Deep Learning for Medical Imaging - Final Project Report
Using CycleGANs to translate MRI to CT scans of the brain

Antonin Duval
antonin.duval@student-cs.fr

Leo Fillioux
leo.fillioux@student-cs.fr

Sebastien Saubert
sebastien.saubert@student-cs.fr

Abstract

Given either a CT scan or a type T2 MRI image of the brain, we want to convert from one to the other. For that, we will show how a CycleGAN can be applied, trained on unpaired images of CT scans and T2 MRI. After a registration step and a bit of preprocessing, the data is fed into a CycleGAN model, with its classical losses along with an additional cycle perceptual loss. To evaluate the overall results, other that visually, we used the structure similarity metric (SSIM) to evaluate the quality and how much information was conserved by our architecture. The important implementation and hyperparameter tuning steps are described.

Introduction

The original GAN (generative adverserial network) paper [7] was published in 2014 by Ian Goodfellow and Yoshua Bengio. A GAN is able to produce realistic images given some input noise. It works by training two networks, adverserial to one another : a generator, which produces images, of a particular distribution, and a discriminator, which tries to predict if the input image is a real one, or one which came out of the generator. The generator's goal is to trick the discriminator, and the discriminator's goal is to be able to differentiate real images from generated images, hence the adjective "adverserial".

CycleGANs were introduced by a paper [6] published in 2017 by Jun-Yan Zhu. The idea is to have two relatively close distributions of images (in our case CT scans and T2 MRIs of the brain), and to be able to go from one to the other. The dataset is composed of unpaired images, which means that for a given CT scan of the brain, we do not have an MRI of the same patient. Here, there are two generators and two discriminators, one for each transformation. The general idea behind the training is that a generator $X \to Y$ must be able to generate realistic $\hat{y}$ images from x, but at the same time, we want for $Y \to X$ applied to $\hat{y}$ to be very similar to x.

It can be very interesting to be able to deduce one type of medical imaging from another for multiple reasons. These images can be quite expensive to acquire and sometimes require injection or radiation of the patient, which is better when avoided.

CT scans are a quick and non invasive type of medical imaging, but which do not give a very high level of detail. Even though this is sometimes necessary, we often need to have a higher level of detail. In MRI, the amount of information which can be retrieved is much greater, but the process is much more time consuming, expensive and can even be dangerous (with the very high magnetic fields involved). Finding a way to deduce the information present in an MRI from a CT scan can therefore be very intersting.

We chose two types of images that have relatively close distributions so that the model will be able to deduce one image from the other. We will therefore train a CycleGAN on datasets of unpaired images of CT scans and MRIs to go from one distribution to the other.

1. Related work

In [5], two new losses are proposed, to apply CycleGANs to medical images. These two losses are added to the already existing CycleGAN losses. The idea is that cycle consistency loss alone does not allow the network to reproduce the same details and image quality as in the original images.

The first proposed loss is the cycle perceptual loss. A pretrained feature extractor is used on the original image and the cycle reconstructed image. The loss is a weighted sum, over the features extracted from different layers, of the MAE between an image, and the cycle reconstructed one. The loss expression is

$$\mathcal{L}_{cPercep} = \sum_{i=0}^{L} \lambda_i (\|F_i(x) - F_i(\hat{x})\| + \|F_i(y) - F_i(\hat{y})\|)$$

Where F_i are the features extracted from the i^{th} layer of the feature extractor, and λ_i is the weight given to the i^{th} layer.

The second proposed loss is a style-transfer loss, which ensures that the cycle reconstructed image should have the same level of details and style as the original image. It is based on feature map correlations between the features of a certain layer for the original and the cycle reconstructed image.

2. Methodology

2.1. Dataset

Because CycleGANs do not require the images it uses to be paired, we were able to chose two separate sources for our CT and MRI images. The dataset used for CT scans is CQ500 [2] and the one used for T2 MRI is IXI [3]. These datasets contain 3D volumes of the brain but, for computational power constraint, we will work only on one specific slice of the brain for the CT and T2 modalities. Figure 1 shows an example of extracted slices for both modalities.

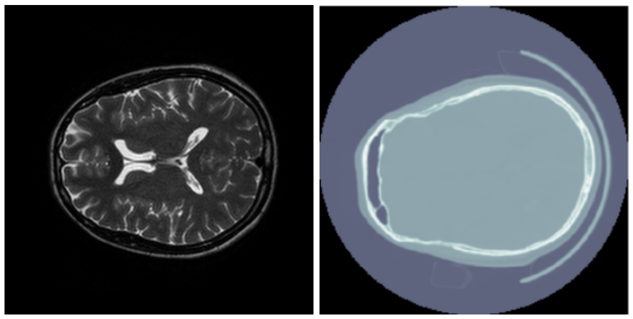


Figure 1. Example of T2 MRI (left), Example of CT scan (right)

As we can see in the sample of a CT scan, colors seems off and we don't have any details of the brain, in comparison of the T2 MRI. This is because the scans are in dicom format, which use a pixel representation with extreme values. One easy way to solve this issue is to clip values that are above 255 and below 0, so that when we plot the image, the normalization can handle smaller variation in pixel. Figure 2 shows a CT scan after the pixel correction. We can now see smaller details inside the brain.

In terms of preprocessing, two majors steps had to be taken. The first one was to do registration on the images with the ANTS library [1]. Training a CycleGAN with unpaired modality is quite difficult by itself, especially if the brains are of different shapes. This is why registration is an important preprocessing step to make the training of the model easier. To proceed on registration, we chose one reference image per modality, these two references images were chosen based on: their shape similarity between each other, plus how well it is centered and oriented. The objective was to take the brain that looked the best as a reference. We chose to do registration using both linear and deformable registration, meaning that each brain will be both placed at the same position as it reference brain, but it will also be reshaped to have a similar shape. Once this had been done, all images were located at the same position and had roughly the same shape but still with a different content.

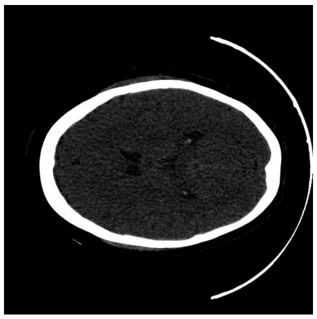


Figure 2. Example of an adjusted CT scan

The second part, which only applied to CT images, was to remove a sort of artefact which was present in all images (see figure 1 right image), and which probably corresponds to a metal plate preventing the x-ray propagation in the room. Since the CT modality has been registered, the zone of interest, i.e. the brain, is localized at a same location for all images. Then, defining a mask was quite easy by taking only a reference image to build it and apply it on all CT images. For building the mask, instead of doing it manually, we apply the following steps on our reference image:

1. apply a binary thresholding on a grayscale image: we keep the bone and the artefact but no more the content of the brain
2. repeat an erosion up until the artefact is removed
3. repeat a dilatation so that the "bone" is reconstructed back with a certain margin
4. fill the content of the bone with binary_fill_holes from scipy library.

Figure 3 shows the resulting mask used then to remove the artefact and figure 4 show the complete preprocessing of CT scans from registration to artefact removal.

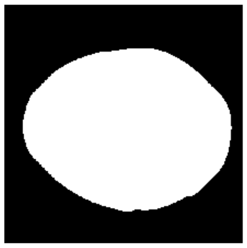


Figure 3. Mask generated to remove artefacts on CT images

The last adjustment performed is to align the resolution of our two modalities to 256×256 pixels.

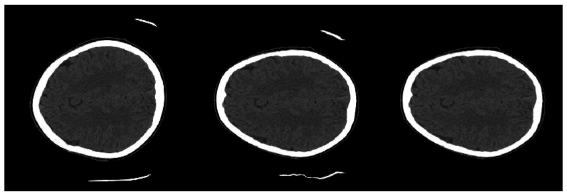


Figure 4. Original CT scan(left), after applying linear and deformable registration (center), and after removing the artefacts (right).

2.2. Model implementation

CycleGANs consist of two generators, each of them in charge of generating images from one modality to the other. To evaluate the quality of the generators, two adversarial networks, the discriminators compare the results from generators to modality's real images.

For the generator, we used U-NET neural network architecture. Indeed, as we need to generate an image from one modality to the other, keeping the same resolution (256×256 pixels), it was straightforward to select U-NET as a basis architecture. To hopefully speed the training and, therefore convergence, we also apply transfer learning to our generators by using pretrained U-NET networks on imagenet. For that, we used models provided by the segmentation-model python library [4] that provides pretrained U-NET models with different backbones. In our case, we used the resnet-18 backbone.

Regarding the two required discriminators, we apply the same strategy: using also a pretrained resnet-18 backbone pretrained on imagenet for these classifiers.

2.3. Losses

The loss used for the discriminators is the binary cross entropy loss, which expresses how well the discriminator is able to distinguish real from generated images. For example, we would give to the T2 discriminator a batch of real T2 images and a batch of fake T2 images and see how well it is able to separate them. We would then do the same for CT images.

$$\begin{aligned}L_{D_{T2}} &= \frac{1}{|T2|}\sum_{T2} -log(D(T2)) \\ &+ \frac{1}{|CT|}\sum_{CT} -log(1 - D(G(CT)))\end{aligned}$$

$$\begin{aligned}L_{D_{CT}} &= \frac{1}{|CT|}\sum_{CT} -log(D(CT)) \\ &+ \frac{1}{|T2|}\sum_{T2} -log(1 - D(G(T2)))\end{aligned}$$

Regarding generators, there are two types of losses. The first one is related to the feedback provided by the discriminator. To get a relevant feedback from discriminators, we must ensure that the discriminator loss as previously described is not too big so that the feedback is valuable but also not too close to zero to avoid vanishing gradient. The loss used by the generators from the discriminators is the following: $L_{G_X}^{D_X} = \frac{1}{|X|}\sum_X -log(D(G(X)))$. We can see that this is the opposite of the second term of discriminators losses. Indeed, as the generator objective is to trick the discriminator, the generator and discriminator losses are somehow opposite.

The second type of loss for generators is the cycle consistency loss, which gives a measure of how well the cycle reconstructed image resembles the original image ($G_X(G_Y(i)) = i$). For this, we first tried to use a simple MSE loss on the images but we found that the results were much better when using the SSIM loss which gives an idea of how well two images compare in terms of luminance, contrast and structure (as a product of all three scores) rather than comparing values pixelwise like the MSE loss.

$$\mathcal{L}_{SSIM}(x, \hat{x}) = lum(x, \hat{x}) + contrast(x, \hat{x}) + struct(x, \hat{x})$$

Unfortunately, with the two losses described previously, the results were not as good as expected. Especially, we were facing a 'mode collapse' where all generated images were the same. To mitigate this, we added a third type of loss to the generators: the perpetual loss [5]. For this, we used a pretrained features extractor (VGG19) and selected the activations from three layers. The loss consists in comparing these activations for an image and its cycle-reconstructed version.

$$\mathcal{L}_{cPercep}(x, \hat{x}) = \sum_{i=0}^{L} \lambda_i(\|F_i(x) - F_i(\hat{x})\|$$

Where F_i are the features extracted from the i^{th} layer of the feature extractor, and λ_i is the weight given to the i^{th} layer.

Therefore, the loss for a generator consisted in summing the loss from discriminator, the cycle consistency loss for its modality with the perpetual loss for the both modalities (with some weights which had to be tuned).

$$\begin{cases} \mathcal{L}_{generator\,x}(x, y) = \mathcal{L}_{SSIM}(x, \hat{x}) + \mathcal{L}_{cPercep}(x, \hat{x}) + \\ \qquad \mathcal{L}_{cPercep}(y, \hat{y}) + \mathcal{L}_{D_y} \\ \mathcal{L}_{generator\,y}(x, y) = \mathcal{L}_{SSIM}(y, \hat{y}) + \mathcal{L}_{cPercep}(y, \hat{y}) + \\ \qquad \mathcal{L}_{cPercep}(x, \hat{x}) + \mathcal{L}_{D_x} \end{cases}$$

2.4. Hyperparameter tuning

This project allowed us to see that training a CycleGAN was not easy, first because of the implementation with multiple losses to take into account, but also because the hyperparameters had to be very finely tuned. The hyperparameters which had the biggest impact on the final result were probably :

1. Learning ratio k : how many iterations of training the generator are done, before doing one iteration of training the discriminator ? This is probably the most important hyperparameter. We want the generator to fool the discriminator, but not too quickly, otherwise there is no feedback to be taken from the discriminator. If the discriminator gets too good too quickly, then the generator is never able to converge. We have to find the right balance so that the discriminator gets fooled at some point (so has an accuracy of 0.5), but not too quickly. Visually, we can see that when the discriminator gets too good, the generator produces either only black images, or very noisy complex structures that do not look at all like the target distribution. During our experiment, we obtain the best result by setting a 4 to 1 ratio of training between the generator and the discriminator.

2. Learning rates : there are two different learning rates, for the generator and the discriminator. How fast we want the generator and the discriminator to learn is the same question we asked in the learning ration k. Here the learning rate also plays the usual role where we don't want the model to diverge. During our experiment, we obtain the best result by setting a $1e^{-4}$ learning rate to the generator and $1e^{-3}$ learning rate for the discriminator.

3. Loss weights : which weights do we give to the cycle perpetual and cycle consistency loss ? The aim of the cycle perpetual loss was to avoid mode collapse, which happened when we only used the cycle consistency loss. However, putting too big of a weight on the cycle perpetual loss gave very bad results, and putting a weight that was too small did not prevent mode collapse.

3. Evaluation

3.1. Evaluation of a CycleGAN

It is challenging to find a good metric to judge how good a GAN is versus another one. One can think of validation by visual inspection, which is sufficient to tell if a model seems to work well. However, in the case of a generative algorithm in the medical domain, we are not qualified to judge if our generated image has kept the important features a doctor would need. Moreover, since we are not working with paired-images, we do not have the real CT image corresponding to the T2 MRI, or the real T2 image for a CT image, so we are just guessing the image is corresponding. A good way to evaluate our model and to compare it with existing one would be to use some quantitative metrics. An efficient GAN evaluation measure should:

1. favor models that generate high fidelity samples, meaning it is hard to distinguish fake sample from real one
2. favor models that generate diverse samples, and reject models that are in mode collapse or mode drop
3. have well-defined bounds
4. be sensitive to image distortions and transformations. GANs are often applied to image datasets where certain transformations to the input do not change semantic meanings. Thus, an ideal measure should be invariant to such transformations.
5. agree with human perceptual judgments and human rankings of models have low sample and computational complexity.

In our case, we decided to use the Fréchet Inception distance as well as the SSIM score.

3.2. Frechet Inception Distance

The Frechet Inception Distance, or FID for short, is a metric for evaluating the quality of generated images and specifically developed to evaluate the performance of generative adversarial networks.

Inception score measures the quality of a generated image by computing the KL divergence between the (logit) response produced by this image and the marginal distribution, i.e., the average response of all the generated images, using an Inception network trained on ImageNet. In other words, Inception score does not compare samples with a target distribution, and is limited to quantifying the diversity of generated samples. Frechet Inception distance compares Inception activations (responses of the penultimate layer of the Inception network) between real and generated images.

The Fréchet Inception Distance is calculated using this equation :

$$\mathrm{FID}=\left\|\mu_r-\mu_g\right\|^2+\operatorname{Tr}\left(\Sigma_r+\Sigma_g-2\left(\Sigma_r \Sigma_g\right)^{1/2}\right)$$

where $X_r \sim \mathcal{N}\left(\mu_r, \Sigma_r\right)$ and $X_g \sim \mathcal{N}\left(\mu_g, \Sigma_g\right)$ are the 2048-dimensional activations of the Inception-v3 pool3 layer for real and generated samples respectively.
To retrieve the logits of our real and generated images, we simply remove the last layer of the Inception network, and make a forward pass of the images. We can then compute the FID score for the set of generated images. However, even though FID indicates better-quality images; conversely, a higher score indicates a lower-quality image, it may not be optimal for medical imaging. Indeed, FID uses the Inception network, that was trained on ImageNet, which is composed of pictures very different from the kind of images we have in our datasets of CT scan and MRI.

3.3. Structural Similarity

The structural similarity (SSIM) is a very common metric to mesure the similarity between two image.

$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$

with: $\cdot\mu_x$ the average of x
$\cdot\mu_y$ the average of y
$\cdot\sigma_x^2$ the variance of x
$\cdot\sigma_y^2$ the variance of y
$\cdot\sigma_{xy}$ the covariance of x and y
$\cdot c_1 = (k_1L)^2, c_2 = (k_2L)^2$ two variables to stabilize the division with weak denominator;
$\cdot L$ the dynamic range of the pixel-values (typically this is $2^{\text{fitits per pixel}} - 1$);
$\cdot k_1 = 0.01$ and $k_2 = 0.03$ by default.
SSIM score range from 0 to 1, where 1 means a very high similarity. This metric can also be used as a loss function to train GAN models. One downside of the SSIM score is that it tends to favor model that are very close to the training data, and doesn't penalize well overfitting.

3.4. Result

Since the training of a CycleGAN can be quite time-consuming, we first decided to use a subset of 100 images of CT and T2 images. This strategy allowed us to have result much quicker and to adapt our model more easily. Once we were satisfied with our result and with the hyperparameters we had chosen, we retrained the model on the whole dataset.

The following metrics were computed on the test dataset, containing 85 T2 MRI images and 111 CT scan.

	CT → T2	T2 → CT
FID	121.82	114.95
SSIM	0.803	0.864

Table 1. Result for our model trained on 100 images

	CT → T2	T2 → CT
FID	106.17	95.73
SSIM	0.800	0.870

Table 2. Result for our model trained on the whole dataset on 120 epochs

	CT → T2	T2 → CT
FID	80.82	71.50
SSIM	0.800	0.879

Table 3. Result for our model trained on the whole dataset on 200 epochs

3.5. Visual results

Even though metrics give a quantitative result of the performance of our CycleGAN, the best feedback that we had was visually. This allowed us to see if the system showed mode collapse, if the transformations were realistic and if the reconstructed image was close to the original one. For this reason, we plotted the original image, the transformed and the cycle reconstructed images for both CT and MRI images on the validation set every 5 training epochs. This was a good way to supervise the training.

Conclusion

Training a CycleGAN is known to be a difficult task by definition in machine learning as we need to address two opposite goals: a generator that must be good at forging fake data and a discriminator who must be good at identifying these forged data. Finally, we must find an unstable equilibrium between these two objectives.

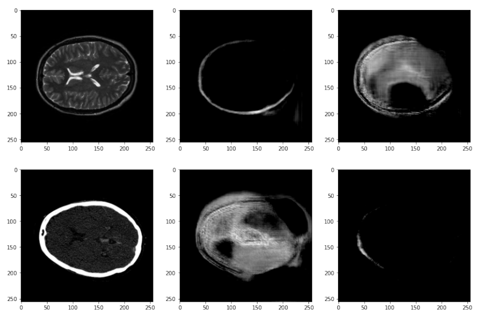
Applying image processing, especially registration, allowed to ease/simplify the task and tweaking the loss functions, especially by adding the VGG loss was a game changer as it allows to progress beyond the 'mode colapse'. At the end, the model provides really good results as demonstrated by several metrics and visual checks.

Training a CycleGan is also a long process requiring a lot of computation power. In medical fields, the data is voxels, which makes the task even harder. Limiting the training on identified slices and performing model and parameters selection on a restricted amount of data (100 images per modality) allow to find a good configuration that worked even better on the full dataset. This was one of the criteria of success as the computation on the full dataset for 200 epochs took 10 hours on Google Colab.

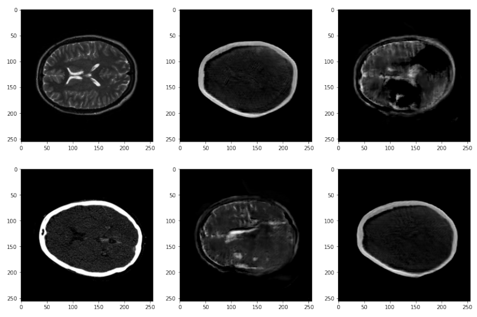
On the other hand, as we are not working on paired images per modality, we don't have any solid measure to confirm that the generated images would be as a real one. Saying that, it would be great to acquire a paired dataset to validate the consistency of our generators against real images.

References

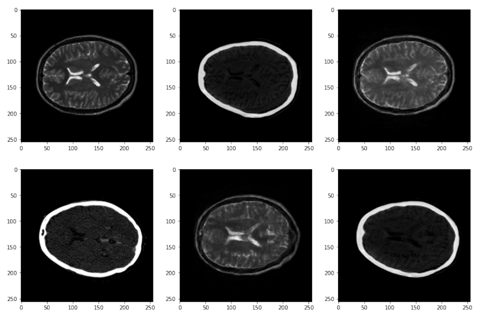
[1] Antspy : https://github.com/antsx/antspy.
[2] Cq500 dataset : http://headctstudy.qure.ai/dataset.
[3] Ixi dataset : https://brain-development.org/ixi-dataset/.
[4] Segmentation model : https://github.com/qubvel/segmentation_models.pytorch.
[5] K. Armanious. Unsupervised medical image translation using cycle-medgan. Mar. 2019.
[6] J.-Y. Z. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. Nov 2018.
[7] I. J. Goodfellow and Y. B. et al. Generative adversarial networks. June 2014.



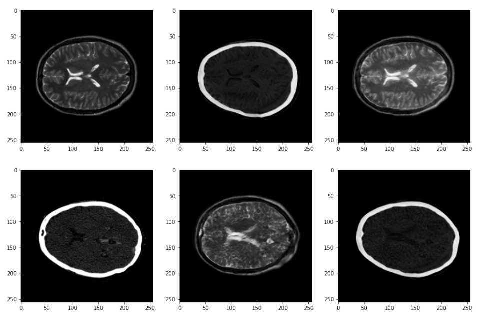
(a) results after 10 epochs



(b) results after 25 epochs



(c) results after 50 epochs



(d) results after 80 epochs

Figure 5. Results on the validation set after 10, 25, 50 and 80 epochs of training. Left column : original image, center column : transformed image, right column : cycle reconstructed image. Top row : MRI original image, bottom row : CT original image.