After explaining how these models based on GANs introduce new breakthroughs in unsupervised learning, we here focus in a second part on a new GAN-based architecture for image-to-image translation task allowing to work directly with unpaired data : CycleGANs. In fact, paired data, which underlies a defined matching for the observations of one dataset with those of another one, is scarse. Such a model allowing to directly work with unpaired data thus allows to extend the possibilities of image-to-image translation tasks to a wider range of datasets. This last architecture is implemented is Python [1], and is applied for an image-to-image translation task of images from the MNIST dataset to the USPS dataset. The three models mentionned (DCGANs, Wasserstein GANs and CycleGANs) are thus ways to extend the aplications of computer vision algorithms to unsupervised settings where there are no labels or no paired data.

1. We used Google Colab

1 Literature review

1.1 DC-GAN

Motivation and research question

Given the lack of literature on unsupervised learning with CNNs in computer vision despite of their success in supervised learning tasks, the article aims at incorporating CNNs in unsupervised learning problems. More precisely, A. Radford (2015) propose to combine CNNs with GANs, which make possible the learning of good intermediate representations of images and which could act as feature extractors for supervised learning tasks. Focusing on the generation of natural images of the real world, the authors thus propose a parametric model underlying the scaling up of GANs with convolutional neural networks (CNNs), despite the unsuccessful attempts in previous research to combine both architectures. Given the important issue of instability during training of GANs, this criterion is carefully taken into account for the assessment of the resulting model, in order to address this limitation.

Model architecture

The network architecture described includes 3 main changes to the CNN architecture. Firstly, DCGANs don't involve any pooling layer. Pooling functions are replaced in the generator, with fractional-strided convolutions, and in the discriminator, with strided convolutions. Such a modification allows to increase the extent of expressiveness of the model[2]. Secondly, no fully connected layers are introduced : the highest convolutional features are directly connected to the input of the generator (the first layer taking as input a uniform noise distribution Z reshaped into a 4-dimensional tensor), and to the output of the discriminator (after being flattened, the last convolution layer is fed into a single sigmoid output).

The third specificity is related to batch normalization, which is used for all layers except for the generator output and discriminator input layers. In fact, if allows to stabilize learning given the high instability of GANs during training, but two layers are nonetheless excluded from this modification because applying it to all layers could engender "sample oscillation and model instability". Such a normalization can also allow to improve the training speed, and reduces covariance shift issues, which are due to the changes in the hidden units' values.

Lastly, the authors use a ReLU function for all layers in the generator except for the output one, for which the Tanh function is applied. The restricted range ([-1,1]) here allows a quicker learning. In contrast to the maxout activation used in the original GAN paper by I. Goodfellow (2014), a LeakyReLU (leaky rectified activation) function is used for the discriminator, and is shown to work especially well for high resolution modeling. The slope of the leak is set to 0.2 for the model training.

To conclude, these changes overall allow a more stable and faster learning in training, and also allows to get rid off internal covariate shift.

Training

3 datasets are used to train the DCGAN : a database of faces scraped from random web queries on which an OpenCV face detector was run (leading to select 350 000 faces), the LSUN (Large-scale Scene Understanding) dataset (including 3 million images for training), and Imagenet-1k (incorporating natural images for unsupervised training). The images were onlu scaled to the range of the tanh activation function [-1,1]. The models were trained with a mini-batch stochastic gradient descent (with a mini-batch size of 128) and a normal distribution with mean 0 and standard deviation 0.02 was used to initialize weights. Whereas momentum was used in previous GANs to accelerate training, the authors here use an Adam optimizer with tuned hyperparameters. Considering that the default learning rate of 0.001 is too high, the authors here choose a value 0.0002. Lastly, the value of the momentum term β_1 is turned from 0.9 to 0.5 to lower instability.

2. J. T. Springenberg (2014) shows that whereas simply removing max-pooling layers and increasing the stride of previous layers lowers the model performance, the replacement of all max-pooling operations by an additional convolution layer with strides r=2 allows an improvement with regard to the base model considered.

Results

Regarding the predictive performance of the model, the authors extract the features trained on the Imagenet-1k dataset and use the discriminator's convolutional features from every layer. They obtain a 4×4 spatial grid by implementing maxpooling on each layer, which is then flattened and concatenated to obtain a 28 672 dimensional vector. They then use this output to train a regularized linear L2-SVM classifier. The accuracy obtained on the CIFAR-10 dataset, used for testing, is 82.8%, which is found to be higher than all K-Means based approaches previously implemented on the CIFAR-10 dataset used for testing, but lower than for Examplar CNNs. In a classification task performed with a regularized linear L2-SVM classifier trained on features selected in a similar way as previously on another database, the SVHN dataset [3], the results obtained were better than those obtained with another modification of CNNs made to take into account unlabeled data. Also, training a purely supervised CNN with the same architecture on the same data led to a significantly higher validation error, showing that the key factor of DCGAN's performance is not the CNN architecture. The authors also assess the presence of overfitting and memorization of training samples and intend to limit memorization issues by doing image de-duplication on the LSUN dataset [4].

Lastly, a main contribution of the article is to contribute to improve interpretability of such networks, given the scarcity of works aiming at understanding and visualizing the mechanisms characterising GANs and the intermediate representations of multi-layer GANs. First, they propose to "walk in the latent space", since the latent space can give insights about possible memorization patterns or about the presence of relevant and interesting representations (this tends to be the case when walking through it results in semantic changes to the image generations, for example with the addition/removal of objects). They then propose to understand the representations learned by the discriminator, by using guided backpropagation. As compared to a baseline for randomly initialized features which don't activate on anything semantically relevant, the authors show that the features learned by the discriminator of a DCGAN trained on a large image dataset can return a hierarchy of features, since the features learnt activate on typical parts of the bedroom (e.g. beds, windows). Lastly, they propose different protocols aiming at understanding what the generator learns. Since the quality of samples suggests that the generator learns specific object representations for major scene components such as windows or beds, the authors test the impact of deleting some objects on image generation. Choosing the object "windows", they first build a dataset by deleting windows from images [5]. A prediction task is then applied to study the samples generated on the basis of the datasets with and without this feature map removal. It is found that when the windows are deleted, the network tends to forget to draw windows in the bedrooms, and that they are replaced by other objects. Thus, specific filters seem to have learnt to draw specific objects. Nonetheless, the composition of the scene stays similar, which leads to conclude that the generator well differenciates scene and object representations. Also, the authors also show that the generators are characterized by interesting arithmetic properties, which opens the possibility to easily manipulate the semantic qualities of the generated samples.

To conclude the authors find a set of architectures leading to good results in terms of training stability on different datasets by combining CNNs with GANs. For the first time, the combination of these two models lead to very positive results, thus being a new breakthrough in unsupervised learning for computer vision. The use of the trained distriminators for image classification tasks has been shown to result in competitive perfomance, and the model allows to reach a stable and fast learning process. Finally, by focusing on the interpretability of their model, the authors show evidence that the DCGAN manages to learn a hierarchy of objects, by managing to disentangle objects from scenes. Another recent breakthrough in order to develop computer vision in unsupervised learning tasks is proposed by M. Arjovsky (2017), with their Wasserstein GAN.

3. StreetView House Number dataset (SVHN) (Y. Netzer, 2011)
4. It allowed them to detect and remove 275 000 near duplicates
5. To do so, they use a selection of 150 samples on which bounding boxes are manually drawn. This will serve as labels in a logistic regression fit is applied to the second highest convolution layer features to predict if the feature activation is on window or not (label 1 : activation in drawn boxes, label -1 : activation outside boxes). Feature maps with a positive activation are then deleted, resulting in images without windows.

1.2 Wasserstein GAN

Research question and motivation

M. Arjovsky (2017) focus on an unsupervised learning framework, where the objectve is often to learn a probability distribution. Whereas a typical answer is to learn the probability density through the maximum likelihood function which asymptotically minimizes the Kullback-Leiber divergence, this approaches raises an issue when the support of the distribution is made of low dimensional manifolds. In fact, in such cases, there is a very low likelihood that the true support of the distribution and the model manifold have a non-negligible intersection, and this engenders the Kullback-Leibler distance to be undefined or infinite.

A solution allowing to apply the likelihood approach that has been used is the addition of a noise component to the model distribution, but it is in fact incorrect. In order to propose an alternative correct approach allowing to represent distributions characterized by their low dimensional manifold, the authors here propose an alternative framework. Whereas approaches such as Variational Autoencoders (VAE) rely on the likelihood approach, GANs are much more flexible and are chosen as a basis for a new approach to solve the issue.

The Earth-Mover distance

The authors define an alternative approach based on the definition of a random variable Z with fixed distribution $p(z)$. By defining X as a compact metric set and $prob(X)$ as the space of probability measures on X, the authors then propose to apply a parametric function $g_\theta : Z \to X$, the output giving a probability distribution P_θ. In order to measure the extent to which the model and real distributions are different, the authors focus on the best way to specify a distance $\rho(P_\theta, P_r)$, where P_θ is the model distribution and P_r the real one (P_θ,$P_r \in Prob(X)$). Given a chosen distance ρ between two distributions, the authors want a continuous loss function $\theta \to \rho(P_\theta, P_r)$. Equivalently, this requires the mapping $\theta \to P_\theta$ to be continuous, and they thus argue that a distribution P_θ satisfacing this property must be looked for. By comparing different distance measures, the authors define a new type of GAN (called Wasserstein-GAN - WGAN) minimizing the approximation of the *Earth-Mover (EM)* distance, which is presented in equation 1. Such a distance both allows to take into account distributions with low dimensional manifolds and to solve the main problems faced by GANs with respect to training. The main impact of such a modification of the distance relates to the convergence of the sequences of distributions. In equation 1, $\Pi(P_r, P_g)$ represents the set of the joint distributions $\gamma(x, y)$ whose marginal distributions are P_r and P_g. The EM distance intuitively represents the cost of optimal transport to transform the distribution from P_r to P_g.

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim y}[\ ||x - y||\ ] \tag{1}$$

Over low dimensional manifolds, EM distance allows to satisfy the continuity of the mapping $\theta \to P_\theta$, and thus to obtain a continuous loss function $\theta \to \rho(P_\theta, P_r)$, while it is not the case for the other distances considered (e.g. Jensen-Shannon (JS) distance). It thus allows to obtain a usable gradient everywhere, which will allow to train the model until the obtention of the optimal parameters.

Regarding the conditions for the EM distance to be continous on θ, the authors find that few are required, and that these conditions don't allow continuity for other measures of distance [6]. These conditions being satisfied for feedforward neural networks, the EM distance appears to be continuous everywhere and differentiable almost everywhere, and seems thus to be adapted to neural networks. Also, EM being weaker than other distances (JS, TV, and KL in increasing order), the former is the only one whose cost function is sensible in the learning of distributions with low dimensional manifolds. Thus, the continuity of EM for statements verified in neural networks and the sensibility of its cost function in the learning of distributions with low dimensional manifolds make it particularly adapted to solve the problem identified.

Approximation of the optimization of the EM distance

6. This can be put in parallel with the fact that the EM distance has been shown to be weaker than the JS distance, i.e. that it is easier for a sequence of distribution to converge with the EM distance.

The Kantorovich-Rubinstein duality allows to obtain equation 2 from equation 1. The supremum is taken over all 1-Lipschitz functions f.

$$K * W(P_r, P_g) = \sup_{||f||_L \leqslant K} E_{x \sim P_r}[f(x)] - E_{x \sim P_g}[f(x)] \tag{2}$$

By considering a parametrized family of functions $\{f_w\}_{w \in W}$ all K-Lipschitz for some K, we can replace $||f||_L \leqslant 1$ by $||f||_L \leqslant K$ and the problem can be rewritten as expressed in equation 3 to obtain the calculation of $W(x, y)$ up to a multiplicative constant. Also up to a constant, $W(x, y)$ could be differentiated by backpropagation in equation 2 (with K = 1) to obtain equation 4. This last result allows to differentiate the approximation of the EM distance, and thus to optimize the neural network.

$$\max_{w \in W} E_{x \sim P_r}[f_w(x)] - E_{z \sim P(z)}[f_w(g_g(z)] \tag{3}$$

$$\nabla_g W(P_r, P_g) = -E_{z \sim P(z)}[\nabla_g f(g_g(z))] \tag{4}$$

WGAN algorithm

In order to find the appropriate function f maximizing equation 2, the authors enforce a Lipschitz constraint by restricting weights w to a compact space W. To implement this idea, the WGAN algorithm thus adjusts weights within a fixed box after each update of the gradient. Regarding the choice of the clipping parameter, it underlies a tradeoff since a too high value would lead to extend the time necessary for weights to be optimized, whereas a too low value could engender vanishing gradient issues. The restriction on weights limits the increase of the « critic » f_w and makes it at most linear in some parts. This ensures its convergence towards a linear function, which guarantees the exploitability of gradients everywhere for every part of the space. Also, because of the continuity and differentiability of the EM distance, the critic can here be trained until optimality is reached. The training of the critic to completion provides a loss to the generator that can be trained. The better the critic, the higher the quality of the gradients used to train the generator.

To conclude, this new algorithm WGAN thus allows to reach a better training stability, while allowing to learn the probability density in cases where the maximum likelihood principle can't be applied (when the support is made of low dimensional manifolds). It also allows to work with a loss function which is meaningful, by being correlated to the degree of convergence of the generator, and thus to the quality of the images generated.

1.3 Conclusion

To conclude, both articles show that GANs seem to be a promising framework for unsupervised learning tasks in computer vision by acting as feature extractors to consecutively apply supervised tasks, and by allowing to measure distance between distributions when the maximum likelihood principle can't be applied. In addition to the unlabeled aspect of image data, another problem related to the specific task of image-to-image translation relates to the absence of established pairs of images, and thus of paired data. In order to deal with the majority of cases where data is unpaired, a method called CycleGAN has been developed, and will be described and implemented in the next section.

2 CycleGAN : model and implementation

2.1 Presentation of the model

Motivation and research question

CycleGAN allows to perform image-to-image translation, which aims at learning the mapping between an input and an output image in order to change some desired features of its appearance. To train such a model, paired examples would traditionally be need (i.e. the use of $\{x_i, y_i\}_{i=1}^N$ for which each pair x_i and y_i are matching. Datasets with paired examples are in reality scarce, and the *CycleGAN* model here proposes to apply image-to-image translation on unpaired data, that is to say that there is no recognized matching relationships between elements of the two sets $\{x_i\}$ ($x_i \in X$) and $\{y_j\}$ ($j_i \in Y$). Thus, the CycleGAN aims at learning a mapping between two image collections rather that between two specific images.

Approach and model

In order to learn a mapping G : X $\rightarrow$ Y such that the output $\hat{y} = G(x)$, with $x \in X$, are not distinguishable from images $y \in Y$, the authors add further specifications to ensure that the mapping is done in a meaninful way. In fact, there is an infinite number of mappings G that could lead to the same distribution over $\hat{y}$. To constrain the mapping, the authors add the property of cycle consistency : translating from A to B and then from B to A should lead to the same initial A. To do that, we couple the mapping G : X $\rightarrow$ Y with an inverse mapping F : Y $\rightarrow$ X[7]. The mappings G and F are thus simultaneously trained. In order for the translated images to not be distinguished from images in the target domain, 2 GANs are introduced, and there are thus two adversarial losses (one per domain). D_X is the discriminator aiming at disentangling between {x} and {F(y)}, and D_Y between {y} and {G(x)}. The corresponding adversarial loss for the mapping function G :X$\rightarrow$Y and the discriminator D_Y can be written as expressed in equation 5.

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[1 - log D_Y(G(x))] \tag{5}$$

In order to avoid the learned mappings G and F to be contradicting each other and for the cycle consistency hypothesis to be satisfied, a cycle-consistency loss aiming at ensuring $F(G(x)) \approx x$ (forward cycle-consistency) and $G(F(y)) \approx y$ (backward cycle-consistency) is introduced. The expression of this loss function is given in 6.

$$\mathcal{L}_{CYC}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \tag{6}$$

The full objective is given by the overall loss expressed in equation 7, where the parameter λ influences the relative importance given to the two objectives.

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{CYC}(G, F) \tag{7}$$

Finally, equation 8 gives the program that has to be solved.

$$G^*, F^* = \underset{G,F}{argmin} \underset{D_X, D_Y}{max} \mathcal{L}(G, F, D_X, D_Y) \tag{8}$$

Applications

The applications of CycleGAN are numerous. Such an architecture can for instance be used for photo enhancement (e.g. generate photos aligned with some changed properties), for collection style transfer or for the conversion of paintings into photos. It has also been used to change an object on a photo (object transfiguration) or for changing the season expressed on an image (season transfer). It has interesting applications in driving, by allowing to simulate driving scenes in different conditions (e.g. day and night).

7. Thus, F and G should be inverses of each other, and both mappings should be bijections

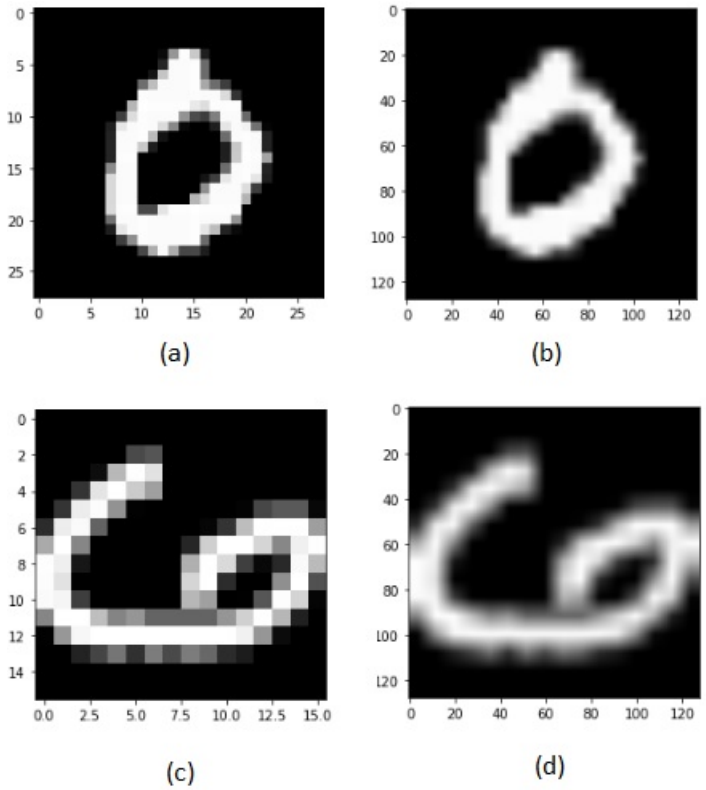

Figure 1 – Example of original and upscaled image for the MNIST dataset (resp. a and b) and for the USPS dataset (resp. c and d)

2.2 Implementation of CycleGAN

Research question

In our implementation of CycleGAN, the image-to-image translation task consists in translating handwritten digits from the MNIST [8] to the USPS [9] dataset.

Data preparation

The MNIST dataset contains 60 000 examples for training, as well as a test set of 10 000 examples. There has been a size normalization of digits, which also have been centered in images of fixed size. The images are in greyscale, and their size is 28 $\times$ 28. Regarding the USPS dataset, the training set contains 7 291 greyscale images, and the test set 2 007. The images of this second database have a size of 16 $\times$ 16 pixels.

As the images of the two datasets used didn't have the same dimensions (resp. 28 $\times$ 28 and 16 $\times$ 16 pixels), and because the CycleGAN here took as input 128×128 images, we bilinearily upsampled images to reach a size of 128 $\times$ 128 [10]. For the MNIST and USPS datasets, examples of original (resp. a and c) and upscaled images (resp. b and d) are presented in figure 1.

All the data was imported into a common file *MNIST_to_USPS* used by the CycleGAN for training. Within this file, the resized images were located in files *trainA* and *testA* for the MNIST images, and in *trainB* and *testB* for the USPS images [11].

Model architecture

8. Source : http ://yann.lecun.com/exdb/mnist/
9. Source : https ://www.kaggle.com/bistaumanga/usps-dataset
10. Z. Murez (2018) for example also bilinearily upsampled images while working on these 2 datasets.
11. For each dataset, the distinction between train and test was kept in order to save a test set.

We use a CycleGAN taking inputs of size $128 \times 128 \times 3$. The model includes two discriminators (D_X and D_Y) as well as two generators ($G_{X \to Y}$ and $G_{Y \to X}$).

The generator is composed of an encoding part (built with convolutional layers), a transformer part, and a decoder part (the two later parts are created using upsampling layers). Regarding the discriminator, it consists in a standard convolutional neural network (*Implementing CycleGAN Using Python*, 2019).

The discriminator and generator losses are minimized in order to reach optimality. The generator loss includes the cycle-consistent loss, which ensures that the network is able, once an image is translated to Y from X, to come back to domain X with a result very similar to the original image. The relative importance of the cycle-consistent loss can here be changed through the parameter λ_{Cycle}.

Training

We use a batch size (number of samples to work through before updating the internal model parameters) of 1, as used in J. Y. Zhu (2017). The updates are thus frequent, which can be insightful regarding the model evaluation and how it improves, and can also lead to a faster learning process. On the contrary, an increased batch size underlies higher memory requirements, and didn't appear to be possible given our resources on Google Colab (RAM = 12.72 GB).

Regarding the number of epochs (number of times the learning algorithm goes through the whole dataset), we used 10 epochs This number has also been constrained by the time of execution of one epoch, since each epoch takes approximately 28 min (2012.5 seconds) to be completed. The number of epochs should be enough important in order to reach the convergence of the training error, but a too high number could lead the model to overfit. By looking at the shape of the error, we see that the loss function regarding the generator and denominator seem to have both almost converge after 10 epochs (figure 2). However, this feature could be only apparent, and hide in reality a slowly decreasing loss.

For each epoch, a batch of images is read and pushed to the generator which generates fake Y images from X

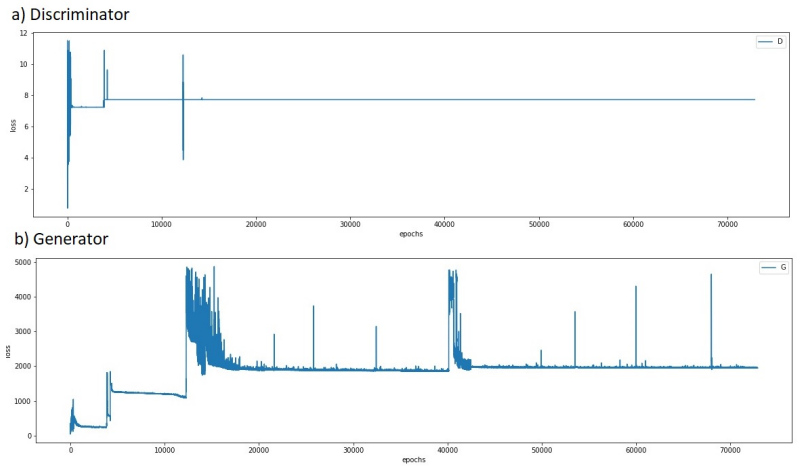

Figure 2 – Discriminator and generator loss function on the 10 epochs

($G_{X \to Y}$) and fake X images from Y ($G_{Y \to X}$). The fake images produced allow to train the discriminators, which have to disentangle fake from real image and are then given feedback about the answer. The discriminator's performance allows to train the generator, which aims at proposing images as close as possible to reality in order to mislead the discriminator.

Results

The results obtained after training the CycleGAN with 10 epochs is displayed in Figure 3. We see that while the number is somewhat well translated at the first epoch, this pattern then surprisingly deteriorates at the 10th epoch. We propose different explanations for this phenomenon in the *2.3. Limitations* section.

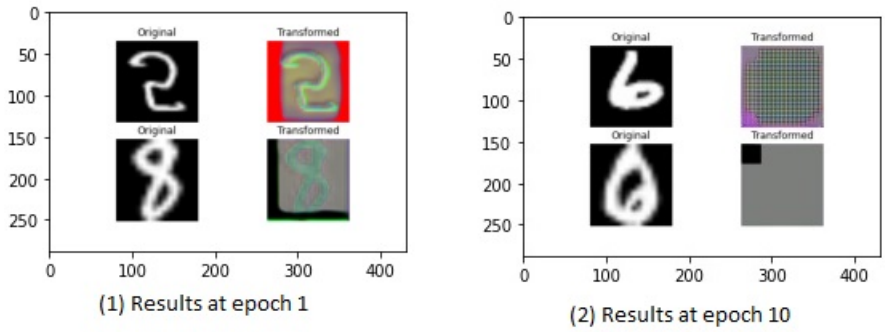

Figure 3 – Results of the image-to-image translation at the 1st and 10th epoch

Model interpretability : tests

A. The role of the cycle-consistency loss : impact of the variation of λ_{Cycle}

We test the impact of varying the relative importance of the cycle-consistent loss component (i.e. $\lambda_{Cycle} = 0$ and $\lambda_{Cycle} = 30$). To do so, we train the CycleGAN model with an unilateral modification of the λ_{Cycle} from 10 to 0 (i.e. corresponding to a model with no cycle-consistent loss component) and another modification from 10 to 30 (corresponding to an increased omportance of this component), in both cases on 5 epochs. The results of the tests are presented in Figure 4. We have the impression that the discrimator converges immediately to a value close to 8. This feature is strange because it is linked to the generator, which faces convergence issues. We suggest that in the baseline, test1 and test4 contexts we didn't manage to get the optimality because the decrease is very slow and we need more epochs to reach it. The change in λ_{Cycle} from 10 to 0 seems to have surprisingly no deleterous effect on the losses. As it is equivalent to neutralizing the cycle-consistent loss component, this seems to show the uninfluence of the cycle lambda. That's quite unexpected ! On the opposite, when λ_{Cycle} is increased relatively to the default value of 10 (to 30), we observe that the behavior of the generator loss deteriorates, since this loss increases step by step with the number of epochs.

B. Optimizer and learning rate

The optimizer Adam is here used as it is computationally efficient and has very little memory requirement. Adam is a combination of Adagrad and RMSprop. While the default learning rate is 0.001, the authors choose to specify it as 0.0002. Regarding the exponential decay rate for the first moment estimates, the default value is set to 0.9, but the authors use 0.5. We here propose to test the impact of a unilateral deviation of the learning rate to higher values, in order to assess its impact on the training process. More specifically, we implemented a training on 5 epochs with a learning rate of 0.001 and another with a learning rate of 0.01. One possible enhancement of the current model could include specifying a varying learning rate, which would decrease during training. It could include the specification of a learning rate warmup. The results of these tests are presented in Figure 4. Tests 3 and 4 aim at assessing the impact of a higher learning rate. We see that a higher learning rate seems to lead to worse results as compared to the baseline, since the level of the generator loss is higher and even increasing with the number of epochs when the learning rate is at the highest value tested (i.e. 0.01). Thus, this confirms that the value of 0.0002 initially chosen is appropriate and gives good results.

Evaluation of the model

Among the existing methods proposed to assess the CycleGAN results, those based on human evaluations can first of all be distinguished. In fact, J. Y. Zhu (2017) evaluate the extent to which the generated images are credible through "real vs fake" perception studies. In such a method, a sequence of image pairs composed of a real and a fake photo is shown to participants, and the task is to recognize the real image. On the other hand, automatic quantitative measures can also be used. In particular, J. Y. Zhu (2017) propose to use the FCN (fully-convolutional network) score, which is an automatic quantitative measure allowing to assess the

extent to which the generated photos are interpretable according to an off-the-shelf semantic segmentation algorithm. Semantic segmentation techniques allow to derivate a human-like ability to understand the contents of an image [12]. Fully convolutional networks is only one type of semantic segmentation network [13]. Such a method could allow to predict a label map for each generated photo. A comparision of the true labels with this label maps using semantic semgmentation metrics could then allow to determine the extent to which the CycleGAN succeeded to generate an image corresponding to the real label. For example, a photo that should represent a "5" has been well generated if the FCN applied to the image created is able to recognize that the number displayed is in fact a 5. This method has however not been implemented here because of the bad quality of images generated (we don't need any algorithm to make this diagnosis) and because of time constraints. To finish, other evaluation methods could encompass relative measures deducted from the comparison of the performance of our CycleGAN model with some other baseline models (J. Y. Zhu, 2017).

2.3 Limitations

Quality of our results

A limitation of the image-to-image translation tasks relates to the low quality of the generated images. This could be due to the fact that the number of epochs used is insufficient for the model to converge towards the optimum. However, by looking at the loss functions, they seem to have converged before the 10th epoch, but it could in fact not be true and reflect a very slow convergence. It can be noticed that the generator uses color whereas it is here not needed. The problem could thus come from *matplotlib.imread*, which, based on the shape of the tensor, automatically converts images (.,.,3) into an RGB format (while here images are in black and white).

Computational resources

The use of Graphics Processing Units (GPU) seems to be an essential part of any Deep Learning task, since Deep learning architectures require high computational resources by stacking many layers together. Such a setting allows to perform parallel computations in a very fast way, could thus allow to decrease the time necessary for training and thus to perform a higher number of epochs, which could allow us to reach a convergence point with better results.

12. https ://medium.com/100-shades-of-machine-learning/https-medium-com-100-shades-of-machine-learning-rediscovering-semantic-segmentation-part1-83e1462e0805

13. It has been built by extending basic convolution networks. As a consequence, they are characterized by an increased number of parameters as well as by a longer training time

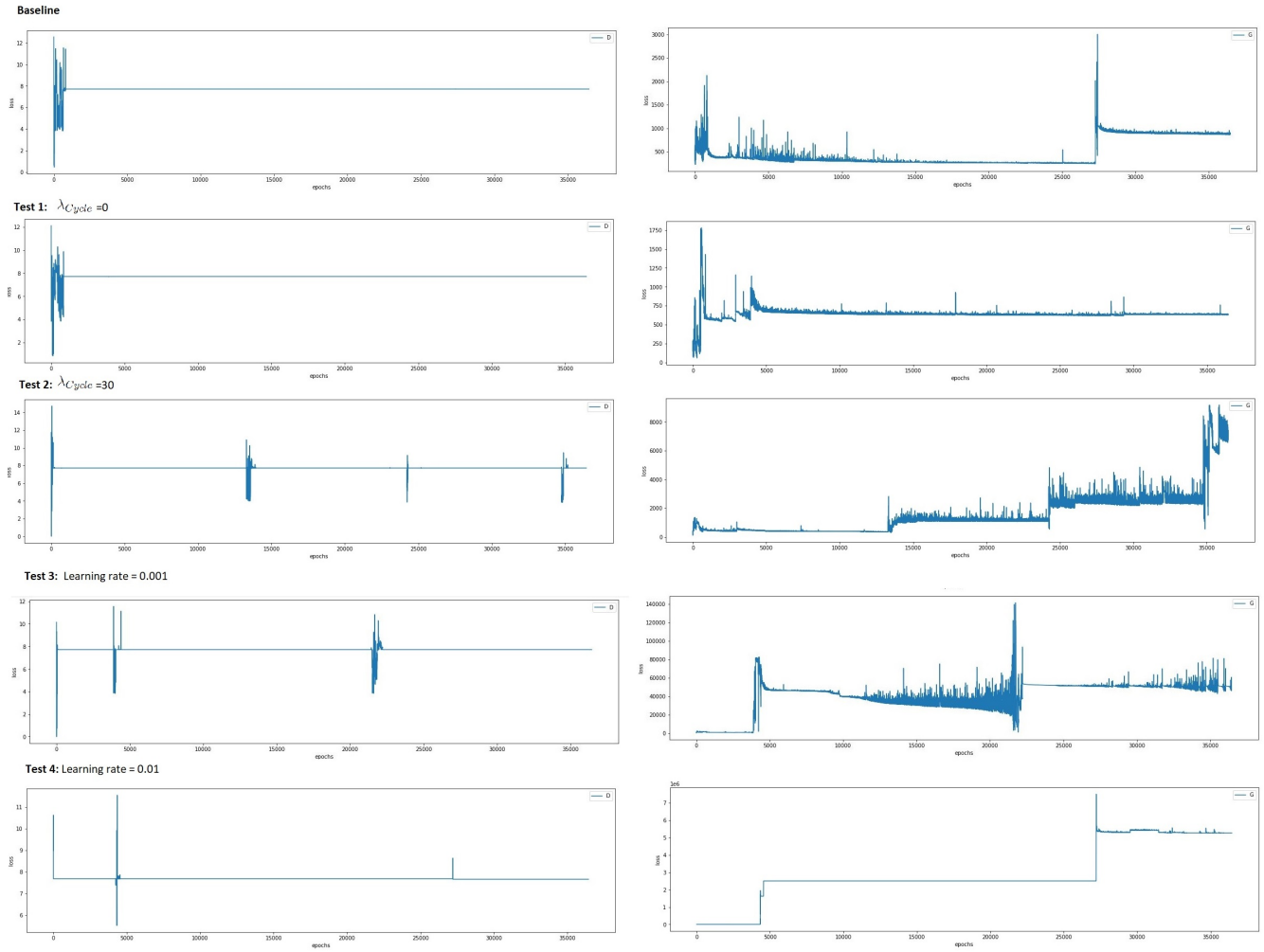

Figure 4 – Discriminator and generator losses obtained during 5 epochs for the different tests

Références

A. Radford, S. C., L. Metz. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv :1511.06434*.

I. Goodfellow, M. M. B. X. D. W.-F. S. O. . . Y. B., J. Pouget-Abadie. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 2672–2680.

Implementing cyclegan using python. (2019). (https ://rubikscode.net/2019/02/11/implementing-cyclegan-using-python/)

J. T. Springenberg, T. B. M. R., A. Dosovitskiy. (2014). Striving for simplicity : The all convolutional net. *arXiv preprint arXiv :1412.6806*.

J. Y. Zhu, P. . A. A. E., T. Park. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE international conference on computer vision*, 2223–2232.

M. Arjovsky, L. B., S. Chintala. (2017). Wasserstein gan. *arXiv preprint arXiv :1701.07875*.

Y. Netzer, A. C. A. B. B. W. A. Y. N., T. Wang. (2011). Reading digits in natural images with unsupervised feature learning. *NIPS workshop on deep learning and unsupervised feature learning*, *2011*.

Z. Murez, D. K. R. R. . K. K., S. Kolouri. (2018). Image to image translation for domain adaptation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4500–4509.