MRI image translation using CycleGAN

Problem Statement:

Misdiagnosis in the medical field is a very serious issue but it's also uncomfortably common to occur. Imaging procedures in the medical field requires an expert radiologist's opinion since interpreting them is not a simple binary process (Normal or Abnormal). Even so, one radiologist may see something that another does not. This can lead to conflicting reports and make it difficult to effectively recommend treatment options to the patient.

One of the complicated tasks in medical imaging is to diagnose MRI (Magnetic Resonance Imaging). Sometimes to interpret the scan, the radiologist needs different variations of the imaging which can drastically enhance the accuracy of diagnosis by providing practitioners with a more comprehensive understanding.

To obtain different variations such as T1 weighted or T2 weighted of the MRI images needs to invest more time and resources. So, is there any deep learning part that comes in handy to solve this problem?

Approach:

The answer to the above question is simply YES! CycleGAN a variant of GAN (generative adversarial network) could somewhat solve this problem. With the help of deep learning, we can use style transfer to generate artificial MRI images of different contrast levels from existing MRI scans. This will help to provide a better diagnosis with the help of an additional image.

Why would this variant a choice? Because it does not require paired data to train on. Also possessing the paired data is more resource consuming and also most of the time simply it does not exist. CycleGAN is a choice.

So in this project, build a model which can translate T1 weighted MRI image into T2 weighted or vice versa.

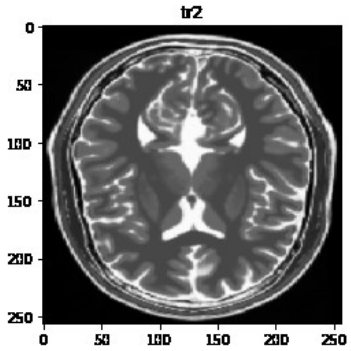
Data Understanding:

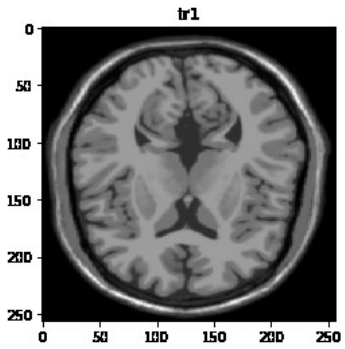
In a dataset, there are two kinds of images. T1-weighted MRI and T2-weighted MRI images. Both types of MRI contrast different regions within the body. And both serve different purposes.

In T1-weighted → fat, tissue appears to be bright

In T2-weighted → both fat and water regions appear to be bright

Images on the next page:





Data Pre-processing:

Image Resizing: The images were of different sizes, so resized them to 256 x 256.

Image Normalization:

Normalization a very important step in pre-processing. Normalizing all of our inputs to a standard scale, we're allowing the network to more quickly learn the optimal parameters for each input node

It helps in gradient propagation. Normalizing the image here between -1 to 1.

Augmentation: It helps to increase the training data, but not all augmentation techniques work in every situation. Here we are not going to see the images upside down, as MRI images are scanned with a particular procedure. There is a chance that we might see side flipped images. So here random flip is used to augment data.

Model Building:

Hyper-parameters:

Drop out = to avoid overfitting.

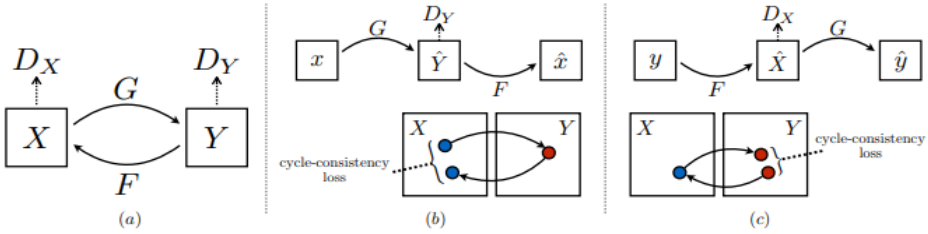
LAMBDA = LAMBDA is deciding the weights of Cycle consistency loss and Identity loss, after experimenting with different values, 1 was giving better results.

LAMBDA adversarial= LAMBDA adversarial decides weightage of discriminator loss, 0.5 giving good results

Adam optimizer is used with a small learning rate, and further decaying the learning rate for smooth training.

EPOCHS = Number of iterations

Network Architecture



CycleGAN consists of 2 Generators and 2 discriminators.

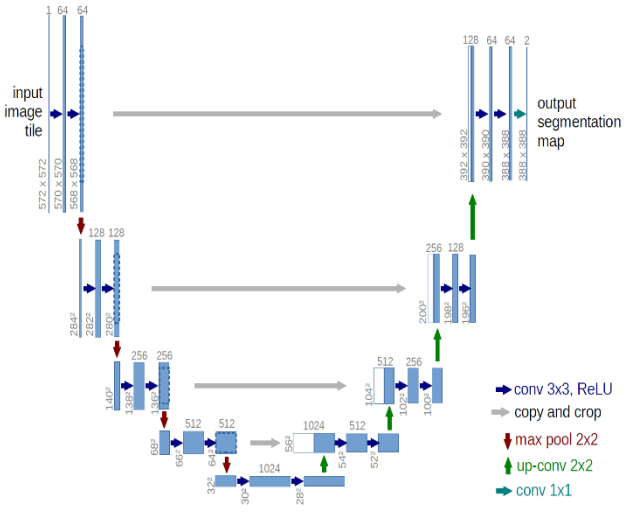
For X domain to Y domain:

- Generator G translate image from the X domain to the Y domain. Discriminator Y will decide the translated image belongs to the Y domain or not.
- Generator Y will try to reconstruct the original X domain image from translated Y domain image. Discriminator X will decide the reconstructed image is the same as the original X domain image or not.

For Y domain to X domain:

- Generator F translates the image from the Y domain to the X domain. Discriminator X will decide the translated image belongs to the X domain or not.
- Generator X will try to reconstruct the original Y domain image from translated X domain image. Discriminator Y will decide the reconstructed image is the same as the original Y domain image or not.

Architecture of UNet:



UNet consists of a contractive and an extensive path, similar to encoder and decoder architecture.

It keeps the dimension of an output image same as that of input to the architecture.

Through copy and crop or skip connection layer we transfer some information from domain X to decoder side because we want particular translation

This architecture used for the generator.

Architecture of Discriminator:

Discriminator in CycleGAN same as CNN binary classifier.

For both architectures, I used the instance normalization technique in which we normalize each channel of the image.

Loss Functions:

Adversarial loss: Both generators are attempting to “fool” their corresponding discriminator into being less able to distinguish their generated images from the real versions.

However, the adversarial loss alone is not sufficient to produce good images, as it leaves the model under-constrained. It enforces that the generated output is of the appropriate domain, but does not enforce that the input and output are recognizably the same.

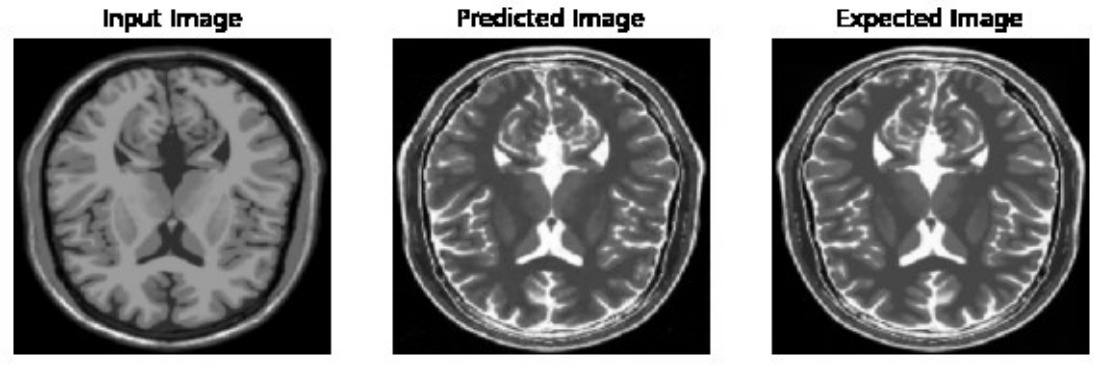
Cycle consistency loss: It relies on the expectation that if you convert an image to the other domain and back again, by successively feeding it through both generators, you should get back something similar to what you put in.

Identity loss: It enforces that CycleGAN preserves the overall colour structure of the picture. It introduces a regularization term that helps to keep the tint of the picture consistent with the original image.

Checkpoints: saving checkpoints at regular interval.

Training: We have a total of 4 networks in place, and at a time only one network should be trained. Suppose the discriminator had successfully caught the fake one. It means a generator needs to improve itself. If the generator is able to fool the discriminator, then the discriminator needs to improve itself. I used Tensor Flow’s inbuilt functionality to achieve a given condition.

At the end of 200 epochs,



Average Generator loss G at epoch 200: 0.86

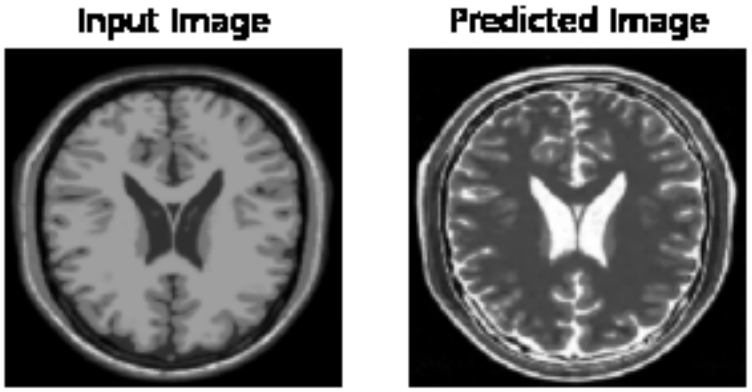
Average Generator loss F at epoch 200: 0.76

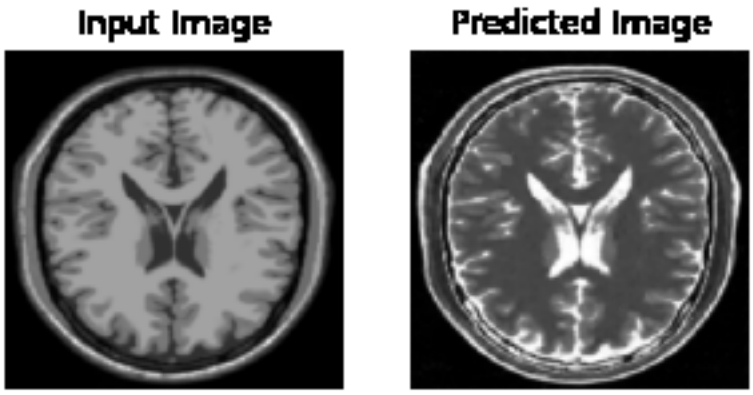
Average Discriminator loss X at epoch 200: 0.69

Average Discriminator loss Y at epoch200: 0.65

Model Testing

The predicted image and Expected image looks pretty much similar. so, we can say that model is performing style transfer decently on the training side.





On the testing side also, we can see that, images from the T1 weighted image decently translated to the T2 weighted image.